



UNIVERSIDAD LA SALLE

FACULTAD DE NEGOCIOS

Con Reconocimiento de Validez Oficial de Estudios de la Secretaría de Educación
Pública según acuerdo número No 954315 De fecha 30 de Octubre de 1995

ELABORACIÓN DE UN ESTUDIO DE CASO

“INTELIGENCIA ARTIFICIAL PARA PREDECIR DISPOSICIONES EN PYMES”

QUE PARA OBTENER EL GRADO DE MAESTRO EN INGENIERÍA ECONÓMICA Y FINANCIERA

PRESENTA:

Jorge Apodaca Gaona

Asesor:

Dr. Héctor Alonso Olivares Aguayo

Ciudad de México,

2023

Ciudad de México, a 01 de junio de 2023

ESP. GABRIEL NUÑEZ GONZÁLEZ
DIRECTOR DE GESTIÓN ESCOLAR
UNIVERSIDAD LA SALLE
P R E S E N T E

Le informo que el (la) C.

Jorge Apodaca Gaona

Pasante de la Escuela o Facultad: FACULTAD DE NEGOCIOS

de la **UNIVERSIDAD LA SALLE**, de la Maestría en:

INGENIERÍA ECONÓMICA Y FINANCIERA

con reconocimiento de validez oficial de estudios de la Secretaría de Educación

Pública según acuerdo número 954315 De fecha 30 de octubre de 1995

Ha elaborado el trabajo de tesis titulado: “INTELIGENCIA ARTIFICIAL PARA PREDECIR DISPOSICIONES EN PYMES”.

De conformidad con la modalidad para la obtención de grado aprobada para esta Maestría de acuerdo a lo establecido en el Reglamento General de las Universidades La Salle Integrantes del Sistema Educativo de las Universidades la Salle.

Cumplió con todos los requisitos y el trabajo que fue elaborado bajo la conducción del Dr. Héctor Alonso Olivares Aguayo que fungió como asesor, tiene la calidad suficiente para ser la base de sustentación de su Examen de Grado por lo que se le autoriza presentarlo.



Mtro. Markel Israel Lehman Elizondo
Director Facultad de Negocios

A Laura por ayudarme en todo
A mi mamá por todo su esfuerzo

INTELIGENCIA ARTIFICIAL PARA PREDECIR DISPOSICIONES EN PYMES

Jorge Apodaca Gaona



Un Caso de Estudio presentado para obtener el
grado de Maestro en Ingeniería Económica y
Financiera

Asesor: Dr. Héctor Alonso Olivares
Aguayo

Facultad de Negocios
México Marzo 2023

Índice

0.1. Introducción	III
0.2. Antecedentes	IV
0.3. Planteamiento del problema	IV
0.4. Justificación	IV
0.5. Objetivo General	V
0.6. Objetivos específicos	V
1. Marco Teórico	1
1.1. Árboles de Decisión	2
1.2. Redes Neuronales	6
1.3. Métricas de efectividad de los modelos	11
1.4. Metodologías para la aplicación de modelos	14
1.5. Herramientas de software utilizadas	16
2. Marco Metodológico	19
2.1. Sampling	19
2.2. Explore	21
2.3. Modify	27
2.4. Model	28
2.5. Asses	31
3. Implementación del modelo	37
3.1. Plan de implementación	37
3.2. Monetización del proyecto	40
3.3. Evidencia de la puesta en producción	42
4. Discusión y conclusiones	45
4.1. Sobre la IA y su implementación	45
4.2. Sobre los resultados de los modelos	45
4.3. Sobre la implementación	46
A. Anexo de Códigos en R y Python	49

B. Anexo de Definiciones Matemáticas	52
B.1. Algoritmo Ada Boost	52
B.2. Algoritmos de Optimización	54
B.3. Sobre Redes Neuronales	55
B.4. Máquinas de Soporte Vectorial	56

0.1. Introducción

En este texto no solo se propone una solución basada en Inteligencia Artificial, sino que se presentan los pasos que se siguieron para su implementación y seguimiento en plataformas tecnológicas, en la metodología AGILE y respetando marcos legales.

La información usada para mostrar en este caso de estudio es publicada por BBVA por lo que no generará controversias y se cuida el anonimato de los clientes, así como la información estratégica del Banco ya que solo se utiliza la información muestra, de hace dos años y sin datos personales. La implementación del modelo en un ambiente productivo significó diversos retos, en este texto se comentarán los más importantes y cómo fueron solucionados en el momento.

A lo largo del texto son presentados diferentes modelos de Machine Learning y diferentes plataformas de programación, para así encontrar las mejores herramientas de generación e implementación del modelo. Con esta información se abordará una breve discusión sobre cuáles son los beneficios de cada una de las herramientas señaladas. En este caso, se selecciona la plataforma de Big Data Jupyter en un entorno distribuido de Spark y que, a manera de ejemplo, se muestra su funcionamiento en una plataforma de Big Data (Datio¹), es por ello que se puede considerar que este modelo aquí presentado es una aplicación de Big Data que es aplicado en un ambiente de cómputo distribuido.

Como se precisa en (Leo et al., 2019) y en (Kaminskyi et al., 2021), tradicionalmente las aplicaciones de Machine Learning están enfocadas en la contención y evaluación del riesgo. Estos enfoques son derivados de la necesidad de obtener la pérdida esperada, el tamaño de la pérdida y la exposición al impago como lo pide la normativa de Basel Committee 2005. En este trabajo, en lugar de centrarse en la exploración de estos aspectos, se busca analizar la necesidad de crédito por parte del cliente en lugar de examinar su riesgo, por lo que esta indagación es una aportación adicional para generar valor de conocimiento y económico sobre los clientes e investigar, cuáles son las razones por las que una PyME solicita un crédito a una institución bancaria debido a que la información disponible sobre el comportamiento económico de las PyMES es escasa.

Por otra parte, en (Guerra y Castelli, 2021) se ha realizado una investigación que pretende mostrar cuáles son los temas más estudiados en el ámbito bancario. En donde, al evaluar 41 artículos con más citas, se estima que el 29.29 % de ellos se enfocan en Machine Learning y que sólo 2.4 % estudian el riesgo de liquidez, por lo que una de las aportaciones del presente caso de estudio es que el modelo propuesto se encuentra en un nicho de investigación poco frecuente debido a que una de las variables más importantes en el modelo es el flujo de liquidez de la PyME por medio del Machine Learning.

En (Kaminskyi et al., 2021) se presenta una recopilación de algoritmos de Machine Learning que permiten enfrentar los problemas a los que la banca se enfrenta en su operación y estrategia. En (Kaminskyi et al., 2021) se especifica

¹Datio es la plataforma de Big Data que utiliza BBVA para elaborar proyectos de Machine Learning con grandes cantidades de datos.

que el conocimiento del comportamiento del cliente suma valor económico al banco que genera este conocimiento.

0.2. Antecedentes

En los últimos años, la IA ha cobrado mayor relevancia en la segmentación de clientes de la Banca y se han generado múltiples propuestas para implementarla en áreas como seguros, prevención de fraudes o para pronosticar ventas. Las entidades financieras buscan talento adecuado y soluciones tecnológicas para aprovechar los beneficios de los avances en el campo de la IA. Además, ésta tiene que contemplar cambios en las políticas de los productos y actualizarse para poder competir con otros modelos de IA con que cuenta la competencia. Es por ello que cada año BBVA lanza el reto Hackathon en el que pone como objetivo encontrar una solución a problemas específicos de cada producto.

Este reto es de carácter abierto; esto es, equipos de especialistas en tecnología generan ideas e innovación para presentar una posible solución. En este caso, debido a la experiencia del autor de este texto en el producto PyME y a la necesidad del Banco de encontrar solución al problema de pronóstico de disposiciones del crédito (necesidad que expone públicamente en el reto) se emprenderá una propuesta de solución al tema y que fue aplicada por el autor de este texto en la empresa con resultados medibles.

0.3. Planteamiento del problema

El problema es descrito tanto por la institución (BBVA. S.A.) como por el autor de este trabajo como se muestra a continuación:

Con base en un conjunto de datos (anónimos) que representan el comportamiento financiero de clientes de Pequeñas y Medianas Empresas (PyME), se requiere encontrar cuáles de estos clientes dispondrán de su tarjeta en los siguientes tres meses para generar una lista que contenga la prioridad con que un ejecutivo de cuenta debe visitar a un cliente, con el fin de disminuir las pérdidas de tiempo derivadas de visitas a clientes con baja propensión al crédito (BBVA, 2021).

0.4. Justificación

Los ejecutivos de cuenta tienen asignada una cartera de clientes que tienen que visitar o contactar para promover los diferentes productos de crédito ofrecidos en el segmento PyME. Sin embargo, la gran cantidad de clientes que tienen asignados genera gran cantidad de tiempo perdido contactando y visitando clientes que no necesitan créditos en este momento. Por lo que el modelo propuesto sirve de guía al ejecutivo de cuenta para solo contactar a los clientes que necesitan un crédito y de esta forma priorizar las visitas a realizar y disminuir los tiempos invertidos en contactar clientes. De esta forma, se promoverá el

aumento en la productividad de los ejecutivos de cuenta debido a que contarán con más tiempo para buscar clientes nuevos.

Por otra parte, el modelo puede ser usado para optimizar la asignación de tasa a los clientes que necesitan un crédito debido a que se les podrá ofrecer una mejor tasa a los clientes que tienen necesidad de crédito mejorando el indicador de ventas debido a que el cliente puede usar como factor de decisión una mejor tasa ofrecida oportunamente. Debido a que un cliente que no necesita el crédito no le importará una mejor tasa y, por otra parte, un cliente con necesidad de crédito que no está totalmente convencido de pedirlo en BBVA, podrá convencerse de solicitarlo en BBVA debido a una promoción en la tasa del crédito.

0.5. Objetivo General

Discutir la aplicación y resultados de un modelo IA puesto en producción por parte del banco BBVA México (en adelante, en ocasiones, se usará el término “el Banco” para referirse a BBVA México) orientado a créditos PyME para generar una lista que contenga la prioridad con que un ejecutivo de cuenta debe visitar a un cliente, con el fin de disminuir las pérdidas de tiempo derivadas de visitas a clientes con baja propensión al crédito. Usando datos proporcionados por BBVA que son públicos, así como estrategias comerciales de BBVA que ha presentado en diferentes foros.

0.6. Objetivos específicos

1. Mostrar la aplicación de los aprendizajes obtenidos en la Maestría de Ingeniería Económica y Financiera de la Universidad La Salle al resolver un problema de impacto social.
2. Mostrar los detalles de la generación de un modelo de IA para el pronóstico de disposiciones de clientes PyME.
3. Mostrar los resultados del modelo y su implementación sin comprometer la información resguardada por BBVA; así como resguardar la imagen de la empresa.
4. Mostrar la eficiencia de la metodología AGILE para la rápida puesta en producción de modelos financieros - económicos.
5. Mostrar la importancia de la IA para la solución de problemas que se presentan en las corporaciones financieras.
6. Mostrar la aplicación de modelos bajo regulaciones Nacionales e Internacionales del cuidado de la información personal de los clientes.

Capítulo 1

Marco Teórico

Desde el comienzo del pensamiento humano, se ha generado interés por automatizar o hacer más fáciles las tareas de los humanos. En el comienzo del siglo XX, las máquinas que calculaban miles de resultados de operaciones presentaron una solución para necesidades matemáticas para la segunda guerra mundial. Por ejemplo, la necesidad de calcular la cantidad necesaria de municiones para poder inhabilitar cierta infraestructura del enemigo, entre otros ejemplos de optimización (Taha, 2011).

Con estos precedentes, y al ver que muchas de las tareas humanas de cálculos eran remplazadas por los llamados ordenadores, tanto en ciencia ficción como en el avance del descubrimiento de las células neuronas y su funcionamiento; el humano ha tratado de generar modelos matemáticos que, traducidos a algoritmos, puedan simular el aprendizaje humano.

Sin embargo, no es casualidad que investigadores en la materia se han preguntado si estos modelos son los óptimos en términos de tiempo de procesamiento y capacidad de los componentes electrónicos de las computadoras. El humano, al hacerse estas preguntas, ha generado algoritmos más eficaces que las llamadas redes neuronales artificiales. Derivado de estos avances, en los últimos años se ha presentado un nuevo campo de conocimiento llamado Machine Learning en lugar de la IA. No solo se pueden dividir los conceptos de aprendizaje e inteligencia y posteriormente unirlos si es necesario; también podemos encontrar algoritmos que aprenden con reglas diferentes a las de las redes neuronales (Brío, 2007).

En adelante, se usarán modelos que aprenden de patrones en los datos y posteriormente, podremos discernir si estos modelos pueden razonar y en qué medida les podemos llamar inteligentes.

El problema matemático que buscan resolver los algoritmos de Machine Learning, en términos generales es el siguiente: Minimizar el error de un modelo que pretende pronosticar un resultado desconocido por medio de un conjunto de información conocida y un algoritmo.

Si bien, muchos de los conceptos usados en la definición pueden ser ambiguos, se supondrá que el lector ya tiene conocimiento sobre términos como error,

modelo, información o datos. Aunque, para eliminar ambigüedades, se esbozará la definición matemática de cada modelo.

1. Árboles de decisión
2. Random forest
3. Ada boost
4. Redes neuronales
5. Regresión logística
6. Máquinas de soporte vectorial

Actualmente existen diferentes tipos de modelos que son usados dependiendo de su efectividad en cada fenómeno a predecir. Sin embargo, la mayoría tienen como objetivo pronosticar resultados con información recopilada de experiencia previa del fenómeno. En este caso, se les llama modelos de aprendizaje supervisado debido a que son entrenados con información previa. Ejemplos de fenómenos a predecir dada información previa, pueden ser:

1. Desempeño de un empleado en el año [bueno, malo]
2. Pronóstico del precio de una acción [baja, sube]
3. Contratación de un préstamo [va a contratar, no va a contratar]

Cabe señalar que se pueden tener más clases pero, por simplicidad, hablaremos solo de dos, y las denotaremos como cero o uno a cada clase. Para poder realizar dicha clasificación, es necesario un conjunto de vectores de datos que serán usados para poder obtener un cero o uno. También, este conjunto debe dividirse en dos subconjuntos; uno de entrenamiento y otro en un conjunto de validación. De tal forma, que necesitamos de una función que nos clasifique al conjunto de datos de dimensión n (n columnas) de la siguiente forma:

$$f := \mathbb{R}^n \rightarrow \{0, 1\} \quad (1.1)$$

Buscando siempre que el error sea el mínimo. De esta forma, el mejor modelo será aquel que nos arroje el menor error posible. Diferentes y múltiples algoritmos se han creado para esta finalidad.

1.1. Árboles de Decisión

El primer clasificador que se verá es el Árbol de Decisión, que usa el concepto de ganancia de información para poder clasificar los datos de la mejor manera. Lamentablemente, solo puede clasificar la información en dos clases lo que supone dudas en algunos aspectos sobre la no linealidad del resultado. Sin

embargo, es ampliamente usado y genera una base conceptual para más algoritmos como veremos a continuación. El árbol de decisión toma una columna de la tabla de datos de entrenamiento (una variable); por ejemplo, supongamos que es seleccionada la variable de volumen en una temporada de conciertos de música clásica y se busca clasificar aquellos conciertos que han dejado mayor satisfacción en los asistentes. Esta variable toma los valores Alto o Bajo y se busca determinar la satisfacción de los asistentes en una variable y que toma los siguientes valores: satisfecho o no satisfecho.

