Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias Físico Matemáticas

Una Introducción A Los Modelos De Machine Learning

Tesis presentada al

Colegio de Matemáticas

como requisito parcial para la obtención del grado de

Licenciado en Matemáticas Aplicadas

por

Brian Romero Rojas

Asesorado por

Dr. Bulmaro Juárez Hernández

Puebla Pue.
26 de octubre de 2020

“ La vida es y siempre seguirá siendo una ecuación incapaz de resolver, pero tiene ciertos factores que conócemos ”

Nikola Tesla

Agradecimientos

Primero que nada, quiero agradecer a Dios por darme fortaleza y salud durante todo este camino, que, aunque fuera muy largo logre concluir, quiero agradecer principalmente a mis 2 madres Margarita y Maguito que durante toda mi vida han estado a mi lado apoyándome y motivándome para mejorar cada día. Aún recuerdo cuando les dije la noticia de que había quedado en la universidad y también el primer día que me fui a Puebla, y se cuanta tristeza les causé en ese momento, por eso, aunque las cosas no fueron fáciles siempre eran la razón para seguir adelante y no rendirme jamás. Sus miradas al verme concluyendo esta etapa de mi vida fue la mejor recompensa que pude haber recibido.

Le agradezco a Paul por apoyarme a mí y a mi madre siempre que lo necesitábamos. Eber y Hugo mis hermanos de otra madre muchas gracias por ser parte de esta etapa de mi vida y hablar conmigo siempre que lo necesite, todas las noches de desvelo jugando fueron una gran ayuda para distraerme y no frustrarme durante los estudios. Chepe tu fuiste la persona que me acompaño durante todo este camino, vivir a unos metros de ti fue de las mejores cosas que la universidad me dejó, espero algún día volver a realizar una aventura nueva juntos. Memo muchas gracias por tu amistad durante la universidad.

Le agradezco a mi Padre, a mis abuelos y a toda mi familia por apoyarme y siempre creer en mí, fueron una razón más para no darme por vencido y lograr todas mis metas. A Kary por ser parte de mi vida y creer en mí, realmente no puedo resumir toda la motivación que me daba.

Por último, quiero darle las gracias al Dr. Bulmaro Juárez Hernández por aceptar trabajar conmigo a pesar de no conocerme tanto, por guiarme en este trabajo que me costó muchas horas de dedicación y esfuerzo. Agradezco a mis sinodales, Dra. Hortensia J. Reyes Cervantes, Dr. José Jacobo Oliveros Oliveros y Dra. Gladys Linares Fleites por tomarse el tiempo de revisar mi trabajo y ayudar a que este fuera mejor.

Solo queda decir que, si tienes una meta no te detengas hasta conseguirla. Es muy importante que a pesar de tener o no tener personas que confíen en ti, tu siempre creas en que puedes lograr las cosas que te propongas.

GRACIAS TOTALES.

Índice general

Resumen

Actualmente el término Inteligencia artificial es usado de manera incorrecta para describir procesos en los cuales no existe Inteligencia Artificial, lo más cercano que estamos a esto es un subconjunto llamado Machine Learning, aquí es donde se están llevando a cabo los avances más importantes en el ámbito de la tecnología, es una nueva forma de programación donde se automatizan tareas específicas como la creación de modelos de Credit Scoring, filtros de spam, etc. Los préstamos son uno de los productos más importantes de los bancos, por lo tanto, están buscando nuevas estrategias comerciales para poder convencer a los clientes de que soliciten préstamos. Sin embargo, algunos clientes se comportan de manera negativa después de que se aprueba su solicitud. Para evitar esta situación, los bancos tienen que encontrar algunos métodos para predecir el comportamiento de los clientes y poder evitar pérdidas. Los algoritmos de Machine Learning tienen un rendimiento bastante bueno en este propósito, por lo que son ampliamente utilizados por los bancos. Aquí, se trabajará en la predicción del comportamiento de los préstamos utilizando modelos de aprendizaje automático.

Palabras clave: *Machine Learning (Aprendizaje Automático), Credit Scoring (Puntaje de crédito)*

Introducción

Actualmente se escucha cada vez más seguido la palabra Inteligencia Artificial y es que se está viviendo en la era de los datos, todos los días se genera una cantidad enorme de datos los cuales pueden convertirse en información importante para las personas que tienen los conocimientos adecuados. Sin embargo, en ocasiones a lo que se le llama Inteligencia Artificial es usada de manera incorrecta para describir acciones en las cuales no existe ésta. En el primer capítulo se desarrollará el concepto de Inteligencia Artificial para comprender mejor el alcance de esta nueva tecnología, los límites que todavía existen y distinguir lo que es Inteligencia Artificial y lo que no. En nuestros días la explosión de los nuevos avances estan enfocados en un subconjunto llamado Machine Learning este tiene un potencial enorme, el cual le da a una computadora la capacidad de aprender de los datos y poder realizar una tarea en específico, ya sea para predecir valores o clasificar, estos datos pueden venir ya con etiquetas establecidas o sin ellas.

Esta tesis tiene como principal objetivo de estudio la creación de modelos de Machine Learning para poder automatizar tareas. Para poder complementar de manera adecuada la teoría, se trabajará en el caso de estudio, en el cual se implementará un modelo de Credit Scoring con una base de datos de un banco de Taiwán del año 2005 (c.f [(17)]. La base de este trabajo se encuentra en los libros " An Introduction to Statistical Learning" (c.f [9]) y "Data Science from Scratch " (c.f [11]). Machine Learning es un proceso que funciona mejor con una cantidad enorme de datos a la cual se le llama Big Data, que gracias al internet no es difícil conseguir o construir una base de datos de un tamaño considerable, ya sea con datos históricos o con datos en tiempo en real, y se puede trabajar con estos datos ya que la capacidad de cómputo y tecnología crecen a pasos agigantados. Décadas atrás sólo se trabajaba con bases de datos que eran llenadas a mano, y eso causaba que sólo se pudieran enfrentar a problemas específicos, pero en la actualidad, las bases de datos contienen diferentes tipos de formatos, lo que permite poder enfrentar distintos tipos de problemas desde perspectivas diferentes y con esto resolver problemas que antes no se habían planteado. Esta es una de las razones del porqué se están logrando tantos avances.

El objetivo de estas tecnologías no es sustituir a los humanos como se cree, al contrario, el objetivo es facilitar las tareas y hacerlas más inteligentes. Para una persona promedio es muy difícil trabajar con demasiados datos rápidamente y de manera eficiente, Una aplicación de la automatización de procesos es la creación de modelos de Credit Scoring, antiguamente una persona que solicitaba un préstamo era evaluada por un trabajador, éste evaluaba las características del solicitante y en base a su experiencia aceptaba o negaba el crédito, pero el problema de esto es cuando llegan miles de solicitudes y se requiere que se evalúen de manera rápida y eficiente, aquí entra el Credit Scoring, ya que en base a la experiencia de solicitudes anteriores este encuentra patrones y tendencias, y automatiza de manera rápida para tomar la decisión de si otorgar o no un crédito. En el Segundo capítulo se verá la historia del Credit Scoring, así como sus fundamentos y definiciones de lo que es un crédito para poder entender el contexto del problema de nuestro caso de estudio.

Los modelos de Credit Scoring no son más que modelos estadísticos que deciden si otorgar

o no un crédito en base a ciertos parámetros y reglas de decisión. En el tercer capítulo se verán a detalle los modelos estadísticos utilizados para la creación de modelos de Credit Scoring, se utilizarán los modelos de regresión logística, árboles de decisión y redes neuronales. Cuando se utiliza el modelo de redes neuronales en un proceso de Machine Learning se trabaja con un subconjunto llamado Deep Learning, este subconjunto es en el cual se han llevado a cabo los avances más importantes, como es el reconocimiento de imágenes y la creación de autos inteligentes.

Para la creación de estos modelos es importante entender cómo funcionan los modelos estadísticos utilizados, para entender mejor por qué se toma cada decisión, como también definir buenas métricas que midan el rendimiento del modelo, las métricas a considerar se deben elegir en base al conjunto de datos con el que se trabajará y con la tarea que se requiere, como se verá más adelante.

Capítulo 1

Inteligencia Artificial

La mayoría, cuando escucha el término *Inteligencia Artificial*, lo primero que se les viene a la cabeza son robots que piensan por ellos mismos y no necesitan de los humanos, esto debido a las películas e historias que están acostumbrados a ver en el cine y en la televisión en el día a día, pero en estos tiempos aún se está lejos de esos objetivos. El término inteligencia artificial está desde 1956, cuando un grupo de investigadores del área lo creó en una conferencia celebrada en la universidad de Darthmouth ubicada en los Estados Unidos. Desde entonces se han presenciado una montaña rusa de avances en el tema, al inicio el objetivo era que la inteligencia humana pudiera ser descrita de forma tan precisa que una máquina fuera capaz de simularla.
La terminología para definir correctamente inteligencia artificial es un poco confusa y cambia con el tiempo sus interpretaciones, según la Real Academia de la lengua (RAE), "La Inteligencia artificial es un programa de computación diseñado para realizar determinadas actividades que se consideran propias de la inteligencia humana, como el aprendizaje o el razonamiento lógico". Una definición que puede resumir las diferentes interpretaciones sería, "La Inteligencia Artificial es la inteligencia llevada a cabo por máquinas".
La Inteligencia Artificial es la combinación de algoritmos planteados con el propósito de crear máquinas que presenten las mismas capacidades que el ser humano. Es probablemente la disciplina técnico-científica con más potencial de los últimos años y sin embargo es difícil dar una definición concisa y clara de que es la inteligencia artificial. Normalmente un sistema de inteligencia artificial es capaz de analizar una gran cantidad de datos (*Big Data*), identificar patrones y tendencias, y, por lo tanto, formular predicciones de forma automática, con rapidez y precisión. Lo importante de la inteligencia artificial es que permite que las experiencias cotidianas sean más "inteligentes".
La Inteligencia Artificial se trata mucho más sobre el proceso y la capacidad de pensamiento superpoderado y el análisis de datos que sobre cualquier formato o función en particular. Aunque la Inteligencia Artificial muestra imágenes de robots de aspecto humano de alto funcionamiento que se apoderan del mundo, la Inteligencia Artificial no pretende reemplazar a los humanos, contrario a esto su objetivo es mejorar significativamente las capacidades y contribuciones humanas, esto mejorando el rendimiento y productividad mediante la automatización de procesos o tareas que antes requería el poder humano. Ya que puede dar sentido a los datos a una escala que ningún humano podría jamás, esto la convierte en un activo comercial muy valioso (c.f [16]).
Los casos de uso en los negocios son amplios: modelos de crédito, modelos de segmentación de clientes (*agrupamiento*), modelos de probabilidad de compra y modelos de migración de clientes, entre otros, como se verá más adelante en el caso de estudio.
A medida que pasa el tiempo y las computadoras se vuelven más capaces, tecnología que una vez se pensó que requería de inteligencia artificial se elimina de la definición, de manera interesante algunas máquinas hacen tareas que antiguamente hacían los humanos y las personas suelen decir que eso no es realmente inteligencia artificial, esto es conocido como el AI effect. Para poder entender todo esto es importante conocer algunos subconjuntos de la inteligencia artificial.

1.1. Inteligencia artificial fuerte

Es una inteligencia artificial que puede actuar exitosamente con cualquier prueba intelectual que un humano puede hacer, incluyendo aprender, planear y tomar decisiones bajo incertidumbre, comúnmente con lenguaje natural, hacer bromas, manipular personas o reprogramarse solas. Esta última es una gran meta, si se crea una inteligencia artificial capaz de reprogramarse sola podría desbloquear un círculo de recursividad que podría dirigir a una explosión de inteligencia sobre algunos periodos de tiempo desconocidos, acortando muchas décadas a un solo día.
La Inteligencia Artificial fuerte implicaría que un ordenador no simula una mente, sino que es una mente, y por consiguiente debería de ser capaz de tener una inteligencia igual o incluso superior a la humana, para entender cómo se estudiará, Machine Learning es un buen lugar para empezar. La Inteligencia Artificial fuerte puede cambiar el mundo para siempre, pero de esta inteligencia se está todavía demasiado lejos.

1.2. Inteligencia artificial débil

La Inteligencia Artificial débil, por otro lado, consiste en construir programas que realicen tareas específicas sin necesidad de tener estados mentales. Es la capacidad de los ordenadores para realizar tareas específicas, incluso mejor que las personas. En ciertos dominios, los avances de la inteligencia artificial débil superan mucho la pericia humana, como para buscar soluciones en diagnósticos médicos y muchos otros aspectos relacionados con la toma de decisiones.
La Inteligencia Artificial débil es capaz de resolver problemas muy bien definidos y acotados, como filtrar correos que son spam. Este tipo de Inteligencia Artificial es la que ha provocado la verdadera explosión en los últimos tiempos. Se han aplicado técnicas como Machine Learning o Deep Learning, ya que han demostrado cómo es posible programar una máquina y entrenarla para resolver todo tipo de tareas.

1.3. Big Data

¿Qué es? y ¿por qué es importante? Big data es un término que describe el gran volumen de datos estructurados y no estructurados. Pero no es la cantidad de datos lo importante. Lo que importa es lo que las organizaciones hacen con los datos. El Big Data puede ser analizado para obtener ideas que conlleven a mejores decisiones y acciones de negocios estratégicos.
Aunque el término "Big Data" es relativamente nuevo, la acción de recopilar y almacenar grandes cantidades de información para su posterior análisis se viene realizando desde hace muchos años. El concepto cobró impulso a principios de la década de los 2000 cuando el analista de la industria Doug Laney articuló la definición ahora muy popular del Big Data como las 4 Vs.

- **Volumen**: El Big Data implica un volumen enorme de datos. Aunque el tamaño utilizado para determinar si un conjunto de datos determinado se considera Big Data no está firmemente definido y sigue cambiando con el tiempo, la mayoría de los analistas y profesionales actualmente se refieren a conjuntos de datos que van desde 30-50 Terabytes a

varios Petabytes. En un inicio los datos eran creados por los propios empleados, pero ahora que los datos son generados automáticamente por máquinas, redes e interacciones personales en sistemas como redes sociales los volúmenes a analizar son masivos. La tecnología para guardar y procesar ha avanzado paralelamente por lo que el mayor problema ahora no es tanto el tamaño comparado con otras dimensiones como la veracidad.

- **Variedad**: La variedad se refiere a las diferentes fuentes y tipos de datos tanto estructurados como no estructurados. Hace pocos años, los únicos datos que se almacenaban eran de fuentes como hojas de cálculo y bases de datos. Ahora, los datos llegan en la forma de emails, fotos, videos, sistemas de monitorización, PDFs, ficheros de sonido, etc. Esta variedad en datos no estructurados crea problemas de almacenamiento, minería de datos y análisis de la información.
- **Velocidad**: La velocidad en Big Data se refiere al ritmo en que los datos de entrada fluyen desde las diversas fuentes como procesos de negocio, máquinas y sensores, redes sociales, dispositivos móviles, etc. El flujo de datos es masivo y continuo. Estos datos recogidos en tiempo real permiten ayudar a investigadores y organizaciones a la hora de tomar decisiones aportando valiosa información que suponen ventajas competitivas estratégicas.
- **Veracidad**: La veracidad en el Big Data se refiere al sesgo, el ruido y la alteración de datos. Los responsables del proyecto Big Data han de preguntarse honestamente si los datos que se almacenan y extraen son directamente relacionados y significativos al problema que se trata de analizar. Esta característica puede ser el mayor reto cuando se comparan con otras como el volumen o la velocidad. Cuando se valore el alcance en su estrategia de Big Data es necesario contar en el equipo con socios imparciales que ayuden a mantener los datos limpios y asegurarse que los procesos no acumulen "datos sucios" en sus sistemas.

1.4. Aprendizaje Automático (Machine Learning)

Mientras que los académicos siguen debatiendo los detalles sobre lo que es Inteligencia Artificial y lo que no, la industria está usando el término Machine Learning (ML) para referirse a un tipo particular de aprendizaje automático. De hecho, la mayoría de las veces la gente usa los términos de manera intercambiable, confundiéndolo con la inteligencia artificial débil, aquí los avances más importantes se están llevando a cabo. La inteligencia artificial es una rama de la ciencia que intenta imitar las habilidades humanas, mientras que el ML es un subconjunto de la inteligencia artificial que entrena a una máquina para aprender.
Antes de hablar del aprendizaje automático se hablará sobre lo que es un modelo. Un modelo es simplemente una especificación de una relación matemática que existe entre diferentes variables. Machine Learning es un método del análisis de datos que automatiza la creación y uso de modelos estadísticos, que da a una computadora la habilidad de "aprender" de los datos, se basa en la idea de que los sistemas pueden aprender de los datos, identificar patrones y tomar decisiones con mínima intervención humana. El aspecto iterativo del ML es importante porque a medida que los modelos son expuestos a nuevos datos, estos pueden adaptarse de forma independiente. La diferencia con técnicas anteriores está en su capacidad para adaptarse a los cambios en los datos a medida que van entrando en el sistema y aprender de las propias acciones del modelo. Ahí radica el aprendizaje y el dinamismo de los que carecían las técnicas previas. Es una ciencia que no es nueva, pero ha cobrado un nuevo impulso, su resurgimiento se debe a los volúmenes y variedades crecientes de datos disponibles, procesamiento computacional más económico y poderoso, y almacenaje de datos

accesible, estos datos alimentan los modelos de ML.

La exactitud de los modelos puede incrementarse de forma importante si son entrenados con Big Data, sin suficientes datos, tratando de tomar decisiones con pequeños subconjuntos de datos se pueden malinterpretar tendencias o perder patrones que estaban por emerger, se necesita un conjunto de datos adecuado para usarse en el proceso.
Los humanos pueden crear, por lo general, uno o dos buenos modelos por semana, el Machine Learning puede crear miles de modelos por semana. En esencia el ML es un etiquetador de cosas que toma tu descripción de algo y te dice la etiqueta que le debería corresponder, consiste en hacer modelos para etiquetar cosas utilizando ejemplos en lugar de instrucciones, contrario a lo que piensan todos, no es una caja mágica. Es una nueva forma de programación, una nueva manera de comunicar tus deseos a una computadora.
En la forma tradicional, un programador crearía un modelo a "mano", pero él tendría que pensar exactamente las instrucciones. No sería mucho mejor si se pudierá decir a la computadora "mira, aquí hay un montón de ejemplos de gatos y aquí un montón de ejemplos de no-gatos, ve y descífralo como puedas". Esa es la esencia del ML, es una forma completamente diferente de programar, en lugar de dar instrucciones especificas paso a paso se puede programar con ejemplos, y el algoritmo de ML encontrará patrones en los datos para convertirlos en esas instrucciones que no se sabía cómo escribir.

En los modelos de ML, para que estos puedan aprender, se necesitan entrenar con un conjunto de datos X, y cada dato de este conjunto es una unidad experimental. Una unidad experimental es cualquier objeto o concepto que se puede medir o evaluar de alguna manera (hombres, animales, compañías, etc.). Cada unidad experimental tiene sus características (color, altura, edad, etc.), a estas características se les va a llamar predictores. Cuando el modelo evalúa el conjunto de datos obtiene una variable respuesta Y. Esta relación se puede ver como $Y = f(X) + \epsilon$.

La función f que conecta a la entrada con la salida en general es desconocida, y existen dos razones muy importantes para querer estimar esta f, predicción e inferencia. En muchas situaciones se tiene un conjunto de características de entrada disponibles pero la salida no puede ser obtenida fácilmente, el objetivo es estimar una f que te pueda predecir estas salidas. Por otro lado, también es interesante investigar como Y es afectado cuando X cambia, en esas situaciones se desea estimar f pero no necesariamente para predecir cosas de Y, se quiere entender la relación entre Y y X, en específico entender cómo Y cambia en función de X.

Los sistemas de Machine Learning pueden ser clasificados de acuerdo con la cantidad y tipo de supervisión que tienen durante su entrenamiento, existen 4 tipos de categorías principales, Aprendizaje Supervisado (Superviced Learning), Aprendizaje No Supervisado (Unsupervised Learning), Aprendizaje Semi Supervisado (Semi Supervised Learning) y Aprendizaje De Reforzamiento (Reinforcement Learning). En este trabajo se dará una introducción a los 2 primeros.

1.4.1. Aprendizaje no supervisado (Unsupervised Learning)

En el aprendizaje no supervisado, como se puede adivinar, los datos de entrenamiento no están etiquetados, el sistema intenta como aprender sin un maestro. En esta situación, se está de una manera trabajando a ciegas, hace falta una variable respuesta que supervise

el análisis, en este caso se busca entender las relaciones entre las variables o entre las observaciones. El objetivo será la extracción de información significativa, sin la referencia de variables de salida conocidas y mediante la exploración de la estructura de dichos datos sin etiquetar.
En esencia los métodos no supervisados agregan etiquetas a los datos y se convierte en supervisado. Existen dos categorías principales del aprendizaje no supervisado: el agrupamiento y la reducción dimensional.

Agrupamiento (Clustering)

El agrupamiento es una técnica exploratoria de análisis de datos, que se usa para organizar información en grupos con significado sin tener conocimiento previo de su estructura. Cada grupo es un conjunto de objetos similares que se diferencia de los objetos de otros grupos. El objetivo es obtener un número de grupos de características similares. Un ejemplo de aplicación de este tipo de algoritmos puede ser para establecer tipos de consumidores en función de sus hábitos de compra, para poder realizar técnicas de marketing efectivas y "personalizadas".

Reducción dimensional (Dimentional reduction)

Es común trabajar con datos en los que cada observación se presenta con alto número de características, en otras palabras, que tienen alta dimensionalidad. Este hecho es un reto para la capacidad de procesamiento y el rendimiento computacional de los algoritmos de Machine Learning. La reducción dimensional es una de las técnicas usadas para mitigar este efecto. La reducción dimensional funciona encontrando correlaciones entre las características, lo que implica que existe información redundante, ya que alguna característica puede explicarse parcialmente con otras (por ejemplo, puede existir dependencia lineal). Estas técnicas eliminan "ruido" de los datos (que puede también empeorar el comportamiento del modelo), y comprimen los datos en un subespacio más reducido, al tiempo que retienen la mayoría de la información relevante.

1.4.2. Aprendizaje supervisado (Supervised Learning)

Se refiere a un tipo de modelos de Machine Learning que se entrenan con un conjunto de ejemplos en los que los resultados de salida son conocidos, los modelos de aprendizaje supervisado son aquellos en los cuales los datos de entrenamiento que alimentan a los algoritmos incluyen ya las soluciones deseadas, que se llamarán etiquetas, lo que se busca es tener muchas observaciones de entrenamiento como sea posible, y hacer que el modelo pueda aprender la relación entre la respuesta y sus predictores, con el objetivo de predecir exactamente las respuestas para futuros datos no procesados previamente, es decir que no se conozca su etiqueta.
Las dos principales pruebas del aprendizaje supervisado son la regresión y la clasificación, las variables que se utilizan en los modelos pueden ser caracterizadas ya sean cuantitativas o cualitativas. Las cuantitativas son aquellas que toman valores numéricos, y las cualitativas son las que toman valores en una de k diferentes clases o categorías. Se dice que los problemas con una respuesta cuantitativa son problemas de regresión y los que tienen una respuesta cualitativa son problemas de clasificación.
Se tiende a elegir qué modelo usar dependiendo si el problema es de regresión o clasificación. Sin embargo, si los predictores son cuantitativos o cualitativos, no es muy importante para elegir el modelo, la mayoría de los modelos se eligen independientemente del tipo de variables

tomadas como predictores.

Clasificación

Un ejemplo de aprendizaje supervisado aplicado a la clasificación es el filtro de spam para emails, este es entrenado con muchos ejemplos de emails con su debida clasificación, ya sea spam o no spam, y este aprende las características de los mensajes que son spam y de los que no son, para poder así clasificar nuevos emails que no estén etiquetados.

