UNIVERSIDAD DE LA REPUBLICA
URUGUAY

FACULTAD DE INGENIERÍA
UDELAR

Algoritmos evolutivos para el aprendizaje de modelos en la industria de procesos

Jimena Ferreira Quagliata

Programa de Posgrado en Ingeniería Química
Facultad de Ingeniería
Universidad de la República

Montevideo – Uruguay
Noviembre de 2022


Algoritmos evolutivos para el aprendizaje de modelos en la industria de procesos

Jimena Ferreira Quagliata

Tesis de Doctorado presentada al Programa de Posgrado en Ingeniería Química, Facultad de Ingeniería de la Universidad de la República, como parte de los requisitos necesarios para la obtención del título de Doctor en Ingeniería Química.

Directores:
Dra. Ana Inés Torres
Dr. Martín Pedemonte

Director académico:
Dra. Soledad Gutiérrez

Montevideo – Uruguay
Noviembre de 2022

Ferreira Quagliata, Jimena

Algoritmos evolutivos para el aprendizaje de modelos en la industria de procesos / Jimena Ferreira Quagliata. - Montevideo: Universidad de la República, Facultad de Ingeniería, 2022.

XXIV, 115 p.: il.; 29, 7cm.

Directores:

Ana Inés Torres

Martín Pedemonte

Director académico:

Soledad Gutiérrez

Tesis de Doctorado – Universidad de la República, Programa en Ingeniería Química, 2022.

Referencias bibliográficas: p. 99 – 104.

1. Aprendizaje automático, 2. Modelos subrogados, 3. Algoritmos evolutivos, 4. Soft-sensors, 5. Ingeniería de sistemas de procesos. I. Torres, Ana Inés, Pedemonte, Martín, . II. Universidad de la República, Programa de Posgrado en Ingeniería Química. III. Título.

INTEGRANTES DEL TRIBUNAL DE DEFENSA DE TESIS

Dr. Prof. Diego Caffaro

Dra. Prof. Libertad Tansini

Dra. Prof. Elena Castelló

Montevideo – Uruguay
Noviembre de 2022

Agradecimientos

Quisiera agradecer a mis tutores Ana Inés y Martín por haber estado presente, y todo lo aprendido e intercambiado en estos años. A Pablo y Soledad, por estar presente. A mis amigos y compañeros, en particular a Mariana, Claudia, Jonathan, Viviana, Ernesto y Manuel, gracias por estar y bancarme en mis reclamos. Y por último, a mi familia por estar siempre presente.

RESUMEN

En esta tesis se presenta el uso de técnicas de aprendizaje automático para obtener modelos subrogados en el contexto de procesos químicos. Los modelos subrogados son modelos simples que aproximan un modelo matemático más complejo con precisión. Suelen ser utilizados para la resolución de problemas que involucran un alto costo computacional, como optimización y control de procesos.

En particular, en esta tesis se utilizan dos técnicas provenientes de la familia de los algoritmos evolutivos, la Programación Genética y la Programación Kaizen. Ambas construyen modelos a partir de datos de entrada y salida sin suposición previa sobre la expresión matemática del modelo.

Como primer acercamiento a estos algoritmos se presenta un estudio comparativo entre ambos tomando como ejemplos funciones de referencia y datos de un modelo de una columna de destilación. Del estudio se desprende que la Programación Kaizen es capaz de obtener modelos que representan mejor los datos que la Programación Genética. Por ello la Programación Kaizen es la técnica a utilizar en los siguientes casos de estudio.

En segundo lugar, se construye un *soft-sensor*, esto es un modelo que permite predecir la salida, de la columna de separación de hidrocarburos C_3 y C_4 en la planta de refinería de La Teja de ANCAP. Para la construcción del modelo se utilizaron datos históricos de la columna. A partir del modelo construido se encuentra que es un buen predictor de la salida del sistema.

Por último, debido a que en la Ingeniería Química es usual trabajar con modelos de varias variables de salida, en donde los modelos comparten expresiones en común, se diseñan y estudian dos algoritmos basados en Programación Kaizen para modelos de múltiples salidas en una sola ejecución del algoritmo. En el primer algoritmo se sustituye en la etapa de regresión lineal, la implementación para una sola variable de salida por una para múltiples salidas. El segundo algoritmo está basado en el modelo de islas para algoritmos evolutivos, en donde cada isla resuelve la búsqueda del modelo de una variable de salida, y cada determinadas iteraciones se intercambia la información entre las islas de las expresiones creadas hasta el momento. Se concluye que la estrategia basada en modelo de islas construye modelos con más precisión y con más expresiones en común que la Programación Kaizen original.

Palabras claves: Aprendizaje automático, Modelos subrogados, Algoritmos evolutivos, Soft-sensors, Ingeniería de sistemas de procesos.

ABSTRACT

The aim of this thesis is to present the application of machine learning techniques to obtain surrogate models in chemical processes. Surrogate models are black-box models that approximate the behavior of a system by fitting input-output data to combinations of simple functions. These models are used to substitute computationally expensive rigorous models, without loss of accuracy, in the context of process control and optimization.

We used Genetic Programming and Kaizen Programming as techniques for the construction of surrogate models. Both techniques belong to the Evolutionary Algorithms family. Both algorithms build data-driven models without assumptions on the mathematical expression of the model.

As a first approach to the use of these algorithms, we applied them to obtain surrogate models from benchmark functions and a distillation column. We found that Kaizen Programming is able to obtain models with better performance than the models obtained from Genetic Programming. For that reason, we use Kaizen Programming in later cases.

Afterwards, we developed a soft-sensor to estimate the composition of C4 hydrocarbons in the distillate stream of a splitter column in the oil refinery of ANCAP. For the construction of the soft-sensor, we used real historical data. In this case, we found that the obtained model had good predictions of the unit operation.

Finally, as in a Chemical Process many outputs, such as concentrations of different species, are highly related by the underlying physicochemical phenomena. We developed and evaluated two multi-output approaches based on Kaizen Programming to obtain multi-output models in a single execution of the algorithm. In the first approach, we used a multi-output linear regression instead of a single-output linear regression as in the original Kaizen Programming. The second approach was based on the islands model for parallelization of evolutionary algorithms, where each island build the model of one output variable, and after a prefixed number of iteration there is a step of exchange of expresssions beetwen the islands. We found that the strategy based on the islands model builds better models and with more terms in common than the models obtained from the single-output Kaizen Programming.

Keywords: Machine Learning, Surrogate models, Evolutionary algorithms, Soft-sensors, Process system engineering.

Lista de figuras

Lista de tablas

Tabla de contenidos

Capítulo 1

Introducción

La cuarta revolución industrial (Industria 4.0) combina técnicas avanzadas de producción con tecnologías inteligentes. En el marco de la Industria 4.0 se incluyen tecnologías y áreas del conocimiento tales como Ciencia de Datos, Internet de las Cosas, Simulación o Modelado de procesos, Robótica e Inteligencia Artificial, Sistemas de integración de procesos, Realidad Virtual, *Cloud Computing*, y Ciberseguridad. En cada uno de estos temas, la ingeniería química se ve involucrada de forma de llevar estas metodologías o herramientas a aplicaciones concretas en la industria de procesos. Esta tesis se centra en el modelado de procesos industriales a través del uso de la inteligencia computacional y la ciencia de datos, de forma de sumar esfuerzos hacia la Industria 4.0.

El modelado de procesos en su forma clásica se basa en modelos fenomenológicos de las operaciones unitarias que permiten obtener los balances de masa y energía, a partir de los cuales, dependiendo del proceso considerado, se generan sistemas de ecuaciones (algebraicas, diferenciales o diferenciales algebraicas) lineales o no lineales. Dependiendo de los fenómenos considerados, el sistema de ecuaciones resultante puede llegar a ser computacionalmente costoso, por ejemplo requiriendo un tiempo de ejecución con un orden superior al polinomial. Estos modelos, en general, son utilizados dentro de un ciclo, por ejemplo para la resolución de un problema de optimización, por lo que los tiempos o costos computacionales que conllevan suelen ser más onerosos. En la literatura reciente se propone, en este tipo de casos, usar modelos subrogados en vez de los modelos fenomenológicos.

Los modelos que aproximan el comportamiento de un sistema mediante el

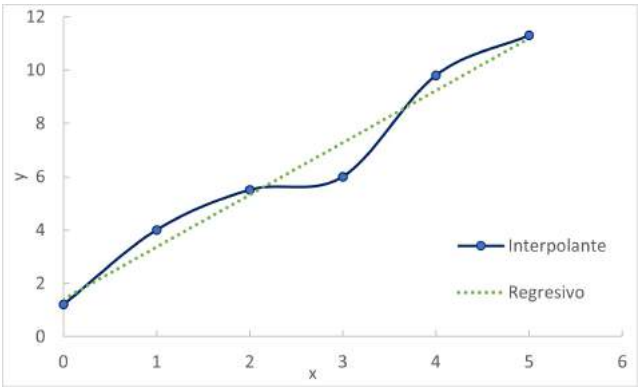


Figura 1.1: Muestra de modelo regresivo y modelo interpolante. Los datos están marcados como círculos azules.

ajuste de datos de entradas y salidas se conocen como modelos subrogados, modelos a partir de datos, modelos de respuesta de superficie o meta-modelos. Los modelos subrogados son considerados modelos de caja negra ya que el comportamiento del sistema es solo visto en términos de sus entradas y salidas, sin ningún otro conocimiento interno de su operación. Pueden clasificarse en interpolantes y regresivos, los primeros son modelos donde la predicción de cada punto experimental y su valor es el mismo, y los segundos son modelos construidos de forma de minimizar el error entre los puntos experimentales y su predicción. En la Figura 1.1 se presenta de forma gráfica la diferencia de los mismos.

1.1. Técnicas de aprendizaje automático en ingeniería de procesos

Los métodos más comunes para construir modelos subrogados son Kriging (que se conoce como regresión por Procesos Gaussianos, ProcGaus, en su versión más simple), Regresión por vector de soporte (SVR) y redes neuronales artificiales (RNA).

Kriging, SVR y ALAMO se basan en la combinación lineal de funciones, esto es: $\hat{y}(x) = \sum_i \omega_i f_i(x)$, para la construcción del modelo. Los tres métodos se diferencian en como son halladas las funciones bases ($f_i(x)$) y los coeficientes (ω_i).

Kriging fue propuesto por Krige (1952). El método asume una distribución previa (o *a priori*) de los datos, en donde el modelo obtenido tiene la forma

de suma ponderada de bases independientes cuyo error sigue una distribución normal con promedio cero. Es un método de interpolación exacta, en donde los coeficientes están relacionados con la covarianza de los datos. Como ejemplo de aplicación de esta técnica está el trabajo de Caballero y Grossmann (2008), en el cual se remplazan los modelos rigurosos de varias columnas de destilación y un reactor de flujo pistón por sus subrogados, para realizar una optimización del diagrama de flujo de forma modular.

Automated Learning of Algebraic Models for Optimization (ALAMO) es un método proveniente del área de Ingeniería Química, propuesto por Cozad et al. (2014). En este caso, los modelos son funciones de bases no lineales predefinidas (por ej. polinomios, exponenciales y logarítmicas). Los coeficientes son calculados minimizando la suma de los errores al cuadrado en un subconjunto de datos llamado conjunto de entrenamiento. ALAMO permite utilizar un método de diseño experimental adaptativo, en donde de forma iterativa se va aumentando la cantidad de datos de entrenamiento, así como la complejidad del modelo a aprender. En el trabajo de Cozad et al. (2014), ALAMO fue utilizado para la búsqueda de un modelo en una columna de captura de carbono.

SVR construye modelos usando funciones bases, conocidas como *kernels*, y los coeficientes son hallados resolviendo un problema de optimización, en el cual se minimiza la norma del vector de coeficientes y la suma de la desviaciones de cada dato con el modelo (Smola y Schölkopf, 2004). SVR y ALAMO se diferencian, principalmente, en el problema de optimización que resuelven para obtener el modelo. Un ejemplo de aplicación de esta técnica es el trabajo de Zaghloul et al. (2020), en donde se utiliza SVR para encontrar un modelo de un reactor anaerobio de lodo granular.

Una técnica de aprendizaje automático muy utilizada, y con características diferentes a las anteriores son las RNA (Goodfellow et al. 2016). En estas, una capa de muchos nodos interiores, en general cientos o miles, conectan los nodos de entrada (datos de entrada) con los nodos de salida (datos de salida), a través de aristas que indican los pesos de los coeficientes de propagación de la conexión. Dichos pesos son encontrados en la etapa de aprendizaje del modelo, al minimizar el error ajustando los costos de las aristas. Para capturar no linealidades, se incluyen funciones de activación en las aristas, como parámetros del algoritmo. Como ejemplo de aplicación se encuentran los trabajos de Schweidtmann et al. (2019), que utiliza modelos de las propiedades termodinámi-

cas en un ciclo Rankine orgánico, obtenidos por RNA, los cuales se utilizan para resolver un problema de optimización global; Henao y Maravelias (2011), que hace uso de los modelos subrogados para remplazar varias operaciones unitarias (RCAs, *mixers*, tanques flash) en el contexto de la optimización de superestructuras de procesos para la producción de anhídrido maleico a partir de benceno; Zhu et al. (2020), que utilizan aprendizaje profundo (AP, RNA con más de una capa de nodos interiores) para obtener modelos subrogados para predecir la dinámica del proceso y las ganancias para un proceso de expansión criogénica para la extracción de gas natural.

Las técnicas de aprendizaje automático basados en algoritmos evolutivos, como la Programación Genética (PG) y la Programación Kaizen (PK), se han utilizado en otras áreas del conocimiento como en las ciencias de la computación, y otras áreas de la ingeniería, pero han sido pocas aplicaciones en la Ingeniería Química. Dentro de los pocos trabajos existentes, se destacan el trabajo de Willis et al. (1997) para obtener el modelo dinámico de una columna de destilación binaria, el trabajo de Hinchliffe y Willis (2003) para la construcción de un modelo dinámico de una extrusora, el trabajo de Cai et al. (2006) para crear modelos de correlación de transferencia de calor, y el trabajo de Dürrenmatt y Gujer (2012) para modelar los reactores de una planta de tratamiento de agua. Debido a que los algoritmos PG y PK son los principales algoritmos utilizados a lo largo de la presente tesis, en las Secciones 2.1 y 2.2 se explican en detalle ambos métodos, respectivamente.

1.2. Objetivos y esquema de la tesis

En la presente tesis se ha planteado como **objetivo general** la aplicación de técnicas evolutivas de aprendizaje automático aplicadas a la industria de procesos. Los **objetivos específicos** son:

- el uso y la aplicación de la Programación Genética como método para obtener modelos subrogados de sistemas de procesos químicos,
- la implementación, el uso y la aplicación de la Programación Kaizen como método para obtener modelos subrogados de sistemas de procesos químicos,
- el diseño, implementación y evaluación de variantes del algoritmo de Programación Kaizen para poder generar varios modelos subrogados en una

sola ejecución del algoritmo, de forma de tener en cuenta la posible relación existente entre las distintas variables de salida cuando comparten los mismos fenómenos físico-químicos con las mismas variables de entrada.

Los algoritmos de PG y PK han sido elegidos debido a que devuelven modelos a partir de datos con una forma interpretable, y, a su vez, son métodos que tienen potencial para ser utilizados en la generación de modelos híbridos (modelos que utilizan técnicas clásicas, como el modelado fenomenológico de procesos, y de aprendizaje automático). Según Pistikopoulos et al. (2021) y Reis y Saraiva (2021) el uso de modelos híbridos conforman las bases para el futuro de la industria 4.0.

Como parte de la elaboración de la presente tesis se han generado los siguientes trabajos:

Artículos en revistas internacionales:

- Ferreira, Pedemonte y Torres (2022a), que fue publicado en la revista *Computers and Chemical Engineering*. En dicho trabajo se obtuvo un modelo (ensamblado) para la concentración de hidrocarburos de 4 carbonos en la corriente de destilación, en una columna de destilación de la refinería de ANCAP, utilizando datos históricos de la planta. Para la obtención de los modelos se utilizó Programación Kaizen y se realizó la comparación de la predicción del modelo ensamblado con un modelo obtenido con Procesos Gaussianos.
- Ferreira, Torres y Pedemonte (2022), que es un artículo que ha sido aprobado para su publicación en la revista *Neural Computing and Applications*. En dicho trabajo se presentó un algoritmo para problemas de aprendizaje automático con múltiples salidas basado en el modelo de islas para la paralelización de algoritmos evolutivos. Se realizó la evaluación de la técnica propuesta con varias estrategias múltiples salidas utilizando datos a partir de funciones de referencia (*benchmark*), y datos de simulaciones de cuatro sistemas de procesos químicos.

Artículos completos en conferencias internacionales arbitradas:

- Ferreira, Pedemonte y Torres (2019), que fue presentado en el congreso internacional *Foundations of Computer-Aided Process Design* (FOCAPD). El trabajo completo se encuentra disponible en la re-

vista *Computer Aided Chemical Engineering*. El artículo presenta el uso de la PG para el modelado de una columna de destilación.

- Ferreira, Torres y Pedemonte (2019), que fue presentado en el congreso latinoamericano *Latin American Conference on Computational Intelligence* (LA-CCI), y se encuentra disponible en el portal de la IEEE. En dicho trabajo se presenta la comparación de los algoritmos de PG y PK para obtener modelos con datos a partir de funciones de referencia.
- Ferreira, Torres y Pedemonte (2021), en el cual se presenta un primer algoritmo para problemas de aprendizaje automático con múltiples salidas, y se utilizaron datos a partir de funciones de referencia. El mismo fue presentado en el congreso latinoamericano *Latin American Conference on Computational Intelligence* (LA-CCI), y se encuentra disponible en el portal de la IEEE.
- Ferreira, Pedemonte y Torres (2022b), en donde se aplica un algoritmo de aprendizaje automático con múltiples salidas en cuatro sistemas de procesos químicos. Dicho trabajo fue presentado en el congreso internacional *International Symposium on Process Systems Engineering* (PSE *Symposium*) y el trabajo completo se encuentra disponible en la revista *Computer Aided Chemical Engineering*.

Presentaciones en conferencias internacionales:

- Ferreira, Pedemonte y Torres (2021a), en donde se obtuvieron modelos para la concentración de hidrocarburos de 4 carbonos en la corriente de destilación, en una columna de destilación de la refinería de ANCAP, utilizando datos históricos de la planta. El trabajo fue presentado en el congreso internacional *American Institute of Chemical Engineers Meeting* (AICHE *Meeting*). Además, a partir de este trabajo se realizó una presentación oral en el Congreso Regional de Ingeniería Química (Ferreira, Pedemonte y Torres, 2021b).

La tesis se organiza de la siguiente forma. El Capítulo 2 presenta una breve descripción de los dos algoritmos utilizados en esta tesis, PG y PK. El Capítulo 3 presenta una primera aproximación a la PG en el modelado de una columna de destilación. En el Capítulo 4 se muestra el uso y comparación de la PG y la PK para obtener modelos a partir de datos de quince funciones de referencia y una columna de destilación. Como resultado principal se encontró que

PK construye modelos con mejor desempeño que los generados por PG. En el Capítulo 5 se presenta la obtención de un modelo a partir de datos históricos de una columna de destilación de la planta de refinación de ANCAP. En dicho capítulo se presentan los pasos realizados desde la recepción de los datos hasta la generación de un *soft-sensor* basado en el ensamblado de modelos generados por PK. En el Capítulo 6 se muestra dos algoritmos, basados en PK, para la construcción de varios modelos en una sola ejecución del algoritmo, utilizando datos a partir de cinco familias de funciones de referencia y cuatro casos de la ingeniería de procesos. La necesidad de generar modelos con múltiples salidas radica en que las variables de salida de un sistema de procesos, en general, comparten los fenómenos físico-químicos que suceden en el mismo, por lo que los modelos construidos deberían reflejar en sus expresiones el fenómenos compartido. Por último, en el Capítulo 7 se presentan las conclusiones y propuestas de trabajos a futuro.

Capítulo 2

Programación Genética y Programación Kaizen

La PG y la PK son algoritmos de la familia de los algoritmos evolutivos para resolver problemas de Regresión Simbólica (RS). En la RS se busca un modelo que a partir de datos de entrada y salidas, sin realizar ningún tipo de suposición sobre el modelo, es decir, se trata de encontrar las funciones y las constantes que forman la expresión que mejor se adaptan a los datos.

La resolución de problemas de RS con métodos de optimización exactos conlleva una complejidad computacional del tipo NP (*Nondeterministic Polynomial time*), esto es, en términos prácticos, que el aumento de la cantidad de variables de entrada del problema aumenta drásticamente su tiempo de resolución. Ejemplos de resolución exacta son los trabajos de Austel et al. (2017, 2020), Cozad y Sahinidis (2018) y Engle y Sahinidis (2022). En contraste con la resolución exacta, la resolución utilizando heurísticas o metaheurísticas como los algoritmos evolutivos, es de complejidad computacional P (*Polynomial time*), por lo que se puede llegar a resolver problemas de mayor cantidad de variables sin aumentar significativamente el costo computacional, pero con la desventaja que no existe certeza de la optimalidad sobre la solución encontrada. Debido a que la PG fue unos de los primeros métodos propuestos para resolver problemas de regresión simbólica, en la bibliografía se suele encontrar el uso de la PG y RS como sinónimos.

2.1. Programación Genética

La PG es una técnica de inteligencia computacional propuesta por Koza (1992), perteneciente a la familia de los algoritmos evolutivos, diseñada especialmente con el fin de evolucionar programas. Estos algoritmos evolucionan de forma iterativa un conjunto de soluciones candidatas (este conjunto se lo denomina población) del problema a resolver. El proceso de búsqueda se desarrolla mediante la aplicación probabilística de operadores evolutivos (cruzamiento, mutación y reproducción) para encontrar mejores soluciones, y es guiado por el principio de supervivencia del más apto, esto es, sobreviven en las siguientes generaciones las soluciones mejor adaptadas. Para determinar adaptación se utiliza una métrica relacionada con el problema a resolver, que se denomina función de *fitness*.

Los programas sobre los que trabaja la PG pueden ser tanto programas generales de computación, como expresiones lógicas o matemáticas. En el caso del uso de expresiones matemáticas, el problema a resolver es una RS.

En la PG las soluciones candidatas son representadas como árboles de expresiones. En contraste con la mayoría de los métodos para resolver la generación de modelos subrogados, que asumen una cierta expresión matemática para el modelo, la PG aprende el modelo directamente de los datos. El espacio de búsqueda para la PG es el espacio de todos los árboles de expresión que pueden ser creados, compuestos por las funciones y terminales disponibles para el problema. Por lo tanto, la PG no utiliza ningún modelo específico para aprender la expresión matemática, sino que es inferido directamente de los datos.

En la Figura 2.1 se muestra las principales etapas del algoritmo PG. El algoritmo comienza generando la población de posibles bases, esto es, los árboles que representan las fórmulas matemáticas, que son los posibles modelos del conjunto de datos. Para cada uno de los individuos de la población se calcula el valor de *fitness*. A partir de allí, se realiza la generación de los árboles hijos (*offsprings*). En cada generación, la cantidad de árboles hijos generados es igual a la cantidad de árboles padres, debido a que el tamaño de la población se mantiene fijo a lo largo de la ejecución del algoritmo. Por último, se realiza el cálculo de la función de *fitness* para los árboles hijos. Los pasos de creación de árboles hijos y cálculo del valor de *fitness* se repiten hasta que se llega a un número máximo de iteraciones (generaciones) o se cumple alguna condición

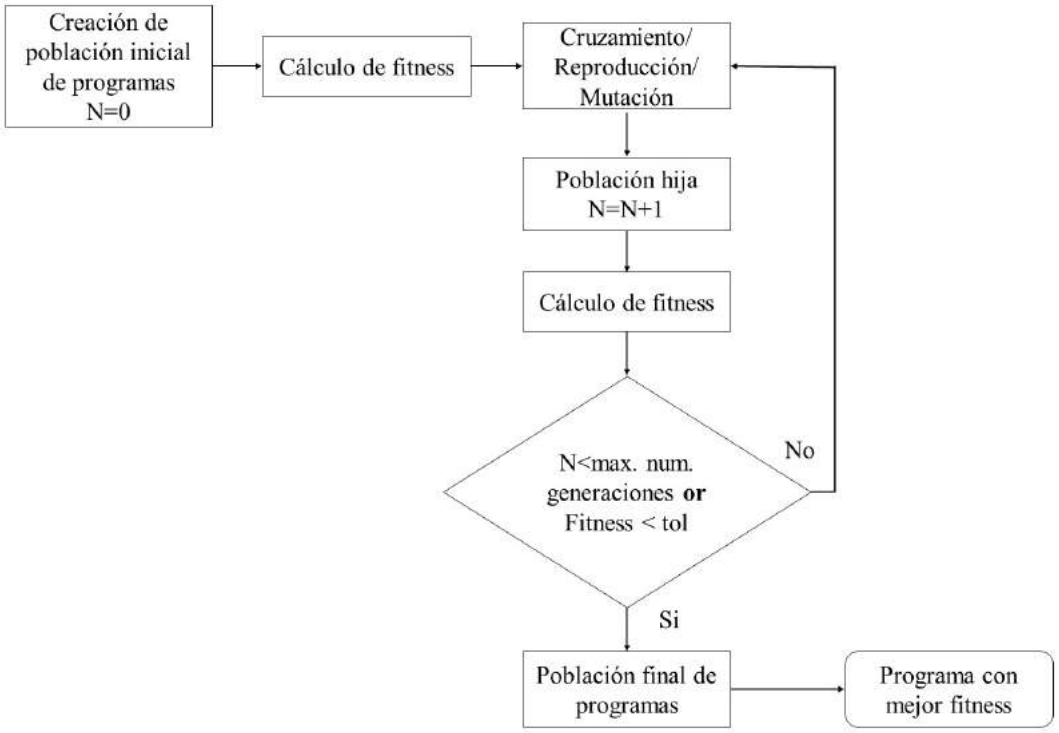


Figura 2.1: Esquema general del algoritmo de PG

relacionada con la función de *fitness* (por ej. si la función de *fitness* corresponde a un valor menor a una tolerancia predefinida, el algoritmo termina). Por último, el individuo con menor valor de *fitness* es devuelto como el modelo de mejor ajuste.

Como se mencionó anteriormente, la PG trabaja con árboles de expresión como estructura para guardar las soluciones candidatas. Para crear los árboles se trabaja con dos conjuntos que son parámetros del algoritmo, el conjunto de funciones y el conjunto de terminales. El conjunto de funciones contiene todas las posibles operaciones matemáticas o funciones que puede tener las expresiones candidatas, por ejemplo suma, resta, división, logaritmo, coseno, etc.; y el conjunto de terminales contiene a todas las variables de entrada y los posibles valores de constantes. Si bien se puede utilizar un conjunto finito de posibles constantes, la forma más usual para la creación de constantes es el uso de Constantes Efímeras Aleatorias (*Ephemeral Random Constant*), de esta forma se crean números de forma aleatoria siguiendo una distribución probabilística predefinida. El uso de Constantes Efímeras Aleatorias permite crear árboles con una gran variedad de valores en las constantes.

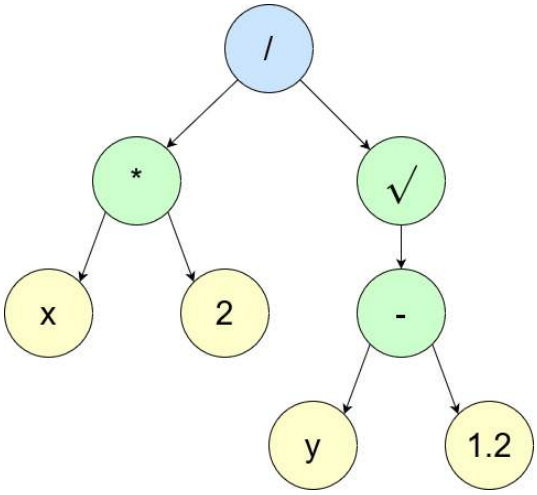


Figura 2.2: Ejemplo de árbol de expresión para $2x/\sqrt{y-1.2}$

2.1.1. Estructuras base e inicialización

Para la creación de árboles de expresión se debe tener en cuenta que cada nodo interior (esto incluye al nodo raíz) de un árbol de expresión es ocupado por una operación aritmética (simple) dada por el conjunto de funciones. El número de hijos de cada nodo interior debe ser la misma que la cantidad de operandos necesarios por la operación (por ej. la suma debe tener dos hijos y el logaritmo debe tener uno). Además, las hojas o nodos exteriores del árbol de expresión se corresponden con una variable o una constante del conjunto de terminales. La Figura 2.2 muestra un ejemplo de árbol de expresión, que representa la expresión $2x/\sqrt{y-1.2}$, en azul se representa el nodo raíz, en verde a los nodos interiores, y en amarillo a las hojas o nodos exteriores).

La población inicial se genera aleatoriamente, con tres posibles mecanismos para la inicialización:

1. en el método *full*, se generan árboles en donde las hojas de cada árbol construido tiene el mismo nivel (distancia o cantidad de nodos entre la raíz y la hoja), y el nivel es especificado previamente;

2. en el método *grow*, se generan árboles de distintos tamaños y formas, en donde se van seleccionando aleatoriamente nodos hasta haber alcanzado la altura máxima especificada;

3. y el método *half-and-half*, en donde cada uno de los métodos anteriores construye la mitad de la población.

2.1.2. Función de *fitness*

Cada solución candidata tiene asociada un valor de *fitness* que mide su calidad y guía el proceso de búsqueda. En regresión simbólica usualmente se utiliza una métrica relacionada con el error entre la salida dada como dato y la solución producida. Ejemplos de estas métricas son: el error cuadrático medio (MSE, *Mean Squared Error*), cuyo cálculo es $MSE = \sum_{i=1}^{n} \frac{(y_i^{dato} - y_i^{estimado})^2}{n}$ y la raíz del error cuadrático medio (RMSE, *Root-Mean-Square Error*), que es la raíz del MSE.

Existen implementaciones en donde se utiliza como función de *fitness* una métrica relacionada con el residuo del modelo obtenido y los datos, y un término que penaliza la cantidad de nodos (o la cantidad de niveles) del árbol evaluado. De esta forma, se busca no solo que las expresiones encontradas sean buenos estimadores de los datos, sino que la expresión sea lo más simple posible.

2.1.3. Selección y operadores evolutivos

El paso de creación de los descendientes (*offsprings*) puede dividirse en dos grandes etapas: selección y aplicación de operadores evolutivos. En la etapa de selección, las soluciones existentes son seleccionadas para la reproducción basadas es su valor de *fitness* (se le da mayor prioridad a las soluciones de mayor calidad). Uno de los métodos de selección más usado es la selección por torneo (*tournament selection*); en dicho método se seleccionan de forma aleatoria N_{selec} individuos de la población, y el ganador del torneo (aquel que tiene el mejor valor de *fitness*) es seleccionado para la etapa de operaciones evolutivas. Por lo que se seleccionan, al menos, tantos individuos como el número de árboles hijos que serán creados.

Por otro lado, en la etapa de operaciones evolutivas, se construyen nuevas soluciones aplicando de forma probabilística los operadores de cruzamiento, mutación y reproducción a las soluciones previamente seleccionadas. En la implementación de PG, en primer lugar se elige probabilísticamente una de las operaciones evolutivas, y posteriormente, sucede la selección por torneo. Por lo que dependiendo de la operación evolutiva a aplicar es la cantidad de árboles padres necesarios para generar el árbol hijo. La nueva población es creada con las nuevas soluciones en cada iteración. En los siguientes párrafos se explica el funcionamiento de los operadores de cruzamiento y mutación más utilizados.

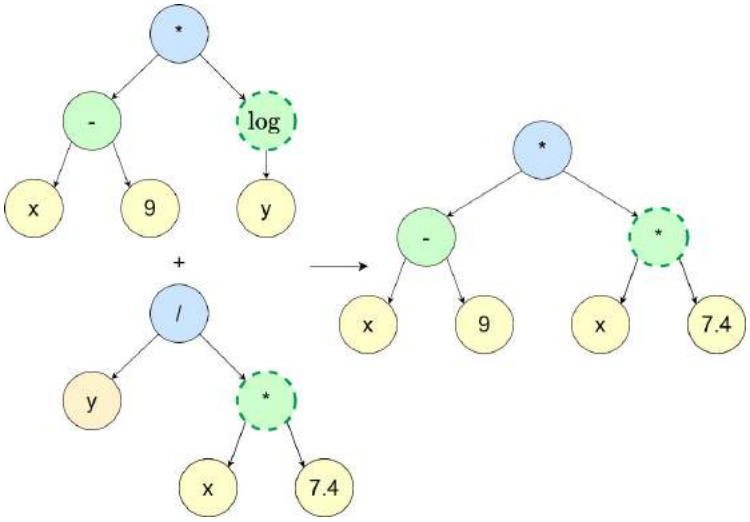


Figura 2.3: Ejemplo de cruzamiento

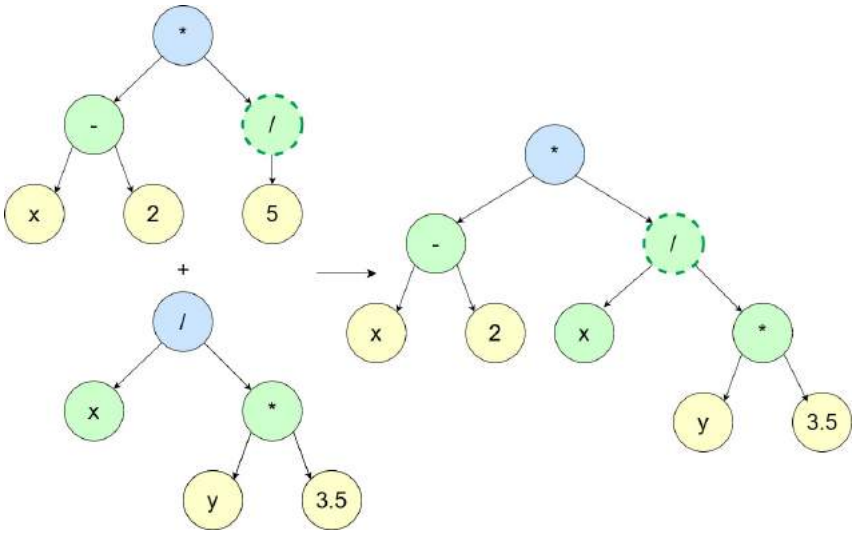


Figura 2.4: Ejemplo de mutación en un punto

En la operación de cruzamiento se toman dos soluciones candidatas y se produce una solución potencialmente nueva. Para ello, se selecciona aleatoriamente un nodo de cada árbol padre, luego el subárbol del primer árbol padre es remplazado con el subárbol del segundo árbol padre. En la Figura 2.3 se muestra el cruzamiento entre dos soluciones candidatas (los árboles padres están a la izquierda y el árbol hijo está a la derecha).

La operación de mutación toma una solución candidata y la modifica aleatoriamente. La operación más usual es la mutación de subárbol, en donde se toma un nodo de forma aleatoria del árbol padre y el subárbol es remplazado por un subárbol nuevo creado con alguno de los métodos de inicialización

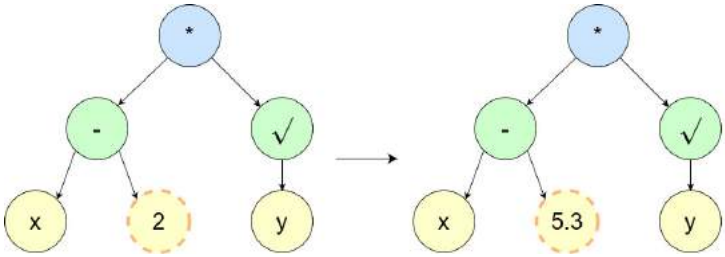


Figura 2.5: Ejemplo de mutación de una constante

mencionados anteriormente.

Por otra parte, existen otros tipos de mutaciones, como por ejemplo, la mutación de nodo, en donde un nodo es seleccionado aleatoriamente e intercambiado por una función del conjunto de funciones con la misma aridad o por un nodo terminal; o la mutación de contantes, en donde se toma aleatoriamente un nodo terminal que tiene una constante, y se la intercambia por otro número de los posibles valores. En la Figura 2.4 se muestra la mutación de subárbol en un punto, mientras que en la Figura 2.5 se muestra un ejemplo de mutación de constante.

