# 4 Machine Learning

El objetivo de este capítulo es visualizar qué aspectos de Machine Learning fueron tomados como componentes de solución al problema de estudio.

En lo que va de la Edad Contemporánea, el hombre siente fuertemente la necesidad de encontrar respuestas a ciertos aspectos de sus propias capacidades, respuestas de cómo funciona el cerebro humano, cómo se van originando e hilando los pensamientos, cómo se va adquiriendo el conocimiento, cómo la racionalidad está presente en las decisiones humanas, cuál es el mecanismo de toma de decisiones de la mente humana, cómo utiliza su inteligencia para resolver problemas e ideas abstractas, cuál es el proceso que sigue la mente para poder aprender, interrogantes sobre la memoria del cerebro y cómo los sentidos van alimentando de percepciones para tener una visión única del universo. La ciencia de la computación ha abierto la puerta a dichas respuestas.

En 1950 Alan Turing dio uno de los saltos más importantes al proponer el enfoque de su “Prueba de Turing”, donde un computador es considerado como agente inteligente si un evaluador humano, sin interactuar con el computador, realiza preguntas y no es capaz de distinguir si las respuestas vienen de una persona o del computador. En su artículo “Computing Machinery and Intelligence” [[Tur50](" \l "biblio-4)] expuso este y otros conceptos y como tal es considerado padre de la Inteligencia Artificial (IA[↓](" \l "nom-ia)).

Aunque hoy en día aún es válida la Prueba de Turing, el desafío no está en construir un agente inteligente total, sino mas bien buscar que un computador tenga capacidades como [[RN04](#biblio-102)]:

1. Procesamiento del lenguaje natural
2. Representación del conocimiento
3. Razonamiento automático
4. Aprendizaje automático (Machine Learning)
5. Visión computacional
6. Robótica

Entre los precursores de IA están John McCarthy, Marvin Minsky, Allen Newell y Herbert Simon, todos ganadores del premio “ACM A.M. Turing Award” [[Awa??](" \l "biblio-1)] por sus notables aportes en los inicios de IA. McCarthy acuño el término Artificial Intelligence en el taller “Dartmouth Summer Research Project on Artificial Intelligence” que organizó y se desarrolló en junio del año 1956.

En palabras transcriptas, el taller perseguía el siguiente objetivo: *“El estudio debe proceder sobre la base de la conjetura de que cada aspecto del aprendizaje o cualquier otra característica de la inteligencia puede, en principio, describirse tan precisamente que se puede hacer que una máquina lo simule. Se intentará encontrar cómo hacer que las máquinas usen el lenguaje, formen abstracciones y conceptos, resuelvan tipos de problemas ahora reservados para los humanos y se mejoren a sí mismos. Creemos que se puede lograr un avance significativo en uno o más de estos problemas si un grupo de científicos cuidadosamente seleccionados trabajan juntos durante un verano”*.

Entre los aportes más destacados de McCarthy se pueden mencionar la definición del lenguaje de alto nivel Lisp, la creación del tiempo compartido y el artículo “Programs with Common Sense” [[McC59](" \l "biblio-54)] donde define su *Generador de Consejos*, todos estos aportes desarrollados en 1958. Minsky diseñó SNARC[↓](" \l "nom-snarc) (Stochastic Neural Analog Reinforcement Calculator) la primera máquina a partir de una red neuronal en el año 1951, junto a Seymour Papert publicó el libro “Perceptrons: an introduction to computational geometry” [[MP87](" \l "biblio-76)] en el año 1969 y también creó un modelo de redes semánticas denominadas “marcos” publicado en “A framework for representing knowledge” en el año 1974 [[Min74](" \l "biblio-77)].

En setiembre de 1956 en el “MIT[↓](" \l "nom-mit) Symposium on Information Theory” se mostraron trabajos muy importantes como el de George Miller que presentó “The Magical Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information”, Noam Chomsky presentó “Three Models for the Description of Language”, y Allen Newell y Herbert Simon presentaron “The Logic Theory Machine”: programa de computador capaz de hacer la demostración de un teorema y considerado el primer programa de IA.

El campo de la ciencia cognitiva echó sus raíces en este simposio evidenciando que los modelos informáticos se pueden utilizar para modelar la psicología de la memoria, el lenguaje y el pensamiento lógico. En 1959 presentaron el programa “General Problem Solver” (GPS[↓](" \l "nom-gps)), pretendía funcionar como una máquina universal para resolver problemas y fue el primer programa de computador que separó su conocimiento de los problemas (reglas representadas como datos de entrada) de su estrategia de cómo resolver problemas (un motor de resolución genérico).

Machine Learning como subcampo de IA está presente en muchas aplicaciones de la vida real, especialmente en aquellas donde se requiere el procesamiento de grandes cantidades de datos. Por esta razón la tecnología de la información la toma como aliado esencial. Muchos avances tecnológicos de última generación utilizan algoritmos de Machine Learning para realizar un análisis inteligente de los datos [[SV08](#biblio-5)]. Entre algunas aplicaciones conocidas se destacan:

* Reconocimiento facial de Facebook: técnicas de Computer Vision and Pattern Recognition. https://research.fb.com/learning-to-segment/
* Kinect para Xbox 360: técnicas de reconocimiento de voz y reconocimiento facial para la identificación automática de los usuarios.
* Voice recognition.
* Campo relacionado del reconocimiento de caracteres manuscritos.
* Motores de búsqueda.
* Recomender system en plataformas como Amazon, Netflix, Facebook.
* Reconocimiento automático de ciertas áreas en el mundo realizado por satélites.

Para dimensionar el gran crecimiento de los datos digitales en el mundo, en un estudio publicado por la International Data Corporation (IDC[↓](" \l "nom-idc)) y patrocinado por Dell EMC que se denomina: “The Digital Universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East” [[GR12](" \l "biblio-53)] se analizó qué tan rápido crecerían los datos cada año. La medida incluyeron todos los datos digitales creados, replicados o consumidos en un solo año. En la Figura [4.1↓](#fig:IDC-exabyes) se muestra una proyección del tamaño hasta el 2020.

Figure 4.1 Proyección de crecimiento de datos del 2005 al 2020.

Los datos incluían: videos de YouTube, películas, datos bancarios, imágenes de cámaras de seguridad en aeropuertos, datos de las colisiones subatómicas del Gran Colisionador de Hadrones en el CERN[↓](" \l "nom-cern), llamadas telefónicas, mensajes de texto, etc. Los resultados indicaron que en el 2005 existían 130 exabytes en datos digitales, en el 2010 llegó a 1200 exabytes, en el 2015 a unos 7900 exabytes y para el 2020 se pronostica que llegará a los 40000 exabytes. IDC estima que para el 2020 hasta el 33% de los datos digitales contendrán información valiosa.

## 4.1 Definición

El término Machine Learning fue propuesto por Arthur Samuel en el año 1959 mientras trabajaba para IBM[↓](" \l "nom-ibm). En el IBM Journal publicó “Some Studies in Machine Learning Using the Game of Checker” en el cual escribió: *“Programming computers to learn from experience should eventually eliminate the need for much of this detailed programming effort”* [[Sam59](" \l "biblio-8)] por lo que de su artículo se interpreta que *“Machine Learning es un campo de estudio que da a las computadoras la capacidad de aprender a resolver problemas sin ser explícitamente programadas*”.

Samuel desarrolló en el año 1952 un programa que aprendió a jugar Damas, hasta llegar a una categoría amateur. Este pionero ya demostraba que los programas a través del aprendizaje pueden efectuar tareas de toma de decisiones sin ser programadas explícitamente dichas decisiones. Cabe destacar que Arthur Samuel fue uno de los asistentes del “Dartmouth Summer Research Project on Artificial Intelligence”.

Otro investigador, Tom Mitchell propuso en 1998 la siguiente definición: *“A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E”*. Donde se nos indica que el aprendizaje en las máquinas deberá ser parecido al aprendizaje en los humanos, por ejemplo cuando una criatura comienza a hablar a través de la experiencia de pronunciar las palabras y de su interacción con otras personas, entonces sucede que su capacidad de hablar se va perfeccionando o mejorando [[Mit97](#biblio-78)].

Otra definición es *“The purpose of machine learning is to learn from training data in order to make as good as possible predictions on new, unseen, data”*[[Pug16](" \l "biblio-49)]. La dificultad radica en que debemos construir modelos que nos acerquen a una buena predicción sobre datos aún no conocidos o imprevistos.

En el contexto de IA y según el enfoque propuesto en el libro “Inteligencia Artificial Un Enfoque Moderno” [[RN04](#biblio-102)], se conciben los sistemas inteligentes como agentes racionales dotados de capacidades específicas. Entre las capacidades principales están el poder percibir el entorno de trabajo (“problemas para los cuales fueron hechos”) con la ayuda de sus sensores, actuar en ese entorno mediante sus actuadores y contar con una medida de rendimiento para medir el éxito.

Estructuralmente un agente = arquitectura + programa. Los sensores y actuadores forman parte de la arquitectura, la arquitectura es el medio físico que puede ser una PC[↓](" \l "nom-pc) o un robot con sensores, etc. El objetivo principal de la IA es construir el programa del agente, que es donde se implementa la función del agente. Para que el agente sea considerado un agente que aprende, el programa debe contener el “Elemento de aprendizaje”. En la Figura [4.2↓](#fig:Agente-racional) se grafica el concepto, un modelo de agente inteligente que aprende.

Figure 4.2 Modelo general para agentes que aprenden [[RN04](#biblio-102)].

## 4.2 Formas de Aprendizaje

Los algoritmos de aprendizaje automático se pueden agrupar según la forma en que se realiza el aprendizaje, de acuerdo a la información que poseen o que pueden llegar a poseer. Para entender la naturaleza del problema es vital conocer el tipo de retroalimentación que se dispone para el aprendizaje. Hay tres tipos distintos de aprendizaje: supervisado, no supervisado y por refuerzo [[RN04](" \l "biblio-102)].

### 4.2.1 Aprendizaje supervisado

Los agentes inteligentes que implementan algoritmos de aprendizaje supervisado (*Supervised Learning*), tienen como objetivo aprender una función de hipótesis *h* que se aproxime a la función verdadera *f* y que es solución al problema que se intenta resolver. Lo que se conoce de *f* son los llamados “ejemplos”, que son puntos concretos dentro de la desconocida línea que representa la función verdadera *f*. En otras palabras, la definición matemática de la función *f* no se conoce, solamente se conoce un conjunto de ejemplos definidos por valores correctos de entradas y salidas de la función *f*.

Otra forma de verlo es de la siguiente manera, se dispone de variables de entrada (*X*) y una variable de salida (*Y*), mediante un algoritmo y a partir de un conjunto de datos de entrenamiento se busca aprender una función *Y* = *f*(*X*) que mapee la salida desde la entrada. Los datos de entrenamiento constituyen las respuestas correctas conocidas o “ejemplos”. El objetivo esencial es aproximar la función de mapeo lo mejor que se pueda, de manera que cuando se tengan nuevos datos de entrada (*X*) se pueda predecir la variable de salida (*Y*) [[CCCC08](" \l "biblio-17)].

