Universidad de las Ciencias Informáticas
Facultad 1

“Herramienta de configuración para análisis y entrenamiento de grandes volúmenes de datos digitales”

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

Autor:

Andy Armas Pérez

Tutores:

Dr. Héctor Raúl González Diez

Ing. Vladimir Campos Kindelán

La Habana, septiembre de 2020

“Año 62 de la Revolución”

"No consideres el estudio como una obligación,
sino como una oportunidad para penetrar
en el bello y maravilloso mundo del saber"

Albert Einstein

DECLARACIÓN DE AUTORÍA

Yo, Andy Armas Pérez, declaro ser el único autor de la presente tesis y reconozco a la Universidad de Ciencias Informáticas los derechos patrimoniales de la misma con carácter exclusivo.

Para que así conste, firmo la presente a los 13 días del mes de octubre del año 2020.

Héctor Raúl González Diez

Firma del Tutor

Vladimir Campos Kindelán

Firma del Tutor

Andy Armas Pérez

Autor

CONTACTOS

Autor: Andy Armas Pérez
Correo electrónico: aarmas@estudiantes.uci.cu

Tutor: Ing. Héctor Raúl González Diez
Correo electrónico: hglez@uci.cu

Tutor: Ing. Vladimir Campos Kindelán
Correo electrónico: vladimirc@uci.cu

AGRADECIMIENTOS

A mi tutor **Vladimir**, por ser mi guía desde el primer momento en que comencé a elaborar esta investigación.

A mi madre **Bárbara**, por apoyarme hasta el último segundo de estudio, perseverancia y dedicación de esta tesis.

A mi hermano **Alejandro**, por darle el cuidado y la forma final de manera profesional a este documento de culminación de estudio.

A mi padre **Elso**, por recordarme constantemente lo importante que es superar esta fase en mi vida.

A **Gallego**, por apoyar todo este proceso de superación.

DEDICATORIA

A mis **padres**, gracias a ellos he podido optar por esta carrera universitaria.

ÍNDICE

RESUMEN

La presente investigación responde a la necesidad de desarrollar una herramienta que facilite la aplicación y gestión de volúmenes de datos por parte de los integrantes de la Línea de Desarrollo: Inteligencia Artificial, de la Universidad de las Ciencias Informáticas (UCI). Se desarrolló una herramienta para configurar el análisis y entrenamiento de grandes volúmenes de datos digitales, la cual, incluye la vinculación de algoritmos de solución relacionados con la inteligencia artificial a partir del uso del Entorno de Desarrollo Integrado (IDE) Pycharm utilizando el lenguaje de programación Python con las librerías sklearn y pyqt5.

La metodología empleada fue XP, por ser una metodología ágil, resumido en cuatro fases: Planeación, Diseño, Codificación y Pruebas, donde, en la primera fase se emplearon las Historias de Usuario para la recolección de información del cliente. Se utilizaron, en adición, la extracción de requisitos funcionales y no funcionales para dar paso a la confección de Tarjetas CRC, mostrando las clases, sus responsabilidades y colaboradores. Se realizaron Tareas de Ingenierías para llevar mediante pasos y acciones la codificación de las Historias de Usuario, anteriormente mencionadas, para la confección de la herramienta. Se mostraron los estándares de codificación efectuados durante el desarrollo del software. Por último, se realizaron las Pruebas de Aceptación para validar que las funciones de la herramienta satisfacen las necesidades del cliente.

Palabras Claves:
Inteligencia Artificial, Minería de Datos, Proceso KDD

SUMMARY

The present investigation responds to the necessity of developing a tool that facilitates the application and administration of data volumes on the part of the members of the Line of Development: Artificial intelligence of the “Universidad de las Ciencias Informáticas” (UCI). a tool was developed to configure the analysis and training of big volumes of digital data, which includes the linking of solution algorithms related with the artificial intelligence starting from the use of the Environment of Integrated Development (IDE) Pycharm using the programming language Python with the bookstores sklearn and pyqt5.

The used methodology was XP, to be an agile methodology, summarized in four phases: Planning, Design, Code and Tests, where in the first phase User's Histories were used for the gathering of the client's information. They were used in addition the extraction of functional and not functional requirements to open the way to the making of Cards CRC, showing the classes, their responsibilities and collaborators. They were made Tasks of Engineerings to take by means of steps and actions the code of User's Histories previously mentioned for the making of the tool. The code standards were shown made during the development of the software. Finally, the Acceptance Tests were carried out to validate that the functions of the tool satisfy the client's needs.

Key words:

Artificial intelligence, Mining of Data, Process KDD

INTRODUCCIÓN

En el mundo de hoy, las mayorías de las actividades y operaciones generan un creciente volumen de información; sin embargo, la habilidad de los humanos para procesarla y asimilarla permanece constante (Olson, y otros, 2008). Además, la información en sí misma tiene pocas ventajas, su sistematización, incorporación y utilización son los elementos que aportan su valor añadido: el conocimiento. Es necesario crear sistemas que generen conocimiento, para asegurar el uso productivo de la información y guiar una toma de decisiones óptima (Canals, y otros, 2003) (Dalkir, 2005).

En las últimas décadas, con el adelanto de la informatización en las diferentes esferas de la sociedad y el perfeccionamiento acelerado y continuo de las Tecnologías de la Información y las Comunicaciones (TIC), el desarrollo de *software* se ha convertido en un elemento de gran importancia para la sociedad actual. Su impacto se manifiesta en diversos ámbitos de la actividad humana, contribuyendo al mejoramiento de la calidad de vida de las personas y cambiando su forma de actuar para con la sociedad. El desarrollo de *software* para la educación ha transformado los métodos tradicionales de enseñanza, aprovechando los beneficios que brindan las TIC, para un mejor desempeño del proceso de enseñanza-aprendizaje y la obtención de resultados superiores en cualquier ámbito. (Canals, y otros, 2003)

Hoy día, tanto las comunidades científicas, organizaciones y gobiernos, invierten en función del desarrollo de la gestión de la información y el conocimiento, a través de proyectos, congresos, postgrados y desarrollo de sistemas con este fin. Algunos ejemplos son: las políticas trazadas por la Unión Europea para incrementar la competitividad de una economía basada en el conocimiento, las acciones realizadas por la OPS/OMS en países en desarrollo y las facilidades brindadas por la UNESCO para desarrollar software para el procesamiento de la información (UNESCO-IBI). (Dalkir, 2005)

La limitación humana y la necesidad de técnicas inteligentes para deducir nuevo conocimiento a partir de grandes volúmenes de datos condicionan el surgimiento de un importante campo, la Minería de Datos. La Minería de Datos (MD) o KDD (Knowledge Discovery in Databases), como se le comenzó a llamar a inicios del año 1996, se define como el proceso de extraer conocimiento útil y comprensible, previamente desconocido, de grandes cantidades de datos almacenados en distintos formatos. La MD se desarrolla por el reconocimiento de un nuevo potencial: el valor de la gran cantidad de datos almacenados informáticamente en los sistemas de información de instituciones, empresas, gobiernos y particulares (Gorunescu, 2011).

La MD permite que los datos pasen de ser un "producto" a ser una "materia prima" que hay que explotar para obtener el verdadero "producto elaborado", el conocimiento (Cios, y otros, 2007). Dado que la MD excede la capacidad humana para el análisis de grandes volúmenes de datos, la utilización plena de los datos almacenados depende del uso de técnicas del descubrimiento del conocimiento (Molina, y otros, 2006), entre las que se encuentran las de Inteligencia Artificial.

Probablemente, en pocos años, el uso de la MD se haya extendido a todas las actividades humanas complejas en las que interviene gran cantidad de datos y variables. Se podrán analizar y comprimir datos para tomar decisiones de poca importancia y servir como medio de apoyo para decisiones complejas o de gran trascendencia. (Gorunescu, 2011)

De acuerdo con la definición dada por Michalski en 1986, aprender es la habilidad de adquirir nuevo conocimiento, desarrollar habilidades para analizar y evaluar problemas a través de métodos y técnicas, así como también por medio de la experiencia propia, siendo un requisito que el resultado del aprendizaje sea entendible para el hombre (Michalski, 1986)

El proceso de MD involucra numerosos pasos e incluye muchas decisiones que deben ser tomadas por el usuario. Entre ellas, se dice que la adecuación de los datos para que las técnicas de descubrimiento puedan utilizarlos demanda el 70% del esfuerzo. Para organizar este proceso han surgido varias metodologías o procedimientos que lo guían. La primera de ellas fue el Proceso KDD (Fayyad, y otros, 1996), estructurado en las siguientes etapas:

- Comprensión del dominio de la herramienta.
- Limpieza y preprocesamiento de los datos.
- Elección de la tarea de minería de datos.
- Elección del algoritmo(s) de minería de datos.
- Minería de datos.

- Interpretación de los patrones encontrados.

En Cuba, una de las estrategias para el progreso tecnológico paulatino de la sociedad está dirigida hacia la creación de productos de software que aporten soluciones novedosas y constituyan pasos importantes hacia la digitalización del país. Para lograrlo, cuenta con la Universidad de las Ciencias Informáticas (UCI), la cual, tiene entre sus principales objetivos proveer productos informáticos que posibiliten el creciente desarrollo tecnológico del país. (Universidad de las Ciencias Informáticas, 2020)

La universidad se encuentra organizada en varios grupos de investigación, entre ellos se encuentra la Línea de Desarrollo Inteligencia Artificial y Reconocimiento de Patrones, encargado del desarrollo de soluciones informáticas dentro del área de Inteligencia Artificial. (Universidad de las Ciencias Informáticas, 2020)

En esta línea de investigación se han identificado algunos aspectos, que conforman el banco de problemas, referido en este caso al tratamiento de grandes registros de información desorganizada, sin una adecuada clasificación, así como una excesiva dispersión de los datos que conforman la misma. Esta falta de ordenamiento de la información se traduce a la falta de indexación de los datos, provocando grandes demoras en la ejecución de los algoritmos de búsqueda, así como costos excesivos de procesamiento y consumo de memoria. Por otra parte realizarlo a través de métodos manuales implicaría para grandes volúmenes de datos, movilizar grandes equipos de trabajo de alrededor de 10 a 20 personas, siendo el proceso demorado y poco factible por las horas hombre empleadas en este. Otro de los problemas presentes, es el trabajo con datos no clasificados, que igualmente hace costoso el proceso en consumo de tiempo, y realizar esta clasificación de forma manual haciendo uso de un gran personal, puede tomar hasta meses de trabajo en dependencia del volumen de datos a procesar. Por último la dispersión de esta información, impide identificar los datos que no aportan una información concluyente, teniendo que realizar una revisión exhaustiva de todo el volumen para la detección y eliminación de los datos sobrantes, una vez más, para ello, requeriría grandes cantidades de tiempo y personal calificado para validar toda la información.

Por estas razones anteriormente expresadas se formula el siguiente **problema de investigación:** ¿Cómo identificar automáticamente patrones de comportamiento y características en grandes volúmenes de datos digitales? Para solucionar dicho problema se tendrán en cuenta referentes teóricos inmersos en el siguiente **objeto de estudio:** La implementación de herramientas KDD en problemas de inteligencia artificial, enfatizando en el **campo de acción:** La implementación de herramientas de configuración para análisis y entrenamiento de grandes volúmenes de datos digitales. Como **objetivo general la investigación** se determina: Desarrollar una herramienta para configurar el análisis y entrenamiento de grandes volúmenes de datos digitales. Con el interés de dar cumplimiento al objetivo general propuesto se identifican los siguientes **objetivos específicos:**

1. Evaluar los principales referentes teóricos acerca de KDD y la implementación de herramientas inteligentes.
2. Identificar requerimientos para implementar una herramienta para configurar el análisis y entrenamiento de grandes volúmenes de datos digitales.
3. Implementar la herramienta para configurar el análisis y entrenamiento de grandes volúmenes de datos digitales.
4. Verificar la herramienta para configurar el análisis y entrenamiento de grandes volúmenes de datos digitales.
5. Validar la herramienta para configurar el análisis y entrenamiento de grandes volúmenes de datos digitales.

Para llevar a cabo esta investigación se aplicaron métodos y técnicas en correspondencia con los objetivos del trabajo, así como una metodología para la obtención de datos verídicos, confiables y útiles para la institución, que posibilitará, además, el diseño de una herramienta de configuración para el análisis y entrenamiento de grandes volúmenes de datos digitales.

Métodos teóricos:

- **Histórico – lógico:** para precisar las principales características que presenta el proceso de desarrollo de un proceso KDD, su trayectoria histórica, evolución desde su surgimiento hasta la actualidad.
- **Análisis sintético:** que permite precisar el marco teórico referencial al consultar la bibliografía especializada en cuanto al tema abordado e identificar elementos claves que contribuyan a la solución del problema de investigación planteado, permite sintetizar

conceptos que ayudaran a comprender la solución del problema. Este facilita el análisis de las principales metodologías, herramientas y pruebas de software existentes para la selección de las indicadas en el contexto del objeto y campo identificado.

- **Modelación:** se empleó para la confección de los prototipos funcionales, la realización de los diagramas necesarios en el proceso de desarrollo de software, el diseño de la base de datos, así como la representación de los requisitos, haciendo una representación abstracta de la solución y facilitando el entendimiento del proceso de desarrollo de la misma.

Métodos empíricos:

- **Análisis documental:** proporcionó la revisión de la literatura necesaria para obtener la información del estado actual del objeto de investigación considerándose diversos autores que han trabajado el tema y sus resultados lo que posibilita llegar a conclusiones certeras del tema.
- **Revisión Bibliográfica:** se empleó en las actividades de búsqueda, identificación y análisis de la información existente.
- **Observación:** permitió identificar las principales deficiencias en las actuales herramientas que emplean un proceso KDD y sus características para una correcta creación de la aplicación.
- **Entrevista al cliente:** se empleó mediante un intercambio verbal con el cliente para obtener la mayor cantidad de información posible, lograr definir las necesidades de los clientes en el levantamiento de los requisitos de la herramienta y obtención de la información necesaria para el desarrollo de la investigación que permitieron definir el problema de investigación.

Métodos estadísticos:

- **Estadística descriptiva**: se utilizó para la determinación y análisis de la información obtenida con las indagaciones empíricas tales como: entrevistas a partir de la mediana conjunta.

La investigación presenta la siguiente estructura:

- Capítulo 1 “Caracterización de los procesos de KDD. Implementación de herramientas para la configuración del análisis y entrenamiento de datos digitales”: Se exponen las principales teorías y posiciones de autores que se han referido a las categorías de análisis relacionadas con la investigación.
- Capítulo 2 “Planificación y diseño de la herramienta de configuración para el análisis y entrenamiento de grandes volúmenes de datos digitales”: Se exponen las bases metodológicas fundamentales para el levantamiento de información y a partir de ellas construir el modelo de dominio, identificar los requisitos funcionales y no funcionales, se redactan las historias de usuario y se plantea a través de la fase de diseño, el estilo arquitectónico, patrones de diseño, diagramas de clases, diagrama de componentes.
- Capítulo 3 “Desarrollo y validación de la herramienta de configuración para análisis y entrenamiento de grandes volúmenes de datos digitales”: En este capítulo se expone la codificación de los principales métodos que responden a las funcionalidades descritas para cada tarea de ingeniería. También se aborda la planificación de los casos de pruebas y la ejecución de pruebas de aceptación.

CAPÍTULO 1, “Caracterización de los procesos de KDD. Implementación de herramientas para la configuración del análisis y entrenamiento de datos digitales”.

Actualmente, múltiples empresas cuentan con grandes volúmenes de datos digitales para analizar y obtener información útil para su gestión y desarrollo, donde el número de posibles relaciones es extenso y resulta prácticamente imposible validar cada una de ellas. Para resolver este problema se utilizan estrategias de búsqueda, extraídas del área de aprendizaje automático (Berry, y otros, 1997), a partir de herramientas que funcionan fijándoles objetivos de búsqueda concretos. Si bien, la minería de datos es la impresión de que se puede aplicar como herramienta a los datos, se debe tener un objetivo, o al menos una idea general de lo que se busca. El coste de esta prospección de datos debe ser coherente con el beneficio esperado teniendo en cuenta que ha bajado su precio, pero el coste en tiempo, personal y consultoría se ha incrementado, llegando en algunos casos a hacer no viable el proyecto. (Berry, y otros, 1997)

Proceso KDD

KDD, Knowledge Discovery in Databases, es un término acuñado en 1989 y se refiere a todo el proceso de extracción de conocimiento a partir de una base de datos, marcando un cambio de paradigma cuyo principal aporte es el conocimiento útil que seamos capaces de descubrir a partir de los datos. (Data Mining and Knowledge Discovery Handbook, 2005)

El descubrimiento de conocimientos en bases de datos KDD, se define como el proceso de identificar patrones significativos en los datos, que sean válidos, novedosos, potencialmente útiles y comprensibles para un usuario. El proceso global consiste en transformar información de bajo nivel en conocimiento de alto nivel. El proceso KDD es interactivo e iterativo, conteniendo los siguientes pasos (Data Mining and Knowledge Discovery Handbook, 2005):

1. Abstracción del escenario

Se precisa entender la problemática de manera global, determinando el contexto en el que se desarrolla para proponer y proyectar soluciones viables y reales. Es importante conocer las propiedades, limitaciones y reglas del escenario en estudio, para posteriormente definir las metas a alcanzar.

2. Selección de los datos

Del conjunto de datos recolectados y una vez definidos los objetivos a alcanzar, se deben elegir datos disponibles para realizar el estudio e integrarlos en uno solo que permita alcanzar los objetivos del análisis. Muchas veces esta información puede encontrarse en una misma fuente (centralizado) o pueden estar distribuidos.

3. Limpieza y preprocesamiento

En esta etapa se determina la confiabilidad de la información, es decir, realizar tareas que garanticen la utilidad de los datos. Para esto, se hace la limpieza de datos (tratamiento de datos perdidos o remover valores atípicos). Esto implica eliminar variables o atributos con datos faltantes o eliminar información no útil para este tipo de tareas como el texto.

4. Transformación de los datos

En esta etapa se mejora la calidad de los datos con transformaciones que involucran, reducción de dimensionalidad (disminuir la cantidad de variables del conjunto de datos) o transformaciones como la discretización que permite convertir los valores que son números a categóricos.

5. Selección de la tarea apropiada de Minería de Datos

Fase en la que se elige el paradigma apropiado de Minería de Datos, ya sea la clasificación, regresión o agrupación, según los objetivos que se hayan planteado para la investigación (predicción o descripción), la primera, diseñada para encontrar un modelo que pueda ser utilizado en casos futuros y desconocidos; mientras que la segunda, solo para observar y describir su comportamiento.

6. Elección del algoritmo de Minería de Datos

Posteriormente, se procede a seleccionar la técnica o algoritmo de Minería de Datos. Se pueden combinar o incluir más de uno para la búsqueda de patrones y la obtención de conocimientos. La meta de aprendizaje se enfoca en explicar la razón por la que un algoritmo funciona mejor que otro en determinadas problemáticas. Cada algoritmo tiene su propia esencia, su propia manera de trabajar y obtener los resultados, por lo que es recomendable conocer las propiedades de aquellos candidatos a utilizar y ver cuál se ajusta mejor a los datos y objetivos planteados.

7. Aplicación del algoritmo

Una vez seleccionada las técnicas, el paso siguiente es su aplicación a los datos ya escogidos, limpios y procesados. Es posible que la ejecución de los algoritmos sean varias, intentando ajustar los parámetros que optimicen los resultados. Estos parámetros pueden variar de acuerdo al método seleccionado.

8. Evaluación

Una vez aplicados los algoritmos al conjunto de datos, se procede a evaluar los patrones que se generaron y el rendimiento que se obtuvo para verificar que cumpla con las metas planteadas en las primeras fases. Para realizar esta evaluación existe una técnica que se llama Validación Cruzada (Devivjer, y otros, 1982), (Refaeilzadeh, y otros, 2008) (Elkan, 2011), la cual realiza una partición de los datos, dividiéndolos en: entrenamiento, que servirán para crear el modelo, y prueba, que serán utilizados para determinar el correcto funcionamiento del algoritmo.

9. Aplicación

Si todos los pasos se siguen correctamente y los resultados de la evaluación se satisfacen, la última etapa consiste en aplicar el conocimiento encontrado al contexto y comenzar a resolver sus problemáticas. Si de lo contrario, los resultados no son satisfactorios entonces es necesario regresar a las anteriores etapas a realizar algún ajuste, analizando desde la selección de los datos hasta la etapa de evaluación.

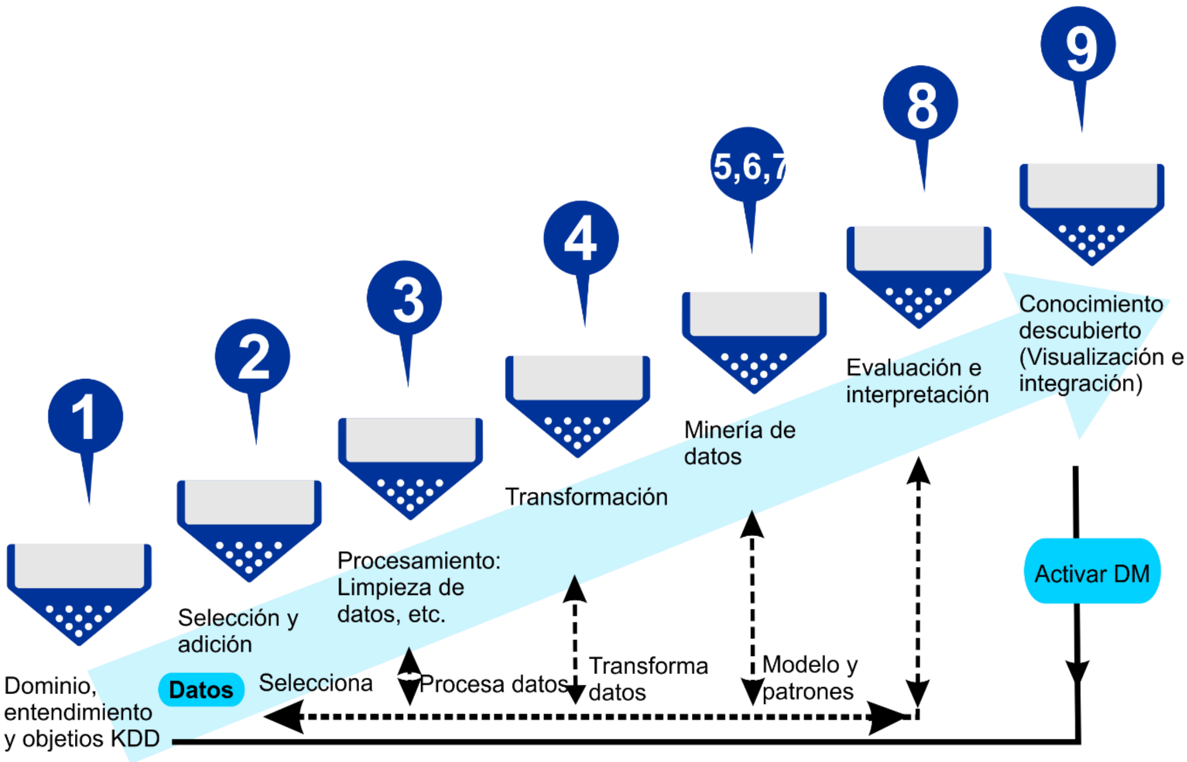


Figura Nº 1. Esquema del Descubrimiento de Conocimiento en Base de Datos. **Fuente:** (Data Mining and Knowledge Discovery Handbook, 2005)

1.1. KDD y MD

Los términos KDD y MD son a menudo confundidos como sinónimos. En general se acepta que la MD Minería de Datos, es un paso particular en el proceso, que consiste en la aplicación de algoritmos específicos para extraer patrones (modelos) de los datos. La KDD por su parte, incluye otros como son: la preparación de los datos, la selección y limpieza de los mismos, la incorporación de conocimiento previo y la propia interpretación de los resultados de minería. Estos pasos aplicados de una manera iterativa e interactiva aseguran que un conocimiento útil se extraiga de los datos. (Landa, 2016)

1.1.1. Inicios de la Minería de Datos

La idea de Minería de Datos no es nueva. (Maimon , y otros, 2010), (Zhu, y otros, 2007). Ya desde los años sesenta los estadísticos manejaban términos como data fishing, data mining o data archaeology, con la idea de encontrar correlaciones sin una hipótesis previa en bases de datos con ruidos. A principios de los años ochenta, Rakesh Agrawal, Gio Wiederhold, Robert Blum y Gregory Piatetsky-Shapiro, entre otros, empezaron a consolidar los términos de data mining y KDD. A finales de los años ochenta sólo existían un par de empresas dedicadas a esta tecnología. Las listas de discusión sobre este tema las forman investigadores de más de ochenta países. Esta tecnología ha sido un punto de encuentro entre personas pertenecientes al ámbito académico y al de los negocios. Es una tecnología compuesta por etapas que integra varias áreas y que no se debe confundir con un gran software. (Revista Científica Mundo de la Investigación y el Conocimiento., 2019)

1.1.2. Minería de Datos

En la actual sociedad de la información, donde cada día se multiplica la cantidad de datos almacenados casi de forma exponencial, la minería de datos es una herramienta fundamental para analizarlos y explotarlos de forma eficaz para los objetivos de cualquier organización. La minería de datos se define también como el análisis y descubrimiento de conocimiento a partir de datos (Maimon , y otros, 2010). La minería de datos hace uso de todas las técnicas que puedan aportar información útil, desde un sencillo análisis gráfico, pasando por métodos estadísticos más o menos complejos, complementados con métodos y algoritmos del campo de la inteligencia artificial (García Serrano, 2012) y el aprendizaje automático que resuelven problemas típicos de agrupamiento automático, clasificación, predicción de valores, detección de patrones, asociación de atributos (Bishop, 2008), (Flach, 2012), (Witten, y otros, 2011) (Raschka, 2015) .

