Proyecto de Grado

DataFish

Técnicas de Data Mining aplicadas a la industria pesquera

Patricia Castro
Ignacio Duarte
Marcelo Fontan

Tutor: Omar Viera

Departamento de Investigación Operativa
Instituto de Computación
Facultad de Ingeniería
Universidad de la República
Marzo de 2005

Resumen

Este documento reporta el trabajo desarrollado en el contexto de Proyecto de Grado (Taller V) de la Facultad de Ingeniería de la Universidad de la Republica. El objetivo principal es estudiar distintas técnicas de Data Mining y realizar una aplicación en un problema particular. El problema resuelto es el de facilitar el análisis visual del comportamiento de la flota pesquera uruguaya, planteado por la Dirección Nacional de Recursos Acuáticos (DINARA).

Se realiza un estado de arte de las distintas técnicas de Data Mining profundizando en uno de los temas de interés para el problema planteado como lo son los algoritmos de clusterización y clasificación especializados en datos espaciales.

Luego de un análisis de los algoritmos especializados en bases de datos espaciales se decide implementar DBRS, DBSCAN y OPTICS; el equipo además desarrolla dos algoritmos propios.

Atendiendo los requerimientos de la DINARA se desarrolla una aplicación con interfase Web integrada totalmente con el esquema de datos, el Sistema de Información Geográfico y el ambiente propuesto por el cliente, la cual facilita el análisis de información visual que se desea.

La aplicación desarrollada esta orientada a la solución del problema planteado por la DINARA, para esto se desarrolla una librería de algoritmos de uso genérico, la cual puede ser utilizada independientemente en otra aplicación. El diseño de este componente permite que el mismo pueda ser ampliado con nuevos algoritmos o inclusive con mejoras a los ya existentes.

Contenido

1. Prefacio

1.1. Contexto del documento

El presente documento es el Informe Final del Proyecto de Grado de la carrera de Ingeniería en Computación de la Facultad de Ingeniería de la Universidad de la República.

En particular el área de trabajo elegida, Data Mining, es de interés del Departamento de Investigación Operativa el cual pretende estudiar y utilizar estas herramientas desde el punto de vista de usuario.

Por otra parte la Dirección Nacional de Recursos Acuáticos (DINARA), presenta una problemática que se resuelve en este proyecto aplicando este tipo de técnicas en particular Clustering y Clasificación (ver Capítulo cuatro y cinco).

El objetivo del informe es presentar y describir el proceso y el trabajo llevado a cabo para la realización del Proyecto de Grado denominado ***DataFish***. Se detallan los objetivos del proyecto, la solución alcanzada y los resultados obtenidos junto a las dificultades encontradas y la forma en que fueron tratadas.

Este documento esta dirigido tanto a docentes como a estudiantes, con la expectativa de que el lector adquiera un claro entendimiento del proyecto realizado y que resulte útil como fuente de consulta para otros proyectos de características similares.

La documentación completa del proyecto está organizada en un informe central (este documento) y varios anexos. Esto es debido a que el alcance y componentes del trabajo son demasiado amplios, y la elaboración de un documento único daría como resultado un informe de complicada lectura. Además algunos anexos pueden ser fuente de consulta independientemente del contexto de este proyecto como por ejemplo los estados del arte.

1.2. Organización del documento

En el Capítulo dos se introduce el problema resuelto en este proyecto, las motivaciones y los objetivos planteados en el mismo.

El Capítulo tres detalla fase a fase el desarrollo del proyecto, describiéndolo más allá del problema específico resuelto, mostrando el trabajo realizado y las distintas decisiones que se tomaron y pautaron el transcurso del mismo. La lectura de este capítulo puede ser obviada por las personas que no estén interesadas en conocer el desarrollo particular de este proyecto.

Los Capítulos del cuatro al seis refieren a la solución del problema planteado por la DINARA. En el Capítulo cuatro se puede ver un resumen de los requerimientos principales, y en el cinco se describen las decisiones de diseño e implementación. El Capítulo seis explica la implementación de la aplicación.

En el Capítulo siete se muestran y explican las pruebas realizadas del sistema tanto de funcionalidad como de performance.

Los Capítulos ocho y nueve están dedicados al análisis de resultados y conclusiones respectivamente.

Las ideas de posible trabajos futuros tanto sobre la aplicación desarrollada como inquietudes planteadas por la DINARA se describen en el Capítulo diez.

Por último en los Capítulos once, doce y trece se enumeran las Referencias, Abreviaciones y Terminología y los Anexos disponibles con este informe.

2. Introducción

Este proyecto surge del interés del Departamento de Investigación Operativa en estudiar el uso de técnicas de Data Mining para su área de trabajo desde el punto de vista de usuario. Para esto se solicita como parte de este taller, por un lado la realización de un Estado del Arte sobre técnicas de Data Mining, y luego la aplicación de alguna de ellas a un caso concreto, tema que se definiría en una etapa posterior.

A la vista de los desafíos planteados aparece una clara separación de etapas, para cada una de ellas se definen objetivos particulares.

En la primera etapa se buscan dos objetivos, el primero es realizar un documento donde se refleje el estado actual de las distintas técnicas de Data Mining, y de esta forma alcanzar un nivel de conocimiento general de las diferentes áreas de aplicación, algoritmos y herramientas disponible. Para esto se realiza una investigación sobre las distintas técnicas existentes y se elabora el documento Estado del Arte **[3]**.

El segundo objetivo de la primera etapa es definir los distintos procedimientos para llevar adelante la administración del proyecto, para ello se definen la metodología **[2]** y el plan de trabajo **[1]** y otros documentos de registración y seguimiento de actividades (Planilla de horas, Minutas de reuniones)

El objetivo de la segunda etapa del proyecto es resolver un problema específico aplicando técnicas de Data Mining. La concretización del problema se realiza por dos vías, buscando problemas abiertos posibles de ser resueltos con técnicas de Data Mining y buscando posibles clientes con requerimientos concretos aplicables en ésta área.

Luego de hacer la evaluación de algunos problemas y varios clientes **[4]**, se decide que como proyecto académico es de interés agregar la interacción con un cliente puntual, y dentro de los clientes evaluados la problemática de realización más factible es la planteada por la DINARA.

Una vez seleccionado el campo de aplicación se estudia y define el problema a resolver, se elige Clustering como la técnica de Data Mining más adecuada para este caso. Encontrándose necesario hacer una especialización en el estudio de dicha técnica, es entonces que se realiza el documento Extensión del Estado del Arte **[5]**. De este trabajo se desprende que de los algoritmos especializados en la problemática planteada se implementarán DBSCAN **[22]**, OPTICS **[20]** y DBRS **[15]**. Además el equipo decide implementar un algoritmo de clasificación simple a los efectos de corroborar los resultados de los otros, DataFishClassificator (DFC). Este último da origen a un segundo algoritmo DataFishClassificatorEmpowered (DFCE) que utiliza conceptos de clustering.

Para la resolución de la problemática planteada por la DINARA se desarrolla el sistema DataFish.

Sistema *DataFish*

Con el fin de satisfacer la necesidad de mejorar la capacidad de información que tienen los datos con los que cuenta la DINARA, es que surge el sistema ***DataFish*** como herramienta de Data Mining visual para ayudar en la toma de decisiones de índole científica o jurídica.

Definición del problema
A efectos de realizar el seguimiento de la flota pesquera Uruguaya y validar la información estadística recabada de la flota pesquera comercial, la DINARA ha desarrollado un Sistema de Información Pesquera Satelital (SIPESAT).

Dicho sistema toma como fuente de información los datos enviados desde la baliza de monitoreo satelital instalada en cada pesquero, que son recibidos y almacenados en un esquema de datos creado exclusivamente para este fin.

Para una explicación más detallada de la realidad y problemática se pude ver el documento Descripción de Realidad **[10]**.

Si bien el sistema SIPESAT permite realizar algunas consultas y visualizaciones, no explota en toda su dimensión la capacidad de información que tienen los datos con los que se cuenta. Uno de los principales objetivos de la DINARA es mejorar la capacidad de información.

Se establecieron objetivos primarios y secundarios, dentro de los objetivos primarios están el de generar dinámicamente vistas de las señales que cumplan determinados criterios como ser flota a la cual pertenecen, fecha etc. y el de visualizar el esfuerzo pesquero por área también con la posibilidad de dinámicamente solicitarlo por distintos criterios.

Como objetivos secundarios se encuentran la configuración de alarmas de zonas de pesca prohibidas y de alarmas de proximidad a puerto, la visualización del histórico de esas alarmas y la cuantificación del esfuerzo pesquero.

Resultados

Se obtiene como resultado de este trabajo un documento de referencia (Estado del Arte **[3]**) con un resumen de las distintas técnicas de Data Mining y puede ser usado como una guía introductoria a las mismas independientemente del área de aplicación.

El documento Extensión del Estado del Arte **[5]** aporta un estudio en profundidad de los algoritmos de clustering especializados en bases de datos espaciales.

El sistema ***DataFish*** resuelve el problema planteado por la DINARA, mejorando la capacidad de análisis del comportamiento de la flota pesquera. Brindando la posibilidad de hacer un análisis visual de los datos para la posterior toma de decisiones.

DataFish brinda una interfaz Web para ejecutar los algoritmos y seleccionar distintos tipos de datos, integrándose con el Sistema de Información Geográfico para la visualización de los resultados.

Este trabajo agrega una comparación tanto en performance como en funcionalidad de los algoritmos clásicos (DBRS **[15]**, DBSCAN **[22]**, OPTICS **[20]**) y propios (DFCE) utilizados para la solución del problema planteado. La cual puede ser de utilidad para otros proyectos que pretendan utilizar estos algoritmos. Se considera especialmente importante este aporte ya que en la etapa de investigación no se encontró una comparación de este estilo entre estos tres algoritmos.

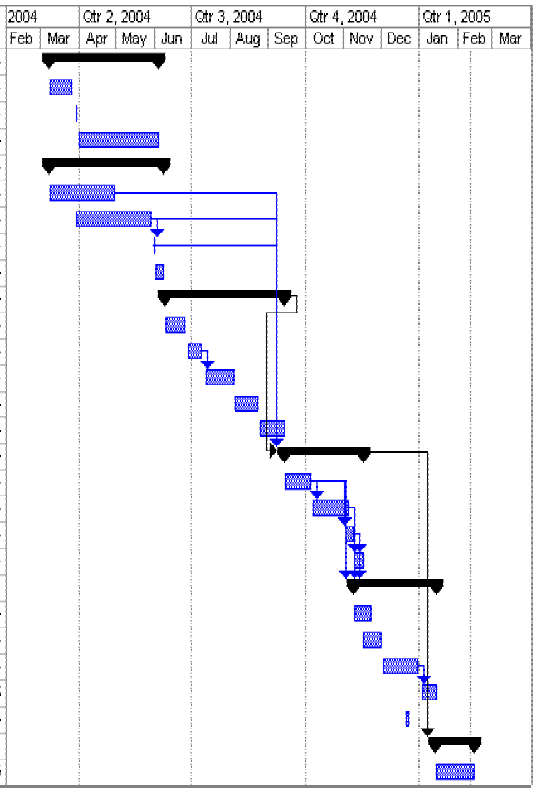
3. Resumen del proyecto

En esta sección se resume el desarrollo del proyecto describiendo brevemente sus fases y las características de las mismas durante su ejecución.

Este capítulo esta orientado a aquellas personas que estén interesadas en el desarrollo particular de este proyecto, en él se comentan las distintas etapas por las cuales atravesó el mismo, pudiendo no ser de interés para el lector focalizado exclusivamente en la aplicación de técnicas de Data Mining. Por lo que puede ser obviado si el principal interés es conocer acerca de la aplicación de técnicas de Data Mining.

El siguiente es el diagrama Gannt de las tareas ejecutadas durante el desarrollo del proyecto.

ID	Task Name	Start	Finish
1	**Fase 0 - Definicion de planes y metodologias**	**Mon 3/8/04**	**Thu 6/3/04**
2	Definición de metodología	Mon 3/8/04	Fri 3/26/04
3	Definición de Plan de Trabajo	Mon 3/29/04	Mon 3/29/04
4	Aprobacion del tutor	Wed 3/31/04	Thu 6/3/04
5	**Fase I - Investigación y Estado Del Arte**	**Mon 3/8/04**	**Tue 6/8/04**
6	Investigación	Mon 3/8/04	Fri 4/30/04
7	Elaboración del Estado del Arte	Mon 3/29/04	Fri 5/28/04
8	Entrega Estado del Arte	Mon 5/31/04	Mon 5/31/04
9	Aprobacion del tutor	Tue 6/1/04	Tue 6/8/04
10	**Fase II - Busqueda campo aplicacion**	**Wed 6/9/04**	**Mon 9/13/04**
11	Localizacion de un posible cliente	Wed 6/9/04	Fri 6/25/04
12	Estudio de situacion del cliente	Mon 6/28/04	Fri 7/9/04
13	Definicion del problema con cliente	Mon 7/12/04	Tue 8/3/04
14	Adaptacion del problema del cliente al Proyecto	Wed 8/4/04	Tue 8/24/04
15	Actualizacion de Estado del arte	Wed 8/25/04	Mon 9/13/04
16	**Fase III - Definicion del Problema y Especificación**	**Tue 9/14/04**	**Wed 11/17/04**
17	Análisis de requerimientos	Tue 9/14/04	Tue 10/5/04
18	Prototipación y especificación de requerimientos	Wed 10/6/04	Fri 11/5/04
19	Estudio y comparación de posibles herramientas de sw	Tue 11/2/04	Mon 11/8/04
20	Aprobacion del tutor y del cliente	Tue 11/9/04	Wed 11/17/04
21	**Fase IV - Diseño e Implentación de la solución**	**Tue 11/9/04**	**Fri 1/14/05**
22	Especificación de Arquitectura y Diseño de la solución	Tue 11/9/04	Mon 11/22/04
23	Prototipación e implementación de la solución	Tue 11/16/04	Tue 11/30/04
24	Analisis de resultados	Thu 12/2/04	Fri 12/31/04
25	Testeo	Mon 1/3/05	Fri 1/14/05
26	Aprobación de prototipo e implementación	Tue 12/21/04	Fri 12/24/04
27	**Fase V - Entrega y Presentación**	**Fri 1/14/05**	**Mon 2/14/05**
28	Preparación de entrega	Fri 1/14/05	Mon 2/14/05



Como se aprecia en el diagrama Gannt, el proyecto se dividió en las siguientes seis fases:

- Fase 0 – Definición de planes y metodologías
- Fase I – Investigación y Estado Del Arte
- Fase II – Búsqueda campo aplicación
- Fase III – Definición del Problema y Especificación
- Fase IV – Diseño e Implementación de la solución
- Fase V – Entrega y Presentación

A lo largo de todas las fases se realizan minutas y se registra en planillas Excel el tiempo invertido en cada tarea. Al finalizar cada una de las fases se elabora un análisis del esfuerzo realizado **[8]** durante la misma, buscando con dicho análisis detectar tempranamente problemas para una mejor toma de decisiones y así evitar un futuro problema similar.

