Predicciones usando mediciones correlacionadas y posiblemente incompletas

Facultad de Ingeniería, Universidad de la República

Sebastián Daloia
Tutor: Pablo Rodríguez Bocca
Usuario Responsable: Diego Kiedanski

24 de abril de 2020

Abstract

La disponibilidad de grandes volúmenes de datos y las mejoras crecientes en recursos y capacidad de cómputo de los ordenadores, permiten que la toma de decisiones basadas en tecnologías y procesos orientados a los datos sean una herramienta de uso frecuente en ámbitos de negocios, investigación y científicos.
El problema de predecir el consumo de energía puede ser abordado para resolver la *demanda agregada*, es decir estimar el consumo total de todos los clientes. Esto es útil en situaciones de decisión como determinar si construir o no una nueva planta de generación de energía en los próximos cinco años. Otro enfoque es el de la *demanda desagregada*, o sea estimar el consumo de energía para cada cliente por separado. Esto permitiría conocer y caracterizar diferentes perfiles de clientes, por ejemplo para saber cuando y en que lugares es más probable que ocurran picos de consumo para determinadas situaciones climáticas o económicas.
Algunas compañías o productores de electricidad, tienen como objetivo predecir el consumo de energía para cada cliente final (*demanda desagregada*). En este caso el problema de predecir la demanda desagregada está asociada al hecho de que cada consumidor se comporta diferente y con frecuencia de manera impredecible (debido a altas tasas de efectos aleatorios) [12, Secc.1.].
En este trabajo se presentan dos técnicas basadas en clustering, que agrupando a los clientes en perfiles de consumo similares, predicen el consumo de cada cliente a partir de la información del perfil. Ambas técnicas se comparan con las del estado del arte (Laurinec et al. [12]).
Como trabajo previo al problema de predicción, se plantea un experimento teórico simplificado denominado *el modelo de las monedas*, en el cual se puede estudiar para que condiciones experimentales es más ventajoso emplear clustering para realizar predicciones en comparación a no emplear clustering y los efectos de factores externos al experimento o ambientales que afectan el resultado del mismo, como por ejemplo la temperatura.
Se implementa, a su vez, una técnica de resumen informativo, algoritmo *flashlight* propuesto por El Gebaly et al.[6], que es utilizada para resumir tablas de resultados de tamaño grande y mediano, y que es empleada para analizar el problema de predicción de consumo y *el modelo de las monedas*.

Índice general

Capítulo 1

Introducción

Dado un proceso aleatorio que evoluciona en el tiempo, un problema clásico es estimar valores futuros en función de observaciones previas. Si el modelo que describe el proceso es conocido, entonces es posible hacer una predicción de forma óptima (posiblemente en forma de distribución de probabilidad). Para problemas reales, los modelos no son exactos y por tanto, las predicciones tampoco.
El problema de interés para el proyecto es el siguiente: si en lugar de trabajar con un solo proceso se trabaja con una familia de procesos, y si se cuenta con observaciones de algunos de los procesos, pero no todos ¿cómo se puede explotar la correlación entre procesos individuales para mejorar las predicciones?
Como ejemplo motivador, considérese el consumo de energía eléctrica por parte de diversas familias en un barrio. Para algunas familias se conoce el consumo, tomado a intervalos regulares en el tiempo, completo de los últimos dos años, mientras que para familias recientemente incorporadas al servicio eléctrico se conoce el consumo de solo las últimas semanas o incluso unos pocos días. A las familias con mayor tiempo en el servicio, y por ende con mayor información de consumo, se las puede caracterizar a través de la información proveniente de su consumo (por ejemplo: tipo de comportamiento en la semana y los fines de semana, tipo de consumo para las diferentes estaciones del año, picos máximos y mínimos mensuales). A partir de esta información se puede pensar, como una solución posible al problema de predecir el consumo futuro, en agrupar a familias con una caracterización parecida en perfiles de consumo y generar modelos que predigan el comportamiento de las familias pertenecientes a cada perfil, en lugar de un modelo para cada familia individual. Aquí es que surgen las siguientes preguntas a responder: ¿Es posible agrupar a las familias de manera tal que se pueda generar un modelo preciso para cada perfil? ¿Cómo se pueden identificar los

perfiles en base a la información histórica de consumo de las familias? A su vez, cada familia recientemente incorporada, para la cual hay poca información de consumo observada, puede presentar un consumo más similar a uno de estos perfiles respecto a otro y por lo tanto ser incorporada a dicho perfil y utilizar su modelo para predecir el consumo de energía. Para estas familias: ¿es posible extrapolar conocimiento de familias antiguas para predecir mejor su comportamiento? y si se puede ¿qué tanto mejora la predicción?
El comportamiento en el consumo de energía también puede ser afectado por eventos externos como la temperatura, la estación del año o por la economía. ¿Cómo tener en cuenta estos eventos externos con el fin de mejorar la predicción?
El objetivo es generar un modelo capaz de predecir el consumo de energía de cada familia, empleando el agrupamiento de estas en perfiles e incorporando, al modelo, información de eventos externos relevantes que afecten al consumo; se busca también mejorar la predicción del consumo de energía en los hogares recientemente incorporados, asignándoles como modelo de predicción el de aquel perfil con mayor similitud en el comportamiento del consumo de energía respecto al del hogar.
La metodología empleada comienza con un trabajo de investigación, estudiando el estado del arte para la predicción del consumo de energía en hogares empleando técnicas de agrupamiento o clustering. Primero se definen las métricas con las cuales se van a evaluar los resultados de las predicciones. Se realiza la formalización del modelo y se formula el objetivo de dicho modelo. Antes de abordar el problema real, es necesario entender algunos aspectos claves del problema como: que tan relevante es para una técnica incorporar información sobre el agrupamiento de los clientes o sobre los eventos externos en relación a no incorporar esta información. Con el objetivo de dar respuesta a estos cuestionamientos y entender la complejidad del problema se define un experimento teórico simple, llamado *modelo de las monedas*, que permita estudiar de manera controlada la relevancia de estos puntos al momento de predecir. Se formaliza un modelo para el experimento, a partir del cual se puedan comparar diferentes técnicas que ignoran o hacen uso del agrupamiento y/o los eventos externos. En la siguiente instancia se elige un conjunto de datos, para el problema real, con el cual trabajar, aplicando un preprocesamiento a los datos y realizando una descripción cuantitativa del mismo. Por último se propone una solución para el problema real, se entrenan las técnicas propuestas junto con las del estado del arte, se prueban sobre los datos elegidos y se comparan los resultados obtenidos por cada técnica.
Es importante resaltar que el estudio y entendimiento del experimento teórico es importante para este trabajo ocupando varios meses en el calendario antes del

abordaje del problema real.
En este trabajo se aborda el problema de la *demanda desagregada*. Para abordar este problema es necesario tener un dataset o juego de datos con mediciones de consumo de energía de clientes que formen parte de una red eléctrica. En la industria de la energía estos datos se pueden conseguir a través del despliegue de grillas inteligentes (*smart grids*). Estas consisten de un conjunto de clientes consumidores (y/o productores) de electricidad, en donde cada cliente está equipado con un medidor inteligente que envía el consumo de electricidad a un servidor cada cierto intervalo de tiempo, usualmente de 15 o 30 minutos.
El Capítulo 2 comienza con una introducción al concepto de *series de tiempo*, presentándose algunas técnicas de forecasting o pronóstico para series de tiempo. Después se presenta el estado del arte para el problema de la *demanda desagregada* de energía, describiendo los resultados obtenidos por Laurinec et al. [12]. El capítulo sigue con una descripción del algoritmo *matrix completion*, utilizado para completar entradas no observadas de una matriz a partir de las entradas que sí tienen observaciones. Este algoritmo motivará una técnica de predicción de consumo de energía, pensando a las predicciones como las entradas no observadas de una matriz incompleta. La última sección trata sobre *explanation table*, un algoritmo que permite hacer resúmenes informativos de tablas de gran tamaño. *Explanation table* será una apoyatura, a lo largo de todo el trabajo, para interpretar las tablas de resultados de las simulaciones.
En el Capítulo 3 se describe formalmente el problema a resolver, además para comprender mejor como pueden influir en un modelo factores como la temperatura o grupos de hogares con comportamientos similiares, se postula un problema teórico sencillo denominado, *el problema de las monedas*, deduciéndose de éste las primeras conclusiones sobre la incorporación de dichos factores a un modelo.
En el Capítulo 4 se realiza un resumen descriptivo del juego de datos de consumo de energía utilizado y de los datos con la información climática.
En el Capítulo 5 se describen las técnicas propuestas en este trabajo para abordar el pronóstico del consumo de energía bajo el enfoque de la *demanda desagregada*. Además se propone una extensión al modelo Laurinec et al. [12] estudiado en el estado del arte, para que este incorpore las predicciones para las familias recientemente agregadas a la grilla de consumo eléctrico, las cuales tienen mucho menos información histórica que el resto de las familias.
En el Capítulo 6 se presentan los resultados de la ejecución de las técnicas propuestas y se comparan con los resultados de las técnicas del estado del arte, descritas en la Sección 2.2.2.
En el Capítulo 7 se finaliza con las conclusiones de este trabajo.

El contenido de los apéndices se reparte de la siguiente manera:
en el Apéndice A se presentan análisis, algoritmos y resultados del experimento sencillo de las monedas, que no fueron incluidos en el cuerpo principal del documento.
El Apéndice B, contiene el detalle de los pasos del algoritmo del método de Laurinec et al. [12] y de una de las técnicas propuestas en este trabajo, para resolver el problema de predicción.
Dada la centralidad de la técnica de clustering K-means, en este trabajo y en el método de Laurinec et al. [12], el Apéndice C, formaliza el problema de clustering a resolver, define el algoritmo K-means y explora sus propiedades.
El Apéndice D es el más extenso, y se dedica a describir las medidas de dependencia más relevantes al momento de estudiar las series de tiempo, se define lo que es un proceso estocástico estacionario y las implicancias, para una serie de tiempo, de que un proceso estocástico sea estacionario. El primer modelo que se analiza es el de regresión lineal múltiple. Luego se especifican criterios para determinar la cantidad óptima de parámetros para un modelo. La sección central de este apéndice está dedicada al modelo clásico de series de tiempo: ARIMA.
Se detallan cada uno de sus componentes y se describe como es realizado el forecasting. Después de ARIMA se definen los tres modelos clásicos de la técnica *Exponential Smooth (ES)*: *Simple ES*, *Holt ES* y *Holt-Winter ES*, junto a sus fórmulas para hacer forecasting. Por último se describe la técnica basada en árboles de desición regresivos denominada *Random Forest* y como hacer forecasting con esta técnica. Los modelos de regresión lineal múltiple, ARIMA, Exponential Smooth y Random Forest son utilizados para resolver el problema de predicción por parte de las diferentes técnicas del método de Laurinec et al. [12] estudiado en el estado del arte.
En el Apéndice E se formaliza el problema del completado de matrices, se describe el Algoritmo *Information Cascading Matrix Completion* presentado por Meka et al. [13], se detallan sus propiedades y se especifica la implementación del algoritmo hecha en este trabajo.
En el Apéndice F se define lo que es una *Explanation table*, sus características y detalles de la implementación hecha en este trabajo.

1.1. Contribuciones

- Se define un modelo para la predicción del consumo de energía en hogares basado en clustering y en la información de las temperaturas observadas

para cada tiempo de registro de consumo (Sección 3.1).
Se define un método para resolver este modelo (Sección 5.1). Se proponen dos técnicas diferentes que implementan el método: *la Técnica de la Media* (Sección 5.2) y *la Técnica del Completado de matrices* (Sección 5.3).
Se deja como antecedente, para trabajos futuros, la comparación en performance *la Técnica de la Media* que predice con un valor de consumo medio para los clientes de cada perfil y de *la Técnica del Completado de matrices* que emplea el algoritmo de completado de matrices (Algoritmo 11) más clustering para la predicción, contra las técnicas de predicción definidas en el estado del arte (Sección 2.2).

- Se define un experimento teórico simple, denominado *el modelo de las monedas* (Sección 3.3), en el cual las monedas se agrupan en dos perfiles y el resultado de salir cara para las monedas en cada perfil depende de la temperatura registrada al momento del experimento. Este experimento permite de manera *controlada* evaluar diferentes técnicas de predicción para el resultado de cada moneda, para las diferentes configuraciones del experimento.
Se prueban sobre el modelo de las monedas una técnica frecuentista basada en clustering y una técnica que emplea el algoritmo de completado de matrices (Algoritmo 11) basada en clustering.

- Implementación en Python de: **1)** Algoritmo *flashlight* (Sección F.3.2) de la técnica *explanation table*, para hacer resúmenes informativos automáticos de tablas. **2)** Algoritmo *iterative scaling* (Algoritmo 12). Técnica clásica para encontrar los pesos w_i asociados a las features f_i definidas para un modelo condicional exponencial de la forma: $\frac{1}{Z}e^{w_i \cdot f_i}$. Este procedimiento es utilizado por la técnica *explanation table*. **3)** Algoritmo *Information Cascading Matrix Completion* (ICMC), para el completado de matrices con valores indefinidos para algunas de sus entradas, presentado en el trabajo de Meka et al. [13]. La implementación atiende los detalles de implementación especificados en Meka et al. [13], para las matrices que no pueden ser totalmente completadas por el Algoritmo (Algoritmo 11). El código se encuentra en gitlab: gitlab.fing.edu.uy/prbocca/energia, véase en el README la sección **Código del proyecto**, para ubicar los archivos que contienen estas implementaciones.

- Extensión del método general de predicción basado en clusters (Algoritmo 8) presentado por Laurinec et al. [12], para poder predecir el consumo de energía de clientes recientemente agregados a la grilla de consumo y que

tienen poca información en comparación a los clientes cuyos registros de consumo comenzaron muy anteriormente (Sección 5.5).
Se define una técnica de clustering, para agrupar clientes, alternativa a la propuesta por Laurinec. Esta técnica considera el agrupamiento de clientes recientemente ingresados a la grilla, lo cual no era posible con la técnica propuesta por Laurinec.
Se comparan los resultados de predicción de: **1)** la técnica de clustering propuesta + las técnicas de predicción de Laurinec; y **2)** la técnica de clustering de Laurinec + las técnicas de predicción empleadas por Laurinec. Consiguiéndose para **1)** una mejora entre el 0,25 % y el 1,1 % en RMSE, en la predicción para las diferentes técnicas en comparación a **2)**.

Capítulo 2

Estado del arte

El capítulo comienza con la presentación de los conceptos principales empleados para la predicción del consumo individual de energía en los hogares. Se definen: *series de tiempo*, *procesos estacionarios y/o estacionales*, y los primeros ejemplos de predicción.
En la parte principal de este capítulo, se aborda un método actual de *forecasting* propuesto por Laurinec et al. [12] que toma como entrada un dataset de consumo de energía y agrupa a los clientes según patrones de consumo similares. Se realiza una descripción de dicho método, se describe y explica el algoritmo propuesto, se resumen los modelos empleados y se presentan los resultados obtenidos.
Una técnica alternativa a las series de tiempo es: pensar el problema de predecir el consumo de cada cliente como una matriz incompleta, cuyas filas representen el consumo de cada cliente, y que las columnas representen el consumo de todos los clientes para un tiempo dado. Algunas entradas de la matriz pueden no tener valores observados e interesa aplicar alguna técnica para completar estos lugares a partir de los valores observados de la matriz. De manera remarcada el interés está en completar las columnas para los tiempos que se quieren predecir. Una descripción del algoritmo *Information Cascading Matrix Completion* Meka et al. [13] es realizada.
Cada una de las técnicas propuestas tendrá parámetros a ajustar, y habrá un resultado determinado por las métricas empleadas para cada juego. Ante la posibilidad de analizar una tabla grande de parámetros de cada modelo y los resultados obtenidos, resulta de ayuda el empleo de una técnica como *Explanation tables* propuesta por ElGebaly et al. [6] para automatizar el resumen de los resultados de la tabla. La técnica es eficiente en tiempo de ejecución y apunta a ser informativa. Se concluye esta sección con la explicación de esta técnica.

2.1. Forecasting y series de tiempo

Las *series de tiempo* son datos experimentales colectados en diferentes puntos del tiempo. Por ejemplo: en la Figura 2.1 los datos del experimento son las ganancias de la compañía Johnson and Johnson por acción y los puntos en el tiempo colectados son los trimestres comprendidos entre Enero de 1960 y Diciembre de 1980. El *análisis de las series de tiempo* tiene como objetivo primario desarrollar modelos matemáticos que provean una descripción plausible de los datos colectados.
A continuación se describirán los conceptos básicos de las series de tiempo, principalmente siguiendo los libros de Rob J Hyndman et al. [10], y Robert H. Shumway et al. [14], se mantienen los términos más relevantes en inglés.

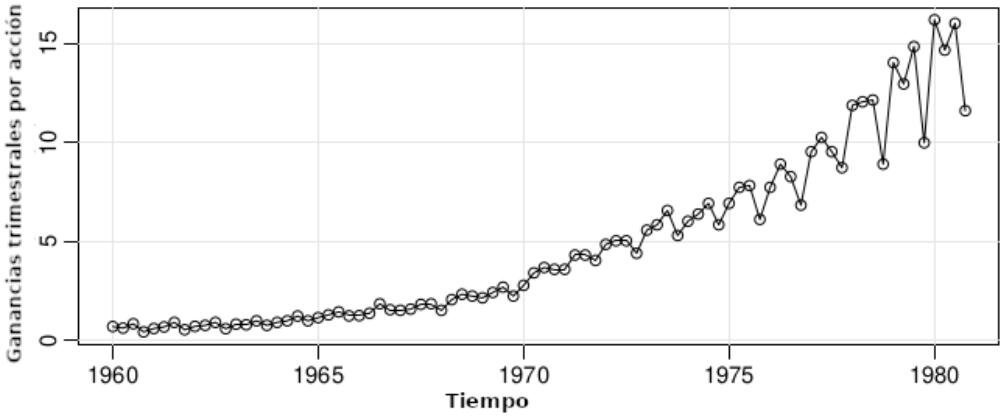


Figura 2.1: Ganancias trimestrales por acción. Johnson and Johnson. Desde el 1^{er} trimestre de 1960 al 4^{to} trimestre de 1980. Figura extraída de la página 3 del Libro [14].

Desde un punto de vista estadístico una *serie de tiempo* puede ser definida como una colección de variables aleatorias indexadas de acuerdo al orden en que son obtenidas en el tiempo. De esta manera se puede considerar a la serie de tiempo como una secuencia de variables aleatorias $x_1, x_2, x_3, \ldots$, donde x_1 representa el valor tomado por la serie para el primer punto, x_2 el valor para el segundo punto en el tiempo, x_3 el valor para el tercer punto en el tiempo y así sucesivamente. En general una colección de variables aleatorias $\{x_t\}$ indexadas por t, se denomina *proceso estocástico*. Si la serie de tiempo representa el consumo de energía de un cliente, entonces la representación más común es usar tiempos discretos (por

ejemplo cada 15 minutos, $t \in \mathbb{N}$) y consumos continuos ($x_t \in \mathbb{R}, \forall t$), en ese caso el proceso estocástico se dice de espacio contínuo y tiempo discreto. Los valores observados de un proceso estocástico se denominan *realización*.
En este trabajo, el término serie de tiempo hará referencia al proceso estocástico o a la realización, quedando claro, del contexto, a cual de los dos conceptos estaráse refiriendo.

2.1.1. Modelos clásicos de series de tiempo

Algunos de los modelos clásicos que definen series de tiempo son los siguientes:
White noise, es una clase de serie de tiempo generada por variables aleatorias no correlacionadas w_t, con media 0 y varianza finita σ_w^2. El proceso se denota $w_t \sim wn(0, \sigma_w^2)$. En la gráfica de arriba en la Figura en 2.2 se ve una colección de 500 variables aleatorias, con $\sigma_w^2 = 1$, cada realización fue graficada por orden de aparición.

Moving average, es una técnica que permite suavizar el comportamiento de una serie de tiempo. Por ejemplo considérese el reemplazo de las variables w_t, del caso anterior, por el promedio del valor actual de w_t y su vecindario inmediato pasado y futuro:

$$v_t = \frac{1}{3}(w_{t-1} + w_t + w_{t+1}) \tag{2.1}$$

La gráfica del medio en la Figura en 2.2 muestra la serie de tiempo resultante luego de aplicar *moving average* a la serie de tiempo anterior. Se observa que las oscilaciones lentas son las que más aparecen y algunas de las oscilaciones rápidas de la primera serie son descartadas.

Autoregression, considérese la serie *white noise* w_t como entrada para la siguiente serie de tiempo:

$$x_t = x_{t-1} + 0{,}9x_{t-2} + w_t \tag{2.2}$$

La Ecuación 2.2 representa una regresión del valor actual x_t de una serie de tiempo en función de los dos valores pasados de la serie más una variable aleatoria w_t representando ruido. En la gráfica de abajo en la Figura en 2.2 se ve una realización para 500 valores generados por la serie de tiempo dada en la Ecuación 2.2.

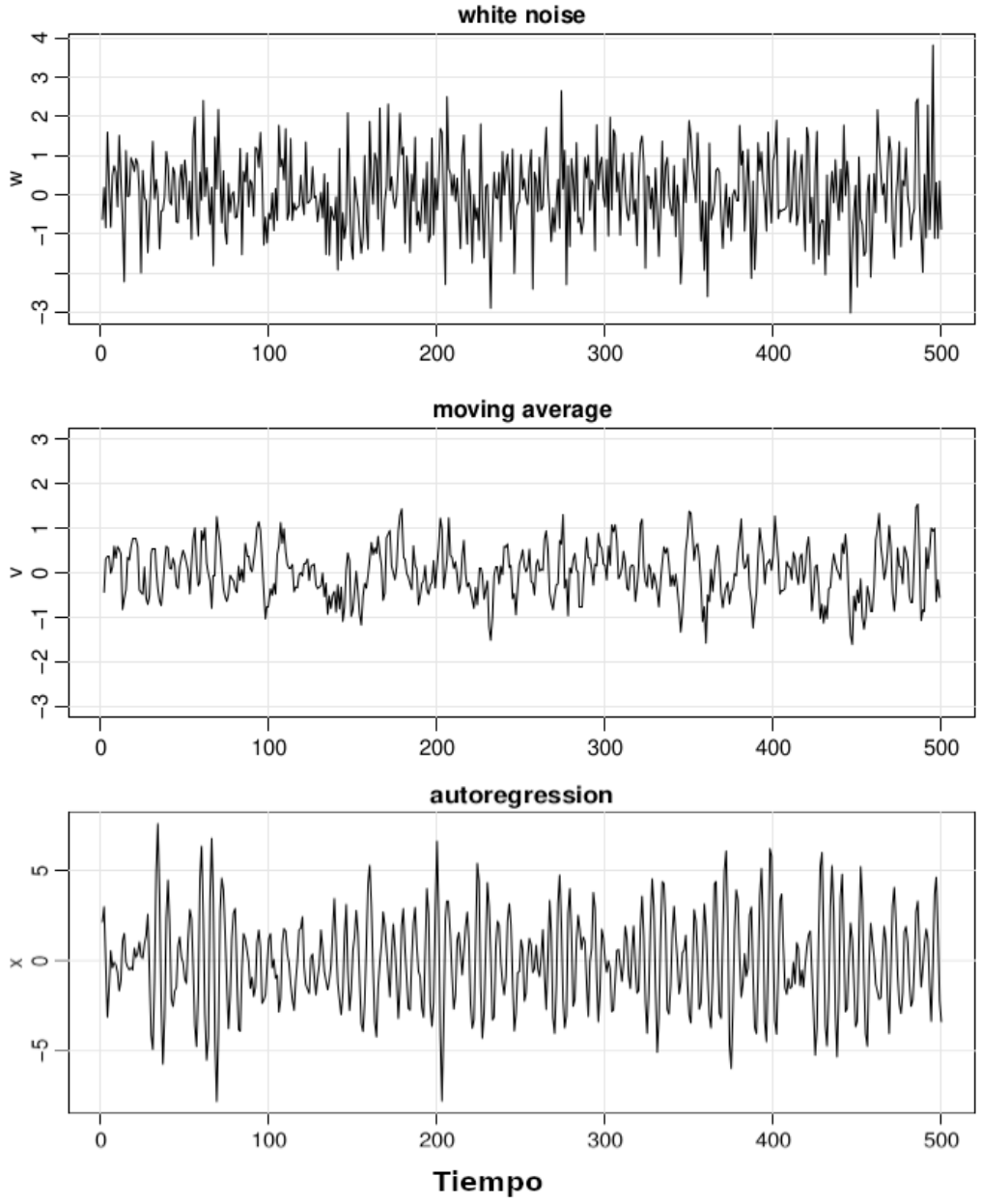


Figura 2.2: La gráfica de arriba representa el proceso estocástico gaussiano $w_t \sim iidN(0, \sigma_w^2)$. La gráfica del medio aplica la técnica de suavizado *moving average* a los w_t anteriores. La gráfica de abajo muestra una serie de tiempo para un modelo autoregresivo. Figuras extraídas de la página 10 a la 11 del Libro [14].

Forecasting: La meta del forecasting o pronóstico sobre una serie de tiempo es predecir valores futuros de la serie de tiempo $x_{n+m}, m = 1, 2, \ldots$ basados en los datos colectados hasta el presente $x_{1:n} = \{x_1, x_2, \ldots, x_n\}$.

2.1.2. Modelos simples de forecasting

Algunos modelos simples de forecasting, que pueden utilizarse como benchmark son *average method*, *seasonal naïve method* y *drift*.

Average method, pronostíca el valor $\hat{x}_{n+m}$ para un tiempo $n+m$, como el promedio de los valores de la serie $\{x\}_{t=1:n}$:

$$\hat{x}_{n+m} = (x_1 + \ldots + x_n)/n \tag{2.3}$$

Seasonal naïve method, es útil en casos como el siguiente: cuando el cliente para cualquier hora del día dada, consume valores similares de energía que los consumidos para la misma hora del día anterior, en este caso **Seasonal naïve method** establece que el valor a predecir para cierta hora del día sea igual al valor observado a la misma hora del último día en la serie de tiempo:

$$\hat{x}_{n+m} = x_{n+m-l(k+1)} \tag{2.4}$$

en donde l es la frecuencia de la estacionalidad, por ejemplo si la serie de tiempo toma 8 mediciones por día del hogar del cliente, el consumo presenta una estacionalidad diaria $l = 8$ y k es igual a la parte entera de $\frac{(m-1)}{l}$.

Drift method, predice aplicando una interpolación lineal:

$$\hat{x}_{n+m} = x_n + \frac{m}{n}\sum_{i=2}^{n}(x_i - x_{i-1}) = x_n + m(\frac{x_n - x_1}{n}) \tag{2.5}$$

La Figura 2.3 presenta los resultados de estos tres métodos prediciendo el consumo de energía de un día de un cliente, usando como información el consumo observado en los tres días anteriores.

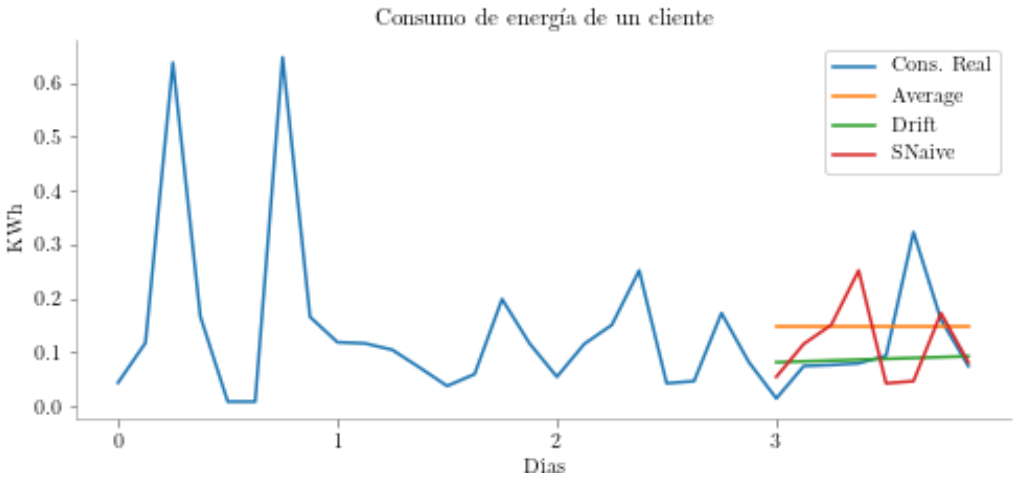


Figura 2.3: Se observa que el pronóstico de **Average** es constante, ya que siempre predice la media de $\{x\}_{t=1:n}$. La interpolación realizada por **Drift** es una recta apenas creciente, debido a que $x_1 < x_n$ en $\{x\}_{t=1:n}$. Mientras que **Seasonal naïve** copia los valores consumidos el día anterior y predice que estos serán los consumidos en el cuarto día, en este caso el parámetro estacional $l = 8$.

2.1.3. Patrones en las series de tiempo

Las series de tiempo evidencian patrones en su estructura, conocer cuales son los patrones involucrados en una serie permite a los modelos incorporar esta información para hacer predicciones más exactas.

- **Trend**, o tendencia, existe en la serie cuando hay una disminución o un aumento, a largo plazo, en los datos.
- **Seasonal**, o estacionalidad, un patrón estacional ocurre cuando una serie de tiempo es afectada por factores estacionales tales como el momento del año o el día de la semana. La estacionalidad siempre es de una frecuencia[1] fija y conocida.
- **Cyclic**, o cíclico, un ciclo ocurre cuando los datos exhiben subidas y bajadas que no son de frecuencia fija.

La Figura 2.4 muestra algunos ejemplos de series de tiempo que exhiben alguna combinación de estos patrones.

[1] número de observaciones antes de que el patrón estacional se repita

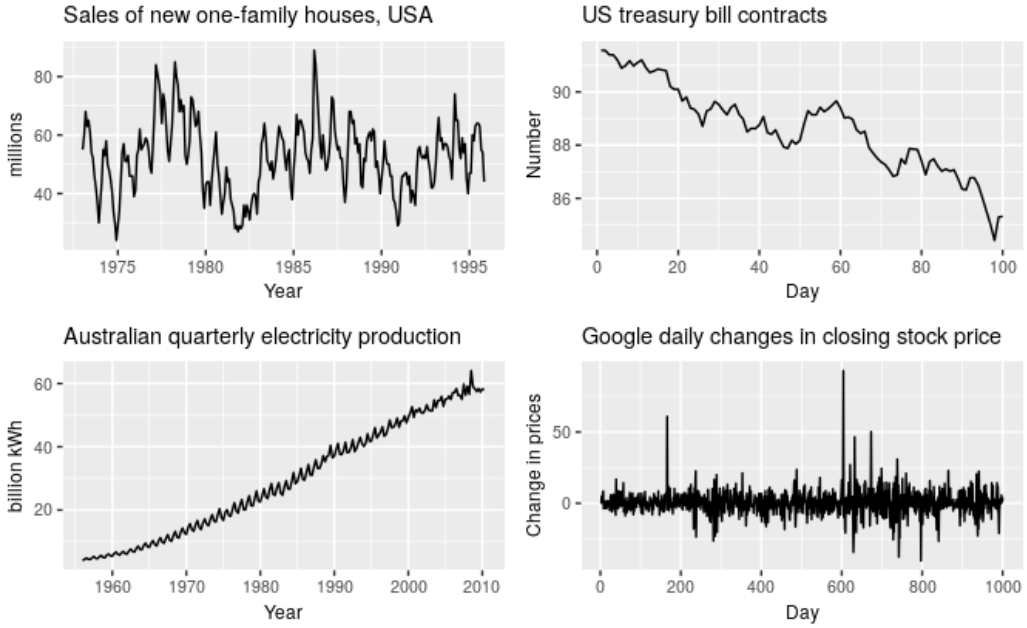


Figura 2.4: Imagen extraída de la Sección 2.3 del Libro [10], la figura muestra diferentes combinaciones de patrones. La gráfica de arriba a la izquierda muestra estacionalidad dentro de cada año, y un comportamiento cíclico para periodos entre 5 y 10 años. La gráfica de arriba a la derecha exhibe una tendencia a la baja, no se refleja estacionalidad ni comportamiento cíclico. La gráfica de abajo a la izquierda muestra un fuerte comportamiento de estacionalidad y tendencia al crecimiento, no hay evidencias de comportamiento cíclico. La gráfica de abajo a la derecha no presenta ninguno de los tres patrones.

2.1.4. Métricas de evaluación

Dos métricas para evaluar el error entre los valores predichos por una técnica y los valores reales son *Root Mean Squared Error* (RMSE) y *Mean Absolute Error* (MAE). Supóngase que se quieren predecir n valores. En donde $\hat{x}_t$ es el valor predicho y x_t el valor real $\forall t \in 1, \ldots, n$. RMSE se define por la Ecuación 2.6 y MAE por la Ecuación 2.7.

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(x_t - \hat{x}_t)^2} \tag{2.6}$$

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|(x_t - \hat{x}_t)| \tag{2.7}$$

Ambas métricas expresan un error promedio en las predicciones. MAE expresa el promedio de la diferencia en valor absoluto entre los valores reales y los predichos. RMSE aplica la raíz cuadrada al promedio de la diferencia al cuadrado entre los valores reales y los predichos. RMSE al aplicar el cuadrado a cada diferencia, penaliza sobre todo a las predicciones $\hat{x}_t$ con mucha diferencia respecto al valor real x_t.
En la Tabla 2.1 se muestran los valores en MAE y RMSE para las predicciones de las técnicas de la Figura 2.3.

	MAE	RMSE
Average	0,0823	0,0089
Drift	0,0521	0,0080
SNaive	0,0841	0,0146

Tabla 2.1: La técnica Drift es la que menor MAE y RMSE presenta para las predicciones de la Figura 2.3. El valor medido para RMSE, en todas las técnicas, es menor al valor medido MAE. Esto ocurre porque RMSE al aplicar el cuadrado a la diferencia de los valores reales vs. predichos disminuye el error, si la diferencia es menor a 1, y para este ejemplo todas las diferencias entre predicción y valor real, en todas las técnicas, son menores a 1.

Ante valores altos de RMSE o MAE puede interesar para cada métrica, cual de los términos de la sumatoria aportó el mayor error. Para esto se definen $MaxError_{RMSE}$ Ec. 2.8 y $MaxError_{MAE}$ Ec. 2.9.

$$MaxError_{RMSE} = max_{t=1,\dots,n}\left\{\frac{1}{n}(x_t - \hat{x}_t)^2\right\} \tag{2.8}$$

$$MaxError_{MAE} = max_{t=1,\dots,n}\left\{\frac{1}{n}|(x_t - \hat{x}_t)|\right\} \tag{2.9}$$

En la Figura 2.5 se muestran cuales fueron los términos de máximo error en MAE y RMSE, para las predicciones de cada técnica en la Figura 2.3.

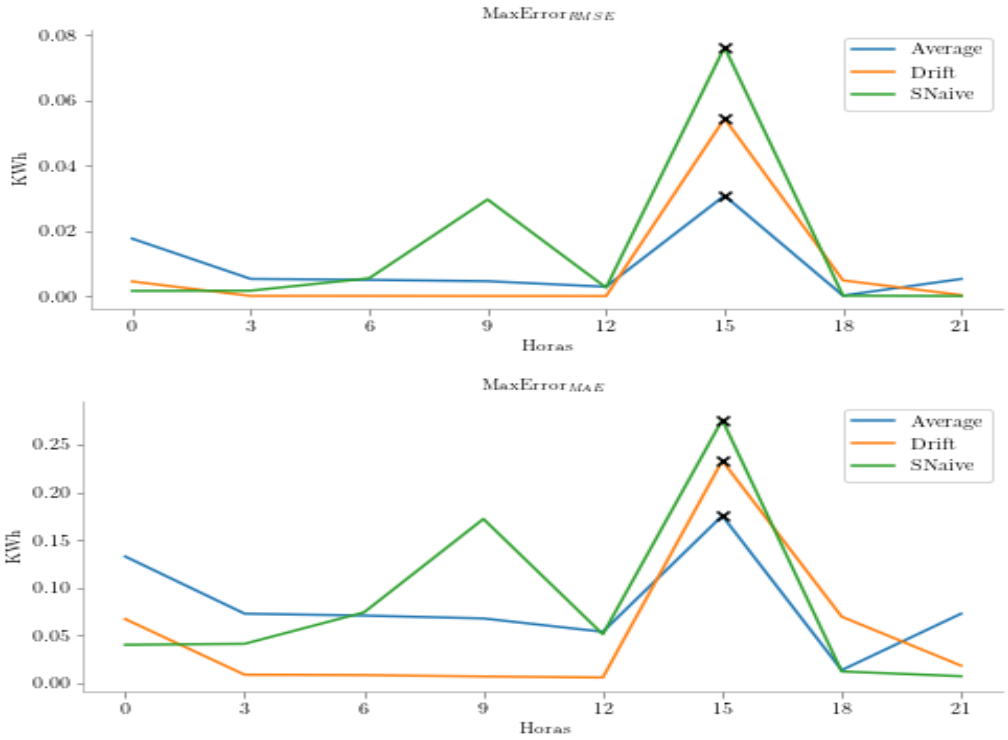


Figura 2.5: En la gráfica de arriba se marcan con una cruz cuales fueron los los términos de máximo error en RMSE, para las técnicas de predicción de la Figura 2.3. La gráfica de abajo es análoga, pero se marcan con una cruz los términos de máximo error en MAE. Se puede observar que la predicción del valor de consumo a las 15 horas fue el que mayor error aportó a las métricas MAE y RMSE, para todas las técnicas. Se observa que SNaive, la técnica con peor resultado en ambas métricas (ver Tabla 2.1), tuvo dos picos altos de error en la predicción: a las 9 horas y a las 15 horas.

F_1-Score se emplea en procesos de clasificación, en donde un algoritmo trata de clasificar o etiquetar elementos, la Tabla 2.2 refleja los posibles resultados para una clasificación binaria de elementos. Esta técnica es útil al momento de predecir cuando el espacio de valores es discreto, y la predicción puede ser vista como un proceso en el cual se quiere asignar la clase (valor real) correcta al valor a predecir. Esta métrica permite evaluar que tan buena es la técnica acertando un valor dado, del conjunto de valores posibles. F1-Score se define por la Ecuación 2.10.

		Real	
		Verdadero	Falso
Pred.	Pred. Verdadero	VP	FP
	Pred. Falso	FN	VN

Tabla 2.2: VP: Verdaderos Positivos, son todos los elementos con etiqueta de Verdadero clasificados como Verdaderos. FP: Falsos Positivos, elementos con etiqueta de Falso clasificados por el algoritmo como Verdaderos (error). FN: Falsos Negativos, elementos con etiqueta de Verdadero clasificados como Falsos (error). VN: Verdaderos Negativos, elementos con etiqueta de Falso clasificados por el algoritmo como Falsos.

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{2.10}$$

$$precision = \frac{VP}{VP + FN} \tag{2.11}$$

$$recall = \frac{VP}{VP + FP} \tag{2.12}$$

F1-Score será utilizado para evaluar la asignación de clientes a perfiles de consumo, y para evaluar la predicción de consumo cuando este es discretizado en solo algunos valores posibles (por ejemplo en el caso del modelo de las monedas). En general RMSE y MAE se usarán para evaluar la calidad de predicción en el consumo.

2.2. Método de predicción para el consumo desagregado de energía

Esta sección resume el trabajo presentado por Laurinec et al. [12], el cual es la referencia principal para este proyecto, dada la cercanía al problema que se quiere resolver, y la similitud de las técnicas empleadas. El método aquí descrito propone una predicción de consumo basada en agrupamientos (clusters) de clientes con perfiles de consumo similares. Se toma de cada perfil un representativo, eliminando ruido e irregularidades, como entrada para los métodos de forecasting. Por último la predicción de consumo para cada cliente es recuperada a partir de la predicción hecha sobre el representativo del cluster al cual el cliente pertenece.
A continuación se define el concepto de cluster, y la técnica de clustering K-means

empleada en el método. Luego se describe el método propuesto para series de tiempo y las técnicas de forecasting utilizadas. Por último las pruebas y resultados del método.

2.2.1. Algoritmos de clustering

Los algoritmos de clustering (agrupamiento) tienen como objetivo crear clusters (grupos) de objetos que sean coherentes internamente, pero claramente diferenciables unos de otros. Es decir, los objetos dentro de un mismo cluster deben ser lo más parecidos entre sí, y los objetos en un cluster dado deben ser lo más distinto posible a objetos pertenecientes a otro cluster. Estos algoritmos entran en la categoría de aprendizaje no supervisado. En la Figura 2.6 se observa una forma de separar en tres grupos o clusters a los objetos, representados por sus features *Feature* 1, *Feature* 2, que logra el objetivo de conseguir para cada cluster una coherencia interna y una clara diferenciación respecto a los otros.

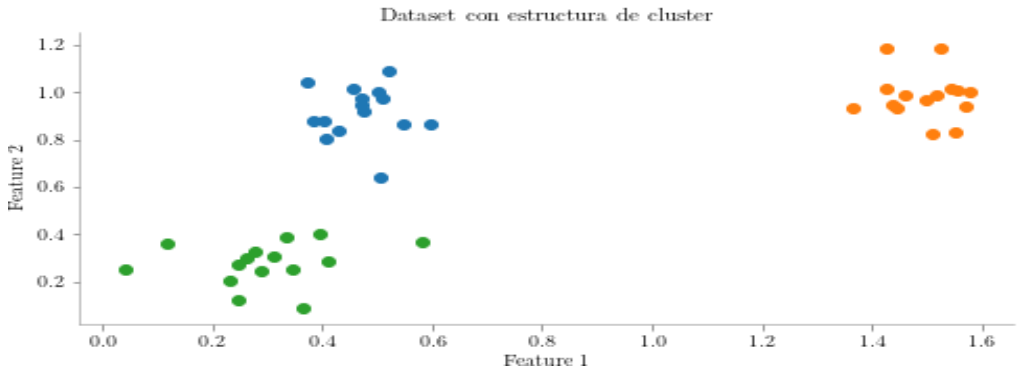


Figura 2.6: En la figura se aprecia una clasificación posible del dataset de este ejemplo en tres clusters o grupos principales. Las dimensiones *Feature 1* y *Feature 2* fueron las usadas para agrupar a los objetos. Figura extraída del libro de Christopher D. Manning et al. [3], Figura 16.1 del Capítulo 16.

K-means

K-means es un algoritmo de agrupamiento categorizado como *flat hard clustering*. Los algoritmos de clustering *flat*, crean un conjunto de clusters planos, sin una estructura específica que los relacione unos con otros, a diferencia de los algoritmos de cluster jerárquicos. Mientras que los algoritmos de clustering *hard* computan una asignación para cada objeto, en la cual cada uno es miembro de un

solo cluster, a diferencia de los algoritmos de tipo *soft*, en donde la asignación de un objeto es sobre una distribución sobre todos los clusters [3, Cap.16].
El problema de clustering se define así, dado:

- un conjunto de objetos, $X = \{x_1, \ldots, x_N\}$;
- un número deseado de clusters, K;
- una función objetivo que evalúa la calidad del cluster.

Se quiere computar una asignación $\gamma : X \rightarrow \{1, \ldots, K\}$ que minimice (o maximice) la función objetivo.
En K-means la función objetivo es la suma de la distancia euclídea cuadrática media entre los objetos $\vec{x}$ pertenecientes a un cluster ω, y el centroide $\vec{u}(\omega)$ de dicho cluster. Sea $\vec{u}$ el vector media del cluster o centroide ω:

$$\vec{u}(\omega) = \frac{1}{|\omega|} \sum_{\vec{x} \in \omega} \vec{x} \tag{2.13}$$

El algoritmo tiene la función objetivo RSS Ec. 2.15 a minimizar:

$$RSS_k = \sum_{\vec{x} \in \omega_k} |\vec{x} - \vec{u}(\omega_k)|^2 \tag{2.14}$$

$$RSS = \sum_{k=1}^{K} RSS_k \tag{2.15}$$

Algunas de las propiedades del algoritmo son las siguientes:

- La complejidad del algoritmo es de orden lineal en todos sus factores relevantes.
- El agrupamiento final depende de la elección inicial al azar de los K clusters a iterar. Existen heurísticas para evitar malas elecciones iniciales.
- El valor de la función objetivo RSS decrece monótonamente en cada iteración del algoritmo.

En el Apéndíce C se describe el algoritmo K-means, incluyendo una prueba para las propiedades arriba enunciadas.

2.2.2. Método de Laurinec

En este método se quiere predecir el consumo de energía durante 7 días, para N clientes pertenecientes a una red de consumo, esto se hace en 7 iteraciones, una por día.
En cada iteración se utiliza el consumo real, de cada cliente, registrado en las últimas 3 semanas. Es decir, si se está prediciendo el tercer día entonces se usan los últimos 21 días de consumo real (lo que incluye los últimos 2 días predichos).
Este procedimiento refleja el problema más común, que es predecir el siguiente día de consumo.
Una vez obtenidos los registros de consumo de los clientes, estos son normalizados aplicando la función z-score definida de la siguiente manera:

$$\hat{x}_i = \frac{x_i - \mu}{\sigma} \tag{2.16}$$

en donde $\hat{x}_i$ es el valor normalizado, x_i es el valor original para $i = 1, \ldots, n$, en donde n representa la cantidad de observaciones para la serie de tiempo de los datos que están siendo normalizados, luego μ y σ son la media y la desviación estándar muestral de la serie.
A continuación se procede a agrupar a los clientes en k clusters.
Las técnicas de predicción se aplican sobre estos k clusters y las predicciones para cada cliente se obtienen desnormalizando las predicciones hechas sobre el cluster al cual cada cliente pertenece, el proceso de desnormalización se hace de acuerdo a la Ecuación 2.16 de la siguiente manera:

$$x_i = \hat{x}_i \sigma + \mu \tag{2.17}$$

Significando en este caso $\hat{x}_i$ la i-ésima predicción para un cluster cl dado y x_i la i-ésima predicción para el cliente (perteneciente al cluster cl) con media y desviación estándar muestral μ y σ.
Esta dinámica se aplica para cada nuevo día a predecir. En la Figura 2.7 se observa el esquema del método para un día de predicción. El algoritmo del método (Alg. 8) y su descripción se encuentran en la Sección B.1 del Apéndice B.

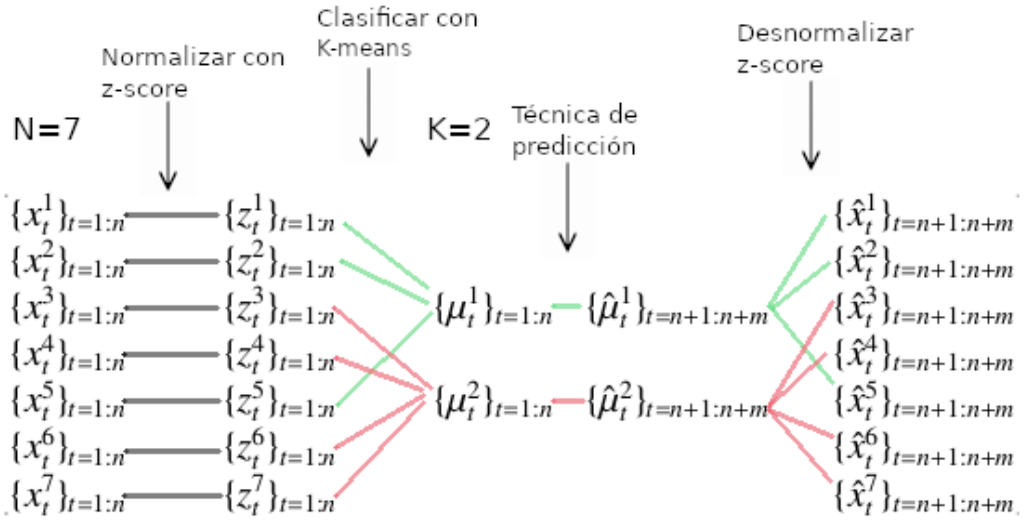


Figura 2.7: La dinámica comienza con las series de tiempo conteniendo la información de las últimas tres semanas de consumo de cada uno de los N clientes. A cada serie de tiempo $\{x_t^j\}, j \in 1, \ldots, N$ se le aplica la función z-score: $z_i^j = \frac{x_i^j - \mu}{\sigma}, i \in 1, \ldots, n$, guardando para cada una de ellas la media μ y la desviación estándar muestral σ. La clasificación empleando K-means se realiza empleando como features una representación de las series de tiempo normalizadas (Ecuación B.3), obteniéndose K clusters, véase la Sección B.1 del Apéndice B para los detalles de esta representación. Sobre los centroides de cada cluster: $\{\mu_t^k\}, k \in 1, \ldots, K$, se aplican a continuación las diferentes técnicas de predicción para obtener los m nuevos pronósticos para el nuevo día. Como último paso se desnormalizan las predicciones para cada cliente a partir de las predicciones hechas para el centroide del cluster al cual el cliente pertenece: $\hat{x}_i^j = \hat{\mu}_i^k \sigma + \mu$, siendo $\hat{x}_i^j, i \in n+1, \ldots, n+m$ las predicciones para el j-ésimo cliente, $\hat{\mu}_i^k$ las predicciones para el k-ésimo centroide al cual el cliente j-ésimo pertenece y μ y σ la media y la desviación estándar muestral de los datos de las últimas tres semanas de consumo del j-ésimo cliente respectivamente.

2.2.3. Evaluaciones y experimentos

Para evaluar la performance de los métodos basados en clustering Laurinec et al. utiliza dos datasets:

- Dataset irlandés, colectado por la *Irish Commision for Energy Regulation* (CER), disponible en el archivo de datos científicos irlandés[2].

[2]http://www.ucd.ie/issda/data/commissionforenergyregulationcer/

- Dataset eslovaco, colectado por el proyecto *International Centre of Excellence for Research of Intelligent and Secure Information-Communication Technologies and Systems*

Para cada uno de estos datasets se eligieron conjuntos de prueba de tres semanas para investigar la performance, cada una de estas fechas tiene una ventana de 21 días para el entrenamiento de los modelos:

- Dataset irlandés, del 1-2-2010 al 21-2-2010
- Dataset eslovaco, del 10-2-2014 al 2-3-2014

La frecuencia de las mediciones evaluadas en los datasets es de 30 minutos. Las métricas utilizadas fueron MAE (Ec. 2.7) y RMSE (Ec. 2.6).

2.2.4. Resultados

Laurinec et al.[12] propone 4 técnicas que emplean clustering y una que no emplea clustering para predecir, estas técnicas se describen en la Sección B.1 del Apéndice B.
La principal diferencia entre ambos tipos de técnicas es que las que emplean clustering siguen los pasos de la Figura 2.7 para realizar las predicciones, mientras que la técnica que no emplea clustering predice para cada usuario individual que el consumo de energía, para el día que se está queriendo predecir, será el mismo el que consumido para el mismo día de la semana pasada.
Para el dataset irlandés la mejor técnica que emplea clustering obtuvo un RMSE promedio un 0,43 % mejor que la técnica que no emplea clustering y en MAE también fue un 0,41 % superior.
Para el dataset eslovaco la mejor técnica que emplea clustering obtuvo un RMSE promedio un 3,83 % mejor que la técnica que no emplea clustering y en MAE también fue un 4,02 % superior.
Los resultados de este trabajo previo motivan la pregunta científica principal de este proyecto, que es evaluar la mejora que surge de considerar series similares para predecir series de tiempo de las cuales aún no se tiene suficiente información. Vale aclarar que este aspecto no fue estudiado por Laurinec et al. [12].

2.3. Matrix Completion

Hasta el momento las técnicas de predicción vistas en el estado del arte se han aplicado sobre series de tiempo. Una alternativa es pensar, en el contexto del con-

sumo de energía, en una matriz $M_{N\times D}$, en donde cada fila $i \in 1, \ldots, N$ representa el consumo de energía en un hogar, y cada columna representa a los diferentes tiempos en que fueron registradas las observaciones para todos los hogares. Supóngase que se quiere predecir el consumo de energía para los últimos $D_p < D$ de estos tiempos. En la matriz M esto se representa con las últimas D_p columnas sin valores observados, por ejemplo inicializando cada entrada igual a $indefinido$. También es válido que haya algunas entradas sin observar en las $D - D_p$ columnas anteriores. En esta situación se puede pensar en encontrar dos matrices $X_{N\times K}, Y_{K\times D}$ tales que XY coincida con los valores de M para cada entrada observada en M y que tenga completadas aquellas entradas que en M no tienen observaciones. *Matrix completion*, es un algoritmo que resuelve este problema aplicando sucesivos sistemas de ecuaciones lineales para ir descubriendo los valores de las matrices X e Y, durante las iteraciones del completado.

2.3.1. Definición del problema

Completar una matriz a partir de pocas entradas dadas es un problema fundamental con aplicaciones en áreas como machine learning, estadística y reconstrucción de señales. Una aplicación conocida es el problema de recomendaciones de Netflix [3], en donde se tiene una matriz cuyas filas representan clientes y las columnas series o películas. Cada celda indica una puntuación dada por el cliente para una serie o película. Como es de esperarse los clientes no califican a todos los productos ofrecidos por Netflix y por ende quedan en la matriz espacios vacíos, sin información:

	Muñeca brava	**Kachorra**	**Sos mi vida**	**Patito feo**	**Miss Tacuarembó**
Mathias	3	1	4	-	2
Matilde	8	4	-	8	5
Sebastián	9	7	10	5	-
Ana	6	-	7	8	4
Mauricio	5	3	8	-	8

Tabla 2.3: Las columnas representan series o películas y las filas a los usuarios de Netflix. Cada entrada observada (i, j) es una calificación dada por el i-ésimo usuario a la j-ésima serie o película. Los guiones $(-)$ representan la ausencia de calificación del i-ésimo usuario para la j-ésima serie o película.

El problema reside en determinar posibles recomendaciones para los clientes,

[3] Véase el siguiente enlace: Recomendaciones basadas en factorización de matrices.

a partir del completado de la matriz. Una suposición natural para este tipo de problemas es aceptar que las matrices sean de rango bajo.

Definición 2.3.1. Rango bajo
Si una matriz $M_{m\times n}$ puede ser factorizada en dos matrices $X_{m\times k}$ y $Y_{k\times n}$. De manera que $XY = M$ entonces M tiene *rango* (no más grande que) k. Si $k \ll min(m, n)$, entonces se dice que M tienen *rango bajo*.

Definición 2.3.2. $[m]$, denota al conjunto de naturales: $1, 2, \ldots, m$

Definición 2.3.3. Ω, denota a un subconjunto de pares de naturales incluido en el producto cartesiano $[m] \times [n]$

Definición 2.3.4. $\mathcal{P}_\Omega : \mathcal{R}^{m\times n} \rightarrow \mathcal{R}^{m\times n}$ denota la proyección de la matriz X sobre el par de índices en Ω:

$$\mathcal{P}_\Omega(X[i, j]) = \begin{cases} X[i, j] & \text{si } (i, j) \in \Omega \\ indefinido & \text{si } (i, j) \notin \Omega \end{cases}$$

El problema es formalizado de la siguiente manera [13]:

Definición 2.3.5. Matrix Completion Problem
Para una matriz desconocida $M \in \mathcal{R}^{m\times n}$, de rango a lo sumo k, dado $\Omega \subseteq [m]\times[n]$, $\mathcal{P}_\Omega$ y k, el problema del completado de la matriz de rango bajo es encontrar una matriz $X \in \mathcal{R}^{m\times n}$ tal que:

$$rango(X) \leq k \text{ y } \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M)$$

2.3.2. Esquema del algoritmo

En el Apéndice E se describe el algoritmo para el completado de matrices. A continuación se presenta un ejemplo de completado, aplicado a la matriz de la izquierda en la Figura 2.8.

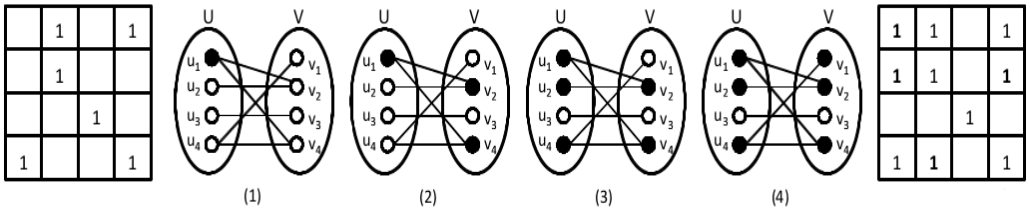


Figura 2.8: En la figura se observa el proceso de *infección* llevado a cabo por el algoritmo *Information Cascading Matrix Completion* (ICMC) propuesto por Raghu Meka et al. [13]. Cuando no hay más vértices que infectar el proceso de detiene. La matriz de la izquierda es la matriz inicial. En la matriz de la derecha se representan con un 1 en los espacios anteriormente vacíos las nuevas entradas completadas. Imagen extraída de Gonca et al. [9].

La Figura 2.8 muestra el proceso de completado para una matriz $M_{4\times4}$ a partir de dos matrices $X_{4\times1}, Y_{1\times4}$ tales que $M = XY$. Inicialmente X e Y no tienen ninguna fila ni columna con valores observados. En primera instancia se crea el grafo bipartito $G = U \cup V$, en donde los vértices $u_i, i = 1, \ldots, 4$ representan las filas de la matriz, los vértices $v_i, i = 1, \ldots, 4$ las columnas, y las aristas (u_i, v_i) representan las conexiones dadas por la matriz a completar a la izquierda de la Figura. Existe una arista (u_i, v_j), si el valor $M[i, j]$ está definido.

- En el paso (1), primer grafo de la Figura 2.8, se marcan k vértices de U, en este caso k=1 y el vértice marcado es u_1. La matriz X se inicializa con $[x_{11}]$ igual a la matriz identidad I_1. Para el caso general de todos los vértices marcados se eligen k de ellos y las filas asociadas de X a estos vértices se igualan a la matriz identidad I_k.

$$X = \begin{bmatrix} 1 \\ x_{21} \\ x_{31} \\ x_{41} \end{bmatrix}, Y = \begin{bmatrix} y_{11} & y_{12} & y_{13} & y_{14} \end{bmatrix}$$

- En el paso (2), segundo grafo de la Figura 2.8, se marcan aquellos vértices de V que tengan al menos k aristas a vértices de U ya marcados por el algoritmo y se resuelven linealmente las filas de Y que correspondan a los nuevos vértices marcados. En el ejemplo los vértices marcados son v_2, v_4. Las ecuaciones a resolver son:

$$M[1, 2] = X[1, 1]Y[1, 2] \rightarrow 1 = 1 \cdot y_{12}$$

$$M[1,4] = X[1,1]Y[1,4] \rightarrow 1 = 1 \cdot y_{14}$$

La matriz Y se actualiza con los nuevos valores

$$X = \begin{bmatrix} 1 \\ x_{21} \\ x_{31} \\ x_{41} \end{bmatrix}, Y = \begin{bmatrix} y_{11} & 1 & y_{13} & 1 \end{bmatrix}$$

- En el paso (3), tercer grafo de la Figura 2.8, se marcan aquellos vértices de U que tengan al menos k aristas a vértices de V ya marcados por el algoritmo y se resuelven linealmente las filas de X que correspondan a los nuevos vértices marcados. En el ejemplo los vértices marcados son u_2, u_4.
  Las ecuaciones a resolver son:

$$M[2,2] = X[2,1]Y[1,2] \rightarrow 1 = x_{21} \cdot 1$$

$$M[4,4] = X[4,1]Y[1,4] \rightarrow 1 = x_{41} \cdot 1$$

  La matriz X se actualiza con los nuevos valores:

$$X = \begin{bmatrix} 1 \\ 1 \\ x_{31} \\ 1 \end{bmatrix}, Y = \begin{bmatrix} y_{11} & 1 & y_{13} & 1 \end{bmatrix}$$

- En el paso (4), cuarto grafo de la Figura 2.8, el único nuevo vértice marcado es v_1.
  La única ecuación a resolver es:

$$M[4,1] = X[4,1]Y[1,1] \rightarrow 1 = 1 \cdot y_{11}$$

  La matriz Y se actualiza con los nuevos valores:

$$X = \begin{bmatrix} 1 \\ 1 \\ x_{31} \\ 1 \end{bmatrix}, Y = \begin{bmatrix} 1 & 1 & y_{13} & 1 \end{bmatrix}$$

- En el paso (5) no hay nuevos vértices por marcar. El algoritmo termina de completar las entradas faltantes mediante el algoritmo heurístico *Alternating Least Squares* [11].

$$X = \begin{bmatrix} 1 \\ 1 \\ 0{,}94871357 \\ 1 \end{bmatrix}, Y = \begin{bmatrix} 1 & 1 & 0{,}94871357 & 1 \end{bmatrix}$$

- La matriz completada $M = XY$ queda:

$$M = \begin{bmatrix} 1. & 1. & 0{,}94865908 & 1. \\ 1. & 1. & 0{,}94865908 & 1. \\ 0{,}94871357 & 0{,}94871357 & 0{,}90000574 & 0{,}94871357 \\ 1. & 1. & 0{,}94865908 & 1. \end{bmatrix}$$

Observaciones sobre el problema:

- Los primeros resultados no triviales que garantizan una solución al problema agregan suposiciones adicionales a la matriz M. Las técnicas que emplean estos resultados dependen que el grafo asociado a la matriz sea un grafo Erdös–Rényi, en donde el grado de los vértices sigue una distribución de Poisson y se concentran alrededor de la media [13, Cap.1. Intr.]. Esto no ocurre en grafos asociados a problemas del mundo real, por ejemplo en la WorldWideWeb el grado de los vértices se desvía significativamente de la distribución de Poisson y en datasets como el de Netflix el número de vértices de grado d sigue una distribución de ley potencial: $d^{-\beta}$ para una constante β.

- El algoritmo *Information Cascading Matrix Completion (ICMC)* [13, Cap.3. ICMC] exhibe buen comportamiento con grafos cuyos vértices están distribuidos por una relación de ley potencial

  $$\text{\#Cant. Vértices de grado d} = C \times (\text{grado d})^p$$

  Véase la Figura E.1 del Apéndice E para un ejemplo de gráfica que sigue una relación de ley potencial.

- ICMC es un enfoque que explota la estructura del grafo subyacente y obtiene mejores resultados para problemas con vértices distribuidos por una relación de ley potencial respecto a otras técnicas como:

- *Singular Value Thresholding (SVT)*
- *Spectral Matrix Completion (SMC)*
- *Alternating Least Squares (ALS)*

2.4. Explanation tables

Como se ha visto en el estado del arte una técnica de predicción puede tener varios parámetros y cada uno de éstos puede tomar un valor de un rango de valores posibles. Una técnica que realice clustering usando K-means deberá elegir la cardinalidad k del algoritmo, una técnica que emplee series de tiempo, al momento del entrenamiento deberá separar los datos, en datos de entrenamiento (DE) y datos de prueba (DP). En *Matrix completion* se debe elegir el rango k de la matriz. Al momento de evaluar la performance de las técnicas para las diferentes combinaciones de valores que estos parámetros pueden tomar, la tabla de resultados asociada puede ser de un tamaño tal (decena o centenas de entradas) que no permita extraer toda la información representada en los resultados a simple vista. Es deseable tener una técnica de resumen automático que sea capaz de asistir en la interpretación de los resultados. El trabajo presentado por ElGebaly et al. [6] propone la técnica *Explanation table*, para resolver el problema de generar resúmenes informativos para una tabla de datos dada de manera eficiente en tiempo de ejecución. En el Apéndice F se presenta el funcionamiento de la técnica, así como los detalles de la implementación que fue realizada en este proyecto. A continuación se bosqueja el problema y se enuncian las cualidades deseadas.
Dado un dataset $D_{m\times n}$, en el cual las primeras $n-1$ columnas representan atributos categóricos que permiten clasificar a cada fila y la n-ésima columna expresa un atributo binario indicador de resultado positivo (1) o negativo (0), el problema es determinar como se puede construir un resumen interpretable e informativo de los factores que afectan al atributo binario en termino de la combinación de valores de los atributos categóricos. Por ejemplo, sea la Tabla 2.4 el registro de la actividad de un deportista. La Tabla 2.5 representa una *explanation table* para la Tabla 2.4. Las cualidades que debe reunir una *explanation table* son:

- *Interpretabilidad*, la tabla debe ser interpretable, es decir revelar tendencias macro de los datos. En los datos reales, en general, los datos se superponen, por lo tanto los patrones deben permitir expresar los solapamientos, como (Sáb., $*$, $*$) y ($*$, $*$, Banana) en la Tabla 2.5.

- *Informatividad*, dados los atributos categóricos, la tabla debería permitir la reconstrucción a nivel micro de los atributos binarios con una exactitud mensurable.
- *Eficiencia*, la tabla debe ser eficiente en tiempo durante su construcción, aún para grandes datasets.

Algunas consideraciones sobre *Explanation Tables*:

- Computar la *explanation table* óptima es un problema NP-Hard ver Sección F.2 del Apéndice F.
- En el trabajo de ElGebaly et al. [6] se presentan dos heurísticas denominadas *Flashlight* y *Laserlight*, cada una de estas técnicas reduce el tamaño del espacio de patrones candidatos, en donde buscar el siguiente patrón a agregar. Ambas técnicas negocian entre *eficiencia* e *informatividad*.
- La técnica *Flashlight* es la implementada en este trabajo.

En el Apéndice F se define el problema de resumir una tabla. Se definen los conceptos necesarios para abordar el problema y se detalla la construcción de una *explanation table* siguiendo la heurística *flashlight*. Se resumen también los resultados sobre *eficiencia* e *informatividad* obtenidos por ElGebaly et al. [6]. Por último se describe la implementación hecha en Python para este proyecto.

id	Día	Tiempo	Comida	Meta Alcanzada
1	Vie.	Amanecer	Banana	1
2	Vie.	Noche	Ens. Verde	1
3	Dom.	Atardecer	Avena	1
4	Dom.	Mañana	Banana	1
5	Lun.	Tarde	Avena	1
6	Lun.	Mediodía	Banana	1
7	Mar.	Mañana	Ens. Verde	0
8	Mié.	Noche	Hambuguesa	0
9	Jue.	Amanecer	Avena	1
10	Sáb.	Tarde	Nuez	0
11	Sáb.	Amanecer	Banana	0
12	Sáb.	Amanecer	Avena	0
13	Sáb.	Atardecer	Arroz	0
14	Sáb.	Mediodía	Tostadas	0

Tabla 2.4: Cada fila contiene información del día en que se realizó el ejercicio, de la comida previa y el momento del día en el que la actividad se realizó, junto con una columna final indicando con un 1 si la meta del ejercicio fue alcanzada y con 0 en caso contrario. Tabla extraída de ElGebaly et al. [6].

