Universidad de las Ciencias Informáticas

Facultad 2

Interfaz gráfica para el diseño de experimentos en MULAN

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

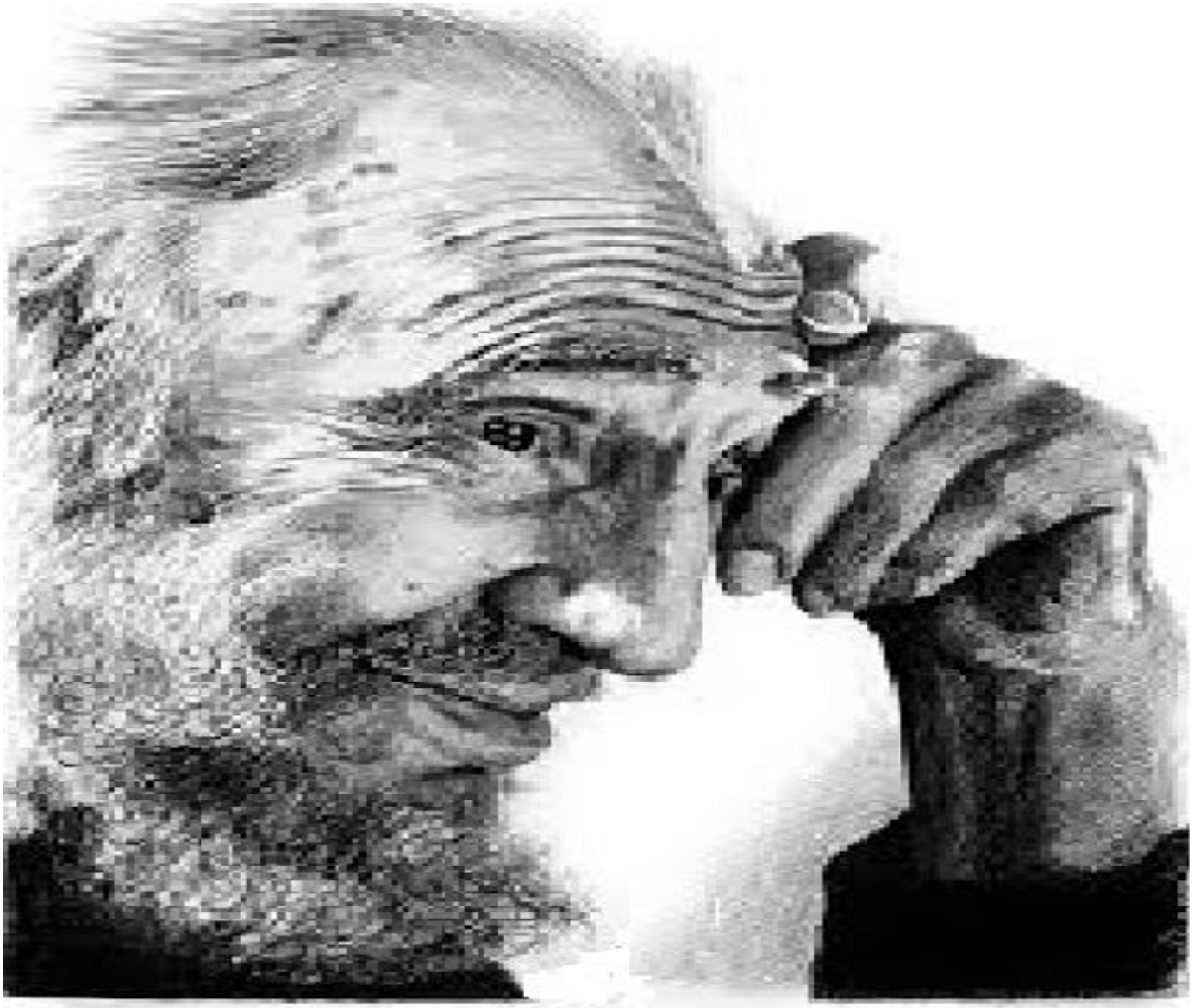
Autores: Lisandra Hernández Colás

Andy Reinoso Brito

Tutores: Msc. Héctor Raúl González Diez.

Ing. Antonio Dario Blanco Rodríguez.

La Habana, 15 de junio de 2017



"El tema relativo al conocimiento y la tecnología es de especial relieve en nuestra agenda, porque en él abordamos los problemas que deciden, en buena medida, el futuro de nuestros países."

Fidel Castro Ruz

(19-9-1999)

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la UCI los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste la presente tesis a los ___ días del mes de _____ del año _____.

_____ _____

Lisandra Hernández Colás Andy Reinoso Brito

_____

Msc. Hector Raúl Gonzalez Diez

Datos de contacto

Lisandra Hernández Colás

Dirección: Calle 16 # 4109 E/ Calle 41 y Calle 45ª, Artemisa, Artemisa

Correo electrónico: lcolas@estudiantes.uci.cu

Andy Reinoso Brito

Dirección: Edificio 2 Apto 18, zona 24, Alamar, La Habana del Este

Correo electrónico: areinoso@estudiantes.uci.cu

Dedicatoria

De Lisandra

Quiero dedicar este trabajo de diploma a mis padres por haberme traído al mundo y por confiar en mí. Por todo el amor, dedicación y empeño que han mostrado en todos estos años. Por ser un persevero ejemplo que siempre he de seguir. Por enseñarme y solidificar los valores necesarios para crecer espiritual y profesionalmente. Por su ilimitada capacidad para entenderme. Quiero que sepan que son las personas que más quiero en el mundo y espero que estén orgullosos de mí. Gracias por existir.

A mi hermano Edgar que espero que algún día siga mis pasos y llegue lejos en la vida.

A mis abuelas Evarista y Josefa por mimarme siempre y ser mi ejemplo e inspiración.

A todos mis familiares y amigos en general que estuvieron al tanto de mis estudios y aportaron lo mejor de sí, para que pudiera culminarlos exitosamente.

A todos muchas gracias.

De Andy

A mi madre (Hilda), que tal vez no sea universitaria pero se gradúa hoy junto a mí,

A mi padre (Jorge Luis), por ser mi guía en todas las situaciones de mi vida, por ser mi modelo de hombre a seguir,

A mi hermana (Arlenis), que me pidió y me enseño muchas cosas de lo que es vivir,

A mi hijo (Alan), que es lo más grande que tengo en la vida y esto de hoy tal vez ya no sea para mí, sino para él y su futuro,

A mis dos hermanos menores (Josué y Arantza), que aun en la distancia que nos separa, siempre los querré con la vida,

A mi sobrino (Brandy), que lo quiero como si fuera hijo mío.

Agradecimientos

De Lisandra:

Hoy es un día especial en mi vida y por tanto quiero agradecer esta victoria a todas aquellas personas que en todo este tiempo han formado parte de mi vida y mi universidad.

A mi mamá Mayelín, por ser mi mejor amiga, consejera, abogada defensora y jueza imparcial; es el mejor ejemplo de sacrificio, de tenacidad, de fuerza espiritual y de amor incondicional. Mi agradecimiento especial para ti, por haberme dado tanto amor, una educación que rinde sus frutos con cada paso que doy, por haberme apoyado siempre e impulsado a lograr metas altas, a luchar por lo que vale la pena, y por mis sueños, ya ves mami valieron mucho tantas lágrimas y tanto sacrificio. Hoy se hace realidad nuestro gran sueño, te quiero mucho.

A mi papá Eduardo, por confiar en mí, es un privilegio ser tu hija mayor; desde que tengo memoria estás en mi vida y, por ende, en mi corazón. Gracias por tu amor, por tus anécdotas tan ocurrentes, y por reafirmarme cuán importante es tener las ideas claras. Siempre te dije que confiaras en mí que yo nunca te iba a defraudar, espero que estés orgulloso de mí, como yo lo estoy de ti por ser tu hija. Gracias por todo, te quiero mucho.

A mis hermanos, por ser parte de mi vida e impulsarme a ser su ejemplo.

A mis abuelas Evarista y Josefa, que son las mejores, por quererme tanto, mimarme, y formar parte de lo que soy hoy.

A mi novio, por ser una fuente segura de amor y de comprensión. Gracias por tu amor y apoyo, sobre todo en este período tan complejo para ambos, por tenerme tanta paciencia y por creer en mí.

A mis tutores Héctor y Dario, por su preocupación y apoyo en este trabajo. Por su paciencia, por defenderme, por guiarme y transmitirme seguridad en este trabajo.

A todos mis amigos que han formado parte de este largo, inolvidable y hermoso camino que es la universidad. ***Gacias a todos.***

Agradecimientos

De Andy

A mis abuelas, primos, primas, tías, tíos a mis padrinos, a toda aquella persona que prendió una vela pensando en mí,

A mi cuñado (Lázaro), que bastantes consejos me ha dado para mi vida,

A mi novia (Dianelis) por ser comprensiva, atenta y paciente conmigo, por aguantarme y brindarme su amor,

A mi compadre y hermano (Abraham), quien anduvo y desanduvo junto a mí en la gran mayoría de mis vivencias en esta escuela,

A la madre de mi hijo, que mucho que me ayudó en mis inicios universitarios,

A mis hermanos de la Orden Caballero de la Luz y hermanos Masones, gracias por su comprensión y apoyo, recuerden "que somos como espigas de trigo y nuestro número es incalculable", siempre seremos hermanos,

A los buenos amigos que hice aquí y a los que tengo fuera de aquí,

A mis suegros por la acogida que me han dado y por hacerme sentir parte de la familia,

A todas las personas que han influido en mí para poder llegar a ser la persona que soy hoy en día,

A todos los profesores desde las seños del círculo hasta los tutores,

A mi compañera de tesis, por entenderme y ayudarme; porque nunca hubo discrepancia entre nosotros y hoy somos mejores amigos.

Gracias a todos.

Resumen

El presente trabajo tiene como objetivo desarrollar una herramienta gráfica para el diseño de experimentos en el contexto de la predicción con salidas compuestas. La herramienta se integra con Mulan, permite cargar múltiples bases de datos y tratar el problema predicción con salidas múltiples a través de los algoritmos basados en transformación. Se utilizaron varias bases de datos estándares disponibles para la evaluación experimental de los resultados. La aplicación de las pruebas de aceptación y unitarias realizadas al sistema mostraron satisfacción del cliente con el desarrollo de la interfaz gráfica.

Palabras Claves:

Aprendizaje Automático, Mulan, Predicción con salidas compuestas

Summary

The present work aims to develop a graphical tool for the design of experiments in the context of prediction with composite outputs. The tool integrates with Mulan, allows loading multiple databases and dealing with the prediction problem with multiple outputs through transformation-based algorithms. Several standard databases were available for the experimental evaluation of the results. The application of acceptance tests and unit tests performed on the system showed customer satisfaction with the development of the graphical interface.

Keywords:

Machine Learning, Mulan, Prediction with composite outputs

Índice de contenidos

Índice de Tablas

Índices de Ilustraciones

INTRODUCCIÓN

El **Aprendizaje Automático** (AA, o *Machine Learning***,** por su nombre en inglés) es la rama de la Inteligencia Artificial que tiene como objetivo desarrollar técnicas que permitan a las computadoras aprender (Witten & Frank, 2005). De forma más concreta, se trata de crear algoritmos capaces de generalizar comportamientos y reconocer patrones a partir de una información suministrada en forma de ejemplos. Es por lo tanto, un proceso de inducción del conocimiento, es decir, un método que permite obtener por generalización un enunciado general a partir de enunciados que describen casos particulares.

Como herramienta, el AA es útil para implementar tareas difíciles de programar (reconocimiento de caras, voz), aplicaciones auto adaptables (interfaces inteligentes, sistemas anti-spam, sistemas de recomendación) y minería de datos (análisis de datos inteligente). Este es importante para dominios de aplicación como: búsquedas web, biología computacional (bioinformática), finanzas, comercio electrónico, exploración del espacio, robótica, extracción de información en redes sociales, entre otros.

Existen problemas y tareas a resolver dentro de *Machine Learning*, aunque las dos más conocidas son el Aprendizaje Supervisado, y el No Supervisado (Bakir et al., 2007). En el caso del **Aprendizaje Supervisado** se genera una función que establece una correspondencia entre las entradas y las salidas deseadas del sistema, donde la base de conocimientos del sistema está formada por ejemplos etiquetados a priori (es decir, ejemplos de los que sabemos su clasificación correcta). Mientras que el **Aprendizaje No Supervisado** es donde el proceso de modelado se lleva a cabo sobre un conjunto de ejemplos formados únicamente por entradas al sistema, sin conocer su clasificación correcta. Por lo que se busca que el sistema sea capaz de reconocer patrones para poder etiquetar las nuevas entradas.

En particular el aprendizaje automático supervisado tiene diferentes tareas dentro de las cuales las más comunes son los problemas de clasificación y predicción (Li, Zhang, & Zhu, 2006). Los mismos son modelados a partir de un conjunto de variables predictoras y una variable de salida la cual se desea predecir o clasificar a partir de un modelo matemático aprendido. Si los posibles valores de la salida se restringen a un conjunto binario, entonces se trata de un problema de clasificación, y cuando la salida toma valores dentro de un conjunto infinito de números reales, entonces es un problema de predicción.

En la actualidad el campo de la clasificación *multi-etiqueta* también conocida en la literatura como *multi-output* o *multi-target*, es un área de creciente interés dentro del campo del aprendizaje automático, debido a la multitud de problemas que pueden ser abordados desde esta perspectiva. Estos problemas para la variable de salida, pueden ser modeladas como Grafos, Jerarquías, Secuencias, Cadenas de texto o Vectores con valores reales o binarios, dependiendo del tipo de problema que se desea estudiar. Como término general este tipo de tarea es conocida como problemas de predicción estructurada o predicción con salidas compuestas (Li et al., 2006).

De estos problemas, merece la pena citar la clasificación semántica de imágenes y textos, problemas de bioinformática relacionados con la selección de la función de proteínas y genes, asociados con el diagnóstico médico e interacciones entre medicamentos o la categorización de sonidos (José Luis Ávila Jiménez, 2013).

Diversas aplicaciones han sido estudiadas en el ámbito de la predicción con salidas compuestas, entre ellas se destacan, el campo de la quimiometría para evaluar diversos parámetros de calidad de agua (S Dzeroski, 2000) al inferir concentraciones de varios analitos de calibración multivariable, usando datos multivariados espectrales. Otro uso ha sido en la predicción de los precios de los billetes de avión para una determinada aerolínea, tomando como variables de entrada detalles de vuelos observados durante varios días como (precio, paradas, fecha de salida) (Spyromitros-Xioufis, Tsoumakas, William, & Vlahavas, 2014).Otro ejemplo es en la predicción de la concentración de metales más costosos de medir utilizando metales que son más baratos para muestrear (Goldberger, Roweis, Hinton, & Salakhutdinov, 2004), en el modelado de los componentes de sistemas ecológicos (Kocev, Vens, Struyf, & Džeroski, 2007a).

En el campo del aprendizaje automático, el problema de clasificación multi-etiqueta es de gran interés y elevada actualidad, ya que existe un grupo de algoritmos que permiten modelar este tipo de problemas de predicción con salidas compuestas, los cuales se encuentran implementados en diversas herramientas como son: Mulan (Tsoumakas, Spyromitros-Xioufis, Vilcek, & Vlahavas, 2011), Clus (Struyf, Zenko, Blockeel, Vens, & Dzeroski, 2010), Meka (Read, 2014), Weka (Durrant et al., 2013), Matlab (Guide, 1998) y Keel (Blockeel, De Raedt, & Ramon, 2000).

Dentro de los algoritmos de predicción con salidas compuestas implementados por Mulan podemos encontrar al Single Target (ST), Multi-Target Stacking (MTS), Multi-Target Stacking Corrected (MTSC), Esemble of Regressor Chain (ERC), Ensemble of Regressor Chains Corrected (ERCC), Multi-objetive RandomForesST (MORF) (Kocev et al., 2007a). En Clus Multi-objetive Random Forest (MORF) (Breiman, 2001a) y Multi-objetive Bagging (ClusBag) (Breiman, 1996a) y otros. Entre los principales algoritmos que implementa Weka se encuentran Apriori, Simple K-Means y ID3 (Durrant et al., 2013). En el caso de Matlab en este se implementan Curds and Whey (C&W) y Reduced Rank Regresion (RRR) (Izenman, 1975a). Keel es una herramienta destinada a crear las experimentaciones que contienen múltiples conjuntos de datos y algoritmos para obtener resultados. Finalmente Meka utiliza el código fuente de Mulan e implementa los algoritmos que se encuentran en la misma.

De manera resumida, luego del estudio de la herramienta solo se identifican a Clus, Mulan y Meka como aquellas que implementan problemas de predicción con salidas múltiples. Clus implementa el marco de trabajo *predictive clustering* (PC) basado en árboles de decisión y reglas de inducción, motivo por el cual no cubre una variedad de enfoques dentro del AA. La implementación de los métodos de inducción y los árboles de decisión no utilizan una herramienta de propósito general como Weka que permita su extensión a nuevos enfoques dentro del AA. Por otra parte Mulan contiene la mayor parte de los algoritmos de predicción con salidas compuestas reportados en la literatura (Eleftherios Spyromitros-Xioufis, 2016). Meka es un primer intento de interfaz gráfica para Mulan, pero tiene problemas de usabilidad porque no permite ejecutar un experimento con múltiples bases de datos y varios algoritmos de manera simultánea. Además de que posee escasa documentación del código fuente para poder extenderlo o realizar modificaciones sobre este. Teniendo en cuenta lo descrito anteriormente se hace necesario crear una nueva interfaz gráfica soportada sobre Mulan que permita a los analistas de datos sin conocimientos de programación modelar experimentos en ella de forma sencilla.

A partir de la problemática existente, se plantea como **problema de la investigación**: ¿Cómo facilitar el uso de la herramienta Mulan para que investigadores convencionales[1] de análisis de datos puedan diseñar nuevos experimentos de manera sencilla?

[1] *Son los usuarios o analistas de datos que tienen conocimientos previos en un dominio específico del conocimiento y es capaz de procesar grandes volúmenes de datos en ese dominio asistido por una herramienta.*

Teniendo en cuenta el problema antes propuesto se define como **Objeto de estudio para esta investigación**, las herramientas de aprendizaje automático no convencionales[1] que modelan algoritmos de predicción con salidas compuestas.

Definiendo como **Campo de Acción** interfaz gráfica para Mulan como herramienta de aprendizaje automático no convencional que modela problemas de predicción con salidas compuestas.

Para dar solución al problema planteado se establece como **Objetivo General** desarrollar una interfaz gráfica que permita diseñar nuevos experimentos sobre la plataforma Mulan para problemas de predicción con salidas compuestas.

Como **Objetivos específicos** se pretende**:**

1. Caracterizar las herramientas que modelan problemas de aprendizaje automático no convencionales[1] centrados en la predicción con salidas compuestas.
2. Diseñar e implementar una interfaz gráfica para el desarrollo de experimentos en Mulan.
3. Validar el funcionamiento de la herramienta propuesta a través de casos de estudio.

Los **métodos científicos** que se utilizaron son:

Métodos teóricos:

Analítico-Sintético: Se emplea para realizar un estudio sobre las herramientas de predicción con salidas compuestas, sustentadas en el estudio de disímiles fuentes de información, el procesamiento de los fundamentos científicos y las apreciaciones de numerosos autores.

Inductivo-Deductivo: Se usa para llegar a conclusiones sobre qué herramienta es más relevantes para así desarrollar la interfaz gráfica.