Regresión

Una prueba típica de regresión es la de predecir un valor numérico, como el precio de un carro dado un conjunto de características (edad, marca, kilometraje, etc.) los cuales son los predictores, para entrenar este sistema se necesita dar muchos ejemplos de carros con sus características y sus precios, para después dependiendo de las características del carro este predecirá su precio.

NOTA: algunos algoritmos de regresión pueden usarse para clasificación, y viceversa, por ejemplo, la regresión logística es comúnmente usada para clasificación, pero también puede dar un valor numérico que corresponda a la probabilidad de pertenecer a alguna clase.

1.4.3. Conjunto de entrenamiento y conjunto de prueba (Training set y Test set)

Al momento de entrenar el modelo con el conjunto de datos, es importante partirlo en dos conjuntos, un conjunto de entrenamiento y un conjunto de prueba, el tamaño en que se divide depende de la situación, pero el tamaño más común es un 70 % en el conjunto de entrenamiento y un 30 % en el conjunto de prueba. Otra manera de particionar el conjunto de datos es mediante una técnica llamada *K- Fold Cross Validation*, para saber más al respecto ser puede ver el Apéndice B donde se profundiza mas en esta técnica.
Las observaciones en el conjunto de entrenamiento forman la experiencia que el algoritmo usa para aprender. Este conjunto va a ayudar al algoritmo a encontrar los patrones y las relaciones entre la variable respuesta y los predictores, el conjunto de entrenamiento contiene las etiquetas, y el modelo puede aprender con estos ejemplos etiquetados.
El conjunto de prueba no tiene etiquetas, es decir, que no se conoce el valor a predecir, es un conjunto de observaciones utilizadas para evaluar el rendimiento del modelo utilizando una medida de rendimiento. Es importante que no se incluyan observaciones del conjunto de entrenamiento en el conjunto de prueba. Si el conjunto de prueba contiene ejemplos del conjunto de entrenamiento, será difícil evaluar si el algoritmo ha aprendido a generalizarse a partir del conjunto de entrenamiento o simplemente lo ha memorizado. Este conjunto ayudará para ver que tan bien funciona el modelo propuesto.

1.5. Proceso de un modelo de aprendizaje automático

Para poder crear un modelo de ML es un proceso de varias etapas, el cual es un ciclo que se repite hasta obtener el resultado deseado.

Comprensión del problema

Este paso rara vez se menciona en los libros y publicaciones acerca de la creación de los modelos, pero es de los más importantes. La etapa inicial para comenzar un proyecto es saber que problema se va a solucionar. Esto se aplica a cualquier proyecto, por supuesto, si no hay problemas, no hay nada que resolver. El problema de este trabajo es que los clientes que no pagan producen pérdidas al banco. Una vez ubicado el problema se tiene que determinar el objetivo de crear un modelo de Aprendizaje Automático, en este caso el objetivo es crear un modelo que permita localizar a los clientes morosos antes de que causen pérdidas.

Reunir los datos (Data collection)

Una vez que se sabe qué problema se va a resolver lo que se necesita son datos, ya que los modelos de ML se entrenan con estos, la cantidad y calidad de los datos van a determinar la precisión del modelo. El resultado de este paso es una representación de datos que se usarán para el entrenamiento del modelo.

Preparación de datos (Data preparation)

Este es uno de los pasos más importantes en cualquier aplicación de ML, y puede ser el paso al que más tiempo se le invierta, a este paso también se le conoce como limpieza de datos ya que al momento en que se tiene un conjunto de datos, estos no van a venir en el formato más óptimo para ser procesados por el modelo, ya que los datos pueden venir con valores faltantes (missing values), valores atípicos (outliers), entre otras cosas. En estos casos el preprocesamiento de datos es una tarea que se debe realizar de manera obligatoria. Muchos algoritmos requieren que las características estén en la misma escala (por ejemplo, en el rango [0,1]) para optimizar su rendimiento, lo que se realiza frecuentemente aplicando técnicas de normalización o estandarización en los datos. También, se puede encontrar en algunos casos que las características seleccionadas están correlacionadas, y por tanto son redundantes para extraer información con significado correcto de ellas. En este caso se tendrá que usar técnicas de reducción dimensional para comprimir las características en subespacios con menores dimensiones. Finalmente, se fragmentará de forma aleatoria el conjunto de datos original en el conjunto de entrenamiento y el conjunto de prueba.

Elección de modelo (Model selection)

Existen diferentes algoritmos para diferentes tipos de pruebas, en el caso de estudio de este trabajo se realizará una prueba de clasificación y se hablará de algunos ejemplos de estos modelos posteriormente. En este paso se analizarán los datos que se tienen y el problema a solucionar, para elegir un modelo que funcione con ello. Se debe elegir el adecuado para que el modelo pueda predecir correctamente.

Entrenar el modelo (Train the model)

Una vez que se tienen los datos en óptimas condiciones ya se pueden utilizar para poder entrenar el modelo elegido, el objetivo del entrenamiento es responder una pregunta o hacer una predicción correctamente lo más a menudo posible. En este paso el modelo va a encontrar las relaciones y patrones en los datos, además de poder estimar de la mejor manera.

Hacer predicciones (Make predictions)
El uso de más datos (conjunto de prueba) que, hasta este punto, han sido retenidos del modelo (y para los que se conocen las etiquetas), se utilizan para probar el modelo, una mejor aproximación de cómo funcionará el modelo en el mundo real.

Evaluación del modelo (Evaluate the model)

Para saber si el modelo se entrenó correctamente o saber si se eligió el modelo adecuado para el problema se utilizará una métrica o combinación de métricas para "medir" el rendimiento del modelo. Se tiene que probar el modelo contra datos no vistos previamente, estos datos no vistos están destinados a ser algo representativos del rendimiento del modelo en el mundo real.

Podría decirse que los modelos de Machine Learning tienen un único propósito, para generalizar correctamente (la generalización es la capacidad del modelo de proporcionar resultados sensibles a conjuntos de datos que nunca antes había visto), una vez que se ha diseñado el modelo, se dice que es un buen modelo de aprendizaje automático, si generaliza los datos de entrada que no conoce de manera adecuada. Esto ayuda a hacer predicciones en los datos futuros, ahora, suponga que se quiere verificar que tan bien el modelo de aprendizaje automático aprende y generaliza a los nuevos datos, existe una terminología en el aprendizaje automático cuando hablamos de que también aprende y generaliza los nuevos datos, Sobreajuste (overfitting) y Falta de ajuste (underfitting), estos son los principales responsables del bajo rendimiento de los algoritmos de aprendizaje automático.

1.6. Sobreajuste (Overfitting)

Overfitting se refiere a cuando un modelo actúa muy bien con los datos de entrenamiento, pero a la hora de generalizar nuevos datos no los predice correctamente. El overfitting ocurre cuando un modelo aprende los detalles y el ruido en los datos de entrenamiento, esto en medida impacta negativamente el rendimiento del modelo en los nuevos datos. Esto significa que el ruido en los datos de entrenamiento es recogida y aprendida como conceptos por el modelo, el problema es que estos conceptos no se aplican a los datos nuevos y afectan negativamente la capacidad de los modelos para generalizar. El sobreajuste es más probable con modelos no paramétricos y no lineales que tienen más flexibilidad al aprender una función objetivo.

1.7. Falta de ajuste (Underfitting)

El underfitting se refiere a cuando un modelo no actúa bien ni siquiera en los datos de entrenamiento, ni tampoco generaliza los datos nuevos. Un modelo de aprendizaje automático con underfitting no es un modelo adecuado y será obvio, ya que tendrá un bajo rendimiento en los datos de entrenamiento, esto a menudo no se discute, ya que es fácil de detectar, dada una buena métrica de rendimiento.
La solución es seguir adelante y probar algoritmos alternativos, sin embargo, proporciona un buen contraste con el problema del overfitting. Un modelo que generaliza bien es uno que no es subajustado ni sobreajustado, puede que esto no tenga mucho sentido todavía, pero se necesita tener esto en cuenta. Esta es la meta, pero es muy difícil hacerlo en la práctica.

Este primer capítulo servirá como una introducción para conocer los alcances del Machine Learning, los elementos que se necesitan para crear un buen modelo y posteriormente en los siguientes capítulos se verán más a detalle con un ejemplo, cómo también los modelos que se utilizarán para el caso de estudio.

Capítulo 2

Preliminares del Credit Scoring

Este capítulo ayudará a entender el contexto y la importancia de establecer un método en la toma de decisiones a la hora de conceder cualquier tipo de crédito por parte de algún individuo u organización, con un enfoque en las tarjetas de crédito.

2.1. Crédito

Etimológicamente la palabra crédito viene del latín *credere* que significa confiar, según la Real Academia Española (RAE) se define a un crédito como: "Un crédito es una cantidad de dinero u otro medio de pago que una persona presta a otro bajo determinadas condiciones de devolución".
Cuando una persona tiene la necesidad de comprar algún bien o servicio y no cuenta a su disposición con el capital para hacerlo, este puede recurrir a un prestamista (persona que se dedica a prestar dinero cobrando por ello un interés) y así poder ir pagando la deuda en pagos más pequeños que el costo total, de esta manera se puede ver al crédito como un "compra ahora y paga después".
Cuando se da un crédito, la persona que pidió el crédito debe pagar por el otorgamiento de este, es decir, los intereses, los cuales son la ganancia que la persona o institución haya limitado en el uso de su dinero, por lo tanto, se estipulan ciertas condiciones que hacen posible llevar a cabo el préstamo, estas condiciones tienen que ver con el plazo para terminar de pagar la deuda, montos a pagar y el tipo de interés. Ante la solicitud de un crédito, el prestamista tiene la necesidad de evaluar las características de su cliente para poder determinar si otorgar o no el crédito, ya que en caso de que este no pague su deuda le puede ocasionar pérdidas importantes.

2.2. Impagos (Default Payment)

Un impago o default es el incumplimiento de las obligaciones legales o condiciones de un préstamo, el incumplimiento es básicamente no realizar un pago de una deuda en tiempo y forma. El mayor problema de los bancos es determinar un mal crédito, es decir, que sus clientes tengan muchos impagos, porque estos pueden causar serios problemas en el futuro, esto puede dirigir a perdida de capital y si esto creciera se puede dirigir hacia una bancarrota.

2.3. Clientes morosos

Para una empresa es importante saber si un crédito otorgado será una buena inversión o no, para esto se tienen que cuidar de las personas que no cumplan con sus obligaciones a la hora de pagar sus deudas ya que esto en un futuro puede ocasionar pérdidas importantes al aceptar este tipo de clientes. Los clientes morosos son aquellas personas que exceden del plazo de tiempo de crédito otorgado para pagar sus deudas, son las personas que no cumplen con su deuda en tiempo y forma.
Cuando los clientes no cumplen con la obligación adquirida, el prestamista los empieza a clasificar como morosos, según sus políticas. En algunos casos se registran este tipo de clientes con el fin de que sirva como referencia del comportamiento de los clientes en cada uno de sus créditos adquiridos. La consecuencia de tener muchos clientes morosos, en su mayoría, puede llegar a tener un nivel de riesgo alto a la institución y en algunos casos puede llegar a la ruina. Para prevenir dicho suceso se hace uso de la probabilidad, estadística y tecnología para construir modelos matemáticos, los cuales pueden obtener una estimación de la probabilidad de incumplimiento y/o calcular un "score" que vaya segmentando las solicitudes y así tomar decisiones correctas en el otorgamiento de este.
La clasificación de los clientes depende de la empresa, esta clasificación ilustra una clasificación general que tienen las instituciones bancarias cuando estas clasifican en función de su tiempo de mora:

- ***Cliente bueno***. - Generalmente no tiene adeudo con el banco y siempre o casi siempre está al corriente de sus pagos o por lo menos pagan antes de los 60 días de retraso.

- ***Cliente intermedio***. – Regularmente necesitan ser observados por más tiempo para poder detectar la tendencia del comportamiento en sus pagos y así llegar a una clasificación correcta.

- ***Cliente malo***. – Se encuentran personas que generan pérdidas económicas al banco. Estos clientes no pagaron su cuenta a pesar de usar técnicas de cobranza, son mayor a 89 días.

  En México la información de este tipo de clientes es enviada al buró de crédito, es enviada mensualmente y se lleva el registro hasta por un periodo de 24 meses.

2.4. Tarjetas de crédito (Credit Cards)

La definición de tarjeta de crédito es: "Tarjeta emitida por una entidad bancaria que permite realizar ciertas operaciones desde un cajero automático y la compra de bienes y servicios a crédito, generalmente es de plástico y tiene un microchip o banda magnética". Las tarjetas de crédito son uno de los productos que las instituciones financieras poseen. Estas son, el medio de pago que un banco otorga a las personas para que puedan realizar el pago de bienes y servicios de forma inmediata sin la necesidad de utilizar dinero en efectivo y que posteriormente se liquidará, a veces en pagos, con el correspondiente pago de interés.
Actualmente es muy utilizada por la sociedad y se ha convertido en un instrumento con mucha demanda por lo que se tiene que tener mayor control con ellas, en el día a día puede ayudar a cubrir las necesidades que no se pueden pagar en efectivo al momento de

necesitarlas y poder pagarlas después. Todo esto es posible siempre y cuando el cliente pague los respectivos intereses por haber utilizado el dinero que pertenece al banco, su uso puede ir desde cosas simples como ropa y alimentos, como hasta para poder pagar viajes y hoteles, todo depende del límite de crédito.

Como un buen instrumento financiero, las tarjetas de crédito tienen sus características para su buen uso y funcionamiento tanto para el cliente como para el emisor de estas.

- **Línea de crédito**: El banco como emisor de crédito concede al cliente, mediante al acuerdo establecido, una línea de crédito revolvente, es decir, una vez pagando la deuda se vuelve a obtener el dinero prestado para, volver a ocuparlo, hasta un límite de crédito determinado por la misma institución.
- **Período**: Es la fecha de inicio y fin que comprende el ciclo en el cual puede ocuparse la tarjeta. Regularmente oscila entre los 30 y 31 días.
- **Fecha de corte**: Es el día del mes en que termina e inicia un nuevo periodo de registro del uso del plástico
- **Fecha límite de pago**: Es la fecha en la cual se tiene que realizar el pago para no caer en morosidad. Generalmente son 20 días naturales a partir de la fecha de corte.
- **Pago mínimo**: Es la cantidad mínima a pagar al banco para no caer en morosidad.
- **Pago para no generar intereses**: Es un monto mínimo que se debe liquidar puntualmente y así evitar el pago de interés (incluye los pagos mensuales correspondientes a promociones a meses sin intereses).
- **Costo Anual Total (CAT)**: Se tiene una medida estandarizada del costo del financiamiento, expresado en términos porcentuales anuales que incorpora la totalidad de los costos y gastos inherentes de los créditos que otorgan las instituciones. Es decir, es un indicador que incorpora en una sola cifra todos los costos relevantes, (intereses, las comisiones y el plazo de pago), en que se incurre al contratar un crédito.

Desafortunadamente si no se llegan a realizar los pagos de la tarjeta de crédito, se harán recargos, hasta llegar al tope crediticio, los pagos atrasados se agregarán al informe de crédito a medida que llegan a 30, 60, 90 y 120 días de retraso. Si una persona tiene un impago en su tarjeta de crédito los acreedores pueden evaluar las tasas de interés al valor predeterminado (o tasa de penalización) o disminuir la línea de crédito. En caso grave de morosidad, el emisor de la tarjeta puede tomar medidas legales para hacer cumplir el pago o embargar salarios. Estos pagos atrasados disminuyen el puntaje de crédito y podrían arruinar la capacidad de obtener un préstamo, tarjeta de crédito o incluso un trabajo en el futuro. La tasa de seguro también podría aumentar como resultado de morosidad en tarjetas de crédito, 6 meses (180 días) después de que se dejen de hacer los pagos a la tarjeta de crédito, la cuenta será cargada y en este caso la compañía de la tarjeta cancela su deuda como una pérdida comercial. Si bien, ya no se debe dinero, se tiene una grave mancha en el informe de crédito que permanecerá ahí por los próximos 7 años, alertando a todos que una vez se incumplió una obligación crediticia.

2.5. Riesgo Crediticio (Credit Risk)

La palabra riesgo viene del latín *Riscare* que significa "atreverse", en finanzas el concepto riesgo se relaciona con la probabilidad de que ocurra un escenario en el cual se tengan

pérdidas para los participantes.
El riesgo crediticio es la probabilidad de sufrir una pérdida causada por un impago, es el riesgo que se tiene a la pérdida de capital debido a la falta de pago en tiempo y forma. Básicamente, el riesgo crediticio es el riesgo que tiene una persona o institución de que no se le pague el dinero que presto.

2.6. Gestión de Riesgo (Risk Managment)

La gestión de riesgo es un enfoque estructurado para manejar la incertidumbre relativa a una amenaza a través de una secuencia de actividades humanas que incluyen la identificación, el análisis y la evaluación de riesgo (Credit Risk).
El problema principal de todo prestamista es diferenciar entre un "buen" cliente y un "mal" cliente para otorgar un crédito. Esta diferenciación es posible usando métodos de credit scoring. La evaluación crediticia es uno de los procesos más cruciales en las decisiones de gestión crediticia de los bancos, este proceso incluye recopilar, analizar y clasificar diferentes elementos, y variables crediticias para evaluar las decisiones crediticias. La calidad de los préstamos bancarios es el determinante clave de la competencia, la supervivencia y la rentabilidad. Uno de los kits más importantes, para clasificar a los clientes de un banco, como parte del proceso de evaluación de crédito para reducir el riesgo actual y esperado que un cliente tenga mal crédito, es la calificación crediticia.

2.7. Probabilidad de incumplimiento

La probabilidad de incumplimiento es una medida de calificación crediticia que se otorga internamente a un cliente o a un contrato con el objetivo de estimar su probabilidad de incumplimiento a un año vista. El proceso de obtención de la probabilidad de incumplimiento se realiza a través de herramientas de scoring y de rating. Esta medida explica que tan probable es que un acreditado deje de cumplir con sus obligaciones contractuales.

2.8. Solvencia Crediticia

Una referencia a la probabilidad de que una de las partes intervinientes en un contrato incumpla sus obligaciones. Cuanto mayor es la probabilidad de incumplimiento, menos solvencia crediticia tiene la parte. La calidad crediticia es la capacidad que posee una entidad emisora de deuda para hacer frente a sus compromisos de pago futuros, tanto en tiempo como en forma.

2.9. Puntaje de crédito (Credit Scoring)

Desde que el primer hombre de las cavernas, Garf, le pidió a su vecino Gug que le prestara un poco de madera para poder hacer fuego, Gug el primer prestamista tuvo que considerar si el préstamo que realizará le será rembolsado, claro que Garf le dijo "prometo hacer fuego con madera, y devolver más madera y carne cocida mañana", pero ¿podría confiar en Garf? ¿Qué pasa si Garf huye a una nueva cueva diferente y nunca vuelve a ver su madera?, quizás Gug podría preguntar a alguno de sus compañeros si Garf es confiable, y así es como funcionaron las cosas durante las siguientes docenas de miles de años.

A pesar de que la historia de los créditos es bastante antigua, desde hace más de 5000 años en la antigua Babilonia, los modelos de credit scoring son relativamente nuevos. Las técnicas de credit scoring se comenzaron a aplicar a partir de 1960 en los Estados Unidos para determinar si los individuos que solicitaban créditos podrían ser sujetos de este utilizando una forma automatizada. Estas técnicas se comenzaron a usar debido al gran volumen de solicitudes de crédito, especialmente de tarjetas de crédito a procesar que hacían a las técnicas tradicionales de evaluación de crédito poco eficientes. El scoring es un método que ha venido evolucionando a lo largo de los años y el interés en su aplicación se basa en calificar a individuos de cualquier población con información propia de cada entidad, posibilitando la aplicación en cualquier mercado. Las entidades bancarias confeccionaron una clasificación para determinar el valor del riesgo en el que incurren cuando conceden un crédito. La clasificación elaborada por las entidades bancarias se denomina credit scoring.

Anderson en 2007 (c.f [2]), sugirió que para definir al Credit Scoring, el término debe dividirse en dos componentes, "credit" y "scoring". En primer lugar, la palabra crédito que viene de "comprar ahora y paga después", en segundo lugar, la palabra puntuación se refiere a "el uso de una herramienta numérica para clasificar los casos en orden de acuerdo a alguna métrica para discriminar entre ellos y garantizar decisiones objetivas y consistentes", en consecuencia, la palabra credit scoring puede definirse como "el uso de modelos estadísticos para transformar datos relevantes en medidas numéricas que guían decisiones crediticias".

Un credit scoring es un sistema de calificación de créditos que intenta automatizar la toma de decisiones en cuanto a conceder o no una determinada operación de riesgo normalmente de crédito. Es una técnica de la minería de datos que, gracias al avance computacional, se puede trabajar de manera más eficiente con el banco de datos que puede poseer alguna institución, donde el objetivo es hallar patrones y relaciones entre la información del solicitante (edad, trabajo, etc.), con el fin de clasificar, siendo este caso una evaluación crediticia, para diferenciar entre clientes cumplidos o incumplidos en cuanto a obligaciones de pago. Estos modelos requieren de dos elementos fundamentales:
La información histórica, las instituciones cuentan con una base de datos donde contiene el comportamiento de sus clientes, y el análisis estadístico, mediante estos algoritmos se encuentra el comportamiento de los clientes y se determinan las probabilidades de ocurrencia en eventos futuros.

Ventajas del Credit Scoring

- Cuantifica el riesgo como una probabilidad: Estos modelos calculan la probabilidad de que un cliente resulte en impago.

- Es explícito: Se conoce y se puede informar el proceso exacto que se utilizó para el pronóstico del riesgo.

- Considera diversos factores: A diferencia de un analista de riesgo, que es la persona que se encarga manualmente de evaluar las solicitudes, los modelos de credit scoring pueden considerar treinta o cincuenta características simultáneamente.

- Puede probarse antes de usarse: puede hacer comparaciones entre el riesgo estimado y el riesgo observado en la práctica, mostrando como habría funcionado el scoring si se hubiera aplicado al momento de las solicitudes de préstamos vigentes.

- Rapidez: Puede evaluar miles de solicitudes rápida e imparcialmente, esto ayuda a las dos partes, al solicitante al darle una pronta respuesta y al banco eliminando costos.

Desventajas del Credit Scoring

- El scoring es una herramienta muy eficaz pero su mal uso puede resultar contraproducente, sí, el prestamista puede reducir costos de evaluar cientos de solicitudes mediante el uso de puntajes, pero si los modelos no son precisos estos ahorros se consumirán con los prestamos mal realizados.

- El scoring supone que el futuro será como el pasado, ya que estos trabajan con información historia se entrenan en base al pasado, para esto se necesita que cada cierto tiempo estos se ajusten con los nuevos datos.

- Existe una posible discriminación indirecta, esto puede pasar ya que ciertas variables como la raza o el sexo estadísticamente tienen cierto peso al decidir si otorgar o no un crédito.

- Funciona con probabilidades no con certezas.

- Es susceptible al mal uso.

Una de las principales limitaciones que se presentan en la clasificación, es la no aleatoriedad en las muestras. En su mayoría los trabajos realizados son con muestras truncadas, es decir, dado que estas muestras se forman sólo con los créditos que son aceptados, esto debido a la imposibilidad de obtener datos de créditos no concedidos. Para que el modelo sea preciso los datos con los cuales el sistema se basa necesitan ser una muestra rica en ambos tipos de desempeños, clientes morosos y no morosos. Uno de los mayores problemas con los que cuenta el scoring es la escasez de información pública disponible (muchas veces desactualizada o incompleta) esto debido a su confidencialidad que las instituciones deben mantener con sus solicitantes y clientes.

