Universidad de las Ciencias Informáticas

FACULTAD 6

Título: Modelo para la predicción del estado ocioso o activo de las estaciones de trabajo en el sistema SIMON.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(es): Karel Rodríguez Carmenates

Reisel González Pérez

Tutor(es): Msc. Longendri Aguilera Mendoza

Ing. Cesar Raúl García Jacas

Co-tutor: Ing. Ernesto Contreras Torres

La Habana, Junio del 2013

"Año 55 de la Revolución"

«Investigar es ver lo que todo el mundo ha visto, y pensar lo que nadie más ha pensado»
Albert Szent-Györgi

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Karel Rodríguez Carmenates

_____

Firma del Autor

Msc. Longendri Mendoza Aguilera

_____

Firma del Tutor

Reisel González Pérez

_____

Firma del Autor

Ing. Cesar Raúl García Jacas

_____

Firma del Tutor

Ing. Ernesto Contreras Torres

_____

Firma del Co-Tutor

DATOS DE CONTACTO

Tutores:

Msc. Longendri Mendoza Aguilera

Centro de Tecnologías de Gestión de Datos (DATEC)

Universidad de las Ciencias Informáticas (UCI)

Email: loge@uci.cu

Ing. Cesar Raúl García Jacas

Centro de Tecnologías de Gestión de Datos (DATEC)

Universidad de las Ciencias Informáticas (UCI)

Email: crjacas@uci.uci.cu

Co-tutor:

Ing. Ernesto Contreras Torres

Centro de Tecnologías de Gestión de Datos (DATEC)

Universidad de las Ciencias Informáticas (UCI)

Email: econtreras@uci.cu

DEDICATORIA

A mi papá, mi mamá, mi segunda mamá, mis hermanos y en especial a mi hermanita del alma.

Karel

A mis abuelos paternos, que hoy no están físicamente… pero siempre los llevo en el corazón.

A mi mamá, por ser mi ejemplo de sacrificio y nunca decepcionarse de mí.

A mi hermanita, por exigirme que me convierta en su ejemplo a seguir.

A mi papá, por obligarme a esforzarme al máximo en cada momento.

A mi AMOR que nunca acaba, por su apoyo y dedicación.

Reisel

AGRADECIMIENTOS

A mi papá por todo el sacrificio realizado para que mis sueños se hicieran realidad, por enseñarme el camino a seguir y servirme de guía con su ejemplo, por brindarme su apoyo incondicional en todo momento, por ser mi eterno compañero y amigo.

A Yami por haber sido la madre que no tuve, por haber llenado ese espacio vacío sin que ni siquiera se notara la diferencia y por haber sido mi consejera durante todos estos años.

A mi hermana querida, quien me da las fuerzas e inspiración necesarias para seguir adelante en cada momento.

A mis hermanos por formar parte de mi vida.

A mi mamá por haberme dado la dicha de nacer.

A Reisel mi dúo de tesis por haberme acompañado todo el tiempo durante esta travesía.

A las amistadas con las que he compartido todos estos años a Félix, Yasel, el Michi, Yisel, Yudy, Mayde, Liuva, Fernando.

A los profesores que aportaron su grano de arena en mi formación en especial mi entrenador Mawad y el profe Manolo.

A los tutores por todo su apoyo brindado.

A todas aquellas personas que de una forma u otra han estado presentes en estos cinco años.

A la revolución por haberme dado esta maravillosa oportunidad de ser útil a la sociedad.

Karel

AGRADECIMIENTOS

Primeramente quiero dedicar un agradecimiento especial a Karel, mi compañero de tesis por tantas horas de sacrificio y dedicación a este trabajo, por soportarme y sobre todo por su paciencia, eres un ejemplo para mí y tienes mi respeto.

A mami "Lile" y papi "Macho", que más que mis abuelos paternos, fueron mis "padres"... ellos que no tuvieron la oportunidad de estudiar, pero desde que era un niño apoyaron mis sueños de superación.

A mi mamá por ser el más grande ejemplo de sacrificio que he conocido en la vida… porque no solo fuiste una niña de 15 años cuidando de un niño, además dejaste tu vida para que yo pudiera tener la mía, porque aun cuando tus calificaciones eran sobresalientes tuviste que abandonar tus estudios y tus sueños para cuidar de mí… porque renunciaste a tus sueños para que yo pudiera cumplir los míos, porque nunca te decepcionaste de mí, ni siquiera en los peores momentos, porque a pesar de todo nunca te he escuchado quejarte y eso nunca lo olvidaré.

A mi papá, que aunque muchas veces no entendí algunas de sus decisiones, hoy le agradezco porque gracias a su rectitud y visión, puedo decir que logré uno de mis mayores sueños y crecí como persona y como profesional.

A mi abuelo "Tormento", porque de ti no solo heredé mi raza, de ti heredé el espíritu, el carácter y los principios. Por ti aprendí a valorar las cosas que son realmente importantes en la vida y aunque no lo creas, contigo aprendí a ser agradecido.

A mi hermanita, que no solo ha sido mi hermana, ha sabido ser mi dolor de cabeza y mi razón para esforzarme siempre un poco más. Yo quería un hermano y la vida me regalo una hermana que vale por tres… yo sé que cada vez que peleamos es un muestra de amor. Yo sé me quieres como yo a ti.

Agradecimientos

A mi tío Robertico, que aunque no llevas mi sangre eres mi familia, porque estuviste a mi lado no solo en los buenos momentos, también en los peores y supiste aconsejarme como si fuera un hijo.

A Ramiro o Hilario, ya da igual cómo te llames… has sido mi padre durante los últimos 15 años, me has educado como persona, me pegaste tus costumbres, tu curiosidad, tus hábitos de lectura, estudio e investigación… En fin, es como dice mi mamá: si fueras mi padre no nos pareceríamos tanto.

A Yindra, Mónica, Loge, Julio Omar, Maikel, Keiler, Jorge Luis, Tellíztico, Rigo, Yixander y al "rubio" César que han sido mi segunda familia durante todo este tiempo, nunca me ha faltado su apoyo, sus consejos han sido oportunos y su amistad la más sincera, más que mi cariño y agradecimiento tiene todo mi respeto y admiración.

A Ernesto por su amistad, sincera y desinteresada, en las buenas y en las malas, porque nunca dejó de confiar en mí… ni siquiera cuando yo lo hice.

Le agradezco sinceramente a mis amigos de todos estos años, a Pototo, al Fernand, a Erick, al Migue, al Puro, a Dasley, a Eudel, a Avilés, a Ernesto Antonio, a Osniel, al Michi, a Alexei, a Tommy a Manito, a Jorge y a Yuned… nunca olvidaré todo lo que vivimos juntos.

Agradezco a todos los buenos profesores que tuve, que me orientaron y supieron guiarme durante mi formación y sobre todo agradezco a este sueño de igualdad que hizo posible que un muchacho del monte hoy pueda graduarse en la Universidad.

Mi más sincero agradecimiento a todas las personas que hicieron posible este momento tan importante en mi vida y por favor le pido disculpas a quienes haya no pude mencionar por espacio y tiempo, han sido muchas personas, necesitaría un libro para agradecerle a todos los que me han apoyado.

Reisel

RESUMEN

La Universidad de las Ciencias Informáticas dispone de un sistema para el monitoreo de los recursos computacionales con los que cuenta la institución, el cual lleva por nombre SIMON. El mismo permite la visualización de un conjunto de reportes para determinar el aprovechamiento que se hace de dichos recursos. El principal objetivo de este trabajo de diploma, es la obtención de un modelo computacional que permita determinar el estado activo u ocioso de las estaciones de trabajo, a fin de poder evaluar el verdadero aprovechamiento de los recursos disponibles. Para ello se realiza el estudio de 4 técnicas de aprendizaje automatizado (k-NN, MLP, SVM y LADTree), con el fin de determinar cuál de ellas ofrece mejores resultados en la clasificación de las estaciones de trabajo. Se seleccionan las métricas que formarán parte de la base de conocimiento necesaria para el entrenamiento de las técnicas mencionadas. Una vez determinado el clasificador que ofrece los mejores resultados, y obtenido el modelo computacional, se procede a la incorporación del mismo al sistema SIMON. La obtención del modelo permite generar nuevos reportes que visualicen la información del porciento de estados activos y ociosos de las estaciones de trabajos, por días, momentos del día y por horas.

Palabras Claves: aprovechamiento, modelo computacional, aprendizaje automatizado.

Tabla de contenido

Introducción

En la actualidad la mayoría de las empresas y organizaciones del mundo generan grandes volúmenes de datos, aportando estos una gran cantidad de información. Para maximizar la utilidad de la información, esta se debe manejar de forma correcta y eficiente; ya que su uso es estrictamente estratégico en el posicionamiento de forma ventajosa de una empresa. El análisis de los datos y los valores acumulados a través de gráficos y tablas, contribuye a la toma rápida de decisiones que conlleven a aprovechar una oportunidad o solucionar alguna deficiencia o problema crítico en el negocio.

Los sistemas de inteligencia de negocios son soluciones informáticas muy utilizadas por las empresas hoy en día, pues por medio de la información almacenada pueden generar escenarios, pronósticos y reportes que apoyen la toma de decisiones, lo que se traduce en una ventaja competitiva para las empresas. La clave de los sistemas de inteligencia de negocios es la información y uno de sus mayores beneficios es la posibilidad de ser utilizados por los especialistas y directivos en la toma rápida de decisiones. En la actualidad la mayoría de las empresas u organizaciones cuentan con sistemas de inteligencia de negocios que son utilizados en diferentes áreas, tales como: ventas, marketing, finanzas y monitoreo.

La Universidad de la Ciencias Informáticas (UCI), cumpliendo con uno de los objetivos para la cual fue creada; producir software y servicios informáticos, se adentra cada vez más en la producción de software empresarial. El Sistema de Monitoreo del uso y explotación de las estaciones de trabajo en su primera versión (SIMON v1.0), es una solución de inteligencia de negocios desarrollada por la UCI, que tiene como objetivo apoyar el proceso de toma de decisiones en los temas relacionados con la explotación de estaciones de trabajo y el control de inventario de software y hardware. Con este fin utiliza datos recopilados por una aplicación cliente instalada en cada estación de trabajo.

La solución incluye varios reportes, tales como: explotación por utilización de mouse y teclado, tráfico de red, consumo de memoria RAM, utilización de CPU, ejecución de procesos por categorías e información de las actividades de los usuarios. Todos estos análisis son realizados desde el nivel de las estaciones de trabajo hasta niveles institucionales.

La versión existente de SIMON contiene una base de datos para la recolección de las métricas de las estaciones de trabajo que se encuentra implementada en MySQL; sistema gestor de base de datos

perteneciente a Oracle[1] y distribuido bajo la licencia GNU GPL[2]. Esto que implica que para incorporarlo en productos privativos se deba comprar una licencia específica que permita este uso. Esta base de datos se encuentra en tercera forma normal y almacena millones de tuplas diarias, incidiendo esto negativamente en el tiempo de respuesta de las consultas realizadas sobre la misma, en el proceso de extracción, transformación y carga de los datos hacia el ODS[3].

El principal inconveniente del sistema SIMON radica en que la evaluación del aprovechamiento de los recursos de cómputo, se realiza considerando únicamente el tiempo de encendido y la actividad de mouse y teclado. Específicamente, no dispone de un mecanismo para determinar si las estaciones de trabajo se encuentran en estado activo u ocioso, desconociéndose el aprovechamiento real de los recursos computacionales.

Por lo antes mencionado se define como **problema a resolver:** ¿Cómo determinar si las estaciones de trabajo se encuentran en estado activo u ocioso?

Se define como **objeto de estudio**: Técnicas de predicción, enmarcado en el **campo de acción:** Técnicas de aprendizaje automatizado para la clasificación.

Para dar solución al problema planteado se define como **objetivo general:** Obtener un modelo computacional para la clasificación de estaciones de trabajo en estado ocioso o activo.

Para dar cumplimiento al objetivo propuesto se trazan los siguientes **objetivos específicos:**

- Seleccionar un modelo computacional para la clasificación del estado de las estaciones de trabajo.
- Realizar la implementación del modelo computacional seleccionado.

---

[1] *Oracle Corporation*: una de las mayores compañías de software del mundo. Sus productos van desde bases de datos (Oracle) hasta sistemas de gestión.

[2] Licencia Pública General de GNU: licencia más ampliamente usada en el mundo del software, garantiza a los usuarios finales la libertad de usar, estudiar, compartir y modificar el software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

[3] Almacén Operacional de Datos: del inglés *Operational Data Store*, es una base de datos diseñada para integrar datos provenientes de diferentes fuentes con el fin de realizar operaciones adicionales sobre los datos.

- Incorporar el modelo para la clasificación de las estaciones de trabajo al sistema SIMON v2.0.

Para dar respuesta a los anteriores objetivos específicos se elaboraron las siguientes **tareas de investigación:**

- Estudio y selección de técnicas de aprendizaje automatizado para realizar inferencias de nuevas conclusiones a partir de los datos recopilados.
- Obtención de un modelo computacional para la clasificación de las estaciones de trabajo en estado ocioso o activo utilizando técnicas de aprendizaje automatizado.
- Determinación de cuál de las técnicas seleccionadas es presumiblemente mejor para la predicción del estado ocioso o no de las estaciones de trabajo mediante tablas de comparación.
- Diseño e implementación de una nueva base de datos no normalizada para la recolección de la información de las estaciones de trabajo.
- Implementación del proceso de Extracción, Transformación y Carga de los datos recopilados de las estaciones de trabajo.

El Trabajo de Diploma está estructurado de la siguiente manera: introducción, 3 capítulos, conclusiones, referencias bibliográficas, bibliografía y glosario de términos.

Capítulo 1: Fundamentación teórica. En este capítulo se abordan los principales conceptos relacionados con el dominio del problema, se exponen algunas características de los principales sistemas de monitoreo existentes y finalmente se fundamenta la utilización de las herramientas y tecnologías utilizadas para dar solución al problema.

Capítulo 2: Obtención del modelo para la clasificación. En este capítulo se exponen los pasos seguidos en la obtención del modelo para la clasificación de las estaciones de trabajo en estado ocioso o activo. Se muestran los resultados de la ejecución de las diferentes técnicas utilizando el software Weka. Se exponen además los principales elementos en la implementación del modelo obtenido.

Capítulo 3: Integración del modelo a SIMON. El capítulo aborda los principales cambios realizados en SIMON para la integración al sistema del modelo obtenido. Se expone además el diseño e

implementación de los nuevos reportes con la clasificación de las estaciones de trabajo incorporados a SIMON.

Capítulo 1: Fundamentación teórica

En el presente capítulo se hará un estudio sobre los principales sistemas de monitoreo de recursos computacionales existentes. Se exponen las principales características de las técnicas de aprendizaje automatizado que han sido propuestas para la obtención del modelo computacional. Finalmente se hace referencia a las herramientas, tecnologías y lenguajes a utilizar en la solución.

1.1 Aprendizaje Automatizado

El aprendizaje automatizado es un campo multidisciplinario de investigación centrado en los fundamentos matemáticos y aplicaciones prácticas de sistemas que aprenden, deciden y actúan. Un gran número de tecnologías modernas se sustentan en el aprendizaje automatizado, tales como el reconocimiento de voz, la robótica, la búsqueda en Internet, la Bioinformática, y más en general el análisis y modelado de datos de gran complejidad. (1). El aprendizaje automatizado es una rama de la Inteligencia Artificial que hace un amplio uso de métodos computacionales y estadísticos. Su principal objetivo es desarrollar técnicas que permitan a las computadoras aprender, permitiendo crear programas capaces de generalizar comportamientos a partir de una información suministrada en forma de ejemplos.

Dentro del aprendizaje automatizado, las técnicas de clasificación son utilizadas con el fin de determinar a qué clase o categoría pertenece un objeto, a partir de un conjunto de datos previamente conformado para el entrenamiento de la técnica o algoritmo a utilizar. En los problemas de clasificación, las técnicas de aprendizaje automatizado aportan grandes ventajas, pues proporcionan tasas de precisión comparables con las de un experto humano.

1.1.1 Clasificación

La idea de que la mente humana organiza su conocimiento usando el proceso de clasificación es generalizada. La clasificación es el proceso de la colocación de un objeto en un conjunto de categorías, sobre la base de las respectivas propiedades del objeto. En los problemas de clasificación se debe disponer de una colección de datos de ejemplo para cada una de las categorías que se desean utilizar, la cual es conocida como base de conocimiento. Luego de una etapa de entrenamiento, el sistema debe quedar ajustado de tal modo que ante nuevos ejemplos, el clasificador es capaz de ubicarlos en alguna de

las categorías existentes. Cuanto mayor sea el conjunto de datos etiquetados, mayor será la información potencial disponible y previsiblemente, mejor resultará el aprendizaje.

El proceso de clasificación se encuentra basado en 4 componentes fundamentales (2):

- **Clases:** es la variable categórica que representa la etiqueta a poner en el objeto después de la clasificación. Constituye la variable dependiente del modelo.
- **Predictores:** constituyen las variables independientes del modelo representando las características de los datos a ser clasificados y sobre los cuales se basa la clasificación.
- **Conjunto de entrenamiento:** es el conjunto de datos que contienen los valores de los dos componentes anteriores, y es usado para el entrenamiento del modelo con vistas a reconocer la clase apropiada, basándose en los predictores disponibles.
- **Conjunto de prueba:** contiene los nuevos datos que serán clasificados por el modelo construido anteriormente, permitiendo luego evaluar la exactitud del modelo.

A continuación se muestra la estrategia de diseño para la construcción de un modelo de clasificación, ver figura 1.1.

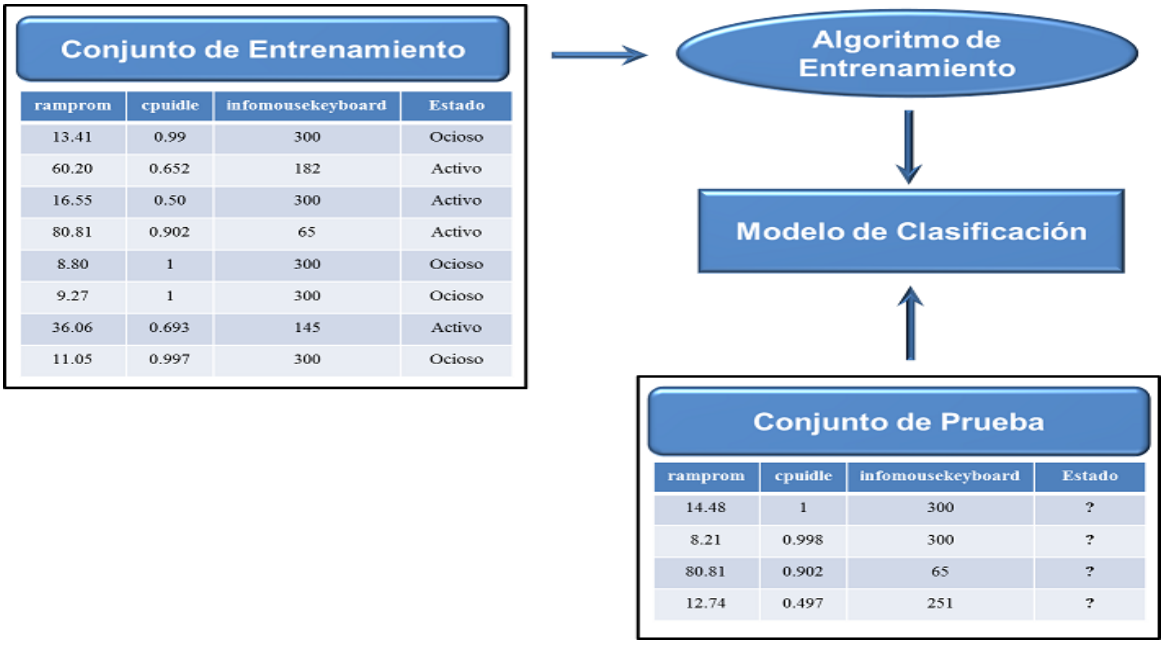


Figura 1.1: Estrategia para la construcción de un modelo de clasificación

Un paso importarte en la construcción de un clasificador, es el proceso de comparación con otros clasificadores con el objetivo de elegir el mejor. A continuación se listan algunos elementos claves que se deben tener en cuenta para realizar la comparación entre clasificadores (2).