Modelación: Se utiliza en el diseño del modelo conceptual, los diagramas de clases, modelación de la arquitectura y diseño de la implementación

Métodos empíricos:

Observación Científica: Permitió definir la situación problemática existente y obtener resultados concretos en virtud de la aplicación de otros métodos, así como validar el correcto funcionamiento de la herramienta.

La entrevista se emplea para conocer el funcionamiento y algunas características de las herramientas y los algoritmos utilizados en la investigación.

Luego de estructurar la presente tesis investigativa de manera coherente y organizada, se describe la misma en un total de 3 capítulos de la siguiente manera:

Capítulo 1. "Fundamentación teórica de los elementos relacionados con las herramientas para algoritmos de predicción con salidas compuestas": Se realiza un estudio relacionado con los principales conceptos y características de la predicción con salidas compuestas, además de realizar un análisis de las principales herramientas utilizadas. Se describe la metodología y el lenguaje de programación seleccionado para realizar la implementación del sistema

Capítulo 2. "Exploración y Planificación": Se describen las características de la solución propuesta, definidas a partir de la modelación de los procesos y la especificación de los requisitos de software. En esta fase se describen los requisitos funcionales a través de las historias de usuario, y se realiza una planificación para darle cumplimiento a las historias de usuario seleccionadas, así como el conjunto de tareas que se ejecutan para su terminación. Además de la planificación y entrega de la aplicación.

Capítulo 3. "Implementación y Resultados": En este capítulo se realiza el diseño del sistema presentándose además los resultados de la fase de implementación y pruebas con el objetivo de solucionar errores y no conformidades presentes en la implementación.

CAPÍTULO 1. Fundamentación teórica de los elementos relacionados con las herramientas y algoritmos para problemas de predicción con salidas compuestas

1.1. Introducción

En este capítulo se relacionan las herramientas y los algoritmos para los problemas de predicción con salidas compuestas, así como en qué consisten estos problemas. Se realiza un estudio de los sistemas similares relacionados con este tipo de problema, además de que se estudian y describen las herramientas existentes que implementan estos algoritmos. Finalmente se describen las herramientas y la metodología de desarrollo de software empleada para elaborar una interfaz gráfica acorde a los requisitos planteados.

1.2. Problemas de predicción con salidas compuestas

Un caso particular de los problemas con salidas estructuradas es la predicción con salidas compuestas donde la variable de salida es un vector de valores reales. De manera general se puede afirmar que, un problema de predicción con salidas compuestas intenta predecir simultáneamente y con un único modelo todos los atributos (reales) del espacio de salida expresada en forma vectorial (Borchani, Varando, Bielza, & Larrañaga, 2015).

Para formalizar este tipo de problemas planteamos la siguiente definición, sean x, y dos vectores aleatorios donde del espacio de entrada y en el espacio de salida para una instancia determinada. Dado el conjunto de instancias de entrenamiento, el objetivo en un problema de predicción de salidas compuestas es aprender un modelo de modo que dada una instancia con entrada conocida es posible predecir, a través de un único modelo, todas las salidas de forma simultánea (Spyromitros-Xioufis, Tsoumakas, William, & Vlahavas, 2014).

En el artículo (Borchani et al., 2015) se establecen dos grandes categorías para los problemas de predicción con salidas compuestas. El primer enfoque contiene los Métodos de Transformación de las salidas, que transforman los problemas de salidas múltiples en diferentes problemas de predicción de una salida; mientras que el otro enfoque se basa en adaptar métodos clásicos conocidos de aprendizaje automático para estimar de manera simultánea las múltiples salidas o sea Adaptación de métodos (o enfoques globales).

En ambas categorías los métodos pueden tener en cuenta o no la interdependencia entre las variables de salidas, aunque de manera natural el segundo enfoque establece esta relación de forma directa.

- Los métodos basados en transformación (también conocidos como métodos locales) transforman los problemas de salida múltiple en problemas independientes de salida única resolviendo cada uno usando un algoritmo de salida simple. Dentro de los métodos basados en transformación se encuentran el MTS, ERC, ERCC y MTSC (Spyromitros-Xioufis, Tsoumakas, William, & Vlahavas, 2014).
- Los métodos basados en adaptación (también conocidos como métodos globales, o métodos *big-bang*) adaptan un método específico de salida simple para el manejo directo en un método de salida múltiple. Dentro de los métodos de adaptación se encuentran MORF (Tsoumakas, Spyromitros-Xioufis, Vrekou, & Vlahavas, 2014), RRR (Izenman, 1975b), C&W (Breiman & Friedman, 1997).

1.3. Algoritmos de predicción con salidas compuestas basados en transformación

Sucesivamente se enuncian los principales algoritmos de predicción con salidas compuestas propuestos en (Spyromitros-Xioufis et al., 2014) que extienden de los enfoques *Stacking y Chain Code*, aplicados a problemas de clasificación multi-etiqueta, los cuales clasifican en la categoría de transformación de las salidas. La comparación de estos enfoques muestra que existen mejoras significativas en aquellas variantes del algoritmo que explotan la interdependencia entre las variables de salidas.

1.3.1. Multi-target Stacking (MTS)

El entrenamiento en MTS está compuesto por dos etapas fundamentales: en la primera se aprenden modelos de salidas simples independientes y en la segunda se aprende un segundo conjunto de meta-modelos por cada salida. Para predecir las salidas de una instancia desconocida se aplica la primera etapa obteniéndose un vector de salida, luego al aplicar la segunda etapa en un vector de una entrada transformado se produce el vector final de salida.

1.3.2. Ensemble of Regressor Chains (ERC)

El método de *Regressor Chains* (RC) se basa en la idea de modelos de encadenamiento de salidas simples. El entrenamiento del RC consiste en seleccionar una cadena aleatoria (permutación) del conjunto de variables de salida y luego construir un modelo de regresión separado para cada salida.

Una propiedad notable de RC es que es sensible en el ordenamiento de la cadena seleccionada. El principal problema que surge de la utilización de una sola cadena al azar, es que las salidas que aparecen antes en una cadena no pueden modelar posibles relaciones estadísticas con las salidas que aparecen más adelante en la cadena. Además el error de predicción es probable que sea propagado y amplificado a lo largo de la cadena cuando hacemos predicciones para una nueva instancia de prueba. Para mitigar estos efectos, se propone un esquema de conjunto llamado *Ensemble of Classifier Chains* donde una prueba de k (típicamente k=10) modelos *Classifier Chains* (CC) con cadenas de diferente orden son construido sobre muestras de arranque del conjunto de entrenamiento y las predicciones finales provienen de la votación por mayoría. Este esquema ha demostrado mejorar consistentemente la exactitud de un solo CC en el dominio de clasificación. Se aplicala misma idea (sin muestreo) en RC y se calcula las predicciones finales tomando la media de las estimaciones de para cada objetivo. El método resultante se llama *Ensemble of Regressor Chains.*

1.3.3. Multi-target Stacking y Ensemble of Regressor Chains (MTSC y ERCC)

El MTSC y el ERCC utilizan un enfoque interno *f-fold cross validation* para obtener estimados fuera de la muestra para las variables de salida. El estimado de esta validación cruzada puede ser menos preciso que los obtenidos durante la predicción ya que los modelos son hechos usando el del conjunto de entrenamiento en su totalidad.

1.3.4. Multi-objetive Bagging (Clusbag) y Multi-objetive Random Forest (MORF)

Clusbag y MORF surgen al aplicar métodos de *ensamble Bagging* (Breiman, 1996b) y *Random Forests* (Breiman, 2001b) respectivamente al método *Multi-objective decision trees* (MODTs) (Kocev, Vens, Struyf, & Džeroski, 2007b).

Los métodos de ensamble no son más que un conjunto de clasificadores construidos con un algoritmo dado. Cada nuevo ejemplo se clasifica mediante la combinación de las predicciones de cada clasificador desde el conjunto. Estas se pueden combinar tomando el promedio (para las tareas de regresión) o el voto de la mayoría (para las tareas de clasificación), como se describe por Breiman ("Bagging predictors," Machine learning, 1996), o tomando combinaciones más complejas (Ho, Hull, & Srihari, 1994) (Kittler, Hatef, Duin, & Matas, 1998).

Para aplicar *Bagging a MODTs*, el procedimiento de *Predictive Clustering Trees PCTs* se utiliza como un clasificador base. Para la aplicación de *Random Forests*, se sigue el mismo enfoque, cambiando el procedimiento *BestTest* de tomar un subconjunto de tamaño aleatorio f(x) de todos los atributos posibles. Con el fin de combinar la salida de las predicciones por los clasificadores base, se toma la media de la regresión, y se aplica un voto distribución de probabilidad en lugar de una simple mayoría de votos para la clasificación, según lo sugerido por Bauer y Kohavi (Bauer & Kohavi, 1999).

1.3.5. Curds and Whey (C&W)

La idea general del algoritmo *Curds and Whey* es tomar las regresiones de mínimos cuadrados, y luego de modificar los valores previstos de esas regresiones por la contracción de ellos, utilizando las correlaciones canónicas entre las variables de salida y las variables predictoras. Breiman y Friedman en su trabajo propone la contracción simultánea tanto en el espacio de entrada como en el de salida.

El enfoque estándar de determinación de la matriz de coeficientes es a través del OLS, donde el resultado que se obtiene es equivalente a la regresión de cada una de las variables de salida a través de las variables predictoras de forma separada. Para este enfoque el error predictivo es pobre en presencia de una correlación alta entre las variables predictivas o cuando existen una cantidad significativa de este tipo de variables.

1.3.6. Reduce Rank Regretion (RRR)

RRR es un método de estimación explícita en la regresión multivariante, que tiene en cuenta la restricción de rango reducido en la matriz de coeficientes. Una incidencia directa del RRR sobre la regresión lineal es el número de restricciones lineales sobre la matriz de coeficientes de la regresión, permitiendo que el número de parámetros efectivos se reduzca y la eficiencia de la estimación aumente.

1.3.7. K - Vecinos más Cercanos (KNN)

El algoritmo de aprendizaje automático K-Vecinos más Cercanos (Cover & Hart, 1967) (KNN por sus siglas en inglés) está considerado uno de los de mejor desempeño en el ámbito del aprendizaje automático. Al mismo tiempo, el KNN construye un modelo sencillo para resolver problemas de predicción el cual está basado en los objetos del conjunto de datos de entrenamiento. En KNN para un nuevo ejemplo, se asigna al atributo de la clase, el valor medio de las clases de los objetos que se encuentran dentro de la frontera de los K-vecinos más cercanos.

1.4. Herramientas para la resolución de problemas de predicción con salidas compuestas.

A continuación se presenta una breve descripción de las herramientas mencionadas anteriormente los cuales implementan algoritmos para problemas de predicción con salidas compuestas.

1.4.1. Mulan

Mulan es una librería de código abierto escrita en Java para el aprendizaje automático, incluye una variedad de algoritmos que solucionan tareas de aprendizaje multi-etiqueta como la clasificación, el *ranking* o la combinación de estas dos. Además permite la selección de características y la evaluación de un grupo de métricas a través de la validación cruzada y evaluación *hold out* (Tsoumakas et al., 2011). Mulan está escrita en Java y se construye encima de Weka para emplear todos los recursos que contiene sobre algoritmos de aprendizaje supervisado. Una característica exclusiva de la librería Mulan es la introducción reciente de un paquete de experimentos que tiene como objetivo reproducir los resultados de los experimentos en un documento de aprendizaje multi-etiqueta.

Esta herramienta actualmente no posee interfaz gráfica aunque se puede decir que la herramienta Meka fue un primer intento para agregarle una interfaz gráfica a esta plataforma pero esta no permite ejecutar todos los algoritmos que están implementados en la misma, ni cubre todas las funcionalidades y los beneficios que brinda en la actualidad la plataforma Mulan. En la misma se encuentran implementados varios algoritmos entre ellos los siguientes: *Single Target* (ST), *Multi-Target Stacking* (MTS), *Multi-Target Stacking Corrected* (MTSC), *Esemble of Regressor Chain* (ERC), *Ensemble of Regressor Chains Corrected* (ERCC), *Multi-objetive RandomForesST* (MORF).

1.4.2. Clus

Clus es un sistema que implementa el marco de trabajo *predictive clustering* (PC) basado en árboles de decisión y reglas de inducción. Este marco de trabajo unifica los métodos no supervisados y los modelos predictivos permitiendo de forma natural la extensión a modelos predictivos más complejos tales como aprendizaje multitarea y clasificación multi-etiqueta (Blockeel et al., 2000). Requiere instancias pero no etiquetas y sirve para entender y resumir los datos. El objetivo de esta herramienta es encontrar grupos de instancias tales que las instancias en un *cluster* sean similares entre sí, y diferentes de las instancias en otros *clusters*. Entre sus aplicaciones están agrupar genes y proteínas con similar funcionalidad, reducir el tamaño de conjuntos de datos grandes y agrupar documentos para explorarlos más rápido. Esta herramienta no posee interfaz gráfica y está limitada solo para modelos basados en árboles de decisión. Se encuentra integrada a la herramienta MULAN desarrollada sobre tecnología JAVA y permite a través de un *wrapper* la ejecución desde MULAN de los algoritmos que se encuentran implementados en esta.

1.4.3. Meka

Meka es una herramienta que permite la clasificación multi-etiqueta que se encarga fundamentalmente de los algoritmos de transformación. Esta fue un primer de interfaz gráfica para la herramienta Mulan pero no cumple con todas las expectativas que presenta la misma, debido a que no tiene elementos para diseñar flujos, ni para diseñar un experimento completo donde se ejecuten varios algoritmos de manera simultánea, y además no permite ejecutar todos los algoritmos que están implementados en Mulan.

1.4.4. Weka

WEKA, acrónimo de *Waikato Environment for Knowledge Analysis*, es una herramienta visual de libre distribución bajo licencia GPL desarrollada por un equipo de investigadores de la Universidad de *Waikato* (Nueva Zelanda). Es un entorno para experimentación de análisis de datos que permite aplicar, analizar y evaluar las técnicas más relevantes de análisis de datos, principalmente las provenientes del aprendizaje automático, sobre cualquier conjunto de datos del usuario. Para ello únicamente se requiere que los datos a analizar se almacenen con un cierto formato, conocido como ARFF (*Attribute-Relation File Format*) (Durrant et al., 2013).

WEKA se distribuye como software de libre distribución desarrollado en Java. Está constituido por una serie de paquetes de código abierto con diferentes técnicas de pre-procesado, clasificación, agrupamiento, asociación, y visualización, así como facilidades para su aplicación y análisis de prestaciones cuando son aplicadas a los datos de entrada seleccionados. Estos paquetes pueden ser integrados en cualquier proyecto de análisis de datos, e incluso pueden extenderse con contribuciones de los usuarios que desarrollen nuevos algoritmos. Con el objetivo de facilitar su uso por un mayor número de usuarios, WEKA incluye una interfaz gráfica de usuario para acceder y configurar las diferentes herramientas integradas.

Esta herramienta es útil para lograr reducir el nivel de complejidad que conlleva la obtención de datos del mundo real, así como la evaluación de su salida, proporcionado una ayuda flexible para la investigación de aprendizaje automático. Incluye algoritmos que permiten resolver problemas de clasificación, clasificación multiclase y de regresión. No permite resolver problemas de aprendizaje a gran escala. Está integrada a la herramienta MULAN y permite la ejecución desde MULAN de los algoritmos que se encuentran implementados en esta.

1.3.5. Matlab

Matlab es un entorno de computación y desarrollo de aplicaciones totalmente integrado, orientado para llevar a cabo proyectos en donde se encuentren implicados elevados cálculos matemáticos y la visualización gráfica de los mismos. Cuenta con un lenguaje de programación propio (lenguaje M), disponible para las plataformas Unix, Windows, Mac OS X y GNU/Linux. Matlab integra análisis numérico, cálculo matricial, proceso de señal y visualización gráfica en un entorno completo donde los problemas y sus soluciones son expresados del mismo modo en que se escribirían tradicionalmente, sin necesidad de hacer uso de la programación tradicional.

Su interfaz gráfica de usuario (GUI - *Graphical User Interface*), es la forma en que el usuario interactúa con el programa o el sistema operativo de una computadora, y en su ventana gráfica se muestra la salida de todos los comandos gráficos (Guide, 1998). Dentro de sus características se encuentran: cálculos intensivos desde un punto de vista numérico, gráficos y visualización avanzada, lenguaje de alto nivel basado en vectores, arreglos y matrices; y una colección muy útil de funciones de aplicación.

1.3.6. Keel

El software está destinado a crear las experimentaciones que contienen múltiples conjuntos de datos y algoritmos para obtener resultados. Ofrece al usuario una interfaz amigable, orientada al análisis de algoritmos. Los experimentos son modelados gráficamente, basados en flujos de datos y representados mediante grafos y conexiones arco-nodo. Permiten escoger el tipo de validación a emplear (validación cruzada de k-particiones o validación cruzada 5x2) y el tipo de aprendizaje (regresión, clasificación o no supervisado). Los experimentos son independientemente script generados a partir de la interfaz de usuario para una línea de ejecución en cualquier máquina que soporta una máquina virtual de Java.

La interfaz genera un árbol de directorios que contiene los *datasets*, los ejecutables de los algoritmos y los *tests* estadísticos, y un script con la secuenciación de los experimentos. El diseño de experimentos de la interfaz gráfica sirve para definir gráficamente experimentos. Su propósito es generar una estructura de directorios con los ficheros necesarios para poder ejecutar el experimento diseñado en una máquina local que el usuario decida. Una vez almacenada esta estructura en la máquina seleccionada el usuario podrá lanzar el experimento ejecutando una clase Java que se encargará de ejecutar toda la experimentación diseñada.

1.3.7. Comparación de las herramientas

En un artículo publicado en marzo del 2017 (Sebastian Ventura, 2017) sobre software de código abierto para minería de datos se plantea la disponibilidad de algunas de las herramientas descritas y la posibilidad de una mejor comprensión de los enfoques proporcionando a la comunidad de investigadores una gran oportunidad para sintonizar y mejorar la experimentación.

El procedimiento de evaluación llevado a cabo en esta investigación sigue un marco de puntuación basado en tres categorías principales (rendimiento, funcionalidad, usabilidad) para evaluar las herramientas. Esta metodología fue descrita por el Centro para *Data Insight (CDI)* en la Universidad del Norte de Arizona como experiencia para evaluar las herramientas de minería de datos.

Cada herramienta se evalúa siguiendo un puntaje específico para cada una. En lugar de anotar en una escala artificialmente absoluta, se utiliza un procedimiento (valores de 1 a 5) tomando una herramienta específica como referencia, la cual recibe un puntaje de 3 para cada criterio, mientras que el resto de las herramientas son valoradas contra la referencia aplicando la siguiente escala de clasificación: (1) mucho peor que la referencia; (2) peor que la referencia; (3) igual que la referencia; (4) mejor que la referencia; (5) mucho mejor que la referencia. Todas estas puntuaciones se asignan a cada herramienta teniendo en cuenta las siguientes categorías:

El primer análisis realizado está relacionado con el **rendimiento** de cada una de las herramientas. El criterio varía según las plataformas en las que se puede ejecutar el sistema. La arquitectura del software ya sea por el tamaño de los datos, la variedad de fuentes de datos manejados y la eficiencia de los algoritmos. Además de la capacidad de interoperar con otras herramientas.

La **funcionalidad** se mide como la inclusión de una variedad de capacidades, técnicas y metodologías para la extracción de datos. La evaluación de esta categoría se basa en varias características tales como: cantidad y variedad de algoritmos, diversidad de tipos de datos manejados por la herramienta; capacidad de afinar los algoritmos y la capacidad de exportar los resultados para su uso continuo.