3.1. Fase 0 – Definición de planes y metodologías

Esta es una fase en la que se definen las formas de trabajo y se planifica como se llevan a cabo las distintas etapas contenidas en el proyecto.

En esta fase se preparan los documentos "Plan de Trabajo" **[1]** y "Metodología de trabajo" **[2]**. En el primero se especifica la duración de cada tarea comprendida en cada fase así como un detalle de cada una y en el segundo se encuentra la descripción de la metodología utilizada para el desarrollo del proyecto entendiendo por metodología a la forma de comunicación del equipo, forma de registrar las horas utilizadas y plantillas de documentos entre otros.

Esta fase no tuvo alteración de esfuerzo ni de tiempo a lo largo del desarrollo del proyecto.

3.2. Fase I – Investigación y Estado del Arte

En esta fase se realiza la investigación de las distintas técnicas de Data Mining. Dicha investigación se realiza básicamente sobre material extraído de Internet aunque se consulta también bibliografía específica de la biblioteca del Instituto de Computación (INCO).

Es en esta fase que se prepara el documento 'Estado del Arte' **[3]**. Para la preparación de dicho documento se aborda el tema por técnica y dentro de cada técnica se analizan las herramientas existentes, los algoritmos y el software disponible en el mercado.

Cabe destacar que en esta investigación nos enfrentamos con dos dificultades. Una de ellas es que existe cierto solapamiento entre las técnicas, lo cual obviamente es lógico y esperable, y otra es que la calidad y cantidad de información disponible es muy variable dependiendo de la técnica descrita.

En esta fase se invierte más esfuerzo que el estimado ya que la información que se encuentra de Data Mining es más diversa y vaga de la esperada por el equipo. Surge la necesidad de especializar o focalizar la investigación. Para esto se busca un área de aplicación concreta, surgiendo dos posibles caminos a tomar, por un lado se busca un cliente con alguna necesidad concreta en ésta área, y por otro se comienzan a investigar problemas abiertos en general que puedan ser resueltos con alguna de las técnicas de Data Mining.

3.3. Fase II – Búsqueda campo aplicación

Esta fase surge luego que se detecta la necesidad de focalizar la investigación y concretar el problema a resolver.

Inicialmente se iba a aplicar alguna de las técnicas investigadas con datos provistos por el tutor, pero al ir avanzando el proyecto es de interés del equipo buscar un problema específico que plantee una problemática real a ser resuelta por el grupo.

La concretización del problema se realizó por dos vías, buscando problemas abiertos posible de de ser resueltos con técnicas de Data Mining y buscando posibles clientes con requerimientos concretos aplicables en ésta área.

Dentro de los problemas abiertos se investiga el uso de redes neuronales en el área de detección de intrusos. También se investiga la aplicabilidad de redes neuronales en programas de detección de correo no deseado (SPAM).

Para localizar un cliente el grupo se entrevista con cuatro “posibles clientes” analizando si la problemática planteada por cada uno se aplica al alcance del proyecto.

Es en esta fase que se prepara el documento “Búsqueda del cliente” **[4]**, en el cual se describe en detalle el rubro de los potenciales clientes entrevistados y el motivo por el cual finalmente se elige a DINARA como el único cliente viable.

El equipo decide que como proyecto académico era de interés agregar la interacción con un cliente puntual, por lo tanto se decide atacar la problemática de la DINARA como campo de aplicación.

Al realizar el análisis del problema planteado por el cliente surge la necesidad de preparar la “Extensión del Estado del Arte” **[5]**, que es una investigación localizada en la problemática a resolver para el cliente elegido.

La toma de esta decisión que insumió un tiempo considerable no se tuvo en cuenta en la planificación original, y fue uno de los principales motivos por los cuales se retrasaron los tiempos de finalización del proyecto. No solo por el tiempo incurrido en la fase sino también por la propia interacción con un nuevo actor en las fases siguientes.

3.4. Fase III – Definición del Problema y Especificación

En esta fase se realizan varias entrevistas con el cliente y reuniones con el tutor para resolver en que puntos de la problemática del cliente se aplican las técnicas investigadas en las fases anteriores de forma tal de maximizar el valor agregado al cliente y al taller.

Es esta una de las fases que consume más tiempo por algunas demoras en conseguir la información necesaria, así como para ajustar las necesidades del cliente con el proyecto en sí.

En las reuniones con el cliente, se discuten sus necesidades y el cliente expone las herramientas con las que cuenta para analizar la información generada por las balizas así como la ausencia de una herramienta adecuada para la toma de distintas decisiones.

Todas las reuniones están debidamente registradas en sus correspondientes documentos de minutas **[7]**, de acuerdo al documento de metodología **[2]**.

En esta fase se realiza un prototipo de la aplicación. Dicho prototipo es aprobado por el cliente y el tutor. En base al prototipo y a las reuniones con el cliente se realiza la especificación de requerimientos **[6]** siguiendo la norma IEEE 830.

La creación de un prototipo de la aplicación no estaba pensada como tal dentro de los planes iniciales, es para una mayor comprensión del problema y un mejor análisis que se realiza el mismo.

La creación del prototipo cumple con dos objetivos, capacitar a los integrantes del equipo en la creación de páginas HTML y la navegación entre las mismas, y una rápida comprensión del cliente de la solución propuesta.

El equipo entiende que realizar la clasificación de los datos y además brindar una forma clara de visualización de los resultados en el sistema utilizado por el cliente maximiza el valor agregado de la solución, permitiendo un primer análisis completamente visual ideal en un proceso de toma de decisiones.

Por esta razón se realiza una investigación y capacitación sobre el Sistema de Información Geográfico GRASS **[14] [19]**. Luego de la investigación y de elegida la forma de interactuar con el GRASS, se tiene una reunión con un especialista en sistemas de información geográficos, quien indica que la misma era la forma adecuada de acuerdo a las posibilidades que brinda

GRASS. Las tareas relativas al manejo de GRASS no estuvieron previstas en la planificación original.

3.5. Fase IV – Diseño e Implementación de la solución

Elección de Técnicas y Algoritmos

En esta fase se determina que la técnica de Data Mining más apropiada es clustering y se decide que los algoritmos más adecuados para la clusterización de datos espaciales son DBSCAN, DBRS y OPTICS.

Luego de determinados los algoritmos a utilizar, se comienza la búsqueda de herramientas que implemente o faciliten la implementación de estos algoritmos. Después de evaluadas las distintas herramientas encontradas se concluye que ninguna herramienta por si sola aporta suficiente valor como para ser tenida en cuenta. Se decide entonces desarrollar un componente que implemente los algoritmos elegidos y sea lo suficientemente flexible para ser extendido o modificado.

Diseño de la solución

En esta fase se analizan las posibles arquitecturas y los posibles diseños aplicables a la herramienta a desarrollar.

Se buscan implementaciones de los algoritmos entrando en contacto con lo creadores de los mismos, quienes nos enviaron una implementación del DBSCAN y otra del OPTICS. Estas librerías fueron usadas mediante técnicas de wrapeo para la implementación de una librería de algoritmos. De esta forma se logra tener el algoritmo implementado sin depender exclusivamente de la implementación externa. Se complementa la librería de algoritmos con un desarrollo integro del algoritmo DBRS ya que no se pudo obtener una implementación del mismo.

Se ve la necesidad de desarrollar un algoritmo simple de clasificación a los efectos de poder corroborar el funcionamiento de los anteriores, este algoritmo fue llamado DataFishClassificator (DFC). Al contar con el DFC y la incorporación del concepto de grilla (concepto diferente al usado por los otros algoritmos) surgió la idea de implementar un algoritmo de clustering nuevo llamado DataFishClassificatorEmpowered (DFCE).

Se define la arquitectura y el diseño del sistema ***DataFish*** **[9]**.

En esta fase se requiere capacitación en las diferentes tecnologías utilizadas tanto en desarrollo como en infraestructura.

Desarrollo de la aplicación

Se configuran dos ambientes, uno sobre Windows XP para el desarrollo y otro sobre Linux para el testeo. Dicha configuración de ambientes sobrepasa levemente el tiempo inicialmente estimado para dicha tarea. Se ve que es mas práctico desarrollar sobre el ambiente Windows y correr la parte del Sistema de Información Geográfico sobre el ambiente Linux.

Se desarrolla la aplicación utilizando JAVA 1.4 en el entorno de desarrollo WebSphere Studio Application Developer (WSAD).

Testeo

Se investiga la existencia de datos de prueba que puedan ser útiles para el testeo de los diferentes algoritmos. Dentro de los juegos de datos disponibles se identifica el benchmark “Sequoia” como candidato para testear los algoritmos implementados en lo que tiene que ver con performance, y se identifican los juegos de datos clásicos para las pruebas de funcionamiento. También se testea con juegos de datos proporcionados por el cliente.

Aprobación
Se presentan el aplicativo y los resultados al tutor y al cliente, aprobando ambos el trabajo realizado.

En el tiempo global de la fase IV se invierte un esfuerzo muy similar al planificado originalmente.

3.6. Fase V – Entrega y Presentación

En esta fase se preparara la entrega final y la presentación **[13]**.

Esta fase esta totalmente de acuerdo a lo planificado en cuanto a tareas y a esfuerzo.

4. Requerimientos

La especificación de requerimientos se puede leer en el documento "Especificación de Requerimientos de Software" especialmente escrito para el caso **[6]**, pero a los efectos de entender el desarrollo de este informe se destacan los puntos más relevantes del mismo.

El sistema con el que cuenta actualmente el cliente, descrito en el documento Descripción de la Realidad **[10]** y resumido en el Capítulo 2 de este informe, no permite, al menos de una forma sencilla, visualizar el comportamiento de una determinada flota marítima, o cuales son las regiones donde se realiza el mayor esfuerzo pesquero, y menos aún deducir el comportamiento de una especie el cual es uno de los objetivos finales de la DINARA.

Lo que pretende el cliente como resultado de este proyecto es mejorar la capacidad de análisis de la información del sistema SIPESAT **[10]**, en especial lo que tiene que ver con análisis visual.

Se establecieron como requerimientos primarios para este proyecto el poder visualizar las áreas donde se realiza el esfuerzo pesquero de determinadas flotas en diferentes rangos de tiempo, y la capacidad de dinámicamente generar vistas de las señales del sistema SIPESAT según diferentes criterios.

Como requerimientos secundarios se establecieron, la capacidad de generar alarmas cuando las embarcaciones entran en zonas de veda o cuando se aproximaran a puerto, y desarrollar una herramienta capaz de cuantificar en horas el esfuerzo pesquero.

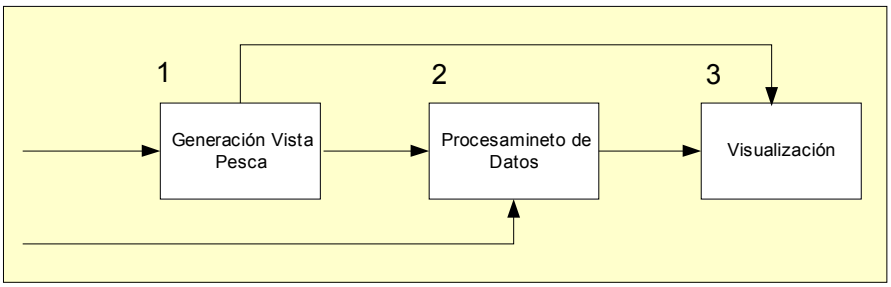
Otro requerimiento importante a destacar es que las herramientas desarrolladas en el proyecto se deben integrar al sistema SIPESAT.

El sistema SIPESAT fue desarrollado sobre una plataforma usando software comercial, basado en Windows **[25]** como sistema operativo, Informix **[26]** como motor de base de datos, y ArcView **[27]** como Sistema de Información Geográfico. Actualmente la DIANARA se encuentra en un proceso de migración hacia software de distribución libre. En este nuevo ambiente han implementado el sistema existente sobre una plataforma basada en Linux **[32]** como sistema operativo, Postgresql **[18]** como motor de base de datos y GRASS **[19]** como Sistema de Información Geográfico.

Se estableció como requerimiento que todas las herramientas desarrolladas debían correr e interactuar con el ambiente basado en software de distribución libre.

Una vez establecidos los requerimientos, el ambiente en el cual debe ejecutar la aplicación, y los sistemas con los que debe interactuar, se definen los procesos que se ejecutan en el sistema ***DataFish***.

La siguiente figura muestra los tres procesos principales del sistema:



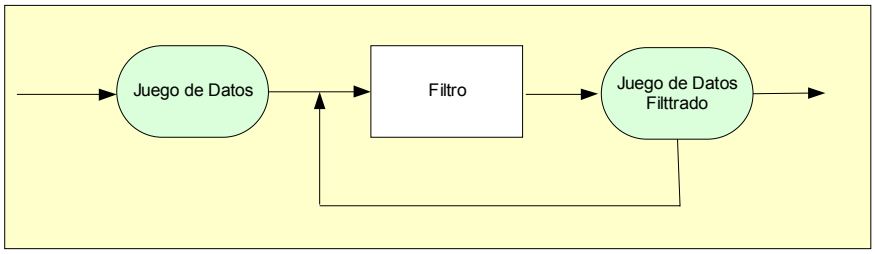
Los tres procesos son:

- Generación Vista Pesca

- Procesamiento de Datos
- Visualización de resultados

El usuario de la aplicación puede iniciar la ejecución de la misma disparando el proceso *Generación Vista Pesca* o directamente mediante la ejecución del proceso *Procesamiento de Datos.* El proceso *Procesamiento de Datos* puede tomar como entrada la salida del proceso *Generación Vista Pesca* o puede iniciarse por la ejecución directa del usuario.
La salida del proceso *Procesamiento de Datos* es la entrada del proceso de *Visualización.*
La aplicación esta preparada para tomar la salida del proceso *Generación Vista Pesca* como la entrada del proceso de *Visualización.*

Generación Vista Pesca

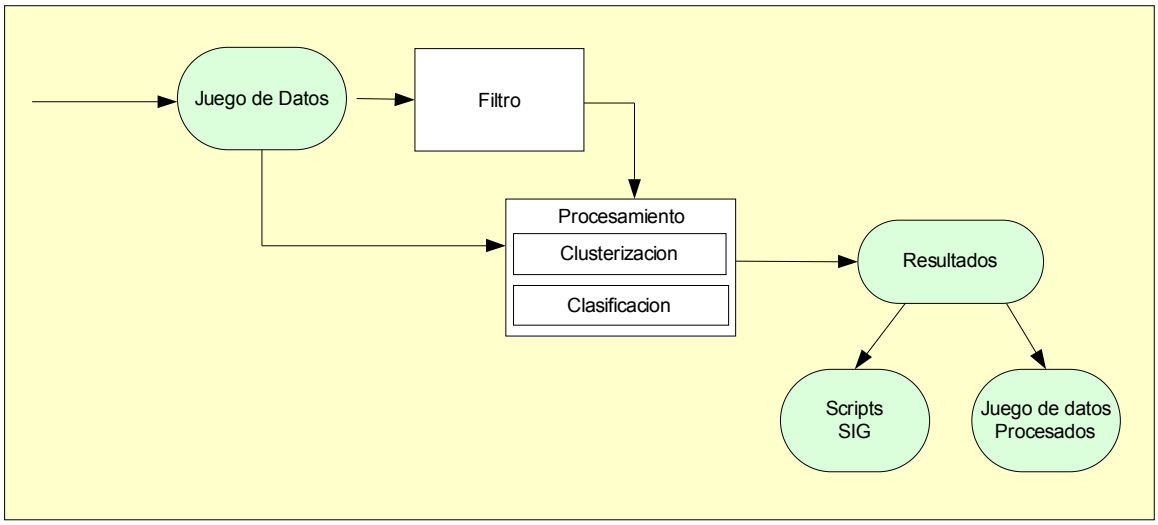


A partir de un conjunto de datos *Juego de Datos* se le aplica un *Filtro,* obteniendo como resultado un *Juego de Datos Filtrado* que pueden a su vez ser la entrada de un nuevo *Filtro* o puede ser el conjunto final de datos a ser procesado o visualizado.