Problemas del tipo clasificación o regresión se resuelven típicamente con aprendizaje supervisado. Un problema de clasificación es cuando la variable de salida es una categoría y un problema de regresión es cuando la variable de salida es un valor real [[JWHT14](" \l "biblio-46)]. Como ejemplo de aplicaciones que implementan esta técnica están los vehículos autoconducidos, que deben aprender a diferenciar una calle de la que no es (salida booleana: es calle o no es calle?), también debe aprender a frenar (salida booleana: frenar o no frenar?), etc. Entre los algoritmos que implementan aprendizaje supervisado están: Linear Regression, Decision Trees, Support Vector Machines (SVM[↓](#nom-svm)), Bayesian Networks, etc [[Kot07](#biblio-62)]. El problema de estudio de la tesis utiliza algoritmos de aprendizaje supervisad.

### 4.2.2 Aprendizaje no supervisado

En los problemas de aprendizaje no supervisado (*Unsupervised Learning*) los algoritmos reciben como entrada datos de entrenamiento (ejemplos) que no tienen respuestas correctas conocidas. Solo se dispone de variables de entrada (*X*) pero no hay correspondiente variable de salida (*Y*). Los algoritmos buscan estructuras presentes y como resultado pueden extraer reglas generales, o reducir sistemáticamente la redundancia, u organizar los datos por similitud. Los problemas de agrupación (clustering) y asociación (association) se resuelven normalmente con algoritmos de aprendizaje no supervisados.

En empresas retail buscan descubrir grupos de clientes que tienen el mismo comportamiento en las compras (los que toman cerveza por ejemplo), lo que representa una situación de agrupación inherente en los datos. Cuando buscan descubrir reglas que se deduzcan del conjunto de datos (por ejemplo, los que toman vino compran agua tónica o mineral), se tiene una situación de asociación inherente en los datos. Entre las aplicaciones que implementan esta técnica está el reconocimiento de dígitos escritos a mano. Entre los algoritmos de aprendizaje no supervisado están: k-Means, Principal Component Analysis (PCA[↓](" \l "nom-pca)), reducción de la dimensionalidad, etc [[Kot07](" \l "biblio-62)].

### 4.2.3 Aprendizaje por refuerzo

El aprendizaje por refuerzo (*Reinforcement Learning*) es el más general entre las tres categorías. En vez de que un instructor indique al agente qué hacer, el agente inteligente debe aprender cómo se comporta el entorno mediante recompensas (*refuerzos*) o castigos, derivados del éxito o del fracaso respectivamente. El objetivo principal es aprender la función de valor que le ayude al agente inteligente a maximizar la señal de recompensa y así optimizar sus políticas de modo a comprender el comportamiento del entorno y a tomar buenas decisiones para el logro de sus objetivos formales.

Los principales algoritmos de aprendizaje por refuerzo se desarrollan dentro de los métodos de resolución de problemas de decisión finitos de Markov, que incorporan las ecuaciones de Bellman y las funciones de valor. Los tres métodos principales son: la Programación Dinámica (*Dynamic Programming* *o DP*[↓](" \l "nom-dp)), los métodos de Monte Carlo y el aprendizaje de Diferencias Temporales (*Temporal-Difference Learning o TD*[↓](" \l "nom-td)) [[SB98](#biblio-90)].

Entre las implementaciones desarrolladas está AlphaGo, un programa de IA desarrollado por Google DeepMind para jugar el juego de mesa Go. En marzo de 2016 AlphaGo le ganó una partida al jugador profesional Lee Se-Dol que tiene la categoría noveno dan y 18 títulos mundiales. Entre los algoritmos que utiliza se encuentra el árbol de búsqueda Monte Carlo, también utiliza aprendizaje profundo con redes neuronales. Puede ver lo ocurrido en el documental de Netflix “AlphaGo” https://www.netflix.com/title/80190844.

## 4.3 Algoritmos de aprendizaje supervisado y no supervisado

El Ph.D. Jason Brownlee es un especialista en aprendizaje automático, desarrollador, escritor y empresario. Ha trabajado en sistemas de aprendizaje automático para la defensa, startups y pronósticos meteorológicos. Tiene una comunidad “Machine Learning Mastery” [[Bro??](#biblio-48)], la cual empezó porque le apasiona ayudar a los desarrolladores profesionales a comenzar y aplicar con confianza Machine Learning que les permita resolver problemas complejos. Los algoritmos de aprendizaje automático se pueden agrupar según la similaridad en términos de su forma o función, como por ejemplo los métodos basados en árboles y los métodos inspirados en redes neuronales. Se muestra en la Figura [4.3↓](#fig:Algoritmos-de-machine) lo propuesto por Brownlee [[Bro??](" \l "biblio-48)][[Bro16](" \l "biblio-13)].

Figure 4.3 Agrupación de algoritmos según Jason Brownlee [[Bro13](" \l "biblio-47)].

No hay un consenso general de cómo agrupar los algoritmos de Machine Learning en términos de su función o de cómo trabajan. La Figura [4.3↑](#fig:Algoritmos-de-machine) muestra un método útil de agrupación, que no es perfecto y ni exhaustivo en los grupos y algoritmos propuesto por Brownlee.

### 4.3.1 Algoritmos de regresión

Los Algoritmos de Regresión (*Regression Algorithms*) reciben como datos de entrada un conjunto de datos llamados instancias o ejemplos. Cada instancia puede verse como un registro de base de datos, donde uno de los campos es la variable dependiente y las demás variables independientes. Estos algoritmos buscan la función que defina lo más adecuadamente la relación existente entre la variable dependiente y las variables independientes. Si la variable dependiente es un valor discreto, entonces se trata de un problema de clasificación, si es un valor continuo se trata de un problema de regresión [[Iko12](" \l "biblio-43)]. Estos algoritmos se implementan en problemas de aprendizaje supervisado.

### 4.3.2 Algoritmos basados en instancia

El modelo de aprendizaje Basados en Instancia (*Instance-Based Learning*) se basa en un conjunto de instancias que pertenecen a la descripción del concepto y que son almacenadas y actualizadas constantemente. A la descripción de concepto se suman dos funciones principales: la función de similitud y la función de clasificación. Por cada instancia de entrenamiento la función de similitud calcula la similitud que tiene con las instancias de la descripción del concepto. La función de clasificación realiza la predicción basada en la función de similitud y en ciertas propiedades de la descripción del concepto [[AKA91](" \l "biblio-18)]. Estos algoritmos se implementan en problemas de aprendizaje supervisado.

### 4.3.3 Algoritmos de regularización

Los Algoritmos de Regularización (*Regularization*) ayudan a evitar modelos sobreajustados agregando penalizaciones a los parámetros. Entre los métodos más destacados se encuentran: regularización L1 (regularización LASSO[↓](" \l "nom-lasso)), regularización L2 (regularización de Tíjonov) y una versión combinada de ambos [[DKAK11](" \l "biblio-22)]. Estos algoritmos se implementan en problemas de aprendizaje supervisado y no supervisado.

### 4.3.4 Algoritmos de árbol de decisiones

Los métodos de Árbol de Decisiones (*Decision Tree*) permiten *“Aproximar funciones objetivo de valores discretos, en el que la función aprendida se representa mediante un árbol de decisiones”* [[Mit97](#biblio-78)]. Estos algoritmos se implementan en problemas de aprendizaje supervisado.

### 4.3.5 Algoritmos bayesianos

Los métodos Bayesianos (*Bayesian*) modelan el aprendizaje en base al teorema de Bayes. Se considera una técnica indicada para problemas donde es obligatorio tener en cuenta la incertidumbre [[Bar12](" \l "biblio-9)]. Estos algoritmos se implementan en problemas de aprendizaje supervisado.

### 4.3.6 Algoritmos de agrupación

Los métodos de Agrupación (*Clustering*) son técnicas que persiguen el objetivo de formar subconjuntos homogéneos a partir de un conjunto principal de datos y en base a determinadas características [[JMF99](" \l "biblio-45)]. Estos algoritmos se implementan en problemas de aprendizaje no supervisado.

### 4.3.7 Algoritmos de reglas de asociación

Los algoritmos basados en Reglas de Asociación (*Association Rules*) construyen un conjunto de reglas del tipo *si* ⇒ *entonces*. Una regla{*chorizo*, *costilla*} ⇒ {*carbón*} podría indicar que los clientes que compran chorizos y costilla de vaca eventualmente compran carbón para hacer asado. [[BR18](" \l "biblio-11)]. Estos algoritmos se implementan en problemas de aprendizaje no supervisado.

### 4.3.8 Algoritmos de redes neurales artificiales

Las Redes Neuronales artificiales (*Neural Networks*) son modelos basados en la abstracción de una red de neuronas del cerebro humano. *“El cerebro es una computadora altamente compleja, no lineal y paralela (sistema de procesamiento de información)”* [[Hay11](" \l "biblio-40)]. Estos algoritmos se implementan en problemas de aprendizaje supervisado, no supervisado y por refuerzo.

### 4.3.9 Algoritmos de aprendizaje profundo

Los algoritmos de Aprendizaje Profundo (*Deep Learning*) se construyen sobre Redes Neuronales con muchas capas ocultas [[GBC16](" \l "biblio-37)]. Para Quoc V. Le [[Le15](" \l "biblio-67)] son algoritmos que utilizan redes neuronales como arquitectura y que además son capaces de aprender automáticamente las características. Estos algoritmos se implementan en problemas de aprendizaje supervisado y no supervisado.

### 4.3.10 Algoritmos de reducción de dimensionalidad

La Reducción de la Dimensionalidad (*Dimensionality Reduction*) es aplicado a problemas donde se tiene una enorme cantidad de características en el modelo, de los cuales no todos son relevantes. Estos métodos proveen una manera eficiente de encontrar un grupo reducido de dimensiones determinantes [[SVP14](" \l "biblio-100)]. Estos algoritmos se implementan en problemas de aprendizaje no supervisado.

### 4.3.11 Algoritmos combinados

Los métodos combinados (*Ensemble*) se basan en la idea de que varios clasificadores participen en el proceso de predicción sobre nuevos datos [[Sew07](" \l "biblio-97)]. Estos algoritmos se implementan en problemas de aprendizaje supervisado y no supervisado.

### 4.3.12 Algoritmos máquinas de soporte vectorial

Las máquinas de vectores de soporte (Support Vector Machines o SVM[↓](" \l "nom-svm)) son una serie de potentes algoritmos que resuelven problemas de regresión y clasificación, se basan en la construcción de hiperplanos n-dimensionales [[Wan05](" \l "biblio-106)]. Estos algoritmos se implementan en problemas de aprendizaje supervisado.

## 4.4 Algoritmos de aprendizaje por refuerzo

Se puede complementar el listado propuesto por el Dr. Jason con algunos de los algoritmos de aprendizaje por refuerzo.

### 4.4.1 Programación dinámica

Estos son los principales algoritmos [[BBSE10](" \l "biblio-14)]:

* Programación dinámica con Policy Iteration.
* Programación dinámica con Value Iteration.
* Programación dinámica con Policy search

### 4.4.2 Método de Monte Carlo

Tiene como principales algoritmos [[HS16](" \l "biblio-39)]:

* On-Policy Monte Carlo Control.
* Off-Policy Monte Carlo Control.

### 4.4.3 Aprendizaje por Diferencias Temporales

Sus principales algoritmos son [[SB98](" \l "biblio-90)]:

* Sarsa: On-Policy TD Control.
* Q-learning: Off-Policy TD Control.
* Actor-Critic Methods.
* R-Learning for Undiscounted Continual Tasks.