En la operación de reproducción simplemente se realiza una copia del árbol que se pasa sin modificaciones a la siguiente generación.

2.1.4. Problemas encontrados en la programación genética

Desde su creación en 1992 se han publicado muchos trabajos que utilizan la PG como técnica para resolución de problemas de RS. Las aplicaciones van desde problemas sintéticos, en donde se conocen las ecuaciones que generan los datos, hasta problemas con datos reales, en las más diversas áreas de aplicación como ingeniería, física, economía, entre otras.

Si bien la PG resuelve y encuentra buenos modelos en varias aplicaciones, se han encontrado algunas desventajas en los modelos generados (Poli et al. 2008). La primera desventaja es el problema de *bloat*, que es el crecimiento desmesurado del tamaño de los árboles, generando expresiones matemáticamente complejas. La forma que se ha encontrado para resolver este problema fue la creación de un parámetro de altura máxima de árbol o la inclusión dentro de la función de *fitness* de un coeficiente de *parsimonia* de forma de penalizar el número de nodos en el árbol. La segunda desventaja es el hecho de que las

constantes de los árboles son producidas en la creación del árbol, pero luego nunca cambian su valor. Para ello se han realizados varios trabajos en busca de mejorar esta restricción. El uso de mutación de constante es una de las medidas creadas para mejorar el valor de las constantes.

Por otra parte, inspirados en la técnica de PG, se han creados métodos basados en la PG en donde las estructuras sobre las que se trabaja no son árboles, como por ejemplo en la PG cartesiana, se utiliza un vector que representa una expresión matemática (Miller, 2020). O métodos que se basan en la estructura de la PG para utilizar los árboles de una población como términos de una sola expresión matemática, como la PK.

2.2. Programación Kaizen

El algoritmo de Programación Kaizen (PK) fue propuesto como una mejora de PG para la resolución de problemas de RS. Dentro de los cometidos a mejorar respecto a PG están la reducción del crecimiento innecesario del tamaño de los árboles de expresión y la presencia de Constantes Efímeras Aleatorias, las cuales permanecen sin cambios durante la ejecución del algoritmo (de Melo, 2014; de Melo y Banzhaf, 2018). Como resultado, en comparación con la PG, la PK usa expresiones que suelen ser más cortas para ajustar los datos.

La PK construye modelos del tipo $y(x) = \sum_{i=1}^{n} \beta_i f_i(x)$, donde cada f_i es generada por operaciones de PG, y los coeficientes β_i son ajustados a través de una regresión lineal, por ejemplo con regresión por mínimos cuadrados ordinarios (OLS, *Ordinary Least Squares*). Es así que a la PK se la puede ver como una algoritmo híbrido que combina técnicas evolutivas con el uso de técnicas estadísticas de regresión determinísticas. El algoritmo tiene 6 etapas principales que se repiten iterativamente: generación de árboles hijos, creación de población extendida (es la unión de la población madre e hija), eliminación de expresiones linealmente dependientes, primera regresión lineal para la estimación de los $p-$valores, eliminación de bases funcionales, y segunda regresión lineal con la población reducida. En la Figura 2.6 se muestra un esquema del algoritmo básico de la PK.

Si bien el algoritmo fue creado por de Melo y Banzhaf (2018) al comienzo del desarrollo de esta tesis no existía una implementación disponible públicamente, por lo que se procedió a realizar una implementación propia. Para su implementación se tuvieron que tomar varias decisiones sobre las distintas

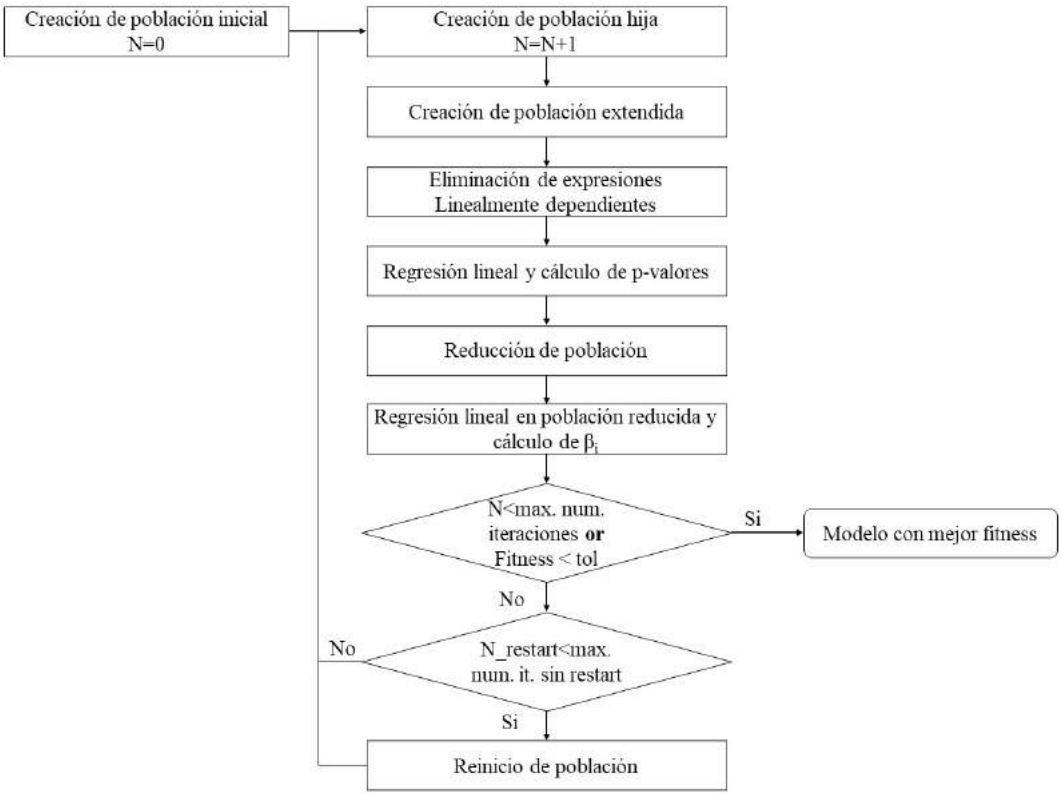


Figura 2.6: Esquema del algoritmo de PK

etapas del algoritmo que no estaban explicitadas en los trabajos de presentación del algoritmo PK. Para la etapa de eliminación de individuos linealmente dependientes (LD) se eligió uno de los métodos utilizados comúnmente para evaluar los términos LD, y se agregó la etapa de la segunda regresión lineal de forma de obtener un modelo que reflejara la información de los individuos de la población final de cada iteración. La implementación del algoritmo se encuentra disponible en un repositorio de acceso público en: GitHub[1], y está basada en la utilización de los paquetes `deap` para las operaciones de PG y `scikit-learn` para las operaciones de la regresión lineal (R^2, p−valores, RMSE, Pedregosa et al., 2011).

2.2.1. Inicialización

Inicialmente, un conjunto de P^{max} funciones bases (f_i) (población inicial) es generado de forma aleatoria. P^{max} representa la población máxima y es ingresado como parámetro del algoritmo; en otras palabras, es la cantidad de

[1] https://github.com/JimenaFerreira/KaizenProgramming

funciones bases máxima que va a tener el modelo a obtener. Así como en la PG, cada (f_i) se corresponde con una expresión construida de la combinación de funciones simples (*exp*, *log*, *sin*, etc.), de operadores matemáticos (+, −, ∗, /) y el conjunto de terminales, ingresados como parámetros del algoritmo. Al igual que en PG, la inicialización puede ser *full*, *grow* o *half-and-half*.

2.2.2. Creación de población extendida

A continuación de la inicialización, comienzan las etapas iterativas. En primer lugar, se crea un conjunto expandido de funciones bases por la adición de nuevas funciones bases a la población actual, mediante la aplicación de cruzamiento y mutación a los individuos actuales. Al conjunto de nuevas funciones se la denomina población hija, y en general, se genera con tamaño P^{max}. Cabe destacar que como en la etapa siguiente se utiliza la población madre e hija, no se utiliza la reproducción como operador evolutivo en la etapa de generación de árboles hijos. El tamaño de la población, la probabilidad de cruzamiento y mutación son parámetros del algoritmo.

2.2.3. Preparación y primera regresión lineal

Como las funciones bases son generadas de forma independiente entre ellas, la población expandida resultante puede ser linealmente dependiente (LD). Por lo tanto, para realizar una regresión lineal por el método de OLS, se requiere una etapa de identificación y eliminación de las funciones bases LD. En nuestra implementación dicho proceso se realizó utilizando técnicas numéricas.

La identificación y eliminación de las funciones bases se realizó a través del uso de la matriz F, conformada por la evaluación de cada función base, f_i, para cada dato, x_i. Para que las funciones bases sean LD, F debe cumplir que sus columnas sean linealmente independientes. A continuación, en la Ecuación 2.1, se muestra un esquema general de la matriz F.

$$F = \begin{pmatrix} \begin{bmatrix} f_1(x_1) \\ f_1(x_2) \\ f_1(x_3) \\ \vdots \\ f_1(x_n) \end{bmatrix} & \begin{bmatrix} f_2(x_1) \\ f_2(x_2) \\ f_2(x_3) \\ \vdots \\ f_2(x_n) \end{bmatrix} & \dots & \begin{bmatrix} f_p(x_1) \\ f_p(x_2) \\ f_p(x_3) \\ \vdots \\ f_p(x_n) \end{bmatrix} \end{pmatrix} \tag{2.1}$$

Luego de hallada la matriz F, se realiza la descomposición de F en las matrices R y Q ($F = QR$) a través de la descomposición QR (`linalg.qr`) que está disponible en el paquete `numpy` de `Python`. Si F tiene columnas LD, R es singular y tiene valores es su diagonal principal que son cero, por lo tanto, las expresiones asociadas a los valores cero deben ser descartados. Debido a las aproximaciones numéricas al realizar cálculos en una computadora, la diagonal de R no siempre tiene valores 0 sino que puede tener valores muy cercanos a 0. Por esta situación, en estos casos, es necesario usar una tolerancia para descartar las expresiones asociadas a valores en la diagonal de R menores a tol_R. En el presente trabajo se utilizó como base la tolerancia que utiliza `Matlab` para la función `rank`. El valor de tol_R utilizado es
$tol_R = \epsilon_{FT} \cdot max(abs(diag_R)) \cdot max(R_{shape})$, donde ϵ_{FT} es la precisión de los números en punto flotante, $diag_R$ es un vector con los elementos de la diagonal principal de R, y $max(R_{shape})$ es el valor máximo entre la cantidad de filas y columnas de la matriz R.

Luego de eliminadas las funciones LD, los coeficientes (β_i) son calculados resolviendo la Ecuación 2.2:

$$\min_{\beta} \sum_{j=1}^{n} (y_j - \sum_{i=1}^{P} \beta_i f_i(x_j))^2, \tag{2.2}$$

donde n es el número de puntos experimentales y P es el tamaño de la población en esta etapa de la iteración (el número de funciones bases linealmente independientes).

En nuestra implementación para la obtención de los coeficientes se utilizó `linear_model.LinearRegression`, que pertenece al paquete `sklearn` de `Python`, así como `sklearn.metrics` para el cálculo de las métricas obtenidas de la regresión (p−valor y R^2).

2.2.4. Reducción de la población y segunda regresión lineal

Luego de realizada la regresión lineal, se lleva a cabo una etapa para identificar el conjunto de individuos más relevantes. Esto se realiza en 3 subetapas. La primera, en donde solo se mantienen los individuos estadísticamente significativos en base al p−valor del test t de Student; en donde la hipótesis nula verifica si $\beta_i = 0$ y la hipótesis alternativa es $\beta_i \neq 0$ con un nivel de signifi-

cancia α. Si el p−valor es menor o igual a α la hipótesis nula es rechazada, por lo tanto el coeficiente no es cero. En otras palabras, se mantienen las bases funcionales cuyo p−valor asociado es menor o igual a α. En el Apéndice 2 se presenta una breve explicación de la prueba t de Student.

En la segunda etapa, aunque sean estadísticamente significativos, los términos cuyos β_i son menores a cierto valor predefinido θ son descartados. Esto se debe a que dichos términos son despreciables frente al resto de términos de la ecuación.

Finalmente, en la tercera etapa, si luego de las subetapas anteriores el tamaño de la población es mayor a P^{max}, se descartan los individuos sobrantes en base al p−valor. Esto es, permanecen aquellos árboles cuyo p−valor es menor.

Las 3 subetapas mencionadas anteriormente llevan a una población más chica de P' funciones bases. Por último, con este conjunto de bases relevantes, se calculan con OLS nuevos coeficientes β_i.

2.2.5. Finalización de la iteración

Por último se realizan los cálculos relativos a la función de *fitness*. En la PK como función de *fitness* se utiliza $1 - R^2.Ajust$, donde $R^2.Ajust$ es una versión ajustada del coeficiente de correlación R^2 obtenido mediante OLS. El valor de $R^2.Ajust$ se calcula a partir de la Ecuación 2.3. Este enfoque incluye la cantidad de términos en la función de *fitness*, lo que permite que los mejores modelos sean aquellos que mejoran la estimación con la mínima cantidad de términos. En cambios si se usara R^2 todo término que incrementara el valor sería incluido.

$$R^2.Ajus = 1 - (1 - R^2)\frac{N - 1}{N - P' - 1} \tag{2.3}$$

El valor de *fitness* de la iteración actual es comparado con el mejor obtenido hasta ese momento; si es mejor el valor actual, se mantiene la población, sino se descarta, y se continua la iteración con la mejor población hasta el momento.

Si el valor de la función de *fitness* es menor a un valor de tolerancia, *tol*, o se ha llegado a la cantidad máxima de iteraciones, el algoritmo devuelve el mejor modelo encontrado hasta ese momento. De lo contrario, puede suceder una etapa de reinicio. La etapa de reinicio tiene como cometido evadir los casos en que el algoritmo haya encontrado un óptimo local. De esta forma, cada cierto número de iteraciones predefinido, si el valor de función de *fitness*

no ha cambiado, se reinicia la población actual, y se continua con la siguiente iteración.

2.2.6. Comentarios adicionales sobre el algoritmo

El algoritmo permite ingresar un tamaño máximo de niveles de los árboles, lo que permite a lo largo de las iteraciones construir árboles más chicos que en PG. Y sumado a la restricción de la cantidad de funciones bases, permite encontrar expresiones matemáticas más simples que en PG.

PK mejora los problemas de *bloat* al restringir el tamaño de las funciones bases y por otro lado al realizar la búsqueda de coeficientes por regresión lineal, permite tener constantes en el modelo que varían su valor a lo largo de las iteraciones. Si bien con estos cambios se mejora la búsqueda de modelos, PK sigue teniendo problemas con las constantes generadas aleatoriamente en la construcción de los árboles. Dichas contantes permanecen sin cambios a lo largo de las iteraciones, condicionando así la forma de la expresión matemática.

Capítulo 3

Primer caso de estudio: Generación de modelos mediante programación genética con datos de un simulador de procesos

En el presente capítulo se utiliza la PG para generar un modelo subrogado a partir de datos generados por la resolución de una hoja de procesos en Aspen Plus. En este caso, es posible definir la toma de datos de entrada y salida. Esto permite definir la toma de datos para que se pueda percibir la influencia (o no) de las variables de entrada en las variables de salida. Para esto, es necesario utilizar un método de diseño experimental (DoE). De esta forma, se puede definir la toma de datos para la obtención del modelo (conjunto de entrenamiento o aprendizaje) y/o para la validación del mismo (conjunto de validación), y así evitar el sobreajuste de los datos (*overfitting*). Cabe destacar, que tanto la PG como la PK no tienen incorporado un mecanismo para evitar el sobreajuste de los datos como parte de la obtención del modelo.

En la siguiente sección se presenta un breve resumen de los tipos de diseños experimentales existentes. Luego se presenta el caso de estudio, los conjuntos de entrenamiento y validación, los parámetros del algoritmo utilizados, y por último, los resultados y la discusión del caso en estudio.

3.1. Introducción al diseño de experimentos

Los distintos métodos se pueden clasificar en clásicos y modernos, y los modernos se pueden subdividir en estáticos y adaptativos. Según Garud et al. (2017) las técnicas clásicas fueron diseñadas para tener en cuenta la naturaleza aleatoria de los experimentos físicos, mientras que las modernas son aquellas que trabajan con datos computacionales y asumen simulaciones determinísticas.

Dentro de los métodos estáticos se puede incluir información del sistema (*System-Aided DoE*) o no (*System-Free DoE*). Por ejemplo, dentro de los primeros se encuentran el método de muestreo por máxima entropía y el método basado en el error cuadrático medio; y como parte de *System-Free DoE* se encuentran el método de Monte Carlo y el muestreo de hipercubo latino. Al contrario de los métodos estáticos, los adaptativos adaptan la cantidad de datos de muestreo en función del valor de un indicador de la calidad del modelo.

Construir modelos subrogados requiere un número mínimo de datos en los conjuntos de entrenamiento y validación. Sin embargo, no hay consenso en la literatura de la cantidad mínima de puntos, o la relación de cantidad de puntos entre los conjuntos de entrenamiento y validación. Basado en esto, se utilizó un método adaptativo para construir el conjunto de entrenamiento.

3.2. Caso de estudio, implementación y parámetros

En la Figura 3.1 se muestra el proceso como primer caso de estudio. Esta hoja de proceso está incluida como recurso de *Aspen Plus*, en la sección *bio-ethanol from corn example*. Se toma de aquí los datos de la primera columna de destilación (*RadFrac*) en el proceso de producción de bioetanol a partir de maíz.

Para la generación de datos de forma iterativa se realizó la automatización de la conexión entre *Python* y *Aspen Plus*, de forma que las simulaciones fueran ejecutadas desde *Python* y no desde la interfaz gráfica.

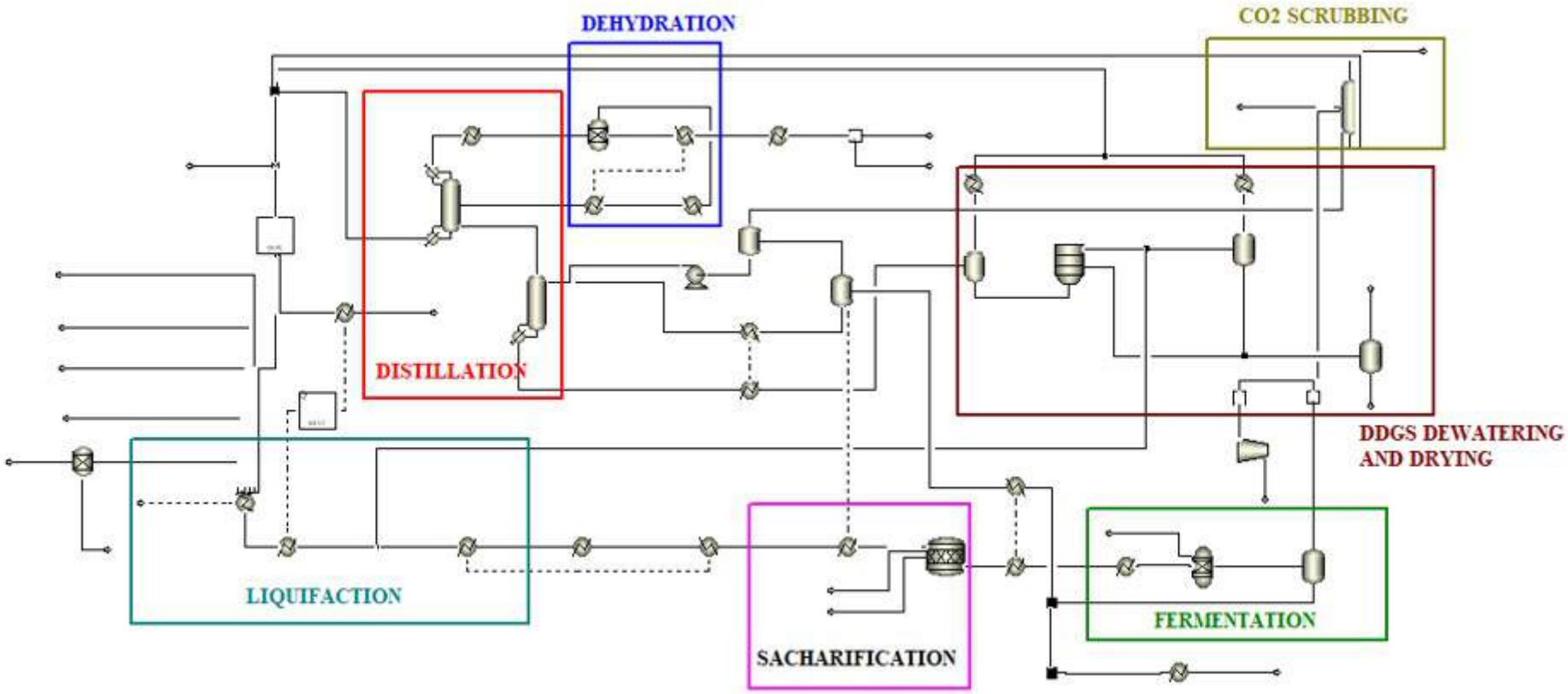


Figura 3.1: Hoja de Aspen Plus utilizada en el caso de estudio.

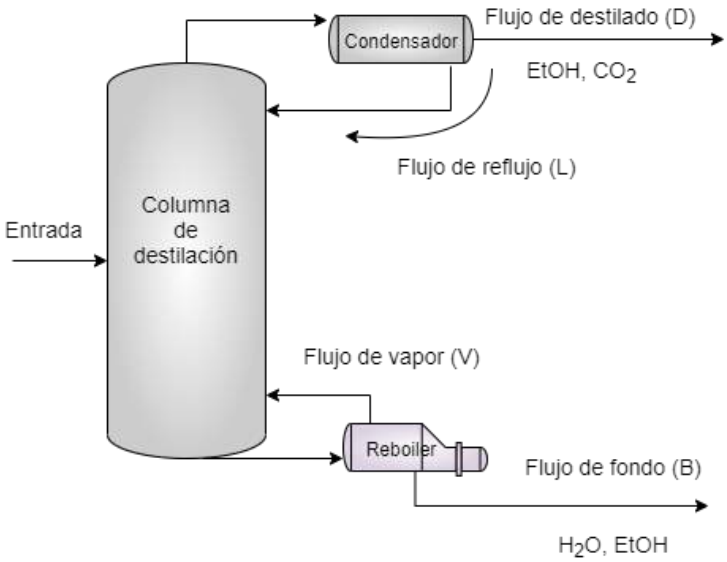


Figura 3.2: Esquema de la operación unitaria estudiada.

Esto permitió acelerar la generación de datos, limitar el consumo computacional a la ejecución de la simulación y no a la interfaz gráfica, y guardar los datos en las estructuras de datos que utiliza *Python*. En el Apéndice 1 se presenta una explicación de la implementación de la conexión de *Python* con *Aspen Plus* y de *Python* con *Aspen Hysys*.

En la Figura 3.2 se muestra un esquema de la columna de destilación considerada, así como las variables de entrada y salida utilizadas. Como variables de entrada se consideraron la relación entre reflujo y destilado (L/D como x_0), la relación entre vapor y líquido de fondo (V/B como x_1) y el número de platos (x_2), y como variables de salida se utilizaron la fracción másica de etanol en el destilado ($EtOH/D$ como y_1), el calor aportado en el *reboiler* (y_2) y el flujo en el destilado (D como y_3).

Las variables de entrada se utilizaron entre los siguientes valores: x_0 entre 1 y 3, x_1 entre 0.1 y 1, y x_2 (variable entera) entre 18 y 24. Las variables de salida y_2 e y_3 se dividieron por un valor referencia, por lo que las mismas se consideraron adimensionales.

En la Figura 3.3 se muestra el algoritmo utilizado. El mismo fue implementado en *Python* utilizando el paquete *gplearn* para las operaciones de PG. Se utilizó un diseño de experimentos adaptativo, en el cual se fueron agregando nuevos datos si luego de N^{max} iteraciones en la ejecución de la PG, el valor de *fitness* del mejor individuo de la población continuaba siendo mayor a la

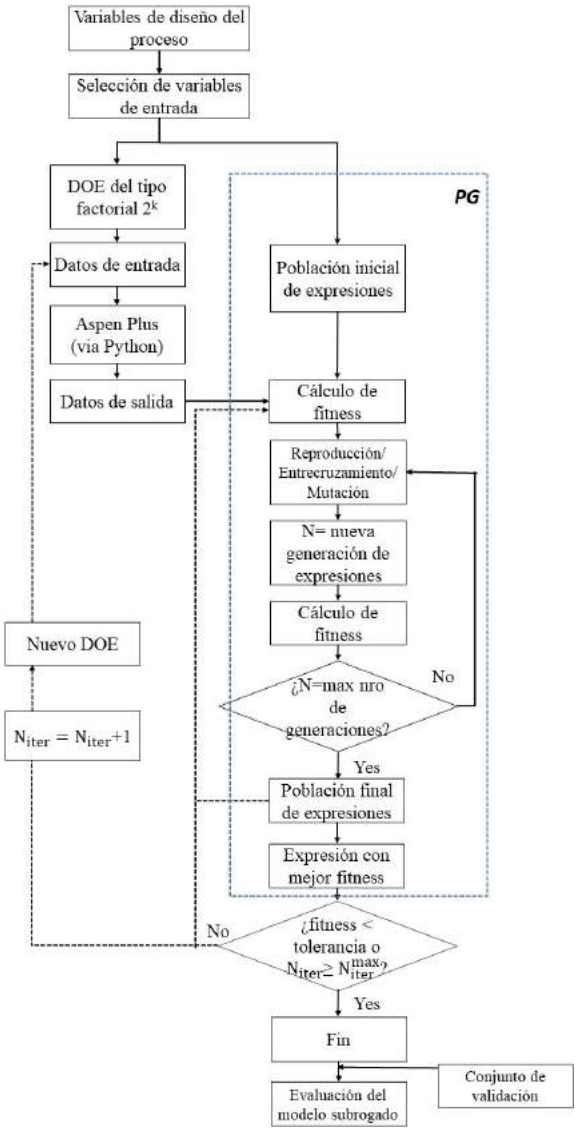


Figura 3.3: Esquema del algoritmo utilizado en el presente caso.

tolerancia. En caso contrario, la búsqueda del modelo termina y el modelo obtenido corresponde al individuo con menor valor de *fitness*.

En este caso se utilizó como función de *fitness* $RMSE + PC * TN$, donde RMSE es el error cuadrático medio sobre el conjunto de entrenamiento, PC es un coeficiente de penalización en el tamaño de nodos del árbol y TN es la cantidad de nodos en el árbol.

Se comenzó con un diseño factorial de 2^3 en las variables de entrada donde x_0, x_1 y x_2 tomaron los valores de los límites del intervalo de trabajo de cada variable de entrada. Con cada nueva generación de puntos se extendieron la cantidad de puntos generados en una sola variable. En cada nuevo DoE se

Tabla 3.1: Cantidad de puntos en cada iteración (N_{iter}).

Iteración	x_0	x_1	x_2
1	2	2	2
2	2	2	3
3	4	2	3
4	4	4	3
5	8	4	3
6	8	8	3
7	16	8	3
8	16	16	3
9	32	16	3
10	32	32	3

Tabla 3.2: Parámetros del algoritmo PG utilizado.

Parámetros	
Generaciones	50
Población	1000
Probabilidad de entrecruzamiento	0.9
Probabilidad de mutación	0.03
Probabilidad de reproducción	0.07
Altura de árbol inicial	$2 \leq d \leq 6$
Método de creación de población inicial	*half-and-half*
Cantidad de árboles en torneo	3
PC	0.0001
N_{iter}^{max}	10

generaron puntos equidistantes en el rango de trabajo de la variable. En la Tabla 3.1 se presenta la cantidad de datos en cada iteración.

Para la ejecución de algoritmo de PG se utilizó como conjunto de funciones las operaciones simples de suma, resta, multiplicación, división y negación, y en algunos casos la función exponencial. En la Tabla 3.2 se presentan los parámetros utilizados. Los casos estudiados con el DoE adaptativo se denominaron A (A_1 con operaciones simples y A_2 con operaciones simples y exponencial), y B fueron los casos base (B_1 con operaciones simples y B_2 con operaciones simples y exponencial), de forma de realizar la comparación de enfoques: diseño experimental fijo con diseño experimental adaptativo. Los casos B se ejecutaron con 8 puntos en el conjunto de entrenamiento y 50 generaciones; con 12 puntos y 100 generaciones; 24 puntos y 150 generaciones, y así, hasta el conjunto de 3072 puntos y 500 iteraciones.

3.3. Resultados

En las Figuras 3.4, 3.5, y 3.6 se muestra la progresión de los valores de RMSE en el conjunto de validación para cada una de las variables de salida, para las cuatro configuraciones estudiadas. El número de valores de puntos de entrenamiento está relacionado con la cantidad de iteraciones y de datos, esto es, en la iteración 1 se utilizan 50 generaciones y se tienen 8 datos de entrenamiento, en la iteración 2 se utilizan otras 50 generaciones y 12 datos, y así con el resto de las iteraciones.

Para las tres variables de salida, se cumple que, en general, los valores de RMSE diminuyen a medida que se tienen mayor cantidad de datos. También se aprecia que a partir de cierta cantidad de datos la cantidad no es relevante a la elección de las condiciones del método. Se puede observar que para algunos casos los valores de RMSE en una configuración son mucho más grandes que para el resto de configuraciones.

La Tabla 3.3 muestra los valores RMSE sobre el conjunto de validación con 3072 puntos calculados a partir de los modelos obtenidos por las distintas configuraciones del algoritmo. A partir de estos resultados, se puede inferir que en todos los casos los modelos obtenidos producen una buena estimación de las variables de salida.

En la Tabla 3.4 se puede observar que para la mayoría de las variables de

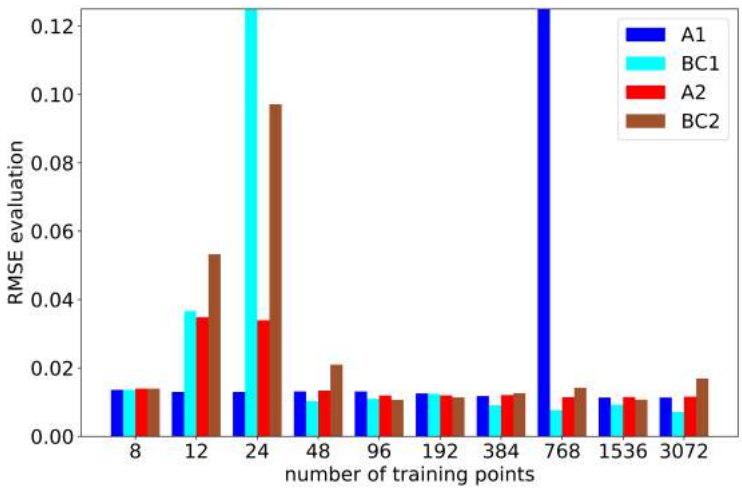


Figura 3.4: RMSE calculado para el conjunto de validación para la fracción másica de etanol en el destilado.

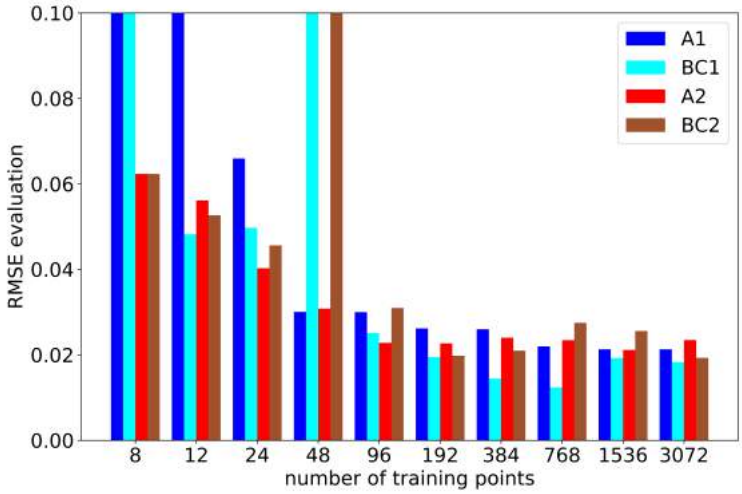


Figura 3.5: RMSE calculado para el conjunto de validación para el flujo en el destilado.

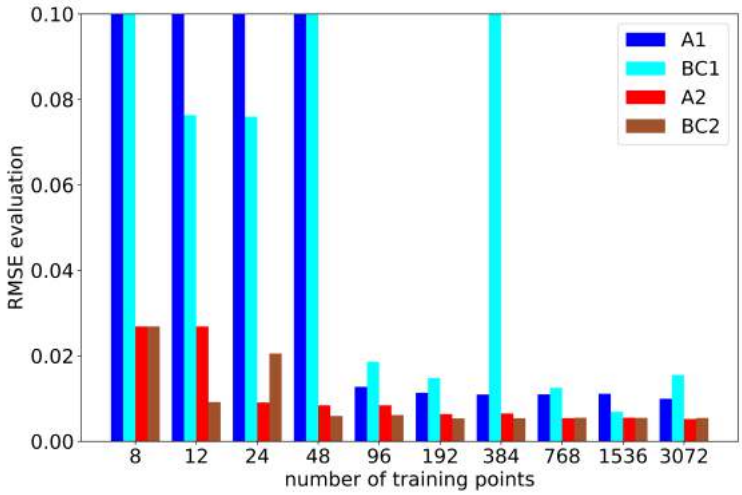


Figura 3.6: RMSE calculado para el conjunto de validación para calor en el reboiler.

Tabla 3.3: RMSE de los modelos obtenidos con 3072 puntos sobre el conjunto de validación para cada condición.

Caso	RMSE y_1	RMSE y_2	RMSE y_3
A1	0.0113	0.0225	0.0105
A2	0.0077	0.0239	0.0053
BC1	0.0076	0.0193	0.0164
BC2	0.0183	0.0197	0.0060

Tabla 3.4: Tiempos de ejecución de entrenamiento (min) en cada caso.

Caso	tiempo y_1	tiempo y_2	tiempo y_3
A1	9.85	8.85	11.27
A2	9.97	11.26	11.22
BC1	9.57	23.33	19.09
BC2	18.13	20.51	18.37

Tabla 3.5: Mejores modelos para cada variable de salida.

y_1	$0.872 - \dfrac{(1 - x_0 + x_1)}{x_0(-x_0(x_0x_1 - x_0 + 0.872 - \frac{(1-x_0+x_1)}{x_0(2x_0-4x_1+5.105-\frac{(x_0x_1-x_0+x_1)}{x_2})}) + x_0 + 5.105)}$
y_2	$\dfrac{x_1x_2(x_0 + x_2)}{(-2x_1 - x_2 + (x_0 - x_1)^2e^{-x_1})(-x_0 + x_1 - x_2 + \frac{(x_0-x_1)^2}{-x_1+e^{x_1}} - 3.467)}$
y_3	$e^{\begin{pmatrix} e^{(x_0-x_1)}e^{(x_1-x_0)}e^{-e^{e^{-e^{-e^{e^{e^{-e^{x_1}}}}}}}} & \\ & e^{-e^{-x_0}}e^{-e^{(x_0-x_1)}} \end{pmatrix}}$

salida, los casos etiquetados como alternativos se ejecutaron en menos tiempo que los casos base, llegando en algunos casos a ser menos de la mitad del tiempo. Cabe destacar, que para el análisis considerado, no se tiene en cuenta el tiempo de toma de datos desde las simulaciones de Aspen Plus.

En la Tabla 3.5 se presentan los modelos que proporcionaron un mejor ajuste para cada variable de salida. Se puede observar que con el algoritmo utilizado las expresiones encontradas son relativamente complejas. Por ejemplo, en la variable de salida y_3 se tiene una combinación de hasta 7 funciones exponenciales. Este es el fenómeno que en la literatura se conoce como *bloat*.

Capítulo 4

Estudio comparativo entre Programación Genética y Programación Kaizen para datos de simuladores

Como se comentó en el Capítulo 2, la programación Kaizen se propuso como una mejora del algoritmo de PG para resolver problemas de regresión simbólica. En el presente capítulo se presenta un estudio comparativo entre PG y la PK para resolver problemas de RS en sistemas de procesos de pocas variables a partir de simulaciones determinísticas.