En conjunto *Juego de Datos* puede ser obtenido o bien de un archivo o bien de una tabla de una base de datos.

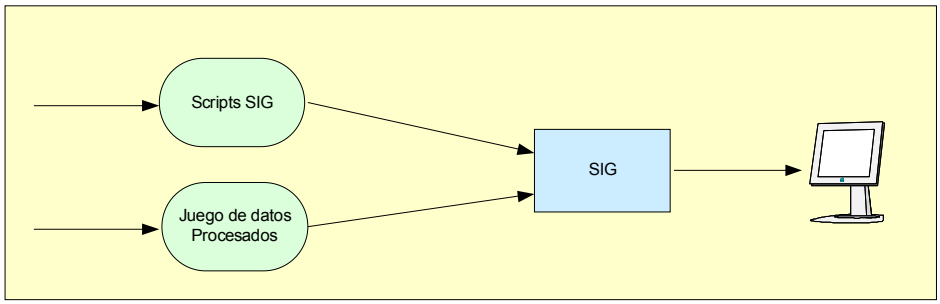
El *Juego de Datos* son las señales de las embarcaciones, y los filtros que se aplican son del tipo, fechas, pescando o no pescando, flota, etc. Por más información ver el "Manual de Usuario" **[11]** y el documento "Arquitectura y Diseño" **[9]**.

Procesamiento de Datos



El proceso de *Clusterización* o *Clasificación* toma como datos de entrada un *Juego de Datos* el cual puede ser filtrado o no, y tiene como salida los datos *Resultados*, que esta compuesto por *Scripts SIG* (scripts para ser ejecutados en el Sistema de Información Geográfico) y *Juego de datos Procesados* (tablas con los datos clisterizados/clasificados)

Visualización de resultados



Los *Scripts SIG* y el *Juego de datos Procesados* son la entrada al Sistema de Información Geográfico, con estos dos elementos el usuario puede visualizar los resultados obtenidos.

5. Diseño de la solución

5.1. Ambiente

Con el objetivo de facilitar la comprensión de las decisiones de diseño tomadas es que se especifica en esta sección el ambiente en el cual debe correr la aplicación. Por un detalle de dicho ambiente se puede consultar la especificación de requerimientos **[7].**

El cliente establece que la aplicación debe correr sobre sistema operativo Linux, la base de datos debe ser Postgresql **[18]**, el Sistema de Información Geográfico especificado es GRASS **[19]** y todo software adicional utilizado debe ser de uso libre.

El esquema de datos es dado por el cliente, y la aplicación ***DataFish*** toma los datos de la base del sistema SIPESAT y genera la información que se visualiza en el Sistema de Información Geográfico GRASS.

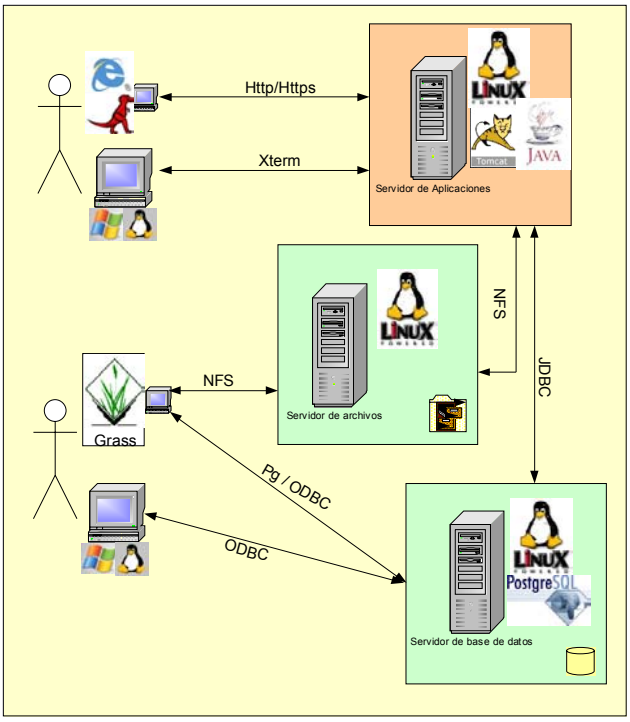
Para satisfacer los requerimientos del cliente, se decide desarrollar una aplicación Web la cual puede ser ejecutada desde cualquier navegador.

La aplicación se instala en un servidor de aplicaciones Tomcat **[28]**, la elección de Tomcat esta fundada en su conocido desempeño en ambientes similares al de este proyecto y por su característica de software de distribución libre.

La aplicación interactúa con la base de datos Postgresql, y genera los datos necesarios para poder ser visualizados en el Sistema de Información Geográfico GRASS, o bien ser consultados desde cualquier cliente SQL **[29]** vía ODBC **[30]**.

En el documento “Arquitectura y Diseño” **[9]** se puede ver en detalle la arquitectura de la aplicación.

En la siguiente figura se puede ver un esquema del ambiente donde se ejecuta la aplicación.



5.2. Características específicas y Técnicas de Data Mining

Antes de mencionar y justificar cuales fueron las técnicas de Data Mining utilizadas e implementadas se resumen cuales son las características más significativas de los requerimientos y de la realidad planteada desde el punto de vista del análisis realizado.

Análisis visual
Era requerimiento fundamental del cliente poder visualizar los resultados obtenidos en su Sistema de Información Geográfico, ya que lo que se pretende como se ve en la especificación de requerimientos **[6]**, es encontrar las zonas de pesca para determinadas especies o flotas dependiendo el caso.

Tipo y Volumen de datos
Una de las características principales de la realidad que condicionan fuertemente la elección de las técnicas a utilizar e implementar, es el volumen de datos con el que se trabaja y el tipo de dato con el que se cuenta.

Tipo:
El tipo de datos con el que se trabaja es un registro de posición, velocidad y tiempo, de una embarcación, en el documento que describe la realidad se puede encontrar un detalle del tipo de datos y cómo es obtenido **[10]**. A los efectos de la terminología manejada los datos son del tipo espacial (posición) con atributos no espaciales. Al momento de realizar el trabajo, no se podía contar con más datos no espaciales, como ser que especie realmente se estaba pescando en el momento que fue emitida la señal, aunque en el análisis se asumió que se va a contar con ese dato en un futuro.

Volumen:
El volumen de datos manejado, por lo visto en la investigación se puede considerar como “Very Large Data Base” (VLDB). Se emiten 24 señales por día, desde más de 100 embarcaciones, lo que hace más de 72000 señales por mes y más de 864000 por año.

Otras características:
Otra característica importante para este análisis es el hecho de que los datos cuentan con una fuerte presencia de “ruido”.

Análisis
En términos generales lo que se pretende es identificar agrupaciones de embarcaciones (puntos) dentro de una gran cantidad de puntos en presencia de ruido, y poder visualizarlas. Si bien otras técnicas estudiadas, como redes neuronales, son utilizadas para encontrar agrupaciones de puntos, la técnica sobre la cual existe mayor desarrollo e investigación es clustering especialmente en realidades similares a la planteada en este proyecto.

Cómo se vio en “Extensión del Estado del Arte” **[5]**, la *Clasificación* es la técnica madre de clustering, tanto así que muchas veces son confundidas en la literatura. La técnica de *Clasificación* es una técnica atendida o dirigida, a diferencia de las técnicas de clustering que son desatendidas, dada esta diferencia se decidió implementar también una herramienta de clasificación persiguiendo básicamente dos objetivos.

Un objetivo es la verificación, dicha verificación se enfoca desde dos puntos de vista, una es la verificación en el ámbito de desarrollo, esto es chequear los resultados de los algoritmos de clustering desarrollados, y otra es la verificación de la aplicabilidad de los algoritmos de clustering a la realidad del cliente.

Otro objetivo es brindarle al usuario una herramienta del tipo atendida para validar cierta información que recibe de los partes de pesca (documento que las embarcaciones entregan al

llegar a puerto) la cual es más fácil de analizar con una herramienta de este tipo, y que a su vez le sirve para cruzar información y validar resultados obtenidos con las herramientas de clustering.

Las técnicas de Data Mining elegidas entonces para resolver la problemática planteada son Clustering y Clasificación.

5.3. Algoritmos

Los algoritmos de clustering elegidos para implementar son DBSCAN **[22]**, DBRS **[15]** y OPTICS **[20]**. Se desarrollan dos algoritmos propios uno de clasificación DataFishClasificator (DFC) y otro de clustering DataFishClassificatorEmpowered (DFCE).

Como se puede ver en el documento "Extensión del Estado del Arte" **[3]**, estos algoritmos fueron elegidos por ser de los más apropiados para trabajar con bases de datos espaciales y grandes volúmenes de datos.

A continuación se da una breve descripción de los mismos y su seudo código, la explicación teórica de los mismos esta fuera del alcance de este documento y se recomienda leerla para una mejor comprensión de su aplicabilidad a este proyecto en sus respectivos papers de presentación, los cuales se pueden encontrar en la carpeta Bibliografía del CD DataFish. **[15]**, **[20]**, **[22]**

DBSCAN, DBRS y OPTICS

Estos tres algoritmos están basados en el concepto de densidad de puntos y la idea general es que la densidad de puntos dentro del cluster es mayor que fuera del mismo. Por ende la densidad en las áreas de ruido es menor que dentro de un cluster. Una vez que se definen las áreas suficientemente densas, estas se unen por un criterio llamado 'densamente alcanzables'.

El algoritmo DBSCAN es el que introdujo este concepto y en sus primeras versiones no tenía en cuenta datos no espaciales. Versiones posteriores (GDBSCAN, DBSCAN*) introdujeron esta opción. **[21]**

Lo que diferencia el algoritmo DBRS de los de la familia DBSCAN es que utiliza una forma de selección de puntos randómica que lo hace más performante y escalable, aunque con una pequeña degradación de exactitud en los puntos y clusters encontrados, la cual es despreciable si el algoritmo es aplicado a un conjunto de datos especialmente denso. **[15]**, **[22]**

El algoritmo OPTICS se diferencia más de los otros, usa el mismo concepto de densidad, pero no retorna los grupos o clusters encontrados, de hecho por si solo no clasifica los puntos en clusters, sino que devuelve una lista jerárquica ordenada de los puntos clasificados, con un valor de proximidad a su punto más cercano. Con esta lista y mediante la aplicación de algún algoritmo de conversión de los resultados es posible extraer los clusters detectados. **[23]**, **[24]** La principal ventaja de OPTICS es la capacidad de detectar sub-clusters y de clasificar jerárquicamente los mismos, cualidades que los otros algoritmos no poseen. **[20]**

DFC y DFCE

El algoritmo de clasificación DataFishClassificator DFC divide el mapa en una grilla de un ancho de celda especificado por el usuario. Clasifica los puntos en sus celdas correspondientes, y devuelve las celdas que superen una cierta cantidad de puntos también especificada.

El DataFishClassificatorEmpowered DFCE se puede considerar un algoritmo de clustering del tipo basado en grilla, en el cual luego de correr el DFC, realiza de un modo inteligente la unión de aquellas celdas vecinas que superan la cantidad de puntos configurada.

Seudo códigos
A continuación se muestran los seudo códigos de los algoritmos tal cual fueron presentados por sus creadores.

- **DBSCAN**

```
DBSCAN (SetOfPoints, Eps, MinPts)
// SetOfPoints is UNCLASSIFIED
ClusterId := nextId(NOISE);
FOR i FROM 1 TO SetOfPoints.size DO
       Point := SetOfPoints.get(i);
       IF Point.ClId = UNCLASSIFIED THEN
              IF ExpandCluster(SetOfPoints, Point,
                     ClusterId, Eps, MinPts) THEN
                ClusterId := nextId(ClusterId)
              END IF
       END IF
END FOR
END; // DBSCAN


ExpandCluster(SetOfPoints, Point, ClId, Eps,MinPts) : Boolean;
seeds:=SetOfPoints.regionQuery(Point,Eps);
IF seeds.size<MinPts THEN // no core point
       SetOfPoint.changeClId(Point,NOISE);
       RETURN False;
ELSE   // all points in seeds are density-
       // reachable from Point
       SetOfPoints.changeClIds(seeds,ClId);
       seeds.delete(Point);
       WHILE seeds <> Empty DO
              currentP := seeds.first();
       result := SetOfPoints.regionQuery(currentP,Eps);
       IF result.size >= MinPts THEN
              FOR i FROM 1 TO result.size DO
                     resultP := result.get(i);
                     IF resultP.ClId IN {UNCLASSIFIED, NOISE} THEN
                            IF resultP.ClId = UNCLASSIFIED THEN
                                   seeds.append(resultP);
                            END IF;
                            SetOfPoints.changeClId(resultP,ClId);
                     END IF; // UNCLASSIFIED or NOISE
              END FOR;
       END IF; // result.size >= MinPts
       seeds.delete(currentP);
       END WHILE; // seeds <> Empty
       RETURN True;
END IF
END; // ExpandCluster
```

- **DBRS**

```
Algorithm DBRS(D, Eps, MinPts, MinPur)
1 ClusterList = Empty;
2 while (!D.isClassified( ))
3       { Select one unclassified point q from D;
4       qseeds = D.matchingNeighbors(q, Eps);
5        if ((|qseeds| < MinPts) or (qseeds.pur < MinPur))
6              q.clusterID = -1; /*q is noise or a border point */
7       else
8           { isFirstMerge = True;
         C_i = ClusterList.firstCluster;
              /* compare qseeds to all existing clusters */
10            while (C_i != Empty)
11                    { if ( hasIntersection(qseeds, C_i) )
12                           if (isFirstMerge)
13                                  { newC_i = C_i.merge(qseeds);
14                                  isFirstMerge = False; }
15                           else
16                                  { newC_i = newC_i.merge(C_i);
17                                  ClusterList.deleteCluster(C);}
18                      C_i = ClusterList.nextCluster;
                      } // while != Empty
              /*No intersection with any existing cluster */
19            if (isFirstMerge)
20                    { Create a new cluster C_j from qseeds;
21                      ClusterList = ClusterList.addCluster(C_j);
                      }//if isFirstMerge
           } //else
       }//while!D.isClassified
```