La minería de datos es un intento de buscarle sentido a la explosión de información que actualmente puede ser almacenada. Hoy en día, los datos no están restringidos a tuplas representadas únicamente con números o caracteres. El avance de la tecnología para la gestión de bases de datos hace posible integrar diferentes tipos de datos, tales como imagen, video, texto y otros datos numéricos en una base de datos sencilla, facilitando el procesamiento multimedia. Como resultado, la mezcla tradicional ad hoc de técnicas estadísticas y herramientas de gestión de datos no son adecuadas para analizar esta vasta colección de datos desiguales. (Maimon , y otros, 2010)

El almacenamiento de información en formatos digitales es cada vez más barato y sencillo, por lo que hay que intentar sacar partido a estos volúmenes de información para la toma de decisiones. La tecnología informática constituye la infraestructura fundamental de las grandes organizaciones y permite, hoy, registrar múltiples detalles de la vida de las empresas. Las bases de datos posibilitan almacenar cada transacción, así como otros muchos elementos que reflejan la interacción de la organización con otras organizaciones, clientes, o internamente, entre sus divisiones y empleados. Es imprescindible convertir los grandes volúmenes de datos existentes en experiencia, conocimiento y sabiduría, formas que atesora la humanidad para que sea útil a la toma de decisiones, especialmente en las grandes organizaciones y proyectos científicos. (Fernández, 2015)

1.1.2.1. Técnicas de minería de datos

Las técnicas de Minería de Datos provienen de la inteligencia artificial (García Serrano, 2012) y de la propia estadística. Dichas técnicas, no son más que algoritmos sofisticados que se aplican sobre un conjunto de datos para obtener resultados, patrones o modelos a partir de los datos recopilados. Las técnicas de Minería de Datos se clasifican en dos grandes categorías: supervisadas o predictivas y no supervisadas o descriptivas. (Técnicas de reducción de datos en KDD. El uso de Algoritmos Evolutivos para la Selección de las Instancias., 2006)

No supervisados o descriptivas:

- Clustering: Numérico, Conceptual, Probabilístico
- Asociación: A-Priori, Partition, Eclat

Supervisados o predictivas:

- Predicción: Regresión, Árboles de Predicción, Estimador de Núcleos
- Clasificación: Tabla de Decisión, Árboles de Decisión, Inducción de Reglas, Basado en Ejemplares, Redes de Neuronas, Lógica Borrosa, Técnicas Genéticas, Bayesiana

1.1.2.2. Herramientas de Minería de Datos

Existen algunas herramientas diseñadas para extraer conocimientos desde bases de datos que contienen grandes cantidades de información. Las más populares de estas herramientas son SPSS Clementine, Oracle Data Miner y Weka. (RCCI, 2009)

Clementine de SPSS: Clementine se centra en la integración de data mining con otros procesos y sistemas de negocio que ayuden a entregar inteligencia predictiva en un tiempo eficiente durante las operaciones de negocio diarias. La funcionalidad abierta de data mining en bases de datos que posee Clementine permite que muchos de los procesos de data mining se realicen en entornos que mejoran tanto el rendimiento como el despliegue de los resultados de data mining. La última versión de Clementine extiende la funcionalidad de data mining al incluir un conjunto de reglas de scoring y modelos de árboles de decisión y carga de resultados de data mining en la base de datos. Sistema integrado de minería de datos que permite encontrar patrones en la información para facilitar la toma de decisiones a los usuarios. (Pardo, y otros, 2002)

Esta herramienta permite seleccionar campos o filtrar los datos, permite mostrar propiedades de los datos, encontrar relaciones, ambiente integrado de minería de datos para usuarios finales y desarrolladores, algoritmos múltiples de minería de datos y herramientas de visualización. La compañía es SPSS/Integral Solutions Limited (ISL) y funciona sobre todas las plataformas hardware y sistemas operativos, incluyendo Unix, VMS y Windows NT. (Pardo, y otros, 2002)

YALE: Es una herramienta bastante flexible creada en la universidad de Dortmund para el descubrimiento del conocimiento y la minería de datos. Puesto que YALE está escrito enteramente en Java (Todo programación, 2005), funciona en las plataformas o sistemas operativos más conocidos. Es un software de código abierto GNU (St.Amant, y otros) y con licencia GPL. Recientemente fue lanzada la última versión, la cual incluye características como las de implicar nuevos formatos de entrada de datos con operadores para Microsoft Excel y SPSS. Desde la perspectiva de la visualización, YALE ofrece representaciones de datos en dispersión en 2D y 3D, representaciones de datos en formato SOM (Self Organizing

Map), coordenadas paralelas y grandes posibilidades de transformar las visualizaciones de los datos. (Todo programación, 2005)

WEKA: Es de libre distribución (licencia GPL) y destacada por la cantidad de algoritmos que presenta, así como, por la eficiencia de los mismos y por los generadores de reglas (Reutemann, y otros, 2004). Está desarrollada por miembros de la Universidad de Waikato y proporciona gran cantidad de herramientas para la realización de tareas propias de minería de datos, la visualización y permite la programación en JAVA de algoritmos más sofisticados para análisis de datos y modelado predicativo, unidos a una interfaz gráfica de usuario para acceder fácilmente a sus funcionalidades. En WEKA, se implementan las técnicas de clasificación, asociación, agrupamiento y predicciones existentes en la actualidad. Su sistema operativo es multiplataforma. (Witten, y otros, 1999)

SAS Enterprise Miner: Su compañía es SAS, es una solución de minería de datos que permite incorporar patrones inteligentes a los procesos de marketing, tanto operativos como estratégicos. (Rocha, 2015). El software de SAS, es un sistema de entrega de información que provee acceso transparente a cualquier fuente de datos, incluyendo archivos planos, archivos jerárquicos, y los más importantes manejadores de bases de datos relacionales. También incluye su propia base de datos de información para almacenar y manejar los datos, es decir, un "data warehouse". Soporta los principales protocolos de comunicación, cubre los cinco modelos de procesamiento cliente/servidor de acuerdo a Gartner Group y cumple con las 12 reglas de OLAP (Codd, 1996). El sistema soporta un amplio rango de aplicaciones, destacándose el análisis estadístico, análisis gráfico de datos, análisis de datos guiado, mejoramiento de la calidad, diseño experimental, administración de proyectos, programación lineal y no lineal, generación de reportes y gráficas, manipulación y despliegue de imágenes, sistemas de información geográfica, visualización multidimensional de datos, aplicaciones de multimedia, así como los sistemas de información ejecutiva. (Rocha, 2015)

1.1.3. Inteligencia Artificial

La Inteligencia Artificial (IA) (García Serrano, 2012) es la habilidad de los ordenadores para hacer actividades que normalmente requieren inteligencia humana. Pero, para brindar una definición más detallada, podríamos decir que la IA es la capacidad de las máquinas para usar algoritmos, aprender de los datos y utilizar lo aprendido en la toma de decisiones, tal, y como lo haría un ser humano. Sin embargo, a diferencia de las personas, los dispositivos basados en IA no necesitan descansar y pueden analizar grandes volúmenes de información a la vez. Asimismo, la proporción de errores es significativamente menor en las máquinas que realizan las mismas tareas que sus contrapartes humanas.

La idea de que los ordenadores o los programas informáticos, puedan, tanto aprender como tomar decisiones es particularmente importante y algo sobre lo que deberíamos ser conscientes ya que sus procesos están creciendo exponencialmente con el tiempo. Debido a estas dos capacidades, los sistemas de inteligencia artificial pueden realizar ahora muchas de las tareas que antes estaban reservadas sólo a los humanos.

Las tecnologías basadas en la IA ya están siendo utilizadas para ayudar a los humanos a beneficiarse de mejoras significativas y disfrutar de una mayor eficiencia en casi todos los ámbitos de la vida. Pero el gran crecimiento de la IA también nos obliga a estar atentos para prevenir y analizar las posibles desventajas directas o indirectas que pueda generar la proliferación de la IA (Russell, y otros, 2004).

1.1.4. Aprendizaje automático

El aprendizaje automático (en inglés, machine learning) es uno de los enfoques principales de la inteligencia artificial (Béjar, 2007). En pocas palabras, se trata de un aspecto de la informática en el que los ordenadores o las máquinas tienen la capacidad de aprender sin estar programados para ello. Un resultado típico serían las sugerencias o predicciones en una situación particular.

Los primeros ordenadores personales, que estuvieron disponibles para los consumidores a partir de la década de 1980, fueron programados explícitamente para realizar ciertas acciones. Por el contrario, gracias al aprendizaje automático, muchos de los dispositivos que verás en el futuro obtendrán experiencia y conocimientos a partir de la forma en que son utilizados para poder ofrecer una experiencia personalizada al usuario. Ejemplos de ello en la actualidad son la personalización de los sitios de medios sociales como Facebook o los resultados del motor de búsqueda de Google. (Béjar, 2007)

Tipos de Aprendizaje Automático

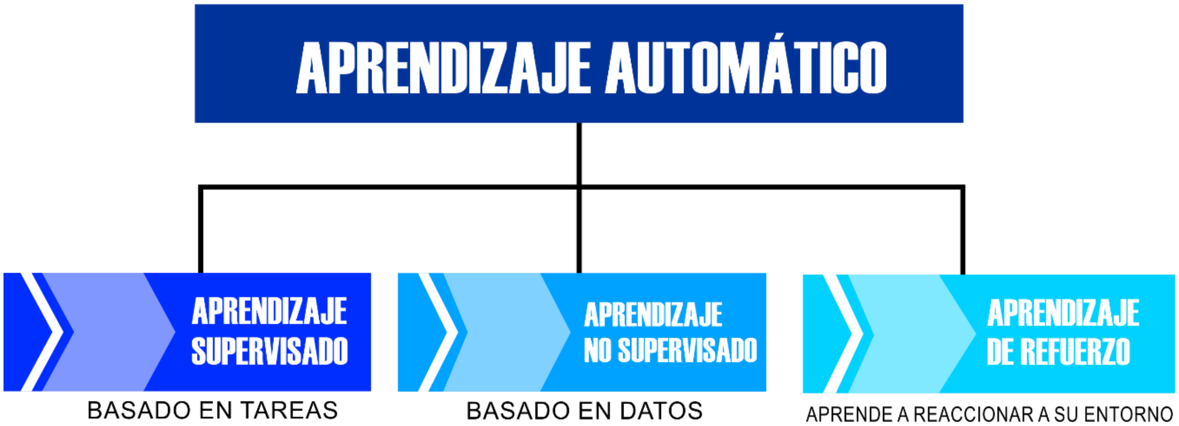


Figura Nº 2: Tipos de Aprendizaje Automático **Fuente:** Elaboración propia

En el **aprendizaje supervisado**, los algoritmos usan datos que ya han sido etiquetados u organizados previamente para indicar cómo tendría que ser categorizada la nueva información. Con este método, se requiere la intervención humana para proporcionar retroalimentación. (Béjar, 2007)

En el **aprendizaje no supervisado**, los algoritmos no usan ningún dato etiquetado u organizado previamente para indicar cómo tendría que ser categorizada la nueva información, sino que tienen que encontrar la manera de clasificarlas ellos mismos. Por tanto, este método no requiere la intervención humana. (Béjar, 2007)

En el **aprendizaje por refuerzo**, los algoritmos aprenden de la experiencia. En otras palabras, tenemos que darles un refuerzo positivo cada vez que aciertan. La forma en que estos algoritmos aprenden se puede comparar, por ejemplo, con la de los perros cuando les damos «recompensas» al aprender a sentarse. (Técnicas de reducción de datos en KDD. El uso de Algoritmos Evolutivos para la Selección de las Instancias., 2006)

1.1.5. Aprendizaje profundo

Una de las aplicaciones más poderosas y de mayor crecimiento de la inteligencia artificial es el aprendizaje profundo (en inglés, deep learning). Se trata de un subcampo del aprendizaje automático que se utiliza para resolver problemas muy complejos y que, normalmente, implican grandes cantidades de datos.

El aprendizaje profundo se produce mediante el uso de redes neuronales, que se organizan en capas para reconocer relaciones y patrones complejos en los datos. Su aplicación requiere un enorme conjunto de información y una potente capacidad de procesamiento. Actualmente, se utiliza en el reconocimiento de voz, el procesamiento del lenguaje natural, la visión artificial y la identificación de vehículos en los sistemas de asistencia al conductor (Rouhianen, 2018).

1.2. Tipos de Datos

El análisis de datos se basa generalmente en dos tipos de información: datos estructurados y datos no estructurados. Para comprender realmente los sistemas de inteligencia artificial, es importante reconocer las diferencias entre estos dos tipos de datos.

Tradicionalmente, los datos estructurados se han utilizado con más frecuencia que los no estructurados. Los primeros incluyen la introducción de información, como valores numéricos, fechas, monedas o direcciones; los segundos contienen tipos de datos que son más complicados de analizar como textos, imágenes y vídeos. Sin embargo, el desarrollo de la inteligencia artificial ha hecho posible examinar mayor número de datos no estructurados y los resultados pueden utilizarse para hacer recomendaciones y predicciones. (Rouhianen, 2018)

Los datos estructurados se almacenan en una base de datos relacional (RDBMS) mientras que los datos no estructurados no pueden almacenarse en estructuras de datos relacionales predefinidas (NoSQL). (Calvo, 2017)

Los datos estructurados al poseer, como su propio nombre indica, una estructura organizada que otorga al usuario de facilidad de análisis para la obtención de resultados medibles. Por el contrario, los datos no estructurados necesitan herramientas analíticas más complejas. (Calvo, 2017)

Los datos no estructurados son más flexibles, es decir, mucho menos sensibles a los cambios que los datos estructurados. Al almacenar toda la información en bruto, permite el acceso de cualquier usuario para configurar y reconfigurar según la finalidad para la que hayan sido concebidos. (Calvo, 2017)

Figura Nº 3: Tipos de Datos Estructurados **Fuente:** (Calvo, 2017)

1.2.1. Herramientas a utilizar. Lenguajes de Programación

1.2.1.1. Python

Es un lenguaje interpretado, orientado a objetos, de alto nivel, que permite escribir códigos con una alta claridad y legibilidad, permitiendo así un rápido aprendizaje del mismo. La legibilidad, permitirá en futuros accesos al código, una clara comprensión de lo antes implementado, haciendo más fácil el mantenimiento de las aplicaciones. Su biblioteca estándar es muy amplia, conteniendo funcionalidades de gran ayuda desde el más bajo nivel hasta el más alto, facilitándole al programador la implementación de aplicaciones sin la necesidad de recurrir continuamente a bibliotecas externas. Además, dispone de una extensa colección de bibliotecas libres disponibles en la mayoría de los repositorios de los sistemas GNU\Linux (Knowlton, 2009).

Este lenguaje permite ser extendido, haciéndolo mucho más favorable para su uso, pues en caso de existir alguna región crítica del proyecto que al ser escrita en Python no sea lo más recomendable u óptimo, ésta puede ser implementada en C/C++ y compilada de modo que sea accesible desde el intérprete Python, aumentando considerablemente el rendimiento de esa sección a programar (Martelli, 2007).

Interfaz gráfica de usuario

PyQt5: Qt para Python

Las bibliotecas gráficas Qt, constituyen actualmente una de las mejores en su tipo en el mundo del software libre. Realizadas en el lenguaje de programación C++, están hechas con el fin de realizar más, con menos código. Su carácter multiplataforma hace fácil su despliegue. PyQt es la combinación de Python y Qt, estableciendo una interfaz transparente para acceder desde Python a dichas bibliotecas gráficas; así, con la facilidad de Python sumada a la excelencia de las Qt, se hace más agradable la programación visual. Cuenta con una amplia documentación proveniente de Qt y de Python, a la vez. Con la biblioteca PyQt se logró una interfaz visual sencilla y sin muchos contratiempos al brindar una amplia variedad de componentes visuales, así como su abundante documentación y ejemplos. Al funcionar en varias plataformas y tener una alta calidad hacen de la biblioteca uno de los productos con más alto uso en la actualidad (Martelli, 2007).

Qt Designer

Qt Designer es la herramienta para el diseño gráfico del FrameworkQt por excelencia. Se creó con el mismo fin para el que fue confeccionado Qt: agilizar el desarrollo y funcionar en una amplia variedad de plataformas. Qt Designer está disponible para las mismas plataformas sobre las que está sustentado el framework para el que confecciona las interfaces. Con la ayuda de una herramienta perteneciente a la mencionada biblioteca PyQt, se puede crear el código Python correspondiente a las interfaces confeccionadas en Qt Designer. La herramienta Qt Designer fue muy útil en el desarrollo de las interfaces visuales de la herramienta, brindando de forma cómoda y sencilla la creación de la misma con la variedad de componentes visuales que posee la biblioteca, así como la manipulación de las variables de configuración de cada uno de ellos (A Brief History of Qt. C++ GUI Programming with Qt 4, 2006).

1.2.1.2. Entornos de Desarrollo Integrado (IDE)

Una de las herramientas que juegan un papel importante en el desarrollo de soluciones informáticas son los Entornos de Desarrollo Integrado (IDE). Estos ofrecen facilidades al equipo de desarrollo cuando se implementan las aplicaciones debido a que permite corrección de errores comunes que se comenten a diario (Brenner, 2016), (Ciementson, 2017).

PyCharm

PyCharm es un IDE o entorno de desarrollo integrado multiplataforma utilizado para desarrollar en el lenguaje de programación Python. Proporciona análisis de código, depuración gráfica, integración con VCS / DVCS y soporte para el desarrollo web con Django. PyCharm es desarrollado por la empresa JetBrains y debido a la naturaleza de sus licencias tiene dos versiones, la Community que es gratuita y orientada a la educación y al desarrollo puro en Python y la Professional, que incluye más características como el soporte a desarrollo web con varios precios (Romeu, 2009).

Ventajas de PyCharm

Trabajar con PyCharm tiene ventajas básicas, similares a las ofrecidas por otros IDE, pero también algunas específicas a las cuales debe su popularidad. Es así que PyCharm tiene un editor inteligente, que permite completar código con algunos atajos de teclado. Asimismo, permite navegar a través de nuestro código, saltando entre las clases y métodos creados, haciendo el flujo de trabajo mucho más dinámico. (Romeu, 2009)

Una de las características notables de PyCharm es la posibilidad que tiene de refactorizar el código, que, en términos generales, significa modificar el código sin comprometer la ejecución del mismo. (Romeu, 2009)

Esta operación se realiza de forma constante dentro de la Ingeniería de Software y es más conocida como limpiar el código para que este pueda ser interpretado con facilidad cuando hay distintas personas integrando un equipo de trabajo. (Romeu, 2009)

Por último, la gran cantidad de desarrolladores que trabajan con PyCharm ha generado que se tenga una gran cantidad de temas y plugins que se pueden usar para trabajar más cómodamente (Romeu, 2009).

Es de vital importancia la presencia de estas características para una programación dinámica y rápida para la elaboración de esta herramienta. El uso de plugins y el acceso a librerías de manera asequible y cómoda como ofrece PyCharm es una ventaja que otros IDEs no portan.

1.3. Origen de las Metodologías Ágiles de desarrollo de software

De acuerdo a (Letelier, y otros, 2006) en un proceso de software existen numerosas propuestas metodológicas que inciden en distintas dimensiones del transcurso de desarrollo. Por una parte, tenemos aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, las herramientas y notaciones que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos, pero también han presentado problemas en otros. Una posible mejora es centrarse en otras dimensiones, como, por ejemplo, el factor humano o el producto software. Esta es la filosofía de las metodologías ágiles, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas.

1.3.1. Principales Metodologías Ágiles

El punto de partida fue el Manifiesto ágil, un documento que resume la filosofía ágil. Este manifiesto ágil comienza enumerando los principales valores del desarrollo ágil, como son:

- Al Individuo y las iteraciones del equipo de desarrollo sobre el proceso y las herramientas.
- Desarrollar software que funciona más que conseguir una buena documentación.
- La colaboración con el cliente más que la negociación de un contrato.
- Responder a los cambios más que seguir estrictamente un plan.

Las Metodologías Ágiles resuelven los problemas surgidos, posteriormente, a la masificación del uso del computador personal, dado que las expectativas y necesidades por parte de los usuarios se hicieron más urgentes y frecuentes (Orjuela, y otros, 2008).

Fue así como al comienzo de los años 90 surgieron propuestas metodológicas para lograr resultados más rápidos en el desarrollo del software sin disminuir su calidad (Orjuela, y otros, 2008).

Entre las principales metodologías ágiles se encuentran las siguientes:

1.3.1.1. XP (Extreme Programming)

La programación extrema o Extreme Programming (XP) es un enfoque de la Ingeniería de Software formulado por Kent (Beck, 2004). Es el más destacado de los procesos ágiles de desarrollo de software.

Al igual que éstos, la programación extrema se diferencia de las metodologías tradicionales, principalmente, pues hace más énfasis en la adaptabilidad que en la previsibilidad. Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del Desarrollo de Proyectos (Letelier, y otros, 2006).

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre. Kent Beck, el padre de XP, describe su filosofía sin cubrir los detalles técnicos y de implantación de las prácticas. Posteriormente, otras publicaciones de experiencias se han encargado de dicha tarea (Wells, 2020).

1.3.1.2. AUP

Es una versión simplificada de Rational Unified Process (RUP). Este define un flujo de trabajo con disciplinas diferentes a las de RUP, aunque conserva sus fases en cada una. La disciplina de modelación incluye la modelación del negocio, requisitos, análisis y diseño. Por otra parte, se integran además la Gestión de Cambios y Gestión de Configuración en una sola disciplina. La metodología está compuesta por cuatro disciplinas que sirven de apoyo a la ingeniería (Modelación, Implementación, Prueba y Despliegue) y cuenta con tres destinadas al soporte

(Gestión de configuración, Gestión de Proyecto y Ambiente). Estas son ejecutadas y guiadas de forma iterativa (Ambler, 2019).

1.3.1.3. SCRUM

Según (Orjuela, y otros, 2008), SCRUM está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo del software se realiza mediante iteraciones denominadas Sprint, con una duración de 30 días, el resultado de cada Sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo del proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración.

1.3.1.4. CRYSTAL METODOLOGÍAS

Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por encontrarse centradas en las personas que componen equipo y la reducción al máximo del número de artefactos producidos. El desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas definidas de trabajo en equipo. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo: Crystal Clear (3 a 8 integrantes), Crystal Orange (25 a 50 integrantes). (Orjuela, y otros, 2008).