Para la comparación de los métodos se utilizan funciones de referencia propuestas por Surjanovic y Bingham (2013) para la comparación de algoritmos, y también, datos provenientes de la simulación del equipo de destilación utilizado en el Capítulo 3.

4.1. Metodología

A continuación se presentan la metodología utilizada para el caso de funciones de *benchmark* y la columna de destilación con sus correspondientes conjuntos de entrenamiento y validación, la implementación de los algoritmos y los parámetros utilizados.

4.1.1. Casos de estudio seleccionados para la comparación

4.1.1.1. Funciones de *benchmark*

Las funciones de *benchmark* consideradas en esta evaluación fueron las propuestas por Surjanovic y Bingham (2013). Estas funciones están clasificadas en cinco categorías, según su forma: muchos mínimos locales (*Many Local Minima*, MLM), forma de taza (*Bowl-shaped*, BS), forma de plato (*Plate-shaped*, PS), forma de valle (*Valley-shaped*, VS), y crestas empinadas/caídas (*Steep Ridges/Drops*, SR/D). Para el presente caso de estudio se eligieron tres funciones de cada forma, como se muestra en la Tabla 4.1.

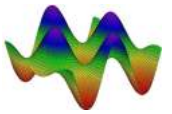
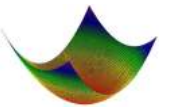
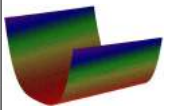
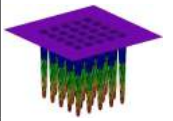
En el trabajo de Davis et al. (2018), se concluye que para este tipo de problemas, a partir de 1600 puntos en el conjunto de entrenamiento, los métodos de muestreo utilizados (hipercubo latino, Sobol y secuencias de Hobol) no afectan el resultado de la búsqueda de modelos subrogados. Para el presente caso se utilizaron conjuntos de entrenamiento de 1800 puntos y conjuntos de validación de 600 puntos utilizando el método de hipercubo latino como método de diseño experimental.

4.1.1.2. Columna de destilación del Capítulo 3

Para estudiar los algoritmos a un caso de procesos, se utilizó el caso ya discutido en el Capítulo 3.

Como variables de entrada y salida fueron utilizados las ya discutidas, pero utilizando un diseño experimental diferente. Al contrario del caso anterior, se consideró y_1 como el flujo de salida de etanol en vez de la fracción másica. Para este caso se utilizaron 32 puntos igualmente espaciados para las variables de entrada continuas (x_0 y x_1) y 3 niveles para la variable de entrada discreta (x_2); y para el conjunto de validación se usaron 16 puntos igualmente distribuidos para las variables de entrada continuas y 3 niveles para x_2. Los valores máximos de las variables de salida son 42785 kg/h, 17010 cal/s y 59825 kg/h, respectivamente para y_1, y_2 y y_3.

Tabla 4.1: Funciones de referencia consideradas

Forma	Nombre	Expresión	Rango en vars. de entrada		Rango en var. de salida	
MLM	Griewank (f_1)	$\sum_{i=0}^{1} \frac{x_i^2}{4000} - \prod_{i=0}^{1} \cos\left(\frac{x_i}{\sqrt{i+1}}\right) + 1$	x_0=[-5,5]	x_1=[-5,5]	$[0, 2]$	
	Bukin N.6 (f_2)	$100\sqrt{\lvert x_1 - 0.01x_0^2\rvert} + 0.01\lvert x_0 + 10\rvert$	x_0=[-15,-5]	x_1=[-3,3]	$[0, 230]$	
	Cross-in-Tray (f_3)	$-10^{-4}\left(\left\lvert \sin(x_0)\sin(x_1)e^{\left\lvert 100 - \frac{\sqrt{x_0^2+x_1^2}}{\pi}\right\rvert}\right\rvert + 1\right)^{0.1}$	x_0=[-10,10]	x_1=[-10,10]	$[-2, 0]$	
BS	Rotated Hyper-Ellipsoid (f_4)	$2x_0^2 + 2x_1^2$	x_0=[-65,65]	x_1=[-65,65]	$[0, 12675]$	
	Sum of Different Powers (f_5)	$\lvert x_0\rvert^2 + \lvert x_1\rvert^3$	x_0=[-1,1]	x_1=[-1,1]	$[0, 2]$	
	Sphere (f_6)	$x_0^2 + x_1^2$	x_0=[-5,5]	x1=[-5,5]	$[0, 50]$	
PS	Booth (f_7)	$(x_0 + 2x_1 - 7)^2 + (2x_0 + x_0 - 5)^2$	x_0=[-10,10]	x_1=[-10,10]	$[74, 2600]$	
	Matyas (f_8)	$0.26(x_0^2 + x_1^2) - 0.48x_0x_1$	x_0=[-10,10]	x_1=[-10,10]	$[0, 100]$	
	McCormick (f_9)	$\sin(x_0 + x_1) + (x_0 - x_1)^2 - 1,5x_0 + 2.5x_1 + 1$	x_0=[-1.5,4]	x_1=[-3,4]	$[-2, 45]$	
VS	Three-Hump Camel (f_{10})	$2x_0^2 - 1.05x_0^4 + 1/6x_0^6 + x_0x_1 + x_1^2$	x_0=[-5,5]	x_1=[-5,5]	$[0, 6375]$	
	Dixon-Price (f_{11})	$(x_0 - 1)^2 + 2(2x_1^2 - x_0)^2$	x_0=[-10,10]	x_1=[-10,10]	$[1, 88325]$	
	Rosenbrock (f_{12})	$100(x_1 - x_0^2)^2 + (x_0 - 1)^2$	x_0=[-5,10]	x_1=[-5,10]	$[1, 1102581]$	
SR/D	De Jong N.5 (f_{13})	$\left(0.002 + \sum_{i=1}^{2} 5\frac{1}{i + (x_0 - a_{1i})^6 + (x_1 - a_{2i})^6}\right)$	x_0=[-65,65]	x_1=[-65,65]	$[12, 500]$	
	Easom (f_{14})	$-\cos(x_0)\cos(x_1)e^{\left(-(x_0-\pi)^2-(x_1-\pi)^2\right)}$	x_0=[-5,5]	x_1=[-5,5]	$[0, 1]$	
	Michalewicz (f_{15})	$-\sin(x_0)\sin^{20}\left(x_0^2/\pi\right) - \sin(x_1)\sin^{20}\left(2x_1^2/\pi\right)$	x_0=[-5,5]	x_1=[-5,5]	$[-1, 1]$	

con $a = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & \dots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \dots & 32 & 32 & 32 \end{pmatrix}$

Tabla 4.2: Parámetros de los algoritmos utilizados

Parámetros	PK	GP
Tamaño de población	10	500
Generaciones	2000	50
Operadores matemáticos	$+, -, *, /, neg, cos, sin, exp, log, \sqrt{}, abs$	$+, -, *, /, neg, cos, sin, exp, log, \sqrt{}, abs$
Altura inicial	$0 \leq d \leq 3$	$2 \leq d \leq 6$
Altura máxima	10	17
Probab. de cruzamiento	1	0.85
Probab. de mutación	1	0.03
α	0.05	N/A
θ	10^{-6}	N/A
Reinicio	25 % de generaciones	N/A

4.1.2. Implementación y parámetros utilizados

El algoritmo de PG fue instanciado e implementado en `Python` utilizando el paquete `deap` (Fortin et al. 2012), mientras que para la PK se utilizó la implementación descripta en la Sección 2.2. Ambos algoritmos utilizan `cxOnePoint` como método de cruzamiento, y `mutEphemeral` y `mutUniform` como métodos de mutación (0.90 de probabilidad para la mutación uniforme y 0.10 de probabilidad para la mutación de la constante). En la Tabla 4.2 se presentan los parámetros utilizados para la ejecución de ambos algoritmos. Los mismos se basaron en el trabajo de de Melo y Banzhaf (2018).

La plataforma para las ejecuciones fue una computadora con Windows 10, un procesador Quad Core Intel i7 7500U de 2.90 GHz. y 12 GB de memoria RAM.

Debido a la naturaleza aleatoria de ambos algoritmos, para comparar los resultados de ambos se aplicaron pruebas estadísticas. En Derrac et al. (2011) y Pedemonte et al. (2018) se presenta un procedimiento para determinar si la distribución de errores para cada algoritmo y cada problema es estadísticamente diferente. En el trabajo de Alba y Luque (2005) se encuentra una breve explicación sobre la elección de la prueba de hipótesis según si los puntos en los conjuntos de entrenamiento utilizados son independientes en cada caso o no. Debido al extensivo uso de los *test* estadísticos a lo largo de la tesis, en el Apéndice 2 se presenta una breve introducción a cada uno de ellas.

Para aplicar dicho procedimiento, en el caso de las funciones de referencia, se realizaron 30 ejecuciones diferentes para cada algoritmo y cada función

(utilizando un conjunto de entrenamiento diferente cada vez), obteniéndose 30 modelos. Luego, se realizó el cálculo del RMSE sobre el conjunto de validación para cada uno de los modelos, y por último fue aplicada la prueba de Wilcoxon-Mann-Whitney con un nivel de significancia de 95 % en el caso de las funciones *benchmark*, y la prueba de los rangos con signo de Wilcoxon con el mismo nivel de significancia en el caso de la columna de destilación.

4.2. Resultados

4.2.1. Funciones *Benchmark*

La Tabla 4.3 presenta los resultados obtenidos para cada algoritmo. Se muestra, en todos los casos, los valores mínimo, el primer cuartil (Q1), la mediana, el tercer cuartil (Q3) y el máximo de la distribución de los valores de RMSE para cada función. Además, en las últimas dos columnas se incluye el valor estadístico obtenido con la prueba estadística U de Mann-Whitney (p−valor) y un símbolo que indica cual distribución de errores es menor, por lo tanto cual algoritmo genera mejores modelos. Se indica con '$\triangleleft$' cuando PK es mejor, con '$\triangledown$' cuando PG es mejor, y con '$-$' cuando no hay diferencia estadística entre ambos algoritmos.

Los resultados obtenidos muestran que PK es el algoritmo con mejor desempeño. PK es mejor que PG en 13 de las 15 funciones, mientras que PG es mejor solamente en un caso. En la función f_4, los resultados numéricos son apenas favorable para PK, a pesar de que dicha diferencia no tiene significancia estadística. Además, comparando los valores mínimos y máximos, PK tiene un mejor desempeño para 14 de las 15 funciones, y PK presenta menor variabilidad en la distribución de RMSE. Se debe hacer notar que para las funciones f_1 y f_5 algunas ejecuciones de PG generaron expresiones que producen NaNs (Not a Number). Esto se debe a que se generan modelos que en la evaluación dan infinito en algunos puntos.

Tabla 4.3: Resultados numéricos de cada algoritmo en el caso de funciones de referencia.

	Programación Kaizen					Programción Genética					Estadístico	
	Mín.	Q_1	Mediana	Q_3	Máx.	Mín.	Q_1	Mediana	Q_3	Máx.	Simb.	p-valor
f_1	2.26e-15	1.08e-8	**4.48e-5**	1.10e-1	2.42e-1	2.64e-1	3.11e-1	3.25e-1	3.31e-1	4.17e-1	$\lhd$	1.51e-11
f_2	1.06e1	1.17e1	**1.28e1**	1.35e1	1.84e1	3.29e1	4.30e1	4.80e1	5.16e1	5.91e1	$\lhd$	1.51e-11
f_3	1.84e-2	3.71e-2	**6.09e-2**	9.69e-2	1.41e-1	1.49e-1	1.98e-1	2.18e-1	2.39e-1	2.49e-1	$\lhd$	1.51e-11
f_4	0	0	**8.80e-14**	1.09e-11	1.26e-9	0	0	**0**	9.55e2	2.40e3	$-$	3.80e-1
f_5	5.55e-7	1.23e-4	**2.98e-4**	4.62e-4	1.44e-3	1.54e-1	2.03e-1	2.83e-1	2.97e-1	3.00e-1	$\lhd$	1.51e-11
f_6	0	2.9e-16	1.13e-14	4.07e-4	1.42e-1	0	0	**1.45e-16**	2.43e-16	5.29e0	$\rhd$	1.14e-5
f_7	9.82e-14	5.20e-13	**1.91e-12**	2.87e-11	4.10e1	1.16e2	2.01e2	2.17e2	2.38e2	2.84e2	$\lhd$	1.51e-11
f_8	2.35e-15	2.46e-15	**2.77e-15**	4.31e-14	2.27e0	1.03e0	4.46e0	5.09e0	7.30e0	1.61e1	$\lhd$	1.67e-11
f_9	2.15e-14	6.16e-2	**3.61e-1**	4.41e-1	6.58e-1	7.03e-1	9.27e-1	1.29e0	1.73e0	2.52e0	$\lhd$	1.51e-11
f_{10}	3.68e-2	1.66e0	**7.82e0**	9.30e0	1.67e1	1.49e1	2.37e1	2.70e1	4.38e1	8.48e1	$\lhd$	1.84e-11
f_{11}	6.79e-12	4.47e2	**1.13e3**	1.40e3	3.04e3	4.97e3	5.94e3	8.47e3	9.68e3	1.20e4	$\lhd$	1.51e-11
f_{12}	1.08e-12	2.79e-12	**4.78e-12**	5.77e-12	8.72e-12	2.08e2	9.78e2	1.02e3	1.18e3	1.98e3	$\lhd$	1.51e-11
f_{13}	7.16e1	7.84e1	**8.25e1**	8.52e1	2.42e2	8.77e1	1.65e2	1.74e2	1.80e2	2.38e2	$\lhd$	3.69e-10
f_{14}	2.68e-2	7.00e-2	**7.42e-2**	8.20e-2	1.15e-1	7.25e-2	8.43e-2	9.03e-2	9.89e-2	1.31e-1	$\lhd$	1.66e-6
f_{15}	2.50e-1	2.88e-1	**2.98e-1**	3.06e-1	3.10e-1	3.12e-1	3.26e-1	3.35e-1	3.49e-1	3.70e-1	$\lhd$	3.16e-10

La mediana de los mejores resultados están en **negrita**.

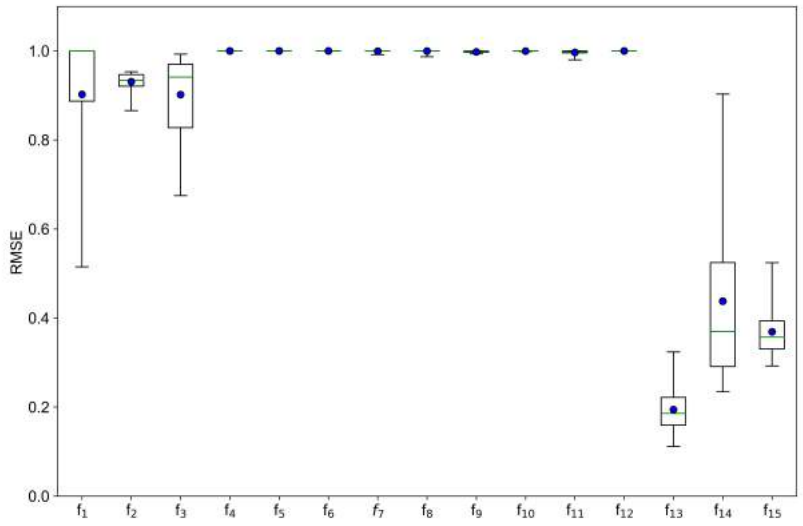


Figura 4.1: R^2 Ajustado promedio para cada función de referencia

La Figura 4.1 muestra la distribución de los valores de R^2 ajustado de PK para cada función considerada como un gráfico de cajas. Las funciones con forma de mínimos múltiples presentan altos valores de R^2 ajustado, variando entre 0.902 y 1.000; mientras que las funciones con forma crestas empinadas/caídas tienen valores por debajo de 0.5. Además, se puede observar que solamente en las formas con una superficie con grandes cambios de pendientes, como las funciones pertenecientes a las familias de muchos mínimos locales y la de crestas empinadas/caídas se presentan grandes variaciones de R^2 y RMSE en los modelos obtenidos.

4.2.2. Caso: Columna de destilación

La Tabla 4.4 muestra los resultados de PK y PG para cada modelo buscado. En la misma se incluyen las métricas para cada distribución y los valores de las pruebas estadísticas para cada variable de salida (en una nomenclatura similar que en la tabla de resultados anterior).

Los resultados muestran que PK también presenta mejor desempeño que PG para las tres variables de salida consideradas. Además, los valores de R^2 ajustado promedio obtenidos con PK para cada variable de salida son 0.923, 0.998 y 0.994, para y_1, y_2 y y_3, respectivamente. Esto indica que los modelos

Tabla 4.4: Resultados numéricos de cada algoritmo en el caso de aplicación industrial.

Var. de salida	Fracción másica, y_1		Energía, y_2		Flujo másico, y_3	
	KP	GP	KP	GP	KP	GP
Mín.	4.78e-01	2.03e3	2.00e4	3.51e5	1.14e2	2.07e3
Q_1	5.45e-01	7.57e3	4.04e4	9.14e5	2.03e2	8.16e3
Mediana	**5.60e-1**	9.94e3	**4.37e4**	1.06e6	**2.18e2**	1.13e4
Q_3	5.76e-01	1.28e4	5.79e4	1.22e6	2.63e2	1.41e4
Máx.	6.61e-01	1.98e4	7.12e4	2.06e6	2.79e2	1.80e4
Test estad.	Sig.	p−valor	Sig.	p−valor	Sig.	p−valor
	◁	1.734e-06	◁	1.734e-06	◁	1.734e-06

Los mejores resultados están en **negrita**.

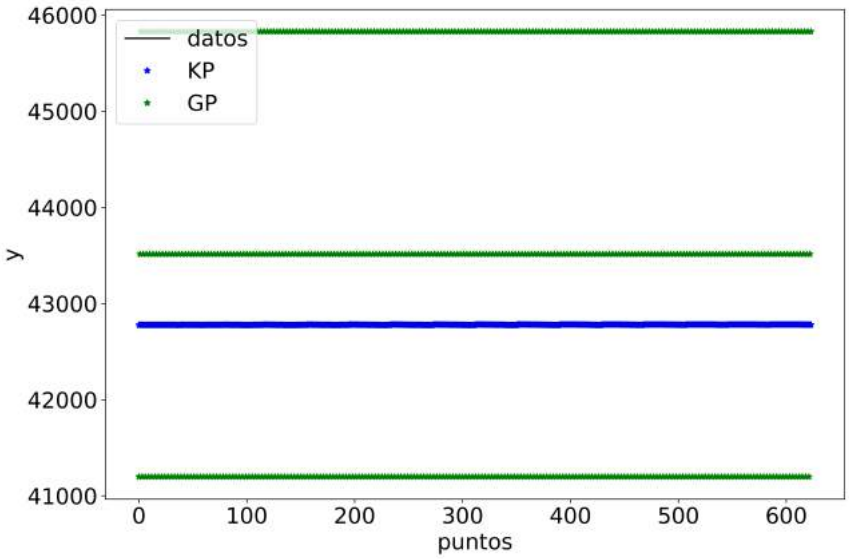


Figura 4.2: Valor de y_1 para cada punto dato, estimado PK y estimado PG.

obtenidos con PK ajustan de buena forma a los datos.

En las Figuras 4.2, 4.3 y 4.4 se muestran los valores de las variables de salida, dados como dato y los estimados por las mejores ecuaciones obtenidas por cada algoritmo. Las ecuaciones usadas son las que devolvieron un mejor RMSE para el conjunto de validación. A partir de las figuras, se puede apreciar que los valores estimados por PK son muchos más cercanos que aquellos estimados por PG para las tres variables. En particular, para la variable de salida y_1, la mejor estimación de PG no tiene un buen desempeño, sin embargo, el modelo obtenido por PK llega a valores muy cercanos. En la Figura 4.5 se muestra

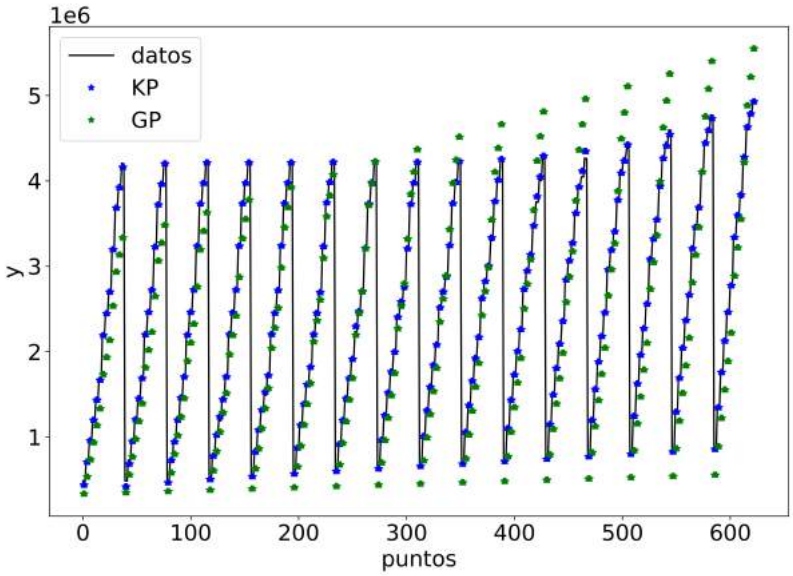


Figura 4.3: Valor de y_2 para cada punto dato, estimado PK y estimado PG.

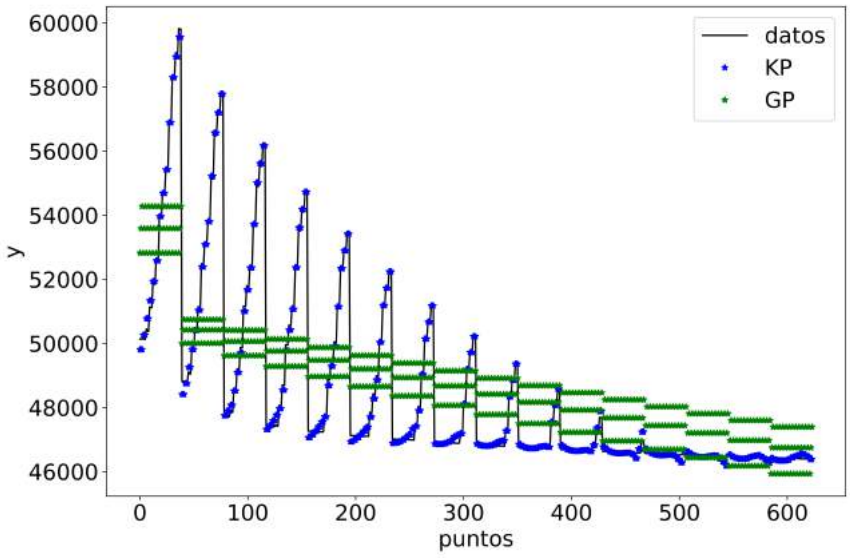


Figura 4.4: Valor de y_3 para cada punto dato, estimado PK y estimado PG.

la estimación para la primera variable de salida con los datos del conjunto de validación y el estimado por el mejor modelo obtenido por PK, de forma de apreciar mejor la similitud de los valores en la Figura 4.2.

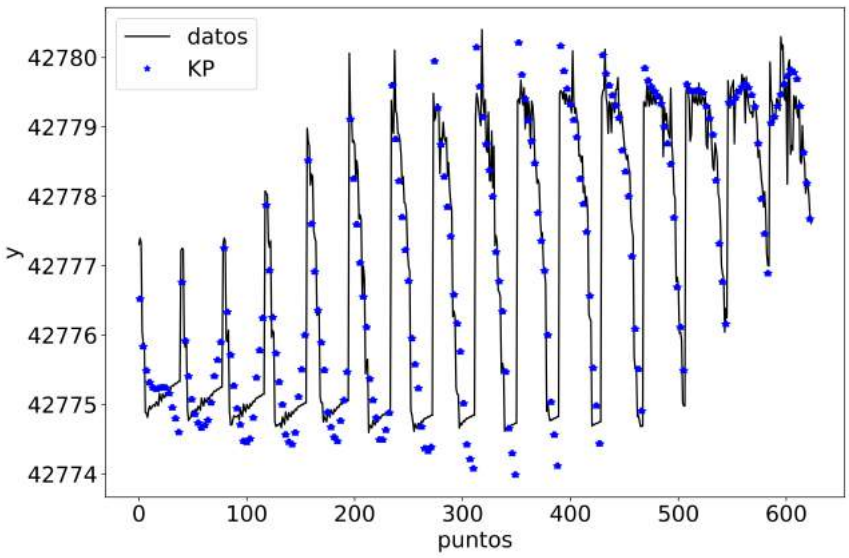


Figura 4.5: valor de y_1 para cada punto dato y estimado PK

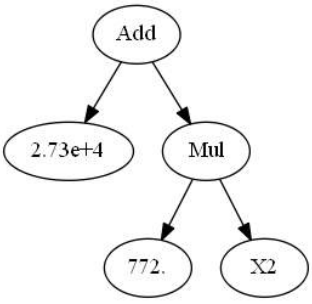


Figura 4.6: Árbol con el modelo obtenido por PG que devuelve menor RMSE en el conjunto de validación para y_1.

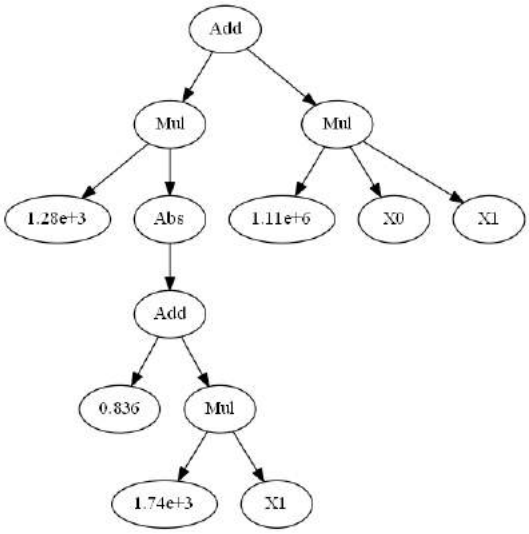


Figura 4.7: Árbol con el modelo obtenido por PG que devuelve menor RMSE en el conjunto de validación para y_2.

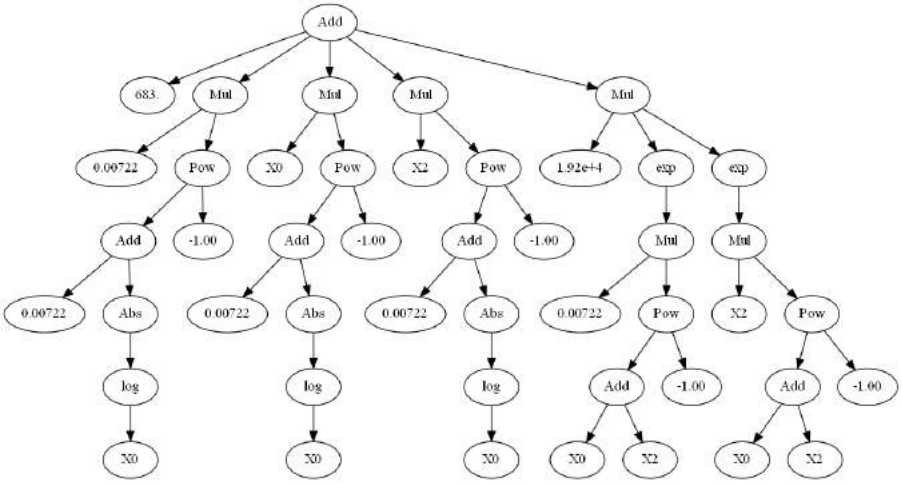


Figura 4.8: Árbol con el modelo obtenido por PG que devuelve menor RMSE en el conjunto de validación para y_3.

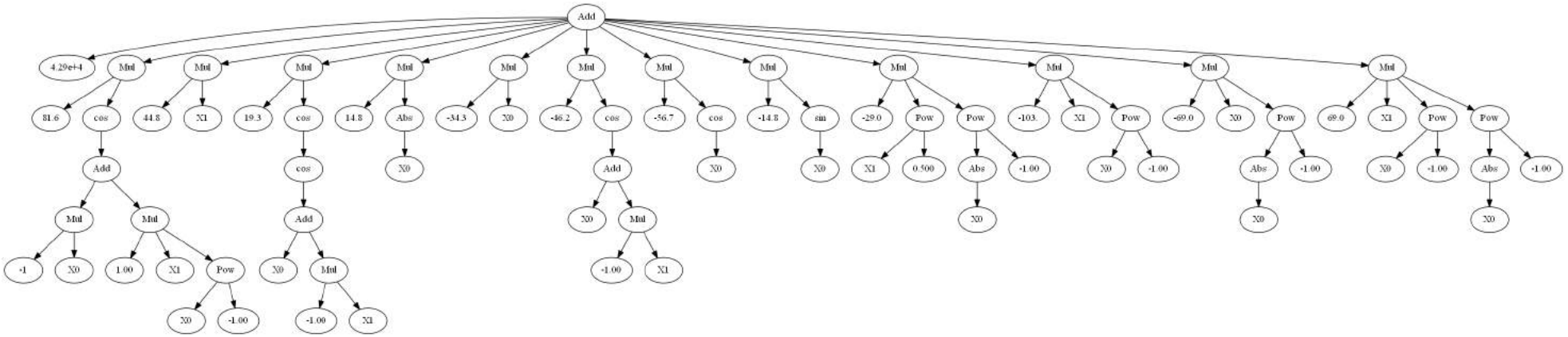


Figura 4.9: Árbol con el modelo obtenido por PK que devuelve menor RMSE en el conjunto de validación para y_1.

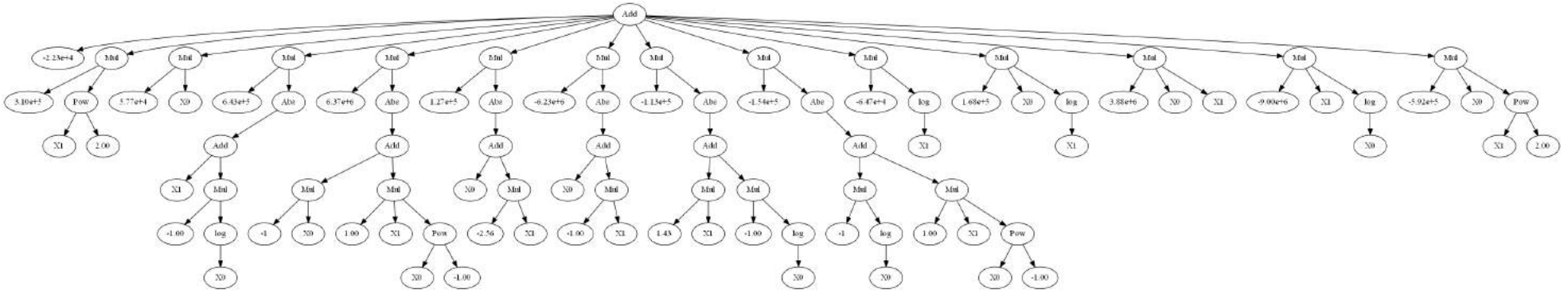


Figura 4.10: Árbol con el modelo obtenido por PK que devuelve menor RMSE en el conjunto de validación para y_2.

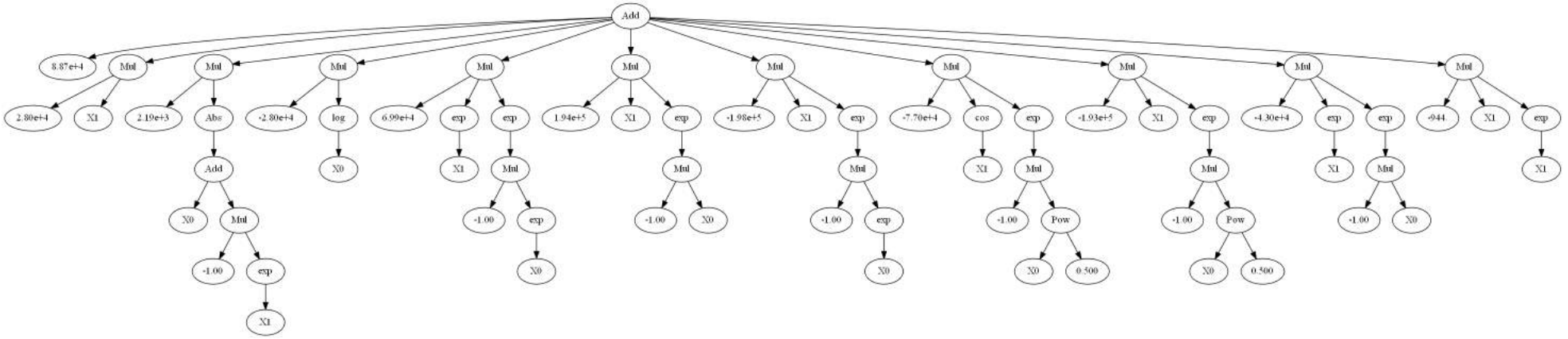


Figura 4.11: Árbol con el modelo obtenido por PK que devuelve menor RMSE en el conjunto de validación para y_3.

Por último, en las Figuras 4.6, 4.7 y 4.8 se presentan los árboles que representan los mejores modelos obtenidos por PG para cada variable de salida, respectivamente, mientras que las Figuras 4.9, 4.10 y 4.11 muestran los árboles obtenidos por PK. Como se puede observar, en los tres casos la expresión obtenida por la PG contiene menos ramificaciones y nodos que la PK, pero con la desventaja de una peor estimación de los valores del conjunto de validación.

Capítulo 5

Generación de *soft-sensor* para una columna de separación en la refinería de ANCAP

Luego de la evaluación comparativa de los algoritmos PG y PK, se definió que se usaría la PK como técnica de aprendizaje automático para obtener modelos de sistemas de procesos. En esta tesis, la aplicación central es la búsqueda de modelos subrogados a partir de datos reales provenientes de uno de los procesos de destilación en la planta de la Teja de ANCAP.

Para este caso, además del uso de la técnica PK, se evaluó el trabajo con ensamblado de modelos como predicción (o modelo) del proceso, de forma de obtener una mayor precisión y reducir la incertidumbre de las predicciones. En este capítulo, en primera instancia se hará una breve introducción a la temática de*soft-sensors* en el contexto de su aplicación a la Ingeniería Química, luego se presenta el caso de estudio que consiste en la estimación de la concentración de los hidrocarburos de cuatro carbonos en la corriente del destilado en una columna de destilación. Posteriormente, se detalla la metodología aplicada que incluye la limpieza de los datos (dado que son datos reales provenientes de sensores y medidas experimentales), creación de los conjuntos de entrenamiento y validación y metodología seguida para comparar los modelos obtenidos. Finalmente, se presentan los resultados obtenidos, incluyendo los modelos generados y una evaluación sobre la degradación de los modelos con el paso del tiempo.

5.1. Aplicaciones de *soft-sensors* en ingeniería química

Un uso extendido de las técnicas de aprendizaje automático en la Ingeniería Química, es en la construcción de *soft-sensors*. Los *soft-sensors* son aplicaciones que actúan en un proceso en sustitución de sensores reales, cuando las variables de interés no son posibles de medir o la toma de medidas es muy costosa para la frecuencia requerida. Esto último sucede, por ejemplo, con las concentraciones de especies químicas donde las medidas se deben hacer en un laboratorio.

En general, los *soft-sensors* son desarrollados con modelos que consideran los fenómenos físicos y químicos que suceden en el proceso. Debido a que se deben realizar simplificaciones de los modelos para poder ser implementados, una alternativa es el uso de modelos generados a partir de datos históricos.

A pesar de las similitudes entre la construcción de modelos subrogados y de *soft-sensors* a partir de datos, la literatura en estas dos temáticas está desconectada, por lo que han sido dos áreas de investigación que han desarrollado distintas técnicas para obtener modelos. Para la realización de modelos en *soft-sensors* es común la utilización de técnicas como Análisis de componentes principales (*Principal Component Analysis*, PCA), Regresión de mínimos cuadrados parciales (*Partial Least Squares*, PLS), así como SVR y ANN o AP, que ya se han nombrado anteriormente.