Lo que hace básicamente el scoring es calcular la probabilidad de que un crédito o préstamo para un cliente en concreto resulte en un impago, si esa probabilidad es menor que un límite puesto por el banco entonces el crédito será aprobado. Estos modelos se pueden ver como una comparación de clientes, cuando una persona solicita un crédito esta es comparada con los clientes actuales, si esta persona tiene las características de un cliente que paga, lo más posible es que este sea aceptado, mientras en caso contrario, si la persona tiene características de una persona que no paga y le causa pérdidas a la empresa, lo más probable es que este sea rechazado. El objetivo de los modelos es asignar prestamos ya sea a buenos clientes o predecir a los malos créditos, por lo tanto, los problemas de credit scoring está relacionado con problemas de clasificación antes mencionados.
Se dice que un buen scoring mejora la tasa de morosidad frente a las decisiones humanas. Ayudan a evaluar el riesgo en los préstamos. La calificación crediticia es una evaluación confiable de la solvencia crediticia de una persona ya que se basa en datos reales. Los modelos de credit scoring precisos y predictivos ayudan a maximizar el rendimiento ajustado al riesgo de una institución financiera. Sin embargo, los mercados y el comportamiento del consumidor pueden cambiar rápidamente durante los ciclos económicos, por esta razón es importante que no sólo se creen los modelos sino también después de un tiempo ajustarlos y validarlos correctamente. Las agencias seleccionan las características estadísticas que se

encuentran en los patrones de pago de crédito de una persona, las analizan y obtienen una puntuación de crédito.
Cualesquiera que sean las técnicas utilizadas, es fundamental que haya una gran muestra de clientes anteriores con los detalles de sus aplicaciones y patrones de comportamiento. La mayoría de estas técnicas usan esta muestra para identificar conexiones entre las características de los consumidores y su historia subsecuente.
El Credit Scoring funciona, ya que este aprende de los errores, es decir, analiza que operaciones han ido peor y trabaja en consecuencia a través de análisis estadísticos. Un buen scoring es mejor mientras más se usa.

2.10. Punto de corte (Cutoff point)

Un puntaje de corte es el puntaje de crédito más bajo posible que se puede tener y aún así calificar para un préstamo. Los puntajes de corte varían ampliamente según el tipo de préstamo solicitado y el prestamista. El puntaje de corte para tarjetas de crédito y otros prestamos de alto interés tenderá a ser de menor valor. Los puntajes de crédito, a veces conocidos como puntajes FICO, se basan en la información proporcionada por la empresa de análisis de datos del mismo nombre.
Existen cuestionamientos sobre cuál es el puntaje de corte correcto para la evaluación de crédito. El puntaje es fundamental para la utilidad y valor de los modelos de credit scoring, ya que dependiendo de este punto predeterminado un cliente puede clasificarse como aceptado o rechazados.
En general no hay un corte de punto óptimo, este varia del entorno en el que está, ya sea el banco o el país, en algunos bancos se quieren aceptar más clientes por lo que el puntaje es más bajo de lo esperado y viceversa, cuando no quieren admitir demasiados clientes. El puntaje que se elegirá en el caso de estudio será de 0.5, el cual es el que está por default en los modelos estadísticos.

En el siguiente capítulo se verán los modelos estadísticos que se utilizarán para la construcción de un modelo de Credit Scoring en el caso de estudio.

Capítulo 3

Modelos Estadísticos

Los modelos de Machine Learning se pueden ver de la siguiente manera:

$$P = f(x_1, x_2, ..., x_p) + \epsilon, \qquad (3.1)$$

donde x_i son las variables explicativas, ϵ es el error agregado por problemas de la vida real, $f()$ la función que determina la relación entre las características y la variable de salida, y enfocándose al problema de clasificación P es la probabilidad de permanecer a cierta clase o categoría. El objetivo principal de los problemas de clasificación se centra en encontrar la relación que permita clasificar con la mayor exactitud posible las observaciones de la muestra, y que los errores de predicción sean los menores posibles. Dependiendo si la relación $f()$ es conocida o no, o si se trata de un modelo paramétrico o no paramétrico.

Los modelos paramétricos se basan en una función de distribución o clasificación conocida, es decir, que la relación $f()$ se establece a priori, de modo que el problema aquí radica en estimar los parámetros que mejor se ajusten a las observaciones de la muestra. Estos modelos son muy efectivos cuando los datos siguen la distribución propuesta, pero son muy sensibles a las violaciones de las hipótesis de partida.
Los modelos no paramétricos tratan de aproximar la función de clasificación a través del uso de formas funcionales flexibles, sin suponer ninguna estructura a priori. Por lo tanto, son más flexibles en las restricciones por lo que se considera que son más fáciles de aplicar que los modelos paramétricos, ya que permiten reconstruir la función de clasificación. Contrario al caso de los modelos paramétricos, no buscan estimar los parámetros de la función, sino que mediante formas funcionales buscan aproximarse a una función objetivo.

A continuación, se explicarán a detalle los modelos que se utilizaran posteriormente para el análisis en el caso de estudio, los cuales son Regresión Logística, Árboles de Decisión y Redes neuronales, de los cuales 2 de ellos son paramétricos y uno no paramétrico.

3.1. Regresión Logística

La regresión logística es un método estadístico que es utilizado para la clasificación binaria, donde se busca estimar la probabilidad de que una observación pertenezca a cierta clase con una probabilidad entre 0 y 1. Generalmente los resultados binarios provienen de una relación no lineal, entre la variable respuesta y las variables independientes del modelo, es de interés

estudiar la relación entre una o más variables independientes o explicativas $x_1, ..., x_p$ y la variable independiente y, de manera que a cada elemento del conjunto X le corresponde un único elemento del conjunto Y.

El modelo que asegurará que las probabilidades de respuesta estimadas estén estrictamente entre cero y uno es el modelo *Logit*, el cual está basado en la función de distribución logística f:

$$f(x) = \frac{1}{1+e^{-x}}, \qquad (3.2)$$

donde se tomará $x = \beta \cdot \underline{x} = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p$,

con $\beta = (\beta_0, ..., \beta_p)$ y $\underline{x} = (1, x_1, ..., x_p)$.

Esta función es estrictamente creciente, y cuando x tiende a $-\infty$ se tiene que $f(x) \to 0$, por otro lado si se toma x cuando tiende a ∞ entonces $f(x) \to 1$.

Otra forma en la cual suele encontrarse la ecuación (3.2) es:

$$f(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x} \cdot \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x}, \qquad (3.3)$$

donde su gráfica es una curva S o sigmoidea, y la cual tiene un único punto de inflexión en el que cambia la concavidad y la rapidez del crecimiento, este punto se puede asociar con el punto de corte en los modelos de Credit Scoring. Observar Figura (3.1).

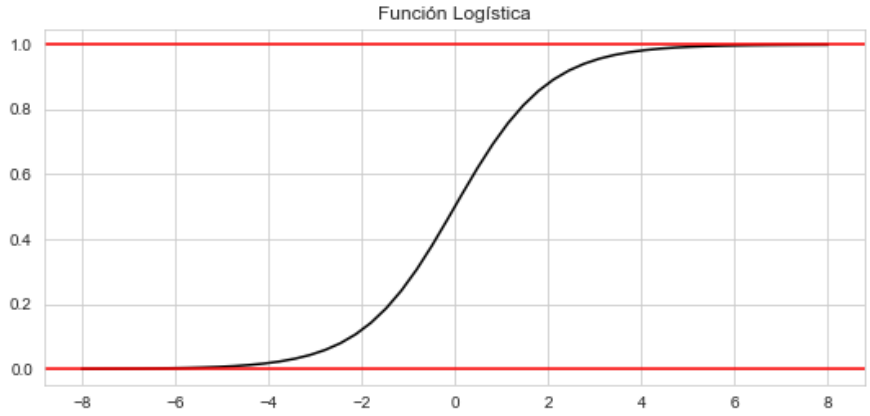

Figura 3.1: Grafica de la función logística.

La función logística cuenta con una función inversa llamada *transformación logit*, la cual es importante para el desarrollo de la regresión. La *transformación logit* que proviene de la función logística, es una transformación que tiene ventajas por admitir variables categóricas, además de tomar valores entre 0 y 1 para la variable dependiente, lo cual se puede asociar a una probabilidad de incumplimiento.

Esta transformación se obtiene mediante un despeje de variables de la ecuación (3.2).

$$f(x) = \frac{1}{1+e^{-x}}$$

$$\Rightarrow 1 + e^{-x} = \frac{1}{f(x)}$$

$$\Rightarrow e^{-x} = \frac{1}{f(x)} - 1$$

$$\Rightarrow e^{-x} = \frac{1-f(x)}{f(x)}$$

$$\Rightarrow e^{x} = \frac{f(x)}{1-f(x)}$$

$$\Rightarrow x = ln(\frac{f(x)}{1-f(x)})$$

$$\therefore logit[f(x)] = ln(\frac{f(x)}{1-f(x)}) = x = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p. \qquad (3.4)$$

Al realizar esta trasformación, el *logit* tiene varias propiedades de un modelo de regresión lineal. El modelo logit es lineal en sus parámetros, puede ser continua, su dominio puede estar en un rango de $(-\infty, \infty)$, y codominio en el intervalo (0,1), teniendo como único caso posible para que $\frac{f(x)}{1-f(x)} > 0$ que el numerador y el denominador sean positivos, tenemos:

1) $f(x) > 0$.

2) $1 > f(x) \Rightarrow 1 - f(x) > 0$.

Para la función logit y la función logística, cualquier $f(x)$ se encuentra dentro del intervalo (0,1). Con base en esto, se define la regresión, donde Y da a $f(x)$ una interpretación de probabilidad:

$$P(y_i = 1|x_i; \beta) = f(x) \text{ y } P(y_i = 0|x_i; \beta) = 1 - f(x), \qquad (3.5)$$

donde se pueden tomar uno de dos valores posibles, $y = 1$ con probabilidad $f(x)$, y si $y = 0$ con probabilidad $1 - f(x)$.

3.1.1. Estimación de parámetros del modelo de regresión

Considerando la ecuación (3.2) se debe desarrollar un método para estimar los parámetros β_j, con $j = 0, ..., p$, los cuales representan la relación entre las variables y los diversos pesos de cada característica a partir de una muestra de n observaciones $(\underline{Y}, \mathcal{X})$, donde $(y_i, \underline{x_i})$, $i = 1, ..., n$, son las características del i-ésimo individuo de la muestra.
En este caso, donde la variable respuesta es dicotómica, se usa el método de máxima verosimilitud, el cual se utiliza para el ajuste de modelos y estimación de parámetros. Con el Estimador de Máxima Verosimilitud (EMV), se puede inferir sobre uno o más parámetros $\beta_0, \beta_1, \beta_2, ..., \beta_p$ basándose en las n observaciones. Se busca el valor estimado que maximiza la Función de Verosimilitud.

Para la función logística $f(x)$, se consideran a las variables y como un evento Bernoulli, donde cada elemento observado tiene la posibilidad de ser un éxito o un fracaso por lo que su función de densidad es:

$$g_i(y_i) = f(x_i)^{y_i}(1 - f(x_i))^{1-y_i},\ y_i = 0, 1, \qquad (3.6)$$

y dado que las n observaciones son independientes, la función de densidad conjunta de los parámetros desconocidos β_j y las observaciones de la muestra $\mathcal{X}$, o la **función de verosimilitud** de $\underline{Y}$ queda de la siguiente manera:

$$L(\beta) = g_1(y_1) \cdot g_2(y_2) \cdot \quad \cdots \quad \cdot g_n(y_n) = \prod_{i=1}^{n} g_i(y_i) = \prod_{i=1}^{n} f_i(x_i)^{y_i}(1 - f(x_i))^{1-y_i}$$
$$= \prod_{i=1}^{n} \left(\frac{e^{\beta_0+\beta_1 x_1+\ldots+\beta_p x_p}}{1 + e^{\beta_0+\beta_1 x_1+\ldots+\beta_p x_p}}\right)^{y_i} \left(1 - \frac{e^{\beta_0+\beta_1 x_1+\ldots+\beta_p x_p}}{1 + e^{\beta_0+\beta_1 x_1+\ldots+\beta_p x_p}}\right)^{1-y_i}. \qquad (3.7)$$

Para obtener los valores que maximizan esta función comúnmente se hace uso de la función logaritmo con lo que es posible facilitar tal cálculo. Esto es:

$$l(\beta) = ln(L(\beta)) = \sum_{i=1}^{n} [y_i ln(f(x_i)) + (1 - y_i) ln(1 - f(x_i))]. \qquad (3.8)$$

Para encontrar el valor del vector $\beta = (\beta_0, \beta_1, ..., \beta_p)$ que maximiza $l(\beta)$, se deriva $l(\beta)$ con respecto a β:

$$\frac{\partial l(\beta)}{\partial \beta} = \sum_{i=1}^{n} \left[\frac{y_i}{f(\beta \cdot \underline{x})} - \frac{1 - y_i}{1 - f(\beta \cdot \underline{x})}\right] \frac{\partial f(\beta \cdot \underline{x})}{\partial \beta}, \qquad (3.9)$$

después se deriva $f(x)$ respecto a β:

$\frac{\partial f(\beta \cdot \underline{x})}{\partial \beta} = \frac{\partial f(\beta \cdot \underline{x})}{\partial \beta \cdot \underline{x}} \frac{\partial \beta \cdot \underline{x}}{\partial \beta}$,

$\frac{\partial f(\beta \cdot \underline{x})}{\partial \beta \cdot \underline{x}} = \frac{-e^{-(\beta \cdot \underline{x})}(-1)}{(1+e^{-(\beta \cdot \underline{x})})^2} = \frac{e^{-(\beta \cdot \underline{x})}}{(1+e^{-(\beta \cdot \underline{x})})^2}$

$= \frac{e^{-(\beta \cdot \underline{x})}}{(1+e^{-(\beta \cdot \underline{x})})} \frac{1}{(1+e^{-(\beta \cdot \underline{x})})}$

$= [1 - \frac{1}{(1+e^{-(\beta \cdot \underline{x})})}] \frac{1}{(1+e^{-(\beta \cdot \underline{x})})}$

$= [1 - f(\beta \cdot \underline{x})] f(\beta \cdot \underline{x}). \qquad (3.10)$

Ademas en $\frac{\partial \beta \cdot \underline{x}}{\partial \beta}$ para cada $\beta_j \neq \beta_0$, con $j = 0, 1, ..., p$, se tiene que $\frac{\partial \beta \cdot \underline{x}}{\partial \beta_j} = x_j$ y $\frac{\partial \beta \cdot \underline{x}}{\partial \beta_0} = 1$. De (3.9) y (3.10) se tienen las derivadas parciales para β_0 y los p coeficientes restantes β_j, estas ecuaciones se igualan a 0 para obtener los estimadores, y se tiene:

$$\frac{\partial l(\beta)}{\partial \beta_j} = \sum_{i=1}^{n} [\frac{y_i}{f(\beta \cdot \underline{x})} - \frac{1-y_i}{1-f(\beta \cdot \underline{x})}][1-f(\beta \cdot \underline{x})][f(\beta \cdot \underline{x})]\underline{x}.$$

$$= \sum_{i=1}^{n} [y_i - f(\beta \cdot \underline{x})]\underline{x}; \qquad (3.11)$$

$$\frac{\partial l(\beta)}{\partial \beta_0} = \sum_{i=1}^{n} [y_i - f(\beta \cdot \underline{x})]. \qquad (3.12)$$

Al tratarse de ecuaciones no lineales, las cuales son más complejas a mayor cantidad de elementos en la muestra y características a evaluar, se requiere de un proceso robusto que implica una gran cantidad de operaciones para su solución, pero en la actualidad existen diversos paquetes estadísticos que facilitan el cálculo de los parámetros.
Los valores obtenidos con la solución de las ecuaciones anteriores, se llaman Estimadores de Máxima Verosimilitud (EMV) y son denotados por $\hat{\beta}$, que son las soluciones de las ecuaciones de verosimilitud, de tal forma que, al evaluar el Hessiano asociado a la función de verosimilitud en $\hat{\beta}$ resulta en una matriz definida negativa.

Es importante verificar la significancia estadística de los parámetros obtenidos, existen diversos métodos para esto, siendo el estadístico de Wald uno de los más usados.

El estadístico de Wald: por definición contrasta la hipótesis de que un coeficiente es distinto de 0, y sigue una distribución normal de media 0 y varianza 1 (Distribución Normal Estándar).

$$H_0 : \beta_i = 0 \qquad vs \qquad H_1 : \beta_i \neq 0.$$

Este estadístico se obtiene dividiendo el valor estimado del coeficiente $\hat{\beta}_i$ entre su correspondiente error estándar $\hat{\sigma}(\beta_i)$.

$$Wald = \frac{\hat{\beta}_i}{\hat{\sigma}(\beta_i)}. \qquad (3.13)$$

La obtención de significación indica que dicho coeficiente es diferente de 0 y merece la pena su conservación en el modelo, la ausencia de significación implica que el modelo sin la variable independiente no empeora respecto al modelo completo (es decir, no importa su presencia o su ausencia) y dicha variable debería ser eliminada del modelo ya que aporta nada. En modelos con errores grandes estándar, el estadístico de Wald puede proporcionar falsas ausencias de significación (es decir, se incrementa el error tipo II, cuando H_0 es falsa y no se rechaza, se comete un error de tipo II).

3.1.2. Interpretación de los parámetros

Para poder interpretar los parámetros obtenidos, se escribe la ecuación en términos de odds o momios. El odds de un evento se define como la razón de probabilidad de que ocurra ese evento a la probabilidad de que no ocurra, siendo este cociente de probabilidades de las estimaciones más comunes que se usan para la regresión:

$$odds = \frac{P(y=1|x)}{P(y=0|x)} = \frac{P(y=1|x)}{1-P(y=1|x)} = \frac{f(\hat{X};\hat{\beta})}{1-f(\hat{X};\beta)} \ . \qquad (3.14)$$

Estos cocientes, cuentan el número de veces que será más probable que ocurra un éxito del evento correspondiente con cada variable i, puede tomar valores entre $[0, \infty)$, valores cercanos a 0 y a ∞ indican poca o muchas probabilidades de un **impago** respectivamente.

La razón de momios (Odds Ratios) sirve para evaluar cuantitativamente el impacto de cambiar el valor de una variable; estos factores de cambio requieren de dos modelos para el análisis de cada variable i, uno que contenga todas las variables ($modA$) y otro que no tenga la variable $i(modB)$, la rázon de momios de ambos modelos es:

$$OR = \frac{\frac{f(\hat{X};\hat{\beta}_{modA})}{1-f(\hat{X};\hat{\beta}_{modA})}}{\frac{f(\hat{X};\hat{\beta}_{modB})}{1-f(\hat{X};\hat{\beta}_{modB})}}. \qquad (3.15)$$

Tomando el logaritmo a la razón de momios se tiene:

$$\begin{aligned} ln(OR) &= ln[\frac{\frac{f(\hat{X};\hat{\beta}_{modA})}{1-f(\hat{X};\hat{\beta}_{modA})}}{\frac{f(\hat{X};\hat{\beta}_{modB})}{1-f(\hat{X};\hat{\beta}_{modB})}}] = ln[\frac{f(\hat{X};\hat{\beta}_{modA})}{1-f(\hat{X};\hat{\beta}_{modB})}] - ln[\frac{f(\hat{X};\hat{\beta}_{modB})}{1-f(\hat{X};\hat{\beta}_{modB})}] \\ &= \sum_{j=1}^{n} \hat{\beta}_j \cdot \hat{x}_j - \sum_{j=1, j\neq i}^{n} \hat{\beta}_j \cdot \hat{x}_j = \hat{\beta}_i \cdot \hat{x}_i. \qquad (3.16) \end{aligned}$$

Ahora eliminando el logaritmo aplicando la función exponencial en ambos lados:

$$OR = e^{\hat{\beta}_i \cdot \hat{x}_i}. \qquad (3.17)$$

Entonces, se tiene el factor de cambio $e^{\hat{\beta}_i \cdot \hat{x}_i}$, para utilizarlo se ocupan intervalos de u unidades para saber el efecto que tendrá la probabilidad de éxito, se calcula $e^{\hat{\beta}_i \cdot u}$.

El coeficiente estimado asociado con un predictor representa el cambio en la función logística por cada cambio de unidad en el predictor, mientras los demás predictores se mantienen constantes, un cambio de unidad en un factor se refiere a una comparación de un determinado nivel con el nivel de referencia.

3.2. Árboles de Decisión (Decison Tree Methods)

Los árboles de decisión son un método que se utiliza para regresión y para clasificación, estos consisten en segmentar el espacio de los predictores en un número de regiones simples. Para hacer una predicción de una observación dada, usualmente se usa el promedio en el caso de regresión o la mayor clase del conjunto de entrenamiento en la región a la que pertenece para la clasificación, donde el conjunto de reglas para dividir usadas para segmentar el espacio de los predictores puede ser resumido en un árbol. Estos árboles pueden usar variables numéricas y/o categóricas.

Un árbol de decisión se llama así debido a su estructura de árbol similar a un diagrama de flujo donde un nodo interno representa una característica (o atributo), la rama representa una regla de decisión y cada nodo hoja representa el resultado. El nodo superior en un árbol de decisión se conoce como el nodo raíz. Aprende a particionar en función del valor del atributo. Divide el árbol de una manera recursiva llamada partición recursiva. Esta

estructura tipo diagrama de flujo lo ayuda a tomar decisiones. Es una visualización como un diagrama de flujo que imita fácilmente el pensamiento a nivel humano. Es por eso que los árboles de decisión son fáciles de entender e interpretar.

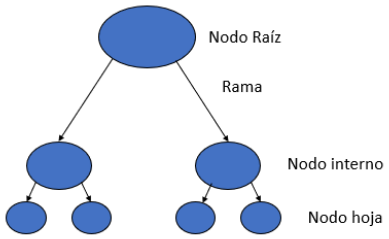

Figura 3.2: Partes de un árbol de decisión.

Para entender mejor este proceso se definirán las partes del árbol:

- **Nodo Raíz(The Root Node)** - Representa a toda la población o muestra y esto divide en dos nodos posteriores, es el primer nodo a evaluar.
- **Rama (Branch)** - Una subsección del árbol de decisión se denomina rama o subárbol. Es la cual une a un nodo con otro.
- **Nodos Internos (Internal nodes)** - Cuando un subnodo se divide en subnodos adicionales, se denomina nodo de decisión.
- **Nodo hoja / terminal (Terminal Nodes or Leaves)** - Los nodos sin hijos (sin división adicional), se denominan nodo hoja o terminal, son llamados hojas en el sentido que están hasta el fondo del árbol.

Este trabajo solo se enfocará a los árboles de clasificación, estos predicen la clase de las observaciones clasificándolas con el árbol desde la raíz hasta algún nodo hoja, con el nodo hoja proporcionando la clasificación de la observación, este enfoque se llama enfoque de arriba hacia abajo. Cada nodo en el árbol actúa como un caso de prueba para algún atributo, y cada borde que desciende de ese nodo corresponde a una de las posibles respuestas al caso de prueba. Este proceso es recursivo y se repite hasta tener un árbol que no pueda mejorar o tenga un mínimo de observaciones.

En general al construir un árbol de clasificación existen dos pasos:

1. Se divide el espacio de predictores, esto es, el conjunto de todos los posibles valores para $X_1, X_2, ..., X_p$ en J regiones diferentes que no estén sobrepuestas, $t_1, t_2, ..., t_J$ (Nodos terminales).
2. Para toda observación que cae en la región t_J, se hará la misma predicción, que simplemente es la clase más común (*most commonly oucurring class*) del conjunto de entrenamiento en la región t_J.

Al construir las regiones $t_1, t_2, ..., t_J$ pueden tener cualquier forma, en teoría, sin embargo, se elige dividir el espacio de los predictores en rectángulos de altas dimensiones, o cajas, por simplicidad y una fácil interpretación de los resultados del modelo predictivo. Al interpretar los resultados de un árbol de clasificación, a menudo se está interesado no solo en la predicción de clase correspondiente a una región de nodo terminal particular, sino también en las proporciones de clase entre las observaciones de entrenamiento que caen en esa región.