- **Precisión en la predicción:** se refiere a la habilidad del modelo para clasificar correctamente cada nuevo objeto desconocido.
- **Velocidad:** se refiere a cuán rápido el modelo puede procesar los datos.
- **Escalabilidad:** se refiere principalmente a la habilidad del modelo de procesar cada vez más grandes volúmenes de datos, además puede referirse a la habilidad de procesar datos provenientes de diferentes campos.
- **Interpretabilidad:** ilustra la característica del modelo de ser fácil de entender e interpretar.
- **Simplicidad:** está relacionado con la habilidad del modelo de no ser complicado. En principio se debe escoger el modelo más simple que pueda resolver efectivamente un problema específico, como en la matemática, donde la demostración más elegante es la más sencilla.

Existen varias técnicas de aprendizaje automatizado que son utilizadas en los problemas de clasificación. Para el desarrollo de la presente investigación se seleccionaron las técnicas siguientes: k- Vecinos más Cercanos, Redes Neuronales, Máquinas de Soporte Vectorial y los Árboles de Decisión.

1.1.2 Método de k-Vecinos más Cercanos

K-Vecinos más Cercanos o k-NN (*k nearest neighbors* por sus siglas en inglés) es un método de clasificación supervisada que predice la clase a la que pertenece una nueva instancia a partir de la similitud a su objeto vecino más cercano. Este es un método de clasificación no paramétrico, que estima el valor de la función de densidad de probabilidad o directamente la probabilidad de que un elemento x pertenezca a la clase C_j, a partir de la información proporcionada por un conjunto de prototipos (2).

K-NN ha sido utilizado en numerosos problemas reales, tales como la correcta pronunciación de las palabras, realización de diagnósticos de enfermedades, reconocimiento de voz, simuladores de vuelo y pilotos automáticos, detectores de fraudes bancarios, expertos en juegos y en la clasificación de

secuencias de ADN y ARN[4].

Para la construcción del algoritmo se necesitan los siguientes elementos:

- Un conjunto de registros almacenados (juego de datos de entrenamiento).
- La distancia para calcular la similitud entre los objetos.
- El valor de k, es decir, el número (necesario) de los objetos que pertenecen al conjunto de datos sobre los cuales se va a lograr la clasificación de un nuevo objeto.

Basándose en estos tres requerimientos un nuevo objeto puede ser clasificado mediante la ejecución de los siguientes pasos:

- Calcular la distancia (similitud) entre todo el conjunto de datos y el nuevo objeto.
- Identificar los k objetos más cercanos, mediante el ordenamiento del conjunto de objetos, teniendo en cuenta la distancia calculada en el paso anterior.
- Asignar la etiqueta que es más frecuente dentro del conjunto de registros que es más cercano al objeto.

Limitaciones

- No construye modelos explícitamente como otros clasificadores.
- El proceso de clasificación requiere de un largo tiempo de ejecución, siendo además relativamente costoso computacionalmente.
- La predicción se basa en la información local, por lo tanto, es probable que sea influenciada por los valores atípicos extremos. (2)

Ventajas

- Es simple, flexible y fácil de implementar.
- Los investigadores han mostrado que la precisión de la clasificación de k-NN puede ser bastante fuerte y en muchos casos tan precisa como otros métodos de clasificación.

[4] Ácido ribonucleico**,** es un ácido nucleico formado por una cadena de ribonucleótidos. Está presente tanto en las células procariotas como en las eucariotas, siendo el único material genético de ciertos virus.

- Puede trabajar con límites de decisión de forma arbitraria (3).

1.1.3 Redes Neuronales Artificiales

Las redes neuronales artificiales constituyen el paradigma de aprendizaje y procesamiento automatizado inspirado en la forma en que funciona el sistema nervioso de los animales. Durante décadas, las redes neuronales aparecen como una tecnología práctica diseñada para resolver de manera exitosa muchos problemas en varios campos, tales como: Matemáticas, Estadísticas, Física, Ciencias de la Computación, Ingeniería y Biología (2).

Perceptrón Multicapa

Uno de los modelos más populares de las redes neuronales es el MLP (*Multilayer Perceptron* por sus siglas en inglés). Este constituye una red neuronal artificial formada por múltiples problemas que no son linealmente separables, lo cual es su principal limitación. MLP usando neuronas ocultas con funciones no lineales, es capaz de aproximar cualquier tipo de función continua y brindar excelentes resultados en las tareas de clasificación. Debido a su característica fundamental; la habilidad de aprender a partir de un conjunto de datos de entrenamiento “con o sin un maestro”, las redes neuronales son aplicadas en diversas áreas como el modelado, análisis de series de tiempo, patrones de reconocimiento y el procesamiento de señales (2).

Las capas pueden clasificarse en tres tipos:

- **Capa de entrada:** constituida por aquellas neuronas que introducen los patrones de entrada en la red. En estas neuronas no se produce procesamiento.
- **Capas ocultas:** formada por aquellas neuronas cuyas entradas provienen de capas anteriores y cuyas salidas pasan a neuronas de capas posteriores.
- **Capa de salida:** neuronas cuyos valores de salida se corresponden con las salidas de toda la red.

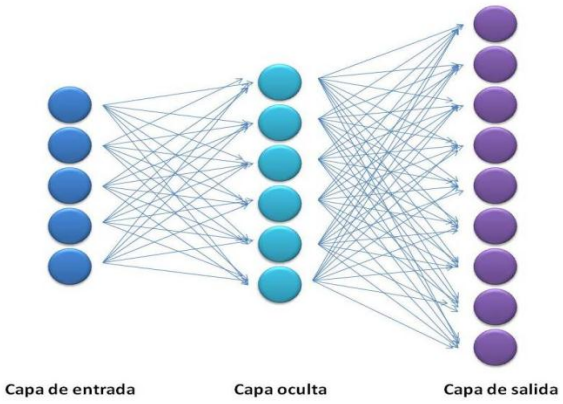


Figura 1.2: Modelo del MLP

MLP se caracteriza por presentar una no linealidad en la salida, un conjunto de capas de neuronas ocultas y un alto grado de conectividad. Utiliza un algoritmo de retro-propagación del error, que está basado en la regla de aprendizaje por corrección de error. Su operación consta de dos fases, una directa y una inversa o de retroceso. En la fase directa, se ingresa el patrón de actividad en la capa de entrada de la red (vector de entrada), que recorre todas las capas subsiguientes. Se obtiene la respuesta real de la red en la capa de salida. En la fase inversa, los pesos sinápticos son ajustados de acuerdo con la regla de corrección del error (4).

Limitaciones

- El MLP no extrapola bien, es decir, si la red se entrena mal o de manera insuficiente las salidas pueden ser imprecisas.
- La existencia de mínimos locales de la función de error dificulta considerablemente el entrenamiento, pues una vez alcanzado un mínimo, el entrenamiento se detiene aunque no haya alcanzado la taza de convergencia fijada (5).

Ventajas

- **Aprendizaje adaptativo:** las redes neuronales aprenden a llevar a cabo ciertas tareas mediante un entrenamiento con ejemplos ilustrativos.
- **Tolerancia a fallos:** comparados con los sistemas computacionales tradicionales, los cuales pierden su funcionalidad en cuanto sufren un pequeño error de memoria , en las redes neuronales,

si se produce un fallo en un pequeño número de neuronas, aunque el comportamiento del sistema se ve influenciado, no sufre una caída repentina (5).

1.1.4 Máquinas de Soporte Vectorial

Las Máquinas de Soporte Vectorial o SVM (*Support Vector Machine* por sus siglas en inglés), constituyen máquinas lineales, equipadas con funciones específicas, y basadas en el método de minimización de riesgo estructural, y la teoría del aprendizaje estadístico. Consecuentemente, las SVM pueden proveer una buena ejecución generalizada en los problemas de reconocimiento de patrones, sin incorporar un conocimiento del dominio del problema, lo que le da una característica única entre las máquinas de aprendizaje (2).

Las SVM pertenecen a la familia de clasificadores lineales debido a que inducen separadores lineales o hiperplanos en espacios de características de muy alta dimensionalidad con un sesgo inductivo muy particular (6). Sin embargo, la formulación matemática de las SVM varía dependiendo de la naturaleza de los datos; es decir, existe una formulación para los casos lineales y, por otro lado, una formulación para casos no lineales. De manera general para los problemas de clasificación, las SVM buscan encontrar un hiperplano óptimo que separe las clases existentes, ver figura 1.3.

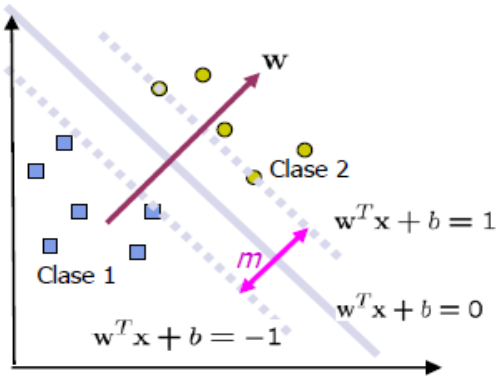


Figura 1.3: Separación de las clases en SVM

En la actualidad, las SVM pueden ser utilizadas para resolver problemas tanto de clasificación como de regresión. Algunas de las aplicaciones de clasificación o reconocimiento de patrones son: reconocimiento de firmas, reconocimiento de imágenes y categorización de textos. Por otro lado, las aplicaciones de regresión incluyen predicción de series de tiempo y problemas de inversión en general.

Limitaciones

- Su modelo de “caja negra” no permite que estas sean fáciles de explicar y entender.
- Requieren del ajuste de grandes parámetros de configuración.

Ventajas

- Con una apropiada selección del tipo de SVM y el kernel a utilizar, estas permiten obtener un modelo computacional con un alto porciento de precisión.
- Se escalan relativamente bien para datos en espacios dimensionales altos.
- El compromiso entre la complejidad del clasificador y el error puede ser controlado explícitamente.
- Datos no tradicionales como cadenas de caracteres y árboles pueden ser usados como entrada a la SVM, en vez de vectores de características.

1.1.5 Árboles de Decisión

Los árboles de decisión constituyen una de las técnicas más utilizadas en los problemas de clasificación, debido a que su predicción es muy efectiva en comparación con otros métodos de aprendizaje automatizado. El modelo de predicción obtenido por esta técnica es representado mediante un árbol, llamado árbol de decisión. Los árboles de decisión son utilizados para predecir la pertenencia de los objetos a diferentes categorías o clases, teniendo en cuenta los valores que le correspondan a sus atributos (3). Entre sus áreas de aplicación se destacan la Medicina (diagnóstico), la Informática (estructuras de datos), la Botánica y la Psicología.

ADTree

Los Árboles de Decisión Alternativo o ADTree (*Alternative Decision Tree* por sus siglas en inglés), constituyen una técnica de clasificación exitosa, que combina los árboles de decisión con la precisión de predicción de la técnica de impulso (conocida como *boosting*), en un conjunto de reglas de clasificación interpretables. Los ADTree proporcionan un mecanismo para combinar las débiles hipótesis generadas durante el impulso en una sola representación interpretable. El algoritmo de impulso utilizado por los ADTree llamado AdaBoost, ha sido usado exitosamente en los árboles de decisión produciendo buenos

resultados en la clasificación (7).

Los grandes resultados obtenidos son logrados debido a que en cada iteración de impulso se añaden tres nodos al árbol; un nodo divisor que intenta dividir conjuntos de instancias en subconjuntos puros y dos nodos de predicción, uno para cada uno de los subconjuntos del nodo divisor. La posición de este nuevo nodo divisor se determina mediante el examen de todos los nodos de predicción eligiendo la posición con mayor puntuación alcanzada (8).

LADTree

Los LADTree (*Logitboost Alternative Decision Tree* por sus siglas en inglés), constituyen un ejemplo de árboles de decisión alternativo que utilizan como estrategia de impulso el algoritmo LogitBoost[5]. Los LADTree producen ADTree capaces de tratar problemas de clasificación que contienen más de dos clases. Este extiende el algoritmo original ADTree al caso multi-clase, al dividir el problema en varios problemas de dos clases. Al igual que con el algoritmo original, un solo atributo de prueba se elige como el nodo divisor para el árbol en cada iteración. El objetivo es ajustar el valor de la media de las instancias, en un subgrupo en particular, al minimizar el valor de mínimos cuadrados entre ellos (7).

Ventajas

- Permiten el uso de datos numéricos y categóricos sin ninguna restricción.
- Son robustos, rápidos y procesan grandes cantidades de datos.
- No requieren del ajuste de grandes cantidades de parámetros de configuración.
- Requieren pequeñas cantidades de datos de entrenamiento en comparación con otras técnicas de clasificación.
- Su modelo de “caja blanca” permite que sean fáciles de entender y de explicar. (2)

[5] Algoritmo de impulso formulado por Jerome Friedman, Trevor Hastie, y Robert Tibshirani. Optimiza la probabilidad (más estándar) binomial en lugar de la función de pérdida exponencial utilizada en AdaBoost.

1.2 Inteligencia de Negocios

La inteligencia de negocios o BI (*Business Intelligence* por sus siglas en inglés), se define como el proceso de analizar los bienes o datos acumulados en las empresas, con el objetivo de extraer cierta inteligencia o conocimiento de ellos. Esta además es el conjunto de productos y servicios que permiten a los usuarios finales acceder y analizar la información de manera rápida y sencilla, para la toma de decisiones de negocio a nivel operativo, táctico y estratégico (9).

La inteligencia de negocios se refiere al uso de la tecnología para recolectar y usar efectivamente la información, con el fin de mejorar las operaciones del negocio. Cuenta con un conjunto de herramientas de reporte, consulta y análisis de datos, que ayudan a los usuarios a navegar a través de la información y obtener conclusiones valiosas que apoyan el proceso de toma de desiciones.

1.3 Proceso de Extracción, Transformación y Carga (ETL)

El término ETL (*Extract-Transform-Load* por sus siglas en inglés) es el proceso de Extracción, Transformación y Carga de los datos en una entidad. Desde el punto de vista tecnológico constituye un factor crítico de éxito en una solución de inteligencia de negocios; pues automatiza y simplifica procesos muchas veces complejos o demandantes en tiempo. Este es el proceso que organiza el flujo de los datos entre diferentes sistemas en una organización y aporta los métodos y herramientas necesarias para mover datos desde múltiples fuentes, reformatearlos, limpiarlos y cargarlos en otra base de datos, mercado de datos o almacén de datos.

Un sistema ETL bien diseñado, extrae la información de los sistemas fuentes y hace cumplir las normas de calidad y la consistencia de los datos. Además conforma datos de forma tal que aún siendo de fuentes diferentes puedan ser usados juntos y los entrega en un formato de presentación, listo para que los desarrolladores puedan construir aplicaciones que permitan a los usuarios finales la toma de decisiones (10).

Las funciones específicas de un proceso ETL son (11):

Extracción de la información desde una o diferentes bases de datos, archivos de texto y otras fuentes. El proceso de extracción puede incluir tareas de validación y el descarte de los datos que no cumplen con los patrones y reglas establecidas.

Transformación de los datos obtenidos para conocer las necesidades técnicas y del negocio requeridas. La transformación implica tareas como la conversión de tipos de datos, realización de algunos cálculos, filtrado de datos irrelevantes, así como el resumen de datos.

Carga de los datos transformados en la base de datos destino. En dependencia de los requerimientos, la carga puede sobrescribir la información existente, o puede adicionar nueva información cada vez que es ejecutada.

1.4 Sistemas de Monitoreo

Los sistemas de monitoreo son aplicaciones que permiten verificar sistemáticamente el desempeño y disponibilidad de los principales componentes de un sistema de cómputo. El monitoreo de recursos computacionales permite detectar afectaciones o cambios en los principales elementos de software y hardware, siendo de gran utilidad para el apoyo a la toma de decisiones. Para el desarrollo de la presente investigación se realizó un estudio de los principales sistemas de monitoreo existentes, a fin de identificar cuál o cuáles de ellos permite realizar la clasificación de las estaciones de trabajo según su estado.

1.4.1 Ganglia

Ganglia es un sistema distribuido de monitoreo de recursos computacionales para sistemas de computación de alto rendimiento. El mismo está escrito en C, Perl, PHP, Python, es multiplataforma y es publicado bajo la licencia *Berkeley Software Distribution* (BSD). Su arquitectura está basada en un diseño jerárquico y escalable que permite monitorear infraestructuras, incluso con más de 2000 nodos (12). Ha sido adaptado a un amplio conjunto de sistemas operativos y arquitecturas de procesadores. Actualmente está en uso en miles de instituciones de todo el mundo.

El sistema está conformado por dos servicios responsables de la recolección, el almacenamiento de los datos y una interfaz web para la visualización de la información.

Ganglia Monitoring Daemon (gmond): es el servicio responsable de la recolección de métricas reales tales como CPU, memoria, disco, red y datos acerca de los procesos activos. Debe ser instalado en cada nodo que desee ser monitoreado por el sistema.

Ganglia Meta Daemon (gmetad): es el servicio responsable de recopilar métricas de otras fuentes

gmetad o gmond y almacenar la información. Proporciona un mecanismo de consultas para recolectar información específica acerca de los grupos de máquinas y admite organización jerárquica, permitiendo la creación de dominios de monitoreo.

Ganglia PHP Web Front-end (gweb): proporciona una vista de la información recolectada en tiempo real a través de páginas web dinámicas, permitiendo visualizar los datos más significativos para los administradores de sistemas.

1.4.2 Pandora FMS (Sistema de Monitoreo Pandora Flexible)

Pandora FMS es una solución de software para el monitoreo de las redes informáticas. Pandora FMS permite monitorizar de forma visual el estado y rendimiento de varios parámetros de los diferentes sistemas operativos, servidores, aplicaciones, firewalls, servidores proxy, bases de datos, servidores web y routers. Es altamente escalable (hasta 2000 nodos con un solo servidor) y completamente basado en la web.

Tiene un sistema de control de acceso multiusuario y una gran cantidad de informes gráficos definidos por el usuario, así como pantallas de control. Se encuentra publicado bajo los términos de la Licencia Pública General GNU, por lo que Pandora FMS es software libre (13).

1.4.3 Hyperic HQ

Hyperic HQ es un sistema de monitoreo y administración de recursos computacionales. Es considerado por los especialistas en el tema, uno de los mejores sistemas de monitoreo existente.

Hyperic ofrece dos versiones de su producto:

- **Hyperic HQ**: oferta de código abierto bajo la Licencia Pública General de GNU v2.
- **vFabric Hyperic:** presenta todas las capacidades de la versión de código abierto, además de la automatización avanzada y funciones de control de la gestión de aplicaciones web a gran escala. Está disponible como versión de prueba gratis para su descarga desde Hyperic bajo una Licencia Comercial. La versión de pruebas se limita a 50 plataformas administradas, y por lo general expira de 30 a 45 días.

Los principales elementos de su arquitectura son los servidores HQ, que proporcionan una administración y persistencia centralizada, el Agente HQ, que facilita el monitoreo y control por plataforma.

