UNIVERSIDAD DEL BÍO BÍO

FACULTAD DE CIENCIAS EMPRESARIALES

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN Y TECNOLOGÍAS DE LA INFORMACIÓN

USO DE TÉCNICAS DE MACHINE LEARNING PARA PREDECIR EL RENDIMIENTO ACADÉMICO DE LOS ESTUDIANTES DE LA CARRERA DE INGENIERÍA CIVIL EN INFORMÁTICA DE LA UNIVERSIDAD DEL BÍO BÍO, CHILLÁN

Trabajo de Titulación Presentado por Gaspar Soto Romero para Obtener el Título de Ingeniero Civil en Informática

---

Profesora Guía
Carola Figueroa Flores
Magister en Ciencias de la Computación

"El talento humano para reconocer patrones es una arma de doble filo: somos especialmente buenos encontrando patrones, aun cuando realmente no están ahí"
Neil deGrasse Tyson

Agradecimientos

Primeramente gracias a Dios pues es él quien me ha ayudado durante estos cinco años de carrera, sin mi fe en él muy poco hubiera podido hacer.

Gracias a mi familia por ser la motivación que me hizo falta a lo largo de este proceso.

A mis profesores, cada uno de los cuales me enseñó no sólo informática, sino que valores y formaron en mi el caracter que hizo falta.

A mi profesora guía, por la paciencia que me tuvo durante este último semestre y por la disposición que siempre mostró a ayudarme y corregirme.

Y a mi amada Belen, por su cariño y su deseo de estar siempre conmigo.

Resumen

Este proyecto se presenta para dar conformidad a los requisitos exigidos por la Universidad de Bío Bío en el proceso de titulación de la carrera de Ingeniería Civil en Informática . El proyecto se titula "Uso de técnicas de Machine Learning para Predecir el Rendimiento Académico de los estudiantes de la carrera de Ingenieria Civil en Informática de la Universidad del Bío Bío, Chillán".

La capacidad de identificar qué factores influyen en los estudiantes es muy importante, pues ésta permitiría tomar las medidas correctivas necesarias para evitar el fracaso de los alumnos, además de guiarlos en las áreas en que se encuentran más afectados.

Este proyecto da a conocer los detalles del desarrollo y uso de una herramienta de clasificación de propósito general usada en el intento de predecir el rendimiento académico de estudiantes de la Universidad del Bio Bio guiándose por los hallazgos de investigación que se señalan en la literatura como posibles factores asociados al rendimiento académico.

Para esto, se ha creado una herramienta de software que utiliza la técnica conocida como *k Nearest Neigbors* con la capacidad de clasificar variables nominales, además considerando la naturaleza subjetiva del problema se ha conseguido alcanzar niveles de acierto aceptables en la tarea de predecir el rendimiento académico de los estudiantes, se consiguió, en el mejor de los casos, una tasa de acierto del $60\,\%$ e indices de error cuadrático medio del $0,4$.

Abstract

This project is presented to provide conformity to the requirements of the University of Bío Bío in the process of titling on degree "Ingeniería Civil en Informática". The project is titled "Using machine learning techniques to predict the academic performance of students studying civil engineering in computer science of the University of Bío Bío, Chillán".

The ability to identify which factors influence students is very important because it would take the necessary corrective measures to prevent the failure of students, in addition to guide them in the areas that are most affected.

This project discloses the details of the development and use of a tool classification general purpose used in the attempt to predict academic performance of students from the Universidad del Bío Bío guided by research findings as identified in the literature as possible factors associated with academic performance.

For this, it has created a software tool that uses the technique known as *k Nearest Neigbors* with the ability to classify nominal variables, also considering the subjective nature of the problem has achieved acceptable levels of success in the task of predicting the academic performance of students was achieved, in the best case, a hit rate of $60\,\%$ and a mean square error of $0{,}4$.

Índice general

Índice de figuras

Índice de cuadros

Lista de algoritmos

Capítulo 1

Introducción

Hoy en dia vivimos en la era de las computadoras, estamos rodeados de ellas, de su influencia y de su impacto en nuestra sociedad (sea este positivo o negativo), hoy más que en ninguna epoca anterior de la humanidad, tenemos libre acceso a bancos gigantescos de información y con tan sólo acceder a una computadora podemos adentrarnos en un mundo de conocimiento inmenso, además, si a eso le añadimos el auge de las redes computacionales, nuestras posibilidades se vuelven infinitas.

Pero no siempre lo que se consigue en primera instancia es conocimiento, por lo general, el hecho de vivir en un mundo computarizado, automatizado y globalizado es sinonimo de vivir rodeado de datos, y estos no son necesariamente información, al menos no información útil.

Procesar cantidades pequeñas de datos es tarea fácil para una persona, manejar un par de variables es parte del trabajo diario de nuestra mente, podemos recordar un par de números telefónicos e identificar a quienes pertenecen, o podemos estudiar los razgos físicos y psicológicos de nuestros familiares y amigos y construir un mapa mental lo suficientemente eficaz para reconocerlos entre otras personas, sin embargo, el problema comienza a tornarse complicado cuando esa cantidad de variables y de datos aumenta y en el último tiempo han aumentado considerablemente.

Y es aquí donde el *Machine Learning* entra a jugar su rol más importante, tareas como el análisis de datos, la *Minería de Datos* y el reconocimiento de patrones que pueden realizarse con las técnicas de esta área vuelven al *Machine Learning* muy atractivo para los sectores que trabajan con grandes cantidades de datos.

En este trabajo se aborda una problemática relacionada con el rendimiento académico de los estudiantes de la carrera de Ingeniería Civil Informática, cuya finalidad es comprender el impacto de los factores internos y externos en ellos, y cómo predecir su rendimiento, de tal manera de aplicar las acciones necesarias para mejorarlo.

Este trabajo no es tarea fácil, son muchos y muy variados los aspectos que se ven involucrados y que deben ser analizados, este problema se ha intentado abordar de diferentes maneras en variadas casas de estudio tales como la Universidad de Atacama, Universidad de Talca, entre otras.

El aspecto diferenciador del informe presentado a continuación es la técnica que se utilizó, *K Nearest Neighbors*, con esta estratégia se ha conseguido una tasa de acierto en la predicción del orden del 60 % en el mejor de los casos.

A continuación se presenta una descripción general del problema enfrentado.

1.1. Descripción del problema

El problema estudiado en este trabajo es el reconocimiento de los factores tanto internos como externos que inciden en el rendimiento académico de los estudiantes, específicamente de los estudiantes de la carrera de Ingeniería Civil Informática, para así intentar predecir cómo será el rendimiento de futuros estudiantes que compartan caracteristicas similares.

El concepto de rendimiento académico es un termino complejo e interpretado de diferentes maneras, por ello, se considera un concepto multidimensional, relativo y contextual, (Gonzáles-Lopez, 2004, citado en Miquel, J., Expósito, M. y Sempere, S., 2014).

Sin embargo, en el contexto de este problema, el concepto de rendimiento académico es definido como la calificación final obtenida por el estudiante durante su participación en alguna de las asignaturas que el alumno haya cursado. (Tejedor, 1998, citado en Miquel, J., Expósito, M. y Sempere, S., 2014).

El intento de predecir el rendimiento académico no es tarea fácil puesto que éste se ve influenciado por caracteristicas subjetivas que complican el trabajo, es por eso que se han realizado distintos esfuerzos, cada uno con caracteristicas propias, para conseguir un resultado de predicción aceptable, estos trabajos se describen brevemente en la siguiente sección.

1.2. Descripción del proyecto

Esta sección tiene como propósito dar a conocer el objetivo general y los objetivos específicos del proyecto.

1.2.1. Objetivo general del proyecto

El objetivo primordial del proyecto es la creación de una herramienta de software que permita al usuario predecir el rendimiento académico de los estudiantes de la carrera de Ingeniería Civil en Informática de la Universidad del Bío-Bío, Chillán, usando para esto, el algoritmo *k nearest neigbors*.

Si bien el objetivo general señala la creación de una herramienta de software para abarcar un problema específico, vale la pena mencionar, que dicha herramienta al ser un software clasificador podría ser usada para otras tareas de características similares, siempre y cuando se siga la metodología de aplicación descrita más adelante en el capítulo 4.

1.2.2. Objetivos específicos del proyecto

Los objetivos específicos del proyecto son los siguientes:

- Realizar una revisión sistemática de la literatura para conocer y entender los factores tanto internos como externos que influyen en el rendimiento académico de los estudiantes.
- Estudiar el grado de acierto que tiene el algoritmo *k nearest neighbors* para la resolución de este problema en específico.
- Recolectar a través de una encuesta y su posterior análisis los datos que describen el comportamiento y rendimiento de los estudiantes de la Facultad de Ciencias Empresariales.
- Diseñar e implementar una encuesta en línea y hospedarla en el servidor de la carrera de Ingeniería Civil Informática, de tal manera que el aaceso a la información sea hecho de manera segura y confiable.
- Estudiar en qué consiste el *Machine Learning* y sus distintas técnicas en el contexto de este proyecto.
- Ofrecer una herramienta que ayude al análisis del comportamiento académico de los estudiantes para así poder apoyarlos de mejor forma en su desempeño durante el transcurso de sus estudios.

El documento presentado a continuación posee la siguiente estructura, primero se describe el problema de investigación, luego se introduce al lector en los conceptos básicos del *Machine Learning*, posteriormente se mencionan los trabajos relacionados.

Finalmente se describe el estudio empírico, se explica detalladamente la metodología utilizada y los resultados de la aplicación de la técnica *K Nearest Neighbors*.

Capítulo 2

Marco teórico

2.1. Machine Learning

Esta sección tiene la finalidad de proveer al lector de una introducción a los conceptos básicos relativos al *Machine Learning*, los cuales serán aplicados para la solución del problema de investigación abordado en este informe.

Este capítulo está organizado de la siguiente manera, primero se dará a conocer una breve descripción sobre qué es el *Machine Learning* para luego presentar los diferentes tipos de enfoques que existen en el área, más adelante en la sección 2.2 se define qué es la clasificación y cómo trabaja ésta, además de describir algunas de las técnicas existentes en el área.

2.1.1. ¿Qué es el Machine Learning?

Es bien sabido que para solucionar problemas utilizando una computadora hace falta un algoritmo que le indique a la máquina los pasos a seguir para dar solución a dicho problema, un clásico ejemplo de esto sería un algoritmo de ordenamiento, en donde existe una clara definición sobre los datos de entrada y de salida, además son conocidos los pasos que se deben seguir para dar solución a los requerimientos e incluso existen diferentes técnicas disponibles, algunas más eficientes que otras para dar solución al mismo problema.

Sin embargo, existen tareas cotidianas que no tienen un algoritmo determinado asociado a ellas, por ejemplo el hecho de analizar nuestro correo electrónico para seleccionar cuál de ellos es correo basura y cuál no lo es. Por otro lado, al leer los correos claramente podríamos decidir cuál es de nuestro interés y cuáles no

lo son, si bien podemos hacer esto no es fácil explicar cómo lo hacemos y cómo no podemos explicar nuestra habilidad es que no podemos escribir programas que hagan eso, para esto es que existe el *Machine Learning*.

Si bien existen tareas fáciles de abordar usando algoritmos, hay otras que son muy difíciles o imposibles de realizar de esta forma y donde el *Machine Learning* juega un rol importantísimo, como lo son los problemas que se describen a continuación (Dietterich, T., 1996).

Primero, los problemas para los cuáles no existen expertos, supongamos el caso de una gran maquinaria industrial de última tecnología, imaginemos también que esta máquina es tan nueva y tan avanzada que no existen aún expertos para prevenir los posibles errores que ésta sufra, sin embargo un programa que utilice las técnicas de *Machine Learning* podría fácilmente estudiar los patrones de falla de la máquina y predecir los errores antes de que estos ocurran.

Segundo, el caso en el que los expertos humanos si existen pero no tienen la capacidad de explicar su expertise, este es el caso de muchas tareas de perspicacia como por ejemplo el reconocimiento del rostro de un ser querido, la habilidad de leer textos escritos a mano o el entendimiento del lenguaje natural, las personas demuestran tener la habilidad para realizar estas tareas pero si se les consulta cuales son los pasos detallados que siguen son muy pocas las que logran dar una respuesta, afortunadamente los humanos pueden entregarle ejemplos a programas de *Machine Learning* para que estos aprendan y logren identificar futuros ejemplos.

Tercero, aquellos problemas en los que los fenómenos estudiados cambian rápidamente, por ejemplo si se quisiera programar una aplicación para estudiar los gustos de las personas y las compras que éstas hacen para predecir en qué área gastan más sus ingresos habría que reprogramarla frecuentemente debido a la cantidad de cambios que estos gustos y tendencias sufren con el tiempo, sin embargo, si se usaran técnicas de *Machine Learning* el programa en cuestión aprendería de dichas tendencias y se adaptaría a los requerimientos existentes.

Cuarto, aquellas aplicaciones que necesitan ser personalizadas acorde al uso de cada usuario, aquí cabe el ejemplo del filtro de spam mencionado al comienzo de esta sección, tal vez para ciertos usuarios algún correo puede significar spam mientras que para otros el mismo tipo de correos puede ser uno de sus favoritos, es por eso que seria importante tener un programa lo suficientemente inteligente como para adaptarse a los gustos y patrones de uso del usuario.

En el año 1959, Arthur Samuel, definió el concepto de *Machine Learning* como:

> *“ Un campo de estudio que le entrega a los computadores la habilidad de aprender sin haber sido explícitamente programado para eso”*

Tom M. Mitchell es el precursor del *Machine Learning* en la Universidad de Carnegie Mellon. Como un reconocido autor del área, su definición es a menudo citada como:

> *“Un programa informático se dice que aprende de la experiencia E con respecto a alguna clase de tareas T y una medida de rendimiento P, si su desempeño en tareas T, medida por P, mejora con la experiencia E”*

En otras palabras, *Machine Learning* es una rama de la Inteligencia Artificial, con la que podemos diseñar sistemas con la capacidad de aprender de los datos con los que se entrenan de manera que vaya mejorando con la experiencia y el tiempo para perfeccionar un modelo que puede ser usado para predecir los resultados de preguntas basadas en el aprendizaje previo.(Bell, J., 2015).

Las técnicas que abarca el *Machine Learning* tienen una caracteristica distintiva, que las hace estar en la frontera de la *inteligencia artificial*, las *estadísticas* y la *ingeniería de software*, y es que poseen la capacidad de aprender con el tiempo, son capaces en cierta medida de añadir el conocimiento adquirido en trabajos pasados para aplicarlos en tareas futuras, sin duda una gran ventaja.(Harrington, P., 2012).

2.1.2. Tipos de algoritmos para Machine Learning

Existen varios enfoques a la hora de trabajar con algoritmos de Machine Learning, pero los dos más usados son el aprendizaje supervisado y el aprendizaje no supervisado.

2.1.2.1. Aprendizaje supervisado

El aprendizaje supervisado es llamado así porque necesita de una etapa previa de entrenamiento en la cual el programa es alimentado con datos etiquetados tanto con los atributos como con la variable objetivo de cada instancia.

Dentro del aprendizaje supervisado existen dos tareas comunes las cuales son clasificación y regresión, en la tarea de clasificación las variables objetivos, son valores etiquetados de manera nominal, mientras que en las tareas de regresión la variable objetivo tiene un dominio dentro de los números reales y por lo tanto pueden tomar infinitos valores.

2.1.2.2. Aprendizaje no supervisado

En la categoría de aprendizaje no supervisado caben los programas a los cuales les entregamos la colección de datos que queremos analizar para descubrir patrones ocultos en ellos, con la diferencia de que estos datos no están etiquetados. En vez de querer "predecir Y para nuestros datos X" más bien podemos "pedirle al programa que nos provea de información acerca de nuestros datos X"

Las tareas más comunes dentro del aprendizaje no supervisado son el *clustering* y *dimension reduction*, los cuales no son objeto de nuestro estudio por lo que sólo son mencionados en esta parte del documento.