Frank y Friedman (1993) realizaron una revisión de las técnicas aplicadas a problemas de Ingeniería Química y el trabajo de Kadlec et al. (2009) presenta una lista de los principales trabajos realizados en el área de *soft-sensors* a partir de datos. Contribuciones más recientes son las de Kaneko y Funatsu (2014), que presentaron la aplicación de SVR con sistemas de aprendizaje ensamblados para adaptar cambios no lineales y variantes en el tiempo, a los procesos de producción de alquil aluminio y de desnitrificación de gases de escape; Zhang et al. (2013), que utilizaron PCA para la reducción de dimensiones de los datos y mínimos cuadrados con SVR para construir un *soft-sensor* para la predicción del tamaño medio de partícula en el proceso de síntesis de oxalato de cobalto; Shang et al. (2014), que trabajaron con una técnica de AP para el desarrollo de un *soft-sensors* aplicado a un caso de la industria petrolera; y Nielsen et al. (2020), en donde utilizaron un modelo híbrido para predecir la cinética de partícula en combinación con un modelo riguroso de balance poblacional.

5.2. Presentación del caso de estudio

La columna de destilación considerada separa principalmente hidrocarburos de 3 carbonos (C3s, destilado) de los hidrocarburos de 4 carbonos (C4s, residuo). La Figura 5.1 muestra un esquema del proceso en donde la corriente de entrada proviene de una columna de destilación previa. Además de los hidrocarburos de 3 y 4 carbonos, la corriente puede contener hidrocarburos de 1, 2, 5 y 6 carbonos, así como H_2 (Hidrógeno) y H_2S (Ácido sulfhídrico).

En la planta, la calidad de las corrientes de salidas es monitoreada para que la cantidad de C4s en el destilado y la cantidad de C3s en el residuo estén por debajo de los límites permitidos en la operación normal de la columna. La medida de la calidad es realizada mediante medidas experimentales en laboratorio de 19 posibles compuestos en las corrientes, en donde las tomas se realizan diariamente o cada dos días. Por otro lado, el proceso es monitoreado a través de varios sensores, como se muestra en la Figura 5.1, donde los datos son adquiridos cada 30 segundos.

El objetivo del caso de estudio y la necesidad de las personas que monitorean el proceso es poder realizar ajustes en el proceso, a partir de un modelo

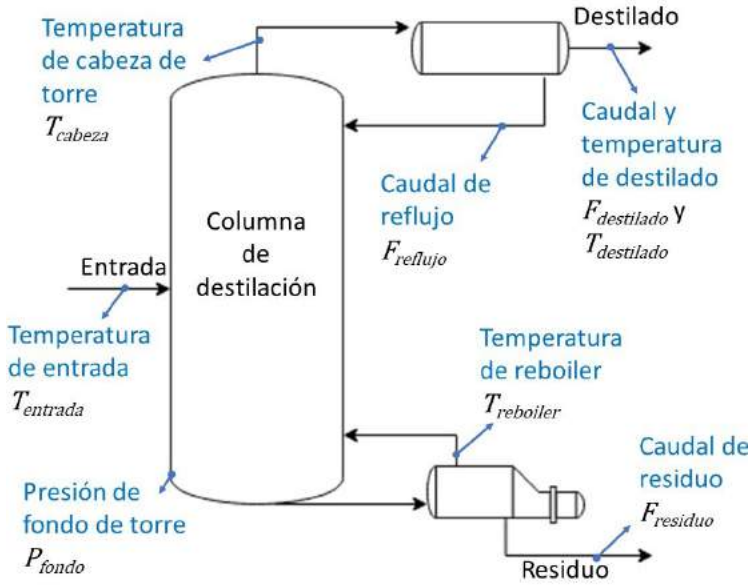


Figura 5.1: Esquema de la columna de destilación considerada. En azul se incluye los sensores en tiempo real: temperatura en: la entrada ($T_{entrada}$), el destilado ($T_{destilado}$), la cabeza de columna (T_{cabeza}) y el reboiler ($T_{reboiler}$); la presión en el fondo de torre (P_{fondo}); y el caudal en: el destilado ($F_{destilado}$), el residuo ($F_{residuo}$) y el reflujo ($F_{reflujo}$).

predictivo y los datos adquiridos en tiempo real, de forma de detectar cuando la concentración de C4s en el destilado está desviada de la especificación, sin necesidad de esperar las medidas experimentales.

Para guardar la confidencialidad de los datos de operación, todos los valores presentados están normalizados a partir de un valor de referencia.

5.3. Metodología

Para la realización del modelo de predicción fueron recibidos 861086 medidas de los 8 sensores mostrados en la Figura 5.1, 282 medidas de laboratorios conteniendo las medidas de las concentraciones de los 19 compuestos posibles en el destilado, y 351 medidas de las concentraciones en el residuo.

Para el modelo fueron considerados 8 variables de entrada, que se corresponden con las medidas realizadas por los sensores: $X = [T_{entrada}, F_{reflujo}, F_{residuo}, F_{destilado}, T_{reboiler}, T_{cabeza}, T_{destilado}, P_{fondo}]$, y una única variable de salida: Y, definida como la suma de las fracciones volumétricas de todos los hidrocarburos de 4 carbonos en el destilado.

$$Y = \sum_i C4_i, \tag{5.1}$$

donde i corresponde con: n-butano, 1-buteno, isobutano, isobuteno, cis-2-buteno, trans-2-buteno and 1,3-butadieno.

El proceso trabaja en dos estados estacionarios, en donde cada uno tiene su correspondiente valor de composición de hidrocarburos de 4 carbonos en el destilado máximo de operación. Se referirá a estos estados como estado estacionario (EE) usual, y EE esporádico. El EE usual es el que permite menor concentración de C4s en el destilado.

Como primer paso para la obtención de los modelos a partir de datos se debe realizar una limpieza de estos, ya que para la utilización de técnicas de aprendizaje automático se debe utilizar conjunto de datos completos, esto es, se deben tener valores de las variables de entrada correspondiente para cada valor de variable de salida. Para la limpieza de los datos, en primera instancia se detectaron los valores en que los sensores no estaban funcionando o desconectados. Luego, utilizando la regla de los 3σ se detectaron y eliminaron los *outliers* de cada variable de entrada y de la variable de salida. De este modo, se detectaron 3177 puntos para X_1, 13613 puntos para X_4, 54685 puntos para

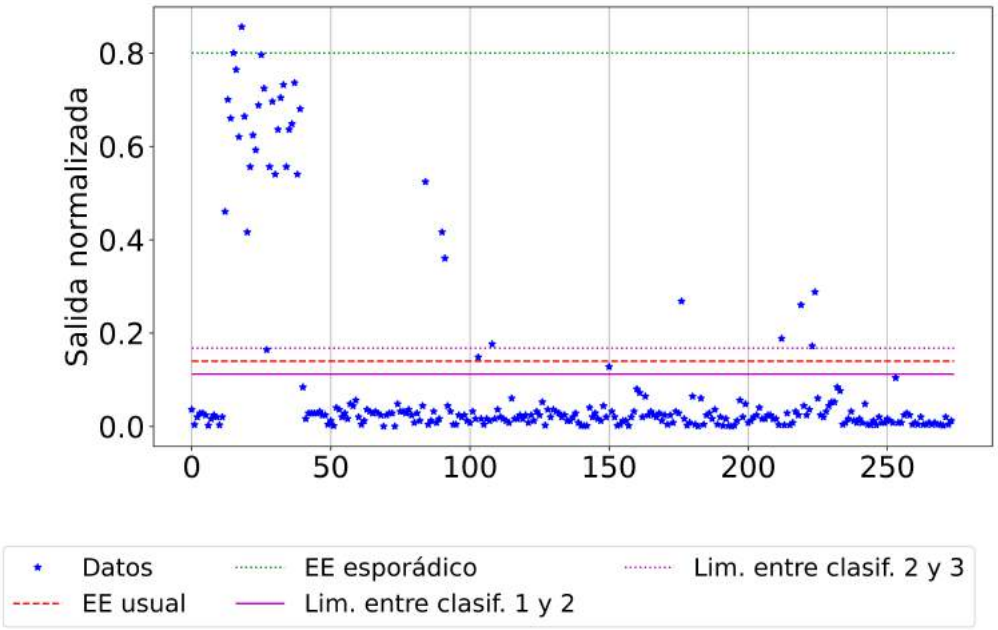


Figura 5.2: Datos de salida normalizados luego de la filtración de *outliers*.

X_5, 35816 puntos para X_6, y 42658 puntos para X_8. Para la variable de salida se clasificaron 8 puntos como *outliers*.

Para construir conjuntos de entrenamiento y validación completos para cada muestra de laboratorio, se tomó como valor de las variables de entrada, el promedio de la medida de cada sensor en la última hora a la toma de muestra. En los casos en donde no existía medida en la última hora, se tomó el promedio de la última hora más cercana a la toma. Esto se justifica debido a que por largos periodos de tiempo los datos permanecen sin mucho cambio. La elección del promedio de la última hora se basó en tomar en cuenta el tiempo de residencia (tiempo promedio en el cual un compuesto permanece en el sistema, entre que entra y sale al sistema) de la columna y de que las tomas para las medidas experimentales se etiquetan con una hora estimada.

Luego de estos pasos, el conjunto de datos completo resultó en 274 valores. Esto pone de manifiesto que si bien se partió de más de 860 mil datos, los datos utilizables para la construcción del modelo son mucho menos. En la Figura 5.2 se muestran los datos normalizados de la variable de salida.

5.3.1. Generación de modelos a partir de los datos

5.3.1.1. Construcción de los conjuntos de entrenamiento y validación

Como el sistema opera en dos estados estacionarios diferentes, son necesarios datos que representen los dos estados en los conjuntos de entrenamiento y validación. Según la Figura 5.2, se pueden visualizar cinco bloques: uno debajo de la menor especificación de concentración máxima (la mayoría de los datos pertenecen a este bloque), otro por arriba de la mayor especificación de concentración máxima, un bloque intermedio (que pueden ser los valores de transición entre un modo de operación y el otro, o que existe un desvío importante de las especificaciones), y dos bloques que se corresponden a que el proceso está trabajando en el límite especificado o en un entorno pequeño a ese valor.

Debido a que los valores mayores al EE usual no son tan fáciles de identificar, en este caso, se ha optado por separar los datos en tres bloques: uno correspondiente a cuando el proceso trabaja en un entorno de 20 % del EE usual, otro por debajo de este, indicando que se está trabajando conforme a los límites de especificados, y un tercer bloque con los datos por encima del entorno.

Una vez separados los datos en estos tres bloques, los conjuntos de entrenamiento y validación fueron generados con puntos de cada bloque seleccionados de forma aleatoria, de forma que 75 % de cada bloque es utilizado para entrenamiento, y 25 % es utilizado para validación. Para prevenir la extrapolación del modelo en la etapa de validación, en el conjunto de entrenamiento se incluyeron el máximo y mínimo punto en la variable de salida. Además, para reducir el posible sesgo inducido por los datos elegidos para entrenamiento y validación, se utilizaron cinco combinaciones de conjuntos de datos, de forma de elegir los mejores parámetros de configuración del algoritmo. Los pares de conjuntos (entrenamiento/validación) se han denominado Conjunto 1, Conjunto 2, ..., Conjunto 5.

5.3.1.2. Construcción del modelo

Para este caso se utilizó la misma implementación de PK que para el caso del Capítulo 4. La Tabla 5.1 muestra los parámetros utilizados que se mantienen fijos para todas las ejecuciones. Además, en la Tabla 5.2 se presentan los

Tabla 5.1: Parámetros del algoritmo que se mantienen constantes para todas las ejecuciones.

Operadores matemáticos*	$+, -, *, /, x^y$
Altura inicial	$0 \leq d \leq 3$
Altura máxima	10
Probab. de entrecruzamiento	1
Probab. de mutación	1
Tamaño máximo de población	Número de bases
Tamaño de la población hija	Número de bases
Método de inicialización de población	GP Half And Half
Método de entrecruzamiento	Entrecruzamiento por un punto
Método de mutación	90 % Mutación uniforme, 10 % Mutación de constantes
α	0.05
θ	10^{-4}
Reinicio	500 iteraciones
Criterio de finalización	Nro. de iteraciones o $(1 - Adj.R^2) < 10^{-10}$

*La división está protegida para el valor 0.

Tabla 5.2: Parámetros del algoritmo que cambian según las ejecuciones.

	Nro. de bases	Iteraciones
KP_{2000}^{8}	8	2000
KP_{4000}^{8}	8	4000
KP_{2000}^{12}	12	2000
KP_{4000}^{12}	12	4000

parámetros que más afectan los resultados, y se eligen 4 combinaciones diferentes de uso. El algoritmo fue ejecutado 100 veces con cada configuración, para cada par de conjuntos de entrenamiento y validación. Así como en el caso del Capítulo 4, fueron utilizados pruebas de hipótesis para evaluar la existencia de diferencias estadísticas entre las distribuciones de los modelos obtenidos. En este caso se utilizaron las pruebas de Friedman y la de rangos con signo de Wilcoxon con la distribución de los RMSE obtenidos luego de aplicarse el modelo a cada conjunto de validación. Para las pruebas estadísticas se utilizó un nivel de confianza del 99 %.

5.3.2. Ensamblado de modelos

Luego de la selección de la mejor configuración, los dos mejores modelos generados con cada uno de los cinco conjuntos de entrenamiento con dicha configuración fueron considerados para la construcción del ensamblado de modelos. Los mejores modelos son aquellos que presentaron menor RMSE para el

conjunto de entrenamiento. Seguidamente se llevó a cabo una etapa de poda (*pruning*), en donde las expresiones duplicadas fueron eliminadas. Finalmente, el modelo final se construye con el promedio de los modelos resultantes en la etapa anterior.

De forma de comparar el modelo obtenido por el ensamblado de modelos, se creó un sexto conjunto de prueba (tanto de entrenamiento como de validación). Cómo técnica de comparación se eligió ProcGaus (implementado en el paquete `scikit-learn` de `Python`). Para ello se aprendió el modelo por ProcGaus con el sexto conjunto de entrenamiento, y luego se realizó la evaluación comparativa de los modelos sobre el sexto conjunto de validación.

5.4. Resultados experimentales

En primera instancia se presentarán los resultados obtenidos para seleccionar la mejor configuración del algoritmo, luego los resultados para el ensamblado de modelos, y por último la evaluación de degradación del ensamble de modelos.

5.4.1. Selección de la mejor configuración del algoritmo

En la Tabla 5.3 se presentan los resultados de los experimentos computacionales para seleccionar la configuración del algoritmo. Como ya fue indicado, cada experimento fue realizado 100 veces, esto es, se obtuvieron 100 modelos para cada configuración y cada conjunto de datos. En la misma tabla se indican los valores estadísticos de las distribuciones para cada configuración y cada conjunto.

A partir de los resultados mostrados en la Figura 5.3 se puede visualizar que para el Conjunto 3 la mediana de las cuatro configuraciones son similares, mientras que para el resto de los Conjuntos (1, 2, 4 y 5) son diferentes. Esto está de acuerdo a lo devuelto por la prueba de Friedman, que se muestra en la Tabla 5.4. En la tabla, SS se utiliza para indicar cuando hay diferencia estadística entre las distribuciones obtenidas entre cada configuración, y no SS cuando no la hay.

La Tabla 5.5 muestra los resultados de la prueba de Wilcoxon para las configuraciones en donde la prueba de Friedman indica que las distribuciones son estadísticamente diferentes. A partir de las pruebas estadísticas no se puede

Tabla 5.3: Indices estadísticos para la distribución de RMSE (normalizado) para los diferentes conjuntos de validación y configuración del algoritmo.

Conjunto	Config.	Promedio	Min	Q_1	Mediana	Q_3	Max
Conjunto 1	KP_{2000}^{8}	5.32e-2	4.84e-2	5.16e-2	5.32e-2	5.48e-2	5.72e-2
	KP_{4000}^{8}	5.32e-2	4.84e-2	5.12e-2	5.24e-2	5.48e-2	5.92e-2
	KP_{2000}^{12}	5.44e-2	4.56e-2	5.00e-2	**5.20e-2**	5.36e-2	1.84e-1
	KP_{4000}^{12}	5.56e-2	4.56e-2	5.00e-2	**5.20e-2**	5.32e-2	1.84e-1
Conjunto 2	KP_{2000}^{8}	5.04e-2	3.92e-2	4.80e-2	**5.00e-2**	5.16e-2	6.08e-2
	KP_{4000}^{8}	5.12e-2	4.24e-2	4.72e-2	**5.00e-2**	5.12e-2	1.20e-1
	KP_{2000}^{12}	5.16e-2	4.36e-2	4.96e-2	5.08e-2	5.44e-2	5.96e-2
	KP_{4000}^{12}	5.12e-2	4.32e-2	4.88e-2	5.08e-2	5.36e-2	6.28e-2
Conjunto 3	KP_{2000}^{8}	4.76e-2	3.88e-2	4.40e-2	4.64e-2	5.16e-2	6.00e-2
	KP_{4000}^{8}	4.68e-2	3.96e-2	4.36e-2	4.56e-2	4.88e-2	6.00e-2
	KP_{2000}^{12}	4.64e-2	3.68e-2	4.40e-2	4.64e-2	4.96e-2	5.36e-2
	KP_{4000}^{12}	4.56e-2	3.68e-2	4.48e-2	4.56e-2	4.72e-2	5.32e-2
Conjunto 4	KP_{2000}^{8}	4.16e-2	3.36e-2	3.76e-2	3.96e-2	4.20e-2	1.30e-1
	KP_{4000}^{8}	4.08e-2	3.32e-2	3.72e-2	3.88e-2	4.08e-2	1.30e-1
	KP_{2000}^{12}	3.86e-2	3.32e-2	3.60e-2	**3.84e-2**	4.08e-2	4.40e-2
	KP_{4000}^{12}	3.76e-2	3.24e-2	3.48e-2	**3.72e-2**	4.04e-2	4.52e-2
Conjunto 5	KP_{2000}^{8}	5.52e-2	4.24e-2	4.80e-2	**5.20e-2**	5.88e-2	9.28e-2
	KP_{4000}^{8}	5.60e-2	4.16e-2	4.76e-2	**5.28e-2**	5.48e-2	8.32e-2
	KP_{2000}^{12}	5.96e-2	4.36e-2	4.96e-2	5.44e-2	6.40e-2	1.63e-1
	KP_{4000}^{12}	6.72e-2	4.60e-2	5.24e-2	6.04e-2	6.88e-2	1.63e-1

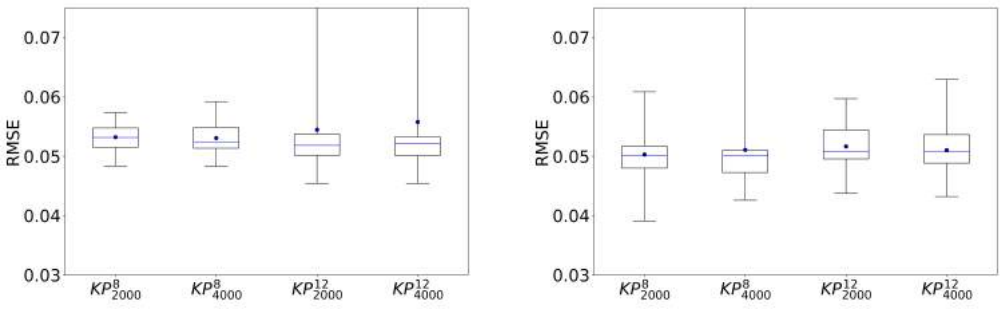
Los mejores resultados están en formato **negrita**.

Tabla 5.4: Resultados de la prueba de Friedman para cada conjunto.

	Conjunto 1	Conjunto 2	Conjunto 3	Conjunto 4	Conjunto 5
$p-valor$	0.68e-3	0.31e-2	0.20e0	0.20e-4	0.28e-4
Resultado	SS	SS	No SS	SS	SS

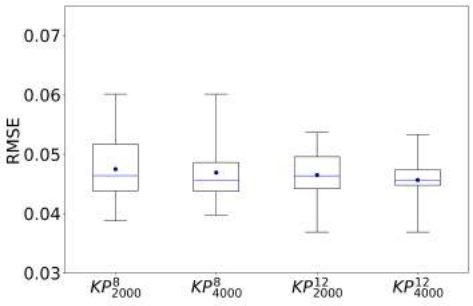
encontrar una clara tendencia de una configuración sobre otra, pero se puede encontrar que para las 12 comparaciones de a pares, la configuración KP_{2000}^{8} es mejor 2 veces, y peor 4 veces, KP_{4000}^{8} es mejor 2 veces y peor solo una vez, KP_{2000}^{12} es mejor 3 veces y peor 2 veces, y KP_{4000}^{12} es mejor 3 veces y peor 3 veces; en los otros casos no hay diferencia estadística entre las comparaciones.

Teniendo en cuenta lo expresado anteriormente, las configuraciones KP_{4000}^{8} y KP_{2000}^{12} son comparativamente mejores que las otras dos. Por ser KP_{4000}^{8} mejor que KP_{2000}^{12} en el Conjunto 2, y no haber diferencia estadística en el resto de los conjuntos, es la configuración que se utilizará para la generación del ensamblado de modelos.

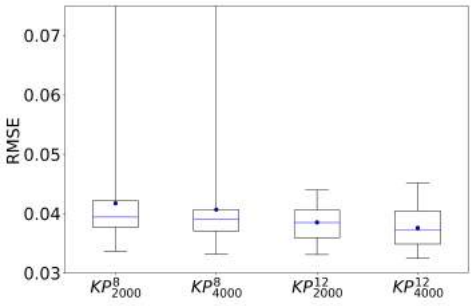


(a) Conjunto 1

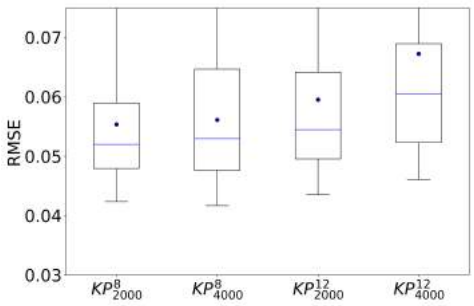
(b) Conjunto 2



(c) Conjunto 3



(d) Conjunto 4



(e) Conjunto 5

Figura 5.3: Diagrama de cajas para la distribución de RMSE en los puntos de validación. •= media; caja: linea inferior=Q1, linea superior=Q3, linea intermedia=mediana; barras: inferior=mín, superior=máx.

5.4.2. Ensamblado de modelos y comparación con Procesos Gaussianos

Como fue mencionado anteriormente, para el ensamblado se consideraron los dos mejores modelos de cada conjunto usando la configuración KP_{4000}^{8}. El modelo final fue construido con 9 modelos, debido a que uno de los modelos estaba duplicado. Los modelos utilizados se presentan en la Tabla 5.6.

Tabla 5.5: Evaluación estadística de a pares por prueba de rangos de signos de Wilcoxon (p−valor y símbolo).

	Conjunto 1			Conjunto 2		
	KP_{4000}^{8}	KP_{2000}^{12}	KP_{4000}^{12}	KP_{4000}^{8}	KP_{2000}^{12}	KP_{4000}^{12}
KP_{2000}^{8}	− 0.61e0	△ 0.22e-2	△ 0.16e-2	− 0.28e0	◁ 0.54e-2	− 0.14e0
KP_{4000}^{8}		− 0.12e-1	− 0.12e-1		◁ 0.13e-3	− 0.24e-1
KP_{2000}^{12}			− 0.90e0			− 0.12e0
	Conjunto 4			Conjunto 5		
	KP_{4000}^{8}	KP_{2000}^{12}	KP_{4000}^{12}	KP_{4000}^{8}	KP_{2000}^{12}	KP_{4000}^{12}
KP_{2000}^{8}	− 0.49e-1	△ 0.32e-2	△ 0.26e-5	− 0.58e0	− 0.47e-1	◁ 0.13e-4
KP_{4000}^{8}		− 0.24e0	△ 0.14e-2		− 0.56e0	◁ 0.38e-3
KP_{2000}^{12}			− 0.31e-1			◁ 0.27e-2

Se indica con '◁' cuando la configuración de la fila es estadísticamente mejor que la de la columna.
Se indica con '△' cuando la configuración de la columna es estadísticamente mejor que la de la fila.
Se indica con '−' cuando no hay diferencia estadística entre las configuraciones.

La Figura 5.4 muestra los resultados obtenidos para cada modelo. Cabe aclarar que los modelos fueron corregidos para que cuando la expresión devolviera valores apenas negativos la concentración resultante fuera 0. La figura incluye el intervalo entre el máximo y mínimo de los valores predichos por los modelos usados para el ensamblado. Como se puede apreciar la distribución de las predicciones es bastante cercana a los datos, tanto para el caso de valores por debajo del EE usual como para aquellos valores por encima de este (caso para el cual se tuvieron menos datos para el aprendizaje). A partir de una comparación visual el ensamble de modelos se desempeña mejor que el modelo obtenido por ProcGaus.

Los resultados numéricos de los modelos comparados se presentan en la Tabla 5.7. En la fila de todos los puntos se muestran los valores de RMSE utilizando el conjunto completo de validación, EE usual indica el RMSE calculado con los datos del conjunto de validación asignados al EE usual, y EE esporádi-

Tabla 5.6: Modelos incluidos en el ensamblado de modelos para KP_{4000}^{8}.

N°	Ecuación
1	$\beta_{1,1} \cdot 0.15^{0.75^{T_{cabeza}} F_{destilado}} + \beta_{1,2} F_{reflujo} \left(F_{reflujo} + P_{fondo}\right) + \beta_{1,3} F_{reflujo} +$ $\beta_{1,5} F_{destilado} + \beta_{1,6} P_{fondo} + \beta_{1,7} \left(-F_{reflujo} + F_{residuo}\right) \left(F_{reflujo} + P_{fondo}\right) + \beta_{1,8}$
2	$\beta_{2,1} F_{residuo} \left(F_{reflujo} + \frac{2F_{reflujo} + F_{residuo} - 1}{F_{residuo} + \beta_{2,2} F_{destilado}}\right) + \beta_{2,3} F_{residuo} + \beta_{2,4} T_{reboiler} +$ $\beta_{2,5} \left(T_{cabeza} - F_{reflujo}\right) \left(2F_{reflujo} + \beta_{2,6} T_{destilado} + P_{fondo}\right) + \beta_{2,7} T_{cabeza} +$ $\beta_{2,8} T_{cabeza} \left(T_{cabeza} - F_{reflujo}\right) + \beta_{2,9} \left(F_{reflujo} + 0.76\right) \left(T_{cabeza} + \beta_{2,10} T_{destilado}\right) +$ $\beta_{2,11} \left(-F_{reflujo} + T_{cabeza}\right) \left(F_{reflujo} + F_{destilado}\right) + \beta_{2,12}$
3	$\beta_{3,1} T_{entrada} + \beta_{3,2} T_{reboiler} + \beta_{3,3} T_{cabeza} - \frac{\beta_{3,4}\left(F_{reflujo} - F_{destilado}\right)\left(F_{residuo} + \beta_{3,5} T_{destilado}\right)}{T_{destilado}} -$ $\frac{\beta_{3,6}\left(F_{reflujo} - F_{destilado}\right)\left(T_{reboiler} + \beta_{3,7} T_{destilado}\right)}{T_{destilado}} + \frac{\beta_{3,8}\left(F_{reflujo} - F_{destilado}\right)\left(1.9 T_{destilado} + 1\right)}{T_{destilado}} +$ $\frac{\beta_{3,9}\left(F_{reflujo} - F_{destilado}\right)\left(F_{reflujo} + T_{reboiler} + \frac{0.6}{P_{fondo}}\right)}{T_{destilado}} + \beta_{3,10} P_{fondo}$
4	$\frac{\beta_{4,1}\left(T_{entrada} - F_{destilado}\right)}{T_{entrada}} - \frac{\beta_{4,2} T_{entrada} F_{reflujo}}{\left(-F_{destilado} + P_{fondo}\right)} + \beta_{4,3} F_{reflujo} T_{cabeza} + \beta_{4,4} F_{reflujo} +$ $\beta_{4,5} F_{destilado} + \beta_{4,6} T_{cabeza} + \beta_{4,7} T_{destilado} + \frac{\beta_{4,8}}{P_{fondo}\left(F_{destilado} + T_{cabeza}\right)} + \beta_{4,9}$
5	$\beta_{5,1} T_{entrada} F_{residuo} + \beta_{5,2} F_{reflujo} + \beta_{5,3} F_{residuo} + \beta_{5,4} F_{destilado} + \beta_{5,5} T_{reboiler}$ $+\beta_{5,6} T_{cabeza} + \beta_{5,7} P_{fondo} + \beta_{5,8} \left(-F_{reflujo} + T_{cabeza}\right) \left(F_{reflujo} - F_{destilado}\right) + \beta_{5,9}$
6	$\frac{\beta_{6,1} F_{reflujo}^{0.53}}{T_{entrada}} - \frac{\beta_{6,2} P_{fondo}}{T_{entrada}} + \beta_{6,3} T_{entrada} + \frac{\beta_{6,4}}{F_{reflujo}} + \beta_{6,5} X_1 + \beta_{6,6} F_{residuo} + \beta_{6,7} F_{destilado} T_{cabeza} +$ $\beta_{6,8} F_{destilado} + \beta_{6,9} T_{reboiler} + \beta_{6,10} T_{destilado} + \beta_{6,11} \left(T_{entrada} - F_{residuo}\right)^{0.39^{F_{reflujo}}} + \beta_{6,12}$
7	$\beta_{7,1} X_0 P_{fondo} + \beta_{7,2} X_0 + \frac{\beta_{7,3} F_{destilado}}{F_{reflujo}} + \frac{\beta_{7,4} F_{residuo} F_{destilado} P_{fondo} \left(-T_{reboiler} + T_{cabeza}\right)}{T_{cabeza} T_{destilado} \left(X_0 + T_{destilado}\right)} + \frac{\beta_{7,5} P_{fondo}}{T_{cabeza}} +$ $\beta_{7,6} T_{cabeza} + \beta_{7,7} P_{fondo}$
8	$\frac{\beta_{8,1} T_{entrada}}{T_{destilado}} + \beta_{8,2} X_1 T_{cabeza} + \beta_{8,3} X_1 \left(F_{reflujo} T_{reboiler} - T_{destilado}\right) + \beta_{8,4} F_{destilado} +$ $\beta_{8,5} T_{reboiler} + \beta_{8,6} T_{cabeza} + \beta_{8,7} T_{destilado} + \beta_{8,8} P_{fondo} +$ $\beta_{8,9} \left(F_{residuo} - T_{destilado}\right) \left(F_{residuo} + T_{destilado}\right)$
9	$\frac{\beta_{9,1} F_{reflujo}^2}{T_{entrada}} + \beta_{9,2} F_{reflujo} T_{cabeza} + \beta_{9,3} F_{reflujo} + \beta_{9,4} F_{destilado} + \beta_{9,5} T_{cabeza} +$ $\frac{\beta_{9,6} T_{destilado}}{P_{fondo}\left(F_{residuo} + T_{reboiler}\right)} + \frac{\beta_{9,8} T_{destilado}}{\left(\frac{T_{cabeza}}{T_{entrada}} + T_{cabeza} + T_{destilado}\right)} + \beta_{9,9} T_{destilado} + \beta_{9,10}$

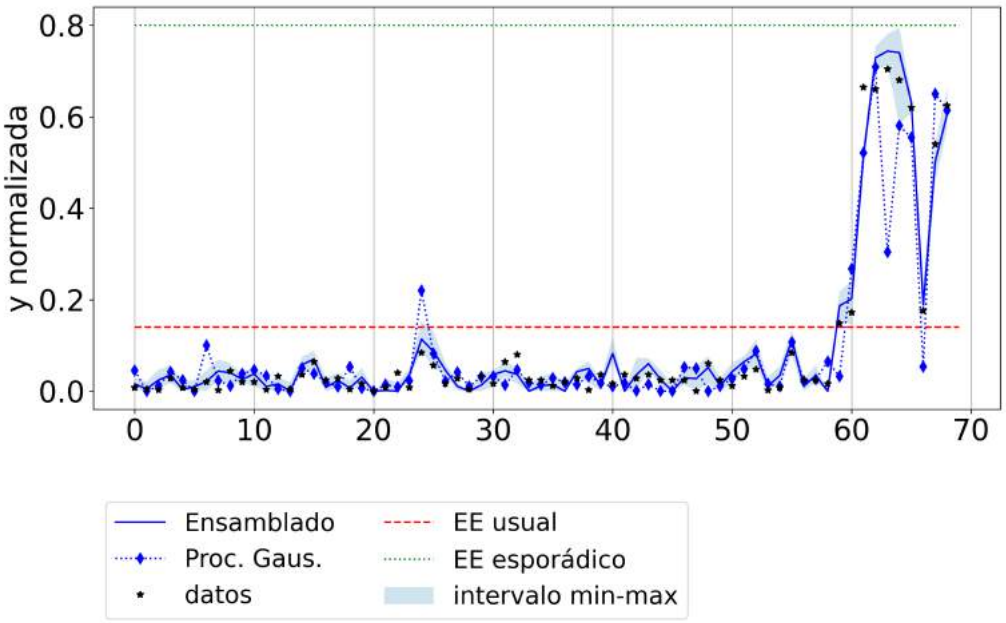


Figura 5.4: Salida y y su estimada y_{est} para cada punto del conjunto de validación para ensamblado de modelos y ProcGaus.

Tabla 5.7: Comparación de RMSE (normalizado) para los diferentes modelos utilizando el conjunto de validación.

	Ensamblado	ProcGaus	Mejor modelo	Modelo ideal
Todos los puntos	3.11e-2	6.64e-2	3.14e-2	1.98e-2
EE usual	2.25e-2	3.12e-2	2.49e-2	1.14e-2
EE esporádico	6.06e-2	1.57e-1	5.59e-2	4.41e-2

cos corresponde al RMSE calculado con los valores cuya salida corresponde a valores mayores al EE usual.

Debido a que ProcGaus trabaja sobre la hipótesis de que los datos siguen una distribución normal, es de esperar que este método estime mejor los puntos de los que se tiene más información (en este caso, los valores correspondientes al EE usual). Sin embargo, para los dos bloques considerados, el modelo ensamblado tiene un mejor desempeño que aquel modelo obtenido por ProcGaus.

Las últimas dos columnas de la Tabla 5.7 corresponden al mejor modelo y al modelo ideal. El primero de ellos es el modelo dentro de los considerados para el ensamblado que tiene un valor de RMSE más pequeño en el conjunto de validación. El segundo de ellos se construye eligiendo los mejores valores obtenidos por alguno de los modelos considerados para el ensamblado, corres-

ponde a una estrategia de ensamblado “ideal” de forma que el estimado sea el mejor valor posible producido por los modelos (esto es, para cada predicción se elige aquella que está más cerca de los datos). Para el mejor modelo se puede señalar, como era de esperar tiene un RMSE mayor al modelo ensamblado, pero tiene un RMSE menor que ProcGaus y tiene un peor desempeño para los valores del EE usual que el modelo ensamblado. Esto último es consistente con lo observado en la Figura 5.4, en donde algunos de los puntos por encima del EE usual tienen un intervalo de valores ancho, indicando que por lo menos uno de los modelos del ensamblado no tiene un buen desempeño en esa zona de operación.

Por último, para el modelo ideal se observa que se corresponde con el menor RMSE obtenido por los otros modelos. Este RMSE mínimo da un valor mínimo al cual se podría llegar si se tuviera una estrategia de ensamblado “ideal” y permite visualizar que pueden existir estrategias de ensamblado mejores que el promedio de los valores.

5.4.3. Evaluación de la degradación del modelo

Debido a los cambios naturales que se dan en los procesos a lo largo del tiempo se obtienen cambios en el proceso que llevan a una degradación del modelo, esto es, el modelo de predicción pierde precisión con el tiempo. Para evaluar la degradación del modelo generado, se utilizaron datos obtenidos en la operación de 2020. Los datos obtenidos para 2020 se corresponden solamente con valores por debajo del límite del estado estacionario usual. En la Figura 5.5 se muestra las predicciones del ensamble de modelos, el modelo obtenido por ProcGaus, el mejor modelo y el modelo ideal para los nuevos puntos. Los valores correspondientes de RMSE son: 1.84e-2, 2.42e-2, 3.18e-2 y 3.96e-3, respectivamente. En comparación con el RMSE de los puntos debajo del estado estacionario usual obtenido para el sexto conjunto, se puede apreciar que el ensamble de modelos predice con una precisión similar. Así como para el sexto conjunto de validación, el ensamble de modelos estima mejor los valores del conjunto de predicción que aquel obtenido por ProcGaus y que el mejor modelo. En comparación con el modelo ideal, el modelo ideal predice mejor, lo que conlleva a pensar que se deben buscar mejores estrategias para el ensamblado de modelos.