- **Agente Hyperic:** es responsable de recolectar información periódicamente de la estación de trabajo donde ha sido instalado y enviarla al servidor Hyperic.
- **Servidor Hyperic:** recibe la información recolectada por los agentes Hyperic y la almacena en la base de datos Hyperic. Es capaz de detectar problemas y notificarlos.
- **Portal Hyperic:** es una interfaz gráfica de usuario altamente personalizable para el servidor Hyperic. La página de inicio del portal es el *Dashboard Hyperic*, que contiene módulos de función configurables, ofreciendo una visión general del inventario de software, componentes problemáticos y gráficos con las métricas más importantes.

Hyperic HQ permite supervisar entornos computacionales, alcanzando decenas de miles de recursos gestionados, desde las interfaces de CPU y de red, hasta servidores de aplicaciones y bases de datos. Su capacidad de auto-descubrimiento, le permite iniciar la supervisión y administración de entornos heterogéneos apenas transcurridos unos minutos desde la instalación, tanto en entornos físicos como virtuales. La efectividad del auto-descubrimiento radica en la librería Sigar de Hyperic (*System Information Gatherer* por sus siglas en inglés), un API[6] multiplataforma para la recolección de datos de estaciones de trabajo. Los agentes HQ Sigar se utilizan para descubrir un amplio conjunto de información del sistema, incluyendo la velocidad de la CPU, tamaño de la RAM, versión del sistema operativo y direcciones IP (14).

1.4.4 SIMON

SIMON es una solución de inteligencia de negocios que tiene como objetivo apoyar el proceso de toma de decisiones en los temas relacionados con la explotación de estaciones de trabajo y el control de inventario de software y hardware, usando para este fin datos recopilados por un programa cliente. La solución incluye análisis de diversas áreas como explotación por utilización de mouse y teclado, tráfico de red, consumo de memoria RAM, utilización de CPU, ejecución de procesos por categorías e información de las

[6] Interfaz de programación de aplicaciones o *Application Programming Interface* (por sus siglas en inglés), es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizados por otro software como una capa de abstracción.

actividades de los usuarios. Todos estos análisis desde nivel de estaciones de trabajo hasta niveles institucionales.

El sistema está divido en tres tipos de aplicaciones:

- Una primera aplicación cliente-servidor (que lleva por nombre T-arenal) cuyo módulo cliente se ejecuta en las estaciones de trabajo para recoger, cada cierto intervalo de tiempo, una muestra del estado de los recursos de hardware y software de la PC y enviarlas al servidor. El módulo servidor se encarga de recibir las muestras y almacenarlas en una base de datos.
- Una segunda aplicación que se encarga, a partir de los datos recopilados por la primera aplicación, del proceso de ETL de los datos hacia un ODS para un posterior análisis.
- Una tercera aplicación web que permite visualizar variedades de tipos de gráficas, para analizar los datos que se encuentran en el ODS poblado por la segunda aplicación.

1.5 Herramientas, Tecnologías y Metodologías de Desarrollo

Un elemento fundamental en todo proceso de desarrollo de software, es sin duda la buena selección de las herramientas y tecnologías a utilizar. A continuación se exponen las principales características de las herramientas y tecnologías seleccionadas para el desarrollo de la presente investigación.

1.5.1 Sistema Gestor de Base de Datos

Un Sistema Gestor de Base de Datos (SGBD) es un sistema de software que permite la definición de bases de datos; así como la elección de las estructuras de datos necesarias para el almacenamiento y búsqueda de los datos, ya sea de forma interactiva o a través de un lenguaje de programación. Se componen de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. El propósito general de los SGBD es el de manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información (15). Existen varios SGBD los cuales se dividen en dos grupos: los de licencia libre como PostgreSQL y los de licencia propietaria como MySQL, Oracle y SQL server.

A continuación se enuncian los principales objetivos de los SGBD (15).

- Evitar la redundancia de los datos.
- Mejorar los mecanismos de seguridad de los datos y la privacidad.
- Asegurar la independencia de los programas y los datos, es decir, la posibilidad de modificar la estructura de la base de datos (esquema) sin necesidad de modificar los programas de las aplicaciones que manejan esos datos.
- Mantener la integridad de los datos realizando las validaciones necesarias cuando se realicen modificaciones en la base de datos.
- Mejorar la eficacia de acceso a los datos, en especial en el caso de consultas imprevistas.

PostgreSQL 9.1

PostgreSQL es el líder en sistemas de bases de datos de código abierto, con una comunidad mundial de miles de usuarios y contribuyentes, además de una docena de empresas y organizaciones. Cuenta con más de 15 años de desarrollo activo y una arquitectura probada que se ha ganado una sólida reputación de fiabilidad. Presenta soporte completo para claves foráneas, uniones, vistas, disparadores y procedimientos almacenados (en varios idiomas). PostgreSQL provee algunas de las más avanzadas características empresariales de cualquier base de datos de código abierto. El conjunto de características maduras que posee PostgreSQL, permiten que este no sólo rivalice con sistemas de bases de datos propietarios, sino que los supere en cuanto a características avanzadas, extensibilidad, seguridad y estabilidad (16).

Principales Características:

- **Atomicidad:** esta característica asegura que las operaciones se hayan realizado o no, y por lo tanto ante un fallo del sistema no pueden quedar a medias.
- **Consistencia:** esta característica asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper las reglas y directrices de integridad de la base de datos.
- **Aislamiento:** esta característica asegura que una operación no puede afectar a otras. Esto asegura que dos transacciones sobre la misma información nunca generarán ningún tipo de error.

- **Durabilidad:** esta característica asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.

PgAdmin III

PgAdmin III es la herramienta de código abierto de administración por excelencia para las bases de datos PostgreSQL. Algunas de sus características son: el soporte completo para UNICODE[7], edición rápida de consultas y datos, así como la inclusión de soporte para todos los tipos de objetos de PostgreSQL. Incluye una interfaz gráfica de administración, una herramienta para el trabajo con SQL, un editor de código de funciones y procedimientos, entre otras funcionalidades. PgAdmin III está diseñado para darle respuesta a las necesidades de la mayoría de los usuarios, desde la escritura de consultas simples en SQL hasta el desarrollo de bases de datos complejas. La interfaz gráfica soporta todas las características presentes de PostgreSQL, por lo que se puede hacer la administración fácilmente. Está disponible en más de 30 lenguajes y para varios sistemas operativos (17).

1.5.2 Plataforma de Aprendizaje Automatizado Weka 3.6.8

La plataforma Weka, cuyo nombre proviene de *Waikato Environment for Knowledge Analysis*, es una plataforma de aprendizaje automatizado y minería de datos desarrollada en java. La plataforma está disponible libremente bajo la Licencia Pública General de GNU. Soporta varias tareas de minería de datos, especialmente, procesamiento de datos, clasificación, regresión, visualización, y selección. Todas las técnicas de Weka se fundamentan en la asunción de que los datos están disponibles en un fichero plano o una relación, en la que cada registro de datos está descrito por un número fijo de atributos (normalmente numéricos o nominales, aunque también soporta otros tipos). Weka también proporciona acceso a bases de datos vía SQL gracias a la conexión JDBC (*Java Database Connectivity*) (18).

Algunas Características:

- Contiene varios paquetes para el aprendizaje automatizado.
- Especializado en la minería de datos.

[7] Norma que proporciona un número único para cada carácter, sin importar la plataforma, programa o idioma.

- Es fácil de utilizar por un principiante gracias a su interfaz gráfica de usuario.
- Es muy portable porque está completamente implementado en Java y puede correr en casi cualquier plataforma.
- Se pueden probar diferentes algoritmos de aprendizaje al mismo tiempo y comparar los resultados.

1.5.3 Lenguaje de Programación

Un lenguaje de programación es un conjunto de reglas y condiciones que permiten crear programas o aplicaciones dentro de una computadora. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones (19).

En el mundo de la informática existen varios lenguajes de programación, algunos más potentes que otros o dirigidos específicamente a algún tipo de programación, ejemplos de estos son C, C++, PHP y Java.

Java

Java es un lenguaje de programación de propósito general, concurrente, basado en clases, y orientado a objetos, que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Cuenta con una máquina virtual que permite que las soluciones sean ejecutadas en diferentes entornos de hardware y software. La versatilidad, eficacia, portabilidad de plataformas y seguridad de Java lo convierten en la tecnología ideal para la informática de redes. Su uso se extiende desde dispositivos móviles, sistemas empotrados, aplicaciones de servidor entre otras. Java es un lenguaje de código abierto que se rige bajo la Licencia Pública General (GPL) de GNU (20).

Java permite a los desarrolladores (20):

- Escribir software en una plataforma y ejecutarla virtualmente en otra.
- Crear programas que se puedan ejecutar en un explorador y acceder a servicios web disponibles.
- Desarrollar aplicaciones de servidor para foros en línea, almacenes, encuestas, procesamiento de formularios HTML y mucho más.

- Combinar aplicaciones o servicios que utilizan el lenguaje Java para crear aplicaciones o servicios con un gran nivel de personalización.
- Escribir aplicaciones potentes y eficaces para teléfonos móviles, procesadores remotos, productos de consumo de bajo coste y prácticamente cualquier otro dispositivo con un latido digital.

1.5.4 Lenguaje de Modelado UML 2.1

El Lenguaje de Modelado Unificado o UML (*Unified Modeling Language* por sus siglas en inglés) se define como un lenguaje que permite especificar, visualizar y construir los artefactos de los sistemas de software. Es un sistema de notación destinado a los sistemas de modelado que utilizan conceptos orientados a objetos. Ayuda a los profesionales a visualizar, comunicar y aplicar sus diseños para proporcionar un entorno de modelado visual que une las tecnologías de software y las necesidades de comunicación. Una de las metas principales de UML es avanzar en el estado de la integración institucional proporcionando herramientas de interoperabilidad para el modelado visual de objetos (21).

1.5.5 Herramientas Case

Las herramientas CASE (*Computer Aided Software Engineering*, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo costes en términos de tiempo y dinero. Estas herramientas ayudan en todos los aspectos del ciclo de vida de desarrollo de software, en tareas como el proceso de realización del diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras (22).

Principales Objetivos:

- Permitir la aplicación práctica de metodologías estructuradas, las cuales permiten agilizar el trabajo al ser realizadas con una herramienta.
- Facilitar la realización de prototipos y el desarrollo conjunto de aplicaciones.
- Mejorar y estandarizar la documentación.
- Aumentar la portabilidad de las aplicaciones.

- Permitir un desarrollo y un refinamiento visual de las aplicaciones, mediante la utilización de gráficos.

Entre las herramientas CASE más conocidas se encuentran EASYCASE, WINPROJECT, RATIONAL ROSE y VISUAL PARADIGM FOR UML.

Visual Paradigm for UML 6.4

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del proceso de desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones con calidad y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación (22).

De manera general, esta herramienta de modelado ofrece:

- Entorno de creación de diagramas para UML.
- Diseño centrado en casos de uso y enfocado al negocio.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad en múltiples plataformas.

1.5.6 Entorno de Desarrollo Integrado Eclipse 3.8

El Entorno de Desarrollo Integrado (IDE) Eclipse es un entorno de desarrollo de Java que emplea módulos para proporcionar mayor funcionalidad, a diferencia de otros entornos monolíticos donde las funcionalidades están generalmente prefijadas, las necesite el usuario o no. El mecanismo de módulos permite que el entorno de desarrollo soporte otros lenguajes de programación además de Java, así como introducir otras aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo (23).

Entre sus principales ventajas se encuentran las siguientes:

- Es una herramienta de código abierto.

- Soporta la construcción de una variedad de herramientas para el desarrollo de aplicaciones.
- Soporta el desarrollo de aplicaciones basadas en GUI[8] y CLI[9].
- Soporta tecnologías que manipulan diferentes tipos de archivos como por ejemplo Java, C, PHP, C++, HTML, GIF, etc.
- Permite su ejecución en múltiples sistemas operativos incluyendo Windows y Linux.

1.5.7 Herramientas de Inteligencia de Negocios

Las herramientas de inteligencia de negocios son aplicaciones diseñadas con el fin de colaborar con el procesamiento de la información valiosa de las empresas. Entre las principales herramientas destacan los productos comerciales como Crystal Reports, Oracle Bussines Intelligence, Microsoft Business Intelligence y los de licencia libre como Freereporting y Pentaho Open Source Business Intelligence.

Pentaho BI Platform / Server 3.8

Uno de los procesos más importantes en todo sistema de inteligencia de negocios es la visualización de los datos, a través de reportes, tablas y gráficas. La Plataforma Pentaho BI provee el soporte y la infraestructura necesaria para crear soluciones de inteligencia empresarial a problemas de negocios. El marco proporciona los servicios básicos, incluidos autenticación, registro, auditoría, servicios web y motor de reglas. La plataforma también incluye un motor de solución que integra reportes, análisis y tableros de comandos. Su arquitectura basada en plugin[10] permite que toda, o parte de la plataforma, esté embebida en aplicaciones de terceros (24).

Algunas de sus ventajas son:

- Integración con procesos de negocio.

[8] Interfaz Gráfica de Usuario o *Graphical User Interface* (por sus siglas en inglés), es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información.

[9] Interfaz de Línea de Comandos o *Command Line Interface* (por sus siglas en inglés), es un método que permite a las personas dar instrucciones a algún programa informático por medio de una línea de texto simple.

[10] Conjunto de componentes de software que agrega capacidades específicas para una aplicación de software más grande.

- Administra y programa reportes.
- Administra seguridad de usuarios.

1.5.8 Herramientas ETL

Las herramientas ETL proporcionan la consolidación de los datos para la construcción de bases de datos permanentes, utilizadas en el análisis o la generación de informes, por lo que constituye un factor crítico contar con una herramienta ETL que permita reducir tiempos, costos de desarrollo y el mantenimiento de los procesos existentes.

Pentaho Data Integration 4.2.1

Pentaho Data Integration también conocido como Kettle, ofrece poderosas capacidades de Extracción, Transformación y Carga (ETL) mediante un enfoque innovador de metadatos; permitiendo a las organizaciones extraer datos de las fuentes más complejas y heterogéneas, para luego analizar los datos de las aplicaciones críticas del negocio. Cuenta con un ambiente intuitivo, gráfico y una arquitectura probada basada en estándares (25). La integración de datos de Pentaho es la elección más demandada por las empresas hoy en día.

Características de Pentaho Data Integration:

- Plataforma: Windows y Linux.
- GUI: Interfaz con indicadores visuales de transformación. Informes disponibles de la capa de metadatos.
- Código: Aplicación 100% Java con transformaciones avanzadas en Java Script mediante una interfaz empotrada. Diseño orientado a metadatos.
- Licencia: *Mozilla Public License*.
- Código fuente: El código fuente está disponible.
- Soporte: Existe un foro, un buscador de problemas y la comunidad Pentaho con varios artículos técnicos.
- Conectividad: Soporta Oracle, DB2, SQL Server, MySQL, PostgreSQL, entre otros.

1.5.9 Metodología de Desarrollo

El desarrollo de software engloba una serie de bases, herramientas y métodos que necesitan imperativamente de una metodología, con el objetivo de encausar cada uno de estos engranajes en lo que sería un proceso coordinado y acorde al fin que se persigue. Una metodología de desarrollo no es más que un conjunto de técnicas, procedimientos, herramientas y un soporte documental que permiten de manera más fácil el desarrollo de software que funcione.

Metodología de Desarrollo de Software OpenUP

OpenUP es un proceso de desarrollo de software de código abierto que aplica propuestas iterativas e incrementales dentro del ciclo de vida, abrazando una filosofía programática y ágil que se centra en la naturaleza colaborativa del desarrollo de software. El esfuerzo personal en un proyecto de OpenUP se organiza en micro-incrementos. Estos representan unidades cortas de trabajo que producen un ritmo constante y medible del progreso del proyecto. Estos micro-incrementos proporcionan un circuito de retroalimentación extremadamente corto, que impulsa las decisiones de adaptación dentro de cada iteración (26).

El ciclo de vida de OpenUP se encuentra estructurado en 4 fases: Inicio, Elaboración, Construcción y Transición. Además provee a los interesados y a los miembros del equipo, puntos de visibilidad y de decisión a través del proyecto, permitiendo esto una supervisión efectiva, así como realizar la toma de decisiones en el momento adecuado.

OpenUP se caracteriza por cuatro principios básicos que se soportan mutuamente:

- Balance para confrontar las prioridades para maximizar el valor para los stakeholders[11].
- Colaboración para alinear los intereses y un entendimiento compartido.
- Enfoque en particular de la arquitectura para facilitar la colaboración técnica, reducir los riesgos y minimizar excesos y trabajo extra.
- Evolución continua, para reducir riesgos, demostrar resultados y obtener retroalimentación de los clientes.

[11]Representa a un grupo de interés cuyas necesidades se deben satisfacer o que está materialmente afectado por el resultado del proyecto.

Los beneficios de esta metodología se destacan de la siguiente forma: permite disminuir las probabilidades de fracaso en los proyectos pequeños, las detecciones tempranas de errores, evita la elaboración de documentación innecesaria y permite un enfoque centrado en el cliente y con iteraciones cortas.

Por lo antes expuesto se considerada esta metodología como las más factible y ajustable al equipo de trabajo para el desarrollo de la presente solución, debido a que está dirigida a proyectos pequeños y logra colocar a los individuos y su interacción, por encima de los procesos y las herramientas. Teniéndose en cuenta además que por sus características es muy utilizada para el desarrollo de los proyectos en el centro DATEC.

Conclusiones del Capítulo

En el presente capítulo se realizó una revisión del estado del arte de los principales sistemas de monitoreo existentes, concluyendo que la mayoría de los sistemas de código abierto no son capaces de realizar análisis exhaustivos sobre los datos y no disponen de funcionalidades como la extrapolación de datos para realizar predicción. Se abordaron las técnicas de aprendizaje automatizado, exponiendo sus principales características y ventajas, lo que se utilizará como referencia para la obtención de un modelo para la clasificación del estado de las estaciones de trabajo. Los clasificadores seleccionados resultaron k-NN, MLP, SVM y LADTree, todos documentados o referenciados en la mayoría de los libros y artículos de la literatura consultada. Para la obtención del modelo computacional se decide utilizar el software de aprendizaje automatizado “Weka”, el mismo incluye implementaciones de las técnicas a utilizar y permite realizar pruebas de validación utilizando diferentes configuraciones y conjuntos de datos. Además se describieron las principales herramientas, tecnologías y metodologías a utilizar para dar solución al problema planteado.

Capítulo 2: Obtención del modelo para la clasificación

En el presente capítulo se realiza la selección y validación de la idoneidad de las variables a partir de las gráficas 2D de las matrices de dispersión y la prueba estadística “Lambda de Wilks”. Además se muestran los resultados y las tablas de comparación obtenidas a partir de las pruebas de validación realizadas a los clasificadores seleccionados, variando los parámetros de configuración de los mismos. Finalmente se selecciona e implementa el modelo computacional más conveniente para la resolución del problema planteado en la presente investigación.

2.1 Selección de las variables

Para la conformación de la base de conocimiento es necesaria la correcta selección e identificación de las variables. En el desarrollo de la presente investigación han sido seleccionadas tres variables independientes y una variable dependiente. Las variables independientes seleccionadas han sido identificadas como **ramprom**, **cpuidle** e **infomousekeyboard**, haciendo referencia al uso promedio de memoria RAM, porciento de inactividad del CPU y tiempo en milisegundos sin actividad de mouse y teclado, respectivamente.

La variable dependiente ha sido identificada como **Estado**, haciendo referencia al estado ocioso o activo de las estaciones de trabajo. El tipo de dato de la misma es **No métrico** o **Cualitativo**, teniendo en cuenta que se refiere a una característica o propiedad que identifica a la estación de trabajo. Es además escala de medida **Nominal** al ser codificados los estados ocioso y activo en valores binarios **{0,1}** respectivamente, y **Discreta** debido a que cada característica o propiedad tiene carácter excluyente sobre las demás.