El objetivo del algoritmo llamado C4.5 (Rim y Liu, 2020), es buscar la Ganancia de la Información en los conciertos con volumen Alto respecto a los conciertos con volumen Bajo. En caso de que exista menor incertidumbre (mayor ganancia de la información) entre una u otra clase de la variable volumen, entonces divide el conjunto de datos en dos (genera una rama nueva del árbol) y aplica este proceso para cada variable una y otra vez dependiendo de la profundidad (número de divisiones) o exactitud (nivel de error) requeridos por el usuario del algoritmo. En específico, la famosa fórmula de Ganancia de la Información o de medida de incertidumbre (Entropía) de Shannon I_E es:

$$I_E(f) = - \sum_{i=1}^n f(x_i) \log_2 f(x_i) \quad (1.2)$$

En donde f es la función de distribución de probabilidad, $f(x_i)$ es la probabilidad del evento x_i , m es el total de eventos del espacio de eventos.

Para ver cómo funciona esta medida, se puede evaluar el fenómeno de lanzar una moneda al aire con las probabilidades de que salga cara o cruz; se podrá ver que es el experimento con mayor incertidumbre ya que las probabilidades de que se consiga una u otra cara son iguales y son los únicos resultados del experimento. En cambio, al tirar dos dados, el resultado con menos incertidumbre (la función alcanza un mínimo) es el número siete.

Una alternativa para la medida de incertidumbre de Shannon, es el índice de Gini que es utilizado para evaluar el desempeño de un modelo como se verá más adelante y se define como:

$$Gini = \sum_{i=1}^n p_i(1 - p_i) \quad (1.3)$$

en donde p_i es la probabilidad del evento i y n es el total de eventos posibles de un experimento.

Un árbol de decisión se compone de hojas, nodos y ramas. Las hojas son la parte final de árbol y representa un segmento (patrón) encontrado en donde, si se divide más, el algoritmo sobrepasa las divisiones que se pre requirieron o ya no se encuentra ganancia de la información por alguna división. Los nodos, son las divisiones que encuentra el algoritmo de acuerdo a la ganancia de información calculada al encontrar un patrón. Las ramas unen los nodos y las hojas para poder mostrar gráficamente las relaciones entre las variables.

Posteriormente, con este algoritmo, podemos construir otros dos que mejoran la precisión. El Random Forest y el Ada Boost (Freund et al., 1999). El Random

Forest, en lugar de medir recursivamente la “Ganancia de Información” variable por variable cuantas veces se le asigne al algoritmo en profundidades genera cientos o miles de combinaciones de pequeños árboles de decisión, por lo general, de una división de profundidad y posteriormente los une poniendo en primer término la unión que genere mayor “ganancia de información”.

Por último, el Ada Boost, se basa en el supuesto de que si se unen pequeños árboles de decisión (de una división de profundidad) como en un Random Forest, se puede caer en el abandono de información valiosa olvidando patrones que generan ganancia de la información pero que se han descartado por ser poco valiosos respecto de otro patrón. De tal forma, que este algoritmo genera múltiples árboles de decisión para generar uno en el que varios caminos pueden llegar al mismo resultado. Por ejemplo:

Supongamos que un volumen alto siempre genera satisfacción en el público, entonces, un árbol de decisión, o un Random Forest, dejarían este resultado como la única hoja ya que la ganancia de información es siempre relevante. Pero, ¿qué sucede si, la calidad de los instrumentos es una variable también relevante, aunque no tanto? La respuesta es que el análisis estaría dejando de lado información importante. Es por ello, que el algoritmo de Ada Boost, generará dos árboles (hasta donde se sabe solo se han presentado estas dos variables), y por ello, nos dirá que, si los instrumentos son de calidad o el volumen es alto, entonces se prevé un concierto con gran satisfacción.

Cabe señalar, que un aspecto importante del Ada Boost es que utiliza la optimización del “Gradiente Descendiente” para poder dividir las ramas buscando un óptimo en ganancia de la información cuando la variable es continua y no discreta (clases).

En la Figura 1.1, se puede apreciar el algoritmo de un árbol de decisión como el que hemos comentado.

Existen diferentes versiones del algoritmo de Random Forest, a continuación se definirá un algoritmo de Random Forest con base en (Rim y Liu, 2020), del cual se han presentado las ideas generarles:

Definición 1.1.1 (Algoritmo Random Forest) *Con base en un conjunto de datos de entrenamiento y un parámetro k , se busca un resultado de clasificación con los siguientes pasos:*

1. *Dividir el conjunto de entrenamiento en k subconjuntos aleatorios.*
2. *Hacer un árbol de decisión condicional para con cada subconjunto k con el algoritmo C4.5.*
3. *Evaluar una entrada de información en cada uno de los k árboles de decisión creados.*
4. *El resultado devuelto del modelo será la moda del conjunto de los k resultados (esto es, el resultado más votado por todos los k árboles generados).*

Una de las ventajas del modelo de Random Forest es que evita el sobreajuste debido a que intervienen los resultados de varios modelos que se han

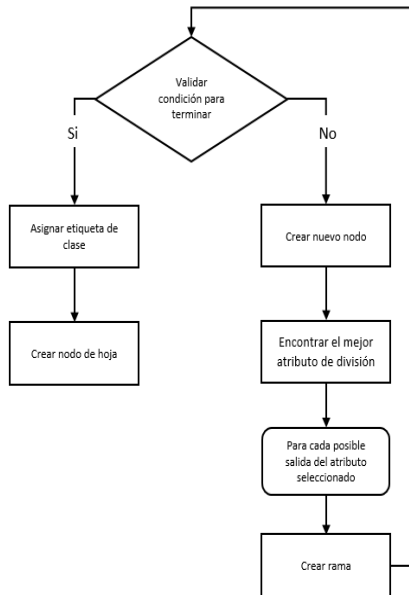


Figura 1.1: Diagrama del flujo de un algoritmo para generar un árbol de decisión C4.5. Fuente: (Brío, 2007).

entrenado con subconjuntos de datos diferentes. Una de las desventajas es que su entrenamiento requiere de más recursos computacionales y tiempo debido a que estima varios árboles de decisión. El modelo de Random Forest es usualmente el que genera mayor precisión debido a que considera las respuestas de un conjunto de modelos. Cabe señalar que al tener más datos, se podrá subdividir la muestra en más subconjuntos, obteniendo más modelos que adicionarán valor al resultado final.

El Algoritmo B.1.1 presentado en 1997 pretende crear un conjunto de entrenamiento en cada iteración (método conocido como algoritmo ensamblador). Este nuevo conjunto de entrenamiento tomará en cuenta, en mayor medida, aquellos registros en los que el mejor modelo no ha podido pronosticar adecuadamente. De esta forma, los árboles se encuentran conectados debido a que un árbol aprende (con una variable diferente usada por el árbol predecesor) de los registros que no fueron pronosticados adecuadamente por un árbol predecesor.

Una de las desventajas del modelo Ada Boost es que utiliza grandes cantidades de recursos computacionales ya que recalcula los una cantidad M de árboles en cada iteración. En el algoritmo (B.1.1) la función I se refiere a la función Caso que arroja el valor uno en caso de cumplir con cierta condición y cero en caso contrario.

1.2. Redes Neuronales

En el caso de las redes neuronales, sus algoritmos son fáciles de comprender¹ y ampliamente usados hoy en día. Existe una amplia investigación al respecto sobre sus formas y cómo ordenarlas para obtener los mejores resultados.

Aunque no se han obtenido resultados contundentes en mejorar los resultados y en cada aplicación es necesario realizar experimentos de prueba y error para comprobar hipótesis sobre su arquitectura (topología) como se precisa en (Brío, 2007). Aunque son fáciles de programar, consumen muchos recursos computacionales, por lo que la mayoría de las aplicaciones solo utiliza topologías de pocas capas o un número reducido de neuronas.

Sin embargo, en los últimos años, el desarrollo del cómputo distribuido y nuevas estrategias para su programación ha generado resultados insospechados hace unas décadas en temas como reconocimiento de imágenes o autonomía de vehículos; generando un resurgimiento de su uso con más potencia que hace unos años. Las mejoras y los nuevos algoritmos se pueden ver en el proyecto Tensor Flow de Google, que pretende ser referente para la programación y uso de estas redes.

El concepto básico de red neuronal se basa en el modelo de neurona de Ramón Cajal (Klein, 2017) que se puede apreciar en la Figura 1.2.

En la Figura 1.2, podemos apreciar diferentes arreglos y elementos que permiten a la neurona guardar información y compartirla con otras neuronas. En el centro observamos el núcleo (B) que recibe de las dendritas (A) información de otras neuronas y que después la transmite a otras neuronas a través del Axón (C) con diferentes grosores (sinapsis) para enviar señales moduladas a otras neuronas. El flujo de información en la Figura 1.2 sería de abajo hacia arriba; en “A” se recibe información de otras neuronas, B activa la salida que es modulada por C que lleva la nueva información (señal) a otras neuronas. El modelo matemático más aceptado (Modelo Estándar) para representar una neurona es el siguiente:

Definición 1.2.1 (Modelo Matemático de Neurona) Sea $\vec{x} = \langle x_1, x_2, \dots, x_n \rangle$ un vector con información, el procesamiento de esta información sucederá de la siguiente forma:

$$y(\vec{x}) = \varphi \left(\sum_i^n w_i x_i - \theta \right) \quad (1.4)$$

La fórmula es muy fácil de programar y entender. Primero, sumamos cada entrada de información (dato del vector) multiplicado por w_i el peso sináptico (brazos de la red con diferentes grosores) y posteriormente le se le resta a Theta

¹Se consideran fáciles de comprender debido a que se pueden interpretar como la unión de varias regresiones lineales que son ampliamente conocidas en el análisis económico y estadístico.

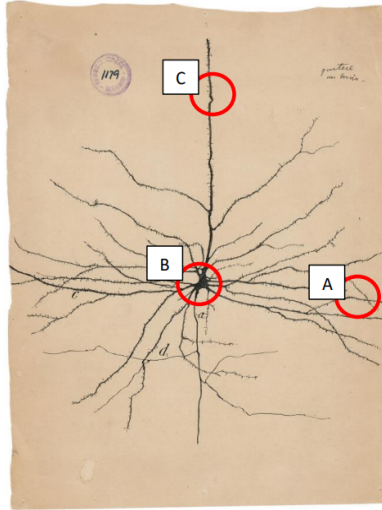


Figura 1.2: Una de las primeras ilustraciones de una neurona en 1889 con sus componentes señalados. Fuente: Elaboración propia con imagen de fondo provista en Klein, 2017.

que usualmente es una constante². Al sumar estos valores, se le aplica una función φ (función de activación) que apaga o enciende una señal³, usualmente funciones que entreguen cero o uno como resultado. Cuando se tiene una red neuronal, usualmente se envía este resultado a otra neurona para que lo procese con otros resultados.

Cuando se tiene una sola neurona, el modelo es el de una regresión lineal y su interpretación es ampliamente conocida. Sin embargo, al unir múltiples neuronas, no se puede concluir lo mismo y no se puede interpretar la sinapsis del modelo, pero si se puede utilizar el conocimiento que se tenga sobre una regresión para explicar cómo aprenden las redes neuronales.

Un aspecto importante sobre las Redes Neuronales es que el aprendizaje depende en gran medida de las funciones de activación con que se cuente, entre las más comunes se tienen:

Definición 1.2.2 (Algunas de las funciones de activación más comunes)

Sea $x \in \mathbb{R}$ y sea $\vec{z} = \langle z_1, z_2, \dots, z_n \rangle$ un vector de tamaño n y con $z_i \in \mathbb{R}$, a las siguientes funciones se les denominará como *Funciones de Activación*:

²Theta es usualmente una constante arbitraria y en ocasiones es conocida como “bias” (véase la Figura 1.3 círculos en color azul). En una regresión lineal, la ecuación de la recta en una regresión es de la forma $y = mx + b$, en este caso, b sería el bias.

³Existen diferentes funciones de activación y cada una puede mejorar el modelo dependiendo de cómo se configure la red neuronal. En el caso de las regresiones, al aplicar las funciones del tipo sigmoidea, se obtienen los llamados modelos probit o logit que clasifican la información de salida en cero o uno (SAS, 2011).

Identidad:

$$\varphi(x) = x \quad (1.5)$$

Escalón:

$$\varphi(x) = \begin{cases} 0 & \text{si } x < 0 \\ 1 & \text{si } x \geq 0 \end{cases} \quad (1.6)$$

Sigmoidal:

$$\varphi(x) = \frac{e^x}{1 + e^{-x}} \quad (1.7)$$

ReLU (Rectified Lineal Unit):

$$\varphi(x) = \max(0, x) \quad (1.8)$$

Softmax:

$$\varphi(z_j) = \frac{e^{z_j}}{\sum_{i=1}^n e^{z_i}} \quad \text{para } j = 1, \dots, n \quad (1.9)$$

La elección de cada una de estas funciones se puede hacer arbitrariamente en un inicio, pero por medio de experimentos sucesivos se podrá elegir a la mejor función de activación. Es importante señalar, que la función 1.8 ha sido de gran importancia en los últimos avances en la materia, probando mejoras sustanciales en los modelos. Esto, se podría explicar debido a que algoritmos de optimización tienen menos problemas al hallar un valor óptimo en esta función, lo que se entenderá si se examinan los principales algoritmos para encontrar máximos y mínimos como se mostrará más adelante.

Existen diferentes tipos de funciones de costo y la elección de una de ellas para el aprendizaje será determinante para el desempeño final de un modelo.

Definición 1.2.3 (Algunas funciones de costo más comunes) *Las funciones de costo más comunes para una muestra de tamaño N con n parámetros son:*

1. *Error Simple (ES):*

$$L(\hat{y}, y) = \sum_{i=1}^N |y_i - \hat{y}_i| \quad (1.10)$$

2. *Error Absoluto Medio (EAM):*

$$L(\hat{y}, y) = \sum_{i=1}^N \frac{|y_i - \hat{y}_i|}{N} \quad (1.11)$$

3. *Error Cuadrático Medio (ECM):*

$$L(\hat{y}, y) = \sqrt{\sum_{i=1}^N \frac{(y_i - \hat{y}_i)^2}{N}} \quad (1.12)$$

4. Cross Entropy (CE):

$$L(\hat{y}, y) = \sum_{i=1}^N -(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)) \quad (1.13)$$

En donde $\hat{y} = \langle \hat{y}_1, \dots, \hat{y}_N \rangle$ es el vector de los valores pronosticados e $y = \langle y_1, \dots, y_N \rangle$ los valores reales.

En el caso del modelo Ada Boost, la función de costo será la función de incertidumbre de Shannon. Es importante señalar, que la función de costo debe contar con derivadas de primer y segundo orden y ser cóncava para poder aportar una solución no divergente.

Es importante comprender el funcionamiento de los algoritmos de optimización debido a que la elección de uno u otro proporcionaría diferentes resultados. El modelo de Gradiente Descendente (GD) es ampliamente usado en Machine Learning, tanto para Redes Neuronales, como para el algoritmo de Ada Boost. El algoritmo GD se explica en el anexo de Definiciones Matemáticas.

Existen diferentes variantes de este algoritmo, la que se utilizará para el aprendizaje de la Red Neuronal es la variante ADAM (Adaptive Moment Estimation), que consiste en el algoritmo de gradiente descendiente, pero en el que la tasa de aprendizaje η se adapta (cambia) en función de los primeros momentos del gradiente. El algoritmo ADAM se explica en el anexo de Definiciones Matemáticas.

Los avances en optimización, en específico, la función de costo CE, la función de activación ReLU y el algoritmo ADAM han generado un avance en la Inteligencia Artificial en los últimos años.

La Regresión Logit, ayudará a mostrar cómo las diferentes definiciones presentadas se pueden unir para encontrar los mejores parámetros de un modelo. El modelo básico de regresión, se podrá definir de la siguiente forma:

$$\hat{y}(\langle x_1, \dots, x_n \rangle) = \sum_{i=1}^n w_i x_i + \theta \quad (1.14)$$

Posteriormente, se podrá aplicar alguna de las funciones de costo, usualmente la de ECM y utilizar un algoritmo de optimización para encontrar los mejores parámetros. Este tipo de análisis se suele llamar modelo obtenido por mínimos cuadrados cuando se utiliza ECM. Es importante resaltar, como se ha mencionado, que los resultados del modelo cambiarán de acuerdo al algoritmo de optimización, la función de costo y la función de activación seleccionados. En el caso de que se le aplique al resultado de la regresión la función φ sigmoideal (lo cual hará que el resultado se encuentre entre cero y uno), entonces se dirá que se cuenta con modelo de regresión Logit, pero por ser un modelo de clasificación y no de regresión, entonces se tendrá que usar la función de costo CE.