En un tercer análisis, el procedimiento de evaluación se centra en la **usabilidad** que describe la facilidad con la que el sistema puede ser utilizado, la visualización de los datos y los resultados, teniendo en cuenta que el criterio de evaluación varía en este caso según la interfaz de usuario.

De las categorías mencionadas anteriormente en esta investigación solo se evaluarán algunas características en cada una de las herramientas. En el caso del rendimiento se tendrán en cuenta las plataformas en las que se puede ejecutar el sistema y la capacidad de interoperar con otras herramientas. Para la categoría funcionalidad se evalúa fundamentalmente la variedad de algoritmos; y en el caso de la usabilidad el criterio de evaluación varía según la interfaz de usuario. Teniendo en cuenta que la herramienta de referencia es Mulan, las puntuaciones se modifican de acuerdo a las ponderaciones asignadas a cada característica (ver tabla 1).

Herramientas		Mulan	Clus	Meka	Weka	Matlab	Keel
Características	**Rendimiento**						
	Plataformas de ejecución	3	2	2	3	2	3
	Capacidad de interoperar con otras herramientas	3	2	1	2	2	3
	Funcionalidad						
	Variedad de algoritmos	3	2	2	3	3	2
	Variedad de datos	3	2	2	2	2	2
	Salidas múltiples	3	3	3	1	1	1
	Usabilidad						
	Interfaz gráfica	3	3	4	4	3	5
Promedio		**3**	**2.3**	**2.3**	**2.5**	**2.16**	**2.7**

Tabla 1. Criterios de evaluación de las herramientas.

Mulan, que es la referencia demuestra que muchas de las herramientas de código abierto analizadas anteriormente son muy prometedoras. La diferencia obtenida entre todas estas herramientas describe que algunas de estas se comportan de manera similar. Para resumir el procedimiento ver (ilustración 14, Anexo 1) que muestra las puntuaciones obtenidas para cada herramienta. Los resultados revelan que Mulan con una puntuación de 3 puntos es la mejor herramienta, mientras que Clus y Meka con 2.3 puntos respectivamente se comportan de forma similar y Keel con 2.7 de puntuación es la más próxima a la referencia de acuerdo con las características descritas en la Tabla 1, destacando en este caso que Mulan no tiene interfaz gráfica por lo que Keel con 5 puntos en este aspecto supera a la referencia actualmente. Teniendo en cuenta que si se implementara, con determinadas facilidades de uso, una interfaz gráfica sobre la plataforma Mulan se lograría un estado deseado como el siguiente: Clus con 1.7, Meka con 2, Weka con 2.2, Matlab con 1.8 y Keel con 2.3 puntos. En estas condiciones se lograría una herramienta significativamente mejor en cuanto a las características que se analizan en la presente investigación.

1.5. Metodología de desarrollo

Según (Roger S. Pressman, 2007) define una metodología como un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un software. Según la filosofía de desarrollo se pueden clasificar las metodologías en dos grupos, las metodologías tradicionales o pesadas que se basan en una fuerte planificación durante todo el desarrollo, como por ejemplo, Métrica V3, *Open Source Development Software* y RUP (*Rational Unified Process*).

Por otra parte las metodologías ágiles, son aquellas en las que el desarrollo de software es incremental, cooperativo, sencillo y adaptado, como por ejemplo *eXtreme Programming* (XP), *Scrum, Crystal Clear*, *Feature -Driven Development* (FDD), *Dynamic Systems Development Method* (DSDM), *Adaptive Software Development* (ASD) y *Agile Unified Process* (AUP).

1.5.1. Programación Extrema

Para emprender el desarrollo del proyecto se decidió utilizar XP (Programación Extrema, en inglés *Extreme Programming*) (Ing. Software (Equipo 02), n.d.), debido a que los elementos que aporta se adaptan en gran medida a las condiciones de trabajo que se imponen para la herramienta a desarrollar. Además de que es sencilla, genera pocos artefactos, y es utilizada para proyectos de corto plazo, pequeños equipos de desarrolladores y breve tiempo de entrega.

Los requisitos tienden a cambiar frecuentemente y según vaya avanzando el trabajo, se pueden agregar nuevas historias de usuario (HU), dividirlas e incluso eliminarlas. La implementación se divide en iteraciones cortas y en cada iteración se entrega una versión funcional de la herramienta.

El ciclo de vida de XP consiste en 4 fases fundamentales:

- Exploración: En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto.

- Planificación de la Entrega: Se establece la prioridad de cada historia de usuario, se estiman los esfuerzos que requieren cada una de las tareas al igual que el tiempo de duración de cada una de ellas.

- Iteraciones: Se incluyen varias iteraciones sobre el sistema antes de ser entregado, las mismas no deben exceder las 3 semanas.
- Producción: En esta fase se realizan pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. De igual forma se toman decisiones sobre añadir nuevas características al sistema.

Con el fin de definir cada una de las etapas y sus artefactos en el transcurso del desarrollo de la aplicación, se decidió ajustar el ciclo de vida a la propuesta de solución como sigue:

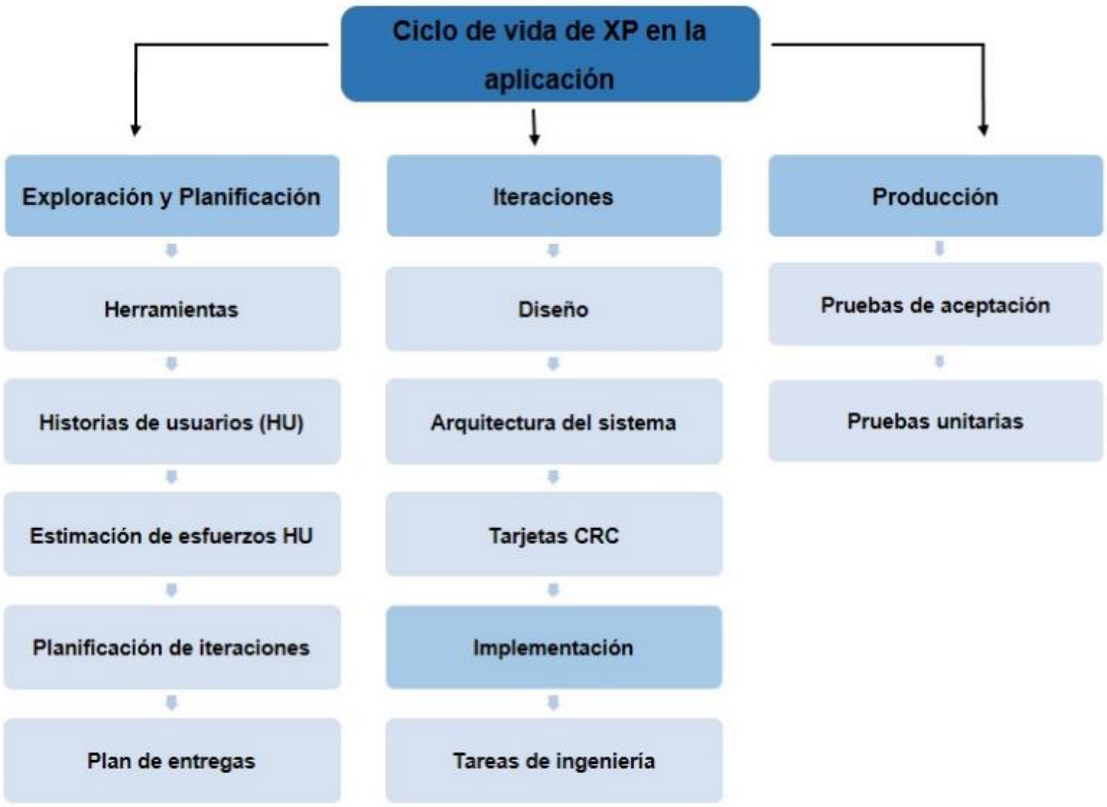


Ilustración 1. Ciclo de vida XP en la aplicación.

1.6. Lenguaje de programación

Un lenguaje de programación es un conjunto de reglas, notaciones, símbolos y/o caracteres que permiten a un programador poder expresar el procesamiento de datos y sus estructuras en la computadora (Oscar Belmonte Fernández, 2004). En esta investigación se utilizara JAVA, potente lenguaje de programación, simple y sencillo de aprender. Se propone su uso muy ligado a la selección de la plataforma MULAN.

1.6.1. JAVA

La principal característica de Java es la de ser un lenguaje compilado e interpretado. Todo programa en Java ha de compilarse y el código que se genera es interpretado por una máquina virtual (Adamson, 2006). De este modo se consigue la independencia de la máquina, el código compilado se ejecuta en máquinas virtuales que si son dependientes de la plataforma. Java es un lenguaje orientado a objetos de propósito general, se puede utilizar para construir cualquier tipo de proyecto. Finalmente posee un gestor de seguridad con el que poder restringir el acceso a los recursos del sistema.

1.7. Herramienta de desarrollo

Para la implementación del sistema se utilizó *NetBeans* en su versión 8.0 debido al número de prestaciones que brinda y su especialización para ambientes de escritorio. Es un producto de código abierto, con todos los beneficios del software disponible en forma gratuita, el cual ha sido examinado por una comunidad de desarrolladores. Proporciona varias ventajas como es el caso de la facilidad de uso durante todo el ciclo de desarrollo y el soporte para lenguajes de modelado, lo que aumenta la productividad, y las razones que motivaron su uso.

1.8. CONCLUSIONES PARCIALES

Con la realización de este capítulo se fundamentaron los conceptos relacionados con la investigación que ayudaron a una mejor comprensión de la situación planteada por lo que se arriba a las siguientes conclusiones:

- Se realizó un estudio de las diferentes herramientas que tratan los algoritmos de predicción con salidas compuestas llegando a la conclusión de que la mejor es Mulan.
- Se definió que la metodología seleccionada ágil XP es adecuada para el desarrollo del software, ya que se adapta a las necesidades del mismo por las características del sistema a implementar, así como el lenguaje de programación Java y el entorno de desarrollo integrado *NetBeans*.

CAPÍTULO 2. Exploración y planificación

2.1. Introducción

En este capítulo se hace referencia a las fases de Exploración y Planificación de la metodología XP detallando cada uno de los artefactos que surgen como resultado de estas dos fases. Se describe la propuesta solución de la herramienta a desarrollar y sus características fundamentales en diferentes fases, dejando esencialmente plasmados en este los requisitos funcionales, los requerimientos de usuario, el diseño de flujo de un experimento, el modelo de dominio y las historias de usuarios necesarias para una mejor comprensión de la solución a implementar. En este capítulo también se detallan las iteraciones llevadas a cabo durante la etapa de construcción del sistema, exponiéndose cada una de las tareas designadas por Historia de Usuarios (HU).

2.2. Propuesta de solución

Para dar solución al objetivo general planteado se propone, desarrollar una herramienta para el diseño de experimentos en Mulan a través de una interfaz gráfica que facilitará el trabajo de los analistas de datos al realizar un experimento. Este experimento toma como punto de partida la selección de una o varias bases de datos (*DataSets*) las cuales tienen que ser cargadas en formato “arff”. Para lograr generalización entre los resultados el sistema permite seleccionar el tipo de evaluación (*cross-validation o Test Split*), los algoritmos de predicción (ST, MTSC, ERC, ERCC, MORF, MTS) que el analista decida y el tipo de base según la preferencia del usuario, además de especificar la cantidad de salidas (t*arget*) según la base de datos seleccionada. Una vez seleccionados los datos se añaden a la interfaz y luego verificas si están todos los datos y son correctos, permitiendo también eliminarlos si el usuario no desea incluirlos en la experimentación. Luego de ejecutadas estas opciones que brinda el sistema se muestran los resultados en pantalla y posteriormente se guardan. A continuación una ilustración de la propuesta solución del sistema.

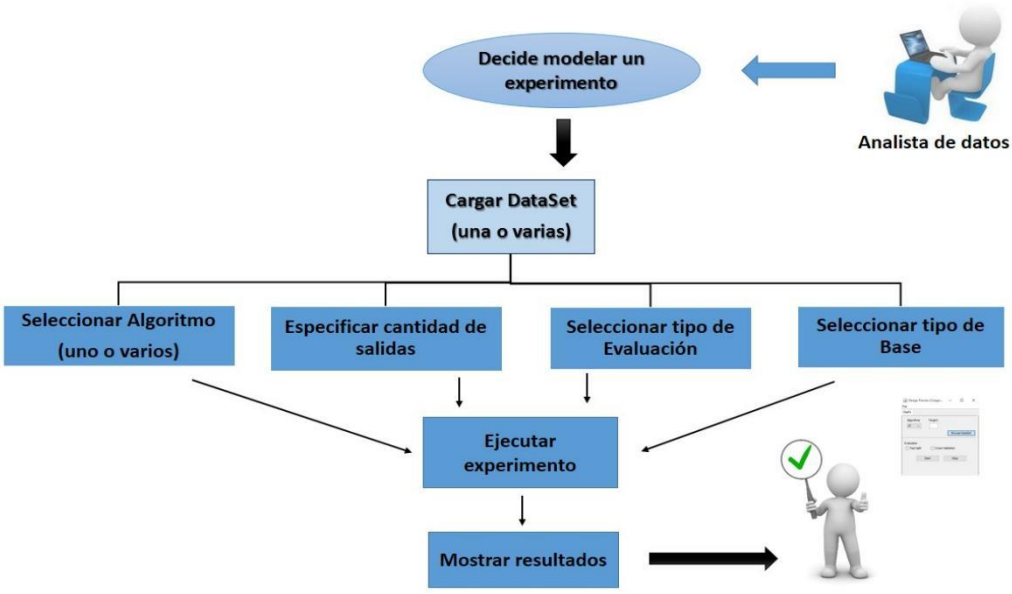


Ilustración 2. Propuesta del sistema.

2.3. Modelo de dominio

El modelo del dominio se realizó durante la fase de exploración y planificación de la aplicación, aunque este artefacto no esté entre los propuestos por la metodología de desarrollo utilizada, se propone su realización para una mejor comprensión del dominio del sistema por parte del cliente y del equipo de desarrollo.

2.3.1. Conceptos fundamentales del dominio

Analista de datos: Persona con conocimientos previos que por medio de una computadora puede acceder al sistema con las bases de datos requeridas para realizar un experimento y obtener los resultados del mismo.

Base datos: Es un archivo “arff” que cargará el sistema y permitirá al analista de datos seleccionar todos los parámetros necesarios para ejecutar el experimento.

Validación cruzada: Divide los datos entrados por el analista en datos de entrenamiento y datos de prueba.

Datos de entrenamiento y prueba: Se aprende un modelo matemático a partir de un conjunto de datos representativos de un problema real.

Entrenamiento del clasificador: Esta colección de datos puede presentar en algunos de los casos del conjunto de datos valores anómalos, valores ausentes (Valores que no han sido medidos para una variable) o variables en diferentes escalas de medición lo cual es muy común en estos problemas.

Medida del error: Métrica de evaluación utilizada para la ejecución de los experimentos.

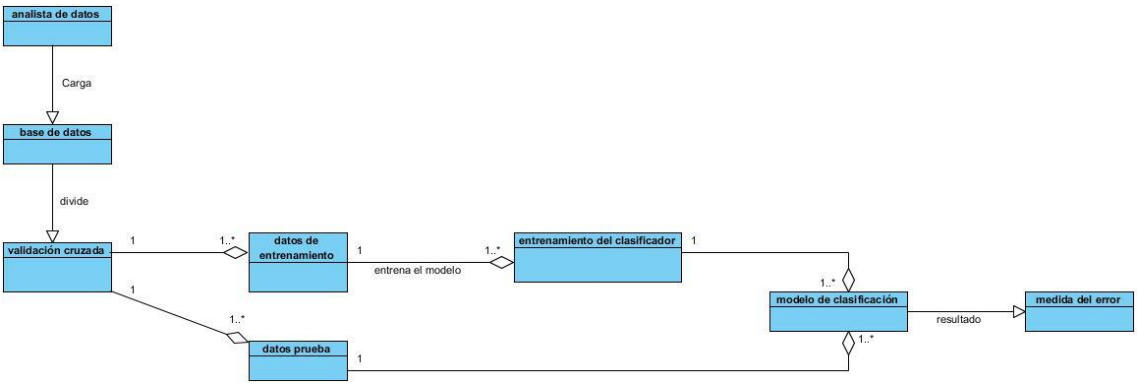


Ilustración 3. Diagrama del Modelo de Dominio.

Como se evidencia en el Diagrama de Modelo de Dominio, un analista de datos tiene las bases de datos en formato “arff” para cargarlas en el sistema. Una vez seleccionada la base de datos el investigador divide los datos en datos de entrenamiento y prueba teniendo muestras significativas en ambos conjuntos de datos relacionados con el problema. Estos son entrenados por el modelo matemático a partir de un conjunto de datos representativos de un problema real. Luego se mostrarán los resultados del experimento realizado de forma detallada. El analista podrá acceder a los resultados obtenidos a través de la medida del error.

2.4. Funcionalidades del sistema

El levantamiento de requerimientos es la primera fase técnica del proceso de ingeniería de software. La tarea del análisis de los requerimientos es un proceso de descubrimiento, refinamiento, modelado y especificación que permite definir las funcionalidades y el rendimiento del software, e indica la interfaz y las restricciones que debe tener el software (Beck, Kent y Wesley, Addison, 1999). La importancia de esta fase radica en la especificidad con que son detallados los requerimientos que permitirán el diseño y posteriormente la implementación del sistema. Existen dos clasificaciones de requerimientos (Funcionales y No funcionales), a través de los cuales se define el alcance del sistema en cuanto a las acciones que debe realizar.

A partir de las funcionalidades requeridas se procede a identificar las HU necesarias para el funcionamiento de la aplicación. Las funcionalidades definidas por el cliente son:

- Leer uno o varios ficheros de datos en formato “arff”.
- Dividir los datos en entrenamiento y prueba para ejecutar el experimento.
- Mostrar en pantalla los elementos que componen el resultado final del experimento.

- Escribir los resultados en forma tabular en ficheros de texto y haciendo uso del formato “csv”[2].

Por otra parte, el sistema debe tener en cuenta los requerimientos no funcionales, que son propiedades o cualidades que el equipo de desarrollo no debe obviar, puesto que estos son los encargados de proporcionar la terminación del producto. Estas propiedades se muestran como las características que hacen al producto atractivo, usable, rápido y confiable.

Hardware:

Para un buen funcionamiento del sistema, se hace necesario contar con un dispositivo con las siguientes características:

- Memoria RAM de 512 MB o superior.
- Microprocesador Dual Core a 2.0 GHz o superior.
- El dispositivo deberá tener mínimo 40 MB de almacenamiento disponible.

Software:

- El dispositivo debe tener instalada la máquina virtual de Java.

Interfaz:

- El sistema se mostrará de forma vertical en todos los dispositivos que se instale.
- Los colores que predominan son (negro, gris y blanco).

Disponibilidad:

- La aplicación debe estar disponible en todo momento que el usuario necesite acceder y manejar la información contenida en la misma.

Usabilidad:

- Para la navegación en la aplicación, se requiere de usuarios que sean capaces de procesar grandes volúmenes de datos en ese dominio específico asistido por una herramienta, lo que posibilitará al usuario sin experiencia una rápida adaptación para operar con el sistema.
- La aplicación deberá presentar una interfaz que permita la fácil interpretación por el usuario, además debe posibilitar que el usuario visualice de manera rápida la acción realizada

  .