2.2 Obtención de la base de conocimiento

Siendo seleccionadas las variables necesarias se debe proceder a la conformación de la base de conocimiento. Para ello fue necesaria la recopilación de las métricas previamente identificadas y clasificar las estaciones de trabajo según su estado. Fueron identificados tres casos para la clasificación de las estaciones de trabajo.

Estado activo: Existe actividad de mouse y teclado durante el periodo de observación.

Estado activo: No existe actividad de mouse y teclado pero se identifica un bajo porciento de inactividad del CPU.

Estado ocioso: No existe actividad de mouse y teclado en el periodo de observación, identificándose además un alto porciento de inactividad del CPU.

Los datos para la conformación de la base de conocimiento fueron recopilados en un entorno controlado con 30 estaciones de trabajo y fueron clasificados según su caso.

2.3 Verificación de la idoneidad de las variables

Los datos recopilados y clasificados son exportados a ficheros con formato ARFF (*Attribute Relationship File Format*) y CSV (*comma-separated values*), para ser interpretados por el paquete de software para aprendizaje automatizado "Weka", así como por el lenguaje y ambiente de programación para análisis estadístico "R", respectivamente. Tanto "R" como "Weka" permiten la conformación y visualización de gráficos 2D de matrices de dispersión para evaluar la idoneidad de las variables seleccionadas.

2.3.1 Gráficas 2D de las matrices de dispersión

En la figura 2.1 se puede identificar claramente la existencia de dos grupos; las estaciones de trabajo clasificadas como activas se identifican en color rojo y las estaciones de trabajo clasificadas como ociosas se identifican en color azul.

La figura 2.2 muestra la dispersión existente entre las estaciones de trabajo basándose en el uso promedio de memoria RAM **(ramprom)**. En la misma pueden ser identificados claramente ambos grupos, donde se evidencia el bajo porciento de uso de la memoria RAM presente en las estaciones de trabajo clasificadas como ociosas.

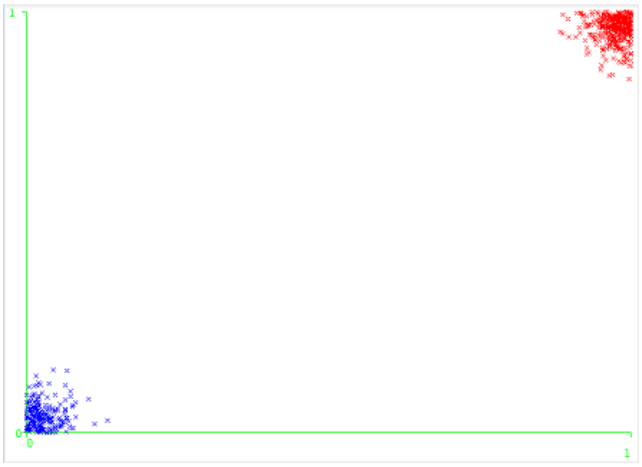


Figura 2.1: Matriz de dispersión (Estado)

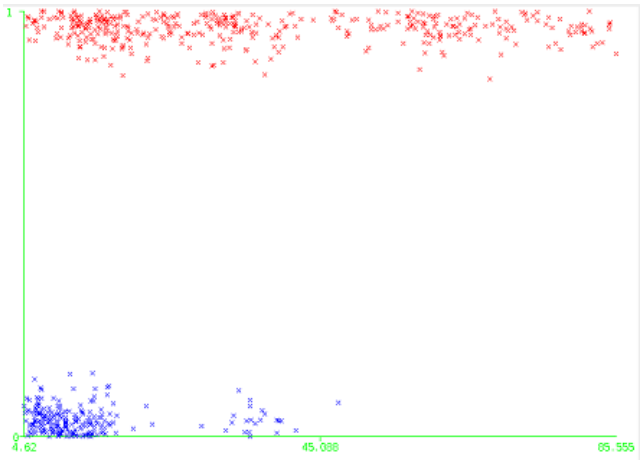


Figura 2.2: Matriz de dispersión (ramprom)

La figura 2.3 muestra la dispersión existente entre las estaciones de trabajo basándose en el porciento de inactividad del CPU **(cpuidle)**. En la misma pueden ser identificados claramente ambos grupos, donde se evidencia el alto porcentaje de inactividad del CPU presente en las estaciones de trabajo clasificadas como ociosas.

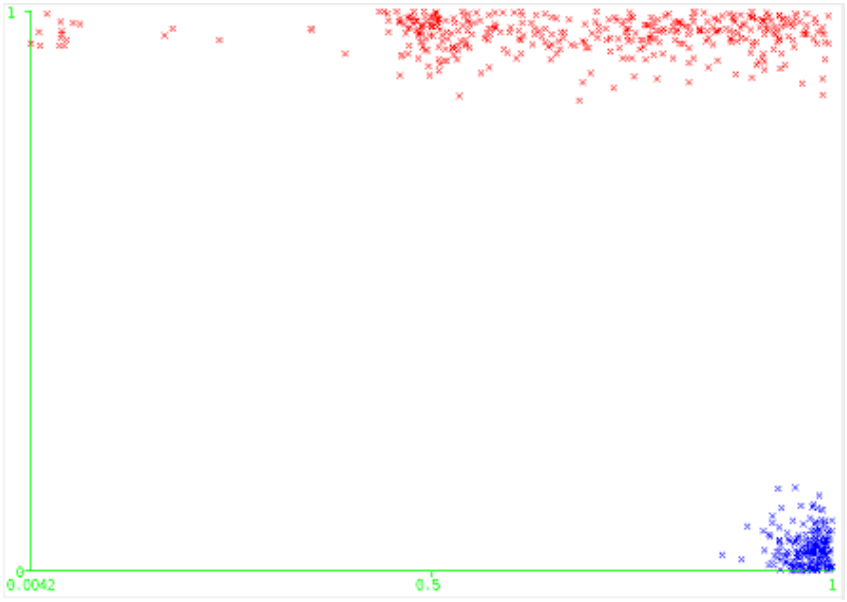


Figura 2.3: Matriz de dispersión (cpuidle)

La figura 2.4 muestra la dispersión existente entre las estaciones de trabajo basándose en la actividad de mouse y teclado **(infomousekeyboard)**. En la misma se evidencia el alto valor en milisegundos presente en las estaciones de trabajo clasificadas como ociosas.

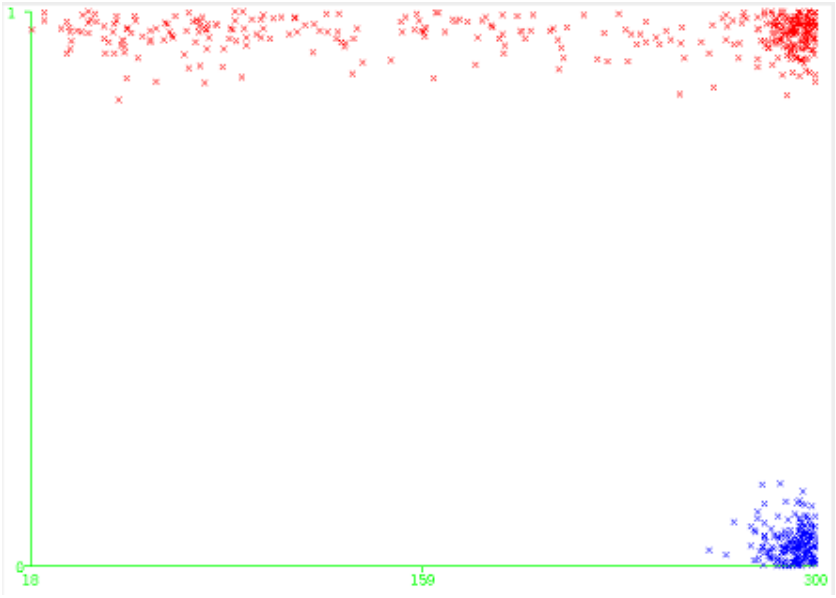


Figura 2.4: Matriz de dispersión (infomousekeyboard)

Para evaluar calidad de las variables independientes seleccionadas se hizo uso de la plataforma estadística "R". Haciendo uso de funciones propias de esta plataforma, se obtiene una gráfica 2D de dispersión, que permite evaluar la relación existente entre las variables independientes.

La figura 2.5 muestra el diagrama 2D de dispersión obtenido durante el desarrollo de la presente investigación. En el mismo se evidencia el alto número de instancias dispersas a ambos lados de la línea de mayor correlación. Este resultado sugiere una correlación muy baja o nula, interpretándose como inexistencia de relación entre las variables.

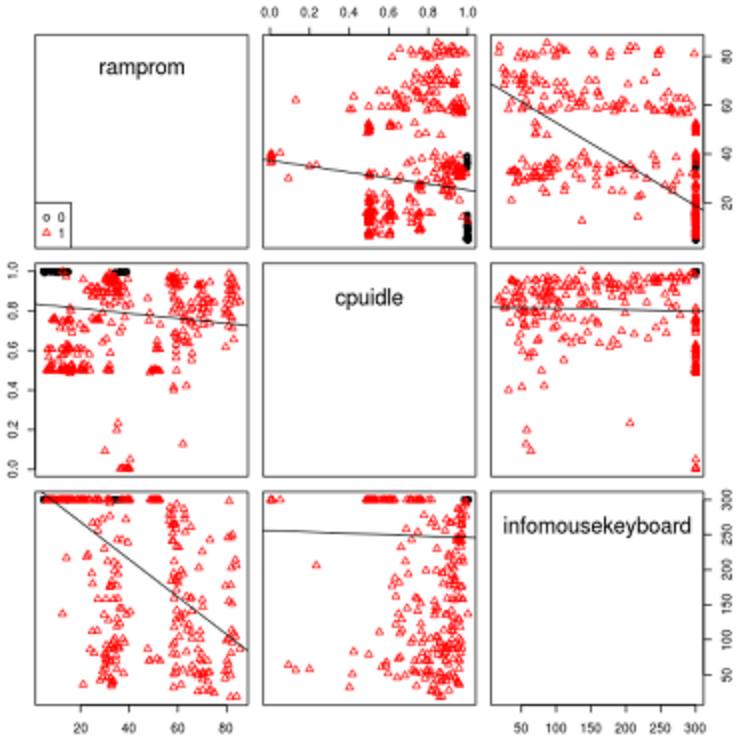


Figura 2.5: Diagrama 2D de dispersión

2.3.2 Prueba estadística “Lambda de Wilks”

La obtención y visualización de las gráficas 2D de las matrices de dispersión, tanto en el software para aprendizaje automatizado “Weka” como en la plataforma estadística “R”, sugiere la idoneidad de las variables seleccionadas. Para comprobar la presumible idoneidad de las mismas, se procede a la realización de una prueba que permita validar estadísticamente si existen diferencias entre los grupos de estaciones de trabajo clasificadas, basándose en las variables seleccionadas.

Las características o propiedades que se utilizan para distinguir entre grupos de objetos se denominan "variables discriminantes". La prueba estadística conocida como “Lambda de Wilks” es una medida de las diferencias entre grupos de objetos sobre variables discriminantes. Valores de lambda próximos a cero indican alta discriminación, por el contrario a medida que el valor de lambda crece progresivamente va indicando menor discriminación. Si el valor de lambda es igual a 1 significa que no existen diferencias

entre los grupos, siendo estos idénticos. Cuanto menor sea el valor del "Lambda de Wilks", más diferenciados estarán los grupos de objetos y será menos probable la hipótesis nula $\mathbf{H_0}$ o hipótesis de igualdad.

La figura 2.6 muestra el resultado de la ejecución de la prueba estadística "Lambda de Wilks" en la plataforma "R". En la misma se evidencia el bajo valor de lambda (0.3321) lo que significa discriminación entre los grupos de objetos y finalmente se rechaza la hipótesis nula o de igualdad $\mathbf{H_0}$, teniendo en cuenta que el valor p asociado al resultado observado es igual o menor que el nivel de significación establecido (0.05).

```
        One-way MANOVA (Bartlett Chi2)

data:  x
Wilks' Lambda = 0.3321, Chi2-Value = 693.875, DF = 3.000, p-value <
2.2e-16
sample estimates:
        V1        V2       V3
0 11.99239 0.9975654 299.9956
1 36.43928 0.6919744 218.2203
```

Figura 2.6: Prueba estadística "Lambda de Wilks"

Mediante el análisis de las gráficas 2D de las matrices de dispersión y los resultados de la prueba estadística "Lambda de Wilks", donde fue rechazada la hipótesis nula o de igualdad, se puede confirmar la idoneidad de las variables seleccionadas, teniendo en cuenta la discriminación existente entre los grupos de objetos.

2.4 Validación de los clasificadores seleccionados

Los clasificadores seleccionados fueron sometidos a un conjunto pruebas de validación cruzada variando los diferentes parámetros de configuración. Además se realizaron pruebas de validación con dos conjuntos de datos, un conjunto conformado por 1623 instancias y otro conformado por 2256 instancias. Ambos conjuntos fueron conformados con métricas previamente clasificadas que no formaban parte de la

base de conocimiento, con el objetivo de evaluar el desempeño de los clasificadores contra nuevas instancias no conocidas por ellos. En las pruebas realizadas con ambos conjuntos fueron variados los mismos parámetros de configuración para cada clasificador.

2.4.1 k-NN

La figura 2.7 muestra los resultados de las pruebas realizadas variando la configuración de los parámetros de k-NN contra el conjunto conformado por 1623 instancias. Los parámetros variados fueron la cantidad de vecinos k y el tamaño de la ventana. En la misma se evidencia que a medida que aumenta la cantidad de vecinos, el porciento de instancias clasificadas correctamente va disminuyendo. En cambio al variar el valor del tamaño de la ventana se evidencian mejores resultados en la clasificación, llegando a obtener el 100% de instancias correctamente clasificadas en todos los casos que el tamaño de la misma sea mayor que cero.

k	Ventana	Correcto	Incorrecto	Error absoluto medio	Raíz del error cuadrático medio	Error absoluto relativo	Raíz del error cuadrático relativo
1		95.9951 %	4.0049 %	0.0415	0.1998	11.457 %	55.166 %
10		92.3598 %	7.6402 %	0.0708	0.246	19.5489 %	67.9059 %
1	10	100 %	0 %	0.0833	0.0833	23.0072 %	23.0072 %
10	10	100 %	0 %	0.0098	0.0098	2.7067 %	2.7067 %
15	15	100 %	0 %	0.0044	0.0044	1.2162 %	1.2162 %
20	**20**	**100 %**	**0 %**	**0.0025**	**0.0025**	**0.6868 %**	**0.6868 %**

Figura 2.7: k-NN con 1623 instancias

La figura 2.8 muestra los resultados de las pruebas realizadas variando la configuración de los parámetros de k-NN contra el conjunto conformado por 2256 instancias. En la misma se evidencia que con mayor cantidad de instancias los resultados varían notablemente. La mejor clasificación obtenida es de 97.1188% de instancias correctamente clasificadas para $k = 1$, en cambio al aumentar el valor de k y el tamaño de la ventana los resultados van empeorando hasta obtener 10.1507% de instancias incorrectamente clasificadas.

k	Ventana	Correcto	Incorrecto	Error absoluto medio	Raíz del error cuadrático medio	Error absoluto relativo	Raíz del error cuadrático relativo
1	**-**	**97.1188 %**	**2.8812 %**	**0.0303**	**0.1695**	**7.7647 %**	**42.481 %**
10	-	94.4149 %	5.5851 %	0.052	0.2102	13.3167 %	52.6939 %
1	10	89.8493 %	10.1507 %	0.1679	0.3025	43.0373 %	75.8341 %
10	10	89.8493 %	10.1507 %	0.1093	0.3156	28.0181 %	79.1101 %
15	15	89.8493 %	10.1507 %	0.105	0.3172	26.9153 %	79.5138 %
20	20	89.8493 %	10.1507 %	0.1035	0.3178	26.5236 %	79.6622 %

Figura 2.8: k-NN con 2256 instancias

2.4.2 MLP

La figura 2.9 muestra los resultados de las pruebas realizadas variando la configuración de los parámetros de MLP contra el conjunto conformado por 1623 instancias. Los parámetros variados fueron el tiempo de entrenamiento, la cantidad de capas y la cantidad de neuronas presentes en la(s) capa(s) oculta(s). En la misma se evidencian resultados favorables al aumentar el tiempo de entrenamiento de la red neuronal. El valor mínimo de tiempo de entrenamiento de la red arroja un 98.0283% de instancias correctamente clasificadas, con una capa oculta conformada por dos neuronas.

Tiempo	Capas - Neuronas	Correcto	Incorrecto	Error absoluto medio	Raíz del error cuadrático medio	Error absoluto relativo	Raíz del error cuadrático relativo
500	1 - 2	98.0283 %	1.9717 %	0.0222	0.1263	6.1343 %	34.861 %
1000	1 - 2	98.2748 %	1.7252 %	0.0182	0.1139	5.0349 %	31.46 %
2500	1 - 2	98.8293 %	1.1707 %	0.0117	0.091	3.2393 %	25.1195 %
3000	1 - 2	98.8909 %	1.1091 %	0.0105	0.0861	2.9111 %	23.7821 %
4000	1 - 2	99.4455 %	0.5545 %	0.0062	0.0632	1.6988 %	17.4563 %
5000	1 - 2	99.5071 %	0.4929 %	0.0051	0.0573	1.4024 %	15.8144 %
5000	**3 – [3,3,3]**	**99.6303 %**	**0.3697 %**	**0.0039**	**0.051**	**1.0869 %**	**14.082 %**

Figura 2.9: MLP con 1623 instancias

El porciento de instancias correctamente clasificadas va aumentando a medida que aumenta el tiempo de entrenamiento de la red, llegando a obtenerse 99.5071% para el valor máximo de tiempo de

entrenamiento definido (5000). Pruebas realizadas para estos mismos valores de tiempo, variando la cantidad de capas y neuronas mostraron idénticos resultados. Las mejoras en el resultado obtenido se evidencian en 99.6303% de instancias correctamente clasificadas. Este resultado se obtuvo con el valor máximo de tiempo de entrenamiento y un total de 3 capas ocultas con 3 neuronas cada capa.

La figura 2.10 muestra los resultados de las pruebas realizadas variando la configuración de los parámetros de MLP contra el conjunto conformado por 2256 instancias. En la misma se evidencian resultados favorables al aumentar el tiempo de entrenamiento de la red neuronal. El valor mínimo de tiempo de entrenamiento de la red proporciona un 98.5372% de instancias correctamente clasificadas, con una capa oculta conformada por dos neuronas. El porciento de instancias correctamente clasificadas va aumentando a medida que aumenta el tiempo de entrenamiento de la red, llegando a obtenerse un 99.6454% para el valor de tiempo de entrenamiento igual a 4500 ms. Las mejoras en el resultado obtenido se evidencian en 99.734% de instancias correctamente clasificadas. Este resultado se obtuvo con valor máximo tiempo de entrenamiento (5000) y un total de 3 capas ocultas con 3 neuronas cada capa.