En una regresión se busca minimizar el error del modelo obteniendo los w_i

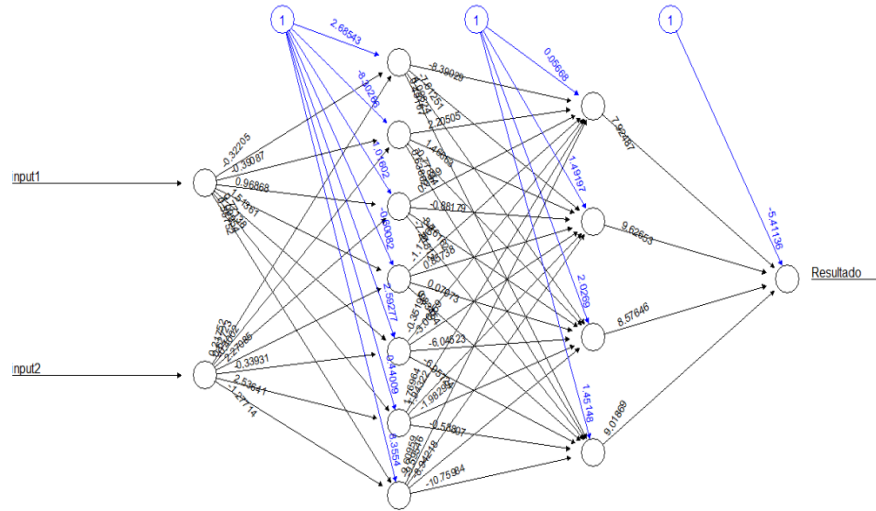


Figura 1.3: Ejemplo de una red neuronal de 2 capas ocultas. Fuente: Elaboración propia con datos simulados (ver anexo).

óptimos con un algoritmo de optimización, usualmente Mínimos Cuadrados⁴. En las redes neuronales sucede lo mismo (Brío, 2007), pero cada neurona va fragmentando el error total del modelo en las múltiples neuronas paso a paso por un orden de atrás hacia adelante. Es por ello, que el nombre del algoritmo más usual para el aprendizaje de las redes neuronales es llamado Back Propagation.

En la Figura 1.3 se puede observar el modelo descrito en la Definición B.3.1. En este caso, dos entradas de información (x_1, x_2) pasan a una capa de siete neuronas que procesa la información por medio de una función g_1 y pasa la información a la capa dos con cuatro neuronas. Por último, una capa de una neurona recibe la información de la capa dos para proporcionar el resultado final.

Este tipo de parámetros de aprendizaje (número de capas, tasa de aprendizaje, funciones de activación para cada capa o función de costo) son llamados hiperparámetros y, por medio de varios experimentos se proponen valores que generen mejores resultados de la red neuronal.

Las máquinas de soporte vectorial (SVM por sus siglas en idioma inglés) se basan en encontrar fronteras (planos) multidimensionales que dividen a un conjunto de puntos en el espacio. Se busca un plano de tal forma que los puntos que se encuentran de un lado (C_1) u otro (C_2) de él tengan características diferentes, esto es, que la distancia entre uno u otro conjunto de datos sea la más

⁴Existe otro algoritmo para encontrar los parámetros de una regresión (máxima verosimilitud) pero no será mencionado en este párrafo debido a que lo que se quiere mostrar es la relación del algoritmo de mínimos cuadrados de una regresión con el algoritmo de aprendizaje de una red neuronal.

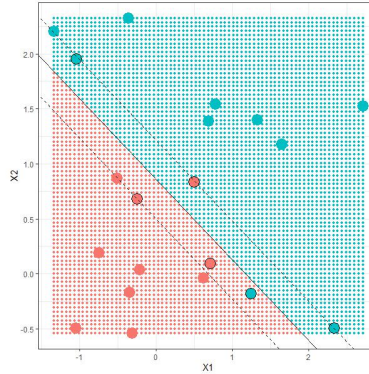


Figura 1.4: Clasificación lineal con una Máquina de Soporte Vectorial para dos clases (rojo y verde). Fuente: Elaboración Propia en lenguaje R con datos simulados.

grande posible. Será conveniente tomar a cada registro de la muestra como un punto en el espacio multidimensional, en donde cada característica (columna de la tabla muestra) será una dimensión. En el anexo de Definiciones Matemáticas se explica a detalle su funcionamiento.

En la Figura 1.4 se pueden observar planos que buscan dividir dos conjuntos de datos (rojos y verdes) y en la Figura 1.5 se puede apreciar la aplicación del Kernel Gaussiano para dividir las dos clases de forma no lineal.

1.3. Métricas de efectividad de los modelos

Para medir la efectividad de los modelos, será preciso entrenarlos con un conjunto elegido aleatoriamente (llamado train set) y probarlo con un conjunto de prueba (test set). De acuerdo con (Hamstra y Zaharia, 2013) y (Brío, 2007) se usará un muestreo aleatorio simple y una proporción de 0,7 vs 0,3 respectivamente que es lo que usualmente se considera apropiado.

Como serán evaluados diferentes modelos, será necesario medir la efectividad de ellos por medio de métricas. En concreto, utilizaremos la matriz de confusión y la curva ROC (SAS, 2011) que es una representación gráfica de la matriz de confusión. Cabe señalar que la matriz de confusión aporta mayor valor si se discuten y analizan sus resultados.

La matriz de confusión se compone de cuatro elementos que, al analizarlos podremos definir qué tan bueno es un modelo para los propósitos que necesitamos. Estos cuatro elementos se pueden apreciar en la Tabla 1.1.

Dependiendo del fenómeno a modelar, serán más valiosos los verdaderos negativos o los verdaderos positivos. Por ejemplo, si se trata de pronosticar la subida de una acción, nos interesa ser adversos al riesgo, centraremos la atención en los falsos positivos; En el caso de un modelo que pronostique una enfermedad

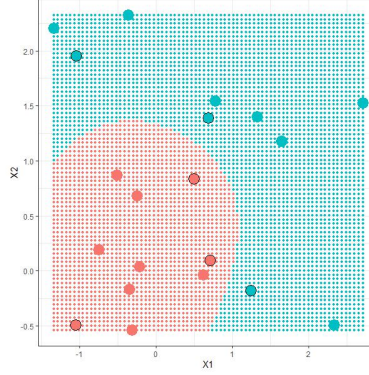


Figura 1.5: Clasificación no lineal con una Máquina de Soporte Vectorial para dos clases (rojo y verde) con un kernel Gaussiano. Fuente: Elaboración Propia en lenguaje R con datos simulados.

	Predicho Clase 1	Predicho Clase 2
Real Clase 1	Verdaderos Positivos (vp)	Falsos Negativos (fn)
Real Clase 2	Falsos Positivos (fp)	Verdaderos Negativos (vn)

Tabla 1.1: Relación de resultados entre lo predicho y el conocido real.

mortal que se pueda prevenir, nos interesarán los verdaderos positivos y los falsos negativos.

En específico, se utilizarán las siguientes métricas:

Definición 1.3.1 *Métricas de efectividad de modelos de clasificación*

1. *Precision (Precisión)*

$$P = \frac{vp}{vp + fp} \quad (1.15)$$

2. *Recall (Sensibilidad, también llamado True Positive Rate o TPR)*

$$TPR = \frac{vp}{vp + fn} \quad (1.16)$$

3. *Accuracy (Exactitud o también llamado Efectividad)*

$$A = \frac{vp + vn}{vp + fn + fp + vn} \quad (1.17)$$

4. *Specificity (Especificidad)*

$$S = \frac{vn}{vn + fp} \quad (1.18)$$

5. $f1$

$$f1 = \frac{2vp}{2vp + fp + fn} \quad (1.19)$$

6. False Positive Rate (FPR)

$$FPR = \frac{fp}{fp + vn} \quad (1.20)$$

La Precisión es adecuada cuando se requiere hacer énfasis en los verdaderos positivos no importando que los verdaderos negativos sean elevados. Un ejemplo para usar este tipo de métricas sobre de otras es el pronóstico de si un caballo ganará o no una carrera. En este caso es importante saber si el caballo elegido por el modelo ganará o no y no nos preocupa que el caballo gane si no fue pronosticado que ganará por el modelo. En el caso del uso del Recall (en el mismo ejemplo de la apuesta a los caballos), pude que además de precisión, sea requerido volumen de carreras acertadas correctamente, es por ello que se adhiere fn a la ecuación, de esta forma se podría apostar también por los caballos que pierden. En el caso de la métrica de Exactitud, se tiene una idea global del resultado del modelo, pero esto no necesariamente es adecuado a tomar en cuenta debido a que depende de la aplicación del modelo si es necesario más precisión o sensibilidad.

Cuando se toma en cuenta la métrica de Especificidad, se requiere que el modelo no pronostique una gran cantidad de fp . Por ejemplo, en caso de que se pronostique que un fraude ocurrirá o no, será necesario que los fp sean los menos posibles para no perder recursos investigando casos que no serán fraude.

En el caso de la métrica $f1$, se busca combinar la Precisión con la Exactitud dando más peso a los verdaderos positivos multiplicando por dos y esto es adecuado para aplicaciones en las que la Precisión es necesaria pero sin dejar de lado la Exactitud.

Para modelos que no son de clasificación existen diferentes métricas. Se considera apropiado mostrar las más importantes. La métrica R^2 es ampliamente usada y se basa en el hecho de que una menor variabilidad de los datos pronosticados hará un mejor modelo:

$$R^2 = \frac{\sum_{i=1}^n (\hat{y}_i - \mu)^2}{\sum_{i=1}^n (y_i - \mu)^2} \quad (1.21)$$

en donde μ es el promedio de las y_i y su versión modificada (promedio por elemento de la muestra):

$$\hat{R}^2 = 1 - \frac{n-1}{n-M-1} (1 - R^2) \quad (1.22)$$

en donde M es el número de variables explicativas. Estas métricas son usadas en modelos de regresión lineal, pero en modelos más avanzados, se ha usado en

mayor medida la propia función de costo que en el caso de modelos de regresión puede ser el ECM.

Otra métrica para modelos de clasificación ampliamente usada es el índice de Gini que se ha mostrado en la fórmula 1.3. En el algoritmo de aprendizaje de Ada Boost se ha usado este índice para encontrar el mejor modelo dentro de un conjunto de modelos. En el caso del índice de Gini, el modelo con el menor índice será el mejor debido a que representa una medida de incertidumbre. Sin embargo, se recomienda contrastar diferentes métricas para elegir el mejor modelo.

Por otra parte, el área bajo la curva de la ROC (Receiver Operating Characteristic) nos ayudará a vislumbrar la efectividad del modelo. Cuando esta medición converge a uno, se dirá que el modelo predice todos los eventos propuestos.

La curva ROC representa de forma gráfica el Accuracy del modelo, esto es, el área bajo la curva es igual al Accuracy. Para graficar la curva ROC, se tienen que unir los puntos resultantes de Recall en el eje vertical y FPR en el eje horizontal a medida que el resultado del modelo va dejando valores de entre cero y uno. Por ejemplo, cuando el modelo predice que el cliente tomará un crédito con un puntaje de 0.8, entonces se calcula la Recall y la FPR para los clientes que el modelo calculó con ese puntaje.

Definición 1.3.2 (Curva ROC) *La curva ROC para una muestra de tamaño n es calculada con la siguiente función de $\mathbb{R} \rightarrow \mathbb{R}^2$:*

$$ROC(l) = \left(1 - \frac{\sum_{i=1}^n I(\{y \neq \hat{y}, \hat{y} < l\})}{\sum_{i=1}^n I(y \neq \hat{y})}, 1 - \frac{\sum_{i=1}^n I(\{y = \hat{y}, \hat{y} < l\})}{\sum_{i=1}^n I(y = \hat{y})} \right) \quad (1.23)$$

en donde l es un número en $[0, 1]$ que es en donde se encontrará la respuesta de la función φ . La función caso $I(\{a, b\})$ será uno si se cumplen las hipótesis a y b y cero en otro caso.

1.4. Metodologías para la aplicación de modelos

Para aplicar un modelo, será necesario seguir una metodología. Para ello, fue seleccionada la metodología SEMMA de SAS (SAS, 2011). La cual pretende ser una guía para generar modelos debido a que es la metodología que toma los principales pasos de las otras dos metodologías de las que se hablará más adelante. Los pasos de la metodología SEMMA son:

1. Sampling: Se refiere a la selección de la muestra de train set y test set y propone una selección de las variables para ingesta en el modelo.
2. Explore: Explorar los datos de forma descriptiva para poder generar hipótesis que nos ayuden a quitar datos redundantes y se pueda aplicar el siguiente paso.

3. Modify: Se refiere a los cambios que deben aplicarse a las variables para que sean más entendibles para el algoritmo; por ejemplo, utilizar variables sintéticas.
4. Model: Aplicar diferentes modelos a los datos y obtener sus resultados.
5. Asses: Este paso consiste en evaluar varios modelos con los indicadores de eficiencia como los mostrados en este texto (matriz de confusión).

Si bien, existen muchas otras metodologías, la mayoría coincide en los pasos y solo difieren unas de otras en el orden para aplicarlos como se precisa en la metodología Box y Jenkins (Apodaca, 2014) y (Sarkar, 2012). La metodología Box y Jenkins fue elaborada en el año 1960 y sigue los siguientes pasos:

1. Identificación: Seleccionar un modelo apropiado con base en las características estadísticas de las variables.
2. Estimación: Encontrar los parámetros que permitan al modelo representar con mayor detalle el fenómeno.
3. Verificación: Que el modelo cumpla con su objetivo y que represente de forma deseada el modelo.
4. Uso del modelo: Implementar el modelo, obtener resultados y analizarlos.

Otra metodología usada para la aplicación de un modelo es la llamada CRISP-DM (Cross Industry Standard Process for Data Mining) la cual pretende ser un estándar para aplicar modelos en áreas de Minería de Datos (Espinosa-Zúñiga, 2020). Esta metodología Europea es creada en el año 1997, por el Programa de Investigación y Desarrollo en Tecnologías de Información de la Unión Europea (ESPRIT) y consiste de los siguientes pasos:

1. Comprensión del problema: Identificación del problema, determinación de objetivos y evaluación de la situación actual (estado actual antes de implementar la solución de minería de datos propuesta).
2. Comprensión de los datos: Recolección de datos, descripción de datos y exploración de datos.
3. Preparación de datos: Limpieza de datos, creación de variables sintéticas y transformación de datos.
4. Modelado: Selección de técnica de modelado, selección de datos de prueba y ajuste del modelo.
5. Evaluación del modelo: Determinar la calidad del modelo con base en el análisis de ciertas métricas estadísticas, comparando los resultados con resultados previos.
6. Implementación del modelo: Documentación de los resultados para el usuario final. Evaluar las acciones para detectar áreas de oportunidad.

En el caso de la metodología CRISP-DM se propone una metodología cíclica en la que la evaluación de los resultados de la implementación permite generar nuevas ideas para poder generar nuevos modelos o repensar los ya existentes. Cabe señalar, que en la metodología CRISP-DM, se propone comprender el problema para delimitar la aplicación y responder mejor a las necesidades del negocio. Este paso no es mencionado en la metodología Box y Jenkins debido a que es más enfocada al concebimiento de un modelo que surge de una necesidad ya existente. Por otra parte, la documentación del modelo es parte de un proceso que ayudará al usuario a comprender cómo funciona la herramienta y supone que el modelo debe ser parte de un ecosistema en el que el generador del modelo esté en contacto con el usuario para mejorar la aplicación del modelo.