El objetivo es encontrar regiones $t_1, t_2, ..., t_J$ que minimicen la *tasa de error (error rate)* de la clasificación, ya que se planea asignar una observación a una región dada por la clase más común del conjunto de entrenamiento en tal región. La clasificación por *error rate* es simplemente la fracción de las observaciones de entrenamiento en la región que no pertenecen a la clase más común:

$$E = 1 - \max_k(\hat{p}_{mk}), \qquad (3.18)$$

donde $\hat{p}_{mk}$ representa la proporción de observaciones de entrenamiento en la *m-ésima* región que son de la *k-ésima* clase. Sin embargo, resulta que a veces el error de clasificación no es lo suficientemente sensible para la creación de árboles, y en la práctica son preferibles otras dos medidas.

El *Índice de Gini (Gini index)* es definido por

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) = 1 - \sum_{k=1}^{K} \hat{p}_{mk}^2. \qquad (3.19)$$

Una medida de la varianza total a través de las K clases. No es difícil ver que el índice de Gini toma un valor pequeño si todos los $\hat{p}_{mk}$s están cerca de 1. Por esta razón, el índice de Gini es tomado como una medida de pureza de los nodos, un valor pequeño indica que en un nodo predominan las observaciones de una sola clase.

Una alternativa al indice de Gini es la entropía (*Cross-entropy*), la entropía se puede ver como una medida de incertidumbre y está dada por:

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} log \hat{p}_{mk}, \qquad (3.20)$$

ya que $0 \leq \hat{p}_{mk} \leq 1$, se sigue que $0 \leq -\hat{p}_{mk} log \hat{p}_{mk}$. Uno puede demostrar que la *cross-entropy* tomará valores cerca de 0 o cerca de 1. Por lo tanto, como el índice de Gini, la entropía tomara valores pequeños si el *m-ésimo* nodo es puro. De hecho, resulta que el índice de Gini y la entropía son numéricamente similares.

Cuando se construye un árbol de clasificación, ya sea con el índice de Gini o con entropía se usa típicamente para evaluar la calidad de una división particular, ya que estos dos enfoques son más sensibles a la pureza del nodo que el enfoque por *error rate*.

Para empezar a construir el árbol se utiliza un enfoque llamado división recursiva binaria (*recursive binary splitting*), este enfoque es de arriba-abajo porque empieza en la cima del árbol y sucesivamente se divide el espacio de los predictores, cada división crea dos nuevas

ramas del árbol, la mejor división es hecha en particular para cada paso donde se elige el predictor que tenga un menor nivel de impureza, eligiendo la división que será mejor para el árbol en el futuro.

El algoritmo optará por la división que mejore la impureza, tomando en consideración ambos nodos descendientes. Se siguen dividiendo los nodos hasta que resulte imposible mejorar realizando más divisiones o el nodo obtenido tenga el valor mínimo de observaciones, en este punto el árbol se considera saturado y está terminado. Cualquiera de los 3 enfoques anteriores puede ser usados cuando se poda el árbol, pero la clasificación por *error rate* es preferible si la exactitud de la predicción del ultima árbol podado es el objetivo.

Una vez obtenidas las regiones $t_1, t_2, ..., t_J$ se puede predecir la respuesta para las observaciones de prueba, simplemente otorgando el valor de la clase más común en cada hoja.

3.2.1. Poda (The Prunning)

En el proceso descrito arriba, cuando se obtiene el árbol final, se pueden producir buenas predicciones en el conjunto de entrenamiento, pero este suele a sufrir de sobreajuste, dirigiéndose a una predicción pobre sobre los datos de prueba. Esto sucede ya que los árboles suelen ser muy complejos. Un árbol pequeño con pocos nodos (esto es, pocas regiones $t_1, ..., t_J$) puede dirigir a una baja varianza y una mejor interpretación del conjunto de prueba y con solamente un poco de *sesgo* (bias).

Por lo tanto, una mejor estrategia es hacer crecer un árbol saturado T_0, y entonces podarlo en orden para obtener un subárbol que no sufra de sobreajuste. ¿Cómo se determina la mejor manera de podar un árbol? intuitivamente, el objetivo es seleccionar un subárbol que lleve a la *prueba de tasa de error (test error rate)* o nivel de impureza más bajo. Dando un subárbol se puede estimar el test error usando cross-validation. Sin embargo, estimar el error de validación cruzada para cada subárbol posible también sería engorroso, ya que hay un número extremadamente grande de posibles subárboles. En cambio, se necesita una forma de seleccionar un pequeño conjunto de subárboles para su consideración. La poda de complejidad de costos (*Cost complexity pruning*), también conocida como poda de enlace más débil, brinda una manera de hacer precisamente esto. En lugar de considerar todos los subárboles posibles, se considerará una secuencia de árboles indexados por un parámetro de ajuste no negativo α.

Uno de los principales problemas que se presentan a la hora de construir árboles es la cantidad de nodos, una gran cantidad de nodos puede llevar al problema del sobre ajuste, una medida de calidad del árbol debe tomar en cuenta tanto la calidad de los nodos terminales, así como el tamaño del árbol, tener en cuenta solo el costo de mala clasificación puede llevar a arboles demasiado grandes. Para evitar este problema existe la función de complejidad de costos para un árbol (*Cost complexity function*) la cual se define de la siguiente manera:

$$R_\alpha(T) = R(T) + \alpha|\dot{T}|. \qquad (3.21)$$

Donde:

- $R(T) = \sum_{t \in \dot{T}} R(t)$ es la medida de rendimiento elegida para el árbol T y es solamente la suma de errores de clasificación errónea en cada hoja t, en el cual $\dot{T}$ es el conjunto de

nodos terminales de T, es una medida de calidad del nodo la cual puede ser cualquiera de las 3 antes mencionadas, pero la más común es la de tasa de error.

- $|\dot{T}|$ es el número de nodos terminales en el árbol T.
- α es un parámetro de complejidad mayor que 0.

Cuantos más nodos terminales tenga el árbol, mayor será la complejidad del árbol, porque tenemos más flexibilidad para dividir el espacio en piezas más pequeñas y, por lo tanto, más posibilidades de ajustar los datos de entrenamiento. También está la cuestión de cuánta importancia poner en el tamaño del árbol, el parámetro de complejidad α ajusta eso, ya que mientras más hojas se tengan, mayor es la penalización. Al final, la medida de complejidad de costos se presenta como una versión penalizada de la tasa de error de mala clasificación. Esta es la función que se debe minimizar al podar el árbol.

Notación

- **Nodo descendiente**.- Un nodo $\underline{t}$ es descendiente de un nodo t si hay una rama que conecta el árbol que va desde t a $\underline{t}$.
- **Subárbol o rama** .- T_t es un subárbol de T, si $t \in T$, T_t consiste de todos los nodos descendientes de t en T.
- **Poda de un árbol**.- La poda de un subárbol T_t de T, consiste en eliminar todos los nodos descendientes de t excepto t, es decir, se elimina T_t de T excepto t, se denota por $T - T_t$.

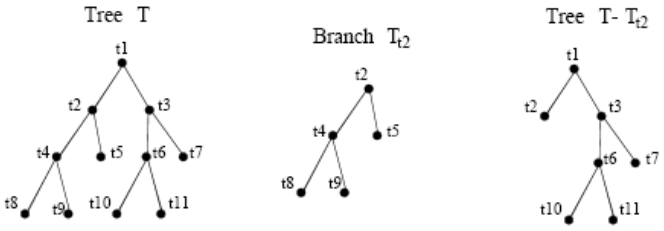

Figura 3.3: Árboles y Subárboles.

En general cuando se da un α, se trata de encontrar un subárbol $T(\alpha)$ que minimice $R_\alpha(T)$, es decir,

$$R_\alpha(T(\alpha)) = \min_{T<T_0} R_\alpha(T). \qquad (3.22)$$

El subárbol que minimiza existe para cualquier α siempre, ya que sólo hay un numero finito de subárboles, si t es un nodo interno de T entonces se garantiza que tiene una tasa de error menor que la tasa de error del subárbol T_t, es decir, $R(t) > R(T_t)$. El método de corte de enlace más débil no solo encuentra el siguiente α que resulta en un subárbol óptimo diferente, sino que encuentra ese subárbol óptimo.

Podando subárboles

Podando un subárbol T_t

A continuación se muestra una variación de la cost-complexity function, es la cost-complexity function cuando se poda un subárbol T_t

$$R\alpha(T - Tt) - R\alpha(T). \qquad (3.23)$$

Así de (3.21) y (3.23)

$$\Rightarrow R(T - T_t) - R(T) + \alpha(|T \dot{-} T_t| - |\dot{T}|) = R(t) - R(T_t) + \alpha(1 - |\dot{T}_t|). \qquad (3.24)$$

Si α se sigue incrementando más, el signo de desigualdad se invertirá, y se tiene la siguiente desigualdad $R_\alpha(T_t) > R_\alpha(t)$. Algunos nodos t pueden alcanzar la igualdad antes que otros. El nodo que logra la igualdad en el α más pequeño se llama el enlace más débil. Si se resuelve la siguiente inecuación $R_\alpha(T_t) < R_\alpha(t)$ se obtiene:

$$\alpha < \frac{R(t)-R(T_t)}{|\dot{T}_t|-1}. \qquad (3.25)$$

El lado derecho es la relación entre la diferencia en las tasas de error y la diferencia en la complejidad, que es positiva porque tanto el numerador como el denominador son positivos.

Se toma

$$g_0 = \frac{R(t)-R(T_t)}{|\dot{T}_t|-1}. \qquad (3.26)$$

El enlace más débil (*weakest link*) $\bar{t_0}$ en T_0 logra el mínimo en g_0, es decir:

$$g_0(\bar{t_0}) = \min_{t \in T_0} g_0(t). \qquad (3.27)$$

Finalmente, el subárbol seleccionado depende de α. Si $\alpha = 0$, entonces, se elegirá el árbol más grande (T_0), porque el término de penalización de complejidad se elimina esencialmente. A medida que α va incrementando, finalmente, se seleccionará el árbol con un solo nodo terminal, es decir, el nodo raíz.

Algoritmo 1: Algoritmo de poda

Se toma el árbol T_0 que es obtenido cuando $\alpha_0 = 0$ minimizando $R(T)$

1. seleccione el nodo $t \in T_0$ que minimice $g_0(t)$.
2. sea $\bar{t_0}$ este nodo y tome $\alpha_1 = g_0(t_0)$ y también $T_1 = T_0 - T_{0,\bar{t_0}}$.

i. seleccione un nodo $t \in T_i$ que minimice $gi(t) = \frac{R(t)-R(T_{i,t})}{|\dot{T}_{i,t}|-1}$, se toma $\bar{t_i}$ como este nodo, sea $\alpha_{i+1} = gi(\bar{t_i})$ y $T_{i+1} = T_i - T_{i,t_i}$.

Las ramas se podan del árbol de forma anidada y predecible, por lo que es fácil obtener la secuencia completa de subárboles en función de α. Para cada valor de α le corresponde un subárbol $T \subset T_0$ tal que al terminar este algoritmo se obtiene una secuencia de subárboles:

$$\{root\} \subset ... \subset T_k \subset ... \subset T_1 \subset T_0.$$

Y una secuencia de parámetros α,

$$... \subset \alpha_k \subset ... \subset \alpha_1 \subset \alpha_0.$$

Se puede seleccionar un valor de α usando un conjunto de validación o usando k fold cross-validation. Luego se vuelve al conjunto de datos completo y se obtiene el subárbol correspondiente a α. Después de esto, se puede ver el proceso completo para obtener un árbol de clasificación adecuado en el siguiente algoritmo.

Algoritmo 2: Árbol de Decisión

1. Use la división recursiva para hacer crecer el árbol con el conjunto de entrenamiento, pare únicamente cuando ya no se pueda mejorar la división o cuando cada nodo terminal tenga menos observaciones que el número mínimo de observaciones.
2. Aplique poda de complejidad de costos al árbol grande para obtener una secuencia de los mejores subárboles, en función de α.
3. Usa K-Fold cross validation para elejir la α. Esto es, dividir el conjunto de entrenamiento en K partes. para cada $k = 1, ..., K$:
   (a) Repetir el paso 1 y 2 en todas las k partes del conjunto de entrenamiento.
   (b) Evalúa el error en los datos de prueba de cada k parte, como función de α.

   se elige el α que minimice el error de validación.
4. Devuelve el subárbol del paso 2 que corresponda con el valor de α elegido.

Los árboles de decisión pueden ser construidos con variables tanto cuantitativas como cualitativas.

Ventajas de los árboles de decisión

- Los árboles son muy fáciles de explicar a las personas. De hecho, son más fáciles de explicar que una regresión lineal.
- Algunas personas piensan que los árboles de decisión son un espejo de las decisiones que toman las personas.
- Se pueden representar gráficamente, pueden ser interpretados por personas que no son expertas en el tema.
- Pueden encargarse de predictores cualitativos sin necesidad de crear variables dummy.
- Robustez a valores atípicos.
- La invarianza en la estructura de sus árboles de clasificación o de regresión y su interpretabilidad.

Desventajas de los árboles de decisión

- Desafortunadamente, los árboles generalmente no tienen el nivel de exactitud en sus predicciones como otros métodos de Machine Learning.

- Son propensos a sufrir de sobreajuste.

Como se ha mencionado muchas veces, el enfoque estructurado en árbol maneja las variables categóricas y ordenadas de una manera simple y natural. Los árboles de clasificación a veces realizan una selección automática de variables escalonadas y una reducción de la complejidad. Proporcionan una estimación de la tasa de clasificación errónea para un punto de prueba. Para cada punto de datos, se sabe en qué nodo hoja aterriza y se tiene una estimación de las probabilidades posteriores de clases para cada nodo hoja. La tasa de clasificación errónea se puede estimar utilizando la clase posterior estimada.

Los árboles de clasificación son invariables en todas las transformaciones monótonas de variables ordenadas individuales. La razón es que los árboles de clasificación dividen los nodos por umbral. Las transformaciones monótonas no pueden cambiar las formas posibles de dividir los puntos de datos por umbral. Los árboles de clasificación también son relativamente sólidos para los valores atípicos y los puntos mal clasificados en el conjunto de entrenamiento. No calculan un promedio o cualquier otra cosa a partir de los puntos de datos. Los árboles de clasificación son fáciles de interpretar, lo cual es atractivo especialmente en aplicaciones médicas

Sin embargo, existen enfoques como lo son *bagging, random forests y boosting* que usan a los árboles de decisión para obtener mejores predicciones. Combinar una gran cantidad de árboles, a menudo, puede resultar en mejoras dramáticas en la precisión de la predicción, a expensas de alguna pérdida en la interpretación.

3.3. Aprendizaje Profundo (Deep Learning)

El aprendizaje profundo (*Deep Learning*) es un subconjunto del Machine Learning donde se usan redes neuronales, todo proceso de aprendizaje profundo es un proceso de Machine Learning pero no todo proceso de Machine Learning es un proceso de aprendizaje profundo. El Deep Learning lleva a cabo el proceso de Machine Learning usando una red neuronal que se compone de un número de niveles jerárquicos. En el nivel inicial de la jerarquía, la red aprende algo simple y luego envía esa información al siguiente nivel, el siguiente nivel toma toda esta información sencilla, la combina, compone una información algo un poco más compleja, y se lo pasa al siguiente nivel, y así sucesivamente.

Para entender mejor el concepto de aprendizaje profundo se muestra el siguiente ejemplo, imagine a un niño cuya primera palabra es "perro". El niño aprende lo que es un perro (y lo que no es), el niño señala diferentes objetos diciendo la palabra perro, el padre dice "si es perro"ó "no, eso no es un perro", mientras el niño continúa apuntando a los objetos se vuelve más consciente de las características que poseen todos los perros. Lo que el niño hace, sin saberlo, es aclarar una abstracción compleja (el concepto de perro), construyendo una jerarquía en la que cada nivel de abstracción se crea con el conocimiento que se obtuvo de la capa precedente de la jerarquía.

Los algoritmos que se utilizan en el aprendizaje profundo pasan por el mismo proceso, cada algoritmo en la jerarquía aplica una transformación no lineal en su entrada y utiliza lo que se aprende como salida. El número de capas de procedimiento a través de las cuales los datos deben pasar es lo que inspiro la etiqueta de profundidad (Deep). Con el fin de lograr un nivel aceptable de precisión, los programas de aprendizaje profundo requieren acceso a inmensas cantidades de datos de entrenamiento y poder de procesamiento. El aprendizaje profundo es capaz de crear modelos precisos a partir de grandes cantidades de datos no etiquetados y

no estructurados. El descubrimiento y reconocimiento de patrones en el mundo es un factor fundamental en los procesos científicos y tecnológicos actuales. La cuestión ahora es como utilizar el Deep Learning para obtener nuevos conocimientos o mejorar lo que se está haciendo. En lugar de organizar datos para que se ejecuten a través de ecuaciones predefinidas, el Deep Learning configura parámetros básicos acerca de los datos y entrena a la computadora para que aprenda por cuenta propia reconociendo patrones mediante el uso de muchas capas de procesamiento. Se necesita mucho poder de cómputo debido a su naturaleza iterativa, su complejidad aumenta conforme aumentan el número de capas que se necesitan para la red.

3.3.1. Red Neuronal (Neural Network)

A pesar de su nombre, las redes neuronales no tienen un concepto demasiado complicado detrás de ellas. El nombre, como se puede imaginar, viene de la idea de imitar el funcionamiento de las redes de neuronas del cerebro humano; un conjunto de neuronas conectadas entre sí y que trabajan en conjunto, sin que haya una tarea concreta para cada una. Con la experiencia, las neuronas van creando y reforzando ciertas conexiones para aprender algo que se queda fijo en el tejido.

Las redes neuronales han ido moviéndose para tener un foco en matemáticas y estadística. Se basan en una idea sencilla: dados unos parámetros hay una forma de combinarlos para predecir un cierto resultado. Las redes son un modelo para encontrar esa combinación de parámetros y aplicarla al mismo tiempo. En lenguaje propio, encontrar la combinación que mejor se ajusta es entrenar la red neuronal. Las redes neuronales están formadas por varias neuronas y esta unidad básica también es llamada perceptrón.

3.3.2. Perceptrón (Neurona)

Los Perceptrones fueron desarrollados en 1950 por el científico Frank Rosenblatt, una neurona es la unidad más básica de procesamiento dentro de una red neuronal, no es más que una función matemática, es una forma de discriminador lineal, a partir de lo cual se desarrolla un algoritmo capaz de generar un criterio de salida. Un Perceptrón es un elemento que tiene varias entradas con un cierto peso cada una.

Un discriminador lineal en el campo del aprendizaje automático toma una decisión de clasificación basada en el valor de una combinación lineal de sus características. Las características de un objeto son típicamente presentadas en un vector llamado vector de características. Si la entrada del clasificador es un vector de características reales $\hat{x}$ y w un vector de pesos, entonces el resultado de salida es $y = f(w \cdot \hat{x}) = f(\sum_j w_j x_j)$). Se puede decir que lo que hace una neurona internamente es un modelo de regresión lineal.

Un perceptrón tiene varias entradas $x_1, x_2, ..., x_p$, donde se le asignan pesos, $w_1, w_2, ..., w_p$, números reales que expresen la importancia de las entradas para la salida, donde se hace una suma ponderada, y se agrega un sesgo b, la salida de la neurona es determinada por la función de activación f, y produce una salida :

$$y = f(W \cdot X + b) \qquad (3.28)$$

donde:

- f es la función de activación, la cual puede variar dependiendo de los objetivos, para evitar problemas a la hora de concatenar las neuronas esta función es no lineal y diferenciable.

- $W \cdot X$ es la suma ponderada de las entradas con los pesos asignados.
- b es el sesgo de la neurona.

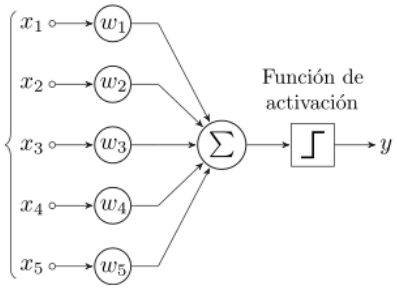

Figura 3.4: Perceptrón.

Así es como se ve un perceptrón, es el modelo matemático básico, una manera de ver al perceptrón es como un dispositivo que toma decisiones al ponderar la evidencia. Una neurona sola y aislada carece de razón de ser, su labor especializada se torna valiosa en la medida en que se asocia a otras neuronas.

Una red neuronal es la unión de varias neuronas, cuando una neurona está en la misma columna se le llama capa, cada neurona en la misma capa recibirá la misma información de la capa anterior y los cálculos que hagan se pasaran a la capa siguiente.

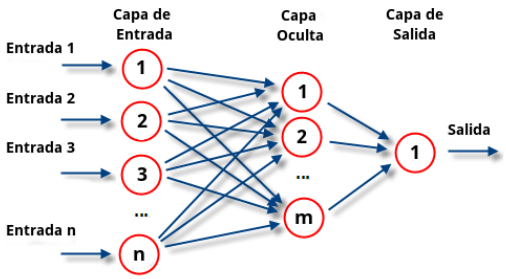

Figura 3.5: Red Neuronal o Perceptrón multicapa.

- **Capa de entrada(*Input Layer*)**: Constituida por aquellas neuronas que introducen las características de entrada en la red, en estas neuronas no se produce procesamiento.
- **Capas intermedias u ocultas (*Hidden Layers*)**: Formada por aquellas neuronas cuyas entradas provienen de capas anteriores y cuyas salidas pasaran a capas posteriores.
- **Capa de salida (*Output Layer*)**: Neuronas cuyo valor de salida corresponden con las salidas de toda la red, estas neuronas harán las predicciones.

Cuando se ponen las neuronas en forma secuencial una recibe la información procesada de la anterior, la ventaja de esto es que la red puede aprender de forma jerarquizada, mientras más capas se agreguen más complejo puede ser el conocimiento obtenido.

Como se mencionó antes, cada neurona tiene el efecto de hacer una regresión lineal, y la concatenación de varias neuronas daría como resultado una sola neurona, para evitar esto es que se utiliza la función de activación. Lo que hace la función de activación es distorsionar la salida, añadiendo deformaciones no lineales, al añadir estas deformaciones no lineales se da por solucionado el problema de encadenar varias neuronas.

3.4. Funciones de Activación

Las funciones de activación de la red neuronal son un componente crucial del aprendizaje profundo. Las funciones de activación determinan el resultado de un modelo de aprendizaje profundo, su precisión y también la eficiencia computacional de entrenar un modelo, que puede hacer o deshacer una red neuronal a gran escala. Las funciones de activación también tienen un efecto importante en la capacidad de convergencia de la red neuronal y la velocidad de convergencia, o en algunos casos, las funciones de activación pueden evitar que las redes neuronales converjan en primer lugar.

Las funciones de activación son ecuaciones matemáticas que determinan la salida de una red neuronal. Las funciones de activación también ayudan a normalizar la salida de cada neurona en un rango entre 0 y 1 o entre -1 y 1. Un aspecto adicional de las funciones de activación es que deben ser computacionalmente eficientes porque se calculan a través de miles o incluso millones de neuronas para cada muestra de datos. Las redes neuronales modernas usan una técnica llamada retropropagación (*BackPropagation*) para entrenar el modelo, lo que aumenta la tensión computacional en la función de activación y su función derivada. A continuación, se mostrarán 3 de las más utilizadas.