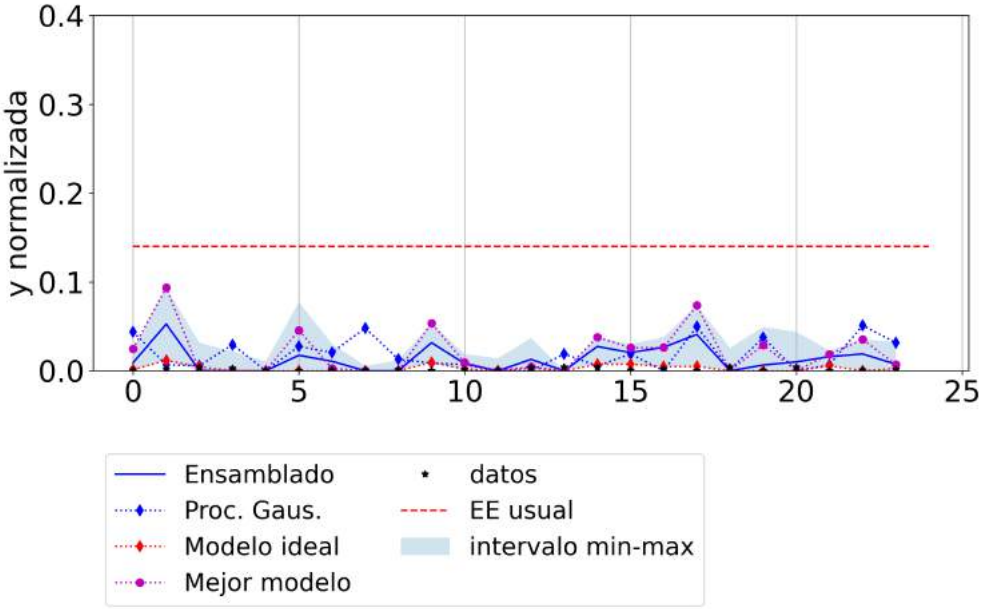


Figura 5.5: Degradación de los modelos: salida y y su estimada y_{est} para cada punto del conjunto de 2020, para ensamblado de modelos, ProcG, el mejor modelo y el modelo ideal.

Capítulo 6

Generación de modelos con múltiples salidas

La mayoría de los algoritmos de aprendizaje automático resuelven el problema de generación de modelos de una variable de salida a partir de varias variables de entrada. En los procesos industriales, lo más usual, es querer obtener varias variables de salida en función de varias variables de entrada. Es de esperar que las ecuaciones que describen las variables de salida de un mismo sistema, al compartir fenómenos físicos y/o químicos, compartan términos que reflejen dichos fenómenos. Por ello, es de utilidad diseñar técnicas que obtengan varios modelos subrogados en una misma ejecución del algoritmo, y que las bases funcionales reflejen en las expresiones resultantes, los posibles fenómenos físico-químicos compartidos entre las variables de salida.

En la Figura 6.1 se presenta un esquema de uso de los algoritmos que trabajan con una variable de salida y con varias variables de salida. Como se puede visualizar, si se requiere hallar modelos para varias variables de salidas utilizando un algoritmo que trabaja con una sola variable de salida, se debe ejecutar el algoritmo de forma independiente, tantas veces como la cantidad de variables de salida del sistema. Por otro lado, un algoritmo que trabaje con varias variables de salida al mismo tiempo, dado los conjuntos de datos de las variables de entrada y las de salida, devuelve luego de su ejecución los modelos para cada variable de salida. En este capítulo se presentarán dos estrategias diseñadas en esta tesis para tratar con sistemas de múltiples salidas en la PK: el uso de múltiple regresiones lineales y el uso del modelo de islas. Luego se presentarán los casos de estudios para comparar el algoritmo de PK

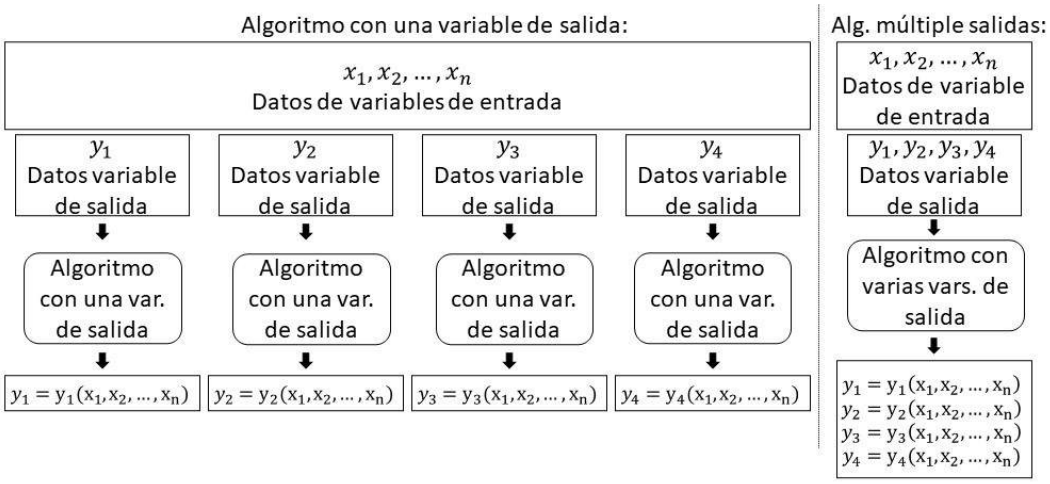


Figura 6.1: A la izquierda algoritmo con una variable de salida y a la derecha un algoritmo con varias variables de salida.

original, y las estrategias para salidas múltiples, y por último la evaluación experimental de los algoritmos. La implementación de ambos algoritmos se encuentra disponible en un repositorio de acceso público en: GitHub[1]

6.1. Estrategia-1: Múltiples regresiones lineales

En la estrategia de múltiples regresiones lineales (MO-OLS) se modificó la implementación del algoritmo de PK para obtener varias salidas en una misma ejecución. De este modo para una misma población, se realizó una regresión lineal múltiple, en lugar de una regresión lineal simple. Este cambio favorece que las variables de salida compartan bases funcionales.

En la Figura 6.2 se presenta el algoritmo de PK diseñado para realizar una búsqueda de múltiple salidas por MO-OLS. El algoritmo de PK para múltiple salidas comienza de la misma forma que la PK para una sola variable (Figura 2.6), las principales diferencias radican en las etapas de realización de regresión lineal, eliminación de individuos de la población y cálculo de la función de *fitness*.

En las etapas de cálculo de coeficientes por mínimos cuadrados, se sustituyó la regresión lineal de una sola variable de salida por una regresión lineal para

[1] https://github.com/JimenaFerreira/KaizenProgramming

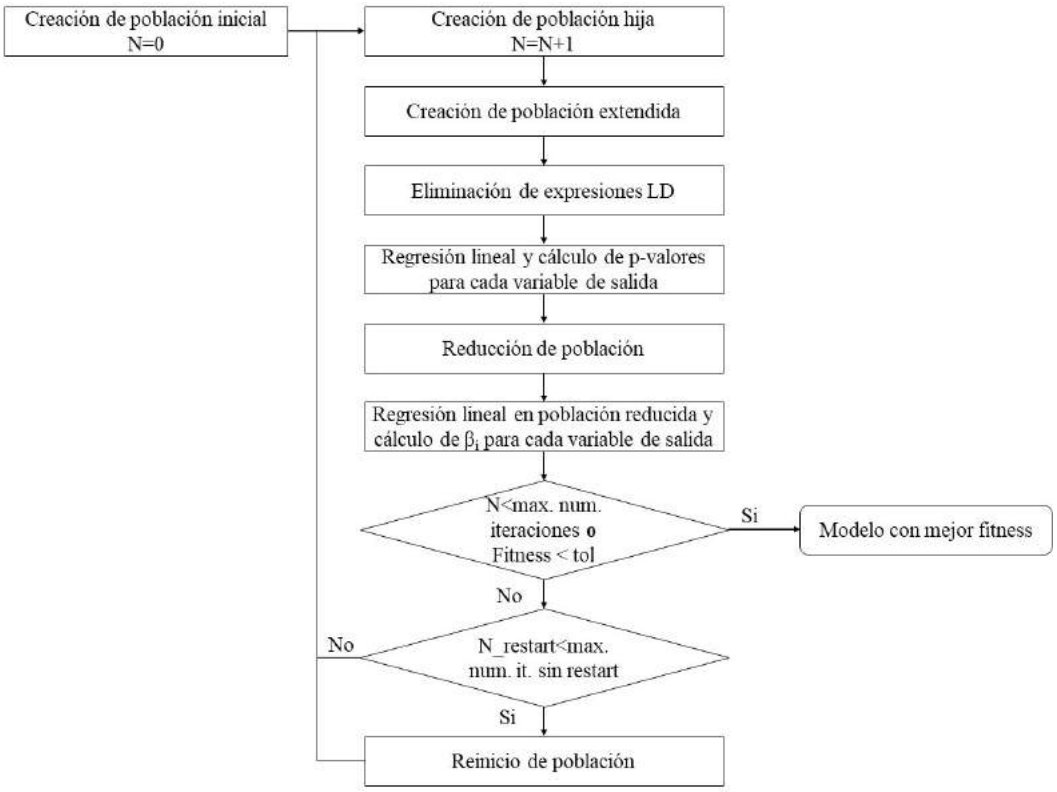


Figura 6.2: Esquema del algoritmo MO-OLS

múltiples salidas, utilizando en la regresión lineal de cada variable de salida las mismas bases o individuos.

Luego, en la etapa de eliminación de individuos, primero se eliminan las bases cuya significancia estadística es baja para todas las variables de salidas. Esto es, si para todas las salidas los p−valores asociados a los coeficientes de una base funcional son mayores a α, la base funcional es descartada porque no es útil para ninguna salida. Luego, en la etapa de eliminación de individuos, primero se eliminan las bases cuya significancia estadística es baja para todas las variables de salidas. Es decir, si para todas las salidas los p−valores asociados a los coeficientes de una base funcional son mayores a α, la base funcional es descartada porque no es útil para ninguna salida. Seguidamente, si los coeficientes asociados a una base funcional en todas las estimaciones de las variables de salida son menores a θ, son descartadas esas bases funcionales. Por último, si luego de la eliminación de bases funcionales por los dos pasos anteriores el tamaño de la población es mayor a P^{max}, se ordenan las bases según el p−valor en cada variable de salida y se toman las P^{max} primeras bases diferentes. El orden considerado es: base funcional más importante para y_1,

base funcional más importante para y_2, ..., base funcional más importante para y_m, segunda base funcional más importante para y_1, segunda base funcional más importante para y_2, y así para el resto de las bases y variables de salida.

La tercera diferencia con el algoritmo de PK de una sola variable de salida es el cálculo de la función de *fitness*. Para el caso de múltiples salidas se utilizan como insumos los R^2 ajustado de cada estimación lineal en forma independiente. Para definir la función de *fitness* se debe decidir como se totalizan estos valores independientes. Algunas alternativas son: el promedio de los R^2 ajustados de las estimaciones de las variables de salida, la mediana de los R^2 ajustado o el mínimo de los R^2 ajustado.

Los cambios realizados en las tres etapas presentadas anteriormente permiten realizar la estimación de varias variables de salida en una sola ejecución del algoritmo, forzando el uso de las mismas bases funcionales para todas las variables de salidas.

6.2. Estrategia-2: Basado en el modelo de islas

El diseño del algoritmo de PK con modelo de islas fue inspirado en el modelo de islas que se utiliza para paralelización de algoritmos evolutivos (Harada y Alba, 2020). El modelo de islas para algoritmos evolutivos se basa en partir la población en varias subpoblaciones, llamadas islas (I), en donde cada isla trabaja de forma independiente el problema de optimización a resolver. Luego, cada determinada cantidad de generaciones, sucede la migración, en donde las islas intercambian información, esto es, cada isla intercambia con una o varias islas el mejor o los mejores individuos hasta ese momento. El modelo de islas va intercambiando etapas de resolución del problema de optimización con etapas de intercambio de información, en donde cada isla resuelve el mismo problema. Este modelo ha mostrado mejorar los resultados respecto a ejecutar los algoritmos evolutivos de forma clásica (Harada y Alba, 2020).

Basado en ese modelo, se diseña una nueva estrategia que permita generar varios modelos subrogados en una sola ejecución. En esta estrategia se propone que cada isla resuelva la RS para cada variable de salida de forma independiente, a diferencia del modelo de islas para algoritmos evolutivos, en donde todas las islas resuelven el mismo problema. Cada cierta cantidad fija de iteraciones, se realiza el intercambio de información en la que cada isla comunica todos sus individuos al resto de islas. De esta forma, en la primera iteración luego del

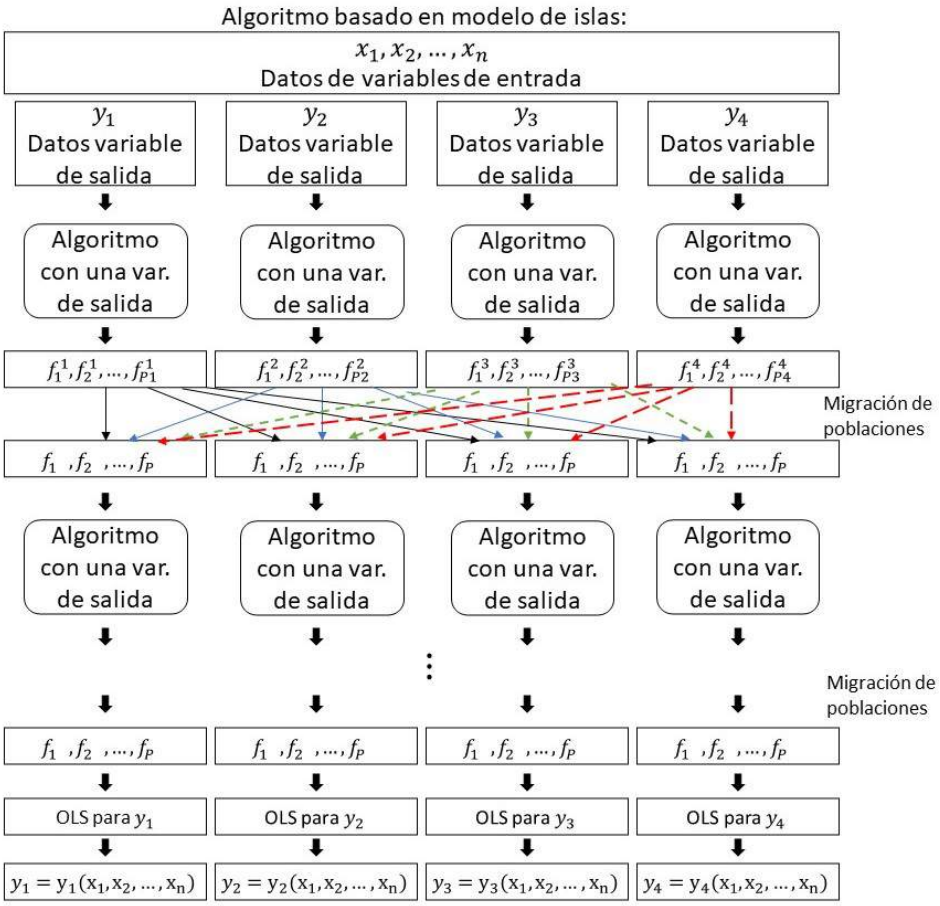


Figura 6.3: Esquema del algoritmo PK con múltiples salidas basado en el modelo de islas.

intercambio de información, todas las islas comienzan con $\sum_{i=1}^{I} P_i$ individuos. Por lo que en la iteración siguiente, se eliminan individuos hasta que queden P^{max} individuos en cada una de las islas. Al igual que en el modelo de islas de algoritmos evolutivos, se realizan iterativamente las etapas de resolución del problema de optimización y de migración, hasta una cantidad máxima de iteraciones. En la Figura 6.3 se presenta un esquema del algoritmo basado en modelo de islas aplicado a la resolución de la RS para múltiples salidas por PK. Cabe destacar que la estrategia propuesta es aplicable a PK y no a PG u otros algoritmos evolutivos clásicos, ya que los individuos de la PK no son soluciones completas, sino términos del modelo final.

Esta forma de intercambio de información permite que todas las variables de salidas compartan las mismas bases en algunas iteraciones, y a la misma vez permite que cada isla genere sus propias bases funcionales en forma inde-

pendiente. Esto es una ventaja respecto a la estrategia de múltiples salidas por regresión lineal mostrada en la sección anterior, ya que en los casos en que las bases funcionales no son iguales o no hay un gran grado de bases iguales en las variables de salida, no se fuerza a utilizar las mismas bases funcionales.

6.3. Metodología experimental

Para el estudio de las estrategias de múltiple salidas se evaluaron dos tipos de escenarios, extensiones de funciones de *benchmarks* de RS y la aplicación a varios sistemas de procesos químicos.

6.3.1. Funciones de *benchmarks* de regresión simbólica

Para analizar y probar las dos estrategias para múltiple salidas se realizó la extensión de 5 funciones de referencia utilizadas para comparar algoritmos de RS. Las funciones consideradas son: Nguyen-4 (Nguyen et al. 2011), Keijzer-12, Keijzer-15 (Keijzer, 2003), Korns-2 y Korns-3 (Korns, 2011).

Se consideraron cuatro esquemas de formas de compartir bases funcionales. En la Figura 6.4 se muestran los esquemas. En el esquema 1, las bases funcionales de y_1 están incluidas en y_2, las de y_2 están incluidas en y_3, y así con el resto de las variables de salida. En el esquema 2, las bases funcionales de y_1, y_2 e y_3 son todas distintas, e y_4 incluye las bases funcionales de las otras 3 variables de salida. En el esquema 3, y_1 e y_3 comparten bases, y_2 e y_3 comparten otras bases que no están presentes en y_1, e y_4 comparte bases con las otras tres variables de salida. De esta forma, el esquema 1 es una situación de gran intersección entre las variables de salida, el esquema 2 es una situación de baja intersección, el esquema 3 es una situación de mediana intersección, y el esquema 4 es una situación sin intersección entre las bases.

Para todos los casos considerados, y_4 son Nguyen-4, Keijzer-12, Keijzer-15, Korns-2 modificado y Korns-3, respectivamente. En las Tablas 6.1, 6.2, 6.3, 6.4 y 6.5 se presentan las ecuaciones para cada variable de salida en cada esquema. Cabe notar, que para todos los esquemas considerados, la mayoría de los coeficientes de un término no son compartidos entre las funciones del esquema correspondiente, de forma de reforzar la idea de compartir bases funcionales entre las variables de salidas. A su vez, todas las variables de salida tienen valores límites distintos, las variables de salidas de la familia de Nguyen-4 tienen

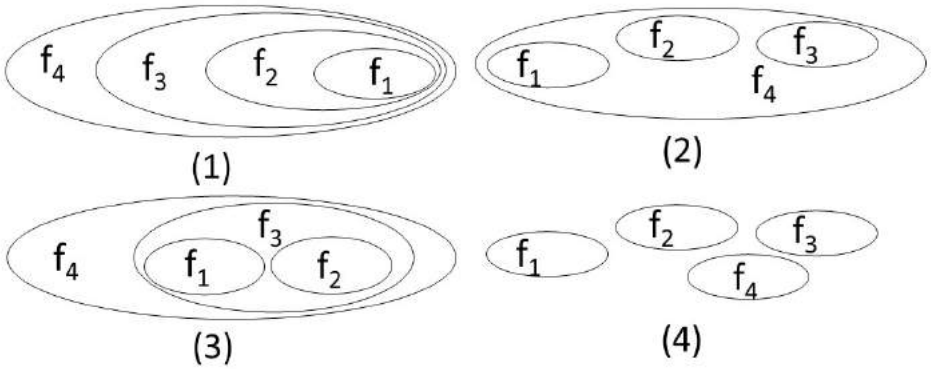


Figura 6.4: Diferentes esquemas de bases compartidas.

Tabla 6.1: Funciones derivadas de Nguyen-4

	Esquema-1	Esquema-2	Esquema-3	Esquema-4
N_1	$x^3 + 1.5x^2 + 3x$	$x^6 + x$	$x^5 + x^4 + x^3$	$x^6 + x$
N_2	$x^4 + \frac{4}{3}x^3 + 2x^2 + 4x$	$x^4 + \frac{4}{3}x^3$	$x^4 + \frac{4}{3}x^3 + 2x^2$	$x^4 + x^3$
N_3	$x^5 + \frac{5}{4}x^4 + \frac{5}{3}x^3 + \frac{5}{2}x^2 + 5x$	x^5	$x^2 + x$	x^5
N_4	$x^6 + x^5 + x^4 + x^3 + x^2 + x$			$x^7 + x^2$

Tabla 6.2: Funciones derivadas de Keijzer-12

	Esquema-1	Esquema-2	Esquema-3	Esquema-4
K_1^{12}	y	$-x^4$	$x^4 - \frac{4}{3}x^3$	x^4
K_2^{12}	$y^2 - 2y$	x^3	$-x^3 + \frac{3}{4}y^2$	$-x^3$
K_3^{12}	$x^3 - 1.5y^2 + 3y$	$-0.5y^2$	$y^2 - 2y$	$0.5y^2$
K_4^{12}	$x^4 - x^3 + 0.5y^2 - y$			$-y$

los valores más pequeños entre -4 y 8.3, los valores de las variables de salida de las familias Keijzer-12 y Keijzer-15, varían entre -80 y 120, y los valores de salida para las funciones de la familias Korns-2 y Korns-3están entre menos y más infinito.

Los conjuntos de entrenamiento y validación fueron basados en el trabajo de de Melo y Banzhaf (2018), en donde para la función de Nguyen-4 se utilizan conjuntos de entrenamiento y validación de 20 puntos creados aleatoriamente utilizando una distribución uniforme entre -1 y 1. Para las funciones Keijzer-12 y Keijzer-15 se utilizaron conjuntos de entrenamiento creados aleatoriamen-

Tabla 6.3: Funciones derivadas de Keijzer-15

	Esquema-1	Esquema-2	Esquema-3	Esquema-4
K_1^{15}	x^3	x^3	$-y-x$	$\frac{1}{5}x^3$
K_2^{15}	x^3+y^3	y^3	y^3-3y	$0.5y^3$
K_3^{15}	x^3+y^3-3y	$-y$	x^3+y^3	$-y$
K_4^{15}	$\frac{1}{5}x^3+0.5y^3-y-x$			$-x$

Tabla 6.4: Funciones derivadas de Korns-2

	Esquema-1	Esquema-2	Esquema-3	Esquema-4
K_1^2	$\frac{x_2}{3x_5}$	$\frac{x_2}{3x_5}$	$\frac{x_2}{3x_5}$	$\frac{x_2}{3x_5}$
K_2^2	$\frac{14.2}{2}\frac{x_2+x_4}{3x_5}$	$\frac{14.2}{2}\frac{x_4}{3x_5}$	$\frac{14.2}{2}\frac{x_2+x_4}{3x_5}$	$14.2\frac{x_4}{3x_5}$
K_3^2	$\frac{14.2}{3}\frac{x_1+x_2+x_4}{3x_5}$	$\frac{14.2}{3}\frac{x_1}{3x_5}$	$\frac{14.2}{3}\frac{x_1}{3x_5}$	$14.2\frac{x_1}{3x_5}$
K_4^2	$0.23+14.2\frac{x_1+x_2+x_4}{3x_5}$			0.23

Tabla 6.5: Funciones derivadas de Korns-3

	Esquema-1	Esquema-2	Esquema-3	Esquema-4
K_1^3	$\frac{x_4}{3x_5}$	$\frac{x_4}{3x_5}$	$4\frac{x_4}{3x_5}$	$4.9\frac{x_4}{3x_5}$
K_2^3	$-2.705+2.45\frac{x_4}{3x_5}$	$2.45\frac{x_2}{3x_5^2}$	$5.41-2.45\frac{x_4+x_2/x_5}{3x_5}$	5.41
K_3^3	$-1.803+1.63\frac{x_4+x_2/x_5}{3x_5}$	2.705	$\frac{x_2}{3x_5^2}$	$4.9\frac{x_2}{3x_5^2}$
K_4^3	$-5.41+4.9\frac{x_4-x_1+x_2/x_5}{3x_5}$			$4.9\frac{1}{3x_5}$

te utilizando una distribución uniforme entre -3 y 3, y para los conjuntos de validación se utilizó el conjunto de puntos entre -3 y 3 separados cada 0.01. Para las funciones Korns-2 y Korns-3 se usaron conjuntos de entrenamiento de 200 puntos y de validación de 2000 puntos creados aleatoriamente utilizando una distribución uniforme entre -50 y 50.

Para cada una de las ejecuciones fueron creados sus correspondientes pares de conjuntos de entrenamiento y validación. Cada algoritmo fue ejecutado 100 veces con cada uno de los conjuntos de entrenamiento creados, de forma que los algoritmos buscaran los modelos a partir de los mismos conjuntos de entrenamiento. Así como en el Capítulo 5, fueron utilizadas las pruebas de Friedman y la de rangos con signo de Wilcoxon para evaluar la existencia

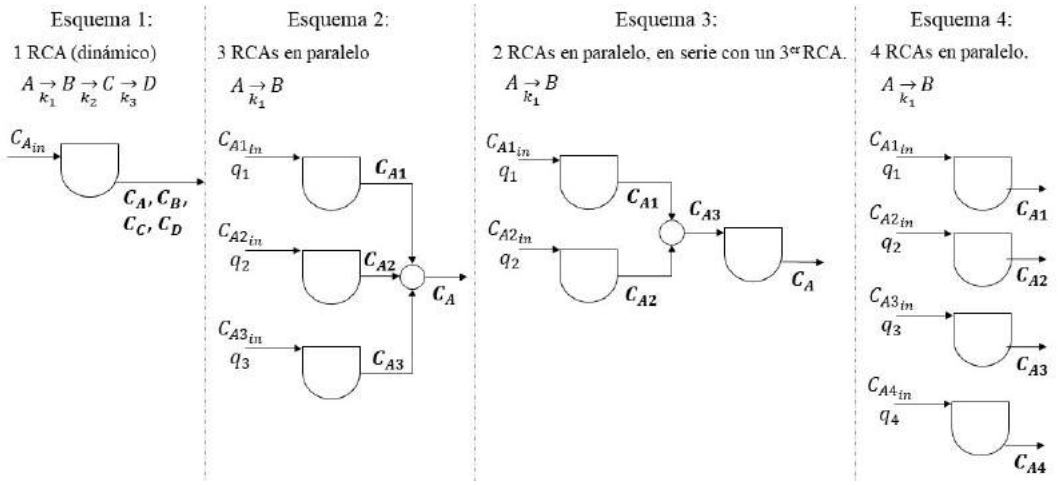


Figura 6.5: Casos de estudio de la industria de procesos.

de diferencias estadísticas entre las distribuciones de los modelos luego de aplicarse el modelo a cada conjunto de validación. Para las pruebas estadísticas fue utilizado un nivel de confianza del 95 %.

6.3.2. Aplicaciones de la industria de procesos

Para los casos de estudio aplicados a la industria de procesos se trabajó con los mismos esquemas de bases compartidas que para los casos de funciones de *benchmark*.

En la Figura 6.5 se representa el sistema para cada esquema, donde C_{ij} representa la concentración de la especie i en la corriente j (las salidas del sistema se etiquetan sin j), q_j el caudal de la corriente j, y k el coeficiente de la velocidad de reacción. En las Tablas 6.6 y 6.7 se presentan las ecuaciones analíticas de cada caso considerado.

Para el esquema 1, de alta intersección entre las variables de salida, se consideró el modelo dinámico de un reactor RCA con tres reacciones en serie de primer orden ($A \rightarrow B \rightarrow C \rightarrow D$). Para este caso, si se considera volumen y flujo de entrada constantes, se obtiene que las bases de y_2 ($C_B(t, C_{A_{in}})$) incluyen las de y_1 ($C_A(t, C_{A_{in}})$), las de y_3 (($C_C(t, C_{A_{in}})$) incluyen las de y_2, y las de y_4 ($C_D(t, C_{A_{in}})$) incluyen las de y_3.

En el esquema 2, que presenta una baja intersección de términos entre las funciones, se utilizó un sistema en EE de tres reactores RCA en paralelo con un punto de mezcla al final de los tres reactores. Las reacciones que suceden ($A \rightarrow B$) son de primer orden. Este modelo podría ser, por ejemplo, tres

Tabla 6.6: Casos de estudio de ingeniería de procesos para el Esquema 1 y 2.

	Esquema-1	Esquema-2
CP_1	$\frac{dC_A}{dt} = \frac{q}{V}(C_{Ain} - C_A) - k_1 C_A$	$C_{A1} = C_{A1,in}\frac{q_1}{q_1 + k_1 V}$
CP_2	$\frac{dC_B}{dt} = -\frac{q}{V}C_B + k_1 C_A - k_2 C_B$	$C_{A2} = C_{A2,in}\frac{q_2}{q_2 + k_1 V}$
CP_3	$\frac{dC_C}{dt} = -\frac{q}{V}C_C + k_2 C_B - k_3 C_C$	$C_{A3} = C_{A3,in}\frac{q_3}{q_3 + k_1 V}$
CP_4	$\frac{dC_D}{dt} = -\frac{q}{V}C_D + k_3 C_C$	$C_A = \frac{C_{A1}q_1 + C_{A2}q_2 + C_{A3}q_3}{q_1 + q_2 + q_3}$

Tabla 6.7: Caso de estudio de ingeniería de procesos para el Esquema 3 y 4.

	Esquema-3	Esquema-4
CP_1	$C_{A1} = C_{A1,in}\frac{q_1}{q_1 + k_1 V}$	$C_{A1} = C_{A1,in}\frac{q_1}{q_1 + k_1 V}$
CP_2	$C_{A2} = C_{A2,in}\frac{q_2}{q_2 + k_1 V}$	$C_{A2} = C_{A2,in}\frac{q_2}{q_2 + k_1 V}$
CP_3	$C_{As} = \frac{C_{A1}q_1 + C_{A2}q_2}{q_1 + q_2}$	$C_{A3} = C_{A3,in}\frac{q_3}{q_3 + k_1 V}$
CP_4	$C_A = C_{As}\frac{q_1 + q_2}{q_1 + q_2 + k_1 V}$	$C_{A4} = C_{A4,in}\frac{q_4}{q_4 + k_1 V}$

plantas de tratamiento de aguas residuales domésticas en donde sus salidas son combinadas para su disposición final. En la solución analítica del sistema, y_1 ($C_{A_1}(q_1)$), y_2 ($C_{A_2}(q_2)$) e y_3 ($C_{A_3}(q_3)$) no comparten bases funcionales, e y_4 ($C_A(q_1, q_2, q_3)$) comparte términos con las otras tres variables.

Para el esquema 3 se consideró un sistema de tres reactores RCA, también en EE, en dónde hay dos RCA en paralelo, y el tercero está en serie con los anteriores. Como ejemplo, también podría considerarse un sistema de plantas de tratamiento de aguas residuales. Para este caso, la solución analítica y_1 ($C_{A_1}(q_1)$) e y_2 ($C_{A_2}(q_2)$) no comparten bases, y_3 ($C_{A_3}(q_1, q_2)$) e y_4 ($C_A(q_1, q_2)$) comparten bases con las anteriores.

En el último esquema, se estudió el sistema en EE de cuatro RCA en paralelo. En este caso y_1 ($C_{A_1}(q_1)$), y_2 ($C_{A_2}(q_2)$), y_3 ($C_{A_3}(q_3)$) e y_4 ($C_{A_4}(q_4)$) no comparten bases funcionales.

Los conjuntos de entrenamiento y validación se obtuvieron a través de

la solución numérica de las ecuaciones del esquema-1 y de las soluciones analíticas de los restantes 3 esquemas. Para las simulaciones se utilizaron los siguientes valores: V=1 L, k_1=3.7 min^{-1}, k_2=4 min^{-1}, k_3=3 min^{-1}, q=2 $L.min^{-1}$, $C_A(0)$=$C_B(0)$=$C_C(0)$=$C_D(0)$=0 $mol.L^{-1}$, $C_{A1_{in}}$ =1 $mol.L^{-1}$, $C_{A2_{in}}$ =2 $mol.L^{-1}$, y $C_{A_{in}}$ =3 $mol.L^{-1}$. Se crearon 100 pares de conjuntos de entrenamiento y validación con 100 datos cada uno, con las siguientes variables de entradas generadas aleatoriamente siguiendo una distribución uniforme entre los siguientes rangos: $C_{A_{in}} \in$ [0.5-3] $mol.L^{-1}$, t$\in$ [0-3] min, q$\in$ [0.5-4] $L.min^{-1}$. De esta forma los valores de las variables de salida en el esquema 1 están entre 0 y 1.05 $mol.L^{-1}$, en el esquema 2 entre 0.55 y 1.55 $mol.L^{-1}$, en el esquema 3 entre 0.11 y 1.04 $mol.L^{-1}$, y en el esquema 4 entre 0 y 1.8 $mol.L^{-1}$.

Así como en el caso de las funciones de referencia se ejecutó cada algoritmo 100 veces utilizando en cada ejecución uno de los conjuntos de entrenamiento creado. Aquí también fueron utilizadas las pruebas de Friedman y la de rangos con signo de Wilcoxon para evaluar la existencia de diferencias estadísticas entre las distribuciones de los modelos con un nivel de confianza del 95 %.

6.3.3. Implementación y parámetros utilizados

Así como en los casos de los estudios anteriores los algoritmos fueron implementados en `Python`, y las ejecuciones fueron realizadas en una computadora con las siguientes características: procesador Quad Core Intel i7 7500U de 2.90 GHz. y memoria RAM de 12 GB.

En la Tabla 6.8 se presentan los valores de los parámetros utilizados para las ejecuciones. Los valores de los parámetros del algoritmo PK donde se resuelve para una sola variable (S-PK) son los mismos que para el caso del Capítulo 5 con la excepción del tamaño de población y las generaciones. Para los algoritmos propuestos, PK con múltiples regresiones OLS (MO-OLS) y PK con modelo de islas (MO-I), se utilizan los mismos valores que para S-PK con excepción de las generaciones y el reinicio. La cantidad de generaciones fueron elegidas en cada caso para tener una cantidad equivalente de evaluaciones de las funciones. En S-PK se realizan cuatro ejecuciones independientes de 2000 iteraciones, es decir 2000 evaluaciones de la función. Equivalentemente en MO-OLS se realizan 8000 generaciones ya que se evalúa 2000 por la cantidad de variables de salida, y para el caso de MO-I se utilizan 100 generaciones para cada ejecución independiente cada 20 migraciones.

6.4. Resultados

6.4.1. Funciones de *benchmarks* de regresión simbólica

En esta sección se presentan los resultados de los experimentos con los escenarios de las funciones de referencia.

En las Tablas 6.9, 6.10, 6.11 y 6.12 se presentan el mínimo, el primer cuartil, la mediana, el tercer cuartil y el máximo de la distribución de RMSE para cada algoritmo y función considerada. Los resultados muestran que para el caso del algoritmo con solo una variable de salida (S-PK) la mediana de RMSE es 0 en 49 de los 80 escenarios, utilizando MO-I se da en 59 casos y para MO-OLS (en los dos tipos de función de *fitness* considerados) en 55 escenarios. Cabe destacar que los escenarios que resultaron con la mediana mayor a 0 se centran en los esquemas de las funciones derivadas de Keijzer-12 , Korns-2 y Korns-3.

Además, considerando los escenarios con mediana menores a 1e-10 se encuentran, 71 casos para S-PK, 77 para MI-I, 67 para MO-OLS usando el promedio de los R^2 ajustado como función de *fitness*, y 62 para MO-OLS usando el mínimo de los R^2 ajustado como función de *fitness*. Esto indica que los tres

Tabla 6.8: Parámetros del algoritmo que se mantienen constantes para todas las ejecuciones.