1.5. Herramientas de software utilizadas

Cuando las computadoras empezaron a generar y guardar datos de forma más eficiente que antes, las empresas que generaban grandes cantidades de información se encontraron con el problema de aplicar algoritmos o consultas a sus datos debido a la limitada capacidad de procesamiento de las computadoras de escritorio o servidores pequeños (Jules, 2019).

Una de las soluciones encontradas y que se ha generalizado su uso, es el cómputo distribuido que pretende relevar múltiples tareas a un conjunto de procesadores en lugar de usar un solo procesador (Jules, 2019) y (Caballero, 2018).

Si bien, esta tecnología existe desde hace decenas de años; las grandes corporaciones han puesto a disposición del usuario los grandes servidores con que cuentan para mejorar la eficiencia de los algoritmos. Esto es, es posible pensar, que no es una solución al problema de eficiencia en el almacenamiento o procesamiento de un CPU⁵, en cambio, multiplicar el número de unidades de CPU que realizan una tarea (Jules, 2019).

En el medio computacional (Google, IBM o Microsoft) sigue sin respuesta consensuada la pregunta de qué es Big Data, como se precisa en (Davenport, 2014).

Sin embargo, existe un consenso en que debe de tratarse de un conjunto de técnicas enfocadas a solucionar problemas que no se pueden resolver en un servidor o equipo personal común por temas de capacidad de memoria o tiempo (Jules, 2019) y (Caballero, 2018). Además, se pueden añadir otras especificaciones como variedad de datos que son formatos o estructuras difíciles de registrar debidamente en bases de datos convencionales (imágenes, sonido, texto, etc.)⁶.

Ya que se cuenta con la capacidad de cómputo suficiente para trabajar con grandes volúmenes de datos, es preciso encontrar software que trabaje de forma distribuida.

El software pionero en popularizar estos temas fue Hadoop creado por la empresa Google. Sin embargo, Spark ha generado mejoras sustanciales y es

⁵CPU son las siglas de: Central Processing Unit

⁶Es conocido en el medio las tres V: velocidad, variedad y volumen (Davenport, 2014).

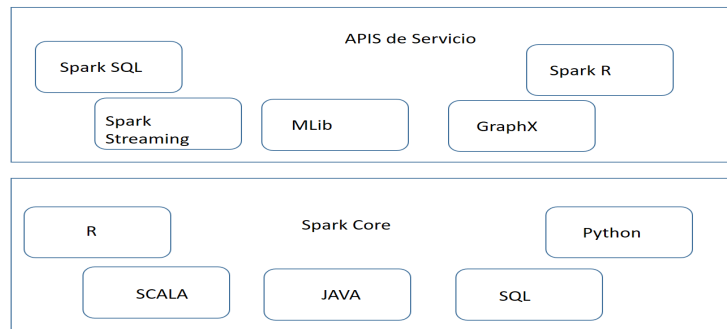


Figura 1.6: Ecosistema de las herramientas que se pueden utilizar en Spark. Fuente: (Eman, 2019).

mucho más fácil de usar (a opinión del autor de este texto) que Hadoop (Jules, 2019). Además, es el software con el que se contaba y es con el que trabajan Datio, DataBricks y Horton.

Entre los algoritmos de Big Data más conocidos, se encuentra Map and Reduce, creado y patentado por Google, el cual mapea una tarea y la reduce en pequeñas tareas que le asigna a cada procesador. Como ejemplo, supóngase que se requiere conocer el saldo total de todas las cuentas de todos los clientes de un banco en el menor tiempo posible. El algoritmo de Map and Reduce tomará conjuntos de n registros de forma ordenada y a cada conjunto le asignará un id nodo (identificador único de cada nodo) dependiendo del nodo al que se le asignará la tarea (Map). Cada conjunto de datos se trasladará a cada nodo (CPU) con que cuente el servidor. Cada nodo, sumará los saldos de los clientes que le fueron asignados y devolverá el resultado. Posteriormente, se sumarán los resultados obtenidos en cada nodo (Reduce).

La principal diferencia entre Spark y Hadoop, es que Spark trabaja in-memori y no escribe la información distribuida en los discos duros de cada servidor del sistema distribuido; esto permite una ganancia sustancial en tiempo.

Por otra parte, es posible programar en los lenguajes de programación R, SCALA y Python (Bill, 2017). Lo cual hace de Spark, la plataforma de Big Data más usada hasta el momento. Para mayor detalle de Spark, se pueden obtener los manuales que se encuentran en su sitio web (spark.apache.org, 2021).

En la Figura 1.6 se puede apreciar que Spark cuenta con herramientas para trabajar los datos en forma distribuida, pero con los lenguajes más comunes. Por ejemplo, Java, R, SQL o Python. Por otra parte, permite, que Bibliotecas de manejo de datos o de Machine Learning sean programadas en estos lenguajes como MLib que permitirá programar en Python modelos de Machine Learning pero haciendo el procesamiento en un ambiente distribuido.

En la Figura 1.7, se presenta una descripción del procesamiento interno de

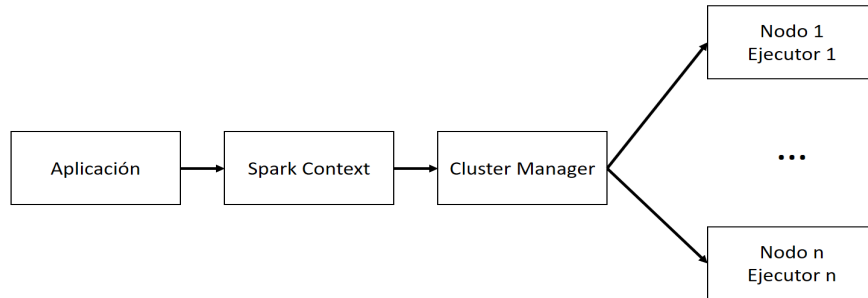


Figura 1.7: Descripción del funcionamiento distribuido de Spark. Fuente: (Eman, 2019).

Spark, se programa en un lenguaje (Python, Scala, R, etc.) una aplicación, después, el módulo de Spark Context, traduce ese programa en algoritmos Map and Reduce y lleva este procesamiento al Cluster Manager quien asigna tareas ordenadas y recibe las respuestas de cada Nodo ordenando la información.

Capítulo 2

Marco Metodológico

A continuación, se seguirán los pasos propuestos por la metodología SEMMA, por lo cual, las subsecciones seguirán el orden de la metodología y tendrán los respectivos nombres de las siglas de SEMMA.

2.1. Sampling

Los datos proporcionados y publicados a los participantes del Hackathon fueron usados por el autor para generar el modelo¹. Los datos proporcionados son descritos en la Tabla 2.1.

Si bien, como se precisa en (BBVA, 2021), se plantean diferentes problemas a resolver; Se eligió el problema de encontrar un clasificador para las PyMES para saber si necesitan o no un crédito revolvente (propensión al crédito revolvente) ya que esta fue la solución que se construyó y quedó en producción en la empresa BBVA.

El total de registros es de 45,718² y difieren de los datos entregados al Hackaton debido a que los desarrolladores del evento cambiaron algunos campos para poder hacer la tabla más entendible para el público en general. El autor de este texto, quien fue responsable de crear la tabla, se da la libertad de mejorarla con la experiencia adquirida en los últimos meses. A saber, los cambios más importantes son, quitar el Código Postal y las ofertas hasta ese momento. En cambio, se agregó la liquidez del cliente.

Sin embargo, debido a que este dato no forma parte de la tabla original, solo se hará mención al campo y no se evaluarán sus descriptivos. Cabe señalar que la variable Y fue incluida por el autor para el reto Hackaton pero fue quitada por los desarrolladores por razones de mala comunicación.

¹Para el modelo final, el autor contaba con más campos que ayudaron al desempeño del modelo. Sin embargo, no se pueden mostrar debido a la protección de secretos industriales.

²La tabla original presentada en el Hackaton tenía 60,048 registros, esta nueva tabla es un subconjunto de la original.

Campo	Valores	Descripción
NU-CTE	1-45,717	Número aleatorio para no mostrar datos confidenciales
Y	0,1	Variable Objetivo (tiene crédito en los próximos 3 meses) 1 si tiene crédito 0 caso contrario.
IMy	0-14,000,000	Variable Objetivo Monto en pesos Mexicanos (tiene crédito en los próximos 3 meses)
MONTO-EXP	0-14,000,000	Monto en pesos Mexicanos de Créditos simples que se le han otorgado en 12 meses
CREDIS-EXP	0-15	Número de créditos simples en 12 meses que se le han otorgado
MoraTarj	0-1	Atraso de más de 30 días en tarjeta, 1 si cuenta cae en mora en ese rango de días en el producto Tarjeta de Crédito
MoraCS	0-1	Atraso de más de 30 días en CS, 1 si cuenta cae en mora en ese rango de días en el producto Crédito Simple
SALIDAS-100	0-2,019	Número de Salidas (cargos) de más de 100 mil en un periodo de 3 meses
SALIDAS-300	0-857	Número de Salidas (cargos) de más de 300 mil en un periodo de 3 meses
SALIDAS-600	0-746	Número de Salidas (cargos) de más de 600 mil en un periodo de 3 meses
SALIDAS-1M	0-698	Número de Salidas (cargos) de más de 1 M en un periodo de 3 meses
Liquidez-pyme	0-46M	Saldo en pesos Mexicanos con cuentas PyME
Liquidez-total	0-166M	Saldo en pesos Mexicanos en todas las cuentas
Bcom	0-1	Canal digital , 1 si cuenta con acceso a banca en línea, 0 otro caso
TDCN	0-1	Tiene tarjeta, 1 si cuenta con tarjeta, 0 otro caso
TP-PERSONA	F,G,M	Tipo de persona SHCP Física, Gobierno o Moral

Tabla 2.1: Breve descripción de la tabla de datos proporcionada para generar el modelo. Fuente: Elaboración Propia.

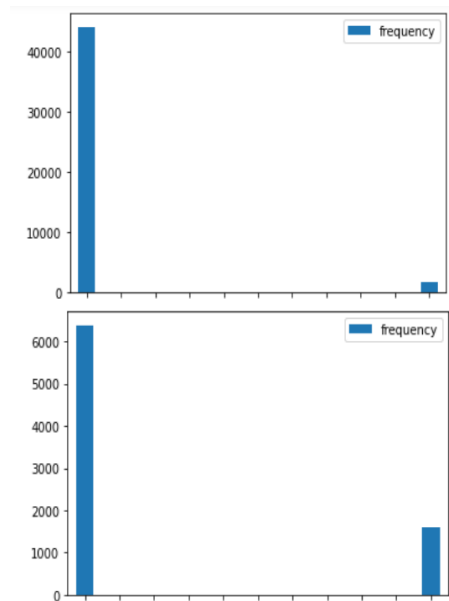


Figura 2.1: Comparación del número de positivos antes (arriba) y después (abajo) del balanceo de la muestra. Fuente: Elaboración Propia.

2.2. Explore

El primer paso para explorar los datos consiste en encontrar la proporción de positivos versus negativos para poder balancear la muestra de aprendizaje y poder encontrar mejor las relaciones. Este paso es poco mencionado, pero es crucial para mejorar los modelos. No solo funciona para que los modelos de Machine Learning aprendan mejor, también, para valorar mejor las relaciones entre las variables. Es por esto último, que es preciso desarrollar este paso primero. La lógica detrás de este paso es que un modelo que aprende de una muestra con una proporción de positivos muy pequeña, por ejemplo, 1 %, aprenderá que lo mejor es marcar cada registro como falso; obteniendo una efectividad del 99 %. En las Figura 2.1, se puede ver un comparativo de cómo queda la muestra cuando balanceamos las cargas de aprendizaje:

En este caso, se ha utilizado una proporción de positivos del 20 %. Sin embargo, es necesario precisar, que la elección de las proporciones debe de hacerse con un ejercicio de prueba y error para alcanzar la mejor muestra de aprendizaje. El código en Spark con Python para balancear la muestra queda de la siguiente forma:

```

1 A=df.filter(df.Y ==0)
2 B=df.filter(df.Y==1)
3 tamb=B.count()
4 tama=tamb*(0.8/0.2)
5 propa=tama/(A.count())

```

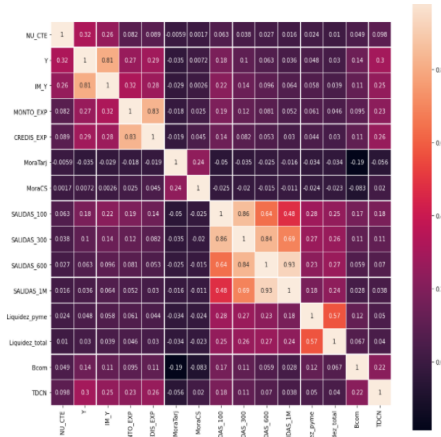


Figura 2.2: Comparación del número de positivos antes del balanceo de muestra.
Fuente: Elaboración Propia.

```
6 (A1, A2) = A.randomSplit([1-propa, propa])
7 df2 = B.union(A2)
```

Código 2.1: Balanceo de la muestra. Fuente: Elaboración propia.

Uno de los argumentos en contra de esta metodología es que reduce el tamaño de la muestra. En las Figuras 2.2 y 2.3, se muestra un ejemplo de mejora en la obtención de correlaciones entre la variable objetivo y las variables de entrada usando el balanceo de la muestra.

En la Figura 2.3, podemos apreciar que las variables que más se relacionan con la variable Y (bajo el coeficiente de correlación de Pearson), son las variables de salidas en los tres últimos meses. Además si el cliente cuenta ya con una tarjeta de crédito o con la experiencia previa en el uso d crédito se aprecia una correlación más alta con la variable Y.

Sin embargo, será necesario buscar otro tipo de relaciones basadas en distribuciones conjuntas. Una metodología para lograr esto es correr un algoritmo random forest en múltiples ocasiones (se recomienda unas 500 ocasiones para este experimento debido a que si se corría en más ocasiones no se obtenían cambios en los resultados, en cambio, sí se corría en menos ocasiones, se obtenían variaciones en los resultados). Para poder obtener las mejores variables (las que más explican Y) el algoritmo random forest (usando combinaciones aleatorias y la medida de ganancia de la información) permitirá saber qué variable o conjunto de variables son las más importantes para predecir la variable objetivo Y.

Sin embargo, este proceso, por su naturaleza será presentado más adelante. No será necesario abundar más sobre las relaciones entre variables, dejando de lado temas como cópulas o la correlación de t de Kendal ya que se espera que

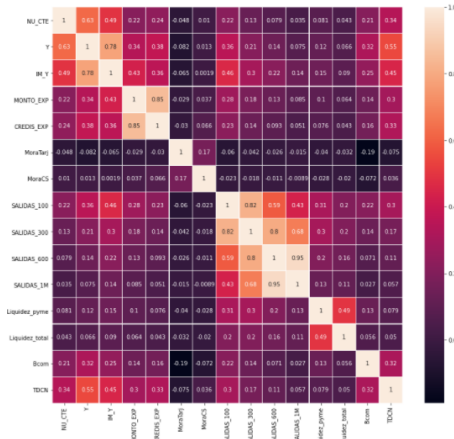


Figura 2.3: Comparación del número de positivos después del balanceo de muestra. Fuente: Elaboración Propia.

los algoritmos aprendan sobre las relaciones entre las variables y nos muestren los resultados posteriormente³.

En la Figura 2.4 se pueden observar las correlaciones gráficamente. Sin embargo, no será necesario profundizar en el análisis ya que la matriz de correlaciones dejará más información que una inspección visual de las distribuciones conjuntas. Se puede ver que las variables de gastos y liquidez se encuentran altamente correlacionadas. En la diagonal de la matriz de dispersiones, se gráfica el histograma de cada variable.