1.3.1.5. ADAPTIVE SOFWARE DEVELOPMENT (ASD)

Presupone que las necesidades del cliente son cambiantes. La iniciación de un proyecto involucra definir una misión para él, determinar las características, las fechas y descomponer el proyecto en una serie de pasos individuales, cada uno de los cuales puede abarcar entre cuatro y ocho semanas. Los pasos iniciales deben verificar el alcance del proyecto, los tardíos tienen que ver con el diseño de la arquitectura, la construcción del código, la ejecución de las pruebas finales y el despliegue (Calderón, y otros, 2007).

1.3.1.6. DYNAMIC SYSTEMS DEVELOPMENT METHOD

Es la única de las metodologías planteadas, surgida de un consorcio formado originalmente por 17 miembros fundadores en enero de 1994. El objetivo del consorcio era producir una metodología de dominio público que fuera independiente de las herramientas y que pudiera ser utilizada en proyectos de tipo RAD (Rapid Application Development). En DSDM se definen cinco fases en la construcción de un sistema, las mismas son: Estudio de factibilidad, Estudio del Negocio, Iteración del modelo funcional, Iteración del diseño y Construcción, Implantación. La Adecuación de DSDM para desarrollo rápido está suficientemente aprobada y se ha aplicado a proyectos grandes y pequeños. (Calderón, y otros, 2007)

1.3.1.7. AUP-UCI

Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable. Se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI (Rodríguez, 2014).

Con la adaptación de AUP que se propone para la actividad productiva de la UCI:

- Se logra estandarizar el proceso de desarrollo de software.
- Se logra hablar un lenguaje común en cuanto a fases, disciplinas, roles y productos de trabajos.
- Se redujo a 1 la cantidad de metodologías que se usaban y de más de 20 roles en total que se definían se redujeron a 11.

De igual forma se definen por parte de la metodología 4 escenarios para la disciplina de Requisitos, los cuales son:

- **Escenario Nº 1:** Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que puedan modelar una serie de interacciones entre los trabajadores del negocio/actores del sistema (usuario), similar a una llamada y respuesta respectivamente, donde la atención se centra en cómo el usuario va a utilizar el sistema. Es necesario que se tenga claro por el proyecto que los casos de uso del negocio muestran como los procesos son llevados a cabo por personas y los activos de la organización.

- **Escenario Nº 2:** Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que no es necesario incluir las responsabilidades de las personas que ejecutan las actividades, de esta forma modelarían exclusivamente los conceptos fundamentales del negocio. Se recomienda este escenario para proyectos donde el objetivo primario es la gestión y presentación de información.
- **Escenario Nº 3:** Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio con procesos muy complejos, independientes de las personas que los manejan y ejecutan, proporcionando objetividad, solidez, y su continuidad. Se debe tener presente que este escenario es muy conveniente si se desea representar una gran cantidad de niveles de detalles y las relaciones entre los procesos identificados.
- **Escenario Nº 4:** Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Se recomienda en proyectos no muy extensos, ya que una historia de usuario (HU) no debe poseer demasiada información.

1.3.2. Metodología a Utilizar

Se utilizará para la elaboración de esta herramienta una metodología ágil, el autor de esta tesis y de la herramienta creada posee una amplia experiencia en el campo de la Inteligencia Artificial y del proceso KDD, con dos años de vinculación a la Línea de Desarrollo Inteligencia Artificial de la Universidad de Ciencias Informáticas. En adición, el tutor Héctor posee un amplio recorrido trabajando en esa Línea de Investigación y una gran experiencia de trabajo en el lenguaje de programación Python.

Para guiar el desarrollo de la solución se decide utilizar XP como metodología de desarrollo en donde se utilizarán las Historias de Usuarios confeccionadas a partir de los requisitos funcionales. En este escenario el cliente acompaña al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Además, se recomienda en proyectos no muy extensos, ya que una HU no debe poseer mucha información.

1.3.3. Metodología Programación Extrema (XP)

1.3.3.1. Características

- Metodología basada en prueba y error para obtener un software que funcione realmente.
- Fundamentada en principios.
- Está orientada hacia quien produce y usa software (el cliente participa muy activamente).
- Reduce el coste del cambio en todas las etapas del ciclo de vida de la herramienta.
- Combina las que han demostrado ser las mejores prácticas para desarrollar software y las lleva al extremo.
- Cliente bien definido.
- Los requisitos pueden cambiar.
- Grupo pequeño y muy integrado (2-12 personas).
- Equipo con formación elevada y capacidad de aprender

1.3.3.2. Artefactos XP

A continuación, los artefactos que utiliza la metodología XP durante sus fases de desarrollo.

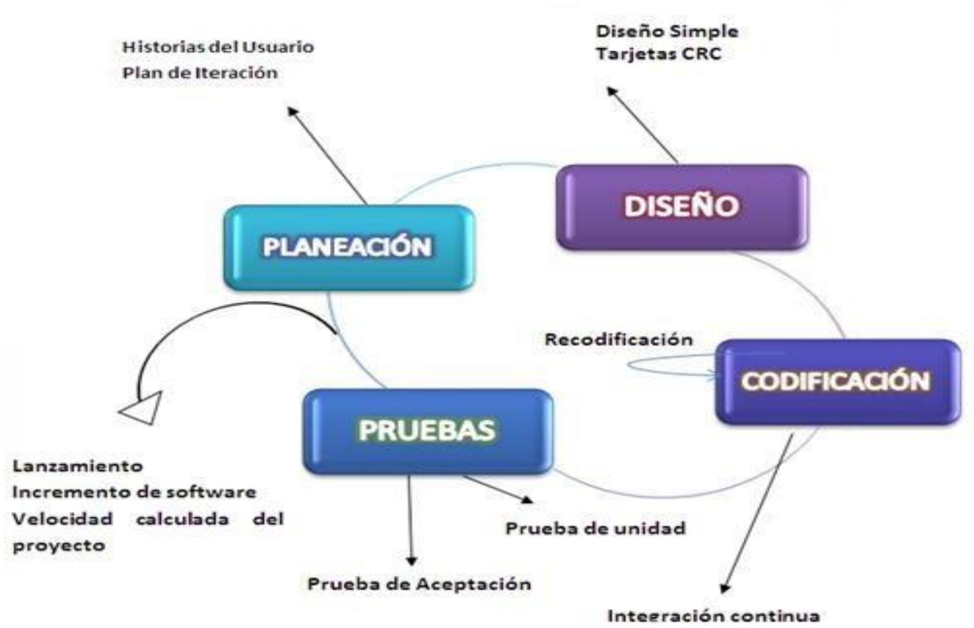


Figura Nº 4: Artefactos XP durante sus Fases de Desarrollo **Fuente:** (Letelier, y otros, 2006)

Historias de usuario

Las Historias de Usuario representan una breve descripción del comportamiento de la herramienta, se realizan por cada característica principal de la misma y son utilizadas para cumplir estimaciones de tiempo y el plan de lanzamientos, así mismo reemplaza un gran documento de requisitos, y presiden la creación de las pruebas de aceptación (Letelier, y otros, 2006).

Cada historia de usuario debe ser lo suficientemente comprensible y delimitada para que los programadores puedan implementarlas en unas semanas (Letelier, y otros, 2006).

HISTORIA DE USUARIO	
Número: Permite identificar a una historia de usuario.	**Usuario:** Persona que utilizará la funcionalidad de la herramienta descrita en la historia de usuario.
Nombre Historia: Describe de manera general a una historia de usuario.	
Prioridad en Negocio: Grado de importancia que el cliente asigna a una historia de usuario.	**Riesgo en Desarrollo:** Valor de complejidad que una historia de usuario representa al equipo de desarrollo.
Puntos Estimados: Número de semanas que se necesitará para el desarrollo de una historia de usuario.	**Iteración Asignada:** Número de iteración, en que el cliente desea que se implemente una historia de usuario.
Programador Responsable: Persona encargada de programar cada historia de usuario.	
Descripción: Información detallada de una historia de usuario.	
Observaciones: Campo opcional utilizado para aclarar, si es necesario, el requerimiento descrito de una historia de usuario.	

Tabla Nº 1: Historia del Usuario. **Fuente:** (Letelier, y otros, 2006)

Tarjetas CRC (Clase-Responsabilidades-Colaboradores)

Las Tarjetas CRC (Clase-Responsabilidades-Colaboradores), permiten conocer que clases componen la herramienta y cuáles interactúan entre sí. Se dividen en tres secciones: Nombre de la Clase, Responsabilidades y Colaboradores. **Fuente:** (Loarte, y otros, 2014)

TARJETAS CRC	
Nombre de la Clase: Nombre de la clase al cual hace referencia la tarjeta.	
Responsabilidades: Atributos y operaciones de la clase.	**Colaboradores:** Clases que colaboran con la clase citada en la tarjeta.

Tabla Nº 2: Tarjetas CRC**. Fuente:** (Loarte, y otros, 2014)

Tareas de ingeniería (TASK CARD)

Una Historias de Usuario se descompone en varias tareas de ingeniería, las cuales describen las actividades que se realizarán en cada historia de usuario, asimismo, las tareas de ingeniería se vinculan más al desarrollador, ya que permite tener un acercamiento con el código (Ferreira, 2013).

TAREAS DE INGENIERÍA	
Número de Tarea: Permite identificar a una tarea de ingeniería.	**Número de Historia:** Número asignado de la historia correspondiente.
Nombre de Tarea: Describe de manera general a una tarea de ingeniería.	
Tipo de Tarea: Tipo al que corresponde la tarea de ingeniería.	**Puntos Estimados:** Número de días que se necesitará para el desarrollo de una tarea de ingeniería.
Fecha Inicio: Fecha inicial de la creación de la tarea de ingeniería.	**Fecha Fin:** Fecha final de la tarea de ingeniería.
Programador Responsable: Persona encargada de programar la tarea de ingeniería.	
Descripción: Información detallada de la tarea de ingeniería.	

Tabla Nº 3: Tarea de ingeniería. **Fuente:** (Ferreira, 2013).

Pruebas de aceptación

Según (Loarte, y otros, 2014) las “Pruebas de aceptación” son de vital importancia para el éxito de una iteración y el comienzo de la siguiente, con lo cual el cliente puede conocer el avance en el desarrollo de la herramienta, y a los programadores, lo que les resta por hacer. Además, permite una retroalimentación para el desarrollo de las próximas historias de usuarios a ser entregadas. Estas, son comúnmente llamadas pruebas del cliente, por lo que son realizadas por el encargado de verificar si las historias de usuarios de cada iteración cumplen con la funcionalidad esperada.

PRUEBAS DE ACEPTACIÓN	
Código: Nº Único, permite identificar la prueba de aceptación.	**Nº Historia de Usuario:** Número único que identifica a la historia de usuario.
Historia de Usuario: Nombre que indica de manera general la descripción de la historia de usuario.	
Condiciones de Ejecución: Condiciones previas que deben cumplirse para realizar la prueba de aceptación.	
Entrada/Pasos de Ejecución: Pasos que siguen los usuarios para probar la funcionalidad de la historia de usuario.	
Resultado Esperado: Respuesta de la herramienta que el cliente espera, después de haber ejecutado una funcionalidad	
Evaluación de la Prueba: Nivel de satisfacción del cliente sobre la respuesta de la herramienta. Los niveles son: Aprobada y No Aprobada.	

Tabla Nº 4: Pruebas de aceptación. **Fuente:** (Loarte, y otros, 2014)

1.3.3.3. Roles XP (Pressman, 2010)

El **programador** escribe las pruebas unitarias y produce el código de la herramienta. Debe existir una comunicación y coordinación adecuada entre los programadores y otros miembros del equipo.

El **cliente** escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio. El cliente es sólo uno dentro del proyecto, pero puede corresponder a un interlocutor que está representando a varias personas que se verán afectadas por la herramienta.

Encargado de pruebas (Tester): ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.

Encargado de seguimiento proporciona realimentación al equipo en el proceso XP. Su responsabilidad es verificar el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, comunicando los resultados para mejorar futuras estimaciones.

Entrenador (Coach): es responsable del proceso global. Es necesario que conozca a fondo el proceso XP para proveer guías a los miembros del equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.

Consultor: Es un Miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto. Guía al equipo para resolver un problema específico.

Gestor (Big Boss): Es el vínculo entre el cliente y programadores. Experto en tecnología y labores de gestión. Construye el plantel del equipo, obtiene los recursos necesarios y maneja los problemas que se generan. Administra a su vez las reuniones (planes de iteración, agenda de compromisos). Su labor fundamental es de coordinación.

1.3.3.4. Fases de la Programación Extrema (XP)

La Programación Extrema consta de 4 fases, las cuales son: planeación, diseño, codificación y pruebas (Pressman, 2010).

Planeación

La Metodología XP plantea la planificación como un diálogo continuo entre las partes involucradas en el proyecto, incluyendo al cliente, a los programadores y a los coordinadores. El proyecto comienza recopilando las historias de usuarios, las que constituyen a los tradicionales casos de uso. Una vez obtenidas estas historias de usuarios, los programadores evalúan rápidamente el tiempo de desarrollo de cada una.

Los Conceptos básicos de la planificación son:

Las Historias de Usuarios: las cuales son descritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que la herramienta debe realizar.

El Plan de Entregas (Release Plan): establece que las historias de usuarios serán agrupadas para conformar una entrega y el orden de las mismas. Este cronograma será el resultado de una reunión entre todos los actores del proyecto.

Plan de Iteraciones (Iteration Plan): las historias de usuarios seleccionadas para cada entrega son desarrolladas y probadas en un ciclo de iteración, de acuerdo al orden preestablecido.

Reuniones Diarias de Seguimiento (Stand – Up Meeting): el objetivo es mantener la comunicación entre el equipo y compartir problemas y soluciones.

Diseño

La Metodología XP hace especial énfasis en los diseños simples y claros (Pressman, 2010). Los conceptos más importantes de diseño en esta metodología son los siguientes:

Simplicidad: Un diseño simple se implementa más rápidamente que uno complejo. Por ello XP propone implementar el diseño más simple posible que funcione.

Soluciones "Spike": Cuando aparecen problemas técnicos o cuando es difícil de estimar el tiempo para implementar una historia de usuario, pueden utilizarse pequeños programas de prueba (llamados "Spike"), para explorar diferentes soluciones.

Recodificación ("Refactoring"): Consiste en escribir nuevamente parte del código de un programa, sin cambiar su funcionalidad, a los efectos de crearlo más simple, conciso y entendible. Las metodologías de XP sugieren recodificar cada vez que sea necesario.

Metáforas: XP sugiere utilizar este concepto como una manera sencilla de explicar el propósito del proyecto, así como guiar la estructura del mismo. Una buena metáfora debe ser fácil de comprender para el cliente y a su vez debe tener suficiente contenido como para que sirva de guía a la arquitectura del proyecto.

Codificación

Disponibilidad del Cliente: Uno de los requerimientos de XP es tener al cliente disponible durante todo el proyecto. No solamente como apoyo a los desarrolladores, sino formando parte del grupo. El involucramiento del cliente es fundamental para que pueda desarrollarse un proyecto con la metodología XP. Al comienzo del proyecto, él debe proporcionar las historias de usuarios. Pero, dado que estas historias son expresamente cortas y de "alto nivel", no contienen los detalles necesarios para realizar el desarrollo del código. Estos detalles deben ser proporcionados por el cliente y discutidos con los desarrolladores, durante la etapa de desarrollo. (Pressman, 2010)

Uso de Estándares: XP promueve la programación basada en estándares, de manera que sea fácilmente entendible por todo el equipo y que facilite la recodificación.

Programación Dirigida por las Pruebas ("Test-Driven Programming"): En las metodologías tradicionales, la fase de pruebas, incluyendo la definición de los test, es usualmente realizada sobre el final del proyecto o el final del desarrollo de cada módulo. La metodología XP propone un modelo inverso, primero se escriben los test que la herramienta debe pasar. Luego, el desarrollo debe ser el mínimo necesario para pasar las pruebas previamente definidas. Las pruebas a las que se refiere esta práctica, son las pruebas unitarias, realizadas por los desarrolladores. La definición de estos test al comienzo, condiciona o "dirige" el desarrollo.

Programación en Pares: XP propone que se desarrolle en pares de programadores, ambos trabajando juntos en un mismo ordenador. Si bien parece que esta práctica duplica el tiempo asignado al proyecto (y, por ende, los costos en recursos humanos), al trabajar en pares se minimizan los errores y se logran mejores diseños, compensando la inversión en horas. El producto obtenido es por lo general de mejor calidad que cuando el desarrollo se realiza por programadores individuales.

Integraciones Permanentes: Todos los desarrolladores necesitan trabajar siempre con la "última versión". Realizar cambios o mejoras sobre versiones antiguas causan graves problemas y retrasan al proyecto. Es por eso que XP promueve publicar lo antes posible las nuevas versiones, aunque no sean las últimas, siempre que estén libres de errores. Idealmente, todos los días deben existir nuevas versiones publicadas. Para evitar errores, solo una pareja de desarrolladores puede integrar su código a la vez.

Propiedad Colectiva del Código: En un proyecto XP, todo el equipo puede contribuir con nuevas ideas que apliquen a cualquier parte del proyecto. Asimismo, una pareja de programadores puede cambiar el código que sea necesario para corregir problemas, agregar funciones o recodificar.

Ritmo Sostenido: La Metodología XP indica que debe llevarse un ritmo sostenido de trabajo. El concepto que se desea establecer con esta práctica es planificar el trabajo de forma a mantener un ritmo constante y razonable, sin sobrecargar al equipo.

Pruebas

Pruebas Unitarias: Todos los módulos deben pasar las pruebas unitarias antes de ser liberados o publicados. Por otra parte, como se mencionó anteriormente, las pruebas deben ser definidas antes de realizar el código ("Test-Driven Programming"). Que todo código liberado pase correctamente las pruebas unitarias, es lo que habilita que funcione la propiedad colectiva del código. (Pressman, 2010)

Detección y Corrección de Errores: Cuando se encuentra un error ("Bug"), éste debe ser corregido inmediatamente, y se deben tener precauciones para que errores similares no vuelvan a ocurrir. Asimismo, se generan nuevas pruebas para verificar que el error haya sido resuelto.

Pruebas de Aceptación: Son creadas en base a las historias de usuarios, en cada ciclo de la iteración del desarrollo. El Cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada. Asimismo, en caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución. Una historia de usuario no se puede considerar terminada hasta que pase correctamente todas las pruebas de aceptación **Fuente:** (Joskowicz, 2008).

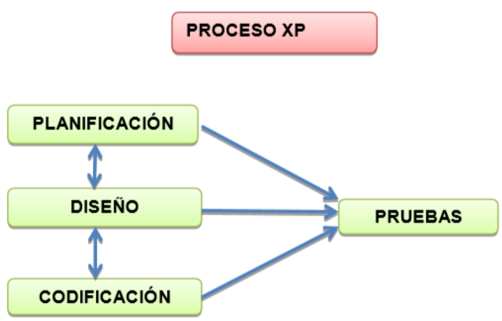


Figura Nº 5: Proceso XP**. Fuente:** (Joskowicz, 2008).

1.3.3.5. Prácticas de la metodología XP

En (Echeverry, y otros, 2007) La Metodología Extreme Programming o XP, está orientada al desarrollo de software cuando los requerimientos son ambiguos o rápidamente cambiantes asumiéndolos como algo natural, por lo que los programadores deben responder a estos cambios cuando el cliente lo solicite.

XP es para pequeños y medianos equipos basándose en la comunicación continúa entre todos los participantes, la simplicidad en las soluciones implementadas y coraje para enfrentar los cambios (Pressman, 2010).

Esta metodología recomienda las siguientes prácticas:

- Comunicación: Conversación continúa entre el equipo de desarrollo y el cliente, para implementar cambios lo antes posible.
- Entregas pequeñas: Entrega en versiones operativas.
- Diseño simple: Diseñar lo más posible, pero con la funcionalidad requerida.
- Pruebas: Se realizan pruebas unitarias por parte de los programadores y pruebas de aceptación por parte del cliente.
- Refactorización (refactoring): Remover código duplicado para facilitar los posteriores cambios.
- Programación en parejas: Se realiza para contar con menor tasa de errores, mejor diseño y mayor satisfacción de los programadores.
- Integración continúa: Cuando un fragmento de código esté listo, puede ser integrado a la herramienta.
- Cliente in-situ: El Cliente debe estar presente y disponible para el equipo de desarrollo.
- Estándares de programación: Normas definidas por los desarrolladores para tener un código legible.
- Juego de la planificación: Desde el comienzo del desarrollo se requiere que el grupo y el cliente tengan una visión general del proyecto. En el transcurso del mismo se realizan diferentes reuniones, con el fin de organizar las tareas e ideas que surgen tanto por parte del cliente como del equipo.
- Propiedad colectiva del código: El Código no es conocido por una sola persona del grupo del trabajo, esto facilita implementar cambios al programa por parte de otros integrantes del grupo.
- Utilización de metáforas de la herramienta: Para mejorar el entendimiento de los elementos de la misma por parte del equipo de desarrollo se acude a la utilización de metáforas, como una forma de universalizar el lenguaje de la herramienta.

- Test del cliente: El Cliente, con la ayuda de los desarrolladores, propone sus propias pruebas para validar las mini versiones.
- Horas por semana: Se debe de trabajar un máximo de 40 horas por semana.

Conclusiones parciales

A partir de la bibliografía consultada se determinó que la metodología ágil (XP) es la más factible para el desarrollo de una herramienta de configuración para el análisis y entrenamiento de grandes volúmenes de datos digitales, pues esta permite:

- Proceso más dinámico.
- Programación sumamente organizada.
- Ocasiona eficiencias en el proceso de planificación y pruebas.
- Su tasa de errores es pequeña.
- Fomenta la comunicación entre los clientes y los desarrolladores.
- Facilita los cambios.
- Puede ser aplicada a cualquier lenguaje de programación.
- El cliente tiene el control sobre las prioridades.
- Se hacen pruebas continuas durante el proyecto.
- Propicia la satisfacción del programador.
- Los artefactos a utilizar son más centrados en la elaboración del producto.

Se pudo determinar a partir de la construcción teórica estudiada, y en función de los objetivos de la presente investigación, que, el proceso de descubrimiento de conocimiento en bases de datos (KDD), es un proceso no trivial de identificar patrones válidos, novedosos, potencialmente útiles y comprensibles a partir de los datos.

La parte central del KDD, el análisis de datos, se suele denominar minería de datos y tiene sus raíces en las técnicas estadísticas multivariantes que se enriquecen de los métodos científicos computacionales para constituir un conjunto de técnicas predictivas o descriptivas capaces de extraer valor de los datos.

Las principales herramientas de análisis de datos son totalmente gratuitas y evolucionan continuamente. El lenguaje de programación Python cuenta con un conjunto excepcional de librerías de minería de datos. PyCharm, por su parte es uno de los Entorno de Desarrollo Integrado más completos para Python.

CAPÍTULO 2, “Planificación y diseño de la herramienta de configuración para el análisis y entrenamiento de grandes volúmenes de datos digitales”.

En este capítulo se exponen las bases metodológicas fundamentales para el levantamiento de información y a partir de ellas construir el modelo de dominio, identificar los requisitos funcionales y no funcionales, se redactan las historias de usuario y se plantea a través de la fase de diseño, el estilo arquitectónico, patrones de diseño, diagramas de clases, diagrama de componentes.

2.1. Modelo de Dominio

A continuación, el Modelo de Dominio de la herramienta, ejemplificando las clases a elaborar y el flujo de información entre las mismas.