Tiempo	Capas - Neuronas	Correcto	Incorrecto	Error absoluto medio	Raíz del error cuadrático medio	Error absoluto relativo	Raíz del error cuadrático relativo
500	1 - 2	98.5372 %	1.4628 %	0.0175	0.109	4.4782 %	27.3187 %
1000	1 - 2	98.7145 %	1.2855 %	0.0144	0.0986	3.6927 %	24.7093 %
2000	1 - 2	99.0691 %	0.9309 %	0.0107	0.084	2.7526 %	21.0669 %
3000	1 - 2	99.1578 %	0.8422 %	0.0086	0.0746	2.2004 %	18.7104 %
4000	1 - 2	99.6011	0.3989 %	0.0054	0.0541	1.3783 %	13.5579 %
4500	1 - 2	99.6454 %	0.3546 %	0.0046	0.0495	1.1763 %	12.4053 %
5000	**3 – [3,3,3]**	**99.734 %**	**0.266 %**	**0.003**	**0.0433**	**0.7773 %**	**10.8458 %**

Figura 1.10: MLP con 2256 instancias

2.4.3 SVM

La figura 2.11 muestra los resultados de las pruebas realizadas variando la configuración de los parámetros de SVM contra el conjunto conformado por 1623 instancias. Los parámetros variados fueron el tipo de SVM y el kernel a utilizar por la misma. Durante el desarrollo de las pruebas fueron utilizados dos

tipos de SVM (C-SVC y nu-SVC), teniendo en cuenta que la codificación binaria utilizada no es admitida por los otros tipos de SVM. En general, se evidencian buenos resultados para los diferentes tipos de SVM y kernel seleccionados. El peor caso se evidencia para la SVM de tipo 0 (C-SVC) y utilizando kernel de tipo 1 ($polynomial: (\ gamma * u' * v\ +\ coef0\ )\hat{}\ degree$), dando como resultado un 99.5687% de instancias correctamente clasificadas.

Tipo	Kernel	Correcto	Incorrecto	Error absoluto medio	Raíz del error cuadrático medio	Error absoluto relativo	Raíz del error cuadrático relativo
0	**0**	**100 %**	**0 %**	**0**	**0**	**0 %**	**0%**
1	**0**	**100 %**	**0 %**	**0**	**0**	**0 %**	**0%**
0	1	99.5687 %	0.4313 %	0.0043	0.0657	1.1908 %	18.1316 %
1	**1**	**100 %**	**0 %**	**0**	**0**	**0 %**	**0%**
0	**2**	**100 %**	**0 %**	**0**	**0**	**0 %**	**0%**
1	**2**	**100 %**	**0 %**	**0**	**0**	**0 %**	**0%**
0	**3**	**100 %**	**0 %**	**0**	**0**	**0 %**	**0%**
1	3	0 %	100 %	1	1	276.087 %	276.087 %

Tipo SVM
0 = C-SVC
1 = nu-SVC
Kernel
0 = linear: u'*v
1 = polynomial: (gamma*u'*v + coef0)^degree
2 = radial basis function: exp(-gamma*|u-v|^2)
3 = sigmoid: tanh(gamma*u'*v + coef0)

Figura 2.11: SVM con 1623 instancias

La figura 2.12 muestra los resultados de las pruebas realizadas variando la configuración de los parámetros de SVM contra el conjunto conformado por 2256 instancias. En este caso se evidencia que independientemente de los buenos resultados obtenidos, para un conjunto de datos con mayor cantidad de instancias, el comportamiento de SVM en el presente problema no resulta tan bueno como para conjuntos de datos con menor cantidad de instancias. El peor de los casos se evidencia para la SVM de tipo 1 (nu-SVC) y utilizando kernel de tipo 3 ($sigmoid: tanh(\ gamma * u' * v\ +\ coef0\ )$), dando como

resultado un 89.8493% de instancias incorrectamente clasificadas. El mejor caso se evidencia para la SVM de tipo 0 (C-SVC) y utilizando kernel de tipo 0 $(linear: u' * v)$, brindando como resultado un 100% de instancias correctamente clasificadas y valores igual a cero en los diferentes tipos de errores.

El alto porcentaje de instancias correctamente clasificadas con diferentes configuraciones obteniéndose en más de un caso 100% de clasificación correcta, permite identificar a SVM como un fuerte candidato para la obtención del modelo computacional.

Tipo	Kernel	Correcto	Incorrecto	Error absoluto medio	Raíz del error cuadrático medio	Error absoluto relativo	Raíz del error cuadrático relativo
0	**0**	**100 %**	**0 %**	**0**	**0**	**0 %**	**0 %**
1	0	98.227 %	1.773 %	0.0177	0.1332	4.5442 %	33.376 %
0	1	99.6897 %	0.3103 %	0.0031	0.0557	0.7952 %	13.9622 %
1	1	96.8528 %	3.1472 %	0.0315	0.1774	8.0659 %	44.4666 %
0	2	99.4238 %	0.5762 %	0.0058	0.0759	1.4769 %	19.0273 %
1	2	97.0301 %	2.9699 %	0.0297	0.1723	7.6115 %	43.1958 %
0	3	89.8493 %	10.1507 %	0.1015	0.3186	26.0155 %	79.8587 %
1	3	10.1507 %	89.8493 %	0.8985	0.9479	230.277 %	237.5917 %

Tipo SVM
0 – C SVC
1 = nu-SVC
Kernel
0 = linear: u'*v
1 = polynomial: (gamma*u'*v + coef0)^degree
2 – radial basis function: exp(gamma*|u v|^2)
3 = sigmoid: tanh(gamma*u'*v + coef0)

Figura 2.12: SVM con 2256 instancias

2.4.4 LADTree

La figura 2.13 muestra los resultados de las pruebas realizadas variando la configuración de los parámetros de LADTree contra el conjunto conformado por 1623 instancias. El único parámetro variado en la configuración es la cantidad de iteraciones. En la misma se evidencian los buenos resultados obtenidos en las pruebas. Para un número mínimo de iteraciones igual a 10, se obtuvo un 100% de instancias

correctamente clasificadas, un error absoluto medio igual a 0.0073 y pequeños valores para los diferentes tipos de errores. Intentando minimizar el valor de los diferentes tipos de errores se aumenta el valor de la cantidad de iteraciones. Con un número de iteraciones igual a 20, el valor del error absoluto medio es igual a 0.0001 y la raíz del error cuadrático medio es igual a 0.0002. Para un número de iteraciones igual a 25, el valor del error absoluto medio y la raíz del error cuadrático medio es igual a 0, el error absoluto relativo es igual a 0.01% y la raíz del error cuadrático relativo es igual a 0.0102%. Finalmente para un número de iteraciones igual a 40, el valor de los diferentes tipos de errores es igual a cero.

Iteraciones	Correcto	Incorrecto	Error absoluto medio	Raíz del error cuadrático medio	Error absoluto relativo	Raíz del error cuadrático relativo
10	100 %	0 %	0.0073	0.0286	2.0271 %	7.8841 %
15	100 %	0 %	0.0009	0.0024	0.2439 %	0.6725 %
20	100 %	0 %	0.0001	0.0002	0.0293 %	0.0431 %
25	100 %	0 %	0	0	0.01 %	0.0102 %
30	100 %	0 %	0	0	0.0013 %	0.0014 %
40	**100 %**	**0 %**	**0**	**0**	**0 %**	**0 %**

Figura 2.13: LADTree con 1623 instancias

La figura 2.14 muestra los resultados de las pruebas realizadas variando la configuración de los parámetros de LADTree contra el conjunto conformado por 2256 instancias. Para un número mínimo de iteraciones igual a 10 se obtuvo un 100% de instancias correctamente clasificadas, un error absoluto medio igual a 0.0063 y pequeños valores para los diferentes tipos de errores. Con un número de iteraciones igual a 20, el valor del error absoluto medio es igual a 0.0001 y la raíz del error cuadrático medio es igual a 0.0001. Para un número de iteraciones igual a 25, el valor del error absoluto medio y la raíz del error cuadrático medio es igual a 0, el error absoluto relativo es igual a 0.0084% y la raíz del error cuadrático relativo es igual a 0.0088%. Finalmente, con un máximo de iteraciones igual a 40, el valor de los diferentes tipos de errores es igual a cero.

Iteraciones	Correcto	Incorrecto	Error absoluto medio	Raíz del error cuadrático medio	Error absoluto relativo	Raíz del error cuadrático relativo
10	100 %	0 %	0.0063	0.0243	1.6123 %	6.1005 %
20	100 %	0 %	0.0001	0.0001	0.0242 %	0.0349 %
25	100 %	0 %	0	0	0.0084 %	0.0088 %
30	100 %	0 %	0	0	0.0011 %	0.0012 %
35	100 %	0 %	0	0	0.0002 %	0.0002 %
40	**100 %**	**0 %**	**0**	**0**	**0 %**	**0 %**

Figura 2.14: LADTree con 2256 instancias

2.4.5 Comparación de los mejores resultados

La figura 2.15 muestra los mejores resultados obtenidos de las pruebas realizadas a los clasificadores seleccionados, contra el conjunto conformado por 1623 instancias. Independientemente de los buenos resultados de esta prueba, se evidencia que los clasificadores SVM y LADTree presentan un mejor comportamiento para la resolución del presente problema, resultando ambos en 100% de instancias correctamente clasificadas y con valor igual a cero para los diferentes tipos de errores. El peor comportamiento lo presenta MLP con un 99.633% de instancias correctamente clasificadas, error absoluto medio igual a 0.0039, raíz del error cuadrático medio igual a 0.0051, error absoluto relativo igual a 1.0869% y la raíz del error cuadrático relativo igual a 14.082%.

Clasificador	Clasificación Correcta	Clasificación Incorrecta	Error absoluto medio	Raíz del error cuadrático medio	Error absoluto relativo	Raíz del error cuadrático relativo
k-NN	**100 %**	**0 %**	**0.0025**	**0.0025**	**0.6868 %**	**0.6868 %**
MLP	**99.6303 %**	**0.3697 %**	**0.0039**	**0.051**	**1.0869 %**	**14.082 %**
SVM	**100 %**	**0 %**	**0**	**0**	**0 %**	**0 %**
LADTree	**100 %**	**0 %**	**0**	**0**	**0 %**	**0 %**

Figura 2.15: Mejores resultados con 1623 instancias

La figura 2.16 muestra los mejores resultados obtenidos de las pruebas realizadas a los clasificadores seleccionados, contra el conjunto conformado por 2256 instancias. En la misma se evidencia que al igual que en las pruebas realizadas con el conjunto formado por 1623 instancias, los clasificadores SVM y LADTree presentan un mejor comportamiento para la resolución del presente problema, resultando ambos en un 100% de instancias correctamente clasificadas y con valor igual a cero para los diferentes tipos de errores. El peor comportamiento lo presenta k-NN con un 97.1188% de instancias correctamente clasificadas, error absoluto medio igual a 0.0303, raíz del error cuadrático medio igual a 0.1695, error absoluto relativo igual a 7.7647% y la raíz del error cuadrático relativo igual a 42.481%.

Clasificador	Clasificación Correcta	Clasificación Incorrecta	Error absoluto medio	Raíz del error cuadrático medio	Error absoluto relativo	Raíz del error cuadrático relativo
k-NN	97.1188 %	2.8812 %	0.0303	0.1695	7.7647 %	42.481 %
MLP	99.734 %	0.266 %	0.003	0.0433	0.7773 %	10.8458 %
SVM	100 %	0 %	0	0	0 %	0 %
LADTree	100 %	0 %	0	0	0 %	0 %

Figura 2.16: Mejores resultados con 2256 instancias

2.5 Selección del modelo computacional

En las pruebas de validación realizadas a los clasificadores seleccionados se evidencian buenos resultados, destacándose notablemente los resultados de los clasificadores SVM y LADTree. Teniendo en cuenta los resultados de las pruebas de ambos clasificadores, donde ambos muestran 100% de instancias correctamente clasificadas y valor igual a cero para los diferentes tipos de errores, surge la necesidad de decidir qué clasificador es más conveniente para la resolución del problema planteado en la presente investigación.

Tomando en consideración el teorema "*No free lunch*" (27), el cual básicamente plantea que no se puede encontrar ningún algoritmo metaheurístico que sea el mejor en comportamiento para cualquier problema, no se puede afirmar que un clasificador es mejor que otro. Debido a este planteamiento, se decide que

para la selección del clasificador, no solo deben tenerse en cuenta los resultados de las pruebas de validación, también deben tenerse en consideración la simplicidad y conveniencia de uso.

Atendiendo a lo planteado en la sección 1.1.1 sobre los elementos claves que se deben tener en cuenta para realizar la comparación entre clasificadores, y las consideraciones del Principio de Parsimonia o Navaja de Ockham[12] (28), que aplicado a la informática establece que ante múltiples soluciones que ofrezcan el mismo resultado, la decisión correcta es seleccionar la de mayor simplicidad; se decide seleccionar LADTree con número de iteraciones igual a 40, como clasificador más conveniente para la resolución del problema planteado en la presente investigación.

2.6 Implementación del modelo obtenido

Siendo seleccionado LADTree como clasificador más conveniente para la resolución del problema en la presente investigación, se procede a la implementación del modelo obtenido. Para su implementación se utiliza el API del software de aprendizaje automatizado "Weka" en su versión 3.6.8 y se diseñó e implementó una clase en el lenguaje de programación Java que permite obtener la clasificación de cada estación de trabajo.

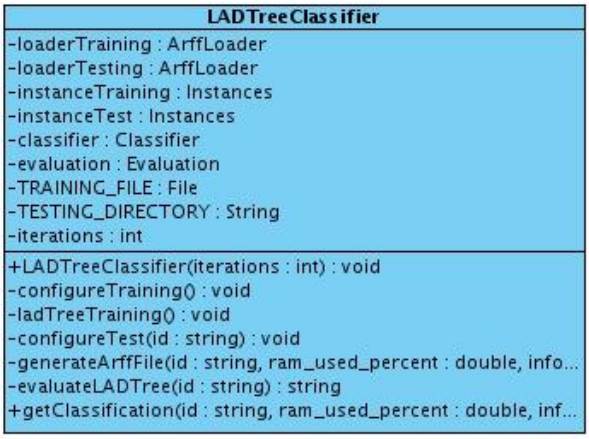


Figura 2.17: Diagrama UML de la clase LADTreeClassifier

[12] Principio metodológico y filosófico atribuido a Guillermo de Ockham (1280-1349), según el cual, "en igualdad de condiciones, la explicación más sencilla suele ser la correcta".

La figura 2.17 muestra el diagrama UML que representa a la clase **LADTreeClassifier** mencionada anteriormente. La misma cuenta con los atributos y métodos necesarios para la configuración y uso del clasificador LADTree implementado en el API de "Weka".

A continuación se explican los principales atributos y métodos de la clase:

Atributos

- **loaderTraining:** responsable de leer la información de la base de entrenamiento.
- **loaderTesting:** responsable de leer un fichero de tipo Arff generado dinámicamente con las métricas de la estación de trabajo a clasificar.
- **intanceTraining:** responsable de instanciar los datos almacenados en el atributo loaderTraining.
- **instanceTest:** responsable de instanciar los datos almacenados en el atributo loaderTesting.
- **classifier**: responsable de instanciar el clasificador LADTree implementado en el API de "Weka".
- **evaluation:** responsable de almacenar el resultado de la evaluación del modelo computacional obtenido y la clasificación de la estación de trabajo.
- **TRAINING_FILE**: corresponde a la dirección del fichero de tipo Arff utilizado como base de conocimiento.
- **TESTING_DIRECTORY**: corresponde a un directorio de intercambio que se crea para almacenar un fichero de tipo Arff con las métricas de la estación de trabajo a clasificar, el cual es generado y eliminado dinámicamente.

Métodos

- **configureTraining:** es responsable de leer los datos y cargar la información de la base de conocimiento para realizar el entrenamiento del clasificador.
- **ladTreeTraining:** es responsable de establecer los parámetros de configuración y construir el clasificador LADTree, a partir de la información almacenada en la base de conocimiento.
- **generateArffFile:** es responsable de generar dinámicamente un fichero de tipo Arff con las métricas de la estación de trabajo a clasificar, recibiendo como parámetros el valor de las mismas y un id que se corresponde con la dirección MAC de la estación de trabajo a clasificar.

- **configureTest:** es responsable de leer los datos y cargar la información del fichero de tipo Arff generado dinámicamente con las métricas de la estación de trabajo a clasificar, utilizando para ello el id que recibe como parámetro y que se corresponde con la dirección MAC de la estación de trabajo.
- **evaluateLADTree:** es responsable de evaluar el modelo computacional obtenido y retornar la clasificación de la estación de trabajo, el mismo recibe como parámetro un id que se corresponde con la dirección MAC de la estación de trabajo a clasificar, para una vez efectuada la clasificación, eliminar el fichero de tipo Arff generado dinámicamente con las métricas de la misma y así evitar ocupar espacio innecesariamente.
- **getClassification:** es el responsable de retornar la clasificación de la estación de trabajo a partir de las métricas recibidas por parámetro, invocando para ello los demás métodos definidos e implementados en la clase en cuestión.

Conclusiones del Capítulo

El análisis de las gráficas 2D de las matrices de dispersión y los resultados de la prueba estadística “Lambda de Wilks” permitieron validar la idoneidad de las variables seleccionadas. Las pruebas de validación realizadas a los clasificadores seleccionados, variando sus parámetros de configuración y los conjuntos de datos, permitieron la selección de LADTree con número de iteraciones igual a 40, como clasificador más conveniente para la resolución del problema planteado en la presente investigación. El diseño e implementación de la clase LADTreeClassifier facilita la configuración y uso del clasificador LADTree implementado en el API del software de aprendizaje automatizado “Weka”, permitiendo obtener la clasificación de las estaciones de trabajo en estado activo u ocioso a partir de las métricas recopiladas.

Capítulo 3: Integración del modelo a SIMON

En este capítulo se procede a la integración del modelo obtenido al Sistema de Monitoreo del uso y explotación de las estaciones de trabajo. Para ello se realizan los cambios necesarios en los subsistemas de recolección, almacenamiento, integración y visualización. Además se explica la necesidad de implementar un nuevo proceso ETL y se lleva a cabo el análisis, diseño e implementación de los nuevos reportes del estado de las estaciones de trabajo.

3.1 Integración al sistema T-arenal

Como fue mencionado con anterioridad, el cliente de T-arenal instalado en cada estación de trabajo es el encargado de recolectar, cada cierto intervalo de tiempo, una muestra del estado de los recursos de hardware y software de la PC, la cual es luego enviada al servidor. Esta información es almacenada por el servidor en la base de datos de recolección, para luego ser cargada al ODS por el proceso ETL. Teniendo en cuenta que el estado de las estaciones de trabajo es una métrica adicional que no era anteriormente almacenada, se hace necesario realizar modificaciones en el subsistema de recolección.

3.1.1 Modificaciones a la base de datos de recolección

Atendiendo a los problemas planteados con anterioridad sobre el diseño de la base de datos de recolección, se decide realizar un nuevo diseño e implementación de la misma, que dé solución a dichos problemas. Para ello se diseña una nueva base de datos de recolección la cual fue implementada en el gestor de base de datos PostgreSQL. La misma cuenta con un menor número de tablas y relaciones, permitiendo además el almacenamiento de un mayor número de métricas. Como se observa en el modelo físico de la base de datos, ver figura 3.1, la misma cuenta con un total de cinco tablas que almacenan la información de las métricas estáticas y dinámicas de las estaciones de trabajo, así como la información de los procesos que se encuentran en ejecución.

La información dinámica es aquella que puede ser diferente en cualquier instante de tiempo, ejemplo de esto son: RAM en uso, % de uso del CPU, estado de la estación de trabajo, tiempo de inactividad del usuario, etc. La información estática es aquella que no varía con frecuencia, representando generalmente

las características de hardware y software de las estaciones de trabajo. La tabla **tb_trace_static_metrics** recolecta la información de las métricas estáticas que por alguna razón cambiaron sus valores.