Una forma adecuada de evaluar las relaciones entre las variables es genera un diagrama de árbol de decisión condicional, en donde se podrán encontrar las correlaciones condicionales entre las variables. Lamentablemente, este tipo de gráficos, no pueden ser generados en todas las plataformas. Una de estas plataformas que no cuentan con la capacidad para graficarlos es Datio, por lo que se recurrirá al uso del lenguaje R para poder mostrar los beneficios de este diagrama en el análisis de la información.

En la Figura 2.5 se pueden apreciar las relaciones que ha construido el algoritmo con los datos de entrenamiento, debido a que el formato de salida de R cuenta con una letra pequeña, no se puede apreciar el resultado de cada rama en la imagen de este texto, el lector interesado podrá ver la imagen en formato PDF y legible en (Apodaca, 2022). En la parte de abajo de la Figura 2.5 se puede apreciar una barra; cuando la barra tiene más color oscuro que claro, se entenderá que es más probable obtener un crédito.

En la Figura 2.6 se pueden apreciar las relaciones que ha construido el algoritmo con los datos de entrenamiento, pero en un nivel de acercamiento en el que podemos dimensionar y apreciar las relaciones entre las variables. Por

³Para mayor información sobre medidas de dependencia se puede ver (Apodaca, 2014).

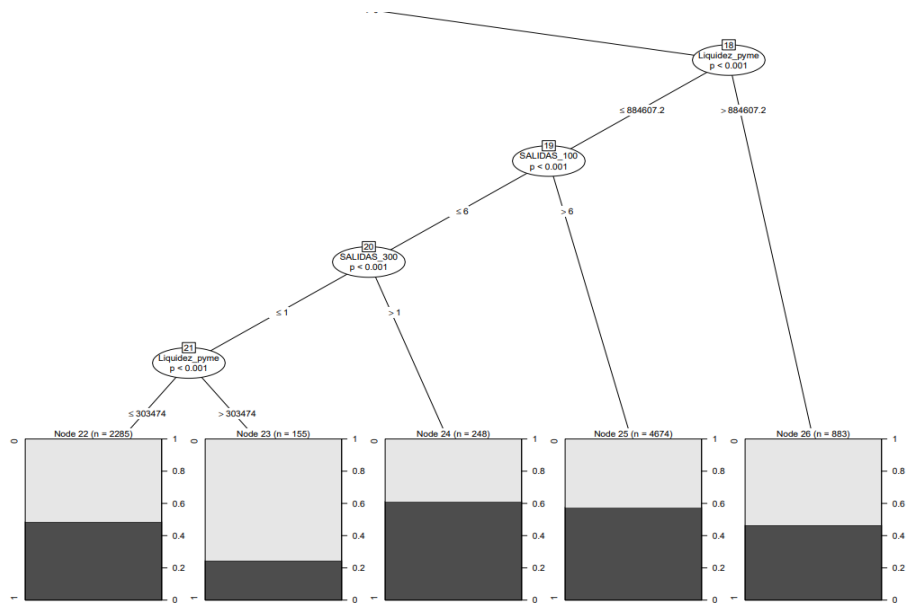


Figura 2.6: . Acercamiento al árbol de decisión condicional generado en R. Fuente: Elaboración Propia.

ejemplo, mientras más salidas contra liquidez realice una PyME, más probable será que requiera un préstamo, pero esto depende de otras variables como mora o experiencia en el producto como se muestra en los niveles más altos del árbol de decisión. El código utilizado para generar las Figuras 2.5 y 2.6 se encuentra en el anexo.

Las relaciones encontradas en el árbol condicional nos ayudaran a determinar qué es lo que los algoritmos de Machine Learning están aprendiendo de la información proporcionada y cómo los patrones encontrados podrán traducirse en un futuro en información para mejorar las estrategias de venta. Los patrones que ha encontrado el algoritmo de árbol condicional son los siguientes:

```

1) TDCN <= 0;
2)* weights = 51843
3) TDCN > 0
4) Bcom <= 0;
5) CREDIS_EXP <= 0;
6) SALIDAS_100 <= 7;
7)* weights = 1348
8) SALIDAS_100 > 7
9)* weights = 159
4) CREDIS_EXP > 0
8)* weights = 32
3) Bcom > 0
9) CREDIS_EXP <= 0;
10) SALIDAS_100 <= 0;

```

```

15 11) Liquidez_total <= 175107.9;
16 12) CARGOS <= 99561.87;
17 13)* weights = 370
18 12) CARGOS > 99561.87
19 14) Liquidez_total <= 23783.29;
20 15)* weights = 1029
21 14) Liquidez_total > 23783.29
22 16)* weights = 1301
23 11) Liquidez_total > 175107.9
24 17)* weights = 257
25 10) SALIDAS_100 > 0
26 18) Liquidez_pyme <= 884607.2;
27 19) SALIDAS_100 <= 6;
28 20) SALIDAS_300 <= 1;
29 21) Liquidez_pyme <= 303474;
30 22)* weights = 2285
31 21) Liquidez_pyme > 303474
32 23)* weights = 155
33 20) SALIDAS_300 > 1
34 24)* weights = 248
35 19) SALIDAS_100 > 6
36 25)* weights = 4674
37 18) Liquidez_pyme > 884607.2
38 26)* weights = 883
39 9) CREDIS_EXP > 0
40 27) SALIDAS_600 <= 1;
41 28)* weights = 991
42 27) SALIDAS_600 > 1
43 29)* weights = 1016

```

Código 2.2: Resultados de las relaciones entre variables por un árbol de decisión condicional. Fuente: Elaboración Propia.

El patrón más fuerte (al tener la mayor ganancia de información con base en el resultado de la fórmula uno) es el de experiencia previa del cliente con el uso de créditos, se puede apreciar, que el uso de tarjeta previo, predispone al consumo futuro del cliente (ramas uno y nueve).

Posteriormente, otra variable será de gran ayuda para medir la propensión del cliente a tomar un crédito, pues una alta liquidez de un cliente (ramas 11, 14, 18 y 21) muestran que, al aumentar la liquidez, el cliente será menos propenso a solicitar un crédito.

También, el algoritmo detectó que los gastos acumulados durante los últimos meses pueden generar la necesidad de crédito como se aprecia en las ramas que cuentan con salidas mayores a 600 mil, a 100 mil o con cargos en sus cuentas.

Una de las ventajas del uso de árboles de decisión condicional con respecto al análisis de regresión, es que para las regresiones se suele considerar que las variables usadas en el modelo tienen una relación lineal con la variable objetivo, en el caso de los árboles de decisión condicional, esto no es necesario. Además, se puede observar cómo estas variables interactúan de forma unida y no por separado para llegar a un resultado, por ejemplo, la experiencia previa del cliente con el uso de crédito combinado con las salidas acumuladas mayores a 600 mil pesos durante tres meses como se aprecia en la rama 27.

Además, el algoritmo del árbol de decisión condicional ha podido generar

puntos de corte obtenidos con función de ganancia de información cuando es máxima. De esta forma, se podrá asignar una estrategia comercial a cada tipo de cliente ya que un cliente que ha tenido muchos gastos en un periodo de tiempo y tiene poca liquidez, es diferente a un cliente que ha tenido los mismos gastos, pero mayor liquidez (en este caso, por mayor, o menos liquidez, se entenderá que un cliente sobrepasa o no el punto de corte determinado por el algoritmo con base en la función de ganancia de la información).

Otra de las ventajas de este algoritmo es que puede incluir información categórica o continua, además de poder trabajar con valores nulos en la información, proporcionando una ventaja con respecto a otros algoritmos debido a que disminuye la cantidad de tiempo dedicado a limpiar o transformar los datos.

2.3. Modify

Para modificar los datos antes de entrenar los modelos con ellos, será preciso elegir las variables y transformarlos para una mejor efectividad. En concreto, no serán seleccionadas las variables categóricas, (tipo de persona y giro) por no mostrar una alta correlación. Sin embargo, se pueden incluir dentro de un experimento para poder validar con los resultados del modelo si fue efectivo o no el hecho de incluir la variable. También, se quitaron por tener poca efectividad con el análisis de relación con la variable objetivo con un random forest que se mostrará más adelante.

En el caso de la transformación de datos, usaremos la estandarización de la muestra; En concreto se trata de aplicar la siguiente fórmula a cada variable:

$$x_2 = \frac{x_1 - \mu}{\sigma} \quad (2.1)$$

En donde x_1 es un valor de la variable x_1 a transformar, x_2 es el valor resultante. Los valores μ y σ son la media y desviación estándar de la variable x_1 respectivamente

Para aplicar estas transformaciones se puede usar el siguiente código que es simple y se explica en (spark.apache.org, 2021) y (Bill, 2017):

```
1 from pyspark.ml.feature import StandardScaler
2 # Normalizamos las variables
3 standard_scaler = StandardScaler(inputCol='features', outputCol='
  standardized_features')
```

Código 2.3: Aplicar la función standard a los campos para normalizarlos. Fuente: Elaboración Propia.

De esta forma, los resultados del modelo serán más precisos debido a que se estarán dimensionando los datos en una sola métrica. Por ejemplo, no se comparará monto de experiencia de cuatro millones contra la variable de tarjeta que arroja valores uno o cero.

Existen otras transformaciones que se pueden aplicar a los datos. Una recomendación será hacer pruebas en un mismo modelo cambiando estas transformaciones para elegir la más adecuada. Las transformaciones que se pueden

encontrar en las bibliotecas de Python y Spark (que generan menos errores al momento de enviar un dato transformado a los algoritmos de Tensor Flow debido a que se encuentran aceptadas y programadas en Spark), son las transformaciones de Box Cox y Yeo Jhonson (Weisberg, 2001). Las cuales se definen de la siguiente forma:

$$y_i^{(\lambda)} = \begin{cases} \frac{y_i^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0, \\ \ln y_i & \text{if } \lambda = 0, \end{cases} \quad (2.2)$$

La función 2.2 es la transformación Box Cox del dato y , en donde λ es un número positivo que puede ser encontrado con un algoritmo de optimización que busca como objetivo que la distribución sea lo más estandarizada posible (media cero y mejor simetría). Es preciso notar que esta transformación requiere que y sea positiva, lo cual ha propiciado el surgimiento de nuevas transformaciones que adicionan parámetros u operaciones para lograr esto; por ejemplo, sumar un número muy alto a y para asegurar que sea positiva. Sin embargo, una solución más apropiada a este problema se encuentra en la transformación Yeo Jhonson:

$$y_i^{(\lambda)} = \begin{cases} ((y_i + 1)^\lambda - 1)/\lambda & \text{if } \lambda \neq 0, y \geq 0 \\ \log(y_i + 1) & \text{if } \lambda = 0, y \geq 0 \\ -((-y_i + 1)^{(2-\lambda)} - 1)/(2 - \lambda) & \text{if } \lambda \neq 2, y < 0 \\ -\log(-y_i + 1) & \text{if } \lambda = 2, y < 0 \end{cases} \quad (2.3)$$

En donde λ es un parámetro que se puede encontrar con un algoritmo de optimización, buscando que la distribución resultante sea lo más estandarizada posible. Como ejemplo de la aplicación de estas transformaciones a los conjuntos de datos, se muestra en el Código 2.4 el cambio en las distribuciones de acuerdo a un experimento realizado en (scikit-learn, 2022):

```
1 from sklearn.preprocessing import PowerTransformer
2 from sklearn.preprocessing import QuantileTransformer
3 from sklearn.model_selection import train_test_split
4 bc = PowerTransformer(method='box-cox')
5 yj = PowerTransformer(method='yeo-johnson')
```

Código 2.4: Aplicar transformaciones a los campos para obtener mejores resultados. Fuente: Elaboración propia con las funciones de la librería PowerTransformer.

2.4. Model

Ha llegado el momento de modelar, esto es, aplicar modelos a los datos analizados y transformados. El primer modelo será una regresión logística, que corre con el Código 3.5 que funcionará como base para los siguientes modelos. Este código es generado fácilmente leyendo el manual de SPARK (spark.apache.org, 2021) y en donde se encontrarán más detalles y modelos para probar.

Para poder encontrar cuáles son las características de cada uno de los modelos se deben realizar múltiples simulaciones de cada modelo con diferentes parámetros. Para hacer esto, se utilizó la librería GridSearchCV de Python que a cada posible combinación de parámetros entrena un modelo y éste es probado en un número finito de muestras distintas del conjunto de validación (validación cruzada) y, de esta forma, calcula el promedio y la desviación estándar de una función de costo. Como se mencionó, este proceso puede tomar horas, días, meses u años, dependiendo del número de parámetros que se requiera combinar para probar diferentes modelos. Los parámetros de los modelos presentados en este caso de estudio son el resultado de aplicar la librería GridSearchCV de Python para encontrar el modelo que más se adapte al fenómeno que se estudió. El código para realizar este ejercicio se muestra en el Anexo.

En el Código 2.5 se muestran las sentencias para entrenar un modelo en un ambiente distribuido con el lenguaje Spark con el ejemplo de la Regresión Logística. Se puede observar que se ha definido un pipeline que permite ordenar los procesos que seleccionan y modifican los datos para después entrenar el modelo. De la misma forma se aplica el código para los diferentes modelos mencionados anteriormente. Los resultados de los modelos se mostrarán en la siguiente sección.

```

1 # Definimos el modelo:
2 log_reg = LogisticRegression(featuresCol='standardized_features',
3                               labelCol='Yfinal') #recibe s f de la funcion normalizadora
4
5 #Generamos un vector (unimos todas las variables en una) para que
6   Spark pueda correr el modelo
7
8 vectorAssembler = VectorAssembler(inputCols =
9   ['MONTO_EXP', 'CREDIS_EXP', 'MoraTarj', 'MoraCS', 'SALIDAS_100', '
10    SALIDAS_300', 'SALIDAS_600', '
11    SALIDAS_1M', 'Liquidez_pyme', 'Liquidez_total', 'Bcom', 'TDCN'],
12    outputCol = 'features')
13
14 #Generamos una nueva tabla para ingestar al modelo
15 vdf = vectorAssembler.transform(df)
16 vdf = vdf.select(['features', 'Yfinal'])
17
18 #Dividimos la muestra en entrenamiento y en validaci n
19 (trainingData, testData) = vdf.randomSplit([0.7, 0.3])
20
21 # Lo ponemos en cola de producci n en Pipeline
22 pipeline_log_reg = Pipeline(stages=[standard_scaler, log_reg])
23
24 #Entrenamos el modelo
25 pipeline_model_log_reg = pipeline_log_reg.fit(trainingData)
26
27 #Calificamos la muestra de test para validar el modelo
28 tarjeta_test_log_reg = pipeline_model_log_reg.transform(testData)

```

Código 2.5: Código en Spark para entrenar los modelos en un ambiente distribuido (ejemplo con la Regresión Logística). Fuente: Elaboración propia.

Los parámetros y funciones usadas para generar los modelos se muestran a continuación:

```

1 #Red Neuronal:
2 layers = [12,45,25,2]
3 MultilayerPerceptronClassifier(featuresCol='standardized_features',
4                                 labelCol='Yfinal',

```

```

4  maxIter=100, layers=layers,
5  seed=1234, )
6
7  #Ada Boost:
8  GBTCClassifier(featuresCol='standardized_features', labelCol='Yfinal',maxIter=10)
9
10 #Arbol de decision:
11 DecisionTreeClassifier(featuresCol='standardized_features',
12   labelCol='Yfinal')
13
14 #Random Forest:
15 RandomForestClassifier(featuresCol='standardized_features',
16   labelCol='Yfinal')
17
18 #Regresi n Logistica:
19 LogisticRegression(featuresCol='standardized_features', labelCol='
20   Yfinal')
21
22 #SVM:
23 LinearSVC(featuresCol='standardized_features', labelCol='Yfinal')

```

Código 2.6: Definición de los modelos en SPARK. Fuente: Elaboración propia.