2.2. Clasificación

Una importante tarea en *Machine Learning* es la clasificación, como se mencionó en la sección anterior, las técnicas de clasificación pertenecen al grupo de los algoritmos de aprendizaje supervisado, esta parte del documento está organizada para definir brevemente en que consiste la clasificación, como se lleva a cabo y cuáles son las principales técnicas de clasificación existentes.

2.2.1. ¿Qué es la clasificación?

La clasificación consiste básicamente en reconocer los atributos del elemento a clasificar utilizando el conocimiento adquirido durante la etapa de entrenamiento para poder asignarle un valor a la variable objetivo de dicho elemento.

Las técnicas de clasificación tienen muchas aplicaciones como lo son la detección de fraude, objetivos dentro del marketing, diagnostico médico, etc, además de ser de gran ayuda para predecir comportamientos futuros en muchas áreas.

2.2.2. ¿Cómo trabaja la clasificación?

La clasificación de datos es un proceso que consta de dos etapas, la etapa de aprendizaje (donde es construido el modelo), y la etapa de clasificación (donde el

modelo es usado para predecir las etiquetas de clases de los datos dados). (Han, J., Kamber, M. y Pei, J., 2012).

La etapa de aprendizaje (también conocida como etapa de entrenamiento) es en la cual se construye el modelo utilizando una serie de datos que sirven de base para el conocimiento del algoritmo, para lograr esto, es que se le entrena utilizando un set de entrenamiento, que no son más que tuplas de datos etiquetados tanto en sus atributos como en su variable objetivo. Una tupla X, generalmente es representada por un vector n-dimensional llamado vector de atributos, $X = (x_1, x_2, \cdots, x_n)$.

Una vez que el algoritmo ha sido enseñado con los datos de entrenamiento, ya está listo para ser usado para la clasificación de una variable x, el algoritmo implementado será capaz de tomar los atributos de dicha variable, analizarlos y tomar una decisión de clasificación basándose en los datos de los cuales fue provisto en el paso anterior.

2.2.3. Técnicas de clasificación

Esta sección introduce brevemente a algunas de las técnicas de clasificación más usadas.

2.2.3.1. Árboles de decision

A continuación, una pequeña introducción a la técnica de *Machine Learning* conocida como *Decision Trees* o *Árboles de decisión* por su nombre en español.

Según encuestas recientes los árboles de decisión son la técnica de clasificación más comúnmente usada ya que no hace falta saber mucho sobre *Machine Learning* para entender como estos funcionan. (Seni, G. y Elder, J., 2010, citado en Harrington, P., 2012)

¿Cómo funciona un árbol de decisión?

Si usted ha jugado con un amigo a intentar de adivinar lo que él está pensando haciendo sólo preguntas cerradas (es decir preguntas que pueden ser respondidas sólo con un sí o un no) sabrá que este juego consiste en que uno de los participantes piensa en algún objeto o situación y el otro trata de adivinar esto por medio de preguntas, bueno así funciona un árbol de decisión.

Pongamos otro ejemplo, todos hemos asistido a la consulta de algún médico, cuando usted entra al despacho del profesional de la salud y toma asiento, dentro de las diferentes herramientas que este posee para intentar descubrir lo que usted padece, están las preguntas que el doctor le realiza con la finalidad de poder realizar un diagnóstico, le pregunta qué le duele, cuándo comenzó la molestia, si se ha realizado alguna actividad extraña, entre otras cosas, entonces, a grandes rasgos, podríamos decir que este proceso está basado en un árbol de decisión, o bien la técnica de los árboles de decisión se basa en situaciones como estas.

Grafiquemos esta situación, supongamos que a usted le duele mucho la cabeza y ha tenido tos durante los últimos tres días, por lo que decide asistir a un famoso médico de su comunidad, al entrar en su despacho y luego del saludo correspondiente se produce la siguiente conversación:

- ¿Cuál es su problema?
- Mire, me he sentido muy mal este último tiempo, creo que los años me están pasando la cuenta.
- ¿Qué edad tiene usted?
- Tengo 55 años.
- Pero es usted muy joven, ¿Le ha dolido la cabeza?
- Si, bastante.
- Mmm, ¿ha tenido usted tos?
- Si, la verdad es que bastante tos, doctor.
- ¿Hace cuánto tiempo que tiene esta tos?
- Déjeme pensar, creo que comenzó alrededor de 4 días atrás.
- De acuerdo, dígame ¿Usted fuma?
- No doctor.
- Ok, ¿ha estado expuesto al frio o humedad últimamente?
- Si, la semana pasada llovía mucho y no llevaba paraguas.
- Comprendo, pase por aquí para revisarlo...

Una vez terminada la charla anterior y la posterior revisión física, el médico está listo para dar su diagnóstico, para él, usted sufre de un resfrío común por lo que le receta un analgésico y le da descanso por tres días.

Pero volvamos a lo nuestro, ¿Qué tiene que ver todo esto con árboles de decisión? Bueno, la charla anterior si fuera modelada en un árbol de decisión luciría como la imagen de la Figura 2.1.

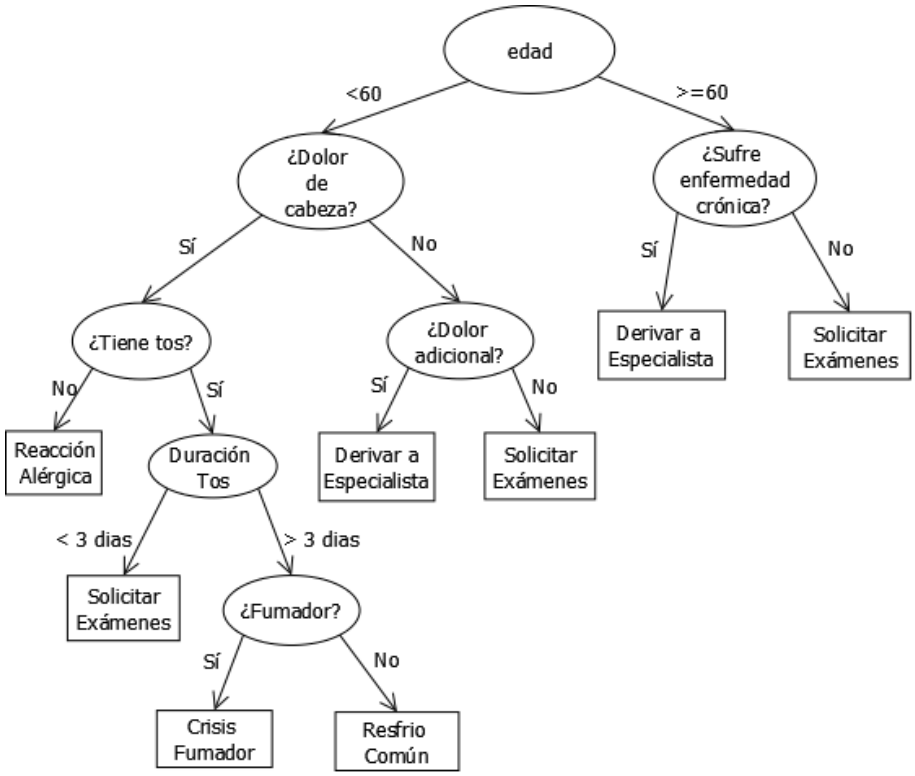

Figura 2.1: Árbol de decisión

Como podrá darse cuenta, el árbol de decisión está compuesto de nodos, donde cada nodo es asociado con una de las variables de entrada, las aristas vienen a ser todos los posibles valores que pueden tomar esas variables. Finalmente, una hoja representa la respuesta o decisión dado un conjunto de variables de entrada y el camino que realiza el algoritmo en el árbol.

Uno de los roles fundamentales del *Machine Learning* en esta técnica es el establecimiento de las reglas del árbol, es decir, a medida que el algoritmo es entrenado con el tiempo va evolucionando y podría crear reglas dinámicamente.

Usos para los árboles de decisión

Los árboles de decisión son usados en muchas áreas, por ejemplo las instituciones financieras los usan para clasificar a sus clientes, y así establecer cuáles de ellos son riesgosos y cuáles no lo son.

En el área de la salud, son usados para realizar diagnósticos de infecciones a la sangre o para predecir ataques al corazón en pacientes de alto riesgo.

Además, en el área del entretenimiento son ampliamente usados, por ejemplo el Microsoft Kinect usa este método para seguir el movimiento del cuerpo además de ser usado para el reconocimiento facial.

Ventajas y desventajas

Las ventajas de utilizar esta técnica es que su ejecución es de bajo coste computacional, los resultados obtenidos son fáciles de entender por las personas, además de trabajar de buena forma inclusive con ruido en los datos. La principal desventaja es que esta técnica es propensa a caer en el sobreajuste, es decir, es posible que llegue a ser manipulado por quien lo implemente (Harrington, P., 2012).

2.2.3.2. Redes neuronales artificiales

Las redes neuronales se modelan principalmente siguiendo la arquitectura de los cerebros animales, no necesariamente sólo de los humanos. (Bell, J., 2015).

A grandes rasgos, las redes neuronales se basan de simples entradas y salidas de información.

En términos biológicos, las neuronas son células que pueden transmitir y procesar señales químicas o eléctricas, las redes de neuronas están formadas por la interconexión que existe entre ellas.

Para nuestro objetivo basta con saber que una neurona está conformada por tres partes principales, las Dendritas (la entrada), el Cuerpo Celular (donde se procesa la información) y el Axon (la salida).

La unidad básica de una red neuronal artificial es el perceptrón, está estructura es la que imita la neurona antes descrita, y tal como la real, la neurona artificial recibe una entrada, ese valor lo pasa luego por una función y el resultado lo convierte en la salida del perceptrón que a su vez hará de entrada para otros perceptrones.

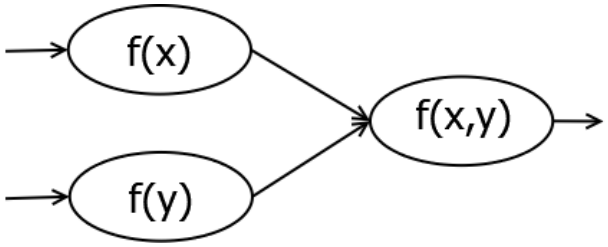

Figura 2.2: Red neuronal sencilla

El perceptron trabaja con números o vectores y puede tener la cantidad de entradas que la máquina donde se está ejecutando lo permita.

Usos para las redes neuronales artificiales

Estas tienen usos muy variados, los cuales van desde el campo médico, en el que son usadas para la predicción de enfermedades, hasta el campo financiero, donde si bien se usan bastante los árboles de decisión, también son las redes neuronales una de las técnicas más utilizadas a la hora de realizar predicción o de administrar transacciones de alta velocidad las cuales serían muy complejas de manejar por un ente humano.

Otra área importante para las redes neuronales es la robótica, donde son ampliamente usadas para el reconocimiento de patrones y para la gestión de la toma de decisiones de los agentes de inteligencia artificial.

2.2.3.3. Redes de Bayes

Para introducir al concepto de Redes de Bayes (también conocidas como Redes Bayesianas) sería util pensar en las siguientes situaciones: Si lanzo una moneda

al aire y sale cara, ¿cuál es la probabilidad de que la proxima vez también sea cara? o bien, si tiro un dado ¿cuál es la probabilidad de que salga el número seis?. Probabilidad es la medida de posibilidades de que un evento ocurra.

Sigamos con el ejemplo de la moneda, si lanzamos una moneda al aire, ciertamente sabemos que existen dos posibles resultados, o la moneda cae con su cara boca arriba, o bien, lo hace con el sello, lo que equivaldría al 50 % de probabilidad para cada uno de los casos, para describir esto, podríamos utilizar una notación como la siguiente:

$$P(sea\,cara) = 0{,}5\,(o\,1/2\,o\,50\,\%)$$
$$P(sea\,sello) = 0{,}5\,(o\,1/2\,o\,50\,\%)$$

Lo anterior representa la probabilidad para un evento, pero qué pasa si quisieramos saber, tal como nos preguntábamos al comienzo, cuál es la probabilidad de que al lanzar una moneda y salga cara, el siguiente lanzamiento arroje como resultado también cara, para eso es que existe la llamada *Prababilidad Condicional*, esto es, la probabilidad de que ocurra un evento A, sabiendo que también sucede otro evento B, la probabilidad condicional para este caso se escribiría $P(A|B)$ y se lee "la probabilidad de A dado B".

Revisemos el caso de la moneda, se tiene una serie de eventos:

Lanzamiento de la moneda 1. (A)
Lanzamiento de la moneda 2. (B)

La probabilidad condicional sería:

$$P(A|B)$$

Sabemos que la probabilidad para una moneda es $1/2$ por lo tanto la probabilidad de que ambas monedas muestren cara se consigue simplemente multiplicando $1/2 * 1/2$ lo que da como resultado $1/4$, es decir la probabilidad de que al lanzar dos monedas ambas sean cara, es de $1/4$.

Ahora que conocemos a grandes razgos que es la probabilidad condicional, podemos introducir el teorema de bayes, llamado así en honor al filosófo inglés Thomas Bayes quien planteó esta proposicion en el año 1763.

Para entender el teorema de Bayes es necesario antes conocer el concepto de probabilidad total el cual afirma lo siguiente:

Sea $A_1, A_2, \ldots, A_n$ una partición muestral sobre el espacio muestral y sea B un suceso cualquiera del que se conocen las probabilidades condicionales $P(B|A_i)$, entonces la probabilidad del suceso B viene dada por la expresión:

$$P(B) = \sum_{i=1}^{n} P(B|A_i)P(A_i)$$

Dicho lo anterior, el teorema de Bayes se utiliza para revisar probabilidades previamente calculadas cuando se posee nueva información, en otras palabras, el teorema de Bayes se apoya en el proceso inverso de la probabilidad total.

Teorema de la probabilidad total: a partir de las probabilidades del suceso A (probabilidad de que llueva o de que haga buen tiempo) deducimos la probabilidad del suceso B (que ocurra un accidente).

Teorema de Bayes: a partir de que ha ocurrido el suceso B (ha ocurrido un accidente) deducimos las probabilidades del suceso A (¿estaba lloviendo o hacía buen tiempo?).

El teorema de Bayes dicta lo siguiente:

Sea $A_1, A_2, \ldots, A_n$ un conjunto de sucesos mutuamente excluyentes y exhaustivos tales que la probabilidad de cada uno de ellos es distinta de cero. sea B un suceso cualquiera del que se conocen las probabilidades condicionales $P(B|A_i)$. Entonces la probabilidad $P(A_i|B)$ viene dada por la expresión:

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{P(B)}$$

Donde:

- $P(A_i)$ son las probabilidades a priori.
- $P(B|A_i)$ es la probabilidad de B dado A_i.
- $P(A_i|B)$ son las probabilidades a posteriori.
- $P(B)$ es la probabilidad total de B.

Para entender mejor esto, veamos el siguiente ejemplo:

El informe meteorológico ha anunciado tres posibilidades para el fin de semana:

a) Que llueva: probabilidad del 50 %

b) Que nieve: probabilidad del 30 %

c) Que haya niebla: probabilidad del 20 %

Según estos posibles estados meteorológicos, la probabilidad de que ocurra un accidente es la siguiente:

a) Si llueve: probabilidad de accidente del 20 %

b) Si nieva: probabilidad de accidente del 10 %

c) Si hay niebla: probabilidad de accidente del 5 %

Luego, ocurre un accidente pero no conocemos qué inclemencia del tiempo fue la que lo produjo, el teorema de Bayes nos permite calcular la probabilidad de que haya estado lloviendo, nevando o con niebla cuando se produjo el accidente.