[2] *Valores separados por coma (Coma Separated Value).*

2.6. Fase de Exploración

La metodología para desarrollo de software *Extreme Programming*, comienza su ciclo de desarrollo durante la fase de Exploración, que corresponde con la fase donde se identifican las historias de usuarios, que no serán más que los elementos rectores dentro del desarrollo del software. Además en la misma se explora la familiarización del equipo de trabajo con las tecnologías y herramientas que se emplearán a lo largo del desarrollo.

2.6.1. Historias de usuarios

Las Historias de Usuarios (HU) son escritas por los propios clientes, tal y como ven ellos las necesidades del sistema. En estas el cliente plantea a grandes rasgos lo que necesita a través de las HU, lo que facilita a los programadores estimar el tiempo de desarrollo. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. El cliente es el encargado de asignarle una prioridad a cada HU y es el equipo de desarrollo el encargado de asignarle un costo, este se traduce en las semanas que llevará el desarrollo de las mismas. Si las HU según lo planificado demoran en desarrollarse, se sugiere dividirla en HU más pequeñas. También, es importante destacar que las HU nuevas pueden describirse en cualquier momento, con esto se comprueba la flexibilidad de la metodología.

Estas se escriben desde la perspectiva del cliente, aunque los desarrolladores pueden brindar también su ayuda en la identificación de las mismas. Estas son descritas en una tabla dividida por las siguientes secciones.

1. **Número**: número de la historia de usuario incremental en el tiempo.
2. **Nombre de Historia de Usuario**: el nombre de la historia de usuario sería para identificarlas mejor entre los desarrolladores y el cliente.
3. **Modificación de HU número**: si sufrió alguna modificación anterior.
4. **Usuario**: involucrados en el desarrollo de la HU.
5. **Iteración Asignada**: número de la iteración.
6. **Prioridad en negocio**: Alta, Media o Baja.
7. **Riesgo en Desarrollo**: Alta, Media o Baja.
8. **Puntos estimados**: tiempo estimado que se demorará el desarrollo de la HU en días.

9. **Puntos Reales**: tiempo que se demoró en realidad el desarrollo de la HU.
10. **Descripción**: breve descripción de la HU.
11. **Observaciones**: señalamiento o advertencia del módulo.
12. **Prototipo de interfaz**: Prototipo de interfaz.

La prioridad en el negocio es otro rol fundamental en las historias de usuarios, estas pueden ser clasificadas en (Alta, Media, Baja). Alta cuando cumple con las HU que resultan funcionalidades fundamentales en el desarrollo del sistema, a las que el cliente define como principales para el control integral del sistema. Media cuando son las funcionalidades a tener en cuenta por el cliente, sin que estas tengan una afectación sobre el sistema que se esté desarrollando. Baja cuando se le otorga a las HU que constituyen funcionalidades que sirven de ayuda al control de elementos asociados al equipo de desarrollo.

El riesgo en su desarrollo también es importante este se clasifica en (Alta, Media, Baja). Alta cuando en la implementación de las HU se considera la posible existencia de errores que lleven a la inoperatividad del código. Media cuando pueden aparecer errores en la implementación de la HU que puedan retrasar la entrega de la versión. Baja cuando pueden aparecer errores que serán tratados con relativa facilidad sin que traigan perjuicios para el desarrollo del proyecto.

Para el desarrollo de la herramienta se definieron las siguientes HU:

1. Cargar *Dataset* (una o varias).
2. Especificar la cantidad de salidas(targets)
3. Seleccionar tipo de Evaluación (validación cruzada, *test split*).
4. Seleccionar algoritmo (ST, MTS, MTSC, ERC, ERCC, MORF).
5. Seleccionar tipo de base (*Zeror, Reptree-bag*).
6. Añadir datos.
7. Eliminar datos.
8. Ejecutar experimento.
9. Mostrar los resultados.
10. Exportar resultados.

En el **Anexo 2: Historias de Usuario**, pueden encontrarse las HU restantes implementadas en el desarrollo de la aplicación.

<table>
<tr><th colspan="3">Historia de Usuario</th></tr>
<tr><td colspan="2">Número: HU1</td><td>Usuario: Cliente</td></tr>
<tr><td colspan="3">Nombre de historia: Cargar DataSet.</td></tr>
<tr><td colspan="2">Prioridad en negocio: Alta

(Alta/ Media/ Baja)</td><td>Riesgo en desarrollo: Alto</td></tr>
<tr><td>Puntos estimados: 5/5</td><td>Puntos reales: 5</td><td>Iteración asignada: 1</td></tr>
<tr><td colspan="3">Programador responsable:

Lisandra Hernández Colas y Andy Reinoso Brito.</td></tr>
<tr><td colspan="3">Descripción:

Permite cargar un archivo con extensión “arff”, de una dirección especificada que contenga una base de datos.</td></tr>
<tr><td colspan="3">Observaciones: Las bases de datos tienen que estar en formato “arff”.</td></tr>
<tr><td colspan="3">Interfaz:

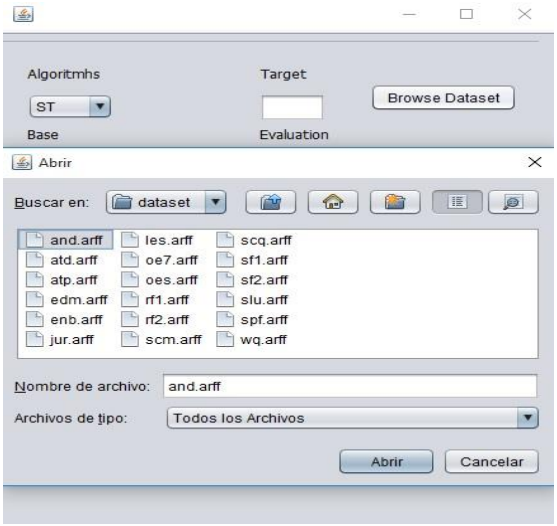
 
</td></tr>
</table>

Tabla 2. HU1.Cargar DataSet.

Historia de Usuario		
Número: HU8	**Usuario:** Cliente	
Nombre de historia: Ejecutar experimento.		
Prioridad en negocio: Alta (Alta/ Media/ Baja)	**Riesgo en desarrollo:** Alto	
Puntos estimados: 5/5	**Puntos reales:** 5	**Iteración asignada:** 3
Programador responsable: Lisandra Hernández Colás y Andy Reinoso Brito.		
Descripción: Permite ejecutar las funcionalidades de las historias de usuarios anteriores sobre las bases de datos seleccionadas.		
Observaciones: Deben haberse añadido correctamente los parámetros necesarios para la ejecución del experimento, o sea las bases de datos, la cantidad de salidas, el tipo de evaluación, el tipo de base y el algoritmo deseado.		
Interfaz: 		

Tabla 3. HU8.Ejecutar experimento.

2.7. Fase de Planificación

En la fase de la planificación de la metodología XP, el cliente prioriza cada una de las HU y los programadores realizan una estimación del esfuerzo por cada una de ellas.

2.7.1. Estimación del esfuerzo por HU

La siguiente tabla contiene la estimación del esfuerzo por cada HU según el orden en que serán realizadas. En esta estimación se incluye todo el esfuerzo asociado a la implementación de cada HU y se expresa utilizando como medida los puntos de estimación (punto máximo). Cada punto se considera una semana ideal de trabajo (5 días), donde se trabaja el tiempo que ha sido planificado sin interrupciones. En la tabla se muestran los puntos estimados por cada HU, donde al final se concluyó que las mismas se realizarán en un período máximo de 15 semanas de trabajo.

No.	Historias de usuarios	Puntos estimados
1	Cargar *DataSet*	2/5
2	Especificar cantidad de salidas	1/5
3	Seleccionar tipo de evaluación	1/5
4	Seleccionar algoritmo	1/5
5	Seleccionar tipo de base	1/5
6	Añadir datos	1/5
7	Eliminar datos	1/5
8	Ejecutar experimento	3/5
9	Mostrar resultados	2/5
10	Exportar resultados	2/5

Tabla 4. Estimación del esfuerzo por HU.

2.7.2. Plan de iteraciones

Luego de identificar las HU y la estimación del esfuerzo por cada una de ellas, se procede a realizar el plan de iteraciones en el cual estarán comprendidas las HU según el orden en que sean realizadas por cada iteración, además del total de semanas que durará cada una de estas. Teniendo en cuenta la prioridad y el riesgo de cada una de las HU, así como otros factores que influyen en la implementación del sistema, se decidió dividir el proyecto en tres iteraciones, detalladas a continuación:

Iteración 1: En esta iteración se implementarán las 5 primeras HU, correspondientes a las funcionalidades de cargar una o varias bases de datos, seleccionar la cantidad de salidas deseadas, seleccionar el tipo de Evaluación, seleccionar los algoritmos integrados en la herramienta para su ejecución y seleccionar tipo de base.

Iteración 2: En esta iteración se llevará a cabo el desarrollo de las HU número 6, 7 y 8 correspondientes a las funcionalidades de añadir datos, eliminar datos y ejecutar experimento.

Iteración 3: En esta iteración se implementan las HU número 9 y 10 correspondiente a la funcionalidad de complementar el resultado de lo implementado en las iteraciones anteriores en una interfaz que muestre los resultados.

La siguiente tabla se encarga de reflejar el orden de las HU por cada iteración, así como la duración total en semanas de las mismas.

Iteración	Orden de las HU por iteraciones	Duración total /semanas
1	1. Cargar *DataSet* 2. Especificar cantidad de salidas 3. Seleccionar tipo de Evaluación 4. Seleccionar algoritmo 5. Seleccionar tipo de base	6
2	6. Añadir datos 7. Eliminar datos 8. Ejecutar experimento	5

3	9. Mostrar resultados 10. Exportar resultados	4
Total	**10**	**15 semanas**

Tabla 5. Tiempo estimado por HU.

2.7.3. Plan de entregas

El plan de entregas define la fecha fin de cada iteración, este plan se obtiene de cada una de las reuniones realizadas entre el equipo de desarrollo y el cliente.

A continuación se muestra una tabla con las fechas aproximadas de entregas por cada iteración.

Herramienta	**Fin de 1^{era} iteración**	**Fin de 2da iteración**	**Fin de 3^{era} iteración**
Interfaz gráfica para el diseño de experimentos en MULAN	23/febrero/2017	31/marzo/2017	21/abril/2017
	Herramienta v0.1	Herramienta v0.2	Herramienta v1.0

Tabla 6. Plan de entregas.

2.7. CONCLUSIONES PARCIALES

Con el desarrollo de este capítulo se describieron las fases de exploración y planificación propuestas por la metodología XP y se arriba a las siguientes conclusiones:

- Se identificaron las características que el sistema debe cumplir para su correcto funcionamiento representado en la HU.
- Se planificaron las diferentes iteraciones que permitieron organizar el desarrollo de la interfaz gráfica.
- Se elaboró el plan de entregas donde se estima que el sistema estará listo en 15 semanas.

CAPÍTULO 3. Implementación y Resultados

3.1. Introducción

En el proceso de desarrollo del software la etapa implementación en la metodología XP es un proceso que se realiza de forma iterativa, obteniendo como resultado de cada una de estas un producto funcional que debe ser sometido a pruebas y mostrado al cliente para permitir una retroalimentación por parte de los desarrolladores. En este capítulo se muestra como se llevó a cabo la implementación del sistema, definiéndose diferentes tareas por iteraciones. Se evalúa la calidad de la solución a través de las pruebas de software realizadas, así como los resultados obtenidos.

3.2. Arquitectura del sistema

La arquitectura de un software es el diseño de más alto nivel de la estructura de un sistema. Toda arquitectura de software debe definir diversos aspectos del software, y generalmente, cada uno de estos se describe de una manera más comprensible si se utilizan distintos modelos o vistas (Roger S. Pressman, 2007).

Para el desarrollo del sistema se empleará la arquitectura N Capas, la cual consiste en dividir el sistema en varias capas lográndose de esta forma reducir el grado de complejidad. La aplicación contienen códigos para la presentación, procesamiento y almacenamiento de datos, la diferencia se encuentra en la organización del mismo. Ofrece ventajas de tipo organizativo debido a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La utilización de esta arquitectura permite desarrollar capas en paralelo, además posibilita un mantenimiento más sencillo ya que se puede modificar un componente de ser necesario en lugar de toda la aplicación y permite agregarle nuevos módulos aumentado su flexibilidad. El sistema se divide en las siguientes capas lógicas:

Interfaz: Representa la información proporcionada por el acceso a datos en un formato que permite la interacción entre el usuario y el sistema, se puede evidenciar en las interfaces de usuario.

Lógica del negocio: Procesa las solicitudes realizadas por el usuario y se envían las respuestas produciendo generalmente cambios en el sistema. Esta capa sirve como intermediaria entre la interfaz y el acceso a datos.

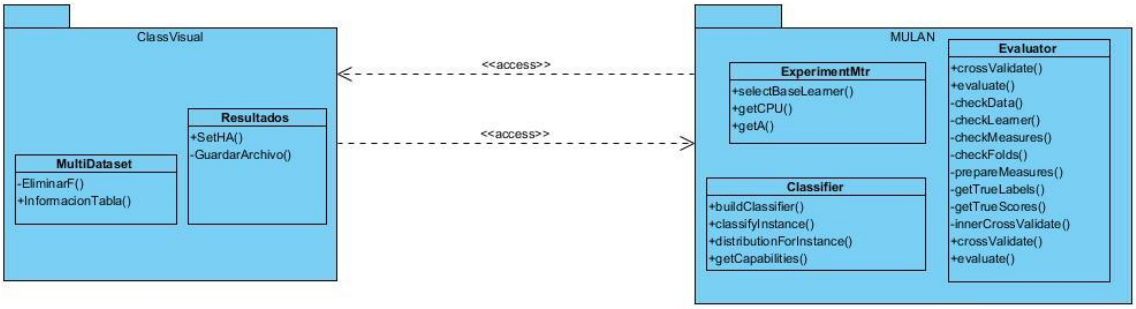


Ilustración 4. Interacción entre las capas del sistema (Vista lógica del sistema).

3.3. Patrones de diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Un patrón de diseño resulta ser una solución a un problema de diseño (E. Gamma, R. Helm, R. Johnson, J. Vlissides, & ., n.d.).

Después de haber realizado un análisis sobre los criterios de expertos en el tema, se puede resumir, que los patrones de diseño se logran combinar en componentes que resuelven grandes problemas. Así como adaptar a problemas de menor complejidad siempre y cuando se haga un uso correcto de los mismos. Además cabe destacar que existen varios grupos de patrones de diseño, pero la solución se basa en los más usados en función del objetivo que se ha planteado en este trabajo.

3.3.1. Patrones GRASP

Los patrones de software para la asignación de responsabilidades (GRASP), describen los principios esenciales de la asignación de responsabilidades a objetos, por tanto fueron considerados en la confección de las clases que componen el modelo de diseño (Craig Larman, 2010).

Patrón experto: Define el principio fundamental en virtud del cual se asignan las responsabilidades en un diseño orientado a objetos. Plantea que la responsabilidad debe recaer en la clase que cuenta con la información necesaria para cumplir con ella. De forma tal que el sistema sea más fácil de entender, mantener y ampliar, permitiendo reutilizar los componentes creados en futuras aplicaciones.

Ejemplo: La clase *MultiplesDataset*, cuenta con la información necesaria para cumplir con cada una de las responsabilidades que le corresponden. Ver Anexo 3.

Patrón creador: Define quien debe ser el responsable de crear una nueva instancia de una clase. Plantea que debe recaer en la clase que agrega, contiene, registra, utiliza o tiene los datos de inicialización de la nueva instancia. La prioridad en caso de que exista más de una posibilidad es en el orden enunciado. El propósito fundamental de este patrón es asignar responsabilidades relacionadas con la creación de objetos producidos en cualquier evento. Facilita un Bajo Acoplamiento, supone menor dependencia respecto al mantenimiento y mejores oportunidades de reutilización.

Ejemplo: La clase *MultiplesDataset* es la responsable de crear instancias de la clase Mulan para la visualización por el usuario. Ver Anexo 4.

Patrón Bajo Acoplamiento: Define como fomentar una menor dependencia de una mayor reutilización. Plantea mantener la fuerza de las conexiones entre clases lo más bajo posible. Este parámetro es conocido como acoplamiento. No puede verse de forma aislada como Experto o Alta Cohesión, el acoplamiento tal vez no sea muy importante, sino se busca la reutilización.

Ejemplo: La clase *ExperimentMTR* se le asigna la tarea de cargar *dataset* pasándole como parámetros el nombre y la dirección del archivo obtenido de la clase *MultiplesDataset.* Ver Anexo 5.

Patrón Alta Cohesión: Define que las responsabilidades asignadas a una misma clase deben guardar relación y estar enfocadas al mismo objetivo. Una Alta Cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas donde todos sus elementos trabajan juntos para proporcionar algún comportamiento bien delimitado.

Ejemplo: En la clase *MultiplesDataset* en los métodos *InformaciónTabla* y *EliminarF* todos los datos trabajan según la responsabilidad asignada, estos trabajan juntos para proporcionar un comportamiento bien delimitado. Ver Anexo 6.

3.4. Tarjetas CRC (Clase - Responsabilidad – Colaborador)

El diseño de aplicaciones, en la metodología XP no requiere la representación del sistema mediante diagramas de clases utilizando notación UML[3], en su lugar se usan otras técnicas como las tarjetas CRC.

[3] *Lenguaje Unificado de Modelado, es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad para visualizar, especificar, construir y documentar un sistema.*

Estas determinan responsabilidades y colaboraciones de las clases. El desarrollo de cualquier proyecto requiere de un buen diseño de sus clases para de esta forma realizarlo con la mejor calidad posible y así el cliente quede satisfecho. En la metodología XP el diseño de las clases se realiza a través de las tarjetas CRC, para de esta forma ayudar al refinamiento de las clases. De forma organizada cada tarjeta representa una clase, donde se describen las responsabilidades que tiene y las clases colaboradoras que se relacionan con la misma. Las tarjetas CRC también ayudan a diseñar el sistema en conjunto entre todo el equipo de desarrollo, aunque su principal objetivo es propiciar el enfoque orientado a objetos y reducir el modo de pensar procedimental.

Estas se encuentran diseñadas en cuatro secciones: nombre de la clase, descripción, responsabilidades y colaboradores. Una clase describe un objeto o evento del sistema, mediante sus atributos y métodos. Las responsabilidades de estas se describen por las tareas que realiza o por los métodos y los colaboradores son las demás clases con las que interactúa para cumplir con sus responsabilidades.

A continuación se describen algunas de las tarjetas CRC diseñadas para la implementación del sistema:

3.4.1. Tarjeta CRC # 1 Entrenamiento

Nombre de la clase: *ExperimentMTR*	
Descripción: Permite procesar y calcular los datos.	
Responsabilidad	**Colaboradores**
Permite entrenar el modelo matemático.	*Multiples DataSet* *Classifier* *MultiLabelLearnerBase* *MultiLabelIntances* *Evaluator* *Resultados*

Tabla 7. Tarjeta CRC # 1 Entrenamiento.

3.1.2. Tarjeta CRC #2 Múltiples Bases de datos

Nombre de la clase: *MultiplesDataSets*	
Descripción: Permite cargar todos los datos para la ejecución del experimento.	
Responsabilidad	**Colaboradores**
Permite entrenar el modelo	*ExperimetMTR*

Tabla 8. Tarjeta CRC # 3 Múltiples bases de datos.

3.1.2. Tarjeta CRC # 3 Resultado

Nombre de la clase: Resultados	
Descripción: Permite mostrar el resultado final del experimento.	
Responsabilidad	**Colaboradores**
Mostrar resultado final del experimento	*Experiment MTR*

Tabla 9. Tarjeta CRC # 4 Resultados.