## 4.5 Problemas de clasificación

### 4.5.1 Clasificación binaria

La formulación se resume en, dado una variable de entrada *x* perteneciete al dominio *X*, se debe estimar el valor de la variable aleatoria binaria de salida *y* ∈ {0, 1} [[KNRD14](" \l "biblio-63)]. Entre aplicaciones comunes que implementan clasificación binaria están: deteminar si un correo electrónico es spam o no, detección de fraude, diagóstico médico, etc.

### 4.5.2 Clasificación multiclase

A diferencia de la clasificación binaria la variable de salida puede tomar muchos valores, *y* ∈ {1, 2, 3.., *N*}[[Aly05](" \l "biblio-79)]. El problema de estudio en cuestión utiliza clasificación multiclase, donde *y* ∈ {*Nada*, *Poco*, *Medio*, *Mucho*}. Un típico problema es el de detección automática del idioma de una documento [[SV08](" \l "biblio-5)]. Para desarrollar un modelo o esquema de Machine Learning para resolver problemas de clasificación multiclase, es necesario conocer los componentes esenciales que la forman [[FHW11](#biblio-27)]:

#### Ejemplos o instancias

La entrada de un esquema de aprendizaje automático es un conjunto de instancias. Estas instancias son las cosas que deben ser clasificadas, asociadas o agrupadas. En el escenario estándar, cada instancia es un ejemplo individual e independiente del concepto que se debe aprender.

#### Características o atributos

Las instancias son caracterizadas mediante los valores de un conjunto predeterminado de atributos. Cada instancia proporciona una entrada al aprendizaje automático y es caracterizado por los valores de un conjunto fijo y predefinido de características o atributos.

#### Etiquetas

Las cantidades nominales tienen valores que son símbolos distintos. Los valores mismos sirven como etiquetas o nombres, de ahí el término nominal, que viene de la palabra latina nombre. Los atributos nominales a veces se llaman categorizados, enumerados o discretos.

#### Conjunto de entrenamiento

El grupo de ejemplos utilizados en el proceso de entrenamiento de los algoritmos de aprendizaje automático constituyen el conjunto de entrenamiento.

#### Algoritmos de clasificación multiclase

Constituye el conjunto de algoritmos de machine learning que soportan problemas de clasificación multiclase. Cada algoritmo se construye con su Función Objetivo (f), Variables de entrada (X), Variable de salida (Y), donde Y = f(X) [[JDEHSZ12](" \l "biblio-51)].

#### Conjunto de prueba

Para medir el rendimiento de un clasificador sobre nuevos datos, necesitamos evaluar su tasa de error en un conjunto de datos que no desempeñó ningún papel en la formación del clasificador. Este conjunto de datos independiente se denomina conjunto de prueba.

## 4.6 Algoritmos de clasificación en Weka

Weka[↓](" \l "nom-weka) es una colección de algoritmos de aprendizaje automático para tareas de minería de datos. Los algoritmos pueden ser aplicados directamente a un conjunto de datos o llamados desde código Java. Weka contiene herramientas para pre-procesamiento de datos, clasificación, regresión, clustering, reglas de asociación y visualización. También es adecuado para desarrollar nuevos esquemas de aprendizaje automático [[Wai??](" \l "biblio-72)]. En el problema de estudio se utiliza el conjunto de algoritmos de clasificación de Weka [[FHWP16](" \l "biblio-26)]. Los algoritmos de clasificación de Weka que se utilizarán son los siguientes [[htt??](" \l "biblio-115)]:

|  |  |  |
| --- | --- | --- |
| Clasificadores Weka | | |
| ***Categoría del clasificador*** | ***Nombre del clasificador*** | ***Modelo, técnica o algoritmo que implementa*** |
| Clasificadores bayesianos | BayesNet | Bayes Network (Red Bayesiana) [[BFHKRSS16](" \l "biblio-89)] |
| Clasificadores bayesianos | NaiveBayes | Naive Bayes [[JL95](" \l "biblio-36)] |
| Clasificadores bayesianos | NaiveBayesMultinomial | Naive Bayes multinomial [[MN98](" \l "biblio-6)] |
| Clasificadores bayesianos | NaiveBayesMultinomialUpdateable | Naive Bayes multinomial actualizable [[MN98](#biblio-6)] |
| Clasificadores bayesianos | NaiveBayesUpdateable | Naive Bayes actualizable [[JL95](#biblio-36)] |
| Basado en funciones | Logistic | Regresión Logística [[lv92](" \l "biblio-116)] |
| Basado en funciones | MultilayerPerceptron | Red Neuronal con *back propagation* |
| Basado en funciones | SimpleLogistic | Regresión Logística lineal con LogitBoost [[LHF05](" \l "biblio-83)] [[SFH05](" \l "biblio-73)] |
| Basado en funciones | SMO[↓](" \l "nom-smo) | Sequential Minimal Optimization con Support Vector [[Pla98](" \l "biblio-55)] [[KSBM01](" \l "biblio-95)] [[HT98](" \l "biblio-105)] |
| Clasificadores perezosos | IBk | K-nearest neighbours (K vecinos más cercanos) [[AKA91](#biblio-18)] |
| Clasificadores perezosos | KStar | K\* con función de distancia basada en entropía [[CT95](" \l "biblio-52)] |
| Clasificadores perezosos | LWL[↓](" \l "nom-lwl) | Locally Weighted Learning (Aprendizaje Ponderado Localmente) [[FHP03](" \l "biblio-25)] [[AMS96](" \l "biblio-16)] |
| Meta algoritmos | AdaBoostM1 | Adaboost M1 [[FS96](" \l "biblio-111)] |
| Meta algoritmos | AttributeSelectedClassifier | Selección de atributos |
| Meta algoritmos | Bagging | Bagging [[Bre96](" \l "biblio-68)] |
| Meta algoritmos | ClassificationViaRegression | Métodos de regresión [[FWIHW98](" \l "biblio-28)] |
| Meta algoritmos | CVParameterSelection | Selección de parámetros [[Koh95](" \l "biblio-93)] |
| Meta algoritmos | FilteredClassifier | Filtro arbitrario |
| Meta algoritmos | IterativeClassifierOptimizer | Optimización del número de iteraciones |
| Meta algoritmos | LogitBoost | Regresión Logística aditiva [[FHT98](" \l "biblio-50)] |
| Meta algoritmos | MultiClassClassifier | Metaclasificador |
| Meta algoritmos | MultiClassClassifierUpdateable | Metaclasificador actualizable |
| Meta algoritmos | MultiScheme | Selección del clasificador |
| Meta algoritmos | RandomCommittee | Conjunto aleatorizado de clasificadores base |
| Meta algoritmos | RandomizableFilteredClassifier | Clasificador arbitrario con filtro arbitrario |
| Meta algoritmos | RandomSubSpace | Árbol de decisión [[Ho98](" \l "biblio-104)] |
| Meta algoritmos | Stacking | Combinación de clasificadores utilizando apilamiento [[Wol92](" \l "biblio-20)] |
| Meta algoritmos | Vote | Combinación de clasificadores [[Kun04](" \l "biblio-71)] [[KHDM98](" \l "biblio-56)] |
| Meta algoritmos | WeightedInstancesHandlerWrapper | Soporte de instancias ponderadas |
| Sistema de reglas | DecisionTable | Tabla de decisión simple [[Koh95](" \l "biblio-92)] |
| Sistema de reglas | JRip | “Repeated Incremental Pruning to Produce Error Reduction” (RIPPER) [[Coh95](" \l "biblio-108)] |
| Sistema de reglas | OneR | Clasificador 1R [[Hol93](" \l "biblio-91)] |
| Sistema de reglas | PART | Divide y vencerás para construir un árbol de decisión C4.5 parcial [[FW98](" \l "biblio-24)] |
| Sistema de reglas | ZeroR | Clasificador 0-R |
| Árboles de decisión | DecisionStump | Decision stump in conjunction with a boosting algorithm |
| Árboles de decisión | HoeffdingTree | Algoritmo de inducción incremental del árbol de decisión [[HSD01](" \l "biblio-34)] |
| Árboles de decisión | J48 | Árbol de decisión C4.5 podado o no podado [[Qui93](" \l "biblio-94)] |
| Árboles de decisión | LMT[↓](" \l "nom-lmt) | “Árboles de Modelos Logísticos” o “Logistic Model Trees” (LMT) [[LHF05](#biblio-83)] [[SFH05](#biblio-73)] |
| Árboles de decisión | RandomForest | “Bosque de Árboles Aleatorios” o “Forest of Random Trees” [[Bre01](" \l "biblio-69)] |
| Árboles de decisión | RandomTree | Considera K atributos elegidos al azar en cada nodo. No realiza poda. |
| Árboles de decisión | REPTree | Construye un árbol de decisión/regresión utilizando la información de ganancia/varianza |

## 4.7 Técnicas de evaluación para aprendizaje supervisado

La evaluación es la clave para lograr avances reales en el aprendizaje automático. Entre las técnicas de evaluación para problemas de aprendizaje supervisado se destacan: hold-out validation y k-fold cross-validation

### 4.7.1 Método de retención

La técnica de Retención (*Hold-out method*) divide el conjunto de datos en dos subconjuntos disjuntos: uno para el entrenamiento (generalmente el 70%) del que aprende el modelo y otro para la prueba (el otro 30%) donde se evalúa el modelo. Esta técnica puede ayudar a evitar el sobreajuste ya que el conjunto de prueba es un conjunto que no desempaña ninguna función en el entrenamiento, contribuyendo así a una evaluación objetiva de cómo se comportaría el modelo sobre nuevos datos [[Ras18](" \l "biblio-87)][[LZ09](#biblio-70)].

### 4.7.2 Validación cruzada de k iteraciones

La técnica de Validación Cruzada de k iteraciones (k-fold Cross-Validation) consiste en dividir los datos en un número de particiones iguales. En la figura [4.4↓](#fig:Esq-general-de-Cross) se muestra un ejemplo de una validación cruzada de 3 iteraciones donde se divide el conjunto de datos en tres particiones de igual tamaño. Luego se entrena el modelo en dos de esas tres particiones y se prueba el modelo en la partición restante. Luego, se repite el proceso, seleccionando una partición diferente para que sea el grupo de prueba y entrenando un nuevo modelo en el conjunto restante de dos particiones. El proceso se repite una vez más, para un total de tres rondas de validación cruzada, una por cada pasada [[LZ09](#biblio-70)].

Finalmente se tienen tres modelos diferentes, cada uno entrenado y testeado en subconjuntos de datos diferentes, se combinan estos modelos promediando sus pesos y estimando un modelo predictivo resultante. La validación cruzada también ayuda a evitar el sobreajuste.

Figure 4.4 Esquema general de iteraciones para 3-fold Cross-Validation [[LZ09](" \l "biblio-70)]

La técnica comúnmente utilizada es Stratified k-fold Cross-Validation, una pequeña variante donde la estratificación se refiere al proceso de reorganizar los datos de manera a que cada partición sea una representación del conjunto. Se acepta que 10 es el número de particiones con el que se obtiene una buena estimación de error, idea sustentada en diversas pruebas sobre diferentes conjuntos de datos y técnicas de aprendizaje [[FHW11](#biblio-27)].