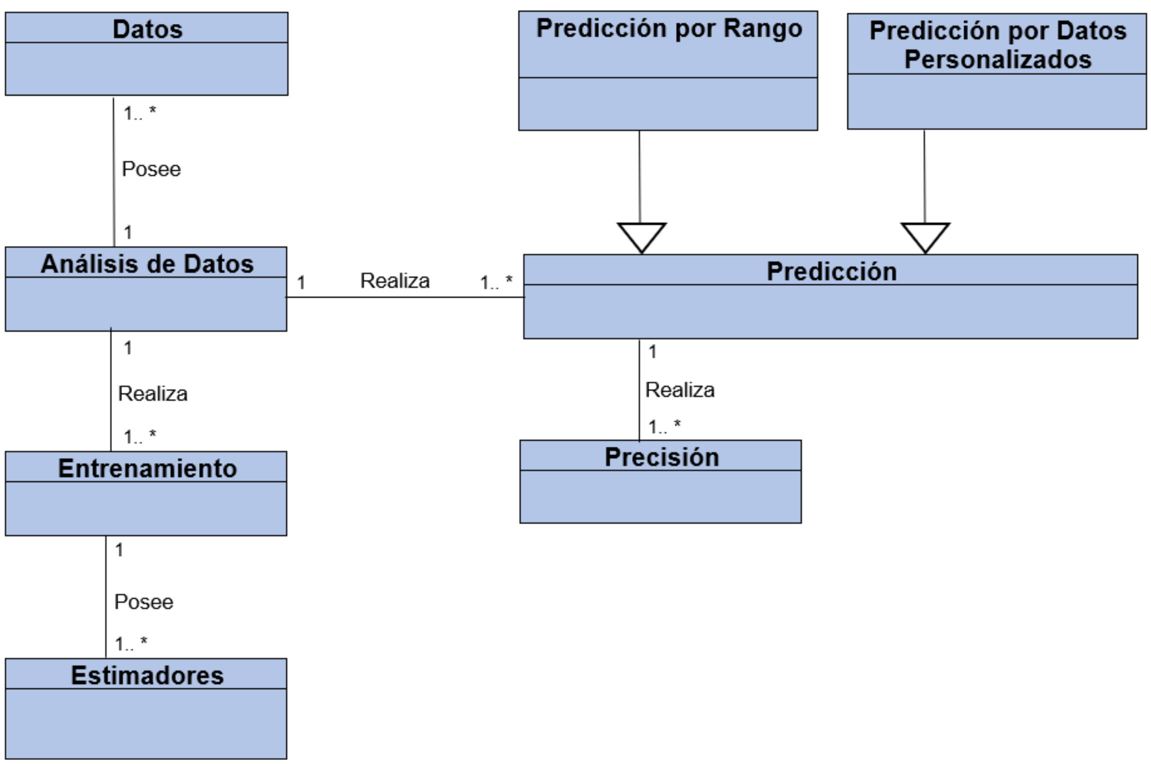


Figura Nº 6: Modelo de Dominio. **Fuente:** Elaboración propia

2.2. Captura de requisitos

La Ingeniería de Requisitos, es el proceso de desarrollar una especificación de Software. Las especificaciones pretenden comunicar las necesidades del cliente a los desarrolladores de la herramienta. Trata de los principios, métodos, técnicas y herramientas que permiten descubrir, documentar y mantener los requisitos para sistemas basados en computadora, de forma sistemática y repetible (McConnell, 1996) (Wiegers, 2003) (Stellman , y otros, 2005) (Landgraf, 2011).

Para la extracción de los requisitos se emplearon las herramientas y métodos existentes garantizando una correcta redacción y análisis de los mismos.

Luego de realizar entrevistas y posteriormente, la discusión con el cliente para determinar los requisitos de la herramienta en las posibles áreas donde se aplicará la misma, se lograron extraer los requisitos. Como resultado de la aplicación de estas técnicas se obtuvo la plantilla de modelos de historias de usuarios del negocio y la lista de reserva del producto, estos elementos son descritos en los siguientes epígrafes. Estos elementos que constituyen los requisitos de la herramienta a implementar fueron validados por el cliente, partiendo de que se describen estos artefactos de conjunto a los desarrolladores y el cliente. Luego se sometieron a tres rondas de revisiones lo cual permitió llegar a un consenso y claridad de las historias de usuario y sus descripciones.

2.2.1. Requisitos Funcionales

I. Seleccionar Dataset
II. Mostrar Descripción de Dataset
III. Mostrar Instancias
IV. Mostar Atributos
V. Seleccionar Instancias a Entrenar
VI. Entrenar Aleatoriamente
VII. Seleccionar Estimadores
VIII. Realizar Proceso de Entrenamiento
IX. Seleccionar Instancias a Predecir
X. Permitir la Predicción de Instancias Personalizadas

XI. Realizar Proceso de Predicción
XII. Mostrar Predicción
XIII. Mostrar Precisión

2.2.2. Requisitos No Funcionales

- **Usabilidad**

El software tendrá siempre la posibilidad de ayuda disponible para cualquier tipo de usuario, lo que le permitirá un avance considerable en la explotación de la herramienta en todas sus funcionalidades, para lo cual debe poseer una interfaz agradable para el cliente.

- **Fiabilidad**

La herramienta de implementación a utilizar tiene soporte para recuperación ante fallos y errores.

- **Seguridad**

Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos. La herramienta debe garantizar la confidencialidad, integridad y disponibilidad de la información que se procese en la herramienta.

- **Soporte**

Soporte para 1 Gb de almacenamiento y velocidad de procesamiento mayor o igual a 100 MHz.

- **Restricciones de diseño**

El lenguaje de programación es Python. La herramienta IDE de desarrollo será Pycharm para Python.

- **Interfaz**

La herramienta tiene que ofrecer una interfaz amigable, fácil de operar. Diseño sencillo, con pocas entradas, permitiendo que no sea necesario mucho entrenamiento para que los usuarios la puedan utilizar.

2.3. Planificación

Es la fase inicial de la metodología, donde se establece una comunicación continua entre el equipo de desarrollo y el cliente, para obtener principalmente los requisitos de la herramienta. Además, permite establecer el alcance del proyecto y fechas de entrega de la herramienta, teniendo en cuenta la prioridad y tiempo estimado para el desarrollo de cada historia de usuario.

Se quiere que la herramienta de configuración para análisis y entrenamiento de grandes volúmenes de datos digitales, mediante las encuestas, nos permita evaluar y analizar la información a tratar. Esto facilitará en enorme escala el proceso de recolección de datos y procesamiento de la información.

Para la entrega de este proyecto, esta herramienta contará con los siguientes módulos:

- Sesión
- Encuestas
- Administración
- Reporte
- Ayuda

Como resultado de la extracción de los requisitos se obtuvo la plantilla de modelos de historias de usuarios, estos elementos son descritos en los siguientes epígrafes. Estos elementos que constituyen los requisitos de la herramienta a implementar fueron validados por el cliente, partiendo de que se describen estos artefactos de conjunto a los desarrolladores y el cliente. Luego se sometieron a tres rondas de revisiones lo cual permitió llegar a un consenso y claridad de las historias de usuario y sus descripciones. Los módulos mencionados anteriormente, se recopilaron en base a reuniones y se definieron las siguientes **historias de usuario**:

I. Selección de Datasets
II. Muestra de datos de entrada y datos objetivos en forma de instancias.
III. Mostrar descripción del Dataset seleccionado
IV. Interfaz de entrenamiento

V. Opciones de entrenamiento de instancias por tipo de rango
VI. Opciones de entrenamiento aleatorio de instancias
VII. Selección de Estimadores
VIII. Interfaz de Predicción
IX. Opciones de predicción por tipo de rango
X. Opción de predicción de Datos Personalizados
XI. Iniciar el Proceso de Predicción
XII. Mostrar la Precisión de la Predicción
XIII. Volver a la Interfaz Anterior

2.4. Historias de Usuario

A continuación, en las tablas, se muestran las historias de usuario, las cuales fueron utilizadas para llevar a cabo el desarrollo de la herramienta:

HISTORIA DE USUARIO	
Número: 1	**Usuario:** Administrador, Usuarios
Nombre Historia: Selección de Datasets	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Media
Puntos Estimados: 2	**Iteración Asignada:** 1
Programador Responsable: Andy Armas	
Descripción: Al iniciarse la herramienta se podrá seleccionar entre diferentes tipos de Base de Datos (Datasets) para analizar, realizar futuros entrenamientos y predicciones.	
Observaciones: Es necesario seleccionar algunos de los Datasets presentes para proseguir con el análisis del mismo.	

Tabla Nº 5: Historia de Usuario. Selección de Datasets.

Fuente: Elaboración propia

HISTORIA DE USUARIO	
Número: 2	**Usuario:** Administrador, Usuarios
Nombre Historia: Muestra de datos de entrada y datos objetivos en forma de instancias.	
Prioridad en Negocio: Baja	**Riesgo en Desarrollo:** Baja
Puntos Estimados: 1	**Iteración Asignada:** 1
Programador Responsable: Andy Armas	
Descripción: Luego de seleccionar el Datasets, una opción permitirá mostrar los Datos del mismo (entrada y salida) por orden de instancias.	
Observaciones: Para ello es necesario seleccionar un Botón con el nombre de “Datos”	

Tabla Nº 6: Historia de Usuario. Muestra de datos de entrada y datos objetivos en forma de instancias.

Fuente: Elaboración propia

HISTORIA DE USUARIO	
Número: 3	**Usuario:** Administrador, Usuarios
Nombre Historia: Mostrar descripción del Dataset seleccionado	
Prioridad en Negocio: Baja	**Riesgo en Desarrollo:** Baja
Puntos Estimados: 1.5	**Iteración Asignada:** 1
Programador Responsable: Andy Armas	
Descripción: Luego de seleccionar el Datasets, una opción permitirá mostrar una breve descripción del mismo, mostrando a su vez: atributos, clases, rango de las mismas, entre otras informaciones características del mismo.	
Observaciones: Para ello es necesario seleccionar un Botón con el nombre de “Descripción”	

Tabla Nº 7: Historia de Usuario. Mostrar descripción del Dataset seleccionado.

Fuente: Elaboración propia

HISTORIA DE USUARIO	
Número: 4	**Usuario:** Administrador, Usuarios
Nombre Historia: Interfaz de entrenamiento	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Puntos Estimados: 3	**Iteración Asignada:** 2
Programador Responsable: Andy Armas	
Descripción: A través de un botón con la opción de “Entrenar”, se pasará a una ventana con informaciones clave de dicho Dataset, mostrando número de instancias, número de atributos y mostrando diferentes opciones de entrenamiento.	
Observaciones: Para ello es necesario seleccionar un Botón con el nombre de “Entrenar”	

Tabla Nº 8: Historia de Usuario. Interfaz de entrenamiento.

Fuente: Elaboración propia

HISTORIA DE USUARIO	
Número: 5	**Usuario:** Administrador, Usuarios
Nombre Historia: Opciones de entrenamiento de instancias por tipo de rango	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Alta
Puntos Estimados: 2	**Iteración Asignada:** 2
Programador Responsable: Andy Armas	
Descripción: Permitirá entrenar las instancias seleccionando el rango requerido, el cual, puede ser tanto por número de instancias o por un porcentaje con respecto al número total de instancias.	
Observaciones: Para ello es necesario seleccionar dentro de una caja de chequeo entre las opciones “Por Instancias” y “Por Porcentaje”, luego de esa acción se procederá a especificar el número de instancias o el porcentaje de las mismas respectivamente. El comienzo del entrenamiento no se procederá si no se tiene un Estimador seleccionado y una opción de entrenamiento de instancias por tipo de rango.	

Tabla No. 9: Historia de Usuario. Opciones de entrenamiento de instancias por tipo de rango.

Fuente: Elaboración propia

HISTORIA DE USUARIO	
Número: 6	**Usuario:** Administrador, Usuarios
Nombre Historia: Opciones de entrenamiento aleatorio de instancias	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Alta
Puntos Estimados: 2.5	**Iteración Asignada:** 2
Programador Responsable: Andy Armas	
Descripción: Permitirá, si se desea, entrenar las instancias seleccionadas de manera aleatoria proporcionando una mayor generalización de aprendizaje dentro del propio entrenamiento.	
Observaciones: Para ello es necesario seleccionar dentro de una caja de chequeo llamada “Entrenar Aleatoriamente” y dejarla marcada, en caso de no hacerlo se entrenará igualmente pero solo de manera sucesiva y por orden de las instancias.	

Tabla Nº 10: Historia de Usuario. Opciones de entrenamiento aleatorio de instancias.

Fuente: Elaboración propia

HISTORIA DE USUARIO	
Número: 7	**Usuario:** Administrador, Usuarios
Nombre Historia: Selección de Estimadores	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Puntos Estimados: 3	**Iteración Asignada:** 2
Programador Responsable: Andy Armas	
Descripción: Permitirá seleccionar diferentes tipos de Estimadores para el entrenamiento.	
Observaciones: Se permitirá seleccionar entre diferentes tipos de Estimadores el adecuado para el entrenamiento. El comienzo del entrenamiento no se procederá si no se tiene un Estimador seleccionado y una opción de entrenamiento de instancias por tipo de rango.	

Tabla Nº 11: Historia de Usuario. Selección de Estimadores.

Fuente: Elaboración propia

HISTORIA DE USUARIO	
Número: 8	**Usuario:** Administrador, Usuarios
Nombre Historia: Interfaz de Predicción	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Baja
Puntos Estimados: 2	**Iteración Asignada:** 3
Programador Responsable: Andy Armas	
Descripción: Luego de seleccionar el Estimador deseado y la opción de entrenamiento de instancias por tipo de rango, presionando un botón con el nombre de "Entrenar" se pasará a una nueva interfaz que mostrará diferentes tipos de opciones para la predicción.	
Observaciones: Para ello habrá que haber seleccionado el botón "Entrenar" de la interfaz anterior, para finalizar el entrenamiento y comenzar la predicción. El comienzo del entrenamiento no se procederá si no se tanto un Estimador seleccionado y una opción de entrenamiento de instancias por tipo de rango.	

Tabla Nº 12: Historia de Usuario. Interfaz de Predicción.

Fuente: Elaboración propia

HISTORIA DE USUARIO	
Número: 9	**Usuario:** Administrador, Usuarios
Nombre Historia: Opciones de predicción por tipo de rango	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Media
Puntos Estimados: 2.5	**Iteración Asignada:** 3
Programador Responsable: Andy Armas	
Descripción: A la hora de predecir se podrá elegir el intervalo de instancias a ser analizadas con el Entrenamiento efectuado. Dichas instancias a entrenar serán seleccionadas a través de las siguientes opciones: - Todas las Instancias - Resto de las Instancias Entrenadas - Rango de Instancias	
Observaciones: Para ello es necesario seleccionar dentro de una caja de chequeo entre las opciones: "Todas las Instancias", "Resto de las Instancias Entrenadas" y "Rango de Instancias", en el caso de esta última se debe especificar el comienzo y el final del rango por número de instancias. Para acceder al proceso de predicción será necesario seleccionar algunas de estas opciones y luego de esta acción presionar un botón llamado "Predicción"	

Tabla Nº 13: Historia de Usuario. Opciones de predicción por tipo de rango.

Fuente: Elaboración propia

HISTORIA DE USUARIO	
Número: 10	**Usuario:** Administrador, Usuarios
Nombre Historia: Opción de predicción de Datos Personalizados	
Prioridad en Negocio: Baja	**Riesgo en Desarrollo:** Media
Puntos Estimados: 1.5	**Iteración Asignada:** 3
Programador Responsable: Andy Armas	
Descripción: A la hora de predecir se podrá elegir si se desea insertar datos personalizados por el usuario, dichos datos serán analizados y se obtendrá una predicción por parte del Entrenamiento elaborado.	
Observaciones: Para ello es necesario seleccionar dentro de una caja de chequeo entre la opción "Predecir Datos Personalizados" en este caso se debe especificar insertando las instancias de entrada personalizadas a predecir. Para acceder al proceso de predicción será necesario seleccionar esta opción y luego de esta acción presionar un botón llamado "Predicción"	

Tabla Nº 14: Historia de Usuario. Opción de predicción de Datos Personalizados.

Fuente: Elaboración propia

HISTORIA DE USUARIO	
Número: 11	**Usuario:** Administrador, Usuarios
Nombre Historia: Iniciar el Proceso de Predicción	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Puntos Estimados: 4	**Iteración Asignada:** 3
Programador Responsable: Andy Armas	
Descripción: Inicia el Proceso de Predicción tomando los Dataset de entrada dentro del intervalos antes seleccionados y devolviendo los resultados de salida de dicha predicción.	
Observaciones: Para acceder al proceso de predicción será necesario seleccionar una de las opciones de tipo de predicciones y luego de esta acción presionar un botón llamado "Predicción"	

Tabla Nº 15: Historia de Usuario. Iniciar el Proceso de Predicción.

Fuente: Elaboración propia

HISTORIA DE USUARIO	
Número: 12	**Usuario:** Administrador, Usuarios
Nombre Historia: Mostrar la Precisión de la Predicción	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Media
Puntos Estimados: 3.5	**Iteración Asignada:** 3
Programador Responsable: Andy Armas	
Descripción: Muestra la Precisión, evaluada en porcentajes, entre los resultados de la Predicción objetivo contra el Dataset objetivo del intervalo de instancias seleccionadas anteriormente. Dicha comparación será mostrada en una barra de progreso donde se evidenciará el porcentaje de igualdad.	
Observaciones: Este proceso de precisión será mostrado solamente si se selecciona cualquier opción de predicción por tipo de rango excepto la opción de "Predecir Datos	

Personalizados", esto ocurrirá debido a la ausencia de datos objetivos de algún Dataset para comparar. Aun así se mostrará los datos objetivos evaluados en la Predicción.

Tabla Nº 16: Historia de Usuario. Mostrar la Precisión de la Predicción.

Fuente: Elaboración propia

HISTORIA DE USUARIO	
Número: 13	**Usuario:** Administrador, Usuarios
Nombre Historia: Volver a la Interfaz Anterior	
Prioridad en Negocio: Baja	**Riesgo en Desarrollo:** Baja
Puntos Estimados: 1	**Iteración Asignada:** 3
Programador Responsable: Andy Armas	
Descripción: Presionando el botón "Atrás" se podrá volver a la interfaz anterior para cambiar los datos a evaluar o el tipo de entrenamiento a efectuar, el botón aparecerá en todas las interfaces excepto en la primera: "Selección de Datasets". Este es el orden de las interfaces en la herramienta: "Selección de Datasets" -> "Descripción de Datasets" -> "Entrenamiento" -> "Predicción"	
Observaciones: Para acceder a la interfaz anterior solo es necesario presionar el botón "Atrás"	

Tabla Nº 17: Historia de Usuario. Volver a la Interfaz Anterior.

Fuente: Elaboración propia

2.4.1. Plan De Entrega Del Proyecto

Basándonos en las historias de usuario definidas para el desarrollo de la herramienta, se ha elaborado el siguiente plan de entrega, el cual muestra las historias de usuario que se llevarán a cabo en cada iteración. Para este plan de entrega se ha tomado en cuenta la prioridad y el esfuerzo de cada historia de usuario.

En la tabla siguiente se muestra el plan de entrega del proyecto:

Historias	Iteración	Prioridad	Esfuerzo	Fecha de Inicio	Fecha Final
Historia 1	1	Alta	2	10/10/2019	20/10/2019
Historia 2	1	Baja	1	22/10/2019	27/10/2019
Historia 3	1	Baja	1.5	30/10/2019	8/11/2019
Historia 4	2	Alta	3	10/11/2019	25/11/2019
Historia 5	2	Media	2	27/11/2019	9/12/2019
Historia 6	2	Media	2.5	11/12/2019	23/12/2019
Historia 7	2	Alta	3	24/12/2019	8/1/2020
Historia 8	3	Alta	2	10/1/2020	19/1/2020
Historia 9	3	Media	2.5	22/1/2020	4/2/2020
Historia 10	3	Baja	1.5	5/2/2020	12/2/2020

Historia 11	3	Alta	4	15/2/2020	7/3/2020
Historia 12	3	Media	3.5	9/3/2020	27/3/2020
Historia 13	3	Baja	1	30/3/2020	7/4/2020

Tabla Nº 18: Cronograma de entrega del proyecto.

Fuente: Elaboración propia

2.5. Diseño

El Diseño es la única manera de materializar con precisión los requerimientos del cliente, es un proceso y un modelado a la vez, con un conjunto de pasos repetitivos que permiten al diseñador describir todos los aspectos de la herramienta a construir. El diseño debe implementar todos los requisitos. Debe ser una guía que puedan leer y entender los que construyan el código y los que prueban y mantienen el software. El diseño debe proporcionar una completa idea de lo que es el software, enfocando los dominios de datos, funcional y comportamiento desde el punto de vista de la Implementación.

La metodología XP sugiere que hay que conseguir diseños simples y sencillos, fácilmente entendible e implementable que, a la larga, costará menos tiempo y esfuerzo desarrollar.

En la fase de Diseño se define el proceso de aplicar ciertas técnicas y principios con el propósito de definir una herramienta, con suficientes detalles como para permitir su interpretación y realización física. Este transforma elementos estructurales de la arquitectura del programa, la importancia del diseño del software se puede definir en una sola palabra: calidad; dentro del diseño es donde se fomenta la calidad del proyecto.

2.6. Tarjetas CRC

La metodología XP requiere, de muy poco, la representación de la herramienta mediante diagramas de clases utilizando la notación UML. En su lugar se usan las tarjetas CRC (Clase, Responsabilidad y Colaboración). Las cuales ayudan al equipo a definir actividades durante el diseño de la herramienta. Cada tarjeta representa una clase en la programación orientada a objetos. Las tarjetas CRC se componen de los siguientes elementos: (Pressman, 2010)

Clase: es cualquier persona, cosa, evento, concepto, pantalla o reporte.

Responsabilidades: es lo que debe hacer la clase, sus atributos y métodos.

Colaboradores: son el resto de las clases con las que interactúa para llevar a cabo sus responsabilidades.

Tarjeta CRC	
Clase: Main	
Responsabilidades	**Colaboraciones**
- Gestionar Interfaces - Configurar Eventos de Interfaces - Seleccionar Dataset - Mostrar Descripción de Dataset - Mostrar Instancias - Mostar Atributos - Seleccionar Instancias a Entrenar - Entrenar Aleatoriamente - Seleccionar Estimadores - Realizar Proceso de Entrenamiento - Seleccionar Instancias a Predecir - Permitir la Predicción de Instancias Personalizadas - Realizar Proceso de Predicción - Mostrar Predicción - Mostrar Precisión	- Select_Dataset - Dataset_Show - Train_Preparation - Classes_not_Enought - Prediction

Tabla Nº 19: Clase: Main

Fuente: Elaboración propia

Tarjeta CRC	
Clase: Select_Dataset	
Responsabilidades	**Colaboraciones**
- Crear Interfaz: Selección de Datos - Reubicar Componentes de Interfaz: Selección de Datos	- Main

Tabla Nº 20: Select_Dataset

Fuente: Elaboración propia

Tarjeta CRC	
Clase: Dataset_Show	
Responsabilidades	**Colaboraciones**
- Crear Interfaz: Muestra de Datos - Reubicar Componentes de Interfaz: Muestra de Datos	- Main

Tabla Nº 21: Dataset_Show

Fuente: Elaboración propia

Tarjeta CRC	
Clase: Train_Preparation	
Responsabilidades	**Colaboraciones**
- Crear Interfaz: Preparación de Entrenamiento - Reubicar Componentes de Interfaz: Preparación de Entrenamiento	- Main

Tabla Nº 22: Train_Preparation

Fuente: Elaboración propia

Tarjeta CRC	
Clase: Classes_not_Enought	
Responsabilidades	**Colaboraciones**
Crear Interfaz: Clases Insuficientes **Reubicar Componentes de Interfaz: Clases Insuficientes**	**Main**

Tabla Nº 23: Classes_not_Enought

Fuente: Elaboración propia

Tarjeta CRC	
Clase: Prediction	
Responsabilidades	**Colaboraciones**
Crear Interfaz: Predicción **Reubicar Componentes de Interfaz: Predicción**	**Main**

Tabla Nº 24: Prediction

Fuente: Elaboración propia

2.7. Estilo arquitectónico Modelo - Vista - Template (MVT)

Este patrón de diseño es muy semejante en teoría al MVC, se diferencia del anterior en que para el MVT las vistas hacen referencia a la lógica del negocio, es decir, esta capa es el puente entre Modelo y Template, es algo así como el controlador en el MVC (Aranguren, 2014).

El término diferente y que varía un poco a comparación del MVC es el Template, este componente lo que hace es tener un conjunto de planillas que se diseñan estandarizadas, a estos Templates se les encarga mostrar la información al usuario y capturar las interacciones de ellos con la herramienta (Aranguren, 2014).