Día	Tiempo	Comida	**Meta Alcanzada** = 1	Total
*	*	*	0.5	14
Sáb.	*	*	0	5
*	*	Banana	0.75	4
*	*	Avena	0.75	4

Tabla 2.5: La tabla contiene patrones que especifican subconjuntos de tuplas, los patrones consisten de atributos categóricos o del símbolo '*' denotando todos los valores posibles. Cada patrón está asociado con el número de tuplas que abarca (**Total**) y la fracción que indica cuantas de esas tuplas tienen como resultado del atributo binario un 1. Tabla extraída de ElGebaly et al. [6].

Capítulo 3

Formalización del modelo

En la primera parte de este capítulo se definen: el modelo para la predicción del consumo de energía, la métrica a emplear y la función objetivo. En la segunda parte se define un experimento teórico más sencillo denominado *el modelo de las monedas*. El objetivo del *modelo de las monedas* es comprender el grado de importancia de conocer información como la temperatura y el empleo de técnicas de agrupamiento para realizar las predicciones. Se definen las métricas a emplear y el objetivo del experimento. Se presentan resultados y conclusiones del experimento.

3.1. Definición del modelo

Sea N el número de clientes de una red de consumo eléctrico. N_c de ellos tienen un histórico de consumo de energía de tamaño H. El histórico de consumo para un cliente c de los N_c es representado como una serie de tiempo $\{x_t^c\}_{t=1:H}$, con el supraíndice c indicando a que cliente corresponde dicha serie. N_{nc} de los clientes tienen una cantidad menor a H de observaciones, por haber comenzado a registrar las observaciones en un tiempo $t > 1$. La serie de tiempo para un cliente c en N_{nc} queda de la siguiente manera $\{x_t^c\}_{t=n:H}, n > 1$. Se cumple la igualdad $N = N_c + N_{nc}$. Los tiempos $1, \cdots, H$ en los que se registra el consumo de energía para los clientes, están separados por un intervalo constante de tiempo. Sea $\mathbf{H}_{N\times H}$ la matriz que guarda los históricos de consumo, cada fila $i = 1, \cdots, N$ contiene los valores observados para la serie de tiempo de cada cliente $\{x_t^{c_i}\}_{t=1:H}$. Si el cliente es uno de los N_{nc} clientes con histórico incompleto, con serie de tiempo $\{x_t^{c_i}\}_{t=n:H}, n > 1$, entonces las entradas de la matriz, para la fila i y columnas de la 1 a la $n-1$, $\mathbf{H}[i, 1:(n-1)]$ toman el valor *indefinido*. Cada columna corresponde

a uno de los tiempos $1, \cdots, H$ de observación.
Existen K perfiles de consumo y cada cliente pertenece a uno de ellos.
$T = \{y_t\}_{t=1:H}$ es el histórico de temperaturas registradas para cada uno de los H tiempos.
Dados dos perfiles distintos cualesquiera k_i, k_j, se asume que si dos clientes pertenecen a perfiles distintos, entonces sus consumos son lo suficientemente diferentes, esto se formaliza así: Dado un cliente c_1, perteneciente al perfil k_i e histórico de consumo $\{x_t^{c_1}\}$ y un cliente c_2, perteneciente al perfil k_j e histórico $\{x_t^{c_2}\}$, se pueden observar diferencias notorias en sus series de tiempo (ej. distinta estacionalidad, tendencias y/o distintos rangos en los valores de consumo). Y dados c_1, c_2 dos clientes pertenecientes al mismo perfil k_i, el comportamiento de sus series coincide (estacionalidad y tendencias similares, mismo rango de valores de consumo).
Para cada perfil de clientes, que se asume lo suficientemente diferente al resto, se quiere caracterizar su comportamiento de consumo basado en las temperaturas. Para esto se clasifica a las temperaturas registradas en T en varios grupos (por ej, temperaturas bajas, medias y altas), y se analiza el comportamiento de cada perfil de consumo para cada uno de estos grupos. Se espera para cada perfil, que sus clientes varíen el comportamiento de consumo dependiendo de la clase temperatura observada.
Supóngase, para simplificar, que existen dos clases de temperaturas T_A y T_B representando temperaturas altas y bajas respectivamente. Una forma sencilla de hacer esta clasificación es con la función $f : T \to \{T_A, T_B\}$:

$$f(y_t) = \begin{cases} T_A & \text{Si } y_t \geq 15 \\ T_B & \text{En otro caso} \end{cases} \tag{3.1}$$

Cada perfil k para cada clase de temperatura T_A, T_B, tendrá asociado un rango de valores de consumo $[c_{min}, c_{\text{máx}}]$ y una distribución de probabilidad sobre los valores c pertenecientes al rango. Esta distribución se puede expresar así:

$$p_{k,t}(x_t^{c_i} = h | T_X, \mathbf{H}[K, :]) \; \forall c_i \in k, k \in K, \forall y \in T_X, X \in \{A, B\} \tag{3.2}$$

para todo cliente c_i con serie de tiempo $x_t^{c_i}$ y perfil de consumo k, en el tiempo t, la probabilidad de que el consumo observado sea h está condicionada a la clase T_X a la cual pertenezca la temperatura y_t en el tiempo t y al histórico de consumo de energía $\mathbf{H}[K, :]$ de los clientes del perfil K. Para cada perfil en un momento dado habrán tantas distribuciones de probabilidad como clases de temperaturas. La probabilidad aproximada con los datos de la muestra $\hat{p}_{k,t}(x_t^{c_i} = h | T_X, \mathbf{H}[K, :])$

se puede calcular usando todas las observaciones en el perfil k, o solo las observaciones de los clientes que tengan el histórico completo.
Se pretende explotar las estructuras de *perfiles* y *clases de temperaturas* empleándolas para predecir el consumo de energía de cada cliente.
La Tabla 3.1 representa con colores un bosquejo del problema.
En la Tabla 3.2 se muestra un ejemplo de como relacionar los datos dados por las estructuras de *perfiles* y *clases de temperaturas*. Los datos extraídos pueden ser utilizados para aprender las probabilidades de predicción Ec. 3.2 o como datos de entrada para el entrenamiento de alguna técnica de predicción.

F=2	1	2	1	2	1	2

T=5	y_1	y_2	y_3	y_4	$\hat{y}_5$	$\hat{y}_6$

N=5 \ H=4	1	2	3	4	5	6
1	x_1^1	x_2^1	x_3^1	x_4^1	$\hat{x}_5^1$	$\hat{x}_6^1$
2	x_1^2	x_2^2	x_3^2	x_4^2	$\hat{x}_5^2$	$\hat{x}_6^2$
3	x_1^3	x_2^3	x_3^3	x_4^3	$\hat{x}_5^3$	$\hat{x}_6^3$
4	x_1^4	x_2^4	x_3^4	x_4^4	$\hat{x}_5^4$	$\hat{x}_6^4$
5	x_1^5	x_2^5	x_{53}^5	x_4^5	$\hat{x}_5^5$	$\hat{x}_6^5$
6				x_4^6	$\hat{x}_5^6$	$\hat{x}_6^6$
7			x_3^7	x_4^7	$\hat{x}_5^7$	$\hat{x}_6^7$

Tabla 3.1: En la tabla de colores se observa que hay 7 clientes, los primeros 5 con 4 valores observados (columnas 1 al 4) y los últimos dos con 1 y 2 valores observados respectivamente. La frecuencia F es de 2 mediciones por día, y con la información del día 1 (col. 1 y 2) y del día 2 (col. 3 y 4), se quieren predecir los valores para el siguiente día (col. 5 y 6). Cada cliente pertenece a uno de dos perfiles de consumo, los clientes 1,2,4 y 7 pertenecen al perfil de consumo verde y los clientes 3,5 y 6 al perfil de consumo rojo. A su vez hay 4 observaciones para las temperaturas hechas durante los tiempos en que se observaron las mediciones, cada temperatura pertenece a una clase: temperatura alta (color magenta) y temperatura baja (color cyan), las temperaturas del siguiente día predícense como: temperatura alta ($\hat{y}_5$) y temperatura baja ($\hat{y}_6$).

F=2	1	2	1

T=5	y_1	y_4	$\hat{y}_5$

H=4 / N=5	1	4	5
1	x_1^1	x_4^1	$\hat{x}_5^1$
2	x_1^2	x_4^2	$\hat{x}_5^2$
4	x_1^4	x_4^4	$\hat{x}_5^4$
7		x_4^7	$\hat{x}_5^7$

3	x_1^3	x_4^3	$\hat{x}_5^3$
5	x_1^5	x_4^5	$\hat{x}_5^5$
6		x_4^6	$\hat{x}_5^6$

Tabla 3.2: Para el caso del tiempo 5, cuya temperatura $\hat{y}_5$ se predijo como alta, los valores de consumo para cada perfil se predicen utilizando como datos históricos aquellos que coincidan con la clase de temperatura de $\hat{y}_5$. Otra posibilidad es predecir los valores de consumo en el tiempo 5 considerando como datos históricos aquellos que coincidan con la clase de temperatura de $\hat{y}_5$ (factor temperatura) y además que coincidan con el tiempo del día que se va a predecir (factor estacional), observando el arreglo de frecuencias F, el tiempo de predicción es el 1, con lo que solo serviría como información la primera columna de información histórica, descartando el consumo observado de la segunda columna.

3.2. Objetivo

Sean D los días a predecir el consumo de energía para cada usuario, sea F la frecuencia con que se predice en el día, ejemplo si las predicciones se realizan cada media hora $F = 48$, si se realizan cada 3 horas entonces $F = 8$. Sea $s \in F * D$ un tiempo dado de la predicción. Sea $RMS\,E_s$ (Ec 2.6) el error medio cuadrático dado por los valores reales de energía consumidos por los clientes en el tiempo s y los valores predichos por el modelo.
El objetivo es minimizar el error medio cuadrático promedio de los $F \cdot D$ tiempos de predicción:

$$min \frac{1}{F \cdot D} \sum_{s=1}^{F \cdot D} RMS\,E_s \tag{3.3}$$

La Ecuación 3.3 equivale a mejorar el error para cada usuario en particular, en contraste con mejorar el promedio.

3.3. El modelo de las monedas

El modelo de las monedas está enfocado en definir un experimento teórico y analizar en él que tan relevante resulta el conocer o no los perfiles de las monedas, y la información de la temperatura como factor que afecta al experimento. *El modelo de las monedas* consta de un experimento sencillo: el arrojar varias monedas cada día y registrar para cada una si sale cara o cruz. En primera instancia se define un experimento en el cual el comportamiento de las monedas depende del perfil al que pertenecen. Cada perfil presenta un único comportamiento que no cambia durante el experimento. En el segundo experimento el comportamiento de las monedas sigue dependiendo del perfil al que pertenecen. Pero cada perfil cambia su comportamiento dependiendo de la temperatura del ambiente al momento de realizar los lanzamientos. Para cada caso se exponen resultados y conclusiones.

3.3.1. Modelo sin incluir temperatura

El experimento consiste en arrojar durante D días N monedas, y registrar para cada una de ellas si el resultado de la tirada es cara o cruz. La matriz $M_{N\times D}$ almacenará en la i-ésima fila los D resultados de las tiradas para la moneda i, asumiendo un valor de 1 si el resultado es cara y 0 si es cruz. Existen dos perfiles de monedas K_1 y K_2. Cada perfil con una probabilidad asociada de salir cara $p_{K_1}(cara)$ y $p_{K_2}(cara)$ respectivamente. Cada moneda pertenece a uno de estos perfiles y la probabilidad de que su resultado luego del lanzamiento sea cara es igual al de la probabilidad asociada a su perfil.
Los D días del experimento se dividen en D_e días de entrenamiento y D_p días de prueba, cumpliendo la igualdad $D = D_e + D_p$.
No todas las monedas comienzan el experimento en el día 1, algunas pueden comenzar en un día d muy próximo al día final del entrenamiento D_e, lo que conlleva a una menor cantidad de información individual en comparación a las que comenzaron a registrar los resultados desde el primer día. La matriz $M_{N\times D}$ guardará el símbolo $indefinido$, para las entradas que no tengan observaciones. Durante el experimento, día a día, se irán completando las D_e columnas de la matriz M. Guardando cada día $d = 1, \cdots, D_e$ el resultado de la tirada de la moneda i en la entrada $M[i, d]$, o el valor $indefinido$, en caso de que la tirada de la moneda

no sea observada. Con la información registrada de los D_e días de entrenamiento, se busca completar o predecir cuales serán los resultados del experimento para cada moneda en los siguientes D_p días de prueba o columnas de la matriz M.

3.3.1.1. Métricas

Las métricas empleadas para evaluar las diferentes técnicas aplicadas al modelo son: RMSE Ec 2.6 y F_1-Score Ec 2.10 sobre el total de las monedas.
$F1$-Score es útil para evaluar, en la predicción de un experimento, la fracción de monedas correctamente etiquetadas como cara sobre el total real de monedas con etiquetas cara (precision) y la fracción de monedas con etiqueta real cara de todas las que fueron predichas con etiqueta cara (recall), lo que permite evaluar que tan buena es la técnica en acertar la salida de cara para la tirada de cada moneda.

3.3.1.2. Objetivo

Sean $F1_{D_e+i}, RMSE_{D_e+i}, 1 \leq i \leq D_p$, los valores de las métricas F1-Score y RMSE para el i-ésimo día de prueba. Los valores promedio de estas métricas durante los D_p días de prueba se definen así:

$$F1_{Prom} = \frac{1}{D_p} \sum_{i=1}^{D_p} F1_{D_e+i} \tag{3.4}$$

$$RMSE_{Prom} = \frac{1}{D_p} \sum_{i=1}^{D_p} RMSE_{D_e+i} \tag{3.5}$$

El objetivo para este modelo es maximizar $F1_{Prom}$ y minimizar $RMSE_{Prom}$.

3.3.1.3. Técnicas

Se proponen 5 técnicas que empleando toda la información del modelo, o parte de esta, resuelven el problema de predicción. A continuación se define un método de aprendizaje y predicción. Todas, salvo una de las técnicas definidas, utilizan este método. La diferencia entre las técnicas que usan el método viene dada por los distintos parámetros de entrada que cada una ingresa al algoritmo del método. Luego de esto se definen las 5 técnicas a comparar.

Método de aprendizaje y predicción: Sea N la cantidad de monedas del experimento, de estas: N_c de ellas comienzan a registrar las observaciones del experimento a partir del primer día (histórico de tiradas completo), mientras que N_{nc} de ellas comienzan las observaciones en días próximos a D_e (histórico de tiradas no completo), el comienzo de las observaciones no tiene que ser el mismo día para cada una de estas N_{nc} monedas. Se cumple que $N = N_c + N_{nc}$. Sea $\mathcal{K}$ un conjunto de perfiles de monedas, sea C un arreglo de etiquetas de tamaño N_c, con $C[i] = k$ indicando que la i-ésima moneda, con histórico completo, pertenece al perfil k. Notar en este punto que el pedir como parámetros de entrada $\mathcal{K}$ y C, no implica que el Algoritmo 1 del método conozca la etiquetas reales de las monedas con histórico completo, sino que esto significa que la técnica que implementa el algoritmo debe proporcionarle las etiquetas al mismo, sean las correctas o no. En principio se desconoce el perfil de las N_{nc} monedas con histórico incompleto y es tarea del algoritmo asignarles un perfil de los existentes en $\mathcal{K}$. El algoritmo toma como entrada la matriz de tiradas históricas $M_{N\times D_e}$, $\mathcal{K}$, C y los D_p días a predecir. Sin pérdida de generalidad se asume que las primeras N_c entradas de la matriz son ocupadas por los clientes con histórico completo.

Algoritmo 1 $MONEDAS_1(M_{N\times D_e}, \mathcal{K}, C, D_p)$

1: Separar la matriz M en M^c, matriz con las monedas que comienzan las observaciones a partir del primer día y M^{nc}, matriz con las monedas que comienzan las observaciones en días próximos al final del entrenamiento.
2: $iter \leftarrow D_e + 1$
3: **while** $iter - D_e \leq D_p$ **do**
4: // *Aprendizaje*
5: Asignar perfiles a las monedas con histórico incompleto
$\hat{C} \leftarrow asignarPerfiles(M^c, M^{nc}, \mathcal{K}, C)$
6: **for** $k \in \mathcal{K}$ **do**
7: Obtener M^c_k, la submatriz con las filas únicamente de las monedas $i \in N_c$ con perfil k. Sea además $|C_k|$ la cantidad de monedas con perfil k.
8: **for** $d \in \{1, \ldots, iter - 1\}$ **do**
9: Calcular la razón entre la cantidad de caras en la d-ésima columna en M^c_k: $M^c_k[:, d]$ y $|C_k|$, sea $\hat{p}_d$ el resultado de este cálculo, que se puede ver como la probabilidad aproximada de salir cara en el día d para las monedas con perfil k.
10: **end for**
11: La probabilidad aproximada de salir cara para las monedas en k se aproxima como el promedio de los $\hat{p}_d$. Entonces $\hat{p}_k(cara) = \frac{1}{iter-1} \sum_{d=1}^{iter-1} \hat{p}_d$.

12: **end for**
13: // *Predicción*
14: **for** $i \in N$ **do**
15: **if** i es moneda con histórico completo **then**
16: $k \leftarrow C[i]$
17: **else if** i es moneda con histórico incompleto **then**
18: $k \leftarrow \hat{C}[i - N_c]$
19: **end if**
20: El resultado de la tirada para el cliente i para el día *iter* se predice con $\hat{p}_k(cara)$.
21: **end for**
22: A la matriz M^c se le agrega una nueva columna, con las predicciones de las N_c monedas y a la matriz M^{nc} se le agrega una nueva columna, con las predicciones de las N_{nc} monedas.
23: $iter = iter + 1$
24: **end while**

Observar en Alg. 1, que para cada uno de los D_p días se repiten los bloques de *Aprendizaje* y *Predicción*.
En el bloque de *Aprendizaje*, paso 5 se resuelve, cada día, que etiqueta le corresponde a cada moneda con histórico incompleto. Nótese como las etiquetas de estas monedas pueden variar, acomodándose a diferentes perfiles según pasen los días, a diferencia de las etiquetas fijas de las N_c monedas con histórico completo.
A su vez en los pasos del 6 al 12, en los que se aprenden las probabilidades $\hat{p}_k(cara)$ de cada perfil k, notar que la única información histórica que se emplea es la de las monedas con histórico completo (ver paso 7).
Ya en el bloque de *Predicción*, en los pasos del 14 al 21 se predice el resultado del lanzamiento para cada moneda, con probabilidad de salir cara igual a la probabilidad aprendida para el perfil al cual la moneda pertenece.
Al final del bucle de predicción, paso 22, se agrega a la matriz M^c y M^{nc} una nueva columna con los resultados de la predicción hechas paras las monedas con histórico completo e histórico incompleto, respectivamente. Notar que a las matrices M^c y M^{nc}, se les agrega en cada paso una columna de predicción, siendo estas las matrices de las cuales se obtendrán los datos de entrenamiento para todos los perfiles en el siguiente paso (en caso de no ser el último), por lo que para predecir los días a futuro se usan: el histórico real más las predicciones que se hayan realizado hasta el momento.
Llegada la ejecución al final del *while*, paso 23, se vuelven a repetir los mismos

pasos para el siguiente día de predicción.
El algoritmo utilizado para asginar perfiles a las monedas con histórico incompleto es el siguiente, sin pérdida de generalidad asúmase que las N_c monedas con histórico completo ocupan las primeras N_c filas de la matriz $M_{N\times D_e}$:

Algoritmo 2 $asignarPerfil(M_{N\times D_e}, \mathcal{K}, C)$

1: **for** k en $\mathcal{K}$ **do**
2: Obtener el índice I en C de las monedas que pertenecen a k.
3: Obtener la submatriz $M_k^c = M^c[I, :]$, dada por todas las filas que pertenecen a las monedas con perfil k.
4: Para M_k^c calcular la media de caras salidas en todas las tiradas observadas μ_k. Este valor se guarda en el arreglo $\mathcal{U}$, de tamaño $|\mathcal{K}|$ con $\mathcal{U}[k] = \mu_k$.
5: **end for**
6: Obtener M^{nc} submatriz con las monedas con histórico incompleto (las últimas N_{nc} filas de $M_{N\times D_e}$).
7: **for** i fila en M^{nc} **do**
8: Calcular la media de caras salidas en las tiradas observadas en la fila i, μ_i, guardándose cada valor en $\mathcal{U}^{nc}$, análogamente al arreglo anterior $\mathcal{U}^{nc}[i] = \mu_i$.
9: **end for**
10: **for** i moneda en N_{nc} **do**
11: Determinar el perfil $k \in \mathcal{K}$ al que pertenece mediante $k_i = \mathcal{K}[k], k = argmin_{k\in\mathcal{K}}\{|\mathcal{U}^{nc}[i] - \mathcal{U}[k]|\}$. Guardar en $\hat{C}$ el perfil resultante, significando $\hat{C}[i] = k_i$.
12: **end for**
13: **return** $\hat{C}$

Observar en el paso 11 en el cual se asigna una etiqueta a cada moneda i con histórico incompleto, que el criterio de la elección corresponde a elegir la etiqueta del perfil k cuya media de caras $\mathcal{U}[k]$ esté más proxima a la media de caras, $\mathcal{U}^{nc}[i]$ de la i-ésima moneda con histórico incompleto. Las medias $\mathcal{U}[k]$ se calculan en los paso del 1 al 5 y las medias de las monedas con histórico incompleto $\mathcal{U}^{nc}[i]$ en los pasos del 7 al 9.

A continuación se describen las técnicas predictivas utilizadas en este trabajo, sin pérdida de generalidad se asume que las primeras N_c filas corresponden a las monedas con observaciones en todos los días de entrenamiento y las $N - N_c$ restantes a las monedas con observaciones a partir de un día próximo al final del entrenamiento.

Técnica Conociendo Perfiles (TCP):
TCP conoce a que perfil $k \in \{K_1, K_2\}$ pertenece cada moneda $i \in N_c$, por lo cual C contiene exactamente a que perfil pertenece cada moneda con histórico completo. Esto no es realista para el problema general, pero busca evaluar la predicción cuando no se introduce error al clasificar en perfiles.
El *algoritmo general de aprendizaje* Alg. 1 toma como entrada $M_{N\times D_e}$, $\mathcal{K} = \{K_1, K_2\}$, C y D_p días a predecir.
Invocación al Algoritmo 1:

$$MONEDAS_1(M_{N\times D_e}, \{K_1, K_2\}, C, D_p)$$

Técnica K-Means (TKM):
TKM no conoce exactamente los perfiles de cada moneda, pero conoce que son 2, por lo que habrán dos perfiles $\{KM_1, KM_2\}$. El arreglo de etiquetas C almacenará las etiquetas devueltas luego de ejecutar el algoritmo $Kmeans(M_{N_c\times D_e}, k = 2)$, el algoritmo K-means toma como rasgos para cada moneda sus tiradas y como cardinalidad $k = 2$.
El *algoritmo general de aprendizaje* Alg. 1 toma como entrada $M_{N\times D_e}$, $\mathcal{K} = \{KM_1, KM_2\}$, $C = Kmeans(M_{N_c\times D_e}, k = 2)$ y D_p días a predecir.
Invocación al Algoritmo 1:

$$MONEDAS_1(M_{N\times D_e}, \{KM_1, KM_2\}, Kmeans(M_{N_c\times D_e}, k = 2), D_p)$$

Técnica Desconociendo Perfiles (TDP):
TDP desconoce la existencia de los perfiles de monedas, presupone que todas las monedas se comportan de manera similiar, por lo que se pueden considerar todas dentro de un mismo perfil. Cabe aclarar que esto no implica que no haya perfiles, simplemente que esta técnica desconoce su existencia y por lo tanto no los ve. Llámese el único perfil presupuesto K, en este caso $C[i] = K, \forall i \in N_c$. El *algoritmo general de aprendizaje* Alg. 1 toma como entrada $M_{N\times D_e}$, $\mathcal{K} = \{K\}$, C y D_p días a predecir.
Invocación al Algoritmo 1:

$$MONEDAS_1(M_{N\times D_e}, \{K\}, C, D_p)$$

Técnica Individualizando Perfiles (TIP):
TIP utiliza como información para predecir la tirada de cada moneda solo su perfil individual, por lo que habrán N_c perfiles diferentes. En este caso $\mathcal{K} = \{1, \ldots, N_c\}$ y $C = [1, \ldots, N_c]$.

El *algoritmo general de aprendizaje* Alg. 1 toma como entrada $M_{N\times D_e}$, $\mathcal{K} = \{1,\ldots,N_c\}$, $C = [1,\ldots,N_c]$ y D_p días a predecir.
Invocación al Algoritmo 1:

$$MONEDAS_1(M_{N\times D_e}, \{1,\ldots,N_c\}, [1,\ldots,N_c], D_p)$$

Técnica Completado de Matrices (TCM):
Esta técnica es la única que no emplea el *algoritmo general de aprendizaje* Alg. 1, el Algoritmo 3 toma como entrada una matriz incompleta $M_{N\times D_e}$,un rango k y D_p días a predecir, devolviendo una matriz completada MC.

Algoritmo 3 $TCM_1(M_{N\times D_e}, k, D_p)$

1: Separar la matriz M en M^c, matriz con las monedas que comienzan las observaciones a partir del primer día y M^{nc}, matriz con las monedas que comienzan las observaciones en días próximos al final del entrenamiento.
2: Clasificar las N_c filas con histórico completo de M^c en dos grupos $\{K_1, K_2\}$ utilizando K-means. Guardar en C los perfiles resultantes de la clasificación.
3: $iter \leftarrow D_e + 1$
4: **while** $iter - D_e \leq D_p$ **do**
5: $\hat{C} \leftarrow asignarPerfil(M^c, M^{nc}, \mathcal{K}, C)$
6: // Crear submatrices para aplicar ICMC
7: **for** $i \in [1,2]$ **do**
8: Crear la matriz del perfil K_i, a la cual aplicar el algoritmo ICMC, siendo M^{K_i} la matriz que contiene todas las filas j_c de M^c tales que $C[j_c] = K_i$ y todas las filas j_{nc} de M^{nc} tales que $\hat{C}[j_{nc}] = K_i$. Guardar en el arreglo I_i^c (resp. I_i^{nc}) la información de a que fila de M^c (resp. M^{nc}) corresponde cada fila de M^{K_i}.
9: **end for**
10: // Aplicar ICMC a las submatrices
11: **for** $M^{K_i} \in [M^{K_1}, M^{K_2}]$ **do**
12: Agregar una columna sin observaciones al final de M^{K_i} para completar con ICMC el siguiente día a predecir.
13: Elegir al azar el 1 % de las filas en M^{K_i} y predecir su resultado para el siguiente día, con probabilidad de salir cara igual a la razón entre la cantidad de caras observadas para dicha moneda y la cantidad de días de entrenamiento $iter - 1$.
14: Aplicar el algoritmo $ICMC(M^{K_i}, rango = k)$
15: **end for**
16: // Agregar las predicciones a la nueva columna en M

17: Crear una nueva columna en M^c y M^{nc}. Para $i = 1, 2$, agregar los valores completados en las matrices M^{K_i}, colocando el último valor de cada columna en la fila correspondiente a la nueva columna de M^c o M^{nc}, de acuerdo a los índices guardados en I_i^c, I_i^{nc}.
18: Agregar a M las nuevas columnas predichas durante la iteración agregadas a M^c y M^{nc}.
19: $iter = iter + 1$
20: **end while**
21: **return** M

El Algoritmo 3 en el paso 1, al igual que el Algoritmo 1, separa las monedas en una matriz para aquellas con histórico completo M^c y otra para las que tienen histórico incompleto M^{nc}.
En los pasos 2 y 5 se observa también un proceso de etiquetado de las monedas. Las monedas con histórico completo son etiquetadas solo una vez al comienzo del algoritmo y las monedas con histórico incompleto son reetiquetadas cada día.
En los pasos del 7 al 9 para cada perfil K_i se obtiene la matriz M^{K_i}, conteniendo en cada fila los valores observados de todas las monedas etiquetadas a dicho perfil.
En los pasos del 11 al 15 luego de agregarle a cada matriz M^{K_i} una nueva columna, al final, sin observaciones, representando la columna para el siguiente día de predicción, se aplica el algoritmo de completado de matrices ICMC. Observar que en el paso 13 se exige completar una fracción pequeña de la columna sin observaciones recientemente agregada a la matriz, esto es necesario ya que el algoritmo ICMC necesita tener al menos una observación en toda fila o toda columna, como condición necesaria para completar con éxito las entradas no observadas de la matriz.
En el paso 17 se agregan nuevas columnas a M^c y M^{nc} con las predicciones (últimas columnas completadas por $ICMC(M^{K_i}, rango = k), i = 1, 2$).
En el paso 18 se agrega una nueva columna a M con las predicciones (últimas columnas completadas de M^c y M^{nc}).

3.3.1.4. Comportamiento esperado

En esta sección se analizarán algunas características de las diferentes técnicas, con el objetivo de encontrar juegos de parámetros para los cuales las técnicas de clustering, TCP y TKM, que utilizan perfiles performen mejor que el resto.
Una característica de las técnicas TCP y TKM es que hacen hincapié en la clusterización para aprender las probabilidades $p_k(cara)$ y predecir. TCP: conociendo exactamente los perfiles de cada usuario, TKM: conociendo cuantos perfiles K hay

y aplicando una técnica de clustering para clasificar a las monedas en K perfiles. En cambio las técnicas TDP y TIP son los dos casos extremos de clusterización en donde la primera asume un solo grupo $K = 1$ y la segunda iguala la cantidad de grupos con la cantidad de monedas $K = N$. En este trabajo se aplicarán técnicas de clustering para la predicción, por lo que el interés está en saber para que juegos de parámetros del modelo las técnicas TCP y TKM son mejores respecto a TDP y TIP y viceversa. Esto permitirá conocer en que situaciones es más favorable tomarse el trabajo de clasificar y para que situaciones no, por ejemplo es de esperar que a medida que los perfiles de monedas se parezcan más entre sí sea menos eficaz su utilización en la predicción.
Algunas consideraciones:

- La probabilidad de cada moneda sigue una distribución Bernoulli $\mathcal{B}er(p)$, con p igual a la probabilidad de éxito. El éxito se define como: salir cara en una tirada.

- La técnica TKM emplea como *features* para clasificar a cada moneda: las D tiradas observadas en el experimento.

Clasificación usando K-means

La técnica TKM conoce la cantidad de perfiles de las monedas, pero no conoce a que perfil pertenece cada una. TKM emplea la técnica de clustering K-means para la tarea de clasificación. En primera instancia, observar que las técnicas TCP y TKM no son lo mismo. TKM tiene el trabajo extra de inferir los perfiles de los usuarios. En la Figura 3.1 se visualizan las regiones de probabilidades más y menos favorables para la tarea de clasificación de TKM usando K-means, véase la Sección A.1.1.1 del Apéndice A para la argumentación de cada región.

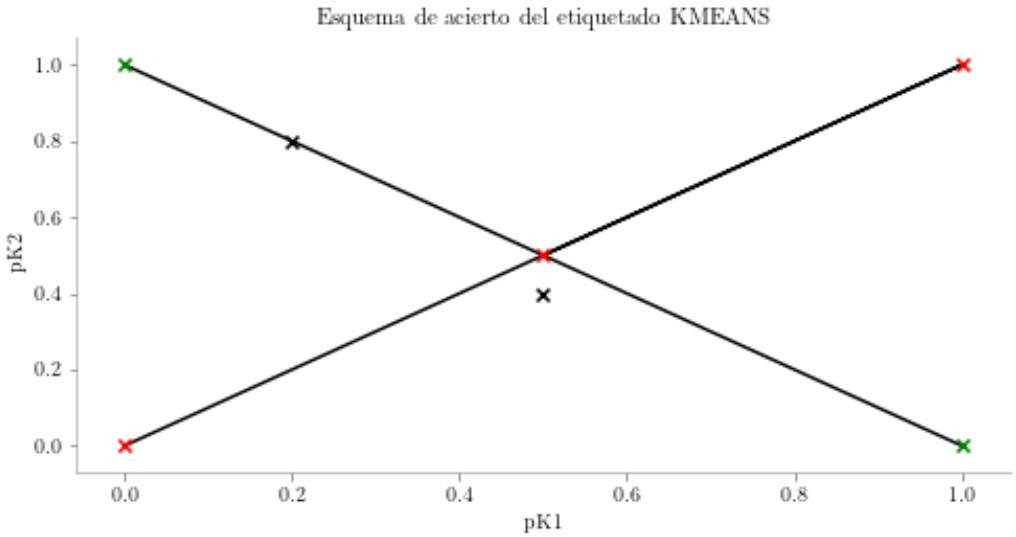


Figura 3.1: El eje x representa la variación de la probabilidad de salir cara en cada tirada para el perfil K_1, el eje y representa análogamente esto para el perfil K_2. Las observaciones hechas se marcan con una cruz. Las cruces de color verde representan los lugares en donde se observa una clasificación perfecta y las rojas los lugares en donde la clasificación es más dificultosa. El comportamiento esperado, usando como función de distancia en el algoritmo K-means la distancia euclídea es el siguiente: para los puntos cerca de la recta con pendiente 1, en la figura, es el de una mala clasificación, dado el parecido entre las probabilidades. Mientras que a medida que los puntos se alejan de esta recta, es decir a medida que las probabilidades se hacen más diferentes entre sí, se ha de esperar un mejor comportamiento en la clasificación. Por ejemplo el punto $(0.2, 0.8)$ más cercano a la recta con pendiente -1 que a la recta con pendiente 1, se espera tenga una mejor clasificación que el punto $(0.5, 0.4)$, el cual guarda mayor cercanía a la recta con pendiente 1.

Del primer punto de las consideraciones se infiere que los lanzamientos de una moneda, perteneciente al perfil K, con probabilidad $p_K(cara)$, durante D días, se puede ver como una distribución binomial $\mathcal{B}in(D, p_K(cara))$.
En la Figura 3.2 se visualiza el rango de valores que contienen el 95 % de la distribución para cada probabilidad p de una distribución binomial de tamaño $D = 100$. Allí se evidencia que dos monedas de perfiles con probabilidades muy distintas tendrán un comportamiento marcadamente diferente durante los D días del experimento, porque las diferencias en la cantidad de resultados cara es muy notoria para monedas con probabilidades alejadas. Mientras que si los perfiles tienen probabilidades similares el comportamiento de los resultados de las tiradas

a lo largo del experimento será más parecido, pues los intervalos se solapan.

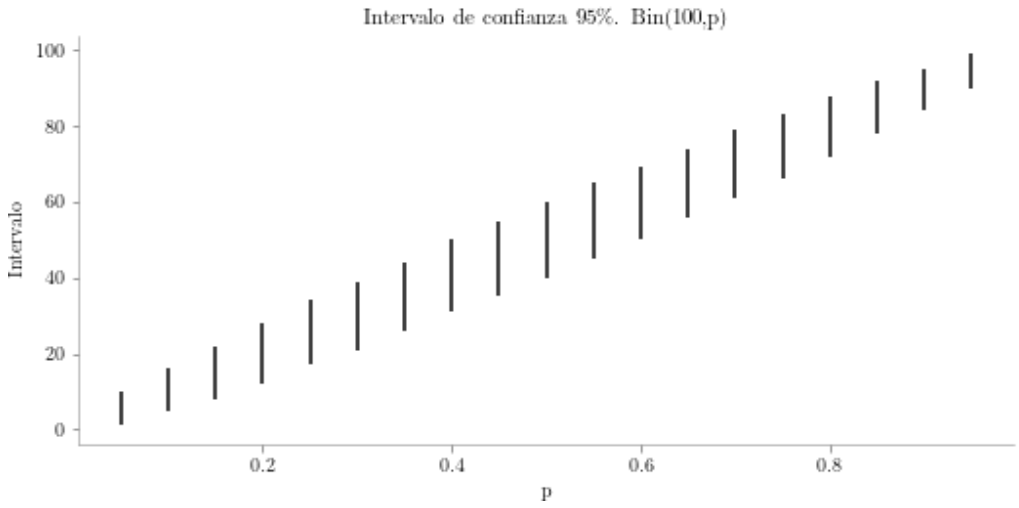


Figura 3.2: Rango de valores que contiene el 95 % de la distribución para cada probabilidad p de una distribución binomial de tamaño $D = 100$.

Comportamiento predictivo de técnicas TDP y TCP

Se quiere comparar la técnica TCP, que conoce exactamente los perfiles, versus la técnica TDP la cual asume que todas las monedas se comportan exactamente igual y asume un solo perfil para todas. La Ecuación 3.6, representa el valor esperado de la probabilidad aprendida por TDP para su único cluster, ver Sección A.1.1.2 del Apéndice A para la obtención de $\hat{p}_{TDP}$.

$$\hat{p}_{TDP} \approx p_{K_1}(cara) + k \times (p_{K_2}(cara) - p_{K_1}(cara)), \quad 0 \leq k \leq 1 \tag{3.6}$$

con $k = \frac{N_2}{N}$ igual a la fracción de monedas del perfil K_2, siendo N el total de monedas y N_2 la cantidad de monedas del perfil K_2.

La Ecuación 3.6 permite analizar el comportamiento de la técnica TDP para la predicción del resultado de las monedas.

A medida que k se aproxima a 0, en el experimento se tienen cada vez menos monedas del perfil K_2 (pues $k = \frac{N_2}{N}$) y la Ecuación 3.6 queda:

$$\hat{p}_{TDP} \approx p_{K_1}(cara) + k \times (p_{K_2}(cara) - p_{K_1}(cara)) \underset{k \to 0}{\approx} p_{K_1}(cara)$$

con lo que la probabilidad aprendida esperada para TDP se acercará a $p_{K_1}(cara)$. Con la consiguiente mala predicción para las pocas monedas que pertenecen a K_2

y bastante buena para las monedas en K_1.
Si k se aproxima a 1, en el experimento se tienen cada vez más monedas del perfil K_2 y la Ecuación 3.6 queda:

$$\hat{p}_{TDP} \approx p_{K_1}(cara) + k \times p_{K_2}(cara) - k \times p_{K_1}(cara) \underset{k \to 1}{\approx} p_{K_2}(cara)$$

con lo que la probabilidad aprendida esperada para TDP se acercará a $p_{K_2}(cara)$, y la mala predicción ahora será para las pocas monedas que haya en K_1.
En el caso de que los perfiles tengan la misma cantidad de monedas, se tiene que $k = \frac{1}{2}$ y la Ecuación 3.6 queda:

$$\hat{p}_{TDP} \approx p_{K_1}(cara) + \frac{1}{2} \times (p_{K_2}(cara) - p_{K_1}(cara)) = \frac{1}{2} \times (p_{K_2}(cara) + p_{K_1}(cara))$$

la probabilidad aprendida aproxima al punto medio entre $p_{K_1}(cara)$ y $p_{K_2}(cara)$. Cuando hay igual cantidad de monedas en cada perfil la probabilidad aprendida por TDP aproxima al punto medio de las probabilidades de ambos perfiles. En esta situación se espera que el valor de la métrica RMSE aumente y de F1 disminuya a medida que la diferencia entre las probabilidades de los perfiles aumente, por la simple razón de que el perfil con la probabilidad más chica irá aumentando la cantidad de falsos positivos y el perfil con las probabilidad más grande irá aumentando la cantidad de falsos negativos.
El modelo TCP en cambio, independientemente de la cantidad de monedas de uno u otro perfil, aprenderá dos probabilidades cercanas a las reales:

$$\hat{p}_{K_1}(cara) \approx p_{K_1}(cara)$$

y

$$\hat{p}_{K_2}(cara) \approx p_{K_2}(cara)$$

evitando los errores que tiene TDP, cuando las proporciones de monedas en los perfiles son iguales y las probabilidades son muy distintas. Se espera una mejor performance en RMSE y F1 para TCP respecto de TDP en estos casos.

Benchmark para históricos incompletos

Las técnicas se comparan con el benchmark de predecir el resultado de cada moneda con observaciones incompletas, solo con los datos de estas. La probabilidad de salir cara es la suma total de caras sobre la cantidad de tiradas observadas para cada moneda, llámese el benchmark: técnica INCompleta (INC).
En primera instancia si el histórico que INC tiene para aprender es de tamaño 30,

entonces tendrá bastante información para aproximar bien la probabilidad real del perfil, dado que el comportamiento de la moneda en el perfil es siempre el mismo (misma probabilidad siempre), lo mismo se puede decir para el caso en donde el histórico es de tamaño 10. Mientras que para el caso en que el histórico es de tamaño 3, INC, igualmente puede aprender con bastante exactitud la probabilidad del perfil, véase la Tabla 3.3 para casos de históricos favorables para INC con históricos de tamaño 3.

$p(cara)$	Histórico	$\hat{p}(cara)$
0.9	[1, 1, 1]	1
0.1	[0, 0, 0]	0
0.3	[0, 1, 0]	0.3 . . .
0.6	[1, 1, 0]	0.6 . . .

Tabla 3.3: La columna $p(cara)$ contiene la probabilidad real del perfil. **Histórico** indica cual es el histórico observado para la moneda. $\hat{p}(cara)$ contiene la probabilidad aprendida por INC.

Resumen de resultados esperados

- Modelo TKM, está técnica emplea el algoritmo K-means para clustering, con la distancia Euclídea utilizada en los pasos de *reasignación* de features a clusters y *recomputación* de centroides en el Algoritmo 10. Mientras más parecidas sean las probabilidades $p_{K_1}(cara)$ y $p_{K_2}(cara)$, es más probable que TKM aumente la cantidad de monedas clasificadas en el perfil equivocado (ej. clasificar una moneda como del perfil K_1, perteneciendo en realidad al perfil K_2).

- Observando los intervalos de confianza, con $\alpha = 0.95\,\%$, para un experimento de lanzamiento de una moneda durante D días (ver Figura 3.2) se evidencia que dos perfiles con probabilidades muy distintas, tendrán un comportamiento marcadamente distinto.

- Modelo TDP, si la cantidad de monedas en ambos perfiles K_1 y K_2 son iguales, TDP tendrá una tasa igual de errores sin favorecer la predicción de ningún perfil. Bajo estas condiciones: a medida que la diferencia entre las probabilidades de los perfiles aumenta, se espera un aumento del error en RMSE y F1. Se espera que TCP performe mejor que TDP en estos casos.

- Combinando los dos puntos anteriores se tiene que TKM y TCP deben performar marcadamente mejor que TDP cuando las probabilidades de los perfiles sean muy distintas y la cantidad de monedas en cada perfil iguales.
- El benchmark INC puede performar bien aún con poca información histórica.

3.3.1.5. Resultados

Sean los siguientes parámetros del problema:

- N, cantidad de monedas.
- fNK_1, fracción de monedas con perfil K_1.
- N_{nc}, cantidad de monedas con histórico incompleto.
- D, cantidad de días (entrenamiento más pruebas).
- D_p, cantidad de días tomados para predicción.
- DI, cantidad de días observados para las monedas con histórico incompleto.
- p_{K_1}, p_{K_2}, probabilidad de salir cara para los perfiles K_1, K_2 respectivamente.
- k, rango de la matriz a completar, parámetro requerido por la técnica TCM.

Se realizaron tres pruebas, la primera intenta verificar para que pares de probabilidades, de los perfiles, K-means clasifica mejor, la segunda apunta a verificar el resto de las conclusiones de la sección anterior para monedas con histórico completo y la tercera a analizar el comportamiento de las técnicas para la predicción de monedas con histórico incompleto.

Prueba ST1: Verificar clasificación K-means

Esta prueba evalúa la capacidad de K-means para clasificar correctamente a las monedas en los dos perfiles K_1 y K_2. Las *features* son los lanzamientos históricos observados para cada moneda y la cardinalidad k es igual a 2. Los parámetros elegidos para evaluar son los dados en la Tabla A.1 del Apéndice A. Para cada juego de parámetros se generan 10 matrices que cumplan con las restricciones dadas por dichos parámetros. Sobre estas diez matrices se calcula el promedio de

monedas que K-means clasificó correctamente, sobre el total de monedas. Los resultados se ven en la Tabla 3.4, los cuales corroboran que K-means clasifica mejor mientras más diferencia haya entre las probabilidades de cada perfil.

ID	N	D	fNK_1	N_{nc}	DP	p_{K_1}	p_{K_2}	Fracc.
1	100	100	0.1	0	10	0.1	0.9	1.0
2	100	100	0.5	0	10	0.1	0.9	1.0
3	100	100	0.9	0	10	0.1	0.9	1.0
4	100	100	0.5	0	10	0.5	0.5	0.497
5	100	100	0.3	0	10	0.3	0.6	0.999
6	100	100	0.6	0	10	0.3	0.6	0.993

Tabla 3.4: Fracc. representa la fracción promedio de aciertos, definida como el promedio sobre las 10 simulaciones de la razón entre las monedas clasificadas correctamente por K-means y la cantidad de monedas del experimento.

Prueba ST2: Verificar resultados esperados

Los valores de los parámetros para esta prueba se detallan en la Tabla 3.5. Para cada juego de parámetros se genera una matriz $M_{N\times D}$ que cumpla con las restricciones impuestas por los parámetros. Las predicciones de los D_p días de prueba se repiten 10 veces para todas las técnicas. Obteniéndose un RMSE y F1 promedio de las 10 simulaciones. La elección de los rangos k se detalla en la Sección A.1.2.1 del Apéndice A.

ID	**N**	**D**	fNK_1	N_{nc}	D_p	p_{K_1}	p_{K_2}	k
1	{10, 100, 200}	{50, 100, 200}	0.1	0	{5, 10, 15}	0.1	0.9	5
2	{10, 100, 200}	{50, 100, 200}	0.5	0	{5, 10, 15}	0.1	0.9	5
3	{10, 100, 200}	{50, 100, 200}	0.9	0	{5, 10, 15}	0.1	0.9	3
4	{10, 100, 200}	{50, 100, 200}	0.5	0	{5, 10, 15}	0.5	0.5	3
5	{10, 100, 200}	{50, 100, 200}	0.3	0	{5, 10, 15}	0.3	0.6	5
6	{10, 100, 200}	{50, 100, 200}	0.6	0	{5, 10, 15}	0.3	0.6	3

Tabla 3.5: Para las pruebas realizadas no se consideraron monedas con histórico incompleto, ver columna N_{nc}. A su vez, se trata de capturar el comportamiento de los parámetros p_{K_1}, p_{K_2} y fNK_1, ante el cambio en las dimensiones de la matriz del experimento $M_{N\times D}$ y ante la variación en la cantidad de días a predecir (columna D_p). El rango k para la técnica TCM fue elegido llevando a cabo el ajuste detallado en la Sección A.1.2.1 del Apéndice A

En la Tabla 3.6 se muestran los resultados promedio de cada técnica en RMSE y F1 para cada conjunto de parámetros especificados en la Tabla 3.5.

ID	**RMSE**					**F1**				
	TCP	TKM	TDP	TIP	TCM	TCP	TKM	TDP	TIP	TCM
1	**0.171**	0.173	0.264	0.175	0.280	**0.895**	**0.895**	0.838	0.893	0.791
2	**0.181**	**0.181**	0.489	0.186	0.222	0.807	**0.808**	0.481	0.803	0.712
3	0.177	**0.176**	0.266	0.180	0.190	**0.403**	0.401	0.145	0.399	0.341
4	0.503	0.501	0.500	0.500	**0.494**	0.486	0.485	**0.487**	**0.487**	0.458
5	0.467	0.463	0.493	**0.461**	0.501	0.547	0.551	0.522	**0.556**	0.434
6	0.440	0.445	0.479	**0.438**	0.469	0.461	0.452	0.413	**0.465**	0.414

Tabla 3.6: En negrita se resaltan los mejores resultados en RMSE y F1 para cada colección de parámetros especificados en la Tabla 3.5.

La Tabla 3.6 muestra que para los casos con $ID = 1, 2, 3$, en donde mayor es la diferencia entre el comportamiento de las monedas del perfil $K1$ y el perfil $K2$, es decir cuando ocurre una mayor diferencia entre las probabilidades p_{K1} y p_{K2}, las técnicas que emplean clustering (TCP y TKM) performan mejor que el resto, tanto en métrica RMSE como en F1.

Para el caso con $ID = 4$ se observa en la Tabla 3.6 que todas las técnicas tanto en RMSE como en F1 obtienen un valor promedio alrededor de 0.5. Debido a que todas las monedas tienen probabilidad 0.5 de salir cara y es igual de dificil predecir el siguiente resultado tanto para una técnica que emplea clustering como para una que no haga uso de dicha información.

Luego, viendo en la Tabla 3.6 los casos con $ID = 5, 6$, en donde la diferencia entre el comportamiento de las monedas del perfil $K1$ y $K2$ no es tan extrema como en los casos con $ID = 1, 2, 3$, se observa que la técnica TIP es la que mejor performa, secundada por las técnicas que emplean clustering (TCP y TKM).

Estos son los resultados principales, para una discusión adicional sobre los resultados ver la Sección A.1.2.2 del Apéndice A.

Resumen de los resultados aplicando Explanation Table

Para finalizar el análisis de la **prueba ST2**, se aplicó el algoritmo *flashlight* (ver Sección F.3.2 del Apéndice F) para generar una *Explanation Table* a modo de resumen, ver Tabla 3.7. Lo que se busca capturar en el resumen es un conjunto

de patrones que expliquen en que situaciones una técnica que explote la información de clustering (TCP) es mucho mejor en la predicción, respecto a una que desconozca los perfiles de las monedas (TDP). La tabla consiste en 6 atributos de clasificación: $N, D, D_p, fK_1, p_{K_1}, p_{K_2}$ y un atributo binario que vale 1 si la diferencia en métrica F1 es mayor a 0.25 a favor de TCP, respecto a TDP, y 0 en caso contrario. La *explanation table* resalta en sus dos primeras filas los resultados más importantes, la fila 1 dice que en un 30 % de los casos el modelo TCP obtuvo una métrica al menos 0.25 puntos mejor que TDP y la fila 2 dice que la situación más favorable para las técnicas que usan la información de perfiles es el caso en donde las probabilidades son muy diferentes $p_{K_2} = 0.9$ (p_{K_1} siempre es igual a 0.1 para este caso), y donde las proporciones entre ambos perfiles es la misma 0.5. Recordar que era esperado que el modelo TDP tuviese una tasa alta de errores para ambos perfiles en estas condiciones, ya que la probabilidad aprendida por TDP estaba alejada a la misma vez de las probabilidades de ambos perfiles. Para el patrón de la tercera fila $(*, *, *, 0.9, *, *)$, los únicos juegos de parámetros que coinciden son los dados por la fila con $ID = 3$ en la Tabla 3.5, pues son los únicos cuya fracción de monedas del perfil $K1$ es igual a $fNK_1 = 0.9$. La Figura 3.3, que representa los resultados para estos parámetros confirma el tercer resultado de la *explanation table*.

N	D	D_p	fNK_1	p_{K_1}	p_{K_2}	Tot.	Fracc.
*	*	*	*	*	*	162	0.296
*	*	*	0.5	*	0.9	27	0.962
*	*	*	0.9	*	*	27	0.814
RMSE 0.037							

Tabla 3.7: *Explanation table* para el caso en donde la métrica F1 tuvo una diferencia mayor a 0.25 a favor del modelo TCP versus TDP. El RMSE es entre la columna binaria real de la tabla y la columna de resultados binarios estimada por el algoritmo *flashlight*. La columna **Tot.** cuenta la cantidad de entradas en la tabla que coinciden con el patrón de cada fila. La columna **Fracc.** es la fracción de resultados con valor 1, del total de entradas de la tabla coincidentes con el patrón de cada fila.

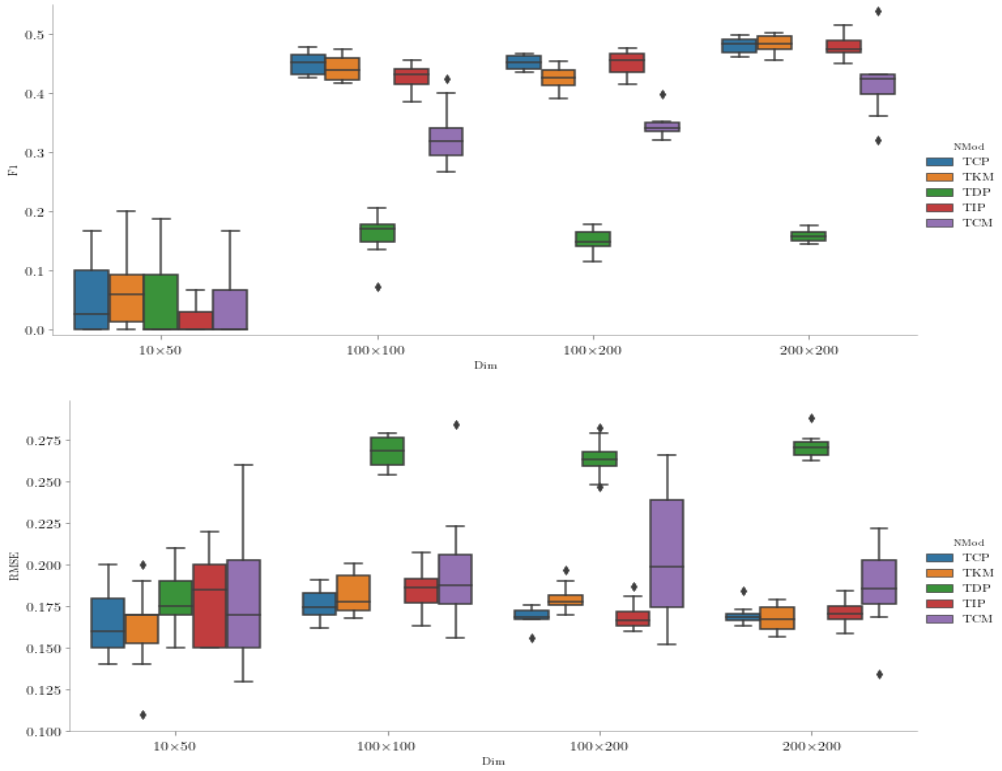


Figura 3.3: Los boxplots presentan los resultados de las simulaciones para los parámetros $p_{K_1} = 0.1$, $p_{K_2} = 0.9$ y $fNK_1 = 0.9$.

Prueba ST3: Monedas con observaciones incompletas

En esta prueba se agregan monedas con el comienzo de las observaciones en días próximos a la finalización del experimento. Los parámetros empleados se visualizan en la Tabla 3.8. Al igual que en la prueba anterior para cada juego de parámetros se genera una matriz $M_{N\times D}$ que cumpla con las restricciones impuestas por los parámetros y las predicciones sobre los D_p días de prueba se repiten 10 veces para todas las técnicas, obteniéndose valores promedios en métrica F1 y RMSE. La elección de los rangos k se detalla en la Sección A.1.2.1 del Apéndice A.

En este caso las técnicas se comparan con el benchmark de predecir el resultado de cada moneda con observaciones incompletas INC, definida en la sección anterior.

ID	N	D	fNK_1	DI	N_{nc}	D_p	p_{K_1}	p_{K_2}	k
1	{10, 100, 200}	{50, 100, 200}	0.1	{3, 10, 30}	10	10	0.1	0.9	5
2	{10, 100, 200}	{50, 100, 200}	0.5	{3, 10, 30}	10	10	0.1	0.9	5
3	{10, 100, 200}	{50, 100, 200}	0.9	{3, 10, 30}	10	10	0.1	0.9	3
4	{10, 100, 200}	{50, 100, 200}	0.5	{3, 10, 30}	10	10	0.5	0.5	3
5	{10, 100, 200}	{50, 100, 200}	0.3	{3, 10, 30}	10	10	0.3	0.6	5
6	{10, 100, 200}	{50, 100, 200}	0.6	{3, 10, 30}	10	10	0.3	0.6	3

Tabla 3.8: La tabla presenta el conjunto de parámetros explorados para esta prueba. Se observa que los días de predicción se mantienen constantes en 10 (ver columna D_p), al igual que la cantidad de monedas con histórico incompleto (columna N_{nc}). Mientras que se varía la cantidad de días observados para las monedas con histórico incompleto (columnas DI). El rango k para la técnica TCM fue elegido llevando a cabo el ajuste detallado en la Sección A.1.2.1 del Apéndice A

En la Tabla 3.9 se muestran los resultados promedio de cada técnica en RMSE y F1 para cada conjunto de parámetros especificados en la Tabla 3.8.

ID	RMSE						F1					
	TCP	TKM	TDP	TIP	TCM	INC	TCP	TKM	TDP	TIP	TCM	INC
1	0.185	0.184	0.301	0.179	0.401	**0.173**	0.877	0.876	0.803	0.882	0.576	**0.886**
2	0.183	0.189	0.502	**0.177**	0.420	0.184	0.783	0.781	0.453	**0.793**	0.252	0.787
3	**0.158**	0.159	0.273	0.169	0.173	0.164	0.315	0.317	0.125	0.304	0.139	**0.325**
4	0.500	0.500	0.505	0.495	0.501	**0.493**	**0.479**	**0.479**	0.467	0.478	0.236	0.462
5	0.470	0.471	0.496	0.471	0.508	**0.458**	0.478	0.484	0.477	0.495	0.255	**0.513**
6	0.465	0.468	0.502	0.463	**0.440**	0.453	0.415	0.410	0.368	0.414	0.266	**0.437**

Tabla 3.9: En negrita se resaltan los mejores resultados en RMSE y F1 para cada colección de parámetros especificados en la Tabla 3.8.

En primera instancia en la Tabla 3.9 se observa que la técnica INC obtiene buenos resultados para todas las colecciones de parámetros, como se infería de los resultados esperados.

La Tabla 3.9 muestra que para los casos con $ID = 1, 2, 3$, las técnicas que emplean clustering (TCP y TKM) performan mejor que la técnica TDP que no hace uso de dicha información.

Para el caso con $ID = 4$ (Tabla 3.9) se observa que todas las técnicas tanto en RMSE como en F1 obtienen un valor promedio alrededor de 0.5, salvo TCM que obtuvo un valor en métrica F1 de 0.236.

Luego en los casos con $ID = 5, 6$ (Tabla 3.9), en donde la diferencia entre el comportamiento de las monedas del perfil $K1$ y $K2$ no es tan extrema como en los casos con $ID = 1, 2, 3$, se observa que la técnica TIP que emplea la información individual de cada moneda, y las técnicas TCP y TKM que emplean clustering performan mejor que la técnica TDP que no hace uso de la información de clusters.

Estos son los resultados principales, para una discusión adicional sobre los resultados ver la Sección A.1.2.3 del Apéndice A.

Resumen de resultados obtenidos:

- Los perfiles con probabilidades muy diferentes favorecen la predicción de las técnicas que explotan la información de clustering: TCP y TKM.
- La técnica TIP, que utiliza solo la información de cada moneda para aprender su probabilidad tiene un rendimiento bueno también en F1 y RMSE para los perfiles con probabilidades muy diferentes.
- Los perfiles con probabilidades cercanas disminuyen la brecha en F1 y RMSE entre las técnicas que emplean clustering y TDP.
- Los valores promedios en F1 y RMSE de la técnica TCM, para predicción sobre las monedas con histórico completo (Prueba ST1), fueron mejores que TDP pero no mejores que TCP, TKM y TIP.
- El peor escenario para TDP es aquel en donde las probabilidades de los perfiles son muy diferentes y existen igual cantidad de monedas en ambos perfiles.
- La técnica INC, para predecir las tiradas de las monedas con histórico incompleto tiene buenos resultados para los diferentes tamaños de históricos evaluados (3,10 y 15).
- TCP, TKM y TIP son mejores técnicas para predecir las tiradas de las monedas con histórico incompleto, en comparación a TDP y TCM.

3.3.2. Modelo incluyendo temperatura

Ahora se hará un experimento un poco más complejo, introduciendo un factor externo (la temperatura) que influye en la probabilidad de obtener cara en cada perfil de moneda.
El experimento consiste en arrojar durante D días N monedas, y registrar para cada una de ellas si el resultado de la tirada fue cara o cruz. La matriz $M_{N\times D}$ almacenará en la i-ésima fila los D resultados de las tiradas para la moneda i, asumiendo un valor de 1 si el resultado es cara y 0 si es cruz. Para cada día se registrará la temperatura que hubo al momento de realizar las tiradas, las D temperaturas se guardarán en el arreglo T, las temperaturas en T serán solo dos: T_A y T_B, temperatura alta y baja respectivamente.
Los D días del experimento se dividen en D_e días de entrenamiento y D_p días de prueba, cumpliendo la igualdad $D = D_e + D_p$. No todas las monedas comienzan el experimento en el día 1, algunas pueden comenzar en un día d muy próximo al día final del entrenamiento D_e, lo que conlleva a una menor cantidad de información individual en comparación a las que comenzaron a registrar los resultados desde el primer día. La matriz $M_{N\times D}$ guardará el símbolo, $indefinido$, para las entradas que no tengan observaciones.
Existen dos perfiles de monedas K_1 y K_2. Cada perfil tendrá dos probabilidades de salir cara asociadas para sus monedas: $p_{K_1}(cara|T_A)$, $p_{K_1}(cara|T_B)$, son las probabilidades para cualquier moneda m de la clase K_1, de que el resultado del lanzamiento sea cara, condicionado a la temperatura que haya al momento de realizar el experimento y $p_{K_2}(cara|T_A)$, $p_{K_2}(cara|T_B)$, son las probabilidades para cualquier moneda m de la clase K_2, de que el resultado del lanzamiento sea cara, condicionado a la temperatura que haya al momento de realizar el experimento.
Se quiere para los D_p días de prueba, predecir para cada moneda si saldrá cara o cruz. La diferencia con el experimento que no incluye temperaturas (Sección 3.3.1) es que ahora los perfiles son sensibles a la temperatura y cambian sus probabilidades de salir cara dependiendo de la clase de temperatura que haya al momento de hacer el experimento. Este modelo es similar a la definición del modelo dado para la predicción del consumo de energía (Sección 3.1), sin embargo en este modelo teórico se quieren predecir valores discretos (cantidad de monedas) y no continuos (consumo de energía), además se tiene la ventaja de poder *controlar* que los perfiles sean distintos entre sí y que efectivamente haya un cambio en el comportamiento de las monedas ante cambios de temperaturas de una clase a otra, algo que de suyo no se puede asegurar para el modelado de sistemas reales.

3.3.2.1. Métricas

Las métricas empleadas para evaluar las diferentes técnicas aplicadas al modelo son: RMSE (Ec. 2.6) y F_1-score (Ec. 2.10).

3.3.2.2. Objetivo

El objetivo para este modelo es el mismo que para el modelo que no incluye temperatura (Sección 3.3.1.2).

3.3.2.3. Técnicas

Se proponen 5 técnicas que resuelven el problema de predicción empleando toda la información del modelo, o parte de esta. A continuación se define un método de aprendizaje y predicción. Todas, salvo una, de las técnicas definidas utilizan este método. La diferencia entre las técnicas que usan el método viene dada por los distintos parámetros de entrada que cada una ingresa al algoritmo del método. Luego de esto se definen las 5 técnicas a comparar.

Método de aprendizaje y predicción: Sea N la cantidad de monedas del experimento. N_c de ellas comienzan a registrar las observaciones del experimento a partir del primer día (histórico de tiradas completo), mientras que N_{nc} de ellas comienzan las observaciones en días próximos a D_e (histórico de tiradas no completo), el comienzo de las observaciones no tiene que ser el mismo día para estas N_{nc} monedas. Sea $\mathcal{K}$ un conjunto de perfiles de monedas, sea C un arreglo de etiquetas de tamaño N_c, con $C[i] = k$ indicando que la i-ésima moneda, con histórico completo, pertenece al perfil k. Notar en este punto que el pedir como parámetros de entrada $\mathcal{K}$ y C, no implica que el Algoritmo 6 del método conozca la etiquetas reales de las monedas con histórico completo, sino que esto significa que la técnica que implementa el algoritmo debe proporcionarle las etiquetas al mismo, sean las correctas o no. En principio se desconoce el perfil de las N_{nc} monedas con histórico incompleto y es tarea del algoritmo asignarles un perfil de los existentes en $\mathcal{K}$. El algoritmo del método de aprendizaje y predicción, denominado $MONEDAS_2$, toma como entrada $M_{N\times D_e}$, $\mathcal{K}$, C y los D_p días de predicción.
$MONEDAS_2$ (Alg. 6) se define en la Sección A.2.1.1 del Apéndice A y su invocación es la siguiente:

$$MONEDAS_2(M_{N\times D_e}, \mathcal{K}, C, D_p)$$

Se observa que el Algoritmo 6 conoce la temperatura para todos los días de entrenamiento.
Se observa que el Algoritmo 1 ($MONEDAS_1$), propuesto para el modelo sin temperaturas, es un caso particular del Algoritmo 6 ($MONEDAS_2$), en donde se puede asumir que hay un solo un tipo de temperatura y por lo tanto el paso de predicción de la temperatura para los días es trivial (bloque de código *Predecir la temperatura* del Algoritmo 6) ya que la única temperatura tiene probabilidad 1 de salir, al igual que el paso de obtención de submatrices (pasos 8 y 9 del bloque de código *Obtener submatrices para aprender las probabilidades, y asignar perfil a las monedas con histórico incompleto*) que también es trivial para el modelo sin temperatura, ya que no se descarta ninguna columna, luego los bloques de *Aprendizaje* y *Predicción* son los mismos.

A continuación se describen las técnicas. Sin pérdida de generalidad se asume que las primeras N_c filas corresponden a las monedas con observaciones en todos los días de entrenamiento y las $N - N_c$ restantes a las monedas con observaciones a partir de un día próximo al final del entrenamiento.