## 4.8 Métricas de desempeño para problemas de clasificación

Las métricas de clasificación son las medidas contra las cuales se evalúan los modelos de clasificación. Para problemas de clasificación es común medir el rendimiento de un clasificador en términos de la tasa de error (*error rate*) o tasa de acierto (*sucess rate*). El clasificador predice la clase de cada instancia: si es correcta se cuenta como un éxito, sino se cuenta como un error. La tasa de error es sólo la proporción de errores cometidos sobre un conjunto de instancias y mide el rendimiento general del clasificador. Por supuesto, lo que interesa es el probable desempeño futuro en nuevos datos, no el rendimiento sobre datos pasados.

Para predecir el rendimiento de un clasificador sobre nuevos datos, necesitamos evaluar su tasa de error en un conjunto de datos que no tuvo ninguna labor en la formación del clasificador. Este conjunto de datos independiente se denomina conjunto de prueba. Se suele hablar de tres conjuntos de datos: datos de entrenamiento, datos de validación y datos de prueba [[FHW11](" \l "biblio-27)]. Los datos de entrenamiento son utilizados por uno o más esquemas de aprendizaje para conocer clasificadores.

Los datos de validación se utilizan para optimizar los parámetros de los clasificadores, o para seleccionar uno determinado. Luego, los datos de prueba se utilizan para calcular la tasa de error del método final optimizado. Cada uno de los tres conjuntos debe ser independiente: El conjunto de validación debe ser diferente del conjunto de entrenamiento para obtener un buen desempeño en la etapa de optimización o selección y el conjunto de prueba debe ser diferente de ambos para obtener una estimación confiable de la tasa de error real.

Para fines prácticos generalmente se utiliza la metodología de dividir los datos en dos conjuntos, el conjunto de entrenamiento y el conjunto de prueba, como es el caso del trabajo actual donde se utilizan valores por defecto en los parámetros de los clasificadores.

### 4.8.1 Porcentaje de acierto

Es el porcentaje de instancias correctamente clasificadas o de predicciones correctas. La métrica más simple y más común es el porcentaje de acierto. Se calcula dividiendo el número de predicciones correctas sobre el número total de predicciones realizadas, multiplicado por 100. Por ejemplo, en un modelo de clasificación supervisada para transacciones bancarias, si probamos el modelo en cien transacciones y se predice correctamente la etiqueta (fraude/no fraude) para noventa de ellas, entonces el porcentaje de acierto del modelo es del 90%.

**(5.1)** *Porcentaje* *de* *acierto* = (*Número* *de* *predicciones* *correctas*)/(*Número* *total* *de* *predicciones* *realizadas*) × 100

### 4.8.2 Matriz de confusión

El porcentaje de acierto es la métrica de clasificación más fácil, simple y entendible que se puede utilizar. Pero la desventaja es que no muestra la distribución de los valores de respuesta. No hay distinción entre falsos positivos, transacciones incorrectamente clasificadas como fraude y falsos negativos, transacciones incorrectamente clasificadas como no fraudulentas. Es por eso que se necesita la matriz de confusión.

Una matriz de confusión para una clasificación binaria es una tabla de 2 x 2 como se muestra en la figura [4.5↓](#fig:matriz-confusion) que clasifica las predicciones en una de cuatro clasificaciones: verdadero positivo (*VP*[↓](" \l "nom-vp)), verdadero negativo (*VN*[↓](" \l "nom-vn)), falso positivo (*FP*[↓](" \l "nom-fp)) y falso negativo (*FN*[↓](" \l "nom-fn)).

En problemas de clasificación de implementaciones reales los errores tienen diferentes costos. Por ejemplo en bancos y financieras el costo de prestar a una persona que no paga sus deudas es mayor que el costo de rechazar un préstamo a una persona que es pagadora. Los Verdaderos Positivos y Verdaderos Negativos son clasificaciones correctas. Un Falso Positivo es cuando el resultado se predice incorrectamente como positivo cuando es realmente negativo. Un Falso Negativo es cuando el resultado se predice incorrectamente como negativo cuando es realmente positivo.

Figure 4.5 Matriz de confusión para clasificación binaria.

En la figura [4.6↓](#fig:matriz-confusion-1) se muestra una generalización de la matriz de confusión para clasificación multiclase.

Figure 4.6 Matriz de confusión para clasificación multiclase.

La matriz de confusión ofrece una visión más completa de cómo está funcionando el clasificador. También permite calcular varias métricas de clasificación que pueden orientar la elección del modelo. La selección de la métrica depende en gran medida del objetivo de la aplicación. Se debe identificar si FP o FN es más crucial reducir y luego elegir la métrica (FP o FN en la ecuación). Entre las métricas calculadas a partir de una matriz de confusión se encuentran [[Zhe15](" \l "biblio-113)]:

#### 4.8.2.1 Precisión (accuracy)

Una medida general de la tasa de acierto del clasificador.

**(5.2)** *Precisión* = (*VP* + *VN*)/(*VP* + *VN* + *FP* + *FN*)

La generalización de la fórmula para clasificación multiclase es la siguiente, donde n es el número total de instancias evaluadas y N el número total de clases.

**(5.3)** *Precisión* = (1)/(*n*)∑*i*= 1*Nfii*

#### 4.8.2.2 Exactitud (precision)

Una medida de la tasa de acierto cuando la predicción es un valor positivo. ¿Cuán exacto es para predecir instancias positivas?.

**(5.4)** *Exactitud* = (*VP*)/(*VP* + *FP*)

La generalización de la fórmula para clasificación multiclase es la siguiente, donde n es el número total de instancias evaluadas y N el número total de clases.

*Exactitudi* = (*fii*)/(∑*j*= 1*Nfji*)

#### 4.8.2.3 Sensibilidad (sensitivity)

Una medida de la tasa de acierto cuando el valor real es positivo. ¿Cuán sensible es para detectar instancias positivas?. También es conocido como True Positive Rate o Recall.

**(5.5)** *Sensibilidad* = (*VP*)/(*VP* + *FN*)

La generalización de la fórmula para clasificación multiclase es la siguiente, donde n es el número total de instancias evaluadas y N el número total de clases.

*Sensibilidadi* = (*fii*)/(∑*j*= 1*Nfij*)

#### 4.8.2.4 Especificidad (specificity)

Una medida de la tasa de acierto cuando el valor real es negativo. ¿Cuán específico es para detectar instancias negativas?.

**(5.6)** *Especificidad* = (*VN*)/(*VN* + *FP*)

#### 4.8.2.5 Medida F1 (f1-measure)

La Medida F1 es un promedio ponderado de la precisión y sensibilidad, donde alcanza su mejor valor en 1 y el peor en 0.

**(5.7)** *Medida* *F*1 = (2 × *Sensibilidad* × *Precision*)/((*Sensibilidad* + *Precision*))

La generalización de la fórmula para clasificación multiclase es la siguiente:

**(5.8)** *Medida* *F*1(*i*) = (2 × *Sensibilidadi* × *Precisioni*)/((*Sensibilidadi* + *Precisioni*))

#### 4.8.2.6 Estadística kappa (kappa Statistic)

En la clasificación multiclase cada elemento de la matriz de confusión muestra el número de ejemplos de prueba y donde la clase real es la fila y la clase predicha es la columna. El mejor resultado se logra cuando todos los números están en la diagonal principal y cero fuera de ella. El coeficiente kappa de Cohen (*κ*) [[Coh60](" \l "biblio-44)] es una estadística que mide el acuerdo entre examinadores sobre ítems cualitativos. Si los examinadores están completamente de acuerdo Kappa es igual a 1. Si no hay total acuerdo entre los observadores Kappa tiene un valor < 1.

**(5.9)** *κ* = (*Pr*(*a*) − *Pr*(*e*))/(1 − *Pr*(*e*))

Donde: *Pr*(*a*)es el acuerdo relativo observado entre examinadores o el número de instancias que fueron clasificadas correctamente sobre el total de instancias, *Pr*(*e*) es la probabilidad de que el acuerdo entre examinadores se deba al azar y está relacionada con el número de instancias de cada clase y el número de instancias que el clasificador concuerda con el valor verdadero de la clase. El resultado del numerador da el grado de acuerdo realmente logrado por encima del azar y el del denominador da el grado de acuerdo que es alcanzable por encima del azar.

La generalización de la fórmula para clasificación multiclase es la siguiente, donde n es el número total de instancias evaluadas, N el número total de clases.

**(5.10)** *Pr*(*a*) = (1)/(*n*)∑*i*= 1*Nfii*

**(5.11)** *Pr*(*e*) = ∑*k*= 1*N*[(1)/(*n*2)(∑*i*= 1*Nfik* × ∑*j*= 1*Nfkj*)]

Otra forma común de interpretar los resultado es la siguiente:

* < 0: Acuerdo deficiente
* 0.01 - 0.20: Acuerdo leve
* 0.21 - 0.40: Acuerdo justo
* 0.41 - 0.60: Acuerdo moderado
* 0.61 - 0.80: Acuerdo sustancial
* 0.81 - 1.00: Acuerdo casi perfecto

## 4.9 Componentes de los errores de predicción: Ruido, Sesgo y Varianza

Los errores de predicción en los modelos de aprendizaje automático tienen tres componentes: Ruido (*Noise*), Sesgo (*Bias*) y Varianza (*Variance*) [[Luy09](" \l "biblio-61)].

### 4.9.1 Ruido

El ruido es considerado un error aleatorio de la variable que está siendo medida [[HPK11](" \l "biblio-38)] y representa el componente fuera de control del error de generalización en una función. Como se trata de una alteración de los datos originales no puede ser reducido. Por ejemplo, en el problema de estudio se encontraron malas entradas de datos donde en algunos casos el valor del costo de un producto era mayor al precio de venta. Otro ejemplo, en un sistema de monitoreo ambiental puede haber ruido proveniente del mal funcionamiento o calibraciones defectuosas en los sensores de medición [[KACN03](" \l "biblio-60)].

### 4.9.2 Sesgo

La precisión de la hipótesis aprendida a través del entrenamiento es una mala estimación del rendimiento sobre futuros ejemplos, ya que la hipótesis se derivó de estos ejemplos y proporcionará una estimación optimista sesgada lo que se conoce como sesgo en la estimación [[Mit97](#biblio-78)]. El sesgo conocido también como “subajuste”, ocurre cuando la función de aprendizaje mapea pobremente los datos de entrenamiento y esto da un error de generalización con datos futuros. En estadística se entiende por sesgo como la diferencia al cuadrado entre el verdadero valor y el valor esperado de la estimación [[FHT01](#biblio-32)].

El sesgo define la capacidad de ajuste del modelo de la función de aprendizaje, en la figura [4.7↓](#fig:Errores-en-la-prediccion) se observa que a menor complejidad, mayor sesgo, esto indica que el error de predicción tanto en el entrenamiento como en el testeo es mayor cuando el modelo seleccionado es mas simple [[DB95](" \l "biblio-23)].

### 4.9.3 Varianza

Cuanto menor es el conjunto de ejemplos de prueba, mayor es la varianza esperada [[Mit97](" \l "biblio-78)]. La varianza llamada también “sobreajuste” se da cuando la función seleccionada aprende casi perfectamente la tendencia de los datos del ejemplo de entrenamiento, pero falla en la generalización con datos nuevos del futuro.