Tamaño población	8
Generaciones (S-PK/MO-OLS/MO-I)	2000/8000/100
Migraciones (MO-I)	20
Operadores matemáticos	$+, -, *, /$
Altura inicial	$0 \leq d \leq 3$
Altura máxima	10
Probab. de entrecruzamiento	1
Probab. de mutación	1
Tamaño máximo de población	Número de bases
Tamaño de la población hija	Número de bases
Método de inicialización de población	GP Half And Half
Método de entrecruzamiento	Entrecruzamiento por un punto
Método de mutación	90 % Mutación uniforme, 10 % Mutación de constantes
α	0.05
θ	10^{-4}
Reinicio	25 % de las generaciones
Criterio de finalización	Nro. de iteraciones o $(1 - Adj.R^2) < 10^{-10}$

Tabla 6.9: Distribución de RMSE para cada función, para los algoritmos S-PK y MO-I, para los esquemas 1 y 2.

		S-PK					MO-I				
		Min.	Q1	Med.	Q3	Max.	Min.	Q1	Med.	Q3	Max.
Esquema-1	N_1	0	0	0	0	2.19e-3	0	0	0	0	3.83e-12
	N_2	0	0	0	0	2.97e-2	0	0	0	0	5.37e-4
	N_3	0	0	8.80e-15	9.57e-4	1.38e1	0	0	0	0	5.84e-2
	N_4	0	0	6.10e-14	2.62e-3	3.50e1	0	0	0	0	2.95e1
	K_1^{12}	0	0	0	0	0	0	0	0	0	0
	K_2^{12}	0	0	0	0	0	0	0	0	0	0
	K_3^{12}	0	0	0	0	6.66e-1	0	0	0	0	1.18e-11
	K_4^{12}	0	0	0	1.26e0	6.94e3	0	0	0	0	3.03e2
	K_1^{15}	0	0	0	0	9.8e-1			0	0	4.18e-1
	K_2^{15}	0	0	0	0	3.70e3	0	0	0	0	1.6e-13
	K_3^{15}	0	0	0	0	1.15e3	0	0	0	0	1.11e-5
	K_4^{15}	0	0	0	0	2.95e1	0	0	0	0	1.30e-13
	K_1^{2}	0	0	0	0	1.02e-5	0	0	0	0	1.38e-5
	K_2^{2}	1.76e-14	1.25e-13	2.96e-13	1.49e-12	3.75e1	4.38e-14	1.09e-13	2.52e-13	2.12e-12	9.42e-5
	K_3^{2}	4.59e-14	1.46e-13	6.17e-13	1.67e0	9.98e4	3.86e-14	1.23e-13	2.93e-13	1.65e-08	7.00e-5
	K_4^{2}	1.09e-13	2.50e-12	2.30e-1	6.32e-1	4.34e5	5.98e-14	1.68e-13	5.18e-13	4.19e-07	2.09e-4
	K_1^{3}	1.25e-15	1.21e-14	2.55e-14	4.00e-10	1.16e-5	2.01e-15	8.42e-15	1.98e-14	8.26e-14	7.11e-4
	K_2^{3}	2.60e-15	1.53e-14	3.11e-12	1.05e-6	8.03e-3	2.63e-15	6.83e-15	1.65e-14	1.32e-7	1.21e2
	K_3^{3}	1.45e-13	4.71e1	9.07e2	1.28e4	1.51e8	5.09e-14	1.28e-12	4.20e-11	1.65e-6	7.69e6
	K_4^{3}	7.43e-13	4.54e2	3.40e3	2.06e4	1.44e10	2.98e-13	7.79e-12	1.83e-8	5.40e-5	5.48e7
Esquema-2	N_1	0	0	0	1.09e-3	4.65e1	0	0	0	1.31e-13	6.19e1
	N_2	0	0	0	0	3.19e-14	0	0	0	0	1.68e-11
	N_3	0	0	0	0	2.90e-2	0	0	0	0	2.37e-5
	N_4	0	0	1.13e-5	3.73e-3	5.24e2	0	0	0	0	3.08e1
	K_1^{12}	0	0	0	0	6.85e-1	0	0	0	0	1.11e0
	K_2^{12}	0	0	0	0	1.63e4	0	0	0	0	1.08e-11
	K_3^{12}	0	0	1.04e-15	6.56e-15	8.24e-2	0	0	0	0	2.20e-13
	K_4^{12}	0	0	0	3.66e-01	2.33e14	0	0	0	0	3.00e0
	K_1^{15}	0	0	0	0	2.59e-2	0	0	0	0	4.18e-1
	K_2^{15}	0	0	0	0	2.95e-2	0	0	0	0	4.37e-4
	K_3^{15}	0	0	0	0	0	0	0	0	0	6.44e-10
	K_4^{15}	0	0	0	0	5.92e1	0	0	0	0	2.89e-05
	K_1^{2}	0	0	0	0	2.58e-5	0	0	0	0	4.87e-6
	K_2^{2}	3.54e-14	1.00e-13	2.05e-13	6.87e-13	1.87e-5	2.94e-14	9.98e-14	2.25e-13	9.65e-13	8.16e-4
	K_3^{2}	0	0	0	0	2.66e-4	0	0	0	0	2.34e-4
	K_4^{2}	6.59e-14	2.77e-12	7.76e-5	3.70e-1	1.33e4	5.67e-14	2.12e-13	6.25e-13	3.29e-6	2.32e-1
	K_1^{3}	0	0	1.78e-15	3.64e-9	8.10e-5	0	0	0	2.73e-14	5.73e-5
	K_2^{3}	0	1.39e-13	2.97e-12	1.40e1	2.99e9	5.99e-15	1.18e-13	6.54e-13	5.98e-8	3.79e5
	K_3^{3}	0	0	0	0	0	0	0	0	0	1.00e20
	K_4^{3}	1.05e-12	2.12e2	1.35e3	8.08e3	2.88e8	7.35e-15	1.65e-13	1.05E-11	1.01e4	7.23e5

Tabla 6.10: Distribución de RMSE para cada función, para los algoritmos MO-OLS, para los esquemas 1 y 2.

		MO-OLS promedio					MO-OLS mínimo				
		Min.	Q1	Med.	Q3	Max.	Min.	Q1	Med.	Q3	Max.
Esquema-1	N_1	0	0	0	0	2.65e-1	0	0	0	0	6.00e-1
	N_2	0	0	0	5.21e-16	1.46e0	0	0	0	0	1.61e0
	N_3	0	0	0	9.51e-5	8.72e0	0	0	0	3.83e-3	4.61e0
	N_4	0	0	0	6.66e-4	9.85e0	0	0	0	2.42e-3	8.36e0
	K_1^{12}	0	0	0	0	0	0	0	0	0	0
	K_2^{12}	0	0	0	0	0	0	0	0	0	2.17e-1
	K_3^{12}	0	0	0	0	4.5e0	0	0	0	0	6.66e-1
	K_4^{12}	0	0	0	0	7.45e0	0	0	0	0	1.28e0
	K_1^{15}	0	0	0	0	9.24e-14	0	0	0	0	4.35e-13
	K_2^{15}	0	0	0	0	4.18e-1	0	0	0	0	2.95e-13
	K_3^{15}	0	0	0	0	2.39e-1	0	0	0	0	5.87e-13
	K_4^{15}	0	0	0	0	1.47e-1	0	0	0	0	1.66e-13
	K_1^2	0	0	0	0	3.10e1	0	0	0	0	1.44e-5
	K_2^2	4.38e-14	1.12e-13	3.17e-13	1.03e-12	2.20e2	4.38e-14	1.51e-13	3.63e-13	1.96e-12	7.67e1
	K_3^2	4.72e-14	1.16e-13	2.63e-13	3.26e-12	1.46e2	4.13e-14	1.29e-13	2.94e-13	2.01e-12	5.79e1
	K_4^2	5.88e-14	5.60e-13	1.04e-7	2.45e-1	1.14e3	7.21e-14	4.51e-13	1.02e-07	2.94e-1	1.65e2
	K_1^3	3.71e-15	1.17e-14	2.62e-14	6.61e-9	1.76e3	4.23e-15	1.27e-14	3.20e-14	1.58e-08	1.96e4
	K_2^3	2.60e-15	1.59e-14	1.35e-13	1.78e-6	4.31e3	4.10e-15	2.57e-14	1.27e-9	1.83e0	4.80e4
	K_3^3	1.43e-13	2.80e1	8.09e2	6.92e3	1.56e14	6.42e-13	1.48e2	1.01e3	7.58e3	1.86e7
	K_4^3	8.17e-13	3.99e2	2.41e3	2.04e4	4.69e14	4.85e-12	5.41e2	3.90e3	3.00e4	5.60e7
Esquema-2	N_1	0	0	0	3.25e-4	4.64e0	0	0	0	8.31e-4	3.34e0
	N_2	0	0	0	0	1.73e0	0	0	0	1.04e-15	2.34e0
	N_3	0	0	0	4.67e-05	3.87e0	0	0	0	3.12e-4	1.70e0
	N_4	0	0	0	5.11e-4	2.46e0	0	0	0	1.22e-3	2.03e0
	K_1^{12}	0	0	0	0	5.02e3	0	0	0	0	7.09e0
	K_2^{12}	0	0	0	0	2.51e3	0	0	0	0	2.79e0
	K_3^{12}	0	0	0	0	0	0	0	0	0	0
	K_4^{12}	0	0	0	0	3.11e3	0	0	0	0	4.76e0
	K_1^{15}	0	0	0	0	3.95e-5	0	0	0	0	3.12e-1
	K_2^{15}	0	0	0	0	1.88e-5	0	0	0	0	3.10e-1
	K_3^{15}	0	0	0	0	0	0	0	0	0	0
	K_4^{15}	0	0	0	0	1.11e-5	0	0	0	0	9.33e-1
	K_1^2	0	0	0	0	2.11e1	0	0	0	0	1.26e1
	K_2^2	1.71e-14	9.79e-14	2.11e-13	1.13e-12	7.68e2	3.17e-14	1.04e-13	2.32e-13	2.60e-12	1.29e2
	K_3^2	0	0	0	0	1.51e2	0	0	0	0	6.75e1
	K_4^2	1.34e-13	6.99e-13	2.07e-6	2.41e-1	1.61e3	1.24e-13	6.83e-13	1.89e-7	2.02e-1	2.88e2
	K_1^3	0	0	1.51e-14	2.14e1	6.73e5	0	0	1.36e1	1.13e2	5.87e5
	K_2^3	1.71e-13	3.28e2	8.60e2	7.91e3	1.60e6	1.56e-14	4.05e2	1.23e3	7.16e3	1.10e6
	K_3^3	0	0	0	0	0	0	0	0	0	0
	K_4^3	1.09e-11	6.91e2	1.96e3	1.54e4	3.39e6	3.48e1	8.65e2	3.55e3	1.55e4	2.20e6

Tabla 6.11: Distribución de RMSE para cada función, para los algoritmos S-PK y MO-I, para los esquemas 3 y 4.

		S-PK					MO-I				
		Min.	Q1	Med.	Q3	Max.	Min.	Q1	Med.	Q3	Max.
Esquema-3	N_1	0	0	0	5.10e-6	6.60e0	0	0	0	0	4.50e-2
	N_2	0	0	0	0	7.39e-14	0	0	0	0	1.78e-9
	N_3	0	0	0	0	0	0	0	0	0	2.74e-16
	N_4	0	0	1.20-5	5.06e-3	3.5e10	0	0	0	0	3.20e-2
	K_1^{12}	0	0	0	1.48e-2	3.07e2	0	0	0	0	1.76e0
	K_2^{12}	0	0	0	0	1.00e1	0	0	0	0	8.49e-11
	K_3^{12}	0	0	0	0	3.71e-13	0	0	0	0	2.88e-11
	K_4^{12}	0	0	0	3.55e-1	4.01e3	0	0	0	0	2.67e0
	K_1^{15}	0	0	0	0	0	0	0	0	0	0
	K_2^{15}	0	0	0	0	4.28e-2	0	0	0	0	2.44e-5
	K_3^{15}	0	1.16e-14	5.11e-14	7.74e-13	2.61e3	0	0	0	0	1.65e-3
	K_4^{15}	0	0	0	0	2.73e0	0	0	0	0	2.98e-5
	K_1^{2}	0	0	0	0	4.20e-6	0	0	0	0	7.81e-1
	K_2^{2}	6.09e-14	1.53e-13	3.44e-13	2.53e-12	2.84e-5	5.32e-14	1.16e-13	2.95e-13	2.94e-12	1.39e-4
	K_3^{2}	0	0	0	0	1.08e-4	0	0	0	0	2.53e-5
	K_4^{2}	8.87e-14	6.77e-13	1.89e-05	2.97e-1	1.10e4	6.25e-14	2.05e-13	6.17e-13	3.22e-7	4.03e-1
	K_1^{3}	0	0	0	2.65e-14	3.06e-5	0	0	0	1.27e-14	2.78e-4
	K_2^{3}	4.82e-14	3.91e2	1.73e3	1.39e4	3.11e8	2.32e-14	1.09e-12	6.38e-9	1.18e-4	2.08e6
	K_3^{3}	0	0	2.63e-14	3.09e1	1.00e6	0	0	0	2.17e-7	2.97e6
	K_4^{3}	4.35e-13	5.93e2	2.74e3	2.66e4	5.59e10	3.49e-13	9.30e-12	5.88e-9	2.39e-5	3.42e6
Esquema-4	N_1	0	0	2.91e-15	8.17e-4	1.21e3	0	0	0	1.31e-15	5.03e-3
	N_2	0	0	0	0	2.84e-4	0	0	0	0	3.59e-11
	N_3	0	0	0	0	3.18e1	0	0	0	0	2.47e-12
	N_4	0	0	0	0	0	0	0	0	0	0
	K_1^{12}	0	0	0	0	1.11e0	0	0	0	0	3.33e2
	K_2^{12}	0	0	0	0	3.10e-1	0	0	0	0	1.63e-3
	K_3^{12}	0	0	0	0	1.43e-12	0	0	0	0	1.28e-5
	K_4^{12}	0	0	0	0	0	0	0	0	0	7.08e-2
	K_1^{15}	0	0	0	0	1.54e-5	0	0	0	0	1.12e-3
	K_2^{15}	0	0	0	0	2.09e-1	0	0	0	0	9.29e-4
	K_3^{15}	0	0	0	0	0	0	0	0	0	0
	K_4^{15}	0	0	0	0	0	0	0	0	0	0
	K_1^{2}	3.74e-14	9.62e-14	1.63e-13	3.79e-13	4.98e-5	2.75e-14	9.74e-14	1.79e-13	8.17e-13	1.95e-5
	K_2^{2}	4.24e-14	9.39e-14	1.70e-13	6.68e-13	6.85e-6	4.31e-14	9.52e-14	1.79e-13	9.79e-13	1.65e-4
	K_3^{2}	4.42e-15	8.10e-14	1.44e-13	3.29e-13	8.59e-3	2.87e-14	8.21e-14	1.52e-13	3.33e-13	4.05-6
	K_4^{2}	0	0	0	0	2.35e-1	0	0	0	0	9.55e-1
	K_1^{3}	2.96e-15	9.68e-14	1.76e-13	8.90e-13	6.93e-5	2.33e-14	1.00e-13	1.88e-13	9.90e-13	1.18e-5
	K_2^{3}	0	0	0	0	8.88e-16	0	0	0	0	0
	K_3^{3}	1.37e-13	4.22e-12	2.86e-11	1.86e-3	1.84e6	2.92e-13	3.15e-12	1.95e-11	9.48e-9	6.75e2
	K_4^{3}	2.36e-16	3.98e-15	1.68e-14	2.75e-8	1.63e-4	1.77e-15	5.25e-15	1.41e-14	6.44e-9	4.41e-2

Tabla 6.12: Distribución de RMSE para cada función, para los algoritmos MO-OLS, para los esquemas 3 y 4.

		MO-OLS promedio					MO-OLS mínimo				
		Min.	Q1	Med.	Q3	Max.	Min.	Q1	Med.	Q3	Max.
Esquema-3	N_1	0	0	0	0	1.64e2	0	0	0	9.12e-4	3.98e3
	N_2	0	0	0	0	9.81e1	0	0	0	0	7.68E+02
	N_3	0	0	0	0	0	0	0	0	0	0
	N_4	0	0	0	1.53e-14	1.94e1	0	0	0	1.89e-3	7.55e3
	K_1^{12}	0	0	0	0	1.29e1	0	0	0	0	3.17e1
	K_2^{12}	0	0	0	0	4.36e-1	0	0	0	0	5.72e0
	K_3^{12}	0	0	0	0	0	0	0	0	0	0
	K_4^{12}	0	0	0	0	1.25e1	0	0	0	0	3.20e1
	K_1^{15}	0	0	0	0	5.58e-15	0	0	0	0	0
	K_2^{15}	0	0	0	0	1.19e-3	0	0	0	0	0
	K_3^{15}	0	0	0	0	9.33e-3	0	0	0	0	2.37e-5
	K_4^{15}	0	0	0	0	1.87e-3	0	0	0	0	4.73e-6
	K_1^2	0	0	0	0	8.39e-1	0	0	0	0	7.00e2
	K_2^2	5.37e-14	1.29e-13	3.31e-13	2.38e-9	3.33e1	5.83e-14	1.40e-13	3.36e-13	1.29e-10	1.89e3
	K_3^2	0	0	0	0	4.96e2	0	0	0	0	3.28e4
	K_4^2	7.20e-14	4.29e-13	1.40e-6	2.37e-1	1.73e3	8.13e-14	4.03e-13	3.17e-12	2.43e-1	9.24e4
	K_1^3	0	0	0	5.56e-14	6.80e3	0	0	0	4.26e-14	2.31e5
	K_2^3	5.83e-14	1.54e2	1.43e3	5.96e3	2.78e7	6.57e-13	5.27e1	1.50e3	9.06e3	3.21e7
	K_3^3	0	2.49e1	1.70e3	7.20e3	3.38e7	0	6.27e-1	1.83e3	1.11e4	3.64e7
	K_4^3	3.31e-12	4.32e2	3.22e3	1.53e4	5.56e7	1.57e-11	3.40e2	3.22e3	1.90e4	6.41e7
Esquema-4	N_1	5.55e-17	1.16e-15	5.96e-15	3.74e-14	3.24e2	0	1.09e-15	6.38e-15	1.42e-3	7.95e0
	N_2	0	0	0	0	1.21e-2	0	0	0	0	2.56e-1
	N_3	0	0	0	1.15e-16	4.78e-1	0	0	0	2.37e-4	1.57e0
	N_4	0	0	0	0	0	0	0	0	0	0
	K_1^{12}	0	0	0	0	2.20e0	0	0	0	0	1.74e0
	K_2^{12}	0	0	0	0	4.99e-1	0	0	0	0	5.66e-1
	K_3^{12}	0	0	0	0	8.24e-2	0	0	0	0	2.98e-15
	K_4^{12}	0	0	0	0	0	0	0	0	0	0
	K_1^{15}	0	0	0	0	7.91e-3	0	0	0	0	3.44e-4
	K_2^{15}	0	0	0	0	2.95e0	0	0	0	0	2.95e-13
	K_3^{15}	0	0	0	0	0	0	0	0	0	0
	K_4^{15}	0	0	0	0	0	0	0	0	0	0
	K_1^2	3.10e-14	1.01e-13	2.06e-13	6.71e-13	4.14e4	6.12e-14	9.56e1	1.86e2	4.92e2	1.20e6
	K_2^2	3.62e-14	1.16e-13	2.27e-13	2.72e-12	3.02e4	4.24e-14	7.23e1	2.04e2	4.67e2	1.20e6
	K_3^2	3.10e-14	1.18e-13	2.76e-13	6.28e-13	6.70e4	5.98e-14	1.16e2	2.36e2	5.08e2	6.43e5
	K_4^2	0	2.30e-1	2.30e-1	0	2.30e-1	0	0	2.30e-1	2.30e-1	2.30e-1
	K_1^3	2.32e-14	1.49e-13	8.99e-13	9.65e1	8.22e6	3.24e-14	5.05e-13	5.78e1	2.27e2	5.91e5
	K_2^3	0	0	0	0	0	0	0	0	0	0
	K_3^3	2.75e-12	7.06e2	3.01e3	1.03e4	1.36e8	5.00e-9	9.38e2	3.48e3	1.32e4	6.89e6
	K_4^3	1.84e-15	1.56e-8	2.60e0	8.10e0	7.91e3	1.15e-15	1.44e0	2.99e0	7.70e0	2.91e4

algoritmos considerados pueden obtener modelos con muy buena precisión.

Se realizó la prueba de Wilcoxon en las distribuciones de RMSE del conjunto de validación del algoritmo MO-OLS usando el promedio y usando el mínimo de los R^2 ajustado como función de *fitness*. Como resultado se encontró que hay diferencia estadística para los modelos obtenidos en:

- la familia de funciones de Nguyen-4, en el esquema 3 para la función N_1, N_2 y N_4 y en el esquema 4 para la función N_1, N_3 y N_4,
- la familia de funciones de Keijzer-15 para el esquema 4 en la función K_3^{15},
- la familia de funciones de Korns-2 en el esquema 3 para la función K_2^2,
- la familia de funciones de Korns-3 para el esquema 4 para la función K_1^3.

Por lo tanto, en 71 de las 80 funciones no hay diferencia estadística entre los modelos obtenidos usando el promedio o el mínimo de los R^2 ajustado como función de *fitness*. Como para la mayoría de los escenarios no hubo diferencia estadística, para los siguientes resultados experimentales, se considerará solamente el algoritmo MO-OLS usando el promedio (MO-OLS-promedio).

En la Tabla 6.13 se presentan los resultados de la prueba de Friedman para todas las funciones y esquemas considerados, y en la Tabla 6.14 se muestran los resultados de la prueba de a pares de Wilcoxon, para los casos en donde la prueba de Friedman dio como resultado la existencia de diferencia estadística entre los algoritmos. Las pruebas de Friedman resultaron en 33 casos sin diferencia estadística entre los 3 algoritmos (S-PK, MO-I y MO-OLS). De los 47 casos donde la prueba de Friedman resultó en diferencia estadística, en 31 escenarios MO-I genera mejores modelos que S-PK y en los restantes 16 casos no hay diferencia estadística. Además, en 32 casos MO-I construye mejores modelos que MO-OLS, en 1 caso MO-OLS produce mejores modelos que MO-I, y en los restantes 15 escenarios no hay diferencias estadísticas. Analizando para S-PK y MO-OLS, S-PK genera mejores modelos que MO-OLS en 15 casos, en 16 casos MO-OLS resulta en mejores modelos que S-PK, y en los 17 casos restantes no hay diferencia estadística.

Si se analizan los resultados de las pruebas estadísticas por esquemas, para casi todos los esquemas de las familias de funciones de Nguyen-4, Korns-2 y Korns-3, MO-I genera mejores resultados que los otros dos algoritmos, y para las otras dos familias de funciones casi no hay diferencia estadística entre las distribuciones de RMSE. Para el algoritmo S-PK se tienen mejores resultados que MO-OLS en el esquema 4, que es la situación de no intersección entre las

Tabla 6.13: Resultados de la prueba de Friedman para cada escenario.

	Esquema-1		Esquema-2		Esquema-3		Esquema-4	
	Res.	*p-val.*	Res.	*p-val.*	Res.	*p-val.*	Res.	*p-val.*
N_1	no SS	1.31e-1	SS	3.79e-2	SS	5.38e-3	SS	4.35e-10
N_2	SS	1.76e-5	SS	2.15e-4	SS	4.77e-4	SS	1.02e-3
N_3	SS	3.67e-8	SS	3.40e-7	no SS	3.70e-1	SS	2.07e-7
N_4	SS	1.27e-10	SS	2.29e-11	SS	1.11e-11	no SS	1.00e0
K_1^{12}	no SS	1.00e0	no SS	2.48e-1	SS	3.34e-6	no SS	4.71e-1
K_2^{12}	no SS	1.00e0	no SS	2.25e-1	no SS	2.62e-1	no SS	6.27e-1
K_3^{12}	no SS	1.15e-1	SS	1.68e-30	no SS	9.40e-2	no SS	7.79e-1
K_4^{12}	SS	6.21e-11	SS	1.35e-11	SS	1.65e-6	no SS	3.68e-1
K_1^{15}	no SS	7.79e-1	SS	3.88e-2	no SS	3.68e-1	no SS	2.02e-1
K_2^{15}	SS	5.36e-3	no SS	7.17e-1	no SS	4.49e-1	no SS	7.06e-1
K_3^{15}	no SS	1.32e-1	no SS	3.68e-1	SS	3.80e-31	no SS	1.00e0
K_4^{15}	no SS	4.30e-1	SS	2.32e-3	no SS	9.40e-2	no SS	1.00e0
K_1^{2}	no SS	5.49e-1	SS	3.27e-2	no SS	9.32e-1	SS	1.57e-2
K_2^{2}	no SS	3.28e-1	no SS	5.84e-2	SS	2.49e-2	SS	1.48e-2
K_3^{2}	SS	7.31e-3	SS	3.14e-2	no SS	4.35e-1	SS	4.67e-6
K_4^{2}	SS	1.19e-15	SS	1.13e-10	SS	3.75e-8	SS	1.21e-23
K_1^{3}	SS	2.11e-3	SS	1.19e-5	no SS	8.58e-1	SS	1.29e-5
K_2^{3}	SS	4.64e-3	SS	2.54e-26	SS	6.01e-25	no SS	3.68e-1
K_3^{3}	SS	3.72e-16	SS	4.98e-2	SS	3.26e-18	SS	2.84e-30
K_4^{3}	SS	7.68e-19	SS	2.18e-27	SS	1.16e-25	SS	1.26e-17

funciones, mientras que MO-OLS es bastante superior en el esquema 1 (gran intersección) y en el esquema 3 (intersección intermedia).

Otra forma de análisis de los resultados es tener en cuenta que los algoritmos son usados para problemas de múltiple variables de salidas. Para este análisis se evaluó cuantas veces (de las 100 ejecuciones) los tres algoritmos construyeron modelos con buen ajuste. Para ello, para los algoritmos que generan modelos con múltiples salidas, se consideró como buen ajuste y se lo etiquetó como *hit* cuando los valores de RMSE para cada una de las variables de salida es menor a 1e-10 (de Melo y Banzhaf, 2018) (RMSE calculado en el conjunto de validación). Debido a que las ejecuciones para S-PK son independientes para cada caso, se calcularon los *hits* de forma independiente para cada función, y luego se calculó la cantidad de *hits* como el mínimo valor entre los *hits* para cada variable de salida.

La Figura 6.6 muestra la cantidad de *hits* para los modelos obtenidos por cada algoritmo, agrupados por familia de funciones. Los resultados muestran que MO-I es mejor en 8 de los 20 casos (3 casos de Nguyen-4, 3 de Korns-

Tabla 6.14: Evaluación estadística de a pares por prueba de rangos de signos de Wilcoxon.

Comparación	Esquema-1				Esquema-2				Esquema-3				Esquema-4			
función	N_1	N_2	N_3	N_4	N_1	N_2	N_3	N_4	N_1	N_2	N_3	N_4	N_1	N_2	N_3	N_4
S-PK vs MO-I	▷	▷	▷	▷	▷	–	–	▷	▷	–		▷	▷	–	▷	
S-PK vs MO-OLS	–	◁	▷	▷	–	◁	▷	▷	–	◁		▷	–	◁	◁	
MO-OLS vs MO-I	–	▷	▷	▷	–	▷	▷	▷	▷	▷		▷	▷	▷	▷	
función	K_1^{12}	K_2^{12}	K_3^{12}	K_4^{12}	K_1^{12}	K_2^{12}	K_3^{12}	K_4^{12}	K_1^{12}	K_2^{12}	K_3^{12}	K_4^{12}	K_1^{12}	K_2^{12}	K_3^{12}	K_4^{12}
S-PK vs MO-I				▷			▷	▷	▷			▷				
S-PK vs MO-OLS				▷			▷	▷	▷			▷				
MO-OLS vs MO-I				–			–	–	–			–				
función	K_1^{15}	K_2^{15}	K_3^{15}	K_4^{15}	K_1^{15}	K_2^{15}	K_3^{15}	K_4^{15}	K_1^{15}	K_2^{15}	K_3^{15}	K_4^{15}	K_1^{15}	K_2^{15}	K_3^{15}	K_4^{15}
S-PK vs MO-I		▷			–			▷			▷					
S-PK vs MO-OLS		▷			–			▷			▷					
MO-OLS vs MO-I		–			◁			–			–					
función	K_1^{2}	K_2^{2}	K_3^{2}	K_4^{2}	K_1^{2}	K_2^{2}	K_3^{2}	K_4^{2}	K_1^{2}	K_2^{2}	K_3^{2}	K_4^{2}	K_1^{2}	K_2^{2}	K_3^{2}	K_4^{2}
S-PK vs MO-I			▷	▷	–		–	▷		–		▷	–	–	–	–
S-PK vs MO-OLS			▷	▷	–		◁	–		–		–	◁	◁	◁	▷
MO-OLS vs MO-I			–	▷	▷		▷	▷		–		▷	–	▷	▷	▷
función	K_1^{3}	K_2^{3}	K_3^{3}	K_4^{3}	K_1^{3}	K_2^{3}	K_3^{3}	K_4^{3}	K_1^{3}	K_2^{3}	K_3^{3}	K_4^{3}	K_1^{3}	K_2^{3}	K_3^{3}	K_4^{3}
S-PK vs MO-I	▷	–	▷	▷	▷	▷	–	▷		▷	▷	▷	–		▷	–
S-PK vs MO-OLS	–	–	–	–	◁	◁	–	–		–	◁	–	◁		◁	◁
MO-OLS vs MO-I	▷	–	▷	▷	▷	▷	–	▷		▷	▷	▷	▷		▷	▷

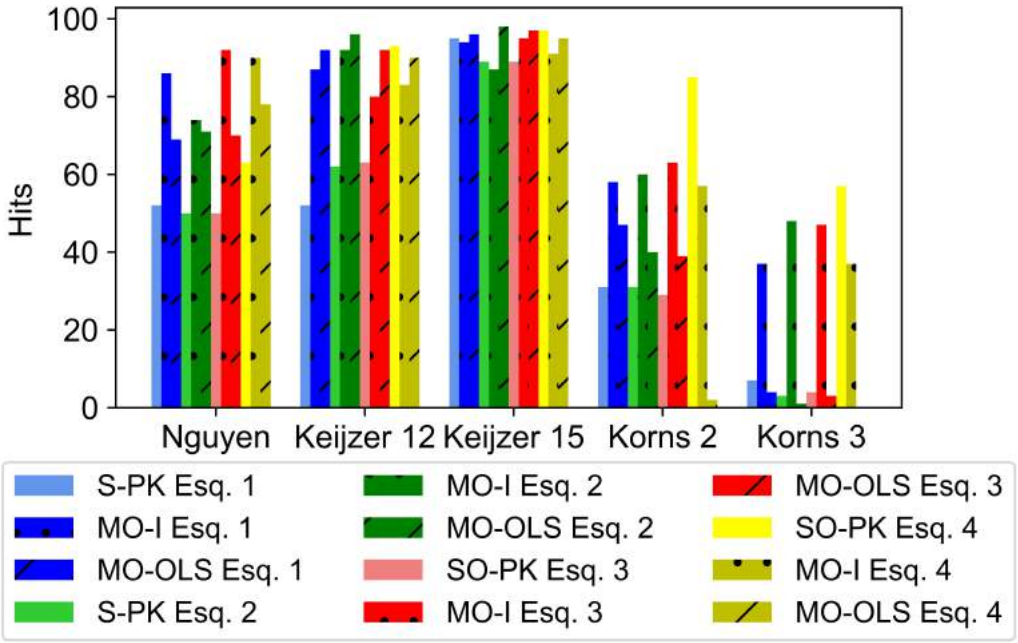


Figura 6.6: *Hits* para cada esquema por familia de función.

2, 2 de Korns-3); MO-OLS-promedio tiene mejores resultados que S-PK en 3 escenarios (1 caso de Keijzer-12, 1 de Keijzer-15 y 1 de Korns-3); y S-PK es

mejor en el esquema 4 de la familia de funciones de Korns-2 y Korns-3. Además, existe un caso en donde los tres algoritmos tienen similar cantidad de *hits*, 3 escenarios en donde MO-I y MO-OLS-promedio tienen la misma cantidad de *hits*, y un caso donde S-PK y MO-OLS-promedio son similares. Por otro lado, si se observan los resultados por esquemas, en 3 de los 4 esquemas de la familia de funciones de Keijzer-15 los tres algoritmos presentan resultados similares, mientras que MO-I en la familia de funciones Korns-2 y Korns-3, para todos los esquemas, tiene una marcada diferencia en la cantidad de *hits* respecto a los otros dos algoritmos. Por último, para la familia de funciones de Nguyen-4 el algoritmo MO-I tiene una clara diferencia respecto a los *hits* de los otros algoritmos en 3 esquemas, y para la familia de funciones de Keijzer-12 los algoritmos para múltiples salidas presentan mayor cantidad de *hits* respecto a la ejecución independiente.

Cabe destacar que a pesar de esperarse a priori un mejor desempeño numérico para los modelos generados por S-PK, ya que los casos son resueltos independientemente y no están forzados a generar los modelos al mismo tiempo. Sin embargo, los resultados muestran que el algoritmo MO-OLS es competitivo con S-PK, y que MO-I no solo produce modelos competitivos con S-PK sino que en muchos casos los modelos generados son mejores.

Debido a que con las estrategias de múltiple salidas se busca obtener modelos que, en el caso de que las variables de salida tengan una estrecha relación, es decir, compartan una gran cantidad de bases funcionales, en las Tablas 6.15 y 6.16 se presentan el promedio y la desviación estándar de las bases compartidas entre pares de funciones de salida obtenidas para cada esquema y cada algoritmo. Como en el caso de las ejecuciones de S-PK son independientes, para realizar una comparación justa con las estrategias para múltiples salidas, se calcula primero todas las posibles combinaciones entre dos funciones para todas las ejecuciones, y luego se resolve un problema de optimización para maximizar la suma de los términos compartidos entre dos funciones (construyendo así, el mayor valor de emparejamiento posible entre las ejecuciones independientes). En la tabla se presenta como valor real, aquel que comparten las funciones originales en cada esquema.

Para que se aprecie mejor los resultados, se incluyen las Figuras 6.7, 6.8, 6.9 y 6.10, en donde se muestra el promedio de las bases compartidas para cada escenario para cada esquema.