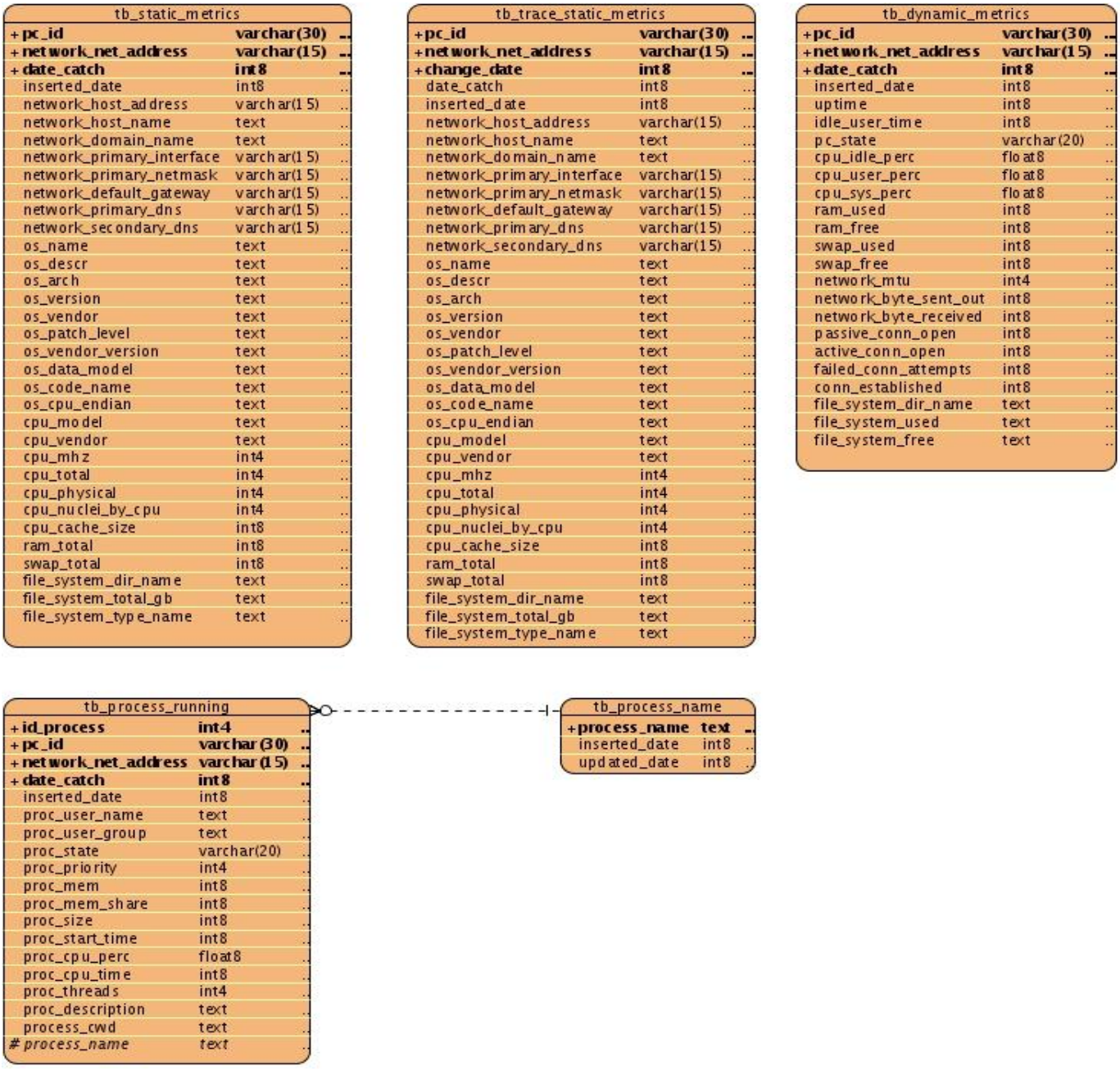


Figura 3.1: Modelo físico de la base de datos de recolección

3.1.2 Modificaciones al sistema T-arenal

Una vez realizado el diseño de la nueva base de datos para la recolección de las métricas, se hace necesario realizar un nuevo diseño de clases para el módulo de tecnología tanto para el cliente como el servidor, que permita la recogida de las métricas en el cliente y la inserción de las mismas en la base de datos por parte del servidor. La figura 3.2 muestra el diagrama de clases del diseño del módulo de tecnología en el cliente.

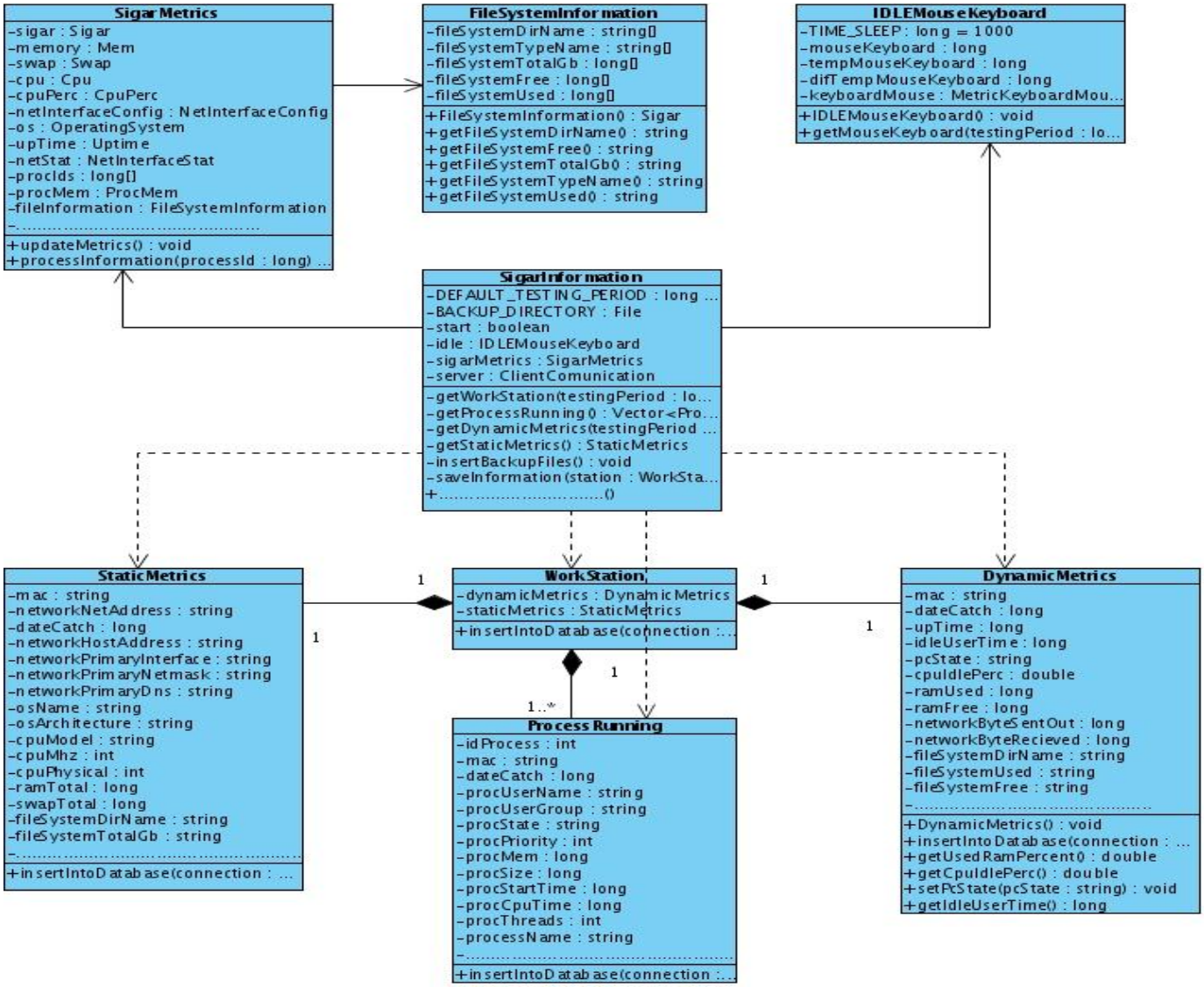


Figura 3.2: Diagrama de clases del diseño del módulo de tecnología en el cliente

A continuación se ofrece una explicación de las responsabilidades de cada clase.

- **StaticMetrics:** almacena la información de las métricas estáticas de la estación de trabajo que posteriormente serán almacenadas en la base de datos de recolección.
- **DynamicMetrics**: almacena la información de las métricas dinámicas de la estación de trabajo que posteriormente serán almacenadas en la base de datos de recolección.
- **ProcessRunning**: almacena la información de uno de los procesos que se encuentra en ejecución en la estación de trabajo.
- **WorkStation**: representa la estación de trabajo como un objeto. Está formada por las métricas estáticas, dinámicas y la información de todos los procesos que están en ejecución en la estación de trabajo.
- **FileSystemInformation**: contiene la información de los sistemas de archivos existentes en la estación de trabajo.
- **IDLEMouseKeyboard**: permite determinar el tiempo que el usuario ha pasado sin movimiento de mouse y teclado desde la última vez en que fueron recogidas las métricas.
- **SigarMetrics**: obtiene la información de todas las métricas de la estación de trabajo.
- **SigarInformation**: es responsable por el proceso de recogida y envió de la información de las estaciones de trabajo hacia el servidor.

Patrones de Diseño

Los patrones de diseño constituyen una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular. Estos identifican clases, instancias, roles, y asignación de responsabilidades. La realización de los diagramas de clases del diseño de la presente investigación, se hizo teniendo en cuenta los patrones (GRASP) de asignación de responsabilidades. A continuación se describen los patrones utilizados.

- **Experto:** es el encargado de asignar responsabilidades a una clase que dispone de la información necesaria para llevar a cabo una tarea. Si esta asignación de responsabilidades se hace

adecuadamente, los sistemas tienden a ser más fáciles de entender, mantener y ampliar. Este patrón está presente en las clases SigarInformation, IDLEMouseKeyboard y WorkStation.

- **Creador:** guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Este patrón está presente en la clase SigarInformation, la cual al ser experta en información, es la encargada de la creación de la mayoría de las instancias. El patrón se evidencia además en la clase SigarMetric.
- **Bajo Acoplamiento:** el acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases. Con la aplicación de este patrón se busca asignar las responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible. Una clase con bajo (o débil) acoplamiento no depende de muchas otras. Este patrón se evidencia en las clases StaticMetric, DynamicMetric, ProcessRunning y FileSystemInformation.
- **Alta Cohesión:** en la perspectiva del diseño orientado a objetos, la cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una clase con alta cohesión tiene responsabilidades moderadas en un área fundamental y colabora con las otras para llevar a cabo las tareas. Este patrón está presente en las clases StaticMetric, DynamicMetric, ProcessRunning y FileSystemInformation, las cuales solamente asumen las responsabilidades necesarias, teniendo además un bajo número de relaciones lo que garantiza tener un bajo acoplamiento y por lo tanto una alta cohesión.

3.1.3 Incorporación del modelo al sistema

Una vez realizados los cambios en el subsistema de recolección, se procese a integrar el modelo para la clasificación de los estados de las estaciones de trabajo. El mismo es añadido al servidor, el cual una vez recibidas las métricas enviadas por los clientes, pasará al modelo los datos necesarios para la clasificación de la estación de trabajo. Posteriormente las métricas serán almacenadas en la base de datos, unido a la clasificación obtenida.

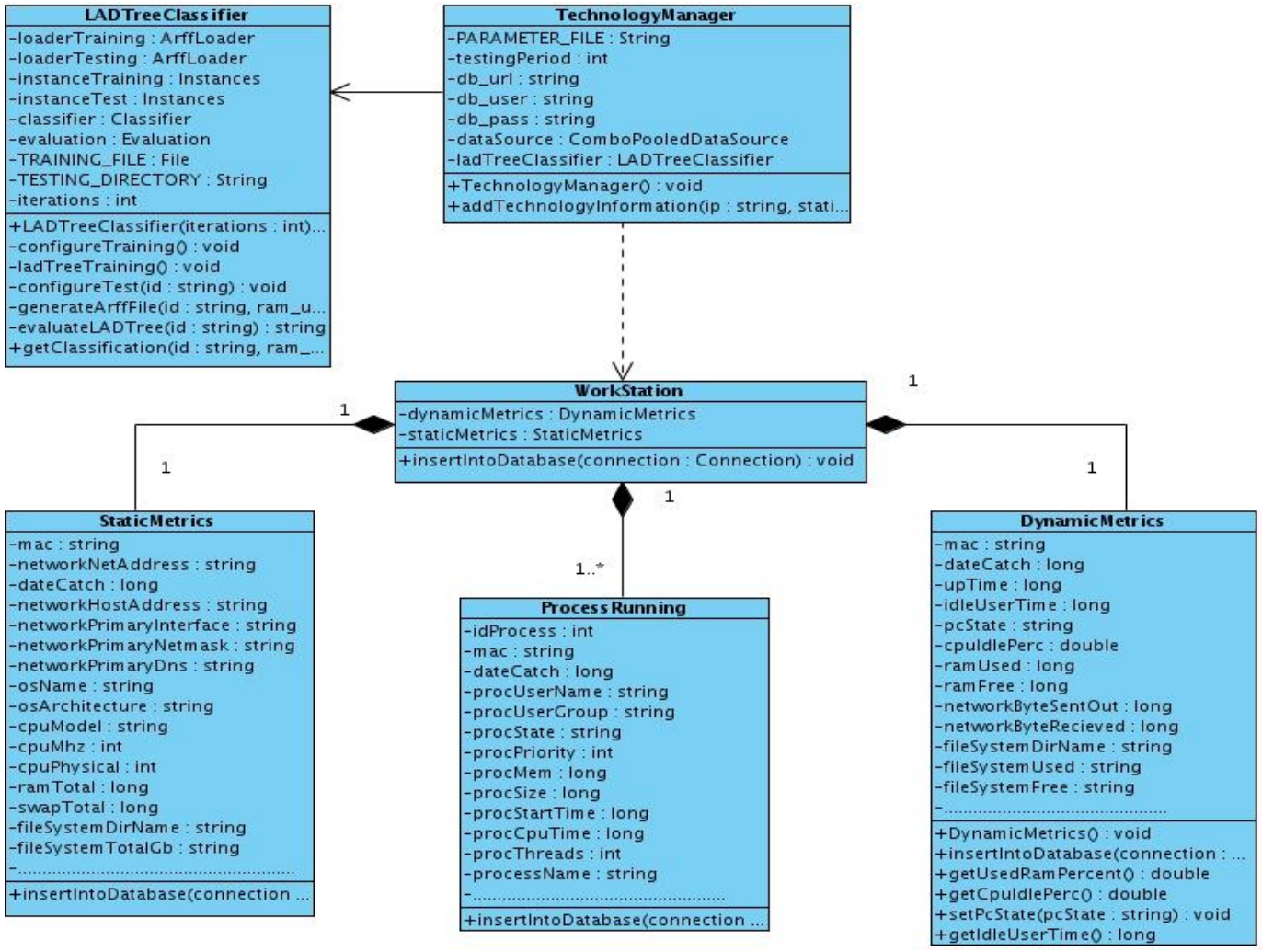


Figura 3.3: Diagrama de clases del diseño del módulo de tecnología en el servidor

Como se observa en la figura 3.3, correspondiente al diagrama de clases del módulo de tecnología en el servidor, se decide añadir a la clase **TecnologyManager** un nuevo atributo de tipo **LADTreeClassifier¸** a través del cual se obtendrá la clasificación de las estaciones de trabajo. A continuación se hace una breve descripción de las clases presentes en el diagrama.

- **LADTreeClassifier:** Descrita en la sección 2.6, es la clase que representa el modelo de clasificación. Su función principal es clasificar el estado de las estaciones de trabajo una vez proporcionados los parámetros necesarios.

- **TecnologyManager:** Es la clase encargada del manejo de la información de las estaciones de trabajo. Su principal tarea es asegurar la clasificación de las estaciones de trabajo a partir de la información enviada por los clientes. Una vez obtenida dicha clasificación, se encarga del almacenamiento de las métricas en la base de datos de recolección.

Las clases StaticMetrics, DynamicMetrics y ProcessRunning presentan las mismas responsabilidades descritas en la sección anterior.

3.2 Integración al mercado de datos de SIMON

El sistema SIMON cuenta con un mercado de datos que se encarga de la carga, almacenamiento y visualización de la información. Para lograr que la información proporcionada por el modelo llegue a los usuarios finales es necesario realizar modificaciones en los subsistemas de almacenamiento, integración y visualización de dicho mercado. Esta información es mostrada a través de tablas y gráficas que forman parte de un conjunto de reportes brindados por el subsistema de visualización. Para determinar los reportes a implementar se hizo necesario conocer las necesidades de información de los usuarios, las cuales son identificadas en el proceso de especificación de requisitos.

3.2.1 Especificación de Requisitos

Uno de los pasos más importantes en el proceso de desarrollo de software es la especificación de los requisitos, teniendo en cuenta que es necesario conocer qué es lo que los usuarios finales necesitan. La implicación de los mismos durante el ciclo de vida del producto es de gran importancia, ya que de ello depende que los resultados sean satisfactorios o no. La especificación de requisitos del sistema SIMON v1.0 se encuentra en el documento “Especificación de Requisitos de Software”, a continuación se hará referencia a los requisitos identificados referentes a los nuevos reportes del estado de las estaciones de trabajo.

Requisitos de Información

Los requisitos de información representan la información que debe estar disponible para los usuarios finales. Los mismos constituyen una entrada principal para el proceso de inteligencia de negocios y para

futuros reportes. En la presente investigación se identificaron un total de 6 requisitos de información, los cuales se listan a continuación.

RI1- Obtener la cantidad de estados activos y ociosos, a nivel de universidad, edificios docentes, tipo de laboratorio (docente, producción, etc.) y laboratorio.

RI2- Obtener la cantidad de estados activos y ociosos por días, a nivel de universidad, edificios docentes, tipo de laboratorio (docente, producción, etc.) y laboratorio.

RI3- Obtener la cantidad de estados activos y ociosos por momento del día (madrugada, mañana, tarde, noche), a nivel de universidad, edificios docentes, tipo de laboratorio (docente, producción, etc.) y laboratorio.

RI4- Obtener el % estados activos y ociosos por cada hora del día.

RI5- Obtener el % de estados activos y ociosos por cada momento del día.

RI6- Obtener el % de estados activos y ociosos de las últimas dos semanas.

3.2.2 Modelo de Casos de Uso del Sistema

Los casos de uso del sistema describen la interacción del usuario con el sistema para obtener un objetivo específico. Estos se utilizan para representar los requisitos de información y funcionales a desarrollar, así como su relación con los actores del sistema. La descripción de los casos de uso del sistema SIMON se encuentra en el documento "Especificación de casos de uso". A continuación se muestran los casos de uso identificados referentes a los nuevos reportes del estado de las estaciones de trabajo.

Actores del Sistema

Los actores del sistema son personas o programas que interactúan con el sistema, es decir, no son parte de este, sino que pueden intercambiar información con él.

Actores	Descripción
Especialista	Es el responsable de realizar el análisis de los diferentes reportes del estado de las estaciones de trabajo.