Y finalmente el componente Modelo no varía del patrón MVC al patrón MVT, sigue siendo una capa de acceso a la base de datos, que maneja permisos, solicitudes, búsquedas, y demás.

Modelo: La librería "sklearn" contiene Datasets almacenados para aplicar durante su entrenamiento. La clase **Selección de Dataset**, es capaz de extraer los datos presentes de la librería, luego durante cualquier proceso de la herramienta, es capaz de modificar, eliminar e incluso crear, nuevos datos a entrenar. Este ejemplo se puede apreciar en la Figura Nº 7.

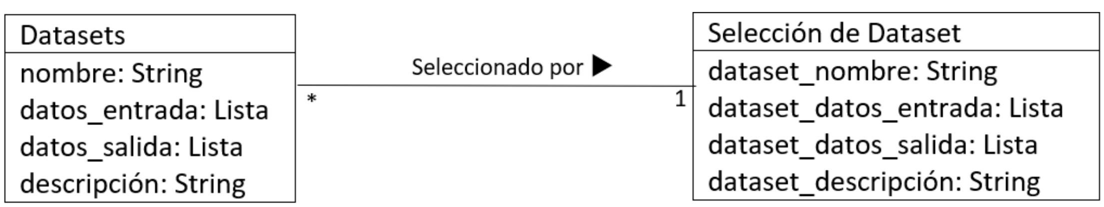


Figura Nº 7: Flujo de Información. **Fuente:** Elaboración propia

Vista: Las clases Muestra de Datos del Dataset y Muestra de Descripción del Dataset, muestran la información almacenada en la clase controladora Muestra de Dataset a través del uso de la librería **PyQt5.** Las variables dataset_datos_entrada, dataset_datos_salida y dataset_descripción serán mostrados respectivamente. Véase en la Figura **Nº** 8 este ejemplo.

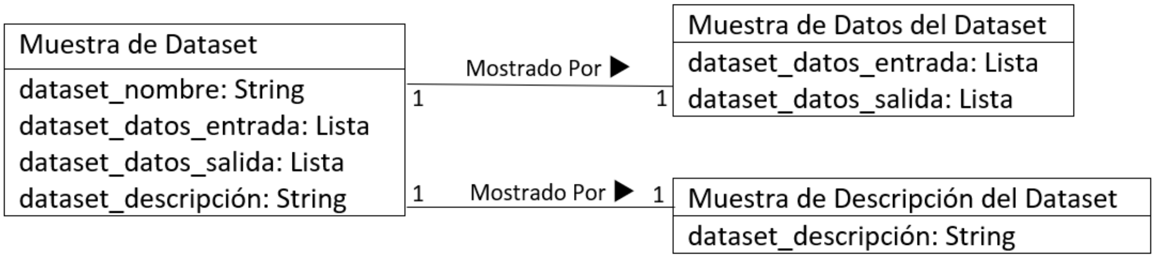


Figura Nº 8: Flujo de Información. **Fuente:** Elaboración propia

Controlador: Realiza iteraciones lógicas y validaciones de requisitos en cada una de las interfaces mostradas (Leff, y otros, 2001). Como ejemplo de clase controladora podemos observar la clase Proceso de Entrenamiento, que realiza las validaciones necesarias, aplicación del estimador escogido para realizar el proceso de entrenamiento dando lugar al clasificador que se utilizará posteriormente para la predicción de los datos. Dicho proceso lógico extraerá las variables necesarias de la clase Preparación de Entrenamiento para dar comienzo a este algoritmo. Véase en la Figura Nº 9 este ejemplo.

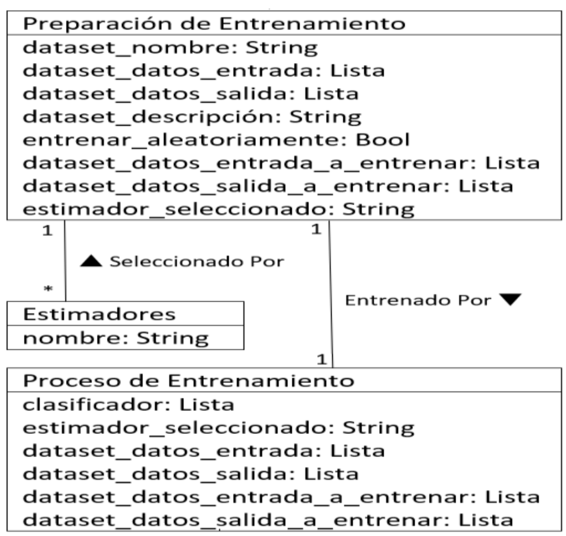


Figura Nº 9: Flujo de Información. **Fuente:** Elaboración propia

2.8. Patrones GRASP

GRASP son las siglas en inglés para denominar a los Patrones Generales de Asignación de Responsabilidades. Estos patrones de diseño son utilizados para la asignación de responsabilidades a una determinada clase. El grupo GRASP está conformado por 4 patrones principales, aplicables al diseño orientado a objetos (2019). A continuación, se refleja la utilización de estos en la solución que se propone:

Experto: Consiste en la asignación de responsabilidades al más competente en información, la clase cuenta con la información necesaria para cumplir la responsabilidad. Es el principio básico de asignación de responsabilidades que suele utilizarse en el diseño Orientado a Objetos. Este patrón se puede observar en la clase Preparación de Entrenamiento ya que se utiliza para realizar todas las acciones que tengan que ver con todo lo relacionado con la preparación del proceso de entrenamiento, siendo capaz de validar todas las entradas posibles. Este patrón se puede observar a continuación en la Figura Nº 10.

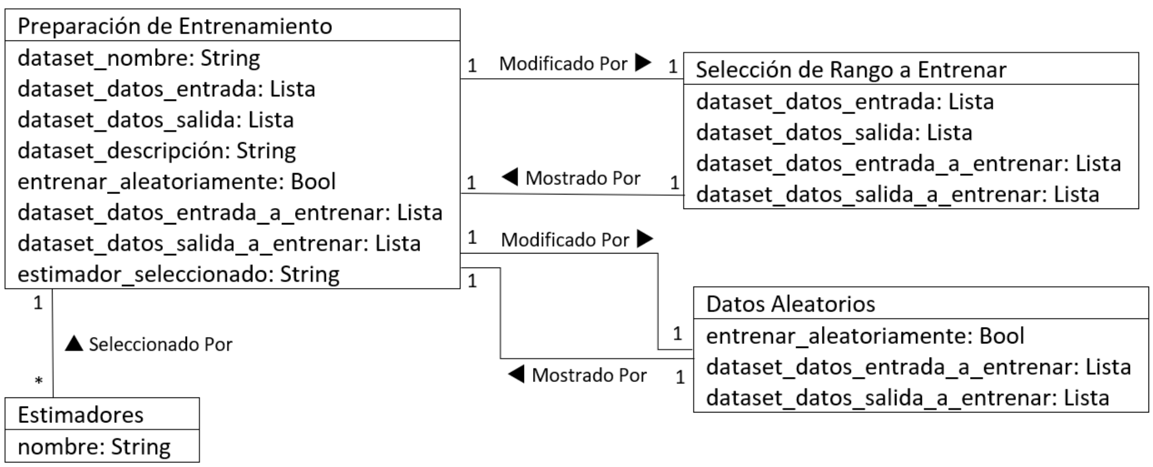


Figura Nº 10: Flujo de la funcionalidad del Patrón Experto. **Fuente:** Elaboración propia

A continuación, ejemplos de métodos que utiliza la clase experta Preparación de Entrenamiento:

```
def set_intances_and_atributes_numbers():
    globals()['window'].label_2.setText("Número de Instancias: " + str(len(dataset.data)))
    globals()['window'].label.setText("Número de Atributos: " + str(len(dataset.data[0])))
    pass
```

Figura Nº 11: Método *set_instances_and_atributes_numbers*. **Fuente:** Código Elaborado

En este método mostrado en la Figura Nº 11 la clase experta Preparación de Entrenamiento, muestra en la interfaz la cantidad de instancias del dataset, así como la cantidad de atributos de cada una de estas.

```python

def on_spin_box_text_changed_values():
    if globals()['window'].radioButton_2.isChecked():
        value = globals()['window'].spinBox.value()

        percent: float = float(value / len( globals()['dataset_data'] ))
        r_percent: int = int(percent * 100)

        globals()['window'].spinBox_2.setValue(r_percent)

    pass

```

Figura Nº 12: Método *on_spin_box_text_changed_values*. **Fuente:** Código Elaborado

En este método mostrado en la Figura Nº 12 la clase experta Preparación de Entrenamiento, valida los valores ingresados para la selección de rango de instancias a entrenar, tanto por cantidad fija o por un porcentaje de las mismas.

```python

def set_train_arrays():
    my_dataset_data = get_dataset_data()
    my_dataset_target = get_dataset_target()

    if train_randomly:
        my_dataset_data = []
        my_dataset_target = []

        list_pos = list(range(len(dataset_data)))
        random.shuffle(list_pos)

        globals()['list_pos_randomly'] = list_pos

        for i in range(len(list_pos_randomly)):
            my_dataset_data.append(dataset_data[list_pos_randomly[i]])
            my_dataset_target.append(dataset_target[list_pos_randomly[i]])
            pass

    if globals()['window'].radioButton.isChecked():
        percent: float = globals()['window'].spinBox_2.value()
        percent *= 0.01
        num = percent * len(dataset_data_randomly)
        num = int(num)
        globals()['dataset_X_train'] = dataset_data_randomly[:num]
```

Figura Nº 13: Método *set_train_arrays*. **Fuente:** Código Elaborado

En este método mostrado en la Figura Nº 13 la clase experta Preparación de Entrenamiento, realiza todo el proceso de validación de rango de instancias seleccionadas, ya sean aleatorias o no.

Alta Cohesión: Asignar una responsabilidad de modo que la unión se mantenga a gran escala. Asignar a las clases responsabilidades que trabajen sobre una misma área de aplicación y que no tengan mucha complejidad. Mejoran la claridad y facilidad con que se entiende el diseño. Se puede observar en las clases Proceso de Entrenamiento, Selección de Rango a Predecir, Datos Personalizados a Predecir, las cuales realizan funcionalidades específicas e independientes unas de otras, otorgando flujo de información a la clase Muestra de Predicción. Este patrón se puede observar a continuación en la Figura Nº 14.

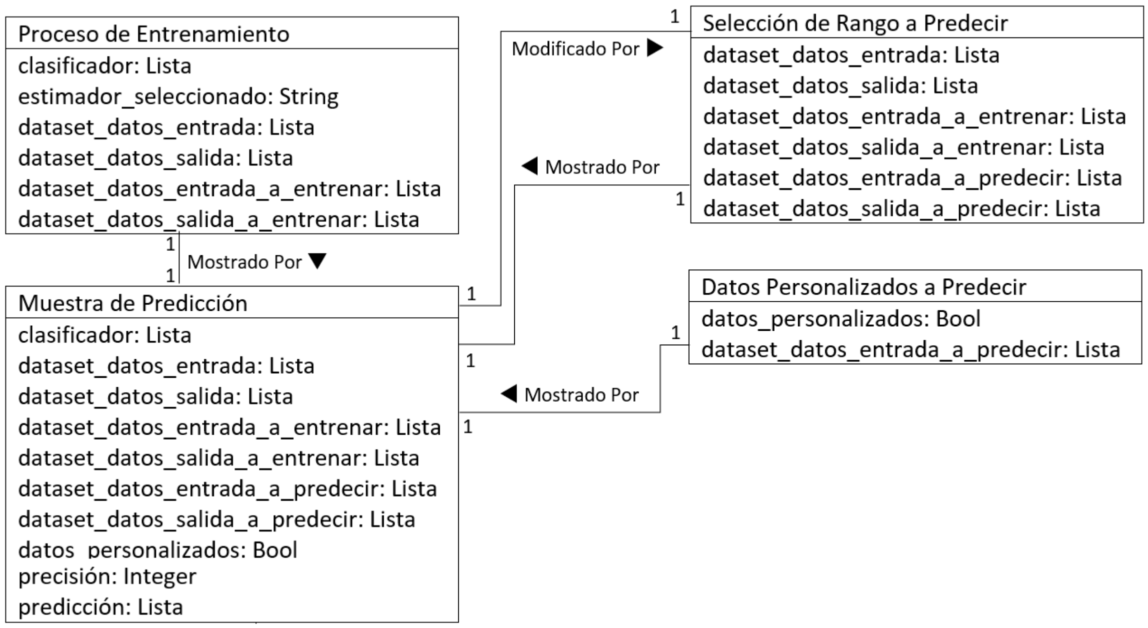
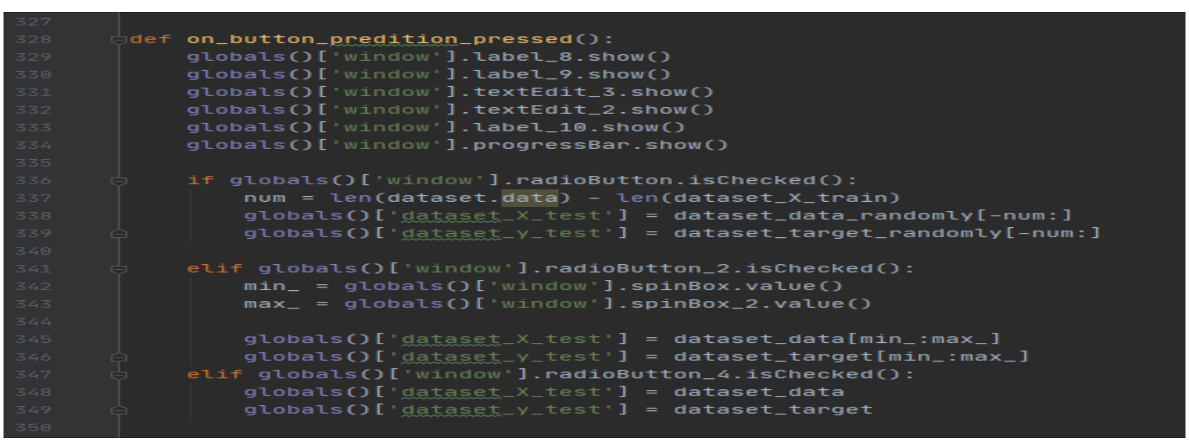


Figura Nº 14: Flujo de la funcionalidad del Patrón Alta Cohesión.

Fuente: Elaboración propia

A continuación, ejemplos de métodos que utilizan el patrón Alta Cohesión


```
def on_button_predition_pressed():
    globals()['window'].label_8.show()
    globals()['window'].label_9.show()
    globals()['window'].textEdit_3.show()
    globals()['window'].textEdit_2.show()
    globals()['window'].label_10.show()
    globals()['window'].progressBar.show()

    if globals()['window'].radioButton.isChecked():
        num = len(dataset.data) - len(dataset_X_train)
        globals()['dataset_X_test'] = dataset_data_randomly[-num:]
        globals()['dataset_y_test'] = dataset_target_randomly[-num:]

    elif globals()['window'].radioButton_2.isChecked():
        min_ = globals()['window'].spinBox.value()
        max_ = globals()['window'].spinBox_2.value()

        globals()['dataset_X_test'] = dataset_data[min_:max_]
        globals()['dataset_y_test'] = dataset_target[min_:max_]
    elif globals()['window'].radioButton_4.isChecked():
        globals()['dataset_X_test'] = dataset_data
        globals()['dataset_y_test'] = dataset_target
```

Figura Nº 15: Método *on_button_predition_pressed*.

Fuente: Código Elaborado

En este método, mostrado en la Figura Nº 15, se valida todo el proceso de rango de predicción seleccionado en la interfaz.

Bajo Acoplamiento: Asignar una responsabilidad para mantener un engranaje pobre. Es un principio que se debe recordar durante las decisiones de diseño. Soporta el diseño de clases más independientes. Asigna las responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible. Se evidencia en la relación que tiene la clase Muestra de Dataset con las clases Muestra de Datos del Dataset y Muestra de Descripción del Dataset. Esta estructura permite que un cambio en otro elemento de configuración no afecte a estas clases. Este patrón se puede observar a continuación en la Figura Nº 16.

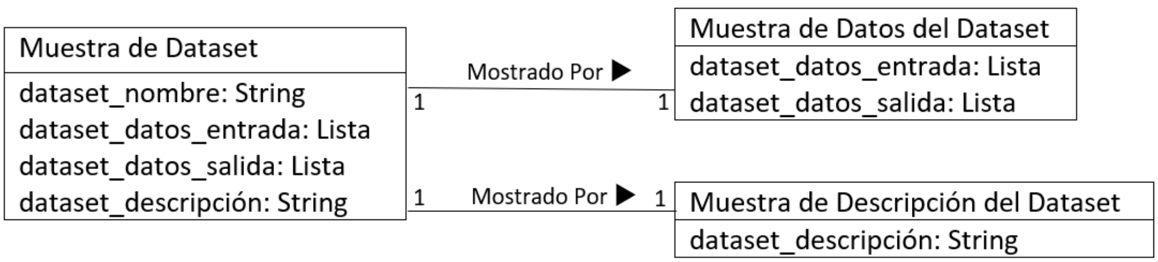


Figura Nº 16: Flujo de la funcionalidad del Patrón Bajo Acoplamiento. **Fuente:** Elaboración propia

A continuación, ejemplos de métodos que utilizan el patrón Bajo Acoplamiento:

```
def on_button_pressed_data():
    globals()['window'].textEdit.setText('')

    cont = 0
    for i in dataset_data:
        globals()['window'].textEdit.setText(
            globals()['window'].textEdit.toPlainText() + str(dataset_data[cont]) + " -> " + str(
                dataset_target[cont]) + '\n')
        cont += 1
        pass
    pass
```

Figura Nº 17: Método *on_button_pressed_data*.

Fuente: Código Elaborado

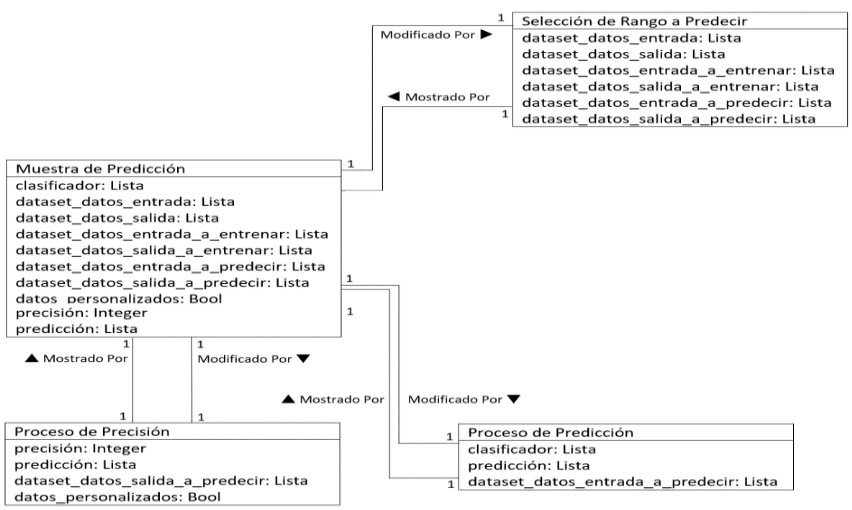
En este método mostrado en la Figura Nº 17 realiza el proceso de muestra de los datos de entrada y de salida en forma de instancias del dataset seleccionado.

```python
def on_button_pressed_description():
    globals()['window'].textEdit.setText(str(dataset_description))
    pass
```

Figura Nº 18: Método *on_button_pressed_description*. **Fuente:** Código Elaborado

En este método mostrado en la Figura Nº 18 realiza el proceso de muestra de la descripción del dataset seleccionado.

Controlador: Asignar la responsabilidad de controlar el flujo de eventos de la herramienta a clases específicas. Este patrón se evidencia en la clase Muestra de Predicción en relación y control de flujo de las clases Selección de Rango a Predecir, Proceso de Precisión y Proceso de Predicción. Estas clases relacionadas en dependencia del método llamado en la clase controladora invocan al algoritmo necesario para cumplir dicha funcionalidad. Este patrón se puede observar a continuación en la Figura Nº 19.



.

Figura Nº 19: Flujo de la funcionalidad del Patrón Controlador. **Fuente:** Elaboración propia

A continuación, ejemplos de métodos que utiliza la clase controladora Muestra de Predicción:

```python
def predict_process(with_target: bool = True):
    if with_target:
        globals()['prediction'] = classifier.predict(dataset_X_test)

        string_prediction_array = ''
        string_target_array = ''


        for i in range(len(prediction)):
            string_prediction_array += ' '
            string_prediction_array += str(prediction[i])
            string_prediction_array += ','

            string_target_array += ' '
            string_target_array += str(dataset_y_test[i])
            string_target_array += ','
            pass

        globals()['window'].textEdit_3.setText(string_prediction_array)
        globals()['window'].textEdit_2.setText(string_target_array)


        accuracity_prediction(prediction, dataset_y_test)
        globals()['window'].progressBar.setValue(int(globals()['accuracity']))
```

Figura Nº 20: Método *predict_process*. **Fuente:** Código Elaborado

En este método mostrado en la Figura Nº 20 la clase controladora Muestra de Predicción, realiza todo el proceso de control relacionado con la predicción dando uso a este método.

```python
def accuracity_prediction(prediction_acc: list = [], target_acc: list = []):
    equals = 0
    total = len(prediction_acc)

    for i in range(total):
        if prediction_acc[i] == target_acc[i]:
            equals += 1
            pass
        pass
    globals()['accuracity'] = (equals / total) * 100

    pass
```

Figura Nº 21: Método *accuracity_prediction*. **Fuente:** Código Elaborado

En este método mostrado en la Figura Nº 21 la clase controladora Muestra de Predicción, realiza todo el proceso de control relacionado con la precisión dando uso a este método.

2.9. Patrones GOF

2.9.1. Creacionales

- Abstrac Factory

Este patrón proporciona una interface para crear familias de objetos relacionados o dependientes sin especificar sus clases concretas. En la clase Selección de Dataset se crea de la familia Datasets el objeto seleccionado en la interfaz a través de las funciones *set_dataset_nombre()* y luego *load_dataset()*

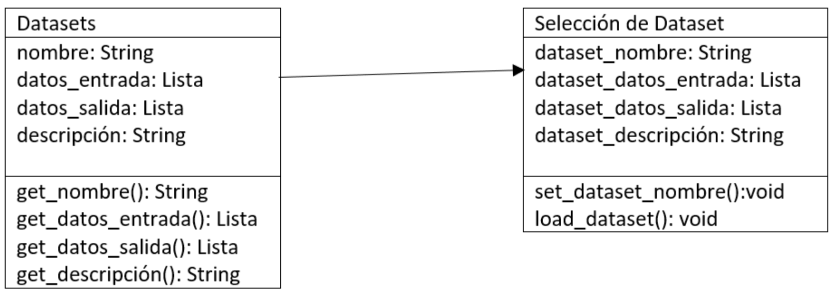


Figura Nº 22: Flujo de la funcionalidad Abstract Factory. **Fuente:** Elaboración propia

- Factory Method

Define una interfaz para crear un objeto, pero deja a las subclases decidir qué clase instanciar. En este caso se puede evidenciar en la clase Muestra de Dataset que posee el método *mostrar_información()* el cual tiene diferentes usos en las clases Muestra de Datos del Dataset y Muestra de Descripción del Dataset respectivamente

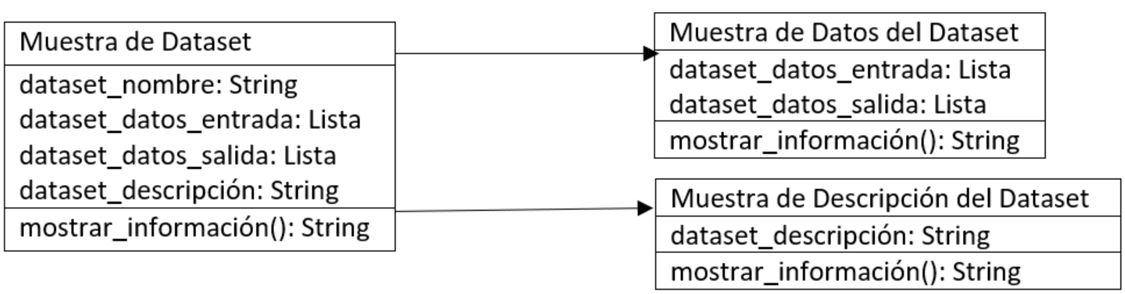


Figura Nº 23: Flujo de la funcionalidad Factory Method. **Fuente:** Elaboración propia

2.9.2. Estructurales

- Composite

Compone objetos en estructuras arborescentes para representar jerarquías. Permite manejar indistintamente objetos individuales y composiciones de objetos. Las funciones *crear_datos_entrada_a_predecir()* y *crear_datos_salida_a_predecir()* de la clase Muestra de Predicción son heredadas por las clases hijas Selección de Rango a Predecir y Datos Personalizados a Predecir, las cuales poseen variaciones y diferentes funcionalidades en estos métodos instanciados.