3.4.1. Función escalonada (Binary Step Function)

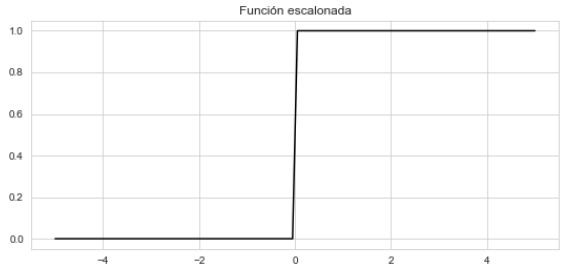

Figura 3.6: Función Escalonada.

Una función escalonada es una función de activación basada en un umbral. Si el valor de entrada está por encima o por debajo de cierto umbral, la neurona se activa y envía exactamente la misma señal a la siguiente capa.

$$f(x) = \begin{cases} 0 \quad si \quad W \cdot X + b \leq 0 \\ \\ 1 \quad si \quad W \cdot X + b > 0 \end{cases} \text{,b} \in \mathbb{R}. \qquad (3.29)$$

El problema con una función escalonada es que no permite salidas de valores múltiples, por ejemplo, no puede soportar clasificar las entradas en una de varias categorías.

3.4.2. Función Sigmoide (*Sigmoid Function*)

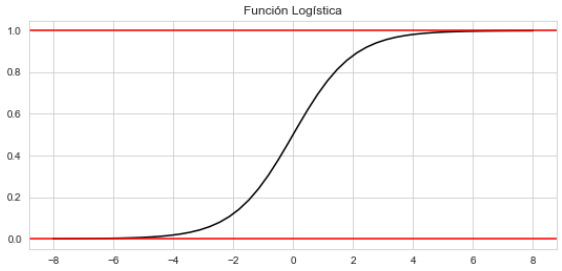

Figura 3.7: Función Sigmoide.

Históricamente, la función sigmoide es la función de activación más antigua y popular, una neurona que utiliza la sigmoide como función de activación se le llama neurona sigmoide. Primero se establece que la variable equivale a la suma ponderada de los datos de entrada con sus pesos, y después se pasa a través de la función sigmoide.

$$f(z) = \frac{1}{1-e^{-z}},\ z = W \cdot X + b.$$

De la cual se habló anteriormente en la sección de regresión logística.

3.4.3. Rectilínea Uniforme (*ReLU – Rectified Lineal Unit*)

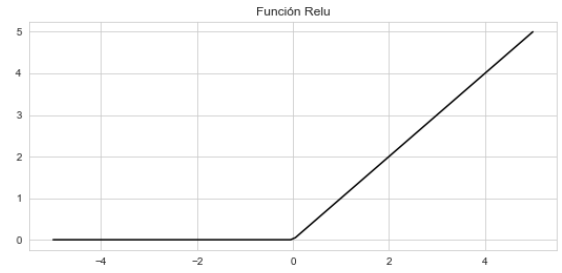

Figura 3.8: Función Rectilínea Uniforme.

En otras palabras, las ReLUs permiten el paso de todos los valores positivos sin cambiarlos, pero asigna todos los valores negativos a 0. Aunque existen funciones de activación aún más recientes, la mayoría de las redes neuronales de hoy utilizan ReLU o una de sus variantes.

$$f(z) = \begin{cases} 0, & si \quad z < 0 \\ \\ z, & si \quad z \geq 0. \end{cases} \tag{3.30}$$

Al añadir estas deformaciones no lineales, se soluciona el problema de concatenar varias neuronas. Al construir un modelo y entrenar una red neuronal, la selección de funciones de activación es crítica. Experimentar con diferentes funciones de activación para diferentes

problemas le permitirá lograr resultados mucho mejores.

Después de fijar la arquitectura de la red, la cual va a determinar la complejidad del problema, se puede utilizar la siguiente fórmula para determinar el tamaño del conjunto de datos por:

$$N \geq \frac{W}{\epsilon} \quad , \epsilon > 0. \qquad (3.31)$$

Donde W es el número de pesos de la red y ϵ es la fracción del error permitido, regularmente se establece en .01, de esta forma se puede asegurar que el número de muestras permitirá a la red generalizar correctamente.

3.5. Propagación hacia adelante (*Forward Propagation*)

Como su nombre indica, los datos de entrada se propagan en la dirección hacia adelante a través de la red. Cada capa oculta acepta los datos de entrada, los procesa según la función de activación y pasa a la capa sucesiva. Para generar algo de salida, los datos de entrada deben alimentarse sólo en la dirección de avance. Los datos no deben fluir en dirección inversa durante la generación de salida, de lo contrario, se formaría un ciclo y la salida nunca podría generarse. Dichas configuraciones de red se conocen como red de avance. En cada neurona, en una capa oculta o de salida, el procesamiento ocurre en dos pasos:

Preactivación: es una suma ponderada de entradas, es decir, la transformación lineal de pesos con las entradas disponibles.

Activación: la suma ponderada calculada de las entradas se pasa a la función de activación y da los valores de salida.

Durante la propagación hacia adelante en cada nodo de la capa oculta y de salida, tiene lugar la preactivación y activación. El pase directo se refiere al proceso de cálculo, los valores de las capas de salida de los datos de entrada. Atraviesa todas las neuronas desde la primera hasta la última capa.

En el contexto de las redes neuronales, el aprendizaje puede ser visto como el proceso de ajuste de los parámetros de pesos y sesgo de la red. Partiendo de un conjunto de pesos aleatorio, el proceso de aprendizaje busca un conjunto de pesos que permitan a la red desarrollar correctamente una determinada tarea. El proceso de aprendizaje es un proceso iterativo, en el cual se va refinando la solución hasta alcanzar un nivel de operación suficientemente bueno. La mayoría de los métodos de entrenamiento utilizados en las redes neuronales con conexión hacia delante consisten en proponer una función de error que mida el rendimiento actual de la red en función de los parámetros. El proceso de entrenamiento de una red neuronal es determinar un conjunto de parámetros que minimizan la diferencia entre el valor esperado y la salida del modelo. Esto se realiza mediante el descenso de gradiente, que por definición comprende dos pasos: calcular los gradientes de la función de error, luego actualizar los parámetros existentes en respuesta a los gradientes, que es cómo se realiza el descenso. Este ciclo se repite hasta alcanzar los mínimos de la función de pérdida.

El proceso de descenso de gradiente se exhibe en la forma del paso de propagación hacia atrás (BackPropagation) donde se calculan los vectores de error δ hacia atrás, comenzando desde la

capa final. Dependiendo de la función de activación, se identificará cuánto cambio se requiere tomando la derivada parcial de la función con respecto a W y b. El valor del cambio se multiplica por la tasa de aprendizaje. Como parte de la salida, se resta este valor de la salida anterior para obtener el valor actualizado. Se continua con esto hasta llegar a la convergencia.

Cuando se crea una red neuronal ésta tiene sus parámetros inicializados de manera aleatoria, por lo que se quiere que la función de costes le dará un valor elevado, se usará este error para enseñar a la red. Es operar de formar recursiva capa atrás capa moviendo el error hacia atrás de la red, los errores son los que se usaran para calcular las derivadas parciales de cada parámetro de la red, para obtener el vector gradiente.

3.6. Propagación hacia atrás (BackPropagation)

El algoritmo de Backpropagation se introdujo originalmente en la década de 1970, pero su importancia no fue plenamente apreciada hasta un famoso artículo de 1986 de David Rumelhart, Geoffrey Hinton y Ronald Williams. Ese documento describe varias redes neuronales en las que la retropropagación funciona mucho más rápido que los enfoques anteriores de aprendizaje, lo que hace posible utilizar redes neuronales para resolver problemas que anteriormente habían sido insolubles. Hoy, el algoritmo de retropropagación es el caballo de batalla del aprendizaje en redes neuronales.

En el corazón de backpropagation hay una expresión para las derivadas parciales $\frac{\partial C}{\partial W}$ y $\frac{\partial C}{\partial b}$ de la función de coste C con respecto a cualquier peso (o sesgo) en la red. La expresión dice qué tan rápido cambia el coste cuando se cambian los pesos y los sesgos. Y aunque la expresión es algo compleja, también tiene un cierto encanto, ya que cada elemento tiene una interpretación natural e intuitiva. Entonces, backpropagation no es solo un algoritmo rápido para el aprendizaje, en realidad, brinda información detallada sobre cómo cambiar los pesos y los sesgos cambia el comportamiento general de la red.

Una vez que se ha aplicado un patrón a la entrada de la red como estímulo, este se propaga desde la primera capa a través de las capas siguientes de la red, hasta generar una salida (forward propagation). La señal de salida se compara con la salida deseada y se calcula una señal de error para cada una de las salidas. Las salidas de error se propagan hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de la capa oculta que contribuyen directamente a la salida. Sin embargo, las neuronas de la capa oculta solo reciben una fracción de la señal total del error, basándose aproximadamente en la contribución relativa que haya aportado cada neurona a la salida original. Este proceso se repite, capa por capa, hasta que todas las neuronas de la red hayan recibido una señal de error que describa su contribución relativa al error total.

La importancia de este proceso consiste en que, a medida que se entrena la red, las neuronas de las capas intermedias se organizan a sí mismas de tal modo que las distintas neuronas aprenden a reconocer distintas características del espacio total de entrada. Después del entrenamiento, cuando se les presente un patrón arbitrario de entrada que contenga ruido o que esté incompleto, las neuronas de la capa oculta de la red responderán con una salida activa si la nueva entrada contiene un patrón que se asemeje a aquella característica que las neuronas individuales hayan aprendido a reconocer durante su entrenamiento.

¿Cuánto varía el error ante un cambio de los parámetros? El objetivo de backpropagation es calcular las derivadas parciales $\frac{\partial C}{\partial W}$ y $\frac{\partial C}{\partial b}$ de la función de coste C con respecto a cualquier

peso W o sesgo b en la red. Para calcular el error de la red se utilizará la función de coste cuadrática:

$$C(W,b) = \frac{1}{2n} \sum_x \parallel y(x) - a^L(x) \parallel^2. \qquad (3.32)$$

W denota la colección de todos los pesos en la red, b todos los sesgos, n es el número total de entradas de entrenamiento, a^L es el vector de salidas de la L-ésima capa cuando se ingresa x, $y(x)$ es el valor real de la observación y la suma es sobre todas las entradas de entrenamiento x. L es el número de capas que tiene la red neuronal.

La función del error cuadrático para un solo ejemplo de entrenamiento x puede escribirse diferente, esto ayudará para hacer los siguientes cálculos más sencillos:

$$C = \frac{1}{2} \sum_j (y_j - a_j^L)^2. \qquad (3.33)$$

Como se mencionó, se va a trabajar hacia atrás, se calcularán las derivadas de los parámetros W y b de la última capa. Para calcular esto se necesita saber cuál es el camino que conecta el valor del parámetro y el coste final, la función de pérdida es una métrica de error, que proporciona un indicador de cuánta precisión se pierde, si se reemplaza la salida real por la salida real generada por el modelo de red neuronal entrenado. Por eso se llama pérdida.

Una forma de ver la pérdida o el error en la última capa de la red es la siguiente

$$ERROR = C(a^L(Z^L)). \qquad (3.34)$$

Donde

- Z^L = Vector con el resultado de las sumas ponderadas en las neuronas de la capa L.
- a^L = Función de activación de la capa L.
- C = Función de coste.

Sin embargo, en la mayoría de los casos, componer las funciones es muy difícil. Además, para cada composición, se tiene que calcular la derivada dedicada de la composición. Para resolver el problema, afortunadamente, la derivada es descomponible, por lo tanto, puede propagarse nuevamente. Se tiene el punto de partida de los errores, que es la función de pérdida, y se sabe cómo calcular su derivada, y si se sabe cómo calcular la derivada de cada función a partir de la composición, se puede propagar el error desde el final hasta el comienzo. Para obtener las derivadas de los parámetros se utilizará la regla de la cadena:

$$\frac{\partial C}{\partial W^L} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial Z^L} \cdot \frac{\partial Z^L}{\partial W^L}. \qquad (3.35)$$

$$\frac{\partial C}{\partial b^L} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial Z^L} \cdot \frac{\partial Z^L}{\partial b^L}. \qquad (3.36)$$

- La derivada parcial $\frac{\partial C}{\partial a^L}$ es la derivada de la función de coste respecto a la función de activación, esto es, como varía el error de la red cuando se varía la salida de la activación de las neuronas en la última capa, en este caso las activaciones de las neuronas son la salida de la red.

- La derivada parcial $\frac{\partial a^L}{\partial z^L}$ es la derivada de la activación respecto a la suma ponderada, esto es, como varia la salida de la neurona cuando se varia la suma ponderada de la neurona, si se recuerda lo único que separa a Z con la activación de la neurona es la función de activación.
- Por último las derivadas parciales $\frac{\partial Z^L}{\partial b^L}$ y $\frac{\partial Z^L}{\partial W^L}$ es como varía la suma ponderada cuando varian los parámetros de los pesos y el sesgo.

Derivando la suma ponderada

$$Z^L = \sum_i a_i^{L-1} w_i^L + b^L.$$

Donde el subíndice indica la i−ésima neurona de la capa L, de esta manera se obtienen las parciales

$$\frac{\partial Z^L}{\partial b^L} = 1 \qquad \text{y} \qquad \frac{\partial Z^L}{\partial w^L} = a_i^{L-1}$$

donde es 1 ya que el término es independiente de lo demás y a_i^{L-1} es el valor de salida de la neurona anterior. Finalmente, la solución que se busca para los parámetros de la última capa se calcula multiplicando las derivadas obtenidas.

Tomando:

$$\frac{\partial C}{\partial Z^L} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial Z^L}. \qquad (3.37)$$

Esto dice como varia el error en función del valor de Z, es decir, cuenta en que grado se modifica el error cuando se produce un cambio en la suma ponderada de la neurona. Si este es grande, significa que un pequeño cambio en la neurona se verá reflejado en el trabajo final. Si este es pequeño, no importa que tanto varié el valor de la neurona, no impactará en el error de la red. Esta derivada va a contar que responsabilidad tiene la neurona en el resultado final, a esta derivada se le llamará *Error imputado a la neurona* y se denotará por δ de la siguiente manera.

$$\frac{\partial C}{\partial Z^L} = \delta^L. \qquad (3.38)$$

Así para simplificar la ecuación inicial se puede poner en función del error de las neuronas de la última capa (capa L).

$$\frac{\partial C}{\partial b^L} = \delta^L \cdot \frac{\partial Z^L}{\partial b^L} \qquad \text{y} \qquad \frac{\partial C}{\partial w^L} = \delta^L \cdot \frac{\partial Z^L}{\partial w^L}.$$

De esta manera se obtiene que

- $\frac{\partial C}{\partial b^L} = \delta^L \cdot 1$ la derivada del coste respecto al término de sesgo es igual al error de las neuronas.
- $\frac{\partial C}{\partial w^L} = \delta^L \cdot a_i^{L-1}$ la derivada del coste respecto a los pesos es igual al error de las neuronas por la activación de la capa previa.
- $\delta^L = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial Z^L}$ el error imputado por la neurona es el producto de la derivada de la función de coste y la derivada de la función de activación.

De esta manera se tienen 3 expresiones que ayudaran a encontrar las derivadas parciales para la última capa, sólo se necesita una expresión más para calcular el resto de derivadas de la red.

Ahora se quiere calcular las derivadas de la capa anterior (L-1) basándose en el mismo razonamiento anterior y usando la regla de la cadena $C(a^L(W^L a^{L-1}(W^{L-1}a^{L-2} + b^{L-1}) + b^L))$:

$$\frac{\partial C}{\partial W^{L-1}} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial Z^L} \cdot \frac{\partial Z^L}{\partial a^{L-1}} \cdot \frac{\partial a^{L-1}}{\partial Z^{L-1}} \cdot \frac{\partial Z^{L-1}}{\partial W^{L-1}} \qquad (3.39)$$

y

$$\frac{\partial C}{\partial b^{L-1}} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial Z^L} \cdot \frac{\partial Z^L}{\partial a^{L-1}} \cdot \frac{\partial a^{L-1}}{\partial Z^{L-1}} \cdot \frac{\partial Z^{L-1}}{\partial b^{L-1}}. \qquad (3.40)$$

Donde se puede ver, que del hecho de que se va propagando el error hacia atrás, algunas derivadas ya se han calculado antes, donde:

- $\delta^L = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial Z^L}$ que es el error de la última capa L.
- $\frac{\partial Z^{L-1}}{\partial b^{L-1}} = 1$ y $\frac{\partial Z^{L-1}}{\partial W^{L-1}} = a^{L-2}$ que operan igual que antes, 1 y la activación de la capa previa.
- $\frac{\partial a^{L-1}}{\partial Z^{L-1}}$ es la derivada de la función de activación en la capa $L-1$.
- $\frac{\partial Z^L}{\partial a^{L-1}}$ es lo único que faltaría de calcular, esto es, como cambia la suma ponderada cuando varia la salida de una neurona en la capa previa, la cual es básicamente la matriz de parámetros W^L que conecta ambas capas. Básicamente lo que hace esto es mover el error de una capa a la capa anterior, distribuyendo el error en función de cuáles son las ponderaciones de las conexiones.

Con esto, ya se tendría nuevamente una expresión a partir de la cual obtener las derivadas parciales que se están buscando, tomando

$$\delta^{L-1} = \frac{\partial C}{\partial Z^{L-1}} = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial Z^L} \cdot \frac{\partial Z^L}{\partial a^{L-1}} \cdot \frac{\partial a^{L-1}}{\partial Z^{L-1}}. \qquad (3.41)$$

La cual vuelve a representar el error imputado de las neuronas en esta capa, con el algoritmo de Backpropagation, lo que se hizo en esta capa ya es extensible al resto de capas de la red aplicando la misma lógica.

1. Cómputo del error de la última capa.
   $\delta^L = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial Z^L}$.
2. Se Retropropaga el error a la capa anterior
   $\delta^{l-1} = W^l \delta^l \cdot \frac{\partial a^{l-1}}{\partial z^{l-1}}$ se toma el error de la capa anterior y se multiplica por la matriz de pesos en una transformación que viene a representar la retropropagación de los errores.
3. Se calculan las derivadas parciales respecto a los parámetros de la capa usando el error
   $\frac{\partial C}{\partial b^{l-1}} = \delta^{l-1}$ y $\frac{\partial C}{\partial w^{l-1}} = \delta^{l-1} a^{l-2}$

y así sucesivamente recorriendo todas las capas de la red hasta llegar al final, de esta manera, con un único pase se habrán calculado todos los errores y todas las derivadas parciales de la red con el solo uso de 4 expresiones, que son las anteriores. Como se puede ver el algoritmo de Backpropagation solamente es derivar 4 expresiones desde la última capa a la segunda capa. Estas expresiones son bastante intuitivas ya que solo están contando como utilizar el error de las capas y propagarlo hacia atrás para calcular el error en esta capa.

Se puede pensar en la retropropagación como una forma de aplicar el gradiente descendiente, calculando el gradiente de la función de costos aplicando sistemáticamente la regla de la

cadena.

Algoritmo 3: BackPropagation

1. Ingresa un vector $X = (x_1, x_2, ..., x_p)$, con las características de las observaciones y establezca la función de activación correspondiente a^l para la *l-ésima* capa, $l = 2, ..., L$.
2. Aplique propagación hacia adelante, para cada $l = 2, ..., L$, calcule la suma ponderada $z^l = w^l \cdot a^{l-1} + b^l$ y $a^l = f(z^l)$, donde f es la función de activación de esa capa.
3. Calcule el error de la capa de salida, δ^L.
4. Propague el error hacia atrás, para cada $l = L-1, L-2, ..., 2$, calculando el error $\delta^l = W^{l+1}\delta^{l+1} \cdot \frac{\partial a^l}{\partial z^l}$.
5. La salida es el gradiente de la función de costo, viene dado por $\frac{\partial C}{\partial b^l} = \delta^l$ y $\frac{\partial C}{\partial w^l} = \delta^l a^{l-1}$.
6. Por último se actualizan los parámetros utilizando el método del gradiente descendente, $W_i = W_{i-1} - \alpha \cdot \frac{\partial C}{\partial W^l}$ y $b_i = b_{i-1} - \alpha \cdot \frac{\partial C}{\partial b^l}$ donde i es el número de ciclos que han pasado y α es un término llamado *learning rate*.
7. Se repite el proceso del paso 1 al paso 6 actualizando los parámetros hasta llegar a un nivel aceptable del error.

La tasa de aprendizaje (learning rate) es una constante positiva entre 0 y 1, esta constante sirve para definir el costo que tiene el gradiente en la actualización de un peso. Entre mayor sea su valor, mayor la magnitud en la que incrementa o decrece el peso, lo cual puede ser bueno o afectar la convergencia de la función de coste. El valor adecuado de este hiperparámetro es muy dependiente del problema en cuestión, pero en general, si este es demasiado grande, se están dando pasos enormes, lo que podría ser bueno para ir rápido en el proceso de aprendizaje, pero es posible que se salte el mínimo y dificultar así que el proceso de aprendizaje se detenga porque al buscar el siguiente punto perpetuamente rebota al azar en el fondo del "pozo". Contrariamente, si la tasa de aprendizaje es pequeña, se harán avances constantes y pequeños, teniéndose una mejor oportunidad de llegar a un mínimo local, pero esto puede provocar que el proceso de aprendizaje sea muy lento. En general, una buena regla es que, si el modelo de aprendizaje no funciona, se disminuya la tasa de aprendizaje. Si se sabe que el gradiente de la función de pérdida es pequeño, entonces es seguro probar con la tasa de aprendizaje que compensen el gradiente.

Al examinar el algoritmo, se puede ver por qué se llama retropropagación. Se calculan los vectores de error δ^l hacia atrás, comenzando desde la capa final. Puede parecer peculiar que se esté atravesando la red hacia atrás. Pero si piensa en la prueba de propagación hacia atrás, el movimiento hacia atrás es una consecuencia del hecho de que el costo es una función de los resultados de la red. Para comprender cómo varia el costo con los pesos y sesgos anteriores, se necesita aplicar repetidamente la regla de la cadena, trabajando hacia atrás a través de las capas para obtener expresiones utilizables. El algoritmo de retropropagación es una forma inteligente de realizar un seguimiento de pequeñas perturbaciones en los pesos (y sesgos) a medida que se propagan a través de la red, alcanzan la salida y luego afectan el coste.

Una vez finalizada la fase de aprendizaje, la red puede ser utilizada para realizar la tarea para la que fue entrenada. Una de las principales ventajas que posee este modelo es que la red aprende la relación existente entre los datos, adquiriendo la capacidad de generalizar conceptos. De esta manera, una red neuronal puede tratar con información que no le fue presentada durante la fase de entrenamiento.

VENTAJAS

- Las redes neuronales pueden solucionar problemas no lineales y de alta complejidad.
- Son un instrumento muy flexible para la solución de problemas, ya que las neuronas pueden reconocer patrones que no han sido aprendidos.
- El algoritmo de retropropagación disminuye el costo y tiempo en el proceso de aprendizaje de la red.
- Los pesos son ajustados basándose en la experiencia, lo que significa que se le tiene que enseñar a la red lo que necesita saber antes de ponerla en funcionamiento.
- Las neuronas son tolerantes a fallos, si parte de la red no trabaja, solo dejará de funcionar la parte para que dicha neurona sea significativa; el resto tendrá su comportamiento normal.

DESVENTAJAS

- Complejidad de aprendizaje para grandes tareas, cuantas más cosas se necesiten que aprenda una red, más complicado será enseñarle.
- Tiempo de aprendizaje elevado. Esto depende de dos factores: primero si se incrementa la cantidad de patrones a identificar o clasificar y segundo si se requiere mayor flexibilidad o capacidad de adaptación de la red neuronal para reconocer patrones que sean sumamente parecidos, se deberá invertir más tiempo en lograr que la red converja a valores de pesos que representen lo que se quiera enseñar.
- No permite interpretar lo que se ha aprendido, la red por si sola proporciona una salida, un número, que no puede ser interpretado por ella misma, sino que se requiere de la intervención del programador y de la aplicación en si para encontrarle un significado a la salida proporcionada.
- Solo la experiencia puede proporcionar el tipo de topología que se utilizará en la red.