Las probabilidades a priori, es decir las que conocemos antes de añadir la información adicional son las correpondientes a la probabilidad de que haya lluvia, nieve o niebla. Una vez que incorporamos la información del accidente, las probabilidades cambian y se denominan probabilidades a posteriori, que son las que calcularemos utilizando el teorema de Bayes.

Probabilidad de que durante el accidente estuviera lloviendo:

$$P(A_i|B) = \frac{0,50 * 0,20}{(0,50 * 0,20) + (0,30 * 0,10) + (0,20 * 0,05)} = 0,714$$

La probabilidad de que haya estado lloviendo es del $71\,\%$

Probabilidad de que durante el accidente estuviera nevando:

$$P(A_i|B) = \frac{0,30 * 0,10}{(0,50 * 0,20) + (0,30 * 0,10) + (0,20 * 0,05)} = 0,214$$

La probabilidad de que haya estado nevando es del $21\,\%$

Probabilidad de que durante el accidente hubiera niebla:

$$P(A_i|B) = \frac{0,20 * 0,05}{(0,50 * 0,20) + (0,30 * 0,10) + (0,20 * 0,05)} = 0,071$$

La probabilidad de que hubiera niebla es del $7\,\%$

Las redes de bayes siguen la lógica de la teoría de grafos, es decir se componen de nodos y vertices.

Para navegar dentro de la red se utiliza el teorema antes descrito, se generan reglas de decisión que implican probabilidades y en base a esas probabilidades y a los resultados obtenidos por el teorema es que se toman las decisiones sobre por cual arco de la red moverse, al final del proceso, el resutado de clasificación será dado por el valor del nodo final de la red. (Bell, J., 2015).

2.2.3.4. K-nearest neighbor

La técnica de clasificación conocida como *k Nearest Neighbor* es sin duda una de las más simples y efectivas, dado que esta es la técnica escogida para llevar a cabo este estudio es que su análisis detallado es presentado en la sección 2.2.4.

2.2.4. Especificación de la técnica K Nearest Neighbor

Esta técnica pertenece al grupo de los algoritmos de aprendizaje supervisado, ya que necesita de datos de ejemplo para poder trabajar, es decir hace falta una etapa de entrenamiento para que este funcione adecuadamente.

Antes de entrar de lleno en la definición de la estrategia utilizada por el algoritmo de los k vecinos más cercanos (llamado así por su traducción al español) es necesario definir el concepto de distancia euclidiana.

La distancia euclidiana es la distancia ordinaria entre dos puntos p y q en el espacio euclídeo, la cual se define como sigue:

$$d(p,q) = d(q,p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \ldots + (q_n - p_n)^2}$$

De forma general:

$$\sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

Logicamente la formula de la distancia varia en función a la cantidad de dimensiones en donde se aplique, por ejemplo:

Si x e y son dos puntos en una línea, entonces para una dimension, la formula es como sigue:

$$\sqrt{(x-y)^2} = |x-y|$$

Si $p = (p_1, p_2)$ y $p = (q_1, q_2)$ son puntos en el plano, entonces la formula para la distancia euclideana toma la siguiente forma:

$$d(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}$$

Para tres dimensiones:

$$d(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + (q_3 - p_3)^2}$$

Por lo tanto para n dimensiones, la distancia euclideana se calcula:

$$d(p,q) = d(q,p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \ldots + (q_n - p_n)^2}$$

Vale la pena mencionar que esta técnica de clasificación puede trabajar con cualquier fórmula de distancia tal como *distancia de Manhattan, distancia de Chebychev, distancia del coseno* entre otras, para los ejemplos presentados en este trabajo y la implementación de los algoritmos se ha escogido la distancia euclideana pero emplear otra formula de distancia emplicaría realizar cambios mínimos al trabajo.

Ahora que conocemos el concepto de distancia estamos capacitados para introducir la técnica de los k vecinos más cercanos, para ilustrar esto de mejor forma emplearemos un ejemplo.

Supongamos que tenemos la misión de clasificar una serie de películas, en si son películas de acción o peliculas de romance, para esto vamos a suponer que el género de una película se define ya sea por la cantidad de escenas de acción (como peleas, patadas, o disparos) y la cantidad de escenas de romance (besos, abrazos, etc.) que poseen.

Si bien por razones de simplicidad en este ejemplo usaremos sólo dos variables, la cantidad de escenas de acción y la cantidad de escenas de romance en la película, es importante dejar en claro que es posible utilizar la cantidad de variables que se estimen convenientes, siempre y cuando la formula de distancia euclideana aplicada sea la correcta.

Para poder utilizar la técnica de los k vecinos más cercanos es necesario poseer una entrada de datos que sirvan de entrenamiento, es decir, datos que le den al algoritmo la base sobre la cual decidir para futuras clasificaciones, para esto es que se ha hecho una recopilación y se han obtenido las siguientes estadísticas de películas cuyo género ya se conocía:

#	Película	Escenas de acción	Escenas de romance	Género
A	300	50	5	acción
B	duro de matar	35	10	acción
C	titanic	5	15	romance
D	calabozos rojos	10	40	romance
E	amor y desastre	25	50	romance

Cuadro 2.1: Datos de entrenamiento

Ahora bien, estos datos deben formar parte de la base de datos de entrenamiento del algoritmo, luego estos son representados en el espacio de n dimensiones, donde n corresponde a la cantidad de variables estudiadas, en este caso dos, por lo que corresponde ubicar cada película en un punto del plano.

En el gráfico de la Figura 2.3 se pueden apreciar las posiciones que mantiene cada una de las tuplas de la tabla de datos de entrenamiento al ser volcadas en el plano, es facilmente identificable la tendencia de los puntos, en la parte superior del plano podemos encontrar los puntos correspondientes a las películas C, D, E correspondientes a *titanic*, *calabozos rojos* y *amor y desastre* respectivamente

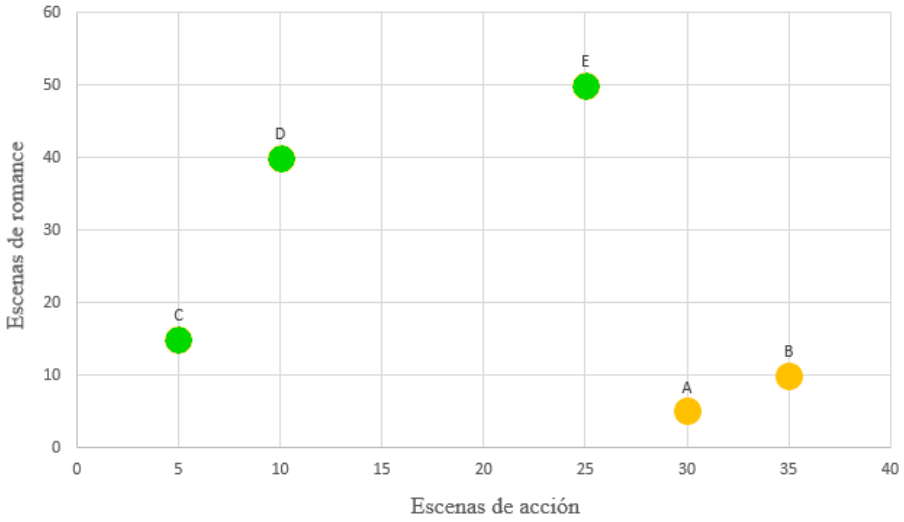

Figura 2.3: Datos de entrenamiento en el plano

mientras que en la parte inferior están ubicados los puntos que representan a las tuplas de *300* y *duro de matar* (A y B).

Una vez que los datos de entrenamiento ya han sido cargados, esta técnica requiere de la tupla o las tuplas que quieren ser clasificadas, para este ejemplo vamos a suponer que sólo queremos clasificar la película que se presenta en la siguiente tabla:

#	Película	Escenas de acción	Escenas de romance
F	romance peligroso	25	30

Cuadro 2.2: Datos a clasificar

Si miramos con atención, el Cuadro 2.2 nos muestra la tupla que deseamos clasificar, pero esta tupla posee una columna menos en comparación con el cuadro de los datos de entrenamiento, ya que éste no posee variable objetivo, pues es esta la que queremos conocer con la aplicación del algoritmo *k nearest neighbors*.

La tupla a clasificar es volcada al plano en la ubicación que le corresponde dentro de él, la Figura 2.4 muestra esto gráficamente, cabe recordar que estas tuplas están siendo representadas como puntos en el plano pues para fines prácticos son sólo dos los atributos estudiados en este ejemplo.

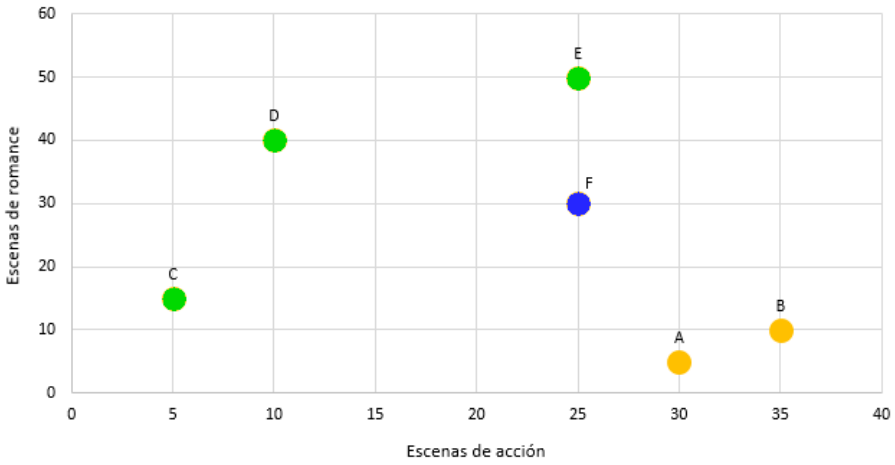

Figura 2.4: Datos de entrenamiento & datos a clasificar en el plano

Para continuar con el proceso, debemos calcular las distancias desde cada uno de los puntos del entrenamiento (representados en la Figura 2.5 de color verde y amarillo) hasta el punto que representa a la tupla a clasificar, en el ejemplo, deberíamos calcular las siguientes distancias:

$$d(C, F)$$
$$d(D, F)$$
$$d(E, F)$$
$$d(A, F)$$
$$d(B, F)$$

Como estamos trabajando en dos dimensiones es que debemos calcular la distancia euclideana entre dos puntos usando la formula que vimos anteriormente:

$$d(p, q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}$$

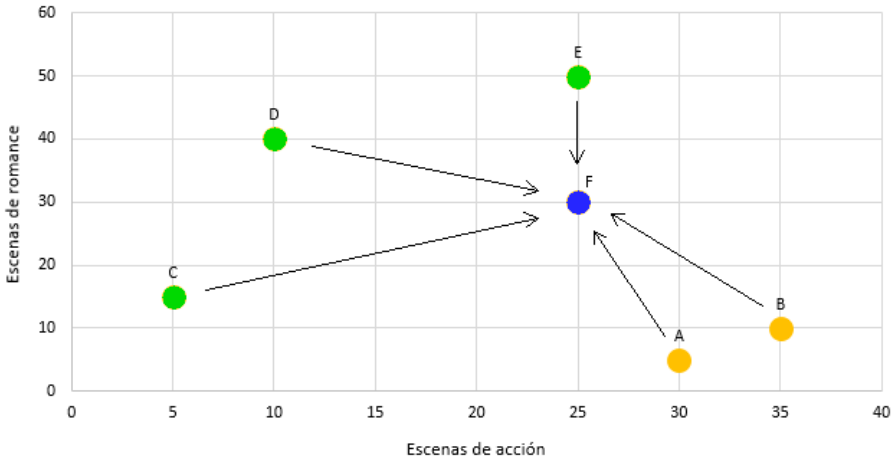

Figura 2.5: Distancias al punto a clasificar

Reemplazando los valores en la formula para cada punto:

$$d(C,F) = \sqrt{(5-25)^2 + (15-30)^2} = 25$$

$$d(D,F) = \sqrt{(10-25)^2 + (40-30)^2} = 18$$

$$d(E,F) = \sqrt{(25-25)^2 + (50-30)^2} = 20$$

$$d(A,F) = \sqrt{(30-25)^2 + (5-30)^2} = 26$$

$$d(B,F) = \sqrt{(35-25)^2 + (10-30)^2} = 22$$

Una vez se han calculado todas las distancias, estas deben ser ordenadas de menor a mayor y agrupadas en un conjunto que llamaremos S, para este ejemplo, S estaría formado como muestra el Cuadro 2.3.

#	Puntos	Distancia
1	D-F	18
2	E-F	20
3	B-F	22
4	C-F	25
5	A-F	26

Cuadro 2.3: Conjunto S de distancias ordenadas ascendentemente

Ahora hace falta ocuparse del factor k, este representa la cantidad de distancias que evaluaremos para tomar la desición al momento de clasificar, este factor k debe ser mayor a cero y menor o igual a la cantidad de tuplas ingresadas en la etapa de entrenamiento, si $k = 1$ entonces esta técnica es conocida como *The Nearest Neighbor* puesto que se toma la desición en base al primer elemento del conjunto S.

Una vez determinado el valor de k se deben tomar los primeros k elementos dentro del conjunto S y escoger como variable objetivo de clasificación la variable objetivo de las tuplas que sea mayoría dentro del sub conjunto de los primeros k elementos en S, en el cuadro 2.4 se muestra el conjunto S con las variables objetivo para cada tupla.

#	Puntos	Distancia	Género
1	D-F	18	romance
2	E-F	20	romance
3	B-F	22	acción
4	C-F	25	romance
5	A-F	26	acción

Cuadro 2.4: Conjunto S con variables objetivo para cada tupla

Para graficar esto, se han tomado distintos valores de k y se han analizado los resultados de estas distintas combinaciones, supongamos que escogemos como nuestro k el valor 1, por lo tanto si $k = 1$ entonces para realizar la clasificación bastaría con tomar la etiqueta correspondiente a la variable objetivo de la primera tupla dentro de S.

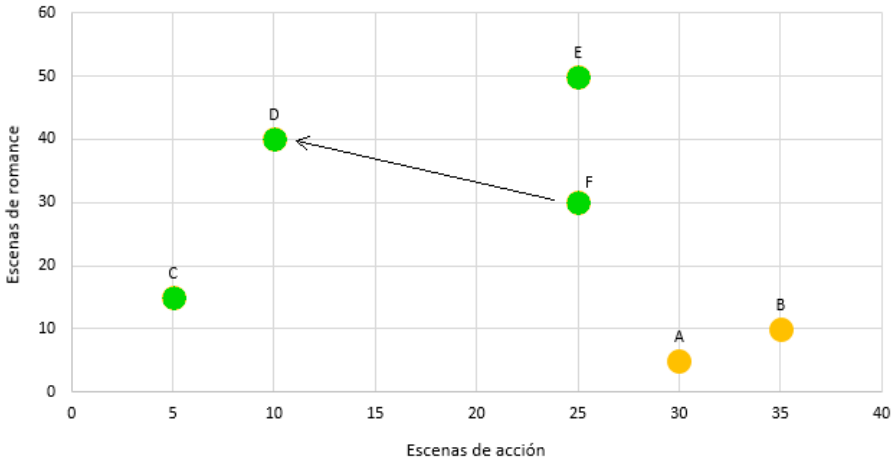

Figura 2.6: Gráfico para k=1

Por lo tanto el resultado de la clasificación sería:

#	Película	Escenas de acción	Escenas de romance	Genero
F	romance peligroso	25	30	romance

Cuadro 2.5: Resultado de la clasificación para k=1

Ahora bien cambiemos el valor de k, si ahora $k = 3$ según lo que se ha dicho en los parráfos anteriores, deberíamos tomar los primeros 3 elementos dentro de nuestro conjunto S y escoger como variable objetivo de la clasificación la variable que más se repita en este subconjunto de S.