Para un mejor entendimiento de cómo están relacionadas estas clases y las funcionalidades que realizan cada una de ellas, se elaboró un diagrama de clases. Ver Anexo 7.

3.5. Tareas de la ingeniería

En cada una de las iteraciones se procede a la implementación de las historias de usuarios seleccionadas. Al inicio de las mismas se lleva a cabo una revisión del plan de iteraciones y se modifica si es necesario. Se descomponen las Historias de Usuarios en tareas de desarrollo, asignando posteriormente cada una de estas a un equipo (o una persona) responsable de su implementación. Estas tareas son para el uso de los programadores, pueden escribirse utilizando un lenguaje técnico y no necesariamente deben ser entendibles para el cliente.

Las Tareas de Ingeniería (TI), se encuentran asociadas fuertemente a las funcionalidades y características que se deben cumplir. Cada tarea pertenece a una HU en específico. Se consideran como la entrada de trabajo para el equipo de programadores. Es una ficha que contiene el número identificador de la tarea, el identificador de la HU con la que está relacionada, el nombre de la tarea, la fecha de inicio, la fecha de fin, el programador responsable y la descripción. También se realizan para llevar a cabo la correcta implementación de las HU descritas por el cliente, que se diseñan en cada una de las iteraciones.

De acuerdo a la planificación realizada, se llevaron a cabo tres iteraciones para el desarrollo del sistema. Cada una de las Historias de Usuario comprendidas en las 3 iteraciones fueron desglosadas en tareas desarrolladas por los programadores.

Historias de Usuarios	Tareas
Cargar *DataSet*	1. Implementar la funcionalidad Cargar *DataSet.* 2. Creación de la interfaz para cargar las bases de datos seleccionadas.

Especificar cantidad de salidas	1. Implementar la funcionalidad especificar cantidad de salidas. 2. Creación de la interfaz para especificar la cantidad de salidas deseadas.
Seleccionar tipo de evaluación	1. Implementar la funcionalidad seleccionar tipo de Evaluación. 2. Creación de la interfaz para seleccionar el tipo de Evaluación.
Seleccionar algoritmo	1. Implementar la funcionalidad seleccionar algoritmo. 2. Creación de la interfaz para seleccionar el algoritmo deseado.
Seleccionar tipo de base	1. Implementar la funcionalidad seleccionar tipo de base. 2. Creación de la interfaz para seleccionar tipo de base.
Añadir datos	1. Implementar la funcionalidad añadir datos. 2. Creación de la interfaz para datos.
Eliminar datos	1. Implementar la funcionalidad eliminar datos. 2. Creación de la interfaz para añadir eliminar datos.
Ejecutar experimento	1. Implementar la funcionalidad ejecutar experimento. 2. Creación de la interfaz para ejecutar el experimento.
Mostrar los resultados	1. Implementar la funcionalidad mostrar los resultados. 2. Creación de la interfaz para mostrar los resultados.
Exportar resultados	1. Implementar la funcionalidad mostrar resultados en una ventana de la herramienta.

	2. Creación de la interfaz para mostrar resultados.

Tabla 10. Asignación de tareas de la ingeniería por HU.

A continuación en los epígrafes siguientes se detallan cada una de las HU que comprenden el desarrollo del sistema, durante las iteraciones previstas. Para ello se hará énfasis en cada una de las tareas comprendidas por HU.

3.5.1. Iteración 1

Durante la primera iteración se abordaron las HU con una prioridad alta donde se seleccionaron las funcionalidades fundamentales y necesarias que permitieron al usuario seleccionar los elementos imprescindibles para comenzar el flujo del experimento y obtener de esa forma una retroalimentación rápida y amplia. En la Tabla 11 se describen las HU correspondientes a esta iteración, el tiempo estimado y el tiempo real en días utilizado para la realización de las tareas de ingeniería. En el **Anexo 8: Tareas de la ingeniería. Iteración 1**, pueden encontrarse las TI restantes implementadas en el desarrollo de la aplicación.

Historia de Usuario	Estimación	Real en días
Cargar *DataSet*	10	10
Especificar cantidad de salidas	5	5
Seleccionar tipo de Evaluación	5	5
Seleccionar algoritmo	5	5
Seleccionar tipo de base	5	5
Totales:	**30**	**30**

Tabla 11. HU implementadas en la Iteración 1.

3.5.1.1. Tareas de la Historia de Usuario Cargar DataSet

Tarea de Ingeniería	
Número de Tarea: 1	**Historia de usuario:** Cargar *DataSet*
Nombre de Tarea: Implementar la funcionalidad Cargar *DataSet.*	
Tipo de Tarea: Desarrollo	**Puntos estimados:** 5
Fecha Inicio: 16/01/2017	**Fecha Fin:** 20/01/2017
Programador responsable: Andy Reinoso Brito	

Descripción: Se implementarán las funcionalidades que permitan al usuario cargar archivos de texto en formato “arff”.

Tabla 12. TI # 1 Implementar la funcionalidad Cargar DataSet.

Tarea de Ingeniería	
Número de Tarea: 2	**Historia de usuario:** Cargar *DataSet*
Nombre de Tarea: Creación de la interfaz para cargar las bases de datos seleccionadas.	
Tipo de Tarea: Desarrollo	**Puntos estimados:** 0.8
Fecha Inicio: 23/01/2017	**Fecha Fin:** 27/01/2017
Programador responsable: Andy Reinoso Brito	
Descripción: Se desarrollarán un grupo de interfaces para cargar archivos de textos en formato “arff”. Además se conectarán las interfaces con las funciones ya implementadas para estos fines.	

Tabla 13. TI # 2 Creación de la interfaz para cargar las bases de datos seleccionadas.

3.5.2. Iteración 2

Durante esta iteración se abordaron las HU con una prioridad alta y media. Al culminar se cuenta con el sistema casi listo para su puesta en funcionamiento con la mayoría de sus funcionalidades más críticas ya implementadas. A continuación se describe en la Tabla 22 las HU implementadas con la estimación y el tiempo real en días de cada una. En el **Anexo 9: Tareas de la ingeniería. Iteración 2**, pueden encontrarse las TI restantes implementadas en el desarrollo de la aplicación.

Historia de Usuario	Estimación	Real en días
Añadir datos	5	5
Eliminar datos	5	5
Ejecutar experimento	15	15
Totales:	**25**	**25**

Tabla 14. HU implementadas en la Iteración 2.

3.5.2.1. Tarea de la Historia de Usuario Ejecutar experimento

Tarea de Ingeniería	
Número de Tarea: 1	**Historia de usuario:** Ejecutar experimento
Nombre de Tarea: Implementar la funcionalidad ejecutar experimento.	
Tipo de Tarea: Desarrollo	**Puntos estimados:** 3
Fecha Inicio: 13/03/2017	**Fecha Fin:** 22/03/2017
Programador responsable: Andy Reinoso Brito	

Descripción: Se implementarán las funcionalidades que le permitan al usuario hacer una llamada a la clase *ExperimentMTR* y realizar la ejecución del experimento.

Tabla 15. TI # 1 Implementar la funcionalidad Ejecutar experimento.

Tarea de Ingeniería	
Número de Tarea: 2	**Historia de usuario:** Ejecutar experimento
Nombre de Tarea: Creación de la interfaz para ejecutar experimento.	
Tipo de Tarea: Desarrollo	**Puntos estimados:** 3
Fecha Inicio: 23/03/2017	**Fecha Fin:** 31/03/2017
Programador responsable: Lisandra Hernández Colás	
Descripción: Se creará la interfaz para manejar la aplicación de la tarea ejecutar experimento.	

Tabla 16. TI # 2 Creación de la interfaz para ejecutar algoritmo seleccionado.

3.5.3. Iteración 3

En esta iteración será implementada la última HU cuyo resultado final será el sistema listo para su funcionamiento. A continuación se describe en la Tabla 32 la estimación y el tiempo real en días de esta. En el **Anexo 10: Tareas de la ingeniería. Iteración 3**, pueden encontrarse las TI restantes implementadas en el desarrollo de la aplicación.

Historia de Usuario	Estimación	Real
Mostrar resultados	10	10
Exportar resultados	5	5
Totales:	**15**	**15**

Tabla 17. HU implementada en la Iteración 3.

3.5.3.1. Tarea de la Historia de Usuario Mostrar resultados

Tarea de Ingeniería	
Número de Tarea: 1	**Historia de usuario:** Mostrar resultados
Nombre de Tarea: Implementar la funcionalidad mostrar resultados.	
Tipo de Tarea: Desarrollo	**Puntos estimados:** 1.3
Fecha Inicio: 03/04/2017	**Fecha Fin:** 07/04/2017
Programador responsable: Andy Reinoso Brito	
Descripción: Se implementarán las funcionalidades que permitan al usuario mostrar los resultados luego de la ejecución del experimento.	

Tabla 18. TI # 1 Implementar la funcionalidad Mostrar resultados.

Tarea de Ingeniería	
Número de Tarea: 2	**Historia de usuario:** Mostrar resultados
Nombre de Tarea: Creación de la interfaz para mostrar resultados.	
Tipo de Tarea: Desarrollo	**Puntos estimados:** 1.8
Fecha Inicio: 10/04/2017	**Fecha Fin:** 14/04/2017
Programador responsable: Lisandra Hernández Colás	
Descripción: Se creará la interfaz para manejar la aplicación de la tarea mostrar resultados.	

Tabla 19. TI # 2 Creación de la interfaz para mostrar resultados.

3.6. Pruebas al sistema

Las pruebas permiten determinar el estado de la calidad del producto de software. La metodología XP propone la realización de dos tipos de pruebas al sistema, que son las llamadas pruebas unitarias y pruebas de aceptación. El desarrollo de software mediante esta metodología se encuentra compuesto por una serie de iteraciones cortas, cada iteración concluye ejecutando un conjunto de pruebas de aceptación, que permitan al cliente comprobar si está satisfecho con el resultado. En otras palabras, las historias de usuario detectadas tendrán asociadas tantas pruebas de aceptación como sean convenientes para verificar el grado de cumplimiento de cada una. Las pruebas unitarias son definidas antes de realizarse la implementación del código.

3.6.1. Pruebas de Aceptación

Las pruebas de Aceptación son generadas a partir de las historias de usuario elegidas para cada iteración donde el cliente verifica que lo que se está probando funcione correctamente. Las mismas son pruebas de caja negra que se crean a partir de las historias de usuario.

Cuando se pasa la prueba de aceptación se considera la historia de usuario finalizada. El objetivo final de estas es garantizar que los requerimientos han sido cumplidos y que el sistema es aceptable.

A continuación se detallan las distintas pruebas de Aceptación empleadas para el correcto funcionamiento de las funcionalidades previstas en cada una de las Historias de Usuario. En el **Anexo 11: Pruebas de Aceptación**, pueden encontrarse las TI restantes realizadas en el desarrollo de la aplicación.

3.6.1. Prueba de Aceptación de la Historia de Usuario Cargar *DataSet*

Prueba de Aceptación	
Código: HU1_P1	**Historia de Usuario:** Cargar *DataSet*
Nombre: Origen de la base de datos, archivo “arff”.	
Descripción: Se prueba la carga de archivos de extensión “arff”.	
Condiciones de Ejecución: El formato dentro del archivo “arff”, debe ser correcto.	
Entrada/ Pasos de ejecución: Se intenta la carga del archivo para la prueba eligiendo el mismo dentro del directorio origen.	
Resultado Esperado: El archivo “arff” es cargado satisfactoriamente.	
Evaluación de la Prueba: Satisfactoria	

Tabla 20. Prueba de Aceptación # 1 origen de la base de datos.

Algunas de las deficiencias detectadas en las iteraciones fueron las que se describen a continuación:

- La interfaz no cargaba múltiples bases de datos.
- La ejecución del experimento no llegaba hasta el final.
- La interfaz solo permitía seleccionar un tipo de base.
- No se mostraban los resultados finales del experimento.
- Errores ortográficos en varias interfaces.

Las no conformidades detectadas por el cliente fueron solucionadas en las iteraciones de las pruebas. Además se probó la solución propuesta en varias ocasiones, formando el cliente una parte significativa de las pruebas y obteniéndose resultados satisfactorios. Los resultados finales de las iteraciones se resumen en el (gráfico 1) de la presente investigación.

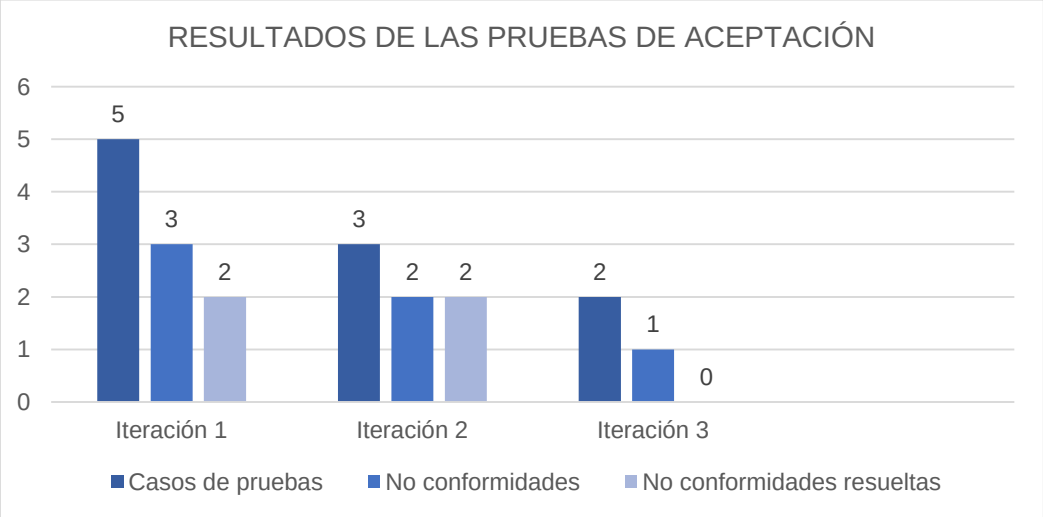


Gráfico 1: Resultados por iteraciones de las pruebas aplicadas.

3.6.2. Pruebas Unitarias

Las pruebas unitarias son enfocadas a los elementos probables más pequeños del software. Aplicable a componentes representados en el modelo de implementación para verificar que los flujos de control y de datos están cubiertos, y que funcionen como se espera. La prueba de unidad siempre está orientada a caja blanca. Todo el código debe tener pruebas unitarias y deben pasarlas antes de ser liberado.

Al culminar la implementación de las funcionalidades propuestas en las iteraciones, las mismas fueron sometidas a un proceso riguroso de pruebas para asegurar que el código propuesto estaba listo para continuar con el desarrollo del sistema, cumpliendo de esta forma con lo que propone la metodología XP, “No pasar a una etapa superior mientras existan errores por corregir” (Ing. José Joskowicz, n.d.). Dichas pruebas arrojaron los siguientes resultados:

- No se cargaban múltiples bases de datos.
- No se agregaban los datos a la interfaz.
- Al seleccionar el tipo de base *Zeror* no se ejecutaba el experimento.
- No se mostraban los datos que el usuario iba a utilizar en la ejecución del experimento.
- No se eliminaban los datos que el usuario no deseaba utilizar.
- No se mostraban los resultados.

Las pruebas unitarias realizadas al código fuente del sistema, permitieron entregar una herramienta gráfica con las características esperadas, ya que el mismo no presenta errores que puedan ocasionar problemas una vez que sea desplegado y utilizado por los usuarios. Se sugiere ver (gráfico 2) el cual muestra los resultados por iteraciones de las pruebas unitarias realizadas.

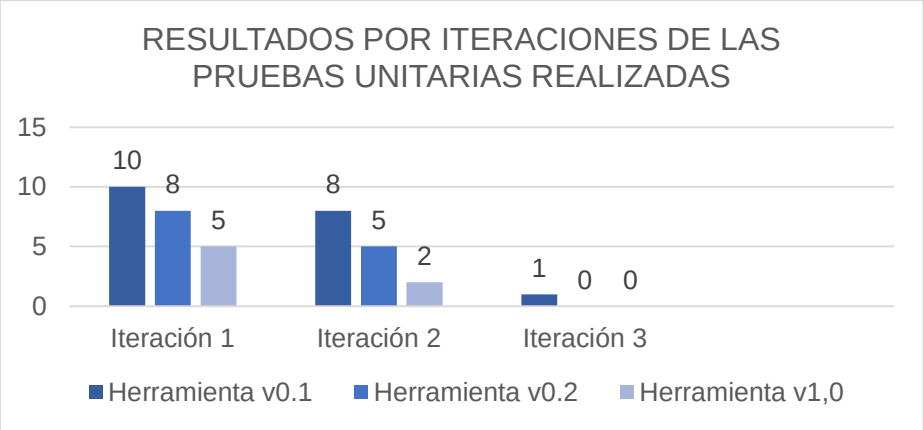


Gráfico 2: Resultados por iteraciones de las pruebas aplicadas.

3.7. Caso de estudio de la Herramienta grafica para el diseño de experimentos en MULAN

En este epígrafe se explica el experimento realizado. Primeramente se hace una descripción de las bases de datos y la métrica de evaluación utilizada para la ejecución de los experimentos. Se muestra un ejemplo de un caso de estudio empleando la interfaz gráfica implementada.

3.7.1. Evaluación

Como medida de evaluación se empleó el promedio del error cuadrático medio (aRRMSE) medido en cada base de datos para las variables de salidas. La misma se determina de la siguiente manera:

$$aRRMSE(h; D_{test}) = \frac{1}{q}\sum_{j=1}^{q}\sqrt{\frac{\sum_{(x,y)\in D_{test}}(h(x_j) - y_j)^2}{\sum_{(x,y)\in D_{test}}(\bar{y} - y_j)^2}}$$

3.7.2. Bases de Datos

La tabla 21 describe algunas de las bases de datos utilizadas en el estudio y sus principales características. La primera columna indica el nombre de las bases de datos, la segunda se refiere al número de observaciones de la base de datos, en esta columna además se indica si los datos están particionados en datos de prueba *test* y datos de entrenamiento *train* o si la base de datos contiene todos los datos. También en el experimento es necesario aplicar una validación cruzada para seleccionar los datos de entrenamiento y los de prueba, se conoce como *K-Fold Cross Validation* (se señala como cv), mientras que el resto de las columnas se refieren a los atributos predictores y las variables de salida respectivamente.

Datos	***Instancias***	***# Atributos***	***# Target***
EDM	154/cv	16	2
ENB	768/cv	8	2
Solar Flare 1	323/cv	10	3
Solar Flare 2	1066/cv	10	3
Atp1d	201/136	411	6
Atp7d	188/108	411	6
OES	343/cv	263	16
OES7	403/cv	298	16

RF1 RF2	4108/5017 4108/5017	64 576	8 8
SCM1 SCM2	8145/1658 7463/1503	280 61	16 16
Water Quality	1060/cv	16	14

Tabla 21. Datos generales de las bases de datos.

EDM *(Electrical Discharge Machining)* es una base de datos que representa un problema de regresión con dos variables de salida y 16 variables de entrada continúas. Cada variable de salida puede tomar 3 valores distintos (-1, 0 ,1). La tarea es acortar el tiempo de mecanizado mediante la reproducción del comportamiento de un operador humano que controla los valores de dos variables.

SF1, SF2 *(Solar Flare)* (Asuncion & Newman, 2007) es una base de datos que contiene 3 variables de salida correspondientes a 3 tipos de llamas solares (común, moderada y severa) que son observadas durante 24 horas. Hay dos versiones de esta base de datos, SF1 contiene los datos del año 1969 y SF2 del año 1978.