- **OPTICS**

```
OPTICS (SetOfObjects, e, MinPts, OrderedFile)
  OrderedFile.open();
  FOR i FROM 1 TO SetOfObjects.size DO
       Object := SetOfObjects.get(i);
       IF NOT Object.Processed THEN
              ExpandClusterOrder(SetOfObjects, Object, e,MinPts, OrderedFile)
  OrderedFile.close();
END; // OPTICS

ExpandClusterOrder(SetOfObjects, Object, e, MinPts,OrderedFile);

neighbors := SetOfObjects.neighbors(Object, e);
Object.Processed := TRUE;
Object.reachability_distance := UNDEFINED;
Object.setCoreDistance(neighbors, e, MinPts);
OrderedFile.write(Object);
IF Object.core_distance <> UNDEFINED THEN
       OrderSeeds.update(neighbors, Object);
       WHILE NOT OrderSeeds.empty() DO
              currentObject := OrderSeeds.next();
              neighbors:=SetOfObjects.neighbors(currentObject, e);
              currentObject.Processed := TRUE;
              currentObject.setCoreDistance(neighbors, e, MinPts);
              OrderedFile.write(currentObject);
              IF currentObject.core_distance<>UNDEFINED THEN
                     OrderSeeds.update(neighbors, currentObject);
END; // ExpandClusterOrder

OrderSeeds::update(neighbors, CenterObject);

c_dist := CenterObject.core_distance;
FORALL Object FROM neighbors DO
       IF NOT Object.Processed THEN
              new_r_dist:=max(c_dist,CenterObject.dist(Object));
              IF Object.reachability_distance=UNDEFINED THEN
                     Object.reachability_distance := new_r_dist;
                     insert(Object, new_r_dist);
              ELSE // Object already in OrderSeeds
                IF new_r_dist<Object.reachability_distance THEN
                       Object.reachability_distance := new_r_dist;
                       decrease(Object, new_r_dist);
END; // OrderSeeds::update
```

- **DataFishClassificator**

```
Function dataFishClassificator ( SetOfPoint set,
                                 Real zoneWide,
                                 Number minAmountPoint ) --> ListOfCluster

      // Make grid
       ListOfZones zoneList = Grid.makeGrid( zoneWide );

       // classify zones
       for each point in set {
           for each zone in zoneList {
              if (point in zone){
                 zone.addPoint(point);
               }
            }
       }
       // discard zones
       for each zone inZoneList {
            if ( satisfaceCondiciones( minAmountPoint ) {
              clusterList.add( zone )
            }
       }
```

- **DataFishClassificatorEmpowered**

```
Function dfcEmpower( SetOfPoint set,
                     Real zoneWide,
                     Number minAmountPoint ) --> ListOfCluster

    // Make grid
    ListOfZones zoneList = Grid.makeGrid( zoneWide );

    // classify zones
    for each point in set {
          for each zone in zoneList {
              if (point in zone){
                        zone.addPoint(point);
               }
          }
    }

    // discard not satisfied zones and merge neighbors zones
    for each zone in zoneList {
           if ( satisfaceCondiciones( minAmountPoint ) {
              ListOfZones neighborsZone = findClassifiedNeighbors(zone);
               if (neighborsZone isNotEmpty ){
                        mergeClusters ( neighborsZone, zone );
               } else {
                        clusterList.add(zone);
               }
           }
    }
    return clusterList;
```

5.4. Decisiones de diseño

Una vez decidido las técnicas y los algoritmos a utilizar en el proyecto, se comienza con la evaluación de herramientas y entornos de trabajo a utilizar. Para ello existieron varias restricciones y requerimientos que se describen a continuación.

Se evalúan siempre herramientas de uso libre por requerimiento del cliente. Es deseable que las herramientas a utilizar interactúen fácilmente con la base de datos del cliente y el Sistema de Información Geográfico. Es necesario que la herramienta soporte el volumen de datos con el cual se esta trabajando.

Se evaluaron un conjunto de herramientas, las cuales no cumplen satisfactoriamente con los requerimientos antes expuestos, por lo que se decide implementar todos los algoritmos como se describe en el documento "Arquitectura y Diseño" **[9]**. A continuación se describen algunas de las herramientas evaluadas y se explica porque finalmente no fueron utilizadas.

5.4.1. Herramientas Evaluadas

R Project [29]

El lenguaje R es un entorno muy utilizado en el ambiente de software libre sobre el cual hay implementados varios algoritmos de cluster entre otras cosas, e incluye una interfaz de visualización.

Ninguno de los algoritmos seleccionados esta implementado en R, se entiende que este entorno no aporta demasiadas herramientas parar implementar los algoritmos en él, y además se debería implementar toda la interfaz con la aplicación Web.

GeoVistaStudio [30]

Este es un entorno de desarrollo y visualización para Data Mining en sistemas de información geográficos, es de uso libre, trae muchas herramientas de visualización y varios problemas de interfaces resueltos. Fue descartada ya que el Sistema de Información Geográfico que usa es ArcGis **[27]**, y además requiere que se trabaje siempre dentro del entorno GeoVista esto último no permite incluir otras funcionalidades necesarias en la aplicación ***DataFish***.

ATLAS [31]

Esta herramienta fue evaluada ya que trae incluida una implementación de DBSCAN. Atlas es un SQL extendido que hace más sencilla la implementación de aplicaciones que realizan muchas operaciones sobre datos que se encuentran en un repositorio relacional. Dado que para usarla hay que implementar una interfase la única ventaja que presenta es el acceso a la base de datos, es por eso que se descarta.

5.4.2. Diseño de algoritmos

Dado que no se encontró ninguna herramienta específica o entorno que pudiera aportar real valor en el desarrollo de los algoritmos y el utilizar cualquiera de las mencionadas implicaba implementar interfaces con el resto de las componentes del sistema que ya eran dadas por los requerimientos, como ser el Sistema de Información Geográfico y el motor de bases de datos, se decide implementar un componente donde se desarrollan los algoritmos.

Los algoritmos implementados deberán cumplir la interfaz requerida, especificada en el documento de arquitectura y diseño **[9]**

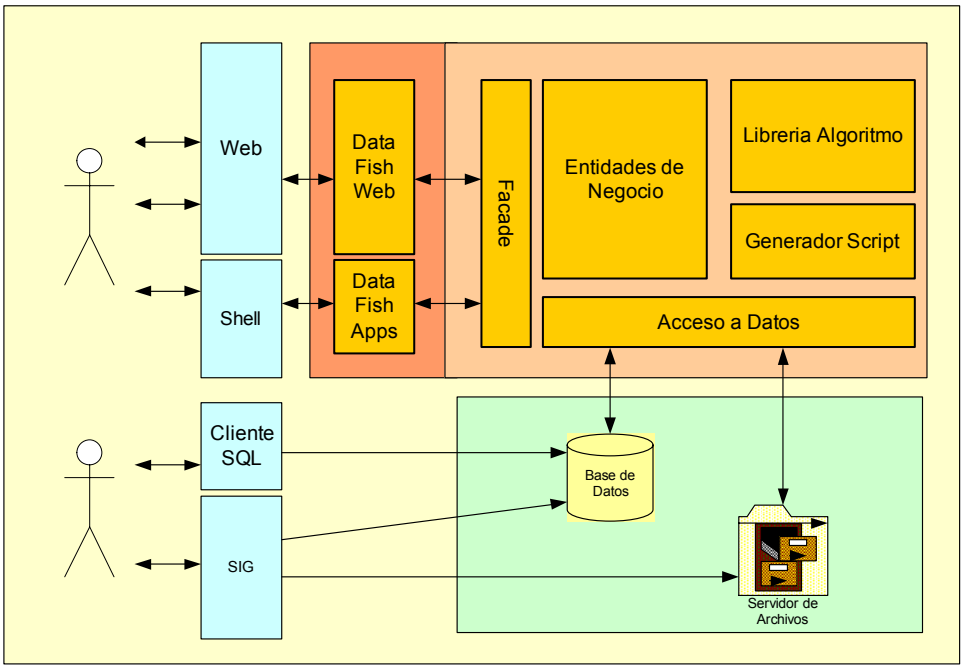
Elección de herramienta de desarrollo

Como se expuso anteriormente uno de los requerimientos de la aplicación es que esta ejecute en un servidor Linux **[32]**. Por otra parte se considero que la mejor opción era tener un servidor de aplicaciones para poder ejecutar remotamente tanto desde línea de comando como desde un navegador, luego como se explico en la Sección 5.1 se opto por Tomcat **[28]**. La herramienta de desarrollo en la cual el equipo tiene mayor experiencia es JAVA **[33] [34]**, además se encontró en la investigación que la mayoría de las implementaciones de algoritmos de clustering hechas últimamente están desarrolladas en JAVA. Por los puntos expuestos anteriormente fue una decisión casi inmediata desarrollar la aplicación en JAVA.

6. Implementación

En este Capítulo se introducen los aspectos más relevantes de la implementación del sistema ***DataFish*** desde un punto de vista global o de alto nivel.

En el Capítulo 5 de este informe se explicó la arquitectura física de la aplicación, en donde se muestran cuales son y donde corren las distintas componentes del sistema. La figura siguiente es un diagrama conceptual de las componentes del sistema.



6.1. Interfaz de usuario

Las interfaces que tiene el usuario para interactuar con el sistema, se pueden ver en la figura anterior con color celeste y se describen a continuación.

Interfaz Web:
La interfaz Web le permite al usuario ingresar los datos para ejecutar tanto sea la generación de vistas como el procesamiento de datos. (Por más detalles ver manual de usuario **[11]**)

Línea de comando:
Las mismas operaciones que se pueden realizar en la interfaz Web, se pueden hacer desde línea de comando ingresando en modo Terminal al servidor donde está instalada la aplicación.

Sistema de Información Geográfico:
A través de esta interfaz es que el usuario del sistema puede visualizar los resultados obtenidos, para ello el sistema genera los scripts y datos necesarios para que el usuario genere los vectores a ser visualizados.

Cliente SQL:
Otra posibilidad de acceso a la información generada es mediante cualquier cliente SQL. El usuario se puede conectar a las tablas generadas como resultado, en donde puede consultar los datos obtenidos, por ejemplo la cantidad de zonas encontradas, la cantidad de puntos clasificados, etc. De esta forma, puede hacer una primera evaluación del resultado antes de pasar a la visualización de lo mismos o tomar estos datos como entrada de cualquier otro sistema.

6.2. Lógica de Presentación

En esta capa se implementan las interfaces entre el usuario y la lógica de la aplicación. Esta decisión de diseño tiene como ventaja, desde el punto de vista lógico, que se puede cambiar íntegramente la presentación sin tener que modificar la lógica del negocio. Desde el punto de vista de infraestructura brinda la posibilidad de distribuir en distintos servidores ambas capas.

Se implementaron dos módulos de comunicación entre el usuario y la aplicación, uno vía Web y otro por línea de comando. Si bien la interfaz Web brinda mejor facilidad de uso para un usuario final, la opción de ejecución mediante línea de comando brinda otras posibilidades, como por ejemplo la creación de procesos batch mediante scripts que ejecuten varios requerimientos en momentos que los recursos (por ej. repositorios de datos) son menos usados. Es de destacar que ambas interfaces brindan exactamente las mismas posibilidades de ejecución.

DataFishWeb
Este módulo implementa toda la interfaz Web con el cliente y está desarrollado siguiendo el patrón Model View Controller (MVC) **[35]**. Fue implementado utilizando el framework Struts **[16]** el cual provee las facilidades necesarias para la correcta aplicación del patrón. La componente View de este patrón fue implementada íntegramente con Java Server Pages (JSP) **[36]**.

DataFishApp
Este módulo implementa la interfaz vía línea de comando desde una shell, desde la cual se puede invocar a la aplicación.

6.3. Lógica de la Aplicación

Esta capa reúne las componentes que implementan la lógica de la aplicación manejando las interacciones entre las mismas, cumpliendo los casos de uso requeridos por el sistema. Esta separación en componentes brinda flexibilidad en la sustitución, mejora o ampliación de los mismos. En los siguientes puntos se describen cada uno de los componentes, por más detalles ver “Arquitectura y Diseño” **[9]**.

Facade
Este módulo implementa el conocido patrón de diseño Façade. Presenta todas las posibles interacciones con los módulos cliente. Esto permite tener una entrada única al sistema, con las ventajas obvias al momento de establecer políticas de seguridad, verificación, etc. También otorga flexibilidad al momento de hacer modificaciones en el resto de los componentes.

Entidades de negocio
En este módulo se implementan los objetos propios del negocio de la DINARA, como ser embarcación, señal, zona, ubicación etc.

Liberia de algoritmos
El corazón de la aplicación radica en los algoritmos de clustering/clasificación con que se procesan los datos. Se desarrolla como un módulo aparte a los efectos de que fuera simple la modificación de los mismos y la incorporación de nuevos algoritmos.

Esta librería comprende la implementación de los algoritmos de clustering/clasificación y puede ser ampliada y/o modificada sin tener que modificar nada en el resto de la implementación del aplicativo ***DataFish***. Por más detalles ver documento de arquitectura **[9]**

Anteriormente se explicó que el algoritmo OPTICS no devuelve el conjunto de clusters detectados sino que devuelve una lista ordenada jerárquicamente **[20]**, y que era necesario utilizar otras herramientas para encontrar los clusters a partir de la mencionada lista **[23][24]**. En la librería de algoritmos se incluyen dos componentes con características diferentes para dicha tarea. El detalle del funcionamiento de dichas componentes y cuales de las opciones

presentadas en los papers especializados **[23] [24]** se implementan se puede leer en el documento de arquitectura **[9]**.

Generador de scripts
Este módulo es el encargado de generar los scripts para visualizar los resultados obtenidos luego de la clusterización/clasificación seleccionada. El desarrollo de este módulo se hace utilizando las facilidades provistas por el framework Velocity **[17]** para la generación e instanciación de plantillas.

Es de destacar que se realiza un módulo específico ya que si bien se implementan scripts solo para GRASS **[19]**, se deja la posibilidad abierta para que se generen en un futuro scripts para cualquier sistema de visualización, simplemente modificando una plantilla. En el manual de instalación **[12]** se puede ver en detalle cómo y donde se modifica esta opción.

Acceso a datos
El acceso a datos como se puede ver en el diagrama conceptual es en doble sentido, uno es de donde toma los datos la aplicación y el otro es donde se guardan los resultados.

Origen de datos:
Los datos que toma la aplicación, además de los suministrados por el usuario vía Web o mediante la invocación desde línea de comando, es el juego de datos a procesar. Dicho juego de datos puede ser tomado tanto desde una base de datos relacional como desde un archivo delimitado por comas. El usuario puede seleccionar el origen de datos en el mismo momento de la invocación.

Resultados:
La aplicación devuelve básicamente dos resultados, los datos clasificados o clusterizados, y los scripts a ser ejecutados en el Sistema de Información Geográfico.

En el caso que el usuario elija como destino de sus resultados la base de datos, el sistema genera y guarda los datos resultado en dos tablas, una donde se guardan las zonas (clusters) denominada *[nombre_proyecto]_areas* , y otra llamada *[nombre_proyecto]_puntos* donde se guardan los puntos clasificados/clusterizados .

Si el usuario elige como destino la opción de archivo, se genera una carpeta con el nombre del proyecto en la ruta configurada en la aplicación **[12]**, la cual contiene un archivo con la información de las zonas (clusters) encontradas, y un archivo por cada zona conteniendo la información de los puntos que contiene.