Las mayores diferencias entre los valores se debe a los casos en donde la

Tabla 6.15: Cantidad de bases compartidas para cada par de funciones para los Esquemas 1 y 2 (Promedio ± desviación estándar).

			y_1-y_2	y_1-y_3	y_1-y_4	y_2-y_3	y_2-y_4	y_3-y_4
Esquema-1	Nguyen-4	Real	3	3	3	4	4	5
		S-PK	3.01 ± 0.39	2.68 ± 0.73	2.64 ± 0.74	3.26 ± 1.04	3.14 ± 1.09	3.07 ± 1.49
		MO-I	2.93 ± 0.35	2.59 ± 0.99	2.76 ± 0.78	3.43 ± 1.37	3.66 ± 1.10	4.16 ± 1.86
		MO-OLS	3.23 ± 0.99	3.23 ± 0.99	3.17 ± 0.99	4.86 ± 1.66	4.83 ± 1.67	5.79 ± 1.49
	Keijzer-12	Real	1	1	1	2	2	3
		S-PK	1.00 ± 0.00	1.00 ± 0.00	0.90 ± 0.30	2.00 ± 0.00	1.71 ± 0.62	2.34 ± 0.90
		MO-I	1.00 ± 0.00	0.99 ± 0.10	0.87 ± 0.34	1.98 ± 0.20	1.76 ± 0.62	2.61 ± 0.95
		MO-OLS	1.00 ± 0.00	0.99 ± 0.10	0.99 ± 0.10	1.98 ± 0.20	1.98 ± 0.20	3.27 ± 1.22
	Keijzer-15	Real	1	1	1	2	2	3
		S-PK	0.02 ± 0.14	0.04 ± 0.20	0.03 ± 0.17	1.77 ± 0.61	1.91 ± 0.55	2.80 ± 0.68
		MO-I	0.94 ± 0.24	0.93 ± 0.26	0.97 ± 0.22	2.00 ± 0.65	2.03 ± 0.50	2.97 ± 0.68
		MO-OLS	1.03 ± 0.33	1.03 ± 0.33	1.03 ± 0.33	2.34 ± 1.31	2.34 ± 1.31	3.33 ± 1.24
	Korns-2	Real	1	1	1	2	2	3
		S-PK	0.19 ± 0.39	0.23 ± 0.44	0.31 ± 0.56	1.89 ± 0.60	1.91 ± 0.55	2.63 ± 0.74
		MO-I	0.21 ± 0.68	0.16 ± 0.46	0.24 ± 0.60	2.45 ± 3.41	2.51 ± 3.17	3.59 ± 3.24
		MO-OLS	0.35 ± 1.43	0.36 ± 1.37	0.37 ± 1.38	2.33 ± 1.30	2.33 ± 1.30	3.49 ± 1.45
	Korns-3	Real	1	1	1	2	2	3
		S-PK	0.36 ± 0.62	0.54 ± 0.68	0.36 ± 0.54	0.78 ± 0.69	0.72 ± 0.65	1.16 ± 1.18
		MO-I	0.26 ± 0.86	0.26 ± 0.84	0.28 ± 0.79	1.84 ± 1.02	1.71 ± 1.15	2.74 ± 1.69
		MO-OLS	1.07 ± 2.20	1.21 ± 2.21	1.17 ± 2.19	2.08 ± 2.29	2.00 ± 2.22	6.55 ± 3.02
Esquema-2	Nguyen-4	Real	0	0	2	0	2	1
		S-PK	0.53 ± 0.73	0.94 ± 0.37	2.03 ± 1.05	0.12 ± 0.47	1.50 ± 0.90	0.22 ± 0.67
		MO-I	0.40 ± 0.72	0.09 ± 0.29	2.16 ± 1.07	0.02 ± 0.14	1.70 ± 0.71	0.89 ± 0.40
		MO-OLS	2.00 ± 3.55	2.54 ± 4.08	4.31 ± 3.44	1.99 ± 3.68	3.33 ± 2.94	3.29 ± 3.72
	Keijzer-12	Real	0	0	1	0	1	1
		S-PK	0.10 ± 0.30	0.00 ± 0.00	0.70 ± 0.57	0.00 ± 0.00	0.02 ± 0.14	0.88 ± 0.32
		MO-I	0.01 ± 0.10	0.01 ± 0.10	0.87 ± 0.36	0.00 ± 0.00	0.99 ± 0.50	0.93 ± 0.29
		MO-OLS	0.38 ± 2.68	0.01 ± 0.10	1.49 ± 2.76	0.01 ± 0.10	1.37 ± 2.87	0.99 ± 0.10
	Keijzer-15	Real	0	0	1	0	1	1
		S-PK	0.00 ± 0.00	0.00 ± 0.00	0.04 ± 0.20	0.03 ± 0.17	0.04 ± 0.20	0.98 ± 0.14
		MO-I	0.00 ± 0.00	0.03 ± 0.17	0.99 ± 0.26	0.04 ± 0.40	0.96 ± 0.20	0.98 ± 0.14
		MO-OLS	0.00 ± 0.00	0.00 ± 0.00	1.07 ± 0.50	0.01 ± 0.10	1.04 ± 0.31	1.00 ± 0.00
	Korns-2	Real	0	0	1	0	1	1
		S-PK	0.92 ± 0.31	0.92 ± 0.27	0.20 ± 0.45	0.88 ± 0.32	0.20 ± 0.40	0.20 ± 0.47
		MO-I	0.85 ± 0.36	0.87 ± 0.34	0.18 ± 0.50	0.83 ± 0.38	1.08 ± 0.64	0.25 ± 0.75
		MO-OLS	0.92 ± 0.54	0.94 ± 0.70	0.47 ± 0.93	0.91 ± 0.68	0.62 ± 1.56	0.74 ± 1.74
	Korns-3	Real	0	0	1	0	1	1
		S-PK	0.14 ± 0.37	0.00 ± 0.00	0.61 ± 0.72	0.00 ± 0.00	0.56 ± 0.73	0.00 ± 0.00
		MO-I	0.03 ± 0.17	0.00 ± 0.00	0.09 ± 0.45	0.00 ± 0.00	0.74 ± 0.52	0.00 ± 0.00
		MO-OLS	1.67 ± 2.23	0.00 ± 0.00	1.70 ± 2.32	0.00 ± 0.00	4.54 ± 2.71	0.00 ± 0.00

estimación no es exacta. Se puede apreciar que, en general, los modelos obtenidos por S-PK estiman valores de términos compartidos menores a los reales y, que por el contrario, MO-OLS-promedio estima valores mayores. Para estos

Tabla 6.16: Cantidad de bases compartidas para cada par de funciones para los Esquemas 3 y 4 (Promedio ± desviación estándar).

			y_1-y_2	y_1-y_3	y_1-y_4	y_2-y_3	y_2-y_4	y_3-y_4
Esquema-3	Nguyen-4	Real	2	0	3	1	3	2
		S-PK	2.04 ± 0.60	0.60 ± 0.89	2.23 ± 1.16	1.02 ± 0.14	2.48 ± 0.87	1.88 ± 0.41
		MO-I	1.95 ± 0.46	0.25 ± 0.64	2.89 ± 0.88	0.99 ± 0.26	2.84 ± 0.86	1.84 ± 0.54
		MO-OLS	3.23 ± 2.68	0.60 ± 0.91	4.61 ± 2.70	1.16 ± 0.37	3.94 ± 2.47	1.97 ± 0.22
	Keijzer-12	Real	1	0	2	1	2	2
		S-PK	0.78 ± 0.48	0.15 ± 0.38	1.33 ± 1.00	1.00 ± 0.14	1.60 ± 0.65	1.81 ± 0.48
		MO-I	0.86 ± 0.35	0.04 ± 0.24	1.70 ± 0.82	0.96 ± 0.42	1.90 ± 0.96	1.81 ± 0.80
		MO-OLS	1.25 ± 1.14	0.05 ± 0.30	2.31 ± 1.14	1.00 ± 0.14	2.22 ± 1.15	1.97 ± 0.22
	Keijzer-15	Real	1	0	2	1	2	2
		S-PK	1.01 ± 0.10	0.15 ± 0.46	1.94 ± 0.31	0.99 ± 0.30	1.88 ± 0.43	1.91 ± 0.57
		MO-I	0.96 ± 0.20	0.05 ± 0.26	1.96 ± 0.28	0.99 ± 0.30	1.94 ± 0.42	2.13 ± 0.64
		MO-OLS	1.01 ± 0.10	0.08 ± 0.42	2.01 ± 0.10	1.12 ± 0.62	2.09 ± 0.53	2.29 ± 1.24
	Korns-2	Real	1	0	1	1	2	1
		S-PK	0.13 ± 0.36	0.80 ± 0.40	0.29 ± 0.50	0.25 ± 0.46	1.97 ± 0.66	0.35 ± 0.52
		MO-I	0.10 ± 0.36	0.86 ± 0.35	0.15 ± 0.38	0.34 ± 0.84	2.07 ± 1.18	1.12 ± 0.79
		MO-OLS	0.55 ± 1.34	0.85 ± 0.57	0.58 ± 1.31	0.56 ± 1.32	2.70 ± 1.46	0.57 ± 1.30
	Korns-3	Real	1	0	1	1	3	1
		S-PK	0.58 ± 0.71	0.24 ± 0.49	0.52 ± 0.61	0.71 ± 0.77	1.30 ± 1.11	0.49 ± 0.70
		MO-I	0.19 ± 0.59	0.06 ± 0.24	0.17 ± 0.68	0.72 ± 3.16	3.09 ± 4.24	0.73 ± 3.31
		MO-OLS	1.52 ± 3.02	1.04 ± 2.41	1.49 ± 2.99	7.33 ± 4.92	8.24 ± 4.07	7.09 ± 4.81
Esquema-4	Nguyen-4	Real	0	0	0	0	0	0
		S-PK	0.52 ± 0.78	1.02 ± 0.80	0.93 ± 0.26	0.14 ± 0.49	0.04 ± 0.20	0.08 ± 0.27
		MO-I	0.25 ± 0.55	0.04 ± 0.20	0.17 ± 0.38	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.10
		MO-OLS	1.05 ± 2.09	1.75 ± 3.14	0.31 ± 0.46	0.88 ± 2.08	0.11 ± 0.31	0.25 ± 0.43
	Keijzer-12	Real	0	0	0	0	0	0
		S-PK	0.01 ± 0.10	0.00 ± 0.00	0.02 ± 0.14	0.00 ± 0.00	0.93 ± 0.26	0.01 ± 0.10
		MO-I	0.03 ± 0.17	0.00 ± 0.00	0.02 ± 0.14	0.00 ± 0.00	0.95 ± 0.22	0.00 ± 0.00
		MO-OLS	0.42 ± 1.47	0.03 ± 0.17	0.05 ± 0.22	0.04 ± 0.20	0.97 ± 0.17	0.00 ± 0.00
	Keijzer-15	Real	0	0	0	0	0	0
		S-PK	0.00 ± 0.00	0.00 ± 0.00	0.03 ± 0.17	0.01 ± 0.10	0.01 ± 0.10	1.00 ± 0.00
		MO-I	0.00 ± 0.00	0.01 ± 0.10	0.06 ± 0.24	0.01 ± 0.10	0.00 ± 0.00	1.00 ± 0.00
		MO-OLS	0.08 ± 0.80	0.01 ± 0.10	0.01 ± 0.10	0.05 ± 0.22	0.03 ± 0.17	1.00 ± 0.00
	Korns-2	Real	3	3	3	4	4	5
		S-PK	1.60 ± 0.77	1.74 ± 0.64	0.01 ± 0.10	1.60 ± 0.79	0.00 ± 0.00	0.03 ± 0.17
		MO-I	0.80 ± 0.40	0.87 ± 0.34	0.02 ± 0.14	1.62 ± 0.85	0.15 ± 0.41	0.13 ± 0.39
		MO-OLS	1.77 ± 0.95	1.96 ± 1.17	0.00 ± 0.00	2.12 ± 1.57	0.00 ± 0.00	0.00 ± 0.00
	Korns-3	Real	0	0	0	0	0	0
		S-PK	0.00 ± 0.00	0.74 ± 0.46	1.21 ± 0.86	0.00 ± 0.00	0.00 ± 0.00	0.50 ± 0.50
		MO-I	0.00 ± 0.00	0.08 ± 0.27	0.62 ± 0.49	0.00 ± 0.00	0.00 ± 0.00	0.67 ± 0.79
		MO-OLS	0.00 ± 0.00	1.75 ± 2.12	1.34 ± 1.51	0.00 ± 0.00	0.00 ± 0.00	2.50 ± 1.97

casos, la estrategia utilizada en MO-OLS fuerza que los modelos generados en una ejecución compartan las mismas bases, a diferencia de las ejecuciones de S-PK que son completamente independientes. Para el caso de los modelos

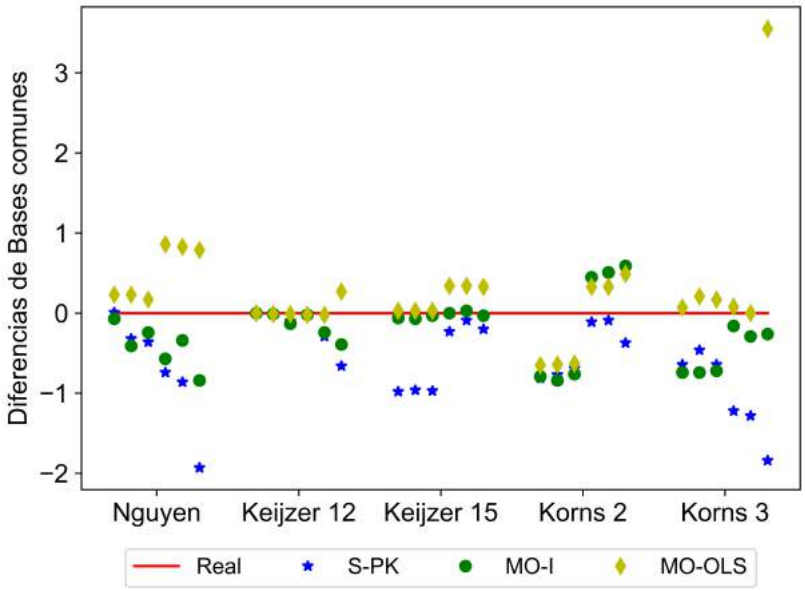


Figura 6.7: Cantidad de bases compartidas para los casos en el Esquema-1. En cada función se muestra las 6 comparaciones en el siguiente orden: y_1-y_2, y_1-y_3, y_1-y_4, y_2-y_3, y_2-y_4, y_3-y_4.

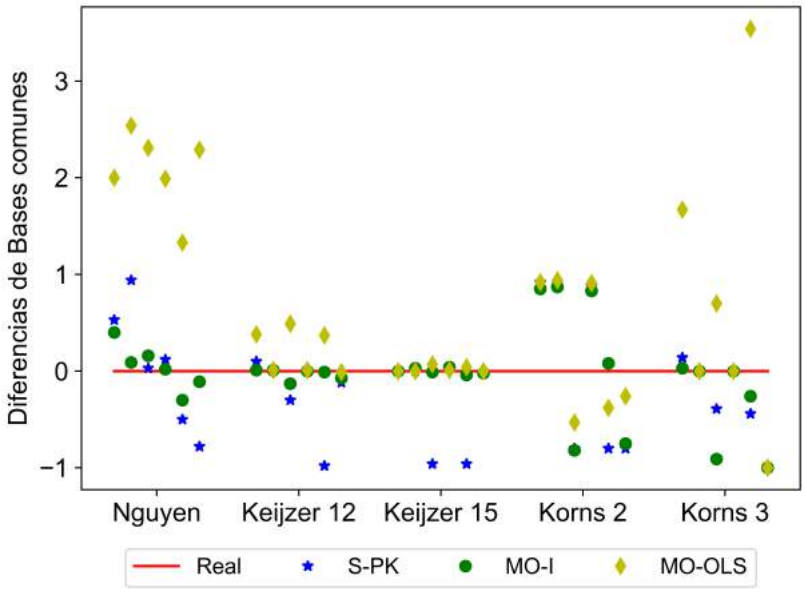


Figura 6.8: Cantidad de bases compartidas para los casos en el Esquema-2. En cada función se muestra las 6 comparaciones en el siguiente orden: y_1-y_2, y_1-y_3, y_1-y_4, y_2-y_3, y_2-y_4, y_3-y_4.

generados por MO-I, el comportamiento es similar a S-PK, pero en algunos casos se obtienen valores muy cercanos a los reales, e indica que esta estrategia,

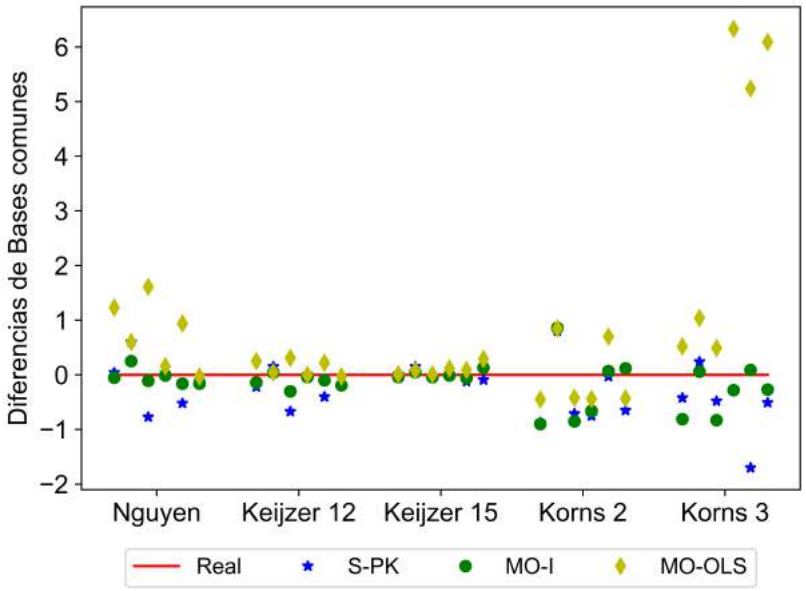


Figura 6.9: Cantidad de bases compartidas para los casos en el Esquema-3. En cada función se muestra las 6 comparaciones en el siguiente orden: y_1-y_2, y_1-y_3, y_1-y_4, y_2-y_3, y_2-y_4, y_3-y_4.

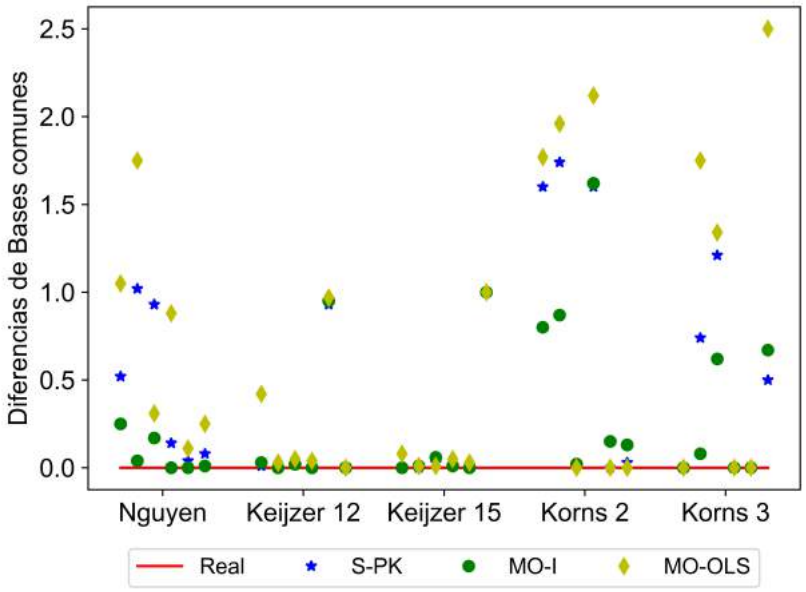


Figura 6.10: Cantidad de bases compartidas para los casos en el Esquema-4. En cada función se muestra las 6 comparaciones en el siguiente orden: y_1-y_2, y_1-y_3, y_1-y_4, y_2-y_3, y_2-y_4, y_3-y_4.

hace que los modelos obtenidos tengan valores de bases compartidas intermedios entre las estrategia MO-OLS-promedio y la S-PK. Esto demuestra que la

Tabla 6.17: Promedio y mediana del tiempo de ejecución en CPU en segundos.

		Nguyen		Keijzer-12		Keijzer-15		Korns-2		Korns-3	
		Prom.	Med.	Prom.	Med.	Prom.	Med.	Prom.	Med.	Prom.	Med.
Esq. 1	S-PK	12.97	4.49	9.92	14.07	9.71	3.52	80.41	82.06	130.67	131.31
	MO-I	3.70	2.37	6.00	4.14	4.04	3.45	25.39	21.34	85.09	67.34
	MO-OLS	41.00	3.12	15.46	2.09	4.90	1.65	179.35	194.95	322.27	333.17
Esq. 2	S-PK	12.92	3.04	10.07	8.55	3.88	1.91	41.71	51.63	97.24	98.89
	MO-I	6.59	4.29	5.92	4.24	1.73	1.58	16.92	13.34	57.97	43.04
	MO-OLS	33.61	1.04	14.28	2.11	4.09	0.97	182.61	184.21	363.29	365.23
Esq. 3	S-PK	10.34	2.85	14.48	14.09	6.36	2.34	49.09	56.43	157.24	157.72
	MO-I	4.71	2.60	8.12	4.78	2.58	2.10	21.2	18.32	77.10	67.52
	MO-OLS	32.22	1.17	14.96	2.02	6.88	1.57	167.32	158.29	308.32	322.22
Esq. 4	S-PK	7.09	1.37	2.76	1.50	0.59	0.38	42.83	47.49	36.93	25.97
	MO-I	4.36	2.52	2.49	1.83	0.62	0.50	61.36	73.16	28.43	23.80
	MO-OLS	26.60	1.10	15.45	1.70	6.64	1.38	309.25	312.34	353.60	352.97

estrategia propuesta basada en el modelo de islas permite identificar con gran efectividad los solapamientos entre las bases funcionales de las distintas salidas del sistema.

Por último, si bien el foco de las estrategias diseñadas para abordar problemas con múltiples variables de salida no está en la mejora de los costos computacionales para la obtención de los modelos, se estudió el tiempo de ejecución de cada estrategia y de las ejecuciones independientes, de forma de visualizar si existe un sobrecosto computacional al utilizar las estrategias propuestas en esta tesis. En la Tabla 6.17 se presentan los valores promedios y medianas de los tiempos de ejecución de CPU para S-PK, MO-I y MO-OLS-promedio (MO-OLS usando el mínimo presenta valores similares) para cada esquema y familia de funciones. Para el caso de las ejecuciones independientes de S-PK el tiempo de ejecución reportado es la suma de los tiempos de ejecución del algoritmo para las cuatro variables de salida.

En los resultados de tiempos de ejecución se encontró que la mediana de los tiempos de ejecución de MO-OLS-promedio fue la más chica en 11 de los 20 casos, en 8 casos la mediana más chica corresponde a MO-I y en los restantes dos casos el menor valor fue el obtenido por S-PK. Esto demuestra que el desempeño de las nuevas variantes propuestas para múltiples salidas es competitivo con el tiempo de ejecución de S-PK. Por otro lado, si se analizan los tiempos promedio, MO-I presenta para casi todos los esquemas menores tiempos que los otros dos algoritmos. Esto demuestra que MO-I tiene tiempos de ejecuciones con poca variabilidad, al contrario de MO-OLS que tiene una

Tabla 6.18: Distribución de RMSE para cada escenario usando ProcGaus.

	Esquema-1			Esquema-2			Esquema-3			Esquema-4		
	Med.	IQR	Max	Med.	IQR	Max	Med.	IQR	Max	Med.	IQR	Max
N_1	1.27e-6	9.85e-6	2.20e-2	9.17e-5	6.46e-4	2.90e-1	1.68e-5	7.44e-5	9.37e-3	1.74e-4	9.66e-4	3.13e-2
N_2	8.74e-6	6.40e-5	3.98e-2	8.81e-6	5.98e-5	9.38e-2	6.72e-6	3.66e-5	3.97e-3	1.67e-5	9.07e-5	2.62e-2
N_3	1.71e-5	8.75e-5	1.09e-1	2.98e-5	1.07e-4	2.69e-2	3.31e-7	2.50e-6	4.89e-4	4.82e-5	1.71e-4	6.81e-2
N_4	1.10e-4	3.82e-4	1.08e-1	9.99e-5	6.78e-4	3.98e-1	8.19e-5	3.93e-4	2.27e-2	1.75e-6	7.67e-6	2.89e-3
K^12_1	7.77e-02	8.33e-02	3.98e-01	8.53e0	3.61e0	1.83e1	8.42e0	5.59e0	2.53e1	8.48e0	4.76e0	2.38e1
K^12_2	3.84e-01	3.24e-01	1.64e0	2.25e0	1.34e0	5.20e0	2.33e0	1.54e0	7.46e0	2.26e0	1.53e0	4.80e0
K^12_3	2.12e0	1.57e0	5.80e0	2.23e-1	1.86e-1	8.20e-1	4.45e-1	4.45e-1	2.78e0	2.17e-1	1.66e-1	6.56e-1
K^12_4	8.88e0	4.76e0	2.11e1	8.80e0	4.59e0	2.11e1	8.18e0	5.25e0	2.37e1	1.01e-1	1.01e-1	3.30e-1
K^15_1	4.13e-2	4.13e-2	1.22e0	2.98e-2	3.39e-2	3.71e-1	3.19e-5	4.27e-5	2.94e-3	4.79e-3	7.27e-3	9.28e-2
K^15_2	5.73e-2	5.66e-2	1.16e0	2.80e-2	2.85e-2	2.14e-1	3.27e-2	3.23e-2	3.64e-1	1.50e-2	1.84e-2	2.37e-1
K^15_3	5.72e-2	5.65e-2	1.16e0	1.82e-5	2.24e-5	8.18e-4	4.71e-2	5.17e-2	5.98e-1	1.18e-5	5.53e-5	3.65e-3
K^15_4	1.97e-2	2.62e-2	5.66e-1	1.59e-2	1.47e-2	1.38e-1	1.71e-2	1.67e-2	1.83e-1	1.14e-5	3.84e-5	3.44e-3
K^2_1	1.32e1	2.08e1	6.68e2	1.61e1	2.40e1	5.87e2	1.33e1	2.80e1	6.51e2	1.92e2	1.93e2	3.27e4
K^2_2	1.22e2	2.14e2	2.14e3	1.13e2	1.85e2	1.79e3	1.28e2	2.01e2	5.21e3	1.93e2	2.41e2	2.05e4
K^2_3	1.07e2	1.96e2	4.04e3	6.03e1	1.18e2	3.40e3	6.56e1	1.12e2	1.65e3	1.94e2	2.70e2	2.51e4
K^2_4	3.21e2	5.87e2	1.21e4	3.73e2	6.06e2	6.59e3	2.93e2	4.47e2	9.90e3	2.30e-1	1.94e-2	2.30e-1
K^3_1	1.52e1	1.70e1	3.09e3	3.59e1	4.10e1	7.37e2	4.88e1	5.97e1	1.76e3	7.09e1	1.03e2	2.35e3
K^3_2	3.73e1	4.14e1	7.58e3	8.65e2	3.69e3	1.10e6	2.10e3	1.02e4	2.69e7	4.98e0	1.58e0	5.41e0
K^3_3	1.52e3	7.34e3	1.43e7	2.61e0	8.38e-01	2.71e0	2.60e3	1.26e4	3.30e7	3.17e3	8.92e3	5.13e6
K^3_4	4.64e3	2.20e4	4.28e7	1.67e3	7.43e3	2.20e6	4.13e3	2.06e4	5.38e7	2.90e0	2.73e0	6.59e1

gran variabilidad de los tiempos de ejecución.

Los valores altos de tiempo de ejecución promedio que registra MO-OLS se pueden deber principalmente a que el criterio de finalización del algoritmo está relacionado con la precisión de los modelos obtenidos o una cantidad fija de iteraciones, por lo que grandes tiempos de ejecución se pueden deber a que no se alcanza la precisión necesaria y se realizan la máxima cantidad de iteraciones. Otro motivo es la complejidad del algoritmo. Para los tres algoritmos se invierte más del 60 % del tiempo de ejecución en realizar las regresiones lineales por OLS. En el caso particular de las regresiones múltiples el tiempo de ejecución de los algoritmos se vería mejorado considerablemente si se realizaran cambios para disminuir el tiempo de ejecución de las regresiones lineales, como por ejemplo utilizar una implementación paralela.

6.4.1.1. Comparación de MO-I con Procesos Gaussianos

Como se ha mencionado anteriormente, se realizaron ejecuciones de ProcGaus para varias variables de salida para las funciones estudiadas anteriormente. En la Tabla 6.18 se presentan los resultados de los escenarios para las

funciones *benchmark* utilizando ProcGaus como técnica de validación externa.

De los datos se puede deducir que en comparación con los algoritmos derivados de PK, los valores de RMSE obtenidos por Procesos Gaussianos son mayores que aquellos obtenidos por PK de una salida, MO-I y MO-OLS. Cabe destacar, que los resultados provenientes del uso de los algoritmos derivados de PK, son claramente mejores que ProcGaus, indicando una muy buena estimación del modelo.

6.4.2. Aplicaciones de la industria de procesos

Para el caso de aplicación en sistemas de procesos químicos se presentan los resultados de la distribución de RMSE en los conjuntos de validación para cada función de cada esquema y cada algoritmo considerado en la Tabla 6.19. Así como en el caso de las funciones de referencia, se evaluó las diferencias estadísticas entre las distribuciones de los RMSE obtenidos por los algoritmos. En la Tabla 6.20 se presentan los resultados de la prueba de Friedman para cada función en cada esquema.

Los resultados de las distribuciones de RMSE muestran similitudes con los casos de las funciones de referencia. Los algoritmos de S-PK y MO-I resultan en mejores modelos que MO-OLS en todos los casos. MO-I fue mejor que S-PK

Tabla 6.19: Distribución de RMSE para cada función y cada algoritmo para el caso de sistemas de procesos químicos.

		SO			MO-I			MO-OLS		
		Med.	IQR	Max	Med.	IQR	Max	Med.	IQR	Max
Esquema-1	CP_1	1.59e-3	2.52e-2	1.32e2	1.74e-5	6.20e-5	3.11e0	4.50e-4	2.02e-3	2.43e0
	CP_2	1.74e-3	7.86e-3	1.69e1	4.52e-5	1.65e-4	3.71e-2	9.93e-4	3.27e-3	1.33e1
	CP_3	1.90e-3	8.00e-3	1.17e5	4.94e-5	4.49e-4	8.96e-2	1.05e-3	3.45e-3	4.15e0
	CP_4	9.58e-4	7.32e-3	4.26e1	5.01e-5	2.23e-4	3.48e-1	1.14e-3	5.00e-3	2.31e1
Esquema-2	CP_1	1.93e-5	6.60e-5	3.71e-3	1.95e-6	1.63e-5	3.15e-3	2.92e-3	3.54e-3	1.49e-2
	CP_2	2.26e-5	9.09e-5	6.40e0	6.03e-6	2.72e-5	2.13e-3	6.65e-3	7.00e-3	2.44e-2
	CP_3	5.24e-5	1.51e-4	7.36e-3	1.35e-5	7.46e-5	4.10e-3	9.43e-3	9.95e-3	3.84e-2
	CP_4	1.16e-2	4.51e-3	3.19e-1	1.36e-2	5.68e-3	5.16e0	2.03e-2	4.20e-3	3.91e-2
Esquema-3	CP_1	4.31e-6	1.58e-5	1.42e-3	1.69e-6	4.40e-6	3.70e-3	1.80e-3	1.15e-3	4.26e-3
	CP_2	6.50e-6	2.02e-5	7.33e-4	4.16e-6	2.56e-5	4.12e-3	2.20e-3	2.22e-3	7.15e-3
	CP_3	1.39e-3	1.20e-3	7.71e-3	1.80e-3	2.34e-3	1.02e-2	2.36e-3	1.28e-3	2.15e-2
	CP_4	5.39e-4	4.08e-4	1.67e-3	7.99e-4	5.17e-4	4.74e-3	1.08e-3	7.90e-4	6.55e-3
Esquema-4	CP_1	1.26e-5	6.82e-5	2.39e-1	1.82e-6	1.13e-5	6.02e-4	3.41e-3	3.28e-4	1.89e-2
	CP_2	2.30e-5	1.55e-4	9.19e0	7.75e-6	2.16e-5	2.12e-2	6.91e-3	8.18e-4	3.87e-2
	CP_3	3.37e-5	2.68e-4	9.62e3	6.20e-6	3.25e-5	1.90e-3	1.03e-2	9.14e-4	1.43e-1
	CP_4	4.11e-5	1.04e-4	3.75e-1	7.21e-6	4.33e-5	1.17e-2	1.38e-2	1.72e-3	9.13e-2

Tabla 6.20: Resultado de la significancia estadística entre las distribuciones de RMSE para el caso de sistemas de procesos químicos.

	Esquema-1		Esquema-2		Esquema-3		Esquema-4	
	Res.	*p-val.*	Res.	*p-val.*	Res.	*p-val.*	Res.	*p-val.*
CP_1	SS	1.83e-18	SS	7.25e-35	SS	1.24e-32	SS	2.95e-35
CP_2	SS	5.36e-19	SS	5.57e-34	SS	5.79e-33	SS	6.49e-31
CP_3	SS	1.04e-13	SS	4.94e-34	SS	1.53e-7	SS	8.96e-32
CP_4	SS	2.10E-13	SS	4.19e-24	SS	5.82e-12	SS	5.50e-33

Tabla 6.21: Evaluación estadística de a pares por prueba de rangos de signos de Wilcoxon para el caso de sistemas de procesos químicos.

Comparación	Esquema-1				Esquema-2				Esquema-3				Esquema-4			
función	N_1	N_2	N_3	N_4	N_1	N_2	N_3	N_4	N_1	N_2	N_3	N_4	N_1	N_2	N_3	N_4
S-PK vs MO-I	$\triangleright$	$\triangleright$	$\triangleright$	$\triangleright$	$\triangleright$	$\triangleright$	$-$	$\triangleleft$	$\triangleright$	$-$	$\triangleleft$	$\triangleleft$	$\triangleright$	$\triangleright$	$\triangleright$	$\triangleright$
S-PK vs MO-OLS	$\triangleright$	$\triangleright$	$-$	$-$	$\triangleleft$	$\triangleleft$	$\triangleleft$	$\triangleleft$	$\triangleleft$	$\triangleleft$	$\triangleleft$	$\triangleleft$	$\triangleleft$	$\triangleleft$	$\triangleleft$	$\triangleleft$
MO-OLS vs MO-I	$\triangleright$	$\triangleright$	$\triangleright$	$\triangleright$	$\triangleright$	$\triangleright$	$\triangleright$	$\triangleright$	$\triangleright$	$\triangleright$	$-$	$\triangleright$	$\triangleright$	$\triangleright$	$\triangleright$	$\triangleright$

Tabla 6.22: Cantidad de bases compartidas para cada par de funciones para caso de sistemas de procesos químicos (Promedio $\pm$ desviación estándar).

		y_1-y_2	y_1-y_3	y_1-y_4	y_2-y_3	y_2-y_4	y_3-y_4
Esquema-1	Real	1	1	1	2	2	3
	S-PK	1.47 $\pm$ 1.04	1.52 $\pm$ 1.12	1.65 $\pm$ 1.31	1.60 $\pm$ 1.25	1.66 $\pm$ 1.38	1.86 $\pm$ 1.42
	MO-I	2.80 $\pm$ 2.47	2.61 $\pm$ 2.42	2.38 $\pm$ 2.38	2.54 $\pm$ 2.36	1.96 $\pm$ 2.16	2.21 $\pm$ 2.69
	MO-OLS	9.69 $\pm$ 2.76	9.80 $\pm$ 2.71	9.78 $\pm$ 2.69	9.78 $\pm$ 2.67	9.75 $\pm$ 2.62	9.85 $\pm$ 2.58
Esquema-2	Real	0	0	1	0	1	1
	S-PK	0.80 $\pm$ 1.19	0.50 $\pm$ 0.92	0.84 $\pm$ 1.4	0.55 $\pm$ 0.94	0.68 $\pm$ 1.28	0.65 $\pm$ 1.28
	MO-I	0.33 $\pm$ 0.80	0.40 $\pm$ 0.79	0.60 $\pm$ 1.19	0.47 $\pm$ 0.91	0.42 $\pm$ 0.96	0.54 $\pm$ 1.12
	MO-OLS	2.80 $\pm$ 2.28	3.13 $\pm$ 2.69	5.02 $\pm$ 2.68	2.86 $\pm$ 2.41	4.83 $\pm$ 2.50	5.34 $\pm$ 2.75
Esquema-3	Real	0	1	1	1	2	1
	S-PK	0.32 $\pm$ 0.80	0.53 $\pm$ 1.14	0.59 $\pm$ 1.30	0.45 $\pm$ 1.11	0.77 $\pm$ 1.51	0.82 $\pm$ 1.51
	MO-I	0.33 $\pm$ 1.02	0.71 $\pm$ 1.28	0.61 $\pm$ 1.10	0.66 $\pm$ 1.26	0.73 $\pm$ 1.39	1.17 $\pm$ 1.55
	MO-OLS	7.91 $\pm$ 2.98	8.67 $\pm$ 2.38	8.73 $\pm$ 2.47	8.53 $\pm$ 2.45	8.58 $\pm$ 2.55	9.36 $\pm$ 1.85
Esquema-4	Real	0	0	0	0	0	0
	S-PK	1.39 $\pm$ 1.06	1.47 $\pm$ 1.04	1.33 $\pm$ 1.03	1.37 $\pm$ 1.23	1.26 $\pm$ 1.15	1.20 $\pm$ 1.05
	MO-I	1.07 $\pm$ 1.16	1.26 $\pm$ 1.35	1.13 $\pm$ 1.12	1.38 $\pm$ 1.38	1.15 $\pm$ 1.15	1.26 $\pm$ 1.36
	MO-OLS	5.74 $\pm$ 1.72	5.53 $\pm$ 1.77	5.45 $\pm$ 1.76	5.97 $\pm$ 1.50	5.80 $\pm$ 1.63	5.62 $\pm$ 1.54

en 11 casos, S-PK fue mejor que MO-I en 3 casos y en los restantes dos casos no hubo diferencias estadísticas entre las distribuciones de RMSE.