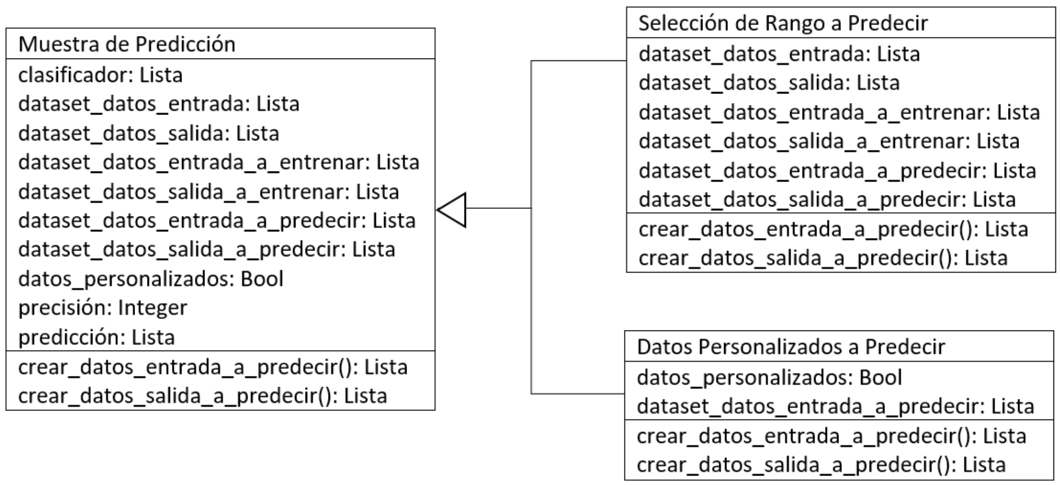


Figura Nº 24: Flujo de la funcionalidad Composite. **Fuente:** Elaboración propia

2.9.3. Comportamiento

- Observer

Define una dependencia de uno a muchos entre objetos así que cuando un objeto cambia de estado, todo el objeto que dependen de él es notificado para que se actualicen automáticamente. En este caso cuando se ejecuta el método *realizar_predicción()* de la clase Muestra de Predicción que realiza una modificación en la variable *clasificador*, inmediatamente se realizan el método respuesta *actualizar_resultados_predicción()* pertenecientes en las clases Proceso de Precisión y Proceso de Predicción.

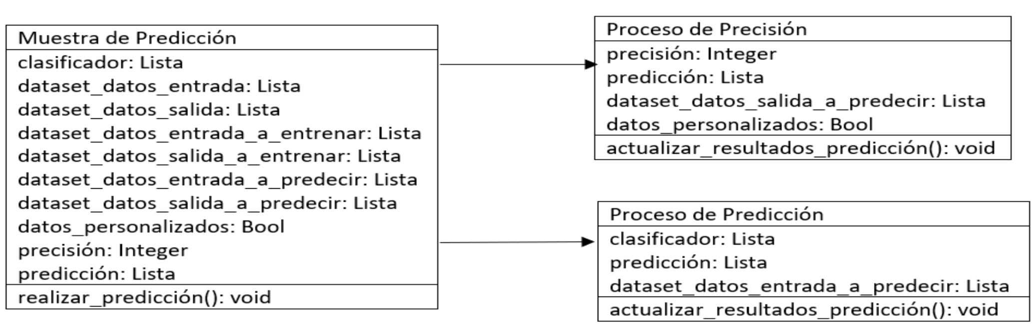


Figura Nº 25: Flujo de la funcionalidad Observer. **Fuente:** Elaboración propia

2.10. Diagrama de componentes

A continuación, en la Figura Nº 26, se muestra el diagrama de componentes elaborado a partir del diagrama de clases antes mostrado:

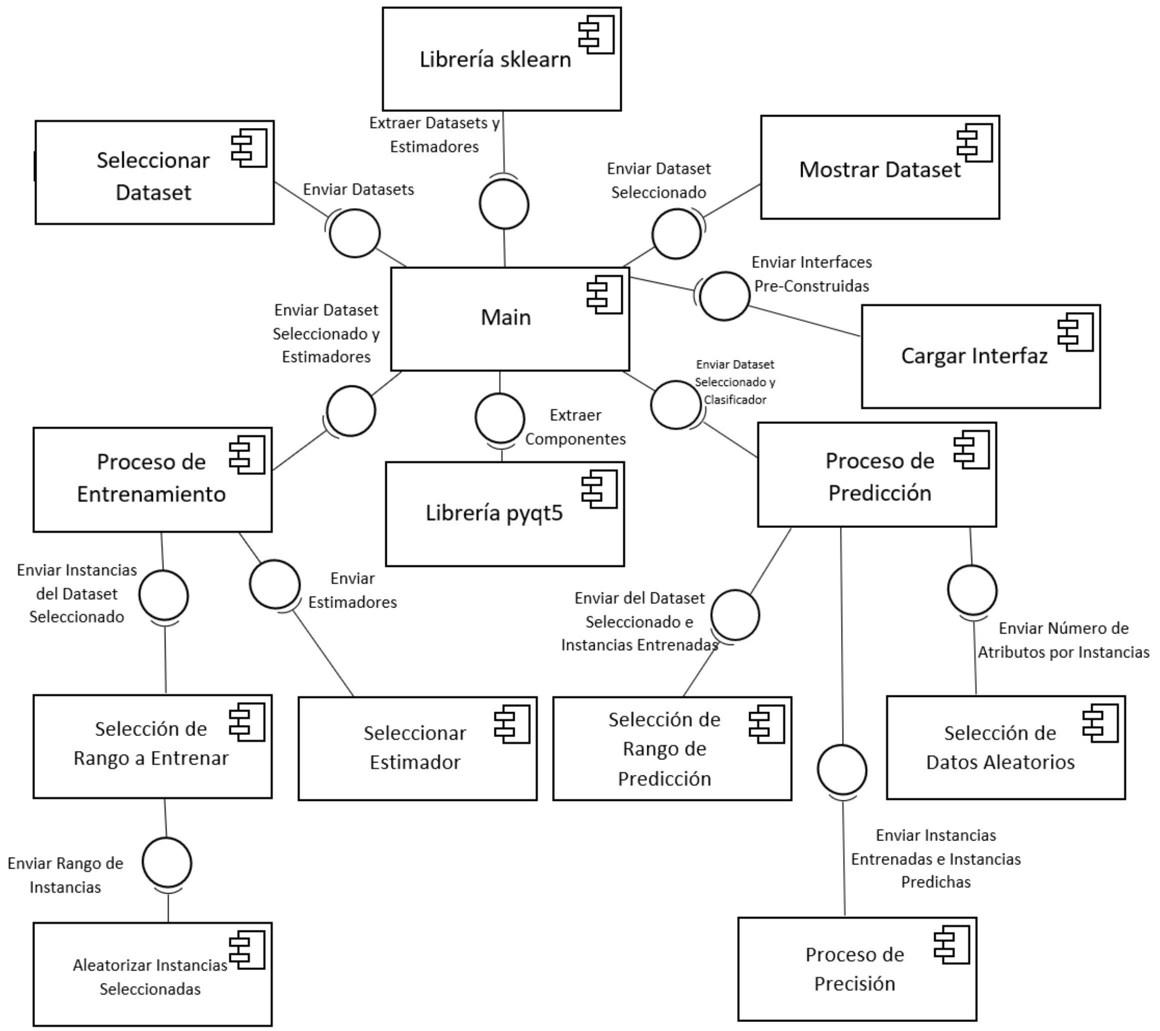


Figura Nº 26: Diagrama de componentes. **Fuente:** Elaboración propia

Conclusiones Parciales

Para un mejor entendimiento de los objetivos fundamentales en la confección de esta herramienta se realizó la entrevista para construir el Modelo de Dominio, representando el flujo de información presente en la elaboración de esta herramienta, además se identificaron los requisitos funcionales y no funcionales, para su diseño e implementación.

Durante el diseño de esta herramienta, para efectuar un buen diagnóstico se determinaron las Historias de Usuario, con las cuales se definieron las tareas y funcionalidades de la herramienta a desarrollar propuestas por el cliente.

En adición, para acumular una mayor cantidad de información valiosa, se representaron las prácticas a realizar según las metodologías utilizadas, se definieron las fuentes de información como los instrumentos de recolección de datos utilizados en su desarrollo.

Con el objetivo de un mejor entendimiento estructural de la herramienta se plantea el estilo arquitectónico, la representación de los patrones GRASP y GOF y el diagrama de componentes construido a partir del diagrama de clases.

CAPÍTULO 3, "Desarrollo y validación de la herramienta de configuración para análisis y entrenamiento de grandes volúmenes de datos digitales".

En este capítulo se expone la codificación de los principales métodos que responden a las funcionalidades descritas para cada tarea de ingeniería. También se aborda la planificación de los casos de pruebas y la ejecución de pruebas de aceptación.

3.1. Codificación

XP propone la definición de un estándar de programación para mantener un código legible. Esto beneficia la comunicación de los programadores a través del código y aún más si la implementación se realiza en pareja. Las convenciones de código, o como también se conocen estándares de codificación son modelos de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. A continuación, se presentan algunas de estas convenciones para la programación de la herramienta (Solís, 2012):

3.1.1. Tareas de ingeniería

La plantilla Tareas de ingeniería, tiene gran importancia, pues permite definir cada una de las actividades asociadas a las Historias de Usuario y que consentirán su implementación. Así se estima el tiempo que se llevará cada historia de usuario en implementarse, de acuerdo a su complejidad. Esta plantilla proporciona ventajas tales como:

- Permite organizar el proceso de implementación, pues las tareas se van implementando de acuerdo a su prioridad.
- Posibilita conocer el grado de complejidad de cada historia de usuario, teniendo en cuenta la cantidad de tareas asociadas.

Roles: Programador

3.1.2.1. Primera iteración

En esta iteración se le dio cumplimiento a la implementación de la Historia de Usuario número 1, 2 y 3, dichas HU brindan funcionalidades como seleccionar el Dataset a analizar, las descripciones básicas del mismo junto a la muestra de sus atributos y cantidad de instancias.

Historias de Usuario	**Tiempo de Implementación (semanas)**	
	Estimación	**Real**
Selección de Datasets	2	1
Muestra de datos de entrada y datos objetivos en forma de instancias.	1	1
Mostrar descripción del Dataset seleccionado	1.5	1

Tabla Nº 25: Tiempo de implementación

Fuente: Elaboración propia

Tareas de ingeniería vinculadas a esta iteración:

Tabla Nº26: Tareas de ingeniería # 1 Selección de Datasets

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 1
Nombre Tarea: Selección de Datasets	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio: 10/10/2019	**Fecha Fin:** 20/10/2019
Programador Responsable: Andy Armas	
Descripción: La Herramienta debe proporcionar la opción de seleccionar entre varios Datasets a analizar escoger uno y proceder a la fase de descripción del mismo.	

Fuente: Elaboración propia

Tabla Nº 27: Tareas de ingeniería # 2 Muestra de datos de entrada y datos objetivos en forma de instancias.

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 2
Nombre Tarea: Muestra de datos de entrada y datos objetivos en forma de instancias.	
Tipo de Tarea: Desarrollo, Mejora	**Puntos Estimados:** 1
Fecha Inicio: 22/10/2019	**Fecha Fin:** 27/10/2019
Programador Responsable: Andy Armas	
Descripción: La Herramienta debe mostrar una vez seleccionado el Dataset, los datos de entrada y los datos objetivos en forma de instancias, dichos datos serán mostrados en un cuadro de texto.	

Fuente: Elaboración propia

Tabla Nº 28: Tareas de ingeniería # 3 Mostrar descripción del Dataset seleccionado

Tarea de Ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** 3
Nombre Tarea: Mostrar descripción del Dataset seleccionado	
Tipo de Tarea: Desarrollo, Mejora	**Puntos Estimados:** 1.5
Fecha Inicio: 30/10/2019	**Fecha Fin:** 8/11/2019
Programador Responsable: Andy Armas	
Descripción: La Herramienta debe mostrar una breve descripción del Dataset seleccionado anteriormente, dicha descripción será mostrada en un cuadro de texto.	

Fuente: Elaboración propia

3.1.2.2. Segunda iteración

En esta iteración de la herramienta se les dará cumplimiento a la Historia de Usuario número 4, 5, 6 y 7, las cuales tienen como funcionalidad el desarrollo de todo el proceso de selección de opciones de entrenamiento y de la aplicación del mismo.

Historias de Usuario	**Tiempo de Implementación (semanas)**	
	Estimación	**Real**
Interfaz de entrenamiento.	**3**	**2**
Opciones de entrenamiento de instancias por tipo de rango	**2**	**2**
Opciones de entrenamiento aleatorio de instancias	**2.5**	**2**
Selección de Estimadores	**3**	**2**

Tabla Nº 29: Tiempo de implementación

Fuente: Elaboración propia

Tabla Nº 30: Tareas de ingeniería # 4 Interfaz de entrenamiento

Tarea de Ingeniería	
Número Tarea: 4	**Número Historia de Usuario:** 4
Nombre Tarea: Interfaz de entrenamiento	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 3
Fecha Inicio: 10/11/2019	**Fecha Fin:** 25/11/2019
Programador Responsable: Andy Armas	
Descripción: La Herramienta debe mostrar a través de una interfaz cómoda y sencilla las diferentes opciones para desarrollar el proceso de entrenamiento.	

Fuente: Elaboración propia

Tabla Nº 31: Tareas de ingeniería # 5 Opciones de entrenamiento de instancias por tipo de rango

area de Ingeniería	
Número Tarea: 5	**Número Historia de Usuario:** 5
Nombre Tarea: Opciones de entrenamiento de instancias por tipo de rango	
Tipo de Tarea: Desarrollo, Mejora	**Puntos Estimados:** 2
Fecha Inicio: 27/11/2019	**Fecha Fin:** 9/12/2019
Programador Responsable: Andy Armas	
Descripción: La Herramienta debe mostrar rangos de selección tanto por porcentaje o por instancias totales la cantidad de instancias a entrenar.	

Fuente: Elaboración propia

Tabla Nº 32: Tareas de ingeniería # 6 Opciones de entrenamiento aleatorio de instancias

Tarea de Ingeniería	
Número Tarea: 6	**Número Historia de Usuario:** 6
Nombre Tarea: Opciones de entrenamiento aleatorio de instancias	
Tipo de Tarea: Desarrollo, Mejora	**Puntos Estimados:** 2.5
Fecha Inicio: 11/12/2019	**Fecha Fin:** 23/12/2019
Programador Responsable: Andy Armas	
Descripción: La Herramienta debe mostrar la posibilidad de entrenar el rango de instancias seleccionadas de manera aleatoria, abarcando de manera más general la información del Dataset.	

Fuente: Elaboración propia

Tabla Nº 33: Tareas de ingeniería # 7 Opciones de entrenamiento aleatorio de instancias

Tarea de Ingeniería	
Número Tarea: 7	**Número Historia de Usuario:** 7
Nombre Tarea: Selección de Estimadores	
Tipo de Tarea: Desarrollo, Mejora	**Puntos Estimados:** 3
Fecha Inicio: 24/12/2019	**Fecha Fin:** 8/1/2020
Programador Responsable: Andy Armas	
Descripción: La Herramienta debe mostrar la posibilidad de entre diferentes tipos de estimadores seleccionar uno, el cual realizará el proceso de entrenamiento.	

Fuente: Elaboración propia

3.1.2.3. Tercera iteración

En esta iteración de la herramienta se les dará cumplimiento a las Historias de Usuario número 8, 9, 10, 11, 12 y 13 las cuales tienen como funcionalidades principales realizar todo el proceso de predicción y mostrar los resultados del mismo.

Historias de Usuario	**Tiempo de Implementación (semanas)**	
	Estimación	**Real**
Interfaz de Predicción	2	1.5
Opciones de predicción por tipo de rango	2.5	2
Opción de predicción de Datos Personalizados	1.5	1
Iniciar el Proceso de Predicción	4	3
Mostrar la Precisión de la Predicción	3.5	3
Volver a la Interfaz Anterior	1	0.5

Tabla Nº 34: Tiempo de implementación

Fuente: Elaboración propia

Tabla Nº 35: Tareas de ingeniería # 8 Interfaz de Predicción

Tarea de Ingeniería	
Número Tarea: 8	**Número Historia de Usuario:** 8
Nombre Tarea: Interfaz de Predicción	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio: 10/1/2020	**Fecha Fin:** 19/1/2020
Programador Responsable: Andy Armas	
Descripción: La Herramienta debe mostrar a través de una interfaz cómoda y sencilla las diferentes opciones para desarrollar el proceso de predicción, así como también los resultados del mismo.	

Fuente: Elaboración propia

Tabla Nº 36: Tareas de ingeniería # 9 Opciones de predicción por tipo de rango

Tarea de Ingeniería	
Número Tarea: 9	**Número Historia de Usuario:** 9
Nombre Tarea: Opciones de predicción por tipo de rango	
Tipo de Tarea: Desarrollo, Mejora	**Puntos Estimados:** 2.5
Fecha Inicio: 22/1/2020	**Fecha Fin:** 4/2/2020
Programador Responsable: Andy Armas	
Descripción: La Herramienta debe permitir elegir el rango de predicción entre las instancias totales del Dataset, los tipos de elección deben ser: - Todas las instancias - El resto de las instancias entrenadas - Rango específico	

Fuente: Elaboración propia

Tabla Nº 37: Tareas de ingeniería # 10 Opción de predicción de Datos Personalizados

Tarea de Ingeniería	
Número Tarea: 10	**Número Historia de Usuario:** 10
Nombre Tarea: Opción de predicción de Datos Personalizados	
Tipo de Tarea: Desarrollo, Mejora	**Puntos Estimados:** 1.5
Fecha Inicio: 5/2/2020	**Fecha Fin:** 12/2/2020
Programador Responsable: Andy Armas	
Descripción: La Herramienta debe permitir insertar datos personalizados para el proceso de predicción. Dichos datos de entrada serán analizados y se mostrarán los datos de salida resultantes del mismo.	

Fuente: Elaboración propia

Tabla Nº 38: Tareas de ingeniería # 11 Iniciar el Proceso de Predicción

Tarea de Ingeniería	
Número Tarea: 11	**Número Historia de Usuario:** 11
Nombre Tarea: Opción de predicción de Datos Personalizados	
Tipo de Tarea: Desarrollo, Mejora	**Puntos Estimados:** 4
Fecha Inicio: 15/2/2020	**Fecha Fin:** 7/3/2020
Programador Responsable: Andy Armas	
Descripción: La Herramienta debe mostrar una opción que permite el comienzo del proceso de predicción, teniendo en cuenta el Dataset seleccionado, las instancias a entrenar, si son o no aleatoriamente seleccionadas, el estimador de entrenamiento y las instancias a predecir.	

Fuente: Elaboración propia

Tabla Nº 39: Tareas de ingeniería # 12 Mostrar la Precisión de la Predicción

Tarea de Ingeniería	
Número Tarea: 12	**Número Historia de Usuario:** 12
Nombre Tarea: Mostrar la Precisión de la Predicción	
Tipo de Tarea: Desarrollo, Mejora	**Puntos Estimados:** 3.5
Fecha Inicio: 9/3/2020	**Fecha Fin:** 27/3/2020
Programador Responsable: Andy Armas	
Descripción: La Herramienta debe mostrar, luego del proceso de predicción, las clases resultantes y compararlas con las clases reales de rango de predicción. Dicha comparación será mostrada en una barra de progreso en forma de porcentaje.	

Fuente: Elaboración propia

Tabla Nº 40: Tareas de ingeniería # 13 Volver a la Interfaz Anterior

Tarea de Ingeniería	
Número Tarea: 12	**Número Historia de Usuario:** 12
Nombre Tarea: Volver a la Interfaz Anterior	
Tipo de Tarea: Mejora	**Puntos Estimados:** 1
Fecha Inicio: 30/3/2020	**Fecha Fin:** 7/4/2020
Programador Responsable: Andy Armas	
Descripción: La Herramienta debe permitir regresar al proceso o interfaz anterior.	

Fuente: Elaboración propia

La implementación en el proceso de desarrollo de un software adquiere gran importancia debido a que le da funcionalidad al producto que se desarrolla. Además, son importantes las pruebas que se le realizan al mismo para validar su correcto funcionamiento. El capítulo abarca las fases de implementación y prueba de la herramienta. Se exponen las tareas asignadas a las HU para llevar a cabo la implementación. Se muestra el estándar de codificación utilizado y el código de las HU con mayor prioridad de desarrollo. Se describen las pruebas realizadas a la herramienta con el objetivo de verificar si se cumplieron los requerimientos de la misma.

3.1.3. Estándares de codificación

- **Declaraciones:** Se realizará una declaración por línea.

```python

def on_combo_box_item_changed(self):
    v = window.comboBox.currentText()
    if v != 'Ninguno':
        window.pushButton.setEnabled(True)
    else:
        window.pushButton.setEnabled(False)
    pass

```

Figura Nº 27: Ejemplo de Declaración por línea en la función ***on_combo_box_item_changed()*** de la clase **Main**. **Fuente:** Elaboración propia

- **Inicialización**

Inicializar las variables locales donde se declaran. La única razón para no inicializar una variable donde se declara es, si el valor inicial depende de algunos cálculos que deben ocurrir.

```python

def on_button_pressed_data():
    globals()['window'].textEdit.setText('')

    cont = 0
    for i in dataset_data:
        globals()['window'].textEdit.setText(
            globals()['window'].textEdit.toPlainText() + str(dataset_data[cont]) + " -> " + str(
                dataset_target[cont]) + '\n')
        cont += 1
        pass
    pass

```

Figura Nº 28: Ejemplo de Inicialización en la función ***on_button_pressed_data()*** de la clase **Main. Fuente:** Elaboración propia

- **Colocación**

Poner las declaraciones solo al principio de los bloques. No esperar al primer uso para declararlas. Con la excepción de los ciclos.

```python

def set_algorithms_on_combo_box():
    globals()['window'].comboBox.addItem('Ninguno')
    alg = globals()['algorimthms']
    cont = 0
    for i in alg:
        globals()['window'].comboBox.addItem(alg[cont])
        cont += 1
        pass
    pass

```

Figura Nº 29: Ejemplo de Colocación en la función ***set_algorithms_on_combo_box()*** de la clase **Main. Fuente:** Elaboración propia

- **Métodos**

Los métodos se separan con una línea en blanco.

```python

def on_combo_box_changed_algorithm():
    check_train_button_avaliable()
    pass

def on_spin_box_text_changed_percent():
    if globals()['window'].radioButton.isChecked():
        percent = globals()['window'].spinBox_2.value()
        r_percent: float = percent * 0.01
        value: int = int(r_percent * len( globals()['dataset_data'] ))


        globals()['window'].spinBox.setValue(value)
    pass

```

Figura Nº 30: Ejemplo de Métodos en las funciones ***on_combo_box_changed_algorithm()*** y ***on_combo_box_text_changed_percent()*** de la clase **Main. Fuente:** Elaboración propia

- **Sentencias simples**

Cada línea debe contener como máximo una sentencia.

```python

def check_class_target_array_non_repeatly(array: list) -> bool:
    count_array = []
    for i in range(len(array)):
        value = array[i]
        if count_array.count(value) == 0:
            count_array.append(value)
        pass

    if len(count_array) > 1:
        return True

    return False

```

Figura Nº 31: Ejemplo de Sentencias Simples en la función ***set_algorithms_on_combo_box()*** de la clase **Main. Fuente:** Elaboración propia

- **Líneas en blanco**

Se debe usar siempre una línea en blanco en las siguientes circunstancias:

Entre métodos.