Se generan dos scripts para la creación y visualización de los vectores en el Sistema de Información Geográfico. Los scripts se graban en el directorio debidamente especificado en la configuración de la aplicación **[12]**.

6.4. Repositorio de Datos

El aplicativo utiliza básicamente dos tipos de repositorios de datos, bases de datos relacionales y archivos planos. Se ideó el sistema de forma tal que el servidor de base de datos y el servidor de archivos puedan estar separados entre ellos y principalmente separado del servidor de aplicación.

La comunicación entre el servidor de aplicaciones y el servidor de base de datos se realiza vía JDBC **[37]**, mientras que la comunicación con el servidor de archivos es vía NFS **[38]**.

Por una explicación detallada de la estructura requerida de la base de datos y del formato de los archivos, donde se encuentran los datos de entrada y de salida, ver documento de arquitectura y diseño **[9]**.

7. Testeo

En este Capítulo se explica cuales son las pruebas realizadas sobre el sistema ***DataFish*** y se exponen los resultados obtenidos. Dado que el sistema involucra varias componentes fue necesario realizar distintos tipos de pruebas. Para reflejar de forma ordenada la evaluación de las distintas partes, este Capítulo se divide en los puntos que se exponen a continuación.

En la Sección 7.1. Pruebas de Infraestructura, se enumeran los distintos ambientes en los que la aplicación es ejecutada. Las comparaciones y validaciones de los distintos algoritmos utilizados se detallan en la Sección 7.2. Pruebas de Algoritmos. Se realizó también un testeo funcional de la aplicación que se muestra en la Sección 7.3. Prueba de la Aplicación.

7.1. Pruebas de Infraestructura

Se detallan a continuación las distintas componentes que tiene el sistema, y cuales fueron las pruebas hechas. Para realizar las pruebas de infraestructura se hicieron corridas completas abarcando todos los procesos expuestos en el Capítulo 4, cambiando las componentes de plataforma según las variantes que se explican a continuación.

En lo que tiene que ver con performance en las distintas plataformas, vale decir que las pruebas si bien se hicieron en maquinas de porte similar no eran idénticas, por lo cual su resultado no tiene valor comparativo.

Base de datos
Las bases de datos sobre las que se prueba el sistema son Postgresql 7.4.5 **[18]** sobre Linux **[32]**, última versión estable del producto, y Postgresql 8.0 versión beta sobre Windows XP **[25]**, única versión existente sobre este sistema operativo.

Se encontraron diferencias de performance, sobre todo en los procesos de inserción de datos masivo, la versión del manejador de bases de datos sobre Linux es bastante más rápida. Otro resultado encontrado es que la versión instalada sobre Windows se comportó de forma muy inestable, lo cual es justificable ya que se usó una versión beta.

Servidor de aplicaciones
El servidor de aplicaciones sobre el que se instala el aplicativo es Tomcat **[28]**. Se realizan pruebas instalando la versión jakarta-tomcat-4.1.31 sobre Linux Debian **[39]** y Windows XP.

En las pruebas realizadas no se encontraron diferencias ni de performance ni de estabilidad.

Se realiza también una instalación de la aplicación en un servidor WebSphere Application Server 5.0 **[40]** sobre Windows XP, corroborando que ejecuta sin problemas en el mismo.

Cliente Web
Se hicieron prueba con distintos clientes Web, Internet Explorer 6.1 sobre Windows XP y Mozilla Firefox 1.0 sobre Linux Debian.

Se encontraron pequeñas diferencias de comportamiento entre ambos, relacionadas con la compatibilidad de JavaScript **[41]** y los navegadores. Estos puntos fueron corregidos, no habiéndose encontrado nuevas diferencias luego de los cambios.

Cliente Shell
La aplicación se puede ejecutar también desde línea de comando, se hacen las mismas pruebas que con el cliente Web, tanto en sistemas Linux como Windows, haciendo la ejecución utilizando la maquina virtual de java provista por la versión j2sdk1.4.2_04 **[42]**.

Los resultados funcionales son idénticos a los obtenidos en las pruebas realizadas con el cliente Web. Tampoco se encontraron diferencias entre las versiones de sistema operativo.

Se observa una marcada diferencia en los tiempos de ejecución entre el cliente Web y el Shell. Es más rápida la ejecución desde línea de comando, lo cual es de esperar porque el mismo accede directamente a la aplicación.

GRASS
Todas las pruebas de visualización de los resultados en un Sistema de Información Geográfico, se hicieron sobre GRASS **[19]** instalado en equipos Linux Debian.

Hubiera sido deseable probar que la visualización de los resultados se podía hacer en otro Sistema de Información Geográfico como ser ArcGis **[27]** pero este tipo de pruebas fue descartado ya que requerían una capacitación especial en ArcGis y una ampliación del generador de scripts, lo cual se considero que estaba fuera del alcance y los tiempos del proyecto.

Conclusión de pruebas de infraestructura

La aplicación funciona independientemente de donde estén instaladas sus componentes, no se detecta ninguna restricción al respecto. Por lo cual se verifica que se pueden distribuir sus componentes tal cual fue diseñado.

Se observa que la aplicación ejecuta levemente más rápido en el servidor Linux.

La aplicación ejecuta más rápido desde línea de comando que desde la interfaz Web.

El sistema es portable a otro servidor de aplicación.

7.2. Pruebas de Algoritmos

Las pruebas de algoritmos se dividen en dos, una es la prueba de que el algoritmo encuentra efectivamente los clusters, a la que llamamos prueba de funcionamiento, y la otra es la prueba de performance.

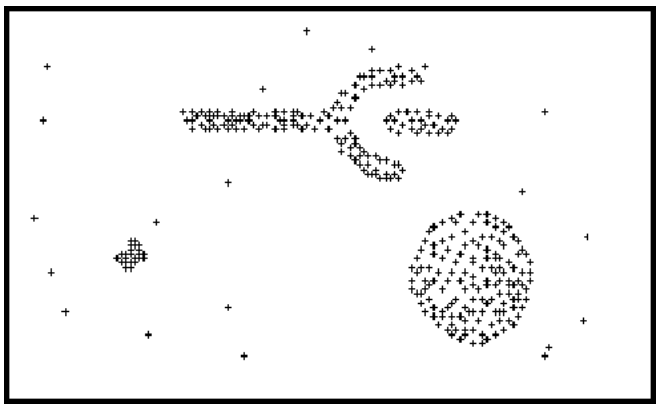
7.2.1. Prueba de Funcionamiento

Una primera prueba para verificar el buen funcionamiento del algoritmo se realiza usando un juego de datos sintético elaborado por el grupo imitando los más usados en la bibliografía manejada. Dado que no se encontró el juego de datos con las coordenadas, solo se cuenta con el dibujo, estas son generadas manualmente ara poder reproducirlo.

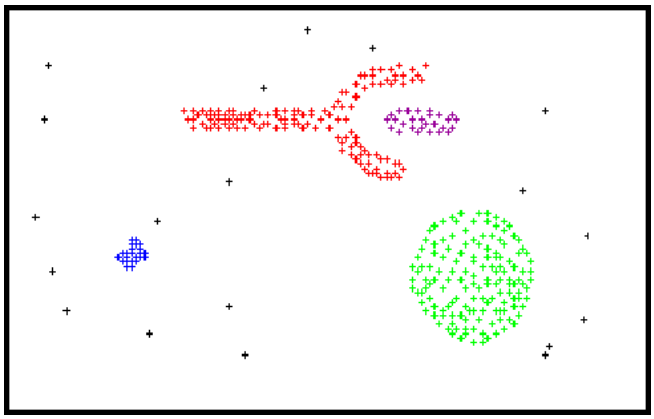
Una segunda prueba de funcionamiento se realiza con un conjunto de datos reales de 70000 puntos provisto por la DINARA. Se realizan tres tipos de evaluaciones; se compara el resultado de los algoritmos de clustering entre si, se desarrolla un algoritmo de clasificación simple llamado DataFishClassificator que se utiliza como validador de los resultados, y por ultimo se verifican los resultados obtenidos con lo esperable. Esto último se hace en conjunto con el cliente quien tiene el conocimiento de la realidad.

7.2.1.1. Prueba con datos sintéticos

En la figura siguiente se ve el conjunto de datos generados para la prueba de funcionamiento de los algoritmos. El mismo se hizo imitando el conjunto de puntos que en la bibliografía se encuentra con el nombre “DS3” **[3] [5]**.



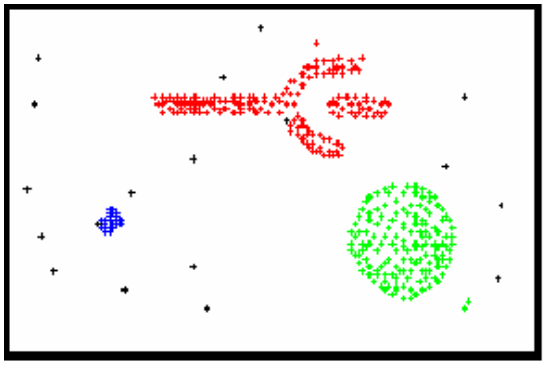
Prueba DBSCAN, DBRS y OPTICS
Los tres algoritmos de clustering utilizados (DBSCAN, DBRS y OPTICS) encuentran correctamente los clusters como lo muestra la siguiente figura. En los tres casos se utilizaron los mismos parámetros de los algoritmos, epsilon igual a 4 y mínima cantidad de puntos por cluster igual a 4.



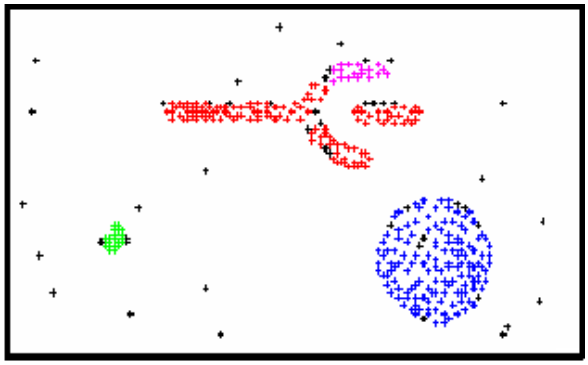
Prueba DataFishClassificatorEmpowered
Se realizan dos ejecuciones con el algoritmo DataFishClassificatorEmpowered (DFCE) con el mismo juego de datos. En la figura siguiente se ven los resultados obtenidos con las dos ejecuciones.

El cuadro Resultado 1 se obtuvo ejecutando DFCE con un ancho de celda igual a 10 y mínima cantidad de puntos por cluster igual a 3. Se puede observar que la detección de las formas es correcta pero dado el ancho de la celda no permite identificar el cluster correspondiente al medio del tridente.

Para la obtención de cuadro Resultado 2 se ejecutó el DFCE con una variación en los parámetros de entrada, ancho de celda igual a 5 y mínima cantidad de puntos por cluster igual a 3. En este caso al achicar el ancho de celda se nota que algunos puntos son evaluados como ruido dado que la densidad de puntos en esa zona no es lo suficientemente grande como para ser reconocida como una celda valida y por lo tanto no participa en la unión de celdas vecinas. Otro hecho a destacar es que la parte superior del tridente (color magenta) se separa del resto de la figura (color rojo) por no existir suficiente densidad en las celdas de la unión entre estas dos zonas.



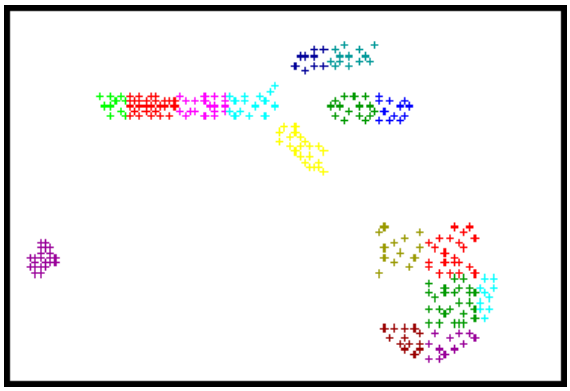
Resultado 1



Resultado 2

Prueba DataFishClassificator

En la siguiente figura se ve una ejecución con el algoritmo DataFishClassificator (DFC) utilizando como ancho de zona 10 y mínima cantidad de puntos por cluster igual a 10. Como este algoritmo no realiza la unión de celdas validas vecinas se puede observar claramente cuales son las celdas que cumplen las condiciones para ser consideradas validas.



7.2.1.2. Pruebas con datos reales

A continuación se pueden ver los resultados obtenidos con una corrida tipo de los algoritmos implementados. En todos los casos se observa que si bien existen leves diferencias entre los resultados, con todos los algoritmos se alcanzan resultados similares y lo que es más importante estos son coincidentes con los resultados esperados. La conclusión de que el resultado es el esperado, y las diferentes discusiones y comparaciones entre los diferentes algoritmos se detallan en el Capítulo 8 Análisis de Resultados.