En el caso de la Red Neuronal, fue seleccionada una estructura descendente, esto es, las neuronas van disminuyendo capa a capa. Después de realizar diferentes experimentos, la efectividad del modelo no cambiaba con un aumento en el número de neuronas en cada capa, es por ello que se eligió esta topología.

Para poder evaluar más parámetros del modelo, será necesario evaluar diferentes escenarios. Para ello, se utilizó la Biblioteca TensorFlow que proporciona facilidades para simular diferentes escenarios de una red neuronal. En el Código 2.7, se aprecia la definición de una red neuronal de cuatro capas, con 12 entradas, y una salida, así como dos capas de 45 y 25 neuronas.

```

1  inputs = tf.keras.layers.Input(shape=(12)) # es el n mero de
2    entradas
3  h1 = tf.keras.layers.Dense(45, activation='relu')(inputs) # capa
4    oculta 1
5  h2 = tf.keras.layers.Dense(25, activation='sigmoid')(h1) # capa
6    oculta 2
7  outputs = tf.keras.layers.Dense(1, activation='sigmoid')(h2) #
8    salidas
9  model = tf.keras.models.Model(
10    inputs=inputs,
11    outputs=outputs)
12
13  model.compile(optimizer="Adam", loss="binary_crossentropy", metrics
14    =["acc"])
15  model.metrics_names

```

Código 2.7: Definición de una red neuronal con la librería Tensor Flow. Fuente: Elaboración propia”.

También se observa, cómo se puede definir la función de salida Sigmoidal o ReLU para cada capa. Se ha elegido el método de optimización ADAM y se ha seleccionado la función de costo CE. Así mismo, es posible definir las métricas

```
[49]: # Training the model
import time
start = time.time()
history=model.fit(train_features, train_Y, epochs=160, callbacks=callbacks
                  , batch_size=3
                  )
end = time.time()
#print("Time elapsed: {}".format(end-start))

Train on 8153 samples
Epoch 1/160
8121/8153 [=====>.] - ETA: 0s - loss: 0.4358 - acc: 0.8578
Learning rate for epoch 1 is 0.001000000474974513
8153/8153 [=====] - 10s 1ms/sample - loss: 0.4377 - acc: 0.8580
Epoch 2/160
8103/8153 [=====>.] - ETA: 0s - loss: 0.3998 - acc: 0.8742
Learning rate for epoch 2 is 0.001000000474974513
8153/8153 [=====] - 6s 742us/sample - loss: 0.3990 - acc: 0.8743
Epoch 3/160
8139/8153 [=====>.] - ETA: 0s - loss: 0.3594 - acc: 0.8747
Learning rate for epoch 3 is 0.001000000474974513
8153/8153 [=====] - 6s 775us/sample - loss: 0.3592 - acc: 0.8746
Epoch 4/160
8118/8153 [=====>.] - ETA: 0s - loss: 0.4876 - acc: 0.8586
Learning rate for epoch 4 is 0.001000000474974513
8153/8153 [=====] - 6s 734us/sample - loss: 0.4870 - acc: 0.8583
Epoch 5/160
8097/8153 [=====>.] - ETA: 0s - loss: 0.3342 - acc: 0.8852
Learning rate for epoch 5 is 0.001000000474974513
8153/8153 [=====] - 6s 746us/sample - loss: 0.3339 - acc: 0.8850
```

Figura 2.7: Salida de código al definirse una Red Neuronal multicapa. Fuente: Elaboración Propia.

para poder evaluar el modelo. En la Figura 2.7, será posible ver los resultados de cinco entrenamientos de la Red Neuronal con los parámetros previamente definidos. Se aprecia que, en este caso, que la efectividad del modelo queda en 0.85.

En la Figura 2.7, se puede apreciar cómo se generan cinco iteraciones de entrenamiento (repite el proceso de entrenamiento cinco ocasiones) con tres muestras aleatorias del grupo de entrenamiento. Este tipo de entrenamiento suele llamarse validación cruzada.

2.5. Asses

A continuación, se mostrará en resumen las diferentes métricas aplicadas a los modelos que fueron generados y aplicados a la muestra de prueba. En específico, se muestra el Recall, Precision, Efectividad y los valores de la matriz de confusión, se muestra en la Tabla 2.2.

Se puede observar que los diferentes modelos muestran diferentes resultados. Si bien el Random Forest muestra la mejor efectividad tanto en valores positivos o negativos, para el pronóstico de los valores de interés (positivos) queda por debajo de otros modelos como la Red Neuronal y el Árbol de decisión. En este caso, buscamos un mejor Recall. Por otra parte, la Regresión Logística y la Máquina de Soporte Vectorial parecen ser un modelo promedio, pero se puede

apreciar que es porque le da mayor peso a pronosticar los negativos que no son de interés.

Para poder valorar mejor qué modelo es más preciso, se puede apreciar en la Figura 3.8 que la curva ROC proporciona un panorama visual que puede ayudar a mostrar la efectividad del modelo.

A continuación, se analizarán las curvas de los modelos puestos a prueba para compararlos. En el caso de la Red Neuronal y la SVM, no se generaron las curvas debido a que la versión de Spark con la que se cuenta, no arroja el parámetro de probability que se encuentra en la salida del entrenamiento de los modelos y que representa el valor que el modelo le otorga al cliente para poder determinar si usará crédito o no. Sin embargo, se cuenta con las otras métricas y no será necesaria su gráfica para poderla comparar con los otros modelos.

Modelo	Random Forest	Ada Boost	Logit	Red Neuronal	Árbol Decisión	SVM
Recall	0.7262	0.7143	0.754	0.754	0.7619	0.6949
Precision	0.7804	0.7725	0.751	0.751	0.7456	0.7610
Efectividad	0.9013	0.8976	0.8976	0.8976	0.8972	0.8890
Verdaderos Negativos	1834	1831	1864	1811	1806	1810
Falsos Negativos	138	144	138	124	120	113
Falsos positivos	103	106	73	126	131	71
Verdaderos positivos	366	360	366	380	384	360

Tabla 2.2: Diferentes métricas para evaluar los diferentes modelos puestos a prueba. Fuente: Elaboración Propia.

Se puede observar (curva azul) que la regresión logística genera mejor efectividad, pero cuenta con dificultades al momento de predecir a toda la población uniformemente (la curva azul se encuentra por debajo de las otras curvas cuando se aproxima a uno en el eje vertical); se observa que predice bien solo ciertos registros y otros no. Esto puede ser ocasionado por no contemplar la no-linealidad en este modelo. El código para elaborar esta gráfica en Spark se muestra a continuación:

```

1 from pyspark.mllib.evaluation import BinaryClassificationMetrics
2 import matplotlib.pyplot as plt
3 class CurveMetrics(BinaryClassificationMetrics):
4     def __init__(self, *args):
5         super(CurveMetrics, self).__init__(*args)
6     def _to_list(self, rdd):
7         points = []
8         for row in rdd.collect():
9             points += [(float(row._1()), float(row._2()))]
10        return points
11    def get_curve(self, method):
12        rdd = getattr(self._java_model, method)().toJavaRDD()
13        return self._to_list(rdd)
14 preds = tarjeta_test_log_reg.select('Yfinal', 'probability').rdd.map(
15     (lambda row:
16      (float(row['probability'][1]), float(row['Yfinal'])))
17 preds2 = tarjeta_test_rf.select('Yfinal', 'probability').rdd.map(
18     (lambda row:
19      (float(row['probability'][1]), float(row['Yfinal'])))
20 preds3 = tarjeta_test_tree.select('Yfinal', 'probability').rdd.map(
21     (lambda row:
22      (float(row['probability'][1]), float(row['Yfinal'])))
23 preds4 = tarjeta_test_gbt.select('Yfinal', 'probability').rdd.map(
24     (lambda row:
25      (float(row['probability'][1]), float(row['Yfinal'])))
26 preds5 = red_NN.select('Yfinal', 'prediction').rdd.map(lambda row: (
27     float(row['probability'][1]),
28     float(row['Yfinal'])))
29 points = CurveMetrics(preds).get_curve('roc')
30 points2 = CurveMetrics(preds2).get_curve('roc')
31 points3 = CurveMetrics(preds3).get_curve('roc')
32 points4 = CurveMetrics(preds4).get_curve('roc')
33 #points5 = CurveMetrics(preds5).get_curve('roc')
34 plt.figure(figsize=(10,10))
35 x_val = [x[0] for x in points]
36 y_val = [x[1] for x in points]
37 x_val2 = [x[0] for x in points2]
38 y_val2 = [x[1] for x in points2]
39 x_val3 = [x[0] for x in points3]
40 y_val3 = [x[1] for x in points3]
41 x_val4 = [x[0] for x in points4]
42 y_val4 = [x[1] for x in points4]
43 recta_x=[0,1]
44 recta_y=[0,1]
45 plt.plot(x_val, y_val)
46 plt.plot(x_val2, y_val2)
47 plt.plot(x_val3, y_val3)

```

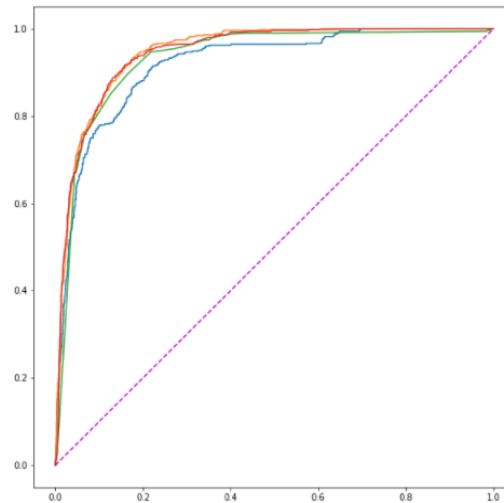


Figura 2.8: Curvas ROC de los modelos evaluados. Regresión Logística azul, Árbol de Decisión naranja, Ada Boost roja, Random Forest verde. Fuente: Elaboración Propia.

```
43 plt.plot(x_val4, y_val4)
```

Código 2.8: Código en Spark para generar curva ROC. Fuente: Elaboración propia.

Por otra parte, se tienen los resultados de las correlaciones encontradas por medio del método de Random Forest. Como se explicó en la sección anterior; este método genera centenares de árboles de decisión para poder determinar cuál es la variable más importante. En este caso, el resultado en orden descendente de importancia se muestra a continuación:

```
1 [('SALIDAS_300'),
2  ('TDCN'),
3  ('SALIDAS_100'),
4  ('SALIDAS_600')]
```

Código 2.9: Variables más importantes con base en un algoritmo de Random Forest. Fuente: Elaboración propia.

Se puede observar, que las salidas mayores a 300 mil pesos son la variable más importante para poder pronosticar si el cliente pedirá un crédito en los siguientes tres meses. Por otra parte, la experiencia que el cliente tiene con el crédito dentro del banco es muy importante, ya sea que cuente con el producto de tarjeta de crédito PyME o con montos de crédito simple en el pasado. Las salidas altas (mayores a 600 mil) son importantes, pero no tanto como las de 100 mil o 300 mil, esto puede generar la hipótesis que es más importante para la PyME

un efecto bola de nieve de gastos pequeños acumulados que una compra de un activo de más de 600 mil pesos. Sin embargo, la discusión de estos resultados se presentará en la sección de conclusiones.

Capítulo 3

Implementación del modelo

3.1. Plan de implementación

Para poder aplicar el modelo generado, se creó una tabla de tiempos de respuesta de cada área implicada en el proceso. Se detectaron cuatro áreas que estarían implicadas y que tendrían diferentes actividades para poder aplicar el modelo y llevarlo a la fuerza de ventas. Antes de continuar con el plan, será necesario definir el objetivo del plan y definir algunos conceptos que son propios de la organización. El objetivo de la implementación se muestra a continuación:

Definición 3.1.1 (Objetivo de implementación.) *Generar un ranking (ordenamiento de mayor a menor de un conjunto de registros con base en un parámetro) de clientes que cuentan con campaña activa (término usado para referirse a un cliente que tiene una oferta aprobada activa), cumpliendo con las siguientes características:*

1. *El ranking deberá de ser por cada portafolio de cada promotor.*
2. *Poner el ranking en la aplicación de Max Pyme para que la fuerza de ventas (cada promotor en el país) pueda visualizar cuáles son los clientes por los que puede ir primero antes que otros.*
3. *Obtener un incremento en la productividad del gestor al priorizar los clientes a los que debe ofrecerle un crédito.*
4. *El ranking debe de ser actualizado cada mes cuando sean entregadas las campañas mensuales.*

Por ranking, se entenderá una lista ordenada por importancia (propensión a tomar el crédito) por cada promotor de ventas. El software Max Pyme es una plataforma que muestra información de los clientes a la fuerza de ventas. Un cliente con campaña activa es aquel que cuenta con una o más ofertas durante un mes y estas ofertas son calculadas de acuerdo a su comportamiento en pagos y uso de los productos financieros ofrecidos.

Periodo	Actividad	Área encargada	Tiempo (horas)	Software
T-4	Generación del modelo	Analytics	2	Spark
T-4	Etiquetar universo de clientes	Analytics	2	Spark
T-3	Ponerles plantilla de ventas	Analytics	2	Spark
T-3	Etiquetar campañas	Analytics	2	Spark
T-2	Enviar a CRM	CRM	4	Excel
T-2	Subir etiquetas a sistema	CRM	2	MaxPyME
T-1	Validación de Implementación	CRM	2	MaxPyME
T-0	Entrega al gestor	Ventas	6	MaxPyME
T+1	Contacto con el cliente	Ventas	1	Canal de ventas
T+2	Visita al cliente	Ventas y Cliente	2	Canal de ventas

Tabla 3.1: Mapa del proceso en áreas, recursos necesarios y tiempos de respuesta para su implementación. Fuente: Elaboración Propia.

Ya que se han definido algunos conceptos clave y de igual forma el objetivo final del modelo, será necesario mapear las actividades de las cuatro áreas involucradas y los tiempos de producción para poder implementar el modelo. En resumen, se tiene la siguiente la Tabla 3.1 con los datos mapeados:

La Tabla 3.1, muestra en orden descendente la prioridad con la que las diferentes áreas reciben insumos para poder implementar el modelo. En la columna Tiempo es expresada la variable T que toma valores enteros entre menos cuatro y dos, ya que expresa en días antes o después de que es lanzado el modelo en la plataforma MaxPyME (puesto en producción). De tal forma que el día de lanzamiento es $T-0$. De esta forma se podrá aprovisionar el tiempo de trabajo anterior para no tener retrasos en las entregas.

La metodología AGILE que ha implementado BBVA (BBVA, 2019) mejora la eficiencia y efectividad para poder implementar un nuevo proyecto. Con ella, se forman equipos multidisciplinarios que solo se dedican a un conjunto de tareas específicas (Rasmusson, 2010), como es en este caso. De tal forma que un nuevo proyecto es implementado en el flujo preconcebido para lanzamientos. Cada área recibe un insumo de otra y ya cuenta con las tecnologías y conocimiento para hacer su parte. De esta forma, la persona que sube el modelo a la plataforma no necesita saber cómo se ha generado el modelo, solo sabe que recibirá en tal fecha una lista de clientes que debe subir a la plataforma.

Por otra parte, la autogestión de los equipos definidos con la metodología AGILE permite dejar de lado la burocracia corporativa, permitiendo que nuevas ideas como ésta sean implementadas fácilmente y sin encontrar problemas políticos dentro de la empresa. De esta forma, los Directores se convierten en proveedores de herramientas para cada una de las áreas y esperan de los equipos nuevas ideas para poder implantarse.

Además, la metodología AGILE permite minimizar costos puesto que la infraestructura ya está puesta para solo recibir nuevas ideas. En cambio, en otro tipo de gestión, en cada proyecto nuevo, habría que organizar un equipo y comprar infraestructura o contratar proveedores que lo hagan. En cambio, la misma infraestructura es usada para varios proyectos.