Por otro lado, cuando se observan los términos compartidos en los modelos obtenidos (Tabla 6.22), se ve la misma tendencia que en el caso de las funciones de *benchmark*, aunque un poco amortiguado el efecto, fundamentalmente para S-PK. En la Figura 6.11 se muestran las diferencias de la cantidad de bases en comunes respecto al valor real. Esto es, los modelos obtenidos por MO-

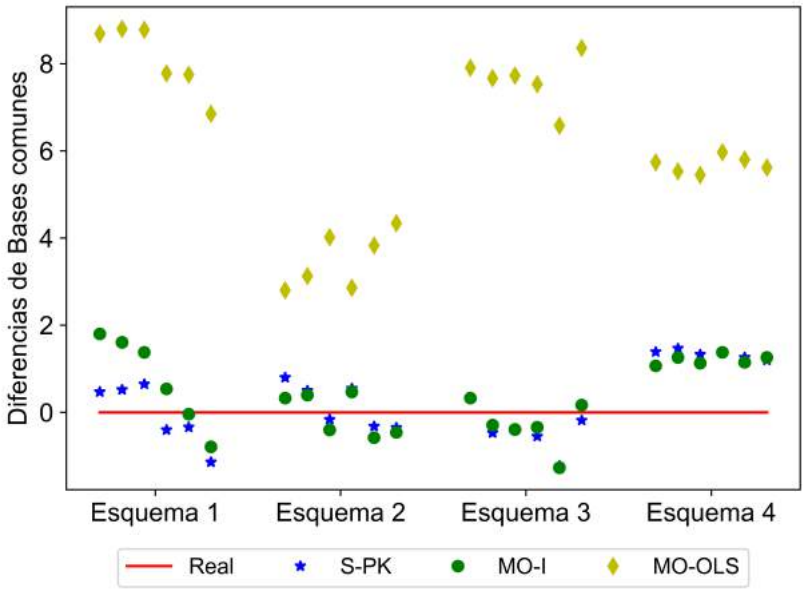


Figura 6.11: Cantidad de bases compartidas para en los 4 esquemas. En cada esquema se muestra las 6 comparaciones en el siguiente orden: y_1-y_2, y_1-y_3, y_1-y_4, y_2-y_3, y_2-y_4, y_3-y_4.

OLS tienden a compartir una gran cantidad de términos en sus expresiones, independientemente de si las salidas reales comparten o no términos. Por el contrario, los modelos obtenidos por S-PK tienden a compartir pocos términos, mientras que los modelos obtenidos por MO-I presentan un comportamiento intermedio entre los dos anteriores. Cabe destacar, que si bien no fue el objetivo inicial del algoritmo MO-I, el mecanismo del mismo permite la convergencia a mejores soluciones que aquellas obtenidas por ejecuciones independientes.

Capítulo 7

Conclusiones y trabajo a futuro

En este tesis se abordó el estudio y diseño de algoritmos para la obtención de modelos subrogados en el contexto de la industria de procesos.

Como punto de partida se estudió el uso de la Programación Genética como método generador de modelos subrogados a partir de datos de simulaciones de una planta de bioetanol. Para el caso de estudio considerado se comprobó que la Programación Genética produce modelos con un bajo valor del error, y se constató los problemas reportados en la bibliografía relacionada, como por ejemplo, el problema de *bloat* y que las constantes de las expresiones encontradas no cambian a lo largo de las iteraciones.

A partir del relevamiento bibliográfico de alternativas para subsanar los inconvenientes presentes en la PG, se identificó al algoritmo de PK como una técnica promisoria que aúna las virtudes de la PG con un tratamiento estadístico más riguroso. Para evaluar este técnica se realizó un estudio comparativo entre la Programación Genética y la Programación Kaizen utilizando funciones de *benchmarks* y de una simulación de una planta de bioetanol en Aspen Plus. Para los casos de estudio se encontró que los modelos obtenidos por la Programación Kaizen presentan un menor residuo en los conjuntos de validación respecto a los modelos obtenidos por Programación Genética y que las expresiones obtenidas son de una complejidad razonable.

Seguidamente, se estudió el uso de la Programación Kaizen para la construcción de un *soft-sensor* de una columna de destilación para estimar la concentración de los hidrocarburos de 4 carbonos en el destilado, utilizando datos históricos de una columna de separación de propanos y butanos de la refinería de ANCAP. A su vez se trabajó siguiendo una metodología para construir mo-

delos a partir de datos reales, en donde se tuvo en cuenta el pre-procesamiento de los datos para elaborar el modelo. Esta etapa es fundamental en la construcción de modelos por técnicas de aprendizaje automático ya que los conjuntos de entrenamiento y validación deben ser completos. En el caso de estudio se encontró que la Programación Kaizen puede producir modelos de la columna con buena precisión. Además, se elaboró un ensamblado de modelos de forma de aprovechar la generación de varios buenos modelos, obteniendo un nuevo modelo de la variable de salida que combina las fortalezas de cada modelo. Para evaluar el método utilizado para la generación de modelos, se comparó la calidad del modelo ensamblado con el modelo obtenido usando Procesos Gaussianos. En este caso, se encontró que el modelo ensamblado supera las estimaciones obtenidas por el modelo por Procesos Gaussianos. En particular, el modelo elaborado a partir de Programación Kaizen es un buen estimador para los dos estados de operación de la columna, en tanto el modelo de Procesos Gaussianos por partir de la hipótesis de distribución normal de los datos no posee una buena estimación sobre el punto de operación esporádico. Para el caso en estudio, se evaluó la posible existencia de degradación de los modelos a partir de datos de la columna de separación tomando datos unos meses después de los utilizados para la construcción del modelo ensamblado. En este caso se encontró que tanto el modelo obtenido por Programación Kaizen como el obtenido por Procesos Gaussianos sufren una degradación en la estimación de la variable de salida.

Por último, se trabajó en la obtención de modelos para un sistema con varias variables de salida en una sola ejecución del algoritmo, de forma que los modelos resultantes puedan compartir términos. Esto permite reflejar en los modelos obtenidos los posibles fenómenos físico-químicos compartidos en un sistema. Se abordó el problema a través del diseño e implementación de dos nuevas estrategias para generar múltiples salidas, una en donde en el paso de PK de regresión lineal, se realiza una regresión lineal múltiple, y la segunda basada en el uso del modelo de islas para la paralelización de algoritmos evolutivos. Para analizar la influencia sobre la calidad de los modelos obtenidos de los términos en común de las variables de salida, se trabajó con cuatro esquemas de bases funcionales compartidas, desde el caso de nula intersección de bases entre las variables de salida, hasta una muy alta intersección entre las bases. Las dos estrategias fueron evaluadas en cinco familias de funciones derivadas de funciones de *benchmark* de regresión simbólica y en varios sistemas de pro-

cesos químicos, para cada uno de los esquemas de bases compartidas. Se realizó la comparación de ambas estrategias con los modelos obtenidos mediante la ejecución independiente del algoritmo de PK para una sola variable de salida, y con los modelos obtenidos por Procesos Gaussianos para múltiples salidas. Se encontró que para casi todos los escenarios considerados la estrategia de múltiples salidas basado en el modelo de islas construyó mejores modelos que las ejecuciones independientes de Programación Kaizen y que la estrategia por regresión lineal múltiple. A su vez, en comparación con los modelos obtenidos por Procesos Gaussianos, todos los modelos obtenidos en los cuatro esquemas de las funciones derivadas de funciones de referencia utilizando Programación Kaizen como base algorítmica resultaron en mejores estimadores de los valores de los conjuntos de validación. Sin embargo, para el caso del sistemas de procesos químicos, la diferencias en los valores de RMSE de los conjuntos de validación no fueron tan significativas como en los otros casos. De cualquier manera, los modelos obtenidos por la estrategia basada en el modelo de islas fue superior en la mayoría de los casos. Con respecto a la cantidad de términos compartidos, se encontró que los modelos obtenidos por S-PK son los que comparten la menor cantidad de términos. Por el contrario, los modelos obtenidos por MO-OLS son los que tienen mayor cantidad de términos en común, y los modelos obtenidos por MO-I presentan un comportamiento intermedio entre los otros dos algoritmos. Finalmente, se encontró que el tiempo de ejecución de las estrategias para considerar múltiples salidas en PK no solo son competitivas con el de PK de forma independiente, sino que el uso de la estrategia basada en el modelo de islas resultó en mejores tiempos de ejecución que los de PK de forma independiente para casi todos los escenarios.

A partir del trabajo realizado se identificaron tres lineas de trabajo a futuro que se consideran interesantes. En primer lugar se plantea como trabajo a futuro el uso de Programación Kaizen como herramienta para la construcción de modelos híbridos de sistemas de procesos químicos, permitiendo la inclusión de significado físico en el mecanismo de búsqueda de expresiones de PK, de forma de obtener modelos que integren los modelos más generales (modelos fenomenológicos con un gran costo computacional) y los modelos a partir de datos del proceso que son particulares del sistema de donde se toman los datos, y tienen un bajo costo computacional.

En segundo caso y a partir de los resultados obtenidos, se considera muy promisoria la aplicación algoritmo para múltiple salidas basado en modelo de

islas para la construcción de varios *soft-sensors* a partir de datos históricos reales, como por ejemplo sobre los datos de la columna de destilación de la refinería de ANCAP.

Por último, la tercera línea de trabajo consiste en el estudio de la estrategia de múltiples salidas basada en el modelo de islas en mayor profundidad. En particular, es interesante considerar otros mecanismos de migración de soluciones (en lugar de enviar todas las soluciones a cada isla), de forma de entender como impacta en la precisión de los modelos obtenidos. También resulta interesante realizar un estudio sobre la escalabilidad del algoritmo, es decir, como se ve afectada la calidad de los modelos y el tiempo de ejecución al aumentar la cantidad de variables de salida del sistema (lo que redunda en aumentar la cantidad de islas y la cantidad de migrantes en el algoritmo propuesto).

Bibliografía

Alba, E., y Luque, G. (2005). *Parallel Metaheuristics*. John Wiley; Sons, Ltd.

Austel, V., Cornelio, C., Dash, S., Goncalves, J., Horesh, L., Josephson, T. R., y Megiddo, N. (2020). Symbolic Regression using Mixed-Integer Nonlinear Optimization. *CoRR*, *abs/2006.06813*. https://arxiv.org/abs/2006.06813

Austel, V., Dash, S., Gunluk, O., Horesh, L., Liberti, L., Nannicini, G., y Schieber, B. (2017). Globally optimal symbolic regression. *31st Conference on Neural Information Processing Systems (NIPS 2017)*.

Caballero, J. A., y Grossmann, I. E. (2008). An algorithm for the use of surrogate models in modular flowsheet optimization. *AIChE Journal*, *54*(10), 2633-2650. https://doi.org/10.1002/aic.11579

Cai, W., Pacheco-Vega, A., Sen, M., y Yang, K. T. (2006). Heat transfer correlations by symbolic regression. *International Journal of Heat and Mass Transfer*, *49*(23-24), 4352-4359. https://doi.org/10.1016/j.ins.2016.08.081

Cozad, A., y Sahinidis, N. V. (2018). A Global MINLP Approach to Symbolic Regression. *Math. Program.*, *170*(1), 97-119. https://doi.org/10.1007/s10107-018-1289-x

Cozad, A., Sahinidis, N. V., y Miller, D. C. (2014). Learning surrogate models for simulation-based optimization. *AIChE Journal*, *60*(6), 2211-2227. https://doi.org/10.1002/aic.14418

Davis, S. E., Cremaschi, S., y Eden, M. R. (2018). Efficient Surrogate Model Development: Impact of Sample Size and Underlying Model Dimensions. En M. R. Eden, M. G. Ierapetritou y G. P. Towler (Eds.), *13th International Symposium on Process Systems Engineering (PSE 2018)* (pp. 979-984, Vol. 44). Elsevier. https://doi.org/https://doi.org/10.1016/B978-0-444-64241-7.50158-0

de Melo, V. V. (2014). Kaizen Programming. *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, 895-902. https://doi.org/10.1145/2576768.2598264

de Melo, V. V., y Banzhaf, W. (2018). Automatic feature engineering for regression models with machine learning: An evolutionary computation and statistics hybrid. *Information Sciences*, *430*, 287-313.

Derrac, J., García, S., Molina, D., y Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, *1*(1), 3-18. https://doi.org/10.1016/j.swevo.2011.02.002

Dürrenmatt, D. J., y Gujer, W. (2012). Automatic reactor model synthesis with genetic programming. *Water Science and Technology*, *65*(4), 765-772. https://doi.org/10.2166/wst.2012.913

Engle, M. R., y Sahinidis, N. V. (2022). Deterministic symbolic regression with derivative information: General methodology and application to equations of state. *AIChE Journal*, *68*(6), e17457. https://doi.org/https://doi.org/10.1002/aic.17457

Ferreira, J., Pedemonte, M., y Torres, A. I. (2019). A Genetic Programming Approach for Construction of Surrogate Models. En S. G. Muñoz, C. D. Laird y M. J. Realff (Eds.), *Proceedings of the 9th International Conference on Foundations of Computer-Aided Process Design* (pp. 451-456, Vol. 47). Elsevier. https://doi.org/https://doi.org/10.1016/B978-0-12-818597-1.50072-2

Ferreira, J., Pedemonte, M., y Torres, A. I. (2021a). A Machine Learning Approach for the Design of a Soft Sensor in an Oil Refinery's Distillation Column. *AIChE Annual Meeting*.

Ferreira, J., Pedemonte, M., y Torres, A. I. (2021b). Aprendizaje automático de modelos en la industria de procesos. *Encuentro Regional de Ingeniería Química*.

Ferreira, J., Torres, A. I., y Pedemonte, M. (2019). A Comparative Study on the Numerical Performance of Kaizen Programming and Genetic Programming for Symbolic Regression Problems. *2019 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, 1-6. https://doi.org/10.1109/LA-CCI47412.2019.9036755

Ferreira, J., Torres, A. I., y Pedemonte, M. (2021). Towards a Multi-Output Kaizen Programming Algorithm. *2021 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, 1-6. https://doi.org/10.1109/LA-CCI48322.2021.9769841

Ferreira, J., Torres, A. I., y Pedemonte, M. (2022). A Kaizen Programming algorithm for multi-output regression based on a heterogeneous island model. *En proceso de revisión en Neural Computing and Applications*.

Ferreira, J., Pedemonte, M., y Torres, A. I. (2022a). Development of a machine learning-based soft sensor for an oil refinery's distillation column. *Computers & Chemical Engineering*, *161*, 107756. https://doi.org/https://doi.org/10.1016/j.compchemeng.2022.107756

Ferreira, J., Pedemonte, M., y Torres, A. I. (2022b). A multi-output machine learning approach for generation of surrogate models in process engineering. En Y. Yamashita y M. Kano (Eds.), *14th International Symposium on Process Systems Engineering* (pp. 1771-1776, Vol. 49). Elsevier. https://doi.org/https://doi.org/10.1016/B978-0-323-85159-6.50295-5

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., y Gagné, C. (2012). DEAP: Evolutionary Algorithms Made Easy. *Journal of Machine Learning Research*, *13*, 2171-2175.

Frank, I. E., y Friedman, J. H. (1993). A Statistical View of Some Chemometrics Regression Tools. *Technometrics*, *35*(2), 109-135. http://www.jstor.org/stable/1269656

Garud, S. S., Karimi, I. A., y Kraft, M. (2017). Design of computer experiments: A review. *Computers & Chemical Engineering*, *106*, 71-95. https://doi.org/10.1016/j.compchemeng.2017.05.010

Goodfellow, I., Bengio, Y., y Courville, A. (2016). *Deep learning*. MIT press.

Harada, T., y Alba, E. (2020). Parallel Genetic Algorithms: A Useful Survey. *ACM Comput. Surv.*, *53*(4). https://doi.org/10.1145/3400031

Henao, C. A., y Maravelias, C. T. (2011). Surrogate-based superstructure optimization framework. *AIChE Journal*, *57*(5), 1216-1232. https://doi.org/10.1002/aic.12341

Hinchliffe, M. P., y Willis, M. J. (2003). Dynamic systems modelling using genetic programming. *Computers & Chemical Engineering*, *27*(12), 1841-1854. https://doi.org/10.1016/j.compchemeng.2003.06.001

Kadlec, P., Gabrys, B., y Strandt, S. (2009). Data-driven Soft Sensors in the process industry. *Computers & Chemical Engineering*, *33*(4), 795-814. https://doi.org/10.1016/j.compchemeng.2008.12.012

Kaneko, H., y Funatsu, K. (2014). Adaptive soft sensor based on online support vector regression and Bayesian ensemble learning for various states in chemical plants. *Chemometrics and Intelligent Laboratory Systems*, *137*, 57-66. https://doi.org/10.1016/j.chemolab.2014.06.008

Keijzer, M. (2003). Improving Symbolic Regression with Interval Arithmetic and Linear Scaling. En C. Ryan, T. Soule, M. Keijzer, E. Tsang, R. Poli y E. Costa (Eds.), *Genetic Programming* (pp. 70-82). Springer Berlin Heidelberg.

Korns, M. F. (2011). Accuracy in Symbolic Regression. En R. Riolo, E. Vladislavleva y J. H. Moore (Eds.), *Genetic Programming Theory and Practice IX* (pp. 129-151). Springer New York. https://doi.org/10.1007/978-1-4614-1770-5_8

Koza, J. R. (1992). *Genetic Programming : On the Programming of Computers By Means of Natural Selection Complex Adaptive Systems*. MIT Press.

Krige, D. G. (1952). A Statistical Approach to Some Basic Mine Valuation Problems on the Witwatersrand. *Journal of the Chemical, Metallurgical and Mining Society of South Africa*, 201-215. https://doi.org/10.2307/3006914

Miller, J. F. (2020). Cartesian genetic programming: its status and future. *Genetic Programming and Evolvable Machines*, *21*, 129-168. https://doi.org/10.1007/s10710-019-09360-6

Nguyen, Q. U., Hoai, N., O'Neill, M., McKay, R., y Galván-López, E. (2011). Semantically-based crossover in genetic programming: Application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines*, *12*, 91-119.

Nielsen, R. F., Nazemzadeh, N., Sillesen, L. W., Andersson, M. P., Gernaey, K. V., y Mansouri, S. S. (2020). Hybrid machine learning assisted modelling framework for particle processes. *Computers & Chemical Engineering*, *140*, 106916. https://doi.org/https://doi.org/10.1016/j.compchemeng.2020.106916

Pedemonte, M., Luna, F., y Alba, E. (2018). A theoretical and empirical study of the trajectories of solutions on the grid of Systolic Genetic Search. *Information Sciences*, *445*, 97-117.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., y Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825-2830.

Pistikopoulos, E. N., Barbosa-Povoa, A., Lee, J. H., Misener, R., Mitsos, A., Reklaitis, G. V., Venkatasubramanian, V., You, F., y Gani, R. (2021). Process systems engineering – The generation next? *Computers & Chemical Engineering*, *147*, 107252. https://doi.org/https://doi.org/10.1016/j.compchemeng.2021.107252

Poli, R., Langdon, W. B., y McPhee, N. F. (2008). *A Field Guide to Genetic Programming*. Lulu Enterprises, UK Ltd.

Reis, M. S., y Saraiva, P. M. (2021). Data-centric process systems engineering: A push towards PSE 4.0. *Computers & Chemical Engineering*, *155*, 107529. https://doi.org/https://doi.org/10.1016/j.compchemeng.2021.107529

Schweidtmann, A. M., Huster, W. R., Lüthje, J. T., y Mitsos, A. (2019). Deterministic global process optimization: Accurate (single-species) properties via artificial neural networks. *Computers & Chemical Engineering*, *121*, 67-74. https://doi.org/https://doi.org/10.1016/j.compchemeng.2018.10.007

Shang, C., Yang, F., Huang, D., y Lyu, W. (2014). Data-driven soft sensor development based on deep learning technique. *Journal of Process Control*, *24*(3), 223-233. https://doi.org/10.1016/j.jprocont.2014.01.012

Smola, A., y Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, *14*, 199-222. https://doi.org/10.1023/B%3ASTCO.0000035301.49549.88

Surjanovic, S., y Bingham, D. (2013). Virtual Library of Simulation Experiments: Test Functions and Datasets [Último acceso: Setiembre, 2022]. https://www.sfu.ca/~ssurjano/optimization.html

Willis, M., Hiden, H., Hinchliffe, M., McKay, B., y Barton, G. (1997). Systems Modelling using Genetic Programming. *Computers & Chemical Engineering*, *21*, 1161-1166. https://doi.org/10.1016/j.aap.2015.04.001

Zaghloul, M. S., Hamza, R. A., Iorhemen, O. T., y Tay, J. H. (2020). Comparison of adaptive neuro-fuzzy inference systems (ANFIS) and support vector regression (SVR) for data-driven modelling of aerobic granular

sludge reactors. *Journal of Environmental Chemical Engineering*, *8*(3), 103742. https://doi.org/https://doi.org/10.1016/j.jece.2020.103742

Zhang, S., Wang, F., He, D., y Jia, R. (2013). Online quality prediction for cobalt oxalate synthesis process using least squares support vector regression approach with dual updating. *Control Engineering Practice*, *21*(10), 1267-1276. https://doi.org/10.1016/j.conengprac.2013.06.002

Zhu, W., Chebeir, J., y Romagnoli, J. A. (2020). Operation optimization of a cryogenic NGL recovery unit using deep learning based surrogate modeling. *Computers & Chemical Engineering*, *137*, 106815. https://doi.org/https://doi.org/10.1016/j.compchemeng.2020.106815

APÉNDICES

Apéndice 1

Conexión Python - Simuladores de Aspen

En el presente anexo se presentan una breve explicación de la conexión de `Python` con Aspen Plus y Aspen Hysys, de forma de ejecutar los archivos de Aspen de forma automatizada (sin utilizar la interfaz de Aspen) a través de `Python`.

1.1. Conexión Python - Aspen Plus

Para la conexión de Aspen Plus a través de Python se debe utilizar el paquete `win32`. Utilizando la función `InitFromArchive2` se carga el archivo de simulación de respaldo (.bkp). A través de esta conexión se puede realizar la copia de datos de la simulación al espacio de trabajo de Python, guardar valores en el archivo de simulación de Aspen Plus y realizar la simulación a través de Python con o sin la interfaz gráfica de Aspen Plus.

Para asignar datos en la simulación o recabar los datos de una simulación se necesita conocer los nombres de las rutas donde se encuentra cada variable a modificar o a obtener dato. Las rutas para cada caso se pueden buscar a través de la pestaña *Customize* de Aspen Plus en *Variable explorer*, en donde están disponible la ruta de cada variable y parámetro del archivo de simulación.

En el siguiente fragmento de código en `Python` se presenta un ejemplo con las 3 acciones posibles de realizar en el archivo *Sim* de Aspen Plus. La base de la conexión Python-Aspen Plus se puede encontrar en url: https://kitchingroup.cheme.cmu.edu/blog/tag/aspen/.

```
#Se entrelaza y se carga el archivo de simulacion
import win32com.client as win32
aspen = win32.Dispatch('Apwn.Document')
aspen.InitFromArchive2('Sim.bkp')

#Se guardan valores en el archivo de simulacion
# Se guarda el valor 1 en la relacion de reflujo del bloque
    RECTIFY
aspen.Tree.FindNode("\Data\Blocks\RECTIFY\Input\BASIS_RR").
    Value= valor1
# Se guarda el valor 2 (dtype=int) en la cantidad de etapas del
     bloque RECTIFY
aspen.Tree.FindNode("\Data\Blocks\RECTIFY\Input\\NSTAGE").Value
     = float(valor2)

#Se ejecuta la simulacion
aspen.Engine.Run2()

#Se obtienen los resultados de la simulacion
# Se guarda en Frac_ETOH la fraccion molar de EtOH en la salida
Frac_ETOH= aspen.Tree.FindNode('\Data\Streams\Salida\Output\
    MASSFRAC\MIXED\ETOH').Value
# Se guarda en F_ETOH el flujo masico de EtOH en la salida
F_ETOH=aspen.Tree.FindNode('\Data\Streams\Salida\Output\
    MASSFLOW\MIXED\ETOH').Value
```

1.2. Conexión Python - Aspen Hysys

La conexión de Python-Aspen Hysys se realiza, así como con Aspen Plus, a través de la COM interface. En este caso tambien se debe utilizar el paquete `win32`.

Así como con Aspen Plus, para asignar datos en la simulación o recabar los datos de una simulación se necesita conocer los nombres de las rutas donde se encuentra cada variable a modificar o a obtener dato. En este caso se puede encontrar las variables a utilizar a través de la inspección del diccionario del objeto asociado al *Flowsheet* en `Python` o en el objeto análogo obtenido de la conexión de Excel-Aspen Hysys. Cabe destacar que en la conexión de `Python` con Aspen Hysys, los valores obtenidos o seteados de una simulación se deben hacer con unidades.

En el siguiente fragmento de código en `Python` se presenta un ejemplo con algunas acciones posibles a realizar en el archivo *runHysys* de Aspen Hysys.

```
from win32com.client import Dispatch
hysysApp = Dispatch("HYSYS.Application.V9.0")
# hysysApp.Visible = True
hysysCase = hysysApp.SimulationCases.Open(Ruta+"\\runHysys.hsc"
    )
mainFlowsheet = hysysCase.Flowsheet
WaterFeed.ComponentMassFlow.SetValues([0,0,x0[pos]],"kg/h")
Feed.ComponentMolarFlow.SetValues([x1[pos],0,0],"kgmole/h")
TempReact.Temperature.SetValue(x2[pos],"C")

Glycol = mainFlowsheet.MaterialStreams.Item("Glycol")
RecyProds = mainFlowsheet.MaterialStreams.Item("RecyProds")
print("The mol flow in RecyProds is", round(RecyProds.MolarFlow
    .GetValue("kgmole/h"), 2), "kmol/h")
print("The mass flow of water in RecyProds is", round(RecyProds
    .ComponentMassFlow.GetValues("kg/h")[2], 2), "kg/h")

hysysApp.quit()
```

Apéndice 2

Pruebas estadísticas

En este anexo se presentan las cinco pruebas estadísticas utilizadas a lo largo de la tesis.

El algoritmo de Programación Kaizen utiliza como parte de su funcionamiento la prueba t de *Student* para identificar los individuos relevantes en los modelos hallados en cada iteración.

Por otra parte, debido a la utilización de algoritmos aleatorizados, la comparación de desempeño de los mismos se debe realizar utilizando un conjunto de ejecuciones de los algoritmos. Para comparar las distribuciones de desempeño de los algoritmos se pueden utilizar pruebas estadísticas paramétricas o no paramétricas. Debido a que en las segundas no es necesario asumir ninguna suposición sobre la distribución de los datos a utilizar en la prueba, es usual el uso de este tipo de pruebas estadísticas para comparar el desempeño entre dos o más algoritmos aleatorizados. Según Alba y Luque (2005), para aplicar las puebas estadísticas sobre los datos de desempeño de algoritmos aleatorizados se debe utilizar conjuntos de por lo menos treinta muestras.

Dependiendo de la cantidad de algoritmos considerados en la evaluación experimental y la dependencia de los datos de las distribuciones a comparar, se recomienda el uso de una prueba para cada caso. En Alba y Luque (2005) se recomienda el uso de la prueba de Friedman para comparar más de dos algoritmos cuando las muestras son a partir de datos dependientes, por ejemplo, utilizar los mismos casos en cada algoritmo. En el caso de comparar solamente dos algoritmos o como prueba *pos-hoc* (prueba que se realiza luego que la prueba que compara más de dos distribuciones devuelve que no hay igualdad estadística sobre las distribuciones), también con datos dependientes, se re-

comienda el uso de la prueba de los rangos con signo de Wilcoxon. Por otro lado, cuando los datos son independientes, para la comparación de más de dos algoritmos se recomienda el uso de la prueba de Kruskal-Wallis, y para comparar solamente dos algoritmos o como prueba *pos-hoc* se recomienda uso de la prueba U de Mann-Whitney. En la presente tesis se utilizan las cuatro pruebas estadísticas según el caso.

En las siguientes secciones se presenta un breve resumen de cada una de las pruebas estadísticas utilizadas. Para el caso de de la prueba de t de *Student* los conjuntos de datos son las evaluaciones de cada individuo de la regresión lineal para cada uno de los datos de variable de entrada. En los otros casos, el conjunto de datos de cada algoritmo, esta compuesto por los valores de RMSE del conjunto de validación para cada una de las ejecuciones del mismo.

2.1. Prueba t de *Student*

En la presente tesis la prueba t de *Student* es utilizada para evaluar si los coeficientes de la regresión lineal tienen significancia estadística diferentes a cero.

La prueba t de *Student* considera como hipótesis nula (H_0) que el coeficiente estimado por la regresión lineal de un término i, β_i, es 0, y como hipótesis alterna que el coeficiente β_i es distinto a 0, i varía entre 1 y m.

Para realizar la prueba se utiliza el estadístico t. A partir de de Melo y Banzhaf (2018), el estadístico para cada coeficiente se calcula como $t_i = \frac{\beta_i}{se_i}$, donde se_i se calcula en base a los datos (x, y), utilizados para la regresión, siendo $se_i = \sqrt{\hat{\sigma}^2 (F^T F)^{-1}}$, con $\hat{\sigma}^2 = \frac{\sum_{j=1}^{n} (y_j - \hat{y}_i)^2}{n-m}$, $F(ij)$ es la evaluación de cada término i para cada valor de dato x_j, n es la cantidad de datos, e $\hat{y}$ son los valores estimados por la regresión.

El método supone que t se aproxima a una distribución t de *Student* con $n - m$ grados de libertad. Por último, se calcula (o se busca en tablas) el valor p correspondiente al estadístico t considerando los grados de libertad. Si $p \geq \alpha$ se acepta la hipótesis nula, y se concluye que β_i es 0 y el término considerado se puede descartar. Un valor usual para la significancia α es 0.05, que se corresponde con el 95 % de significancia.

2.2. Prueba de Friedman

La prueba de Friedman tiene como H_0 que las medianas de las distribuciones de los conjuntos de datos son iguales. Por lo tanto, con esta prueba se responde la pregunta de si dado k algoritmos, con $k \geq 2$, existen por lo menos dos algoritmos cuyos conjunto de datos de desempeño presentan diferentes valores de mediana.

Como primer paso del método se debe ordenar, de forma ascendente, todos los datos manteniendo la etiqueta del conjunto del que provienen. Luego se le asigna el rango a cada dato, de forma que el primero tiene el rango 1, el segundo el rango 2, y así sucesivamente. Si existen empates entre los datos, es usual utilizar como rango el promedio del intervalo de rangos correspondiente; por ejemplo, si existe empate entre los valores que corresponden al rango 3, 4 y 5, se utiliza el rango 4 para los tres datos considerados. Seguidamente, para cada conjunto de datos, se realiza el promedio de los rangos de los datos, obteniendo el valor de $R_j = 1/n \sum r_i^j$, en donde $j = 1, 2, ..., k$, siendo k es la cantidad de algoritmos, n la cantidad de datos en cada conjunto j, e $i = 1, 2, ...,$ cantidad de datos en el conjunto j.

A partir de los valores R_j, se calcula el estadístico Q como
$Q = \frac{12n}{k(k+1)} \left[\sum j R_j^2 - \frac{k(k+1)^2}{4} \right]$.

El método supone que Q se aproxima a una distribución χ^2 con $k-1$ grados de libertad. Por último, se calcula (o buscado en tablas) el valor p correspondiente al estadístico Q considerando los grados de libertad $k-1$. Si $p \geq \alpha$ se acepta la hipótesis nula, y se concluye que que no existe diferencia estadística entre las medianas.

En la presente tesis se utilizó la implementación de la prueba de Friedman disponible en el paquete `scipy.stats.friedmanchisquare` en `Python`.

2.3. Prueba de Kruskal-Wallis

La prueba de Kruskal-Wallis, también llamada prueba H de Kruskal-Wallis, es la prueba análoga a la prueba de Friedman, pero su aplicación es sobre conjunto de datos independientes.

La prueba comienza de la misma forma que la prueba de Friedman, se ordenan todos los datos, se les asigna un rango a cada dato, y se calcula el valor de R_j para cada conjunto de datos. El estadístico de la prueba se calcula

como $H = \frac{12}{N(N+1)} \sum j \frac{R_j^2}{n_j} - 3(N+1)$, con N la cantidad de datos totales y n es la cantidad de datos en cada conjunto.

Así como en la prueba de Friedman, H se aproxima a una distribución χ^2 con $k-1$ grados de libertad. Luego de calculado p, si $p \geq \alpha$ se acepta la hipótesis nula, y se concluye que que no existe diferencia estadística entre las medianas.

En la presente tesis se utilizó la implementación de la prueba de Kruskal-Wallis disponible en el paquete `scipy.stats.kruskal` en `Python`.

2.4. Prueba de los rangos con signo de Wilcoxon

En el caso de la prueba con signo de Wilcoxon, H_0 indica que en el conjunto de diferencias entre pares estudiado, la mediana de las diferencias es igual a 0. Por el contrario, la hipótesis alterna indica que la mediana de las diferencias no es igual a 0.

Para la prueba de Wilcoxon es necesario utilizar los datos de los conjuntos de a pares. Como primer paso, se emparejan los datos según de donde provengan; por ejemplo, en el caso de comparación de algoritmos, se emparejan los datos de desempeños de dos algoritmos si provienen de resolver el problema bajo las mismas condiciones. Luego, se calcula la diferencia de cada pareja de datos. Seguidamente se ordenan de menor a mayor según su valor absoluto. Para cada diferencia se le asigna un rango según el orden (si existen empates, se realiza la misma operación de ajuste que en las pruebas anteriores). Las diferencias iguales a cero se eliminan del análisis. A continuación, a cada rango se le asigna el signo de la diferencia.

Como siguiente paso, se calculan las sumas de las diferencias positivas (R^+), y las negativas (R^-). Luego, se calcula T como el mínimo valor entre R^+ y R^-. El estadístico de esta prueba (z_T) se calcula como $z_T = \frac{(T-\mu_T)}{\sigma_T}$,
con $\mu_T = n(n+1)/4$, $\sigma_T = \sqrt{\frac{n(n+1)(n+2)}{24}}$, siendo n la cantidad de datos de cada conjunto de datos.

El método supone que z_T se aproxima a una distribución Normal con media 0 y desviación estándar 1. A partir del valor de z_T y las suposiciones de la distribución se calcula el valor de p. Así como en las pruebas anteriores, si $p \geq \alpha$ se acepta la hipótesis nula, y se concluye que que no existe diferencia

estadística entre las medianas de las distribuciones de datos.

En la presente tesis se utilizó la implementación de la prueba de los rangos con signo de Wilcoxon disponible en el paquete `scipy.stats.wilcoxon` en `Python`.

2.5. Prueba U de Mann-Whitney

La prueba U de Mann-Whitney, también conocida como la prueba de Mann–Whitney–Wilcoxon, tiene como H_0 la misma que en la prueba de los rangos con signo de Wilcoxon.

De la misma forma que en la prueba Friedman, se ordenan todos los datos, se les asigna un rango, y se calcula el valor de R_j para cada conjunto de datos. Luego se calculan los valores de U_1 y U_2 como: $U_1 = n_1 n_2 + n_1(n_1 + 1)/2 - R_1$ y $U_2 = n_1 n_2 + n_2(n_2 + 1)/2 - R_2$, donde n_1 es la cantidad de datos en el conjunto de datos 1, y n_2 es la cantidad en el conjunto de datos 2. Por último, el estadístico U es calculado como el mínimo entre U_1 y U_2.

El método supone que U se aproxima a una distribución Normal con media 0 y desviación estándar 1. A partir del valor de U y las suposiciones de la distribución se calcula el valor de p. Así como en las pruebas anteriores, Si $p \geq \alpha$ se acepta la hipótesis nula, y se concluye que que no existe diferencia estadística entre las medianas de las distribuciones de datos.

En la presente tesis se utilizó la implementación de la prueba U de Mann-Whitney disponible en el paquete `scipy.stats.mannwhitneyu` en `Python`.