3.7. Capacidad predictiva del modelo

Es de interés clasificar a los individuos dependiendo de que si su probabilidad supera un punto de corte p o no, en particular si el valor de la probabilidad estimada excede a p entonces se tendrá una variable igual a 1, es decir que sea moroso, de otra forma será igual a 0, donde no se es moroso. El valor que se utilizó como punto de corte en los modelos fue de 0.5.

$$\textit{Clasificación} = \begin{cases} \textit{Probabilidad} > p & \Rightarrow \quad y_i = 1 \\ \textit{Probabilidad} \leq p & \Rightarrow \quad y_i = 0. \end{cases}$$

Cuando se construye un modelo de clasificación binaria como será en el caso de estudio, ¿Será este cliente un posible moroso? Dado un conjunto de datos etiquetados y el modelo creado, todos los puntos del conjunto pueden caer en 4 categorías:

TP = True Positives (Verdaderos Positivos).- Este cliente es moroso, y se clasificó correctamente como moroso.

FP= False Positives (Falsos Positivos).- Este cliente no es moroso, pero se clasificó como moroso (Error tipo 1).

FN= False Negatives (Falsos Negativos).- Este cliente es moroso, pero se clasificó como no moroso (Error tipo 2).

TN= True Negatives (Verdaderos Negativos).- Este cliente no es moroso, y se clasificó correctamente como no moroso.

A menudo se encuentran estos valores en una matriz de confusión (Confusion matrix).

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Figura 3.9: Matriz De Confusión.

Se pueden usar estos valores para calcular algunas métricas que ayuden a evaluar el desempeño de los modelos, como son las siguientes:

- Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$, se define como la fracción de predicciones correctas.
- Specificity = $\frac{TN}{TN+FP}$,se conoce como tasa verdadera negativa de una prueba indica la probabilidad de obtener un resultado negativo, es la proporción de casos negativos que son bien detectadas por la prueba.
- Precisión = $\frac{TP}{TP+FP}$, es una medida de cuán precisas fueron las predicciones positivas del modelo.
- Recall = $\frac{TP}{TP+FN}$, es una medida sobre qué fracción de los positivos identificó el modelo.
- F1 score = $\frac{2*Precisin*Recall}{Precisin+Recall}$, es una combinación entre Presición y Recall, es su promedio harmónico.

- Área bajo la curva ROC (Receiver Operating Characteristic) construida para todos los posibles puntos de corte p para la clasificación de individuos. La curva ROC es un gráfico en el que se observan todos los pares de sensibilidad/especifidad resultantes de la variación continua de los puntos de corte en todo el rango de resultados observado.

  En el eje y de coordenadas se sitúa la sensibilidad o fracción de verdaderos positivos, en el eje x se sitúa la fracción de verdaderos positivos. El área bajo la curva está dentro de un rango de 0 a 1, otorgando una medida de la capacidad del modelo para discriminar entre los sujetos que experimentan los resultados de interés contra los que no lo hacen.

En el caso de datos no balanceados accuracy no es una buena medida de rendimiento, ya que construyendo un modelo que asigne todos los datos a la clase con mayor número de datos se obtendría un porcentaje bastante alto, y esto no implicaría que el modelo creado fuera un buen modelo que generalice nuevos datos. Usualmente la elección de un buen modelo toma en cuenta un balance entre Recall y Presición, ya que un modelo que prediga *SI* cuando incluso no es muy probable que sea cierto, tendrá un alto valor de Recall pero un bajo valor en Precisión, por otro lado un modelo que prediga *SI* solo cuando es extremadamente posible que sea cierto tendrá un valor bajo de Recall y un valor alto en Precisión. Otra manera de ver esto es haciendo un balance entre **FP** Y **FN**, diciendo *SI* demasiado se obtendrá una cantidad grande de **FP** y diciendo demasiado *No* se obtendrán una cantidad grande de **FN**.

Si se tiene un desbalance en los datos y se necesita mantenerse lejos de falsos positivos y falsos negativos, una buena opción es f1-score. tiene un buen balance entre las dos métricas precisión y recall, puede optimizar clasificadores sin verse éste afectado porque no se tuvo un conjunto de datos balanceado. El puntaje F1 se puede interpretar como un promedio ponderado de la precisión y el recuerdo, donde un puntaje F1 alcanza su mejor valor en 1 y el peor puntaje en 0.

Por último, un rendimiento de la curva ROC es:

a) Si ROC $= 0.5$ se sugiere no discriminación.
b) Si $0.7 \leq$ ROC < 0.8 se considera discriminación aceptable.
c) Si $0.8 \leq$ ROC < 0.9, se considera discriminación excelente.
d) Si ROC ≥ 0.9, se considera discriminación extraordinaria.

Capítulo 4

Caso de estudio

La República de China más conocida internacionalmente como Taiwán, también conocida en el pasado como Formosa del portugués *Isla Hermosa*, es un estado con reconocimiento limitado situado en el extremo oriente de Asia. Los estados vecinos incluyen la República Popular de China (RPC) al oeste, Japón al norte y Filipinas al sur. La isla de Taiwán tiene un área de 35,808 kilómetros cuadrados, con cadenas montañosas que dominan los dos tercios orientales y llanuras en el tercio occidental, donde se concentra su población altamente urbanizada. Taipéi es la capital y el área metropolitana más grande. Con 23,7 millones de habitantes, Taiwán se encuentra entre los estados más densamente poblados y es el estado más poblado y la economía más grande que no es miembro de las Naciones Unidas.

Desde 1945, la isla y otras cercanas han estado bajo el régimen político de la República de China, el estado que gobernaba toda China hasta el final de la guerra civil entre el Kuomintang y el Partido Comunista de China, cuando este último se hizo con el poder en la China continental. Desde entonces, el antiguo régimen chino se ha mantenido en la isla de Taiwán, dando lugar a una compleja situación jurídica y diplomática, aunque en la práctica es un estado independiente parcialmente reconocido como República de China o Taiwán. La República de China (RDC) tiene su propia moneda, el Nuevo Dólar Taiwanés (*NTDollar*).

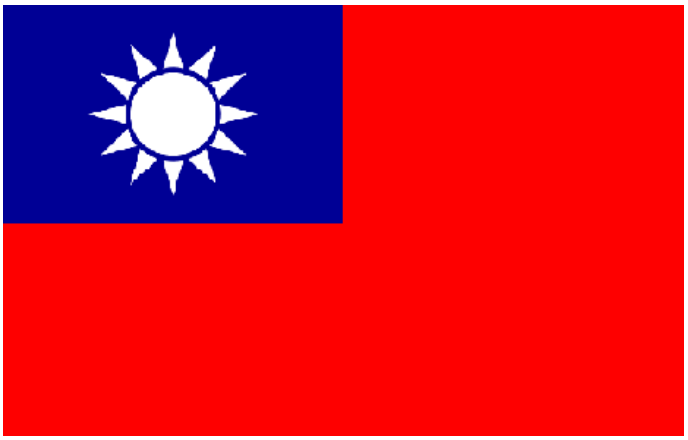

Figura 4.1: Bandera De Taiwán.

4.1. Crisis de tarjetas de crédito en Taiwán

A partir de 1990, el gobierno taiwanés permitió la formación de nuevos bancos. Estos nuevos bancos prestaron grandes sumas de dinero a empresas inmobiliarias con el objetivo de expandir sus negocios y aumentar sus ganancias. Sin embargo, después de un par de años de expansión, el mercado inmobiliario se saturó y las ganancias del sector dejaron de crecer. Los nuevos bancos recurrieron a otros negocios nuevos: tarjetas de crédito y tarjetas de efectivo. Al expandir esta área de negocios, los bancos prodigaron dinero en comerciales que animaban a las personas a solicitar tarjetas de crédito para consumir, aparentemente sin consecuencias. Estos bancos redujeron los requisitos para las aprobaciones de tarjetas de crédito para obtener más clientes. Con el tiempo, los jóvenes se convirtieron en clientes objetivo. Aunque los jóvenes tienden a no tener suficientes ingresos, los bancos todavía les emitieron tarjetas de crédito.

En Taiwán, en febrero de 2006, la deuda de tarjetas de crédito y tarjetas de efectivo alcanzó los 268 mil millones de dólares. Más de medio millón de personas no pudieron pagar sus préstamos. Se convirtieron en "esclavos de tarjetas de crédito", un término acuñado en Taiwán para referirse a personas que sólo podían pagar el saldo mínimo de su deuda de tarjeta de crédito cada mes. Este problema dio lugar a importantes problemas sociales. Algunos deudores y sus familias se suicidaron debido a la deuda, algunos quedaron sin hogar debido al embargo de sus hogares, otros no pudieron pagar la matrícula de sus hijos.

Algunos esclavos de tarjetas de crédito vendieron drogas ilegales para pagar a los bancos. Las prácticas de cobro a veces violentas y amenazantes de ciertos bancos aumentaron la presión sobre los prestamistas, particularmente aquellos en grupos de bajos ingresos. El gobierno taiwanés se vio obligado a resolver estos problemas para salvar el sistema financiero y evitar más problemas sociales. Según el informe del Departamento de Salud de Taiwán, 2,172 personas se suicidaron en 1997, 4,406 en 2006 y 4,128 en 2008. La tasa de suicidios en Taiwán es la segunda más alta del mundo. La tasa de suicidios en 2006 aumentó un 22,9 % en comparación con la tasa de 2005, y la razón principal es el desempleo y la deuda de las tarjetas de crédito.

Para evitar este problema lo que actualmente hacen los bancos es utilizar métodos de Credit Scoring, para el caso de estudio se utilizó una base de datos de un banco de Taiwán del año 2005.

4.1.1. Descripción de base de datos

Este conjunto de datos contiene información sobre pagos predeterminados, factores demográficos, datos crediticios, historial de pagos y estados de cuenta de clientes de tarjetas de crédito en Taiwán desde abril de 2005 hasta septiembre de 2005. El conjunto de datos proviene del Depósito de Aprendizaje Automático UCI Irvine, CA: Universidad de California, Escuela de Información e Informática (c.f [(17)]), otorgada por el Ph.D. I-Cheng Yeh (c.f [(27)]).

La base cuenta con 30,000 observaciones, y cuenta con 23 variables explicativas y una variable respuesta:

- **X1 - LIMIT_BAL (Monto del crédito otorgado) en dólares NT**: Incluye crédito individual y familiar / suplementario.
- **X2 - SEX (SEXO)**: 1 = masculino, 2 = femenino.
- **X3 - EDUCATION (EDUCACIÓN)**: 1 = escuela de posgrado, 2 = universidad, 3 = escuela secundaria, 4 = otros.
- **X4 - MARRIAGE (MATRIMONIO)**: Estado civil , 1 = casado, 2 = soltero, 3 = otros.

- **X5 - AGE (EDAD)**: Edad en años.
- **X6 - PAY_0 (PAGO 0)**: Estado de reembolso en septiembre de 2005 , -1 = pago debidamente, 1 = retraso en el pago durante un mes, 2 = retraso en el pago durante dos meses, ... 8 = retraso en el pago durante ocho meses, 9 = retraso en el pago durante nueve meses y más.
- **X7 - PAY_2 (PAGO 2)**: Estado de reembolso en agosto de 2005 (escala igual a la anterior).
- **X8 - PAY_3 (PAGO 3)**: Estado de reembolso en Julio de 2005 (escala igual a la anterior).
- **X9 - PAY_4 (PAGO 4)**: Estado de reembolso en Junio de 2005 (escala igual a la anterior).
- **X10 - PAY_5 (PAGO 5)**: Estado de reembolso en Mayo de 2005 (escala igual a la anterior).
- **X11 - PAY_6 (PAGO 6)**: Estado de reembolso en Abril de 2005 (escala igual a la anterior).
- **X12 - BILL_AMT1 (Monto del estado de cuenta):** En septiembre del 2005 (dólar NT).
- **X13 - BILL_AMT2 (Monto del estado de cuenta):** En Agosto del 2005 (dólar NT).
- **X14 - BILL_AMT3 (Monto del estado de cuenta):** En Julio del 2005 (dólar NT).
- **X15 - BILL_AMT4 (Monto del estado de cuenta):** En Junio del 2005 (dólar NT).
- **X16 - BILL_AMT5 (Monto del estado de cuenta):** En Mayo del 2005 (dólar NT).
- **X17 - BILL_AMT6 (Monto del estado de cuenta):** En Abril del 2005 (dólar NT).
- **X18 - PAY_AMT1 (Monto del pago anterior):** En septiembre del 2005 (dólar NT).
- **X19 - PAY_AMT2 (Monto del pago anterior):** En Agosto del 2005 (dólar NT).
- **X20 - PAY_AMT3 (Monto del pago anterior):** En Julio del 2005 (dólar NT).
- **X21 - PAY_AMT4 (Monto del pago anterior):** En Junio del 2005 (dólar NT).
- **X22 - PAY_AMT5 (Monto del pago anterior):** En Mayo del 2005 (dólar NT).
- **X23 - PAY_AMT6 (Monto del pago anterior):** En Abril del 2005 (dólar NT).
- **Y - default.payment.next.month (Impago en el mes siguiente):** 1 = sí, 0 = no.

4.2. Análisis exploratorio de datos (EDA)

El análisis exploratorio de datos se refiere al proceso crítico de realizar investigaciones iniciales sobre los datos para descubrir patrones, detectar anomalías, probar hipótesis y verificar supuestos con la ayuda de estadísticas resumidas y representaciones gráficas. Es una buena práctica comprender los datos primero e intentar reunir la mayor cantidad de información posible. EDA trata de dar sentido a los datos disponibles, antes de ensuciarlos con ellos. Dicho análisis se basa en gráficos y estadísticos que permiten explorar la distribución identificando características tales como: valores atípicos o outliers, saltos o discontinuidades, concentraciones de valores, forma de la distribución, etc.

En los procesos de Machine Learning es importante hacer un EDA para darse una idea de los datos con los que se trabaja, en esta sección se van a analizar las variables explicativas y se dará información importante sobre ellas.

El **Monto De crédito** es la suma total de dinero que la institución financiera está prestando. El valor promedio del Monto De Crédito de las tarjetas es de NT$167,484 y la desviación estándar es de NT$129,747 que es inusualmente grande, esto se debe a que los montos de crédito varían bastante, un ejemplo es que el valor máximo es de NT$1,000,000 y el valor mínimo es de NT$10,000. En la Figura 4.2 se muestra un histograma con las frecuencias de los montos.

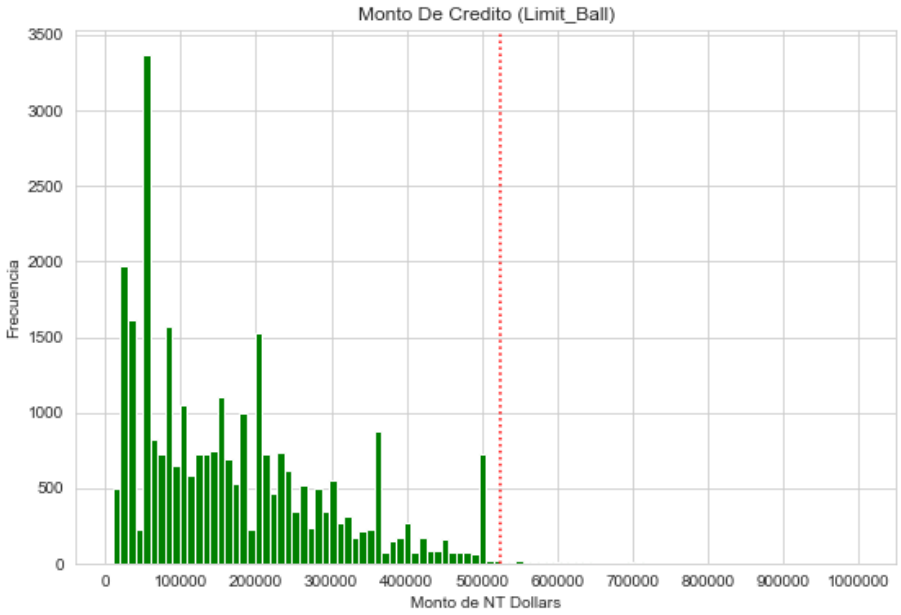

Figura 4.2: Histograma de la variable LIMIT_BALL.

Como se puede observar en la figura anterior existen montos que son más frecuentes, los cuales son NT$50,000 (11.21 %), NT$20,000 (6.58 %), NT$30,000 (5.36 %), NT$80,000 (5.22$) y NT$200,000 (5.09 %), los cuales conforman un 33.48 % de los datos. La línea vertical roja de la Figura 1.1 significa que a partir de este valor las cantidades más grandes pueden ser consideradas como valores atípicos, ya que un valor se considera atípico cuando es mayor que $Q3 + 1.5 * IQR$ (donde Q3 = 3er cuartil, IQR = Rango intercuartil), en este caso el valor es NT$525,000, una verificación rápida revela que los valores atípicos simplemente son clientes con mucho dinero (se hizo esto verificando las variables de montos de pago mayores a NT$300,000). Aun así, se sugiere filtrar observaciones con límites superiores a NT$750,000 ya que algunos algoritmos son sensibles a escalas.

A continuación se mostrarán 3 tablas de las frecuencias de las variables cualitativas **Sexo, Matrimonio y Educación**.

	Frecuencia	Porcentaje
Hombre	11,888	39.6 %
Mujer	18,112	60.4 %
Total	30,000	100 %

Tabla 4.1: Frecuencias de Género.

	Frecuencia	Porcentaje
Posgrado	10,585	35.3 %
Universidad	14,030	46.8 %
Preparatoria	4,917	16.4 %
Otros	468	1.5 %
Total	30,000	100 %

Tabla 4.2: Frecuencias de Educación.

	Frecuencia	Porcentaje
Casado	13,659	45.5 %
Soltero	15,964	53.2 %
Otro	377	1.3 %
Total	30,000	100 %

Tabla 4.3: Frecuencias del Estado Civil.

Observando las tablas de frecuencia de Matrimonio y Educación se puede ver que existen valores que no se encuentran en la documentación. Para el caso de Educación la documentación solo contiene [1,2,3,4] pero en la tabla se registran los valores [0,5,6], y en el caso de Matrimonio los valores documentados son [1,2,3], pero se puede ver el valor 0. En los problemas de Machine Learning la mayoría de veces los datos no vienen en la manera deseada, el proceso de preparar en una forma adecuada la base de datos se llama **limpieza de datos**. Como no se tiene más información de lo que pueden significar estos valores no documentados, en este trabajo se cambiarán [0,5,6] al valor de 4 en la variable Educación y a 0 el valor de 3 en la variable Matrimonio. En la Figura 4.3 se muestran los 3 gráficos de las variables una vez modificadas.

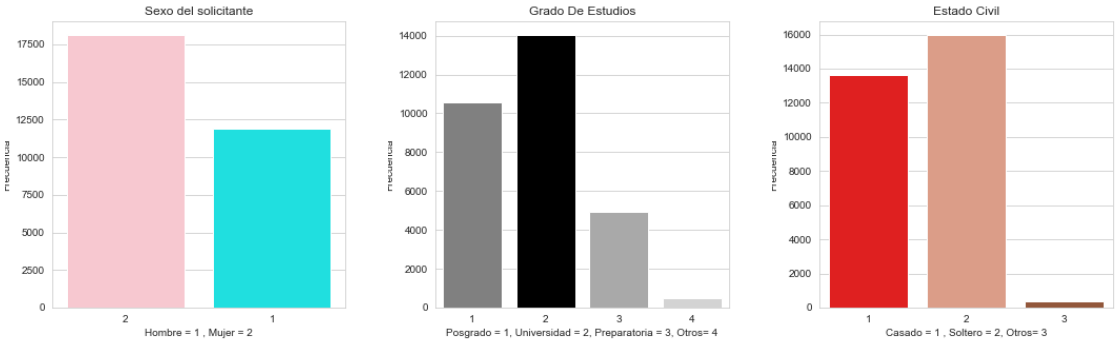

Figura 4.3: Gráficas de barras de las variables SEX, EDUCATION Y MARRIAGE.

Para la variable Sexo del cliente se tiene un porcentaje mayor por parte de las mujeres con un 60.37 %, en el caso de Educación la mayor parte de clientes tiene un nivel universitario los cuales conforman un 46.76 %, como observación el valor de otro en educación puede ser una educación inferior al nivel secundaria, y finalmente la mayoría de personas son solteras las cuales son un 53.21 %. Una observación en este último caso, es que parece que el estado Casado 3 (otros), la media de la edad es mayor a 40 años, esto podría significar que son viudos o divorciados.

En la Figura 4.4 se muestra un histograma de la frecuencia de la edad de los clientes, la edad mínima es de 21 años y la máxima es de 79 años, los solicitantes tienen una media de 34 años con una desviación estándar de 9 años. La línea roja como en el caso del Monto De Crédito es para detectar valores atípicos, en este caso son valores mayores a 61 años.

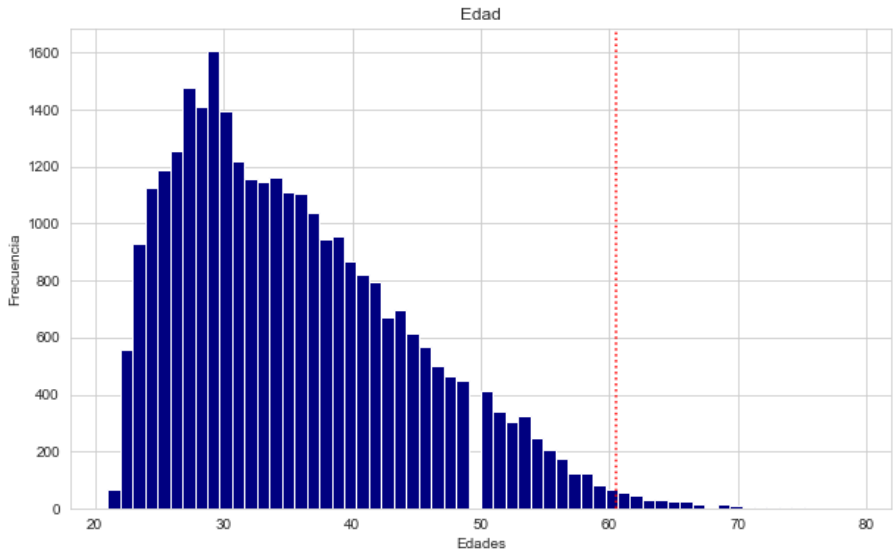

Figura 4.4: Histograma de la variable AGE.

La variable **Pago** es el reembolso, que es el acto de devolver dinero prestado previamente de un prestamista. Por lo general, la devolución de fondos se realiza a través de pagos periódicos que incluyen capital e intereses. Los préstamos generalmente también pueden pagarse en su totalidad en una suma global en cualquier momento, es importante saber esto ya que para las variables Pago también existen valores no documentados como es el caso de [-2,0], y estos forman aproximadamente un 58.5 % del total de datos de las variables de Pago, por lo que una buena idea es investigar que significan estos valores. Una opción es, si no se realizó consumo el valor de -2, si el crédito fue pagado el valor de -1 y si se hizo uso de crédito revolvente el valor 0.

En la Figura 4.5 se pueden ver los gráficos de los Pagos de los meses de abril del 2005 a septiembre del 2005.