Esta vez como son más de uno los elementos dentro de S' es necesario analizar las variables objetivo de cada tupla, en el ejemplo podemos observar que hay dos tuplas que son películas románticas mientras que existe una sóla que pertenece al género de acción por lo tanto, el resultado de la clasificación sigue siendo *romance*.

#	Puntos	Distancia	Género
1	D-F	18	romance
2	E-F	20	romance
3	B-F	22	acción

Cuadro 2.6: S' para k=3

Como se podría suponer, en la estratégia usada por *k Nearest Neighbors* la selección del valor para la variable *k* es muy relevante, si el valor de k es demasiado alto, el resultado de las clasificaciones será generalizado en exceso y si el valor es muy pequeño la predicción sería muy variada una de otra, es por eso que es necesario que este sea escogido de manera tal que produzca la clasificación más exacta. (Myatt, G., Johnson, W., 2014).

Para conseguir lo anterior es recomendable escoger distintos valores de k, calcular el error en la predicción y compararlos, para esto se puede usar la siguiente formula correspondiente a la suma de cuadrados de error (SCE):

$$SCE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Si los conjuntos comparados tienen distinto tamaño, es posible usar una variación de esta formula para obtener el error cuadrático medio:

$$ECM = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Una vez calculado el error para cada valor de k, el conjunto cuya sumatoria de error sea la menor, será el que tenga mayor grado de acierto en la predicción.

Capítulo 3

Trabajos relacionados

En su trabajo investigativo, Timarán (2009), aplicó técnicas de minería de datos a la información almacenada en las bases de datos de la Universidad de Nariño, Colombia con el fin de detectar patrones de bajo rendimiento en los estudiantes de dicha casa de estudio, para poder determinar que alumnos se verían en esta situación en el futuro y así tomar las acciones correctivas que hicieran falta.

Este proceso se apoyó con el software TariyKDD, una herramienta de minería de datos de distribución libre, desarrollada en los laboratorios de KDD del Departamento de Ingeniería de Sistemas de la Facultad de Ingeniería de la Universidad de Nariño.

Timarán en su artículo describe el proceso que llevo a cabo para lograr la detección de patrones de bajo rendimiento y deserción en los estudiantes, el cual cuenta con las siguientes etapas: etapa de selección de los datos, etapa de pre procesamiento de los datos, etapa de transformación de los datos, etapa de minería de Datos y finalmente una etapa de interpretación y evaluación de los resultados.

Dentro de las conclusiones de Timarán están el hecho de que las etapas de pre procesamiento y transformación de los datos fueron las que requirieron más tiempo debido a la mala calidad de los datos presentes en las bases de datos de la universidad.

Si bien Timarán menciona las reglas de asociación utilizadas en el proceso de minería de datos, no da a conocer los resultados brutos de su proceso de detección de patrones.

En otro trabajo, Spositto, Etcheverry, Ryckeboer, Bossero, (2010) realizaron un ejercicio similar al mencionado anteriormente para el cual utilizó herramientas como MS SQL Server, el software SPSS para realizar el pre procesamiento de los datos y el software Weka (Waikato Environment for Knowledge Analysis).

Esta vez la finalidad del proyecto fue la evaluación del rendimiento académico y la identificación de los patrones determinantes en la deserción estudiantil de los estudiantes de la Universidad Nacional de La Matanza, Argentina.

Para llevar a cabo el proceso de descubrimiento de conocimiento se definieron cinco etapas: recopilación e integración de los datos, limpieza, selección y transformación de los datos, minería de datos, evaluación e interpretación y difusión y uso, como el lector habrá notado estas etapas son muy parecidas a las usadas por Timarán en su propio trabajo.

Para lograr los objetivos de la investigación se utilizaron datos de los alumnos desde el año 2003 hasta el año 2008 presentes en las bases de datos de la Universidad Nacional de La Matanza.

Durante la fase de minería de datos se utilizó la clasificación como tipo de minería de datos, árboles de decisión como tipo de modelo y los algoritmos J48 y FT, es importante mencionar también que para entrenar los modelos se utilizó el 30 % de los datos disponibles, los cuales fueron escogidos de forma aleatoria.

El algoritmo J48 es una implementación en *Java* y de licencia open source del algoritmo C4.5, el que a su vez es un algoritmo para generar árboles de decisión desarrollado por Ross Quinlan.

Por otra parte, el algoritmo FT es una sigla para *"Functional Trees"*, el cual es un clasificador que podría utilizar funciones de regresión logística para la generación de los árboles de decisión.

En Spositto et al. (2010) se utilizaron 15 atributos para el estudio, dentro de los que destacan el sexo del individuo, su edad, se estado civil, carrera, su promedio, la cantidad de materias reprobadas, la educación de sus padres entre otras.

Ya como resultado Spositto presenta la siguiente tabla de porcentaje de acierto en su clasificación tanto para el algoritmo J48 como el FT.

	FT	J48
Rendimiento académico	78 %	72 %
Deserción estudiantil	77 %	72 %

Cuadro 3.1: Resultados Clasificación Sposito et al.

Como conclusión Sositto et al. reconocen que si bien no lograron encontrar un clasificador de rendimiento académico y deserción estudiantil lo suficientemente exitoso se logró adquirir experiencia en el uso de los programas SPSS y Weka lo que sin duda permitirá el avance en esta línea de investigación.

Dapozo, Porcel, Lopez y Greiner (2008) Aplicaron una red neuronal de tipo perceptrón multicapa además de algoritmos se aprendizaje supervisado con el fin de obtener información que ayude a conocer los factores que influyen en el rendimiento académico de los estudiantes de la Facultad de Ciencias Exactas de la Universidad Nacional del Nordeste, Argentina.

Si bien en este artículo no se presentan detalles sobre los resultados obtenidos por la aplicación de los algoritmos y técnicas mencionadas en el párrafo anterior, dicho documento sirve de evidencia del esfuerzo puesto por variadas instituciones educacionales en el intento de predecir el rendimiento de sus estudiantes.

Vera, Romero y Ventura (2012) en su trabajo titulado “Predicción del fracaso escolar mediante técnicas de minería de datos” utilizan estás técnicas para detectar los factores que más influyen para que los estudiantes de enseñanza media fracasen, además reconocen la dificultad del problema y proponen utilizar diferentes técnicas para así definir la que posea el mejor rendimiento.

El método utilizado, tal como en otros trabajos, nuevamente utilizó cuatro etapas: recopilación de los datos, pre procesamiento, minería de datos y finalmente la interpretación de los resultados.

Los alumnos objetos del estudio fueron aquellos ingresantes al programa II de la Unidad académica Preparatoria de la Universidad Autónoma de Zacatecas (UAPUAZ) de México, para la recopilación de los datos se aplicaron encuestas a los estudiantes, además se tuvo acceso a bancos de datos del Departamento de Servicios Escolares del Programa II y del Centro Nacional de Evaluación (CENEVAL).

Nuevamente los autores reconocen la importancia de las etapas de recolección y pre procesamiento de los datos, ya que la calidad y fiabilidad de los datos afecta considerablemente en los resultados.

Este trabajo posee los porcentajes de acierto más altos de entre todos los artículos revisados, alcanzando porcentajes de acierto de 94 % para el algoritmo JRip, 93 % usando el algoritmo SimpleCart, el algoritmo con peor rendimiento en este trabajo fue Prism con un 54 % de acierto.

Al ser uno de los trabajos con mayor indice de acierto, sería interesante conocer los parámetros que fueron medidos en su estudio, cómo se mencionó anteriormente, este trabajo contó con tres fuentes diferentes desde las cuales se obtuvo la información, una encuesta, los datos de CENEVAL y los datos existentes en el Departamento Escolar.

Dentro de las variables medidas, destaca el nivel de motivación del estudiante, si sufre alguna discapacidad física, si posee becas de estudio, si toma apuntes, su estado civil, su interés en la institución en la que participa entre otras, a continuación en el Cuadro 3.2 se presenta una breve tabla resumen de algunas de las variables estudiadas y la fuente donde estas fueron adquiridas.

Fuente	Variables
Encuesta	Nivel de motivación, número de amigos, tiempo de estudio, forma de estudio, lugar de estudio, número de hermanos, asistencia a clases, toma apuntes, bebe alcohol, fuma, número de alumnos en el curso, entre otros.
CENEVAL	edad, sexo, modelo de secundaria, nota en matemáticas, nota en español, nota en química, trabajo de los padres, número de computadores en la familia, ciudad de origen, entre otros.
Departamento Escolar	nota en física, nota en ciencias sociales, nota en talleres de lectura, nota en computación, estado académico, entre otros.

Cuadro 3.2: Variables estudiadas por Vera, Romero y Ventura

Otros trabajos interesantes de mencionar son los de Zambrano, Rojas y Carvajal (2011) “Análisis del rendimiento académico estudiantil usando Data Warehouse y Redes Neuronales” y el trabajo titulado “Redes Neuronales para predecir la aptitud del alumno y sugerir acciones” de Salguerio et al. (2006).

Capítulo 4

Estudio empírico

El capítulo que es presentado a continuación tiene como finalidad documentar el trabajo realizado y las conclusiones de éste, para lograr ésto primero se define el método que se siguió y luego se procede a la realización de cada una de las etapas de dicha estrategia.

4.1. Método utilizado

El método utilizado para llevar a cabo este proyecto se basa en la estrategia presentada en el libro *Machine Learning in Action* (Peter Harrington, 2012), el presenta allí, en la sección *"Steps in developing a machine learning application"*. un procedimiento que consta de 6 pasos para la implementación de una aplicación que utilize técnicas de *Machine Learning*, sin embargo, con el fin de adaptar este proceso al actual proyecto es que añadiremos un paso adicional a esta estrategia, la implementación del algoritmo:

1. **Colección de los datos de entrada.** El primer paso para implementar una aplicación que trabaje utilizando técnicas de *Machine Learning* es coleccionar los datos que serán analizados, hay muchas formas en las que se pueden obtener grandes cantidades de datos, este paso se refiere a esas opciones y al esfuerzo aplicado para obtener dichos datos.

2. **Preparación de los datos de entrada.** Una vez que ya se poseen los datos, es necesario asegurarse que estén en el formato correcto para ser procesados por el algoritmo de *Machine Learning* seleccionado, por ejemplo, existen algunos algoritmos que necesitan datos en formato de números enteros, otros algoritmos aceptan etiquetas para cada columna de la tupla estudiada, etc.

Por lo que este paso involucra, si fuera necesario, formatear los datos para adaptarlos a la necesidad de cada algoritmo.

3. **Análisis de los datos de entrada.** Este paso tiene como finalidad asegurar que los datos obtenidos en el paso 1 y preparados en el paso 2 sean los correctos, por ejemplo que no posean caracteres extraños, espacios en blanco (si es que la técnica seleccionada no los permite), entre otras validaciones útiles.

4. **Implementación del algoritmo:** Con los datos ya en nuestro poder, hace falta implementar el algoritmo en el lenguaje de programación que hayamos escogido, asegurándonos que la entrada de nuestro algoritmo sea acorde al formato de los datos que poseemos, en este paso además, debe quedar la mayor cantidad de documentación relacionada al desarrollo que sea posible, tales como diagramas de flujo, pseudocódigo, etc.

5. **Entrenamiento del algoritmo:** Con la aplicación ya creada hace falta entrenar el algoritmo de *Machine Learning*, esta etapa es primordial para el algoritmo pues es aquí donde se desarrolla toda la inteligencia de aprendizaje de éste, los datos de entrenamiento y la forma en que son pasados como entrada al algoritmo dependerán de la técnica utilizada además vale la pena recordar que si se trabaja con algoritmos no supervisados, no hace falta aplicar un paso de entrenamiento.

6. **Testeo del algoritmo:**Aquí es donde la información aprendida por el algorimo es testeada, es decir, se mide el nivel de acierto que tiene nuestro algoritmo, utilzando los datos de entrenamiento podremos establecer el grado de eficacia de nuestra implementación de la técnica seleccionada, si los resultados no son los esperados es probable que haya que volver a etapas previas para intentar identificar el error, que tal vez sea en los datos de entrada o en el algorimo en sí, y tratar de solucionarlo. Una vez realizado los cambios hará falta volver a pasar por todos los pasos anteriores una vez más.

7. **Uso del algoritmo:** Una vez se han consumado todos los pasos, no queda más que usar el algoritmo, esta estapa implica tener que volver a ejecutar los pasos 1, 2, 3 y 5.

Tenga en cuenta que el proceso anterior puede ser desarrollado en prácticamente cualquier lenguaje de programación, para el caso particular de este proyecto se ha escogido *Java* por su versatilidad, por ser orientado a objetos y por la experiencia técnica que posee el desarrollador con este lenguaje.

4.2. Colección de los datos de entrada

Para poder comenzar la implementación de la herramienta de predicción, hace falta primeramente definir qué datos son los que se van a coleccionar y cómo se va a hacer, en esta sección, primero se dará una extendida justificación de los datos recolectados para luego presentar el método escogido para poder coleccionarlos.

4.2.1. Factores que influyen en el rendimiento académico

Desde el punto de vista científico, el rendimiento académico es un proceso mediante el cual se exterioriza información, actitudes, destrezas y habilidades adquiridas a lo largo del proceso enseñanza-aprendizaje.

Si bien existen varias investigaciones relacionadas con el proceso de aprendizaje, muchas de estas se han enfocado en estudiar los componentes que influyen en estos procesos pero de manera aislada unos de otros y por lo tanto es necesario desarrollar modelos integradores (Pintrich, 1994, citado en Valle, Gonzales, Nuñez y Gonzales, 1998).

Es completamente razonable y necesario buscar las causas de los resultados académicos más allá del estudiante mismo. El fracaso o éxito de los estudiantes es un conjunto de factores interrelacionados, tanto internos como externos al estudiante. (Fullana 1996, citado en Plank, 2014).

Es más, existe un consenso en que los factores asociados al desempeño académico puede tener su origen en dos grandes ámbitos: en los determinantes personales y los determinantes sociales.

Según varios autores, (Weiner, 1990; McCombs, 1986, 1989; McCombs y Marzano, 1990, citado en Valle et al. 1998) el autoconcepto, definido como la percepción que el ser tiene de si mismo, desempeña un papel central en la motivación y en el aprendizaje escolar.

Otra variable es el enfoque de aprendizaje, (Marton y Säljö, 1976; Biggs, 1987; Entwistle, 1988, citados en Valle et al. 1998), los enfoques de aprendizaje designan los procesos de aprendizaje que surgen de las percepciones de los estudiantes de las tareas académicas, influenciadas por sus características personales. Según Biggs, cuando un estudiante se enfrenta a una situación de aprendizaje, le surgen dos importantes cuestiones; una relacionada con los motivos y metas que desea

conseguir (¿qué quiero conseguir con esto?) y la otra vinculada con las estrategias y recursos cognitivos que debe poner en marcha para lograr dichas metas (¿cómo hago para conseguirlo?). De esta forma el enfoque de aprendizaje se compone de motivo y estrategia, combinados ambos mediante un proceso meta cognitivo que Biggs denomina "meta-aprendizaje".

En relación al párrafo anterior, se ha logrado identificar tres enfoques de aprendizaje adoptados por los estudiantes al verse enfrentados a una situación de aprendizaje y son los siguientes:

Superficial: aquellos estudiantes que quieran cumplir con los requisitos mínimos de la tarea con el mínimo esfuerzo, activaran un enfoque superficial dirigido a aprender mecánica y repetitivamente la información para reproducirla en el momento oportuno en el que se le solicite.

Profundo: este enfoque es aquel que siguen los estudiantes que tienen la intención de aprender de manera significativa la materia en cuestión, y para lograrlo seguirán estrategias lo suficientemente exigentes para alcanzar dicha meta.