ATP1d, ATP7d *(The Airline Ticket Price)* los conjuntos de datos se refiere a la predicción de los precios de los billetes de avión. Las filas son una secuencia de observaciones en tiempo-ordenado lo largo de varios días. Cada muestra en esta base de datos representa un conjunto de observaciones de una fecha de observación y salida par específico. Las variables de entrada para cada muestra son valores que pueden ser útiles para la predicción de los precios de los billetes de avión para una fecha de salida específica. El resultado es un conjunto de características de 411 variables. Para obtener detalles específicos sobre cómo se construyen estos conjuntos de datos por favor consulte (Groves & Gini, 2011). La naturaleza de estos conjuntos de datos es heterogénea con una mezcla de varios tipos de variables que incluyen variables booleanas, los precios, y los recuentos.

OES, OES7 *(Occupational Employment Survey)* son bases de datos obtenidas en los años 1997 (OES97) y 2010(OES10) del *Occupatinal Employement Survey* compilada por *US Bareu of Labor Statistics*. Cada instancia (ciudad) provee un estimado del tiempo completo equivalente a la cantidad de empleados entre un número determinado de empleos para un área metropolitana específica. Existen 334 y 403 ciudades (instancias) en la base de datos de 1997 y mayo de 2010 respectivamente. Las variables de entrada en estas bases de datos son una secuencia aleatoria de un conjunto de empleos (ejemplo: doctor, dentista, mecánico) observados en al menos el 50 % de las ciudades (algunas de las categorías no tiene valor para determinadas ciudades).

Las variables de salida en los dos años son aleatorias, seleccionadas de todo el conjunto de categorías alrededor del 50 % del umbral. Los valores perdidos tanto en las variables de entrada como en las de salida son reemplazados por la media de la muestra.

RF1, RF2 *(The river flow)* los conjuntos de datos de flujo de los ríos se refieren a la predicción de la red fluvial fluye durante 48 horas en el futuro en lugares específicos. El conjunto de datos contiene los datos de las observaciones de flujo por hora de los 8 sitios en la red del río Mississippi en los Estados Unidos y se obtuvieron del Servicio Meteorológico Nacional de Estados Unidos. Cada fila incluye la observación más reciente para cada uno de los 8 sitios, así como observaciones en tiempo lag de 6, 12, 18, 24, 36, 48 y 60 horas en el pasado. En RF1, cada sitio contribuye 8 variables de atributos para facilitar la predicción. Hay un total de 64 variables más 8 variables objetivo. El conjunto de datos RF2 extiende los datos RF1 mediante la adición de información de pronóstico de precipitación para cada uno de los 8 sitios (lluvia esperada reportado como valores discretos: 0,0, 0,01, 0,25, 1,0 pulgadas).

SCM1, SCM2 *(The Supply Chain Management)* los conjuntos de datos de gestión de la cadena de suministro se derivan de la Competencia Agente Operador en la cadena de suministro (SCM TAC) del torneo a partir de 2010. Cada fila corresponde a un día de observación en el torneo (hay 220 días en cada juego y 18 partidos del torneo en un torneo). Las variables de entrada en este ámbito son los valores de un día de torneo específico. Además, se incluyen 4 observaciones temporizadas para cada producto observado y el componente (1, 2,4 y 8 días de retardado) para facilitar cierta anticipación de las tendencias en el futuro. Los conjuntos de datos contienen 16 objetivos, cada objetivo se corresponde con el valor medio del día siguiente (SCM1d) o el valor por 20 días en el futuro (SCM20d). Los días sin valores objetivos se excluyen de los conjuntos de datos (es decir, los días con etiquetas que están más allá del final del juego están excluidos).

WQ *(Water Quality)* es una base de datos que contiene 14 variables de salida que representan las especies de plantas y animales en los ríos de Eslovenia y 16 variables de entrada que se refieren a los parámetros físicos y químicos que miden la calidad del agua.

3.7.3. Resultados del experimento

Para mostrar las características de la interfaz gráfica implementada, se va a realizar un pequeño experimento involucrando dos de las bases de datos que se encuentran disponibles para problemas de predicción con salidas compuestas.

El objetivo del estudio será destacar las facilidades de uso que ofrece el sistema obteniendo mejores resultados en la experimentación.

A continuación se muestra a través de interfaces utilizando dos bases de datos de las descritas en la tabla 45, el desarrollo del experimento en la herramienta implementada.

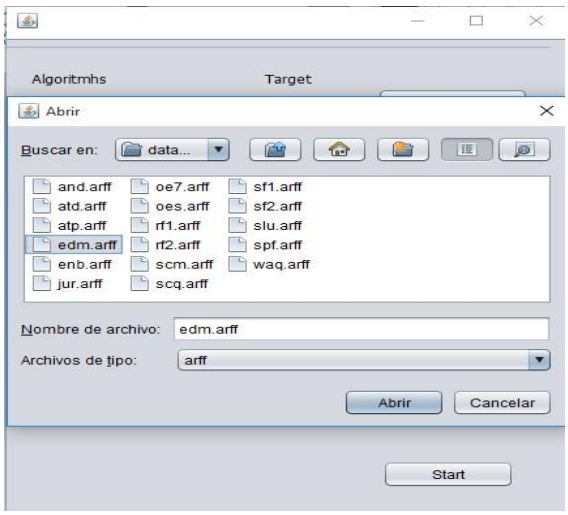


Ilustración 5. Interfaz gráfica para cargar DataSet.

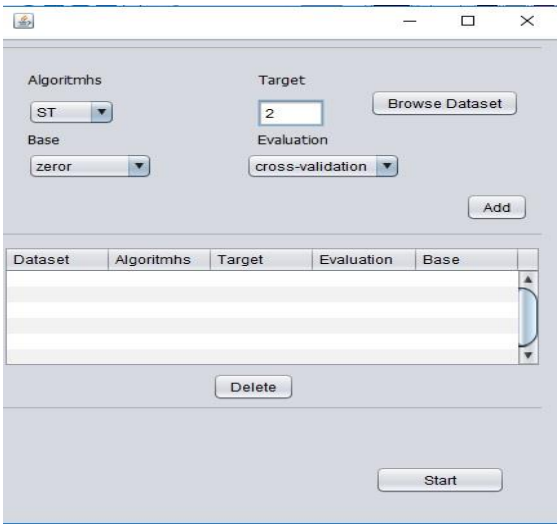


Ilustración 6. Interfaz para especificar la cantidad de salidas del experimento.

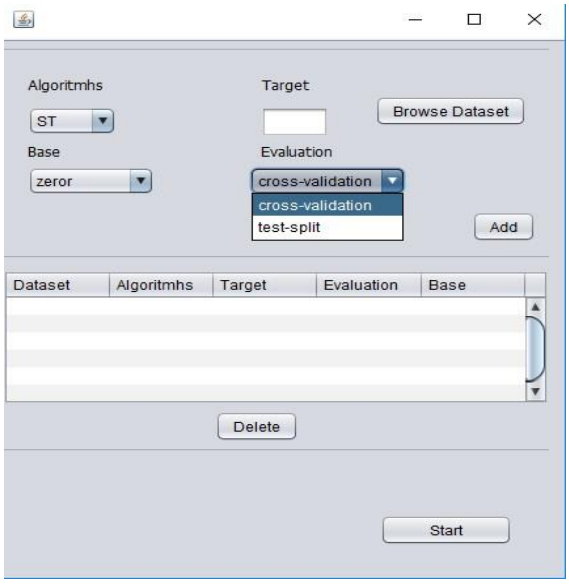


Ilustración 7. Interfaz gráfica para seleccionar tipo de evaluación.

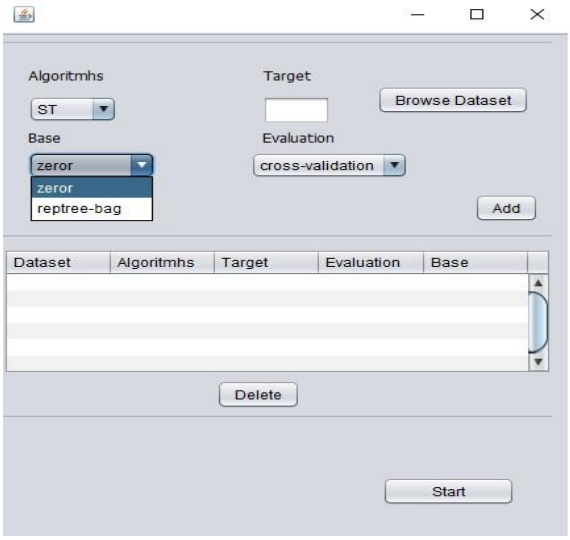


Ilustración 8. Interfaz gráfica para seleccionar tipo de base.

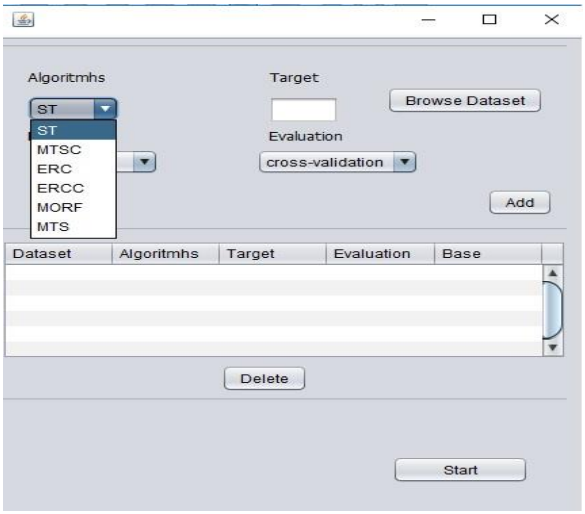


Ilustración 9. Interfaz gráfica para seleccionar algoritmo.

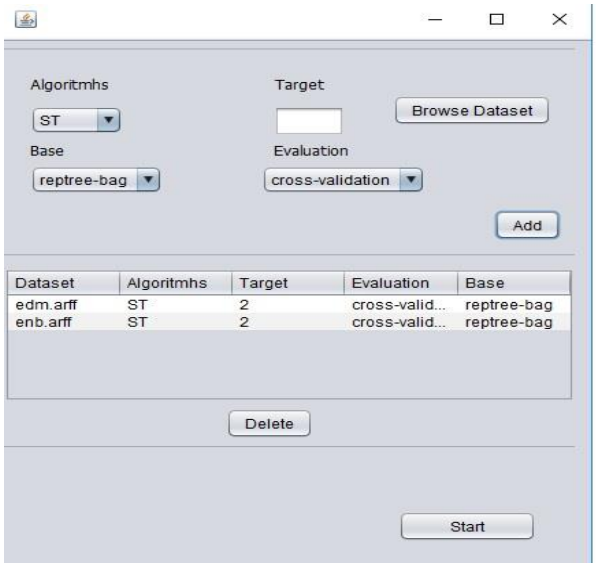


Ilustración 10. Interfaz gráfica para añadir datos.

Una vez seleccionados todos los parámetros necesarios para la puesta en marcha del sistema y añadidos los datos a la tabla de la interfaz, el siguiente paso consiste en la ejecución del experimento. Luego la interfaz permitirá mostrar los resultados finales de la experimentación y exportarlos a formato “csv”.

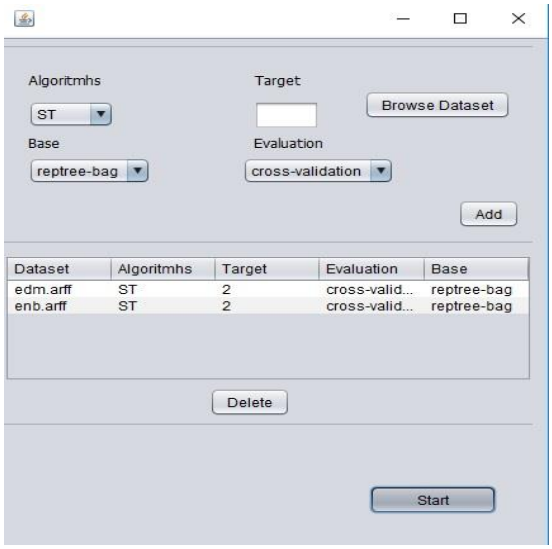


Ilustración 11. Interfaz gráfica para ejecutar experimento.

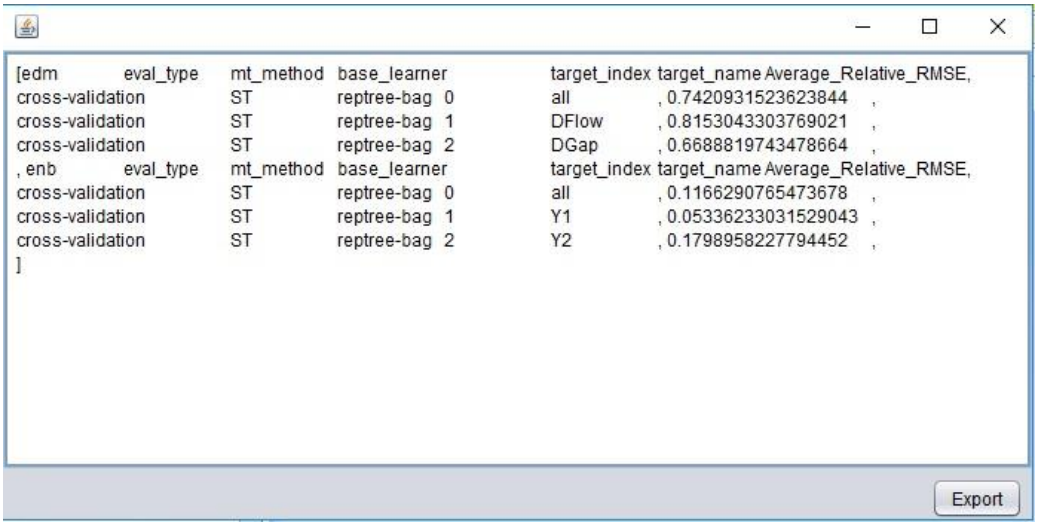


Ilustración 12. Interfaz gráfica para mostrar resultados del experimento.

La Tabla 22 muestra los resultados medios obtenidos (expresados en el intervalo [0; 1]). Para la evaluación de los resultados experimentales se utilizó como medida de evaluación el error cuadrático medio relativo por cada variable de salidas y el promedio para todas las salidas.

Dataset	***Algoritmo***	***# Targets***	***Average_Relative (aRRMSE)***
EDM	ST	2	Target_0 =0.7420931523623844 Target_1= 0.8153043303769021 Target_2= 0.6688819743478664
ENB	ST	2	Target_0= 0.1166290765473678 Target_1= 0.05336233031529043 Target_2= 0.1798958227794452
EDM	MTSC	2	Target_0 =0.7420931523623844 Target_1= 0.8122813189994877 Target_2= 0.6669765793060406
ENB	MTSC	2	Target_0 = 0.12053537850037291

			Target_1= 0.06333849666086776 Target_2= 0.17773226033987805
EDM	ERC	2	Target_0 = 0.7434879840999548 Target_1= 0.8179322620581745 Target_2= 0.6690437061417351
ENB	ERC	2	Target_0 = 0.12529859231409607 Target_1= 0.0637309465764009 Target_2= 0.18686623805179123

Tabla 22. Resultados del experimento.

3.8. CONCLUSIONES PARCIALES

Con el desarrollo de este capítulo se implementó la propuesta de solución, se realizaron las pruebas necesarias y se arriba a las siguientes conclusiones:

- Con la implementación de la herramienta se cumplió con el objetivo propuesto.
- Las pruebas unitarias realizadas ofrecieron una muestra detallada de los errores encontrados en las funcionalidades facilitando así la detección de los mismos en la implementación.
- Las pruebas de aceptación permitieron verificar que el sistema realiza correctamente los requisitos solicitados por el cliente.
- Se realizó un caso de estudio mediante el cual a partir de los resultados arrojados se puede concluir que la herramienta cumple satisfactoriamente con los requerimientos especificados por el cliente.

CONCLUSIONES GENERALES

Al término de la presente investigación, donde se han logrado cumplir los objetivos propuestos, se arriba a las siguientes conclusiones:

- Se hizo un análisis acerca de las metodologías usadas en el desarrollo de software haciendo una selección conveniente, concluyendo con el uso de la Metodología XP.
- Se realizó una selección de la herramienta Mulan como base para la ejecución de algoritmos de predicción de salidas compuestas.
- Se obtuvo como resultado de la presente investigación una herramienta gráfica para el desarrollo de experimentos aplicando algoritmos de predicción con fácil acceso y uso en el contexto de la predicción con salidas múltiples.

RECOMENDACIONES

Como resultado final del proceso investigativo, se plantean las siguientes recomendaciones a tener en cuenta para investigaciones futuras:

- Como todo producto de software se debe continuar el desarrollo de nuevas algoritmos que fortalezca la tarea de predicción con salidas múltiples.
- Poner a disposición de la comunidad internacional en los repositorios de código abierto la aplicación desarrollada para que sea evaluada y probada por los investigadores.

REFERENCIAS BIBLIOGRÁFICAS

Adamson, C. (2006). What Is Java. Retrieved from http://www.onjava.com/pub/a/onjava/2006/03/08/what-is-java.html

Asuncion, A., & Newman, D. (2007). *UCI machine learning repository*.

Bakir, G. H., Hofmann, T., Schölkopf, B., Smola, A. J., Taskar, B., & Vishwanathan, S. V. N. (2007). *Predicting Structured Data (Neural Information Processing)*. The MIT Press.

Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, *36*(1–2), 105–139.

Beck, Kent y Wesley, Addison. (1999). Extreme Programming Explained.

Blockeel, H., De Raedt, L., & Ramon, J. (2000). Top-down induction of clustering trees. *arXiv Preprint cs/0011032*.

Borchani, H., Varando, G., Bielza, C., & Larrañaga, P. (2015). A survey on multi-output regression. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *5*(5), 216–233.

Breiman, L. (1996a). Bagging predictors. *Machine Learning*, *24*(2), 123–140.

Breiman, L. (2001a). Random forests. *Machine Learning*, *45*(1), 5–32.

Breiman, L., & Friedman, J. H. (1997). Predicting multivariate responses in multiple linear regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *59*(1), 3–54.

Craig Larman. (2010). *UML y Patrones* (2da ed.). Murcia : Prentice Hall.

Durrant, B., FRANK, E., HUNT, L., HOLMES, G., MAYO, M., PFAHRINGER, B., … WITTEN, I. (2013). *Weka 3: Data Mining Software in Java*.

E. Gamma, R. Helm, R. Johnson, J. Vlissides, & . (n.d.). Desing Patterns.

Eleftherios Spyromitros-Xioufis. (2016). Multi-Target Regression via Input Space Expansion: Treating Targets as Inputs.

Goldberger, J., Roweis, S., Hinton, G., & Salakhutdinov, R. (2004). Neighbourhood components analysis. *Advances in Neural Information Processing Systems (NIPS)*, *17*, 513–520.

Groves, W., & Gini, M. (2011). *A regression model for predicting optimal purchase timing for airline tickets*. Technical Report 11-025, University of Minnesota, Minneapolis, MN.

Guide, M. U. (1998). The mathworks. *Inc., Natick, MA*, *5*, 333.

Ho, T. K., Hull, J. J., & Srihari, S. N. (1994). Decision combination in multiple classifier systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *16*(1), 66–75.

Ing. José Joskowicz. (n.d.). *Reglas y Prácticas en eXtreme Programming*. Virgo, España.