La varianza es una medida de generalización, tanto el sesgo como la varianza están relacionadas con la complejidad del modelo de la función, la figura [4.7↓](#fig:Errores-en-la-prediccion) muestra que a medida que aumenta la complejidad del modelo, la varianza tiende a aumentar y el sesgo tiende a disminuir [[FHT01](#biblio-32)]. En Machine Learning es esencial encontrar modelos con un balance entre sesgo y varianza, o visto de otra forma, un equilibrio entre error y complejidad.

## 4.10 Complejidad del modelo y curvas de aprendizaje

La capacidad de generalización está determinada principalmente por dos factores: la complejidad del modelo (model complexity) [[FHT01](#biblio-32)] y el tamaño del conjunto de entrenamiento (learning curves) [[ZP16](" \l "biblio-114)].

La figura [4.7↓](#fig:Errores-en-la-prediccion) muestra la relación existente entre la complejidad del modelo (eje x) y el error de predicción (eje y). Indica cómo va variando el error de predicción en los conjuntos de entrenamiento y de prueba a medida que aumenta la complejidad del modelo. Los modelos muy simples dan una situación no deseada de sesgo (subajuste) y los modelos muy complejos igualmente una situación no deseada de varianza (sobreajuste). Para obtener una buena capacidad de generalización se debe encontrar un compromiso entre sesgo y varianza.

Este compromiso entre sesgo y varianza se puede encontrar probando distintas configuraciones de los parámetros de ajuste (tuning parameters) de los algoritmos de aprendizaje y luego eligiendo el que arroje mejores resultados. Normalmente se recomienda probar las distintas configuraciones de los parámetros sobre un conjunto de datos llamado de validación, que no se debe confundir con los conjuntos de entrenamiento y de prueba.

Algunos ejemplos de familias de algoritmos y sus parámetros de ajuste son:

* Algoritmos de regresión lineal: la cantidad de atributos del modelo (que determinará la cantidad de variables independientes).
* Modelos bayesianos: en las redes bayesianas por ejemplo se debe elegir el algoritmo estimador (estimación de las tablas de probabilidad condicional de las redes bayesianas) y el método para encontrar estructuras de red.
* Árboles de decisión: el factor de confianza de la poda, mínimo número de instancias, número de pasadas para reducir el error de poda [[MHCVC16](" \l "biblio-85)].
* Redes neuronales artificiales: la topología de la red donde se debe elegir la cantidad de neuronas para la capa de entrada, salida y la oculta. También se deben probar distintas tasas de aprendizaje (learning rate) y funciones de activación.
* Support vector machines: los parámetros gamma (γ), costo (C), y epsilon (ε). γ es una constante que reduce el espacio modelo y controla la complejidad de la solución, C es una constante positiva que es un parámetro de control de capacidad, mientras que ε es la función de pérdida que describe el vector de regresión sin todos los datos de entrada [[MA16](#biblio-101)].

Figure 4.7 Complejidad del modelo [[FHT01](" \l "biblio-32)]

La figura [4.8↓](#fig:Curvas-de-aprendizaje) muestra la relación existente entre el tamaño del conjunto de entrenamiento (eje x) y el error de predicción (eje y). Indica cómo va variando el error de predicción en los conjuntos de entrenamiento y de prueba a medida que aumenta el tamaño del conjunto de entrenamiento. Los modelos muy simples dan problema de alto sesgo (figura [4.8↓](#fig:Curvas-de-aprendizaje) (a)) y los modelos muy complejos dan problema de alta varianza (figura [4.8↓](#fig:Curvas-de-aprendizaje) (b)).

Cuando el modelo está sesgado (figura [4.8↓](#fig:Curvas-de-aprendizaje) (a)) por mas que se aumenten los datos del conjunto de entrenamiento el nivel de error del conjunto de entrenamiento y el de prueba se mantienen alto, por lo que para modelos sesgados no es una solución aumentar significativamente la cantidad de los datos del conjunto de entrenamiento. La solución se debería buscar aumentando la complejidad del modelo.

Cuando el modelo tiene alta varianza (figura [4.8↓](#fig:Curvas-de-aprendizaje) (b)) el error en el conjunto de entrenamiento se mantiene bajo aún aumentando su cantidad de datos. El conjunto de prueba en cambio inicia con alto porcentaje de error con pocos datos, pero a medida que aumentan los datos en el conjunto de entrenamiento el error del conjunto de prueba disminuye significativamente. Por tanto, ante alta varianza es una posible solución aumentar el tamaño de los datos. También se puede probar disminuyendo la complejidad del modelo.

Figure 4.8 Curvas de aprendizaje

## 4.11 Trabajos relacionados a Pronóstico - Business Intelligence - Machine Learning

En esta sección se pretende dar a conocer un análisis de los trabajos relacionados con los tres puntos más importantes del este trabajo. En los textos revisados cabe destacar que no se ha encontrado un trabajo o estudio que aborda el problema del Pronóstico de la Demanda con Business Intelligence y Machine Learning en la misma solución. Sin embargo, publicaciones que comprenden los últimos años presentan trabajos muy interesantes en el ámbito de Machine Learning asociado al Pronóstico de la Demanda (*Demand Forecasting*), como así también Pronóstico de la demanda en conjunto con Business Intelligence. Se citan a continuación algunos de los problemas afrontados en publicaciones, que dan una idea del estado del arte en este tema:

En el año 2016 [[Cas16](" \l "biblio-59)] se afronta el problema de mejorar el tiempo y la calidad en la toma de decisiones gerenciales en la empresa “Figueri SRL”, el tiempo y el esfuerzo requerido por el área de sistemas para la generación de informes a la gerencia afectan en la toma de decisiones. El enfoque adoptado para la solución consiste en la implementación de un sistema de Business Intelligence basado en un modelo de pronóstico de ventas.

En el desarrollo de Business Intelligence se construye un Data mart bajo la metodología de Kimball para guardar los datos históricos, se realiza el proceso ETL para obtener y cargar los datos al dataware. El desarrollo del pronóstico se basa en el modelo de series de tiempo autorregresivo ARMA de la demanda de productos. Como resultado de la solución propuesta la empresa obtuvo una disminución en el porcentaje de devolución y el porcentaje de rotación de productos, los pronósticos ayudaron a la toma de decisiones, generando mayores ventas y nuevas estrategias de marketing.

En el trabajo presentado en el año 2016 [[MA16](" \l "biblio-101)] los autores abordan cómo pronosticar la demanda de agua urbana en la ciudad de Montreal (Canadá) en plazos de 1 y 3 días. Para llegar a la solución propuesta se compararon cuatro modelos de pronóstico: Artificial Neural Network (ANN), Support Vector Regression (SVR) y Extreme Learning Machine (ELM) y el tradicional Multiple Linear Regression (MLR).

Los modelos se basaron en combinaciones de variables de entrada como la temperatura diaria máxima, precipitación diaria total y la demanda diaria de agua con datos disponibles desde 1999 a 2010. Los dos índices de rendimiento utilizados para evaluar los modelos MLR, ANN, SVR y ELM fueron el Coeficiente de Determinación (*R*2) y el Error Cuadrático Medio (RMSE). ELM alcanzó una mayor precisión en ambos períodos de pronóstico 1 y 3 días, en comparación con MLR, ANN y SVR que alcanzaron buena precisión para el plazo de 1 día. En general, ELM resulta ser un método de aprendizaje eficiente cuando se trata de pronosticar a corto plazo.

En el año 2015 se presentó [[FLS15](" \l "biblio-64)] un caso de estudio basado en la empresa de retailer online *Rue La La* (http://www.ruelala.com). Los autores buscan estrategias para estimar la demanda de estilos nunca antes vendidos y también buscan un algoritmo que optimize combinaciones de precios para que sirva como herramienta de apoyo diario en la toma de decisiones de precios y que maximice los ingresos de dichos primeros estilos de exposición. Los datos de transacciones de ventas abarcaron desde principios de 2011 hasta mediados de 2013, donde cada registro representa una venta con su sello de tiempo, cantidad vendida del artículo, precio, fecha/hora de inicio del evento, duración del evento y el inventario inicial del artículo.

Además, se disponen de datos relacionados al producto como la marca, el tamaño, el color, el precio sugerido por el fabricante (MSRP o Manufacturer’s Suggested Retail Price) y la clasificación jerárquica del producto. Los modelos probados fueron: Least Squares Regression, Principal Components Regression, Partial Least Squares Regression, Multiplicative (power) Regression, Semilogarithmic Regression y Regression Trees. Para comparar estos modelos se dividió aleatoriamente los datos en conjuntos de entrenamiento y prueba, y se utilizó los datos de entrenamiento para construir los modelos de regresión.

Para los modelos que requerían parámetros de ajuste se utilizó validación cruzada de cinco pasadas sobre los datos de entrenamiento. Se encontró que Regression Trees con Bagging superó a los demás métodos de regresión. Luego de poner en producción la herramienta las ventas no disminuyen debido a la implementación de aumentos de precios recomendados por el algoritmo de optimización de precios. Se logró un aumento en los ingresos del grupo de prueba en aproximadamente 9.7%.

El trabajo presentado en el año 2014 [[SLH14](" \l "biblio-112)], plantea como objetivo pronosticar la demanda del petróleo crudo importado (Imported Crude Oil - ICO) en Taiwán. Para el estudio se utilizaron datos reales, desde los años 1993 a 2010, que fueron registrado por la Oficina de Energía de Taiwán. Como solución se propone un modelo híbrido de dos etapas. Para evidenciar la efectividad de los modelos híbridos se lo comparan con modelos de una sola etapa.

Los modelos de pronóstico de una sola etapa incluyen: Multiple Linear Regression (MLR), Support Vector Regression (SVR), Artificial Neural Networks (ANN), y Extreme Learning Machine (ELM), y una selección de 23 variables influyentes como posibles datos de entrada. Para ANN y SVR se utilizaron las 23 variables, para ELM entre 1 y 15 variables, y para MLR se utilizaron 2 variables significativas.

Para los modelos de pronóstico híbridos se utilizaron las 2 variables seleccionadas para MLR como variables de entrada, dando así 3 modelos híbridos: MLR(sel)-ANN, MLR(sel)-SVR y MLR(sel)-ELM. Como medidas de rendimiento de los pronósticos se utilizaron: Mean Absolute Percentage Error (MAPE), Root Mean Square Error (RMSE), y el Mean Absolute Difference (MAD) fueron utilizados. Los resultados mostraron que los enfoques híbridos propuestos son más precisos que los de una sola etapa, por lo tanto son capaces de predecir con mayor precisión la demanda de petróleo crudo en Taiwán.

# 5 Propuesta de solución

## 5.1 Esquema general de solución

En la figura [5.1↓](#fig:5.1.1) se ilustra el esquema general de la solución.

Figure 5.1 Esquema general de solución

En este capítulo se desarrolla el primer componente de la solución propuesta que es Business Intelligence. Para ello, se obtienen los datos de la fuentes de información, se sigue el proceso de ETL y se construye el datawarehouse. Luego se definen y obtienen los valores de los KPI, a continuación se define la estrategia de etiquetado y se asignan las etiquetas a cada tupla de KPI.

En el capítulo siguiente se desarrollan el segundo y tercer componente da la solución propuesta que son: Machine Learning y la Técnica de solución propuesta.