Estratégico: también conocido como enfoque de logro este se basa en la planificación es decir, tiene como objetivo alcanzar el mayor rendimiento posible, (la mejor calificación por ejemplo) siguiente una planificación minuciosa de las actividades de modo que se logre la meta con la energía y el tiempo disponible para ello.

Además del enfoque de aprendizaje adoptado, es posible reconocer a lo menos tres tipos de estudiantes:

Los orientados al dominio. Sujetos que tienen éxito escolar, se consideran capaces, presentan alta motivación de logro y muestran confianza en sí mismos.

Los que aceptan el fracaso. Sujetos derrotistas que presentan una imagen propia deteriorada y manifiestan un sentimiento de desesperanza aprendido, es decir que han aprendido que el control sobre el ambiente es sumamente difícil o imposible, y por lo tanto renuncian al esfuerzo.

Los que evitan el fracaso. Aquellos estudiantes que carecen de un firme sentido de aptitud y autoestima y ponen poco esfuerzo en su desempeño; para "proteger" su imagen ante un posible fracaso, recurren a estrategias como la participación mínima en el salón de clases, retraso en la realización de una tarea, trampas en los exámenes, etc. (Covington, 1984, p. 1, citado en Edel, 2003).

Numerosas investigaciones ponen de manifiesto la importancia de los factores socioculturales sobre el rendimiento académico (Caravana, 1997; Mori, 1992, citado en Iglesias y Vera, 2010).

En relación al status socioeconómico de los padres, (Velez, Schiefelbein y Valenzuela, 1993 citado en Iglesias y Vera, 2010) llegaron a la conclusión de que este factor influye de manera positiva en 49 de 80 casos, además, los alumnos que provienen de bajos estratos sociales tienen un rendimiento menor que aquellos que provienen de estratos sociales más altos, esto más que nada por el hecho de no poder acceder a una educación de mejor calidad durante los años previos al ingreso a la universidad. (Cárdenas, 1988, citado en Gonzáles y Oyarzo, 2000).

Hay autores que relacionan el rendimiento académico con la inteligencia emocional, es decir a mayor inteligencia emocional, mejor será la medida de éxito académico. (Abanto, Higueras y Cueto, 2000; Martín y Beck ,1997 citado en Iglesias y Vera, 2010).

En relación a la variable de inteligencia emocional y su correlación con el rendimiento académico, (Briceño, 2005, citado en Iglesias y Vera, 2010) en su trabajo de investigación concluye que existe suficiente evidencia para establecer que la inteligencia emocional influye considerablemente en el rendimiento académico de los alumnos.

Como dato interesante se podrían mencionar los resultados de una investigación de la universidad de Stanford, la cual determinó que el cociente intelectual interviene solo en un 20 % de los factores que determinan el éxito académico y el 80 % restante está vinculado a otros factores que incluyen lo que se denomina inteligencia emocional. (Chavanne, 1997, citado en Iglesias y Vera, 2010).

Ademas otros autores (Malabrigo, 2001, Villalobos, 2005 y Ushinahua, 2007, citados en Iglesias y Vera, 2010) sostienen que la variable autoestima influye considerablemente en el rendimiento académico del estudiante.

Otro autores concluyen también que altos niveles de ansiedad, presión o stress tienden a reducir el rendimiento de los alumnos.

La variable interés se correlaciona positivamente con el rendimiento académico. Los alumnos que postulan en los primeros lugares de determinada carrera corresponden a los mejores rendimientos (Royo 1979, citado en Gonzáles y Oyarzo, 2000).

En un estudio realizado sobre alumnos ingresados a la carrera de pedagogía de la Universidad de Chile, se concluye que existe relación directa entre el lugar de postulación a la carrera y el rendimiento de los alumnos. (Nazar, 1980, citado en Gonzáles y Oyarzo, 2000).

Un estudio realizado en la Universidad del Salvador (Beguet, Kohan, Castro Solano y Renault, 2001, citado en Corengia, Pita, Mesurado, y Centeno, 2011), analiza y concluye que las variables socio-demográficas, reputación escolar, auto percepción de las razones para comenzar y seguir estudiando, expectativas laborales, tienen gran implicancia en el rendimiento académico del estudiante.

Autores como Betts y Morell, concluyeron que variables como sus características personales, sus escuelas de origen, así como sus resultados del promedio de calificaciones de secundaria y la pruebas de selección universitaria rendidas influyen significativamente en el rendimiento académico universitario. (Betts y Morell 1999, citado en Corengia at al., 2011).

4.2.2. Recolección de los datos

Esta subsección se divide en tres partes, primero se presenta el diseño del instrumento para recabar datos, luego la implementación y justificación de este para finalmente presentar la estratégia usada para poder llegar a la población objetivo de dicha recolección.

4.2.2.1. Diseño del instrumento

Para recolectar los datos necesarios para el análisis se definió una encuesta, la cuál contó de 33 preguntas destinadas a conocer información sobre los factores que influyen en el rendimiento académico de los estudiantes que ingresaron los años 2013, 2014 y 2015 a la carrera de Ingeniería Civil Informática, de la Universidad del Bío Bío en Chillán.

Cada una de las preguntas de la encuesta fue diseñada basándose en los resultados de la revisión de los factores que influyen en el rendimiento académico.

De la sección anterior se concluyeron los siguientes conceptos como factores que influyen en el rendimiento de los estudiantes:

- Autoconcepto.
- Motivación.
- Enfoque de aprendizaje.
- Factores socio-culturales.
- Educación de los padres.
- Inteligencia Emocional.
- Autoestima.
- Factores económicos.
- Factores emocionales.
- Interés en la carrera.
- Escuelas de origen.
- Determinantes personales.
- Puntaje de acceso a la universidad.
- Motivos para estudiar.
- Expectativas laborales.

De estos 15 factores, se escogieron 13 para ser estudiados por medio de la encuesta. A continuación se presentan las 33 preguntas que formaron parte de dicha encuesta, estas preguntas fueron agrupadas teniendo en cuenta el factor que se aborda en ellas.

- **Autoconcepto.**
  - *¿Qué tan responsable se considera?*

    Esta pregunta podía ser respondida seleccionando una de las siguientes opciones:

    1. muy irresponsable
    2. irresponsable
    3. indiferente
    4. responsable
    5. muy responsable

  - *¿Considera que tiene las aptitudes suficientes para estudiar Ingeniería Civil Iinformática?*

    Esta pregunta podía ser respondida seleccionando una de las siguientes opciones:

    1. muy en desacuerdo
    2. en desacuerdo
    3. no sabe
    4. de acuerdo
    5. muy de acuerdo

  - *¿Se considera una persona inteligente?*

    Esta pregunta podía ser respondida seleccionando una de las siguientes opciones:

    1. muy en desacuerdo
    2. en desacuerdo
    3. no sabe
    4. de acuerdo
    5. muy de acuerdo

- **Motivación.**
  - *¿Le gusta programar?*

    Esta pregunta podía ser respondida seleccionando una de las siguientes opciones:

    1. muy en desacuerdo
    2. en desacuerdo
    3. no sabe
    4. de acuerdo
    5. muy de acuerdo

  - *¿Cuántas horas al dia dedica a estudiar para una asignatura?*

    Esta pregunta podía ser respondida ingresando un valor entre 0 y 24
  - *¿Cuántas horas al dia dedica a escribir código?*

    Esta pregunta podía ser respondida ingresando un valor entre 0 y 24
- **Enfoque de aprendizaje.**
  - *Cuando está cerca la fecha de un examen, ¿cón cuántos dias de anticipación estudia?*

    Esta pregunta podía ser respondida ingresando un valor entre 0 y 200

  - *Respecto al estudio, ¿qué considera más importante?*

    Esta pregunta podía ser respondida seleccionando una de las siguientes opciones:

    1. aprender de manera significativa
    2. obtener sólo una buena calificación

- **Factores socio-culturales.**
  - *¿Cuántos integrantes hay en su grupo familiar?*
  - *¿Cuántos integrantes de su grupo familiar trabajan?*

- *¿Cuántos integrantes de su grupo familiar estudian?*
- *¿Cuántos integrantes de su grupo familiar son pensionados?*

  Este grupo de preguntas podía responderse ingresando un número entre 0 y 20

- **Educación de los padres.**
  - *¿Qué nivel de educación posee su madre?*
  - *¿Qué nivel de educación posee su padre?*

    Este grupo de preguntas podía responderse escogiendo una de las siguientes alternativas:

    1. desconocido
    2. básico
    3. medio
    4. técnico
    5. superior

- **Inteligencia Emocional.**
  - *¿Cómo reacciona frente a un problema en su entorno social?*

    Esta pregunta podía ser respondida escogiendo una de las siguientes alternativas:

    1. reacciona con violencia física o verbal
    2. intenta conversar sobre la situación
    3. con indiferencia

  - *¿De qué forma influye el stress en su vida académica?*

    Esta pregunta podía ser respondida escogiendo una de las siguientes alternativas:

    1. muy negativamente
    2. negativamente
    3. no influye

4. positivamente
5. muy positivamente

- *¿Qué nivel de empatía presenta hacia sus compañeros?*

  Esta pregunta podía ser respondida escogiendo una de las siguientes alternativas:

  1. sin empatía
  2. muy poco empático
  3. indiferente
  4. poco empático
  5. muy empático

- **Factores económicos.**

  - *¿Cuántas horas a la semana trabaja?*

    Esta pregunta podía ser respondida con un número entre 0 y 50

  - *¿Cuántas becas de estudio posee?*

    Esta pregunta podía ser respondida con un número entre 0 y 10

  - *¿Cuál es el ingreso aproximado mensual por integrante de su grupo familiar?*

    Esta pregunta podía ser respondida con un número entre 0 y 1000000

- **Factores emocionales.**

  - *¿Posee actualmente una relación amorosa estable?*

    Esta pregunta podía ser respondida con un si o un no.

- **Interés en la carrera.**
  - *Al postular a la carrera ¿en qué posición de preferencia la ubicó?*

    Esta pregunta podía ser respondida con un número entre 0 y 10

  - *¿Si pudiera se cambiaría de carrera?*

    Esta pregunta podía ser respondida con un si o un no.

  - *¿Si pudiera seguiría estudios de postgrado dentro del área de estudio?*

    Esta pregunta podía ser respondida con un si o un no.

- **Escuelas de origen.**
  - *¿En qué tipo de establecimiento completó sus estudios básicos?*
  - *¿En qué tipo de establecimiento completó sus estudios medios?*

    Este grupo de preguntas podia responderse escogiendo una de las siguientes alternativas:
    1. público
    2. privado
    3. subvencionado

- **Determinantes personales.**
  - *Género*

    Esta pregunta podía ser respondida escogiendo masculino o femenino.

  - *¿A qué edad ingresó a la carrera?*

    Esta pregunta podía ser respondida ingresando un valor entre 1 y 100.

- **Puntaje de acceso a la universidad.**

  - *¿Cuál fue su puntaje de ingreso a la carrera?*

    Esta pregunta podía ser respondida ingresando un valor entre 350 y 850.

- **Motivos para estudiar.**

  - *¿Qué lo motiva a estudiar informática?*

    Esta pregunta podía responderse escogiendo una de las siguientes alternativas:

    1. conseguir una mejora económica
    2. conseguir estatus social
    3. es una meta personal
    4. nada en particular

- **Variable objetivo**

  - *Ingrese la nota obtenida la primera vez que cursó la asignatura "introducción a la programación".*
  - *Ingrese la nota obtenida la primera vez que cursó la asignatura "programación orientada a objetos".*
  - *Ingrese la nota obtenida la primera vez que cursó la asignatura "estructura de datos".*

    Estas últimas tres preguntas podian ser respondidas seleccionando una de las siguientes alternativas:

    0. aún no la he cursado
    1. entre 1 y 2
    2. entre 2,1 y 3
    3. entre 3,1 y 4
    4. entre 4,1 y 5
    5. entre 5,1 y 6
    6. entre 6,1 y 7

Una vez que el instrumento de recolección de datos fue definido hizo falta establecer el método para hacer llegar dicha encuesta a los estudiantes, para eso se decidió utilizar una plataforma en línea.

4.2.2.2. Implementación del instrumento

Esta subsección tiene la finalidad de describir la forma en la que se implementó el instrumento de recolección de datos, las herramientas utilizadas en el proceso entre otras cosas importantes.

Para implementar la encuesta online se despreciaron herramientas existentes en el mercado como los *Formularios de Google* o *Surveymonkey*, en su lugar se implementó una pequeña aplicación web la cual se montó en el servidor de la Universidad de Bío Bío, accesible desde la url:

http://colvin.chillan.ubiobio.cl/

Es importante recordar que la implementación de esta encuesta no forma parte de los objetivos de este proyecto, sino más bien fue usada como herramienta para conseguir los datos necesarios para el correcto funcionamiento de la aplicación de clasificación, además, cabe mencionar que una de las razones por las que no se utilizaron las herramientas comunes existentes en el mercado es debido al formato de los datos, los cuales sin duda eran mucho mas fáciles de adquirir utilizando una herramienta de obtención de datos hecha a la medida.

Sin embargo, a pesar de lo mencionado en el párrafo anterior, de todas formas se realizará una breve descripción de la manera en que fue desarrollada la encuesta, la arquitectura utilizada y de las herramientas y lenguajes usados para su implementación.

Arquitectura

La programación por capas es un estilo de programación que tiene como finalidad separar la lógica de negocio de la lógica del diseño, en este caso, se utilizan cuatro capas, las cuales se comunican entre si de la forma que muestra la figura 5.1.

Una de las mayores ventajas de organizar el software de esta manera es que los cambios que se produzcan pueden ser implementados de manera más fácil, ayudando así a mantener el orden del código desarrollado.

A continuación se define cada capa de manera genérica y posteriormente como fue abordada particularmente en la implementación de la encuesta.

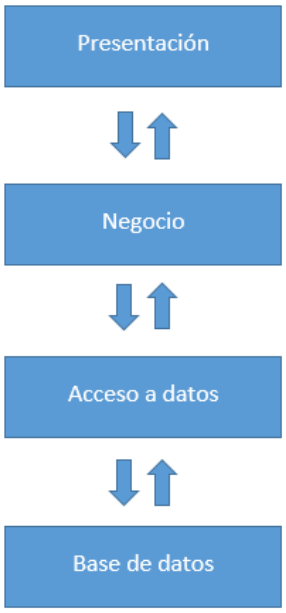

Figura 4.1: Arquitectura de capas

Presentación:

Capa encargada de presentar al usuario la información requerida por este y de además capturar las entradas que este realice, una vez que las obtiene, las valida y envia los datos a la capa de Negocio.

Para el desarrollo de esta capa se utilizó *HTML, HyperText Markup Language*, además del lenguaje *Javascript*, además de librerías como *JQuery* v.1.11 y *Bootstrap* en relación a la validación de los datos, se hizo una verificación para asegurarse que el usuario que participaba de la encuesta pertenecía a la población objetiva del estudio.

Negocio:

La capa de negocio es la encargada de la lógica del negocio o del problema, aquí se ejecutan los programas encargados de comunicar la capa de presentación y los datos que de allí se obtienen con la capa de acceso a datos.

En la implementación de esta capa se utilizó el lenguaje de programación de lado del sevidor *PHP, Hypertext Pre-processor* en su versión 5.3, aquí se recibierón los datos ingresados por el usuario, nuevamente se verificó que fueran datos validos, se procesaron y se enviaron las peticiones requeridas a la capa de acceso a datos.

Acceso a datos:

Capa cuya tarea principal es recibir las peticiones desde la capa de negocio que requieran de los datos almacenados en la base de datos, esta capa se encarga de realizar las consultas en la base de datos y de obtener las respuestas para volver a enviarlas a la capa de negocio, luego la capa de negocio se encargará de enviar a la capa de presentación estas respuestas.