Ing. Software (Equipo 02). (n.d.). METODOLOGIA XP.htm.

Izenman, A. J. (1975a). Reduced-rank regression for the multivariate linear model. *Journal of Multivariate Analysis*, *5*(2), 248–264.

Izenman, A. J. (1975b). Reduced-rank regression for the multivariate linear model. *Journal of Multivariate Analysis*, *5*(2), 248–264.

José Luis Ávila Jiménez. (2013, Abril). Modelos de Aprendizaje Basados en Programación Genética para Clasificación Multi-Etiqueta.

Kittler, J., Hatef, M., Duin, R. P., & Matas, J. (1998). On combining classifiers. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *20*(3), 226–239.

Kocev, D., Vens, C., Struyf, J., & Džeroski, S. (2007a). Ensembles of Multi-Objective Decision Trees. In *Proceedings of the 18th European conference on Machine Learning* (pp. 624–631). Springer-Verlag.

Kocev, D., Vens, C., Struyf, J., & Džeroski, S. (2007b). *Ensembles of multi-objective decision trees*. Springer.

Li, T., Zhang, C., & Zhu, S. (2006). Empirical Studies on Multi-label Classification. In *IcTAI* (Vol. 6, pp. 86–92).

Oscar Belmonte Fernández. (2004). Introducción al lenguaje de programación Java.

Read, J. (2014). Tutorial. Meka 1.7. 3.

Roger S. Pressman. (2007). *"Ingeniería del Software. Un enfoque práctico"* (6ta Parte I).

S Dzeroski, J. G., J.Demsar. (2000). Predicting chemical parameters of river water quality from bioindicator data. *Applied Intelligence 1(13), 7–17*.

Sebastian Ventura. (2017). Evaluation and Comparison of Open Source Software Suites for Data Mining and Knowledge Discovery. *259*.

Spyromitros-Xioufis, E., Tsoumakas, G., William, G., & Vlahavas, I. (2014). Drawing Parallels between Multi-Label Classification and Multi-Target Regression. *arXiv Preprint arXiv:1211.6581v2*.

Spyromitros-Xioufis, E., Tsoumakas, G., William, G., & Vlahavas, I. (2014). Drawing Parallels between Multi-Label Classification and Multi-Target Regression. *arXiv Preprint arXiv:1211.6581v2*.

Spyromitros-Xioufis, E., Tsoumakas, G., William, G., & Vlahavas, I. (2014). Drawing Parallels between Multi-Label Classification and Multi-Target Regression. *arXiv Preprint arXiv:1211.6581v2*.

Struyf, J., Zenko, B., Blockeel, H., Vens, C., & Dzeroski, S. (2010). *Clus: User's Manual*.

Tsoumakas, G., Spyromitros-Xioufis, E., Vilcek, J., & Vlahavas, I. (2011). Mulan: A java library for multi-label learning. *The Journal of Machine Learning Research*, *12*, 2411–2414.

Tsoumakas, G., Spyromitros-Xioufis, E., Vrekou, A., & Vlahavas, I. (2014). Multi-Target Regression via Random Linear Target Combinations. *arXiv Preprint arXiv:1404.5065*.

Witten, I. H., & Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

BIBLIOGRAFÍA

“Bagging predictors,” Machine learning. **Breiman, L. 1996.** 2, 1996, Vol. 24, págs. 123-140.

ALGORITMO CURDS & WHEY PARA PROBLEMAS DE PREDICCIÓN CON SALIDAS COMPUESTAS EN LA HERRAMIENTA MULAN. **González, Héctor R., Hernández, Eduardo y Blanco, Antonio D. . 2016.** La Habana : s.n., 2016.

BORCHANI, H., VARANDO, G., BIELZA, C., & LARRAÑAGA, P. 2015. A survey on multi-output regression. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery.* 5, 2015, 216–233.

Chain, Nassir Sapag y Sapag Chain, Reynaldo. *Preparación y Evaluación de proyectos.* s.l. : Editorial Mac Graw Hill.

Evaluation of KNN-SP algorithm for multi-target prediction problems . **González, Héctor, y otros. 2016.** 3, La Habana : Ediciones Futuro, 2016, Vol. 10. ISSN: 2227-1899 .

Evaluation and Comparison of Open Source Software Suites for Data Mining and Knowledge Discovery. **Ventura, Sebastian. 2017.** 2017.

Fdez, Jesus Alcala. 2015. *Proyecto KEEL: Desarrollo de una Herramienta para el Análisis e Implementación de Algoritmos de Extracción de Conocimiento Evolutivos.* Granada, España : s.n., 2015.

Gonzalez, Héctor R., y otros. 2016. “ALGORITMO CURD & WHEYS PARA PROBLEMAS DE PREDICCION CON SALIDAS COMPUESTAS EN LA HERRAMIENTA MULAN. 2016.

Gutiérrez, José Luis Corredera y Bedoya López, Andres Alejandro. 2016. *Herramienta para la integración de algoritmos de predicción con salidas compuestas.* La Habana : s.n., 2016.

León, Rolando Alfredo Hernández y Coello González, Sayda. *El proceso de investigación científica.* Ciudad de La Habana : Editorial Universitaria, 2011. ISBN 978-959-16-1307-3.

ANEXOS

Anexo 1. Gráfica de resultados por categorías de cada herramienta

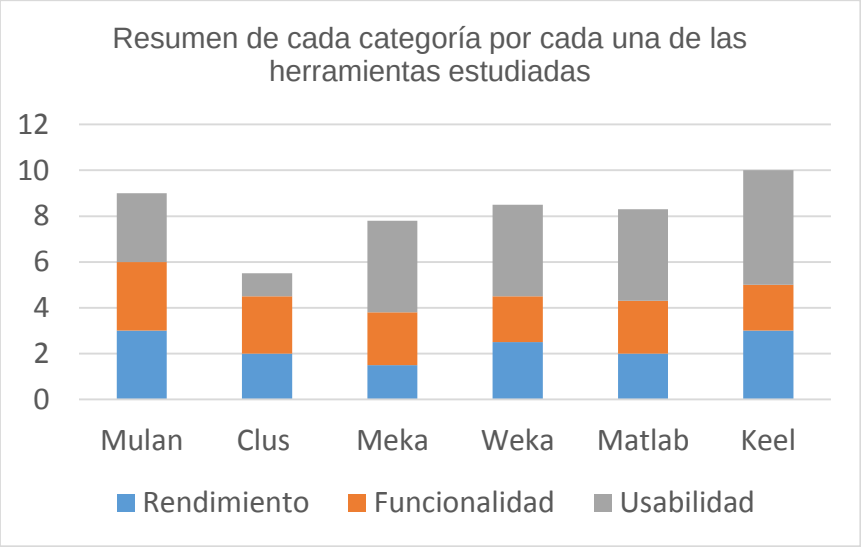


Ilustración 13. Resultados por categorías de cada herramienta.

Anexo 2. Historias de Usuarios

Historia de Usuario		
Número: HU2	**Usuario:** Cliente	
Nombre de historia: Especificar cantidad de salidas.		
Prioridad en negocio: Alta (Alta/ Media/ Baja)	**Riesgo en desarrollo:** Alto	
Puntos estimados: 2/5	**Puntos reales:** 2	**Iteración asignada:** 1
Programador responsable: Lisandra Hernández Colás y Andy Reinoso Brito.		
Descripción: Permite especificar la cantidad de salidas deseadas.		

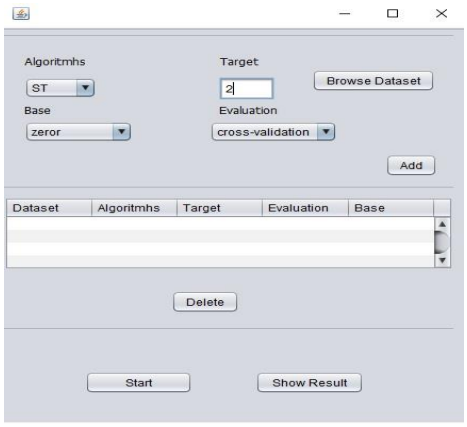
Observaciones: La especificación de las salidas tiene que ser según la base de datos cargada.
Interfaz: 

Tabla 23.HU2.Especificar cantidad de salidas.

Historia de Usuario		
Número: HU3	**Usuario:** Cliente	
Nombre de historia: Seleccionar tipo de evaluación.		
Prioridad en negocio: Alta (Alta/ Media/ Baja)	**Riesgo en desarrollo:** Alto	
Puntos estimados: 3/5	**Puntos reales:** 3	**Iteración asignada:** 1
Programador responsable: Lisandra Hernández Colás y Andy Reinoso Brito.		
Descripción: Permite seleccionar el tipo de evaluación que se va a utilizar al cargar la base de datos (validación cruzada o *test split*).		
Observaciones: Se debe seleccionar la evaluación que el usuario decida utilizar.		
Interfaz:		

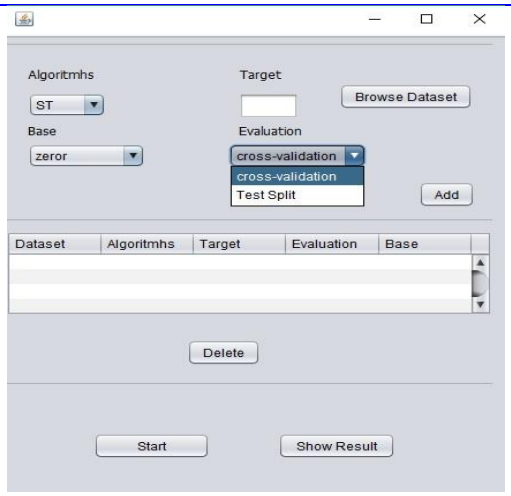


Tabla 24.HU3.Seleccionar tipo de evaluación.

Historia de Usuario		
Número: HU4	**Usuario:** Cliente	
Nombre de historia: Seleccionar algoritmo.		
Prioridad en negocio: Alta (Alta/ Media/ Baja)	**Riesgo en desarrollo:** Alto	
Puntos estimados: 3/5	**Puntos reales:** 3	**Iteración asignada:** 1
Programador responsable: Lisandra Hernández Colás y Andy Reinoso Brito.		
Descripción: Permite seleccionar el algoritmo a ejecutar (ST, MTS, MTSC, ERC, ERCC, MORF).		
Observaciones: Se debe seleccionar el algoritmo que el usuario desee para cada una de las bases de datos cargadas.		
Interfaz:		

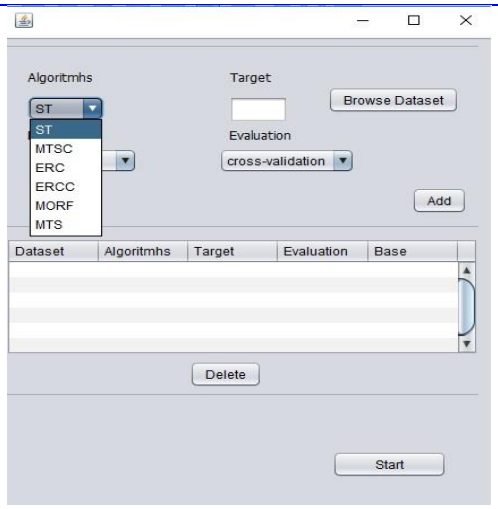


Tabla 25.HU4. Seleccionar algoritmo.

Historia de Usuario		
Número: HU5	**Usuario:** Cliente	
Nombre de historia: Selección del tipo de base		
Prioridad en negocio: Alta (Alta/ Media/ Baja)	**Riesgo en desarrollo:** Alto	
Puntos estimados: 3/5	**Puntos reales:** 3	**Iteración asignada:** 1
Programador responsable: Lisandra Hernández Colás y Andy Reinoso Brito.		
Descripción: Permite seleccionar el tipo de base a trabajar (*zeror, reptree-bag*)		
Observaciones: Se selecciona el tipo de base según la preferencia del usuario.		

Interfaz:

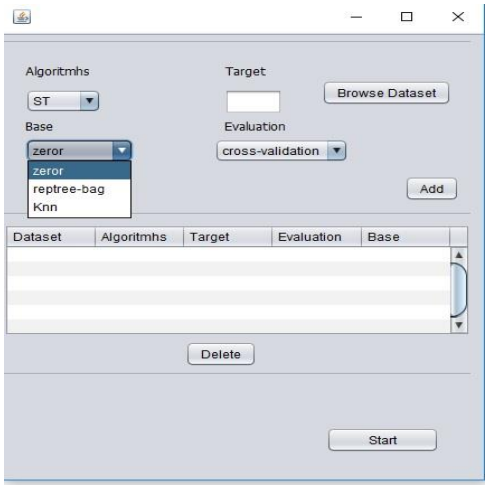


Tabla 26. HU5.Seleccionar tipo de base.

Historia de Usuario		
Número: HU6	**Usuario:** Cliente	
Nombre de historia: Añadir datos.		
Prioridad en negocio: Alta (Alta/ Media/ Baja)	**Riesgo en desarrollo:** Medio	
Puntos estimados: 3/5	**Puntos reales:** 3	**Iteración asignada:** 2
Programador responsable: Lisandra Hernández Colás y Andy Reinoso Brito.		
Descripción: Se añaden los datos en la tabla para la ejecución del experimento.		
Observaciones: Se deben haber seleccionado todos los parámetros para la ejecución del experimento.		

Interfaz:

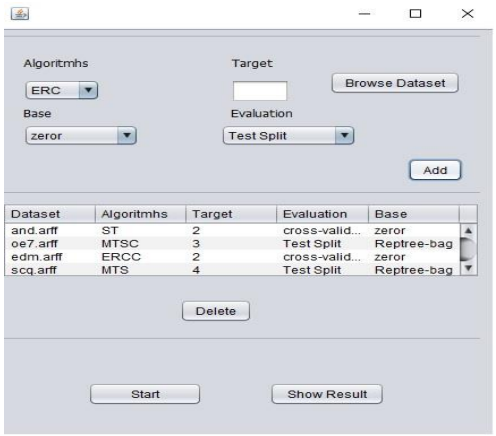


Tabla 27. HU6.Aañadir datos.

Historia de Usuario		
Número: HU7	**Usuario:** Cliente	
Nombre de historia: Eliminar datos.		
Prioridad en negocio: Alta (Alta/ Media/ Baja)	**Riesgo en desarrollo:** Medio	
Puntos estimados: 4/5	**Puntos reales:** 4	**Iteración asignada:** 2
Programador responsable: Lisandra Hernández Colás y Andy Reinoso Brito.		
Descripción: Permite eliminar los datos que el usuario no desee incluir en la ejecución del experimento.		
Observaciones: Deben haberse seleccionado los datos a eliminar.		

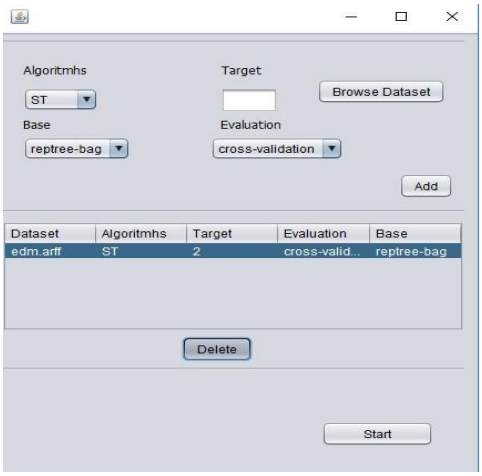
Interfaz:


Tabla 28. HU7.Eliminar datos.

Historia de Usuario		
Número: HU9	**Usuario:** Cliente	
Nombre de historia: Mostrar los resultados.		
Prioridad en negocio: Alta (Alta/ Media/ Baja)	**Riesgo en desarrollo:** Alto	
Puntos estimados: 4/5	**Puntos reales:** 4	**Iteración asignada:** 3
Programador responsable: Lisandra Hernández Colás y Andy Reinoso Brito.		
Descripción: Muestra el resultado del experimento.		
Observaciones: Debe haberse ejecutado al menos un experimento.		
Interfaz:		

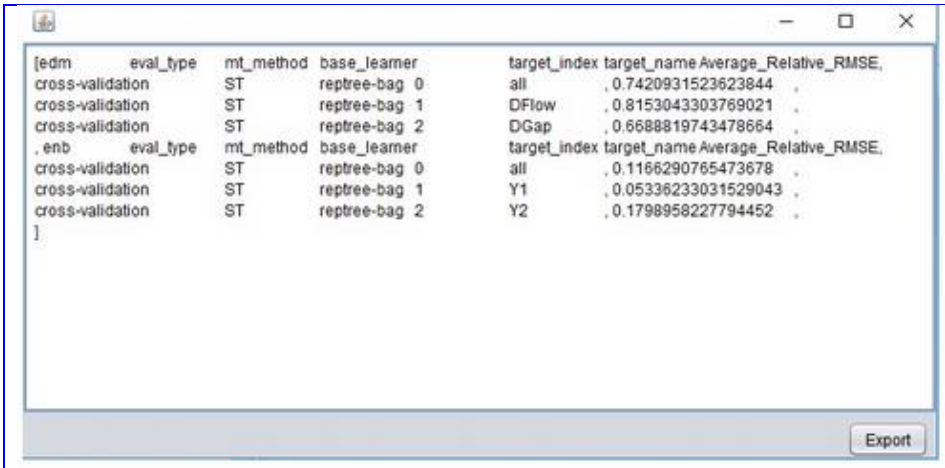


Tabla 29. HU9.Mostrar resultados.

Historia de Usuario		
Número: HU10	**Usuario:** Cliente	
Nombre de historia: Exportar resultados		
Prioridad en negocio: Media (Alta/ Media/ Baja)	**Riesgo en desarrollo:** Medio	
Puntos estimados: 3/5	**Puntos reales:** 3	**Iteración asignada:** 3
Programador responsable: Lisandra Hernández Colás y Andy Reinoso Brito.		
Descripción: Exporta los resultados en formato “csv”.		
Observaciones: Debe haberse ejecutado el experimento satisfactoriamente y arrojado un resultado final.		
Interfaz:		

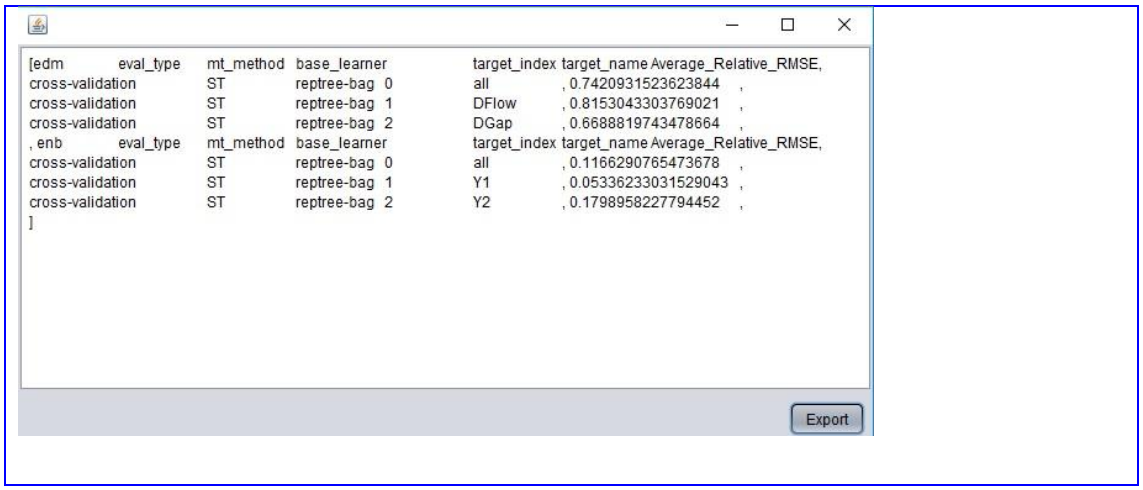


Tabla 30. HU10.Exportar resultados.