Se utilizo una corrida tipo con 11000 puntos, usando epsilon igual a 0.1, y mínima cantidad de puntos igual a 50

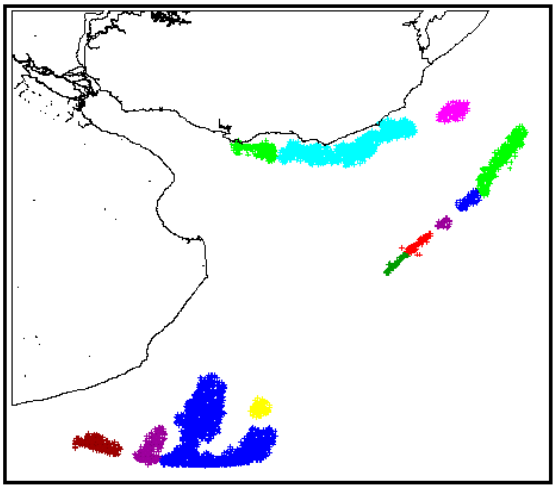


Figura 1 - DBRS

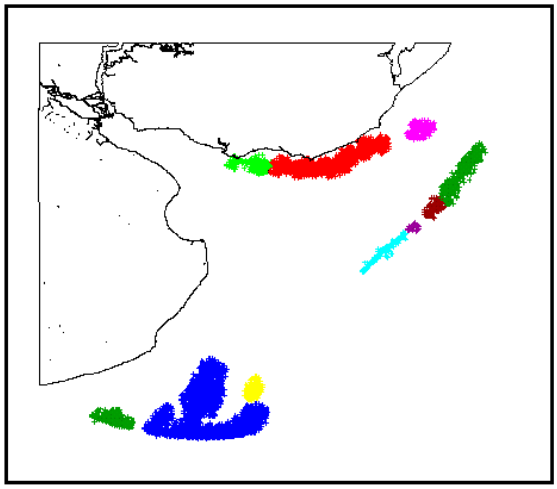


Figura 2 - DBSCAN

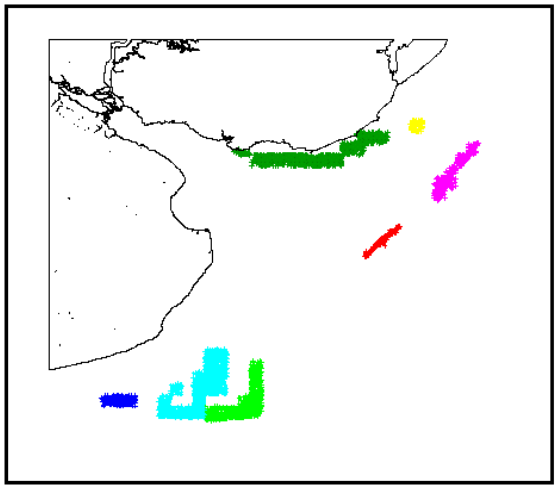


Figura 3 - DFCE

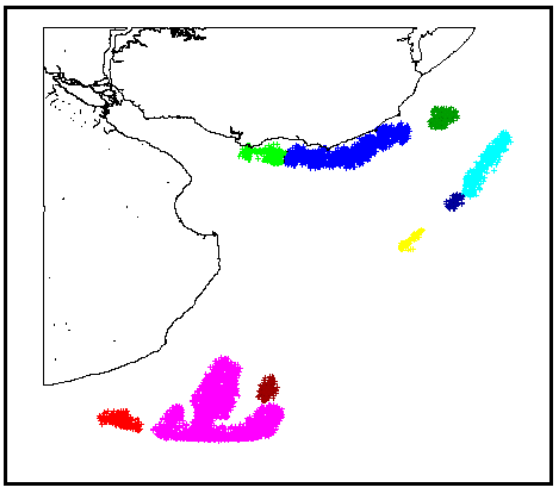


Figura 4 - OPTICS

7.2.2. Pruebas de Performance

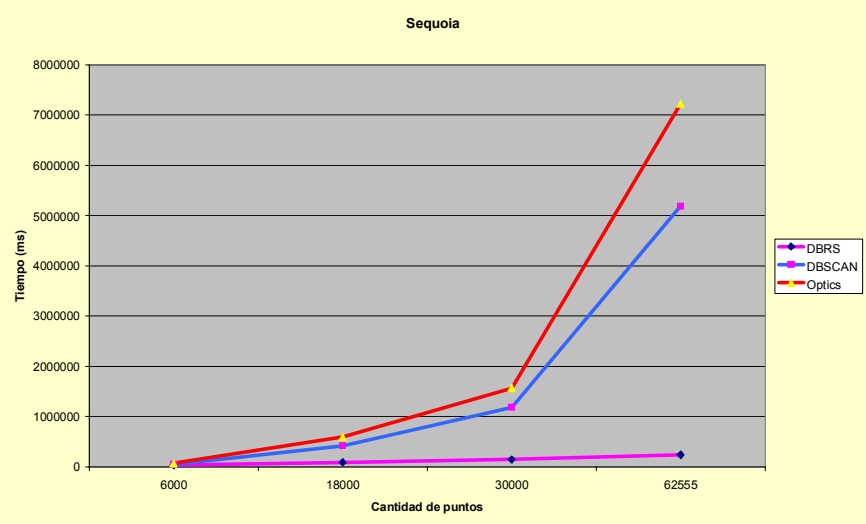
Para la evaluación desde el punto de vista de performance se decide utilizar un benchmark. Se encuentra que el más utilizado para la prueba de algoritmos de este tipo es el llamado “Sequoia”, el mismo se encuentra en la mayoría los papers utilizados en este proyecto. Se adapta la aplicación para ejecutar dicho benchmark y se realizan las pruebas y se grafica.

También se realizan pruebas con datos reales a fin de corroborar que los resultados obtenidos con la “Sequoia” se repiten con el juego de datos con el cual el sistema ***DataFish*** trabaja.

Pruebas con SEQUOIA [43]

Los algoritmos utilizados, en especial el DBRS varían drásticamente en sus tiempos frente a cambios de los parámetros de entrada (epsilon y mínima cantidad de puntos por cluster). Por lo cual los parámetros utilizados en la prueba son elegidos de forma tal que los mismos sean razonables (en cantidad de zonas encontradas) y no buscando el obtener mejores tiempos en alguno de ellos. Las pruebas se hicieron con epsilon igual a 0.1 y cantidad de puntos mínima igual a 10.

El gráfico siguiente muestra los valores comparativos obtenidos de ejecutar los algoritmos usando como juego de datos la benchmark “Sequoia”.



Conclusiones
Los resultados obtenidos están dentro de lo esperado, teniendo en cuenta que las implementaciones hechas no utilizan la estructura de optimización Rtree **[44]**.

El algoritmo OPTICS es levemente inferior en performance que el DBSCAN, aproximadamente 1.5 veces. Y el algoritmo DBRS es muy superior en performance a los otros, especialmente en juegos de datos como el de este benchmark que son altamente densos.

Pruebas con datos reales
Como se mencionó anteriormente la variación del tiempo de ejecución varía según la cantidad de puntos, los parámetros de entrada y obviamente la densidad de los mismos. Por tal motivo los resultados obtenidos no son idénticos a los de la “Sequoia” pero sí son análogos. En el Capítulo ocho de este informe se pueden ver estos resultados junto a un análisis de los mismos.

7.3. Prueba de la aplicación

Dada las características principales de la aplicación resulta difícil realizar un testeo funcional clásico. Para garantizar el correcto funcionamiento del aplicativo se realizan entonces pruebas unitarias de los distintos componentes.

Capa de Presentación

Se verifica que dados los parámetros de entrada ingresados por el usuario la invocación a la capa de negocio, componente façade, se hace con los parámetros correctos.

La otra componente de la capa de presentación es la visualización de los resultados, el testeo de esto se hizo con pruebas unitarias, con datos ficticios, de los componentes generación de scripts y acceso a datos.

Capa de Negocio

El testeo de esta capa se hizo mayoritariamente con pruebas unitarias de cada una de las componentes que la integran. Para esto se trabajo con datos ficticios de los cuales era fácil de corroborar el resultado.

El componente principal de esta capa, que es la librería de algoritmos, fue testeado en profundidad según se detalla en la Sección anterior.

8. Análisis de resultados

Este Capítulo se dedica principalmente al análisis de los resultados obtenidos con los distintos algoritmos. El objetivo fundamental es explicar como afectan las características principales de cada uno de ellos a los resultados, y de esta forma poder concluir cual aplica mejor en cada caso.

En la Sección 8.1 se muestran dos resultados y se explica porque los mismos se ajustan a lo que se conoce de la realidad.

Si bien el análisis esta hecho de cara a la realidad del sistema DataFish, esto es con el conjunto de datos del sistema SIPESAT, se desprenden de esta sección características generales del desempeño de los algoritmos.

El segundo objetivo de este Capítulo, es realizar un análisis del resultado final al cual accede la DINARA a través del uso del sistema y evaluar cual puede ser al aporte real del sistema ***DataFish*** al sistema SIPESAT.

8.1. Análisis visual de los resultados

Este trabajo se realizo en conjunto con el cliente quien tiene el conocimiento del negocio para evaluar los resultados. Para eso se realizaron varias corridas cambiando los algoritmos, los parámetros y el juego de datos aplicando diferentes filtros. Para cada uno de los resultados el cliente hizo un análisis visual corroborando que el resultado obtenido es coherente con el comportamiento ya conocido de la flota pesquera.

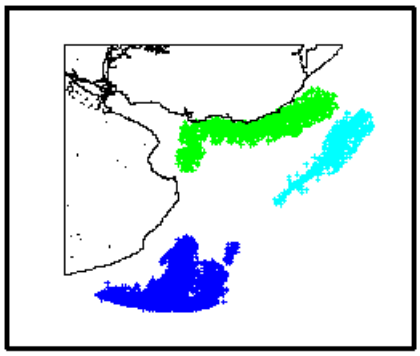


Figura 1

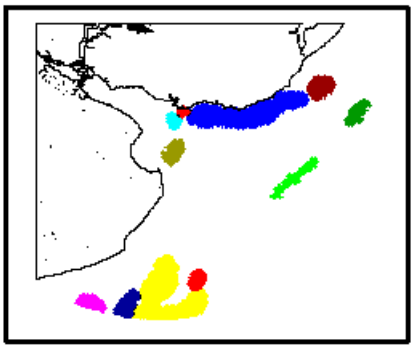


Figura 2

Las figuras muestran dos resultados del sistema ***DataFish***, en los cuales se varió la densidad de puntos requerida para formar un cluster y el algoritmo aplicado. Para la “Figura 1” la densidad requerida es menor, y para la “Figura 2” se ejecuta un algoritmo que identifica mejor las diferencias de densidad entre clusters.

En la “Figura 1” se puede observar que se encuentran tres grandes zonas, la zona costera (verde), y una zona coincidente con un escalón de profundidad (celeste y azul), donde es conocida la alta concentración de diferentes tipos de buques pesqueros.

En la “Figura 2” se observa a grandes rasgos el mismo dibujo, con menor cantidad de puntos, y subdividido en mayor cantidad de zonas. La menor cantidad de puntos clasificados se debe al requerimiento de mayor densidad, esto hace que algunos puntos, que se encuentran principalmente sobre los bordes, no se clasifiquen por no alcanzar la densidad necesaria. La mayor cantidad de zonas se explica, en parte por lo anterior, y también por utilizar un algoritmo más sensible a las variaciones de densidad entre las zonas.

8.2. Algoritmos

Con el objetivo de hacer lo más claro posible este análisis, es que el mismo se basa en ejemplos concretos realizados con datos reales. El formato consiste en plantear casos que serán comparados y explicar los mismos desde el punto de vista funcional y de performance.

En cada caso se elabora una tabla, en donde aparecen los parámetros de entrada del algoritmo, y la cantidad de puntos del juego de datos elegido. Los resultados que se exponen de cada caso, son, el tiempo de ejecución en segundos, la cantidad de zonas encontradas, la cantidad de puntos clasificados y el ruido. Cabe aclarar que el ruido son los puntos no clasificados, esto es, lo puntos que el algoritmo considera que están en una zona sin la densidad requerida como para ser clasificados.

8.2.1. Caso 1 – Consecuencias del cambio de densidad (Epsilon)

El conjunto de datos seleccionado son los puntos con velocidad entre 3 y 8 nudos, el mismo fue elegido ya que se conoce que tiene alta densidad. El caso pretende mostrar como el parámetro epsilon afecta a la performance y al descubrimiento de zonas.

La diferencia entre las dos corridas es el parámetro de entrada epsilon. Lo que se hace en la segunda corrida es aumentarlo, dejando constante la cantidad mínima de puntos requerida para generar un cluster, esto en otras palabras es disminuir la densidad necesaria para formar un cluster.

Nro. Corrida	Puntos totales	Epsilon	Puntos por Cluster
1	20414	0.1	50

Algoritmo	Tiempo (s)	Zonas Encontradas	Puntos Clasificados	Ruido
DBRS	269	10	16424	3990
DBSCAN	1248	9	17053	3361
OPTICS	1502	8	16251	4163

Nro. Corrida	Puntos totales	Epsilon	Puntos por Cluster
2	20414	0.4	50

Algoritmo	Tiempo (s)	Zonas Encontradas	Puntos Clasificados	Ruido
DBRS	60	2	19534	880
DBSCAN	1292	2	19599	815
OPTICS	1478	2	18854	1560

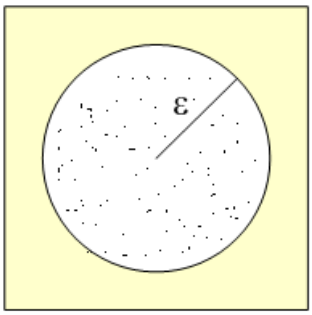
Puntos Clasificados
En ambas corridas se puede ver que la cantidad de zonas es aproximadamente igual. Se encuentra que el DBSCAN **[22]** clasifica más puntos que el DBRS **[15]**. Esto se debe a la forma de procesar los puntos de este último, que en general afecta sobre todo a las zonas de borde de los clusters. Esto hace que algunas veces el DBRS encuentre más zonas, que en realidad son los mismos puntos que el DBSCAN clasifica en menos zonas.

En la segunda corrida, se descubren menos zonas que en la primera, esto es debido a que al bajar el requerimiento de densidad más puntos son clasificados, y por lo tanto lo que antes se separaba en varias zonas ahora queda unido.

El descubrimiento de zonas del algoritmo OPTICS depende de su segunda componente que es el módulo de extracción de clusters **[5] [20] [23] [24]**, por lo que no es directamente comparable con DBRS y DBSCAN

Performance
Se puede apreciar que el cambio en el requerimiento de densidad, afecta sensiblemente a la performance del DBRS y es imperceptible para los otros algoritmos. Esto último se justifica en los puntos siguientes.

Justificación de los resultados y conclusiones
Los tres algoritmos utilizados en las corridas de ejemplo usan el concepto de densidad, esto es que la densidad dentro del cluster es mayor que fuera del mismo. La medida de densidad se obtiene en función de un epsilon (máxima distancia entre un punto y su vecino más cercano dentro de un cluster) y la mínima cantidad de puntos requerida para formar un cluster.



Esto hace que al aumentar el epsilon, más puntos puedan ser clasificados, y con esto se justifica la mayor cantidad de puntos clasificados o visto de otra forma la disminución de ruido en la segunda corrida.

El algoritmo DBSCAN como se puede leer en el Capítulo 5 y en la Extensión del Estado del Arte **[5]**, tiene orden n^2 (u orden n*log(n) si se implementa usando una estructura particular llamada RTree **[44]**). Esto es porque procesa todos los puntos y para cada uno de ellos se fija si todos los otros son vecinos de él. Entonces es constante frente al cambio en la densidad. **[22]**

El algoritmo DBRS, en cambio a medida que va procesando los puntos, si logra ubicarlos dentro de un cluster, no los vuelve a procesar. Por lo tanto, en la medida que tengamos datos lo suficientemente densos (recordar que el nivel de densidad requerido es determinable) la performance mejora notoriamente a medida que los puntos van siendo clasificados en algún cluster.

Este último punto, de no procesar los datos nuevamente, hace que el DBRS clasifique menos puntos que el DBSCAN, que son justamente aquellos puntos que suelen quedar cercanos al borde del cluster. Por más datos acerca de esto ver “A Density-Based Spatial Clustering Method with Random Sampling” **[15]**

8.2.2. Caso 2 – Consecuencias del cambio de densidad (Mínima cantidad de puntos)

Este caso es similar al anterior en el sentido que se varía la densidad requerida, ahora lo que se cambia es la cantidad de puntos mínima necesaria para formar un cluster. El conjunto de datos que se elige son los puntos dentro del espacio marítimo uruguayo, con velocidad entre 3 y 5 nudos. Se incluye en este caso también la corrida del algoritmo DFCE.