Nuevamente, en la implementación de esta capa se utilizó el lenguaje de programación de lado del sevidor *PHP, Hypertext Pre-processor* en su versión 5.3, además de la libreria *ADOdb* en su versión 5, el cual a la fecha en que se escribió este informe puede ser descargada desde *http://adodb.sourceforge.net/*

Base de datos:

Tal como su nombre lo indica, esta capa alberga la base de datos propiamente tal, estos datos son generalmente administrados por un gestor de base de datos.

Para el desarrollo de esta encuesta se utilizó una base de datos relacional y el gestor de base de datos *MySQL* en su versión 5.5.

Una vez que la encuesta fue desarrollada, se cargaron los archivos y el código fuente al servidor antes mencionado, la cual está accesible en la siguiente dirección web:

http://colvin.chillan.ubiobio.cl:8070/gsoto/encuesta/Presentacion/

4.2.2.3. Aplicación de la encuesta

El instrumento fue aplicado de manera presencial, así como también fue solicitada su completación vía correo electronico a todos el universo de estudiantes cuyo ingreso a la carrera de Ingeniería Civil en Informática haya sido durante los años 2013, 2014 o 2015.

Como la encuesta fue de caracter voluntaria, se obtuvo un total de 70 encuestas completamente constestadas, la cual representará la muestra de datos de entrada para este trabajo.

4.3. Preparación de los datos de entrada

El segundo paso descrito como metodología para implantar una solución de *Machine Learning* corresponde a la preparación de los datos que serán usados como *input* para el algoritmo.

Tal cómo se indicó anteriormente, el universo de datos disponibles para ser usados en la clasificación corresponde a la suma total de 70 encuestas, es decir, 70 tuplas, con 33 atributos cada una, los datos se obtuvieron de la tabla *preguntas* de la base de datos, donde se aplicó una simple consulta SQL, como la que aparece a continuación:

*SELECT * FROM preguntas*

Una vez ejecutada dicha consulta SQL se obtuvo la tabla completa, es decir los 34 atributos que la componen (33 preguntas de la encuesta además de una columna correspondiente al run del alumno que respondió las preguntas), luego estos datos fueron exportados a un documento llamado *encuestas* con formato *CSV, comma-separated values* el cual es usado para representar tablas.

Este documento fue exportado al programa *MS Excel* donde se procedió a eliminar la primera columna, la correspondiente al run del estudiante, ya que este dato no es relevante para el estudio por lo que se obtuvo una tabla con un total de 70 filas y 33 columnas. cuyo formato es presentado en el cuadro 5.1:

De no haber sido porque se diseño una encuesta personalizada para este problema específico, la preparación de los datos pudo haber tomado mucho más tiempo, sin embargo, el haberse esforzado en el diseño del instrumento permitió que con

p1	p2	p3	...	p33
1	3	3	...	1
2	2	2	...	2
3	1	3	...	2
...	...	...	...	2
3	1	3	...	2

Cuadro 4.1: Formato del documento encuestas.CSV

una simple consulta *SQL* a la base de datos se obtuvieran los datos casi totalmente formateados.

4.4. Análisis de los datos de entrada

Tal como se indicó en la descripción de los pasos a seguir para la implantación de una solución de *Machine Learning*, el tercer paso de esta estratégia corresponde al análisis de los datos que serán usados como entrada al algoritmo, cabe recordar que del paso anterior,se obtuvo un documento llamado *encuestas.CSV* el que contenia los datos de todas las encuestas que fueron aplicadas.

Dentro de las actividades que se realizan en esta etapa de análisis se encuentra la revisión de los datos, para realizar tal actividad fue necesario llevar a cabo una lectura de los datos, con el fin de detectar los caracteres extraños, datos incorrectos o que pertenezcan a un dominio distinto al que puede ser manejado por el algoritmo clasificador, entre otros.

En este caso, como el algoritmo aún no es implementado, la primera vez que se hace el análisis de los datos debe hacerse pensando en como será en el futuro el algoritmo de *Machine Learning*, es por esta razón que el primer análisis de los datos, cuando el algoritmo aún no está definido, es susceptible a cambios.

Una vez más, al igual que en la etapa anterior, el hecho de haber desarrollado una encuesta a la medida redujo prácticamente a cero los posibles datos incorrectos, caracteres extraños, etc. esto debido a que fueron validados tanto en la capa de presentación como en la capa de negocio de la aplicación web, (para ver una descripción de las tareas llevadas a cabo en estas capas dirijase al la sección 5.2.2.2.).

4.5. Implementación del algoritmo K Nearest Neighbors

La cuarta etapa de este proyecto es sin lugar a dudas una de las más importantes pues es aquí donde se desarrollará la herramienta de clasificación, como se mencionó anteriormente, para este proyecto se ha seleccionado el algoritmo *K Nearest Neighbors*, esta técnica fue presentada por sus autores Fix y Hodges en el año 1951 en su trabajo titulado "An Important Contribution to Nonparametric Discriminant Analysis and Density Estimation".

Esta sección está organizada de la siguiente manera, primero se da a conocer la estructura general del funcionamiento del programa, para posteriormente pasar a la implementación del software usando el lenguaje de programación *JAVA*.

4.5.1. Estructura general del algoritmo

La implementación del algoritmo clasificador forma parte escencial del objetivo general de este proyecto, para lograrlo se siguió la forma general del funcionamiento de la técnica *K Nearest Neighbors* y se diseñó el siguiente pseudocódigo:

Algorithm 1 Pseudocódigo k nearest neighbors

Require: tuplas de entrenamiento e, tupla de clasificación c, $0 < k_0 <= |e|$.
Ensure: clase a la que pertenece la tupla de clasificación.

1: $k \leftarrow k_0$
2: $tuples \leftarrow e$
3: $s \leftarrow c$
4: $s' \leftarrow \emptyset$
5: **for** $i = 0$ **to** $|tuples|$ **do**
6: $\quad setDistance(s, tuples[i])$
7: **end for**
8: $sortingDistances(tuples)$
9: **for** $i = 0$ **to** K **do**
10: $\quad s' \leftarrow tuples[i]$
11: **end for**
12: $r \leftarrow getMajority(s')$
13: $tuples \leftarrow s$
14: **return** r

El pseudocódigo del algoritmo 1 representa el flujo principal de la implementación de la técnica K Nearest Neighbors, las funciones *setDistance, sortingDis-*

tances y getMajority son descritas en las figuras del algoritmo 2, algoritmo 3 y algoritmo 4 respectivamente.

Antes de continuar con las demás funciones, es necesario describir en detalle el flujo de trabajo del algoritmo, esta implementación necesita tres entradas, la primera de ellas, el conjunto *e* corresponde a las tuplas que servirán de entrenamiento al algoritmo de *Machine Learning*, cabe recordar que estas tuplas deben estar ya etiquetadas con su variable objetivo respectiva. La siguiente variable de entrada, la variable *c* corresponde a la tupla que quiere ser clasificada, lógicamente esta no debe venir etiquetada con su variable objetivo, ese es el trabajo del algoritmo y por último, la última entrada, es la correspondiente al valor de *k*, como se ha mencionado anteriormente este valor debe ser mayor a cero y menor o igual a la cantidad de tuplas que pertenecen a *e*. La salida del algoritmo es la clase a la que pertenece la tupla *c*

Las líneas 1, 2, 3 y 4, definen las variables internas necesarias para el funcionamiento del algoritmo, *K* almacena el valor de k recibido como parámetro, *tuples* es un conjunto que almacena las tuplas recibidas como parámetro *e*, *s* almacena la tupla a ser clasificada *c* y por último el conjunto *s'* se inicializa vacio, es allí donde se almacenarán las k tuplas más cercanas al objeto de clasificación.

Algorithm 2 Pseudocódigo cálculo de distancias

for $i = 0$ **to** $|tuples|$ **do**
  $setDistance(s, tuples[i])$
end for

En las líneas 5, 6 y 7 del algoritmo 1, representadas en el algoritmo 2 se calculan todas las distancias entre la tupla que quiere ser clasificada y cada una de las tuplas dentro del conjunto de entrenamiento, la función *setDistance* se describe a continuación en el algoritmo 3.

Algorithm 3 Pseudocódigo setDistance

Require: tupla a ser clasificada s, tupla i del conjunto tuples.
Ensure: distancia calculada.

1: $t \leftarrow tuples[i]$
2: $n \leftarrow |s.features|$
3: $sum \leftarrow 0$
4: **for** $i = 0$ **to** n **do**
5: $\quad sum \leftarrow sum + (s.feature[i] - t.feature[i])^2$
6: **end for**
7: $t.distance \leftarrow \sqrt{sum}$

Respecto a la función *setDistance*, esta requiere de dos datos de entrada, la tupla *s* y la tupla *i* del cunjunto de tuplas de entrenamiento *tuples*, las líneas 1, 2 y 3 definen las variables auxiliares a usar durante la ejecución de este programa, luego en la línea 4, 5 y 6 existe un ciclo para calcular e ir acumulando las diferencias de los cuadrados para cada atributo de las tuplas. Por último se establece la distancia entre s y t como la raíz cuadrada de dicha sumatoria, esto debido a que se está usando la fórmula de distancia euclideana.

Algorithm 4 Pseudocódigo ordenamiento de distancias

$sortingDistances(tuples)$

En la línea 8 del algoritmo 1 se hace referencia a la función *sortingDistances* la cual se encarga de ordenar de manera ascendente las tuplas del conjunto tuples basándose en el atributo de distancia establecido durante la ejecución repetitiva de la función *setDistance*.

El ordenamiento de tuplas mencionado en el párrafo anterior puede ser realizado utilizando cualquiera de las técnicas de ordenamiento existentes, por lo tanto, no se especificará dicho procedimiento en este documento.

Una vez que las tuplas han sido ordenadas hace falta tomar las primeras *k* tuplas del conjunto *tuples* y ponerlas en el subconjunto *s'*.

La línea 12 del algoritmo 1 hace referencia a la función *getMajority* la cual se describe a continuación:

Algorithm 5 Pseudocódigo getMajority

Require: subconjunto s'
Ensure: variable objetivo con mayor número de apariciones

```
1:  class ← ""
2:  cont ← 0
3:  contAux ← 0
4:  for i = 0 to |s| do
5:      aux ← s[i].target
6:      for j = i to |s| do
7:          if aux == s[j].target then
8:              contAux + +
9:          end if
10:     end for
11:     if contAux > cont then
12:         cont ← contAux
13:         class ← aux
14:         contAux ← 0
15:     end if
16:     contAux ← 0
17: end for
18: return class
```

El pseudocódigo del algoritmo 5, describe la función encargada de recorrer el conjunto *s'* en busca de la variable objetivo que más se repita, si es que se produciera un empate, este código, se quedaría con la primera ocurrencia del elemento mayoría, si bien este código es presentado así, al ser una tarea común como buscar el elemento mayoría dentro de un conjunto, es que podría usarse cualquier otra estratégia que lograra este mismo cometido y el funcionamiento del algoritmo *k nearest neighbors* no se vería afectado en lo absoluto.

Ya para finalizar el análisis del funcionamiento general del algoritmo presentado, sólo hace falta describir las dos últimas líneas de código, la línea 13 del algoritmo 1 añade la tupla s, es decir el elemento que ya fue clasificado, al conjunto de entrenamiento, de esta forma, nos aseguramos que la próxima clasificación se realice teniendo en cuenta esta nueva información adquirida durante este proceso de clasificación.

Finalmente, la línea número 14 del algoritmo 1 retorna la variable *r* a la cual se le había asignado anteriormente el nombre de la clase a la que pertenece *s*, es decir,

el resultado de la clasificación. con esto se da por concluido el procedimiento.

4.5.2. k nearest neighbors usando JAVA

Si bien la técnica de clasificación *K Nearest Neighbors* puede ser implementada utilizando distintos paradigmas de programación, para el desarrollo de esta aplicación se ha escogido la *programación orientada a objetos* y para esto se utilizó el lenguaje de programación *JAVA SE*, para hacer aún más gráfico el funcionamiento y la interacción de las distintas clases que componen el software es que a continuación en la figura se presenta el *diagrama de clases UML* del software de clasificación desarrollado.

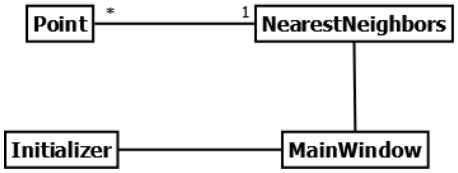

Figura 4.2: Diagrama de clases

Como se puede apreciar en el diagrama de clases de la Figura 5.2, esta aplicación consta de tres clases principales que interectuan entre si, *NearestNeighbors.java, Point.java y MainWindow.java*.

4.5.2.1. NearestNeighbors.java

A continuación se adjunta el código fuente de la clase NearestNeighbors.java

Algoritmo 6: NearestNeighbors.java

```
1  package cl.ubiobio.knearestneighbors;
2
3  import java.util.ArrayList;
4  import java.util.Collections;
5
6  /**
7   * @author gaspar soto
8   *
9   * */
10 public class NearestNeighbors {
11
```

```
12      int k = 0;
13      ArrayList<Point> tuples = new ArrayList<Point>();
14      double[] distances;
15      String[] definedClass;
16
17      /**
18       * Class constructor
19       * @param k an integer to set the value of k for the k
             nearest neigbors.
20       * */
21      public NearestNeighbors(int k){
22         this.k = k;
23      }
24
25      /**
26       * make the training step.
27       * */
28      public void training(Point p){
29         tuples.add(p);
30      }
31
32      /**
33       * Get the class to the point receive how parameter.
34       * @param p a Point object to classify.
35       * @return a String with the class of the point
             classified.
36       * */
37      public String getClass(Point p){
38         setAllDistances(p);
39         sortingDistances();
40         p.setTarget(getMajority());
41         training(p);
42         System.out.println("Class: "+p.getTarget());
43         System.out.println("
              ------------------------------------------\n");
44         return p.getTarget();
45      }
46
47      /**
48       * Get the class with the most appearances in the
             neighborhood.
49       * @return a String with the class with most appearances.
50       * */
51      private String getMajority(){
52         String aux ="";
```

```
53          String clase = "";
54          int cont = 0;
55          int contAux = 0;
56          System.out.println("\n****** The K NearestNeighbors
                are: ****** ");
57          for(int i=0;i<k;i++){
58             aux=tuples.get(i).getTarget();
59             for(int j=i;j<k;j++){
60                if(aux.equalsIgnoreCase(tuples.get(j).getTarget
                      ())){
61                   contAux++;
62                }
63             }
64             if(contAux>cont){
65                cont = contAux;
66                clase = aux;
67                contAux = 0;
68             }
69             contAux=0;
70             System.out.println("Distance to target: " + tuples.
                   get(i).getDistanceToTarget()
71                         + "\t class: "+ tuples.get(i).getTarget
                              ());
72          }
73          System.out.println();
74          return clase;
75       }
76
77       /**
78        * Set all distances from the point p to every point
             saved in the training step.
79        * */
80       private void setAllDistances(Point p){
81          System.out.println("****** All Distances *******");
82          for(int i=0;i<tuples.size();i++){
83             tuples.get(i).setDistanceToTarget(p.getDistance(
                   tuples.get(i)));
84             System.out.println("distance point "+(i+1)+": "+ +
                   p.getDistance(tuples.get(i)));
85          }
86       }
87
88       /**
89        * Sort all distances.
90        * */
```

```
91      private void sortingDistances(){
92         Collections.sort(tuples);
93      }
94   }
```

4.5.2.2. Point.java

Una de las ventajas de utilizar la orientación a objetos es que nos permite crear clases que asistan a la solución general del problema, ayudando así a la abstracción de este.