- Entre las variables locales de un método y su primera sentencia.
- Antes de un comentario de bloque o de un comentario de una línea.

```python

#### TRAIN PREPARATION ########

def on_radio_button_clicked():
    if globals()['window'].radioButton.isChecked():
        globals()['window'].spinBox_2.setEnabled(True)
    else:
        globals()['window'].spinBox_2.setEnabled(False)

    if globals()['window'].radioButton_2.isChecked():
        globals()['window'].spinBox.setEnabled(True)
    else:
        globals()['window'].spinBox.setEnabled(False)

    check_train_button_avaliable()
    pass
```

Figura Nº 32: Ejemplo de Espacios en Blanco en la función ***on_radio_button_clicked()*** de la clase **Main. Fuente:** Elaboración propia

- **Nomenclatura Métodos**

Los métodos cuando son compuestos tendrán todas sus letras en minúsculas y sus palabras serán separadas por el signo "_".

```
def on_radio_button_clicked():
```

Figura Nº 33: Ejemplo de Nomenclatura de Métodos en la función ***on_radio_button_clicked()*** de la clase **Main. Fuente:** Elaboración propia

- **Nomenclatura Variables**

Los nombres de las variables serán en minúsculas en caso de que sean palabras compuestas serán separadas por el signo "_". Los nombres de variables de un solo carácter solo se utilizarán para variables índices temporales.

```
classifier = []
fit = []
prediction = []
algorithm_current = ''
algorimthms = ['Support Vector Classification (SVC)']
```

Figura Nº 34: Ejemplo de Nomenclatura de Variables en las variables ***classifier***, ***fit***, ***prediction***, ***algorithm_current*** y ***algorimthms***. **Fuente:** Elaboración propia

3.2. Planificación de pruebas. Pruebas de aceptación o caso de prueba

Plan de Pruebas

#	Prueba	Requisitos Funcionales a Verificar	Variables de Entrada	Variables de Salida	Resultado
1	Selección de Datasets	- Seleccionar Dataset	datasets: Lista	dataset_nombre: String dataset_datos_entrada: Lista dataset_datos_salida: Lista dataset_descripción: String	Se muestra la interfaz para la elección de dataset.
2	Muestra de datos de entrada y datos objetivos en	- Mostrar Instancias - Mostar Atributos	dataset_nombre: String dataset_datos_entrada: Lista dataset_datos_salida: Lista		Se mostraron los datos de entrada y objetivos en forma de instancias.

	forma de instancias.				
3	Mostrar descripción del Dataset seleccionado	- Mostrar Descripción de Dataset	dataset_nombre: String dataset_descripción: String		Se mostró la descripción característica del dataset seleccionado.
4	Interfaz de entrenamiento	- Mostrar Instancias - Mostar Atributos - Realizar Proceso de Entrenamiento	dataset_datos_entrada: Lista dataset_datos_salida: Lista dataset_datos_entrada_a_entrenar: Lista dataset_datos_salida_a_entrenar: Lista entrenar_aleatoriamente: Bool estimador_seleccionado: String		Se muestra la interfaz de entrenamiento, así como todas sus opciones.
5	Opciones de entrenamiento de instancias por tipo de rango	- Mostrar Instancias - Seleccionar Instancias a Entrenar	dataset_datos_entrada: Lista dataset_datos_salida: Lista	dataset_datos_entrada_a_entrenar: Lista dataset_datos_salida_a_entrenar: Lista	Se ha optado por una de las dos opciones de instancias por tipo de rango, así como la cantidad a elegir.
6	Opciones de entrenamiento aleatorio de instancias	- Entrenar Aleatoriamente		entrenar_aleatoriamente: Bool	Se eligió la opción de Entrenar Aleatoriamente, dando paso a este proceso.
7	Selección de Estimadores	- Seleccionar Estimadores	estimadores: Lista	estimador_seleccionado: String	Se ha elegido una de las opciones disponibles de los estimadores.
8	Interfaz de Predicción	Mostrar Instancias Mostar Atributos	dataset_datos_entrada: Lista dataset_datos_salida: Lista dataset_datos_entrada_a_entrenar: Lista dataset_datos_salida_a_entrenar: Lista		Se muestra la interfaz de predicción con todos sus elementos a trabajar.

			dataset_datos_entrada_a_predecir: Lista dataset_datos_salida_a_predecir: Lista datos_personalizados: Bool precisión: Integer predicción: Lista		
9	Opciones de predicción por tipo de rango	- Mostrar Instancias - Seleccionar Instancias a Predecir	dataset_datos_entrada: Lista dataset_datos_salida: Lista dataset_datos_entrada_a_entrenar: Lista dataset_datos_salida_a_entrenar: Lista	dataset_datos_entrada_a_predecir: Lista dataset_datos_salida_a_predecir: Lista	Se han seleccionado algunas de las opciones por tipo de rango.
10	Opción de predicción de Datos Personalizados	- Mostar Atributos - Permitir la Predicción de Instancias Personalizadas	dataset_datos_entrada: Lista	dataset_datos_entrada_a_predecir: Lista datos_personalizados: Bool	El cuadro de texto fue rellenado con datos personalizados.
11	Iniciar el Proceso de Predicción	- Realizar Proceso de Predicción - Mostrar Predicción	clasificador: Lista dataset_datos_entrada_a_predecir: Lista	predicción: Lista	Se muestra los datos de salida elaborados por la predicción.
12	Mostrar la Precisión de la Predicción	- Mostrar Predicción - Mostrar Precisión	datos_personalizados: Bool	precisión: Integer	Se muestra una ventana con el valor del índice o por ciento de precisión de la predicción.

3.2.1. Diseño de los casos de prueba

Las pruebas de caja negra también conocidas como: pruebas funcionales, pruebas de caja opaca, pruebas de entrada/salida, pruebas inducidas por los datos, son las que no toman en cuenta el código, solo necesita saber cuáles pueden ser las posibles entradas sin necesidad de entender cómo se deben obtener las salidas, en donde se trata de encontrar errores en la interfaz mientras se está usando (Presman, 2003). Este grupo de pruebas se utilizó para la validación de las funcionalidades de la herramienta. Las pruebas de caja negra se centran principalmente en lo que "se quiere" de un módulo o sección específica de un software, es decir, es una manera de encontrar casos específicos en ese módulo que atiendan a su especificación.

Figura Nº 35: Esquema de Caja Negra. **Fuente:** (Presman, 2003)

Según (Presman, 2003), las pruebas de caja negra permiten identificar problemas tales como:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

3.2.1.1. Plantilla Caso de prueba de aceptación

La plantilla de Caso de prueba de aceptación, se genera de la etapa de pruebas. El objetivo de las pruebas de aceptación es validar que una herramienta cumple con el funcionamiento esperado y permitir al usuario de dicha herramienta que determine su aceptación, desde el punto de vista de su funcionalidad y rendimiento.

Roles: Cliente y Tester

Tabla Nº 41: Prueba de aceptación para la HU “Selección de Datasets”

Caso de Prueba de Aceptación	
Código Caso de Prueba: PL_ HU1_P1	**Nombre Historia de Usuario:** Selección de Datasets
Nombre de la persona que realiza la prueba: Andy Armas	
Descripción de la Prueba: Se realiza una prueba para comprobar que la herramienta muestre diferentes tipos de Datasets, luego de seleccionar uno, el botón “Seleccionar” puede ser presionado para dar paso a la siguiente interfaz.	
Condiciones de Ejecución: El cliente debe probar que se muestre los Datasets a elegir.	
Entrada / Pasos de ejecución: Se procede a seleccionar a través de una pestaña desplegable uno de los Datasets admisibles. Luego de seleccionar cualquier opción que no sea “Ninguno”, el botón “Seleccionar” se mostrará funcional para proseguir, se presiona para dar fin al Caso de Prueba.	
Resultado Esperado: La herramienta no presenta errores.	
Evaluación de la Prueba: Satisfactoria	

Fuente: Elaboración propia

Tabla Nº 42: Prueba de aceptación para la HU “Muestra de datos de entrada y datos objetivos en forma de instancias”

Caso de Prueba de Aceptación	
Código Caso de Prueba: PL_ HU2_P2	**Nombre Historia de Usuario:** Muestra de datos de entrada y datos objetivos en forma de instancias.
Nombre de la persona que realiza la prueba: Andy Armas	
Descripción de la Prueba: Se realiza una prueba para se muestren a través de instancias los datos de entrada y de salida.	
Condiciones de Ejecución: El cliente debe probar que se muestre los datos de entrada y de salida del Dataset seleccionado.	
Entrada / Pasos de ejecución: Se procede a seleccionar a través de un botón nombrado “Datos” para la visualización de los datos en un cuadro de texto.	
Resultado Esperado: La herramienta no presenta errores.	
Evaluación de la Prueba: Satisfactoria	

Fuente: Elaboración propia

Tabla Nº 43: Prueba de aceptación para la HU “Mostrar descripción del Dataset seleccionado”

Caso de Prueba de Aceptación	
Código Caso de Prueba: PL_ HU3_P3	**Nombre Historia de Usuario:** Mostrar descripción del Dataset seleccionado
Nombre de la persona que realiza la prueba: Andy Armas	
Descripción de la Prueba: Se realiza una prueba para se muestre una breve descripción característica del Dataset.	

Condiciones de Ejecución: El cliente debe probar que se muestre los atributos principales de la descripción del Dataset seleccionado.
Entrada / Pasos de ejecución: Se procede a seleccionar a través de un botón nombrado "Descripción" para la visualización de esta en forma de un cuadro de texto.
Resultado Esperado: La herramienta no presenta errores.
Evaluación de la Prueba: Satisfactoria

Fuente: Elaboración propia

Tabla Nº 44: Prueba de aceptación para la HU "Interfaz de entrenamiento"

Caso de Prueba de Aceptación	
Código Caso de Prueba: PL_ HU4_P4	**Nombre Historia de Usuario:** Interfaz de entrenamiento
Nombre de la persona que realiza la prueba: Andy Armas	
Descripción de la Prueba: Se realiza una prueba para se proceda a la interfaz de entrenamiento y la definición de las características del mismo.	
Condiciones de Ejecución: El cliente debe verificar que se proceda a esta interfaz correctamente.	
Entrada / Pasos de ejecución: Se procede esta interfaz a través de un botón nombrado "Entrenar" para acceder a la Interfaz de entrenamiento.	
Resultado Esperado: La herramienta no presenta errores.	
Evaluación de la Prueba: Satisfactoria	

Fuente: Elaboración propia

Tabla Nº 45: Prueba de aceptación para la HU "Opciones de entrenamiento de instancias por tipo de rango"

Caso de Prueba de Aceptación	
Código Caso de Prueba: PL_ HU5_P5	**Nombre Historia de Usuario:** Opciones de entrenamiento de instancias por tipo de rango
Nombre de la persona que realiza la prueba: Andy Armas	
Descripción de la Prueba: Se realiza una prueba para verificar que se permita seleccionar diferentes tipos de rango de selección de instancias a entrenar.	
Condiciones de Ejecución: El cliente debe verificar que se muestren y que se puedan elegir las opciones de entrenamiento: por instancias y por porcentaje.	
Entrada / Pasos de ejecución: Se procede seleccionar una de las dos opciones disponibles de: *selección de instancias a entrenar por tipo de rango,* dichas opciones son: - Por Instancias - Por Porcentaje Luego de esta elección se procederá a insertar el número que representa la cantidad de esta decisión.	
Resultado Esperado: La herramienta no presenta errores.	
Evaluación de la Prueba: Satisfactoria	

Fuente: Elaboración propia

Tabla Nº 46: Prueba de aceptación para la HU "Opciones de entrenamiento aleatorio de instancias"

Caso de Prueba de Aceptación	
Código Caso de Prueba: PL_ HU6_P6	**Nombre Historia de Usuario:** Opciones de entrenamiento aleatorio de instancias
Nombre de la persona que realiza la prueba: Andy Armas	
Descripción de la Prueba: Se realiza una prueba para verificar que se permita seleccionar una opción para entrenar aleatoriamente la cantidad de instancias seleccionadas.	
Condiciones de Ejecución: El cliente debe verificar que se muestren esta opción para proceder a un entrenamiento aleatorio de instancias.	
Entrada / Pasos de ejecución: Se procede seleccionar un cuadro de chequeo nombrado "Entrenar Aleatoriamente".	
Resultado Esperado: La herramienta no presenta errores.	
Evaluación de la Prueba: Satisfactoria	

Fuente: Elaboración propia

Tabla Nº 47: Prueba de aceptación para la HU "Selección de Estimadores"

Caso de Prueba de Aceptación	
Código Caso de Prueba: PL_ HU7_P7	**Nombre Historia de Usuario:** Selección de Estimadores
Nombre de la persona que realiza la prueba: Andy Armas	
Descripción de la Prueba: Se realiza una prueba para verificar las diferentes opciones de estimadores a elegir.	
Condiciones de Ejecución: El cliente debe verificar que se muestren a través de una pestaña desplegable los diferentes estimadores a elegir.	
Entrada / Pasos de ejecución: Se procede seleccionar una pestaña desplegable a la derecha de una etiqueta nombrada "Estimadores", la cual mostrará los estimadores a elegir, se seleccionará cualquier opción excepto la que nombre "Ninguno", para dar paso al siguiente Caso de Prueba.	
Resultado Esperado: La herramienta no presenta errores.	
Evaluación de la Prueba: Satisfactoria	

Fuente: Elaboración propia

Tabla Nº 48: Prueba de aceptación para la HU "Interfaz de Predicción"

Caso de Prueba de Aceptación	
Código Caso de Prueba: PL_ HU8_P8	**Nombre Historia de Usuario:** Interfaz de Predicción
Nombre de la persona que realiza la prueba: Andy Armas	
Descripción de la Prueba: Se realiza una prueba para verificar el acceso a la Interfaz de Predicción.	
Condiciones de Ejecución: El cliente debe verificar que se acceda a través de una opción a la Interfaz de Predicción abandonando la Interfaz de Entrenamiento. Para poder seleccionar esta opción será necesario haber elegido alguna de las opciones de selección de instancias a entrenar por tipo de rango, y a su vez haber seleccionado algún Estimador.	
Entrada / Pasos de ejecución: Se procede seleccionar un botón nombrado "Entrenar", el cual accederá a la Interfaz de Predicción.	
Resultado Esperado: La herramienta no presenta errores.	
Evaluación de la Prueba: Satisfactoria	

Fuente: Elaboración propia

Tabla Nº 49: Prueba de aceptación para la HU "Interfaz de Predicción"

Caso de Prueba de Aceptación	
Código Caso de Prueba: PL_ HU9_P9	**Nombre Historia de Usuario:** Opciones de predicción por tipo de rango
Nombre de la persona que realiza la prueba: Andy Armas	
Descripción de la Prueba: Se realiza una prueba para verificar la selección de diferentes opciones de predicción por tipo de rango.	
Condiciones de Ejecución: El cliente debe verificar existan y se puedan seleccionar estas opciones de predicción, las cuales serán: - Predecir Todas las Instancias - Predecir Resto de las Instancias Entrenadas - Predecir Rango de Instancias Solo será posible elegir solo una de estas opciones.	
Entrada / Pasos de ejecución: Se procede seleccionar entre unas opciones de chequeo las anteriormente mencionadas selecciones. En el caso de la opción "Predecir Rango de Instancias", en adición, se deberá especificar mediante la inserción de datos numéricos el rango de predicción.	
Resultado Esperado: La herramienta no presenta errores.	
Evaluación de la Prueba: Satisfactoria	

Fuente: Elaboración propia

Tabla Nº 50: Prueba de aceptación para la HU "Opción de predicción de Datos Personalizados"

Caso de Prueba de Aceptación	
Código Caso de Prueba: PL_ HU10_P10	**Nombre Historia de Usuario:** Opción de predicción de Datos Personalizados
Nombre de la persona que realiza la prueba: Andy Armas	
Descripción de la Prueba: Se realiza una prueba para verificar la posibilidad de inserción de Datos Personalizados para el proceso de predicción.	
Condiciones de Ejecución: El cliente debe verificar si es posible seleccionar una opción para la inserción de Datos Personalizados para la Predicción, dichos datos serán estrictamente de entrada y serán escritos en el mismo formato de las instancias del Dataset seleccionado.	
Entrada / Pasos de ejecución: Se procede seleccionar una opción nombrada "Predecir Datos Personalizados", luego de que está sea marcada, un cuadro de texto se volverá ejecutable, en el mismo, se insertarán los Datos Personalizados de entrada en el formato de instancias del Dataset seleccionado.	
Resultado Esperado: La herramienta no presenta errores.	
Evaluación de la Prueba: Satisfactoria	

Fuente: Elaboración propia

Tabla Nº 51: Prueba de aceptación para la HU "Iniciar el Proceso de Predicción"

Caso de Prueba de Aceptación	
Código Caso de Prueba: PL_ HU11_P11	**Nombre Historia de Usuario:** Iniciar el Proceso de Predicción
Nombre de la persona que realiza la prueba: Andy Armas	
Descripción de la Prueba: Se realiza una prueba para verificar la posibilidad de inserción de Datos Personalizados para el proceso de predicción.	
Condiciones de Ejecución: El cliente debe verificar que el proceso de Predicción se lleve a cabo a través de una opción. Esta opción será accesible, si y solo si, una de las opciones por tipo de rango o la opción de predicción de Datos Personalizados ha sido seleccionada.	
Entrada / Pasos de ejecución: Se procede presionar un botón llamado "Predecir" el cual iniciará el proceso de Predicción.	
Resultado Esperado: La herramienta no presenta errores.	
Evaluación de la Prueba: Satisfactoria	

Fuente: Elaboración propia

Tabla Nº 52: Prueba de aceptación para la HU "Mostrar la Precisión de la Predicción"

Caso de Prueba de Aceptación	
Código Caso de Prueba: PL_ HU12_P12	**Nombre Historia de Usuario:** Mostrar la Precisión de la Predicción
Nombre de la persona que realiza la prueba: Andy Armas	
Descripción de la Prueba: Se realiza una prueba para verificar la muestra de los datos de salida predichos en comparación con los datos de salida del Dataset seleccionado, dichos datos de salida serán en base a las instancias seleccionadas a predecir. Dicha comparación será mostrada a través de la Precisión en forma de porcentaje.	
Condiciones de Ejecución: El cliente debe verificar que tanto los datos de salida predichos como los reales del Dataset sean mostrados en cuadros de texto y a su vez la Precisión se evidencie en forma de Barra de Progreso mostrando su progreso. En el caso de haber seleccionado la opción de predicción de datos personalizados, no se mostrará la Precisión puesto que no se seleccionaron unas instancias propias del Dataset con las que comparar sus datos de salida con respecto con la predicción, solo se mostrarán el resultado de salida de las instancias insertadas.	
Entrada / Pasos de ejecución: Se procede presionar a verificar que los datos se muestren y se comparan manualmente, verificando si el porcentaje es correspondiente al resultado.	
Resultado Esperado: La herramienta no presenta errores.	
Evaluación de la Prueba: Satisfactoria	

Fuente: Elaboración propia

CAPTURAS DE PANTALLAS

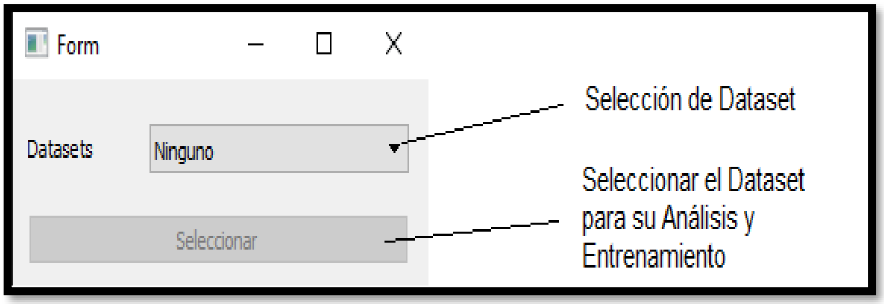


Figura Nº 36: Imagen Interfaz Selección de Datasets. **Fuente:** Elaboración propia

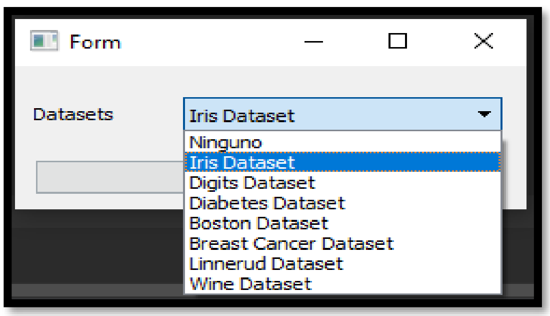


Figura Nº 37: Imagen Interfaz Selección de Datasets. **Fuente:** Elaboración propia

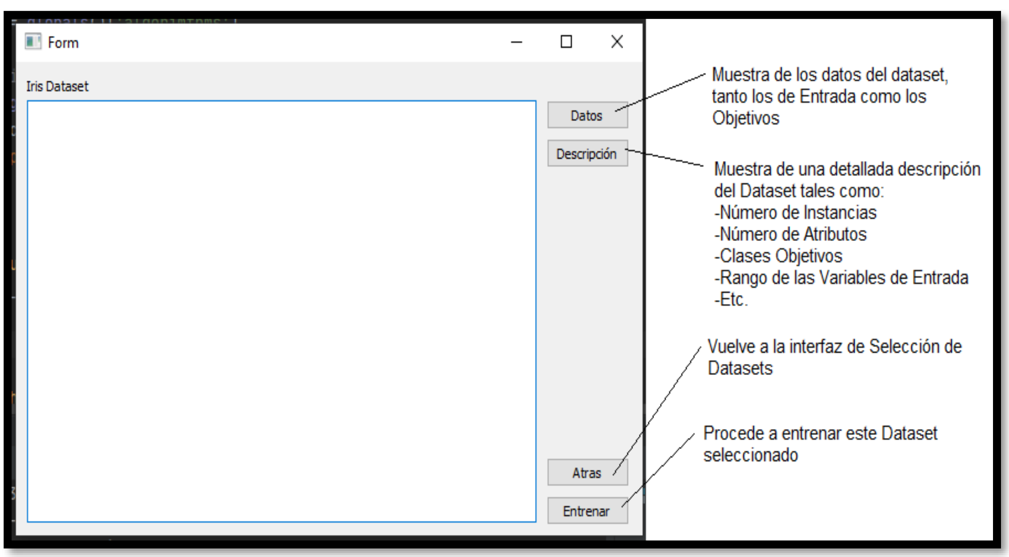


Figura Nº 38: Imagen Interfaz Descripción de Datasets. **Fuente:** Elaboración propia

Figura Nº 39: Imagen Interfaz Descripción de Datasets. **Fuente:** Elaboración propia

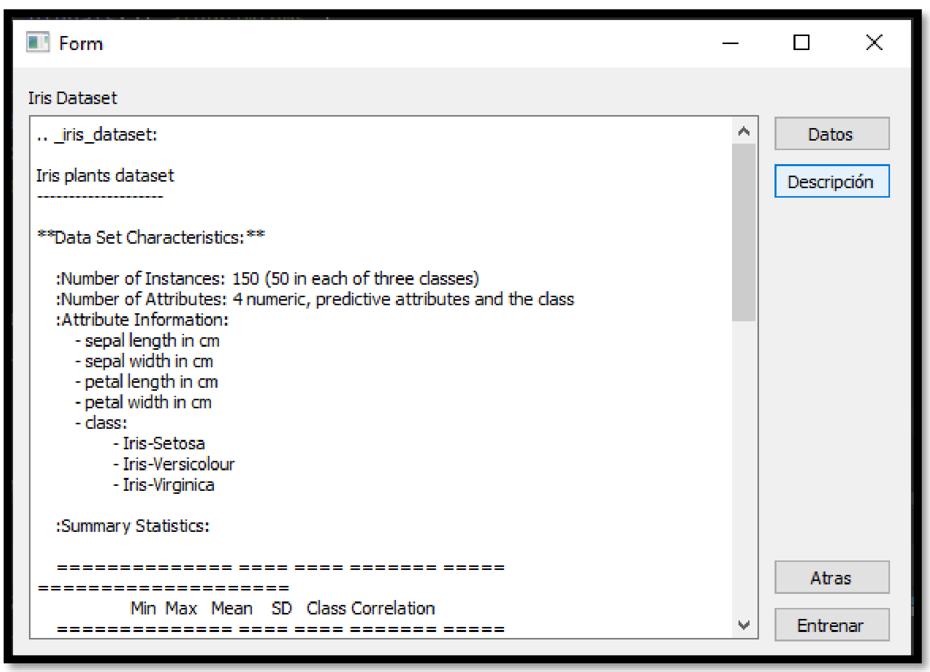


Figura Nº 40: Imagen Interfaz Descripción de Datasets. **Fuente:** Elaboración propia