Anexo 3. Patrón Experto

Ilustración 14. Evidencia de patrones GRASP (Experto).

Anexo 4. Patrón Creador

Ilustración 15. Evidencia de patrones GRASP (Creador).

Anexo 5. Patrón Bajo Acoplamiento

```
public ExperimentMTR() {
}

@SuppressWarnings("empty-statement")
public static void main(String[] args) throws Exception {
    String recibe = args[1];
    String fileStem = args[0];
    String ta = args[3];
    int tar = Integer.parseInt(ta);
    String evaluacion = args[2];
    String metodo = args[5];
    String base1 = args[4];
    String slots = "";
```

Ilustración 16. Evidencia de patrones GRASP (Bajo Acoplamiento).

Anexo 6. Patrón Alta Cohesión

```
private void InformacionTabla() {
    tipoSalid.add(jTextField1.getText());
    informacion[0] = lnombre.get(c);
    informacion[1] = metodos.get(c);
    informacion[2] = tipoSalid.get(c);
    informacion[3] = tipoEval.get(c);
    informacion[4] = tipoBase.get(c);
    jTextField1.setText("");
    c++;
    jTable2.setModel(model);
    jTable2.setSelectionMode(1);
    model.addRow(informacion);
    for (int j = 0; j < informacion.length; j++) {
        informacion[j]=null;}
 }
private void EliminarF(int elim){
lnombre.remove(elim);
metodos.remove(elim);
tipoBase.remove(elim);
tipoSalid.remove(elim);
tipoEval.remove(elim);
ldireccion.remove(elim);
if(c>0)c--;
}
```

Ilustración 17. Evidencia de patrones GRASP (Alta Cohesión).

Anexo 7. Diagrama de clases del sistema

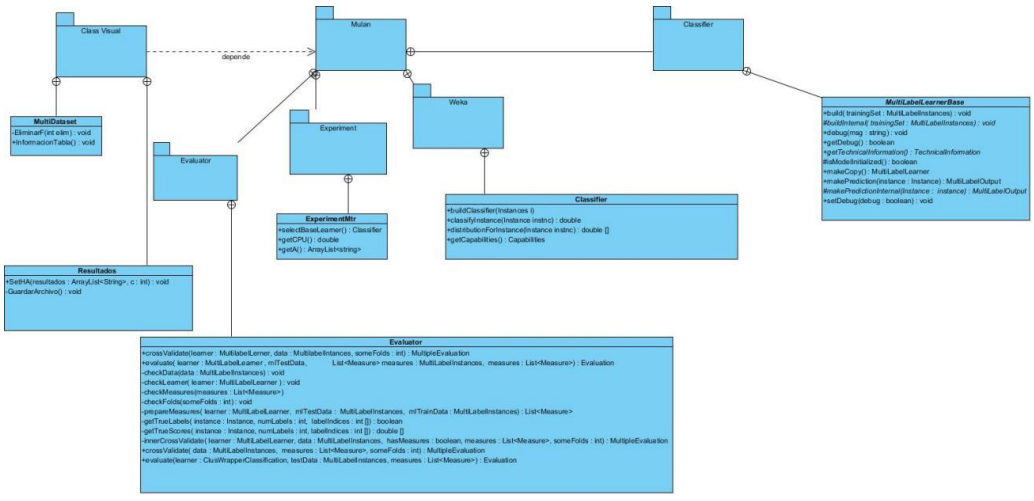


Ilustración 18. Diagrama de clases.

Anexo 8. Tareas de la ingeniería. Iteración 1

Tarea de Ingeniería	
Número de Tarea: 1	**Historia de usuario:** Especificar cantidad de salidas
Nombre de Tarea: Implementar la funcionalidad especificar cantidad de salidas.	
Tipo de Tarea: Desarrollo	**Puntos estimados:** 0.6
Fecha Inicio: 30/01/2017	**Fecha Fin:** 01/02/2017
Programador responsable: Lisandra Hernández Colás	
Descripción: Se implementarán las funcionalidades que permitan al usuario especificar la cantidad de salidas deseadas.	

Tabla 31. TI # 1 Implementar la funcionalidad especificar cantidad de salidas.

Tarea de Ingeniería	
Número de Tarea: 2	**Historia de usuario:** Especificar cantidad de salidas
Nombre de Tarea: Creación de la interfaz para especificar la cantidad de salidas deseadas.	
Tipo de Tarea: Desarrollo	**Puntos estimados:** 0.8
Fecha Inicio: 02/02/2017	**Fecha Fin:** 03/02/2017
Programador responsable: Lisandra Hernández Colás	
Descripción: Se creará una interfaz que permita especificar la cantidad de salidas.	

Tabla 32. TI # 2 Creación de la interfaz para especificar la cantidad de salidas deseadas.

Tarea de Ingeniería	
Número de Tarea: 1	**Historia de usuario:** Seleccionar tipo de Evaluación
Nombre de Tarea: Implementar la funcionalidad seleccionar tipo de Evaluación.	
Tipo de Tarea: Desarrollo	**Puntos estimados:** 1
Fecha Inicio: 06/02/2017	**Fecha Fin:** 08/02/2017
Programador responsable: Andy Reinoso Brito	
Descripción: Se implementarán las funcionalidades que permitan manejar el tipo de Evaluación para cada base de datos cargada, según la preferencia del usuario.	

Tabla 33. TI # 1 Implementar la funcionalidad seleccionar tipo de Evaluación.

Tarea de Ingeniería	
Número de Tarea: 2	**Historia de usuario:** Seleccionar tipo de Evaluación
Nombre de Tarea: Creación de la interfaz para seleccionar tipo de Evaluación.	
Tipo de Tarea: Desarrollo	**Puntos estimados:** 0.8
Fecha Inicio: 09/02/2017	**Fecha Fin:** 10/02/2017
Programador responsable: Andy Reinoso Brito	

Descripción: Se desarrollarán una interfaz para seleccionar el tipo de evaluación según la preferencia del usuario.

Tabla 34. TI # 2 Creación de la interfaz para seleccionar el tipo de Evaluación.

Tarea de Ingeniería	
Número de Tarea: 1	**Historia de usuario:** Seleccionar tipo de base
Nombre de Tarea: Implementar la funcionalidad seleccionar tipo de base.	
Tipo de Tarea: Desarrollo	**Puntos estimados:** 1
Fecha Inicio: 13/02/2017	**Fecha Fin:** 15/02/2017
Programador responsable: Andy Reinoso Brito	
Descripción: Se implementarán las funcionalidades que permitan manejar el tipo de base para cada base de datos cargada, según la preferencia del usuario.	

Tabla 35. TI # 1 Implementar la funcionalidad seleccionar tipo de Evaluación.

Tarea de Ingeniería	
Número de Tarea: 2	**Historia de usuario:** Seleccionar tipo de Evaluación
Nombre de Tarea: Creación de la interfaz para seleccionar tipo de Evaluación.	
Tipo de Tarea: Desarrollo	**Puntos estimados:** 0.8
Fecha Inicio: 16/02/2017	**Fecha Fin:** 17/02/2017
Programador responsable: Lisandra Hernández Colás	
Descripción: Se desarrollarán una interfaz para seleccionar el tipo de evaluación según el algoritmo seleccionado por el usuario.	

Tabla 36. . TI # 2 Creación de la interfaz para seleccionar el tipo de Evaluación.

Tarea de Ingeniería	
Número de Tarea: 1	**Historia de usuario:** Seleccionar algoritmo
Nombre de Tarea: Implementar la funcionalidad seleccionar algoritmo.	
Tipo de Tarea: Desarrollo	**Puntos estimados:** 1.2
Fecha Inicio: 20/02/2017	**Fecha Fin:** 22/02/2017
Programador responsable: Lisandra Hernández Colás	
Descripción: Se implementarán las funcionalidades que permitan manejar la tarea seleccionar algoritmo, o sea la aplicación de los algoritmos y las auxiliares que sea necesarias, según la preferencia del usuario.	

Tabla 37. TI # 1 Implementar la funcionalidad seleccionar algoritmo.

Tarea de Ingeniería	
Número de Tarea: 2	**Historia de usuario:** Seleccionar algoritmo
Nombre de Tarea: Creación de la interfaz para seleccionar el algoritmo deseado.	
Tipo de Tarea: Desarrollo	**Puntos estimados:** 1.6

Fecha Inicio: 23/02/2017	**Fecha Fin:** 24/02/2017
Programador responsable: Andy Reinoso Brito	
Descripción: Se crearán una interfaz para seleccionar el algoritmo deseado por el usuario.	

Tabla 38. TI # 2 Creación de la interfaz para seleccionar el algoritmo deseado.

Tarea de Ingeniería	
Número de Tarea: 1	**Historia de usuario:** Añadir datos
Nombre de Tarea: Implementar la funcionalidad añadir datos.	
Tipo de Tarea: Desarrollo	**Puntos estimados:** 3
Fecha Inicio: 27/02/2017	**Fecha Fin:** 01/03/2017
Programador responsable: Andy Reinoso Brito	
Descripción: Se implementarán las funcionalidades que le permitan al usuario añadir los datos para realizar el experimento.	

Tabla 39. TI # 1 Implementar la funcionalidad añadir datos.

Tarea de Ingeniería	
Número de Tarea: 2	**Historia de usuario:** Añadir datos
Nombre de Tarea: Creación de la interfaz para añadir datos del experimento.	
Tipo de Tarea: Desarrollo	**Puntos estimados:** 3
Fecha Inicio: 02/03/2017	**Fecha Fin:** 03/03/2017
Programador responsable: Lisandra Hernández Colás	
Descripción: Se creará la interfaz para la tarea añadir datos del experimento.	

Tabla 40. TI # 2 Creación de la interfaz para añadir datos del experimento.

Anexo 9. Tareas de la ingeniería. Iteración 2

Tarea de Ingeniería	
Número de Tarea: 1	**Historia de usuario:** Eliminar datos
Nombre de Tarea: Implementar la funcionalidad eliminar datos.	
Tipo de Tarea: Desarrollo	**Puntos estimados:** 1.5
Fecha Inicio: 06/03/2017	**Fecha Fin:** 08/03/2017
Programador responsable: Lisandra Hernández Colás	
Descripción: Se implementarán las funcionalidades que permitan al usuario eliminar los datos que no desee utilizar en la ejecución del experimento.	

Tabla 41. TI # 1 Implementar la funcionalidad para eliminar experimento.

Tarea de Ingeniería	
Número de Tarea: 2	**Historia de usuario:** Eliminar datos
Nombre de Tarea: Creación de la interfaz para eliminar datos.	
Tipo de Tarea: Desarrollo	**Puntos estimados:** 1.4
Fecha Inicio: 09/03/2017	**Fecha Fin:** 10/03/2017
Programador responsable: Andy Reinoso Brito	
Descripción: Se creará una interfaz que permitan eliminar los datos que e usuario no desee utilizar en la ejecución del experimento.	

Tabla 42. TI # 2 Creación de la interfaz para eliminar datos.

Anexo 10. Tareas de la ingeniería. Iteración 3

Tarea de Ingeniería	
Número de Tarea: 1	**Historia de usuario:** Exportar resultados
Nombre de Tarea: Implementar la funcionalidad exportar resultados.	
Tipo de Tarea: Desarrollo	**Puntos estimados:** 1.3
Fecha Inicio: 17/04/2017	**Fecha Fin:** 19/04/2017
Programador responsable: Andy Reinoso Brito	
Descripción: Se implementarán las funcionalidades que permitan al usuario exportar los resultados en formato “csv” luego de la ejecución del experimento.	

Tabla 43. TI # 1 Creación de la interfaz para exportar resultados.

Tarea de Ingeniería	
Número de Tarea: 2	**Historia de usuario:** Exportar resultados
Nombre de Tarea: Creación de la interfaz para exportar resultados.	
Tipo de Tarea: Desarrollo	**Puntos estimados:** 1.8
Fecha Inicio: 20/04/2017	**Fecha Fin:** 21/04/2017
Programador responsable: Lisandra Hernández Colás	
Descripción: Se creará la interfaz para manejar la aplicación de la tarea exportar resultados.	

Tabla 44. TI # 2 Creación de la interfaz para Exportar resultados.

Anexo 11. Pruebas de Aceptación

Prueba de Aceptación	
Código: HU2_P1	**Historia de Usuario:** Especificar cantidad de salidas.
Nombre: Añadir cantidad de salidas.	
Descripción: Se prueba la cantidad de salidas según la base de datos seleccionada.	

Condiciones de Ejecución: La cantidad de salidas tiene que ser un número entero y no puede ser cero.
Entrada/ Pasos de ejecución: Introducir el número.
Resultado Esperado: La cantidad de salida es cargada satisfactoriamente.
Evaluación de la Prueba: Satisfactoria

Tabla 45. Prueba de Aceptación # 1 Añadir cantidad de salidas

Prueba de Aceptación	
Código: HU3_P1	**Historia de Usuario:** Seleccionar tipo de Evaluación.
Nombre: Elegir Evaluación.	
Descripción: Se elige el tipo de evaluación.	
Condiciones de Ejecución: Se selecciona el tipo de Evaluación que se utilizará al cargar la Base de Datos.	
Entrada/ Pasos de ejecución: Permite elegir el tipo de evaluación deseada por el usuario (*cross validation, test split*).	
Resultado Esperado: El tipo de evaluación es cargado satisfactoriamente.	
Evaluación de la Prueba: Satisfactoria	

Tabla 46. Prueba de Aceptación # 1 Elegir Evaluación.

Prueba de Aceptación	
Código: HU4_P1	**Historia de Usuario:** Seleccionar algoritmo.
Nombre: Elegir algoritmo deseado por el usuario.	
Descripción: Se elige el algoritmo según la preferencia del usuario.	
Condiciones de Ejecución: Se selecciona el algoritmo que se utilizará al cargar la Base de Datos.	
Entrada/ Pasos de ejecución: Permite seleccionar el algoritmo a ejecutar (ST, MTS, MTSC, ERC, ERCC, MORF).	
Resultado Esperado: El algoritmo es cargado satisfactoriamente.	
Evaluación de la Prueba: Satisfactoria	

Tabla 47. Prueba de Aceptación # 1 Elegir algoritmo deseado por el usuario.

Prueba de Aceptación	
Código: HU5_P1	**Historia de Usuario:** Seleccionar tipo de base.
Nombre: Elegir el tipo de base deseado por el usuario.	
Descripción: Se elige el tipo de base según la preferencia del usuario.	

Condiciones de Ejecución: Se selecciona el tipo de base que se utilizará al cargar la base de datos.
Entrada/ Pasos de ejecución: Permite seleccionar el tipo de base deseado (*Zeror, Reptree-bag*).
Resultado Esperado: El tipo de base es cargado satisfactoriamente.
Evaluación de la Prueba: Satisfactoria

Tabla 48. Prueba de Aceptación # 1 Elegir el tipo de base deseado por el usuario.

Prueba de Aceptación	
Código: HU6_P1	**Historia de Usuario:** Añadir datos
Nombre: Añadir los datos para la ejecución del experimento.	
Descripción: Se muestra en una tabla los datos seleccionados. Permite visualizar el o los experimentos a ejecutar.	
Condiciones de Ejecución: Se debe añadir a la tabla la base de datos, cantidad de salidas, tipo de evaluación, tipo de algoritmo y el tipo de base.	
Entrada/ Pasos de ejecución: Permite eliminar los datos según la referencia del usuario.	
Resultado Esperado: Los parámetros para la ejecución del experimento son cargados satisfactoriamente.	
Evaluación de la Prueba: Satisfactoria	

Tabla 49. Prueba de Aceptación # 1 Añadir experimento.

Prueba de Aceptación	
Código: HU7_P1	**Historia de Usuario:** Eliminar datos
Nombre: Eliminar datos de la interfaz.	
Descripción: Se eliminan los datos deseados por el usuario de la tabla visualizada en la interfaz.	
Condiciones de Ejecución: Se deben seleccionar los datos que el usuario decida eliminar.	
Entrada/ Pasos de ejecución: Permite eliminar los datos según a preferencia del usuario.	
Resultado Esperado: El experimento es eliminado satisfactoriamente.	
Evaluación de la Prueba: Satisfactoria	

Tabla 50. Prueba de Aceptación # 1 Eliminar experimento.

Prueba de Aceptación	
Código: HU8_P1	**Historia de Usuario:** Ejecutar experimento.
Nombre: Verificar la adición de los datos.	
Descripción: Se prueba que todos los datos se añadan correctamente.	
Condiciones de Ejecución: Seleccionar todos los parámetros necesarios para la ejecución del experimento.	
Entrada/ Pasos de ejecución: Se procede a la ejecución del experimento.	
Resultado Esperado: El experimento es ejecutado satisfactoriamente.	
Evaluación de la Prueba: Satisfactoria	

Tabla 51. Prueba de Aceptación # 1 Ejecutar experimento.

Prueba de Aceptación	
Código: HU8_P2	**Historia de Usuario:** Ejecutar experimento.
Nombre: Aplicación de la ejecución del experimento.	
Descripción: Se procede a aplicar la ejecución del experimento y se calibran los parámetros de acuerdo al resultado deseado.	
Condiciones de Ejecución: La estación de trabajo debe contener espacio en memoria libre, para su correcto funcionamiento, de acuerdo al volumen de los datos.	
Entrada/ Pasos de ejecución: Permite cargar todos los datos añadidos para la puesta en marcha del experimento.	
Resultado Esperado: La ejecución del experimento visualiza los resultados satisfactoriamente.	
Evaluación de la Prueba: Satisfactoria	

Tabla 52. Prueba de Aceptación # 2 Ejecutar experimento.

Prueba de Aceptación	
Código: HU9_P2	**Historia de Usuario:** Mostrar resultados
Nombre: Mostrar los resultados del experimento.	
Descripción: Se muestra el resultado del experimento.	
Condiciones de Ejecución: La tarea para especificar la cantidad de salidas debe haberse ejecutado correctamente.	
Entrada/ Pasos de ejecución: Se procede a introducir el número para especificar la cantidad de salidas.	
Resultado Esperado: La cantidad de salidas es cargada satisfactoriamente.	
Evaluación de la Prueba: Satisfactoria	

Tabla 53. Prueba de Aceptación # 1 Mostrar resultados.

Prueba de Aceptación	
Código: HU10_P1	**Historia de Usuario:** Exportar resultados
Nombre: Exportar los resultados en formato “csv”.	
Descripción: Se guardan los resultados del experimento en formato “csv”.	
Condiciones de Ejecución: Se debe de haber realizado la corrida de al menos un experimento.	
Entrada/ Pasos de ejecución: Luego de ejecutado el experimento se procede a guardarlo en formato “csv”.	
Resultado Esperado: Los resultados obtenidos son guardados correctamente.	
Evaluación de la Prueba: Satisfactoria	

Tabla 54. Prueba de Aceptación # 1 Exportar resultados.

Glosario de Términos

Validación cruzada o *cross-validation*: Técnica utilizada para evaluar los resultados de un análisis estadístico y garantizar que son independientes de la partición entre datos de entrenamiento y prueba.

Test A/B: Término que se utiliza en el ámbito del marketing para describir experimentos aleatorios con dos variantes, A y B, siendo una la de control y la otra la variante.

ZeroRule: Algoritmo que usa la media (para variables numéricas) o la moda (para variables nominales) de la variable de salida.