Técnica Conociendo Perfiles Conociendo Temperaturas (TCPCT),
TCPCT conoce a que perfil $k \in \{K_1, K_2\}$ cada moneda pertenece, por lo que $C[i] = k, i \in N_c$ contiene el perfil real de la moneda i.
El *algoritmo general de aprendizaje* toma como entrada $M_{N\times D_e}$, $\mathcal{K} = \{K_1, K_2\}$, C y los D_p días de predicción.
Invocación al Algoritmo 6:

$$MONEDAS_2(M_{N\times D_e}, \{K_1, K_2\}, C, D_p)$$

Técnica K-Means Conociendo Temperaturas (TKMCT)
TKMCT no conoce exactamente los perfiles de cada moneda, pero conoce que son 2, por lo que habrán dos perfiles $\{KM_1, KM_2\}$. El vector de etiquetas C almacenará las etiquetas devueltas luego de ejecutar el algoritmo $Kmeans(M_{N_c\times D_e}, k = 2)$, tomando como rasgos para cada moneda sus tiradas y cardinalidad $k = 2$.
El *algoritmo general de aprendizaje* toma como entrada $M_{N\times D_e}$, $\mathcal{K} = \{KM_1, KM_2\}$, $C = Kmeans(M_{N_c\times D_e}, k = 2)$ y los D_p días de predicción.
Invocación al Algoritmo 6:

$$MONEDAS_2(M_{N\times D_e}, \{KM_1, KM_2\}, Kmeans(M_{N_c\times D_e}, k = 2), D_p)$$

Técnica Desconociendo Perfiles Conociendo Temperaturas (TDPCT)
TDPCT presupone que todas las monedas se comportan de manera similiar, por

lo que se pueden considerar todas dentro de un mismo grupo, cabe aclarar que esto no implica que no haya perfiles, simplemente que esta técnica desconoce su existencia y por lo tanto no los ve. Llámese el único perfil presupuesto K, en este caso $C[i] = K, i \in N_c$. El *algoritmo general de aprendizaje* toma como entrada $M_{N\times D_e}$, $\mathcal{K} = \{K\}$, C y los D_p días de predicción.
Invocación al Algoritmo 6:

$$MONEDAS_2(M_{N\times D_e}, \{K\}, C, D_p)$$

Técnica Individualizando Perfiles Conociendo Temperaturas (TIPCT)

TIPCT utiliza como información para predecir la tirada de cada moneda solo su perfil individual, por lo que habrán N_c perfiles diferentes. En este caso $\mathcal{K} = \{1, \ldots, N_c\}$ y $C = [1, \ldots, N_c]$.
El *algoritmo general de aprendizaje* toma como entrada $M_{N\times D_e}$, $\mathcal{K} = \{1, \ldots, N_c\}$, $C = [1, \ldots, N_c]$ y los D_p días de predicción.
Invocación al Algoritmo 6:

$$MONEDAS_2(M_{N\times D_e}, \{1, \ldots, N_c\}, [1, \ldots, N_c], D_p)$$

Técnica de Completado de Matrices (TCMCT)

Esta técnica es la única que no emplea *el algoritmo general de aprendizaje* Alg. 6. El Algoritmo 7, utilizado por esta técnica, toma como entrada la matriz $M_{N\times D_e}$, el rango k de la matriz y la cantidad de días a predecir D_p. Devuelve una matriz completada aplicando el Algoritmo ICMC (ver Apéndice E).
Invocación al Algoritmo 7:

$$TCM_2(M_{N\times D_e}, \mathcal{K}, C, k, D_p)$$

En este caso también se observa que el Algoritmo 3 (TCM_1) es un caso particular del Algoritmo 7 (TCM_2), en donde se puede pensar que solo hay una temperatura durante todo el experimento y por lo tanto el paso de predicción de la temperatura (bloque de código *Predecir temperatura* del Algoritmo 7) y el paso de extracción de las submatrices (pasos 8 y 9 del bloque de código *Obtener submatrices para aplicar ICMC y asignar perfiles a monedas con histórico incompleto* del Algoritmo 7) son triviales, por las mismas razones dadas en esta sección para el Algoritmo 6 ($MONEDAS_2$).

3.3.2.4. Comportamiento esperado

Variación del error según la dimensión de la matriz de entrenamiento

El modelo de las monedas incluyendo temperaturas agrega una nueva complejidad, respecto al anterior, que es la predicción de la temperatura para los D_p días

de prueba. Notar que en el Algoritmo 6, luego de elegir la temperatura, el aprendizaje de las probabilidades para cada perfil se realiza igual que en el Algoritmo 1, con la salvedad de que ahora no se eligen todos los días para aprender las probabilidades sino que se elige el subconjunto de días cuyo registro de temperatura coincida con la pronosticada para el día que se quiere realizar la predicción. Lo mismo ocurre para el Algoritmo 7, luego de predecir la temperatura, el completado de las matrices es igual que en Algoritmo 3, con la salvedad de que la matriz incompleta contiene como columnas, aquellas asociadas a los días que coinciden con la temperatura predicha T_{sig} más la columna extra de predicción. Se infiere que mientras mayor dimensionalidad tenga la matriz M del experimento, mejor será para la etapa del aprendizaje de probabilidades para las técnicas que aplican el Algoritmo 6. Por ejemplo si se quieren predecir los últimos diez días en $M_{30\times20}$, solo habrán diez días de aprendizaje, de los cuales un subconjunto será utilizado para aprender las probabilidades de los perfiles para las temperaturas altas T_A y otro para las temperaturas bajas T_B.

Variación del error según la configuración de la temperatura

La proporción de temperaturas altas, en los D días, está relacionada con la posibilidad de error en el pronóstico de las temperaturas durante los D_p días de prueba, ya que la probabilidad de que el siguiente día del experimento tenga temperatura alta: $\hat{p}(T_A)$ aprendida en cada tiempo de predicción se calcula como la cantidad de temperaturas altas (de las observaciones registradas para los tiempos de entrenamiento y de las predicciones previas al tiempo actual de predicción) sobre el total de temperaturas registradas.
Por ejemplo si la proporción de temperaturas altas es 1, entonces la predicción es trivial, siempre se aprenderá: $\hat{p}(T_A) = 1$ y $\hat{p}(T_B) = 0$ para los siguientes D_p días. En cambio si la proporción es 0.5 entonces para cada predicción durante los D_p días habrá aproximadamente un 50 % de posibilidad de error: las probabilidades aprendidas aproximadamente serán $\hat{p}(T_A) \approx 0.5 \approx \hat{p}(T_B)$.
El error en la predicción de la temperatura implica elegir el subconjunto equivocado de días para aprender las probabilidades.
Es de interés evaluar un caso en donde haya un 50 % de probabilidades de error por ejemplo con la fracción de la temperatura del perfil A: $fT_A = 0.5$ y un caso en donde haya más certeza de acierto, por ejemplo $fT_A = 0.8$.

Comparación del modelo TCPCT vs. TCP

Otra cuestión a responder es cuanto pierde y cuanto gana un modelo al conocer la información de la temperatura. El análisis hecho en la Sección A.2.2.2 del

Apéndice A para los juegos de parámetros dados en la Tabla 3.10, proporciona un caso en el cual si el modelo TCPCT se equivoca en la predicción de la temperatura real T_A, prediciendo el nuevo día con temperatura T_B, entonces en métrica F1, TCPCT obtendrá un valor igual a 0, mientras que el modelo TCP que no conoce la temperatura obtendrá un valor en métrica F1 igual a 0.935. Esto permite advertir que si bien es ventajoso conocer la información de la temperatura, para un experimento en el cual esta es relevante, el agregado de una nueva complejidad también implica conocer como esta afectará al error asociado a la predicción y en que casos un error en la determinación de la temperatura conllevará a resultados considerablemente erróneos de predicción.

N	D	fNK_1	N_{nc}	D_p	$p_{K_1}(cara\|T_A)$	$p_{K_1}(cara\|T_B)$	$p_{K_2}(cara\|T_A)$	$p_{K_2}(cara\|T_B)$
100	100	0.1	10	10	0	1	1	0

Tabla 3.10

Benchmark para históricos incompletos

Las técnicas se comparan con el benchmark de predecir el resultado de cada moneda con observaciones incompletas, solo con los datos de éstas. La probabilidad de salir cara para cada moneda se calcula como la suma total de caras sobre la cantidad de tiradas observadas, para los días de entrenamiento que coincidan con la temperatura predicha, llámese el benchmark: técnica INCompleta (INC).
Se puede pensar que para históricos incompletos de tamaño pequeño, por ejemplo 3, la técnica INC debería performar mal en F1 y RMSE, sin embargo INC igualmente puede aprender con bastante exactitud la probabilidad del perfil, véase la Tabla 3.11 para casos de históricos favorables para INC con históricos de tamaño 3.

p_{Talta}	p_{Tbaja}	Histórico	$\hat{p}_{Talta}$	$\hat{p}_{Tbaja}$
0.1	0.9	**0** 1 1	0	1
0.1	0.9	1 1 1	0	1
0.3	0.6	**0** **0** **1**	0.3 . . .	0
0.3	0.6	0 1 0	0	0.6 . . .

Tabla 3.11: Los valores históricos encuadrados indican resultados de tiradas en días con temperaturas altas, los valores históricos sin encuadrar indican resultados de tiradas en días con temperaturas bajas. En verde se indican las probabilidades que fueron bien aproximadas por INC, en rojo aquellas que no corrieron tal fortuna.

Resumen de resultados esperados

- A medida que el número de filas y columnas de la matriz de entrenamiento aumenta, se espera que las técnicas de predicción obtengan mejores resultados en RMSE y F1.
- Se espera que mientras más iguales sean la cantidad de días con temperaturas altas y la cantidad de días con temperaturas bajas, aumente el error de predicción para las técnicas.
- La equivocación en la predicción de la temperatura para una técnica que emplee esa información, puede implicar una baja performance en F1 y RMSE, para los días de los D_p en los cuales se registra esta equivocación.
- El benchmark INC puede performar bien aún con poca información histórica.

3.3.2.5. Resultados

Los parámetros utilizados en las pruebas son los dados para el primer modelo Sección 3.3.1.5, con la salvedad de que ahora por cada perfil habrán dos probabilidades asociadas, más el parámetro de temperatura:

- $p_{K_1}(cara|T_A), p_{K_1}(cara|T_B)$, probabilidades asociadas al perfil K_1 de salir cara para las dos temperaturas del experimento.
- $p_{K_2}(cara|T_A), p_{K_2}(cara|T_B)$, probabilidades asociadas al perfil K_2 de salir cara para las dos temperaturas del experimento.
- fT_A, fracción de temperaturas altas en los D días.

En esta sección se realizarán tres pruebas. La primera prueba realizada está enfocada en evaluar el comportamiento de las técnicas sin considerar monedas con histórico incompleto y la segunda considerando la existencia de este tipo de monedas. A modo de evaluar por separado el funcionamiento de las técnicas para ambas situaciones. La tercera y última apunta a comparar los modelos TCP y TCPCT, el primero de ellos ignora la información de la temperatura y el segundo hace uso de esta información al momento de predecir el resultado de los lanzamientos.

Prueba CT1: Todos los históricos completos

Los valores de los parámetros para esta prueba se detallan en la Tabla 3.12. Para

cada juego de parámetros se genera una matriz $M_{N\times D}$ que cumpla con las restricciones impuestas por los parámetros. Las predicciones de los D_p días de prueba se repiten 10 veces para todas las técnicas. Obteniéndose un valor en métrica RMSE y F1 promedio sobre las 10 simulaciones.
Además, para cada tupla de parámetros

$$(fNK_1, fT_A, p_{K_1}(cara|T_A), p_{K_1}(cara|T_B), p_{K_2}(cara|T_A), p_{K_2}(cara|T_B))$$

se elige el mejor rango k, para el modelo TCMCT. La elección de los rangos k se detalla en la Sección A.2.2.1 del Apéndice A.

ID	N	D	fNK_1	N_{nc}	D_p	fT_A	$p_{K_1}(cara\|T_A)$	$p_{K_1}(cara\|T_B)$	$p_{K_2}(cara\|T_A)$	$p_{K_2}(cara\|T_B)$
1	{30, 100, 200}	{30, 100, 200}	{0.1, 0.5, 0.9}	0	{1, 5, 10}	0.8	0.1	0.9	0.9	0.1
2	{30, 100, 200}	{30, 100, 200}	{0.1, 0.5, 0.9}	0	{1, 5, 10}	0.5	0.1	0.9	0.9	0.1
3	{30, 100, 200}	{30, 100, 200}	{0.1, 0.5, 0.9}	0	{1, 5, 10}	0.8	0.3	0.6	0.6	0.3

Tabla 3.12: Los parámetros son N-cantidad de monedas, D-cantidad de días del experimento, fNK_1-fracción de monedas con perfil K_1, N_{nc}-cantidad de monedas con histórico incompleto (en esta primera prueba 0 en todos los casos), D_p-días de predicción, fT_A-fracción de temperaturas T_A en el total de D días, $p_{K_1}(cara|T_A)$,$p_{K_1}(cara|T_B)$,$p_{K_2}(cara|T_A)$,$p_{K_2}(cara|T_B)$-probabilidades de las monedas de cada perfil para cada temperatura.

En la Tabla 3.13 se muestran los resultados promedio de cada técnica en RMSE y F1 para cada conjunto de parámetros especificados en la Tabla 3.12.

ID	**RMSE**					**F1**				
	TCPCT	TKMCT	TDPCT	TIPCT	TCMCT	TCPCT	TKMCT	TDPCT	TIPCT	TCMCT
1	0.387	**0.383**	0.446	0.386	0.425	0.566	**0.577**	0.482	0.564	0.464
2	0.516	0.505	**0.496**	0.502	0.514	0.452	0.458	**0.461**	0.459	0.415
3	**0.467**	0.475	0.485	0.468	0.496	**0.463**	0.456	0.440	**0.463**	0.378

Tabla 3.13: En negrita se resaltan los mejores resultados en RMSE y F1 para cada colección de parámetros especificados en la Tabla 3.12.

En la Tabla 3.13 se observa que para los casos con $ID = 1$ y 3 en donde hay mayor probabilidad de acertar la temperatura del día para el cual se predecirán las tiradas (80 % de probabilidad de que la temperatura sea alta) las técnicas que emplean clustering (TCPCT, TKMCT) y la técnica que predice con la información individual de las monedas (TIPCT) performan mejor en RMSE y F1 en comparación a TDPCT que no hace uso de la información de los perfiles y a TCMCT que

emplea clustering más completado de matrices. A su vez los resultados en RMSE y F1 son mejores para TCPCT y TKMCT para el caso con $ID = 1$ en comparación a los obtenidos para el caso con $ID = 3$ debido a que hay mayor diferencia en el comportamiento entre las monedas de diferentes perfiles.

Observando los resultados para el caso con $ID = 2$ en la Tabla 3.13, se ve que a pesar de que la diferencia en el comportamiento de las monedas entre los perfiles es muy diferente, los resultados en RMSE y F1 no son mejores para las técnicas que emplean clustering. Esto se debe a la alta tasa de error dada en la predicción de la temperatura para el nuevo día (50 % de probabilidad de que la temperatura sea alta), lo que conlleva a una disminución en la performance de las técnicas que emplean clustering.

En la Sección A.2.2.3 del Apéndice A se pueden apreciar los boxplots para diferentes juegos de parámetros de la prueba.

Resumen de los resultados aplicando Explanation Table

Con el fin de resumir cuales son las mejores condiciones, en la **prueba CT1**, para que las técnicas de clustering que utilizan la información de los perfiles den mejores resultados que la técnica TDPCT, que asume un único perfil para todas las monedas, se aplica el algoritmo *flashlight* (ver Sección F.3.2 del Apéndice F) para generar una *explanation table*. La técnica TCPCT que conoce exactamente los perfiles de cada moneda es la utilizada para comparar con TDPCT. La métrica elegida para la comparación es F1. Los atributos de clasificación son los parámetros utilizados en las pruebas y el atributo binario vale 1 si la medición F1 de TCPCT es mayor por una diferencia de 0.15 a la de TDPCT. En la Tabla 3.14 se describe el problema y se muestran los 4 patrones del resumen.
En la Sección A.2.2.3 del Apéndice A se presentan el análisis exhaustivo de cada patrón de la Tabla 3.14.
Resumiendo lo más importante: el primer patrón indica que solo en un 11 % de los casos la meta de la diferencia en métrica F1 fue obtenida y el segundo patrón reafirma que ante mayor certidumbre en la predicción de la temperatura $fT_A = 0.8$, la técnica TCPCT aprovecha mejor la información de entrenamiento, en comparación a TDPCT.

N	D	D_p	fK_1	fT_A	$p_{K_1}(cara\|T_A)$	$p_{K_1}(cara\|T_B)$	$p_{K_2}(cara\|T_A)$	$p_{K_2}(cara\|T_B)$	Tot.	Fracc.
*	*	*	*	*	*	*	*	*	243	0.115
*	*	*	*	0.8	*	*	*	0.1	81	0.333
*	*	*	0.9	*	*	*	*	0.1	54	0.018
*	*	1	0.1	0.5	*	*	*	*	9	0.111
*	*	*	*	*	*	*	*	0.1	162	0.172
RMSE: 0.1007										

Tabla 3.14: *Explanation table* que resume los patrones claves para entender cuando el modelo TCPCT es mejor en métrica F1 a TDPCT, por una diferencia de 0.15 puntos. Las primeras columnas son los atributos categóricos: N-cantidad de monedas, D-cantidad de días en que se realiza el experimento, D_p-cantidad de días de predicción, fK_1-proporción de monedas con perfil K_1, fT_A-proporción de días con temperaturas T_A, $p_{K_1}(cara|T_A)$,$p_{K_1}(cara|T_B)$,$p_{K_2}(cara|T_A)$,$p_{K_2}(cara|T_B)$-probabilidades de los perfiles. La columna **Tot.** resume el total de entradas en la tabla que coinciden con cada patrón de cada fila, **Fracc.** fracción de entradas de la tabla que coinciden con el patrón y satisfacen la condición de que el resultado promedio en F1 de TCPCT sea mayor al de TDPCT, por una diferencia mayor a 0.15 puntos. El RMSE es entre la columna binaria real de la tabla y la columna de resultados binarios estimada por el algoritmo *flashlight*.

Prueba CT2: Históricos incompletos

Los valores de los parámetros para esta prueba se detallan en la Tabla 3.15. Para cada juego de parámetros se genera una matriz $M_{N\times D}$ que cumpla con las restricciones impuestas por los parámetros. Las predicciones sobre los D_p días de prueba se repiten 10 veces para todas las técnicas. Obteniéndose valores promedios sobre las 10 simulaciones para las métricas RMSE y F1. La elección de los rangos k, que no se muestran en la tabla, se detalla en la Sección A.2.2.1 del Apéndice A.

ID	N	D	fNK_1	N_{nc}	D_p	fT_A	$p_{K_1}(cara\|T_A)$	$p_{K_1}(cara\|T_B)$	$p_{K_2}(cara\|T_A)$	$p_{K_2}(cara\|T_B)$
1	{30, 100, 200}	{30, 100, 200}	{0.1, 0.5, 0.9}	{3, 10, 30}	{1, 5, 10}	0.8	0.1	0.9	0.9	0.1
2	{30, 100, 200}	{30, 100, 200}	{0.1, 0.5, 0.9}	{3, 10, 30}	{1, 5, 10}	0.5	0.1	0.9	0.9	0.1
3	{30, 100, 200}	{30, 100, 200}	{0.1, 0.5, 0.9}	{3, 10, 30}	{1, 5, 10}	0.8	0.3	0.6	0.6	0.3

Tabla 3.15: Los parámetros son N-cantidad de monedas, D-cantidad de días del experimento, fNK_1-fracción de monedas con perfil K_1, N_{nc}-cantidad de monedas con histórico incompleto, D_p-días de predicción, fT_A-fracción de temperaturas T_A en el total de D días, $p_{K_1}(cara|T_A)$,$p_{K_1}(cara|T_B)$,$p_{K_2}(cara|T_A)$,$p_{K_2}(cara|T_B)$-probabilidades de las monedas de cada perfil para cada temperatura.

En la Tabla 3.16 se muestran los resultados promedio de cada técnica en RMSE y F1 para cada conjunto de parámetros especificados en la Tabla 3.15.

ID	RMSE						F1					
	TCPCT	TKMCT	TDPCT	TIPCT	TCMCT	INCCT	TCPCT	TKMCT	TDPCT	TIPCT	TCMCT	INCCT
1	0.362	0.368	0.439	**0.358**	0.469	**0.358**	0.519	0.541	0.438	0.535	0.269	**0.555**
2	0.498	0.501	**0.497**	0.499	**0.497**	0.501	0.424	0.420	0.394	0.420	0.206	**0.425**
3	0.481	0.484	0.485	0.474	**0.468**	0.471	0.427	0.420	0.420	0.423	0.175	**0.430**

Tabla 3.16: En negrita se resaltan los mejores resultados en RMSE y F1 para cada colección de parámetros especificados en la Tabla 3.15.

En primera instancia en la Tabla 3.16 se observa que la técnica INC obtiene buenos resultados, sobre todo en métrica F1, para todas las colecciones de parámetros, como se infería de los resultados esperados.
Luego se observa que para el caso con $ID = 1$ en la Tabla 3.16, las técnicas que emplean clustering (TCPCT y TKMCT) performan mejor que la técnica TDPCT que no hace uso de esta información. Tanto TCPCT y TKMCT performan peor en métrica RMSE y F1, en comparación a la técnica TIPCT, que hace uso de la información individual de cada moneda, con el costo de tener que aprender N probabilidades, en lugar de las 2 que son aprendidas por TCPCT y TKMCT.

Para el caso con $ID = 2$ en la Tabla 3.16, todas las técnicas tanto en RMSE como en F1 obtienen un valor promedio alrededor de 0.5, salvo TCMCT y TDPCT que obtuvieron valores menores a 0.4 en métrica F1. En comparación a los juegos de parámetros para el caso con $ID = 1$, el comportamiento entre los perfiles sigue siendo muy diferente, pero aumenta la tasa de error en el acierto de la temperatura para el siguiente día en que se van a predecir los lanzamientos (aproximadamente 80 % de probabilidad de salir cara para el caso $ID = 1$, contra aproximadamente un 50 % de salir cara para el caso con $ID = 2$).

Observando el caso con $ID = 3$ en la Tabla 3.16, en donde la probabilidad de acertar la temperatura es igual al caso con $ID = 1$, pero el comportamiento entre los perfiles es más parecido, se ve que la diferencia en los valores de las métricas RMSE y F1 entre las técnicas que emplean clustering (TCPCT y TKMCT) y TDPCT que no hace uso de esta información disminuye, siendo aún mejor en resultados emplear la información de clustering en comparación a no emplearla.

En la Sección A.2.2.4 del Apéndice A se pueden apreciar los boxplots para diferentes juegos de parámetros de la prueba.

Prueba CT3: Comparar modelo TCP vs TCPCT

La Figura 3.4 muestra los resultados en F1 y RMSE de dos simulaciones para comparar las técnicas TCP y TCPCT. En este caso hay dos versiones de TCPCT:

- TCPCT1, conoce exactamente la temperatura de los 10 días a predecir.
- TCPCT2, no conoce la temperatura de los 10 días a predecir y debe predecirla.

Se observa que el modelo que debe predecir la temperatura $TCPCT2$ está por debajo de $TCPCT1$ que sí conoce las temperaturas de los días a predecir. Representándose de esta manera el error asociado con la predicción de la temperatura. Mientras que el modelo que no conoce la información de la temperatura, TCP, se encuentra en ambas métricas por debajo de $TCPCT1$ y $TCPCT2$. Este último resultado es positivo por el hecho de que hay casos en donde conocer la temperatura mejora la predicción.

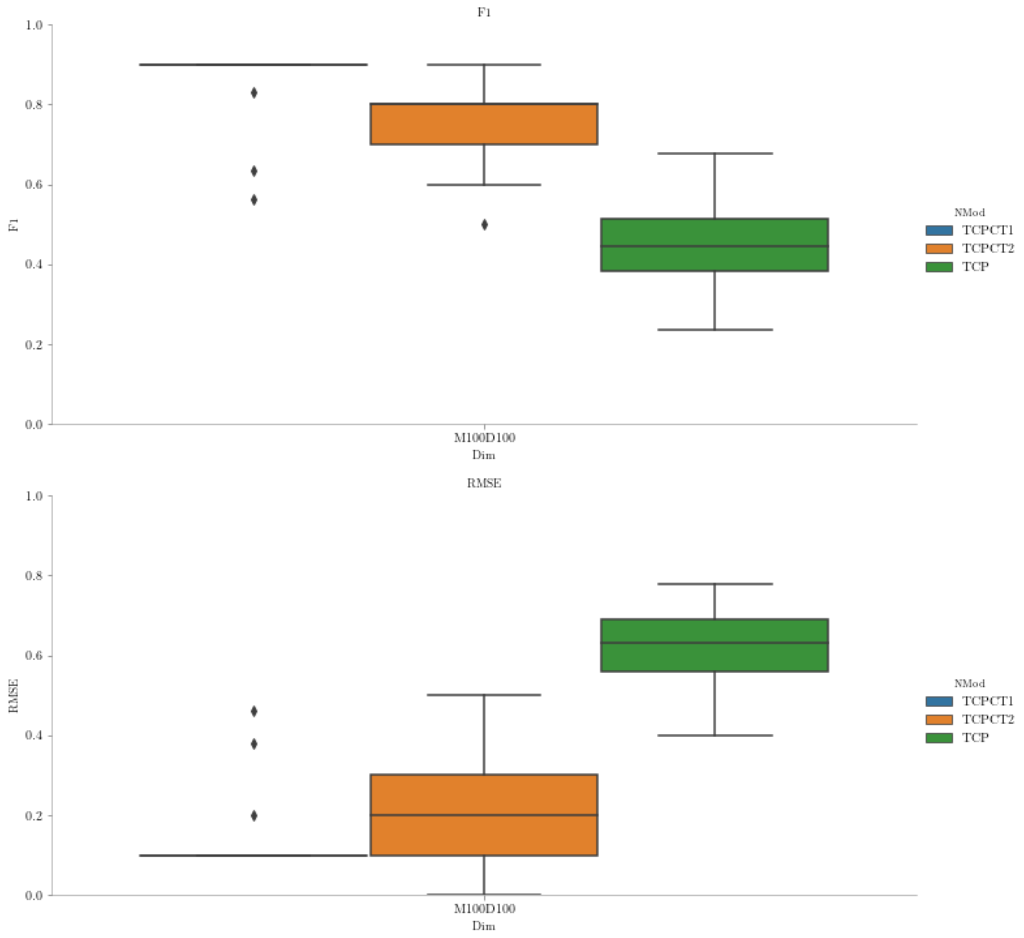


Figura 3.4: Las gráficas muestran los resultados, F1 y RMSE, para los parámetros dados en la Tabla A.8 del Apéndice A.

Resumen:

- La técnica INC con perfiles incompletos arroja buenos resultados. En los casos en donde los perfiles de las monedas se comportan muy diferentes INC puede performar correctamente aún con pocos valores históricos observados para las monedas con histórico incompleto.
- El error en la predicción de la temperatura afecta de manera negativa a los resultados de la predicción. Sobre todo a las técnicas que explotan la información de los perfiles. El valor $fT_A = 0.5$ por ejemplo aporta un 50 % de

probabilidad de error, en la predicción de la temperatura del siguiente día a predecir.

- Los modelos que emplean clustering TCPCT, TKMCT, y la técnica que emplea el histórico individual de cada moneda TIPCT en general performan mejor que TDPCT y TCMCT.
- Los resultados de la *explanation table* afirman la aseveración de que una vez superada la correcta predicción de la temperatura, el modelo se comporta de igual manera que el anterior.
- De acuerdo a los patrones observados en la *explanation table* los parámetros más relevantes que favorecen el usar la información de los perfiles son las probabilidades y las fracciones de monedas de los perfiles y de temperaturas. La cantidad de monedas y el número de días del experimento no reflejaron un cambio relevante en el comportamiento de las técnicas.
- Hay casos en donde conocer la temperatura mejora visiblemente la predicción respecto a modelos que no hacen uso de dicha información.

Capítulo 4

Descripción del Dataset

Dejando de lado los datos experimentales teóricos, es tiempo de definir los datasets con los cuales trabajar en el modelo real, formalizado en la Sección 3.1. En este trabajo se emplean dos datasets: uno con la información del consumo individual de energía en los hogares para clientes del sureste de Australia (Smart Grid Smart City dataset) y el otro con información de las temperaturas registradas en la zona al momento de tomar el consumo (timeanddate dataset). El capítulo se divide en dos partes. En la primera parte se describe la información general del dataset de consumo de energía. Con el objetivo de entender algunos de los aspectos del comportamiento de los clientes a considerar por las técnicas de forecasting, se realiza un análisis de consumo para un cliente elegido al azar. Al final de la primera parte se resumen los pasos del preprocesamiento realizado a los datos del dataset de consumo.
En la segunda parte se describe el dataset de temperaturas, junto con un resumen de la variación de la temperatura a lo largo de un año. Se realiza un análisis enfocado en estudiar la correlación entre el consumo de energía y la temperatura. Al final de la segunda parte, se explica como fueron obtenidos los datos de temperatura para este trabajo.

4.1. Smart Grid Smart City dataset (SGSC)

SGSC es un dataset generado en conjunto por el Departamento de Medioambiente y Energía del Gobierno de Australia y el consorcio industrial de Ausgrid, en el marco del proyecto *Smart Grid Smart City*. El proyecto comenzó en Octubre del 2010 y culminó en Julio del 2014. El objetivo fue recolectar información robusta

sobre los costos y beneficios de la implantación de *smart grids* en las ciudades, para informar sobre desiciones futuras sobre el empleo de estas tecnologías.
Los *smart grids* fueron desplegados en el sureste de Australia, con el punto principal de instalación en la ciudad de Newcastle, New South Wales.

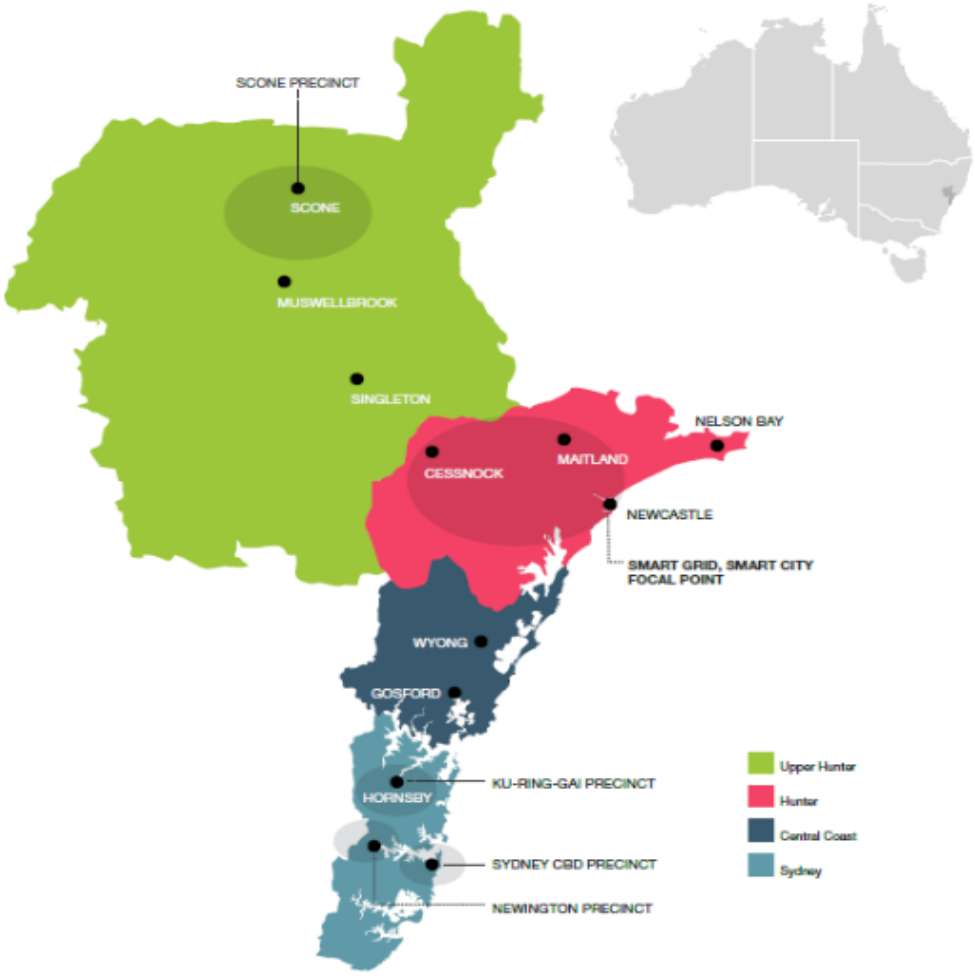


Figura 4.1: El punto central del proyecto estuvo enfocado en la ciudad de Newcastle. También se instalaron medidores en otras regiones de New South Wales, como Upper Hunter (región verde), Hunter (región roja), Central Coast (región azul oscuro) y Sydney (región celeste). Imagen extraída del Reporte ejecutivo [5] generado para el proyecto SGSC.

Los datos generales del dataset son los siguientes:

Campo	Valor
Título	Smart-Grid Smart-City Customer Trial Data
Tipo de contenido	Dataset
Agencia	Department of the Environment and Energy
Fecha de publicación	2015-09-09
Cobertura temporal	2011-10-10 al 2014-02-13
Cobertura geoespacial	GA1318: NEWCASTLE
Cantidad de clientes	402852
Intervalo entre cada medición	30 minutos

4.1.1. Análisis de consumo de un cliente

En la Figura 4.2 se visualiza el consumo de energía para el cliente con ID = 10042756, cada tres horas, durante el Sábado 14 de Abril del año 2012. Se observan dos picos de consumo con un intervalo de 12 horas, a las 3 y 15 horas, mientras que al mediodía se da el nivel más bajo de consumo. En primera instancia no se puede asegurar que este sea el comportamiento del cliente todos los días. Incluso el comportamiento del consumo puede cambiar dependiendo de si el día corresponde a un fin de semana (como lo es en este caso) o a un día de semana.

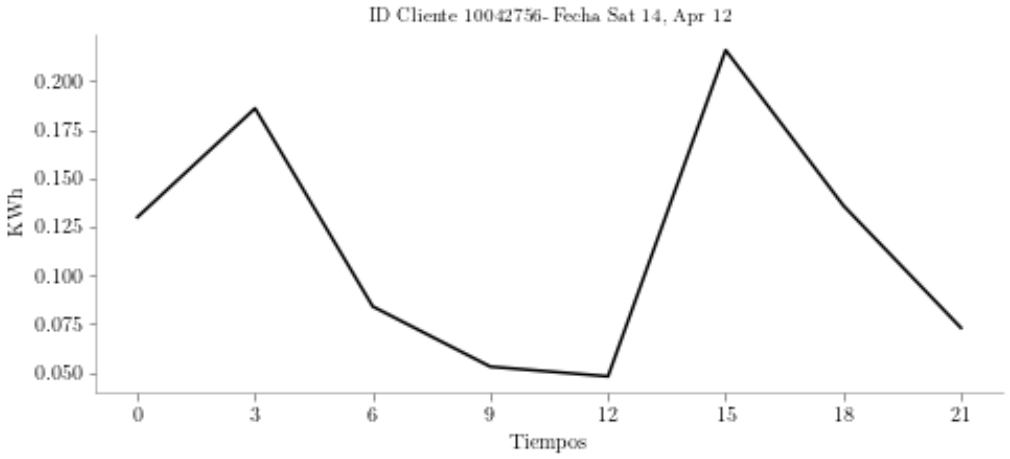


Figura 4.2: Consumo de energía del cliente con ID = 10042756, cada tres horas, durante el Sábado 14 de Abril del año 2012.

Si se observa la Figura 4.3, que representa el comportamiento en el consumo

del cliente con $ID = 10042756$ durante la semana con inicio el 14 de Enero del 2013, el análisis se torna un poco menos intuitivo. No obstante se nota una tendencia a la disminución del consumo a medida que transcurre la semana, con dos picos de consumo a media semana, uno a la medianoche del martes y el otro al mediodía del miércoles. A pesar de la dificultad del análisis, empleando técnicas de descomposición estacional, la curva de consumo puede ser descompuesta en tres términos, un término que refleje *la tendencia* a disminuir o aumentar el consumo (*Trend* Secc. 2.1.3), un término que exprese la componente estacional del consumo (*Seasonal* Secc. 2.1.3) y el último término representando el residuo o parte del consumo cuya información no es capturada por los dos términos anteriores. En la Figura 4.4 se puede ver la descomposición para una estacionalidad de frecuencia diaria. Esta gráfica confirma la disminución en la tendencia del consumo a medida que transcurren los días y llega el fin de semana. Mientras que permite observar que la curva de consumo presenta un comportamiento estacional diario, con un pico de consumo a la medianoche y dos más pequeños a las 9 y 18 horas.

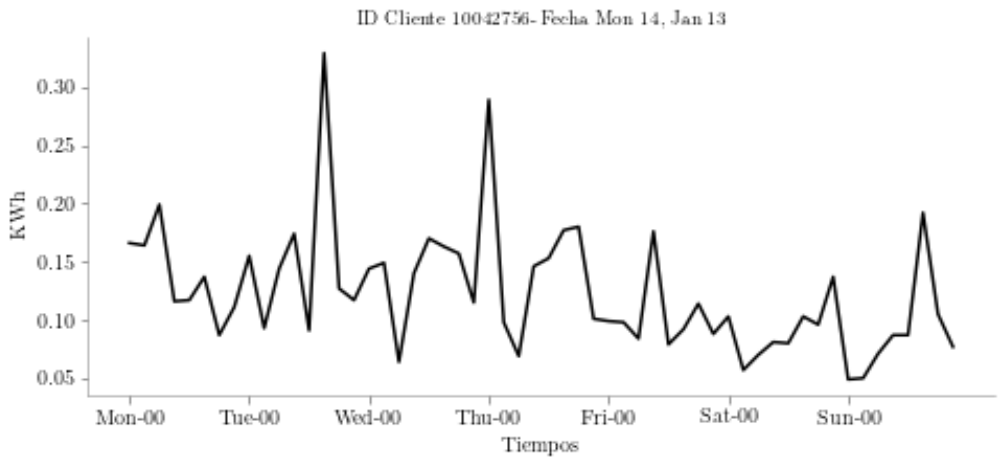


Figura 4.3: El eje y es el consumo en KWh. El eje x es el tiempo (Dia-Hora) de observación de la semana.

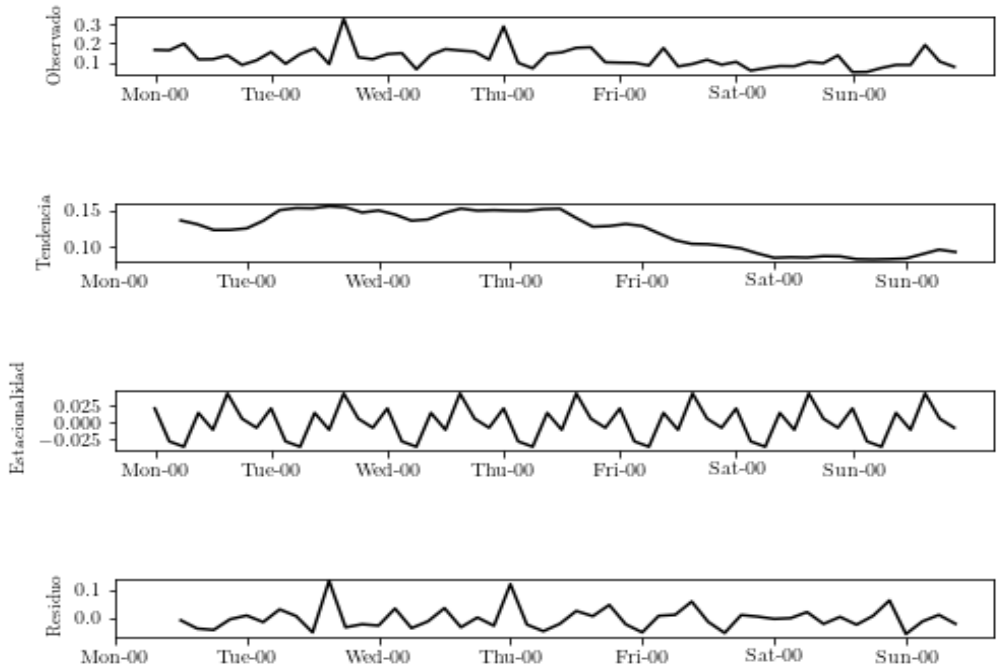


Figura 4.4

Es importante que la técnica de predicción empleada pueda capturar la información estacional y los cambios de tendencia en pos de mejorar la predicción. El ejemplo anterior refleja un patrón estacional diario. Otra posibilidad es considerar patrones estacionales semanales o anuales como los cambios de estaciones. Por ejemplo en la Figura 4.5 se visualiza el consumo del cliente con $ID = 10042756$ durante los años 2012 y 2013. Se observa en primera instancia que en los meses de Julio y Agosto (invierno) se presenta un mayor consumo en comparación a Diciembre (verano) con un menor consumo.

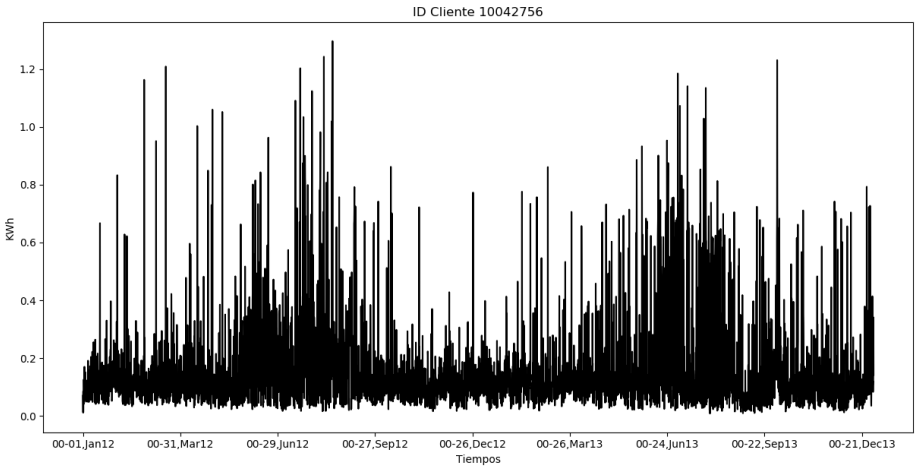


Figura 4.5: El eje y es el consumo en KWh. El eje x es el tiempo (Hora-Dia, MesAño) de observación.

4.1.1.1. Preprocesamiento

SGSC cuenta con datos de consumo para 402852 clientes, cada consumo registrado de cada cliente se encuentra en el archivo Electricity Use Interval Reading, que en su formato descomprimido *.xml* pesa 17.7GB. El archivo se encuentra ordenado por el campo identificador (ID) de cada cliente.
Para poder tratar el problema, el primer paso del preprocesamiento es extraer un subconjunto de clientes del total. Como el archivo no puede cargarse de forma completa en memoria, este se debe leer en forma parcial, colocando en memoria en cada iteración de la lectura un nuevo subconjunto de registros de consumo cada vez. En este caso se eligió leer un registro de consumo nuevo por cada acceso de lectura al archivo. Si bien esto es más costoso en tiempo de ejecución, ya que requiere un acceso a disco para leer cada nuevo registro de consumo, la lógica para detectar cuando terminan los registros de un cliente y comienzan los registros del otro es más simple, pues se pregunta si el ID del cliente de la lectura actual es igual o distinto al ID del cliente de la lectura anterior.
Haciendo un balance entre el tiempo que puede demorar la extracción de todos los registros del subconjunto de clientes y el tener una cantidad de clientes suficientes como para capturar diferentes perfiles de consumo, la cantidad de clientes elegida para conformar el primer paso del preprocesamiento fue de 1000.

El segundo paso es elegir un rango de fechas para trabajar solo con los consumos de los clientes en ese rango. La cobertura temporal de las mediciones en SGSC van desde Octubre del 2011 a Febrero del 2014.
Considerando que es posible que no todos los clientes hayan tenido instalados desde el primer día los *Smart Meters*, para registrar la generación y consumo de energía en sus hogares, se eligió un periodo de tiempo de un año, en el cual el proyecto ya estuviese lo suficientemente maduro como para tener la gran parte de *Smart meters* funcionando. Es por esto que las fechas iniciales y finales corresponden al inicio y fin del año 2013:

- Fecha inicial: $2013-01-01,\ 00:00:00$
- Fecha final: $2013-12-31,\ 23:59:59$

La frecuencia de toma de medición de consumo en SGSC es cada 30 minutos, es decir 48 mediciones por día. Para el rango de fechas elegido se deben registrar, en caso de que no ocurran interrupciones en los medidores un total de 17520 ($365 \cdot 48$) mediciones. Como parte del segundo paso del preprocesamiento, si alguno de los 1000 clientes en este periodo presenta menos de 17520 mediciones, este es descartado del subconjunto final de clientes. Quedando luego de este paso un total de 539 clientes.
El tercer paso del preprocesamiento consiste en bajar la frecuencia de consumos diarios de una medición cada 30 minutos a una cada tres horas, esto se logra quedándose solo con los valores de consumo diario de los clientes para las siguientes horas: $0, 3, 6, 9, 12, 15, 18, 21$ y descartando las mediciones para el resto de los tiempos. Las dimensiones del dataset final de consumo de energía, con el cual se va a trabajar, es de 539 clientes y 2920 ($365 \cdot 8$) registros de consumo medidos para cada cliente.

4.2. Timeanddate dataset

Time and Date AS es una empresa ubicada en Stavanger, Noruega, dedicada a recolectar, presentar y predecir información climática y horaria a nivel mundial. Los servicios están disponibles en el sitio https://www.timeanddate.com/. De este sitio se extrajo la información de la temperatura de la ciudad de Newcastle, New South Wales, cada tres horas durante el año 2013. El dataset está conformado por un archivo csv, con dos columnas, la primera indicando la fecha y hora de la temperatura registrada y la segunda columna registrando la temperatura para este tiempo. La información general, del dataset, es la siguiente:

Campo	Valor
Página	https://www.timeanddate.com/
Tipo de contenido	Dataset
Cobertura temporal	2013-01-01 al 2013-12-31
Cobertura geoespacial	NEWCASTLE, NEW SOUTH WALES
Intervalo entre cada medición	3 horas

En la Figura 4.6 se visualiza la variación de la temperatura durante el año 2013 en Newcastle, New South Wales, Australia.

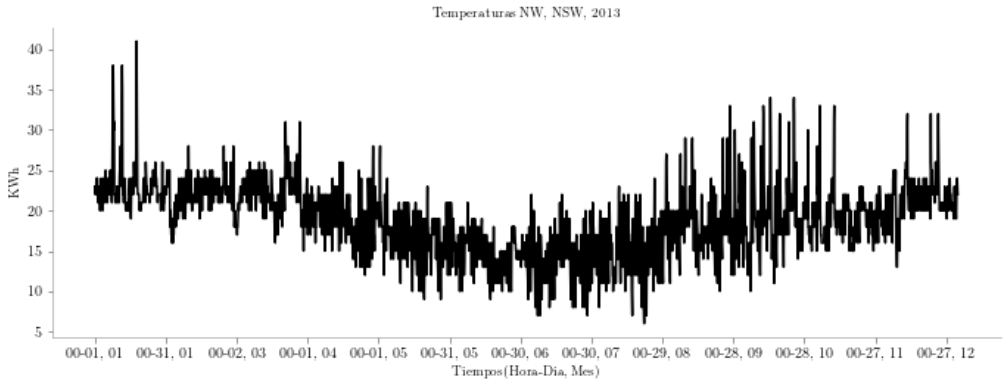
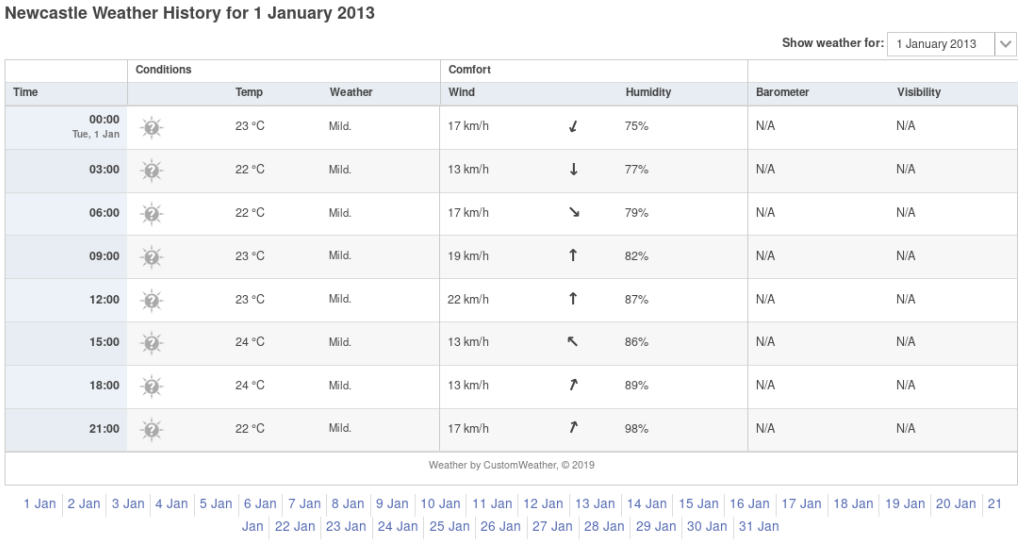


Figura 4.6: El eje y representa el valor de la temperatura en grados Celsius. El eje x representa el tiempo en que fue registrada la temperatura, el formato del eje es: `Hora-Día, Mes`. Se observa que para los meses de invierno Junio-Julio-Agosto ocurren los valores más bajos de temperatura. Mientras que en solo cinco ocasiones, a comienzo de año, las temperaturas superaron los 35 grados Celsius.

4.2.1. Preprocesamiento

Dentro de la información climática y de horarios para la ciudad de Newcastle en la página de timeanddate, existe una sección con datos climáticos pasados, que van desde los datos actuales hasta Febrero del año 2010. En la Figura 4.7 se ve la información climática cada tres horas para el 1° de Enero del año 2013.



Newcastle Weather History for 1 January 2013

Show weather for: 1 January 2013

	Conditions			Comfort				
Time		Temp	Weather	Wind		Humidity	Barometer	Visibility
00:00 Tue, 1 Jan		23 °C	Mild.	17 km/h	↙	75%	N/A	N/A
03:00		22 °C	Mild.	13 km/h	↓	77%	N/A	N/A
06:00		22 °C	Mild.	17 km/h	↘	79%	N/A	N/A
09:00		23 °C	Mild.	19 km/h	↑	82%	N/A	N/A
12:00		23 °C	Mild.	22 km/h	↑	87%	N/A	N/A
15:00		24 °C	Mild.	13 km/h	↖	86%	N/A	N/A
18:00		24 °C	Mild.	13 km/h	↗	89%	N/A	N/A
21:00		22 °C	Mild.	17 km/h	↗	98%	N/A	N/A

Weather by CustomWeather, © 2019

1 Jan | 2 Jan | 3 Jan | 4 Jan | 5 Jan | 6 Jan | 7 Jan | 8 Jan | 9 Jan | 10 Jan | 11 Jan | 12 Jan | 13 Jan | 14 Jan | 15 Jan | 16 Jan | 17 Jan | 18 Jan | 19 Jan | 20 Jan | 21 Jan | 22 Jan | 23 Jan | 24 Jan | 25 Jan | 26 Jan | 27 Jan | 28 Jan | 29 Jan | 30 Jan | 31 Jan

Figura 4.7

El dataset extraído de timeanddate contiene información climática cada tres horas. Se indica para cada entrada del dataset la fecha y hora de la medición y la temperatura registrada en grados Celsius. La creación del dataset fue hecha a mano, recorriendo la información histórica de la ciudad de Newcastle en el sitio para los 365 días del año 2013. Cuando en un día no se observaba registro para las horas de interés: $0, 3, 6, 9, 12, 15, 18$ y 21 entonces la información de esa hora era completada copiando el valor de la temperatura del horario del mismo día más cercano.

4.2.2. Análisis de correlación entre temperatura y consumo

Si la temperatura influye en el consumo de energía es de esperar observar una relación entre la temperatura y el consumo promedio de energía de los clientes. En la Figura 4.8 se observa la variación de la temperatura y del consumo promedio de energía para los 539 clientes finales, durante el año 2013. Se observa una correlación negativa, es decir, en el invierno, cuando se dan las menores temperaturas, el consumo promedio de energía para los clientes aumenta. Otra manera de observar la correlación entre temperatura y consumo es consultando la Figura 4.9. La curva representa el consumo promedio de estos 539 clientes para cada temperatura que se dió en el año 2013. Se observa que el consumo promedio ronda alrededor de 0.2 KWh entre los 6 y 30 grados Celsius, para luego aumentar,

superando los 0.3 KWh en varias ocasiones para temperatura mayores a los 35 grados.

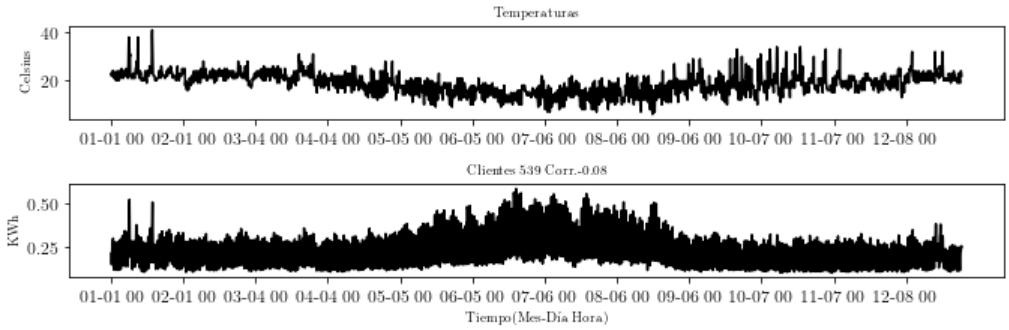


Figura 4.8: La gráfica de arriba muestra la temperatura cada 3 horas durante el año 2013 en Newcastle, la de abajo el consumo medio para los 539 clientes del dataset final para el mismo periodo de tiempo.

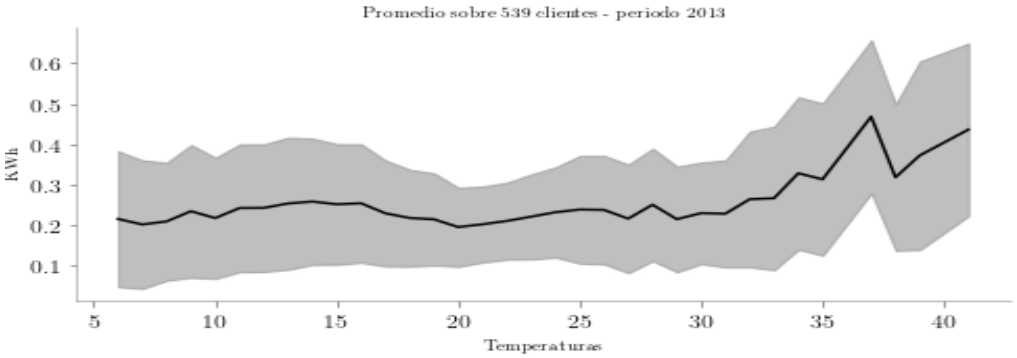


Figura 4.9: La gráfica muestra el consumo medio de temperatura de 539 clientes para cada temperatura registrada durante el año 2013.

Otro punto de interés es saber como varía esta correlación si los clientes se agrupasen en clusters bajo algún criterio. El interés está en obtener mejores valores de correlación para los clusters en comparación con el consumo promedio de todos los clientes. La Figura 4.10 muestra la variación de la temperatura y el consumo promedio de los 539 clientes durante el mes de Julio del año 2013. La correlación en este mes es de 0.11, la cual es una correlación baja. Se puede pensar que dentro del representativo de clientes elegidos, algunos pueden tener una

correlación positiva, otros negativa y otros no presentar correlación, respecto a la temperatura registrada en el mes; y si se pudiese agrupar correctamente a estos clientes, entonces la correlación de alguno de los grupos (sino todos), podría mejorar, en relación a la correlación del consumo total promedio. La Figura 4.11 muestra el consumo promedio de 6 clusters y la correlación con la temperatura para cada uno de ellos. La clasificación se realizó usando K-means, y como features para los clientes se empleó el consumo medio por hora, la tendencia y estacionalidad diaria y la correlación del consumo del cliente con la temperatura. Los clusters 2 y 3 tienen una correlación negativa. Mientras que los clusters 0,1,2 y 5 en valor absoluto obtuvieron más del doble de la correlación del consumo promedio de los 539 clientes.

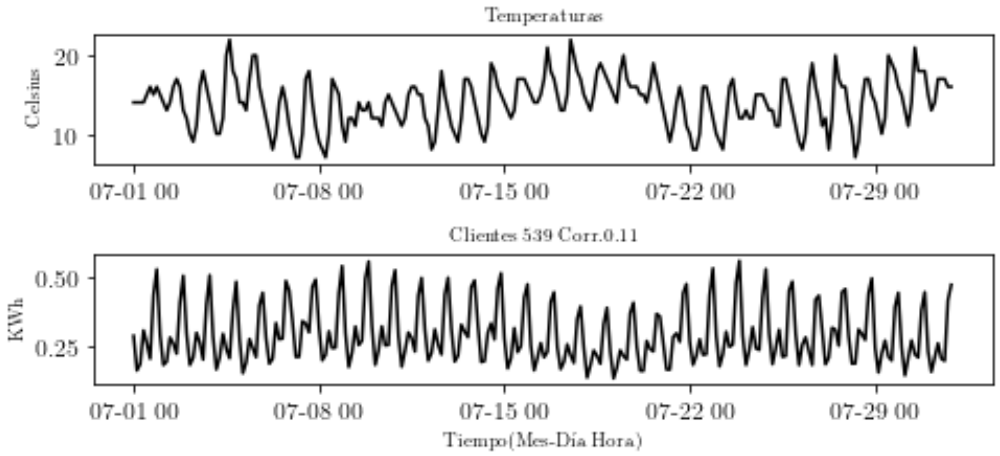


Figura 4.10: La gráfica de arriba muestra las diferentes temperaturas registradas cada 3 horas durante el mes de Julio. La gráfica de abajo muestra el consumo promedio de los 539 clientes cada 3 horas durante el mismo mes.

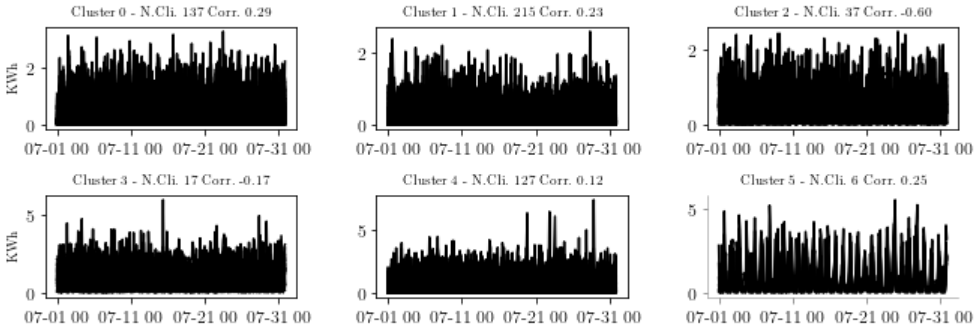


Figura 4.11: La Figura muestra las diferentes relaciones en el consumo de los distinto clusters, resultantes de aplicar K-means para agrupar a los 539 clientes según el consumo, durante el mes de Julio. Cada gráfica muestra el consumo promedio de cada cluster, indicándose arriba de cada gráfica: el número del cluster, la cantidad de clientes en el cluster (N.Cli.) y la correlación entre el consumo promedio del cluster y la temperatura para los tiempos de consumo (Corr.).

Capítulo 5

Solución propuesta

Este capítulo comienza describiendo un método general para resolver el problema de predecir el consumo de energía de los clientes de una red eléctríca basado en clustering más el empleo de la información de las temperaturas para cada tiempo de consumo. Luego se describen dos técnicas que implementan el método: TM, que aprende un valor de consumo medio para cada perfil y TC, que emplea el Algoritmo ICMC (Alg. 11) para completar los valores de una matriz de predicción. TM y TC se comparan para visualizar esquemáticamente sus similitudes y diferencias. Luego se describe la extensión propuesta en este proyecto al Algoritmo 8 de Laurinec et al. [12], para incorporar al modelo clientes recientemente ingresados a la grilla de consumo, con menos información histórica que aquellos que han sido ingresados hace más tiempo. El contenido final de este capítulo atiende las consideraciones del modelo de las monedas aplicados al modelo real.

5.1. Método propuesto

La situación que se quiere resolver es la siguiente: Se quiere predecir el consumo de energía para N clientes de una red eléctrica durante D_p días de predicción. Se conocen para cada día de predicción: datos históricos de temperaturas y de consumo de energía de los N clientes de la red. Denomínese como D_e a la cantidad de días anteriores para los cuales hay datos históricos.
El registro del consumo de energía para los N clientes, y de las temperaturas, se realiza a intervalos regulares. Sea $frec$ la cantidad de registros por día de temperatura y energía. Por ejemplo si $frec = 48$, entonces el registro de los valores se hace cada 30 minutos y si $frec = 8$ entonces el registro es realizado cada 3 horas.

El desafío a plantear es: como utilizar la información histórica para predecir los $frec$ consumos de cada cliente para cada uno de los días de predicción.
En este problema además hay que considerar la siguiente situación: de los N clientes, N_c de ellos tienen los consumos para todos los tiempos de entrenamiento. Pero N_{nc} de estos clientes tienen un cantidad menor de consumos registrados, por haber ingresado a la red mucho después que los N_c clientes anteriores. Por consiguiente se tiene menos información del comportamiento de estos N_{nc} clientes con ingreso tardío.
En base a los resultados del trabajo de Laurinec et al. [12] que propone un método de predicción basado en la clusterización de los clientes y que fue expuesto en el estado del arte (Sección 2.2 Capítulo 2), se propone un método que: **1)** agrupe en clusters a las $D_e \cdot frec$ temperaturas, **2)** agrupe en clusters a los N clientes, **3)** caracterice el comportamiento de cada cluster de clientes para cada cluster de temperaturas, en busca de emplear esta información en pos de mejorar la predicción de consumo.
El Algoritmo 4 propone una linea base para la clasificación y predicción de los datos, para cada tiempo de cada día de predicción.
El algoritmo toma como entrada la frecuencia $frec$ con que se registran los valores medidos de consumo para cada cliente cada día y la cantidad D_p de días que se quieren predecir.
En el paso 1 se inicializan las variables relevantes para el problema como la matriz de consumo de clientes $M_{N \times D_e \cdot frec}$, y el arreglo de temperaturas registradas $T_{D_e \cdot frec}$.
El bucle del paso 2 se repite para cada día de predicción.
Se observa que en los pasos del 3 al 6, el método se enfoca en la clasificación de las temperaturas (la de entrenamiento -paso 3- y las $frec$ predichas para el nuevo día -paso 4-) y de los clientes (con histórico completo -paso 5- e incompleto -paso 6-).
Los bloques de *Aprendizaje* y *Predicción* se aplican para cada tiempo *i* del día de predicción, para cada cluster de clientes.
En el paso 9, de *Aprendizaje*, es donde cada técnica debe ser capaz de caracterizar a cada cluster de clientes, según el comportamiento de consumo observado en los tiempos cuyas temperaturas registradas pertenezcan al mismo cluster que la temperatura predicha para el tiempo del día *i* y extraer la información relevante para realizar las predicciones en el paso 10 del método.
En suma el método propone para cada nuevo día hacer una clasificación de los clientes y las temperaturas. Y para cada cluster de clientes en cada tiempo *i* hacer un proceso de aprendizaje y luego de predicción basado en la correlación entre los

datos de los clusters de clientes con los de temperatura.

Algoritmo 4 ($frec, D_p$)

```
1: //Inicialización de variables
2: for iter ← 1; iter ≤ D_p; iter = iter + 1 do
3:    Clasificar_temperaturas()
4:    Predicción_y_etiquetado_de_las_temperaturas_para_el_nuevo_día()
5:    Clasificar_clientes()
6:    Etiquetado_de_clientes_con_histórico_incompleto()
7:    for i ∈ frec do
8:       for cada cluster de clientes do
9:          Aprendizaje()
10:         Predicción()
11:      end for
12:   end for
13: end for
```

5.2. Técnica de la Media (TM)

TM es una técnica que en el bloque de *Aprendizaje*, para cada cluster *cl*, aprende un valor de consumo medio, c_{hc}, para todos los clientes con histórico completo, y otro valor de consumo medio, c_{hi}, para todos los clientes con histórico incompleto. Por consiguiente en el bloque de *Predicción*, predice que el consumo de los clientes pertenecientes al cluster *cl* con histórico completo es c_{hc} y el consumo de los clientes con histórico incompleto es c_{hi}.
La elección del subconjunto de los datos históricos, para cada tipo de cliente, se realiza de la siguiente manera: En el bloque *Predicción y etiquetado de las temperaturas para el nuevo día*: se realiza la predicción y etiquetado, en uno de los clusters, de las $frec$ temperaturas para el nuevo día y en particular para la temperatura del tiempo *i*. Luego en el bloque de *Aprendizaje*: Dado el tiempo *i* del día que se está queriendo predecir, y dada la temperatura T_x predicha para ese tiempo, la técnica se queda con el subconjunto de las columnas, de la matriz de entrenamiento $M_{N \times D_e \cdot frec}[cl]$, asociadas al tiempo *i*, cuya temperatura registrada pertenezca al mismo cluster de temperaturas que T_x. Es decir si se está prediciendo el consumo para las 3 de la mañana y la temperatura predicha para esta hora es $25°C$, las columnas de entrenamiento elegidas serán aquellas asociadas a las 3 de la mañana cuya temperatura a esa hora también pertenezca al mismo cluster

que la temperatura predicha. Mientras que el subconjunto de filas corresponde a aquellas que tengan el consumo de los clientes pertenecientes al cluster *cl* para el cual se está realizando el proceso de aprendizaje.
En el bloque de *Aprendizaje*, se hace una separación de la submatriz resultante en dos matrices: M^{hc}, la matriz con la información extraída para los clientes con histórico completo y M^{hi}, la matriz con la información extraída para los clientes con histórico incompleto. A partir de M^{hc}, se calcula c_{hc}, el consumo medio de sus valores, y a partir de M^{hi}, se calcula c_{hi}, el consumo medio de sus valores.
En el bloque de *Predicción*, TM predice que todos los clientes pertenecientes al cluster *cl*, con histórico completo consumirán el valor c_{hc}, y todos los clientes con histórico incompleto consumirán el valor c_{hi}, para el tiempo i que se está actualmente prediciendo.

5.3. Técnica de Completado (TC)

TC en el bloque de *Aprendizaje* aplica el algoritmo ICMC (Alg. 11) de completado de matrices. ICMC recibe una matriz incompleta MI_{cl} para cada cluster *cl* y devuelve una matriz completada MC_{cl}. La matriz MI_{cl}, representa un subconjunto de datos históricos que se extraen, para cada cluster, de la misma manera que la explicada para el bloque de *Aprendizaje* en TM. Luego de la extracción de estos datos históricos se agrega a MI_{cl} una columna sin valores de consumo al final, que representan los valores de consumo que se quieren predecir.
Al igual que ocurrió para las técnicas TCM y $TCMCT$, en el modelo de las monedas, es necesario completar al menos una entrada de la última columna agregada antes de ejecutar el algoritmo ICMC. Pues ICMC requiere que en la matriz de visibilidad, asociada a la matriz incompleta MI_{cl}, cada columna (resp. fila) tenga al menos visibilidad a una fila (resp. columna). En donde visible significa que hay un valor observado para la entrada de dicha fila y columna en la matriz. Y no visible significa que no hay un valor observado para dicha fila y columna. En este caso se completa el 1 % de las entradas de la nueva columna, eligiendo las entradas al azar. Si la entrada i es elegida para completar, el valor se calcula como el consumo medio de los datos históricos del cliente i que hay en $MI_{cl}[i]$.
En el bloque de *Predicción*, el consumo predicho c_i de cada cliente i perteneciente al cluster *cl* es igual al valor de la última columna en la fila i de la matriz completada: $c_i = MC_{cl}[i, -1]$, con -1 refiriendo a la ubicación de la última columna en la matriz. Recordar que la última columna agregada a la matriz incompleta MI_{cl} representa los valores no observados de predicción, por lo que la última columna

en la matriz completada MC_{cl} representa los valores predichos.
En el Algoritmo 9 se detallan todos los aspectos de la implementación.