Sobre los retos que presentó la puesta en marcha de la implementación, fue necesario convencer a los tomadores de decisiones que este modelo podría mejorar la productividad de la fuerza de ventas. Sin embargo, usando el hecho de que el costo sería mínimo debido a que ya se cuenta con la infraestructura y que lo mejor sería probar el uso de esta nueva herramienta en la fuerza de ventas para ver cómo aceptaba el usuario final (la fuerza de ventas) la herramienta; antes de desechar la idea, entonces, no se tuvieron mayores obstáculos para la aprobación de la puesta en marcha del proyecto. La herramienta fue aceptada fácilmente y de forma positiva por la fuerza de ventas como se mostrará posteriormente.

La descripción del proceso que se muestra resumido en la Tabla 3.1 comienza en un primer paso con la actualización del modelo para calificar a todos los clientes que se tienen con una puntuación que es el resultado de la variable probability que nos da el output del modelo de árbol de decisión (el que mayor efectividad tiene en verdaderos positivos). Ya que se obtiene este campo que representa la propensión de un cliente a pedir un crédito, se cruza esta infor-

mación con la plantilla de colaboradores que se tienen asignados a los clientes, esto es, se obtiene el colaborador que gestionará al cliente.

Ya que se tiene esta lista, se esperan las ofertas por parte del área de Riesgos para poder filtrar solo a los clientes que tienen una oferta en específico. Ya que el área de Analytics¹ tiene esta lista, ordena de mayor a menor la variable de probability y por promotor. De esta manera se obtiene el ranking para poder enviar al área de CRM que sube la información a la herramienta de visualización de ofertas MaxPyMe.

El proceso no termina con la carga en la herramienta; el siguiente paso es que cada promotor contacte en orden del ranking a los clientes que aparecen con mayor necesidad de un crédito.

Posteriormente, el promotor cierra el trato con el cliente, y con ello, un mes después, será necesario medir la efectividad del modelo. Bajo este proceso, surgen preguntas, principalmente: ¿cuánto fue el aumento en productividad traducido a términos monetarios?

Resulta, que se tuvo un aumento mayor del 0,01 en la efectividad de éstas, posiblemente, por dedicar más tiempo a cada una de las ofertas con mayor probabilidad de formalizarse en lugar de dedicar tiempo a clientes que no tenían la intención de pedir un préstamo.

3.2. Monetización del proyecto

Se sabe que el consumo de bebidas gaseosas genera propensión a que las personas que las consumen obtengan obesidad. Sin embargo, la pregunta de ¿cuánto aportan estas bebidas a la obesidad de las personas? es más compleja de responder debido a que son más variables las que intervienen en el resultado; por ejemplo, la genética de la persona, su actividad laboral, la educación que ha recibido, su estado anímico, su dieta, etc. Es por ello que definir o probar con certeza cómo una variable afecta a un fenómeno es complicado aún con todas las herramientas estadísticas con que se cuenta debido a que son muchos otros los factores que intervienen en los fenómenos complejos.

Sin embargo, no es necesario descartar las variables por completo; Por ejemplo, un vigilante parece poco productivo a la entrada de una empresa, pero no debería ser quitado jamás. De la misma forma, se tratará de mostrar la necesidad del uso de este modelo si es usado correctamente, y, por otra parte, si bien se tienen los resultados estadísticos de la mejora en la productividad, estos son considerados por el autor como estratégicos de la empresa y no se mostrarán en este texto ya que este análisis está basado en la mejora de la efectividad de las ofertas que se encuentran en las primeras 10 que se presentan en el ranking.

Con los precedentes del párrafo anterior, se generó el siguiente modelo económico para probar la utilidad de la herramienta:

Sean las siguientes variables:

¹Área encargada de analizar la información y proponer estrategias de negocio basadas en información.

1. C: Costo por hora por promotor
2. U: Número de clientes promedio asignados a un promotor
3. P: Número de promotores en la empresa
4. T: Tiempo en horas invertidas para analizar un cliente
5. i : La tasa de interés
6. w: Costo de capital (TIE 4 a semanas)
7. r: Porcentaje a reserva de riesgo
8. b: Porcentaje de prima pagada al promotor por generar el crédito
9. g: Costo mensual del cliente (solo la parte que concierne al producto)
10. M: Monto promedio del préstamo

La ganancia neta será dada por la siguiente fórmula:

$$G_1 = (i - r - w - b)M(vp + fn) - TUC - gU \quad (3.1)$$

En donde fácilmente podemos traducirla como las ganancias brutas menos los gastos por evaluar todos los clientes. Aplicando el modelo tendríamos los siguientes cambios:

$$G_2 = (i - r - w - b)M(vp - fn) - T(vp + fp)C - gU \quad (3.2)$$

En esta nueva fórmula solo se tienen las ganancias brutas por motivo de los vp y se descuentan las ganancias por los fn (costo de oportunidad por aplicar el modelo). Por otra parte, solo son descontados los costos por evaluar los vp+fp y no toda la cartera de clientes.

De Ecuación 3.2, se observa que el mejor modelo será el que obtenga la mayor diferencia $vp - fn$. Es por ello que se seleccionó como uno de los mejores modelos al árbol de decisión ya que cuenta con una diferencia de 264. Aunque una aproximación al problema consistiría en disminuir costos, esto es, utilizar el modelo que tenga menor $vp + fp$ que en este caso, se eligió por querer maximizar la ganancia bruta por temas de rentabilidad en el momento de la aplicación del modelo, pero la decisión debe tomar en cuenta qué tasa es mayor (la de rentabilidad o la de costo).

El modelo tendrá beneficios si sucede lo siguiente:

1.
$$vp > fn \quad (3.3)$$

2.
$$(i - r - w - b)M(vp - fn) - T(vp + fp)C > gU \quad (3.4)$$

3.

$$(i - r - w - b)M(vp - fn) > gU + T(vp + fp)C \quad (3.5)$$

En caso de que alguna de estas desigualdades no se cumpla, el modelo no podrá mostrar rentabilidad si se aplica de esa forma. Aunque el punto de vista que se puede tener respecto a la monetización del modelo pueda cambiar si el enfoque es distinto; por ejemplo, si solo se le asignan al promotor los clientes con baja propensión, ya que, si el cliente necesita el crédito y lo va a pedir, no tendría sentido pagarle al gestor por ese trabajo. Bajo ese supuesto, el modelo quedaría así:

$$G_3 = (i - r - w)M(vp) - (i - r - w - b)M(fn) - T(fn + vn)C - gU \quad (3.6)$$

En este caso, la ganancia incrementará por no pagar beneficio por los clientes que son propensos a solicitar el crédito. Como tanto la primera propuesta como la segunda serían irreales de aplicar, se supondrá que el modelo servirá de herramienta para mejorar la productividad del gestor dándole más tiempo para gestionar mejor y más rápido los créditos. En este caso, la ganancia monetaria será $TUC - T(1 - q)(vp + fp)C$, con q como la proporción de T que el promotor invierte en estudiar al cliente gracias al modelo y a priorizar sus tareas. Esta proporción debería de obtenerse por medio de la observación, pero hasta ahora no se tienen resultados concretos sobre ella.

Otra forma de monetizar el modelo es aumentando un valor z a la tasa de interés a los clientes propensos debido a que cuentan con mayor demanda del crédito. O en caso de que se requiera premiarlos por quedarse en el Banco y no ir a la competencia, disminuir la tasa a estos clientes. En el primer caso, la ganancia sería de la forma:

$$G_4 = (i + z - r - w - b)M(vp) - (i - r - w - b)M(fn) - TUC - gU \quad (3.7)$$

Obteniendo una ganancia por incremento de tasa a los clientes que necesitan el crédito. Sin embargo, se recomienda estudiar las posibilidades de aumento de tasa para este tipo de clientes antes de aplicar la Ecuación 3.7. Si bien, todos estos modelos muestran formas de monetizar la aplicación del modelo, en el caso del Banco, el modelo se aplicó sin estos puntos de vista financieros ya que se encontró que el modelo es más valioso para generar una mejor experiencia al cliente y no, por el contrario, aprovecharse de su necesidad. El modelo fue aceptado para anticiparse a las necesidades del cliente y ofrecer un mejor servicio, lo cual, a largo plazo generará fidelidad y rentabilidad constante.

3.3. Evidencia de la puesta en producción

Para mostrar el resultado final, se presenta la Imagen 4.1, en donde se puede observar la lista de clientes en orden por prioridad en la que los promotores pueden consultar cada día a cada uno de sus clientes.

	Propietario del cliente	No. de cliente	UG	CR Usuario	Banca	División	Oficina
1			645509	1777	Red PyME	SUR PyME	XALAPA
2			645509	1777	Red PyME	SUR PyME	XALAPA
3			645509	1777	Red PyME	SUR PyME	XALAPA
4			645509	1777	Red PyME	SUR PyME	XALAPA
5			645509	1777	Red PyME	SUR PyME	XALAPA
6			645509	1777	Red PyME	SUR PyME	XALAPA
7			645509	1777	Red PyME	SUR PyME	XALAPA
8			645509	1777	Red PyME	SUR PyME	XALAPA
9			645509	1777	Red PyME	SUR PyME	XALAPA
10			645509	1777	Red PyME	SUR PyME	XALAPA

Imagen 4.1: Resultado final de la implementación. Fuente: Elaboración Propia.

Por otra parte, se muestra la Imagen 4.2, en donde la carga de los clientes falló y, por lo tanto, esta imagen la ha enviado un promotor al que le urgía tener el ranking para gestionar mejor su cartera. Este hallazgo muestra la necesidad generada en la fuerza de ventas y su funcionalidad.

Por último, se muestra la Imagen 4.3 de un correo en el que el autor de este trabajo envía al CRM la lista generada para cargarla en el sistema.

Con estas imágenes podemos observar la puesta en marcha del modelo aplicándolo a una herramienta que es apreciada y usada por la fuerza de ventas para mejorar la experiencia del cliente.

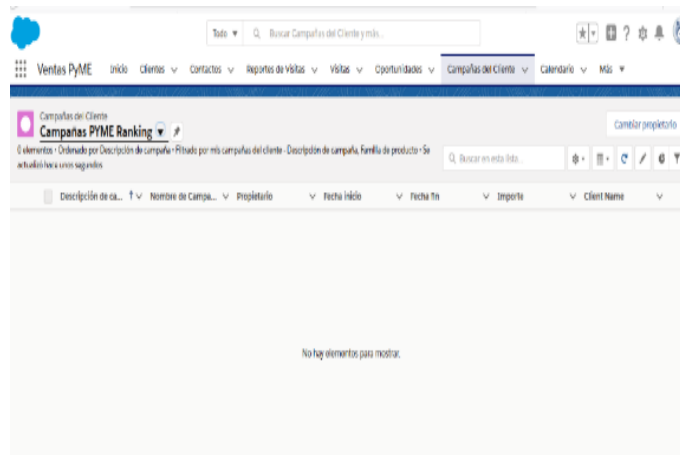


Imagen 4.2: Queja sobre problemas en la carga. Fuente: Elaboración Propia.

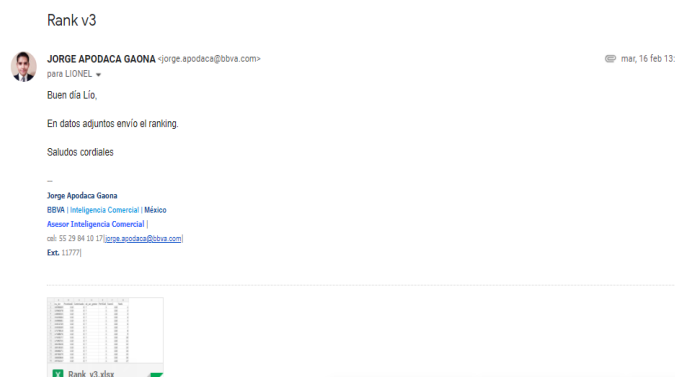


Imagen 4.3: Parte del proceso de generación del ranking. Fuente: Elaboración Propia.

Capítulo 4

Discusión y conclusiones

4.1. Sobre la IA y su implementación

El Machine Learning ha tomado relevancia en los últimos años debido a la facilidad con que pueden ser generados los modelos con algoritmos de aprendizaje y aprovechando el auge de las plataformas de BigData. Sin embargo, es necesario tener conocimiento del fenómeno a modelar e investigar la mayoría de las fuentes de información que puedan nutrir y abundar al resultado del modelo, pues no importa el tamaño del de la Red Neuronal si no se cuenta con la información que le proporcione aprendizaje.

Por otra parte, cada que se tenga oportunidad, será buena práctica incluir nuevos modelos y valorar diferentes simulaciones de sus parámetros debido a que no se tiene certeza aún sobre una metodología segura para determinar el conjunto de parámetros de los modelos; por ejemplo, la cantidad de neuronas o capas o el tamaño final de los árboles de decisión (Brío, 2007).

El balanceo de la muestra usando la variable objetivo cuando toma un valor positivo es muy importante para mejorar el aprendizaje del modelo y evitar el sobreajuste. Sin embargo, disminuye en gran medida el tamaño de la muestra de aprendizaje.

4.2. Sobre los resultados de los modelos

Los diferentes resultados de los modelos proporcionan aplicaciones distintas. Esto es, dependiendo de la métrica que se evalúe o el objetivo del modelo, será necesario elegir una u otra métrica para determinar cuál es el mejor modelo que represente un fenómeno o que sea útil para aplicarlo en la vida real. Como en el ejemplo que se mostró en este trabajo, el mejor modelo resultó ser un árbol de decisión debido a que el fenómeno económico en el que se aplicaba así lo requería.

De la misma forma que se recomienda entrenar los modelos bajo diferentes parámetros, se recomienda evaluar el modelo bajo diferentes métricas.

Una de las formas de evaluar el modelo para futuras investigaciones es la validación cruzada, en la que se evalúan los modelos bajo diferentes muestras aleatorias diferentes entre sí. Esto permite obtener mayor certeza en los resultados del modelo.

4.3. Sobre la implementación

La metodología AGILE permite aumentar la creatividad del área de Analytics debido a que proporciona el ambiente adecuado y la infraestructura para poder probar e implementar modelos que ayuden a obtener mejores resultados en la empresa. El flujo de información durante todo el proceso fue siempre claro para todas las partes involucradas y el tiempo de planeación fue menor debido a que la aplicación del modelo es parecida a la aplicación de otros en producción.

Por otra parte, la evaluación del desempeño del modelo no se puede obtener desde el punto de vista económico debido a que se trata de una aplicación que tiene como objetivo mejorar la experiencia del cliente y la vida laboral del promotor. Estos objetivos son intangibles en muchas ocasiones por lo que pueden no siempre ser medidos los resultados. Sin embargo, se muestran diferentes formas de sacar ventaja económica del modelo (Sección 4.2.).

Si bien, se puede usar para desplazar campos de trabajo o sacar provecho de las necesidades encontradas de los clientes, en cambio, se le puede dar un enfoque en el que todos sean beneficiados.

Con este modelo, se permitirá realizar trabajo futuro para la mejora de la productividad del promotor y ayudar al cliente a tener una mejor experiencia, pero será necesario aplicar nuevos modelos y obtener mayor información sobre el cliente que permita desarrollar nuevas ideas.

En un futuro, se propone indagar en la información de redes sociales o de navegación del cliente para poder ayudarlo a tomar mejores decisiones económicas. En este caso, el modelo sería enfocado en mostrar a la PyME cuál es la mejor tasa y plazo para sus diferentes proyectos de inversión.

Una variable importante, que se plantea utilizar en un futuro es la situación del giro del negocio en un momento y lugar dados para saber si la correlación con el aumento o disminución de la demanda de cierto producto puede generar propensión a la obtención del crédito. Por ejemplo, si un cliente PyME se dedica al hospedaje de personas en una zona vacacional, necesitará del crédito para mejorar sus instalaciones algunos meses antes de la temporada alta de turismo. Analizando estas correlaciones, se le podrá generar un mayor servicio anticipando sus necesidades y generando estrategias diferenciadas.