Tabla 3.1: Actores del Sistema

Diagrama de Casos de Uso

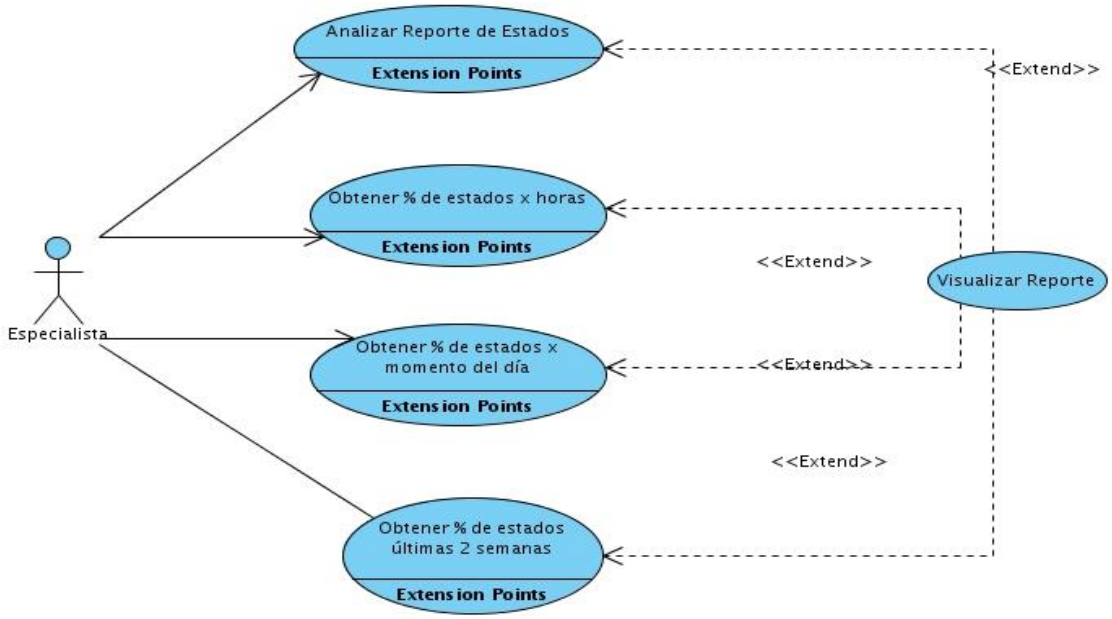
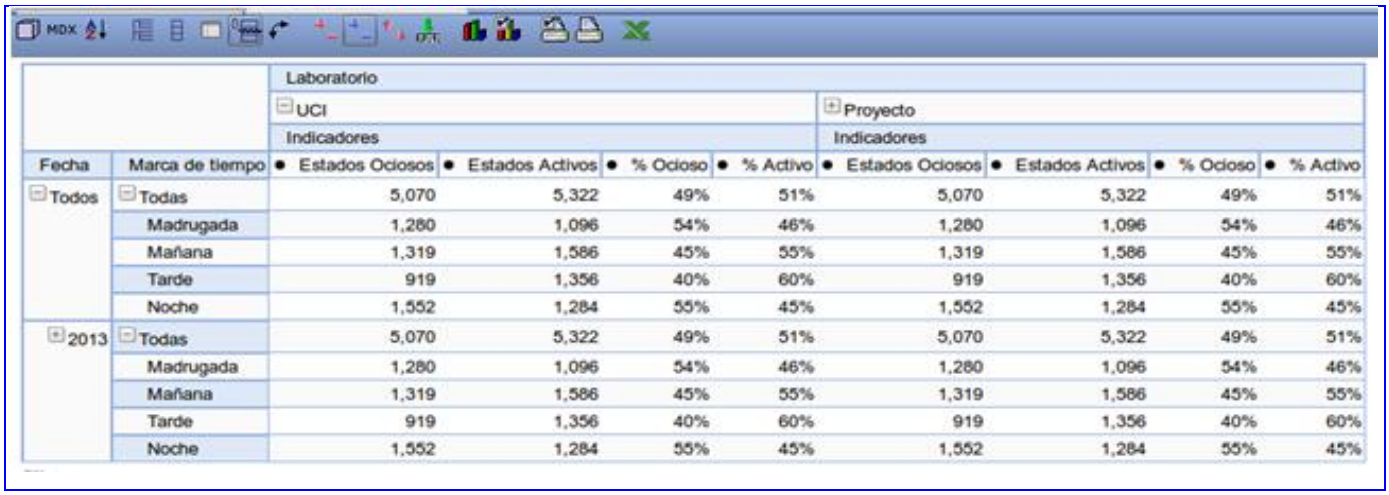


Figura 3.4: Diagrama de Casos de Uso del Sistema

Descripción del Caso de Uso "Analizar Reporte de Estados"

Caso de Uso:	**Analizar Reporte de Estados**
Tipo:	Información
Actores:	Especialista
Resumen:	El CU comienza cuando el especialista se autentica en el sistema y selecciona la opción "Reporte de Estados" en el área de análisis (A.A Reportes de Estados). El CU termina cuando se muestra el reporte de los estados activos y ociosos de las estaciones de trabajo.
Precondiciones:	El usuario debe de estar autenticado. Completitud del almacén. Los datos deben estar cargados.
Referencias	RI1, RI2, RI3

<table>
<tr><td>Prioridad</td><td colspan="2">Crítico.</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td>Acción del Actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td>1. El especialista entra al sistema.</td><td colspan="2">2. El sistema muestra el área de análisis general A.A.G Administración de Tecnología.</td></tr>
<tr><td>3. El especialista despliega el A.A.G.</td><td colspan="2">4. El sistema muestra las áreas de análisis existentes dentro del A.A.G.</td></tr>
<tr><td>5. El especialista despliega el área de análisis A.A. Explotación.</td><td colspan="2">6. El sistema muestra las áreas de análisis contenidas dentro del A.A seleccionada.</td></tr>
<tr><td>7. El especialista selecciona el área de análisis A.A Reportes de Estados.</td><td colspan="2">8. El sistema muestra el listado de reportes asociados al área de análisis Reportes de Estados.</td></tr>
<tr><td>9. El especialista selecciona el reporte ‘‘Reporte de Estados’’.</td><td colspan="2">10. El sistema muestra el reporte seleccionado (ver CU Visualizar reporte).</td></tr>
<tr><td colspan="3">Opciones de reportes de Analizar Reporte de Estados</td></tr>
<tr><td>Perspectivas de análisis</td><td colspan="2">Posibles resultados</td></tr>
<tr><td></td><td>Medidas</td><td>Periodicidad</td></tr>
<tr><td>Dimensiones disponibles relacionadas con la vista materializada v_metricas_estado.
Estructura
Marca de Tiempo
Temporal del día</td><td>Medidas disponibles en el caso de uso “Analizar Reporte de Estados”.
Estados Ociosos
Estados Activos
% Ocioso
% Activo</td><td>Rango de tiempo en que se solicitan las variables de salida:
Por solicitud</td></tr>
<tr><td colspan="3">Prototipo de interfaz</td></tr>
</table>



		Laboratorio							
		UCI				Proyecto			
		Indicadores				Indicadores			
Fecha	Marca de tiempo	Estados Ociosos	Estados Activos	% Ocioso	% Activo	Estados Ociosos	Estados Activos	% Ocioso	% Activo
Todos	Todas	5,070	5,322	49%	51%	5,070	5,322	49%	51%
	Madrugada	1,280	1,096	54%	46%	1,280	1,096	54%	46%
	Mañana	1,319	1,586	45%	55%	1,319	1,586	45%	55%
	Tarde	919	1,356	40%	60%	919	1,356	40%	60%
	Noche	1,552	1,284	55%	45%	1,552	1,284	55%	45%
2013	Todas	5,070	5,322	49%	51%	5,070	5,322	49%	51%
	Madrugada	1,280	1,096	54%	46%	1,280	1,096	54%	46%
	Mañana	1,319	1,586	45%	55%	1,319	1,586	45%	55%
	Tarde	919	1,356	40%	60%	919	1,356	40%	60%
	Noche	1,552	1,284	55%	45%	1,552	1,284	55%	45%

3.2.3 Modificaciones en el subsistema de almacenamiento

La implementación de un subsistema de almacenamiento es de vital importancia para todos los sistemas de inteligencia de negocios que necesiten trabajar con datos históricos. En el documento “Modelo de Datos” del sistema SIMON se exponen detalladamente los modelos conceptual, lógico y físico del ODS de tecnología, el cual conforma el subsistema de almacenamiento de dicho sistema. El mismo se encuentra implementado en PostgreSQL y cuenta con un conjunto de dimensiones, metadatos y hechos, los cuales fueron identificados durante el proceso de diseño una vez realizado el levantamiento de requisitos.

Cambios realizados al ODS

Con vistas a añadir la nueva información de los estados de las estaciones de trabajo fue necesario realizar modificaciones al ODS, manteniendo las pautas planteadas en el documento mencionado en la sección anterior. Teniendo en cuenta que la información de los estados es dinámica, se decide añadir un nuevo campo a la tabla que corresponde al hecho **metricas_dinamicas** llamado **estado**, el cual almacenará los estados de las estaciones de trabajo, en una cadena con el siguiente formato: **idle=[cantidad de estados ociosos] ; nodidle=[cantidad de estados ociosos].** Además se decide implementar una nueva vista materializada que permitirá agilizar el proceso de búsqueda de información. A continuación se muestra cómo quedaría el modelo físico del hecho **metricas_dinamicas** una vez añadido el nuevo campo, ver figura 3.5.

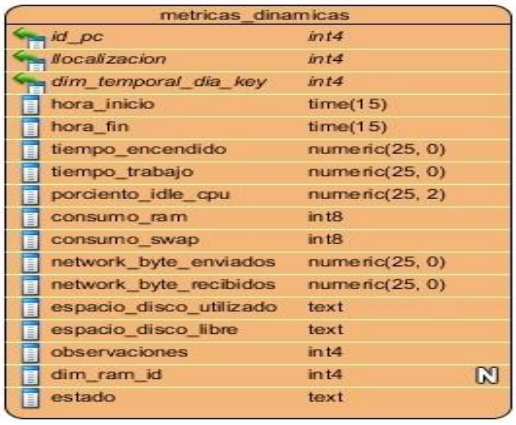


Figura 3.5: Modelo físico del hecho metricas_dinamicas

Vista materializada v_metricas_estado

Una vista materializada constituye una tabla física que se comporta como una vista. Se usa para agilizar las búsquedas de información cuando existen grandes volúmenes de datos en una base de datos, y con el objetivo de tener análisis comunes y frecuentes ya calculados y disponibles para visualizar.

```
CREATE OR REPLACE VIEW staging_tecnologia.v_metricas_estado AS
   SELECT
        dk.dim_temporal_dia_key
        ,lb.dim_laboratorio_id
        ,d.dim_marca_tiempo_lab_id
        ,sum((string_to_array((string_to_array(m.estado_cpu,';'::text))[1],','::text))[2]::integer) AS idle
        ,sum((string_to_array((string_to_array(m.estado_cpu,';'::text))[2],','::text))[2]::integer) AS noidle
   FROM
        staging_tecnologia.metricas_dinamicas m JOIN dimensiones.dim_temporal_dia dk
        ON m.dim_temporal_dia_key = dk.dim_temporal_dia_key JOIN dimensiones.dim_laboratorio lb
        ON m.localizacion = lb.dim_laboratorio_id, dimensiones.dim_marca_tiempo_dia d
   WHERE
        m.hora_inicio >= d.hora_inicio AND m.hora_fin <= d.hora_fin
   GROUP BY
         dk.dim_temporal_dia_key
        ,lb.dim_laboratorio_id
        ,d.dim_marca_tiempo_lab_id
   ORDER BY
         dk.dim_temporal_dia_key
        ,lb.dim_laboratorio_id
        ,d.dim_marca_tiempo_lab_id;
```

Figura 3.6: Sentencia SQL para la creación de la vista v_metrica_estado

La figura 3.6 muestra la sentencia SQL necesaria para la creación de la vista materializada **v_metricas_estado** en PostgreSQL. La misma cuenta con la información resumida de los estados ociosos y activos agrupados por día, momento del día y por laboratorio.

3.2.4 Modificaciones en el subsistema de integración

Para el correcto funcionamiento del ODS de tecnología es de vital importancia la integración de los datos provenientes de las bases de datos del sistema T-arenal, que contienen las métricas de las estaciones de trabajo. La integración de datos consiste en la combinación de los datos que residen en diferentes fuentes para brindar a los usuarios una vista unificada de los mismos. Esta integración se puede lograr mediante los procesos de extracción, transformación y carga. A continuación se explican los principales elementos tenidos en cuenta en la implementación del subsistema de integración de SIMON, así como las causas que dieron origen a la necesidad de implementar un nuevo proceso ETL.

Necesidad de ealizar un nuevo proceso ETL

Todo proceso ETL utiliza una fuente de datos para realizar la extracción y transformación de los datos, para luego realizar la carga de estos a otra base de datos, mercado o almacén de datos. Como se había planteado anteriormente fue necesario el rediseño de la base de datos que almacena la información de las estaciones de trabajo brindada por el sistema T-arenal, disminuyendo el número de tablas y relaciones entre las mismas, así como incrementado el número de métricas almacenadas. Teniendo en cuenta este planteamiento, unido a los cambios realizados en el hecho **metricas_dinamicas**, se hizo necesario realizar un nuevo proceso ETL para garantizar la carga de los datos hacia el ODS.

Extracción, Transformación y Carga de los Datos

Como ha sido definido en la versión 1.0 del sistema SIMON, el proceso ETL cuenta con un conjunto de pasos encaminados a lograr un correcto y eficiente proceso de extracción, transformación y carga de los datos hacia el ODS. Siguiendo este flujo previamente definido fue implementado el nuevo proceso, realizando importantes transformaciones en la extracción de los datos, teniendo en cuenta que la fuente de datos fue totalmente modificada.

A continuación se enumeran los pasos principales que se llevan a cabo en el proceso.

1- Inicialización de las variables de entorno.

2- Extracción de los datos de la fuente hacia el área temporal.

3- Carga de las dimensiones.

4- Transformación de los datos, eliminación de valores nulos.

5- Carga de los hechos.

6- Borrado de los datos de la fuente.

Implementación de los trabajos y las transformaciones

Los trabajos (Jobs) y las transformaciones constituyen un conjunto de tareas que tienen como objetivo realizar una operación específica. Las transformaciones están compuestas por un conjunto de pasos que representan la unidad más pequeña del proceso, los cuales se encuentran unidos a través de saltos. Los Jobs son los encargados de ejecutar una o varias de las transformaciones diseñadas, siguiendo una secuencia lógica de ejecución. En la presente investigación se implementaron varias transformaciones que permiten realizar la carga de los datos hacia el ODS, así como un conjunto de Jobs que guían y controlan el flujo de ejecución de las transformaciones. La figura 3.7 muestra el flujo de ejecución del Job principal, el cual garantiza la ejecución de otros Jobs, dirigidos a lograr un correcto proceso ETL.

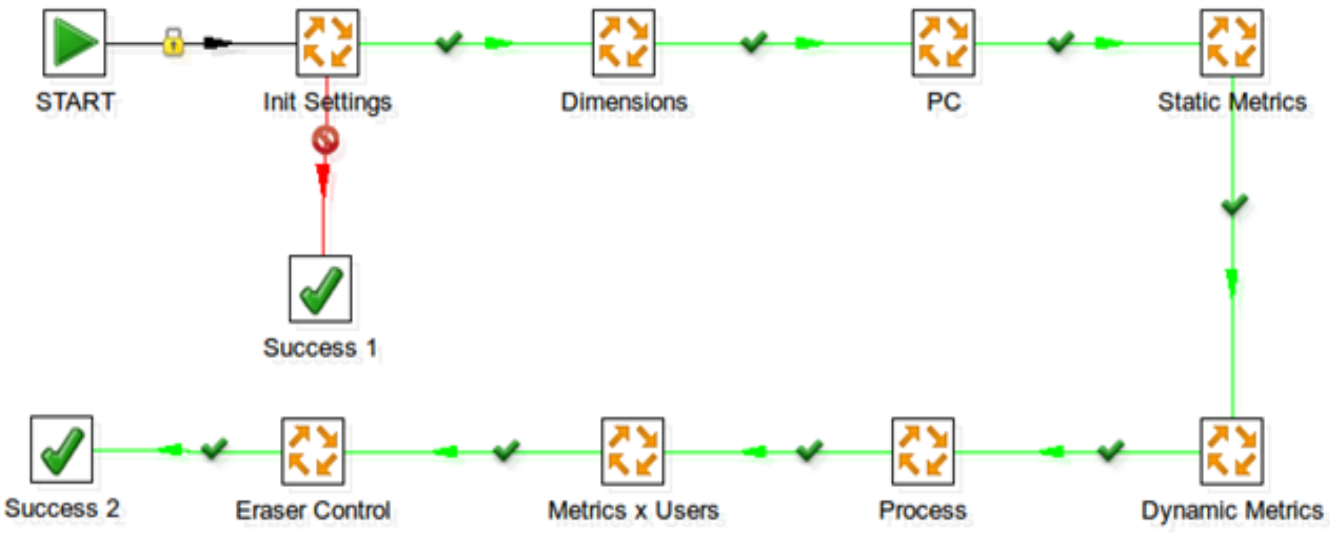


Figura 3.7: Job principal del proceso ETL

3.2.5 Modificaciones en el Subsistema de Visualización

La visualización de la información mediante tablas y gráficas, ofrece un entorno de análisis, monitoreo y control que constituye un elemento fundamental en el proceso de toma de decisiones.

Implementación de los cubos OLAP

Existen diversas estructuras de datos, a través de las cuales se puede representar la información contenida en un almacén de datos, de las cuales, los cubos multidimensionales constituyen una de las estructuras más utilizadas. Un cubo multidimensional representa o convierte los datos planos que se encuentran en las filas y columnas de una tabla. Por esta razón el sistema SIMON v1.0 cuenta con un conjunto de cubos OLAP[13] representando cada tabla de hechos, los cuales se encuentran almacenados en un fichero (tecnología.xml). Para los nuevos reportes fue necesario agregar un nuevo cubo OLAP correspondiente a la vista creada en el ODS, la cual almacena la información referente a los estados de las estaciones de trabajo, ver figura 3.8.

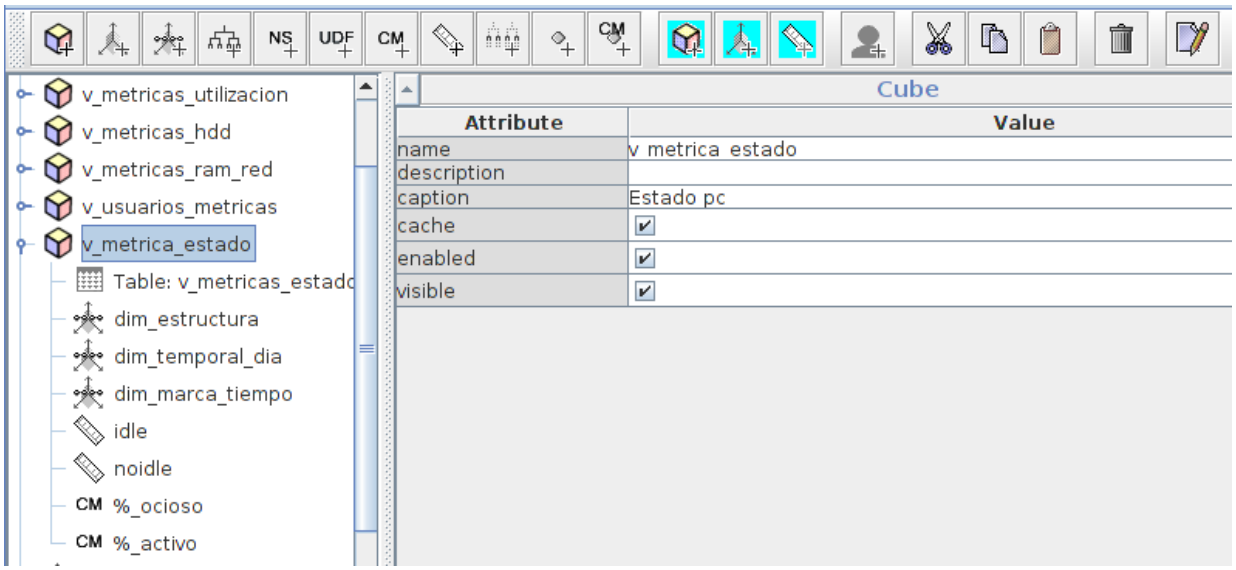


Figura 3.8: Diseño de Cubos OLAP

[13] *Online Analytical Processing* o procesamiento Analítico en Línea, es una base de datos multidimensional, en la cual el almacenamiento físico de los datos se realiza en un vector multidimensional.