5.4. Comparativa de las técnicas

En la descripción de las técnicas TM y TC se hizo énfasis en los bloques de *Aprendizaje* y *Predicción*, pues allí están las diferencias entre estas. Los pasos de clasificación y predicción de las temperaturas son los mismos y se detallan en la Sección B.2. Incluso las técnicas comparten el primer paso del aprendizaje: la extracción de un subconjunto de los datos históricos de consumo. Para ilustrar como es este proceso en las Tablas 5.1 y 5.2 se copia la explicación dada en la Sección 3.1. La Tabla 5.1 plantea el problema en forma de tablas de colores y la Tabla 5.2 describe como es el proceso de extracción de datos para cada cluster de clientes.

F=2	1	2	1	2	1	2

T=5	y_1	y_2	y_3	y_4	$\hat{y}_5$	$\hat{y}_6$

N=5 \ H=4	1	2	3	4	5	6
1	x_1^1	x_2^1	x_3^1	x_4^1	$\hat{x}_5^1$	$\hat{x}_6^1$
2	x_1^2	x_2^2	x_3^2	x_4^2	$\hat{x}_5^2$	$\hat{x}_6^2$
3	x_1^3	x_2^3	x_3^3	x_4^3	$\hat{x}_5^3$	$\hat{x}_6^3$
4	x_1^4	x_2^4	x_3^4	x_4^4	$\hat{x}_5^4$	$\hat{x}_6^4$
5	x_1^5	x_2^5	x_{53}^5	x_4^5	$\hat{x}_5^5$	$\hat{x}_6^5$
6				x_4^6	$\hat{x}_5^6$	$\hat{x}_6^6$
7			x_3^7	x_4^7	$\hat{x}_5^7$	$\hat{x}_6^7$

Tabla 5.1: En la tabla de colores se observa que hay 7 clientes, los primeros 5 con 4 valores observados (columnas 1 al 4) y los últimos dos con 1 y 2 valores observados respectivamente. La frecuencia F es de 2 mediciones por día, con la información del día 1 (col. 1 y 2) y del día 2 (col. 3 y 4), se quieren predecir los valores para el siguiente día (col. 5 y 6). Cada cliente pertenece a uno de dos perfiles de consumo, los clientes 1,2,4 y 7 pertenecen al perfil de consumo verde y los clientes 3,5 y 6 al perfil de consumo rojo. A su vez hay 4 observaciones para las temperaturas hechas durante los tiempos en que se observaron las mediciones, cada temperatura pertenece a una clase: temperatura alta (color magenta) y temperatura baja (color cyan), las temperaturas de los siguientes días predícense como: temperatura alta ($\hat{y}_5$) y temperatura baja ($\hat{y}_6$).

F=2	1	1

T=5	y_1	y_5

N=7 \ H=2	1	5
x_t^1	x_1^1	x_5^1
x_t^2	x_1^2	x_5^2
x_t^4	x_1^4	x_5^4
x_t^7		x_5^7
x_t^3	x_1^3	x_5^3
x_t^5	x_1^5	x_5^5
x_t^6		

Tabla 5.2: Para el caso del tiempo 5, cuya temperatura $\hat{y}_5$ se predijo como alta, los valores de consumo para cada perfil se predicen utilizando como datos históricos aquellos que coincidan con la clase de temperatura de $\hat{y}_5$ para el tiempo 1 (de los F tiempos registrados cada día).

Momento de ilustrar las diferencias. Sea la atención centrada en el cluster *verde*.
TM aprende, para los clientes de cada cluster *cl*, un solo valor de consumo para los clientes con histórico completo c_{hc} y otro para los clientes con histórico incompleto c_{hi}, :

$$c_{hc} = media\left[\begin{array}{cccccc} x_1^1 & x_5^1 & x_1^2 & x_5^2 & x_1^4 & x_5^4 \end{array}\right]$$

$$c_{hi} = media\left[\begin{array}{c} x_5^7 \end{array}\right]$$

La predicción en **TM** es, para un cluster *cl*: c_{hc} para los clientes con histórico completos y c_{hi} para los clientes con histórico incompleto.
TC, aplica el algoritmo de completado de matrices ICMC (Alg. 11), se observa que a la matriz se le agrega una nueva columna y se predice al menos una de sus entradas, antes de aplicar el algoritmo ICMC:

$$ICMC\left(\left[\begin{array}{ccc} x_1^1 & x_5^1 & ? \\ x_1^2 & x_5^2 & \frac{x_1^2+x_5^2}{2} \\ x_1^4 & x_5^4 & ? \\ ? & x_5^7 & ? \end{array}\right], K\right) = \left[\begin{array}{ccc} x_1^1 & x_5^1 & \hat{x}_6^1 \\ x_1^2 & x_5^2 & \frac{x_1^2+x_5^2}{2} \\ x_1^4 & x_5^4 & \hat{x}_6^5 \\ \hat{x}_1^7 & \hat{x}_5^7 & \hat{x}_6^7 \end{array}\right]$$

La predicción en **TC** es la columna final, resultante de aplicar el algoritmo ICMC a la matriz incompleta, el parámetro de entrada K es el rango asociado a la matriz.

5.5. Extensión al método del estado del arte

El método de predicción utilizando clustering propuesto por Laurinec et al. [12], estudiado en el estado del arte, no considera la posibilidad de que los clientes tengan un histórico de observaciones incompleto. Para poder evaluar las técnicas propuestas con las del estado del arte se realiza una extensión al método.
En resumidos pasos el método sin extensión, para cada día de predicción, transforma las series de tiempo de consumo aplicando la normalización de los datos (Ec. 5.1). En donde x_i es el consumo para el tiempo i, μ es la media muestral y σ es la desviación estándar muestral. Guardando para cada serie: la media μ y la desviación σ:

$$z_i = \frac{x_i - \mu}{\sigma} \tag{5.1}$$

Posteriormente se computa una representación para cada cliente, siendo luego agrupados alrededor de los K centroides resultantes de la aplicación de K-means sobre las representaciones. Las técnicas de predicción predicen los valores futuros para estos K centroides.
Finalmente para obtener las predicciones de los N clientes a partir de los K centroides se realiza el paso de desnormalizacion (Ec. 5.2):

$$x_i = z_i\sigma + \mu, \tag{5.2}$$

siendo z_i la predicción hecha para el centroide, y x_i la predicción para el cliente con media μ y desviación σ, perteneciente al cluster del centroide.
En el Algoritmo 5 se detalla la extensión del método.
En la extensión al método, los clientes con histórico incompleto son clasificados en alguno de los K clusters resultantes de la aplicación de K-means, para las series de tiempo de los clientes con histórico completo. La clasificación se realiza de la siguiente manera: sea ts_i la serie de tiempo para el cliente con histórico incompleto i. ts_i no tiene valores observados hasta un tiempo t_{obs} en donde se comienzan a registrar los valores de consumo (paso 12).
Se obtienen los valores de ts_i y de los K centroides para los tiempos a partir de t_{obs} y se calcula la distancia euclídea entre la serie de tiempo a partir de t_{obs}: $ts_{i|t_{obs}}$ y los valores de los centroides K a partir de este tiempo. El cliente i es etiquetado

al cluster que dé el mínimo valor en la distancia (paso 13).
Para la predicción del consumo del cliente i se obtienen las predicciones del cluster k al cual fue etiquetado y se desnormalizan los valores usando la Ecuacion 5.2. En este caso μ y σ, son la media y desviación estándar en el consumo del cliente i (paso 14).

Algoritmo 5 ExtensionDelMetodo($N, D_e, D_p, frec$)

1: $iter \leftarrow 0$
2: **while** $iter < D_p * frec$ **do**
3: //Clientes con histórico completo.
4: Crear una serie de tiempo para cada cliente con histórico completo del tamaño $D_e * frec$ (de tres semanas de entrenamiento).
5: Normalización de cada serie de tiempo aplicando z-score (guardando la media y desviación estándar de cada serie de tiempo).
6: Computar una representación para cada serie de tiempo.
7: Clustering usando K-means.
8: Extracción de K centroides.
9: Usar los K centroides como datos de entrenamiento en todos los métodos de entrenamiento.
10: Desnormalizar los K pronósticos usando las medias y desviaciones guardadas para cada usuario para producir las predicciones para los clientes con histórico completo.
11: //Clientes con histórico incompleto.
12: Crear una serie de tiempo para cada cliente con histórico incompleto del tamaño $D_e * frec$ (de tres semanas de entrenamiento). La serie de tiempo no tendrá valores observados desde el tiempo 1 hasta $t_{obs}-1$ y tendrá valores observados desde t_{obs} hasta $D_e * frec$.
13: Para cada cliente con histórico incompleto, determinar a cual de los K centroides se encuentra a menor distancia euclídea. Usando para el cálculo de la distancia los valores del centroide y del cliente para los tiempos con observaciones de consumo del cliente, desde t_{obs} hasta $D_e * frec$. Etiquetar al cliente al cluster más cercano.
14: Para cada cliente con histórico incompleto obtener los pronósticos hechos por los métodos para el cluster asociado al cliente. Obtener la media μ y la desviación estándar σ del cliente. Desnormalizar los pronósticos usando la media μ y la desviación estándar σ, para producir la predicción para el cliente.
15: $iter = iter + frec$

16: **end while**

5.6. Consideraciones del modelo de las monedas aplicados al modelo real

Los bloques del Algoritmo 4, para el modelo real, tienen su parangón con los bloques del Algoritmo 6, del modelo teórico de las monedas. En ambos casos para cada nuevo día:

- Se predicen las temperaturas del siguiente día.
- Se clasifican los clientes (resp. monedas) con histórico completo.
- Se clasifican los clientes (resp. monedas) con histórico incompleto.
- Para cada tiempo de predicción de cada nuevo día para cada cluster:
    - Hay un bloque de *Aprendizaje*, en donde las técnicas para los clientes y para las monedas extraen la información relevante de la matriz de entrenamiento, con desiciones similares para extraer las filas y columnas, según cada cluster y según la temperatura predicha.
    - Se predice para cada cliente (moneda) de cada cluster, en base a la información obtenida en el bloque de *Aprendizaje*.

La Técnica de la Media (TM) copia las mismas ideas de aprendizaje y predicción que la técnica TKMCT, con la salvedad de que en lugar de aprender probabilidades para un conjunto discreto de valores (*cara* y *cruz*), TM aprende el valor medio de consumo para cada cluster.
La Técnica de Completado (TC) copia las mismas ideas de aprendizaje y predicción que la técnica TCMCT.

Capítulo 6

Resultados

Se ha llegado al punto de la lectura en la que se describen y muestran los resultados de este trabajo. El capítulo consta de tres partes. En la primera se define el subconjunto del dataset sobre el cual las diferentes técnicas entrenarán y realizarán las predicciones de consumo de energía. La segunda parte está dedicada a determinar los mejores parámetros para las técnicas propuestas en el Capítulo 5. Por último se comparan las técnicas propuestas con las descritas en el trabajo de Laurinec et al. [12] y que fueron explicadas en el estado del arte (Secc. 2.2.2 Cap. 2).

6.1. Dataset de entrenamiento y predicción

En el Capítulo 4 se especificó que el dataset de trabajo consta del consumo de energía, registrado cada tres horas durante todo el año 2013, de clientes nucleados en Newcastle, Australia. La cantidad final de clientes, luego del preprocesamiento realizado, es de 539 usuarios. Con este panorama, los siguientes periodos de tiempo son los elegidos para entrenar los parámetros de las técnicas propuestas (Capítulo 5) y para comparar estas técnicas con las del estado del arte:

- Periodo de entrenamiento de parámetros: del 1° al 28 de Marzo.
- Periodo de comparación entre técnicas: del 1° al 28 de Abril.

A su vez, de los 539 usuarios, a 450 de ellos se les conocerá el consumo de energía para todos los tiempos de entrenamiento y a 89 de estos clientes solo se les conocerá el consumo de energía del último día de entrenamiento. Estos 89 clientes

conforman el conjunto de clientes con histórico incompleto, por haber ingresado en forma tardía a la grilla de consumo.

6.2. Entrenamiento

El objetivo del entrenamiento es encontrar un conjunto de parámetros para las dos técnicas propuestas que minimicen el RMSE entre las predicciones de cada técnica y los valores reales de consumo. Ambas técnicas comparten los parámetros *kmIter* y $clTamMax$, que determinan la cantidad de iteraciones durante la reclasifiación de los clientes usando K-means y la cantidad mínima de clientes que un cluster debe tener para ser reclasificado en algunas de las *kmIter* iteraciones, respectivamente. La técnica TC posee el parámetro *rango* que especifica cual es el rango de la matriz a completar aplicando el algoritmo ICMC.
El conjunto de valores a probar durante el entrenamiento es el siguiente:

- técnica ∈ {TM, TC}
- $kmIter \in \{1, 3, 5, 10\}$
- $clTamMax \in \{20, 50, 100\}$
- $rango \in \{1, 3, 5, 10, 20, 30\}$ (solo para TC)

El periodo de entrenamiento va del 1° al 28 de marzo del 2013. Los primeros 21 días son elegidos como datos de entrada para cada técnica y los siguientes 7 días (del 22 al 28) son elegidos para predicción.
Para cada juego de parámetros: se predice el consumo de energía para los días comprendidos del 22 al 28. Cada técnica repite 5 veces la predicción, para estos días, obteniéndose un resultado promedio para las métricas RMSE, MaxErr_{RMSE}, MAE, MaxErr_{MAE}.
En la Tabla 6.1 se ven los mejores 5 resultados ordenados por menor RMSE.

ID	técnica	kmIter	clTamMax	rango	RMSE	MaxErr_{RMSE}	MAE	MaxErr_{MAE}
1	TM	10	20	-	0.0904 ± 0	0.0030 ± 0	0.1435 ± 0	0.0003 ± 0
2	TM	5	20	-	0.0909 ± 0	0.0030 ± 0	0.1453 ± 0	0.0003 ± 0
3	TM	1	20	-	0.0935 ± 0	0.0028 ± 0	0.1644 ± 0	0.0003 ± 0
4	TM	1	50	-	0.0935 ± 0	0.0028 ± 0	0.1644 ± 0	0.0003 ± 0
5	TM	1	100	-	0.0935 ± 0	0.0028 ± 0	0.1644 ± 0	0.0003 ± 0

Tabla 6.1: Mejores 5 resultados para la predicción del consumo de energía durante 7 días del mes de marzo. Predicciones hechas del 22 de Marzo al 28 de Marzo.

Como se puede observar la técnica TM, fue la que obtuvo los mejores resultados. La desviación estándar igual a 0 se debe a que siempre aprende los mismos valores medios con los cuales hacer las predicciones para cada cluster.
Para resumir cuales fueron los mejores parámetros de cada una de las técnicas se ejecuta, para cada una, el algoritmo *flashlight* para generar una *explanation table*. En la Tabla 6.2 se ve una *explanation table*, resumiendo los principales parámetros que se deben reunir para que el RMSE promedio de cada técnica, sobre las 5 simulaciones, sea menor a 0.15.
Para TM el resumen es dilapidario, ver los resultados en la Tabla de arriba 6.2, todos los parámetros obtuvieron un valor de RMSE menor a 0.15.
Los resultados para TC se muestran en la Tabla de abajo 6.2. Observando la fila con $ID = 1$ se tiene que un porcentaje muy bajo de casos obtuvieron un RMSE menor a 0.15, solamente un 4 % de los casos. La fila con $ID = 2$ da una pista de en que situación el modelo TC se encuentra por debajo del umbral. El patrón para esta fila es (kmIter $= *$, clTamMax $= *$, rango $= 30$), con un 25 % de casos con RMSE menor a 0.15. Lo que indica que TC obtuvo todos los resultados positivos solo para el rango más alto evaluado, $rango = 30$. Las filas con $ID = 3$ y 4 indican que para ninguno de estos patrones el resultado fue positivo:

- $(kmIter = 3, clTamMax = *, rango = 30)$
- $(kmIter = 5, clTamMax = *, rango = 30)$

O sea que los 3 casos positivos deben estar repartidos entre los siguientes patrones

- $(kmIter = 1, clTamMax = *, rango = 30)$
- $(kmIter = 10, clTamMax = *, rango = 30)$

Haciendo una observación directa de la tabla de resultados generada se concluye que todos los casos positivos vienen del patrón con $kmIter = 1$.
Los parámetros elegidos para cada técnica se muestran en la Tabla 6.3.
Para TM simplemente se eligió aquella tupla, del total, con menor RMSE.
Para TC la tupla de parámetros

$$(kmIter = 1, clTamMax = 20, rango = 30)$$

que pertenece al patrón $p = (*, *, 30)$, fue la que menor RMSE obtuvo comparando los resultados de todas las tuplas que coincidieron con el patrón p.

Técnica TM					
ID	**kmIter**	**clTamMax**	**rango**	**Tot.**	**Fracc.**
1	*	*	-	12	1
RMSE: 0.000199					
Técnica TC					
ID	**kmIter**	**clTamMax**	**rango**	**Tot.**	**Fracc.**
1	*	*	*	72	0.0416
2	*	*	30	12	0.25
3	3	*	*	18	0
4	5	*	30	3	0
RMSE: 0.0278351					

Tabla 6.2: Resumen del algoritmo *flashlight* para la *explanation table* para cada técnica. El RMSE es entre la columna binaria, que contiene 1 si $RMSE_{tecnica} < \epsilon = 0.15$ o 0 en caso contrario, y la estimación de dicha columna hecha por el algoritmo. Las entradas (i, j) de la tabla que contienen un guión (-) indican que la técnica i no hace uso del parámetro j.

Técnica	kmIter	clTamMax	rango
TM	10	20	-
TC	1	20	30

Tabla 6.3: Las entradas (i, j) de la tabla que contienen un guión (-) indican que la técnica i no hace uso del parámetros j.

6.3. Comparación entre técnicas

En esta sección se comparan las técnicas propuestas (Capítulo 5) contra las del estado del arte Laurinec et al. [12]. El periodo de tiempo sobre el cual se aplican los algoritmos de cada técnica es el comprendido desde 1° al 28 de Abril. Los primeros 21 días son elegidos como datos de entrenamiento para cada técnica. Los días elegidos para predecir son los comprendidos entre el 22 y el 28. Las técnicas son las siguientes:

- Técnicas propuestas
  - Técnica del Promedio (TM)

    - Técnica de Completado (TC)

- Estado del arte: empleando clustering
    - Random Forest (RF)
    - Triple Exponential Smooth (ES)
    - Multiple Linear Regression (MLR)
    - Seasonal Naive (SNAIVE)
- Estado del arte: benchmark sin clustering
    - Random Forest (RF BM)
    - Seasonal Naive (SNAIVE BM)

Para cada técnica la predicción del consumo de energía para los días comprendidos del 22 al 28 se repiten 10 veces, obteniéndose un resultado promedio para las métricas RMSE, MaxErr$_{RMSE}$, MAE, MaxErr$_{MAE}$.
En la Tabla 6.4 se ven los resultados para cada técnica.
Comparando solo las técnicas propuestas se observa que TM fue la que mejor resultados obtuvo en RMSE y MAE. En RMSE, TM obtuvo un resultado un 47 % menor, respecto al de TC.
TC performó peor que todas las técnicas del estado del arte.
Comparando a TM con las técnicas del estado del arte, se tiene que performó mejor que las técnicas que no hacen uso de clustering, obteniendo un resultado en RMSE un 13 % menor respecto a la mejor de las técnicas sin clustering (RF BM). Sin embargo no fue mejor que las técnicas que sí emplean clustering. Obteniendo un resultado en RMSE un 5.4 % mayor respecto a la mejor de las técnicas (ES).
Observando las tablas con los resultados separados por clientes con histórico completo (450) e histórico incompleto (89), se observa que el peor rendimiento de TM fue en la predicción de consumo para los clientes con histórico incompleto (ver Tabla de abajo 6.4). En donde el resultado en RMSE fue un 18 % mayor, respecto a la técnica SNAIVE, que fue la que mejor resultados obtuvo para los clientes con histórico incompleto.
Si se observan los resultados de TM para los clientes con histórico completo, se puede notar que TM se vuelve más competitiva, respecto a las técnicas del estado del arte que emplean clustering. TM en este caso obtuvo resultados en RMSE un 2.3 % mayor respecto a la mejor técnica para histórico completo (RF). Incluso TM obtuvo un valor promedio en RMSE un 1.5 % menor a la técnica SNAIVE (ver

Tabla del medio 6.4).
TM y las técnicas del estado del arte que emplean clustering, fueron mejores en RMSE en comparación a las técnicas que no emplean clustering, al momento de predecir el consumo para ambos tipos de clientes: los clientes con histórico completo y aquellos con histórico incompleto.

Resultados sobre el total de clientes: 539				
Técnica	**RMSE**	**MáxError**$_{RMSE}$	**MAE**	**MáxError**$_{MAE}$
TM	0.0743 ± 0	0.0009 ± 0	0.1456 ± 0	**0.0001 ± 0**
TC	0.1390 ± 0.0006	0.0011 ± 0.00003	0.1948 ± 0.0012	0.0001 ± 0
RF	0.0724 ± 6,4468$e-04$	0.0011 ± 1.3160$e-04$	0.1400 ± 4.1948$e-04$	0.0002 ± 1.9236$e-05$
ES	**0.0705 ± 4.5322e-04**	0.0011 ± 2.2741$e-05$	0.1429 ± 5.1130$e-04$	0.0002 ± 3.2234$e-06$
MLR	0.0718 ± 5.1339$e-04$	0.0011 ± 3.5505$e-05$	**0.1397 ± 3.6886e-04**	0.0002 ± 4.9543$e-06$
SNAIVE	0.0727 ± 4.9711$e-04$	0.0011 ± 2.6294$e-05$	0.1415 ± 3.5243$e-04$	0.0002 ± 3.3217$e-06$
RF BM	0.0853 ± 4.8309$e-18$	**0.0008 ± 9.5414e-20**	0.1486 ± 2.4426$e-17$	0.0002 ± 1.8870$e-20$
SNAIVE BM	0.1119 ± 0	0.0014 ± 1.9082$e-19$	0.1567 ± 0	0.0002 ± 4.7707$e-20$
450 clientes con histórico completo				
Técnica	**RMSE**	**MáxError**$_{RMSE}$	**MAE**	**MáxError**$_{MAE}$
TM	0.0693 ± 0	0.0007 ± 0	0.1392 ± 0	**0.00009 ± 0**
TC	0.1227 ± 0.0004	0.0009 ± 0.00001	0.1829 ± 0.0012	0.00019 ± 0
RF	**0.0677 ± 1.192e-04**	0.0011 ± 2.321$e-05$	0.1333 ± 2.433$e-04$	2.14$e-04$ ± 2.161$e-06$
ES	0.0679 ± 2.20$e-04$	0.0010 ± 1.546$e-05$	0.1383 ± 3.923$e-04$	2.08$e-04$ ± 1.479$e-06$
MLR	0.0680 ± 1.80$e-04$	0.0011 ± 2.4605$e-05$	**0.1332 ± 1.678e-04**	2.12$e-04$ ± 2.302$e-06$
SNAIVE	0.0703 ± 2.86$e-04$	0.0011 ± 2.1536$e-05$	0.1368 ± 1.8224$e-04$	2.16$e-04$ ± 1.9838$e-06$
RF BM	0.0764 ± 0	**0.0006 ± 1.1428e-19**	0.1371 ± 2.9256$e-17$	1.57$e-04$ ± 0
SNAIVE BM	0.1128 ± 0	0.0013 ± 2.2856$e-19$	0.1566 ± 0	2.33$e-04$ ± 5.7142$e-20$
89 clientes con histórico incompleto				
Técnica	**RMSE**	**MáxError**$_{RMSE}$	**MAE**	**MáxError**$_{MAE}$
TM	0.0997 ± 0	0.0016 ± 0	0.1779 ± 0	0.0005 ± 0
TC	0.2212 ± 0.0014	0.0023 ± 0.00007	0.2547 ± 0.0010	0.0006 ± 0.00001
RF	0.0960 ± 0.0029	0.0014 ± 0.00068	0.1738 ± 0.0013	0.0005 ± 0.0001
ES	0.0855 ± 0.0016	**0.0012 ± 0.00006**	0.1663 ± 0.0011	**0.0004 ± 0.00001**
MLR	0.0906 ± 0.0021	0.0012 ± 0.00009	0.1728 ± 0.0013	0.0004 ± 0.00001
SNAIVE	**0.0848 ± 0.0015**	0.0012 ± 0.0006	**0.1654 ± 0.0012**	0.0004 ± 0.00005
RF BM	0.1304 ± 2.9256$e-17$	0.0018 ± 0	0.2065 ± 0	0.0006 ± 1.1428$e-19$
SNAIVE BM	0.1073 ± 0	0.0018 ± 0	0.1572 ± 0	0.0006 ± 0

Tabla 6.4: Tablas con los resultados promedios de cada técnica sobre 10 ejecuciones. La tabla de arriba muestra los resultados promedios generales. La tabla del medio los resultados solo considerando a los 450 clientes con el histórico completo. La tabla de abajo los resultados solo considerando a los 89 clientes con histórico incompleto.

Visto que los mejores resultados fueron obtenidos por las técnicas del estado del arte, resulta de interés saber cuanta incidencia ha tenido en el resultado la clusterización de los clientes. Recordar que el algoritmo general del estado del arte (Algoritmo 8), aplica a las series de tiempo la función z-score (Ec. 2.16), luego para cada serie de tiempo obtiene una representación con componentes estacionales diarios y semanales y sobre esta representación se aplica K-means. Mien-

tras que el algoritmo propuesto en este trabajo (Algoritmo 4), en los pasos del 13 al 24, agrega como features componentes estacionales y de tendencia diaria y semanal, y la correlación de cada serie con la temperatura. Puede ocurrir que la técnica de clustering propuesta sea buena, pero las técnicas de predicción no logren capturar la información que las técnicas del estado del arte sí, o al contrario que la técnica de clustering propuesta no sea tan buena como la del estado del arte incidiendo en el mal resultado de las técnicas de predicción.
Los resultados de la Tabla 6.5 muestran la ejecución de las técnicas del estado del arte utilizando la técnica de clustering propuesta en este trabajo. ES vuelve a ser la mejor técnica al igual que en los resultados anteriores (Tabla 6.4). Incluso mejorando en RMSE y MAE un 0.25 %, en comparación con los anteriores. En general todas las técnicas mejoraron en RMSE y MAE respecto a los resultados obtenidos en la Tabla 6.4. Con MLR obteniendo la mayor mejoría en RMSE 1.08 % y en MAE 0.5 %.

Resultados sobre el total de clientes: 539				
Técnica	**RMSE**	**MáxError$_{RMSE}$**	**MAE**	**MáxError$_{MAE}$**
ES CL	**0.070320 ± 0.000184**	**0.001081 ± 0.000013**	0.142540 ± 0.000328	**0.000249 ± 0.000001**
MLR CL	0.071019 ± 0.000150	0.001133 ± 0.000021	**0.139001 ± 0.000140**	0.000256 ± 0.000002
RF CL	0.071632 ± 0.000161	0.001179 ± 0.000019	0.139380 ± 0.000203	0.000263 ± 0.000002
SNAIVE CL	0.072228 ± 0.000238	0.001150 ± 0.000018	0.141327 ± 0.000152	0.000255 ± 0.000002
450 clientes con histórico completo				
Técnica	**RMSE**	**MáxError$_{RMSE}$**	**MAE**	**MáxError$_{MAE}$**
ES CL	**0.066777 ± 0**	**0.001168 ± 0**	0.135601 ± 2.925e − 17	**0.000215 ± 0**
MLR CL	0.066805 ± 1.462e − 17	0.001224 ± 0	0.133191 ± 2.925e − 17	0.000220 ± 2.857e − 20
RF CL	0.067052 ± 1.462e − 17	0.001224 ± 0	**0.133147 ± 0**	0.000220 ± 2.857e − 20
SNAIVE CL	0.070008 ± 1.462e − 17	0.001226 ± 2.285e − 19	0.137109 ± 2.925e − 17	0.000221 ± 2.857e − 20
89 clientes con histórico incompleto				
Técnica	**RMSE**	**MáxError$_{RMSE}$**	**MAE**	**MáxError$_{MAE}$**
ES CL	0.082352 ± 0	0.001037 ± 0	**0.163851 ± 2.925e-17**	0.000456 ± 1.142e − 19
MLR CL	0.085866 ± 1.462e − 17	0.001141 ± 2.285e − 19	0.168086 ± 2.925e − 17	0.000479 ± 0
RF CL	0.091060 ± 0	0.001323 ± 0	0.169969 ± 0	0.000515 ± 1.142e − 19
SNAIVE CL	**0.081780 ± 0**	**0.001029 ± 2.285e-19**	0.163992 ± 2.925e − 17	**0.000454 ± 0**

Tabla 6.5

Capítulo 7

Conclusiones

En la primera parte de este trabajo se formalizó *el modelo de las monedas* Sección 3.3. Para el cual se realizaron simulaciones para entender como afecta a una técnica de predicción el empleo de clustering y la incorporación de factores que inciden en los resultados, como la temperatura.
Se evaluaron diferentes técnicas para solucionar el problema de predecir el resultado de salir cara o no para cada moneda, durante D_p días de predicción, Secciones 3.3.1.3 y 3.3.2.3.
Se identificó en el espacio de valores de los parámetros del modelo, aquellos juegos en los cuales las técnicas que emplean clustering (TCP, TCPCT, TKM, TKMCT) performaron mucho mejor que aquellas cuya probabilidad era aprendida sobre los resultados del total de monedas (Modelos TDP, TDPCT).
Las técnicas frecuentistas que emplean clustering (TCP, TCPCT, TKM, TKMCT) obtuvieron valores en RMSE y F1 similares a las técnicas que no emplean clustering (TIP, TIPCT). Con la ventaja que para las técnicas frecuentistas que emplean clustering solo es necesario aprender 2 probabilidades, durante la etapa de entrenamiento, una para cada cluster. Cantidad menor que las N probabilidades que deben aprender las técnicas frecuentistas que utilizan solo el histórico individual de cada moneda para el aprendizaje de las probabilidades.
En la Sección 3.3.2.4, en la comparación entre las técnicas TCP vs. TCPCT, se advierte que la incorporación de la temperatura al modelo *de las monedas* también acarrea errores, que en algunos casos pueden ser mayores al error de las técnicas que ignoran esta información.
En cambio, en la prueba 3 de la Sección 3.3.2.5, se muestra que para las predicciones durante 10 días, en promedio las técnicas que emplean la información de las temperaturas pueden performar mucho mejor que aquellas que ingoran esta

información.
En la segunda parte de este trabajo se formalizó el modelo para predecir el consumo de energía empleando técnicas de clustering e información de las temperaturas registradas para cada tiempo de medición del consumo, para clientes antiguos de la grilla de consumo y para clientes recientemente ingresados (Sección 3.1).
Se propusieron 2 técnicas para dicho modelo: TM (Sección 5.2), una técnica que para cada cluster aprende un valor de consumo medio para los clientes con histórico completo c_{hc} y otro para los clientes con histórico incompleto y TC (Sección 5.3), técnica que emplea clustering más *matrix completion*. En el Apéndice E se brindan los detalles de la implementación del algoritmo *matrix completion*.
Se realizó una extensión del método de Laurinec et al. [12], para que las diferentes técnicas propuestas pudieran predecir para clientes con histórico incompleto, es decir recientemente ingresados a la grilla.
De las técnicas propuestas la mejor fue TM, que sin embargo performó peor en cuanto a métrica RMSE respecto a la mejor técnica del estado del arte, ES, obteniendo TM un valor en RMSE un 5.4 % mayor del valor obtenido por ES.
El error mayor estuvo en la predicción de los clientes con histórico incompleto, con un RMSE de 0.0997 para TM, mayor al RMSE de TM para los clientes con histórico completo, valiendo en este caso 0.0693, ver Tabla 6.4.
Como resultado de la extensión del método de Laurinec et al. (Sección 5.5), se observa en la Tabla 6.4 que las técnicas que emplean clustering obtuvieron mejor RMSE para los clientes con histórico incompleto, respecto a las técnicas que no emplean clustering. Este resultado debe ser remarcado ya que el empleo de clustering (habiendo clientes con histórico incompleto) mejora la predicción en comparación a usar solo los pocos registros de información que este tipo de cliente posee.
Todas las técnicas del estado del arte lograron una mejora entre un 0.25 % a un 1.1 % en RMSE al predecir con la técnica de clustering propuesta en este trabajo, en comparación a la propuesta en el estado del arte.
Como tarea adjunta a este proyecto se implementó en Python el algoritmo *flashlight* para la técnica *explanation table*, ver Apéndice F; con la finalidad de automatizar resumenes informativos para las simulaciones hechas en *el modelo de las monedas* y en el problema del consumo de energía. La automatización de estos resúmenes coadyuvó en el análisis y toma de desiciones a lo largo del trabajo.

Apéndice A

Detalles del modelo de las monedas

A.1. Modelo de las monedas sin temperatura

A.1.1. Comportamiento esperado

A.1.1.1. Clasificación usando K-means

Una situación en donde la clasificación de TKM es perfecta, definiendo perfecta el etiquetar a cada moneda con el perfil al que realmente pertenece, es cuando la probabilidad del perfil K_1: $p_{K_1}(cara)$ es igual a 1 y la probabilidad del perfil K_2: $p_{K_2}(cara)$ es igual a 0, en este caso no hay lugar a confusión en los rasgos: si la moneda pertenece a K_1 su *feature* será $[1, \ldots, 1]$ y si pertenece a K_2 será $[0, \ldots, 0]$. En cambio si ocurriese que $p_{K_1}(cara) = p_{K_2}(cara) = 1$, el trabajo de clasificar es irrelevante ya que las monedas de ambas clases K_1 y K_2 tendrían las mismas *features*: un arreglo con D entradas igualadas a 1: $[1, \ldots, 1]$. Una situación caótica ocurre cuando $p_{K_1}(cara) = p_{K_2}(cara) = 0.5$, en este caso las *features* de las monedas serán una secuencia irregular de 0s y 1s, con aproximadamente la mitad de sus entradas igual a 1 y la otra igual a 0. La irregularidad o carencia de patrones en la secuencia viene dado por el hecho de que cada lanzamiento individual, de cada moneda, tiene igual chance de obtener como resultado cara o cruz. La matriz Ec. A.1 refleja el grado de confusión. Cada fila representa las *features* de una moneda, las filas 1 y 2 pertenecen al perfil *verde* y las filas 3 y 4 al perfil *rojo*. Dentro de cada perfil las features son secuencias con patrones diferentes, y si se mira la fila 1 (perfil *verde*) y la fila 3 (perfil *rojo*) se observa que hay coincidencia de valores en las primeras tres columnas, por lo que un algoritmo de clustering podría tender a clasificarlos dentro de un mismo perfil, a pesar de que en el ejem-

plo pertenecen a perfiles diferentes.
Es interesante notar que el Algoritmo K-means (Alg. 10), en los pasos de reasignación de los vectores de features a cada centroide (paso 9) y de recomputación de los centroides (paso 12), hace uso de la noción de distancia. La distancia Euclídea, la cual fue utilizada en los experimentos, favorecería la clusterización alrededor del mismo centroide, para la fila 1 y la fila 3. Pudiendo otra noción de distancia mejorar la correcta clasificación para este tipo de casos.
En la Figura 3.1 se resumen estas observaciones.

$$\begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \tag{A.1}$$

A.1.1.2. Comportamiento predictivo de técnicas TDP y TCP

Sean N_1 y N_2 la cantidad de monedas para los perfiles K_1 y K_2 respectivamente. Considerando un solo día de tirada en el experimento, el valor esperado de la cantidad de caras salidas para cada perfil es:

$$\#\widehat{caras}_{K_1} \approx N_1 \times p_{K_1}(cara)$$

y

$$\#\widehat{caras}_{K_2} \approx N_2 \times p_{K_2}(cara)$$

El valor esperado de monedas un día es la suma de $\#\widehat{caras}_{K_1}$ y $\#\widehat{caras}_{K_2}$:

$$\#\widehat{caras}_{K_1} + \#\widehat{caras}_{K_2} = N_1 \times p_{K_1}(cara) + N_2 \times p_{K_2}(cara)$$

En el modelo TDP, por la forma de aprender las probabilidades del Algoritmo General 1, la probabilidad aprendida esperada $\hat{p}_{TDP}$ es:

$$\hat{p}_{TDP} \approx \frac{1}{D} \sum_{i=1}^{D} \frac{\#\widehat{caras}_{K_1} + \#\widehat{caras}_{K_2}}{N} \approx \frac{D}{D} \cdot \frac{\#\widehat{caras}_{K_1} + \#\widehat{caras}_{K_2}}{N}$$

$$= \frac{\#\widehat{caras}_{K_1} + \#\widehat{caras}_{K_2}}{N} = \frac{N_1 \times p_{K_2}(cara) + N_2 \times p_{K_1}(cara)}{N}$$

$$= \frac{N_1}{N} \times p_{K_1}(cara) + \frac{N_2}{N} \times p_{K_2}(cara)$$

Provistos de la consecuencia dada en la Ecuación A.2 y renombrando $k = \frac{N_2}{N}$, $\hat{p}_{TDP}$ se puede reformular como la Ecuación A.3, esta ecuación nos dice que la probabilidad esperada aprendida por TDP se encuentra entre las probabilidades $p_{K_1}(cara)$ y $p_{K_2}(cara)$.

$$N_1 + N_2 = N \rightarrow N_1 = N - N_2 \tag{A.2}$$

$$\hat{p}_{TDP} \approx p_{K_1}(cara) + k \times (p_{K_2}(cara) - p_{K_1}(cara)), \quad 0 \leq k \leq 1 \tag{A.3}$$

A.1.2. Resultados

A.1.2.1. Elección de rango para TCM

La determinación del rango k, para cada juego de parámetros, se realizó entrenando a la técnica TCM para matrices del experimento que cumpliesen las restricciones de los parámetros dados en la Tabla A.1, los cuales son un subconjunto de los especificado en la Tabla 3.5. Notar que los únicos parámetros que varían son la fracción de monedas del perfil K_1: fNK_1 y las probabilidades de cada perfil p_{K_1}, p_{K_2}. Con lo que el ajuste no es sensible a los cambios en las dimensiones de las matrices ni a la presencia de monedas con histórico incompleto.

ID	**N**	**D**	fNK_1	N_{nc}	DP	p_{K_1}	p_{K_2}
1	100	100	0.1	0	10	0.1	0.9
2	100	100	0.5	0	10	0.1	0.9
3	100	100	0.9	0	10	0.1	0.9
4	100	100	0.5	0	10	0.5	0.5
5	100	100	0.3	0	10	0.3	0.6
6	100	100	0.6	0	10	0.3	0.6

Tabla A.1

Para cada juego de parámetros en la Tabla A.1 se generaron 10 matrices distintas que cumpliesen las restricciones dadas por los parámetros y se completaron aplicando la técnica TCM para diferentes valores del rango k. Los valores evaluados fueron:

$$k \in \{1, 3, 5, 10, 20, 30\}$$

El rango elegido para cada juego de parámetros fue aquel que mejor resultados, en métrica F1, en promedio obtuviese para las 10 matrices. Los resultados se muestran en la Tabla A.2

ID 1		**ID 2**		**ID 3**		**ID 4**		**ID 5**		**ID 6**	
k	**F1**	**k**	**F1**	**k**	**F1**	**k**	**F1**	**k**	**F1**	**k**	**F1**
1	0.809	1	0.689	1	0.324	1	0.499	1	0.505	1	0.451
3	0.870	3	0.803	3	**0.336**	3	**0.501**	3	0.509	3	**0.454**
5	**0.896**	5	**0.833**	5	0.266	5	0.494	5	**0.514**	5	0.449
10	0.880	10	0.711	10	0.165	10	0.471	10	0.499	10	0.415
20	0.730	20	0.517	20	0.160	20	0.410	20	0.452	20	0.365
30	0.644	30	0.396	30	0.159	30	0.345	30	0.395	30	0.278

Tabla A.2: Para cada ID de la Tabla A.1 se muestran los resultados promedio en F1 para el completado de las matrices variando el rango k. El mejor valor promedio en F1 de todos los rangos es resaltado en **negrita**. El rango con mejor promedio en F1 es el elegido para cada parámetro especificado en la Tabla A.1.

A.1.2.2. Comparativa de técnicas sin conocer temperatura, monedas con histórico completo

En esta sección se cubre el comportamiento esperado para los juegos de parámetros de la Tabla 3.5, para las simulaciones especificadas en la **Prueba ST2** de la Sección 3.3.1.5. Se muestran los 5 mejores resultados en métrica F1 y RMSE y se presentan los boxplots, con los resultados obtenidos por las diferentes técnicas, para los juegos de parámetros de la Tabla 3.5. Las gráficas que se muestran son solo para 10 días de predicción, ya que no se observan mayores cambios para las simulaciones con 5 y 15 días de predicción.

El comportamiento esperado para cada juego de parámetros es el siguiente:

ID	Comportamiento esperado
1	Las monedas tienen un comportamiento bien diferenciado. TDP aproxima mejor la probabilidad de las monedas del perfil K_2.
2	Comportamiento diferenciado idem que anterior. TDP aproxima una probabilidad intermedia, no favoreciendo a ninguno de los perfiles.
3	Comportamiento diferenciado idem que anterior. TDP aproxima mejor la probabilidad de las monedas del perfil K_1.
4	Comportamiento impredecible. Equiprobabilidad en el resultado para las tiradas de las monedas.
5	Comportamiento moderadamente diferenciado entre los perfiles. La diferencia en las probabilidades no es tan extrema como en los casos con **ID**= $1, 2, 3$. TDP favorece moderadamente a la clase K_2.
6	Comportamiento moderadamente diferenciado entre los perfiles. La diferencia en las probabilidades no es tan extrema como en los casos con **ID**= $1, 2, 3$. TDP favorece moderadamente a la clase K_1.

Tabla A.3

En la Tabla A.4 se visualizan los 5 mejores resultados promedios en métrica F1 y RMSE. En cuanto a los mejores promedios en F1, se ve que el juego de parámetros $p_{K_1} = 0.1$, $p_{K_2} = 0.9$ y $fNK_1 = 0.1$ es favorable para todos los modelos, incluso para el modelo TDP que obtuvo un promedio máximo, en F1, de 0.938. Observando los mejores promedios en RMSE: se tiene que los parámetros $p_{K_1} = 0.1$, $p_{K_2} = 0.9$ son factores favorables para disminuir el RMSE promedio en los modelos. Esta tabla permite avizorar cuales son las mejores condiciones del experimento pero no permite comparar realmente un modelo con otro.

Mejores promedios en F1									
ID	pK_1	pK_2	fNK_1	DP	Mod	N	D	$F1_{prom}$	$RMSE_{prom}$
1	0.1	0.9	0.1	5	TCP	10	50	0.946	0.098
2	0.1	0.9	0.1	5	TDP	10	50	0.938	0.110
3	0.1	0.9	0.1	5	TKM	10	50	0.933	0.120
4	0.1	0.9	0.1	5	TIP	10	50	0.924	0.134
5	0.1	0.9	0.1	15	ICM	200	50	0.918	0.136
Mejores promedios en RMSE									
ID	pK_1	pK_2	fNK_1	DP	Mod	N	D	$F1_{prom}$	$RMSE_{prom}$
1	0.1	0.9	0.1	5	TCP	10	50	0.946	0.098
2	0.1	0.9	0.1	5	TDP	10	50	0.938	0.110
3	0.1	0.9	0.1	5	TKM	10	50	0.933	0.120
4	0.1	0.9	0.9	5	TKM	10	100	0.763	0.132
5	0.1	0.9	0.1	5	TIP	10	50	0.924	0.134

Tabla A.4: Primeros 5 mejores juegos de parámetros para métrica F1-Score y RMSE

En la Figura A.1 se muestran los boxplots, para 10 días de predicción, para los parámetros con $ID = 1$. El comportamiento es el esperado ya que dado que existe un comportamiento bien diferenciado entre los perfiles, las técnicas que utilizan clustering con 2 perfiles TCP y TKM tienen en todos los casos valores altos en F1-Score y bajo RMSE. La técnica TIP que utiliza la información individual de las monedas es igual de buena en resultados que las dos antes mencionadas, con la ventaja para TCM y TKM que solo deben aprender 2 probabilidades (una para cada perfil) en lugar de las N probabilidades que debe aprender TIP (una para cada moneda). TDP se encuentra un escalón por debajo, igualmente los valores en $F1$ son altos para TDP por la observación de que la probabilidad aprendida por esta técnica se acerca mucho al perfil con mayor fracción de monedas, en este caso K_2, el cual tiene una probabilidad igual a 0.9 de salir cara, lo que le permite, a TDP, tener una buen *precision* y *recall* para el 90 % de las monedas. En cuanto a TCM, se observa en la Figura A.1, que la técnica, con los rangos elegidos para cada juego de parámetros, perfoma bien para matrices con dimensiones grandes, sobre todo cuando las filas (cantidad de monedas) es mayor a 100. En la elección de los rangos para TCM Sección A.1.2.1, el entrenamiento se fijó para matrices de dimension 100 monedas $\times$ 100 días, provocando un sobreajuste evidenciado en los malos resultados para las matrices con solo 10 filas.
Otra observación que se desprende es que mientras más información histórica tenga la matriz M los modelos variarán menos en sus resultados, obsérvese en los

boxplots (Figura A.1) la alta variabilidad que hay en las técnicas para el caso en donde la matriz tiene dimensiones $N = 10$ monedas y $D = 50$ días de experimento, en relación a la matriz con $N = 200$ monedas y $D = 200$ días de experimento.
En la Figura A.2 se visualizan los boxplots para los juegos de parámetros con $ID = 2$, que en principio se concluyeron menos favorables para TDP, y efectivamente se ve un aumento del error y disminución sustancial en la capacidad de aciertos de salir cara (F1-Score) para el modelo que no conoce los grupos TDP, en comparación al resto, esto se explica por el resultado esperado para TDP, concluido en la Sección 3.3.1.4, que vaticina que la probabilidad aprendida por TDP en este caso está alejada de las probabilidades reales de ambos perfiles. Respecto a TCM se vuelve a evidenciar el sobreajuste desfavorable para el completado de las matrices con 10 filas.
La Figura A.3, muestra los boxplots para el juego de parámetros con $ID = 3$. Todos los casos tienen un valor bajo en F1-Score, esto es por el hecho de que la mayoría de las monedas tiene probabilidad baja de sacar cara en una tirada, ya que el 90 % de las monedas pertenece al perfil K_1, el cual tiene una probabilidad asociada de salir cara igual a 0.1, haciéndose más dificil acertar el día en que esas pocas caras salieron para cada moneda.
La Figura A.4 muestra los boxplots para el juego de parámetros con $ID = 4$. Para este último caso obsérvese en la Tabla 3.4, $ID = 4$, que la tasa de errores de TKM en clasificar correctamente las monedas es alta. Sin embargo TCP, que sabe exactamente a que perfil pertenece cada moneda, no obtiene resultados sensiblemente mejores; sobre todo por el hecho de que la diferencia entre los perfiles no se distingue y clasificar de manera incorrecta en estas situaciones no es tan importante.
Las Figuras A.5 y A.6, muestran los boxplots para los parámetros con $ID = 5$ y 6. Se ve un comportamiento más parecido entre los modelos, ya que la diferencia entre las probabilidades y proporciones de monedas de un perfil u otro no son tan extremas como en los primeros tres casos. Se observa que a medida que la matriz M aumenta en dimensionalidad, sobre todo en cantidad de monedas, los modelos que emplean más de un cluster o perfil obtienen mejores resultados que TDP.

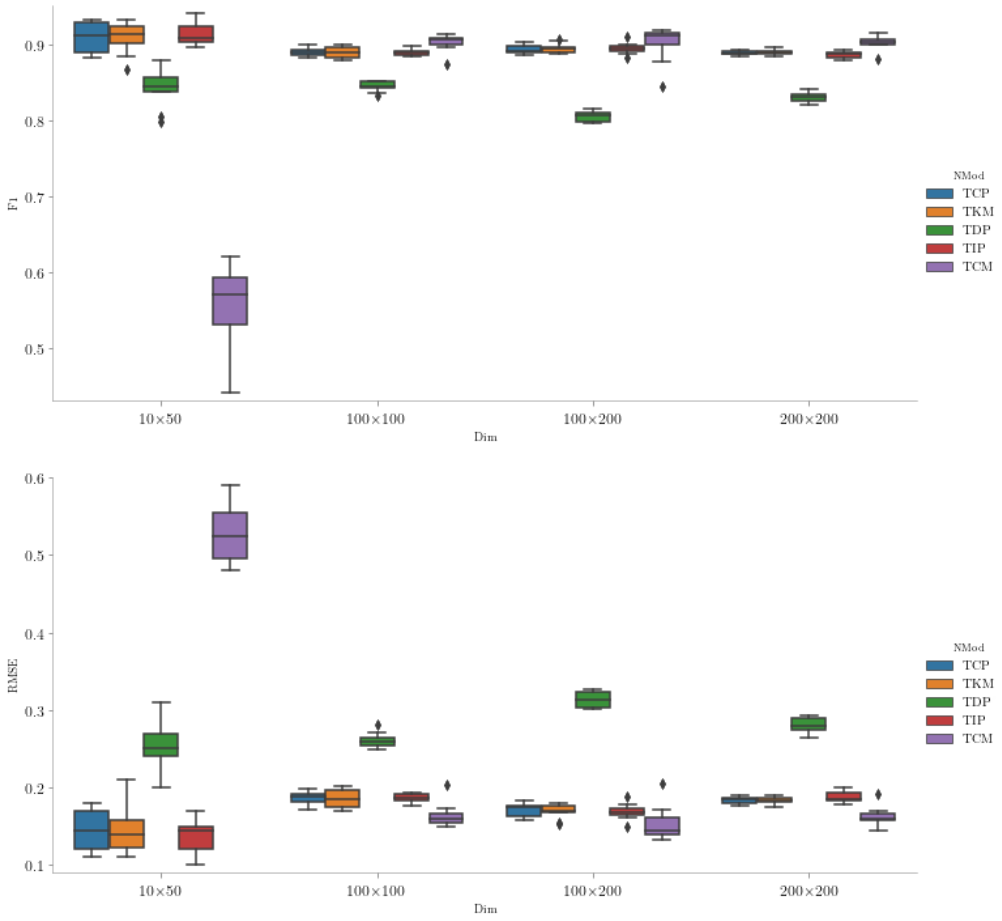


Figura A.1: Los boxplots presentan los resultados de las simulaciones para los parámetros $p_{K_1} = 0.1$, $p_{K_2} = 0.9$ y $fNK_1 = 0.1$. Los boxplots de la gráfica de arriba muestran los resultados respecto a F1-score para 10 días de predicción y distintas dimensiones de la matriz M con los datos históricos del experimento. Los boxplots de la gráfica de abajo tienen el mismo significado, pero respecto a la métrica RMSE. El eje x representa el tamaño en filas y columnas de la matriz sobre la cual se hicieron las predicciones, el eje y representa la medida F1-score y RMSE, respectivamente.

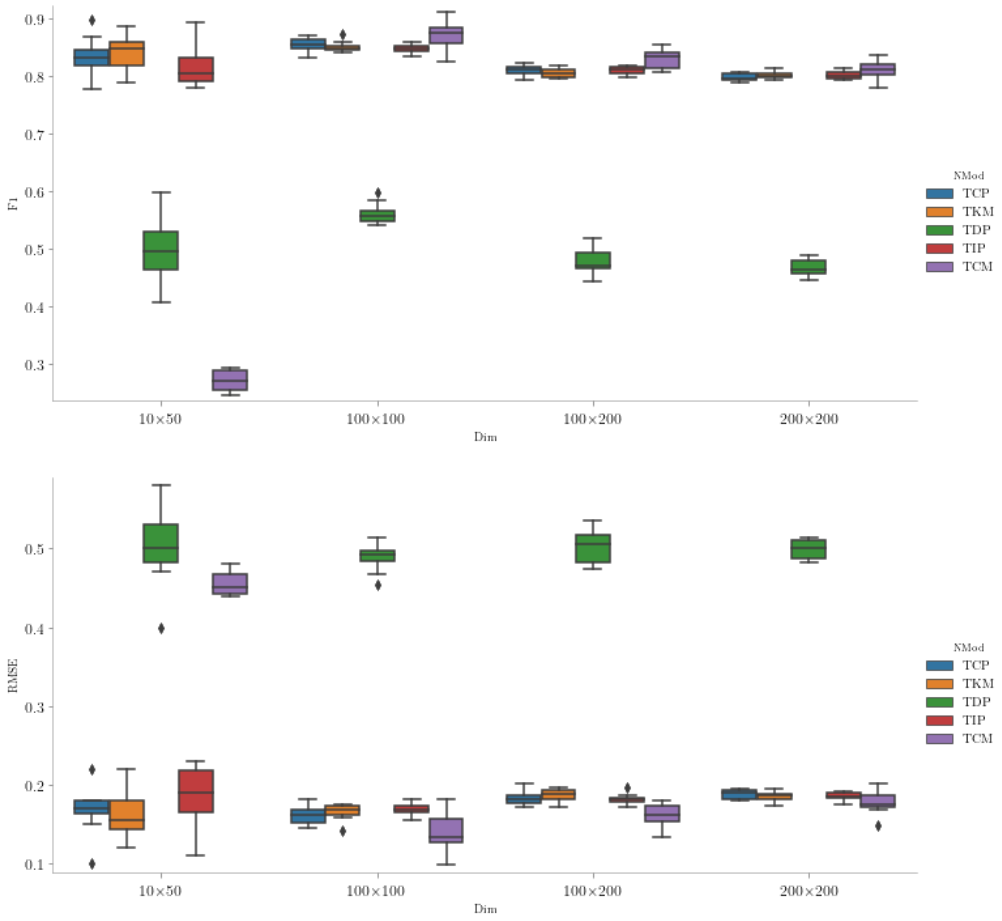


Figura A.2: El significado de las gráficas es el mismo dado en la Figura A.1. Las gráficas representan los resultados en F1 y RMSE para 10 días de predicción para los parámetros $p_{K_1} = 0.1$, $p_{K_2} = 0.9$ y $fNK_1 = 0.5$.

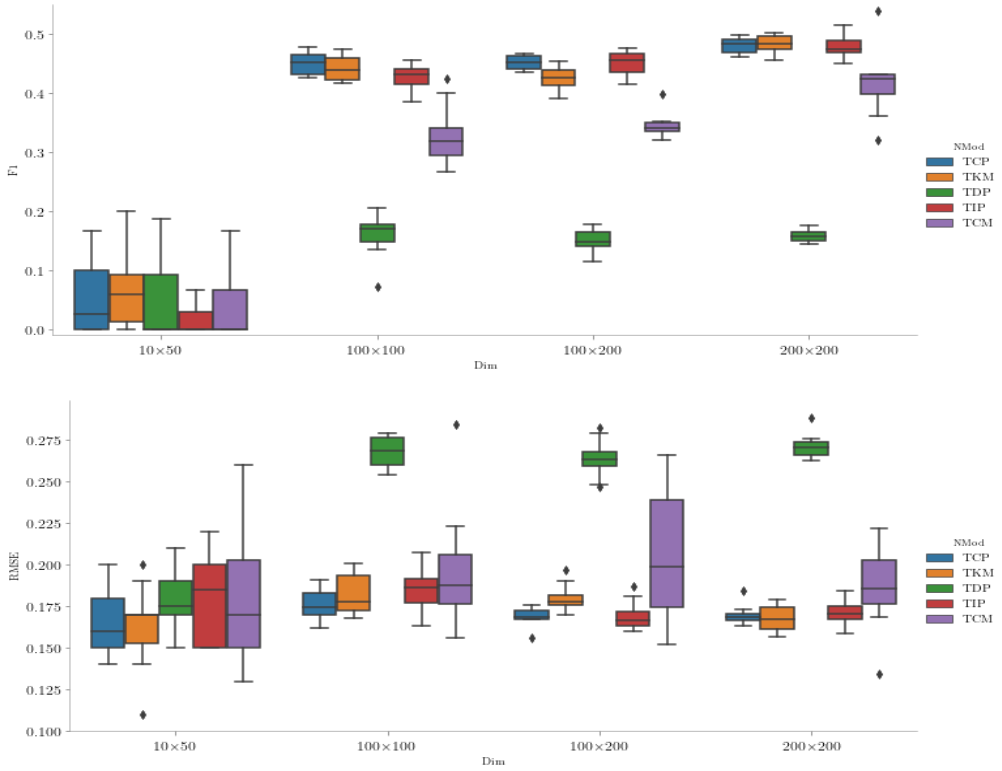


Figura A.3: Los boxplots presentan los resultados de las simulaciones para los parámetros $p_{K_1} = 0.1$, $p_{K_2} = 0.9$ y $fNK_1 = 0.9$.

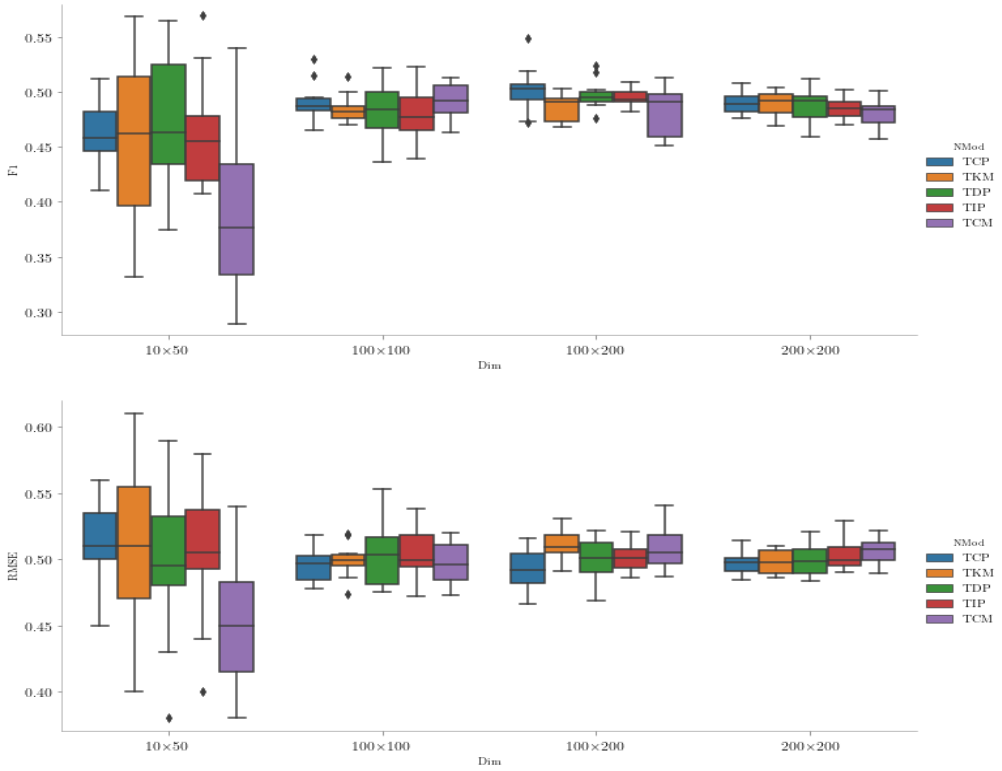


Figura A.4: Los boxplots presentan los resultados de las simulaciones para los parámetros $p_{K_1} = 0.5$, $p_{K_2} = 0.5$ y $fNK_1 = 0.5$.

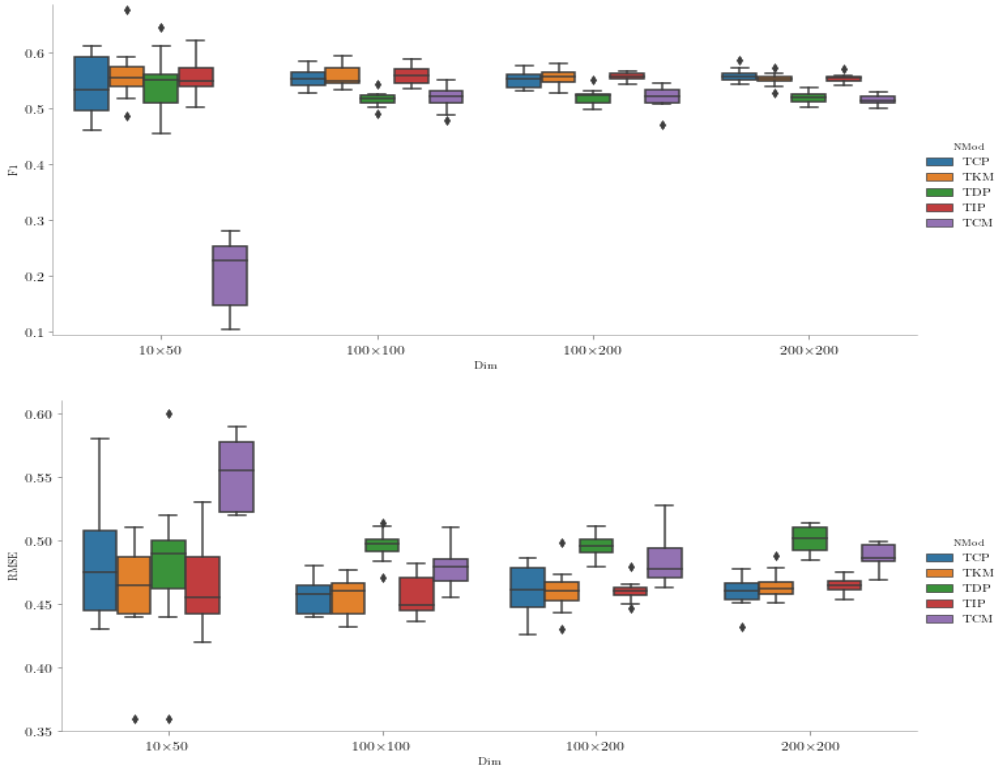


Figura A.5: Los boxplots presentan los resultados de las simulaciones para los parámetros $p_{K_1} = 0.3$, $p_{K_2} = 0.6$ y $fNK_1 = 0.3$.

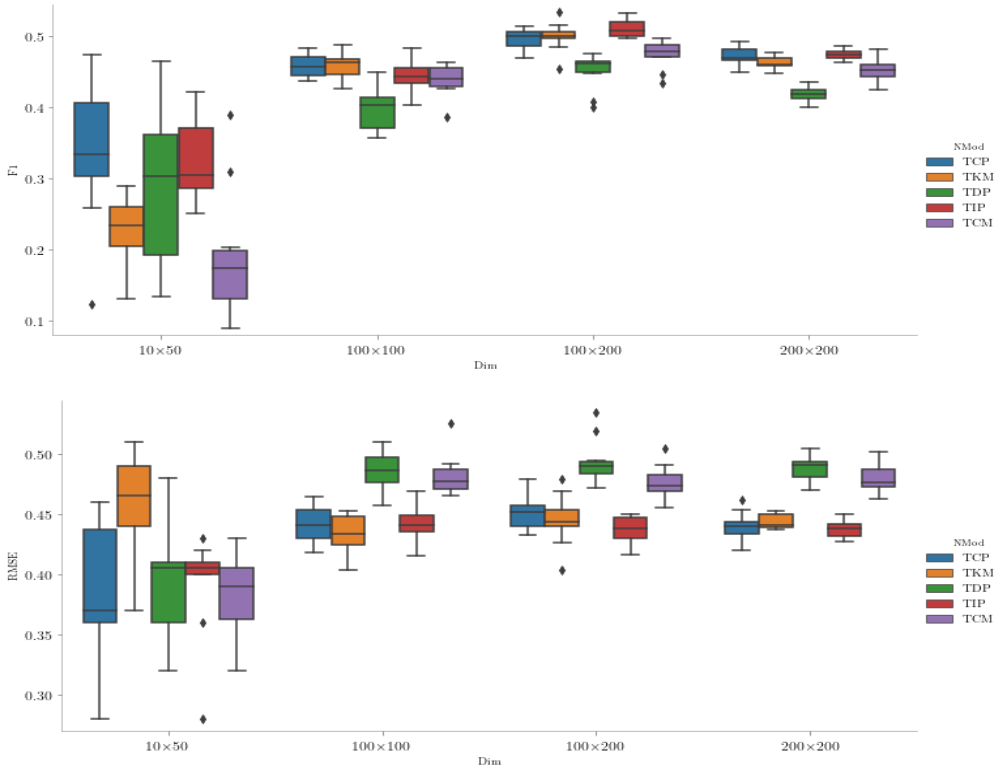


Figura A.6: Los boxplots presentan los resultados de las simulaciones para los parámetros $p_{K_1} = 0.3$, $p_{K_2} = 0.6$ y $fNK_1 = 0.6$.

A.1.2.3. Comparativa de técnicas sin conocer temperatura, monedas con histórico incompleto

En esta sección, para las simulaciones especificadas en la **Prueba ST3** de la Sección 3.3.1.5 se muestran los 5 mejores resultados en métrica F1 y RMSE y se presentan los boxplots, con los resultados obtenidos por las diferentes técnicas. Todas las monedas con histórico incompleto, para los resultados de las simulaciones que se muestran en las gráficas, tienen solo 10 lanzamientos observados.

En la Tabla A.5 se visualizan los 5 mejores resultados en RMSE y F1-Score. La tabla deja avizorar la importancia de tener dos perfiles con comportamiento muy diferentes. El modelo TCM es el que obtuvo los máximos resultados en ambas métricas.