## 5.2 Componente de solución “Business Intelligence”

Esta sección se centra en los tres primeros componentes de Business Intelligence siguiendo el proceso de modelado dimensional[[Kim92](#biblio-86)]. Uno de los principales problemas con los que se enfrentan las empresas retail, trata acerca de pronosticar la demanda, es decir, determinar la cantidad de productos que se deben disponer para satisfacer la demanda de los clientes en el siguiente periodo. Utilizando los conceptos y herramientas de Business Intelligence, se parte con el análisis de una fuente de información auténtica obtenida de una empresa retail; se diseña el datawarehouse que será poblado con información de las transacciones operacionales diarias del negocio.

Posteriormente se definen los indicadores claves de rendimiento como métricas, se calculan los valores de los indicadores con los datos históricos almacenados en el datawarehuose para cada periodo de tiempo establecido y se asigna una etiqueta a cada tupla de valores KPI que finalmente servirán como parámetros de entrada de las herramientas de aprendizaje automático para diseñar una solución que pronostique la demanda.

### 5.2.1 Fuente de Información

Para el presente trabajo, se cuenta con una base de datos relacional Oracle 10g con las operaciones transaccionales de una empresa retail dedicada a la venta de productos alimenticios y artículos de limpieza, algunas de las líneas de productos con que cuenta la empresa son: aceites corporales, acondicionadores, aromatizantes, cuidado corporal, desodorantes, limpiadores, salud e higiene, salud y belleza, aguas, gaseosas, cervezas, vinos, chocolates, galletitas, enlatados, lácteos, yerbas y varias líneas de productos más. La base de datos almacena datos de las operaciones comprendidas entre noviembre de 2013 y octubre de 2016. A continuación una reseña de las principales tablas tenidas en cuenta para el diseño del datawarehouse.

* **Tabla Productos**: almacena datos como descripción, unidad de medida, categoría, tipo de impuesto, línea del producto, marca, código de barras, proveedor, costo, precio de venta, fecha de registro, etc., de los productos disponibles para la venta, la tabla cuenta con 13.200 artículos registrados.
* **Tabla Proveedores**: almacena datos como denominación, dirección, teléfono, ruc, email, página web, ciudad, país, propietario, etc., de los proveedores de la empresa, la tabla cuenta con 1.623 proveedores registrados.
* **Tabla de Ventas Cabecera**: es una de las tablas principales donde se registran los movimientos de ventas de la empresa. Contiene datos como número de factura, moneda, tipo de comprobante, caja, usuario, fecha, cliente, monto total, monto gravado, monto impuesto, monto exenta, etc., la tabla cuenta con 301.316 registros de ventas correspondientes al periodo mencionado previamente.
* **Tabla de Ventas Detalle**: contiene los registros de los productos que fueron comercializados, cada detalle está relacionado a un registro de la tabla venta cabecera. Contiene datos de la fecha, el producto, precio de costo unitario, precio de venta unitario, cantidad, importe grabado, importe del impuesto entre otros datos, la tabla cuenta con 981.402 detalles de ventas registrados.
* **Tabla de Movimientos de Stock:** contiene los registros de movimientos de stock de ventas y compras detallado. Contiene datos de fecha, producto, cantidad, tipo de movimiento, costo unitario, precio unitario, entre otro datos, la tabla cuenta con 1.062.440 movimientos de compra y ventas registradas.

### 5.2.2 Proceso ETL.

En el proceso ETL posterior a la extracción y previa a la carga de los datos al datawarehouse fueron realizadas algunas transformaciones y limpieza sobre los datos de origen.

* **Tabla de Productos**: se detectaron registros de artículos con las siguientes inconsistencias:
  + Datos del proveedor con valores nulos, los cuales eran completados con un proveedor por defecto de la tabla dimensional de proveedores.
  + Artículos con valores de costo nulo, en tales casos los valores eran asignados con un costo promedio tomados de la tabla de Ventas Detalle.
  + Artículos con valores donde el costo eran mayor al precio de venta unitario, en dichos casos los datos fueron completados con el costo promedio tomados de la tabla de Ventas Detalle.
  + Artículos cuyo precio de venta unitario era nulo, en los cuales los datos eran completados con el precio de venta mas reciente tomado de la tabla de Ventas Detalle.
* **Tabla de Ventas Cabecera**: se encontraron registros donde los datos del cliente eran nulos, en dichos casos fueron asignados un cliente por defecto tomados de la tabla dimensional de clientes.
* **Tabla de Ventas Detalle**: se detectaron registros con las siguientes falencias:
  + Registros de detalles donde los valores de costo eran iguales a cero, los cuales eran modificados por el costo promedio de la tabla dimensional de productos.
  + Registros de detalle donde el costo unitario eran mayores al precio de venta unitario, las cuales fueron corregidas con el costo promedio, tomados de la tabla dimensional de productos.

### 5.2.3 Datawarehouse

El datawarehouse se diseña partiendo de la definición de las tablas de hechos y dimensiones utilizando el modelo en esquema estrella [[Kim92](" \l "biblio-86)].

### Tablas de hechos

De las tablas transaccionales se definen 3 tablas de hechos a partir de las cuales se obtienen los valores de los KPI que se utilizan en la solución propuesta.

* **Tabla de hechos Cabecera:** almacena los datos históricos de las ventas, cada registro de la tabla de hechos guarda datos de: fecha, cliente, caja, número de factura y monto total, monto exento, monto gravado IVA. Las métricas definidas para la tabla de hechos son: monto total, monto exento, monto gravado IVA[↓](" \l "nom-iva).
* **Tabla de hechos Detalles:** almacena los datos históricos en detalle por cada producto vendido, cada registro guarda información del número de comprobante, fecha, producto, proveedor, cliente, cantidad, costo unitario, precio unitario, impuesto e importe total. Las métricas asociadas a la tabla de hechos son: cantidad, precio unitario, impuesto, costo unitario y el importe total.
* **Tabla de hechos Stock:** almacena los datos históricos de cada movimiento ya sea compra o de venta realizada, cada registro contiene información como fecha, producto, tipo de movimiento, cantidad, precio unitario y costo unitario. Las métricas definidas para la tabla de hechos son: cantidad, precio unitario y costo unitario.

### Dimensiones

Las tablas de dimensiones diseñadas para el modelado del datawarehouse y que se encuentran relacionadas a las tablas de hechos son:

* **Dimensión Fecha:** La tabla de dimensión fecha está ligada a todas las tablas de hechos, sirve para limitar o agrupar los datos de las tablas de hechos al momento de realizar consultas sobre estas en el tiempo. Con la dimensión fecha se pueden establecer niveles jerárquicos en días, semanas, meses, trimestres, semestres y años.
* **Dimensión Productos:** La tabla de dimensión producto está relacionada a las tablas de hechos Detalles y Stock. Contiene los atributos o campos (ej.: nombre del producto) por las cuales se pueden filtrar o agrupar datos.
* **Dimensión Proveedores:** La tabla de dimensión proveedores está relacionada a la tabla de hechos Detalles. Contiene los atributos o campos (ej.: nombre del proveedor) por las cuales se pueden realizar filtros o agrupaciones de datos.
* **Dimensión Clientes:** La tabla de dimensión Clientes está relacionada a las tablas de hechos Cabecera y Detalles. Contiene los atributos o campos (ej.: nombre del cliente) por las cuales se pueden hacer consultas de datos.
* **Dimensión Cajas:** La tabla de dimensión Cajas está relacionada a la tabla de hechos Cabecera. Contiene los atributos o campos (ej.: número de caja) por las cuales se pueden filtrar o agrupar datos.

En figura [5.2↓](#fig:Hechos_detalles_5.1) observamos un ejemplo del modelo en esquema estrella de la tabla de hechos Detalles.

Figure 5.2 Tabla de hechos Detalles

## 5.3 Definición de los Indicadores Claves de Rendimiento

En el marco de este trabajo, en esta sección se definen los KPIs y se obtienen los valores que serán utilizados como datos de entrada en las herramientas de aprendizaje automático para el desarrollo de una técnica para pronosticar la demanda y que servirá de apoyo en la toma de decisiones para la reposición de stock del siguiente periodo[[Alv13](" \l "biblio-75)] (Ej.: cantidad a comprar para satisfacer la demanda de la siguiente semana, quincena, o mes).

Los KPIs definidos fueron adaptados a la solución planteada debido a que en los textos consultados los KPI engloban a toda la organización en áreas como compras, ventas, marketing, recursos humanos y otros. Cada KPI mide un valor obtenido de los datos históricos almacenados en el datawarehouse. El cálculo de cada KPI se realiza por cada producto y periodo de tiempo (semanal, quincenal o mensual). A continuación se definen los KPIs

**Ticket Medio.**

Es la cantidad media por cada transacción de venta que se realiza de un determinado producto. El indicador viene determinado por dos variables: a) la cantidad total vendida del producto, b) el total de tickets en las que fue vendido el producto. Aplicando la siguiente fórmula obtenemos el valor de la cantidad media de venta para cada producto.

**(6.1)** *TM*  = (*Cantidad* *Vendida* *Periodo*)/(*Total* *Tickets* *Periodo*)

**Cifra de Ventas**

La cifra de ventas es un KPI que sirve para explicar el importe total de ventas que se ha obtenido para un producto. Se obtiene de la siguiente fórmula.

**(6.2)** *CV*  = *Precio* × *Cantidad* *Vendida* *Periodo*

**Margen Comercial**

Es la razón entre el precio de venta y precio de costo del producto, es un indicador que permite conocer el porcentaje de rentabilidad del producto. Se obtiene de la siguiente fórmula.

**(6.3)** *MC*  = ((*Precio* − *Costo*) × *Cantidad* *Vendida* *Periodo*)/(*Precio* × *Cantidad* *Vendida* *Periodo*) × 100

**Rotación de Stock**

Este indicador mide la cantidad de veces que el stock del producto se renueva durante un determinado ciclo comercial, es decir, la cantidad de veces que se recupera la inversión. Se obtiene de la siguiente fórmula.

**(6.4)** *RS*  = (*Cantidad* *Vendida* *Periodo*)/(((*Stock* *Inicial* *Periodo* + *Stock* *Final* *Periodo*)/(2)))

**Coeficiente de Rentabilidad**

El indicador mide la rentabilidad obtenida por la empresa basada en el margen y la rotación, el objetivo de toda empresa retail es aumentar los niveles de rotación. El coeficiente se obtiene de la siguiente fórmula.

**(6.5)** *CR* = ((*Precio* − *Costo*) × *Cantidad* *Vendida* *Periodo*) × *RS*

**Cobertura de Stock**

Este indicador muestra el periodo de tiempo (habitualmente se expresa en días o semanas) que el negocio puede continuar vendiendo con el stock de que dispone en el momento, sin incorporar nuevas cantidades de ese producto.

**(6.6)** *CS*  = (*Stock* *Inicio* *Periodo*)/(*Promedio* *Cantidad* *Vendida* *Ultimos* *N* *Periodos*)

## 5.4 Obtención de los valores para los Indicadores Clave de Rendimiento.

Se obtiene los valores de KPI por cada producto y periodo, mediante codificación de sentencias SQL[↓](" \l "nom-sql) que operan sobre los datos almacenados en el datawarehouse, y todos los resultados obtenidos de la ejecución se almacenan en una tabla base. En dicha tabla base además de los valores de KPI, en cada registro se guarda adicionalmente información de cantidad, fecha, año, mes, quincena, semana y periodo. Durante el cálculo de los valores de los KPI se establecieron ciertas restricciones, como por ejemplo si un producto no fue vendido durante un número consecutivo de periodos entonces era descartado, ya que los resultados para algunos KPI daban como resultado un valor cero, el cual no tiene relevancia.