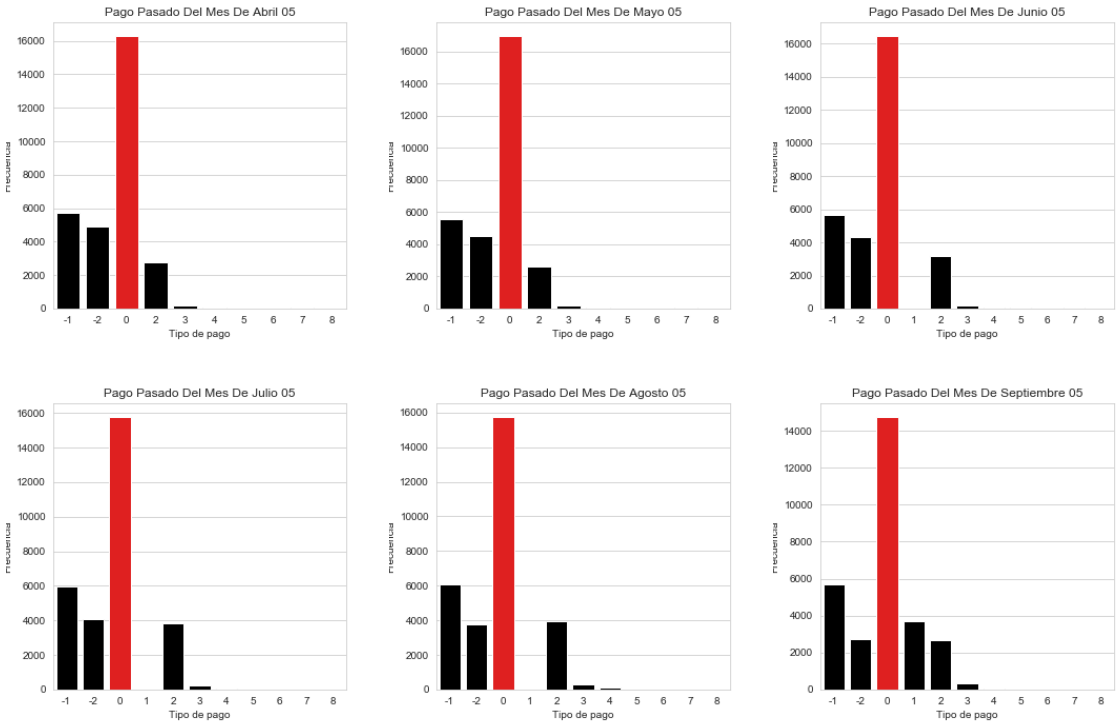

Figura 4.5: Gráficas de barras de las variables PAY_0 - PAY_6.

Se puede observar que la mayoría de los clientes usa un Crédito revolvente y paga debidamente en todos los meses, esto tiene sentido ya que esta muestra de datos en su mayoría es de clientes no morosos.

La variable **Monto Del Estado De Cuenta** muestra el saldo de crédito de cada mes, desde Abril del 2005 a Septiembre del 2005, a continuación, en la Figura 4.6 se muestran los histogramas de los estados de cuenta junto con la Tabla 4.4 que contiene sus estadísticos descriptivos.

	Promedio	Desviación std	Máximo	Mínimo
Septiembre 05	51,223	73,635	964,511	-165,580
Agosto 05	49,179	71,173	983,931	-69,777
Julio 05	47,013	69,349	1,664,089	-157,264
Junio 05	43,262	64,332	891,586	-170,000
Mayo 05	40,311	60,797	927,171	-81,334
Abril 05	38,871	59,554	961,664	-339,603

Tabla 4.4: Estadísticos Descriptivos.

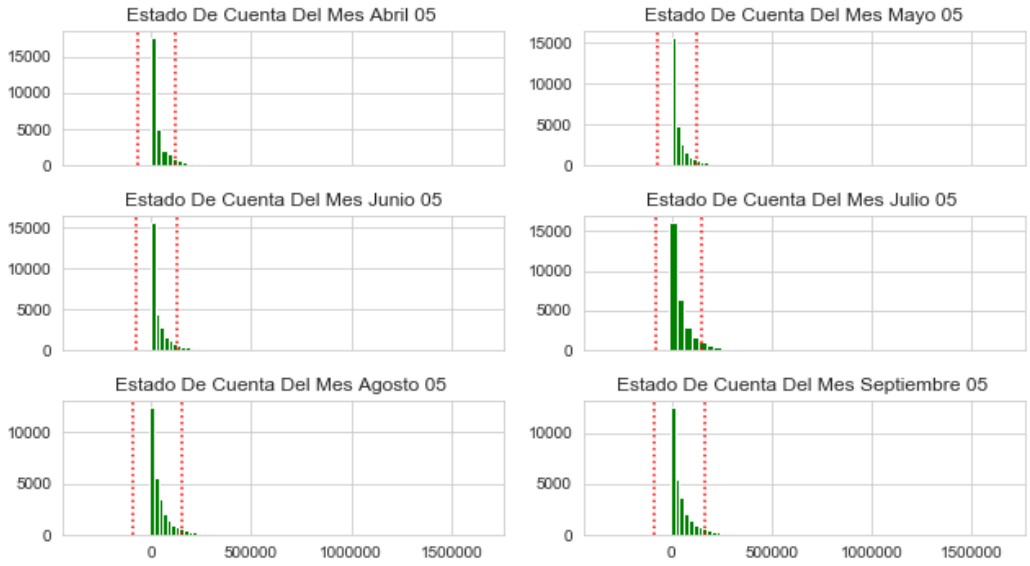

Figura 4.6: Histogramas de las variables BILL_AMT1 - BILL_AMT6.

Las dos líneas rojas son para detectar valores atípicos, las cuales están dadas por $Q3+1.5*IQR$ y $Q1-1.5*IRQ$. El rango intercuartil (IQR) es una medida de variabilidad, basada en dividir un conjunto de datos en cuartiles. Los cuartiles dividen un conjunto de datos ordenado por rango en cuatro partes iguales. Los histogramas generados muestran que los intervalos de *Estado De Cuenta* más comunes son inferiores a NT$1,000. Existen valores negativos, estos significan que pagaron más dinero del que se necesitaba. El promedio de los *Montos De Credito* es NT$44,976 y las desviaciones estándar de cada mes son bastante grandes, también se puede notar que el promedio de cada mes va aumentando cada mes que pasa.

Para la variable **Monto De Pago Previo** se muestran los histogramas con las frecuencias de los pagos que se hicieron. Para estas variables hay una acumulación de datos en las cifras menores que NT$5,000, para esto se ampliaran los histogramas, para ver con más detalle estos datos. En la Figura 4.7 se muestran los histogramas enfocados a los datos menores a NT$6,000.

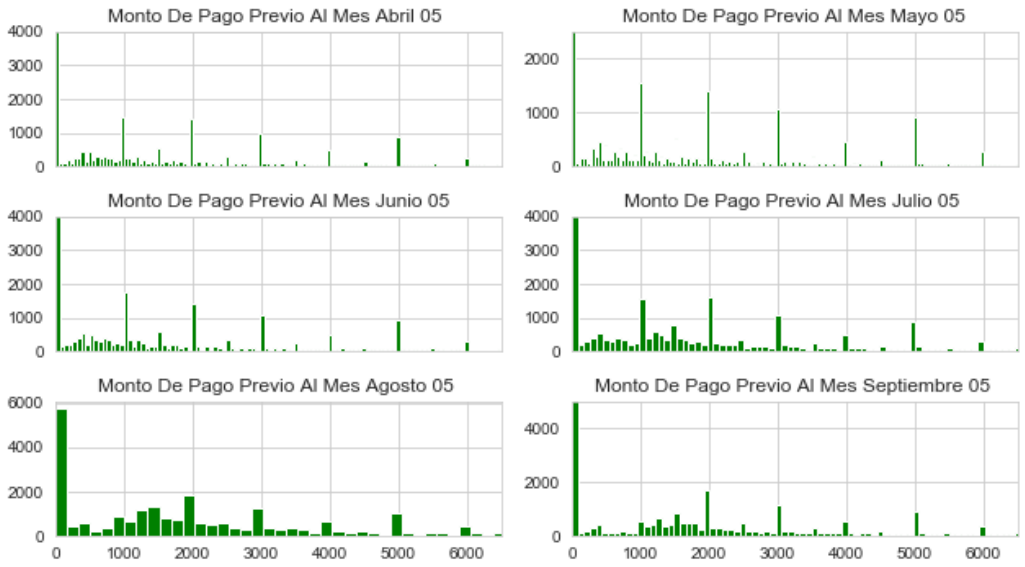

Figura 4.7: Histogramas de las variables PAY_AMT1 - PAY_AMT6.

Al observar de cerca los histogramas para pagos inferiores a NT$5,000 parece que las personas prefieren pagar cifras redondas de dinero, las principales 5 cifras son NT$0, NT$1000, NT$2000, NT$3000 y NT$5000.

	Promedio	Desviación std	Máximo	Mínimo
Septiembre 05	5,663	16,563	873,552	0
Agosto 05	5,921	23,040	1,684,259	0
Julio 05	5,225	17,606	896,040	0
Junio 05	4,826	15,666	621,000	0
Mayo 05	4,799	15,278	426,529	0
Abril 05	5,215	17,777	528,666	0

Tabla 4.5: Estadísticos Descriptivos.

En la Tabla 4.5 se pueden ver los estadísticos descriptivos de las variables *Monto De Pago Previo*, donde se puede observar que los meses de Agosto y Julio fueron los que tuvieron valores más altos.

Por último, se analizará la variable respuesta, si el cliente es moroso, o no, es decir, si incurre o no en el no pago. En la Figura 4.8 se muestra una gráfica con la frecuencia de los solicitantes.

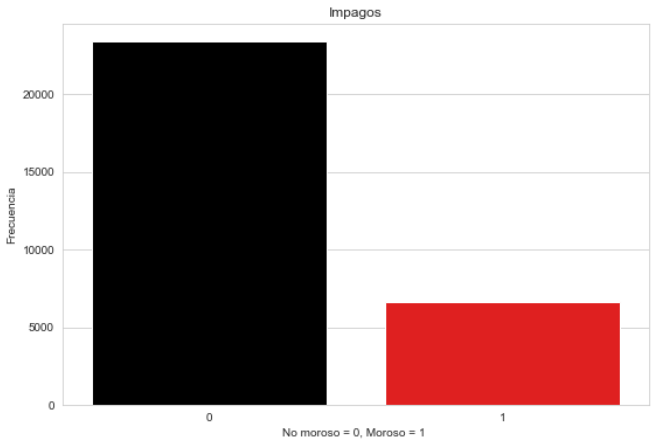

Figura 4.8: Gráfica de barras de clientes con impago.

Se tiene que el 77.88 % de clientes que no son morosos, se puede observar claramente que se tiene un desbalance de los datos con respecto a la variable respuesta, pero en los problemas de Machine Learning es común trabajar con datos desbalanceados, y un número de 6,636 de 30,000, es decir un 22 % de clientes que incumplirán con sus pagos el próximo mes no tiene un gran desequilibrio con respecto al valor objetivo. Existen diferentes formas de trabajar con datos desbalanceados, pero no se verán en este trabajo.

4.3. Aplicación del Credit Scoring

Una vez que se realizó un análisis exploratorio de datos, se tiene una mejor idea sobre las variables, y además los datos están en la forma adecuada para poder implementarse en los modelos, y así poder avanzar. En esta sección se implementarán los 3 modelos de Machine Learning antes vistos, para la creación de un modelo de Credit Scoring que permita predecir si un cliente pagará o no el siguiente mes. Se utilizará la base de datos con la que se trabajó en el EDA (c.f [27]). En esta base de datos se tienen 30,000 observaciones y 23 características con una variable respuesta. En este trabajo no se realizará selección de características, ingeniería de características (feature engineering), ni reducción de dimensión en los datos.

Este problema es una tarea de aprendizaje supervisado ya que se tienen las etiquetas correspondientes a cada cliente, por lo que se entrenará al modelo para poder clasificar la etiqueta de cada uno. Para la creación de los modelos se utilizó el lenguaje de programación python (c.f [17]) y las librerías de scikit learn (c.f [21]) y Tensor Flow (c.f [23]), las cuales son librerías para aprendizaje automático de software libre. Se programarán los modelos de regresión logística, Árboles de decisión y redes neuronales para la clasificación de los clientes en moroso y no moroso. En el Apéndice A se muestrá el código de cada algoritmo con la explicación de los parámetros elegidos.

Como se vio en el Capítulo 1, los modelos de ML tienen el mismo proceso, lo único que cambia es el modelo estadístico elegido, por lo cual, el proceso con los 3 modelos fue el mismo. Primero, se dividió el conjunto de datos total en dos conjuntos, se tomó el 70 % de los datos para el conjunto de entrenamiento y el 30 % restante se convirtió en el conjunto de prueba. En la tabla 4.6 se muestra como quedaron divididos los conjuntos.

	Observaciones	Porcentaje
Conjunto De Entrenamiento	21,000	70 %
Conjunto De Prueba	9,000	30 %
Total	30,000	100 %

Tabla 4.6: Conjunto de Entrenamiento y Prueba.

El conjunto de entrenamiento se utilizará para que los 3 modelos aprendan y posteriormente puedan hacer predicciones con el conjunto de prueba.

4.3.1. Resultados

Una vez entrenados los modelos se utilizará el conjunto de prueba para ver que tan bien actúan con datos no vistos anteriormente.

Se comenzó con el modelo de Regresión Logística, a continuación, se muestran los coeficientes de cada variable, junto con la matriz de confusión obtenida.

Variable	Coeficiente	Variable	Coeficiente
Intersección	-0.00050135	LIMIT_BAL	-3.55575587e-06
SEX	-9.07043841e-04	EDUCATION	-1.03370644e-03
MARRIAGE	-1.03082706e-03	AGE	-1.30100463e-02
PAY_0	1.73186742e-03	PAY_2	1.31434988e-03
PAY_3	1.17877635e-03	PAY_4	1.08974497e-03
PAY_5	1.01497635e-03	PAY_6	9.71295345e-04
BILL_AMT1	-8.84391265e-06	BILL_AMT2	5.41797140e-06
BILL_AMT3	2.28467251e-06	BILL_AMT4	8.22458671e-07
BILL_AMT5	3.44145296e-06	BILL_AMT6	1.58705568e-06
PAY_AMT1	-2.75293345e-05	PAY_AMT2	-2.41567208e-05
PAY_AMT3	-1.06481706e-05	PAY_AMT4	-6.33192976e-06
PAY_AMT5	-6.99060940e-06	PAY_AMT6	-1.98194607e-06

Tabla 4.7: Regresión Logística.

	Predicción (1)	Predicción (0)
Observación (1)	0	1942
Observación (0)	2	7056

Tabla 4.8: Matriz de Confusión.

Como se puede observar en la matriz de confusión obtenida, el rendimiento del modelo no fue bueno, a pesar de que la métrica de Accuracy no es la recomendable en estos casos, apenas tuvo un 1 % mejor que si se hubiera elegido simplemente la clase más común, que hubiera sido de un 77.88 %. Este modelo es muy malo para predecir si un cliente es moroso, el cual sería el principal objetivo de un modelo de Credit Scoring, ya que no tuvo ningún acierto en predecir que un cliente moroso fuera moroso, tuvo una cantidad muy grande de **FN**, por lo que si este modelo fuera implementado podría ocasionar perdidas a la empresa, ya que estaría aceptando clientes que posiblemente no pagarían, por lo que no sería recomendable utilizarlo. Por otro lado tuvo

un alto número de **TN**, por lo que este modelo hace demasiadas predicciones de clientes no morosos.

Se continuó con el modelo de Árboles de Decisión, en la figura 4.9 se muestra el gráfico del árbol obtenido y posteriormente su matriz de confusión.

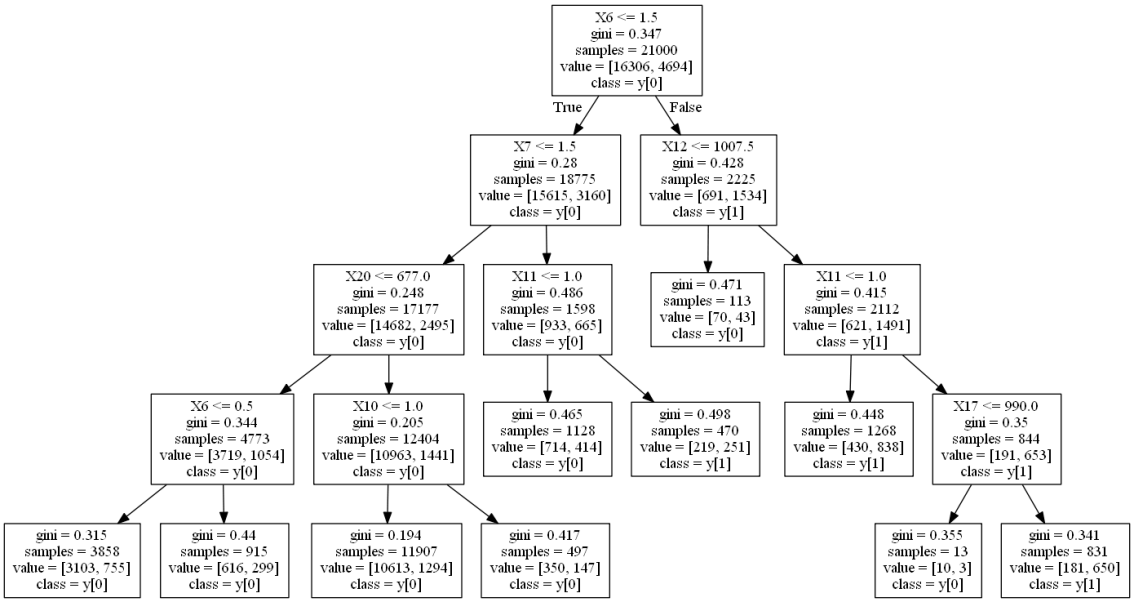

Figura 4.9: Arquitectura del Árbol de Decisión.

	Predicción (1)	Predicción (0)
Observación (1)	709	1233
Observación (0)	339	6719

Tabla 4.9: Matriz de Confusión.

Para la matriz de confusión del Árbol de Decisión, se ve una gran mejora a diferencia de la de regresión logística, este modelo tuvo un mejor rendimiento para predecir **TP**, y tuvo un mejor equilibrio entre **FP** Y **FN**, a pesar de obtener mejores resultados este modelo no es muy bueno, ya que como se verá en la Tabla 4.9 tiene un f1 bastante bajo el cual es una relación entre Precisión y Recall.

Finalmente se culminó con el modelo de redes neuronales, en la Tabla 4.10 se muestra la arquitectura de la red que se eligió junto con su matriz de confusión. Para la estructura de la red se eligieron 3 capas ocultas, en la primera capa oculta se encuentran 15 neuronas con la función de activación Sigmoide, en la segunda capa oculta se encuentran 10 neuronas con la función de activación Relu, en la tercera capa oculta se encuentran 5 neuronas con la función de activación Relu y por último en la capa de salida se eligió 1 neurona con la función de activación Sigmoide para representar la probabilidad de que el cliente sea moroso o no.

Capa de entrada	variables de entrada	23
Capa oculta 1	número de neuronas, Función de activación	15, Sigmoide
Capa oculta 2	número de neuronas , Función de activación	10 , Relu
Capa oculta 3	número de neuronas , Función de activación	5, Relu
Capa de salida	número de neuronas, Función de activación	1, Sigmoide

Tabla 4.10: Arquitectura de la red neuronal.

	Predicción (1)	Predicción (0)
Observación (1)	713	1229
Observación (0)	348	6710

Tabla 4.11: Matriz de Confusión.

Como se puede ver en la matriz de confusión de la Red Neuronal, el rendimiento fue bastante parecido al del modelo de Árbol de Decisión a diferencia de pequeñas predicciones, por lo que no es un gran modelo tampoco, en la tabla 4.12 se muestra una comparativa del rendimiento de los 3 modelos sujeto a las distintas métricas vistas anteriormente y el tiempo de entrenamiento.

Métrica	Regresión Logística	Árbol de Decisión	Redes Neuronales
Accuracy	78.4 %	82.5 %	82.4 %
Specifity	0.99	0.95	0.95
Recall	0	0.36	0.36
Precisión	0	0.67	0.67
F1	0	0.474	0.474
AUC ROC	0.49	0.6585	0.6589
Tiempo de entrenamiento	1.1 s	101 ms	31 s

Tabla 4.12: Comparativa de modelos.

Viendo la comparativa de los 3 modelos ante diferentes métricas, se puede notar que el modelo con el rendimiento más bajo es el de Regresión logística, ya que es muy malo para predecir correctamente a los clientes morosos, y el cual es uno de los objetivos de crear un modelo de Credit Scoring, los únicos aspectos positivos fueron el tiempo de entrenamiento y que predijo un porcentaje alto de clientes no morosos, pero basándonos en lo mencionado anteriormente, se nota que el modelo no aprendió correctamente ni generalizo correctamente los patrones de los datos. El modelo de Árbol de Decisión y el modelo de Redes Neuronales tuvieron desempeños parecidos, el único punto diferente fue el tiempo de entrenamiento, donde el Árbol de decisión toma ventaja, pero en cuento a predicción de clientes morosos y no morosos tuvieron un desempeño parecido, pero como se vio anteriormente en la sección de Capacidad predictiva del modelo, se busca tener un buen balance entre **FP** Y **FN**, lo cual está ligado con la Precisión y Recall, y como se puede observar en la Tabla 4.12 se tiene un valor bajo tanto en Precisión como en Recall. A pesar de no tener el rendimiento deseado, es un buen comienzo, como se vio anteriormente los modelos de Machine Learning son un ciclo, el cual puede repetir los pasos hasta llegar a un objetivo deseado, por lo que se podrían hacer más cosas en la preparación de los datos para obtener mejores resultados. Por estas razones la elección del modelo en estas instancias sería el modelo de Árbol de Decisión.

4.4. Conclusiones

Los datos son un activo muy importante actualmente, las personas que saben cómo convertirlos en información están sacando ventaja de las que todavía no lo hacen, la acción de recopilar datos para su posterior análisis no es nuevo, sin embargo, la cantidad de datos que se crean cada día es una cantidad masiva, lo que se definió como Big Data permite enfrentarse a otro tipo de problemas y tareas. Para poder convertir los datos en información surgen los términos de Inteligencia Artificial y Machine Learning. Es muy importante entender la diferencia entre lo que se pensaba que era Inteligencia Artificial y lo que verdaderamente es, en mi opinión el efecto AI effect no parará hasta llegar a una Inteligencia Artificial fuerte, donde más que recrear a una mente, la computadora sea una mente, igual o superior a la mente humana. En los últimos años se ha avanzado a pasos agigantados, pero todavía falta mucho camino por recorrer. Los modelos de Machine Learning son los que están ayudando a automatizar tareas y procesos, y así poder ahorrar tiempo y trabajo del que normalmente se llevaría una persona común.

Aunque existen 4 tipos de aprendizaje automático, los mas comunes son el aprendizaje supervisado y no supervisado, el objetivo de enfocarse en el aprendizaje supervisado fue el de comprender mejor el proceso ya que es el mas utilizado en problemas de la vida real. Para poder lograr un buen modelo se necesita trabajar correctamente cada parte del proceso, siendo una de las más importantes la preparación de datos, si no se puede contar con datos adecuados por más que mejoremos en todos los aspectos nunca se obtendrá el rendimiento adecuado, o peor aún, se puede caer en patrones equivocados, existe una frase que dice que si se mete basura, sale basura. Tener unos datos de la forma adecuada para entrenar los modelos será un requisito indispensable. Los modelos de ML ayudan a tener una vida más "fácil", un ejemplo de las aplicaciones es el Credit Scoring, tener un buen modelo puede ser beneficioso tanto para la entidad crediticia como para los solicitantes, por parte de la entidad, se puede ahorrar una cantidad de dinero, así como evitar pérdidas y en el caso del solicitante una respuesta rápida e inclusive una posible deuda que no pudiera pagar en el futuro.

La automatización en la toma de decisiones crediticias permite a las entidades poder ahorrar tiempo y dinero a la hora de aceptar nuevos clientes. Para la creación del modelo de Credit Scoring se usaron 3 modelos estadísticos, Regresión logística, Árboles de Decisión y Redes Neuronales. En general los resultados obtenidos no fueron los mejores, ya que este trabajo fue enfocado al proceso de la creación de un modelo de ML, por lo cual no se fijó como objetivo el aumentar el rendimiento en alguna métrica, en especial utilizando técnicas que maximizarán los resultados para no obtener resultados sesgados.