Mejores promedios en F1									
ID	p_{K_1}	p_{K_2}	fNK_1	DI	Mod	N	D	$F1_{prom}$	$RMSE_{prom}$
1	0.1	0.9	0.1	10	TCM	100	100	0.958	0.070
2	0.1	0.9	0.5	30	TCM	200	200	0.938	0.053
3	0.1	0.9	0.1	30	TCM	200	200	0.935	0.111
4	0.1	0.9	0.1	30	TCM	200	50	0.932	0.090
5	0.1	0.9	0.1	3	TCM	200	100	0.929	0.088
Mejores promedios en RMSE									
ID	N	D	fNK_1	DI	p_{K_1}	p_{K_1}	Mod	$F1_{prom}$	$RMSE_{prom}$
1	0.1	0.9	0.9	3	TCM	50	50	0.001	0.052
2	0.1	0.9	0.5	30	TCM	200	200	0.938	0.053
3	0.1	0.9	0.9	10	TCM	50	50	0.666	0.054
4	0.1	0.9	0.9	10	TCM	200	50	0	0.070
5	0.1	0.9	0.9	10	TCP	50	200	0	0.070

Tabla A.5: Primeros 5 mejores juegos de parámetros para métrica F1-Score y RMSE

Observando los boxplots en la Figura A.8, se ve que para todas las combinaciones de parámetros con $ID = 1$ los resultados arrojaron mejores valores en F1-Score y RMSE para las técnicas que emplean clustering y para la técnica INC. La Figura A.7 muestra los boxplots con los resultados obtenidos por cada técnica para los juegos de parámetros con $ID = 2$. Esta combinación de parámetros, por lo concluido para TDP en la Sección 3.3.1.4, es la más favorable para las técnicas que emplean clustering (TCP, TKM), en comparación a las técnicas que ignoran la existencia de clusters (TDP). Se observa que las técnicas que emplean información de clusters en sus predicciones (TCP, TKM) son mejores que TDP. En cuanto a TCM, se observa una baja performance en F1 y RMSE. La performance fue peor en comparación a la observada en TCM para el mismo juego de parámetros en el caso de monedas solo con histórico completo, en este caso el RMSE promedio fue de 0.222 menor al RMSE promedio para el experimento con monedas con histórico incompleto que arrojó un valor de 0.420. Lo que permite inferir que el agregado de monedas con histórico incompleto, incide negativamente en el resultado general de la predicciones hechas aplicando *matrix completion*.
Para el caso con $ID = 3$, los resultados observados en los boxplots de la Figura A.9 muestran que hubieron situaciones en donde el modelo TDP fue mejor que el resto respecto a la métrica F1, a pesar de haber obtenido peor resultado en RMSE en comparación con el resto. Esto se puede explicar por el hecho de que en

un escenario en donde la mayoría de las monedas (90 %) tiene baja probabilidad de salir cara (0.1), es más dificil acertar en que momento saldrá cara. La técnica TDP en este escenario para todas las combinaciones de parámetros siempre tiene una medición F1-Score fluctuando alrededor de 0.1 (ver Figura A.9), mientras que el resto de las técnicas tienen en ocasiones mucho mejores tasas de acierto y una medición F1-Score mejor (alrededor de 0.6), sobre todo explicada por la presencia de monedas con observaciones incompletas perteneciente al perfil con probabilidad 0.9 de salir cara y en otras oportunidades las tasas de acierto son menores a 0.2, explicada por la dificultad de acertar la salida de cara para las monedas pertenecientes al perfil con probabilidad baja.
Observando los boxplots para los $ID = 5$ y 6, Figuras A.11, A.12, respectivamente, el comportamiento se mantiene, la mayoría de las veces las técnicas que emplean clustering e INC son mejores en F1-Score, más en algunas situaciones TDP arroja el mejor resultado.

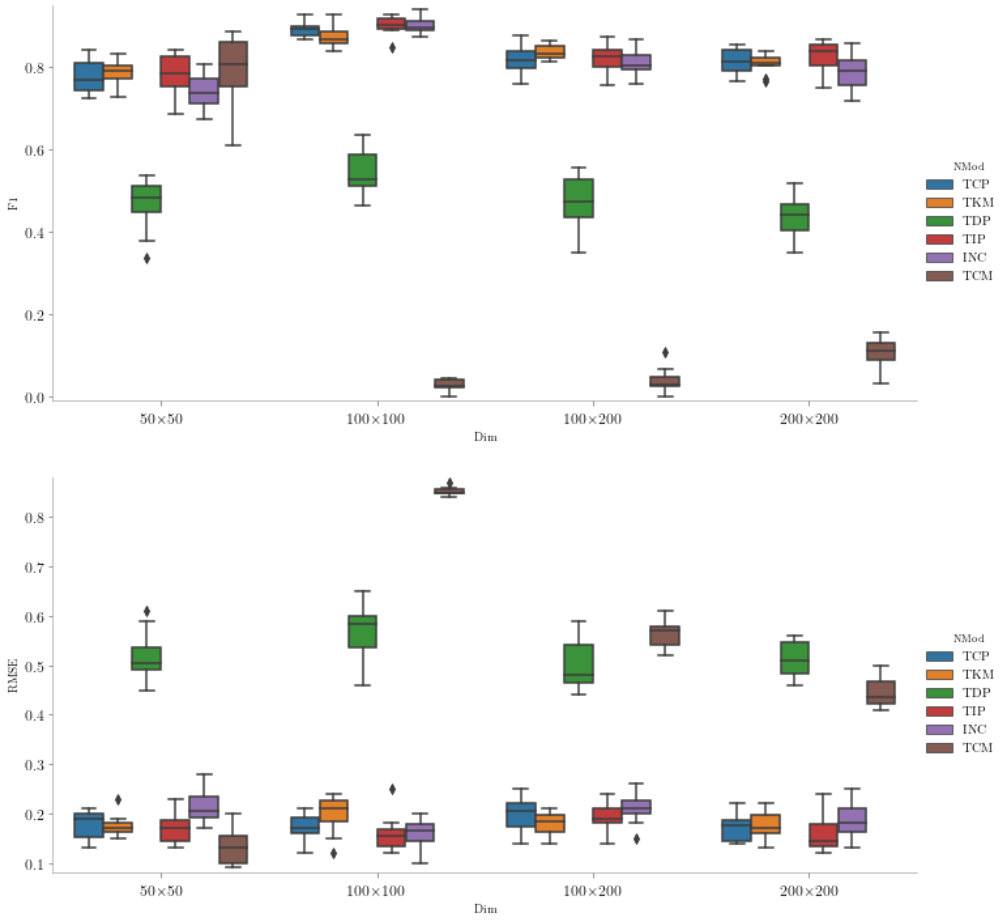


Figura A.7: En cada gráfica el eje x representa las diferentes dimensiones de la matriz M para las cuales se aplicaron las técnicas de predicción. En el eje y se encuentran los resultados en métrica F1-Score (gráfica de arriba) y RMSE (gráfica de abajo). Las gráficas muestran los resultados para el caso de monedas con histórico incompleto con 10 observaciones. Las gráficas muestran los resultados para el caso de monedas con histórico incompleto con 10 observaciones. Los boxplots presentan los resultados de las simulaciones para los parámetros $p_{K_1} = 0.1$, $p_{K_2} = 0.9$ y $fNK_1 = 0.5$.

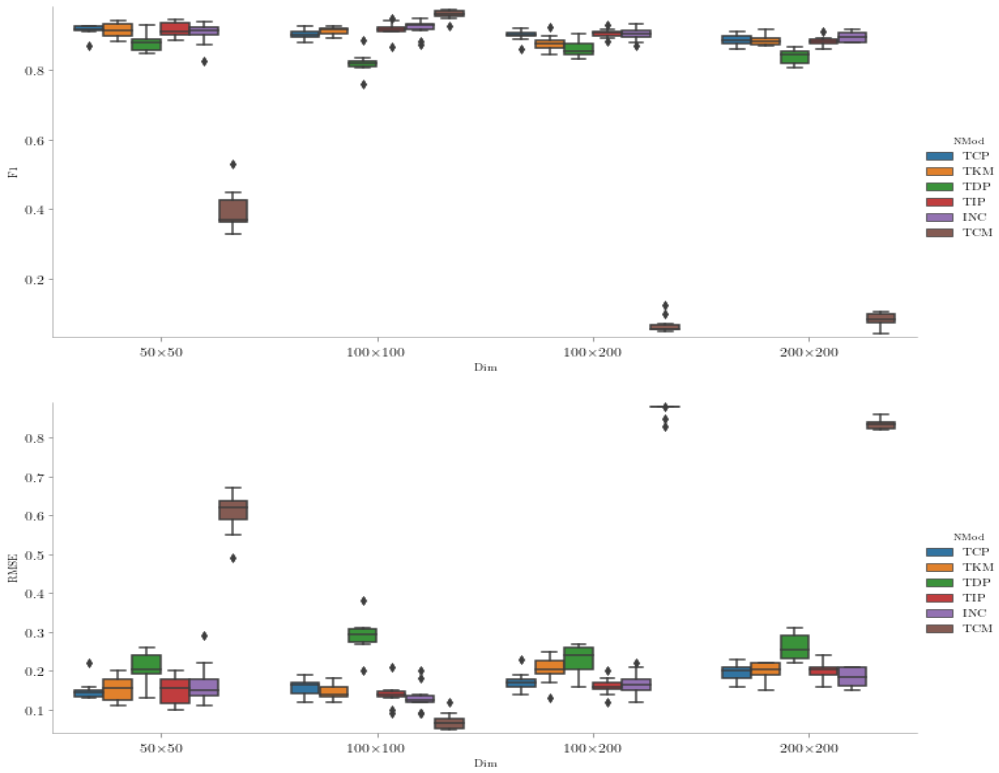


Figura A.8: Los boxplots presentan los resultados de las simulaciones para los parámetros $p_{K_1} = 0.1$, $p_{K_2} = 0.9$ y $fNK_1 = 0.1$.

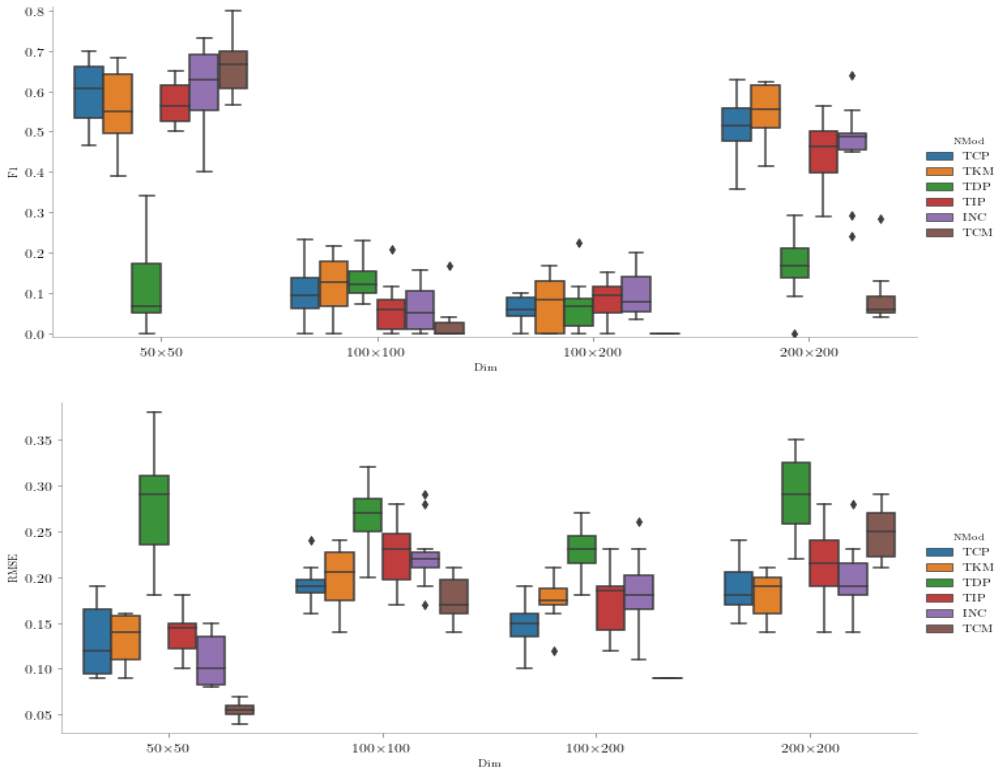


Figura A.9: Los boxplots presentan los resultados de las simulaciones para los parámetros $p_{K_1} = 0.1$, $p_{K_2} = 0.9$ y $fNK_1 = 0.9$.

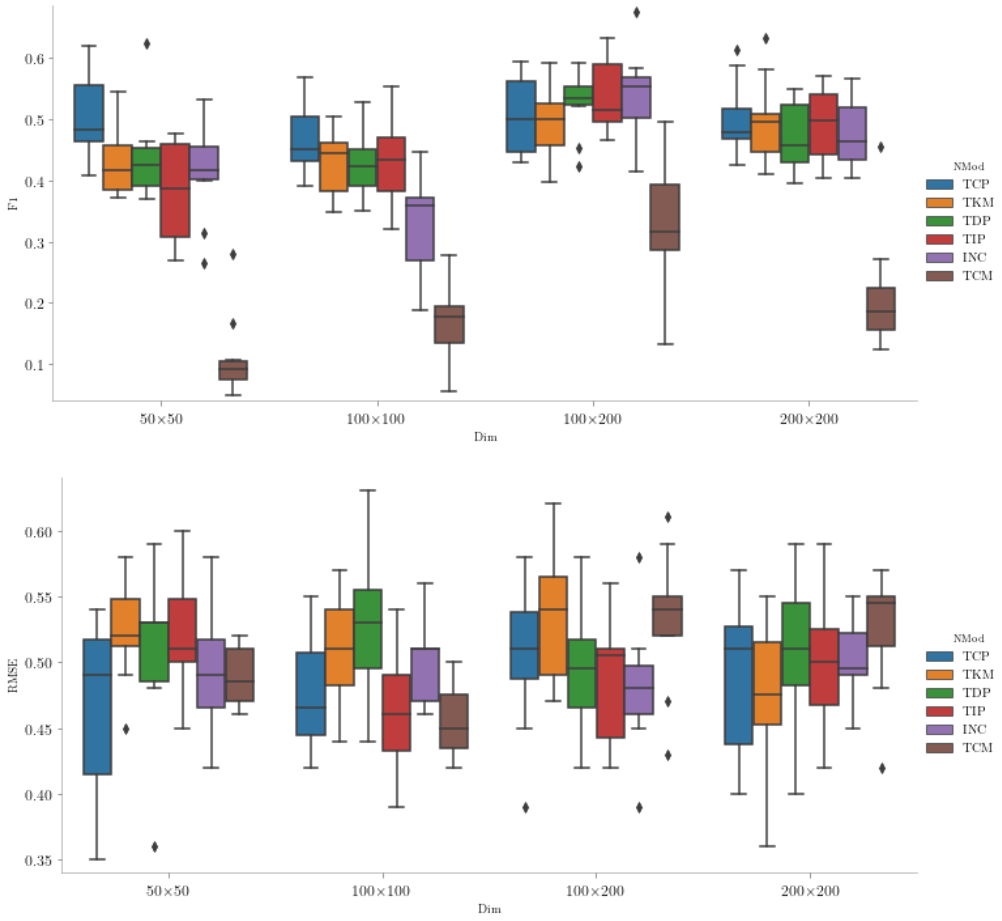


Figura A.10: Los boxplots presentan los resultados de las simulaciones para los parámetros $p_{K_1} = 0.5$, $p_{K_2} = 0.5$ y $fNK_1 = 0.5$.

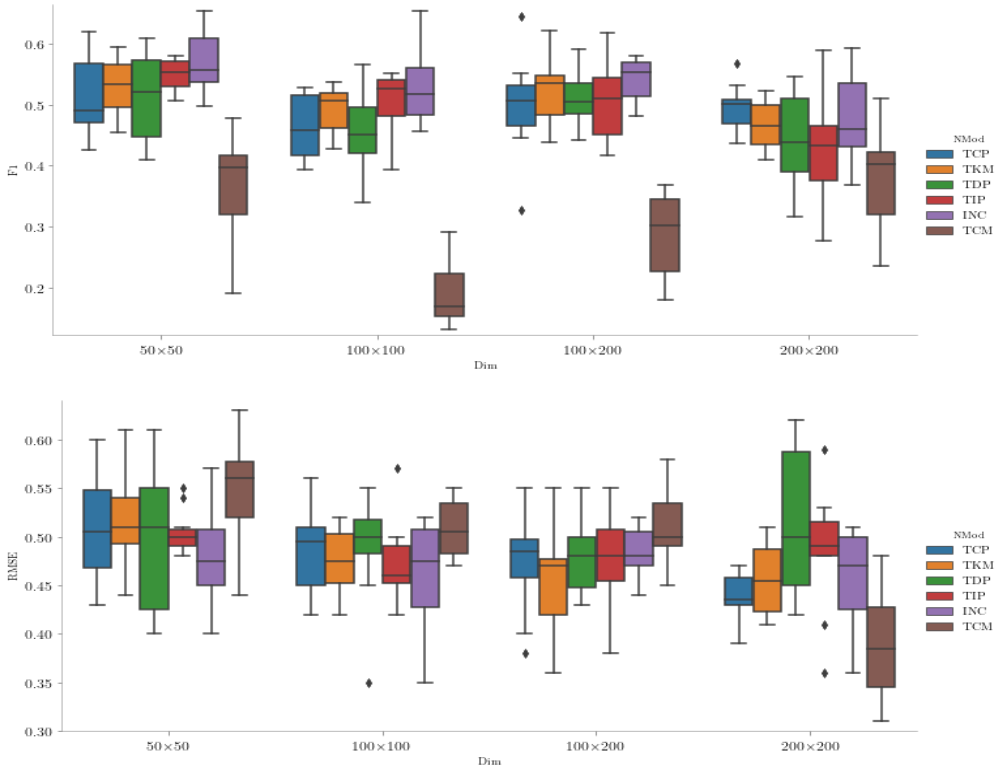


Figura A.11: Los boxplots presentan los resultados de las simulaciones para los parámetros $p_{K_1} = 0.3$, $p_{K_2} = 0.6$ y $fNK_1 = 0.3$.

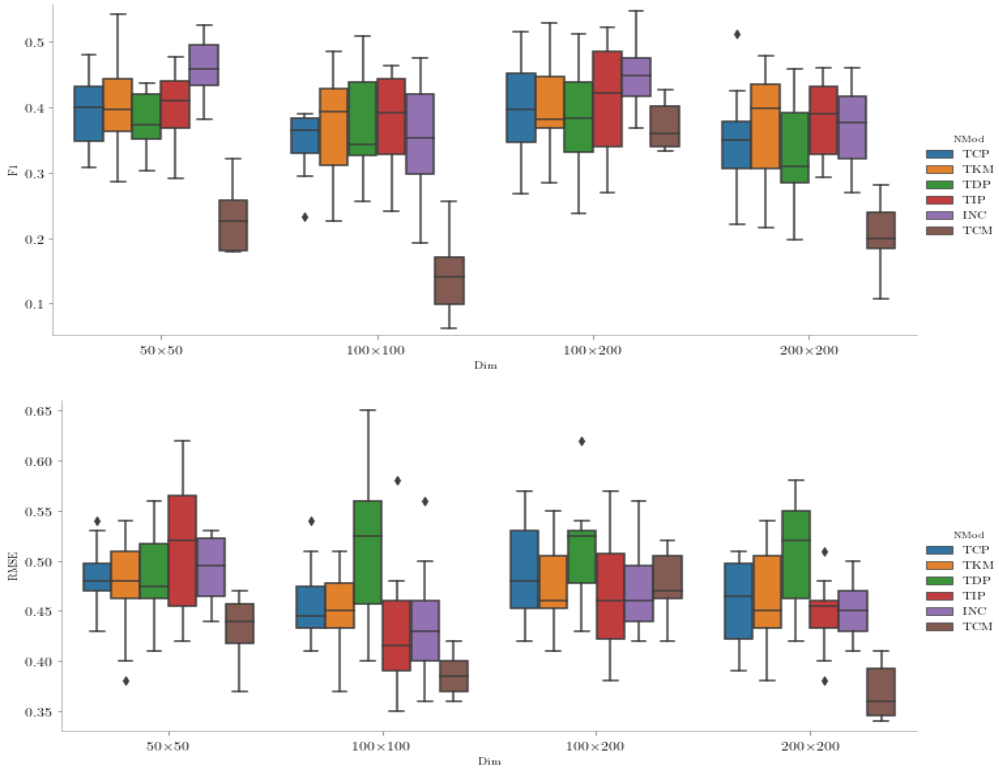


Figura A.12: Los boxplots presentan los resultados de las simulaciones para los parámetros $p_{K_1} = 0.3$, $p_{K_2} = 0.6$ y $fNK_1 = 0.6$.

A.2. Modelo de las monedas con temperatura

A.2.1. Detalles de implementación

A.2.1.1. Algoritmo del método general de aprendizaje y predicción

Sean $M_{N\times D_e}, N_c, N_{nc}, \mathcal{K}, C$ y D_p definidas de igual manera que en la Sección 3.3.2.3.
Sin pérdida de generalidad asúmase que las N_c monedas con histórico completo ocupan las primeras N_c filas de la matriz $M_{N\times D_e}$.

Algoritmo 6 $MONEDAS_2(M_{N\times D_e}, \mathcal{K}, C, D_p)$

1: $T_e \leftarrow$ arreglo con las temperaturas observadas para todos los tiempos del entrenamiento.
2: Separar la matriz M en M^c, matriz con las monedas que comienzan las observaciones a partir del primer día y M^{nc}, matriz con las monedas que comienzan las observaciones en días próximos al final del entrenamiento.
3: $iter \leftarrow D_e + 1$
4: **while** $iter - D_e \leq D_p$ **do**
5: // Predecir la temperatura
6: Predecir la temperatura del siguiente día $T_{sig} \in \{T_A, T_B\}$, con probabilidad estimada $\hat{p}(T_A)$ igual a la razón entre la cantidad de veces que en T_e las temperaturas fueron T_A y $|T_e|$. Almacenar la temperatura del siguiente día en T_{sig}. Agregar T_{sig} a T_e.
7: // Obtener submatrices para aprender las probabilidades, y asignar perfil a las monedas con histórico incompleto
8: A partir de M^c obtener $M^c_{T_{sig}}$, la submatriz con todas las columnas que coincidan con los días en que hubo temperatura igual a la predicha T_{sig}.
9: A partir de M^{nc} obtener $M^{nc}_{T_{sig}}$, la submatriz con todas las columnas que coincidan con los días en que hubo temperatura igual a la predicha T_{sig}.
10: $\hat{C} \leftarrow asignarPerfil(M^c_{T_{sig}}, M^{nc}_{T_{sig}}, \mathcal{K}, C)$
11: // Aprendizaje
12: **for** $k \in \mathcal{K}$ **do**
13: Obtener $M^c_{k,T_{sig}}$, la submatriz con las filas únicamente de las monedas $i \in N_c$ con perfil k y con columnas que coincidan con los días en que hubo temperatura igual a la predicha T_{sig}. Sea además $|C_k|$ la cantidad de monedas con perfil k y $|T_{e|T_{sig}}|$ la cantidad de temperaturas T_{sig} que hay en el arreglo de temperaturas T_e
14: **for** j columna en $M^c_{k,T_{sig}}$ **do**
15: Calcular la razón entre la cantidad de caras en $M^c_{k,T_{sig}}[:, j]$, la columna con los resultados del j-ésimo día con temperatura igual a T_{sig} y $|C_k|$. Sea $\hat{p}_j$ el resultado de este cálculo, que se puede ver como la probabilidad aproximada de salir cara en el día j-ésimo con temperatura igual a T_{sig}, para el perfil k
16: **end for**
17: La probabilidad aproximada de salir cara para las monedas en k se aproxima como el promedio de los $\hat{p}_j$. Entonces $\hat{p}_k(cara|T_{sig}) = \frac{1}{|T_{e|T_{sig}}|} \sum_{j=1}^{|T_{e|T_{sig}}|} \hat{p}_j$.
18: **end for**
19: // Predicción

20: **for** $i \in N$ **do**
21: **if** i es moneda con histórico completo **then**
22: $k \leftarrow C[i]$
23: **else if** i es moneda con histórico incompleto **then**
24: $k \leftarrow \hat{C}[i - N_c]$
25: **end if**
26: El resultado de la tirada para el día *iter* se predice con $\hat{p}_k(cara|T_{sig})$.
27: **if** i es moneda con histórico completo **then**
28: Agregar al final de $M^c[i]$ la predicción hecha en el paso anterior.
29: **else if** i es moneda con histórico incompleto **then**
30: Agregar al final de $M^{nc}[i - N_c]$ la predicción hecha en el paso anterior.
31: **end if**
32: **end for**
33: $iter = iter + 1$
34: **end while**

El Algoritmo 6, para cada uno de los D_p días de predicción, ejecuta los bloques de código // *Predecir la temperatura*, paso 6, // *Obtener submatrices para aprender las probabilidades, y asignar perfil a las monedas con histórico incompleto*, pasos del 8 al 10, // *Aprendizaje*, pasos del 12 al 18 y // *Predicción*, pasos del 20 al 32. El primer paso es asignar a T_e las temperaturas observadas para todos los tiempos de entrenamiento, luego en el segundo paso se separa la matriz M en M^c y M^{nc}, las submatrices con solo los datos de las monedas con histórico completo e histórico incompleto, respectivamente.
En el bloque // *Predecir la temperatura*, el algoritmo aprende una probabilidad $\hat{p}(T_A)$ de que en el próximo día la temperatura sea T_A, a partir de los datos históricos de temperatura. Prediciendo luego con esta probabilidad T_{sig}, la temperatura para el siguiente día de predicción.
En el bloque // *Obtener submatrices para aprender las probabilidades, y asignar perfil a las monedas con histórico incompleto*, de las matrices M^c y M^{nc} se extraen las columnas de los días que coincidan en temperatura con T_{sig}, creando respectivamente las submatrices: $M^c_{T_{sig}}$ y $M^{nc}_{T_{sig}}$. El etiquetado de las monedas con histórico incompleto se hace igual que en el Algoritmo 1. En este caso el cálculo del consumo medio de las monedas con histórico incompleto (paso 8 Alg. 2) y de los perfiles (paso 4 Alg. 2) será realizado sobre los días en que las temperaturas coincidan con T_{sig}.
En el bloque // *Aprendizaje*, para cada perfil k se aprende una probabilidad $\hat{p}_k(cara|T_{sig})$, en función de los datos de consumo de las monedas de dicho perfil

para los días cuya temperatura coinciden con T_{sig}.
En el bloque // *Predicción*, para cada moneda se predice que el resultado del lanzamiento, para el día D_e + *iter*, será cara, con probabilidad igual a la aprendida para el perfil al cual la moneda pertenece.

A.2.1.2. Algoritmo del método de completado de matrices

Sean $M_{N\times D_e}, N_c, N_{nc}, \mathcal{K}, C, T_e$ y D_p definidas de igual manera que para la técnica TCM en la Sección 3.3.2.3.
Sin pérdida de generalidad asúmase que las N_c monedas con histórico completo ocupan las primeras N_c filas de la matriz $M_{N\times D_e}$

Algoritmo 7 $TCM_2(M_{N\times D_e}, \mathcal{K}, C, T_e, k, D_p)$

1: Separar la matriz M en M^c, matriz con las monedas que comienzan las observaciones a partir del primer día y M^{nc}, matriz con las monedas que comienzan las observaciones en días próximos al final del entrenamiento.
2: Clasificar las filas en M^c en dos perfiles $\{K_1, K_2\}$, empleando K-means. Guardar esta clasificación en C.
3: $iter \leftarrow D_e + 1$
4: **while** $iter - D_e \leq D_p$ **do**
5: //Predecir temperatura
6: Predecir la temperatura del siguiente día $T_{sig} \in \{T_A, T_B\}$, con probabilidad estimada $\hat{p}(T_A|T_e)$ igual a la razón entre la cantidad de veces que en T las temperaturas fueron T_A y $|T_e|$. Almacenar la temperatura del siguiente día T_{sig}. Agregar T_{sig} a T_e.
7: //Obtener submatrices para aplicar ICMC y asignar perfiles a monedas con histórico incompleto
8: A partir de M^c obtener $M^c_{T_{sig}}$, la submatriz con todas las columnas que coincidan con los días en que hubo temperatura igual a la predicha T_{sig}.
9: A partir de M^{nc} obtener $M^{nc}_{T_{sig}}$, la submatriz con todas las columnas que coincidan con los días en que hubo temperatura igual a la predicha T_{sig}.
10: $\hat{C} \leftarrow asignarPerfil(M^c_{T_{sig}}, M^{nc}_{T_{sig}}, \mathcal{K}, C)$
11: //Aplicar ICMC a cada perfil
12: Agregar una nueva columna al final en $M^c_{T_{sig}}$ y en $M^{nc}_{T_{sig}}$, estas columnas guardarán los resultados de la siguiente predicción.
13: **for** $k \in \mathcal{K}$ **do**

14: Obtener de $M^{c}_{T_{sig}}$ y $M^{nc}_{T_{sig}}$, las filas únicamente de las monedas con perfil k. Llámese esta matriz $M_{k,T_{sig}}$. Guardar en I^{c}, I^{nc} los índices de estas filas.

15: Elegir al azar el 1 % de las filas en $M_{k,T_{sig}}$ y predecir su resultado para el siguiente día, con probabilidad de salir cara igual a la razón entre la cantidad de caras observadas para dicha moneda y la cantidad de días de entrenamiento $iter - 1$.

16: Aplicar el algoritmo $ICMC(M_{k,T_{sig}}, rango = k)$

17: En la nueva columna *iter* agregada en M^{c} y M^{nc}, de acuerdo a los índices dados en I^{c}, I^{nc}, completar las entradas predichas para las monedas con perfil k.

18: **end for**

19: Rearmar M a partir de las matrices M^{c} y M^{nc}, sobre las cuales se agregaron las D_p columnas de predicción.

20: $iter = iter + 1$

21: **end while**

El Algoritmo 7, en el paso 1, comienza con la separación de las monedas en aquellas con histórico completo M^{c} y aquellas con histórico incompleto M^{nc}.
En el paso 2, se realiza una clasificación usando K-means, de las monedas con histórico completo.
Luego para cada uno de los D_p días de predicción se ejecutan los bloques: *//Predecir temperatura*, paso 6, *//Obtener submatrices para aplicar ICMC y asignar perfiles a monedas con histórico incompleto*, pasos del 8 al 10, *//Aplicar ICMC a cada perfil*, pasos del 12 al 18.
Los bloques *//Predecir temperatura* y *//Obtener submatrices para aplicar ICMC y asignar perfiles a monedas con histórico incompleto* son idénticos a los primeros dos bloques dentro del `while` en el Algoritmo 6.
En el bloque *//Aplicar ICMC a cada perfil*, para cada perfil k, se obtienen de las matrices $M^{c}_{T_{sig}}$ y $M^{nc}_{T_{sig}}$, la matriz incompleta $M_{k,T_{sig}}$, conteniendo en cada fila i a las monedas con histórico completo e incompleto, pertenecientes al perfil k (paso 14).
Luego del `for`, con bloque de ejecución determinado por los pasos del 13 al 18. La matrices M^{c} y M^{nc} tendrán una nueva columna, agregada al final, con las predicciones resultantes de aplicar el Algoritmo *ICMC* 11 a cada una de las matrices $M_{k,T_{sig}}$ de cada perfil.
En el paso 19 las predicciones de la última columna en M^{c} y M^{nc} son agregados a la matriz M.

En el paso 20 se termina el bloque de ejecución del `while` y un nuevo día de predicción vuelve a comenzar.

A.2.2. Comportamiento esperado

A.2.2.1. Elección de rango para TCMCT

La determinación del rango k, para cada juego de parámetros, se realizó entrenando a la técnica TCMCT para matrices del experimento que cumpliesen las restricciones de los parámetros dados en la Tabla A.6, los cuales son un subconjunto de los especificado en la Tabla 3.8. Notar que los únicos parámetros que varían son la fracción de monedas del perfil K_1: fNK_1, la temperatura fT_A y las probabilidades de cada perfil. Con lo que el ajuste no es sensible a los cambios en las dimensiones de las matrices ni a la presencia de monedas con histórico incompleto.

ID	**N**	**D**	fNK_1	N_{nc}	DP	fT_A	$p_{K_1}(cara\|T_A)$	$p_{K_1}(cara\|T_B)$	$p_{K_2}(cara\|T_A)$	$p_{K_2}(cara\|T_B)$
1	100	100	$\{0.1, 0.5, 0.9\}$	0	10	0,8	0.1	0.9	0.9	0.1
2	100	100	$\{0.1, 0.5, 0.9\}$	0	10	0,5	0.1	0.9	0.9	0.1
3	100	100	$\{0.1, 0.5, 0.9\}$	0	10	0,8	0.3	0.6	0.6	0.3

Tabla A.6

Para cada juego de parámetros en la Tabla A.6 se generaron 10 matrices distintas que cumpliesen las restricciones dadas por los parámetros y se completaron aplicando la técnica TCMCT para diferentes valores del rango k. Los valores evaluados fueron:

$$k \in \{1, 3, 5, 10, 20, 30\}$$

El rango elegido para cada juego de parámetros fue aquel que mejor resultados, en métrica F1, en promedio obtuviese para las 10 matrices. Los resultados se muestran en la Tabla A.7

$fNK_1 = 0{,}1$					
ID 1		**ID 2**		**ID 3**	
k	**F1**	**k**	**F1**	**k**	**F1**
1	0,441	1	**0.388**	1	0,402
3	0,440	3	0,361	3	0,418
5	0,411	5	0,386	5	0,411
10	0,391	10	0,387	10	**0.426**
20	**0.449**	20	0,379	20	0,412
30	0,432	30	0,360	30	0,397
$fNK_1 = 0{,}5$					
ID 1		**ID 2**		**ID 3**	
k	**F1**	**k**	**F1**	**k**	**F1**
1	0,452	1	0,548	1	**0.479**
3	**0.517**	3	0,550	3	0,471
5	0,468	5	0,555	5	0,461
10	0,440	10	0,579	10	0,460
20	0,444	20	0,551	20	0,469
30	0,497	30	**0.603**	30	0,471
$fNK_1 = 0{,}9$					
ID 1		**ID 2**		**ID 3**	
k	**F1**	**k**	**F1**	**k**	**F1**
1	**0.420**	1	0,581	1	0,504
3	0,392	3	0,607	3	0,492
5	0,416	5	**0.630**	5	0,488
10	0,406	10	0,612	10	0,497
20	0,414	20	0,559	20	0,493
30	0,392	30	0,585	30	**0.507**

Tabla A.7: Para cada ID de la Tabla A.6 se muestran los resultados promedio en F1 para el completado de las matrices variando el rango k. El mejor valor promedio en F1 de todos los rangos es resaltado en **negrita**. El rango con mejor promedio en F1 es el elegido para cada parámetro especificado en la Tabla A.6.

A.2.2.2. Comparación del modelo TCPCT versus TCP

Considérse el comportamiento de las monedas para los juegos de parámetros de la Tabla A.8.

N	D	fNK_1	N_{nc}	DP	$p_{K_1}(cara|T_A)$	$p_{K_1}(cara|T_B)$	$p_{K_2}(cara|T_A)$	$p_{K_2}(cara|T_B)$
100	100	0,1	10	10	0	1	1	0

Tabla A.8

Esta configuración se representa esquemáticamente en la Tabla A.9 (sin tener en cuenta las N_{nc} monedas con histórico incompleto). En donde las filas de los datos *Históricos* se agrupan por perfil de moneda y las columnas por tipo de temperatura. La fila 1 representa al grupo de monedas con perfil K_1 y la fila 2 representa al grupo de monedas con perfil K_2. La columna 1 representa al grupo de días con temperaturas altas T_A y la columna 2 representa al grupo de días con temperaturas bajas T_B. Los valores 0 y 1 dentro de cada celda representan el resultado de los lanzamientos según el perfil de la moneda y la temperatura que hubo en el día.

Temperaturas		
	T_A	T_B
Históricos		
K_1	0	1
K_2	1	0

Tabla A.9: Valores de los consumos históricos para las monedas de los perfiles K_1 y K_2 según las dos temperaturas T_A y T_B.

Se puede observar que conociendo la temperatura de los días de predicción se puede conocer exactamente que resultados tendrán las monedas para cada perfil. La Tabla A.10 muestra los resultados de las monedas para un día con temperaturas bajas T_B y uno con temperaturas altas T_A.

Temperaturas		
	T_A	T_B
Históricos		
K_1	0	1
K_2	1	0

Tabla A.10: Valores futuros para los perfiles K_1 y K_2 si la temperatura futura es T_A o T_B.

Un modelo del tipo TCP que no tiene información sobre la temperatura que se observa durante los experimentos, aprenderá las probabilidades $p_{K1}(cara)$ y $p_{K2}(cara)$, agrupando las monedas según los perfiles y calculando el promedio de caras a partir de los datos históricos.
Independientemente de que la temperatura sea T_A y T_B, en promedio TCP predecirá las siguientes probabilidades:

$$\hat{p}_{K1}(cara) = \frac{\#\text{caras celda sup. izq.} + \#\text{caras celda sup. der.}}{\#\text{monedas en K1} \cdot \#\text{días históricos}}$$

$$\hat{p}_{K2}(cara) = \frac{\#\text{caras celda inf. izq.} + \#\text{caras celda inf. der.}}{\#\text{monedas en K2} \cdot \#\text{días históricos}}$$

haciendo los cálculos se tiene:

$$\hat{p}_{K1}(cara) = \frac{(0.1 \cdot 100) \cdot (0.8 \cdot 100) \cdot 0 + (0.1 \cdot 100) \cdot (0.2 \cdot 100) \cdot 1}{0.1 \cdot 100 \cdot 100} = 0.2$$

$$\hat{p}_{K2}(cara) = \frac{(0.9 \cdot 100) \cdot (0.8 \cdot 100) \cdot 1 + (0.9 \cdot 100) \cdot (0.2 \cdot 100) \cdot 0}{0.9 \cdot 100 \cdot 100} = 0.8$$

con estas probabilidades aprendidas la cantidad de caras predichas para cada perfil por TCP, independientemente de la temperatura que haya, será aproximadamente:

$$\#\widehat{caras}(K_1) \approx \hat{p}_{K1}(cara) \cdot \#\text{cantidad de monedas en } K_1$$

$$\#\widehat{caras}(K_2) \approx \hat{p}_{K2}(cara) \cdot \#\text{cantidad de monedas en } K_2$$

haciendo las cuentas se tiene:

$$\#\widehat{caras} \approx \#\widehat{caras}(K_1) + \#\widehat{caras}(K_2) \approx 0.2(0.1 \cdot 100) + 0.8(0.9 \cdot 100) = 2 + 72 = 74$$

Si se quisiera calcular la métrica F1-score para TCP se tiene que habría un valor F1-score para los días de temperaturas altas y otro para los días de temperaturas bajas. La Tabla A.11 muestra el cálculo, con los resultados intermedios. En donde

se ve que el resultado de TCP es bueno para los días a predecir con temperatura T_A y malo para los días con temperatura T_B.

Temp. predicha: T_B			Temp. predicha: T_A		
VP	**FP**	**FN**	**VP**	**FP**	**FN**
2	72	$(0.2 \cdot 100) - 2 = 18$	72	2	$(0.8 \cdot 100) - 72 = 8$
Precision			**Precision**		
$\frac{2}{2+18} = \frac{2}{20}$			$\frac{72}{72+8} = \frac{72}{80}$		
Recall			**Recall**		
$\frac{2}{2+72} = \frac{2}{74}$			$\frac{72}{72+8} = \frac{72}{74}$		
F1			**F1**		
$2\frac{\text{presicion} \cdot \text{recall}}{\text{presicion}+\text{recall}} = 0.04$			$2\frac{\text{presicion} \cdot \text{recall}}{\text{presicion}+\text{recall}} = 0.935$		

Tabla A.11

Para el modelo TCPCT las probabilidades aprendidas y la cantidad de caras para un día predicho con temperatura T_A o T_B son más directas de calcular, son los números que se observan en cada bloque de las Tablas A.9 y A.10:

$$\hat{p}_{K_1}(cara|T_A) = 0$$

$$\hat{p}_{K_1}(cara|T_B) = 1$$

$$\hat{p}_{K_2}(cara|T_A) = 1$$

$$\hat{p}_{K_2}(cara|T_B) = 0$$

para predicciones en temperaturas altas:

$$\#\widehat{caras}|T_A(K_1) = 0 \cdot \#\text{cant. monedas en } K_1 = 0$$

$$\#\widehat{caras}|T_B(K_2) = 1 \cdot \#\text{cant. monedas en } K_2 = 90$$

y para temperaturas bajas:

$$\#\widehat{caras}|T_A(K_1) = 1 \cdot \#\text{cant. monedas en } K_1 = 10$$

$$\#\widehat{caras}|T_B(K_2) = 0 \cdot \#\text{cant. monedas en } K_2 = 0$$

La métrica F1 valdría siempre 1 si TCPCT acertara en la temperatura del día a predecir.
La pregunta, en cambio es ¿qué sucedería con F1 si TCPCT se equivoca en la

predicción de la temperatura? La Tabla A.12 muestra los cálculos para un día con temperatura alta y otro con temperatura baja. El resultado es $\frac{0}{0}$ = indef ya que tanto *precision* como *recall* valen 0. Por defecto ante esta situación se puede establecer un valor F1 de 0.

Temp. predicha: T_B			Temp. predicha: T_A		
VP	**FP**	**FN**	**VP**	**FP**	**FN**
0	90	10	0	10	90
Precision			**Precision**		
$\frac{0}{0+10} = 0$			$\frac{0}{0+90} = 0$		
Recall			**Recall**		
$\frac{0}{0+90} = 0$			$\frac{0}{0+10} = 0$		
F1			**F1**		
indef $\rightarrow 0$			*indef* $\rightarrow 0$		

Tabla A.12

Observando los resultados de las tablas se puede notar que para TCP Tabla A.11, si el día de predicción registra temperaturas altas, el valor F1 es alto (0.935) y para TCPTP Tabla A.12, si el día de predicción registra temperaturas altas y TCPTP se equivoca en la predicción de la temperatura, el valor F1 será bajo (0).

A.2.2.3. Comparativa de técnicas conociendo temperaturas, monedas con histórico completo

En esta sección se cubre el comportamiento esperado para los juegos de parámetros de la Tabla 3.12, para las simulaciones especificadas en la **Prueba CT1** de la Sección 3.3.2.5. Se muestran los 5 mejores resultados en métrica F1 y RMSE y finalmente se presentan los boxplots, con los resultados obtenidos por las diferentes técnicas, para los juegos de parámetros de la Tabla 3.12.

ID	Comportamiento esperado
1	La fracción de temperaturas de clase T_A es alta, por lo que habrá mayoría de días con temperatura igual a T_A y se espera que haya una alta tasa de aciertos al momento de predecirla, lo que conlleva a la elección correcta del subconjunto de columnas para predecir. Los perfiles tienen probabilidades muy diferentes en ambos tipos de temperaturas. Se espera un buen comportamiento de las técnicas que usan clustering TCPCT y TKMCT.
2	La fracción de temperaturas de clase T_A es del 50 %, se espera que aumente el error en la predicción de la temperatura, con la consecuente disminución en la performance tanto en F1 como en RMSE para todas las técnicas.
3	En este caso la diferencia entre las probabilidades para ambos perfiles, comparando para ambas temperaturas, no es tan extrema como en los primeros casos, se espera una mejor performance de las técnicas de clustering, pero con valores en F1 y RMSE más bajos que para el caso 1.

Tabla A.13

En la Tabla A.14 se muestran los primeros cinco mejores resultados promedios en F1 y RMSE. El valor máximo en F1 es menor y el mínimo en RMSE es mayor en comparación a los conseguidos en el experimento que no incluye la temperatura como factor (Tabla A.4, **Prueba ST2**). Los mejores resultados se dan para los perfiles con mayor diferencia en sus probabilidades, para ambas temperaturas. En la mayoría de los casos la fracción de temperaturas T_A (fT_A) es igual a 0.8, estando esto asociado a una mejora en el acierto de la siguiente temperatura a predecir, en comparación a fT_A valiendo 0.5.

Mejores promedios en F1												
ID	$pK_1(cara\|T_A)$	$pK_1(cara\|T_B)$	$pK_2(cara\|T_A)$	$pK_2(cara\|T_B)$	fT_A	fNK_1	DP	Mod	N	D	$F1_{prom}$	$RMSE_{prom}$
1	0.1	0.9	0.9	0.1	0.8	0.5	1	TIPCT	30	30	0.900	0.113
2	0.1	0.9	0.9	0.1	0.8	0.9	1	TCPCT	100	30	0.895	0.167
3	0.1	0.9	0.9	0.1	0.8	0.9	1	TCMCT	200	100	0.854	0.189
4	0.1	0.9	0.9	0.1	0.8	0.9	1	TKMCT	30	30	0.848	0.193
5	0.1	0.9	0.9	0.1	0.8	0.9	5	TKMCT	100	100	0.846	0.216
Mejores promedios en RMSE												
ID	$pK_1(cara\|T_A)$	$pK_1(cara\|T_B)$	$pK_2(cara\|T_A)$	$pK_2(cara\|T_B)$	fT_A	fNK_1	DP	Mod	N	D	$F1_{prom}$	$RMSE_{prom}$
1	0.1	0.9	0.9	0.1	0.8	0.5	1	TIPCT	30	30	0.900	0.113
2	0.1	0.9	0.9	0.1	0.8	0.1	1	TCPCT	30	200	0.500	0.140
3	0.1	0.9	0.9	0.1	0.8	0.1	1	TIPCT	30	200	0.513	0.146
4	0.1	0.9	0.9	0.1	0.5	0.1	1	TCMCT	30	100	0.033	0.153
5	0.1	0.9	0.9	0.1	0.8	0.1	1	TIPCT	200	30	0.545	0.157

Tabla A.14: Primeros 5 mejores juegos de parámetros para métrica F1-Score y RMSE

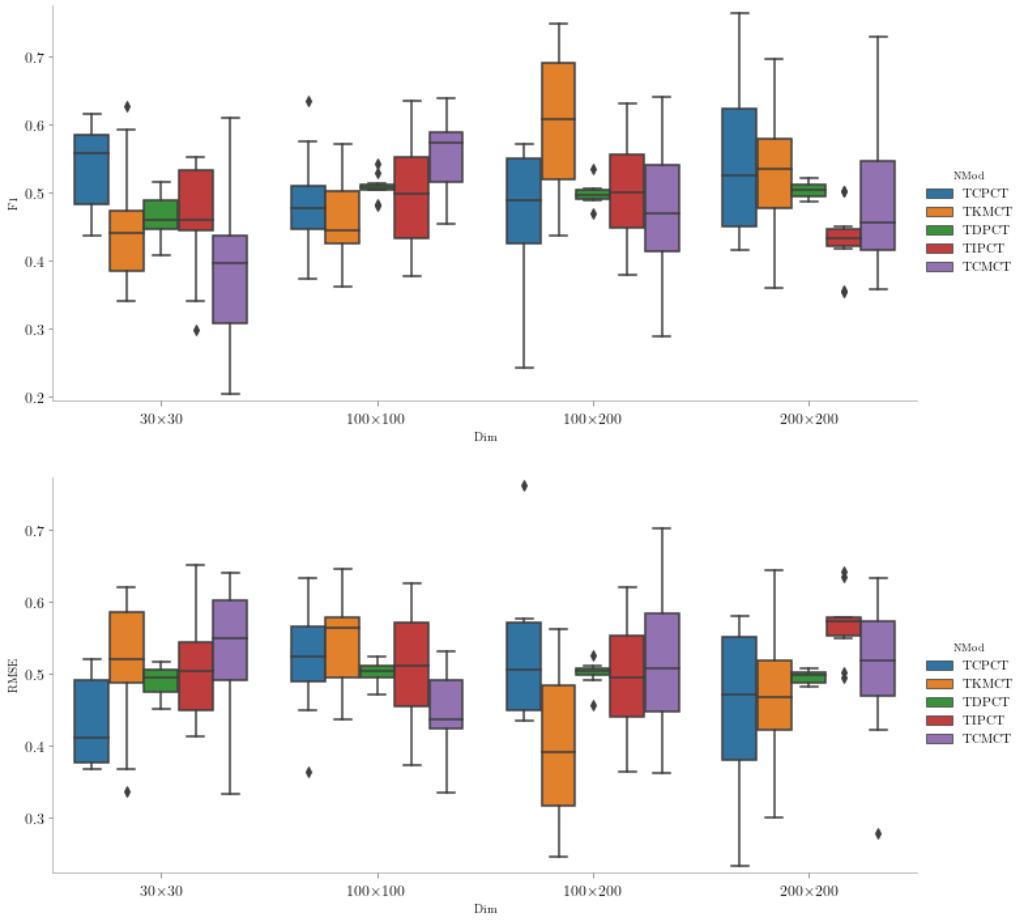


Figura A.13: La gráfica de arriba muestra los boxplots de los resultados de la métrica F1 para 10 días de predicción. La gráfica de abajo muestra los boxplots de los resultados de la métrica RMSE para 10 días de predicción. El eje x representa las diferentes dimensiones de la matriz M que contiene las observaciones. Los parámetros de la prueba son $p_{K_1}(cara|T_A) = 0.1$, $p_{K_1}(cara|T_B) = 0.9$, $p_{K_2}(cara|T_A) = 0.9$, $p_{K_2}(cara|T_B) = 0.1$, $fT_A = 0.5$, $fNK_1 = 0.5$.

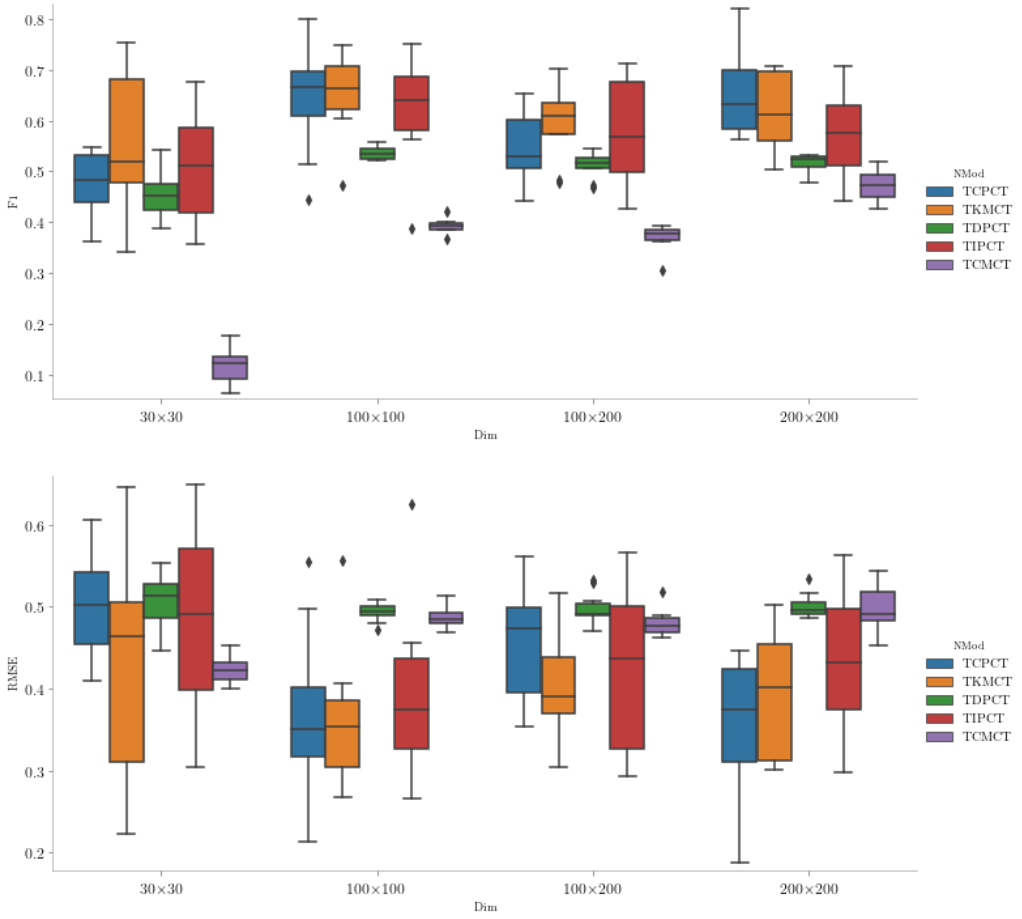


Figura A.14: Los parámetros de la prueba son $p_{K_1}(cara|T_A) = 0.1$, $p_{K_1}(cara|T_B) = 0.9$, $p_{K_2}(cara|T_A) = 0.9$, $p_{K_2}(cara|T_B) = 0.1$, $fT_A = 0.8$, $fNK_1 = 0.5$.

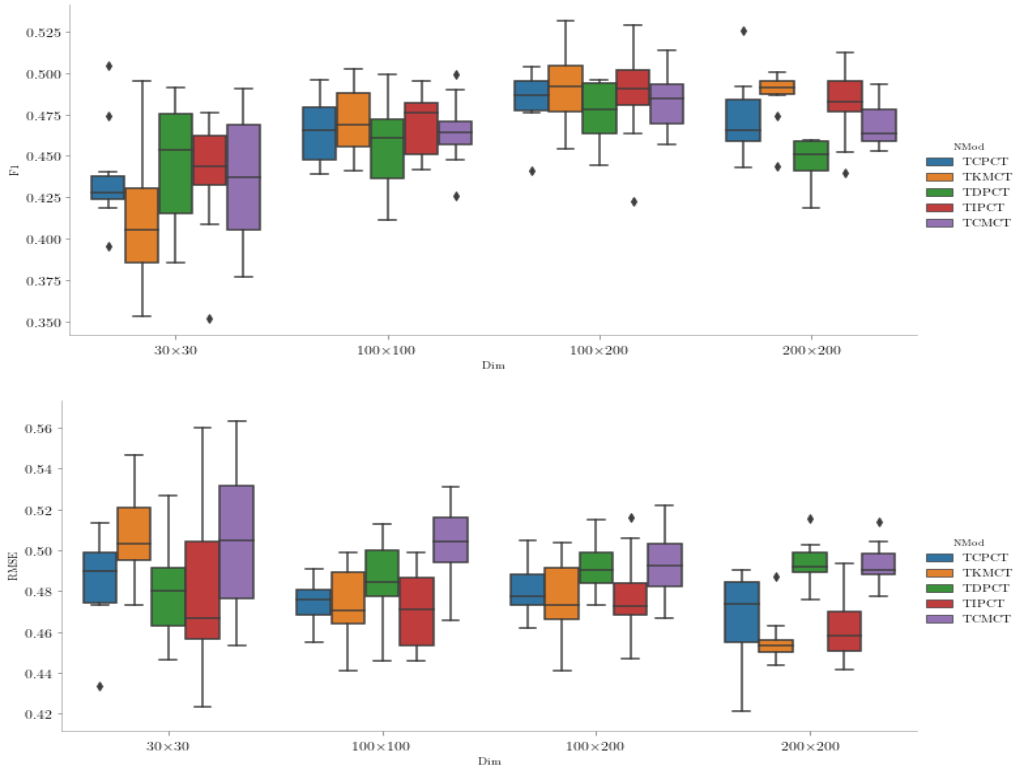


Figura A.15: Los parámetros de la prueba son $p_{K_1}(cara|T_A) = 0.3$, $p_{K_1}(cara|T_B) = 0.6$, $p_{K_2}(cara|T_A) = 0.6$, $p_{K_2}(cara|T_B) = 0.3$, $fT_A = 0.8$, $fNK_1 = 0.5$.

Explanation table

El primer patrón indica que el modelo TCPCT obtuvo un porcentaje bajo (11 %) en métrica F1 superior a TDPCT con una diferencia mayor a 0.15.

El segundo patrón, coincide solamente con los juegos de parámetros con $ID = 1$ en la Tabla 3.12, y reafirma la conclusión de la Sección 3.3.2.4: ante mayor certidumbre en acertar la temperatura para el siguiente día a predecir ($fT_A = 0.8$), las técnicas de clustering (TCPCT, TKMCT), aprovechan mejor la información de los perfiles para superar a TDPCT (33 % de casos positivos).

El tercer patrón coincide con todos los juegos de parámetros en las filas con $ID = 1$ y 2 en la Tabla 3.12, cuya fracción de monedas pertenecientes al perfil K_1 (fNK_1), sea igual a 0.9. La fracción de aciertos es baja en este caso, sobre todo dado por el hecho de que para los parámetros con $ID = 1$ y 2, la mitad o la mayoría de los días tienen temperatura alta T_A: $fT_A = 0.8, 0.5$, respectivamente. Y para la temperatura T_A, la probabilidad de salir cara, del perfil K_1 es: 0.1. Lo

que hace difícil el acierto de en cual de los días saldrá cara, para las monedas de dicho perfil. Esto conlleva a una dismiunción de la métrica F1 y el consiguiente acercamiento de los valores en F1 entre las técnicas TCPCT y TDPCT.
El cuarto patrón, indica que para los juegos de parámetros con $ID = 2$ en la Tabla 3.12, para un día de predicción y para una fracción de monedas del perfil K_1 igual a 0.1 (9 casos), en solo una oportunidad ocurrió que TCPCT fuese mejor que TDPCT, en una diferencia de más de 1.5 puntos en métrica RMSE.
El quinto patrón, indica que para los juegos de parámetros con $ID = 1$ y 2 en la Tabla 3.12, con probabilidad $p_{K_2}(cara|T_B) = 0.1$, en un 17 % de los casos se logró la meta de una diferencia en F1 mayor o igual a 1.5.

A.2.2.4. Comparativa de técnicas conociendo temperaturas, monedas con histórico incompleto

En esta sección, para la **Prueba CT2** especificada en la Sección 3.3.2.5, se muestran los 5 mejores resultados en métrica F1 y RMSE y se presentan los box-plots, con los resultados obtenidos por las diferentes técnicas, para los juegos de parámetros de la Tabla 3.15.

La Tabla A.15 muestra los cinco mejores resultados obtenidos en F1 y RMSE. El valor máximo en F1 es menor y el mínimo en RMSE es mayor en comparación a los conseguidos en el experimento que incluye monedas con histórico completo y que no incluye la temperatura como factor (Tabla A.5, **Prueba ST3**).
Se observa que INC con solo tres días de histórico performa bien, esto se explica nuevamente porque INC con 3 días de histórico puede aproximar bastante bien las probabilidades reales de los perfiles, sobre todo para perfiles muy diferentes, ver la Tabla 3.11 para un ejemplo de esto.

Mejores promedios en F1												
ID	$pK_1(cara\|T_A)$	$pK_1(cara\|T_B)$	$pK_2(cara\|T_A)$	$pK_2(cara\|T_B)$	fT_A	fNK_1	N_{nc}	Mod	N	D	$F1_{prom}$	$RMSE_{prom}$
1	0.1	0.9	0.9	0.1	0.8	0.9	10	INC	30	30	0.910	0.140
2	0.1	0.9	0.9	0.1	0.8	0.9	10	TKMCT	30	30	0.894	0.165
3	0.1	0.9	0.9	0.1	0.8	0.9	10	TCPCT	30	30	0.890	0.168
4	0.1	0.9	0.9	0.1	0.8	0.9	10	TIPCT	30	30	0.887	0.172
5	0.1	0.9	0.9	0.1	0.8	0.9	3	INC	100	30	0.849	0.250
Mejores promedios en RMSE												
ID	$pK_1(cara\|T_A)$	$pK_1(cara\|T_B)$	$pK_2(cara\|T_A)$	$pK_2(cara\|T_B)$	fT_A	fNK_1	N_{nc}	Mod	N	D	$F1_{prom}$	$RMSE_{prom}$
1	0.1	0.9	0.9	0.1	0.8	0.1	10	TCPCT	30	30	0	0.120
2	0.1	0.9	0.9	0.1	0.8	0.9	10	INC	30	30	0.910	0.140
3	0.1	0.9	0.9	0.1	0.8	0.1	3	TCPCT	30	100	0	0.150
4	0.1	0.9	0.9	0.1	0.8	0.9	10	TKMCT	30	30	0.894	0.165
5	0.1	0.9	0.9	0.1	0.8	0.9	10	TCPCT	30	30	0.890	0.168

Tabla A.15: Primeros 5 mejores juegos de parámetros para métrica F1-Score y RMSE

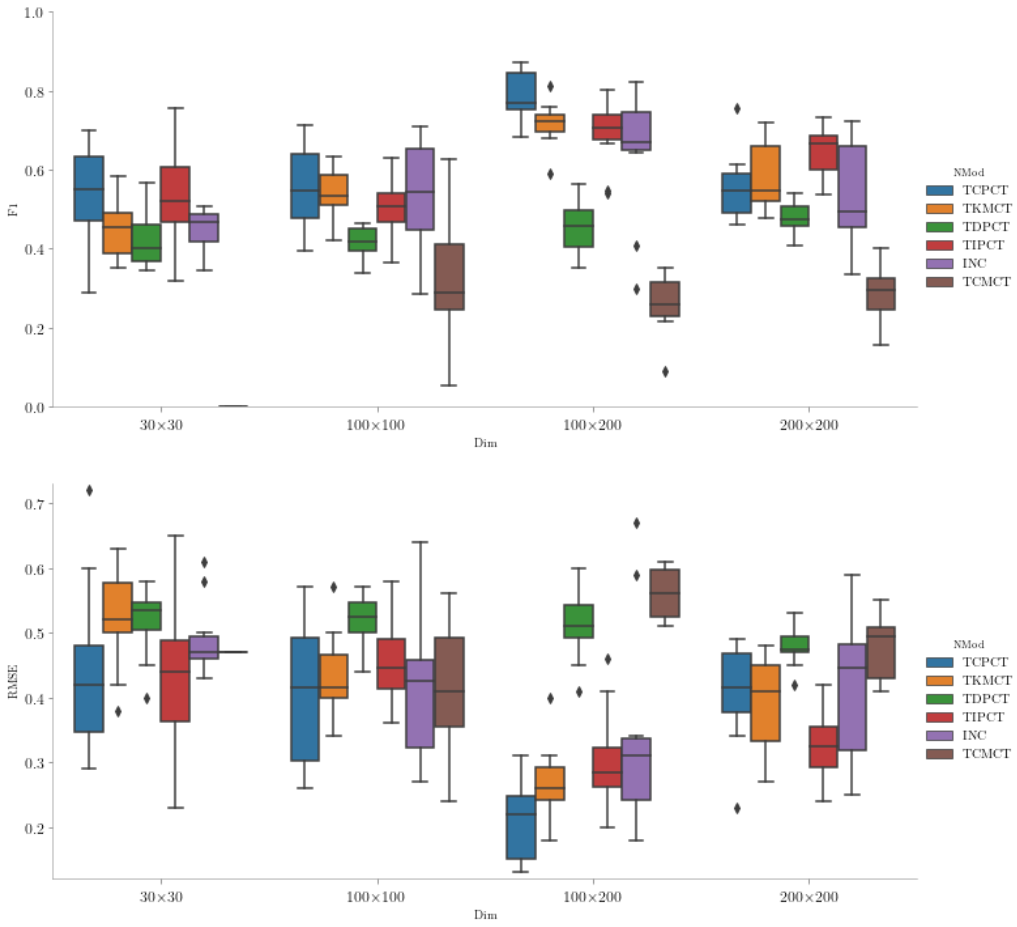


Figura A.16: La gráfica de arriba muestra los boxplots de los resultados de la métrica F1 para monedas con histórico incompleto con 10 días observados para 10 días de predicción. La gráfica de abajo tiene el mismo significado, mostrando los boxplots de los resultados para la métrica RMSE. Los parámetros de la prueba son $p_{K_1}(cara|T_A) = 0.1$, $p_{K_1}(cara|T_B) = 0.9$, $p_{K_2}(cara|T_A) = 0.9$, $p_{K_2}(cara|T_B) = 0.1$, $fT_A = 0.8$, $fNK_1 = 0.5$.

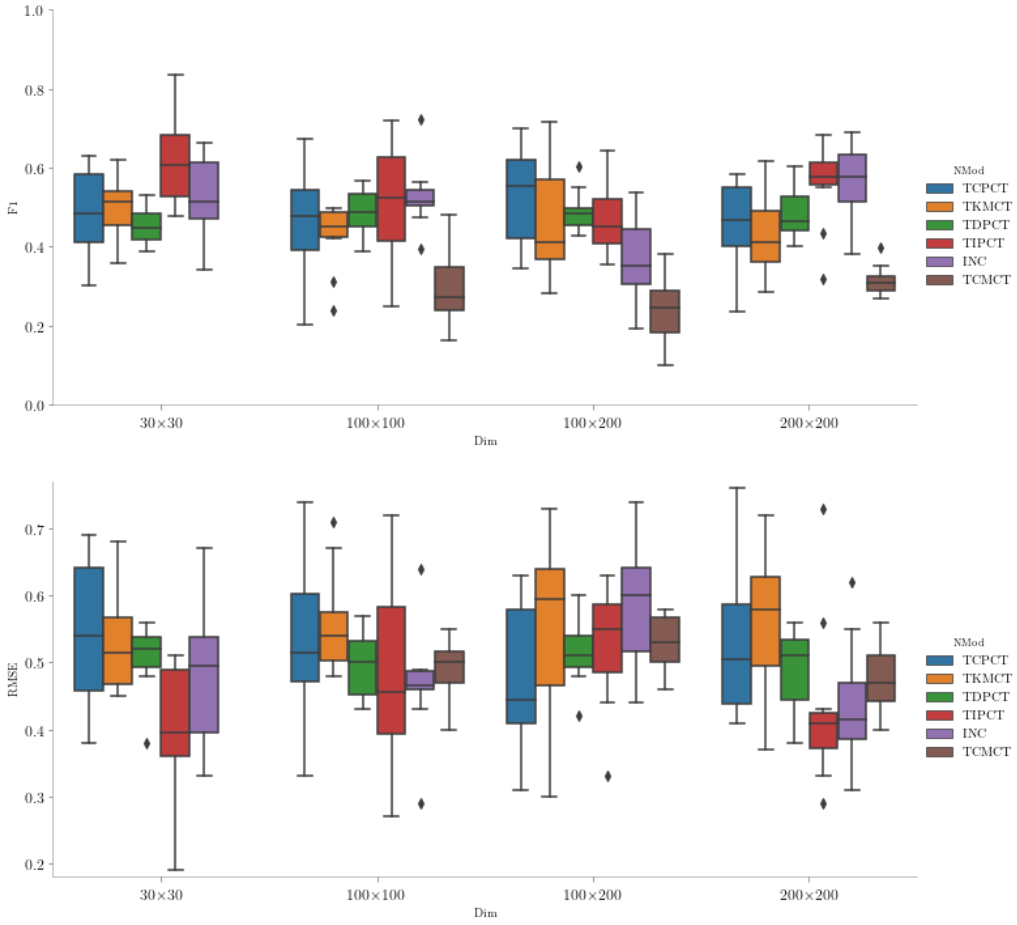


Figura A.17: Los parámetros de la prueba son $p_{K_1}(cara|T_A) = 0.1$, $p_{K_1}(cara|T_B) = 0.9$, $p_{K_2}(cara|T_A) = 0.9$, $p_{K_2}(cara|T_B) = 0.1$, $fT_A = 0.5$, $fNK_1 = 0.5$.