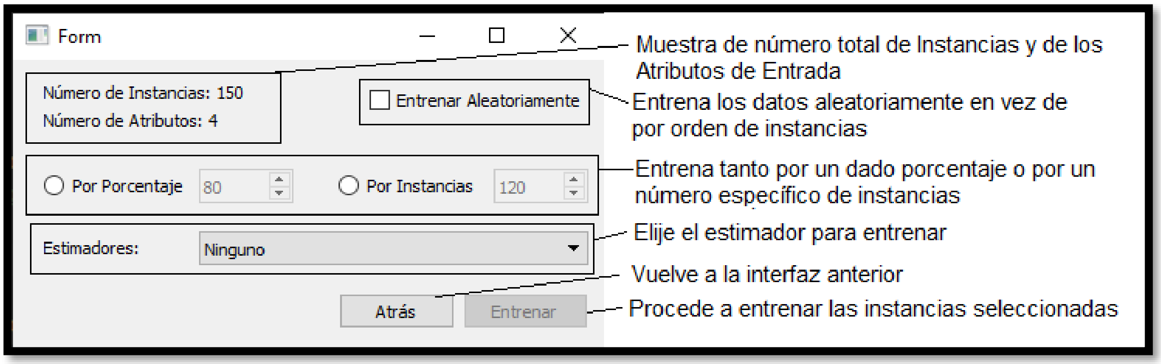


Figura Nº 41: Imagen Interfaz Entrenamiento. **Fuente:** Elaboración propia

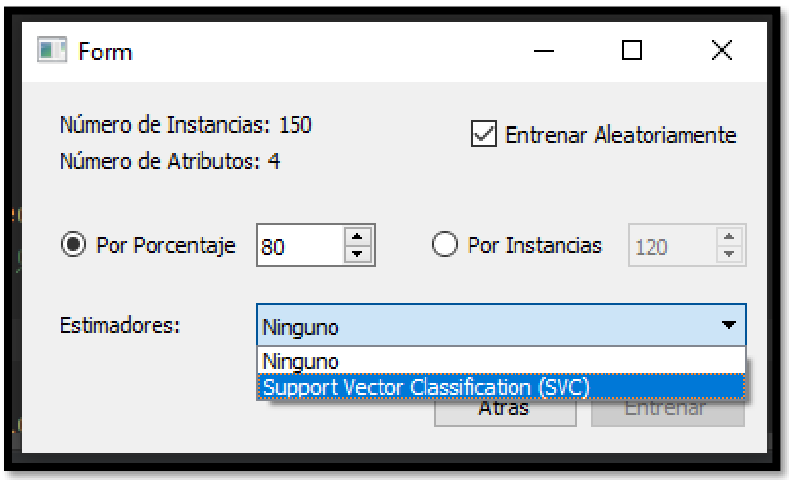


Figura Nº 42: Imagen Interfaz Entrenamiento. **Fuente:** Elaboración propia

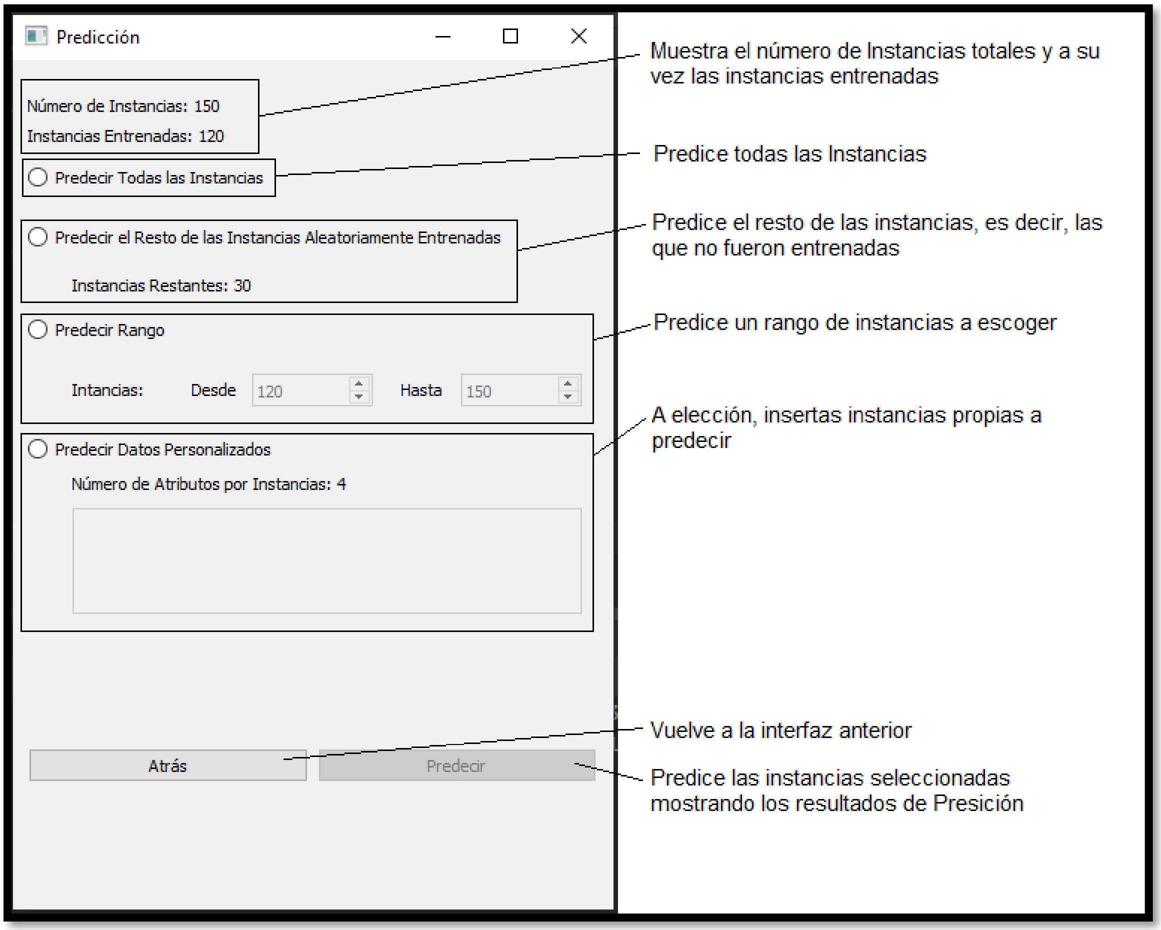


Figura Nº 43: Imagen Interfaz Predicción. **Fuente:** Elaboración propia

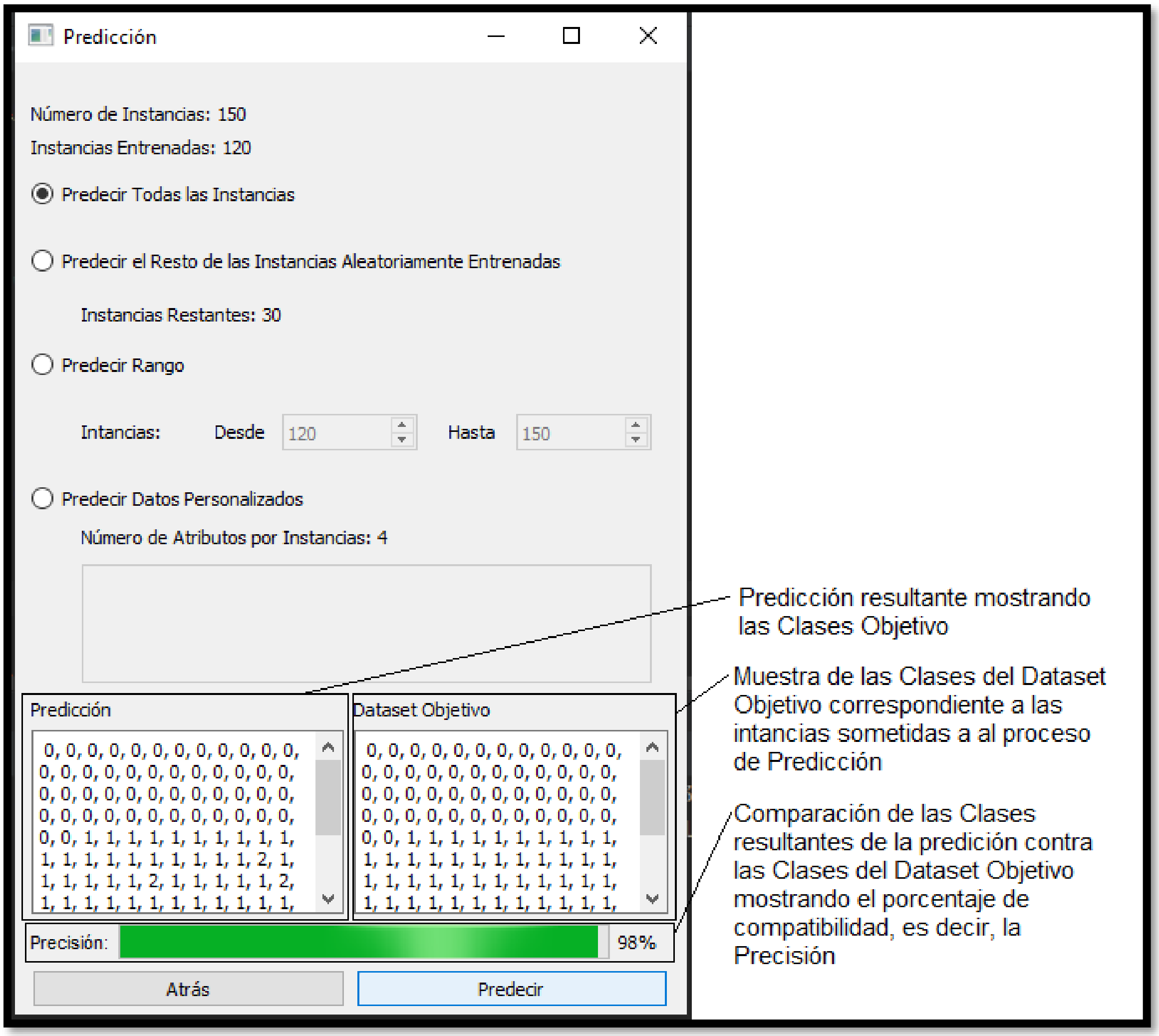


Figura Nº 44: Imagen Interfaz Predicción. **Fuente:** Elaboración propia

Conclusiones parciales

1. En este capítulo hemos abordado la fase de codificación de esta herramienta utilizando las estrategias de codificación para mantener un código legible, hecho que beneficia la comunicación de los programadores a través del código, lo que facilita la lectura, comprensión y mantenimiento del mismo. De esta manera hemos obtenido un código eficiente y limpio como resultado de estas estrategias.
2. Las tareas de ingeniería han sido de gran utilidad a esta herramienta, proporcionando un buen desarrollo de la misma, lo que derivó en mejor funcionabilidad del producto.
3. Con la elaboración de este capítulo hemos abordado la fase de pruebas planteada por la metodología XP, obteniendo un rendimiento positivo y satisfactorio de la herramienta elaborada.
4. La realización de las pruebas de aceptación, en las que el cliente se asegura de que las funciones implementadas cumplen su objetivo satisfactoriamente, fueron probadas de manera individual para cada HU asignándosele la evaluación correspondiente.
5. Todas las pruebas que se realizaron fueron efectivas y el cliente estuvo satisfecho, con las Historias de Usuarios definidas inicialmente.

Conclusiones generales

1. Partiendo de la aplicación de métodos teóricos como la entrevista se pudo determinar el estado de ejecución de los procesos vinculados al problema de investigación, así como criterios valiosos de algunos expertos para identificar necesidades que debía satisfacer la herramienta.
2. Fueron consultadas 68 fuentes bibliográficas a partir de los referentes teóricos identificados. Ello permitió determinar las bases teórico-metodológicas para la caracterización, diseño e implementación de la herramienta.
3. A partir del ciclo descrito por la metodología XP se logró materializar las Historias de Usuario, los Requisitos Funcionales, los Requisitos No Funcionales, las Tarjetas CRC, las Tareas de Ingeniería y las Pruebas de Aceptación, como evidencia del proceso de desarrollo llevado a cabo para obtener la herramienta.
4. El aprendizaje y reconocimiento del marco de trabajo que ofrece Python, además de la herramienta Pycharm, facilitó la codificación para la confección de la herramienta.
5. En el proceso planificado de diseño y aplicación de pruebas, a partir de los 13 requisitos identificados, no se determinaron errores en la elaboración de la herramienta.

BIBLIOGRAFÍA

A Brief History of Qt. C++ GUI Programming with Qt 4. **Blanchette, Jasmin y Summerfield, Mark. 2006.** Madrid : Prentice-Hall, 2006.

Ambler, Scott. 2019. The Agile Unified Process (AUP). *www.ambysoft.com.* [En línea] 19 de 11 de 2019. [Citado el: 5 de 7 de 2020.] http://www.ambysoft.com/unifiedprocess/agileUP.html..

Aranguren, Gilber. 2014. Comparación de los patrones de arquitectura MVC y MVT. *www.ingsoftwarei2014.wordpress.com.* [En línea] 2014. [Citado el: 25 de 9 de 2020.] http://ingsoftwarei2014.wordpress.com.

Beck, Kent. 2004. *Extreme Programming Explained: Embrance Change.* s.l. : Addison-Wesley, 2004.

Béjar, Javier. 2007. S. Kotsiantis, supervisado Aprendizaje Automático: Una Revisión de la Clasificación de las técnicas de Informática. *www.informatica.si.* [En línea] 2007. [Citado el: 8 de 23 de 2020.] http://www.informatica.si/PDF/31-3/11_Kotsiantis% 20 -%%% 20Supervised 20Machine 20Learning 20% -% 20A% 20de% ... 2.

Berry, M.J. y Linoff, G. 1997. *Data Mining Techniques For Marketing, Sales and Customer Support.* 1997.

Bishop, Christopher. 2008. *Pattern Recognition and Machine Learning.* s.l. : Springer Verlag, 2008. ISBN 978-0-3873-1073-2.

Brenner, Joseph. 2016. Emacs as a Perl IDE. *www.obsidianrook.com.* [En línea] 2016. [Citado el: 9 de 7 de 2020.] http://obsidianrook.com/perlnow/emacs_as_perl_ide.html.

Calderón, Amaro y Valverde, J.C. 2007. *Metodologías Ágiles.* Perú : Trujillo, 2007.

Calvo, Diego. 2017. diegocalvo.es. [En línea] 2017. http:/www.diagocalvo.es/tipos-de-datos-estructuardos/.

Ciementson, Bill. 2017. Using Emacs as a Lisp IDE. *www.cl-cookbook.sourceforge.net.* [En línea] 2017. [Citado el: 9 de 7 de 2020.] http://cl-cookbook.sourceforge.net/emacs-ide.html.

Codd, Edgar F. 1996. www.computerworld.es. *Las doce reglas de OLAP.* [En línea] 1996. [Citado el: 8 de 13 de 2020.] http://www.computerworld.es/archive/olap-acceso-rapido-a-los-datos/html.

Data Mining and Knowledge Discovery Handbook. **Maimon, Oded y Rokach, Lior. 2005.** New York : Secaucus, 2005.

Devivjer, P.A. y Kittler, J. 1982. *Pattern Recognition: A Statical Approach.* London : Prentice-Hall, 1982.

Elkan, Charles. 2011. *Evaluating Classifiers.* San Diego : University of California, 2011.

Fernández, Javier. 2015. *Módulo de minería de datos para el gestor de recursos de hardware y software.* La Habana : Universidad de Ciencias Informáticas, 2015. Tesis de Grado. Tutores: MSc. Darián Horacio Grass Boada y Ing. Husseyn Despaigne Reyes.

Flach, Peter. 2012. *Machine Learning: The Art and Science of Algorithms that Make Sense of Data.* New York : Cambridge University Press, 2012. ISBN 978-1-107-42222-3.

García Serrano, Alberto. 2012. *Inteligencia Artificial: Fundamentos, práctica y aplicaciones.* s.l. : Editorial RC Libros, 2012. ISBN 978-84-939450-2-2.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 1999. *The Unified Software Development Process.* 1999. ISBN 0-201-57169-2.

Knowlton, Jim. 2009. *Python.* España : Anaya Multimedia-Anaya Interactiva, 2009. ISBN 978-84-415-2317-3.

Kruchten, Philippe. 2003. *Rational Unified Process.* 2003.

Landa, Francisco Javier. 2016. Tratamiento de los datos. *fcojlanda.me.* [En línea] 2016. [Citado el: 26 de 7 de 2020.] http://fcojlanda.me/es/ciencia-de-los-datos/kdd-y--mineria-de-datos.

Landgraf, Katja. 2011. *Requirement Management in Product Development.* s.l. : Symposion Publishing, 2011. ISBN 978-3-939707-84-4.

Leff, Avraham y Rayfield, James T. 2001. *Web-Application Development Using the Model/View/Controller Design Pattern.* s.l. : IEEE Enterprise Distributed Object Computing Conference, 2001.

Letelier, P. y Penades, M.C. 2006. Metodologías Ágiles para el desarrollo del software: Extreme Programming (XP). [En línea] 2006. [Citado el: 5 de 7 de 2020.] http://www.cyta.com.ar/ta0502/v5n2a1.html.

Loarte. 2014. 2014.

Loarte, B. G y Chiluisa, A. P. 2014. *Desarrollo e implantación del sistema de control de inventarios y gestion de laboratorios para la facultad de ciencias de la escuela politecnica nacional.* Quito : Escuela Politécnica Nacional de Quito, 2014.

Maimon , Oded y Rokach, Lior. 2010. *Data Mining and Knowledge Discovery Handbook.* New York : Springer, 2010. ISBN 978-0-387-09823-4.

Martelli, Alex. 2007. *Python. Guía de referencia.* España : s.n., 2007.

McConnell, Steve. 1996. *Rapid Development: Taming Software Schedules.* 1st ed., Redmon, WA : Microsoft Press, 1996. ISBN 1-55615-900-5.

Orjuela, A. y Rojas, M. 2008. Las Metodologías de desarrollo Ágil como una oportunidad para la Ingeniería del software educativo. *www.bdigital.unal.edu.co.* [En línea] 24 de 5 de

2008. [Citado el: 9 de 7 de 2020.] http://www.bdigital.unal.edu.co/15430/1/10037-182161-PB.pdf.

Pardo, A y Ruiz, M.A. 2002. SPSS 11. Guía para el análisis de datos. Madrid : McGraw-Hill, 2002. ISBN 978844137502.

2019. Patrones de software para asiganción de responsailidades. *www.pararoni.net.* [En línea] 28 de 7 de 2019. [Citado el: 28 de 7 de 2020.] http://www.pararoni.net/2019/07/28/principios-de-software-para-asiganción-de-responsabiidades-grasp/.

Presman, R. S. 2003. *Ingeniería del Software, un enfoque práctico.* s.l. : El Proceso, 2003.

Raschka, Sebastian. 2015. *Pynthon Machine Learning.* s.l. : Packt Open Source, 2015. ISBN 978-1-78355-513-0.

RCCI. **Rodríguez, Yuniet y Díaz, Anolandy. 2009.** 3-4 julio-diciembre, 2009, Vol. 3.

Refaeilzadeh, Payam, Tang, Lei y Lui, Huan. 2008. *k-fold Cross-Validation.* Arizona : Arizona State University, 2008.

Reutemann, P., Pfahringer, B. y Frank, E. 2004. Proper: A Toolbox for Learnin from Relational Data with Propositional and Multi-Instance. *17th Australian Joint Conference on Artificial Intelligence.* Sidney : Springer-Verlag, 2004.

Revista Científica Mundo de la Investigación y el Conocimiento. **Flores, Galo. 2019.** 1, 2019, Vol. 3, págs. 955-970. enero.

Rocha, Roberto. 2015. [En línea] 2015. http://dtyoc-com.cdn.ampproject.org/v/s/dtyoc.com/2015/05/09/sas-analytics-y-enterprise/amp/.

Rodríguez, Tamara. 2014. *Metodología de desarrollo para la Actividad productiva de la UCI.* La Habana : Universidad de Ciencias Informáticas, 2014.

Romeu, J Alcaide. 2009. Proyecto de sistemas informáticos. *www.ucm.es.* [En línea] 2009. [Citado el: 5 de 7 de 2020.] http://www.uam.es/EPS//documento/14477472272/17837_PSI_1718.pdf?blobheader=application/pdf.

Rouhianen, Lasse. 2018. Inteligencia artificial, 101 cosas que debes saber sobre nuestro futuro. *www.planetadelibros.com.* [En línea] 2018. [Citado el: 5 de 7 de 2020.] http://www.planetadelibros.com/Inteligencia-artificial-101-cosas-que-debes-saber-sobre-nuestro-futuro/html.

Russell, S y Norving, P. 2004. *Inteligencia artificial, un enfoque moderno.* s.l. : American Association for Artificial Intelligence, 2004.

Solís, C.M. 2012. *Una explicación de la programación extrema XP.* 2012.

St.Amant, Kirk y Still, Brian. *Handbook of Research on Open Source Software: Technological, Economic, and Social Perspectives.* 1591409993.

Stellman , Andrew y Greene, Jennifer. 2005. *Applied Software Project Management.* Cambridge, MA : O'Reilly Media, 2005. ISBN 0-596-00948-8.

Técnicas de reducción de datos en KDD. El uso de Algoritmos Evolutivos para la Selección de las Instancias. **Herrera, Francisco y Cano, J.R. 2006.** Madrid : Universidad Rey Juan Carlos, 2006. Actas del I Seminario sobre Sistemas Inteligentes (SSIO6). págs. 165-181.

Todo programación. **Domínguez-Dorado, M. y Som, Guillermo. 2005.** 11, s.l. : Iberprensa Madrid, 2005.

Torossi, G. 2012. El Proceso Unificado de Desarrollo de Software. . *www.carlosfau.com.ar.* [En línea] 2012. [Citado el: 5 de 7 de 2020.] http://carlosfau.com.ar/nqi/nqifiles/Proceso%20Unificado%20Manual.pdf.

2020. Universidad de las Ciencias Informáticas. [En línea] 2020. http://www.uci.cu.

Universidad de las Ciencias Informáticas. **2020.** 2020.

Wells, Don. 2020. *Extreme Programming: A gentle introduction.* 2020.

Wiegers, Karl E. 2003. *Software Requirements 2: Practical techniques for gathering and managing requirements throughout the product development cycle .* Redmond : Microsoft Press, 2003. ISBN 0-7356-1879-8.

Witten, Ian H. y Frank, Eibe. 2011. *Data Mining: Practical machine learning tools and techniques Morgan Kaufmann.* s.l. : McGraw Hill, 2011. ISBN 978-0-12-374856-0.

Witten, Ian H., y otros. 1999. *Weka: Practical Machine Learning Tools and Techniques with Java Implementations. Proceedings of the ICONIP/ANZIIS/ANNES'99 Workshop on Emerging Knowledge Engineering and Connectionist Based Information Systems.* 1999.

Zhu, Xingquan y Davidson, Ian. 2007. *Knowledge Discovery and Data Mining: Challenges and Realities.* Hershey, New York : s.n., 2007. ISBN 978-1-59904-252-7.