Un ejemplo de lo anterior es la clase *Point.java*, la cual como puede deducirse de su nombre, ayuda a representar puntos en el espacio de n dimensiones para así poder facilitar el cálculo de distancia entre ellos, como para ayudar al entendimiento general del problema y su solución. A continuación, en la figura del *Algoritmo 7* se provee el código fuente de la clase *Point.java*.

Algoritmo 7: Point.java

```
1
2   package cl.ubiobio.knearestneighbors;
3
4   /**
5    * @author gaspar soto
6    *
7    * */
8   public class Point implements Comparable<Point>{
9
10    private double[] features;
11    private String[] labels;
12    private String target;
13    private String labelTarget;
14    private double distanceToTarget;
15
16    /**
17     * Class constructor.
18     * @param l a String array to represent labels of this
          point.
19     * @param f a double array to save the features of this
          point.
20     * */
21    public Point(String[] l, double[] f){
22      this.features = f;
```

```
23        this.labels = l;
24        this.labelTarget = "";
25      }
26
27      /**
28       * Class Constructor.
29       * @param l a String array to represent labels of this
             point.
30       * @param f a double array to save the features of this
             point.
31       * @param lt a String to save the label of the target of
             this point.
32       * @param t a String to save the value of the target of
             this point.
33       * */
34      public Point(String[] l, double[] f, String lt, String t)
           {
35        this.features = f;
36        this.labels = l;
37        this.labelTarget = lt;
38        this.target = t;
39      }
40
41      /**
42       * Calculates the distance between two points.
43       * @param Point a Point object to calculates the distance
             .
44       * @return the distance to point set by param.
45       * */
46      public double getDistance(Point a){
47        double sum = 0;
48        double[] featuresA = a.getFeatures();
49        for(int i=0;i<amountOfFeatures();i++){
50          sum+=Math.pow(this.features[i]-featuresA[i],2);
51        }
52        return Math.sqrt(sum);
53      }
54
55      /**
56       * Get the list with the features of this point.
57       * @return a double array with the features of this point
             .
58       * */
59      public double[] getFeatures(){
60        return features;
```

```
61      }
62
63      /**
64       * Get the list with the labels of this point.
65       * @return a String array with the labels of this point.
66       * */
67      public String[] getLabels(){
68        return labels;
69      }
70
71      /**
72       * Get the label in the position i of this point.
73       * @return a String with the labels in the position i of
             this point.
74       * */
75      public String getLabelInPosition(int i){
76        return labels[i];
77      }
78
79      /**
80       * Get the features in the position i of this point.
81       * @return a double with the feature in the position i.
82       * */
83      public double getFeatureInPosition(int i){
84        return features[i];
85      }
86
87      /**
88       * Get the amount of features of this point.
89       * @return the amount of features of this point.
90       */
91      public int amountOfFeatures(){
92        return features.length;
93      }
94
95      /**
96       * Get the target of this point.
97       * @return a String with the target of this point.
98       * */
99      public String getTarget(){
100       return this.target;
101     }
102
103     /**
104      * Set the target to this point.
```

```
105      * */
106     public void setTarget(String t){
107       this.target = t;
108     }
109
110     public String getLabelTarget(){
111       return this.target;
112     }
113
114     public void setLabelTarget(String t){
115       this.target = t;
116     }
117
118     public void setDistanceToTarget(double d){
119       this.distanceToTarget = d;
120     }
121
122     /**
123      * Get the distance of this point unto the target.
124      * @return a double with the distance unto the target.
125      * */
126     public double getDistanceToTarget(){
127       return this.distanceToTarget;
128     }
129
130     /**
131      *
132      */
133     public int compareTo(Point o) {
134       return new Double(this.distanceToTarget).compareTo(o.
              distanceToTarget);
135     }
136   }
```

Estas dos clases presentadas anteriormente (*NearestNeighbors.java* y *Point.java*) forman parte de los requerimientos mínimos para hacer funcionar la herramienta de software de *Machine Learning*, además de estas dos clases, se ha creado una interfáz gráfica para ayudar a la interacción del usuario con el programa, sin embargo, el código fuente de esta clase no se describe en este documento.

4.6. Entrenamiento del algoritmo

Durante la ejecución de esta etapa es cuando el programa de *Machine Learning* adquiere el primer conjunto de conocimiento, este es muy importante pues es desde aquí donde obtendrá la base de conocimiento para la toma de futuras decisiones, además de esto, es importante asegurarse que las decisiones que sean tomadas posteriormente al proceso de entrenamiento se añadan a la base de conocimiento del algoritmo para futuras ejecuciones de este.

Como se mencionó anteriormente, se cuenta con un total de 70 encuestas, de las cuales 70 servirán para clasificar a los alumnos que han tenido el curso de *Introducción a la programación*, 42 encuestas pertenecientes a este mismo universo de datos servirán para clasificar a los alumnos de *Programación orientada a objetos* y por último, 24 encuestas serán útiles en la clasificación de los alumnos del curso de *Estructuras de datos*.

4.6.1. Organización de los datos

Se han realizado tres conjuntos de clasificaciones, una para cada asignatura estudiada, en relación a la organización de los datos disponibles se ha utilizado la siguiente configuración:

	Curso 1	Curso 2	Curso 3
Tuplas totales	70	42	23
Tuplas para entrenamiento	53	32	16
Tuplas para prueba de clasificación	17	10	7

Cuadro 4.2: Organización de los datos

En el cuadro 5.2 se define la cantidad de datos usados para clasificación y para entrenamiento, se estableció dejar para la etapa de entrenamiento un total correspondiente al 70 % de los datos disponibles, dejando así el resto de los datos para las pruebas de clasificación.

4.7. Testeo del algoritmo

Como se mencionó durante la descripción de cada etapa, la etapa de testeo del algoritmo tiene la finalidad de medir que tan acertado es nuestro programa, por

razones de organización del documento es que esta etapa tiene su apartado propio en el capítulo 5 “Resultados obtenidos”.

4.8. Uso del algoritmo

Esta última sección de la metodología usada para la implantación de una solución de *Machine Learning* corresponde al uso de la herramienta ya lista con los parámetros óptimos para el problema.

En los siguientes párrafos se presenta un breve tutorial de la herramienta de clasificación desarrollada.

4.8.1. Instrucciones básicas de uso

Para usar este software tenga bien ejecutar el archivo o bien compilar el código fuente adjunto a este documento (este programa fue compilado con la versión 1.8.0_66 provista por Oracle, lo que no quiere decir que con versiones anteriores posteriores o previas a esta no sea posible realizar la compilación) una vez hecho esto usted verá una ventana como la de la Figura 5.3.

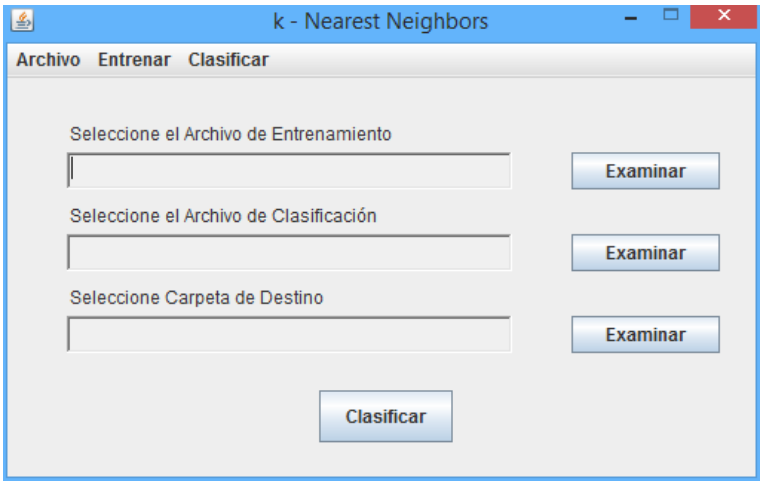

Figura 4.3: Ventana principal K Nearest Neighbors

Como puede apreciar la ventana principal del programa posee una barra de herramientas y cuatro botones principales, que serán lo que usaremos, lo primero que

debe hacer es hacer clic en el boton correspondiente a la sección *"Seleccione el Archivo de Entrenamiento"* al hacer esto verá una ventana emergente como la de la figura 5.4.

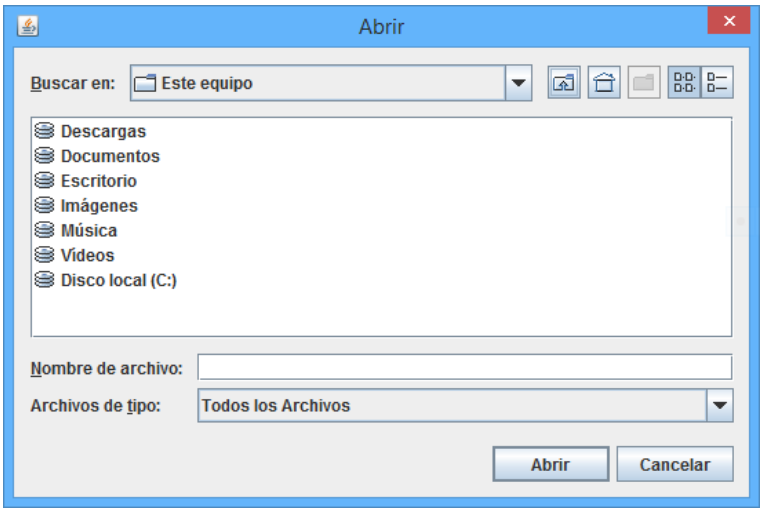

Figura 4.4: Ventana emergente para seleccionar archivos

Aquí busque el archivo CSV con los datos de entrenamiento, pueden ser todas las tuplas que estime conveniente siempre y cuando cumplan con el siguiente formato:

A1	A2	A3	...	VO
1	3	3	...	1
2	2.2	2.5	...	2
3	1.8	3	...	2
...	...	...	...	2
3	1	3	...	2

Cuadro 4.3: Formato genérico del archivo de entrenamiento

Como puede apreciar en el Cuadro 5.3, el documento CSV de entrenamiento debe traer en la primera fila los nombres de las etiquetas para cada atributo o columna, además es necesario recordar que durante la etapa de entrenamiento los datos deben ir con la variable objetivo ya identificada.

Una vez seleccionado el archivo de entrenamiento haga clic en el segundo botón *"Examinar"*, el correspondiente a la sección de la ventana que dice *"Seleccione el Archivo de Clasificación"*, verá una ventana emergente identica a la de la Figura 5.6 pero esta vez seleccione el archivo CSV con los datos que quiere clasificar.

El archivo de datos a clasificar debe estar formateado de manera similar a lo indicado en el cuadro 5.3 salvo algunas diferencias, primero, el archivo con los datos a clasificar no posee una fila con los nombres de las etiquetas, sino sólo los datos a clasificar, además tampoco posee la última columna correspondiente a la variable objetivo ya que esa es la columna que queremos identificar, el cuadro 5.4 muestra el formato general para los archivos de clasificación.

1	3.5	...	1
2.2	2	...	2
3	3.1	...	2.9
...	...	...	2
3	3	...	2

Cuadro 4.4: Formato genérico del archivo de clasificación

Para ambos archivos, tanto el de entrenamiento como el de clasificación, es posible usar valores de punto flotanto o enteros.

Ya con los archivos de entrenamiento y clasificación seleccionados, es necesario escoger el nombre, el formato y la carpeta de destino del archivo con los resultados. Haga clic en el tercer botón *"Examinar"* el correspondiente a la sección de la ventana que dice *"Seleccione Carpeta de Destino"*, al hacer esto verá una ventana emergente como la que muestra la Figura 5.5.

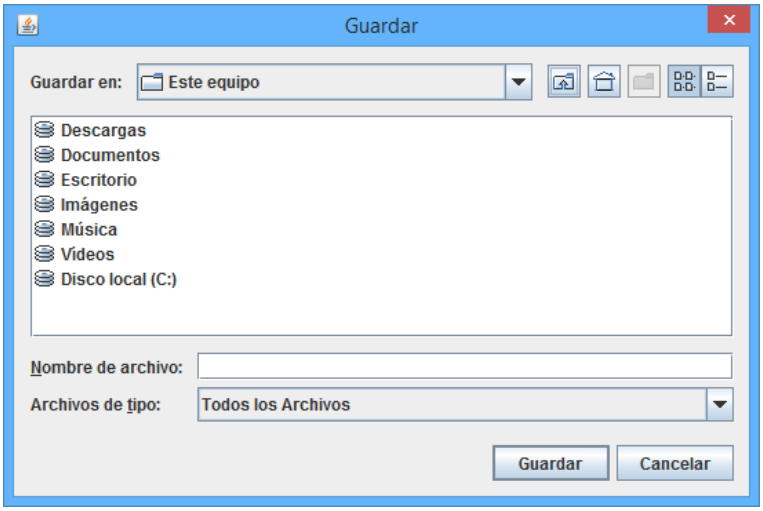

Figura 4.5: Ventana emergente para guardar archivos

En esta ventana emergente busque entre sus directorios la carpeta donde quiere que sean almacenados los resultados, luego, en el campo *Nombre del archivo* ingrese el nombre para el documento de resultados, por ejemplo *"resultadosClasificacion.CSV"*, una vez hecho esto haga clic en *Guardar*, la ventana principal del programa se debería ver como muestra la figura 5.6.

Figura 4.6: Ventana principal K Nearest Neighbors completa

Para finalizar haga clic en el botón central *"Clasificar"*, verá una ventana emergente como la de la figura 5.7 pidiendole que ingrese un valor para el factor k, recuerde que este debe ser mayor a cero y menor o igual a la cantidad de datos pasados como entrenamiento.

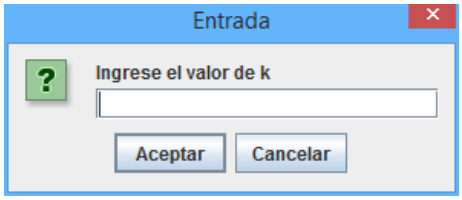

Figura 4.7: Ventana emergente para el valor de k

Si todo el proceso se realizó de manera correcta, verá un mensaje diciendo: *"Clasificación realizada con éxito"* y verá dos archivos de resultado, el archivo *CSV* con los datos de la clasificación, y un archivo *LOG* con el detalle de la clasificación realizada, las distancias al objetivo, las distancias mínimas, entre otros datos útiles.

4.8.2. Descripción de los documentos de resultado

Tal como se mencionó en el párrafo anterior, la salida del programa incluye dos archivos de texto, uno con extensión *CSV* (puede ser otro, pero este es el formato recomendado) y otro con extensión *LOG*, el próposito de esta sección es entregar una breve descripción de estos.

CSV

El archivo de salida *CSV*, es un archivo prácticamente igual al archivo de clasificación falicitado como entrada del programa, salvo una diferencia, cada tupla tiene su variable objetivo ya definida.

LOG

El archivo *LOG* es un archivo que contiene los conjuntos *S* y *S'* para cada tupla, además del detalle de la clasificación de estas, este documento es similar al de la Figura 5.8.