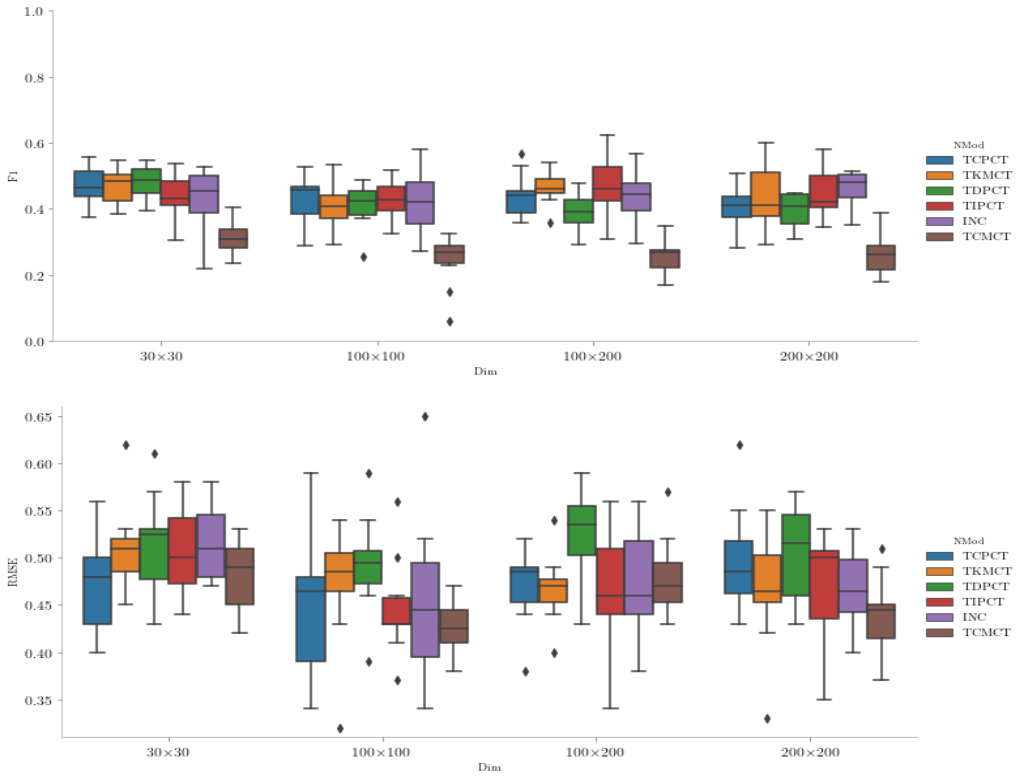


Figura A.18: Los parámetros de la prueba son $p_{K_1}(cara|T_A) = 0.3$, $p_{K_1}(cara|T_B) = 0.6$, $p_{K_2}(cara|T_A) = 0.6$, $p_{K_2}(cara|T_B) = 0.3$, $fT_A = 0.8$, $fNK_1 = 0.5$.

Apéndice B

Detalles del modelo real

B.1. Algoritmo de Laurinec

En esta sección se presenta el algoritmo del método propuesto por Laurinec et al. [12] aplicado a series de tiempo, junto con una explicación de los pasos del algoritmo.
A modo ilustrativo algunas de las variables del problema serán instanciadas.
Sea N el número de clientes, sean $D_e = 21$ los días de entrenamiento (3 semanas), $D_p = 7$ los días a predecir y $frec = 48$ la cantidad de mediciones registradas por día, en este caso una medición cada media hora. El algoritmo realiza una predicción de consumo para el siguiente día, es decir cada técnica devuelve $frec$ pronósticos por iteración, una para cada tiempo a observar del siguiente día.

Algoritmo 8 $(N, D_e, D_p, frec)$

1: $iter \leftarrow 0$
2: **while** $iter < D_p * frec$ **do**
3: Crear una serie de tiempo para cada cliente de tamaño $D_e * frec$ (tres semanas de entrenamiento).
4: Normalizar cada serie de tiempo aplicando z-score (guardando la media y desviación estándar de cada serie de tiempo).
5: Computar una representación para cada serie de tiempo.
6: Clustering usando K-means.
7: Extracción de K centroides.
8: Usar los K centroides como datos de entrenamiento en todos los métodos.
9: Desnormalizar los K pronósticos usando las medias y desviaciones guardadas para cada cliente para así producir los N pronósticos de los clientes.

10: $iter = iter + frec$
11: **end while**

El algoritmo itera produciendo un pronóstico de consumo para el siguiente día. Como las mediciones se toman cada media hora ($frec = 48$), en cada iteración se producen 48 predicciones de consumo para cada pronóstico.
En el paso 3 se crea una serie de tiempo para cada cliente, conteniendo como datos de entrenamiento sus últimas tres semanas de consumo.
El paso 4 de normalización se aplica con el objetivo de clusterizar clientes con patrones similares de consumo. La normalización z-score (Ec. B.1), se desglosa por los términos: $\hat{x}_i$ que es el valor normalizado y x_i el valor original, para $i = 1, \ldots, n$ con n igual al tamaño de la serie de tiempo, y μ y σ que son la media y desviación estándar muestral.

$$\hat{x}_i = \frac{x_i - \mu}{\sigma} \tag{B.1}$$

$$x_i = \hat{x}_i \sigma + \mu \tag{B.2}$$

El paso 5 es el de computación de la representación de cada serie de tiempo, la cual será utilizada como feature para el algoritmo de clasificación. Se emplea una regresión lineal múltiple para extraer los coeficientes de regresión de dos estacionalidades (diaria y semanal) de la serie de tiempo, estos coeficientes serán la representación. Formalmente el modelo puede ser escrito de la siguiente manera:

$$x_t = \beta_{d1} u_{td_1} + \ldots + \beta_{ds} u_{td_s} + \beta_{w1} u_{tw_1} + \ldots + \beta_{w7} u_{tw_7} + \epsilon_t \tag{B.3}$$

para $t = 1, \ldots, n$ donde x_t es el t-ésimo consumo de electricidad. $\beta_{d1}, \ldots, \beta_{ds}$ son los coeficientes de regresión para la estacionalidad diaria. $\beta_{w1}, \ldots, \beta_{w7}$ son los coeficientes de regresión para la estacionalidad semanal. $u_{td_1}, \ldots, u_{td_s}, u_{tw_1}, \ldots, u_{tw_7}$ son variables independientes binarias, sus valores se computan como se muestra en la Tabla B.1. u_{td_i} valdrá 1 cuando el tiempo t esté registrando la i-ésima medición del día y valdrá 0 en otro caso. u_{tw_j} valdrá 1 cuando el tiempo t esté registrando una medición para el j-ésimo día de la semana y valdrá 0 en otro caso. ϵ_t es un error aleatorio teniendo distribución normal $\mathcal{N}(0, \sigma^2)$.
Para cada serie de tiempo se obtienen los coeficientes de regresión:

$$(\beta_{d_1}, \ldots, \beta_{d_s}, \beta_{w_1}, \ldots, \beta_{w_7})$$

los cuales son la representación buscada. El procedimiento más común para obtener un estimativo de los coeficientes es aplicar el método de *mínimos cuadrados*

al sistema de n ecuaciones.
Para resolver el problema de *mínimos cuadrados*, la Ecuación B.3, puede ser reescrita de la siguiente manera:

$$x_t = \beta^T u_t + \epsilon_t \tag{B.4}$$

con $\beta^T = (\beta_{d_1}, \ldots, \beta_{d_s}, \beta_{w_1}, \ldots, \beta_{w_7})$ y $u_t = (u_{td_1}, \ldots, u_{td_s}, u_{tw_1}, \ldots, u_{tw_7})$.
La estimación de los coeficientes β puede realizarse minimizando el error para la Ecuación B.5.

$$\sum_{t=1}^{n} \epsilon_t = \sum_{t=1}^{n} (x_t - \beta^T u_t)^2 \tag{B.5}$$

siendo n la cantidad de series de tiempo, cada una asociada al consumo observado de un cliente.
Diferenciando la Ecuacion B.5 respecto al vector β, se llega a la estimación:

$$\hat{\beta} = (\sum_{t=1}^{n} u_t^T u_t)^{-1} \sum_{t=1}^{n} u_t x_t \tag{B.6}$$

que es la solución para el método de *mínimos cuadrados*.

t	Frec. Diaria	Frec. Sem.	u_{td_1}	u_{td_2}	u_{td_3}	u_{td_4}	u_{tw_1}	u_{tw_2}	u_{tw_3}	u_{tw_4}	u_{tw_5}	u_{tw_6}	u_{tw_7}
1	1	1	1	0	0	0	1	0	0	0	0	0	0
2	2	1	0	1	0	0	1	0	0	0	0	0	0
3	3	1	0	0	1	0	1	0	0	0	0	0	0
4	4	1	0	0	0	1	1	0	0	0	0	0	0
5	1	2	1	0	0	0	0	1	0	0	0	0	0
6	2	2	0	1	0	0	0	1	0	0	0	0	0
7	3	2	0	0	1	0	0	1	0	0	0	0	0
...	...	...	...										

Tabla B.1: En este ejemplo la frecuencia de mediciones diarias es de 1 cada 6 horas por lo que hay 4 variables independientes para representar los 4 tiempos en que se toman las mediciones en el día. Las variables independientes para representar el día de la semana en que se toman las mediciones son 7.

En el paso 6 se aplica K-means tomando como features para los clientes las representaciones del paso anterior. La elección de la cantidad óptima de clusters se realiza eligiendo sobre un rango de valores, aquel que dé mejor resultado en métrica Davies-Bouldin, esta métrica es utilizada para evaluar internamente los clusters finales generados por una técnica de clustering; penalizando la dispersión

interna de cada cluster y el parecido entre dos clusters (igual dispersión) si estos son cercanos. En la Sección C.2.2 se encuentra la descripción de esta métrica y sus propiedades.
En el paso 7 se guardan los K centroides que serán la entrada para las técnicas de predicción utilizadas.
En el paso 8, una vez extraídos los K centroides, se emplean diferentes modelos de series de tiempo para predicción:

- Seasonal Naive (SNAIVE). El valor predicho es igual al valor observado en la última estación que coincide con el mismo momento estacional del valor a predecir. En este caso los valores futuros son iguales a los valores observados para los mismos tiempos de la última semana.

- Regresión Lineal Múltiple, MLR por sus siglas en inglés. Se define un modelo como en Ec B.3, pero en este caso los atributos diarios y semanales interactúan unos con otros, así que en lugar de tener $d_s + w_7$ atributos se tienen $d_s \times w_t$ números de interacciones.

$$x_t = \sum_{i=1}^{s} \sum_{j=1}^{7} \gamma_{ij} u_{td_i} \cdot u_{tw_j} + \epsilon_t \tag{B.7}$$

- Random Forest (RF). Es un método combinado de aprendizaje, en donde un número grande de modelos no correlacionados predicen un valor cada uno, siendo el resultado devuelto el promedio de las predicciones. *RF* es utilizado para predecir la parte estacional de la serie de tiempo de cada cliente y el modelo autoregresivo lineal *ARIMA* (Secc. D.5) para predecir la parte no estacional. Véase la Sección D.7 para una descripción más detallada del funcionamiento de RF.

- Triple Exponential Smoothing (ES). Es un método de forecasting en donde las observaciones pasadas tienen un peso asociado que decrece exponencialmente con el tiempo. Se tienen tres modelos: modelo aditivo simple con una componente de tendencia, modelo aditivo con tendencia amortiguada y modelo aditivo sin tendencia. En cada predicción el resultado del mejor modelo es el utilizado como predicción. Para determinar cual de los tres resultados es el mejor, se emplea el criterio Akaike Information Criterion, AIC (Sección D.4.2) el cual evalúa el residuo de cada modelo ajustado a los datos y penaliza la cantidad de parámetros. Véase la Sección D.6 para tener una expresión del modelo *exponential smooth* con tendencia, tendencia amortiguada y sin tendencia.

El método *SNAIVE* fue empleado también como benchmark, para hacer predicciones individuales según el histórico de cada cliente, sin clustering.
El paso 9 de desnormalización es dado por la Ecuación B.2 y se emplea para pasar de la predicción hecha para el cluster k, a una prediccion para los clientes que pertenecen a dicho cluster.
El paso 10 termina la iteración y se pasa al siguiente día de pronóstico, $iter = iter + frec$.

B.2. Algoritmo de la Técnica de la Media

El Algoritmo 9 describe como esta técnica implementa el método propuesto en la Sección 5.1. Luego del algoritmo síguese una explicación detallada de cada uno de sus pasos. Además de los parámetros $frec$ y D_p, definidos por el método propuesto, se tienen los siguientes: **1)**: $kmIter$ define el número de iteraciones para la reclasificación de los clusters usando K-means, **2)** $clTamMax$: define el tamaño máximo de los clusters, si un cluster, en un paso de reclasificación, tiene más de $clTamMax$ clientes agrupados, entonces debe ser reclasificado en k clusters para reducir su tamaño.
Sin pérdida de generalidad se asume que el consumo de los N_{nc} clientes con consumo incompleto se encuentra en las últimas N_{nc} filas de la matriz de consumo M.

Algoritmo 9 $TM(frec, D_p, kmIter, clTamMax)$

1: //Inicialización de variables
2: Inicializar matriz de consumo $M_{N \times D_e * frec}$.
3: Inicializar arreglo de temperaturas $T_{D_e * frec}$.
4: $N_c \leftarrow$ contar cuantos clientes tienen $D_e * frec$ observaciones en $M_{N \times D_e * frec}$.
5: **for** $iter \leftarrow 1; iter \leq D_p; iter = iter + 1$ **do**
6: //Clasificar temperaturas
7: $(E_T, C_T) \leftarrow Kmeans(T_{D_e * freq}, cardinalidad = 2)$. $C_T = \{1, 2\}$, conjunto de etiquetas identificativas de los dos clusters, determinados por la cardinalidad igual a 2. $E_T[i] \in C_T$, $i \in D_e * frec$, indica a que cluster fue etiquetada la i-ésima temperatura.
8: //Predicción y etiquetado de las temperaturas para el nuevo día
9: $(p, d, q, P, D, Q, m) \leftarrow obtenerOrdenes(T_{D_e * frec})$
10: $T_{pred} \leftarrow SARIMA(T_{D_e * frec})(p, d, q) \times (P, D, Q)m$, $|T_{pred}| = frec$
11: Etiquetar a cada temperatura en T_{pred}, en uno de los clusters C_T. Sea c_1, el centroide del cluster con etiqueta 1 y c_2 el centroide del cluster con

etiqueta 2. Si la temperatura predicha $T_{pred}[i], 1 \leq i \leq frec$ cumple que $|T_{pred}[i] - c_1| \leq |T_{pred}[i] - c_2|$ entonces $T_{pred}[i]$ pertenece al cluster 1, sino $T_{pred}[i]$ pertenece al cluster 2. Las etiquetas se guardan en el arreglo $E_{T_{pred}}$. $E_{T_{pred}}[i] \in C_T, i \in frec$, indica a que cluster pertenece la temperatura predicha i.

12: //Clasificar clientes

13: $(E_{N_c}, C_{N_c}) \leftarrow$ inicializar con un solo cluster para los N_c clientes con histórico completo.

14: **for** $i \leftarrow 1; i \leq kmIter; i = i + 1$ **do**

15: **for** $cl \in C_{N_c}$ **do**

16: $cantEtcl \leftarrow$ cantidad de etiquetas asociadas al cluster cl en E_{N_c}.

17: **if** $cantEtcl > clTamMax$ **then**

18: $M_{cl} \leftarrow$ obtener la submatriz de $M_{N_c \times D_e * frec}$ con las filas asociadas al cluster cl.

19: $(k, features) \leftarrow obtenerFeaturesOptm(M_{cl})$

20: $E_{cl} \leftarrow Kmeans(features, cardinalidad = k)$. Obtener etiquetas a partir de la clasificación.

21: Actualizar las etiquetas en E_{N_c} asociadas al cluster cl, con las nuevas etiquetas E_{cl}.

22: **end if**

23: **end for**

24: **end for**

25: //Etiquetado de clientes con histórico incompleto

26: $E_{nc} \leftarrow$ inicializar arreglo de etiquetas para clientes con histórico incompleto.

27: $M_{nc} \leftarrow$ obtener matriz con los históricos incompletos.

28: **for** i numero de fila en M_{nc} **do**

29: $t_{nc_i} \leftarrow$ obtener los tiempos para los cuales hay observación de consumo para el cliente i

30: Sea $M[E[j], t_{nc_i}]$ la submatriz de todos los clientes, con histórico completo, con etiqueta j y con solo las columnas asociadas a los tiempos observados para el cliente con histórico incompleto i. Sea $\mu_{M[E[j],t_{nc_i}]}$ el promedio por columnas de la matriz $M[E[j], t_{nc_i}]$. Sea $M_{nc}[i, t_{nc_i}]$ la fila de M_{nc} con los consumos observados para el cliente i. La etiqueta para el cliente i se elije de la siguiente manera: $E_{nc}[i] = argmin_j |M_{nc}[i][t_{nc_i}] - \mu_{M[E[j],t_{nc_i}]}|$.

31: **end for**

32: **for** i en $frec$ **do**

33: **for** cl en Cl **do**

34: //Aprendizaje
35: $M_{pred} \leftarrow$ Dada la matriz $M_{N \times D_e * frec}$. Obtener de las $D_e * frec$ columnas de las filas de los clientes con histórico completo que pertenezcan al cluster *cl*, aquellas que coincidan con el tiempo *i* que se está queriendo predecir y cuya temperatura para ese tiempo tenga la misma etiqueta que la asociada a la temperatura predicha para el tiempo *i*.
36: $M^{hc} \leftarrow$ obtener de M_{pred}, las filas correspondientes a los clientes con histórico completo.
37: $M^{hi} \leftarrow$ obtener de M_{pred}, las filas correspondientes a los clientes con histórico incompleto.
38: $c_{hc} \leftarrow$ valor medio $\left(M^{hc}\right)$
39: $c_{hi} \leftarrow$ valor medio $\left(M^{hi}\right)$
40: //Predicción
41: **for** *cli* cliente del cluster *cl* **do**
42: **if** *cli* es cliente con histórico completo **then**
43: El consumo de *cli* predícese como c_{hc}.
44: **else**
45: El consumo de *cli* predícese como c_{hi}.
46: **end if**
47: **end for**
48: **end for**
49: **end for**
50: $De = De + 1$
51: Agregar a las nuevas $frec$ columnas en $M_{N \times De * frec}$ los valores de consumo reales de cada cliente.
52: Agregar a las nuevas $frec$ entradas en $T_{D_e * frec}$ los valores de las temperaturas reales del día.
53: **end for**

Inicialización de variables

En los pasos 2, 3 y 4 del algoritmo se inicializan: la matriz de entrenamiento $M_{N \times D_e * frec}$ y el arreglo de temperaturas $T_{D_e * frec}$. Se calculan: los N_c clientes que tienen el histórico de observaciones completo.

Clasificar temperaturas

En el paso 7 se realiza una clasificación usando K-means de $T_{D_e * frec}$ en 2 clusters. La feature de entrada a K-means, de cada temperatura es la temperatura misma. La clasificación se puede pensar como un agrupamiento de temperaturas altas por un lado y bajas por el otro. Por ejemplo, en la Figura B.1 se observa una clasi-

ficación usando Kmeans para los primeros 28 días del mes de Abril del 2013 en Newcastle.
La fijación de la cardinalidad en 2 es porque en el bloque de *Aprendizaje* solo son empleadas las columnas $d \in \{1, \ldots, D_e * frec\}$ de la matriz $M_{N \times D_e * frec}$ que cumplan que la temperatura $T_{D_e * frec}[d]$ esté en el mismo cluster que la temperatura predicha para el momento del día para el cual se está prediciendo. Una consecuencia de esto es que mientras más clusters k haya para las temperaturas, menos columnas de consumo estarán asociadas a cada cluster, pues las columnas deben ser repartidas entre los k clusters de temperaturas. Conllevando a una menor cantidad de información para utilizar en la etapa de aprendizaje.
Predicción y etiquetado de las temperaturas para el nuevo día
En el paso 9 se obtienen los ordenes para el modelo autoregresivo SARIMA Secc. D.5.6. La elección de cada orden se hace eligiendo del siguiente conjunto de valores:

- $p, q, P, Q \in \{0, 1, 2\}$, p y P son los ordenes autoregresivos, q y Q los ordenes de media móvil, globales y estacionales, respectivamente. Se admiten hasta dos términos de cada uno.

- $d, D \in \{0, 1\}$, d y D son los ordenes de diferenciación global y estacional, respectivamente. Se permite una diferenciación de hasta orden 1, para transformar un proceso no estacionario en uno estacionario o debilmente estacionario (Secc. D.2).

- $m \in \{0, frec, 7 * frec\}$, m determina la frecuencia estacional. El valor 0 no admite estacionalidad en el modelo. $frec$ contempla una estacionalidad diaria. $7 * frec$ una estacionalidad semanal.

Se define un modelo SARIMA para cada combinación de valores y se ajusta a la serie de tiempo de temperaturas $T_{D_e * frec}$. Para cada modelo se calcula la métrica AIC (Secc. D.4.2), para determinar el juego de parámetros óptimo del conjunto de combinaciones posibles. Los ordenes (p, d, q, P, D, Q, m) elegidos son aquellos que ajusten los datos con el menor valor AIC. En la Figura B.2 se observa la aplicación de SARIMA para predecir la temperatura durante la cuarta semana de Abril del 2013 en Newcastle, un día a la vez, siguiendo la metodología explicada en este paso.
En el paso 10 se pronostícan las $frec$ temperaturas para el nuevo día a predecir.
En el paso 11 se clasifican las temperaturas predichas en uno de los dos clusters determinados en el paso 7. Cada temperatura es etiquetada al cluster, cuyo centroide sea el más cercano a ella. Utilízase la distancia euclídea como función de

distancia entre la temperatura y cada centroide.
En la Tabla B.2 se muestran las métricas para los pasos de pronóstico y etiquetado, para las temperaturas predichas durante la cuarta semana del mes de Abril del 2013. El proceso de etiquetado, de las temperaturas predichas, arrojó un valor F1-Score de 0.835.

Clasificar clientes

El bloque de clasificación de clientes es el comprendido por los pasos del 13 al 24.
En primera instancia en el paso 13: $c_{N_c} = \{1\}$ y por lo tanto se inicializa un arreglo de etiquetas E_{N_c} de tamaño N_c, con una única etiqueta asociada al cluster único inicial. Durante las *kmIter* iteraciones se irán reagrupando los clientes alrededor de nuevos clusters. Un cluster actual es reclasificado en k nuevos clusters si cumple la condición dada en el paso 17. Si la cantidad de clientes del cluster supera el máximo de clientes $clTamMax$.
En el paso 18 para cada cluster cl se obtiene la submatriz M_{cl}, con las filas de los clientes que pertenecen a cl.
En el paso 19 se obtienen la cardinalidad y las features óptimas para clasificar usando K-means. Los valores posibles para la cardinalidad y las features son los siguientes:

- $k \in \{2, \ldots, 6\}$
- $features \in \{cDiario, dDiaria, dSemanal\}$
  - $cDiario = [\mu_1, \ldots, \mu_{frec}, \sigma_1, \ldots, \sigma_{frec}]$.
    Cada par $\mu_i, \sigma_i, i \in frec$ representa el consumo medio diario y la desviación estándar media diaria para los valores registrados en el tiempo del día i, durante los D_e días de entrenamiento.
  - $dDiaria = [\mu_1, \ldots, \mu_{frec}, \sigma_1, \ldots, \sigma_{frec}, tend_1, \ldots, tend_{frec}, est_1, \ldots, est_{frec}, corrCoef(T_{D_e * frec}, consumo(j))]$.
    Cada par $\mu_i, \sigma_i, i \in frec$ se interpreta idem que en $cDiario$.
    $tend_i, est_i, i \in frec$ representan la tendencia media diaria y la estacionalidad media diaria para los valores registrados en el tiempo del día i, durante los D_e días de entrenamiento.
    Dada la descomposición aditiva de la serie de tiempo del cliente j:
    $consumo(j) = tendencia(j) + estacionalidad(j) + ruido(j)$.
    El cálculo de $tend_i$ y est_i es igual a μ_i. Con la diferencia que los μ_i se calculan a partir de la serie $consumo(j)$, los $tend_i$ a partir de la serie $tendencia(j)$ y los est_i a partir de $estacionalidad(j)$.

$corrCoef(T_{D_e*frec}, consumo(j))$ es el coeficiente de correlación entre las temperaturas y el consumo del usuario j durante los tiempos de entrenamiento.

- $dSemanal = [\mu_1, \ldots, \mu_{7*frec}, \sigma_1, \ldots, \sigma_{7*frec}, tend_1, \ldots, tend_{7*frec}, est_1, \ldots, est_{7*frec}, corrCoef(T_{D_e*frec}, consumo(j))]$.
  El significado de cada feature en $dSemanal$ es igual al de las features en $dDiario$. La diferencia es que la tendencia, estacionalidad y consumo medio son semanales ($7 * frec$) en lugar de diarias ($frec$).

La elección del mejor par (k, $features$) se hace eligiendo aquella combinación que menor valor dé en métrica Davies-Bouldin (Secc. C.2.2).
En el paso 20 se ejecuta K-means con el par cardinalidad-features elegido.
En el paso 21 se actualizan las etiquetas en E_{Nc} asociadas a los clientes pertenecientes al cluster cl, que fuese reclasificado en el paso anterior, dando paso a la creación de k nuevos clusters a partir de cl. En la Figura B.3 se observan los resultados finales de aplicar los pasos de clasificación de clientes (13 al 24) a 325 clientes, durante los primeros 28 días del mes de Abril del 2013. En la Figura B.4 se muestra como a partir de un cluster inicial, se crean los clusters finales de la figura anterior.

Etiquetado de clientes con histórico incompleto

El bloque de etiquetado para los N_{nc} clientes con histórico incompleto se lleva a cabo entre los pasos 26 y 31.
En el paso 30 se determina la etiqueta de cada cliente con histórico incompleto, comparando el consumo del cliente a etiquetar con el consumo promedio de cada cluster para los tiempos observados del cliente. Se elige el cluster, con el centroide de consumo más cercano al consumo del cliente. La función de distancia empleada es la distancia euclídea.

Aprendizaje

Para cada cluster en el paso 35 se obtiene la submatriz a partir de la cual la técnica aprenderá las probabilidades. En las Tablas 5.1 y 5.2 se bosqueja el proceso de obtención de la submatriz M_{pred}.
En los pasos del 36 al 39, esta técnica a partir de la submatriz M_{pred} obtiene, para cada tipo de cliente, el valor medio de consumo.

Predicción

En los pasos 43 y 45 se realiza la predicción del consumo para ambos tipos de clientes, valiendo c_{hc} si el cliente tiene el histórico de consumo completo, o c_{hi} en caso contrario.

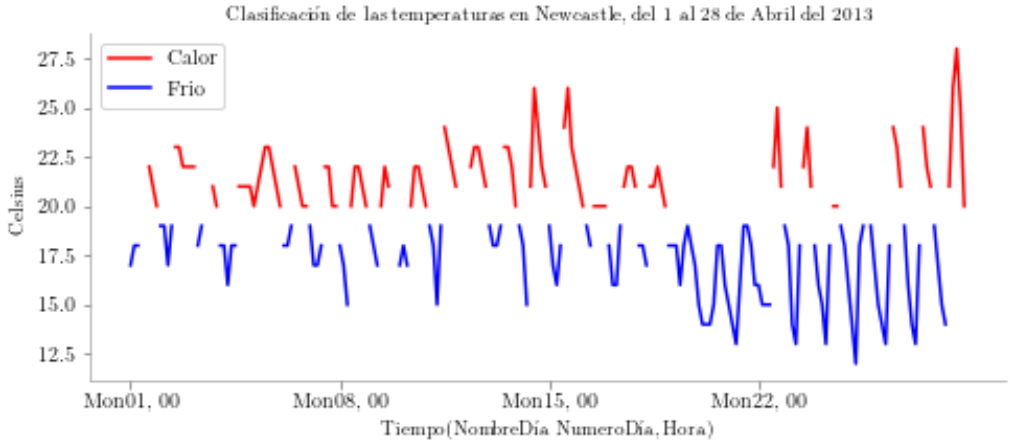


Figura B.1: Las temperaturas mayores o iguales a $20°C$ pueden ser interpretadas como temperaturas *templadas tendiendo a calurosas*, y aquellas de menos de $20°C$ se pueden interpretar como temperaturas *templadas tendiendo a frías*

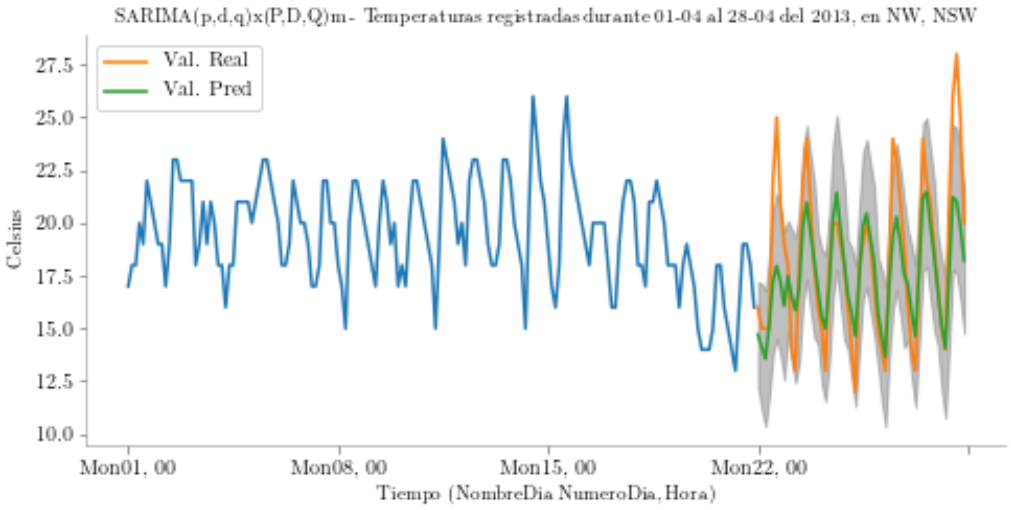


Figura B.2: Para cada uno de los 7 días de predicción se elige la combinación de ordenes (p, d, q, P, D, Q, m) con menor valor en métrica AIC.

Dia	p	d	q	P	D	Q	m	AIC
1	1	1	1	1	0	1	$frec$	574.4
2	1	1	1	1	0	1	$frec$	616.7
3	1	1	1	1	0	1	$frec$	650.1
4	1	1	1	1	0	2	$frec$	674.6
5	1	1	1	1	0	2	$frec$	701.3
6	1	1	1	1	0	2	$frec$	731.3
7	1	1	1	1	0	2	$frec$	757.7
RMSE: 5.731								
MAE: 1.740								
F1-Score: 0.835								

Tabla B.2: Cada fila muestra los ordenes elegidos para cada día de predicción, para la ejecución de SARIMA(p,d,q)×(P,D,Q)m que generaron los resultados en la Figura B.2. Las grandes diferencias entre los picos máximos de consumo y el valor para ellos predicho hacen que el valor de RMSE escale a un error de 5.731 durante la semana de predicción. El Error Medio Absoluto fue de casi $2°C$ en promedio. El valor 0.835 en F1-Score mide que tan bien fueron etiquetadas las temperaturas predichas, respecto a las etiquetas de las temperaturas reales, durante el paso 11 de etiquetado del Algoritmo 9.

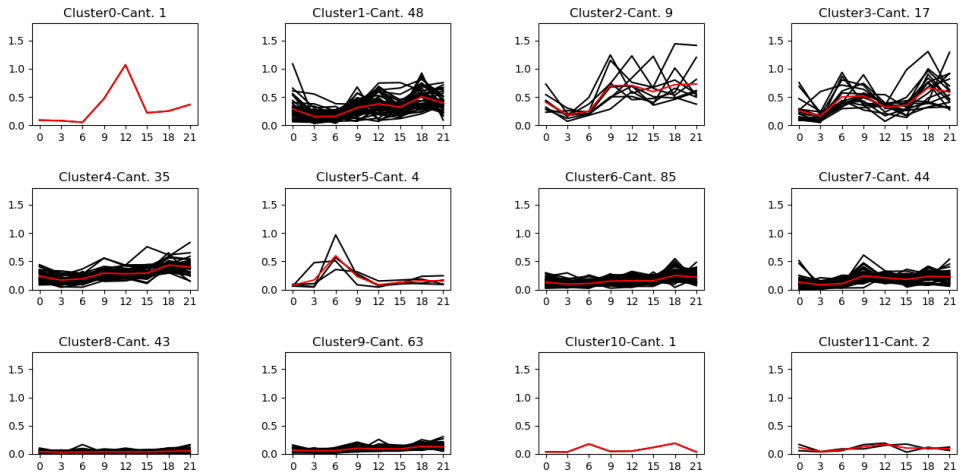


Figura B.3: La Figura muestra el consumo promedio diario en KWh, cada tres horas, para todos los usuarios pertenecientes a cada cluster. En rojo se ve el consumo promedio diario de todos los clientes de cada cluster.

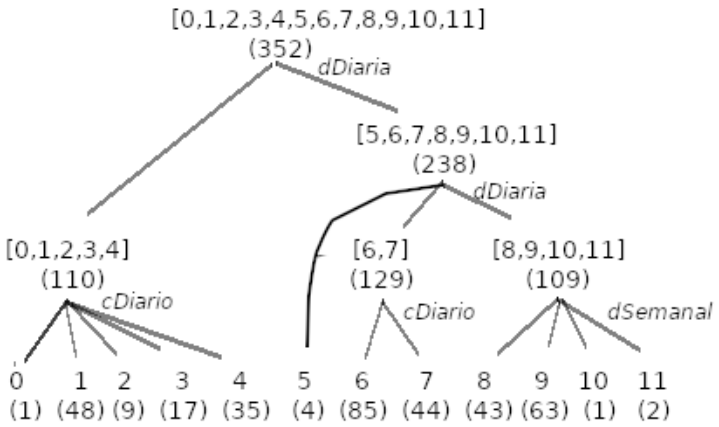


Figura B.4: La Figura muestra una estructura de árbol. Cada nodo representa un cluster. La altura a la que se encuentra cada nodo, representa en cual de las *kmIter* iteraciones fue creado el cluster asociado al nodo. En este caso la profundidad del árbol es igual a 3 ya que para este ejemplo $kmIter = 3$. El nodo raíz es el cluster inicial. Las hojas son los clusters finales representados en la Figura B.3. Entre paréntesis se detalla el número de clientes en cada nodo. La cantidad de hijos de un nodo representa la cardinalidad k con la cual se subdividió el cluster asociado. Las etiquetas $cDiario, dDiaria, dSemanal$ debajo de cada nodo padre indican con que feature se realizó la subdivisión. El tamaño máximo de clientes por cada cluster en el ejemplo ($clTamMax$) es igual a 50. Obsérvese que los clusters finales número 6 y 9 terminaron el proceso de clasificación con más de 50 clientes. Esto es debido a que la reclasificación de los cluster con más de $clTamMax$ clientes se detuvo en la tercera iteración ($kmIter = 3$). Los clusters 6 y 9 requieren una cuarta iteración para poder ser reclasificados y generar clusters hijos con menor cantidad de clientes.

Apéndice C

K-means

La Sección 2.2.1 introdujo la técnica de clustering K-means, junto con una definición del problema que K-means resuelve. En este apéndice se describe el Algoritmo 10; una técnica gráfica y otra analítica para elegir una cardinalidad óptima para el algoritmo y las propiedades directas del algoritmo K-means. El contenido de esta sección está basado en las definiciones y propiedades detalladas en el Capítulo 16 del libro escrito por Christopher D. et al. [3].

C.1. Algoritmo K-means

El algoritmo toma como entrada N objetos $\{\vec{x}_1, \ldots, \vec{x}_N\}$ para clasificar en K clusters.

Algoritmo 10 K-MEANS($\{\vec{x}_1, \ldots, \vec{x}_N\}, K$) [3, Cap.16.4]

1: $(\vec{s}_1, \ldots, \vec{s}_K) \leftarrow$ elegir K semillas al azar de $\{\vec{x}_1, \ldots, \vec{x}_N\}$
2: **for** $k \leftarrow 1$ **to** K **do**
3: $\quad \vec{u}_k \leftarrow \vec{s}_k$
4: **end for**
5: **while** criterio de parada aún no satisfecho **do**
6: $\quad \omega_k \leftarrow \{\}$
7: $\quad$ **for** $n \leftarrow 1$ **to** N **do**
8: $\quad\quad j \leftarrow argmin_{j'} |\vec{u}_{j'} - \vec{x}_n|$
9: $\quad\quad \omega_j \leftarrow \omega_j \cup \{\vec{x}_n\}$ //Reasignación de vectores
10: $\quad$ **end for**
11: $\quad$ **for** $k \leftarrow 1$ **to** K **do**

12: $\vec{u}_k \leftarrow \frac{1}{|\omega_k|} \sum_{\vec{x} \in \omega_k} \vec{x}$ //Recomputación de vectores
13: **end for**
14: **end while**
15: **return** $\{\vec{u}_1, \ldots, \vec{u}_K\}$

El algoritmo comienza en el paso 1 eligiendo K objetos al azar $(\vec{s}_1, \ldots, \vec{s}_K)$ de $\{\vec{x}_1, \ldots, \vec{x}_N\}$.
En el bloque de repetición dado por los pasos del 2 al 4, se inicializan los K centroides $(\vec{u}_1, \ldots, \vec{u}_K)$ mediante la regla $\vec{u}_k = \vec{s}_k$.
El bloque de código del `while`, paso 5, se repite mientras el criterio de parada no haya sido cumplido.
En el paso 6, se inicializan los ω_k clusters, para los $\vec{u}_k$ centroides.
En el bloque de repetición dado por los pasos del 7 al 10, se le asigna a cada objeto $\vec{x}_i$ el cluster ω_k cuyo centroide $\vec{u}_k$ sea el más cercano en distancia euclídea a $\vec{x}_i$.
Luego de asignar (o reasignar) a cada objeto un cluster, en el bloque de repetición dado por los pasos del 11 al 13 se recomputan los centroides según la fórmula $\vec{u}_k = \frac{1}{\omega_k} \sum_{\vec{x} \in \omega_k} \vec{x}$.

C.2. Cardinalidad en K-means

La cardinalidad de un algoritmo de tipo *flat clustering* es el número k de clusters. En K-means éste es ingresado como parámetro de entrada al Algoritmo 10. Una pregunta a responder es: si k es un parámetro de entrada ¿cómo determinar un valor óptimo o plausible para k? Una primera aproximación podría ser seleccionar el valor óptimo de k que minimice la función objetivo RSS (Ec. 2.15). Sin embargo $\widehat{RSS}_{min}(k)$ (Ec. 2.14) es monótona decreciente en k Proposición C.3.3 y por lo tanto el mínimo 0 se alcanza cuando $k = N$, igual a la cantidad de objetos que hay que clasificar.

C.2.1. Método gráfico para encontrar k óptimo

El método consiste en los siguientes pasos:

- Realizar i (ej $i = 10$) clusterings con K-means con cardinalidad k
- Definir $\widehat{RSS}_{min}(k)$ como el valor mínimo de RSS para las i pruebas realizadas.
- Inspeccionar los valores de $\widehat{RSS}_{min}(k)$ a medida que k aumenta.

- Encontrar los puntos a partir de los cuales los decrementos de la curva se vuelven considerablemente más pequeños. Ver Figura C.1.

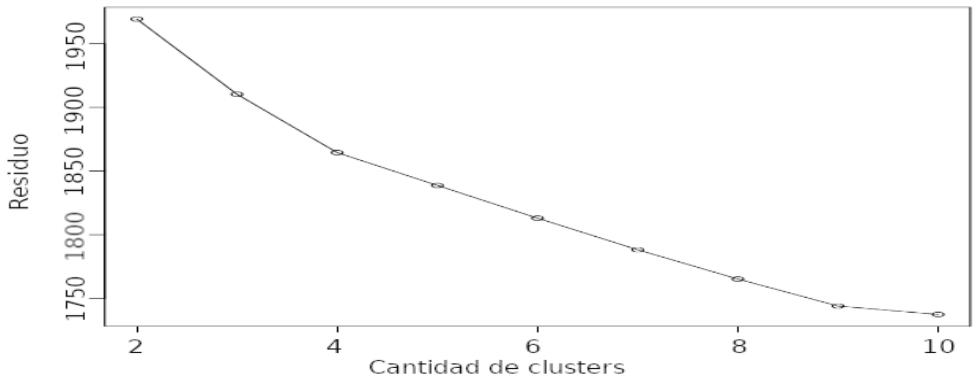


Figura C.1: Observar que para $k = 4{,}9$ ocurre una disminución considerable en los decrementos de la curva para los siguientes valores de k. Figura extraída de la página 366, capítulo 16, Manning et al. [3].

C.2.2. Métrica Davies-Bouldin

Esta métrica fue presentada en el trabajo de Laurinec et al. [4], en su computación se penaliza la dispersión de los objetos alrededor del cluster (comparación intra-cluster) y se penaliza la cercanía entre los clusters (comparación inter-clusters). La métrica refuerza la idea de que dos clusters no deben ser similares.
La medida de dispersión se define como:

$$S_i = \{\frac{1}{T_i}\sum_{j=1}^{Ti} |X_j - A_i|^p\}^{\frac{1}{q}}$$

en donde T_i es la cantidad de elementos en el cluster i, A_i es el centroide del cluster i y los X_j son los objetos agrupados alrededor del cluster A_i.
Mientras que la distancia entre clusters se define como:

$$M_{ij} = \{\sum_{k=1}^{N} |a_{ki} - a_{kj}|^p\}^{\frac{1}{p}}$$

en donde a_{ki} es la k-ésima componente del vector N dimensional a_i, el cual es el centroide del cluster i.

La medida se define así:

$$\hat{R} = \frac{1}{N} \sum_{i=1}^{N} R_i$$

con R_i igual al máximo R_{ij}:

$$R_{ij} = \frac{S_i + S_j}{M_{ij}}, j \neq i$$

$\hat{R}$ tiene el significado de ser el promedio de similitud de cada cluster y su cluster más similar. La mejor elección del cluster por lo tanto será aquella que minimice la similitud promedio.
Las propiedades directas de R_{ij} son las siguientes:

1. $R_{ij} \geq 0$
2. $R_{ij} = R_{ji}$
3. Si $S_j = S_k$ y $M_{ij} < M_{ik} \rightarrow R_{ij} > R_{ik}$
4. Si $M_{ij} = M_{ik}$ y $S_j > S_k \rightarrow R_{ij} > R_{ik}$

La propiedad 1 dice que R_{ij} es una función no negativa.
La propiedad 2 es la propiedad de simetría.
La propiedad 3 indica que si la distancia entre dos clusters aumenta mientras su dispersión se mantiene constante, entonces su similitud disminuye.
La propiedad 4 indica que si la distancia entre dos clusters se mantiene constante mientras su dispersión aumenta, entonces su similitud aumenta.

C.3. Propiedades del algoritmo K-means

Proposición C.3.1 (Convergencia). El valor de la función objetivo, RSS Ec. 2.15, decrece monótonamente en cada iteración del algoritmo.

Primero que nada notar que RSS decrece en cada paso de reasignación (*Reassignment of vectors*), dado que todo vector es asignado al centroide más cercano. En segunda instancia RSS decrece también en el paso de recomputación (*Recomputation of vectors*) debido a que el nuevo centroide es el vector $\vec{v}$ para el cual RSS_k (Ec. 2.14) alcanza el valor mínimo:

$$RSS_k(\vec{v}) = \sum_{\vec{x} \in \omega_k} |\vec{v} - \vec{x}|^2 = \sum_{\vec{x} \in \omega_k} \sum_{m=1}^{M} (v_m - x_m)^2 \tag{C.1}$$

$$\frac{\partial RSS_k(\vec{v})}{\partial v_m} = \sum_{\vec{x} \in \omega_k} 2(v_m - x_m)$$

En donde x_m y v_m son los m-ésimos componentes de sus respectivos vectores. Igualando la derivada parcial a 0 se obtiene:

$$v_m = \frac{1}{|\omega_k|} \sum_{\vec{x} \in \omega_k} x_m \; \square$$

Esta proposición demuestra la convergencia de K-means, pero el óptimo encontrado no necesariamente es global. Véase el siguiente ejemplo:

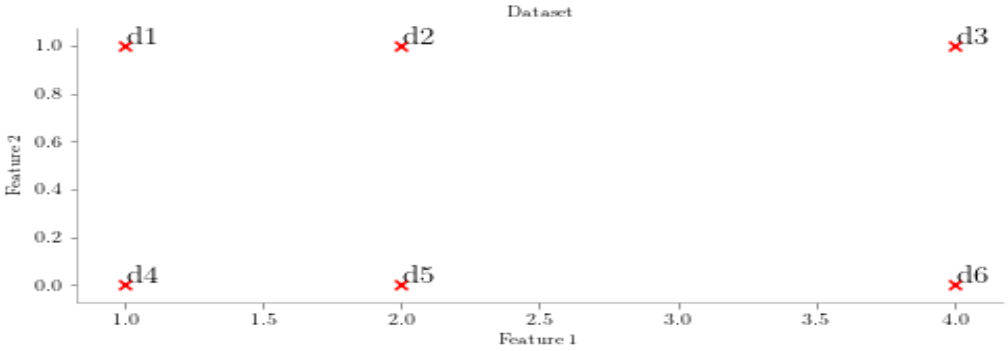


Figura C.2: El resultado de clusterizar con K-means depende de las semillas iniciales. Para k=2, si son elegidos los puntos d_1, d_2, K-means converge a $\{\{d_1, d_2, d_3\}, \{d_4, d_5, d_6\}\}$, un subcluster óptimo. Si son elegidos los puntos d_2, d_3, K-means converge a $\{\{d_1, d_2, d_4, d_5\}, \{d_3, d_6\}\}$, un óptimo global. Figura extraída de la página 364, capítulo 16, Manning et al. [3].

Proposición C.3.2 (Complejidad)**.** La complejidad de K-means es lineal en todos sus factores relevantes.

La mayor parte del tiempo K-means se pasa computando distancias entre vectores. Una operación de ese estilo tiene un costo de $O(M)$, siendo M la dimensión de los vectores $\vec{x_i}$. El paso de reasignación computa KN distancias, así que su complejidad total es de $O(KNM)$. En el paso de recomputación el vector es sumado a su centroide una sola vez, así que la complejidad de este paso es $O(NM)$. Para un número fijo I de iteracciones la complejidad es dada por $O(IKNM)$. Se ve que K-means es lineal en todos sus factores relevantes: iteraciones, número de clusters, número de vectores y dimensionalidad $\square$

Proposición C.3.3. $RSS_{min}(K)$ es monótona decreciente en K.

Bosquejo: Se puede emplear inducción en K para la prueba, teniendo $RSS(K+1) \leq RSS(K)$. Con paso base $RSS(2) \leq RSS(1)$. Para $K = 1$ el único cluster tiene como centro la media de todos los puntos. Ejemplo:

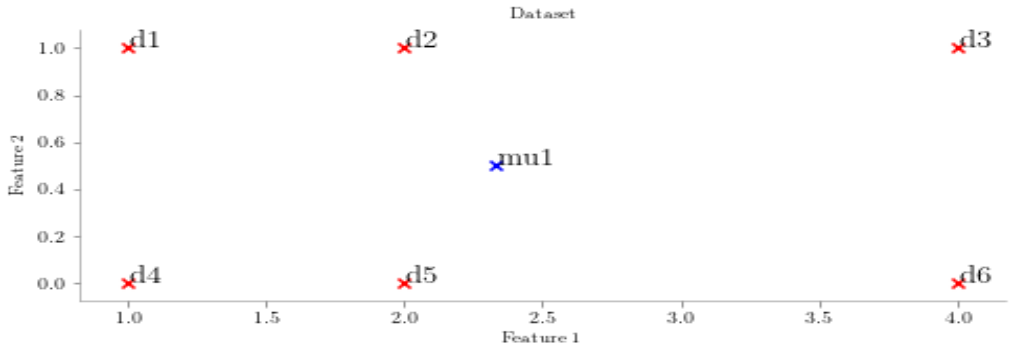


Figura C.3: *mu*1 es la media entre todos los puntos.

Si ocurriese que

$$RSS(1) < RSS(2)$$

entonces

$$RSS(1) < \sum_{\vec{x} \in w_1} |\vec{x} - \vec{u}(w_1)|^2 + \sum_{\vec{x} \in w_2} |\vec{x} - \vec{u}(w_2)|^2$$

entonces

$$\sum_{\vec{x} \in w_1} |\vec{x} - \vec{u}|^2 + \sum_{\vec{x} \in w_2} |\vec{x} - \vec{u}|^2 < \sum_{\vec{x} \in w_1} |\vec{x} - \vec{u}(w_1)|^2 + \sum_{\vec{x} \in w_2} |\vec{x} - \vec{u}(w_2)|^2$$

Siendo u el centro de todos los puntos. Esto implicaría que al menos o (i) $\sum_{\vec{x} \in w_1} |\vec{x} - \vec{u}|^2 < \sum_{\vec{x} \in w_1} |\vec{x} - \vec{u}(w_1)|^2$ o (ii) $\sum_{\vec{x} \in w_2} |\vec{x} - \vec{u}|^2 < \sum_{\vec{x} \in w_2} |\vec{x} - \vec{u}(w_2)|^2$. Supongamos que (i) es verdadero. Pero esto absurdo ya que por Proposición C.3.1 $\vec{u}(w_1)$ es el valor que minimiza la Ecuación C.1.

Apéndice D

Series de tiempos

Las series de tiempo permiten expresar matemáticamente datos experimentales, caracterizarlos estadísticamente como un *proceso estocástico* y definir modelos de comportamiento y predicción. En este apartado se definirán algunos de los modelos de series de tiempo utilizados en este trabajo y en el trabajo estudiado en el estado del arte (Laurinec et al. [12]). En la primera parte se definen algunas medidas de dependencia, que permiten evaluar la relación entre el valor de la serie para un punto del tiempo *t*, respecto a sus valores pasados. Medidas como *la media*, *la función de autocovarianza*, *la función de autocorrelación* (ACF) y *la función de autocorrelación parcial* (PACF) son definidas. La segunda parte está dedicada al estudio de condiciones en las series de tiempo que aseguran cierta regularidad en su comportamiento. Se definen: *los procesos estrictamente estacionarios* y *los débilmente estacionarios*. El tercer punto del apéndice se dedica a la descripción del modelo *Multiple Linear Regression* (MLR). En la cuarta sección se presentan dos técnicas para determinar cual es la mejor cantidad de parámetros para un modelo. En primera instacia se revisa la técnica *Analysis of Variance for Regression* (ANOVA). Luego el criterio *Akaike's Information Criterion* (AIC). La quinta parte describe el modelo SARIMA. Este modelo combina conceptos clásicos de las series de tiempo tales como: autoregresión, diferenciación, media móvil, procesos estacionales y procesos estacionarios, para caracterizar las series. Se verá como SARIMA aplica cada uno de estos conceptos en el modelo. Se presentan las definiciones y conceptos para hacer forecasting con SARIMA, junto con un ejemplo de predicción para un caso particular del modelo, denominado AR(2). La sexta parte presenta el modelo *Exponential Smoothing* (ES) para series de tiempo y la séptima parte presenta el modelo *Random Forest* (RF), que no es esencialmente un modelo para series de tiempo, pero puede ser utilizado para

predicción.

D.1. Medidas de dependencia

Sea $F_t(x) = P\{x_t \leq x\}$ la función de distribución acumulada, CDF por sus siglas en inglés, para la variable aleatoria x_t que representa el valor que la serie toma en el tiempo t. Y sea $f_t(x)$ la función de densidad asociada a CDF.

Definición D.1.1. Media La media es definida como

$$\mu_{x_t} = E[x_t] = \int_{-\infty}^{+\infty} x f_t(x) dx \tag{D.1}$$

en donde E denota al operador valor esperado.

Ejemplo D.1.2. Media de la serie moving average
Si w_t denota al modelo **white noise**, visto en la Sección 2.1.1, entonces $\mu_{w_t} = E[w_t] = 0 \ \forall t$. Este resultado se corresponde con la gráfica de arriba en la Figura 2.2, en donde la serie fluctúa los valores alrededor del 0. Si en la Figura 2.2 se observa la gráfica del medio que corresponde al modelo **moving average**, definido en la Sección 2.1.1, se puede observar que también los valores de la serie de tiempo fluctúan alrededor del 0. El cálculo de la media confirma esta observación:

$$\mu_{v_t} = E[v_t] = E[\frac{1}{3}(w_{t-1} + w_t + w_{t+1})] = \frac{1}{3}(E[w_{t-1}] + E[w_t] + E[w_{t+1}]) = 0 \tag{D.2}$$

Definición D.1.3. Autocovarianza

$$\gamma_x(s,t) = cov(x_s, x_t) = E[(x_s - \mu_{x_s})(x_t - \mu_{x_t})] \tag{D.3}$$

para cualesquiera tiempos t, s.
La autocovarianza mide la dependencia lineal entre dos puntos de la misma serie observados a diferentes tiempos.

Ejemplo D.1.4. Autocovarianza de la serie moving average
El modelo **white noise**, por como está definido $w_t \sim wn(0, \sigma_w^2)$, tiene media $\mu_{w_t} = 0$.
La autocovarianza para este modelo es el siguiente:

$$\gamma_w(s,t) = cov(w_s, w_t) = \begin{cases} cov(w_t, w_t) = var(w_t, w_t) = \sigma_w^2 & s = t \\ 0 & s \neq t \end{cases} \tag{D.4}$$

Para el modelo **moving average** se tiene que:

$$\gamma_v(s,t) = cov(v_s, v_t) = cov(\frac{1}{3}(w_{s-1} + w_s + w_{s+1}), \frac{1}{3}(w_{t-1} + w_t + w_{t+1})). \quad \text{(D.5)}$$

Usando los resultados expuestos en la Ecuación D.4 y las propiedades conocidas de la covarianza de la combinación lineal de variables aleatorias se tiene, cuando $s = t$:

$$\gamma_v(t,t) = \frac{1}{9}cov((w_{t-1} + w_t + w_{t+1}), (w_{t-1} + w_t + w_{t+1})) = \\ \frac{1}{9}[cov(w_{t-1}, w_{t-1}) + cov(w_t, w_t) + cov(w_{t+1}, w_{t+1})] = \frac{3}{9}\sigma_w^2. \quad \text{(D.6)}$$

Computaciones similares permiten resumir en la Ecuación D.7 cual es la dependencia lineal entre las variables aleatorias, en la serie de tiempo **moving average**, en función de la separación en el tiempo entre ellas:

$$\gamma_v(s,t) = \begin{cases} \frac{3}{9}\sigma_w^2, & s = t \\ \frac{2}{9}\sigma_w^2, & |s-t| = 1 \\ \frac{1}{9}\sigma_w^2, & |s-t| = 2 \\ 0, & |s-t| > 2 \end{cases} \quad \text{(D.7)}$$

Definición D.1.5. Función de AutoCorrelación, ACF por sus siglas en inglés, se define como:

$$\rho(s,t) = \frac{\gamma_x(s,t)}{\sqrt{\gamma_x(s,s)\gamma_x(t,t)}} \quad \text{(D.8)}$$

para cualesquiera tiempos t, s.
ACF mide la predictabilidad de la serie en el tiempo t, para el valor x_t usando solo la información del valor x_s. Se tiene que $-1 \leq \rho(s,t) \leq 1$. Si x_t se puede predecir a partir de x_s como $x_t = \beta_0 + \beta_1 x_s$, entonces $\rho(s,t) = 1$ si $\beta_1 > 0$ y $\rho(s,t) = -1$ si $\beta_1 < 0$.

Dada una serie de tiempo, sean dos variables aleatorias x_t y x_{t+h}, para dos tiempos separados por una distancia $h \geq 2$. Sean además:

$$\hat{x}_{t+h} = \beta_1 x_{t+h-1} + \beta_2 x_{t+h-2} + \ldots + \beta_{h-1} x_{t+1} \quad \text{(D.9)}$$

y

$$\hat{x}_t = \alpha_1 x_{t+h-1} + \alpha_2 x_{t+h-2} + \ldots + \alpha_{h-1} x_{t+1} \quad \text{(D.10)}$$

las regresiones respectivas de x_{t+h} y x_t sobre las variables $\{x_{t+h-1}, x_{t+h-2}, \ldots, x_{t+1}\}$.

Definición D.1.6. La **Función de AutoCorrelación Parcial**, PACF por sus siglas en inglés, denotada $\phi_{hh}, h = 1, 2, \ldots$ se define de la siguiente manera:

$$\phi_{11} = corr(x_{t+1}, x_t) = \rho(t+1, t) \tag{D.11}$$

y

$$\phi_{hh} = corr(x_{t+h} - \hat{x}_{t+h}, x_t - \hat{x}_t), \quad h \geq 2 \tag{D.12}$$

PACF, viene a representar la correlación existente entre x_{t+h} y x_t, con la dependencia lineal respecto a las variables $\{x_{t+h-1}, x_{t+h-2}, \ldots, x_{t+1}\}$ removidas en ambos.

D.2. Estacionalidad

En esta sección se impondrán algunas condiciones a las series de tiempo para asegurar cierta regularidad en su comportamiento.

Definición D.2.1. Estacionalidad estricta
Una serie de tiempo estricamente estacionaria cumple que el comportamiento probabilístico de cada colección de valores

$$\{x_{t_1}, x_{t_2}, \ldots, x_{t_k}\}$$

es idéntico al del conjunto desplazado

$$\{x_{t_1+h}, x_{t_2+h}, \ldots, x_{t_k+h}\}$$

es decir,

$$Pr\{x_{t_1} \leq c_1, \ldots, x_{t_k} \leq c_k\} = Pr\{x_{t_1+h} \leq c_1, \ldots, x_{t_k+h} \leq c_k\} \tag{D.13}$$

para todo $k = 1, 2, \ldots$ para todos los tiempos $t_1, t_2, \ldots, t_k$ para todos los valores $c_1, \ldots, c_k$, para todo desplazamiento $h = 0, \pm 1, \pm 2, \ldots$

Si $k = 1$, la Ecuación D.13 implica que

$$Pr\{x_s \leq c\} = Pr\{x_t \leq c\} \tag{D.14}$$

para cualquier tiempo t y s. Esto quiere decir por ejemplo, que la probabilidad de que los valores en una serie de tiempo, con mediciones cada una hora, sean negativos a la 1 de la mañana es la misma que a las 10 de la mañana. Y si la media μ_t existe, entonces debido a que la distribución no cambia con el tiempo se tiene

que esta es constante: $\mu_t = \mu_s$.

Si $k = 2$, por la Ecuación D.13 se tiene que

$$Pr\{x_s \leq c_1, x_t \leq c_2\} = Pr\{x_{s+h} \leq c_1, x_{t+h} \leq c_2\} \tag{D.15}$$

y si la varianza existe en el proceso, las Ecuaciones D.14-D.15 implican que la autocovarianza de la serie satisfaga

$$\gamma(s,t) = \gamma(s+h, t+h)$$

es decir la autocovarianza depende solo de la distancia h entre los puntos en el tiempo.
Este resultado se ve desarrollando las expresiones a ambos lados de la igualdad:

$$\begin{aligned}\gamma(s,t) = cov(x_s, x_t) = E[(x_s - \mu)(x_t - \mu)] = E[x_s x_t] - \mu^2 \\ \gamma(s+h, t+h) = cov(x_{s+h}, x_{t+h}) = E[(x_{s+h} - \mu)(x_{t+h} - \mu)] = \\ E[x_{s+h} x_{t+h}] - \mu^2\end{aligned}$$

como la probabilidad conjunta es la misma para $x_s x_t$ y $x_{s+h} x_{t+h}$ se tiene que $E[x_s x_t] = E[x_{s+h} x_{t+h}]$.
La definición de estacionalidad dada en Def. D.2.1 es demasiado fuerte para la mayoría de las aplicaciones. En lugar de imponer condiciones sobre todas las distribuciones posibles de la serie de tiempo, la siguiente definición impone restricciones solo a la media y a la autocovarianza.

Definición D.2.2. Estacionalidad débil
Una serie de tiempo débilmente estacionaria, con varianza finita, cumple las siguientes condiciones:

i) la media μ_t es constante y no depende del tiempo t

ii) la función de autocovarianza $\gamma(s,t)$ depende de s y t, solo a través de su diferencia $|s - t|$

La estacionalidad débil requiere regularidad en la media y en la autocovarianza, de manera que estas cantidades pueden ser estimadas promediando.
Debido a que la media es constante, esta se puede escribir como

$$\mu_t = \mu. \tag{D.16}$$

Sea $s = t + h$, debido a que la autocovarianza depende solamente del desplazamiento h, entonces se llega a la siguiente igualdad

$$\gamma(t+h,t) = cov(x_{t+h}, x_t) = cov(x_h, x_0) = \gamma(h,0) \tag{D.17}$$

dado que la diferencia entre $t + h$ y t es la misma que entre h y 0.
La autocovarianza γ y la función de autocorrelación ρ, para una serie estacionaria pueden ser reescritas de la siguiente manera.

Definición D.2.3. Autocovarianza de serie estacionaria

$$\gamma(h) = cov(x_{t+h}, x_t) = E[(x_{t+h} - \mu)(x_t - \mu)] \tag{D.18}$$

Definición D.2.4. Función de AutoCorrelación de serie estacionaria

$$\rho(h) = \frac{\gamma(t+h,t)}{\sqrt{\gamma(t+h,t+h)\gamma_x(t,t)}} = \frac{\gamma(h)}{\gamma(0)} \tag{D.19}$$

Ejemplo D.2.5. Estacionariedad de la serie moving average
En el Ejemplo D.1.4 se dedujo que la media era constante $\mu_v = 0$ y que la autocovarianza dependía de la distancia h en el tiempo

$$\gamma_v(h) = \begin{cases} \frac{3}{9}\sigma_w^2, & h = 0 \\ \frac{2}{9}\sigma_w^2, & h = 1 \\ \frac{1}{9}\sigma_w^2, & h = 2 \\ 0, & h > 2 \end{cases} \tag{D.20}$$

La función de autocorrelación Ecuación D.2.4 en este caso es la siguiente

$$\rho_v(s,t) = \begin{cases} 1, & h = 0 \\ \frac{2}{3}, & h = 1 \\ \frac{1}{3}, & h = 2 \\ 0, & h > 2 \end{cases} \tag{D.21}$$

la Figura D.1 muestra la autocorrelación como una función de la distancia en el tiempo h (LAG o retraso).

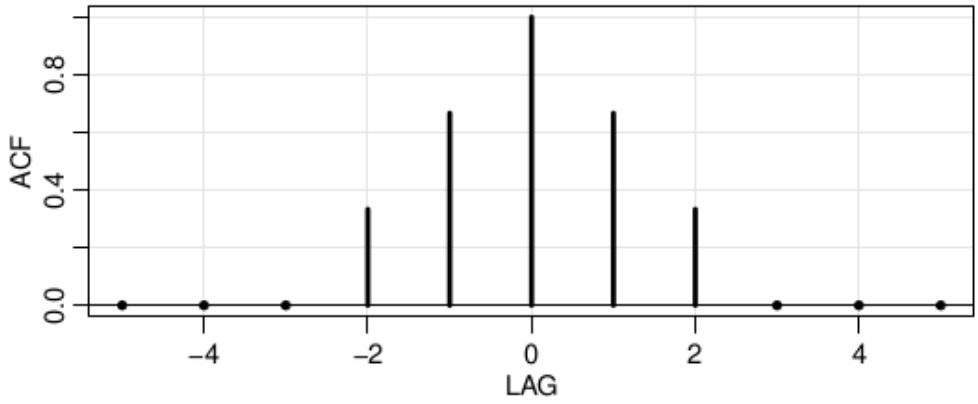


Figura D.1: Imagen extraída de la página 22 del Libro [14].

D.3. Regresión Lineal Múltiple

Sea x_t una serie de tiempo *dependiente*, influenciada por q series *independientes*: $z_{t1}, z_{t2}, \ldots, z_{tq}$. La relación entre ellas en un modelo lineal se expresa así

$$x_t = \beta_0 + \beta_1 z_{t_1} + \beta_2 z_{t_2} + \ldots + \beta_q z_{t_q} + w_t \tag{D.22}$$

donde $\beta_0, \beta_1, \ldots, \beta_q$ son coeficientes de regresión fijos y $\{w_t\}$ es un proceso estocástico que representa un error aleatorio o ruido, con media 0 y desviación σ_w^2. La Ecuación D.22 puede representarse en forma vectorial

$$x_t = \beta^T z_t + w_t \tag{D.23}$$

con $\beta = (\beta_0, \beta_1, \ldots, \beta_q)$ y $z_t = (1, z_{t_1}, z_{t_2}, \ldots, z_{t_q})$.
La estimación de los coeficientes β puede realizarse aplicando la técnica de mínimos cuadrados, minimizando el error dado en la Ecuación D.24.

$$Q = \sum_{t=1}^{n} w_t = \sum_{t=1}^{n} (x_t - \beta^T z_t)^2 \tag{D.24}$$

Diferenciando la Ecuación D.24 respecto al vector β, se llega a la estimación $\hat{\beta}$

$$\hat{\beta} = (\sum_{t=1}^{n} z_t^T z_t)^{-1} \sum_{t=1}^{n} z_t x_t \tag{D.25}$$

asumiendo la no singularidad de $\sum_{t=1}^{n} z_t^T z_t$.
El modelo presentado en la Ecuación D.22 es *aditivo*, en el sentido de que x_t depende linealmente de las variables independientes.

D.4. Criterios para la elección del modelo

El objetivo de los criterios en esta sección radica en determinar la cantidad óptima de parámetros para un modelo. Las dos técnicas que se presentan son ANOVA y AIC.

D.4.1. ANOVA

ANOVA, es un acrónimo para *ANalysis Of VAriance for regression*. En esta técnica se tienen q variables de regresión. Un modelo selecciona $r < q$ de ellas, digamos $z_{t,1:r} = \{z_{t_1}, z_{t_2}, \ldots, z_{t_r}\}$, y propone que la variable dependiente x_t es influenciada por este subconjunto. Se tiene que el modelo dado por la Ecuación D.22, es reducido al siguiente.

$$x_t = \beta_0 + \beta_1 z_{t_1} + \beta_2 z_{t_2} + \ldots + \beta_r z_{t_r} + w_t \tag{D.26}$$

La hipótesis nula en este caso es $H_0 : \beta_{r+1} = \ldots = \beta_q = 0$. El modelo dado por la Ecuación D.26 puede ser comparado al de la Ecuación D.22, de la siguiente manera

$$F = \frac{(SSE_r - SSE)/(q-r)}{(SSE)/(n-q-1)} \tag{D.27}$$

en donde SSE es el error minimizado de la suma de cuadrados para el modelo de la Ecuación D.22

$$SSE = \sum_{t=1}^{n} (x_t - \hat{\beta}^T z_t) \tag{D.28}$$

y SSE_r es el error minimizado de la suma de cuadrados para el modelo de la Ecuación D.26.
Bajo la hipótesis nula H_0 la Ecuación D.27 tiene una distribución F con $q - r$ y $n - q - 1$ grados de libertad, cuando Ec. D.26 es el modelo correcto. La hipótesis nula es rechazada con un nivel de confianza α si $F > F_{q-r}^{n-q-1}(\alpha)$, el percentíl $1 - \alpha$ de la distribución F.