Nro. Corrida	Puntos totales	Epsilon	Puntos por Cluster
1	14756	0.1	100

Algoritmo	Tiempo (s)	Zonas Encontradas	Puntos Clasificados	Ruido
DBRS	296	10	8371	6385
DBSCAN	655	10	9200	5556
OPTICS	720	10	8382	6374
DFCE (0.2 - 100)	0.146	9	6874	7882

Nro. Corrida	Puntos totales	Epsilon	Puntos por Cluster
2	14730	0.1	50

Algoritmo	Tiempo (s)	Zonas Encontradas	Puntos Clasificados	Ruido
DBRS	130	11	12118	2612
DBSCAN	648	11	12681	2049
OPTICS	820	11	11775	2955
DFCE (0.2 - 50)	0.146	10	10907	3823

Puntos Clasificados
Como se justifica más adelante el disminuir la cantidad de puntos, el impacto es menor que al aumentar el epsilon, pero se mantiene el comportamiento del caso anterior en lo que se refiere a la cantidad de puntos clasificados. Esto es, a cuanta menor exigencia de densidad de puntos para formar un cluster, mayor es la cantidad de puntos clasificados.

El algoritmo DataFishClassificatorEmpowered, trabaja de forma muy diferente al resto de los algoritmos implementados. Este arma una grilla de ancho de zona dado, en el ejemplo 0.2, luego busca las celdas de la grilla que superan una cantidad dada de puntos, en el ejemplo de 50, y por último une las celdas vecinas que cumplen las condiciones antes dichas. Los valores de ancho de celda y cantidad de puntos guardan relación con los valores de epsilon y mínima cantidad de puntos usados en los otros algoritmos. En general el ancho de la zona debe ser superior al epsilon en más o menos el doble para obtener resultados similares, esto último es muy relativo a la densidad del juego de datos.

Los resultados obtenidos con el DataFishClassificatorEmpowered, son de menor exactitud o calidad que los del resto de los algoritmos pero si se observa que aproxima muy bien al resultado.

Performance
Al igual que en el caso anterior el algoritmo DBRS mejora su performance al disminuir la densidad requerida.

Se puede apreciar que la performance del DataFishClassificatorEmpowered es muy superior al resto de los algoritmos.

Justificación de los resultados y conclusiones
En este caso el cambio de densidad es menos significativo, esto está directamente relacionado con el juego de datos, por lo que la cantidad de zonas encontrada es similar en ambas corridas. Esto indica que las zonas son exactamente las mismas, lo único que sucede es que se abarca más extensión por el requerimiento de menor densidad, en otras palabras se extienden los bordes.

La excelente performance del algoritmo DataFishClassificatorEmpowered y su buena aproximación a los resultados hacen que sea una muy buena herramienta para una primera aproximación a las soluciones.

8.2.3. Caso 3 – Performance

Este caso tiene como objetivo mostrar la diferencia de performance entre los algoritmos, algo que ya se apreció en los casos anteriores, pero aquí se muestra más detalladamente. El juego de datos elegido para la corrida uno son más de 9000 puntos dentro del espacio marítimo uruguayo, y el segundo es un conjunto de más de 20000 puntos dentro del mismo espacio.

Como ya vimos en los puntos anteriores el algoritmo que tiene mejor performance de los basados en densidad es el DBRS por lo que en la segunda corrida solo se compara este contra el DataFishClassificatorEmpowered.

Nro. Corrida	Puntos totales	Epsilon	Puntos por Cluster
1	9575	0.1	50

Algoritmo	Tiempo (s)	Zonas Encontradas	Puntos Clasificados	Ruido
DBRS	136	12	5298	4377
DBSCAN	282	10	5725	3850
OPTICS	208	9	5429	4146
DFCE (0.2 - 50)	0.1	7	4422	5133

Nro. Corrida	Puntos totales	Epsilon	Puntos por Cluster
2	20260	0.1	50

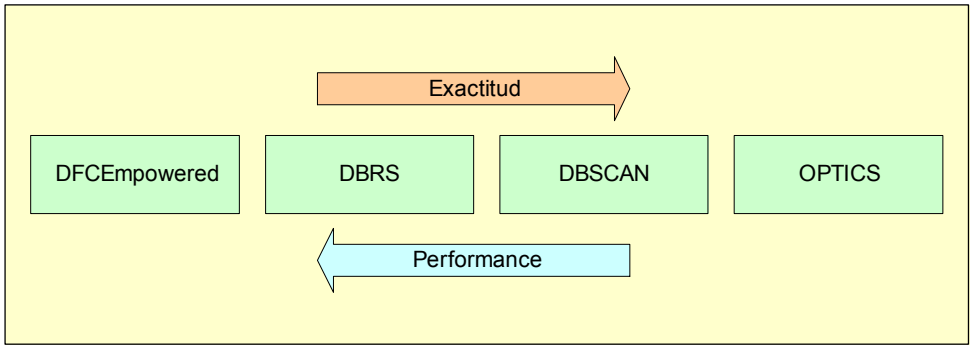
Algoritmo	Tiempo (s)	Zonas Encontradas	Puntos Clasificados	Ruido
DBRS	244	10	16424	3836
DFCE (0.3 – 50)	0.146	4	17078	3182
DFCE (0.2 – 50)	0.209	8	15148	5112
DFCE (0.1 – 30)	3.028	17	11000	9260

Justificación
En este ejemplo podemos ver como el principal factor que afecta al DataFishClassificatorEmpowered es la construcción de la grilla. Cuanto más grande es el espacio por el cual están dispersos los puntos y cuanto menor es el ancho de la celda elegida, más es el tiempo insumido por el algoritmo.

También se observa como se aproxima la relación de ancho de zona y epsilon, recordando siempre que esto es muy relativo a las características del juego de datos.

8.2.4. Conclusión de casos

En general, de lo casos expuestos y de la experiencia adquirida en la ejecución de los algoritmos con diversos juegos de datos se puede afirmar que se cumple la figura que se muestra a continuación.



Los algoritmos enumerados de la siguiente forma; DFCE, DBRS, DBSCAN y OPTICS se encuentran en un orden creciente de exactitud de resultados y en un orden decreciente en cuanto a tiempo insumido para la conclusión del algoritmo.

Por lo que parece razonable ejecutar el DFCE y/o el DBRS en grandes volúmenes de datos, para determinar una primera aproximación de las zonas, y luego ejecutar el DBSCAN y/o el OPTICS si se quieren encontrar los clusters mejor definidos.

Vale aclarar que la diferencia en exactitud entre el DBRS y el DBSCAN dependen fuertemente del juego de datos, pero en general en datos altamente densos, como los del sistema SIPESAT, la diferencia es despreciable.

Lo que ubica al OPTICS en el máximo de exactitud, es que es el único capaz de encontrar subclusters, esto es clusters más densos dentro de otros clusters. **[20][23][24]**

8.3. Sistema *DataFish*

En esta sección se comentan los resultados obtenidos por el sistema ***DataFish*** dentro del proyecto SIPESAT. Se exponen los resultados alcanzados, se dimensiona su aporte al proyecto, y su posible proyección tanto dentro del sistema SIPESAT como fuera de él.

Comportamiento de la flota pesquera
El proveer herramientas para el análisis del comportamiento de la flota pesquera es el objetivo principal del proyecto, para el cual se desarrolla el sistema ***DataFish***. Los resultados obtenidos y mostrados en las Secciones anteriores, reflejan que el sistema brinda un conjunto de herramientas (distintos algoritmos con diferentes capacidades), a través de las cuales es posible determinar las áreas donde se concentran las señales de las balizas GPS. Y de esta forma poder deducir entre otras cosas cuales son las áreas donde hay más concentración de pesca o donde hay más concentración de buques, etc... En las imágenes que siguen se visualiza la información antes de procesarla y luego de procesada por sistema ***DataFish***.

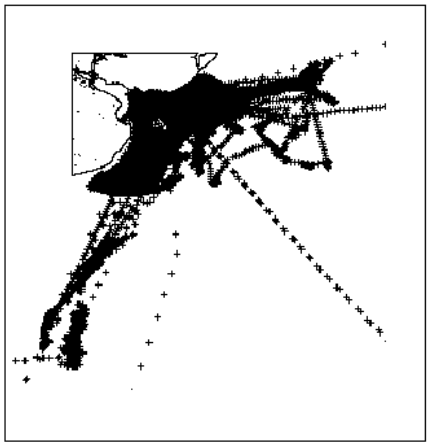


Figura 1

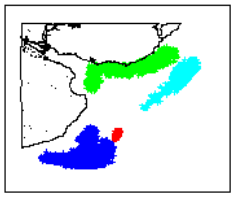


Figura 2a

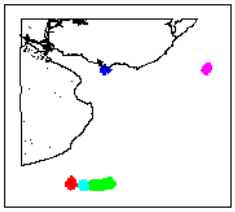


Figura 3

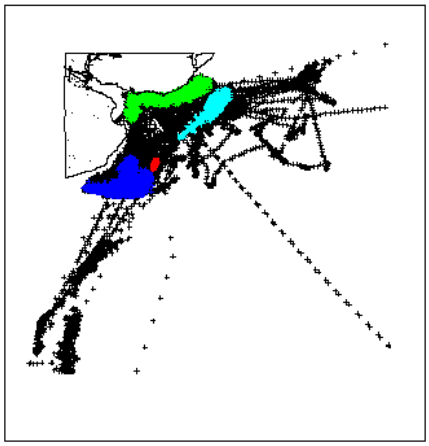


Figura 2b

En la “Figura 1” se observan todas las señales de las balizas tal cual es visualizable por la DINARA sin contar con el sistema ***DataFish***. Es claro que es muy difícil sacar alguna conclusión acerca del comportamiento de la flota en este tipo de figura.

En las Figura 2a se observa un resultado obtenido con el sistema ***DataFish*** en el cual se pretende encontrar concentraciones de buques con ciertas características. En la “Figura 2b” se contrasta el resultado obtenido con la “Figura1”. Con estas figuras es posible determinar donde y en que magnitud se encuentran las agrupaciones de embarcaciones y con esto poder deducir una característica del comportamiento de la flota.

La “Figura 3” es el resultado de una depuración de lo obtenido en la “Figura 2a”, esto es buscar agrupaciones de puntos “más densas”. Es un ejemplo de las utilidades que brinda el sistema, pudiendo el usuario iterar sobre varias soluciones para aumentar la capacidad de análisis. Esto se puede hacer variando el juego de datos y/o aplicando distintos algoritmos.

Se desarrollan varios algoritmos, ya que para distintas realidades (tanto de datos de entrada como de información buscada) algunos aplican mejor que otros, no habiendo ninguno que se destaque sobre los otros en el cien por ciento de los casos.

DataFish brinda una interfaz Web para ejecutar los algoritmos y seleccionar datos por fecha, tipo de embarcación, latitud y longitud. Agrega un generador de scripts que realiza el trabajo de creación de vectores en el Sistema de Información Geográfico. Con este sistema los usuarios de la DINARA logran de manera sencilla analizar mejor la información del sistema SIPESAT.

Se utilizaron características del Sistema de Información Geográfico para poder visualizar el área convexa que contiene a los puntos de cada zona, y poder consultar datos tales como el área total y la cantidad de puntos que contiene.

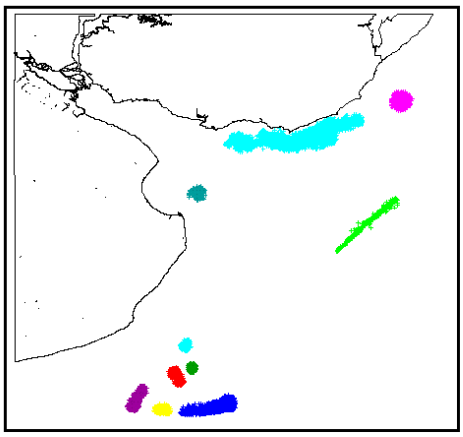


Figura 1

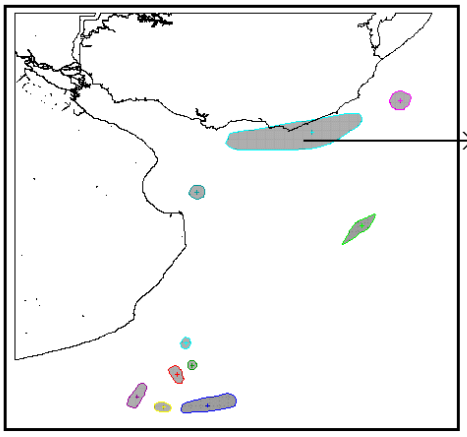


Figura 2a

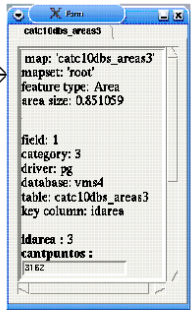


Figura 2b

En la Figura1 se observa el resultado obtenido de una ejecución del algoritmo DBSCAN, y la visualización de los puntos clusterizados, en la Figura2a se pueden ver las áreas convexas encontradas, y en la tercera (Figura 2b) se muestra el resultado de consultar a través del Sistema de Información Geográfico los datos de la zona, donde se ve el área y la cantidad de puntos.

Se entiende entonces que se alcanza el objetivo planteado en lo referente a facilitar el análisis del comportamiento de la flota pesquera, contando con una herramienta que brinda distintas posibilidades, de una manera sencilla de utilizar y visualizar.

Creación Dinámica de Vistas
Esta utilidad que los usuarios del sistema necesitaban para no tener que recurrir a un administrador de la base de datos cada vez que querían visualizar alguna información. Si bien es algo que el usuario antes podía conseguir con un poco más de trabajo, la herramienta aporta mejoras al dinamismo y agilidad de la búsqueda y al análisis de la información por parte del usuario. En otras palabras este punto, no aporta nueva información al sistema SIPESAT, lo que si aporta es agilidad al proceso de análisis de la información. Por lo que se entiende que el objetivo en este punto fue alcanzado totalmente.

Perspectiva de este proyecto
Es destacar que como resultado del análisis y la investigación realizada en este trabajo se dejan planteado varios proyectos relacionados tanto con el sistema ***DataFish*** como con el sistema SIPESAT. En el Capítulo diez de este informe, “Trabajos Futuros”, se pueden leer las perspectivas y trabajos planteados.

Análisis del comportamiento de las especies
Uno de los objetivos del sistema SIPESAT es investigar el comportamiento migratorio y poblacional de las especies. El uso actual del sistema ***DataFish*** colabora con este objetivo y da un primer paso en esta dirección. El aporte no es mayor por un problema de falta de información actualmente en el sistema SIPESAT, la cual se espera contar en el futuro.

El sistema ***DataFish*** puede ser usado como base para el análisis de mucha de la información con la que la DINARA cuenta actualmente pero que no ha podido incorporar al sistema SIPESAT. Con este aporte se van a potenciar los resultados obtenidos.

9. Conclusiones

Se realizo un estudio en amplitud de las diferentes técnicas de Data Mining existentes, lo que nos permite tener un Estado del Arte de esta rama de la informática.

Se estudiaron varias alternativas para la solución de un caso real aplicando técnicas de Data Mining, eligiéndose para resolver la problemática planteada por la DINARA.

Se determinó que la técnica de clustering sobre base de datos espaciales era la más adecuada para este problema, realizándose un estudio en profundidad de los algoritmos que se basan en esta técnica.

Queda como resultado de este trabajo un aporte de información e investigación sobre técnicas de Data Mining, en especial sobre clustering sobre bases de datos espaciales.