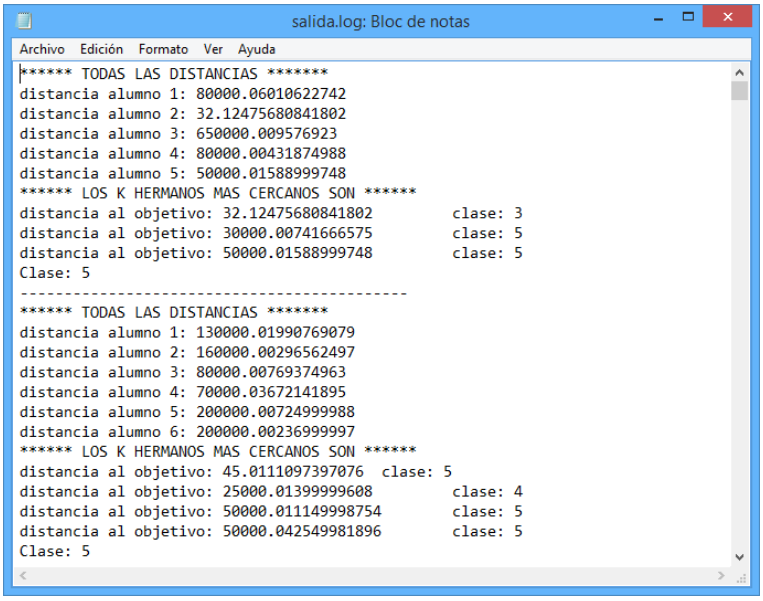

Figura 4.8: Ventana emergente para el valor de k

Capítulo 5

Resultados obtenidos

En este capítulo se da a conocer los resultados obtenidos con la aplicación del algoritmo *K Nearest Neighbors* en el intento de dar solución al problema planteado.

Cada intento de predicción de las notas de los estudiantes se ha tomado como un problema individual , es decir, la predicción del rendimiento académico de los alumnos del curso de introducción a la programación corresponde a una ejecución distinta a la de los alumnos de programación orientada a objetos, del mismo modo sucede con el curso de estructuras de datos.

A su vez, los resultados son presentados en tablas separadas, las cuales están compuestas de cuatro columnas, la primera corresponde al valor de k utilizado en la clasificación, la segunda columna muestra la tasa de acierto del algoritmo para dicha ejecución, la tercera columna enseña el porcentaje de datos mal clasificados, mientras que la última columna da a conocer el valor del error cuadrático medio para dicha ejecución. Es necesario mencionar que cada intento de clasificación se realizó utilizando diferentes valores para el parametro K (su relevancia fue conversada en capítulos anteriores).

Además de los cuadros resumen, se han añadido gráficos de barras para el mejor resultado de clasificación obtenido para cada caso, el objetivo de estos gráficos es facilitarle al lector la visualización de los valores esperados en comparación con los valores obtenidos por el algoritmo.

Para facilitar la lectura de los resultados es importante recordar que la variable objetivo para todos los casos corresponde a rangos de calificaciones, desde el 1 al 6, donde:

1. Calificación entre 1 y 2
2. Calificación entre 2,1 y 3
3. Calificación entre 3,1 y 4
4. Calificación entre 4,1 y 5
5. Calificación entre 5,1 y 6
6. Calificación entre 6,1 y 7

El primer resultado corresponde al intento de predecir el rendimiento académico de los estudiantes en el curso de introducción a la programación, el resumen de los resultados se muestra en la tabla del Cuadro 6.1.

Valor de K	Correctas	Incorrectas	ECM
1	29,41 %	70,59 %	1,35
4	41,18 %	58,82 %	1,24
7	35,29 %	64,71 %	1,59
27	23,53 %	76,47 %	1,88

Cuadro 5.1: Resultados de predicción del curso de introducción a la programación

La mayor tasa de efectividad fue alcanzada con la parametrización $k = 4$, logrando un $41\,\%$ y un índice de error del $1,24$.

Este resultado es presentado también en el gráfico de la Figura 6.1, donde se puede apreciar fácilmente los valores esperados y los valores obtenidos con el programa.

El segundo resumen de resultados corresponde a la clasificación de los resultados de los estudiantes del curso de programación orientada a objetos, aquí, en el Cuadro 6.2. se muetran los detalles.

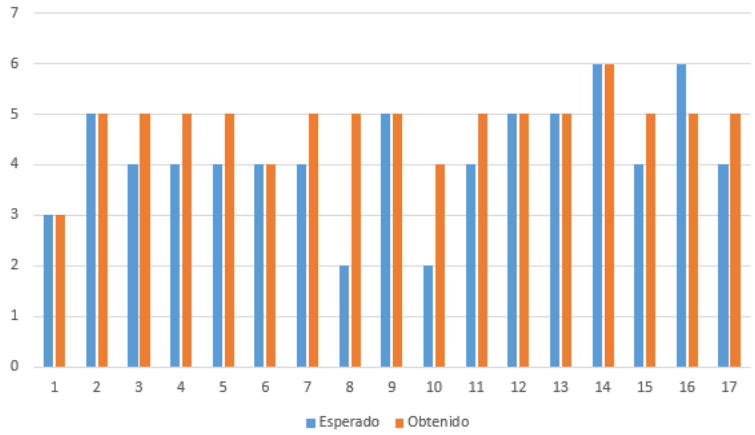

Figura 5.1: Gráfico comparativo Introducción a la programación

Valor de K	Correctas	Incorrectas	ECM
1	20 %	80 %	1,4
8	40 %	60 %	0,9
16	60 %	40 %	0,4
32	60 %	40 %	0,4

Cuadro 5.2: Resultados de predicción del curso de programación orientada a objetos

Como puede apreciar, para esta ejecución, la tasa de acierto mejoró considerablemente respecto a la anterior, aquí, para un $k = 16$ el porcentaje de acierto en la predicción alcanza el $60\,\%$ con un índice de error muy bajo, lo que indica no sólo una buena clasificación sino también que cuando la clasificación fue errónea, la diferencia entre el valor esperado y el obtenido fue muy pequeña.

El gráfico respectivo se presenta a continuación, en la Figura 6.2.

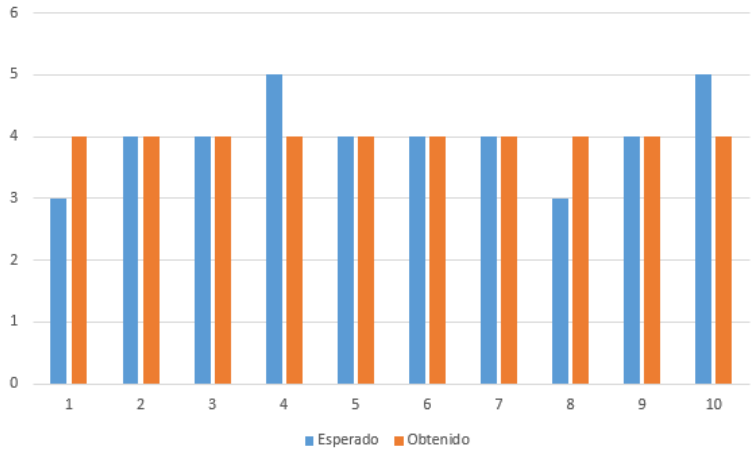

Figura 5.2: Gráfico comparativo programación orientada a objetos

Finalmente, se presentan en el Cuadro 6.3 los resultados de la clasificación de los datos referentes al curso de estructura de datos.

Valor de K	Correctas	Incorrectas	ECM
1	42,56 %	57,14 %	1,43
6	28,57 %	71,43 %	1,57
12	57,14 %	42,86 %	0,43
16	57,14 %	42,86 %	0,43

Cuadro 5.3: Resultados de predicción del curso de estructuras de datos

Aquí, la tasa de éxito en el mejor de los casos, fue similar a la anterior, alcanzando un $57,14\,\%$ y un índice de error del $0,43$, a su vez, el peor caso de clasificación, correspondiente a la parametrización $k = 6$, obtuvo sólo un $28,57\,\%$ de efectividad.

Al igual que en los casos anteriores, para una mejor comprensión de los resultados de la mejor clasificación, se presenta un gráfico de barras en la Figura 6.3.

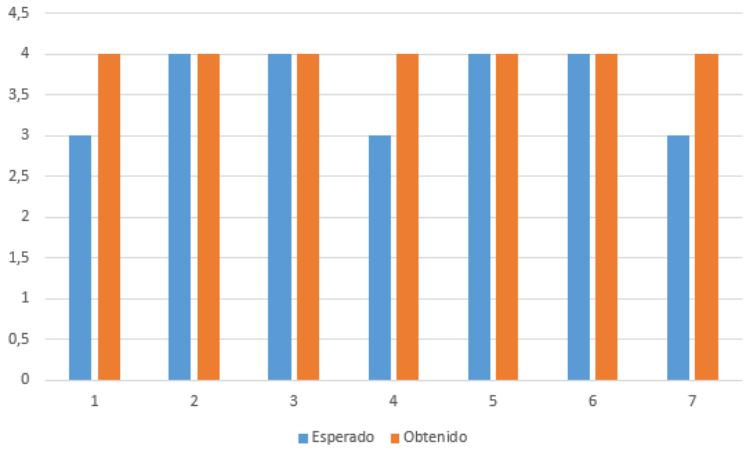

Figura 5.3: Gráfico comparativo estructuras de datos

Así concluye esta sección, se ha pretendido presentar de manera objetiva los resultados obtenidos, es importante mencionar que se trabajó sólo con algunas de las combinaciones posibles a la hora de organizar los datos, y que por lo tanto, otras combinaciones, ya sea en la etapa de entrenamiento o de clasificación del algoritmo podrían haber generado otros resultados, sin embargo, al realizar una mirada holística del trabajo realizado es posible concluir que los resultados presentados en este capítulo son una prueba fidedigna de la capacidad actual del algoritmo implementado.

Capítulo 6

Conclusiones del trabajo

La primera impresión que ha dejado el desarrollo de este proyecto es que la tarea de predecir el rendimiento académico de estudiantes es complicada y como se dio a conocer en el capítulo 3, muchos autores han abordado este problema, de variadas maneras, algunos con más éxito que otros.

Y es en este documento, donde se ha puesto en evidencia una manera distinta de enfrentarlo, sin embargo, distinto no siempre es sinónimo de mejor, la técnica empleada en en este proyecto, *K Nearest Neighbors*, si bien ha conseguido indices de error bastante bajos, no se ha logrado una gran tasa de acierto en la clasificación de las variables, al menos no la que se podría haber esperado.

Esto puede deberse a la técnica misma, es decir, podrían ser las caracteristicas propias de este algoritmo la que lo hacen poco eficiente para este problema, otra posible causa de esto podría tener relación con la calidad de los datos utilizados, ya que al ser obtenidos por medio de la participación de terceros podrían carecer de veracidad y objetividad, influyendo así en el correcto funcionamiento del clasificador.

La razón principal por la cual se escogió esta técnica en desmedro de otras con alto índice de acierto, por ejemplo *JRip*, tiene su fundamento en la relación existente entre la complejidad de la implementación y los potenciales resultados del uso de dicha técnica.

Si bien algoritmos de aprendizaje basado en reglas como *JRip*, pueden tener altas tasas de acierto en la clasificación, su implementación es compleja, sin embargo,

K Nearest Neighbors es bastante más simple de implementar y ha demostrado su potencia en problemas de clasificación similares. (Texeira y Oliveira, 2010).

Sería recomendable seguir intentando predecir el rendimiento académico, ya que lograrlo permitiría tomar medidas tempranas para evitar la deserción de estudiantes o el estancamiento en su desarrollo educacional.

Estos nuevos intentos podrían usar variaciones a la técnica *K Nearest Neighbors* o derechamente utilizar otras estratégias de clasificación, como JRip, además de modificar la herramienta de clasificación, es estrictamente necesario poner énfasis en el modelo de factores que influyen en el rendimiento de los estudiantes, y en los datos recolectados, ya que al parecer, allí reside la clave del éxito.

Respecto a las lecciones aprendidas, sin lugar a dudas, uno de los aspectos que representó el mayor desafío, fue el modo con el cual sobreponerse a la naturaleza subjetiva de las variables que influyen en el rendimiento de un estudiante, ya que muchas de estás son difíciles de identificar, medir y ponderar, haciendo muy complicado el trabajo para cualquier algoritmo de *Machine Learning*.

Además de lo anterior, este proyecto permitió un acercamiento a las técnicas de *Machine Learning* que sin lugar a duda estarán muy de moda debido al crecimiento exponencial de los datos existentes en la actualidad.

Lista de referencias

Bell, J., (2015). *Machine Learning Hands-On for Developers and Technical Professionals*, Indianapolis, USA, Wiley.

Caso, J. y Hernández, L., (2007). Variables que inciden en el rendimiento académico de adolescentes mexicanos. *Revista Latinoamericana de Psicología*, 39.

Corengia, A., Pita, M., Mesurado, B., y Centeno, A., (2011). La predicción de rendimiento académico y deserción en estudiantes universitarios. *LIBERABIT*, 19.

Dapozo, G., Porcel, E., Lopez, M., Greiner, C., (2008). Técnicas de clasificación aplicadas al estudio del rendimiento de ingresantes universitarios. In *X Workshop de Investigadores en Ciencias de la Computación.*.

Dietterich, T., (1996). Machine learning. ACM Computing Surveys.

Edel, R. (2003). El rendimiento académico: concepto, investigación y desarrollo. *Revista Electrónica Iberoamericana sobre Calidad, Eficacia y Cambio en Educación*, 1.

Garbanzo, G. (2007). Factores asociados al rendimiento académico en estudiantes universitarios, una reflexión desde la calidad de la educación superior pública. *Revista Educación*, 31.

Gonzáles, P.y Oyarzo, P. (2000). Factores que determinan el rendimiento académico de los alumnos de la Universidad de Talca. Universidad de Talca, Chile

Han, J., Kamber, M. y Pei, J. (2012). *Data Mining, Concepts and Techniques*, Waltham, USA, Elsevier.

Harrington, P., (2012) *Machine Learning in Action*, Shelter Island, USA, Manning.

Iglesias, L. y Vera, V., (2010). Factores psicológicos, sociales y demográficos asociados al rendimiento académico en estudiantes universitarios. Psicol, 12.

Miquel, J., Expósito, M. y Sempere, S., (2014). Determinantes del rendimiento académico en los estudiantes de grado. Un estudio en administración y dirección de empresas. *RIE*, 32.

Myatt, G. y Johnson, W., (2014). *Making Sense of Data I*, USA, Wiley.

Planck, U. (2014). Factores determinantes del rendimiento académico de los estudiantes de la Universidad de Atacama. *Estudios Pedagógicos*, 11.

Salgueiro, F., Costa, G., Cánepa, S., Lage, F., Kraus, G., Figueroa, N. y Cataldi, Z., (2006). Redes neuronales para predecir la aptitud del alumno y sugerir acciones. *VIII Workshop de Investigadores en Ciencias de la Computación.*

Spositto, O., Etcheverry, M., Ryckeboer, H., Bossero, J. (2010). Aplicación de Técnicas de minería de datos para la evaluación del rendimiento académico y la deserción estudiantil. In *Novena Conferencia Iberoamericana en Sistemas, Cibernética e Informática, CISCI* Vol. 29, pp. 06-2.

Teixeira, L. A., Oliveira, A. L. (2010). A method for automatic stock trading combining technical analysis and nearest neighbor classification. *Expert Systems with Applications*, 37, pp. 6885–6890.

Timarán, R., (2009). Detección de Patrones de Bajo Rendimiento Académico y Deserción Estudiantil con Técnicas de Minería de Datos. In *Memorias de la 8ª Conferencia Iberoamericana en Sistemas, Cibernética e Informática CISCI.*

Torres, L y Rodriguez, N. (2006). Rendimiento académico y contexto familiar en estudiantes universitarios. *Enseñanza e investigación en Psicología*, 11.

Valle, A., Gonzáles, R., Núñez, J. y Gonzáles, J., (1998). Variables cognitivo-motivacionales, enfoques de aprendizaje y rendimiento académico. *Psicothema*, 10.

Vera, Carlos., Romero, C. y Ventura, S., (2012). Predicción del fracaso Escolar mediante técnicas de Minería de Datos. *IEEE-RITA*, 7.

Zambrano, C., Rojas, D. y Carvajal, K., (2011). Análisis de rendimiento académico estudiantil usando data warehouse y redes neuronales. *Ingeniare. Revista chilena de ingeniería*, 19, pp. 369-381.