D.4.2. AIC

Definición D.4.1. AIC, *Akaike's Information Criterion*.

$$AIC = log(\hat{\sigma}_k^2) + \frac{n + 2k}{n} \tag{D.29}$$

con

$$\hat{\sigma}_k^2 = \frac{SSE(k)}{n} \tag{D.30}$$

representando el residuo de la suma de cuadrados bajo el modelo elegido con k coeficientes de regresión y n ecuaciones.

El valor k con el mínimo AIC especifica el mejor modelo. La idea es que minimizar $\hat{\sigma}_k^2$ es un objetivo razonable, excepto por el hecho de que decrece monótonamente a medida que k aumenta. Por ello se debe penalizar por un término proporcional al número de parámetros. Esto es lo que ocurre con el término $\frac{n+2k}{n}$ en la Ecuación D.29.

D.5. Modelo ARIMA

ARIMA, *Autoregressive Integrated Moving Average*, es un modelo clásico que permite capturar la dinámica de las series de tiempo y hacer predicciones. El modelo ARIMA fue popularizado por el trabajo de Box, G. y Jenkins, G. (1970) [2]. Y sus componentes autoregresivos y de media móvil datan de la tesis de Peter Whittle en 1951. En esta sección se resumen los conceptos fundamentales del modelo y de la predicción usando ARIMA.

D.5.1. Operador Backshift

El operador Backshift es útil para facilitar la notación y estudiar el modelo ARIMA.

Definición D.5.1. Operador Backshift
El operador Backshift se define como

$$Bx_t = x_{t-1} \tag{D.31}$$

en su extensión a potencias

$$B^k x_t = x_{t-k} \tag{D.32}$$

D.5.2. AR

Los modelos autoregresivos están basados en la idea de que el valor actual x_t, de la serie, puede ser explicado como función de p valores pasados $x_{t-1}, x_{t-2}, \ldots, x_{t-p}$.

Definición D.5.2. AR
Un modelo autoregresivo de orden p, abreviado $AR(p)$, es de la forma

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \ldots + \phi_p x_{t-p} + w_t \tag{D.33}$$

donde la serie x_t es estacionaria, $w_t \sim wn(0, \sigma_w^2)$ y $\phi_1, \phi_2, \ldots, \phi_p$ son constantes con $\phi_p \neq 0$. Si la media μ de x_t es distinta de 0 entonces reemplazar x_t por $x_t - \mu$ en Ec.D.33

$$x_t - \mu = \phi_1(x_{t-1} - \mu) + \phi_2(x_{t-2} - \mu) + \ldots + \phi_p(x_{t-p} - \mu) + w_t. \tag{D.34}$$

Una forma útil de escribir la Ecuación D.33 es la siguiente

$$(1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p) x_t = w_t \tag{D.35}$$

o más conciso

$$\phi(B) x_t = w_t \tag{D.36}$$

esta nueva expresión lleva a la definición de un nuevo operador.

Definición D.5.3. Operador autoregresivo ϕ

$$\phi(B) = (1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p) \tag{D.37}$$

Ejemplo D.5.4. El modelo AR(1)
De la Ecuación D.33 se deduce la forma del modelo con $p = 1$

$$x_t = \phi x_{t-1} + w_t \tag{D.38}$$

iterando la Ecuación D.38 hacia atrás, se obtiene

$$x_t = \phi x_{t-1} + w_t = \phi(\phi x_{t-2} + w_{t-1}) + w_t =$$

$$\phi^2 w_{t-2} + \phi w_{t-1} + w_t$$

$$\vdots$$

$$\phi^k x_{t-k} + \sum_{j=0}^{k-1} \phi^j w_{t-j}$$

esto sugiere que si se sigue iterando e imponiendo para asegurar la convergencia en todo tiempo t que $|\phi| < 1$ y $sup_t\ var(x_t) < \infty$, entonces se puede representar AR(1) como un proceso lineal

$$x_t = \sum_{j=0}^{\infty} \phi^j w_{t-j} \tag{D.39}$$

la Ecuación D.39 se denomina solución estacionaria del modelo, y el modelo dado por la Ecuación D.33 queda

$$\sum_{j=0}^{\infty} \phi^j w_{t-j} = \phi(\sum_{k=0}^{\infty} \phi^k w_{t-1-j}) + w_t \tag{D.40}$$

analizando la media y la autocovarianza que caracterizan a los procesos débilmente estacionarios, para AR(1) se tiene que la media $\mu = 0$

$$E[x_t] = \sum_{j=0}^{\infty} \phi^j E[w_{t-j}] = 0$$

y la autocovarianza $\gamma(h) = \frac{\sigma_w^2 \phi^h}{1-\phi^2}$

$$\gamma(h) = cov(x_{t+h}, x_t) = E[(\sum_{j=0}^{\infty} \phi^j w_{t+h-j})(\sum_{k=0}^{\infty} \phi^k w_{t-k})]$$

$$= E[(w_{t+h} + \ldots + \phi^h w_t + \phi^{h+1} w_{t-1} + \ldots)(w_t + \phi w_{t-1} + \ldots)]$$

$$= \sigma_w^2 \sum_{j=0}^{\infty} \phi^{h+j} \phi^j = \sigma_w^2 \phi^h \sum_{j=0}^{\infty} \phi^{2j} = \frac{\sigma_w^2 \phi^h}{1-\phi^2}$$

mientras que la función ACF (Ecuación D.2.4) para AR(1) es

$$\rho(h) = \frac{\gamma(h)}{\gamma(0)} = \phi^h \tag{D.41}$$

en la Figura D.2 se observan dos procesos que ajustan al modelo AR(1). La gráfica de arriba tiene como constante $\phi = 0.9$ y la de abajo $\phi = -0.9$. Para el primer caso

se tiene el siguiente ACF $\rho(h) = 0.9^h$, con lo cual la correlación entre puntos contiguos es alta, reflejándose esto en la regularidad de la gráfica de arriba. El segundo caso tiene ACF $\rho(h) = -0.9^h$, lo que implica que la correlación, entre dos puntos, para una distancia $h = 1$ sea alta y negativa y para una distancia $h = 2$ sea alta y positiva. Esto se refleja viendo en la gráfica de abajo que para un valor x_t positivo, típicamente el siguiente valor x_{t+1} es negativo, y vuelve a ser positivo para x_{t+2}. Lo que genera que la gráfica sea bastante entrecortada.

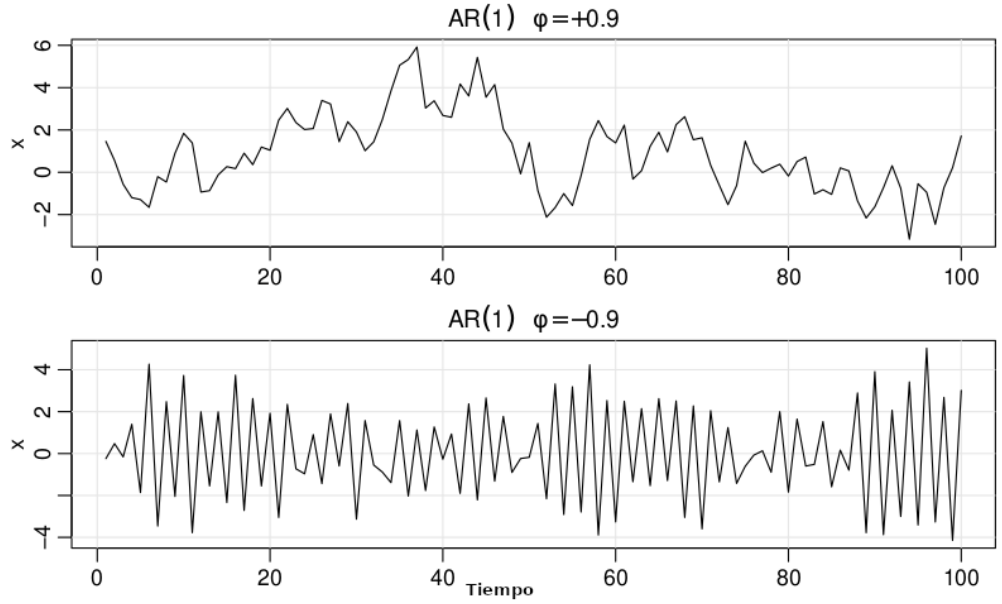


Figura D.2: Imagen extraída de la página 81 del Libro [14].

D.5.3. MA

Este modelo es una alternativa a la representación autoregresiva Ec. D.38, en la cual se asume que el x_t del lado izquierdo de la ecuación es una combinación lineal de sus valores pasados. El modelo *Moving Average* (MA) de orden q asume que el ruido w_t está combinado linealmente para formar los datos observados.

Definición D.5.5. MA
Un modelo *Moving Average* de orden q, abreviado $MA(q)$, es de la forma

$$x_t = w_t + \theta_1 w_{t-1} + \theta_2 w_{t-2} + \ldots + \theta_q w_{t-q} \tag{D.42}$$

donde $w_t \sim wn(0, \sigma_w^2)$ y $\theta_1, \theta_2, \ldots, \theta_p$ son parámetros con $\theta_p \neq 0$.

El sistema puede representarse como

$$x_t = \theta(B)w_t$$

lo cual motiva la definición del operador.

Definición D.5.6. Operador moving average θ

$$\theta(B) = 1 + \theta_1 B + \theta_2 B^2 + \ldots + \theta_q B^q \tag{D.43}$$

Ejemplo D.5.7. El modelo MA(1)
Considérese el proceso MA(1) $x_t = w_t + \theta w_{t-1}$. La caracterización como proceso débilmente estacionario es la siguiente:
Media:

$$\mu = E[x_t] = E[w_t] + \theta E[w_{t-1}] = 0$$

Autocovarianza:

$$\gamma(h) = \begin{cases} (1+\theta)\sigma_w^2 & h = 0 \\ \theta\sigma_w^2 & h = 1 \\ 0 & h > 1 \end{cases}$$

ACF:

$$\rho(h) = \begin{cases} \frac{(1+\theta^2)}{\theta} & h = 1 \\ 0 & h > 1 \end{cases}$$

En este modelo se tiene una correlación baja $|\rho| \leq \frac{1}{2}$. Para cualquier punto x_t existe una correlación con x_{t-1}, pero no con $x_{t-2}, x_{t-3}, \ldots$, en contraste con AR(1), en donde la correlación entre x_t y x_{t-k} nunca es cero. En la Figura D.3 se observan dos procesos MA(1). La gráfica de arriba muestra un proceso con $\theta = 0.9$ y la gráfica de abajo un proceso con $\theta = -0.9$. Para el primer caso $\theta = 0.9$ x_t y x_{t-1} se correlacionan positivamente y la gráfica presenta mayor regularidad en comparación a la gráfica para $\theta = -0.9$, en donde x_t y x_{t-1} se correlacionan negativamente.

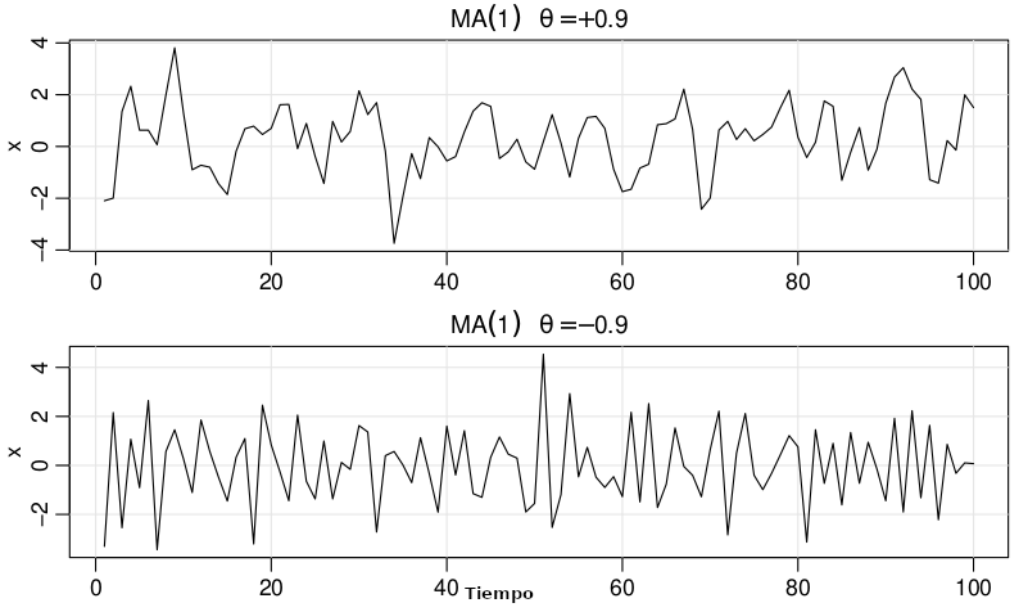


Figura D.3: Imagen extraída de la página 84 del Libro [14].

D.5.4. ARMA

Definición D.5.8. ARMA Una serie de tiempo $\{x_t; t = 0, \pm 1, \pm 2, \ldots\}$ es ARMA(p,q) si es estacionaria y se puede representar de la siguiente manera:

$$x_t = \phi_1 x_{t-1} + \ldots + \phi_p x_{t-p} + w_t + \theta_1 w_{t-1} + \ldots + \theta_q w_{t-q} \tag{D.44}$$

con $\phi_p \neq 0$, $\theta_q \neq 0$ y σ_w^2. Los parámetros p y q se denominan orden autoregresivo y de media móvil, respectivamente. Si x_t tiene media distinta de cero μ, se define $\alpha = \mu(1 - \phi_1 - \ldots - \phi_p)$ y el modelo se escribe

$$x_t = \alpha + \phi_1 x_{t-1} + \ldots + \phi_p x_{t-p} + w_t + \theta_1 w_{t-1} + \ldots + \theta_q w_{t-q} \tag{D.45}$$

con $w_t \sim wn(0, \sigma_w^2)$.

El modelo puede ser reescrito usando el operador autoregresivo Def. D.5.3 y el operador moving average Def. D.5.6:

$$\phi(B)x_t = \theta(B)w_t. \tag{D.46}$$

Para encontrar una representación lineal para una serie de tiempo x_t, en función de sus valores pasados $x_t = \sum_{j=1}^{\infty} \phi_j x_{t-j} + w_t$ (Ec. D.33) o en función de los valores

pasados del ruido $x_t = \sum_{j=1}^{\infty} \theta_j w_{t-j} + w_t$ (Ec. D.42), se necesitan asumir sobre el proceso ARMA las condiciones de *causalidad* y/o *invertibilidad*.

Definición D.5.9. Los **Polinomios AR y MA** son definidos respectivamente como

$$\phi(z) = 1 - \phi_1 z - \phi_2 z^2 + \ldots + \phi_p z^p, \quad \phi_p \neq 0 \tag{D.47}$$

y

$$\theta(z) = 1 + \theta_1 z - \theta_2 z^2 + \ldots + \theta_q z^q, \quad \phi_q \neq 0 \tag{D.48}$$

siendo z un número complejo. Se dice que los polinomios $\phi(z)$ y $\theta(z)$ están en su *forma simple* si no tienen factores en común.

Definición D.5.10. Causalidad Un proceso ARMA(p,q) dícese ser *causal*, si la serie de tiempo $\{x_t; t = 0, \pm 1, \pm 2, \ldots\}$ puede ser escrita como un proceso lineal

$$x_t = \sum_{j=0}^{\infty} \psi_j w_{t-j} = \psi(B) w_t \tag{D.49}$$

donde $\psi(B) = \sum_{j=0}^{\infty} \psi_j B^j$, $\sum_{j=0}^{\infty} |\psi_j| < \infty$; y $\psi_0 = 1$.

Proposición D.5.11. Caracterización de causalidad en un proceso ARMA(p,q)

Un proceso ARMA(p,q) es *causal* si y solo si $\phi(z) \neq 0$ para $|z| \leq 1$. Los coeficientes del proceso lineal dados en la Ecuación D.49 pueden ser determinados resolviendo

$$\psi(z) = \sum_{j=0}^{\infty} \psi_j z^j = \frac{\theta(z)}{\phi(z)}, \quad |z| < 1 \tag{D.50}$$

Definición D.5.12. Invertibilidad Un proceso ARMA(p,q) dícese ser *invertible*, si la serie de tiempo $\{x_t; t = 0, \pm 1, \pm 2, \ldots\}$ puede ser escrita como

$$\pi(B) x_t = \sum_{j=0}^{\infty} \pi_j x_{t-j} = w_t \tag{D.51}$$

donde $\pi(B) = \sum_{j=0}^{\infty} \pi_j B^j$, $\sum_{j=0}^{\infty} |\pi_j| < \infty$; y $\pi_0 = 1$.

Proposición D.5.13. Caracterización de invertibilidad en un proceso ARMA(p,q)

Un proceso ARMA(p,q) es *invertible* si y solo si $\theta(z) \neq 0$ para $|z| \leq 1$. Los coeficientes del proceso lineal dados en la Ecuación D.51 pueden ser determinados resolviendo

$$\pi(z) = \sum_{j=0}^{\infty} \pi_j z^j = \frac{\phi(z)}{\theta(z)}, \quad |z| < 1 \tag{D.52}$$

Ejemplo D.5.14. Forma simple, causalidad e invertibilidad
Sea el siguiente proceso

$$x_t = 0.4x_{t-1} + 0.45x_{t-2} + w_t + w_{t-1} + 0.25w_{t-2}$$

empleando la notación *backshift* el proceso queda

$$(1 - 0.4B - 0.45B^2)x_t = (1 + B + 0.25B^2)w_t$$

que en principio asemeja un proceso ARMA(2,2). Sin embargo al factorizar ambos operadores se tiene que los polinomios asociados no están en su *forma simple*

$$\phi(B) = (1 - 0.4B - 0.45B^2) = (1 + 0.5B)(1 - 0.9B)$$

y

$$\theta(B) = (1 + B + 0.25B^2) = (1 + 0.5B)^2$$

el factor $(1 + 0.5B)$ puede ser cancelado. Una vez con ambos polinomios en su forma simple, el proceso puede ser representado como un ARMA(1,1).

$$(1 - 0.9B)x_t = (1 + 0.5B)w_t$$

$$x_t = 0.9x_{t-1} + w_t + 0.5w_{t-1}$$

El modelo es *causal* porque $\phi(z) = (1 - 0.9z) = 0$ cuando $z = 10/9$ y se encuentra fuera del círculo unidad. El modelo también es invertible porque $\theta(z) = (1 + 0.5z) = 0$ cuando $z = -2$, que también se encuentra fuera del círculo unidad. Los coeficientes ψ_j pueden ser calculados usando la Proposición D.5.11: $\phi(z)\psi(z) = \theta(z)$

$$(1 - 0.9z)(1 + \psi_1 z + \psi_2 z^2 + \ldots + \psi_j z^j + \ldots) = (1 + 0.5z)$$

reordenando

$$1 + (\psi_1 - 0.9)z + (\psi_2 - 0.9\psi_1)z^2 + \ldots + (\psi_j - 0.9\psi_{j-1})z^j + \ldots = (1 + 0.5z)$$

igualando los coeficientes de ambos polinomios se tiene que $\psi_1 = 1.4$ y $\psi_j = 0.9\psi_{j-1}$ para $j > 1$. Obteniéndose la siguiente expresión $\psi_j = (1.4)(0.9)^{j-1}$ para $j \geq 1$. Con estos resultados la Ecuación D.49 se puede escribir

$$x_t = 1.4 \sum_{j=1}^{\infty} (0.9)^{j-1} w_{t-j} + w_t$$

los coeficientes π_j pueden ser calculados usando la Proposición D.5.13: $\theta(z)\pi(z) = \psi(z)$

$$(1 + 0.5z)(1 + \pi_1 z + \pi_2 z^2 + \ldots + \pi_j z^j + \ldots) = (1 - 0.9z)$$

en este caso se tiene la siguiente expresión para los coeficientes $\pi_j = (-1)^j (1.4)(0.5)^{j-1}$ para $j \geq 1$. Con estos resultados la Ecuación D.51 se puede escribir

$$x_t = 1.4 \sum_{j=1}^{\infty} (-0.5)^{j-1} x_{t-j} + w_t$$

D.5.5. ARIMA

Sea la siguiente serie de tiempo, conocida como *random walk*

$$x_t = \delta t + \sum_{j=1}^{t} w_j$$

con condición inicial $x_0 = 0$.
x_t no es estacionaria, pues su media depende del tiempo t

$$E[x_t] = \delta t + E[\sum_{j=1}^{t} w_j] = \delta t$$

sin embargo la diferencia de esta serie sí es estacionaria

$$\nabla x_t = x_t - x_{t-1} = (\delta t + \sum_{j=1}^{t} w_j) - (\delta(t-1) + \sum_{j=1}^{t-1} w_j) = \delta + w_t$$

con media $\mu = \delta$. Esto motiva a la definición del modelo ARIMA, *AutoRegressive Integrated Moving Average*.

Definición D.5.15. ARIMA Un proceso x_t dícese ser ARIMA(p,d,q) si el modelo con diferencia de orden d:

$$\nabla^d x_t = (1 - B)^d x_t$$

es ARMA(p,q). En general el modelo se escribe como

$$\phi(B)(1 - B)^d x_t = \theta(B) w_t \tag{D.53}$$

si $E[\nabla^d x_t] = \mu$, el modelo se escribe

$$\phi(B)(1 - B)^d x_t = \delta + \theta(B) w_t \tag{D.54}$$

con $\delta = \mu(1 - \phi_1 - \phi_2 - \ldots - \phi_p)$.

D.5.6. SARIMA

A menudo la dependencia con el pasado tiende a ocurrir más fuertemente con puntos en el tiempo múltiplos de un valor s estacional. Por ejemplo, para datos económicos mensuales existe un fuerte componente anual, en este caso el valor $s = 12$ indica una dependencia anual. Mientras que para datos tomados trimestralmente $s = 4$ denota el periodo anual.
El modelo $ARMA(P, Q)_s$ *puramente estacional*, toma la siguiente forma

$$\Phi_P(B^s) x_t = \Theta_Q(B^s) w_t \tag{D.55}$$

donde los operadores

$$\Phi_P(B^s) = 1 - \Phi_1 B^s - \Phi_2 B^{2s} - \ldots - \Phi_P B^{Ps} \tag{D.56}$$

y

$$\Theta_Q(B^s) = 1 + \Theta_1 B^s + \Theta_2 B^{2s} + \ldots + \Theta_Q B^{Qs} \tag{D.57}$$

son los operadores *autoregresivos* y de *media móvil* respectivamente.

Ejemplo D.5.16. AR(1) estacional
Considérese la serie autoregresiva estacional de primer orden

$$(1 - \Phi B^{12}) x_t = w_t$$

o

$$x_t = \Phi x_{t-12} + w_t$$

este modelo exhibe al valor de la serie x_t en término de los puntos, en el tiempo pasado, múltiplos de $s = 12$. Si el intervalo de tiempo en que se toman los valores es mensual, s representaría un periodo anual.
Usando la expresión de la autocovarianza para AR(1), deducida en el Ejemplo D.5.4: $\gamma(h) = \frac{\sigma_w^2 \phi^h}{1-\phi^2}$ se tiene el siguiente resultado

$$\gamma(0) = \sigma_w^2/(1 - \Phi^2)$$

$$\gamma(\pm 12k) = \sigma^2 \Phi^k/(1 - \Phi^2), \quad k = 1, 2, \ldots$$

$$\gamma(h) = 0, \quad \text{en otro caso}$$

estos resultados pueden ser generalizados de la siguiente manera

$$\gamma(h) = \Phi\gamma(h - 12)$$

para este caso las correlaciones no negativas son

$$\rho(\pm 12k) = \Phi^k, \quad k = 1, 2, \ldots$$

en la Figura D.4 se puede ver, en la gráfica de arriba, el resultado de la simulación del modelo autoregresivo estacional de orden 1, con periodo $s = 12$: $x_t = 0.9x_{t-12} + w_t$. Las dos gráficas de abajo muestran el ACF y PACF. Obsérvese que la correlación disminuye con los años pasados, en concordancia con la expresión de la correlación: $\rho(k) = 0.9^k$.

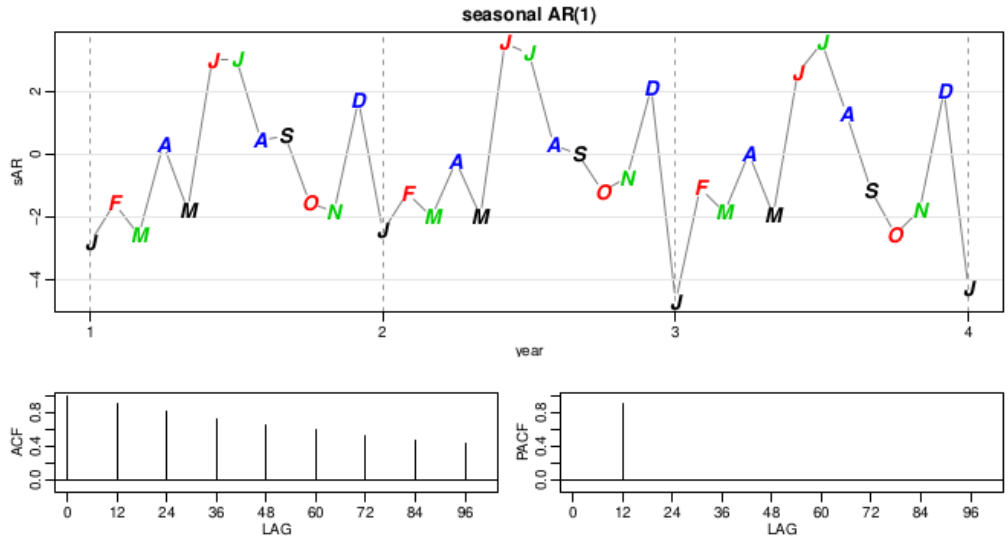


Figura D.4: Modelo $x_t = 0.9x_{t-12} + w_t$. En la gráfica del arriba, cada punto es representado por la letra del mes en que se registró la medición. El eje x representa a los años. El eje y representa al valor arrojado por el modelo para cada punto del tiempo en la serie. Imagen extraída de la página 149 del Libro [14].

En general se pueden combinar los operadores estacionales y no estacionales para dar paso a un modelo *multiplicativo*:

$$\Phi_P(B^s)\phi(B)x_t = \Theta_Q(B^s)\theta(B)w_t. \tag{D.58}$$

Ejemplo D.5.17. Modelo $ARMA(0,1) \times (1,0)_{12}$
Considérese el siguiente modelo

$$x_t = \Phi x_{t-12} + w_t + \theta w_{t-1}$$

donde $|\Phi| < 1$ y $|\theta| < 1$. Entonces debido a que w_t, w_{t-1} y x_{t-12} no están correlacionados se tiene que

$$\gamma(0) = cov(x_t, x_t) = var(\Phi x_{t-12} + w_t + \theta w_{t-1}, \Phi x_{t-12} + w_t + \theta w_{t-1})$$

$$= \Phi^2 var(x_{t-12}, x_{t-12}) + var(w_t, w_t) + \theta^2 var(w_{t-1}, w_{t-1})$$

$$= \Phi^2\gamma(0) + \sigma_w^2 + \theta^2\sigma_w^2$$

o

$$\gamma(0) = \frac{1+\theta^2}{1-\Phi^2}\sigma_w^2$$

en adición multiplicando el modelo por x_{t-h}, $h > 0$ y tomando el valor esperado se tiene

$$E[x_t x_{t-h}] = \Phi E[x_{t-12}x_{t-h}] + E[w_t x_{t-h}] + \theta E[w_{t-1}x_{t-h}]$$

$$\gamma(h) = \Phi\gamma(h-12) + E[w_t x_{t-h}] + \theta E[w_{t-1}x_{t-h}]$$

y si $h = 1$

$$\gamma(1) = \Phi\gamma(-11) + E[w_t x_{t-1}] + \theta E[w_{t-1}x_{t-1}]$$

$$= \Phi\gamma(-11) + E[w_t(\Phi x_{t-13} + w_{t-1} + \theta w_{t-2})] + \theta E[w_{t-1}(\Phi x_{t-13} + w_{t-1} + \theta w_{t-2})]$$

$$= \Phi\gamma(-11) + 0 + \theta E[w_{t-1}w_{t-1}] = \Phi\gamma - 11) + \theta\sigma_w^2$$

usando la relación obtenida en el ejemplo D.5.16: $\gamma(h) = \Phi\gamma(h-12)$ se tiene

$$\gamma(1) = \Phi^2\gamma(1) + \theta\sigma_w^2$$

$$\rightarrow \gamma(1) = \frac{\theta\sigma_w^2}{1-\Phi^2}$$

si $h \geq 2$, la expresión de la autocovarianza es más sencilla de obtener

$$\gamma(h) = \Phi\gamma(h-12)$$

con estos resultados, la función de autocorrelación queda así

$$\rho(12h) = \phi^h, \quad h = 1, 2, \ldots$$

$$\rho(12h \pm 1) = \frac{\theta}{1+\theta^2}\Phi^h, \quad h = 0, 1, 2, \ldots$$

$$\rho(h) = 0, \quad \text{en otro caso}$$

en la Figura D.5 se visualiza el ACF y PACF para el modelo con $\Phi = 0.8$ y $\theta = -0.5$.

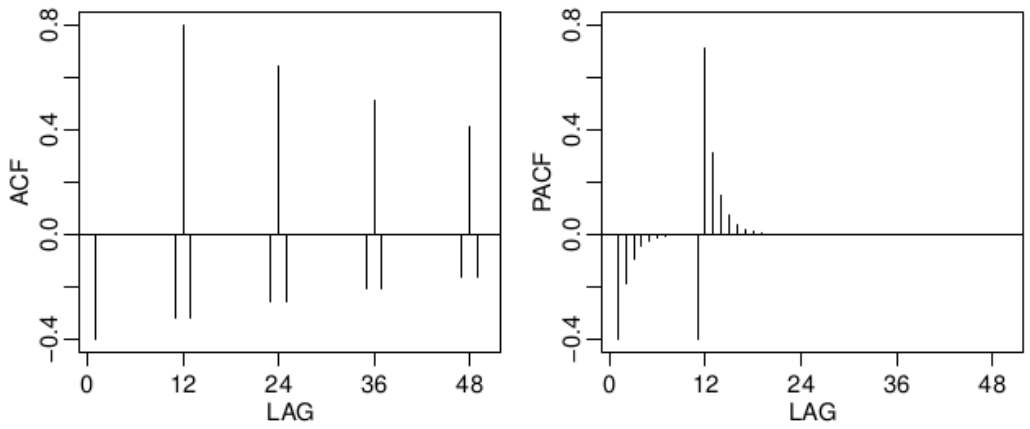


Figura D.5: Modelo $x_t = 0.8x_{t-12} + w_t - 0.5w_{t-1}$. Obsérvese que como el coeficiente $\theta = -0.5$ es negativo, y $\Phi = 0.8$ positivo, la autocorrelacion $\rho(12h \pm 1)$ determina una correlación negativa entre el tiempo actual t respecto a los tiempos $t - (12h \pm 1)$. Imagen extraída de la página 151 del Libro [14].

La extensión del modelo *multiplicativo* $ARMA(p, q) \times (P, Q)_s$, para series no estacionarias implica incorporar el operador de diferenciación ∇.

Definición D.5.18. $ARIMA(p, d, q) \times (P, D, Q)_s$ (SARIMA)
El modelo *multiplicativo estacional autoregresivo integrado de media móvil*, o SARIMA por sus siglas en inglés, viene dado por

$$\Phi_P(B^s)\phi(B)\nabla^D\nabla^d x_t = \gamma + \Theta_Q(B^s)\theta(B)w_t \tag{D.59}$$

donde $w_t \sim iidN(0, \sigma_w^2)$. El modelo general es denotado $ARIMA(p, d, q)\times(P, D, Q)_s$. Los componentes autoregresivos y de media móvil no estacionales son representados, por los polinomios de orden p y q, $\phi(B)$ y $\theta(B)$, respectivamente. Mientras que los componentes autoregresivos y de media móvil estacionales son representados por los polinomios de orden P y Q, $\Phi_P(B)$ y $\Theta_Q(B)$, respectivamente. Las componentes de diferenciación ordinaria y estacional quedan representadas por $\nabla^d = (1 - B^d)$ y $\nabla^D = (1 - B^s)^D$, respectivamente.

D.5.7. Forecasting

En los apartados anteriores se describieron los modelos AR, MA, ARMA, ARIMA y SARIMA, para representar series de tiempo. Ahora se verán dos técnicas de forecasting para resolver el problema de pronosticar el valor de x_{n+m}, $m =$

$1, 2, \ldots$, basado en los datos colectados hasta el presente $x_{1:n} = \{x_1, \ldots, x_n\}$. Se asume que los parámetros del problema son conocidos. Ver Sección 3.5 del Libro [14] para conocer detalles sobre la estimación de los parámetros.
Sea el siguiente resultado

Proposición D.5.19. La función $g(x)$ que minimiza el error cuadrático medio

$$MSE = E[(y - g(x))^2] \tag{D.60}$$

con x e y variables aleatorias conjuntamente distribuidas y con función de densidad $f(x, y)$ es

$$g(x) = E[y|x].$$

Recurriendo a este resultado se tiene entonces que el estimador que minimiza el error cuadrático medio de

$$MSE = E[(x_{n+m} - g(x_{1:n}))^2] \tag{D.61}$$

es

$$x^n_{n+m} = E[x_{n+m}|x_{1:n}]$$

pero en primera instancia restrínjase el estimador a la siguiente forma lineal

$$x^n_{n+m} = \alpha_0 + \sum_{k=1}^{n} \alpha_k x_k \tag{D.62}$$

con $\alpha_0, \ldots, \alpha_n$ números reales. Los predictores lineales que minimizan el error cuadrático medio dado en la Ecuación D.61, se denominan *mejores predictores lineales*, (BLP) por su sigla en inglés.
Una forma de resolver el problema con predictores lineales es pensar en las $x_{1:n}$ observaciones, como una base de un subespacio vectorial W y en los $\alpha_0, \ldots, \alpha_n$ como los coeficientes del vector $w \in W$ que a menor distancia de x_{n+m} se encuentra. Es decir w es la proyección de x_{n+m} en W. La siguiente propiedad es un resultado de esto.

Proposición D.5.20. Dados $x_1, \ldots, x_n$, el *mejor predictor lineal*,

$$x^n_{n+m} = \alpha_0 + \sum_{k=1}^{n} \alpha_k x_k$$

de x_{n+m} para $m \geq 1$, se encuentra resolviendo

$$E[(x_{n+m} - x^n_{n+m})x_k] = 0, \quad k = 0, 1, \ldots, n \tag{D.63}$$

donde $x_0 = 1$, para $\alpha_0, \ldots, \alpha_n$.

Si $E[x_t] = \mu$, la primera ecuación en Ec. D.63 ($k = 0$) implica que

$$E[x_{n+m}] = E[x^n_{n+m}] = \mu$$

tomando el valor esperado en Ec. D.62 se tiene que

$$\mu = \alpha_0 + \sum_{k=1}^{k} \alpha_k \mu \;\; \text{o} \;\; \alpha_0 = \mu(1 - \sum_{k=1}^{n} \alpha_k)$$

por lo que la forma del *mejor estimador lineal* es

$$x^n_{n+m} = \mu + \sum_{k=1}^{n} \alpha_k (x_k - \mu)$$

a continuación, sin pérdida de generalidad, se analizará el caso en que $\mu = 0$.
Considérese la predicción de un solo paso. Dados $\{x_1, \ldots, x_n\}$, se desea predecir x_{n+1}. La forma del *mejor estimador lineal* es

$$x^n_{n+1} = \phi_{n1} x_n + \phi_{n2} x_{n-1} + \ldots + \phi_{nn} x_1 \tag{D.64}$$

usando la Proposición D.5.20, los coeficientes $\{\phi_{n1}, \ldots, \phi_{nn}\}$ satisfacen

$$E[(x_{n+1} - \sum_{j=1}^{n} \phi_{nj} x_{n+1-j}) x_{n+1-k}] = 0, \;\; k = 1, 2, \ldots, n$$

o

$$\sum_{j=1}^{n} \phi_{nj} \gamma(k - j) = \gamma(k), k = 1, 2, \ldots, n. \tag{D.65}$$

Las ecuaciones de predicción D.65 pueden ser escritas en forma matricial

$$\Gamma_n \phi_n = \gamma_n \tag{D.66}$$

donde $\Gamma_n = \{\gamma(k - j)\}^n_{k,j=1}$ es una matriz $n \times n$, $\phi_n = (\phi_{n1}, \ldots, \phi_{nn})$ es un vector de dimensiones $n \times 1$ y $\gamma_n = (\gamma(1), \ldots, \gamma(n))$ un vector de dimensiones $n \times 1$.
Para los modelos ARMA, la matriz Γ_n es semidefinida positiva y se tiene que

$$\phi_n = \Gamma_n^{-1} \gamma_n. \tag{D.67}$$

El error cuadrático medio para la predicción en un paso es

$$P^n_{n+1} = E[x_{n+1} - x^n_{n+1}]^2 = \gamma(0) - \gamma_n^T \Gamma_n^{-1} \gamma_n. \tag{D.68}$$

Ejemplo D.5.21. Predicción para AR(2)
Sea el proceso $x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + w_t$. Supóngase que se quiere predecir el valor x_3 basado en las observaciones x_1 y x_2. Entonces usando la Ecuación matricial D.66 para $x^2_{2+1} = x^2_3 = \phi_{21} x_2 + \phi_{22} x_1$, se resuelven los estimadores ϕ_{21}, ϕ_{22}

$$\begin{bmatrix} \phi_{21} \\ \phi_{22} \end{bmatrix} = \begin{bmatrix} \gamma(0) & \gamma(1) \\ \gamma(1) & \gamma(0) \end{bmatrix}^{-1} \begin{bmatrix} \gamma(1) \\ \gamma(2) \end{bmatrix}$$

De hecho, a partir las ecuaciones de prediccion Ec. D.63, se deduce que $\phi_1 = \phi_{21}$ y $\phi_2 = \phi_{22}$:

$$E[(x_3 - (\phi_1 x_2 + \phi_2 x_1))x_1] = E[w_3 x_1] = 0$$

$$E[(x_3 - (\phi_1 x_2 + \phi_2 x_1))x_2] = E[w_3 x_2] = 0$$

Y por unicidad de coeficientes de la proyección dada en la Proposición D.5.20, se tienen las dos igualdades.

Ahora se verá una técnica que define un predictor $\widetilde{x}^{n+m}_n$ en cuya definición se considera la estructura del proceso ARMA. Defínase, en primera instancia, x_t como un proceso ARMA(p,q), causal e invertible, $\theta(B)x_t = \phi(B)w_t$, con $w_t \sim iidN(0, \sigma^2_w)$. En el caso de media $\mu \neq 0$ reemplazar x_t por $x_t - \mu$. Sea

$$x^n_{n+m} = E[x_{n+m}|x_n, \ldots, x_1]$$

el predictor que minimiza el error medio cuadrático de x_{n+m}, basado en las observaciones $\{x_n, \ldots, x_1\}$.
Sea a su vez

$$\widetilde{x}^n_{n+m} = E[x_{n+m}|x_n, \ldots, x_1, x_0, x_{-1}, \ldots]$$

el predictor de x_{n+m}, basado en el pasado infinito $\{x_n, \ldots, x_1, x_0, x_{-1}, \ldots\}$. En general $\widetilde{x}^n_{n+m}$ no es igual que x^n_{n+m}, pero para muestras grandes, $\widetilde{x}^n_{n+m}$ puede ser una buena aproximacion de x^n_{n+m}.
Ahora escríbase x_{n+m} en su forma causal e invertible

$$x_{n+m} = \sum_{j=0}^{\infty} \psi_j w_{n+m-j}, \quad \psi_0 = 1 \tag{D.69}$$

$$w_{n+m} = \sum_{j=0}^{\infty} \pi_j x_{n+m-j}, \quad \pi_0 = 1 \tag{D.70}$$

Tomando el valor esperado condicional para la Ecuación D.69, se tiene

$$\widetilde{x}_{n+m} = \sum_{j=0}^{\infty} \psi_j \widetilde{w}_{n+m-j} = \sum_{j=m}^{\infty} \psi_j w_{n+m-j} \tag{D.71}$$

en donde la expresión del lado derecho en la segunda igualdad se deduce debido a la causalidad e invertibilidad del proceso

$$\widetilde{w}_t = E[w_t|x_n, x_{n-1}, \ldots, x_0, x_{-1}, \ldots] = \begin{cases} 0 & t > n \\ w_t & t \leq n \end{cases}$$

Tomando el valor esperado condicional ahora para la Ecuación D.70, se tiene

$$0 = \widetilde{x}_{n+m} + \sum_{j=1}^{\infty} \pi_j \widetilde{x}_{n+m-j}$$

o

$$\widetilde{x}_{n+m} = -\sum_{j=1}^{m-1} \pi_j \widetilde{x}_{n+m-j} - \sum_{j=m}^{\infty} \pi_j x_{n+m-j} \tag{D.72}$$

usando el hecho de que $E[x_t|x_n, x_{n-1}, \ldots, x_0, x_{-1}, \ldots] = x_t$, para $t \leq n$. La predicción por lo tanto puede ser hecha recursivamente con la Ecuación D.72. Comenzando por $m = 1$.
Usando la Ecuación D.71 y la Ecuación D.69, se tiene que

$$x_{n+m} - \widetilde{x}_{n+m} = \sum_{j=0}^{m-1} \psi_j w_{n+m-j}$$

de manera que el predictor para el error medio cuadrático P_{n+m}^n puede ser escrito como

$$P_{n+m}^n = E[(x_{n+m} - \widetilde{x}_{n+m})^2] = \sigma_w^2 \sum_{j=0}^{m-1} \psi_j^2 \tag{D.73}$$

La Ecuación D.72 debe ser usada mediante truncamiento, dado a que no se observan los valores $x_0, x_{-1}, \ldots$
Esto se hace igualando a cero $\sum_{j=n+m}^{\infty} \pi_j x_{n+m-j} = 0$. El predictor truncado entonces es escrito así:

$$\widetilde{x}_{n+m}^n = -\sum_{j=1}^{m-1} \pi_j \widetilde{x}_{n+m-j}^n - \sum_{j=m}^{n+m-1} \pi_j x_{n+m-j}^n \tag{D.74}$$

Ejemplo D.5.22. Pronósticos a largo plazo
Sea x_t con media μ distinta de 0. Reemplazando x_{n+m} por $x_{n+m}-\mu$ en la Ecuación D.69 y tomando el valor esperado condicional, como en la Ecuación D.71, se tiene

$$\widetilde{x}_{n+m} = \mu + \sum_{j=m}^{\infty} \psi_j w_{n+m-j}$$

Los ψ_j tienden exponencialmente a cero, con la consecuencia de que $\widetilde{x}_{n+m} \to \mu$ cuando $m \to \infty$.
Mientras que el predictor del error $P^n_{n+m} \to \sigma^2_w \sum_{j=0}^{\infty} \psi_j^2 = \gamma_x(0) = \sigma^2_x$. De donde se observa que el proceso ARMA tiende al valor de la media, con una predicción del error constante, a medida que el horizonte del pronóstico, m, crece. Véase la Figura D.6, para un horizonte de predicción de dos años (m=24).

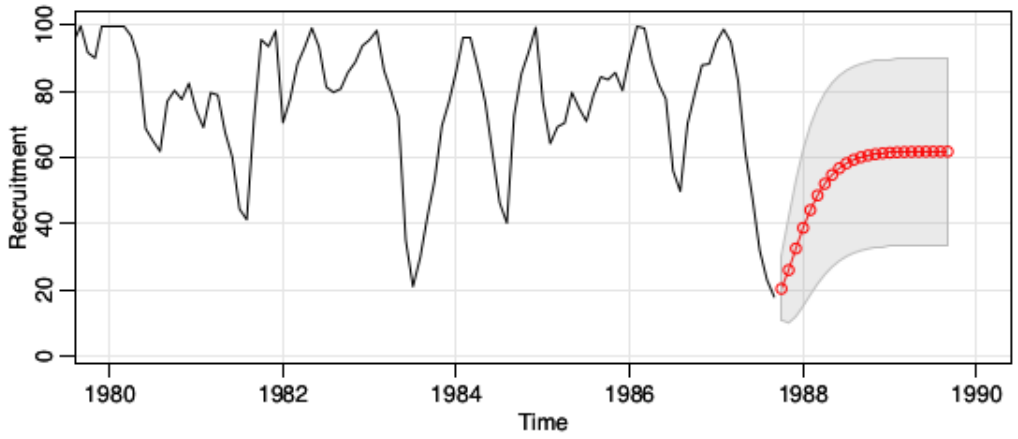


Figura D.6: 24 meses de predicción para una serie de reclutamientos (número de pescados nuevos por mes) proporcionados por el Grupo de Pesca Ambiental del Pacífico (PEFC, por sus siglas en inglés). Los datos recolectados van desde Enero de 1980 a Septiembre de 1987. Imagen extraída de la página 112 del Libro [14].

D.5.8. Comportamiento de ACF y PACF

Las funciones de autocorrelación (ACF) y de autocorrelación parcial (PACF), en general, son de ayuda para determinar los ordenes del modelo ARIMA para una serie de tiempo. La Tabla D.1 resume algunas heurísticas.

	$AR(p)$	$MA(q)$	$ARMA(p,q)$
ACF	Desvanecimiento paulatino de las LAGS	Corte abrupto de las LAGS en el orden q	Desvanecimiento paulatino de las LAGS
PACF	Corte abrupto luego de las LAGs en el orden p	Desvanecimiento paulatino de las LAGS	Desvanecimiento paulatino de las LAGS
	$AR(P)_s$	$MA(Q)_s$	$ARMA(P,Q)_s$
ACF*	Desvanecimiento paulatino en las LAGS ks, $k = 1, 2, \ldots$	Corte abrupto luego de la LAG Qs	Desvanecimiento paulatino en las LAGS ks
PACF*	Corte abrupto luego de la LAG Ps	Desvanecimiento paulatino en las LAGS ks, $k = 1, 2, \ldots$.	Desvanecimiento paulatino en las lags ks
$*$Los valores de las LAGS no estacionales $h \neq ks$, para $k = 1, 2, \ldots$ valen cero.			

Tabla D.1

D.6. Exponential Smooth

En esta sección se describen los tres métodos más conocidos de suavizado exponencial: *Simple Exponential Smoothing* (SES), *Holt's linear trend method*, *Holt-Winter's seasonal method*.

D.6.1. Simple Exponential Smoothing

Una serie de tiempo con media cambiando lentamente con el tiempo puede ser modelada de la siguiente manera

$$X_t = \mu_t + w_t$$

Con w_t representando ruido blanco con distribución $w_t \sim w(0, \sigma_w^2)$ y μ_t cambiando lentamente con el tiempo. Sea l_t el estimador de μ_t, para un tiempo t. Dado l_{t-1} y una vez observado el valor de X_t, SES actualiza el nivel del estimador vía la ecuación

$$l_t = \alpha X_t + (1 - \alpha) l_{t-1} \tag{D.75}$$

siendo α un parámetro de suavizado tomando valores en el rango $(0, 1]$.
El pronóstico para el tiempo $t + h$ hecho en el tiempo t por SES es

$$\hat{X}_{t+h}^{1:t} = l_t, \quad h > 0 \tag{D.76}$$

y el error correspondiente

$$w_{t+h} = X_{t+h} - \hat{X}_{t+h}^{1:t} = X_{t+h} - l_t \tag{D.77}$$

La Ecuación de recurrencia D.75 tiene la siguiente ecuación equivalente

$$l_t = l_{t-1} + \alpha w_t \tag{D.78}$$

con el error

$$w_t = X_t - \hat{X}_t^{t-1} = X_t - l_{t-1} \tag{D.79}$$

mediante sustitución sucesiva de los estimadores l_i en la Ecuación D.78 se llega a la siguiente expresión para l_t

$$l_t = \alpha \sum_{i=0}^{\infty} (1-\alpha)^i X_{t-i}$$

en donde se observa que el estimador puede ser escrito en función de las observaciones pasadas multiplicadas por un peso que declina exponencialmente con el tiempo, debido a ello el nombre *exponential smoothing*.

D.6.2. Holt's method

SES no performa tan bien para series de tiempo con una tendencia lineal del siguiente tipo

$$X_t = \mu_t + w_t = \beta_0 + \beta_1 t + w_t$$

con β_0 y β_1 constantes para un rango de tiempo, pero pudiendo variar lentamente con el paso del tiempo. El método Holt para una tendencia lineal agrega un estimador para la pendiente β_1. Quedando las ecuaciones de recurrencia así

$$l_t = \alpha_1 X_t + (1-\alpha_1)(l_{t-1} + b_{t-1}) \tag{D.80}$$

$$b_t = \alpha_2 (l_t - l_{t-1}) + (1-\alpha_2) b_{t-1} \tag{D.81}$$

con α_1, α_2 parámetros de suavizado tomando valores en el intervalo $(0, 1]$.
El pronóstico para el tiempo $t+h$ hecho en el tiempo t por el método Holt es

$$\hat{X}_{t+h}^{1:t} = l_t + hb_t, \quad h > 0 \tag{D.82}$$

con el error asociado

$$w_{t+h} = X_{t+h} - \hat{X}_{t+h}^{1:t} = X_{t+h} - l_t - hb_t \tag{D.83}$$

Al igual que en SES, los estimadores del método Holt pueden ser reescritos en las formas equivalentes

$$l_t = l_{t-1} + b_{t-1} + \alpha_1 w_t \tag{D.84}$$

$$b_t = b_{t-1} + \alpha_1 \alpha_2 w_t \tag{D.85}$$

con $w_t = X_t - l_{t-1} - hb_{t-1}$ el error para un paso de predicción.
El pronóstico generado por este método muestra una tendencia constante (en aumento o en disminución) indefinida hacia el futuro. Gardner y McKenzie (1985) introdujeron un parámetro que amortigua la tendencia, transformándola, en el futuro, a una línea plana. En conjunción con los parámetros de suavizado α_1 y α_2 el nuevo método agrega un parámetro de amortiguación $0 < \phi < 1$. Las ecuaciones se reescriben de la siguiente manera:

$$\hat{X}^{1:t}_{t+h} = l_t + (\phi + \phi^2 + \ldots + \phi^h) b_t, \quad h > 0 \tag{D.86}$$

$$l_t = \alpha_1 X_t + (1 - \alpha_1)(l_{t-1} + \phi b_{t-1}) \tag{D.87}$$

$$b_t = \alpha_2 (l_t - l_{t-1}) + (1 - \alpha_2)\phi b_{t-1} \tag{D.88}$$

Observar que para los valores de ϕ la predicción converge a la siguiente expresión:

$$l_t + \frac{\phi}{1 - \phi} b_t$$

a medida que $h \to \infty$.

D.6.3. Holt-Winter's method

Este método es apropiado para manejar series de tiempo con cambios estacionales. Una serie de tiempo con tendencia lineal y estacionalidad aditiva puede ser representada por el modelo

$$X_t = \beta_0 + \beta_1 t + s_t + w_t$$

con s_t siendo el factor estacional para el tiempo t.
Denotando como c_t al estimador de s_t, las ecuaciones para actualizar los estimadores están dadas por:

$$l_t = l_{t-1} + b_{t-1} + \alpha_1 w_t \tag{D.89}$$

$$b_t = b_{t-1} + \alpha_1\alpha_2 w_t \tag{D.90}$$

$$c_t = c_{t-M} + (1-\alpha_1)\alpha_3 w_t \tag{D.91}$$

en donde M es el tamaño del ciclo estacional completo, α_1,α_2,α_3 son parámetros de suavizado tomados del intervalo $(0, 1]$.
El pronóstico para el tiempo $t + h$ en el tiempo t es

$$\hat{X}^{1:t}_{t+h} = l_t + hb_t + c_{t-M+h} \tag{D.92}$$

D.7. Random Forest

Random forest es una técnica de *ensamblaje*, es decir las predicciones de múltiples algoritmos son combinadas para generar una predicción conjunta más exacta que cualquier predicción de cada algoritmo individual.
Los bloques constructivos de esta técnica son los árboles de desición. Durante la fase de entrenamiento el modelo es ajustado con *features* o datos históricos. Las relaciones entre los datos y los resultados son aprendidos y un árbol de desición, como el de la Figura D.7, es generado.

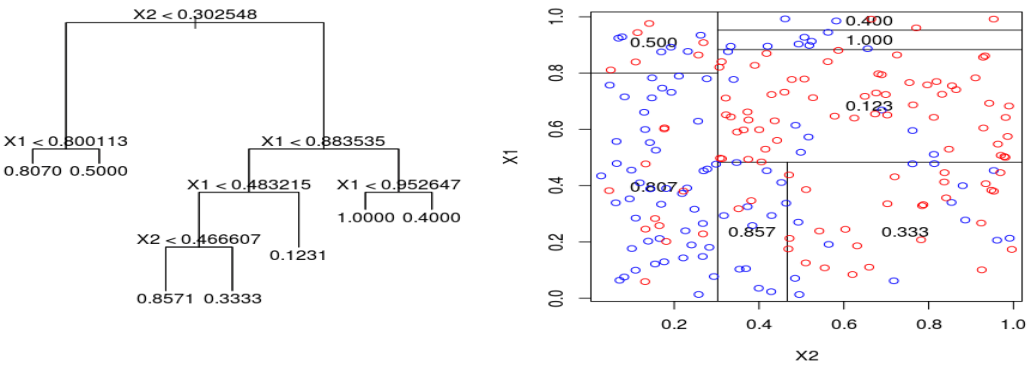


Figura D.7: Dados dos rasgos numéricos $X1$, $X2$, en cada nodo se decide con que rasgos tomar la desición y cual es el criterio; en el ejemplo el criterio es la relación < respecto al valor que mejor separe los datos. Imágen extraída del Artículo online [8]

Random forest opera construyendo múltiples árboles de desición durante la etapa de entrenamiento y arrojando como resultado el promedio de cada una de las predicciones de los árboles individuales.

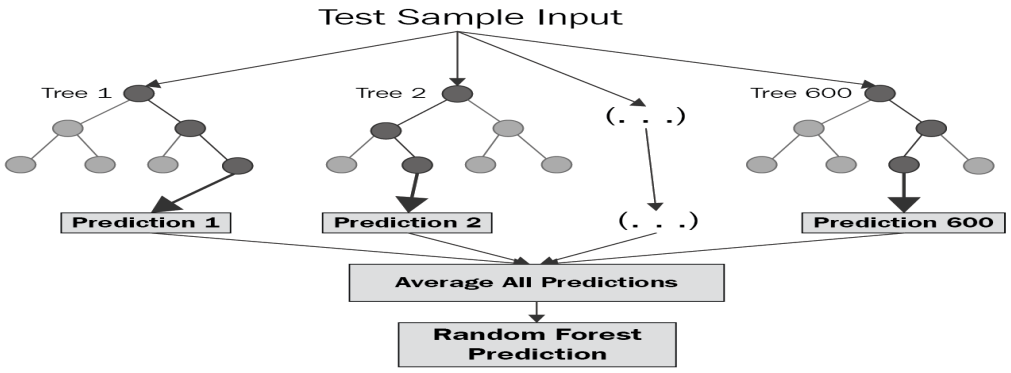


Figura D.8: La figura muestra un ejemplo con 600 árboles de desición no correlacionados. Cada uno realiza una predicción. La técnica predice con el valor medio de las 600 predicciones individuales. Imágen extraída del Artículo online [7]

Una de las claves de la técnica radica en que los árboles no estén correlacionados. Esto le brinda una mayor pluralidad al modelo y evita la preponderancia de grupos altamente correlacionados al momento de la predicción. Para lograr esto *random forest* emplea las siguientes técnicas.

Bagging

Cada árbol, durante el entrenamiento, elige aleatoriamente muestras de tamaño N con reemplazo. Por ejemplo si los datos de entrenamiento son $[1, 2, 3, 4, 5, 6]$, entonces uno de los árboles puede elegir el siguiente conjunto para entrenar $[1, 2, 2, 3, 6, 6]$, en donde los valores fueron elegidos al azar y con reemplazo, por ello la repetición del 2 y el 6.

Aleatoriedad en la elección del rasgo

A su vez hay aleatoriedad en los rasgos de desición, por ejemplo si un árbol en un nodo tiene los rasgos de la Tabla D.2 para elegir uno cualquiera, tiene total libertad de elegir el mejor rasgo para el nodo actual de desición, en este caso el mejor rasgo es el que está subrayado:

Rasgo1
Rasgo2
Rasgo3
Rasgo4

Tabla D.2: Tabla extraída del Artículo online [8].

Sin embargo para un árbol de desición, en cada nodo, aleatoriamente se elige

un subconjunto de rasgos, que puede contener o no al mejor para tomar la desición. En la Tabla D.3 se ven dos elecciones aleatorias de rasgos, una conteniendo el mejor rasgo y otra no.

Árbol 1	**Árbol 2**
Rasgo1	Rasgo2
Rasgo3	Rasgo3

Tabla D.3: Tabla extraída del Artículo online [8].

Apéndice E

Matrix Completion

El completado de una matriz a partir de unas pocas entradas es un problema con aplicaciones en *machine learning*, estadística y procesamiento de señales. En este apartado se define el problema. Se describe el algoritmo de completado de matrices (ICMC). El algoritmo ICMC puede reducirse a un problema análogo sobre grafos, esta reducción permite llegar a conclusiones, sobre la efectividad del algoritmo respecto ciertas clases de grafos. Luego se presentan los resultados experimentales hechos en el trabajo de Meka et al. [13]. Por último se brindan detalles de la implementación hecha en Python para este proyecto.

E.1. Formalización del problema

Antes de formalizar el problema defínase la función $\mathcal{P}_\Omega : \mathcal{R}^{m\times n} \to \mathcal{R}^{m\times n}$, como la proyección de una matriz X sobre los pares de índices especificados en $\Omega \subseteq [m] \times [n]$:

$$\mathcal{P}_\Omega(X) = \begin{cases} X[i,j] & (i,j) \in \Omega \\ \text{indefinido} & \text{en otro caso} \end{cases} \tag{E.1}$$

Definición E.1.1. Sea $M \in \mathcal{R}^{m\times n}$, una matriz de rango al menos k. Dados $\Omega \subseteq [m] \times [n]$, $\mathcal{P}_\Omega(M)$ y k. *El problema de completado de matrices de rango bajo* es encontrar una matriz $X \in \mathcal{R}^{m\times n}$ tal que

$$rango(X) \leq k, \quad \text{y} \quad \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M) \tag{E.2}$$

E.2. Algoritmo ICMC

Considérese la siguiente formulación del *problema de completado de matrices de rango bajo*: Dados k, Ω, $\mathcal{P}_\Omega(M)$ para una matriz M de rango k, encontrar X, Y tales que

$$\mathcal{P}_\Omega(XY^T) = \mathcal{P}_\Omega(M), \quad X \in \mathcal{R}^{m\times k}, \quad Y \in \mathcal{R}^{m\times k} \tag{E.3}$$

Se asume que la matriz objetivo M *no degenera* en el siguiente sentido:

Definición E.2.1. Una matriz M de rango k *no degenera* si existen $X \in \mathcal{R}^{m\times k}$ e $Y \in \mathcal{R}^{m\times k}$ con $M = XY^T$, tales que cualesquiera k filas de X son linealmente independientes y cualesquiera k filas de Y son linealmente independientes.

Sea $G_\Omega = (U, V, \Omega)$ un grafo bipartito con vértices $U = \{u_1, \ldots, u_m\}$, $V = \{v_1, \ldots, v_n\}$ y aristas dadas por los pares ordenados en Ω. G_Ω es el grafo asociado al problema.
El método de ICMC progresivamente va computando las filas de X e Y, satisfaciendo en cada progreso la Ecuación E.2. El vértice $u_i \in U$ dícese *infectado* si la i-ésima fila de X ha sido computada. Análogamente el vértice $v_j \in V$ dícese *infectado* si la j-ésima fila de Y ha sido computada.
Supóngase ahora un paso intermedio de la iteración, en donde los vértices $L \subseteq U$ y $R \subseteq V$ han sido marcados como *infectados*. Es decir las filas de X con índice en L y las filas en Y con índices en R han sido computadas.
Ahora, sea el caso de un vértice no infectado, con índice $j \in [n]$. Para computar la fila correspondiente en Y: $y_j^T \in R^k$, son necesarias k ecuaciones linealmente independientes. Es decir si M *no degenera*, para computar y_j^T solo son necesarias k entradas de la j-ésima columna de M con índices para las filas en L. Traduciendo esta condición en términos del grafo G_Ω, y_j^T puede ser computada y v_j marcado como *infectado* si hay al menos k aristas de v_j a vértices en L. Análogamente x_i^T puede ser computado y el vértice $u_i \in U$ marcado como *infectado* si hay al menos k aristas de u_i a vértices en R.
Sin pérdida de generalidad, para una matriz M que *no degenera*, al principio para iniciar el proceso de infección se pueden fijar k filas de X que sean iguales a la matriz identidad $I_{k\times k}$. Sean L_0 las filas de X elegidas para esta fijación. El proceso de infección en cascada de los vértices en U y V y la computación de las filas en X e Y, se especifican en el Algoritmo 11 ICMC.

Algoritmo 11 $ICMC(G_\Omega, \mathcal{P}_\Omega(M), L_0)$

1: Comenzar con los conjuntos infectados $L = L_0 \subseteq U$, $R = \emptyset$. Inicializar la sumbmatriz $k \times k$ de X con filas en L_0 igual a la identidad I_k.
2: **repeat**
3: Marcar como infectados a todos los vértices en V que tienen al menos k aristas a los vértices infectados L. Agregar los nuevos infectados a R.
4: Para cada nuevo vértice infectado $v_j \in R$, computar la j-ésima fila de Y, usando las entradas de M que corresponden a las aristas de v_j a L.
5: Marcar como infectados a todos los vértices en U que tienen al menos k aristas a los vértices infectados R. Agregar los nuevos infectados a L.
6: Para cada nuevo vértice infectado $u_i \in L$, computar la i-ésima fila de X, usando las entradas de M que corresponden a las aristas de u_i a R.
7: **until** Convergencia

En base a la representación de grafos del proceso en cascada se puede plantear la siguiente abstracción. Sea $G = (W, E)$ un grafo no dirigido, fíjense $A \subseteq W$ y $k > 0$. Se define $\sigma_{G,k}(A, 0) = A$, e inductivamente

$$\sigma_{G,k}(A, t+1) = \sigma_{G,k}(A, t) \cup \{u \in W : u \text{ tiene al menos } k \text{ aristas a } \sigma_{G,k}(A, t)\}$$

Definición E.2.2. La influencia de un conjunto $A \subseteq W$, $\sigma_{G,k}(A)$ es el número de vértices infectados por el proceso en cascada hasta su terminación, comenzando por los vértices iniciales A. Es decir $\sigma_{G,k}(A) = |\cup_t \sigma_{G,k}(A, t)|$. Se dice que A es *un conjunto completo del proceso en cascada de orden k* si $\sigma_{G,k} = |W|$.

Como conclusión de lo discutido en esta sección se obtiene el siguiente teorema.

Teorema E.2.3. Sea M una matriz que *no degenera*, de rango k. Dados $G_\Omega = (U, V, \Omega)$, $\mathcal{P}_\Omega(M)$ y $L_0 \subseteq U$ con $|L_0| = k$. Se tiene que $ICMC(G_\Omega, \mathcal{P}_\Omega(M), L_0)$ recupera la matriz M exactamente si L_0 es *un conjunto completo del proceso en cascada de orden k* en G_Ω.

E.3. ICMC sobre grafos aleatorios

La reducción del *problema de completado de matrices* al problema de encontrar *un conjunto completo del proceso en cascada de orden k* para un grafo, permite estudiar el algoritmo sobre diferentes clases de grafos. Los resultados enunciados en esta sección se refieren al modelo de grafos aleatorios CLV propuesto Chung, Vu y Lu. En primera instancia defínase el modelo CLV:

El modelo CLV genera grafos con una secuencia esperada de grados arbitraria, $p_1, \ldots, p_m, q_1, \ldots, q_n$.
Con $p_1 + \ldots + p_m = q_1 + \ldots + q_n = w$. En el modelo un grafo bipartito, $G = (U, V, E)$, queda definido por los vértices $U = \{u_1, \ldots, u_m\}$ y $V = \{v_1, \ldots, v_n\}$ y las aristas que son colocadas independientemente entre los vértices u_i, v_j, con probabilidad $\frac{p_i q_j}{w} \forall i \in [m], j \in [n]$. La *densidad* de la instancia CLV se define como el valor esperado de grados en los vértices $(p_1 + \ldots + p_m)/(mn) = w/(mn)$.

Si se eligen los pesos $p_1, \ldots, p_m, q_1, \ldots, q_n$ siguiendo una relación de *ley de potencial*, entonces el modelo CLV puede generar grafos con grados distribuidos también bajo la relación de *ley de potencial*. La relación de *ley potencial* entre dos números reales x, y se expresa de la siguiente manera: $y = Cx^p$. En la Figura E.1 se ve un ejemplo gráfico de relación *ley potencial*.

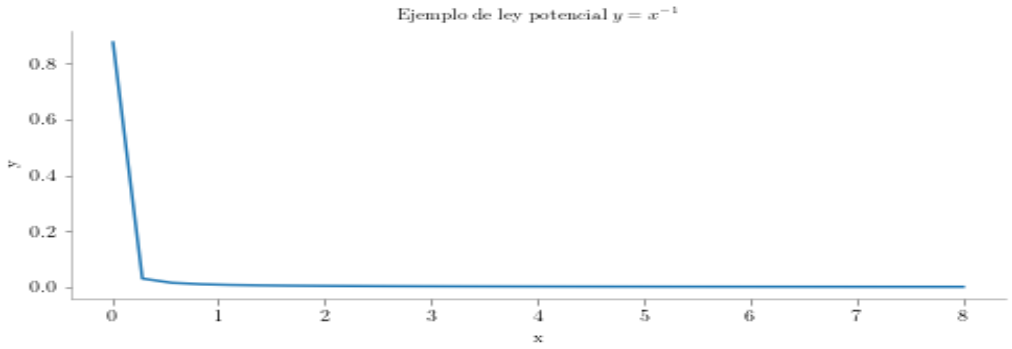


Figura E.1

En cambio si los pesos se eligen de la siguiente manera: $p_i = np$, $q_j = mp$, entonces la distribución de los grados de los vértices del grafo se asemeja a una distribución de *Poisson*, con los valores concetrados ligeramente alrededor de la media, ver Figura E.2. El modelo de grafo *Erdös–Rényi* sigue esta distribución, siendo por lo tanto un caso particular del modelo CLV.