#### PERIODOS DE ANÁLISIS

De la tabla base donde se almacenan los valores obtenidos de los KPI, por la columna *periodo* se pueden agrupar los registros en los tres periodos de análisis: semanal, quincenal y mensual.

**SEMANAL**

En la figura [5.3↓](#fig:5-2) se puede observar un extracto de los valores de la tabla base agrupados por periodo semanal para un determinado producto. Por ejemplo, el primer registro corresponde al producto con Id 133, periodo ’S’ (semanal), semana 49, mes 12 y año 2013, observamos que para el KPI Rotación de Stock según la fórmula [6.4↑](#eq:Rotación_Stock) se obtuvo un valor 0,769230.

Figure 5.3 Ejemplo de valores KPI para un periodo semanal

**QUINCENAL**

En la figura [5.4↓](#fig:5-3) se puede observar un extracto de los valores de la tabla base agrupados por periodo quincenal para un determinado producto. Por ejemplo, el primer registro corresponde al producto con Id 133, periodo ’Q’ (quincenal), quincena 23, mes 12 y año 2013, observamos que para el KPI Cobertura de Stock según la fórmula [6.6↑](#eq:Cobertura_Stock) se obtuvo un valor de 1,2.

Figure 5.4 Ejemplo de valores KPI para un periodo quincenal

**MENSUAL**

En la figura [5.5↓](#fig:5-4) se puede observar un extracto de los valores de la tabla base agrupados por periodo mensual para un determinado producto. Por ejemplo, el primer registro corresponde al producto con Id 133, periodo ’M’ (mensual), mes 12 y año 2013, observamos que para el KPI Margen Comercial según la fórmula [6.3↑](#eq:Margen_Comercial) se obtuvo un valor 23619,047619.

Figure 5.5 Ejemplo de valores KPI para un periodo mensual

## 5.5 Asignación de etiquetas

A cada tupla de KPI se le debe asignar una sola etiqueta de entre las siguientes *“Nada”, “Poco”, “Medio” o “Mucho*”. El etiquetado es uno de los puntos focales para el aprendizaje automático.

Para una mayor fiabilidad esta asignación de etiquetas debe ser realizada y revisada por el experto del área de compras (que podría ser el gerente de administración de compras u otra persona a cargo de la reposición de stock), es decir, el experto debería analizar los KPI por cada tupla y tomar la decisión en base a sus conocimientos y experiencia el volumen de compra apropiado para la reposición de stock para el siguiente periodo. Por ejemplo si el experto escoge la etiqueta *Nada* para una tupla dada, significa que el considera no realizar ninguna compra para el siguiente periodo.

Para el presente trabajo el etiquetado fue realizado en forma empírica, sin la intervención de un experto por la dificultad de contar con una persona especializada en el área. La estrategia para el etiquetado se basa en los siguientes pasos:

#### Paso 1:

Para cada KPI se define una tabla con un rango de valores, luego de acuerdo al valor calculado para dicho KPI se le asigna una determinada letra. A continuación mostramos la tabla de rangos para cada KPI.

Table 5.1 Rango KPI Ticket Medio

|  |  |
| --- | --- |
| (=) igual a 0 | a |
| > (mayor) a 0 y < (menor) a 1 | b |
| >= (mayor o igual) a 1 y <= (menor o igual) a 3 | c |
| > (mayor) a 3 | d |

Table 5.2 Rango KPI Cifra Ventas (%)

|  |  |
| --- | --- |
| >= (mayor o igual) a 0 y <= (menor o igual) a 20 | e |
| > (mayor) a 20 y <= (menor o igual) a 50 | f |
| > (mayor) a 50 y <= (menor o igual) a 80 | g |
| > (mayor) a 80 y <= (menor o igual) a 100 | h |

Table 5.3 Rango KPI Margen Comercial (%)

|  |  |
| --- | --- |
| >= (mayor o igual) a 0 y <= (menor o igual) a 20 | i |
| > (mayor) a 20 y <= (menor o igual) a 50 | j |
| > (mayor) a 50 y <= (menor o igual) a 80 | k |
| > (mayor) a 80 y <= (menor o igual) a 100 | l |

Table 5.4 Rango KPI Rotación Stock

|  |  |
| --- | --- |
| (=) igual a 0 | m |
| > (mayor) a 0 y < (menor) a 1 | n |
| >= (mayor o igual) a 1 y <= (menor o igual) a 3 | o |
| > (mayor) a 3 | p |

Table 5.5 Rango KPI Cobertura Stock

|  |  |
| --- | --- |
| (=) igual a 0 | q |
| > (mayor) a 0 y < (menor) a 1 | r |
| >= (mayor o igual) a 1 y <= (menor o igual) a 3 | s |
| > (mayor) a 3 y <= (menor o igual) a 10 | t |
| > (mayor) a 10 | u |

#### Paso 2:

Posteriormente cada tupla queda definida por un conjunto de 5 letras, luego a cada combinación de letras se asignó de forma empírica una de las posibles etiquetas *“Nada”, “Poco”, “Medio” o “Mucho*” denotado en el cuadro [5.6↓](#tab:un_cuadro_flotante-4-1).

Table 5.6 Tabla de etiquetado por el experto

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| aeimq | Nada | bejnq | Poco | bejoq | Poco |
| aeimr | Nada | bejnr | Poco | bejpq | Medio |
| aeims | Nada | bejns | Nada | beknq | Poco |
| aeimt | Nada | bejnt | Nada | beknr | Poco |
| aeimu | Nada | bejnu | Nada | bekns | Nada |
| ... | ... | ... | ... | ... | ... |

En la figura [5.6↓](#fig:Ejemplo-de-etiquetado_5.5) se muestra un extracto del etiquetado para un periodo semanal. La última columna contiene las etiquetas como resultado del proceso.

Figure 5.6 Ejemplo de etiquetado para periodo semanal.

## 5.6 Resumen

Sintetizando el capítulo se puede resumir que el modelado del problema del componente BI se basa en tres grandes procesos: a) proceso de Business Intelligence y sus etapas b) proceso de obtención de KPI y c) proceso de etiquetado.

El proceso de Business Intelligence incluyó tres de sus componentes, 1) el acceso a las fuentes de información, 2) el proceso de extracción, transformación y carga de los datos obtenidos de la fuente de origen y 3) el diseño y construcción del datawarehouse. El acceso a la fuente de información de los datos no representó mayores inconvenientes, el proceso consistió en restaurar en un servidor local el backup de los datos de la empresa que proporcionó la información.

Durante el ETL se tuvo que identificar las columnas de las tablas relevantes para la *extracción* que posteriormente generan las métricas de las tablas de hechos, los problemas encontrados en esta etapa se dieron por ciertas inconsistencias de datos, valores vacíos y datos nulos los cuales fueron corregidos en la *transformación* y posterior *carga* de los datos al datawarehouse. Para construir el datawarehouse el trabajo clave consistió en definir las tablas de hechos que sirven de base para el cálculo de los KPI junto con sus principales dimensiones utilizando el esquema estrella.

En el proceso de obtención de KPI, el principal trabajo fue la selección de los KPI relevantes en las áreas de negocios ventas y productos. Debido a que en la literatura los KPI miden los procesos de negocios a un nivel de generalización donde se ven a los productos como un todo, las fórmulas fueron adaptadas de modo que las mediciones fueron bajadas a un nivel de productos individuales.

El en etiquetado, lo esencial es contar con la participación de un experto en administración en compras con conocimientos y capacidad de interpretación de KPI. En este proceso se simula al experto mediante un conjunto de reglas para indicar la etiqueta a cada tupla de KPI.

# 6 Experimentación

## 6.1 Componente de solución “Machine Learning”

Se describirá cómo es la implementación del proceso de aprendizaje automático para este caso de estudio. Se mostrará primeramente cómo está constituida la salida del proceso de Business Intelligence, que en esencia proveen las instancias necesarias para la entrada del proceso de aprendizaje automático. También se verá qué clasificadores fueron utilizados, cómo se realizó el proceso de entrenamiento y de evaluación, y cuáles son las métricas de evaluación consideradas para medir el rendimiento de los clasificadores.

## 6.2 Datos de entrada de Machine Learning

La salida de Business Intelligence provee tres conjuntos de datos independientes que se corresponden con los períodos de análisis: Mensuales, Quincenales y Semanales.

* Períodos Mensuales: Se analizaron 309 productos diferentes, y por cada producto se tiene un máximo de 34 instancias. Cada instancia tiene los siguientes atributos: Ticket Medio, Cifra de Ventas, Margen Comercial, Rotación de Stock, Coeficiente de Rentabilidad, Cobertura de Stock, Cantidad, Año, Mes. La clase de cada instancia está definida por *y* ∈ {*Nada*, *Poco*, *Medio*, *Mucho*}.
* Períodos Quincenales: Se analizaron 228 productos diferentes, y por cada producto se tiene un máximo de 68 instancias. Cada instancia tiene los siguientes atributos: Ticket Medio, Cifra de Ventas, Margen Comercial, Rotación de Stock, Coeficiente de Rentabilidad, Cobertura de Stock, Cantidad, Año, Quincena. La clase de cada instancia está definida por *y* ∈ {*Nada*, *Poco*, *Medio*, *Mucho*}.
* Períodos Semanales: Se analizaron 127 productos diferentes, y por cada producto se tiene un máximo de 151 instancias. Cada instancia tiene los siguientes atributos: Ticket Medio, Cifra de Ventas, Margen Comercial, Rotación de Stock, Coeficiente de Rentabilidad, Cobertura de Stock, Cantidad, Año, Semana. La clase de cada instancia está definida por *y* ∈ {*Nada*, *Poco*, *Medio*, *Mucho*}.

## 6.3 Entrenamiento, testeo y evaluación

En la figura [6.1↓](#fig:Esquema-de-entrenamiento) se muestra la estrategia general utilizada para el entrenamiento, testeo y evaluación en los periodos semanales, quincenales y mensuales. El conjunto de entrenamiento se basa en el 70% de las instancias y el conjunto de prueba corresponde al 30% restante. Esta estrategia es la utilizada en el curso The Machine Learning Masterclass [[Sci16](" \l "biblio-29)], un curso moderno de Machine Learning para proyectos de análisis predictivo.

Figure 6.1 Esquema de entrenamiento y prueba.

En el algoritmo [6.1↓](#alg:Pseudocódigo-clasificacion) se presenta el pseudocódigo de aprendizaje y selección de los clasificadores.

**for cada período de análisis {mensual, quincenal,semanal}:**

* **for cada producto con sus instancias:**
  + **establecer conjunto de entrenamiento;**
  + **establecer conjunto de testeo;**
  + **for cada algoritmo de clasificación:**
    - **construir clasificador (conjunto de entrenamiento);**
    - **validar clasificador (stratified k-folds cross-validation);**
    - **obtener métricas de validación;**
  + **endfor;**
  + **criterios de línea de base (ZeroR, criterios del experto u otro);**
  + **seleccionar mejor clasificador (max(Kappa));**
  + **evaluar mejor clasificador (conjunto de testeo);**
  + **guardar clasificador;**
* **endfor;**

**endfor;**

Algorithm 6.1 Pseudocódigo para el proceso de clasificación.