Los resultados del modelo de regresión estuvieron por arriba del porcentaje de la mayor clase del conjunto, fue muy bueno para predecir los clientes que no eran morosos ya que tuvo una especificidad de 0.99, pero fue demasiado malo para predecir clientes morosos por lo cual tiene una sensibilidad de 0, tuvo un AUC ROC de 0.49 el cual es un mal resultado. La ventaja de este modelo es que no es muy complejo por lo que el tiempo de entrenamiento es bastante pequeño. Como el modelo no es muy complejo, una opción que podría mejorar el rendimiento es la reducción de dimensionalidad, ya que tener una gran cantidad de predictores puede causarle ruido al modelo y no permitirle encontrar los patrones correctos de los datos.

El modelo de redes neuronales también estuvo por arriba del porcentaje de la mayor clase del conjunto, fue muy bueno para predecir clientes que no son morosos, tuvo un mejor desempeño que el modelo de regresión para predecir los clientes morosos, pero aún así bastante bajo, tuvo un 0.47 de f1 score, el cual no es muy alto. La contra más grande para este modelo es el tiempo de entrenamiento, ya que supera por bastante a los otros dos. Una manera que podría ayudar a tener

un mejor rendimiento del modelo sería elegir una arquitectura diferente a la red, pero se tendrá que tener en cuenta que más capas podrían llevar a tener mejores resultados, pero un tiempo de entrenamiento mucho mayor.

El modelo que en promedio lo hizo mejor con un menor tiempo de entrenamiento fue el Árbol de Decisión, tuvo un porcentaje mayor que la mayor clase del conjunto, tuvo resultados parecidos a los de Redes Neuronales, pero con un mejor tiempo de entrenamiento. Si se tuviera que elegir alguno de los 3 modelos, el modelo de Árbol de Decisión sería el que tiene en promedio un rendimiento mejor, por lo que su elección sería la más adecuada para este caso de estudio. Como se vio en la sección de Árboles de Decisión una manera de obtener mejores resultados puede ser utilizar técnicas como Ramdom Forest y Bagging.

En los modelos de Credit Scoring que se usan en casos reales es necesario que el porcentaje de clasificación incorrecta de clientes morosos sea el mínimo posible, ya que se considera que este tipo de mala clasificación representa el mayor costo, ya que el que los clientes no paguen representa pérdidas. Los modelos por default tienen un cut off de 0.5, sería interesante intentar cambiar este punto para poder ver si varían los resultados.

Para un futuro trabajo sería una buena idea empezar con algún objetivo específico, elegir algunos métodos de muestreo, selección de características e ingeniería de características para maximizar los resultados. Un buen modelo de Machine Learning es aquel que aprendió correctamente los patrones de los datos y puede hacer buenas predicciones ante datos no vistos anteriormente, es aquel que tiene un buen balance en las predicciones, por ejemplo, en el caso de estudio un buen modelo sería aquel que predijera correctamente tanto clientes morosos como no morosos. Para crear un buen modelo se tienen que tomar en cuenta que tipo de datos se tienen, para elegir correctamente las métricas a utilizar.

Apéndice A

TensorFlow, Keras y Sckit Learn

Scikit-learn es una biblioteca de software de aprendizaje automático para el lenguaje de programación Python. Este paquete se centra en llevar el aprendizaje automático a los no especialistas que utilizan un lenguaje de alto nivel de propósito general. Se enfatiza la facilidad de uso, el rendimiento, la documentación y la coherencia de la API (Application Programming Interface). Tiene dependencias mínimas y se distribuye bajo la licencia BSD (Berkeley Software Distribution) simplificada, lo que fomenta su uso en entornos académicos y comerciales. El código fuente, los archivos binarios y la documentación se pueden descargar desde http://scikit-learn.sourceforge.net

TensorFlow es una interfaz para expresar algoritmos de aprendizaje automático y una implementación para ejecutar dichos algoritmos. Un cálculo expresado con TensorFlow se puede ejecutar con poco o ningún cambio en una amplia variedad de sistemas heterogéneos, desde dispositivos móviles como teléfonos y tabletas hasta sistemas distribuidos a gran escala de cientos de máquinas y miles de dispositivos computacionales como tarjetas GPU. El sistema es flexible y se puede utilizar para expresar una amplia variedad de algoritmos, incluidos los algoritmos de entrenamiento e inferencia para modelos de redes neuronales profundas, y se ha utilizado para realizar investigaciones y desplegar sistemas de aprendizaje automático en producción en más de una docena de áreas de ciencias de la computación y otros campos, incluidos el reconocimiento de voz, la visión por computadora, la robótica, la recuperación de información, el procesamiento del lenguaje natural, la extracción de información geográfica y el descubrimiento computacional de drogas. Este documento describe la interfaz TensorFlow y una implementación de esa interfaz que hemos creado en Google. La API de TensorFlow y una implementación de referencia se lanzaron como un paquete de código abierto bajo la licencia Apache 2.0 en noviembre de 2015 y están disponibles en www.tensorflow.org.

Keras es una biblioteca de Redes Neuronales de Código Abierto escrita en Python. Es capaz de ejecutarse sobre TensorFlow.

En este trabajo se utilizaron las bibliotecas Scikit Learn y Keras para la creación de los modelos de ML. A continuación, se mostrará el código para cada modelo junto con la explicación de los parámetros elegidos.

Para los 3 modelos se utilizo la función **train_test_split()** con los mismos parámetros: **X, y, test_size=0.3, random_state=101**.

Lo que hace esta función es dividir el conjunto de datos total en un conjunto de entrenamiento y un conjunto de prueba. Los parámetros elegidos tienen la siguiente función:

- **Arrays**: Las entradas permitidas son listas, numpy arrays, matrices scipy-sparse ó pandas dataframes. En este caso las entradas fueron 2 data frames. **X** son las características e **y** las etiquetas.
- **test_size**: Representa la proporción del conjunto de datos a incluir en el conjunto de prueba, se toman valores entre 0 y 1. Para estos modelos se utilizó 0.3 ya que se quería el 30 % de los datos para hacer predicciones.
- **random_state**: Controla la combinación aleatoria aplicada a los datos antes de aplicar la división.
- **shuffle**: Si se barajan o no los datos antes de dividirlos. Por default se barajan los datos.

Para el modelo de Regresión Logística se utilizó el clasificador **LogisticRegression()**, en este caso no se usaron parámetros para la creación de este modelo. Una vez creado el modelo, este se entrena con los datos de entrenamiento utilizando la función **fit()**, para posteriormente hacer predicciones con el conjunto de prueba usando la función **predict()**.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

features =['X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X7', 'X8', 'X9','X10',
           'X11', 'X12', 'X13', 'X14', 'X15', 'X16', 'X17', 'X18', 'X19',
           'X20', 'X21', 'X22', 'X23']

X= datos[features]
y= datos["Y"]

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state=101)


logmodel = LogisticRegression()

logmodel.fit(X_train,y_train)

prediction = logmodel.predict(X_test)
```

Figura A.1: Código Regresión Logística.

Para el modelo de Árboles de Decisión se utilizó el clasificador **DecisionTreeClassifier()**, para este modelo se eligieron los siguientes parámetros:

- **criterion**: La función para medir la calidad de una división. Los criterios admitidos son Gini para la impureza de Gini y entropía para la ganancia de información.
- **max_depth**: La profundidad máxima del árbol. Si no se especifica, los nodos se expanden hasta que todas las hojas sean puras o hasta que todas las hojas contengan menos de **min_samples_split** samples.
- **max_features**: La cantidad de características a considerar cuando se busca la mejor división. *None* es para tomar todas las características.
- **max_leaf_nodes**: Cultiva un árbol con **max_leaf_nodes** de la mejor manera. Los mejores nodos se definen como la reducción relativa de impurezas. Si se elige *None*, entonces hay un número ilimitado de nodos hoja.

- **min_impurity_decrease**: Un nodo se dividirá si esta división induce una disminución de la impureza mayor o igual a este valor.
- **min_impurity_split**: Umbral para detenerse temprano en el crecimiento de los árboles. Un nodo se dividirá si su impureza está por encima del umbral; de lo contrario, es una hoja.
- **min_samples_leaf**: El número mínimo de muestras necesarias para estar en un nodo hoja. Un punto de división a cualquier profundidad sólo se considerará si deja al menos **min_samples_leaf** muestras de entrenamiento en cada una de las ramas izquierda y derecha. Esto puede tener el efecto de suavizar el modelo.
- **min_samples_split**: El número mínimo de muestras necesarias para dividir un nodo interno.
- **min_weight_fraction_leaf**: La fracción mínima ponderada de la suma total de pesos (de todas las muestras de entrada) requerida para estar en un nodo hoja. Las muestras tienen el mismo peso cuando no se proporciona **sample_weight**.
- **splitter**: La estrategia utilizada para elegir la división en cada nodo. Las estrategias admitidas son *best* para elegir la mejor división y *random* para elegir la mejor división aleatoria.

Después de crear el modelo, como se hizo anteriormente, se entrena el modelo usando la función **fit()**, y para hacer predicciones la función **predict()**.

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

features =['X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X7', 'X8', 'X9','X10',
           'X11', 'X12', 'X13', 'X14', 'X15', 'X16', 'X17', 'X18', 'X19',
           'X20', 'X21', 'X22', 'X23']

X= datos[features]
y= datos["Y"]

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state=101)

dtre = DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=3,
            max_features=None, max_leaf_nodes=10,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')

dtre.fit(X_train,y_train)

predictions = dtre.predict(X_test)
```

Figura A.2: Codigo Árbol de Decisión.

Para la creación del modelo de Redes Neuronales se utilizó la librería Keras de TensorFlow, antes de entrenar al modelo con el conjunto de datos, estos se transforman a una escala entre 0 y 1, la cual está dada por:

Xstd = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
Xscaled = Xstd * (max - min) + min .

Para poder crear la red se utilizó el método **Sequential()** y para agregar el número de capas se utilizó la función **add()**, en cada capa se utilizó la función **Dense()**, este método hace el proceso de la evaluación de la suma ponderada de los valores de entrada de cada neurona con sus pesos, para posteriormente evaluarla en la función de activación de cada neurona, los parámetros que se usaron son:

- **Units**: Entero positivo, es el número de neuronas en la capa.
- **activation**: Es la función de activación elegida para cada capa. En caso de no elegir ninguna, por default se utiliza la función lineal.

Antes de entrenar a la red, se tienen que configurar los parámetros con los cuales va a ser entrenada, para esto se utilizará **compile()**, los parámetros elegidos fueron:

- **loss**: El nombre de la función de costos.
- **optimizer**: El nombre del optimizador, el optimizador que se eligió se llama adam y es una variante del gradient decent con un learning rate específico.
- **metrics**: Lista de métricas a ser evaluadas por el modelo durante el entrenamiento y las pruebas.

Una vez configurado, se puede entrenar el modelo con la función **fit()** al igual que con los modelos anteriores, pero a diferencia de los anteriores se agregó el parámetro *epochs* que es el número de veces que se realizara el Backpropagation. Por último se realizan las predicciones del conjunto de prueba con la función **predict_classes()**.

```
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.preprocessing import MinMaxScaler

features =['X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X7', 'X8', 'X9','X10', 'X11', 'X12', 'X13',
           'X14', 'X15', 'X16', 'X17', 'X18', 'X19','X20', 'X21', 'X22', 'X23']
X= datos[features]
y= datos["Y"].values

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state=101)
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

red = Sequential()
red.add(Dense(15,activation="sigmoid"))
red.add(Dense(10,activation="relu"))
red.add(Dense(5,activation="relu"))
red.add(Dense(1,activation="sigmoid"))

red.compile(loss="binary_crossentropy",optimizer = "adam", metrics=['accuracy','AUC'])
red.fit(x=X_train,y=y_train,epochs= 30,validation_data =(X_test,y_test))

predictions = red.predict_classes(X_test)
```

Figura A.3: Codigo Red Neuronal.

Apéndice B

Validación Cruzada (k fold Cross-Validation)

La validación cruzada es un método estadístico utilizado para estimar la habilidad de los modelos de aprendizaje automático. Es un procedimiento de remuestreo utilizado para evaluar modelos de aprendizaje automático en una muestra de datos limitada.

Aprender los parámetros de una función de predicción y probarlo con los mismos datos es un error metodológico, un modelo que simplemente repita las etiquetas de las muestras que acaba de ver tendría una puntuación perfecta pero no podría predecir nada útil todavía. Por lo que se lleva a un sobreajuste. Para evitarlo, es una práctica común, cuando se realiza un experimento de aprendizaje automático (supervisado) mantener parte de los datos disponibles como un conjunto de pruebas.

Para resolver este problema, otra parte del conjunto de datos se puede presentar como un conjunto de validación, el entrenamiento continúa en el conjunto de entrenamiento, después de lo cual se realiza la evaluación en el conjunto de validación y cuando el experimento parece ser exitoso, la evaluación final se puede hacer en el conjunto de prueba. Sin embargo, al dividir los datos disponibles en tres conjuntos, se reduce drásticamente el número de muestras que se pueden usar para aprender el modelo, y los resultados pueden depender de una elección aleatoria particular para el par de conjuntos (entrenamiento, prueba).

Una solución a este problema es un procedimiento llamado validación cruzada (CV). Un conjunto de prueba aún debe extenderse para la evaluación final, pero el conjunto de validación ya no es necesario al hacer CV. En el enfoque básico, llamado k-fold CV, el conjunto de entrenamiento se divide en k conjuntos más pequeños. Una vez divididos se sigue el siguiente procedimiento para cada uno de los k conjuntos:

Se entrena un modelo utilizando los k-1 conjuntos como datos de entrenamiento. El modelo resultante se valida en la parte restante de los datos (es decir, se utiliza como un conjunto de prueba para calcular una medida de rendimiento como la precisión). En la figura B.1 se muestra cómo se realiza esto.

La medida de rendimiento informada por la validación cruzada k-fold es entonces el promedio de los valores calculados en el bucle. Este enfoque puede ser computacionalmente costoso, pero no desperdicia demasiados datos (como es el caso al fijar un conjunto de validación arbitrario), lo cual es una gran ventaja en problemas como la inferencia inversa donde el número de muestras es muy pequeño.

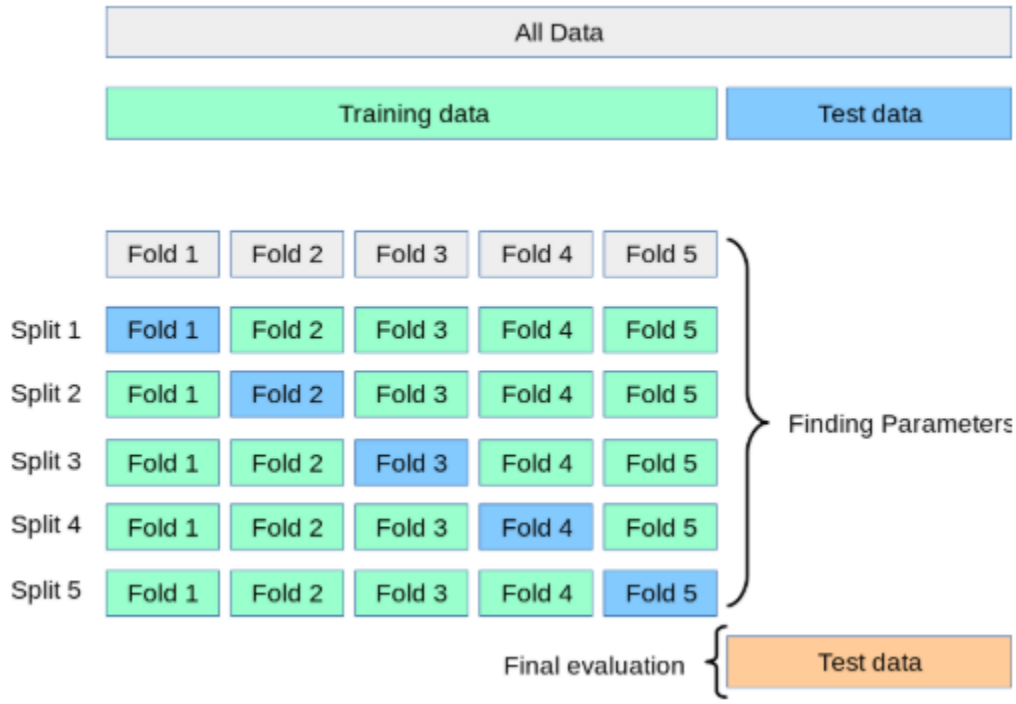

Figura B.1: K fold CV.

Es un método popular porque es simple de entender y porque generalmente da como resultado una estimación menos sesgada o menos optimista de la habilidad del modelo que otros métodos, como una división simple de entrenamiento y prueba. El algoritmo de K Fold CV es el siguiente:

Algoritmo 4: Algoritmo de K Fold Cross Validation

1. Mezcle aleatoriamente el conjunto de datos.
2. Dividir el conjunto de datos en k grupos.
3. Para cada grupo único:
   - Tome el grupo como un conjunto de datos de prueba.
   - Tome los grupos restantes como un conjunto de datos de entrenamiento.
   - Ajuste un modelo en el conjunto de entrenamiento y evalúelo en el conjunto de prueba.
   - Conserve el puntaje de evaluación y descarte el modelo.
4. Resuma la habilidad del modelo utilizando la muestra de puntajes de evaluación del modelo.

Es importante destacar que cada observación en la muestra de datos se asigna a un grupo individual y permanece en ese grupo durante la duración del procedimiento. Esto significa que cada muestra tiene la oportunidad de ser utilizada en el conjunto de espera 1 vez y se usa para entrenar el modelo k-1 veces.

El valor k debe elegirse cuidadosamente para su muestra de datos. Un valor mal elegido para k puede dar como resultado una idea mal representativa de la habilidad del modelo, como un puntaje con una alta varianza (que puede cambiar mucho en función de los datos utilizados para ajustar el modelo), o un alto sesgo, (como una sobreestimación de la habilidad del modelo).

El k generalmente se elige por el número de instancias que se tienen en el conjunto de datos. Por ejemplo, si tiene 10 instancias en sus datos, la validación cruzada 10 veces no tendría sentido. La validación cruzada de k-fold se utiliza para dos propósitos principales, para ajustar los hiper-parámetros y evaluar mejor el rendimiento de un modelo. seleccionar k no es una ciencia exacta porque es difícil estimar qué tan bien el fold representa el conjunto de datos general. Un k pequeño tendrá menor varianza y más sesgo, ya que se toma un porcentaje grande del conjunto de datos, por otro lado, un k más grande tiene más varianza y menor sesgo, ya que se toma un conjunto de datos más pequeño. Por lo que la elección de ese k depende completamente del tamaño de los datos, si el conjunto de entrenamiento es muy grande, elija un valor más alto de k, de lo contrario, elija un valor k más bajo.

Bibliografía

[1] ABDOU, HAH AND POINTON, J. (2011): *Credit Scoring, Statistical techniques and Evaluation Criteria: A Review of the literature*: [En línea] Disponible en http://usir.salford.ac.uk/id/eprint/16518/ : (Visitada el 5.03.20)

[2] ANDERSON, R. (2007): *The Credit Scoring Toolkit: Theory and Practice for Retail Credit Risk Managment and Decision Automation*: Oxford University Press, New York.

[3] BHARGAVA, ADITYA (2016): *Grokking Algorithms: An illustrated guide for programmers and other curious people*: Manning Publications.

[4] CHANG, HSIUNG CHIH (2017): *Credit A Study on the Coping Strategy of Financial Supervisory Organization under Information Asymmetry: Case Study of Taiwan's Credit Card Market*: Universal Journal of Management.

[5] DABOS, MARCELO: *Credit Scoring*: [En línea] Disponible en `https://mba.americaeconomia.com/sites/mba.americaeconomia.com/files/creditscoring.pdf` : (Visitada el 12.02.20)

[6] DRUGOV, VLADIMIR G: *Default Payments of Credit Card Clients in Taiwan from 2005*: [En línea] Disponible en `https://rstudio-pubs-static.s3.amazonaws.com/281390_8a4ea1f1d23043479814ec4a38dbbfd9.html` : (Visitada el 2.02.20)

[7] EL RHAZOUI, NASSIM: *Taiwan´s credit card*: [En línea] Disponible en `https://prezi.com/cvjgdiaotykb/taiwans-credit-card-crisis/` : (Visitada el 05.02.20)

[8] EVGENIOU, THEOS: *Classification for Credit Card Default*: [En línea] Disponible en `http://inseaddataanalytics.github.io/INSEADAnalytics/CourseSessions/ClassificationProcessCreditCardDefault.html` : (Visitada el 05.02.20)

[9] Gareth, James (2013): *An Introduction to Statistical Learning*: Springer.

[10] Geron, Aurelien (2017): *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*: O'Reilly Media.

[11] Grus, Joel (2015): *Data Science from Scratch*: O´Reilly media.

[12] Herrera, Adriana (2017): *Riesgo de Crédito: Análisis Mediante Redes Neuronales (Tesis de Licenciatura)*: Benemérita Universidad Autónoma de Puebla, Puebla.

[13] Islam Rabiul, Sheikh and William Eberle: *Credit Default Mining using combined Machine Learning and Heuristic Approach*: [En línea] Disponible en `https://arxiv.org/ftp/arxiv/papers/1807/1807.01176.pdf`: (Visitada el 08.02.20)

[14] Ji, Cheng: *Credit Card Default Prediction with Logistic Regression*: [En línea] Disponible en `https://medium.com/@guaisang/credit-default-prediction-with-logistic-regression-b5bd89f2799f` : (Visitada el 10.02.20)

[15] Ji, Cheng: *Credit Scoring with Machine learning*: [En línea] Disponible en `https://medium.com/henry-jia/how-to-score-your-credit-1c08dd73e2ed` : (Visitada el 15.02.20)

[16] Jim Goodnight: *Inteligencia Artificial Qué es y por qué es importante*: [En línea] Disponible en `https://www.sas.com/esmx/insights/analytics/what-is-artificial-intelligence.html`: (Visitada el 20.04.20)

[17] Lichman M.: UCI *Machine Learning Repository* : [En línea] Disponible en `https://archive.ics.uci.edu/ml`: (Visitada el 07.02.20)

[18] Python (Versión 3.7.3): *Windows*: `https://www.python.org/downloads/release/python-373/`

[19] Sharma, Sunakshi and Vipul Mchra: *Default Payment Analysis of credit Card*: [En línea] Disponible en https://www.researchgate.net/publication/326171439DefaultPaymentAnalysisofCreditCardClients: (Visitada el 08.02.20)

[20] Skantzos, Nikos y Nicolas Castelein (2016): *Credit Scoring Case studio in data analytics*: Deloitte.

[21] skitlearn (Versión 0.20.3): *Windows*: http://scikit-learn.sourceforge.net

[22] scikit-learn developers: *Cross-validation: evaluating estimator performance*: [En línea] Disponible en https://scikit-learn.org/stable/modules/cross_validation.html: (Visitada el 15.06.20)

[23] tensorflow (Versión 2.0.0): *Windows*: www.tensorflow.org

[24] Vishal, Maini, Samer, Sabri (2017): *Machine learning for humans*: Medium.

[25] Vojtek, Martin: *Credit Scoring Methods*: [En línea] Disponible en https://www.researchgate.net/publication/285873211 : (Visitada el 15.02.20)

[26] Wang Eric: *Taiwan's credit card crisis 2005*: [En línea] Disponible en https://sevenpillarsinstitute.org/case-studies/taiwans-credit-card-crisis/ : (Visitada el 07.02.20)

[27] Yeh, I. C., Lien, C. H. (2009): *The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients*: Expert Systems with Applications