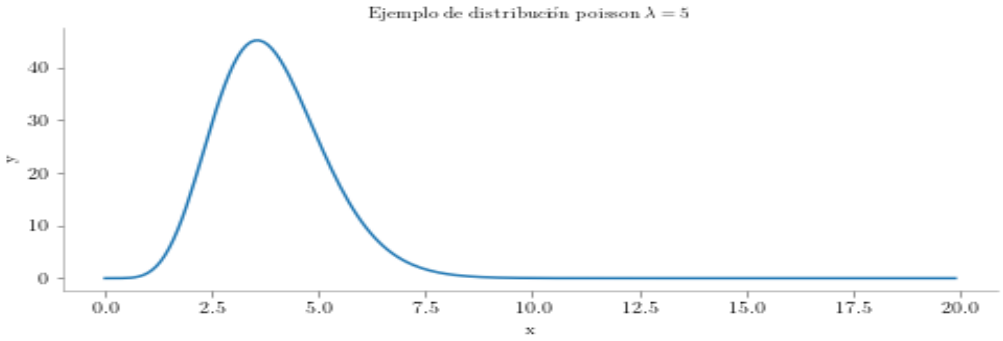


Figura E.2

Los siguientes teoremas resumen los resultados esperados del algoritmo ICMC para grafos que siguen una *ley potencial* para el grado de sus vértices.

Teorema E.3.1. Para todo $\gamma > 0$ existe una constante $c(\gamma)$ tal que ocurre lo siguiente. Considérese una instancia del modelo CLV dado por los pesos $p_1, \ldots, p_m$, $q_1, \ldots, q_n$ con densidad p y $min(p_i, q_j) \geq c(\gamma) \cdot k \cdot ln(n/p^k)$. Entonces para el grafo $G = (U, V, E)$ generado por el modelo, los k vértices de mayor grado en U forman un *conjunto completo del proceso en cascada de orden k* con probabilidad al menos $1 - n^{-\gamma}$.

Teorema E.3.2. Sea M una matriz que *no degenera* de rango k. Entonces si el grafo asociado G_Ω satisface las condiciones del Teorema E.3.1 entonces ICMC recupera la matriz M exactamente con alta probabilidad.

En el trabajo de Meka et al. [13], luego de la demostración de los teoremas, se remarca que debería ser posible mejorar los resultados para grafos con una distribución de grados siguiendo una *ley de potencial*.

E.4. Resultados

El modelo ICMC se compara con los siguientes algoritmos

- *Singular Value Thresholding* (SVT)
- *Alternating Least Squares* (ALS)
- *Spectral Matrix Completion* (SMC)

Los modelos de grafos con los que se comparan los algoritmos son:

- *Erdös–Rényi*
- *Chung-Lu-Vu*
- *Preferential Attachment*
- *Forest-fire*

De estos 4 modelos *Erdös–Rényi* sigue una distribución de Poisson, para los grados de los vértices. El resto presenta una distribución siguiendo la *ley de potencial*.

Detalles de implementación de ICMC: Es posible que los valores observados de la matriz tengan o no ruido, o que ICMC falle en completar la matriz.
Para el primer punto, considérese el paso 4 del Algoritmo 11. Sea L_j el conjunto de vértices en L que tienen una arista a v_j. Sea L_j^k cualquier subconjunto en L_j de tamaño k y sea $X[L_j^k, :]$ la submatriz de X conteniendo las filas correspondientes a los vértices L_j^k. Si la matriz subyacente es de rango bajo y no hay ruido en las entradas observadas, entonces para un nuevo vértice infectado v_j, la fila correspondiente de Y: y_j^T, puede ser computada resolviendo el sistema: $M[L_j^k, j] = X[L_j^k, :]y_j$. En caso de haber ruido en los datos y_j puede ser computado resolviendo el problema de mínimos cuadrados regularizado:

$$y = argmin_y ||M[L_j, j] - X[L_j :]y||_2^2 + \lambda ||y||_2^2$$

y analogamente se computa x_i^T:

$$x = argmin_x ||M[i, R_i]^T - Y[R_i, :]x||_2^2 + \lambda ||x||_2^2$$

Para el segundo punto: si ICMC falla en infectar todos los vértices, se tiene que o $L \subsetneq U$ o $R \subsetneq V$ y las filas de X e Y no serán computadas para los vértices $U \backslash L$, $V \backslash R$. Sea $X = [X_L, X_{\widetilde{L}}]$, en donde X_L es el conjunto de filas computadas de X y $X_{\widetilde{L}}$ denota el resto de las filas en X. Análogamente se define $Y = [Y_R, Y_{\widetilde{R}}]$. Para estimar $X_{\widetilde{L}}$ e $Y_{\widetilde{R}}$ se aplica la técnica de mínimos cuadrados arbitrarios (ALS) basado en la heurística [11] que resuelve el siguiente objetivo:

$$min_{X_{\widetilde{L}}, Y_{\widetilde{R}}} \left\| \mathcal{P}_\Omega \left(M - \begin{bmatrix} X_L \\ X_{\widetilde{L}} \end{bmatrix} \begin{bmatrix} Y_R & Y_{\widetilde{R}} \end{bmatrix} \right) \right\|_F^2 + \lambda ||X_{\widetilde{L}}||_F^2 + \lambda ||Y_{\widetilde{R}}||_F^2 \tag{E.4}$$

El experimento llevado a cabo en Meka et al. [13] genera las matrices $M \in R^{n \times n}$, variando el tamaño n, y los grafos asociados $G_\Omega(M)$ para cada modelo. Cada uno de los algoritmos se aplican sobre M y G_Ω para 20 corridas. De estas 20 corridas se calcula el RMSE promedio y el tiempo promedio que les llevó a los algoritmos completar la matriz.
La Figura E.3 muestra los resultados de cada uno de los algoritmos para los 4 modelos de grafos. Se observa que para los grafos *Erdös–Rényi*, ICMC obtuvo peores resultados, en comparación al resto de las técnicas. Sin embargo para los modelos que siguen una *ley de potencial*, ICMC obtuvo los mejores resultados. La Figura E.4 muestra los tiempos promedio, en segundos, de computación de las técnicas. Se observa que el tiempo de computación de ICMC es considerablemente menor al de SVT y SMC, y comparable con el de ALS.

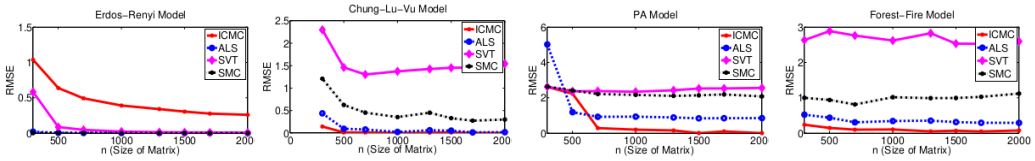


Figura E.3: Figuras extraídas de la sección *Experimental results* de Meka et al. [13]

(a) Erdős-Rényi Graphs

n/Method	SMC	SVT	ALS	ICMC
500	45.51	8.88	1.09	1.28
1000	93.85	17.07	2.39	3.30
1500	214.65	38.81	4.85	6.28
2000	343.76	59.88	7.20	9.89

(b) Chung-Lu-Vu Graphs

n/Method	SMC	SVT	ALS	ICMC
500	35.32	14.69	1.24	0.49
1000	144.19	17.55	2.24	2.02
1500	443.48	30.99	3.89	3.91
2000	836.99	46.69	5.67	5.50

(c) Preferential Attachment Graphs

n/Method	SMC	SVT	ALS	ICMC
500	15.05	14.40	3.97	1.94
1000	67.96	16.49	5.06	2.01
1500	178.35	24.48	9.83	3.65
2000	417.54	32.06	15.07	7.46

(d)Forest-fire Graphs

n/Method	SMC	SVT	ALS	ICMC
500	22.63	5.53	0.57	0.39
1000	85.26	11.32	1.75	1.23
1500	186.81	21.39	3.30	2.99
2000	350.98	27.37	4.84	5.06

Figura E.4: Tabla extraída de la sección *Experimental results* de Meka et al. [13]

E.5. Implementación en Python

Para este trabajo se realizó la implementación en Python 3 del Algoritmo 11. Para resolver el completado de la matriz en caso de que los valores observados

tengan ruido y ante la situación de que ICMC falle en completarla se aplican los *detalles de implementación* descritos en la Sección E.4.
Si los valores observados no tienen ruido: la fila $y_j^T \in Y$ se resuelve mediante la ecuación $M[L_j^k, j] = X[L_j^k, :]y_j$, en donde $L_j^k \subseteq L$ son los k vértices elegidos para infectar a v_j. Análogamente la fila $x_i^T \in X$ se resuelve mediante la ecuación $M[i, R_i^k] = Y[R_i^k, :]x_i$, en donde $R_i^k \subseteq R$ son los k vértices elegidos para infectar a x_i. La función empleada para resolver estas ecuaciones lineales es `numpy.linalg.solve`, que computa la solución x para una matriz de rango completo.

```
y_j = numpy.linalg.solve(M[L_j^k,j], X[L_j^k, :])
x_i = numpy.linalg.solve(M[i,R_i^k], Y[R_i^k, :])
```

Si los valores observados tienen ruido, se emplea un método iterativo para resolver el problema de mínimos cuadrados regularizado

$$min_x = \left\| \begin{bmatrix} b \\ \vec{0} \end{bmatrix} - \begin{bmatrix} A \\ \lambda \cdot I_{k \times k} \end{bmatrix} x \right\|$$

Siendo λ un parámetro del problema. La función `scipy.sparse.linalg.lsmr` es la utilizada para resolver el problema, siendo el parámetro de entrada *damp* el que especifica el valor del parámetro de amortiguamiento λ. En el caso de este proyecto se eligió el parámetro $damp = 0{,}01$.

```
y_j = scipy.sparse.linalg.lsmr(M[L_j^k,j], X[L_j^k, :],
        damp=0.01)
x_i = scipy.sparse.linalg.lsmr(M[i,R_i^k], Y[R_i^k, :],
        damp=0.01)
```

En el caso de que el algoritmo ICMC no complete todas las filas de X e Y. Se aplica la técnica de mínimos cuadrados arbitrarios (ALS) Ecuación E.4. El método heurístico utilizado se define en el trabajo [11].
La invocación del algoritmo ICMC está definida por

```
icmc(G, M, k, lambda_ = 0.1, nIter = 20, ruido=False,
     mayorGrado=False)
```

- En donde G es la matriz de adyacencia del grafo bi partito con particiones U y V. Siengo U el conjunto de nodos correspondientes a las filas de la matriz M, y V el conjunto de nodos correspondientes a las columnas de la matriz M.

- M, matriz incompleta
- k, ranking asumido para el completado de la matriz
- lambda_, valor del parámetro λ para el método heurístico ALS, Ecuación E.4.
- nIter, número de iteraciones para el método heurístico ALS, Ecuación E.4
- ruido, variable booleana. Si las mediciones tienen ruido entonces se resuelve el completado como un problema de minimización cuadrático regularizado, sino se resuelve como un problema de minimización cuadrática estándar
- mayorGrado, si `mayorRango == True` la infección se realiza eligiendo como nodos iniciales aquellos nodos de la partición U que tengan mayor grado en el grafo G, sino la infección se realiza eligiendo k nodos al azar de U.

Apéndice F

Explanation Table

Los datos que son sujeto de análisis, pueden tener atributos binarios, por ejemplo asociados a la satisfación de un predicado o al resultado de una hipótesis probada sobre los datos. Las aplicaciones incluyen: identificar grupos para filtrar datos (ej. `edad=adolescente` & `género=masculino`) o identificar factores de riesgo (ej. `peso=obeso` & `hábito=fumador`). Antes de performar análisis estadísticos complejos, los usuarios con frecuencia desean explorar y resumir los datos.

El concepto de *explanation tables* provee un resumen conciso de los datos en base a un atributo binario. Como ejemplo motivador véase la Tabla 2.5, en la Sección 2.4 del estado del arte, que resume la información del hábito alimenticio de un deportista antes del entrenamiento, con un atributo binario indicando si la meta del entrenamiento fue o no cumplida.

En las siguientes páginas se definirán los conceptos necesarios para entender y formular el problema, además de una descripción detallada de como cuantificar que tan informativa es una *explanation table*. Luego se analizará su complejidad. A continuación se describirá el algoritmo base y la versión *flashlight*. Los resultados del trabajo de ElGebaly et al. [6] serán presentados. Por último se brindarán detalles de la implementación hecha en Python para este proyecto.

F.1. Definiciones y conceptos

Considérese un instancia relacional D sobre un esquema $\mathcal{R}$ con atributos A_1,A_2, ..., A_d y un resultado binario v. Sea $v : D \rightarrow \{0,1\}$ una función que mapea cada tupla $t \in D$ a su resultado. Se asume que la semántica de v tiene alguna estructura

asociada respecto a la combinación de valores de A_1 a A_d.
Sea $\mathcal{PR}) = (dom(A_1) \cup \{*\} \times dom(A_2) \cup \{*\}) \times \ldots \times dom(A_d) \cup \{*\}$. Un *patrón* p, es una tupla de símbolos en donde cada símbolo i toma el valor del dominio A_i respectivo o es igual al comodín $*$.

Definición F.1.1. Un tupla t *coincide* con un *patrón* p, si para cada A_j en $\mathcal{R}$ pasa una de las siguientes condiciones

I $p[A_j] = *$

II $t[A_j] = p[A_j]$

La *coincidencia* se denota como $t \asymp p$.

Obsérvese que una tupla puede coincidir con muchos patrones, y un patrón pude coincidir con muchas tuplas. Por ejemplo el patrón (Sábado, $*$, $*$) de la Tabla 2.5 coincide con todas las tuplas de Tabla 2.4 con Día = Sábado.

Definición F.1.2. El *conjunto soporte* de un patrón p sobre una instancia relacional D, $S_D(p)$, es el conjunto de tuplas en D que *coinciden* con p, esto es

$$\{t \in D : t \asymp p\}$$

El *soporte* de p es

$$|S_D(p)|$$

Sea $\mathcal{P}(\mathcal{D}) = \{p \in \mathcal{R} : S_D(p) \neq \emptyset\}$. Y sea $f_{D,v}(p) : \mathcal{P}(D) \to [0, 1]$ la función que da la fracción de tuplas coincidentes con resultado 1

$$f_{D,v}(p) = \frac{1}{|S_D(p)|} \sum_{t \asymp p} v(t) \tag{F.1}$$

Definición F.1.3. Una *explanation table* (ET) es un resumen de resultados, dado como una colección T de patrones, teniendo cada patrón p en T asociada la fracción $f_{D,v}(p)$ de tuplas coincidentes con resultado $v = 1$ y el *soporte* $|S_D(p)|$.

A continuación se definirá como cuantificar la información de una ET T. La observación clave reside en que las fracciones de cada patrón $f_{D,v}(p)$ pueden ser usadas para *estimar* la distribución del resultado v, respecto a la combinación de valores A_j. Para computar la estimación es aplicado *el principio de máxima entropía*, que establece que la distribución de probabilidad que mejor representa

el estado actual del conocimiento, sujeto a restricciones conocidas, es aquella con mayor entropía.
Dada una ET T, la aproximación de máxima entropía u^* de v, sobre todo $t \in D$, es elegida de aquel u que maximice

$$\sum_{t \in D} -u(t)log(u(t)) \tag{F.2}$$

sujeto a:

$$\begin{cases} 0 \leq u(t) \leq 1 & \forall t \in D \\ \sum_{t \asymp p} u(t) = |S_D(p)| f_{D,v}(p) \end{cases}$$

Las restricciones indican que para cada tupla t en D, el valor estimado de su atributo binario, debe estar entre 0 y 1, es decir se permite una relajación de los valores binarios sobre el intervalo continuo $[0, 1]$. Y además las estimaciones deben ser consistentes con la fracción de resultados valiendo 1, $f_{D,v}(p)$, asociada a los patrones p.

Ejemplo F.1.4. Resolver $u^*(t)$ por cálculo directo
Sea T un ET para la Tabla 2.4. Supóngase que T consiste de solo dos patrones: $p_1 = (*, *, *)$ y $p_2 = (\text{Sábado}, *, *)$ que serán usados para estimar v.
La segunda restricción del problema requiere que

$$\sum_{t \asymp p_1} u(t) = 14 \cdot 0{,}5 = 7$$

y

$$\sum_{t \asymp p_2} u(t) = 5 \cdot 0 = 0$$

Esta última igualdad implica que todas las tuplas de la tabla con día igual a sábado deben tener $u^*(t) = 0$. Esto también significa que la suma de todas las tuplas que no caen en día sábado debe ser 7. La solución de máxima entropía pasa por asignar a todas las tuplas, con día distinto al sábado, la misma probabilidad, esto es $u^*(t) = \frac{7}{9}, t \in S_D(p_1) \backslash S_D(p_2)$.

No siempre ocurre que la resolución de la Ecuación F.2 sea tan directa como en el ejemplo anterior. En general para resolver el problema se puede recurrir a la técnica *iterative scaling*. Extendida esta técnica al problema de esta sección, un resultado clave del trabajo presentado en Berger et al. [1], establece que la distribución de u^*, tiene la siguiente forma

$$u^*(t) = \frac{e^{\sum_{p \in T : t \asymp p} \lambda(p)}}{1 + e^{\sum_{p \in T : t \asymp p} \lambda(p)}} \tag{F.3}$$

donde los $\lambda(p)$ son multiplicadores asociados a cada patrón. El Algoritmo 12 define los pasos, para encontrar los lambdas, de la Ecuación F.3, empleando la técnica *iterative scaling* Berger et al. [1]. Las entradas del algoritmo son: el conjunto de patrones P y las tuplas de atributos D.

Algoritmo 12 *IterativeScaling*(P, D)

1: //Inicializar
2: **for** $p \in P$ **do**
3: Inicializar $\lambda(p) \leftarrow 0$, multiplicador de p
4: **end for**
5: **while** valor actual u^*- valor anterior de $u^* > \epsilon$ **do**
6: //Calcular
7: **for** $p \in P$ **do**
8: Calcular $\Delta\lambda(p)$ usando $\sum_{t \asymp p} \frac{1}{|D|} e^{\Delta\lambda(p) f^{\#}(t)} u^*(t) = E[p]$
con $t \in D$, $f^{\#}$ es la cantidad de patrones con los que t coincide y $E[p]$, es el valor esperado del patrón p.
9: **end for**
10: //Actualizar
11: **for** $p \in P$ **do**
12: $\lambda(p) = \lambda(p) + \Delta\lambda(p)$
13: **end for**
14: **for** $t \in D$ **do**
15: $u^*(t) = \frac{e^{\sum_{p \in T: t \asymp p} \lambda(p)}}{1 + e^{\sum_{p \in T: t \asymp p} \lambda(p)}}$
16: **end for**
17: **end while**

En el paso 3, el algoritmo inicializa los multiplicadores de cada patrón igual a 0. Dentro de la iteración del `while`, en el paso 8 se calcula el incremento $\Delta\lambda(p)$ para cada multiplicador $\lambda(p)$, resolviendo la ecuación allí dada.
En el paso 9 se actualizan los multiplicadores y en el paso 10 se actualizan las estimaciones $u^*(t)$ de $v(t)$.

Ejemplo F.1.5. Resolver $u^*(t)$ aplicando iterative scaling Al igual que en el Ejemplo F.1.4, sea T un ET para la Tabla 2.4 y supóngase que T consiste de solo dos patrones: $p_1 = (*, *, *)$ y $p_2 =$ (Sábado, $*$, $*$).
En el paso 8 del Algoritmo 12 se define la forma de la ecuación para actualizar $\Delta\lambda(p_1)$ y $\Delta\lambda(p_2)$.
Ecuación para $\Delta\lambda(\mathbf{p}_1)$

- $f^{\#}(t) = \begin{cases} 1 & \text{tuplas } t \text{ cuyo día no es sábado} \\ 2 & \text{tuplas } t \text{ cuyo día sí es sábado} \end{cases}$
- $|D| = 14$
- $E[p_1] = \frac{7}{14} = 0{,}5$
- Hay 5 tuplas cuyo día es el sábado y 9 para las cuales no

La ecuación, separando las tuplas en aquellas cuyo día es sábado de las que no, queda:

$$\sum_{t \in \text{Sáb.}} \frac{1}{14} e^{2\Delta\lambda(p_1)} u^*(t) + \sum_{t \notin \text{Sáb.}} \frac{1}{14} e^{\Delta\lambda(p_1)} u^*(t) = 0{,}5$$

notar que para cada sumatoria, la estimación $u^*(t)$, definida por la Ecuación F.3, es la misma para cada tupla involucrada. Para todas las tuplas de la primera sumatoria se tiene que $u^*(t) = u^*(\lambda(p_1), \lambda(p_2))$ y para las tuplas de la segunda sumatoria $u^*(t) = u^*(\lambda(p_1))$, en donde los λ especifican cuales multiplicadores participan en la estimación. Por lo tanto la ecuación queda

$$\frac{5}{14} e^{2\Delta\lambda(p_1)} u^*(\lambda(p_1), \lambda(p_2)) + \frac{9}{14} e^{\Delta\lambda(p_1)} u^*(\lambda(p_1)) = 0{,}5$$

Con el cambio de variable $x = e^{\Delta\lambda(p_1)}$, la ecuación queda

$$\frac{5}{14} u^*(\lambda(p_1), \lambda(p_2)) x^2 + \frac{9}{14} u^*(\lambda(p_1)) x = 0{,}5$$

Luego de resolver este polinomio, obteniendo la raíz real positiva r, $\Delta\lambda(p_1)$ se despeja aplicando logaritmo a r

$$\Delta\lambda(p_1) = ln(r)$$

Ecuación para $\Delta\lambda(\mathbf{p}_2)$

- En este caso todas las tuplas que coinciden con p_2, también coinciden con p_1 por lo que $f^{\#}(t) = 2, \quad \forall t \asymp p_2$, y no hay necesidad de separar en varias sumatorias la ecuación dada en el paso 8 del Algoritmo 12
- $|D| = 14$
- $E[p_2] = \frac{0}{5} = 0$

- Hay 5 tuplas que coinciden con p_2

$$\sum_{t \in \text{Sáb.}} \frac{1}{14} e^{2\Delta\lambda(p_2)} u^*(t) = 0{,}01$$

En este caso, en vez del valor esperado 0, del lado derecho de la igualdad hay un valor de 0,01, pues sino al momento de resolver, el logaritmo no está definido para 0. En este caso se tiene la igualdad $u^*(t) = u^*(\lambda(p_1), \lambda(p_2)), \forall t \in$ Sáb.. Y la ecuación, con esta última consideración queda

$$\frac{5}{14} u^*(t) e^{2\Delta\lambda(p_2)} = 0{,}01$$

Con el cambio de variable $x = e^{\Delta\lambda(p_2)}$, la ecuación queda

$$\frac{5}{14} u^*(t) x^2 = 0{,}01$$

Resolviendo este polinomio, quedándose con la raíz real positiva r, $\Delta\lambda(p_2)$ se resuelve aplicando logaritmo a r

$$\Delta\lambda(p_2) = ln(r)$$

Luego de la convergencia, el Algoritmo 12 devuelve los valores $\lambda(p_1) = 1{,}1822$ y $\lambda(p_2) = -9$ para los multiplicadores. Para las tuplas con día igual a sábado, que coinciden con los dos patrones p_1 y p_2 el algoritmo devuelve

$$u^*(t) = \frac{e^{1,1822-9}}{1 + e^{1,1822-9}} = 0{,}00004$$

y para las tuplas con día distinto a sábado

$$u^*(t) = \frac{e^{1,1822}}{1 + e^{1,1822}} = 0{,}77$$

Para cuantificar que tan informativa es una *explanation table*, se computa la función *Kullback Leibler Divergence* ($D_{KL}(v||u^*)$), entre la distribución verdadera v de las diferentes combinaciones de valores de los A_j y la estimación de máxima entropía u^*.
Para una tupla dada t la función queda definida así

$$D_{KL}(v(t)||u^*(t)) = v(t) ln\left(\frac{v(t)}{u^*(t)}\right) + (1 - v(t)) ln\left(\frac{(1 - v(t))}{(1 - u^*(t))}\right) \tag{F.4}$$

Dado a que el atributo v es binario, $D_{KL}(v||u^*) = \sum_{t \in D} D_{KL}(v(t)||u^*(t))$, se puede escribir así

$$\sum_{t \in D_{|v(t)=1}} ln\left(\frac{1}{u^*(t)}\right) + \sum_{t \in D_{|v(t)=0}} ln\left(\frac{1}{1 - u^*(t)}\right)$$

F.2. Formulación del problema

Definición F.2.1. Problema del descubrimiento de la Explanation Table
Dada una instancia relacional D, con un atributo binario v y un umbral τ para la función *Kullback Leibler Divergence*.
El *problema del descubrimiento de la Explanation Table* es encontrar una *explanation table* $T \subseteq \mathcal{P}(D)$, del menor tamaño, con $u^*(t)$ determinado por la Ecuación F.3 de máxima entropía, tal que $D_{KL}(v\|u^*) \leq \tau$.

Proposición F.2.2. El problema del descubrimiento de la Explanation Table es NP-Hard
Prueba Se hará una reducción, en tiempo polinomial, del problema NP-Hard de *cubrimiento de vértices* en un grafo tripartito.
Sea G un grafo tripartito con partición de vértices (A, B, C), con m aristas. Los vértices de A, B, C se denotan a_i, b_j, c_k, respectivamente. Dada una arista $e \in G$, se puebla una instancia de relación como sigue:

- Si $e = \{a_i, b_j\}$ agregar la tupla (a_i, b_j, z_{ij})
- Si $e = \{a_i, c_k\}$ agregar la tupla (a_i, y_{ik}, c_k)
- Si $e = \{b_j, c_k\}$ agregar la tupla (x_{jk}, b_j, c_k)

donde cada z_{ij}, y_{ik}, x_{jk} es único. A estas tuplas se le asigna el atributo binario $v = 1$.
En adición

- A cada tupla (a_i, b_j, z_{ij}) agregar una tupla $(\alpha_i, \beta_j, z_{ij})$
- A cada tupla (a_i, y_{ik}, c_k) agregar una tupla $(\alpha_i, y_{ik}, \gamma_k)$
- A cada tupla (x_{jk}, b_j, c_k) agregar una tupla $(x_{jk}, \beta_j, \gamma_k)$

Las restricciones para estas tuplas son: $\alpha_i \neq a_{i'} \forall i, i'$, $\beta_j \neq b_{j'} \forall j, j'$, $\gamma_k \neq c_{k'} \forall k, k'$, $\alpha_i \neq \alpha_{i'} \forall i, i'$, $\beta_j \neq \beta_{j'} \forall j, j'$, $\gamma_k \neq \gamma_{k'} \forall k, k'$. A estas tuplas adicionales agréguesele como atributo binario $v = 0$.
D ahora contiene $2m$ tuplas, m de ellas con $v = 1$ y m de ellas con $v = 0$. Esta reducción claramente puede hacerse en tiempo polinomial.
Considérense las diferentes posibilidades para la *explanation table* más pequeña T, con $D_{KL}(v\|u^*) = 0$. Agréguese, en primera instancia, el patrón $(*, *, *)$, con $f_{D,v} = 0{,}5$, ya que exactamente la mitad de las tuplas tienen como valor binario

$v = 0$ y la otra mitad $v = 1$. Para lograr que $D_{KL}(v||u^*)$ valga 0, se necesitan encontrar patrones que cubran solamente aquellas tuplas con $v = 0$ o $v = 1$; estos patrones estarán asociados con valores de $f_{D,v}$, que permitirán hacer que los valores de los $u^*(t)$, correspondan exáctamente a los de $v(t)$.
Si queremos agregar patrones para las tuplas con $v = 0$, entonces se necesitarán m patrones, por la forma en que la instancia D fue construida: cada tupla con $v = 0$ requiere un patrón, distinto al de las demás, para ser identificada. Claramente esto no puede ser más eficiente que agregar patrones que cubran las tuplas con $v = 1$. Así que se elige encontrar patrones para identificar tuplas con $v = 1$.
Considérese el patrón p y asúmase que su primera componente es a_i. Ahora reemplácese p por $p' = (a_i, *, *)$. El nuevo patrón cubre, al menos, tantas tuplas como p, todas con $v = 1$. Es más, cada patrón de ese estilo, (o $(*, b_j, *)$ o $(*, *, c_k)$) corresponde a un vértice, en una forma intuitiva.
De esta manera, el tamaño del cubrimiento de vértices más pequeño es a lo sumo el número de patrones que se necesitan para cubrir todas las tuplas con $v = 1$.
Como resultado, una respuesta óptima puede ser obtenida eligiendo el patrón $(*, *, *)$ con $f_{D,v} = 0{,}5$ más los patrones asociados, de manera intuitiva, al cubrimiento de vértices (todos con $f_{D,v} = 1$). Se sigue que el tamaño mínimo de una *explanation table* es igual al tamaño mínimo de un cubrimiento de vértices, más 1 ■

Como caso ilustrativo de la demostración, considérese el siguiente ejemplo sobre un grafo tripartito.

Ejemplo F.2.3. Reducción del grafo tripartito $K_{1,1,3}$
La Figura F.1, muestra el grafo tripartito $K_{1,1,3}$, con $m = 7$ aristas.

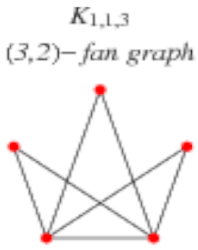


Figura F.1: Imagen extraída de:
http://mathworld.wolfram.com/CompleteTripartiteGraph.html

La partición de este grafo está dada por los tres vértices de arriba conformando el subconjunto A, el vértice izquierdo de la base perteneciendo al subconjunto B, y el vértice derecho de la base perteneciendo a C. Véase este esquema en la Figura F.2.

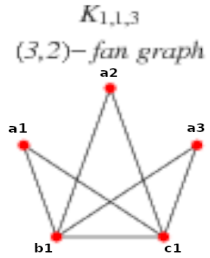


Figura F.2

La instancia D, según la construcción dada en la Propisición F.2.2, queda definida por las tuplas de la Tabla F.1.

$t_1 = (a_1, b_1, z_{11})$,	$v(t_1) = 1$
$t_2 = (a_1, y_{11}, c_1)$,	$v(t_2) = 1$
$t_3 = (a_2, b_1, z_{21})$,	$v(t_3) = 1$
$t_4 = (a_2, y_{21}, c_1)$,	$v(t_4) = 1$
$t_5 = (a_3, b_1, z_{31})$,	$v(t_5) = 1$
$t_6 = (a_3, y_{31}, c_1)$,	$v(t_6) = 1$
$t_7 = (x_{11}, b_1, c_1)$,	$v(t_7) = 1$
$t_8 = (\alpha_1, \beta_1, z_{11})$,	$v(t_8) = 0$
$t_9 = (\alpha'_1, y_{11}, \gamma_1)$,	$v(t_9) = 0$
$t_{10} = (\alpha_2, \beta_1\prime, z_{21})$,	$v(t_{10}) = 0$
$t_{11} = (\alpha'_2, y_{21}, \gamma'_1)$,	$v(t_{11}) = 0$
$t_{12} = (\alpha_3, \beta''_1, z_{31})$,	$v(t_{12}) = 0$
$t_{13} = (\alpha'_3, y_{31}, \gamma''_1)$,	$v(t_{13}) = 0$
$t_{14} = (x_{11}, \beta'''_1, \gamma'''_1)$,	$v(t_{14}) = 0$

Tabla F.1

Los patrones minimales elegidos para hacer $D_{KL}(v|u^*|) = 0$ serían, $p_1 = (*, *, *)$ y $p_2 = (*, b_1, *), p_3 = (*, *, c_1)$. La elección de los dos últimos patrones

se justifica por el hecho de que los vértices b_1 y c_1 son vértices de cubrimiento mínimos para el grafo $K_{1,1,3}$ y resuelven el problema del *cubrimiento de vértices*. Ahora se verá como los patrones, asociados a estos vértices, resuelven el problema del *descubrimiento de la explanation table*.
Las ecuaciones para la segunda restricción del problema de máxima entropía (Ec. F.2) quedan:

$$\sum_{t \asymp p_1} u(t) = u(t_1) + u(t_2) + u(t_3) + u(t_4) + u(t_5) + u(t_6) + u(t_7) + u(t_8) + u(t_9) + u(t_{10}) + u(t_{11}) + u(t_{12}) + u(t_{13}) + u(t_{14}) = 7 \quad \text{(F.5)}$$

$$\sum_{t \asymp p_2} u(t) = u(t_1) + u(t_3) + u(t_5) + u(t_7) = 4 \quad \text{(F.6)}$$

$$\sum_{t \asymp p_3} u(t) = u(t_2) + u(t_4) + u(t_6) + u(t_7) = 4 \quad \text{(F.7)}$$

De la Ecuación F.6 se tiene que los valores de $u(t_1) = u(t_3) = u(t_5) = u(t_7) = 1$ maximizan la Ecuación F.2 de máxima entropía. Con $u(t_7)$ valiendo 1, la Ecuación F.7 queda

$$\sum_{t \asymp p_3} u(t) = u(t_2) + u(t_4) + u(t_6) = 3 \quad \text{(F.8)}$$

Y los valores $u(t_2) = u(t_4) = u(t_6) = 1$ maximizan la Ecuación F.2 de máxima entropía. Con estos valores hayados la Ecuación F.5 queda:

$$\sum_{t \asymp p_1} u(t) = u(t_8) + u(t_9) + u(t_{10}) + u(t_{11}) + u(t_{12}) + u(t_{13}) + u(t_{14}) = 0 \quad \text{(F.9)}$$

y la única manera que esto sea así es con estos términos valiendo 0:

$$u(t_8) = u(t_9) = u(t_{10}) = u(t_{11}) = u(t_{12}) = u(t_{13}) = u(t_{14}) = 0$$

Observando los valores de las estimaciones se tiene que $u^*(t) = v(t)\ \forall t \in D$ y el valor $D_{KL}(v \| u^*) = 0$.
Con este ejemplo se ve como se puede realizar la reducción del problema de *cubrimiento de vértices*, en tiempo polinomial, a partir del problema del *descubrimiento de la explanation table*. Simplemente se construye la instancia D asociada al grafo k-partito, se hayan los patrones óptimos y se determinan los vértices asociados a cada patrón de la solución, salvo el patrón $(*, \ldots, *)$ que no tiene vértice alguno asociado.

F.3. Algoritmos

Dado que la computación de una *explanation table* es un problema NP-Hard, en esta sección se propondrá una solución heurística denominada *flashlight*, definiéndose en primera instancia un algoritmo base Alg. 13.

F.3.1. Baseline

El Algoritmo 13 presenta una linea base para construir *explanation tables* siguiendo una técnica *greedy*.

Algoritmo 13 $BaseLineET(D, \tau)$

1: //Inicializar
2: $T \leftarrow \{(*, \ldots, *)\}$
3: $u^* \leftarrow f_{D,v}(*, \ldots, *)$
4: **while** $D_{KL}(v||u^*) > \tau$ **do**
5: //Elección greedy
6: $p_{max} \leftarrow argmax_{p \in \mathcal{P}(D)} ganancia(p)$
7: $T \leftarrow T \cup \{p_{max}\}$
8: Actualizar u^* basado en T vía *iterative scaling*
9: **end while**

En el paso 2, se selecciona como primer patrón a $(*, \ldots, *)$, el cual coincide con todas las tuplas en D.
En el paso 3, se inicializa u^* con la estimación de cada tupla igual a la fracción de resultados $v = 1$ en D.
El bucle de los pasos 4 al 9, selecciona y agrega a T un patrón a la vez hasta que $D_{KL}(v||u^*)$ sea menor al umbral τ.
El paso 6 examina el espacio de todos los patrones posibles que ocurren en D y selecciona aquel con mayor *ganancia de información*. Una primera implementación poco *eficiente* de este paso sería: para cada posible nuevo patrón p, computar la nueva estimación $u^*(t)$ via *iterative scaling*, asumiendo que el patrón ha sido agregado a T, luego calcular $D_{KL}(v||u^*)$ y en el paso 7 agregar el patrón que mejor resultado haya dado en D_{KL}. Luego en el paso 8 actualizar $u^*(t)$, basado en el patrón que fue agregado en el paso 7.
Una implementación más eficiente pero inexacta del paso 6 es: *estimar* la *ganancia* del patrón p, solo basados en la diferencia entre las estimaciones nuevas y anteriores de u^*, para las tuplas que pertenecen al *conjunto soporte* de p. En lugar

de aplicar *iterative scaling* para cada patrón candidato y así obtener las nuevas estimaciones $u^*(t)$, se realiza lo siguiente: se computa, para las tuplas del *conjunto soporte* de p, la razón entre la suma de las estimaciones para estas tuplas y el *soporte* de p:

$$g_{D,u^*}(p) = \frac{1}{|S_D(p)|} \sum_{t \asymp p} u^*(t) \tag{F.10}$$

Luego para las tuplas del *conjunto soporte*, se computa la diferencia entre la fracción estimada de tuplas con resultado 1: $g_{D,u^*}(p)$ y la fracción real de tuplas con resultado 1: $f_{D,v}(p)$. A mayor diferencia entre estas, más beneficioso será agregar el patrón p, a modo de estimar las tuplas del *conjunto soporte* de p con mayor presición.
En este caso, la *ganancia* de p se define:

$$ganancia(p) = |S_D(p)| \cdot \left(f_{D,v}(p) ln \left(\frac{f_{D,v}(p)}{g_{D,u^*}(p)} \right) + (1 - f_{D,v}(p)) ln \left(\frac{1 - f_{D,v}(p)}{1 - g_{D,u^*}(p)} \right) \right) \tag{F.11}$$

F.3.2. Flashlight

Flashlight es una técnica que en vez de comparar la ganancia de todos los patrones posibles y elegir el de mayor beneficio, realiza la elección del siguiente patrón a incorporar a la *explanation table T* sobre un subconjunto de $\mathcal{P}(\mathcal{R})$. Para explicar el algoritmo, algunas definiciones son requeridas.

Definición F.3.1. Sea $p = lcap(t_1, t_2)$ el menor ancestro común de los patrones para las tuplas t_1, t_2. Es decir $p[A_j] = t_1[A_j]$ cuando $t_1[A_j] = t_2[A_j]$, y $p[A_j] = *$ en otro caso.

Ejemplo F.3.2.

$$lcap(\text{Vie.}, Amanecer, Banana), (\text{Vie.}, Noche, \text{Ens.Verde}) = (\text{Vie.}, *, *)$$

y

$$lcap(\text{Vie.}, Amanecer, Banana), (\text{Sáb.}, \text{Mediodía}, Tostadas) = (*, *, *)$$

Definición F.3.3. El *producto cruzado* entre los menores ancestros comunes, de dos instancias D_1, D_2: $LCA(D_1, D_2)$ es

$$\cup_{t_i \in D_1} \cup_{t_j \in D_2} lcap(t_i, t_j) \tag{F.12}$$

Ejemplo F.3.4. Sea $s \subset D$ una instancia con un subconjunto de 3 tuplas de D, especificadas en la Tabla F.2. La Tabla F.3 muestra el *producto cruzado* entre D y s.

Día	Tiempo	Comida	¿Meta cumplida?
Dom.	Mañana	Banana	Si
Jue.	Amanecer	Avena	Si
Sáb.	Amanecer	Banana	No

Tabla F.2: Muestra de tres tuplas de la instancia dada en la Tabla 2.4. Tabla extraída de ElGebaly et al. [6].

Relación \ Muestra	(Dom., Mañana, Banana)	(Jue., Amanecer, Avena)	(Sáb,Amanecer,Banana)
(Vie., *Amanecer*, *Banana*)	$(*, *,$ *Banana*$)$	$(*,$ *Amanecer*$, *)$	$(*,$ *Amanecer*, *Banana*$)$
(Vie., *Noche*, Ens.Verde)	$(*, *, *)$	$(*, *, *)$	$(*, *, *)$
(Dom., *Atardecer*, *Avena*)	(Dom., $*, *$)	$(*, *,$ *Avena*$)$	$(*, *, *)$
(Dom., Mañana, *Banana*)	(Dom., Mañana, *Banana*)	$(*, *, *)$	$(*, *,$ *Banana*$)$
(Lun., *Tarde*, *Avena*)	$(*, *, *)$	$(*, *,$ *Avena*$)$	$(*, *, *)$
(Lun., Mediodía, *Banana*)	$(*, *,$ *Banana*$)$	$(*, *, *)$	$(*, *,$ *Banana*$)$
(Mar., Mañana, Ens.Verde)	$(*,$ Mañana$, *)$	$(*, *, *)$	$(*, *, *)$
(Mié, *Noche*, *Hamburguesas*)	$(*, *, *)$	$(*, *, *)$	$(*, *, *)$
(Jue., *Amanecer*, *Avena*)	$(*, *, *)$	(Jue., *Amanecer*, *Avena*)	$(*,$ *Amanecer*$, *)$
(Sáb., *Tarde*, *Nueces*)	$(*, *, *)$	$(*, *, *)$	(Sáb., $*, *$)
(Sáb., *Amanecer*, *Banana*)	$(*, *,$ *Banana*$)$	$(*,$ *Amanecer*$, *)$	(Sáb., *Amanecer*, *Banana*)
(Sáb., *Amanecer*, *Avena*)	$(*, *, *)$	$(*,$ *Amanecer*, *Avena*$)$	(Sáb., *Amanecer*, $*$)
(Sáb., *Atardecer*, *Arroz*)	$(*, *, *)$	$(*, *, *)$	(Sáb., $*, *$)
(Sáb., Mediodía, *Tostadas*)	$(*, *, *)$	$(*, *, *)$	(Sáb., $*, *$)

Tabla F.3: Cada fila corresponde a una de las 14 tuplas de la Tabla 2.4 y cada columna a una de las tres tuplas de la muestra (Tabla F.2). La computación del LCA asume la preservación de los ancestros duplicados, por ejemplo en la tabla el patrón $(*, *,$ *Banana*$)$ aparece 5 veces. Tabla extraída de ElGebaly et al. [6].

La idea de esta técnica es usar $LCA(D, s)$ para computar la ganancia de p: *ganancia*(p). La descripción de *flashlight* se hará a través de un ejemplo.
En primer lugar sea D la Tabla 2.4 y la muestra s la Tabla F.2. Asúmase que el patrón $(*, *, *)$ ha sido el primero en agregarse a T. Lo que significa que la estimación inicial para cada tupla t en D es $u^*(t) = 0{,}5$. *Flashlight* implementa el paso 6 del Algoritmo 13, computando la *ganancia*, dada por la Ecuación F.11, sobre todas las tuplas en $\mathcal{P}(s)$ y eligiendo aquella con la mayor ganancia.
El primer paso es computar el producto cruzado $LCA(D, s)$, seguido de una agregación por grupo computando para cada patrón único lo siguiente:

- $contar(D, *)$, para cada patrón p, se cuenta la cantidad de apariciones del patrón en $LCA(D, s)$
- $SUM(D, v) = \sum_{t \preceq p} nv(t)$, en donde $v(t)$ es el resultado binario de la tupla t que propicia la aparición de p en $LCA(D, s)$, y n cuenta la cantidad de veces que p aparece en $LCA(D, s)$ gracias a la tupla t.
- $SUM(D, u^*) = \sum_{t \preceq p} nu^*(t)$,en donde $u^*(t)$ es la estimación actual de la tupla t que propicia la aparición de p en $LCA(D, s)$, y n cuenta la cantidad de veces que p aparece en $LCA(D, s)$ gracias a la tupla t.

Ejemplo F.3.5. La Tabla F.4 muestra las agregaciones para los 13 patrones distintos de $LCA(D, s)$. Por ejemplo el patrón $(*, *, Banana)$ en $LCA(D, s)$ es derivado de las tuplas (Vie., *Amanecer*, *Banana*) que tiene $v = 1$, (Dom.,Mañana,Banana) con $v = 1$, dos veces de (Lun.,Mediodía,Banana) con $v = 1$, y finalmente de (Sáb., *Amanecer*, *Banana*) que tiene $v = 0$. La suma de los v es 4 y de los u^* es 2,5, incluida la repetición de la tupla (Lun., Mediodía, *Banana*). Recordar que en este primer paso, solo con el patrón $(*, *, *)$ agregado, todas las estimaciones $u^*(t)$ valen 0,5.

El segundo paso es, para cada patrón producido como resultado del paso previo, generar los *patrones ancestros*, definidos como p más todos los otros patrones que son iguales a p, con la excepción de que uno o más de un atributo en p, que no sean el símbolo $*$, son reemplazados por el $*$. La columna de más a la derecha en la Tabla F.4 lista a los ancestros de cada patrón.
A continuación se toman todos los ancestros y se computa otra agregación sobre cada uno de los ancestros generados, sin repetir. Cada ancestro tiene las funciones $contar(D, *)$,$SUM(D, v)$,$SUM(D, u^*)$ asociadas. Para cada ancestro p se describen esos cálculos:

- $contar(D, *)$ es la suma de todas las apariciones de los patrones en $LCA(D, s)$ de los que p es ancestro
- $SUM(D, v) = \sum_{p_h = ancestro(p)} SUM_{p_h}(D, v)$, suma de los resultados binarios v de todos los patrones únicos $p_h \in LCA(D, s)$ que son descendientes de p.
- $SUM(D, u^*) = \sum_{p_h = ancestro(p)} SUM_{p_h}(D, u^*)$, suma de las estimaciones u^* de todos los patrones únicos $p_h \in LCA(D, s)$ que son descendientes de p.

Ejemplo F.3.6. En la Tabla F.5 se listan todos los ancestros, sin repetir. Las columnas $2, 3$ y 4 describen el resultado del cálulo de las tres funciones antes descritas: $contar(D, *)$,$SUM(D, v)$ y $SUM(D, u^*)$. Por ejemplo para $(*, *, Banana)$ sus ancestros son: él mismo ($contar = 5$,$SUM(D, v) = 4$ y $SUM(D, u^*) = 2{,}5$), $(*, Amanecer, Banana)$ ($contar = 1$,$SUM(D, v) = 0$ y $SUM(D, u^*) = 0{,}5$), (Sáb., $Amanecer$, $Banana$) ($contar = 1$,$SUM(D, v) = 0$ y $SUM(D, u^*) = 0{,}5$) y (Dom., Mañana, $Banana$) ($contar = 1$,$SUM(D, v) = 1$ y $SUM(D, u^*) = 0{,}5$).
Lo que deja para el patrón $(*, *, Banana)$ los siguientes valores: $contar = 8$, $SUM(D, v) = 5$ y $SUM(D, u^*) = 4$.

Notar que en este punto, después de generar los *patrones ancestros*, del segundo paso, el conjunto resultante de 20 patrones es exactamente el conjunto

$$\mathcal{P}(s) = \{p \in \mathcal{R} : S_s(p) \neq \emptyset\}$$

El tercer paso es corregir el conteo y las sumas agregadas, para que correspondan con $|S_D(p)|$, $f_{D,v}(p)$ y $g_{D,u^*}(p)$ respectivamente. Para esto se dividen las columnas $2, 3$ y 4 de la Tabla F.5, por la cantidad de veces que el patrón coincide con las tuplas de la muestra. La cantidad de coincidencias de los patrones con la muestra s se visualiza en la columna 5 de la Tabla F.5, en donde por ejemplo $(*, *, Banana)$ coincide con dos tuplas de s. Esto da la suma corregida: $contar = \frac{8}{2} = 4$,$SUM(D, v) = \frac{6}{2} = 3$ y $SUM(D, u^*) = \frac{4}{2} = 2$ para $(*, *, Banana)$. De estos resultados se computa: $f_{D,v}(*, *, Banana) = \frac{SUM(v)}{count} = \frac{3}{4}$, $g_{D,u^*}(*, *, Banana) = \frac{SUM(u^*)}{count} = \frac{2}{4}$. Las sumas corregidas de todos los patrones se encuentran en las tres columnas de más a la derecha de la tabla. El paso final es elegir el patrón con de $\mathcal{P}(s)$ con mayor ganancia para la Ecuación F.11, empleando la información de las últimas tres columnas de la tabla, para cada patrón.

Ejemplo F.3.7. El patrón (Sáb., $*$, $*$), es el que mayor ganancia tiene: $5 \cdot (0 + ln(\frac{1}{0{,}5})) = 5 \cdot ln(2)$. Y por lo tanto es agregado a la *explanation table* T.

Patrón	cuenta	SUM(v)	SUM(u^*)		Todos los ancestros
$(*,*,*)$	21	9	10,5	→	$(*,*,*)$
$(*,*,$ *Banana*$)$	5	4	2,5	→	$(*,*,$ *Banana*$)$, $(*,*,*)$
$(*,$ *Amanecer*$,*)$	3	2	1,5	→	$(*,$ *Amanecer*$,*)$, $(*,*,*)$
(Sáb., $*,*$)	3	0	1,5	→	(Sáb., $*,*$), $(*,*,*)$
$(*,*,$ *Avena*$)$	2	2	1	→	$(*,*,$ *Avena*$)$, $(*,*,*)$
$(*,$ *Amanecer*, *Banana*$)$	1	0	0,5	→	$(*,$ *Amanecer*, *Banana*$)$, $(*,$ *Amanecer*$,*)$, $(*,*,$ *Banana*$)$, $(*,*,*)$
$(*,$ *Amanecer*, *Avena*$)$	1	0	0,5	→	$(*,$ *Amanecer*, *Avena*$)$, $(*,$ *Amanecer*$,*)$, $(*,*,$ *Avena*$)$, $(*,*,*)$
(*, Mañana, *)	1	0	0,5	→	(*, Mañana, *), $(*,*,*)$
(Sáb., *Amanecer*, *)	1	0	0,5	→	(Sáb., *Amanecer*, *), (Sáb., *, *), $(*,$ *Amanecer*$,*)$, $(*,*,*)$
(Sáb., *Amanecer*, *Banana*)	1	0	0,5	→	(Sáb., *Amanecer*, *Banana*), . . . , (Sáb., *, *), $(*,$ *Amanecer*$,*)$, $(*,*,*)$
(Dom., *, *)	1	1	0,5	→	(Dom., *, *), $(*,*,*)$
(Dom., Mañana, *Banana*)	1	1	0,5	→	(Dom., Mañana, *Banana*), . . . , (*, Mañana, *), $(*,*,$ *Banana*$)$, $(*,*,*)$
(Jue., *Amanecer*, *Avena*)	1	1	0,5	→	(Jue., *Amanecer*, *Avena*), . . . , (*, *Amanecer*, *), $(*,*,$ *Avena*$)$, $(*,*,*)$

Tabla F.4: Tabla extraída de ElGebaly et al. [6].

Patrón	cuenta	SUM(v)	SUM(u^*)	Frec. en Muestra	$S_D(p)$	$f_{D,v}(p)$	$g_{D,u^*}(p)$
$(*,*,*)$	42	21	21	3	14	0,5	0,5
$(*,*,$ *Banana*$)$	8	6	4	2	4	0,75	0,5
$(*,*,$ *Avena*$)$	4	3	2	1	4	0,75	0,5
$(*,$ *Amanecer*$,*)$	8	4	4	2	4	0,5	0,5
(*, Mañana, *)	2	2	1	1	2	0,5	0,5
(Sáb., *, *)	5	0	2,5	1	5	0	0,5
(Dom., *, *)	2	2	1	1	2	1	0,5
(Jue., *, *)	1	1	0,5	1	1	1	0,5
$(*,$ *Amanecer*, *Banana*$)$	2	1	1	1	2	0,5	0,5
$(*,$ *Amanecer*, *Avena*$)$	2	1	1	1	2	0,5	0,5
(*, Mañana, *Banana*)	1	1	0,5	1	1	1	0,5
(Sáb., *, *Banana*)	1	0	0,5	1	1	0	0,5
(Dom, *, *Banana*)	1	1	0,5	1	1	1	0,5
(Jue., *, *Avena*)	1	1	0,5	1	1	1	0,5
(Sáb., *Amanecer*, *)	2	0	1	1	2	0	0,5
(Dom., Mañana, *)	1	1	0,5	1	1	1	0,5
(Jue., *Amanecer*, *)	1	1	0,5	1	1	1	0,5
(Dom., Mañana, *Banana*)	1	1	0,5	1	1	1	0,5
(Sáb., *Amanecer*, *Banana*)	1	0	0,5	1	1	0	0,5
(Jue., *Amanecer*, *Avena*)	1	1	0,5	1	1	1	0,5

Tabla F.5: Tabla extraída de ElGebaly et al. [6].

Proposición F.3.8. La complejidad del algoritmo *flashlight* es $O(|s||D| + |\mathcal{P}(s)||s|)$.
Prueba: El cómputo de la ganancia, paso 6 del Algoritmo 13, requiere $|\mathcal{P}(s)||D|$ operaciones. El primer paso de *flashlight* requiere $|s||D|$ operaciones para computar $LCA(D, s)$. El segundo paso requiere $|\mathcal{P}(s)|$ operaciones para generar los ancestros de los patrones. El tercer paso requiere $|\mathcal{P}(s)||s|$ operaciones para para computar las funciones agregadas de los $\mathcal{P}(s)$ patrones respecto de las tuplas de la muestra s.

F.4. Resultados

Los experimentos descritos en el paper [6], se basaron en tres datasets:

- `Upgrade`, 5795 registros de tickets de *United Airline* extraídos de la página https://www.diditclear.com/.
- `Adult`, datos del censo norteamericano, conteniendo datos demográficos tales como ocupación, eduación y género y atributos binarios denotando si los ingresos de una persona dada superan los $U\$S\,50{,}000$.
- `Income`, datos del censo norteamericano, conteniendo datos demográficos como cantidad de hijos y estado civil y atributos binarios denotando si los ingresos de una persona dada superan los $U\$S\,100{,}000$.

La medición empleada para cuantificar la información del contenido de los resumenes es el *promedio de la ganancia* dada por la Ecuación F.11.

Comparación de la explanation table con otros enfoques

La comparación se hace contra tres enfoques:

- Árboles de desición
- Operador SURPRISE para exploración de datacubos
- Pattern Tableaux, técnica que encuentra la menor cantidad de patrones que cubren la mayoría de las tuplas con resultado $v = 1$

Para los árboles de desición se considera una técnica que permite a un nodo tener varios hijos (denominada DT) y otra que restringe la división del nodo en solo dos hijos (denominada DTN). El operador SURPRISE requiere un parámetro ϵ, que representa un umbral de exactitud. Se definen dos técnicas: la primera $SURPRISE_{cmp}$, con un valor ϵ tal que el tiempo de ejecución es comparable al de *flashligth*, la segunda $SURPRISE_{bst}$ que emplea el ϵ que mayor *ganancia* dé. La técnica *Pattern Tableaux*, requiere un parámetro $\hat{c}$ que denota la fracción mínima de resultados $v = 1$ por patrón, $Cover_{bst}$ experimenta un rango de valores para $\hat{c}$ y reporta el resultado de aquel que mayor *ganancia* generó.

El primer punto a comparar es cuanta *ganancia* de información, cada técnica obtiene, en base al número de patrones. En la Figura F.3 se muestan las gráficas, con los resultados, para el dataset *Upgrade* (izquierda) y *Adult* (derecha). *Flashlight* supera a todas las técnicas, excepto para una cantidad pequeña de patrones (menos

de 8) en el dataset *Adult*, en cuso caso SURPRISE tiene una ganancia levemente superior (véase la Sección 5.2 en ElGebaly et al. [6], para una explicación de estos resultados).
El segundo punto de interés son los tiempos de ejecución de cada técnica. La Figura **??** muestra una tabla comparativa de los tiempos, medidos en segundos. La conclusión de estos resultados es que *flashlight* ofrece la mejor ganancia de información, tiempo de ejecución y escalabilidad respecto a los algoritmos probados.

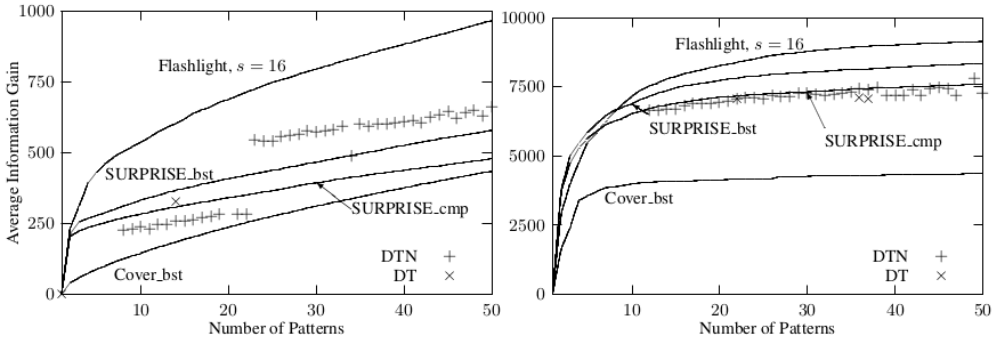


Figura F.3: Imagen extraída de ElGebaly et al. [6].

	Upgrade	Adult	Income
Flashlight	47 s	252 s	0.6 h
DT	1021 s	2599 s	56 h
DTN	6120 s	8348 s	N/A
SURPRISE_cmp	61 s ($\epsilon = 0.2$)	243 s ($\epsilon = 0.1$)	N/A
SURPRISE_bst	88 s ($\epsilon = 0.08$)	875 s ($\epsilon = 0.02$)	N/A
cover_bst	240 s ($\hat{c} = 0.99$)	2051 s ($\hat{c} = 0.6$)	N/A

Figura F.4: Imagen extraída de ElGebaly et al. [6].

Comparación de flashlight con otras técnicas que resuelven explanation table
En primera instancia se compara el tiempo de ejecución de *flashlight* contra las siguientes técnicas:

- *Baseline*, técnica que al momento de agregar un nuevo patrón evalúa cual de todos los patrones genera la mayor ganancia de información.

- *SampleCube gain*$_D$, técnica que elije el nuevo patrón a agregar de $\mathcal{P}(s)$, el conjunto de todos los patrones posibles de un subconjunto de tuplas s. La *ganancia*, para elegir el siguiente mejor patrón, es computada sobre todas las tuplas de D.

La Figura F.5 muestra los resultados del tiempo de ejecución de cada técnica en función del tamaño de la muestra s. Los resultados son dramáticos, a medida que el tamaño de la muestra aumenta, *flashlight* es varios órdenes de magnitud más rápido.

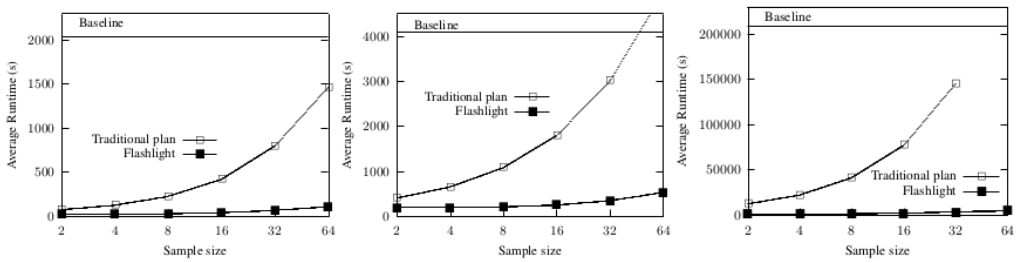


Figura F.5: La gráfica de la izquierda representa los resultados para el dataset *Upgrade*, la del medio para *Adult* y la de la derecha para *Income*. Imagen extraída de ElGebaly et al. [6].

Por último se comparan las técnicas respecto a la mejor *ganancia* posible que puedan obtener dados plazos límites en los tiempos de ejecución. Para esto cada algoritmo se ejecutó para diferentes tamaños de la muestra s y de la tabla T. La Figura F.6 muestra los resultados. Cada punto en la gráfica representa la mejor combinación de $|T|$ y $|s|$ para la mejor *ganancia* de información. Además de forma separada se ejecutó una versión de *flashlight* con tamaño de muestra constante $s = 16$, para verificar que *flashlight* performa bien, incluso con tamaños de muestra pequeños. A su vez, para poner en contexto, se graficaron los resultados para la técnica *baseline*. Aunque sus tiempos de ejecución fueron muy altos y quedaron fuera de la escala. Las dos observaciones sobre *flashlight* son las siguientes: a medida que aumentan los tiempos de ejecución, *flashlight* se aproxima a los resultados de *baseline* (por ejemplo a medida que $|T|$ aumenta), pero el tiempo de ejecución de *flashlight* es mucho menor que el de *baseline*. La segunda observación es que *flashlight* ofrece la mejor relación entre tiempo de ejecución y ganancia en la información.

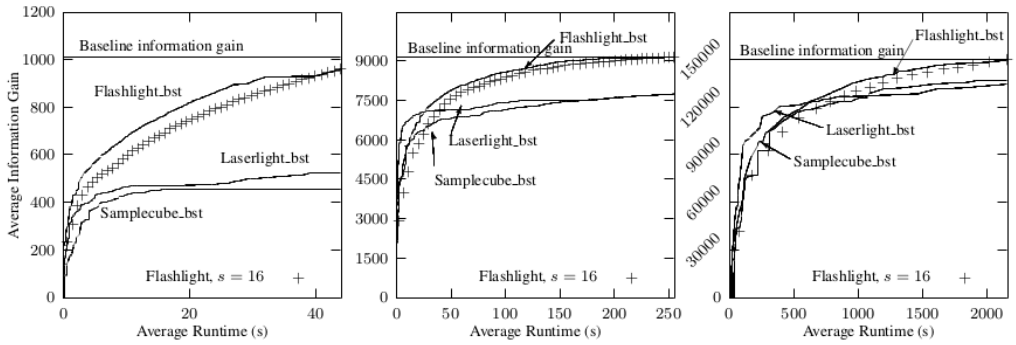


Figura F.6: Imagen extraída de ElGebaly et al. [6].

F.5. Implementación de flashlight en Python

Sea $D_{m\times n}$ una instancia relacional, de m tuplas, con $A_1, \ldots, A_n$ atributos; sea v_m, la columna de valores binarios; *cantMuestra*, el tamaño de la muestra, con un valor por defecto igual a 3 y $|T|$ el tamaño de la *explanation table*. La función `explanationTable`, toma a $D_{m\times n}$, v_m, $|T|$ y *cantMuestra*, como entrada; devolviendo: $T_{|T|\times n}$, la *explanation table*; $rmse(v_m, u_m^*)$, el error medio cuadrático entre los resultados binarios de cada tupla (v_m) y los resultados estimados por el algoritmo (u_m^*); u_m^*, las estimaciones, de v_m, hechas por el algoritmo:

$$T_{|T|\times n}, rmse(v_m, u_m^*), u_m^* = explanationTable(D_{m\times n}, v_m, |T|, cantMuestra = 3)$$

La implementacion en Python sigue los pasos detallados en el Algoritmo 13. Con la salvedad de que en vez de terminar el bucle cuando la estimación u^* y el valor real binario v, tengan una divergencia *Kullback - Leibler* menor a un umbral τ, el bucle en este caso termina luego de la $|T|$-ésima iteración. Es decir cuando se haya agregado el patrón número $|T|$ a la *explanation table*.
En el bloque de inicialización se agrega a la *explanation table* el patrón $p_1 = (*, *, \ldots, *)$ y se inicializa u_m^* con las m entradas valiendo $f_{D,v}(p_1)$, definida en la Ecuación F.1 como la fracción de tuplas con valor 1 en el conjunto *soporte* de p_1.
Dentro del bucle la primera tarea es elegir el próximo patrón que dé mayor ganancia a la *explanation table*. El conjunto de patrones candidatos se obtiene eligiendo primeramente *cantMuestra* tuplas al azar de $D_{m\times n}$ y, a partir de estas tuplas, computando los patrones candidatos de la manera en que se especifican para el algoritmo *flashlight* en la Sección F.3.2. La Tabla F.5 muestra el resultado final en

donde cada patrón p candidato tiene asociado: su soporte $|S_{D,v}(p)|$, la fracción de tuplas con valor 1 asociada $f_{D,v}(p)$ y la fracción de tuplas con valor 1 estimadas $g_{D,u^*}(p)$. Con estos valores asociados a cada patrón se elige aquel que dé mayor ganancia de acuerdo a la Ecuación F.11.
Una vez elegido p_{max}, el patrón que maximiza la ganancia, este es agregado a la *explanation table* T.
El paso final de la iteración es actualizar las estimaciones u^* aplicando el Algoritmo *iterative scaling* 12.

Bibliografía

[ALB96] V. J.D. P. Adam L. Berger Stephen A. Della Pietra. «A Maximum Entropy Approach to Natural Language Processing». En: (1996) (vid. págs. 207, 208).

[BJ70] G. Box y G. Jenkins. «Time Series Analysis: Forecasting and Control.» En: (1970) (vid. pág. 171).

[CDMS08] P. R. Christopher D. Manning y H. Schütze. *An Introduction to Information Retrieval*. Cambridge University Press, 2008 (vid. págs. 22, 23, 157, 159, 161).

[DB79] D. L. Davis y D. W. Bouldin. «A Cluster Separation Measure». En: (mayo de 1979). ISSN: 1084-6654 (vid. pág. 159).

[EESF14] Energeia, F. Economics e I. for Sustainable Futures. «Smart Grid, Smart City: Shaping Australia's Energy Future Executive Report». En: (2014) (vid. pág. 74).

[ElG+14] K. ElGebaly y col. «Interpretable and Informative Explanations of Outcomes». En: (2014) (vid. págs. 1, 12, 32-34, 205, 217, 220-224).

[Ensa] *Random Forest Regression: Along with its implementation in Python*. URL: https://towardsdatascience.com/random-forest-and-its-implementation-71824ced454f (vid. pág. 194).

[Ensb] *Understanding Random Forest*. URL: https://towardsdatascience.com/understanding-random-forest-58381e0602d2 (vid. págs. 193-195).

[GC12] G. Gursun y M. Crovella. «On Traffic Matrix Completion in the Internet». En: (2012) (vid. pág. 29).

[HA18] R. J. Hyndman y G. Athanasopoulos. *Forecasting: Principles and Practice, 2nd Edition*. Monash University, Australia, 2018 (vid. págs. 13, 18).

[HL15] M. L.Y. P. Haoming Li Bangzheng He. «Matrix Completion via Alternating Least Square(ALS)». En: (2015) (vid. págs. 31, 201, 203).

[LL17] P. Laurinec y M. Lucká. «New Clustering-based Forecasting Method for Disaggregated End-consumer Electricity Load Using Smart Grid Data». En: (2017) (vid. págs. 1, 8-10, 12, 21, 26, 85, 86, 92, 95, 98, 103, 144, 163).

[RMD09] P. J. Raghu Meka e I. S. Dhillon. «Matrix Completion from Power-Law Distributed Samples». En: (2009) (vid. págs. 9, 10, 12, 28, 29, 31, 196, 200, 202).

[SS16] R. H. Shumway y D. S. Stoffer. *Time Series Analysis and its Applications, 4th Ed.* Springer, 2016 (vid. págs. 13, 15, 169, 174, 176, 182, 184, 185, 189).