Implementación de los reportes

Los reportes constituyen la unidad básica para mostrar la información almacenada, estos representan la información que el cliente desea que se muestre como finalidad del producto. En el presente trabajo se implementaron un total de 4 reportes, los cuales brindan una medida del aprovechamiento de los recursos computacionales a partir del análisis de los estados de las estaciones de trabajo. A continuación se muestra un reporte que brinda la información de los estados activos y ociosos de las estaciones de trabajo por cada hora del día, ver figura 3.9.

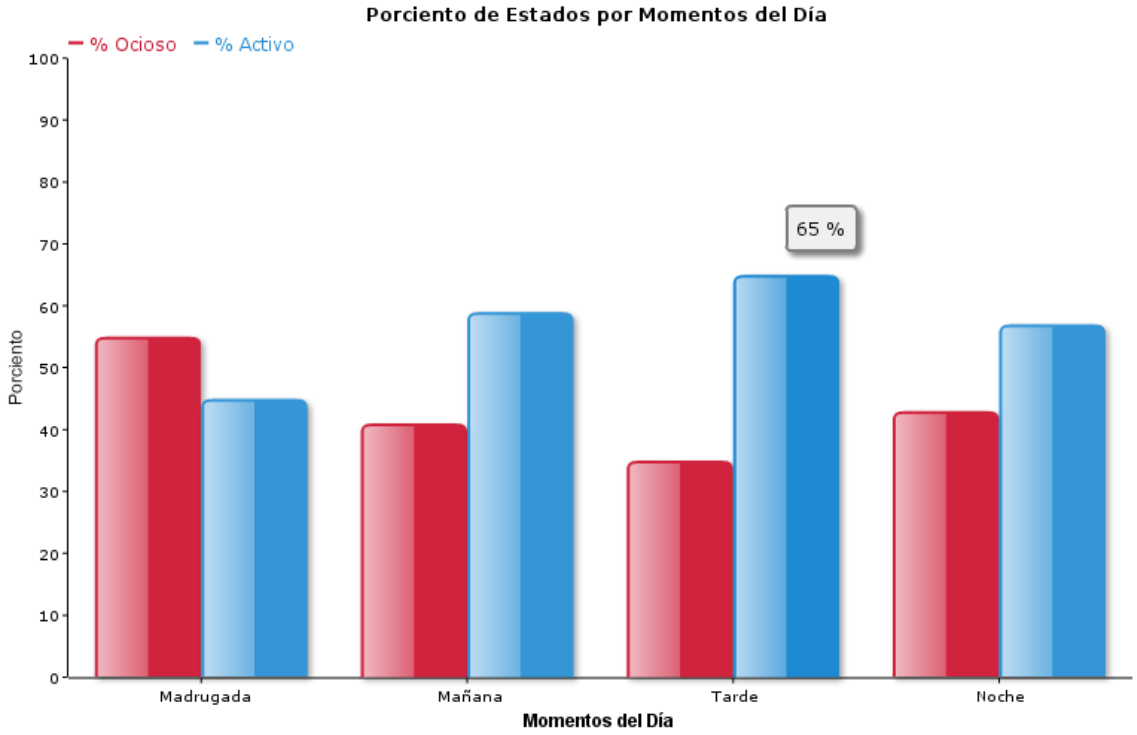


Figura 3.9: Reporte del porciento de estados por momentos del día

Los reportes implementados permiten realizar un análisis de la información de forma más detallada, apoyando de esta forma el proceso de toma de decisiones. En la gráfica mostrada en la figura 3.10 se puede observar el porciento de estados en diferentes momentos del día. En la misma se evidencia que durante la madrugada, el porciento de estaciones de trabajo en estado ocioso es mayor que el porciento de estaciones de trabajo en estado activo, significando esto que existe desaprovechamiento de los

recursos de cómputo. Por otra parte, durante la mañana, tarde y noche el porciento de estaciones de trabajo en estado activo es mayor que el porciento de estaciones de trabajo en estado ocioso, evidenciándose que durante la tarde existe mayor aprovechamiento de los recursos.

Conclusiones del Capítulo

Para lograr la incorporación del modelo de clasificación de las estaciones de trabajo, fue necesario realizar un nuevo diseño del módulo de tecnología del sistema T-arenal. El modelo computacional obtenido fue incorporado al servidor, el cual garantiza el almacenamiento de la información enviada por las estaciones de trabajo, unido a la clasificación de las mismas según su estado.

La identificación de los requisitos de información permitió determinar los cambios necesarios a realizar en los subsistemas de almacenamiento, integración y visualización del mercado de datos de SIMON. Se implementó un nuevo subsistema de integración que garantiza la carga de los datos provenientes de la base de datos de recolección hacia el ODS.

Finalmente fueron implementados varios reportes para garantizar la disponibilidad de la información de manera eficiente y organizada, facilitando el proceso de toma de decisiones.

Conclusiones Generales

Luego de la realización del presente trabajo de diploma se concluye lo siguiente:

- El estudio realizado sobre las técnicas de aprendizaje automatizado constituyó el punto de partida para la selección de los clasificadores candidatos, para la obtención del modelo computacional.
- A partir de las pruebas realizadas a las técnicas de clasificación utilizadas, se determinó que el clasificador LADTree provee buenos resultados en la clasificación de los estados de las estaciones de trabajo, resultando ser el más conveniente para la resolución del problema planteado en la presente investigación; permitiendo obtener un modelo de clasificación con un porciento de acierto aceptable en los casos probados.
- La implementación del modelo obtenido permitió su incorporación al sistema SIMON, brindando una funcionalidad que permite evaluar el verdadero aprovechamiento que se hace de las estaciones de trabajo.
- Se implementó un conjunto de reportes que permiten garantizar la disponibilidad de la información de manera eficiente y organizada, facilitando el proceso de toma de decisiones.

Referencias Bibliográficas

1. **Dias de Oliveira Santos, Victor.** *AUTOMATIC ESSAY SCORING: MACHINE LEARNING MEETS APPLIED LINGUISTICS.* 2011.

2. **Gorunescu, Florin.** *Data Mining Concept, Models and Techniques.* Berlin Heidelberg: Springer, 2011. ISBN 978-3-642-19720-8.

3. **Bing, Liu.** *Web Data Mining. Exploring Hyperlinks, Contents and Usage Data.* New York : Springer, 2011. ISBN 978-3-642-19459-7.

4. **Kleisinger, Lucrecia, H. Valdez, Gretchen y Monzón, Alberto D.** *Comparación de Topologías MLP y LVQ de Redes Neuronales para la Detección de Arritmias Ventriculares.*

5. **Tamarit Muñoz, Indira y Miranda Bermudez, Ana.** *Propuesta del Módulo Decisor del Motor de Clasificación Inteligente por Contenidos (MOCIC).* 2009.

6. **C Hernández, Ferri.** *Introducción a la Minería de Datos.* España : Pearson, 2004.

7. **Holmes, Geoffrey, Pfahringer, Bernhard, Kirkby, Richard, Frank, Eibe y Hall, Mark.** *A Logistic Boosting Approach to Inducting Multiclass Alternating Decision Trees.* Hamilton, New Zealand : Working Paper Series, 2002. ISSN 1170-487X.

8. **Conference, 12th Pacific-Asia.** *Advances in Knowledge Discovery and Data Mining.* Osaka, Japan : springer, 2008.

9. **Alonso Llombart, Oscar.** *BI: Inteligencia aplicada al negocio.* España : DAA Contenidos Digitales, S.L, 2007.

10. **Kimball, Ralph.** *The Data Warehouse ETL Toolkit.* Indianapolis : Wiley Publishing, Inc., 2004. eISBN: 0-764-57923-1.

11. **Roldán, María Carina.** *Pentaho 3.2 Data Integration.* Birmingham : Packt Publishing, 2010. ISBN 978-1-847199-54-6.

12. **Massie, Matt, Li, Bernard, Nicholes, Brad y Vuksan, Vladimir** *Monitoring with Ganglia.* s.l. : O'Reilly Media, Inc., 2013. ISBN: 978-1-449-32970-9.

13. **Hyperic.** Sitio Oficial de Hyperic. [En línea] [Citado el: 12 de 1 de 2013.] http://www.hyperic.com.

14. **Pandora FMS.** Pandora Flexible Monitoring System. [En línea] [Citado el: 12 de 1 de 2013.] http://www.pandorafms.org/.

15. **Mauri Garcia, Antonio y Pérez Falcón, Abel.** *Estrategia de Migración hacia Sistemas Gestores de Bases de Datos Libres.* 2008.

16. **The PostgreSQL Global Development Group.** PostgreSQL. [En línea] [Citado el: 10 de 1 de 2013.] http://www.postgresql.org.

17. **PgAdmin.** PostgreSQL administration and management tools. [En línea] [Citado el: 15 de 1 de 2013.] http://www.pgadmin.org/.

18. **H. Witten, Ian y Frank, Eibe.** *Data Mining, Practical Machine Learning Tools and Techniques.* San Francisco : Elsevier, 2005. ISBN: 0-12-088407-0.

19. **Bonanata, Maximiliano.** *Programación y Algoritmos.* Buenos Aires, Argentina : MP Ediciones S.A.

20. **Oracle.** Sitio Oficial de Java. *Java.* [En línea] [Citado el: 28 de 11 de 2012.] http://www.java.com/es/about/.

21. **Larman, Craig.** *Applying UML and Pattern. An Introduction to Object-Oriented Analysis and Design.* EUA : Prentice Hall, Inc. ISBN 0-13-748880-7.

22. **Visual Paradigm.** UML tool for software application development. [En línea] [Citado el: 13 de 1 de 2013.] ent. http://www.visual-paradigm.com/product/vpuml/.

23. **García Jacas, César Raúl y Miralles Taset, Daniel Marino.** *T-arenal v2.0: Desarrollo del back - end.* 2009.

24. **Pentaho.** Pentaho BI Platform Community Edition. *Pentaho BI Platform and Server.* [En línea] 2012. [Citado el: 11 de 12 de 2012.] http://community.pentaho.com/projects/bi_platform/.

25. **Pentaho**. Pentaho Data Integration. [En línea] 2011. [Citado el: 9 de 12 de 2012.] http://www.pentaho.com/explore/products/.

26. **OpenUP.** Introduction to OpenUP. [En línea] [Citado el: 15 de 1 de 2013.] http://epf.eclipse.org/wikis/openup/.

27. **Wolpert, David H., y Macready, William G.** *No free lunch theorems for optimization.* 1, s.l. : IEEE Transactions on Evolutionary Computation, 1997, Vol. 1.

28. **Sammut, Claude y I. Webb, Geoffrey.** *Encyclopedia of Machine Learning.* New York : Springer, 2011. ISBN 978-0-387-30768-8.

Bibliografía

1. **Dias de Oliveira Santos, Victor.** *AUTOMATIC ESSAY SCORING: MACHINE LEARNING MEETS APPLIED LINGUISTICS.* 2011.

2. **Gorunescu, Florin.** *Data Mining Concept, Models and Techniques.* Berlin Heidelberg: Springer, 2011. ISBN 978-3-642-19720-8.

3. **Bing, Liu.** *Web Data Mining. Exploring Hyperlinks, Contents and Usage Data.* New York : Springer, 2011. ISBN 978-3-642-19459-7.

4. **Kleisinger, Lucrecia, H. Valdez, Gretchen y Monzón, Alberto D.** *Comparación de Topologías MLP y LVQ de Redes Neuronales para la Detección de Arritmias Ventriculares.*

5. **Tamarit Muñoz, Indira y Miranda Bermudez, Ana.** *Propuesta del Módulo Decisor del Motor de Clasificación Inteligente por Contenidos (MOCIC).* 2009.

6. **C Hernández, Ferri.** *Introducción a la Minería de Datos.* España : Pearson, 2004.

7. **Holmes, Geoffrey, Pfahringer, Bernhard, Kirkby, Richard, Frank, Eibe y Hall, Mark.** *A Logistic Boosting Approach to Inducting Multiclass Alternating Decision Trees.* Hamilton, New Zealand : Working Paper Series, 2002. ISSN 1170-487X.

8. **Conference, 12th Pacific-Asia.** *Advances in Knowledge Discovery and Data Mining.* Osaka, Japan : springer, 2008.

9. **Alonso Llombart, Oscar.** *BI: Inteligencia aplicada al negocio.* España : DAA Contenidos Digitales, S.L, 2007.

10. **Kimball, Ralph.** *The Data Warehouse ETL Toolkit.* Indianapolis : Wiley Publishing, Inc., 2004. eISBN: 0-764-57923-1.

11. **Roldán, María Carina.** *Pentaho 3.2 Data Integration.* Birmingham : Packt Publishing, 2010. ISBN 978-1-847199-54-6.

12. **Massie, Matt, Li, Bernard, Nicholes, Brad y Vuksan, Vladimir** *Monitoring with Ganglia.* s.l. : O'Reilly Media, Inc., 2013. ISBN: 978-1-449-32970-9.

13. **Hyperic.** Sitio Oficial de Hyperic. [En línea] [Citado el: 12 de 1 de 2013.] http://www.hyperic.com.

14. **Pandora FMS.** Pandora Flexible Monitoring System. [En línea] [Citado el: 12 de 1 de 2013.] http://www.pandorafms.org/.

15. **Mauri Garcia, Antonio y Pérez Falcón, Abel.** *Estrategia de Migración hacia Sistemas Gestores de Bases de Datos Libres.* 2008.

16. **The PostgreSQL Global Development Group.** PostgreSQL. [En línea] [Citado el: 10 de 1 de 2013.] http://www.postgresql.org.

17. **PgAdmin.** PostgreSQL administration and management tools. [En línea] [Citado el: 15 de 1 de 2013.] http://www.pgadmin.org/.

18. **H. Witten, Ian y Frank, Eibe.** *Data Mining, Practical Machine Learning Tools and Techniques.* San Francisco : Elsevier, 2005. ISBN: 0-12-088407-0.

19. **Bonanata, Maximiliano.** *Programación y Algoritmos.* Buenos Aires, Argentina : MP Ediciones S.A.

20. **Oracle.** Sitio Oficial de Java. *Java.* [En línea] [Citado el: 28 de 11 de 2012.] http://www.java.com/es/about/.

21. **Larman, Craig.** *Applying UML and Pattern. An Introduction to Object-Oriented Analysis and Design.* EUA : Prentice Hall, Inc. ISBN 0-13-748880-7.

22. **Visual Paradigm.** UML tool for software application development. [En línea] [Citado el: 13 de 1 de 2013.] ent. http://www.visual-paradigm.com/product/vpuml/.

23. **García Jacas, César Raúl y Miralles Taset, Daniel Marino.** *T-arenal v2.0: Desarrollo del back - end.* 2009.

24. **Pentaho.** Pentaho BI Platform Community Edition. *Pentaho BI Platform and Server.* [En línea] 2012. [Citado el: 11 de 12 de 2012.] http://community.pentaho.com/projects/bi_platform/.

25. **Pentaho**. Pentaho Data Integration. [En línea] 2011. [Citado el: 9 de 12 de 2012.] http://www.pentaho.com/explore/products/.

26. **OpenUP.** Introduction to OpenUP. [En línea] [Citado el: 15 de 1 de 2013.] http://epf.eclipse.org/wikis/openup/.

27. **Wolpert, David H., y Macready, William G.** *No free lunch theorems for optimization.* 1, s.l. : IEEE Transactions on Evolutionary Computation, 1997, Vol. 1.

28. **Sammut, Claude y I. Webb, Geoffrey.** *Encyclopedia of Machine Learning.* New York : Springer, 2011. ISBN 978-0-387-30768-8.

29. **Krugman, Paul y Wells, Robin.** Introducción a la Economia: Macroeconomía. *Google Books.* [En línea] [Citado el: 14 de 04 de 2013.] http://books.google.com.cu/books?isbn=8429126325.

30. **LAROSE, DANIEL T.** *DISCOVERING KNOWLEDGE IN DATA: An Introduction to Data Mining.* Hoboken, New Jersey. : John Wiley & Sons, Inc., 2005.

31. **F. Hair, Joseph, E. Anderson, Rolph, L. Tatham, Roanld y C. Black, William.** *Análisis Multivariante.* Madrid : Prentice Hall, 1999.

32. **Johnson, Dallas E.** *Métodos Multivariados Aplicado al Análisis de Datos.* s.l. : International Thomson Editores, S.A, 200. ISBN 968-7529-90-3.

33. **Ayed Mezghani, D. Ben, Boujelbene, S. Zribi y Ellouze, N.** *Evaluation of SVM Kernels and Conventional Machine Learning Algorithms for Speaker Identification.* 3, s.l. : International Journal of Hybrid Information Technology, 2010, Vol. 3.

34. **The R-Proyect.** The R Project for Statistical Computing. [En línea] [Citado el: 15 de 1 de 2013.] http://www.r-project.org/.

35. **University of Waikato.** Machine Learning Group at the University of Waikato. *Weka: Data Mining Software in Java.* [En línea] [Citado el: 14 de 1 de 2013.] http://www.cs.waikato.ac.nz/ml/weka/.

36. **Monitis.** 11 Top Server Management & Monitoring Software. [En línea] [Citado el: 10 de 1 de 2013.] http://blog.monitis.com/index.php/2011/02/22/11-top-server-management-monitoring-software/.

Glosario de Términos

- **Área de Análisis:** la agrupación de información según su propósito, aunque el criterio depende de las necesidades de la institución o empresa donde se aplica el sistema.
- **Entrenamiento:** acción que se realiza para el aprendizaje de ciertas muestras.
- **Estación de Trabajo:** se refiere a un ordenador personal (PC) o computadora.
- **GRASP:** patrones generales de software para asignar responsabilidades (GRASP) describen los principios fundamentales del diseño de objetos para la asignación de responsabilidades.
- **Hiperplano:** concepto de geometría, es una generalización del concepto de plano.
- **Inteligencia Artificial:** es la ciencia que enfoca su estudio a lograr la comprensión de entidades inteligentes.
- **Matriz de Dispersión:** gráfica que permite evaluar el tipo de relación existente entre dos variables.
- **Mercado de Datos (Data Mart):** base de datos departamental, especializada en el almacenamiento de los datos de un área de negocio específica. Se caracteriza por disponer la estructura óptima de datos para analizar la información al detalle desde todas las perspectivas que afecten a los procesos de dicho departamento.
- **Metaheurística:** el término heurística proviene de la palabra griega "heuriskein" que significa “hallar, inventar”. El término metaheurística se obtiene de anteponer a heurística el prefijo meta que significa “más allá” o “a un nivel superior”, por tanto las metaheurísticas son estrategias inteligentes para diseñar o mejorar procedimientos heurísticos para buscar una solución óptima o casi óptima.
- **Métrica:** es una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado. (*IEEE Standard Glossary of Software Engineering Terms*).
- **Minería de Datos:** proceso en el que se convierten los datos en conocimiento.
- **Modelo Computacional:** conjunto de representaciones que pueden ser procesadas por un computador y así proveer datos simulados, comparables con datos obtenidos de personas reales.

- **Predicción:** es la función de la minería que predice los valores posibles de datos faltantes o la distribución de valores de ciertos atributos en un conjunto de objetos.
- **Sesgo:** propiedad de una muestra estadística que hace que los resultados sea representativos para toda la población.
- **Tabla de Hechos (Fact Tables):** tabla central de un esquema dimensional que contiene los valores de las medidas del negocio.
- **Tupla:** una ocurrencia de artículo o tupla consiste en un grupo de ocurrencias de campos relacionados, representando una asociación entre ellos.
- **Validación Cruzada:** método de estimación aleatorio que indica el porciento de exactitud de los datos que serán correctamente clasificados en una muestra.