Los algoritmos utilizados para construir los clasificadores son los proveídos por la herramienta WEKA según tabla [4.6↑](#tab:Clasificadores-Weka) y los parámetros de configuración establecidos por defecto. Otra forma de categorizar los clasificadores incluidos en WEKA es como sigue:

* Bayesianos: BayesNet, NaiveBayes, NaiveBayesUpdateable.
* Basados en funciones: Logistic, MultilayerPerceptron, SimpleLogistic, SMO.
* Basados en reglas: OneR, DecisionTable, JRip, PART, ZeroR.
* Basados en árboles: DecisionStump, J48, LMT, RandomForest, RandomTree, REPTree.

La validación se hace por el método Stratified k-fold Cross-Validation para un valor de k igual a 10 y las métricas de desempeño consideradas son el *Porcentaje de Acierto* y la *Estadística Kappa*. El criterio de línea de base utilizado es el clasificador ZeroR, el cual es uno de los criterios de línea de base más representativos para problemas de clasificación. Se considera que también puede resultar conveniente que el experto en compras establezca su propio criterio de línea de base, como puede ser un umbral mínimo de porcentaje de acierto aceptado.

## 6.4 Análisis del desempeño

### 6.4.1 Comparación con línea base ZeroR

En este apartado se contrastan los resultados obtenidos mediante una comparación entre el porcentaje de acierto de la línea base ZeroR, el porcentaje de acierto del mejor algoritmo alcanzado en la validación y el porcentaje de acierto en el testeo del mismo algoritmo. Como se mencionó la línea base es el clasificador ZeroR, el cual es el criterio mínimo de aceptación de un clasificador. Los puntos de la línea de entrenamiento son los porcentajes de acierto de los clasificadores seleccionados durante la validación, según el algoritmo [6.1↑](#alg:Pseudocódigo-clasificacion). La línea de test contiene los puntos que representan los porcentajes de aciertos durante la evaluación sobre el conjunto de prueba.

Los productos incluidos en el gráfico constituyen un subconjunto representativo, esto debido a la extensa cantidad de productos y su dificultad para graficarlos por completo.

En las figuras [6.2↓](#fig:6.3) y [6.3↓](#fig:6.4) los gráficos contienen las líneas comparativas para periodos semanales y quincenales respectivamente. De acuerdo a las experimentaciones el porcentaje de acierto en el entrenamiento es siempre mayor o igual a la línea base, en concordancia con el algoritmo [6.1↑](#alg:Pseudocódigo-clasificacion). En el testeo por tratarse de un conjunto de datos separado del conjunto de entrenamiento los porcentajes de aciertos varían.

Se dieron dos casos atípicos donde el porcentaje de acierto fue menor al de la línea base, estos casos aparecen cuando hay una etiqueta dominante, en periodos semanales fue el producto 222 con el 88% de etiquetas “Nada” y en periodos quincenales fue el producto 400 con el 92% de etiquetas “Nada”. De los resultados se puede deducir que hay aprendizaje, ya que los porcentajes de acierto alcanzados en el testeo se encuentran por encima de la línea base en la mayoría de los casos.

Figure 6.2 Entrenamiento vs Test vs ZeroR para períodos semanales.

Figure 6.3 Entrenamiento vs Test vs ZeroR para períodos quincenales.

La figura [6.4↓](#fig:6.5) muestra las comparativas para periodos mensuales. Al igual que los periodos semanales y quincenales el porcentaje de acierto en el entrenamiento es siempre mayor o igual a la línea base. En el testeo por tratarse de un conjunto de datos separado del conjunto de entrenamiento los porcentajes de aciertos varían. Se tiene un caso atípico donde el porcentaje de acierto fue menor al de la línea base, este caso aparece cuando hay una etiqueta dominante el cual se dio en el producto 309 con el 93% de etiquetas “Medio”.

Figure 6.4 Entrenamiento vs Test vs ZeroR para períodos mensuales.

### 6.4.2 Recuentos por tipo de clasificador

Por cada producto y período de análisis se elige como clasificador aquel que haya alcanzado el mayor valor de *Kappa*. En la figura [6.5↓](#fig:6.6) se muestra como quedó la distribución de clasificadores para períodos mensuales. Por ejemplo, el clasificador *IterativeClassifierOptimizer* resultó una mejor solución en 23 productos para periodos semanales.

Figure 6.5 Conteo de clasificadores para períodos semanales.

En la figura [6.6↓](#fig:6.7) se muestra como quedó la distribución de clasificadores para períodos quincenales. Por ejemplo, el clasificador *IterativeClassifierOptimizer* resultó una mejor solución en 60 productos para periodos quincenales.

Figure 6.6 Recuento de clasificadores para períodos quincenales.

En la figura [6.7↓](#fig:6.8) se muestra como quedó la distribución de clasificadores para períodos mensuales. Por ejemplo, el clasificador *Logistic* resultó una mejor solución en 92 productos para periodos mensuales.

Figure 6.7 Recuento de clasificadores para períodos mensuales.

### 6.4.3 Porcentajes de acierto por período

Se puede observar en la figura [6.8↓](#fig:6.9) que la técnica propuesta alcanza altos porcentajes de aciertos en promedio, tanto para periodos mensuales, quincenales como semanales.

Figure 6.8 Promedio de porcentaje de aciertos para los tres períodos de análisis.

Como se trata de una prueba exhaustiva, por cada producto se intenta con todos los algoritmos de clasificación posible y se evalúa con el método Stratified 10-fold Cross Validation. Estas métricas de desempeño preliminares dan indicio de que la técnica propuesta en este trabajo puede alcanzar altos grados de confiabilidad. Obtener buenos resultados depende en gran medida de que la obtención de valores de *KPI* y la asignación de etiquetas hayan sido realizadas correctamente.

## 6.5 Componente de solución “Técnica de solución propuesta”

En el algoritmo [6.2↓](#alg:6.2) se muestra el mecanismo para obtener el pronóstico de la demanda en un ambiente de producción.

**for cada próximo período a pronosticar {mensual,quincenal,semanal}:**

* **for cada producto:**
  + **obtener KPIs del período actual finalizado;**
  + **ejecutar su mejor clasificador (KPIs);**
  + **obtener etiqueta {nada,poco,medio,mucho}**
  + **extrapolar a valores continuos(criterio experto);**
* **endfor;**

**endfor;**

Algorithm 6.2 Pseudocódigo para el proceso de pronóstico de la demanda.

Como se mencionó anteriormente, la técnica propuesta arroja como resultado un valor discreto *y* ∈ {*Nada*, *Poco*, *Medio*, *Mucho*}. Luego en función de la etiqueta resultante, del tipo de producto y del período seleccionado, el experto extrapola a un valor continuo que representa la cantidad en la orden de compra. El significado de las etiquetas varía en función del tipo de producto.

## 6.6 Discusión

### 6.6.1 Impacto del período de análisis

Una de las decisiones que se debe tomar es acerca del tiempo asignado al período de análisis. En este trabajo se analizaron tres períodos distintos: mensuales, quincenales y semanales con propósitos experimentales y por ser los más comunes en el ámbito comercial. En la práctica, la elección del período es una decisión estratégica a nivel gerencial que depende en gran medida del sector y tamaño de la empresa, tipos de productos y otros criterios.

En el presente trabajo, por tratarse de períodos de tiempo muy cercanos (1, 2 y 4 semanas) no se observan diferencias significativas en los porcentajes de aciertos. Otro factor a tener en cuenta es que para períodos de tiempo muy extensos (6, 12 meses) existe mayor incertidumbre en el pronóstico.

### 6.6.2 Impacto del etiquetado

La técnica propuesta se trata de un sistema parametrizado, donde las variables principales son el período comercial y las etiquetas seleccionadas para la clasificación. Por cuestiones de practicidad y generalidad se eligió para este trabajo un enfoque de problema de clasificación. El etiquetado proporciona mayor flexibilidad al sistema y un entorno más controlable, en comparación a un sistema de asignación de valores continuos. La flexibilidad del sistema permitió emular la opinión del experto en compras y encontrar una cantidad eficiente de etiquetas.

# 7 Conclusiones

## 7.1 Conclusiones Generales

Este trabajo se enfocó en proponer una nueva técnica de pronóstico de la demanda de productos, para reposición de stock en empresas retail. En el capítulo 2 se mencionó que la gestión de compras es uno los ejes centrales en la actividad empresarial y la decisión del volumen de compras para cada producto es un desafío que enfrentan las empresas al momento de reponer el stock. Partiendo de esta premisa y analizando las técnicas de pronóstico de la demanda empleadas en la actualidad y el creciente incremento del uso de tecnologías de Business Intelligence en las organizaciones, se encontró la oportunidad de desarrollar una nueva técnica de pronóstico.

En esta nueva técnica se utilizan los Indicadores Claves de Rendimiento y apoyados en la experiencia de un experto en compras (gerente o encargado de compras) se realiza la experimentación utilizando algoritmos de clasificación de Machine Learning.

De acuerdo a los resultados experimentales se obtuvieron altas tasas de aciertos, haciendo pruebas exhaustivas con varios algoritmos de clasificación y validando con un método ampliamente aceptado. Es importante considerar que para una implementación real es primordial tener en cuenta los siguientes aspectos: el etiquetado multiclase puede ser reemplazado por valores continuos; el experto en compras debe realizar o guiar todo el proceso de etiquetado ya sea con valores continuos o discretos.

Para periodos de análisis cortos se obtuvieron buenos resultados, pero también podrían considerarse periodos de tiempo mas extensos; el conjunto de KPI seleccionados fueron en base a las limitaciones de la base de datos, es importante incluir otros KPI que consideren más aspectos como costos de inventario, marketing, clientes y otros.

Como se mencionó en la introducción, para el sector retail la mayoría de las empresas puede pronosticar la demanda total de todos los productos como un grupo con errores menores al 5%, no obstante el pronóstico de la demanda de un producto puede generar errores considerablemente mayores con las técnicas vigentes. Con la técnica propuesta se logran errores menores al 5% por cada producto. Se pretende que esta técnica se convierta en una herramienta de apoyo en la toma de decisiones del gerente de compras en el proceso de reposición de stock.

## 7.2 Aporte de este Trabajo

Este trabajo de grado brinda los siguientes resultados:

* Definición de KPI adaptados para la medición del rendimiento de productos individuales.
* Esquema básico y automatizado para el etiquetado de las instancias.
* Integración de las tecnologías de Business Intelligence centrado en los Indicadores Claves de Rendimiento y Machine Learning basado en aprendizaje supervisado.
* Herramienta de pronóstico de la demanda de productos para reposición de stock.

## 7.3 Trabajos Futuros

Con el propósito de futuras mejoras del presente trabajo, a continuación se citan una serie de propuestas:

* Incorporar mas KPI referentes a otros procesos de negocios que inciden en la venta de un producto.
* Asignación por parte del experto en compras de valores continuos a la clase de las instancias.
* Etiquetado basado en clustering (aprendizaje no supervisado) con la aprobación del experto en compras.
* Optimizar los algoritmos de aprendizaje mediante el ajuste de los parámetros (*Tuning parameters*).
* Incluir costos asociados a un producto (costos de almacenamiento, seguro, mantenimiento, etc.).
* Desarrollar un software SaaS[↓](" \l "nom-saas) (Software as a Service) que provea un servicio de pronóstico de la demanda.