Con la implementación del sistema ***DataFish*** se cumple uno de los objetivos del proyecto facilitando el análisis del comportamiento de la flota pesquera por parte de la DINARA, contando con una herramienta que brinda distintas posibilidades, de una manera sencilla de utilizar y visualizar.

Queda como aporte la implementación de un módulo de librerías de algoritmos de clustering el cual puede ser utilizado en infinidad de áreas y problemas.

Otro aporte brindado por este proyecto es la evaluación comparativa de los algoritmos DBSCAN, OPTICS y DBRS, en cuanto a funcionalidad y performance. Elemento que en la etapa de investigación no se encontró, por lo que se entiende es un aporte importante.

En cuanto al aporte académico que este proyecto le brindo al equipo, se destacan, el uso de metodología en desarrollo de proyectos, la aplicación de conceptos de Data Mining a una problemática concreta, y la interacción con un cliente.

Otro aporte a destacar son:

- Especificación de requerimiento de software según el estándar IEEE-STD 830-1998
- Desarrollo de aplicaciones Web con JAVA y páginas JSP
- Aplicación de patrones de diseño, en particular MVC, Façade, Strategy, DataAccessObject
- Manejo de Sistemas Información Geográfico en particular GRASS **[19]**
- Manejo y administración de bases de datos PostgreSql **[18]**
- Proceso de revisión y verificación dentro del desarrollo de software

10. Trabajos Futuros

En este Capítulo se exponen posibles trabajos futuros a realizar con el objetivo de ampliar el sistema ***DataFish***, y agregar herramientas asociadas al sistema SIPESAT. Los trabajos expuestos van más allá de la aplicación desarrollada en este proyecto, la cual sin embargo es una buena base para la incorporación de nuevas funcionalidades de apoyo a la toma de decisiones.

En este marco se identifican tres áreas bien diferenciadas en las que se pueden desarrollar nuevos proyectos. La primera es la realización de mejoras a la aplicación ya desarrollada, incorporando los puntos y objetivos que quedaron fuera del alcance de este proyecto.

Otra área de interés es una serie de temas que surgieron en el análisis del problema resuelto en este proyecto, que abren un gran abanico de temas a desarrollar apuntando a una mejor utilización de la diversidad de datos con la que cuenta, o tiene posibilidades de contar, la DINARA.

Una tercera área es la incorporación de nuevos algoritmos a la librería de algoritmos ya desarrollada.

10.1. Mejoras a la aplicación DataFish

Visualización Web
El sistema ***DataFish*** genera los datos y los scripts necesarios para visualizar los resultados en el Sistema de Información Geográfico GRASS y esta pensado para fácilmente generar scripts para cualquier otro SIG que se desee. Sería interesante poder desplegar el resultado visual en el propio entorno Web, eliminando la necesidad de que el usuario final cuente con los conocimientos necesarios para el manejo de este tipo de sistemas. Esto se llego a evaluar en el proyecto pero quedo fuera del alcance del mismo. Se deja como referencia para trabajos futuros el link a una conocida aplicación llamada MapServer la cual permite visualizar resultados de GRASS en un Web Server (http://mapserver.gis.umn.edu/)

Aplicación asincrónica
La aplicación actualmente es sincrónica, se ejecuta la aplicación desde el navegador, se eligen los filtros deseados, se escoge el algoritmo y se ingresan los parámetros necesarios, luego el sistema informa de éxito o error, y genera los datos resultados. Tal cual esta planteado el sistema, se pueden hacer leves modificaciones para que ejecute de forma asincrónica. Con esta característica se pueden enviar varios conjuntos de datos a procesar y que el sistema realice el trabajo, por ejemplo, aprovechando horas de baja utilización de los recursos.

Soporte para mayor cantidad de puntos
Los algoritmos desarrollados ejecutan cargando todos los puntos en memoria, es claro que esto tiene un límite determinado por las características físicas de los servidores, y con ciertas cantidades de puntos se inviable su ejecución. Existen modificaciones a los algoritmos que hacen un mejor uso de la memoria, logrando de esta forma procesar más cantidad de puntos con el mismo consumo de memoria. Esta es otra área de interés que se vio en el proyecto pero quedo fuera del alcance del mismo.

Paralelización
Uno de los grandes problemas de las aplicaciones de Data Mining en general es el tiempo que lleva procesar la información, así como también la capacidad de memoria necesaria para poder ejecutar ciertos volúmenes de información. Por sus características se vio durante el proyecto que algunos algoritmos son paralelizables. Se entiende que puede ser de interés la implementación de alguno de ellos usando técnicas de paralelización.

10.2. Temas a desarrollar

El proyecto ***DataFish*** es el primer proyecto de análisis de información que se realiza en el DINARA en lo que respecta al estudio posicional de las embarcaciones tomando como datos las señales emitidas por las mismas. Al terminar este proyecto la DINARA cuenta con un sistema que le permite clasificar y estudiar el comportamiento posicional de su flota tomando como referencia la velocidad de desplazamiento de las embarcaciones. En las reuniones que se tuvo con el personal de la DINARA se manejaron varios temas de interés para ellos y que a su vez entendemos son de interés académico. Muchos de ellos son imposibles de concebir actualmente por falta de información, pero se está en vías de incorporar de alguna forma esa información. De todas formas el sistema ***DataFish*** fue siempre pensado de forma tal de contemplar alguna de esas futuras necesidades. A continuación se exponen los temas que parecen ser de más interés para la DINARA y tienen a nuestro entender algún interés académico como para la realización de futuros proyectos.

Detección de puntos de pesca
Para el análisis de información es importante saber si el punto registrado por la señal de la baliza es un punto en el cual el barco está o no efectivamente pescando. La forma de estimar esto con los datos con los que cuenta el sistema actual es por la velocidad, se sabe que este método no considera algunos de los métodos de pesca utilizados. Agregando nuevos datos al sistema SIPESAT, se entiende que un proyecto interesante puede ser el determinar si los registros de una embarcación son o no registros de pesca, por ejemplo haciendo un estudio de comportamiento por patrones usando alguna técnica de Data Mining.

Agregar datos no espaciales
Al momento de la realización de este proyecto no se cuenta con el dato de saber que especie esta pescando la embarcación en el momento del registro de la baliza. Si se contara con ese dato se puede extender el sistema ***DataFish*** para considerar estos datos no espaciales. Algunos de los algoritmos desarrollados ya soportan el agregar datos no espaciales, pero el contar con esta posibilidad abre nuevos campos de investigación en lo que se refiere a estudios de comportamiento y descubrimiento de información.

Búsqueda de comportamientos extraños
Uno de los roles de la DINARA es controlar que las embarcaciones solo pesquen aquellas especies que su permiso les autoriza. Se vio que con técnicas de Data Mining se puede determinar por ejemplo si una embarcación con un tipo de permiso dado se esta comportando diferente a las de su clase, o de forma similar a embarcaciones de otra clase.

Otro comportamiento extraño que se pretende detectar es el de la pesca en áreas de veda o áreas no autorizadas a las pesca.

Con las herramientas brindadas por ***DataFish*** es posible determinar algunos comportamientos extraños de una forma visual, pero no automática, sería de interés para la DINARA contar con un sistema inteligente que le permita detectar estos comportamientos.

Cruzamiento de datos con otras fuentes
Para el análisis del comportamiento de las especies el sistema ***DataFish*** ayuda a determinar las zonas donde las mismas se pescan. Es de interés el poder cruzar estos datos con información de temperaturas del agua, corrientes marinas, y otros no disponibles hoy pero si en un futuro, como ser volúmenes exactos de captura en ciertas áreas. Con estos datos se podría llegar a tener sistemas más inteligentes que puedan encontraran patrones de comportamiento de las especies, y con esto intentar deducir el movimiento y la variación de población de las especies.

10.3. Algoritmos nuevos

Los algoritmos desarrollados fueron los que se consideraron más adecuados para resolver el problema planteado por la DINARA. El sistema fue desarrollado de forma tal que fuera sencilla la incorporación de nuevos algoritmos, por lo que cualquier extensión puede ser incorporada al sistema ***DataFish***.

Es de esperar que en los próximos años aparezcan evoluciones de los algoritmos implementados en este proyecto, así como también se cree que los algoritmos implementados pueden ser mejorados.

Con el agregado de nueva información al sistema SIPESAT por parte de la DINARA, podría resultar de interés desarrollar algunos de los algoritmos estudiados para este proyecto como ser STING y CLIQUE **[5]**. Dado que estos se destacan en el tratamiento información de datos espaciales con varias dimensiones no espaciales.

11. Referencias

Nro.	Datos
1.	Plan de Trabajo - Anexo A
2.	Metodología de Trabajo - Anexo B
3.	Estado del Arte - Anexo D
4.	Búsqueda del cliente - Anexo F
5.	Extensión del Estado del Arte - Anexo E
6.	Especificación de Requerimientos de Software - Anexo H
7.	Minutas - Anexo O
8.	Análisis de seguimiento.doc – Anexo M
9.	Arquitectura y Diseño - Anexo I
10.	Descripción de Realidad - Anexo G
11.	Manual de Usuario.doc - - Anexo K
12.	Manual de Instalación - Anexo L
13.	Presentación Final
14.	http://grass.itc.it/gdp/tutorials.php (Ultimo acceso 1-3-2005)
15.	Wang X. and Hamilton, H. J.: DBRS: A Density-Based Spatial Clustering Method with Random Sampling. In: Proc. of the 7th PAKDD, Seoul, Korea (2003) 563 – 575 PDF disponible en CD DataFish: ../Biblografia/DBRS.pdf
16.	http://struts.apache.org (Ultimo acceso 1-3-2005)
17.	http://jakarta.apache.org/velocity/index.html (Ultimo acceso 1-3-2005)
18.	http://www.postgresql.org (Ultimo acceso 1-3-2005)
19.	http://grass.itc.it/ (Ultimo acceso 1-3-2005)
20.	Ankerst M., Breunig M. M., Kriegel H.-P., Sander J.: “OPTICS: Ordering Points To Identify the Clustering Structure”, Proc. ACM SIGMOD, Philadelphia, PA, 1999, pp 49-60. PDF disponible en CD DataFish: ../Biblografia/abks99.pdf

Nro.	Datos
21.	Sander J. O., Martin Ester, Hans-Peter Kriegel, Xiaowei Xu (1998), Density-Based Clustering in Spatial Data sets: The Algorithm GDBSCAN and Its Applications, Data Mining and Knowledge Discovery 2, 169–194 (1998), Kluwer Academic Publishers. PDF disponible en CD DataFish: ../Biblografia/gdbscan.pdf
22.	M. Ester, H. Kriegel, J. Sander, and X. Xu, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise,” *Proc. of 2nd KDD*, Portland, 1996, pp.226-231. PDF disponible en CD DataFish: ../Biblografia/dbscan.pdf
23.	Visually Mining Through Cluster Hierarchies Stefan Brecheisen Hans-Peter Kriegel Peer Kröger Martin Pfeie PDF disponible en CD DataFish: ../Biblografia/sdm04_037.pdf
24.	Automatic Extraction of Clusters from Hierarchical Clustering Representations Jörg Sander, Xuejie Qin, Zhiyong Lu, Nan Niu, Alex Kovarsky PDF disponible en CD DataFish: ../Biblografia/PAKDD03.pdf
25.	http://www.microsoft.com/windows/ (Ultimo acceso 1-3-2005)
26.	http://www-306.ibm.com/software/data/informix/se/ (Ultimo acceso 1-3-2005)
27.	http://www.esri.com/software/arcgis/index.html (Ultimo acceso 1-3-2005)
28.	http://jakarta.apache.org/tomcat/ (Ultimo acceso 1-3-2005)
29.	http://www.r-project.org/ (Ultimo acceso 1-3-2005)
30.	http://www.geovistastudio.psu.edu (Ultimo acceso 1-3-2005)
31.	http://magna.cs.ucla.edu/atlas/ http://www.cs.ucla.edu/~lc/paper/cluster_on_atlas.rtf (Ultimo acceso 1-3-2005)
32.	http://www.linux.org/dist/list.html (Ultimo acceso 1-3-2005)
33.	http://java.sun.com (Ultimo acceso 1-3-2005)
34.	Java How to Program (6th Edition) (How to Program (Deitel)) by Harvey M. Deitel, Paul J. Deitel Prentice Hall; Sixth, book only edition (August 4, 2004) ISBN – 0131483986
35.	http://java.sun.com/blueprints/patterns/MVC-detailed.html (Ultimo acceso 1-3-2005)
36.	http://java.sun.com/products/jsp/ (Ultimo acceso 1-3-2005)
37.	http://www.jdbc.org/ (Ultimo acceso 1-3-2005)
38.	http://nfs.sourceforge.net/ (Ultimo acceso 1-3-2005)
39.	http://www.debian.org/ (Ultimo acceso 1-3-2005)

Nro.	Datos
40.	http://www-1.ibm.com/servers/eserver/iseries/software/websphere/wsappserver/indexb51.html (Ultimo acceso 1-3-2005)
41.	JavaScript: The Definitive Guide, 4th Edition By David Flanagan 4th Edition November 2001 ISBN: 0-596-00048
42.	http://java.sun.com/j2se/1.4.2/docs/index.html (Ultimo acceso 1-3-2005)
43.	Michael Stonebraker, James Frew, Kenn Gardels, Jeff Meredith: The Sequoia 2000 Benchmark. SIGMOD Conference 1993: 2-11
44.	Beckmann N., Kriegel H.-P., Schneider R., Seeger B.: *"The R*-tree: An Efficient and Robust Access Method for Points and Rectangles",* Proc. ACM SIGMOD Int. Conf. On Management of Data, Atlantic City, NJ, ACM Press, New York, 1990, pp. 322-331.

12. Abreviaciones y terminología

Abreviación	Palabra
DM	Data Mining
DT	Decision Tree
SIG	Sistema de Información Geográfico
VLDB	Very Large Data Base
GRASS	Sistema de visualización de información geográfica
IEEE	Institute of Electrical and Electronics Engineers
INCO	Instituto de Computación de la Facultad de Ingeniería de la Universidad de la Republica
DINARA	Dirección Nacional de Recursos Acuáticos
WSAD	WebSphere Studio Application Developer
SIPESAT	Sistema de Información Pesquera Satelital
MVC	Model View Controller
JSP	Java Server Pages
OPTIC	Nombre de algoritmo de clustering
RN	Redes neuronales
DBSCAN	Nombre de algoritmo de clustering
DBRS	Nombre de algoritmo de clustering
CL	Clustering
DFC	DataFish Classificator
DFCE	DataFishClassificatorEmpowered
ArcGis	Sistema de visualización de información geográfica
GPS	Global Positioning System

13. Anexos

Anexo A – Plan de Trabajo

Anexo B – Metodología de Trabajo

Anexo C – Estándar de Documentación

Anexo D – Estado del Arte

Anexo E – Extensión del Estado del Arte

Anexo F – Búsqueda del Cliente

Anexo G – Descripción de Realidad

Anexo H – Especificación de Requerimientos de Software

Anexo I – Arquitectura y Diseño

Anexo J – Estándar de Implementación

Anexo K – Manual Usuario

Anexo L – Manual de Instalación

Anexo M – Análisis de Seguimiento

Anexo N – Glosario

Anexo O – Minutas