Bibliografía

- Apodaca, J. (2014). *Simulación por Computadora de Modelos Estocásticos*. UNAM.
- Apodaca, J. (2022). *Arbolito Propen*. Consultado el 1 de marzo de 2022, desde https://github.com/Elyorch35/ML/blob/main/Arbolito_Propen.pdf
- BBVA. (2019). *innovation BBVA*. Consultado el 1 de julio de 2021, desde <https://www.youtube.com/watch?v=xooKa-uwXI>
- BBVA. (2021). *BBVA Open Innovation*. Consultado el 22 de junio de 2021, desde <https://openinnovation.bbva.com/es/evento/hackathon-bbva-2020>
- Bill, C. (2017). *Spark: The Definitive Guide: Big Data Processing Made Simple*. USA O'Reilly.
- Brío, B. M. (2007). *Redes Neuronales y Sistemas Borrosos*. Alfaomega, Madrid, España.
- Caballero, R. (2018). *Big Data con Python*. España Alfaomega.
- Davenport, T. (2014). *Big data at work: dispelling the myths, uncovering the opportunities*. Harvard Business Review Press.
- Eman, S. (2019). Apache Spark: A Big Data Processing Engine. *Ingeniería, investigación y tecnología*.
- Espinosa-Zúñiga, J. J. (2020). Aplicación de metodología CRISP-DM para segmentación geográfica de una base de datos pública. *Ingeniería, investigación y tecnología*, 21(1).
- Freund, Y., Schapire, R., & Abe, N. (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780), 1612.
- Fuentes, E. J. (2020). *Introducción a los métodos Deep Learning*. España, Universidad de Vigo.
- Guerra, P., & Castelli, M. (2021). Machine learning applied to banking supervision a literature review. *Risks*, 9(7), 136.
- Hamstra, M., & Zaharia, M. (2013). *Learning Spark: lightning-fast big data analytics*. O'Reilly.
- Hansen, V. L. (1999). *Fundamental Concepts in Modern Analysis*. World Scientific Publishing, USA.
- Jules, D. (2019). *Learning Spark: Lightning-Fast Data Analytics*. USA O'Reilly.
- Kaminskyi, A., Nehrey, M., & Zomchak, L. (2021). Machine learning methods application for consumer banking. *SHS Web of Conferences*, 107, 12001.
- Klein, J. (2017). Santiago Ramón y Cajal, el hombre que dibujó los secretos del cerebro.

- Leo, M., Sharma, S., & Maddulety, K. (2019). Machine learning in banking risk management: A literature review. *Risks*, 7(1), 29.
- Rasmusson, J. (2010). The agile samurai: How agile masters deliver great software. *The Agile Samurai*, 1-264.
- Rim, P., & Liu, E. (2020). Optimizing the C4. 5 Decision Tree Algorithm using MSD-Splitting. *International Journal of Advanced Computer Science and Applications*, 11(10), 41-47.
- Rodrigo, J. A. (2017). *Máquinas de Vector Soporte (Support Vector Machines, SVMs)*. Consultado el 1 de julio de 2022, desde https://rpubs.com/Joaquin_AR/267926
- Sarkar, M. A. (2012). *Events identification using Box-Jenkins methodology with application to accelerated durability tests of ground vehicles* (Tesis de maestría).
- SAS, I. (2011). *Applied Analytics Using SAS*. NC, USA: SAS Institute.
- Schapire, R. E. (2013). *Explaining adaboost*. Springer.
- scikit-learn. (2022). *scikit-learn*. Consultado el 1 de marzo de 2022, desde https://scikit-learn.org/stable/auto_examples/preprocessing/plot_map_data_to_normal.html
- spark.apache.org. (2021). *spark.apache.org*. Consultado el 6 de julio de 2021, desde <https://spark.apache.org/docs/latest/ml-classification-regression.html#multilayer-perceptron-classifier>
- Taha, H. A. (2011). *Operations research: an introduction* (Vol. 790). Pearson/-Prentice Hall Upper Saddle River, NJ, USA.
- Weisberg, S. (2001). Yeo-Johnson power transformations. *Department of Applied Statistics, University of Minnesota*. Retrieved June, 1, 2003.

Apéndice A

Anexo de Códigos en R y Python

```
1 # Lenguaje R
2 # install package
3 install.packages("neuralnet")
4 # Crear datos de entrada
5 input1=c(20,10,30,20,80,30,20,10,10,30,40,60)
6 input2=c(90,20,40,50,50,80,20,10,10,30,40,60)
7 Resultado=c(1,0,0,0,1,1,0,0,0,0,1,1)
8 # Generar dataframe
9 df=data.frame(input1,input2,Resultado)
10 # Cargar libreria
11 require(neuralnet)
12 # Entrenar Red
13 nn=neuralnet(Resultado~input1+input2,data=df, hidden=c(7,4),act.fct
    = "logistic",linear.output = FALSE)
```

Código A.1: Código en R para generar red neuronal Fuente: Elaboración propia.

```
1 library(party, quietly=TRUE)
2 Arbolin_Propen1 <- ctree(
3   Y ~
4   MONTO_EXP +
5   CREDIS_EXP+
6   MoraTarj+
7   MoraCS+
8   Perdidas+
9   UNO+
10  DOS+
11  TRES+
12  CUATRO+
13  Liquidez_pyme+
14  Liquidez_total+
15  Bcom+
16  TDCN+
17  buro_antes+
18  buro_vs_salidas+)
```

```

19 Salidas_vs_liquidez+ TP_PERSONA
20 ,data=datos
21 ,control=ctree_control(maxdepth=5))

```

Código A.2: Código en R para generar árboles de decisión. Fuente: Elaboración propia.

```

1
2 modelo_svm <- svm(formula = y ~ X1 + X2, data = datos, kernel = "
   linear",
3                   cost = 10, scale = FALSE)
4
5 rango_X1 <- range(datos$X1)
6 rango_X2 <- range(datos$X2)
7
8 new_x1 <- seq(from = rango_X1[1], to = rango_X1[2], length = 75)
9 new_x2 <- seq(from = rango_X2[1], to = rango_X2[2], length = 75)
10 nuevos_puntos <- expand.grid(X1 = new_x1, X2 = new_x2)
11
12 predicciones <- predict(object = modelo_svm, newdata = nuevos_
   puntos)
13
14 colores <- data.frame(nuevos_puntos, y = predicciones)
15
16 beta <- drop(t(modelo_svm$coefs) %*% as.matrix(datos[,c("X1","X2")
   ])[modelo_svm$index,])
17 beta0 <- modelo_svm$rho
18
19
20 ggplot() +
21   geom_point(data = colores, aes(x = X1, y = X2, color = as.factor(
   y)),
22             size = 1) +
23   geom_point(data = datos, aes(x = X1, y = X2, color = as.factor(y)
   ),
24             size = 6) +
25   geom_point(data = datos[modelo_svm$index, ],
26             aes(x = X1, y = X2, color = as.factor(y)),
27             shape = 21, colour = "black",
28             size = 6) +
29   theme_bw() +
30   theme(legend.position = "none")

```

Código A.3: Código en R para las gráficas de Máquina de Soporte Vectorial. Fuente: Elaboración propia con código base de (Rodrigo, 2017).

```

1
2 # Ejemplo de selección de parámetros con GridSearchCV para un
   random Forest y con validación cruzada
3
4 parameters = {
5   'max_depth': [5,10,20],
6   'min_samples_split': [10,20,60],
7   'min_samples_leaf': [10,20,60],
8   'max_features': ['sqrt', 'log2', 'None']
9
10 }

```

```

11 rf = RandomForestClassifier()
12 clf = GridSearchCV(rf, parameters, cv=10)
13 clf.fit(X_train, y_train)
14
15 # Ejemplo de selección de parámetros con GridSearchCV para una
16   máquina de soporte vectorial y con validación cruzada
17
18 parameters = {
19     'kernel': ('linear', 'poly', 'rbf', 'sigmoid'),
20     'C': [1, 10],
21     'degree': [2, 3, 4],
22     'gamma': ('scale', 'auto')
23 }
24
25 svc = svm.SVC()
26 clf = GridSearchCV(svc, parameters, cv=10)
27 clf.fit(X_train, y_train)
28
29 # Ejemplo de selección de parámetros con GridSearchCV para una
30   red neuronal y con validación cruzada
31
32 parameters = {
33     'hidden_layer_sizes': [(20, 15, 5), (40, 20, 10), (10, 5, 3)],
34     'solver': ('sgd', 'adam'),
35     'activation': ['relu', 'logistic', 'tanh'],
36     'learning_rate_init': [0.001, 0.01],
37     'max_iter': [300, 500]
38     , 'learning_rate': ('constant', 'invscaling', 'adaptive')
39 }
40
41 svc = MLPClassifier()
42 clf = GridSearchCV(svc, parameters, cv=10)
43 clf.fit(X_train, y_train)
44 #clf.cv_results_
45
46 df = pd.DataFrame(clf.cv_results_)
47 df2=df[df.rank_test_score==1].params

```

Código A.4: Código en Python para evaluar diferentes parámetros de los modelos. Fuente: Elaboración propia con código base de GridSearchCV.

Apéndice B

Anexo de Definiciones Matemáticas

B.1. Algoritmo Ada Boost

El algoritmo busca un resultado ponderado de un vector de entrada $X_j = \langle x_1, \dots, x_M \rangle$ con M modelos $(G_1, \dots, G_M)$ con la siguiente función:

$$AB(X_j) = \varphi \left(\sum_{i=1}^M \alpha_i G_i(X_j) \right) \quad (\text{B.1})$$

en donde $j = 1, 2, \dots, n$ (n es el tamaño de la muestra de aprendizaje). φ es la función 1.7.

A continuación se muestran los pasos para pronosticar una clase con base en un modelo de Ada Boost que se ha mencionado.

Definición B.1.1 (Algoritmo Ada Boost) *Con base en un conjunto de datos de entrenamiento denominado A de tamaño n con M variables. En donde y_i es la respuesta categórica (1 o 0) de cada registro i para cada registro de entrada $X_i = \langle x_1, \dots, x_M \rangle$. Mientras $k < T$ (T un número natural definido por el usuario) seguir los siguientes pasos:*

1. Para $k = 0$ asignar $A_k \leftarrow A$ y asignar un peso $w_i = \frac{1}{n}$ para cada observación i de un conjunto de entrenamiento de tamaño n .
2. Dada una muestra A_k , generar un árbol de decisión G_m para cada variable m del modelo $(G_1, \dots, G_M)$ con la restricción de solo tener un nivel de nodos¹.

¹Por un nivel de nodos, se entenderá que el árbol solo utiliza una variable de aprendizaje y solo se podrá encontrar una división (dos clases) obteniendo un nodo y una rama para cada clase.

3. Calcular el índice de Gini para cada árbol G_m y seleccionar al que tenga menor índice de Gini y denominarlo G_m^1 .

4. Calcular el error de este modelo para cada entrada:

$$E_m = \frac{\sum_{i=1}^n w_i I(y_i \neq G_m^1(X_i))}{\sum_{i=1}^n w_i} \quad (\text{B.2})$$

5. Aplicar logaritmo al error para que no se disparen sus valores:

$$\alpha_m = \frac{1}{2} \log \left(\frac{(1 - E_m)}{E_m} \right) \quad (\text{B.3})$$

6. Asignar el nuevo valor de las ponderaciones:

$$w'_i = w_i \exp(\alpha I(y_i \neq G_m^1(X_i))) \quad (\text{B.4})$$

7. Normalizar las ponderaciones:

$$w_i = \frac{w'_i}{\sum_{i=1}^n w'_i} \quad (\text{B.5})$$

8. Seleccionar una nueva muestra A de tamaño n con repetición con muestreo aleatorio ponderado por los pesos $(w_1, \dots, w_n)$. Asignar $A_{k+1} \leftarrow A$.

9. Actualizar k ($k = k + 1$) e ir a paso 2.

La la fórmula B.3, es propuesta por los autores del algoritmo y se deriva de despejar α_m del gradiente del error exponencial:

$$\frac{\partial E}{\partial \alpha_m} = 0 \quad (\text{B.6})$$

en donde si

$$E = \sum_{y_j = G_m(X_j)} w'_j e^{-\alpha_m} + \sum_{y_j \neq G_m(X_j)} w'_j e^{\alpha_m} \quad (\text{B.7})$$

en donde $w'_j = e^{-y_j \hat{y}_j}$ para $m > 1$ y 1 otro caso. Entonces, se tiene que:

$$\alpha_m = \frac{1}{2} \log \left(\frac{(1 - E_m)}{E_m} \right) \quad (\text{B.8})$$

esto es, el resultado $AB(X_j)$ será la ponderación de un conjunto de árboles de decisión con base en la aportación del error mínimo de cada modelo.

Existen diferentes versiones de este algoritmo, en este caso se trata de la versión presentada en (Schapire, 2013). Existen dos aspectos clave en el algoritmo, el primero es aplicar logaritmo para hacer manejable el error pues de esta forma, el error no diverge cuando es muy grande y esto genera un apropiado aprendizaje del modelo. Por otra parte, en el paso seis, al aplicar la función exponencial se evita que los valores del error se disparen y el peso w_i pueda mantenerse menor a uno lo que facilita el posterior cálculo y que se disparen los pesos en la muestra.

B.2. Algoritmos de Optimización

Definición B.2.1 (Algoritmo de Gradiente Descendiente) *Dados los parámetros $w^{(t)}$ y la función de pérdida $L^{(t)}$, en donde t indexa el número de iteración (iniciando en 0) y sea m el número de iteraciones seleccionadas previamente, el nuevo parámetro $w^{(t+1)}$ se obtendrá al seguir los siguientes pasos:*

1. Establecer aleatoriamente los parámetros $w^{(t)} = w_{t1}, \dots, w_{tn}$
2. Para cada iteración calcular el error con una función de costo y su gradiente de cada parámetro (derivada parcial de la función de costo respecto a cada parámetro w_{ti} para $i = 1, \dots, n$):

$$E_t = L^{(t)}(w_{t1}, \dots, w_{tn}) \quad (\text{B.9})$$

$$\nabla E_{ti} = \frac{\partial L^{(t)}(w_{t1}, \dots, w_{tn})}{\partial w_{ti}} \quad \text{para } t = 1, \dots, m \quad (\text{B.10})$$

3. Obtener el nuevo valor de los parámetros:

$$w_{(t+1)i} = w_{(t)i} - \nabla E_{ti} \eta \quad (\text{B.11})$$

en donde η es la tasa de aprendizaje, puede ser variable o constante con respecto a t .

4. Detener si el valor de la función de costo no disminuye lo deseado.

Definición B.2.2 (Algoritmo ADAM) *Dados los parámetros $w^{(t)} = \langle w_{t1}, \dots, w_{tn} \rangle$ y la función de pérdida $L^{(t)}$, en donde t indexa el número de iteración (iniciando en 0), el nuevo parámetro $w^{(t+1)}$ será estimado con las siguientes ecuaciones:*

$$\nabla E_{ti} = \frac{\partial L^{(t)}(w_{t1}, \dots, w_{tn})}{\partial w_{ti}} \quad (\text{B.12})$$

$$m_{wi}^{(t+1)} = \beta_1 m_{wi}^{(t)} + (1 - \beta_1) \nabla E_{ti} \quad (\text{B.13})$$

$$v_{wi}^{(t+1)} = \beta_2 v_{wi}^{(t)} + (1 - \beta_2) (\nabla E_{ti})^2 \quad (\text{B.14})$$

$$\hat{m}_{wi} = \frac{m_{wi}^{(t+1)}}{1 - \beta_1^t} \quad (\text{B.15})$$

$$\hat{v}_{wi} = \frac{v_{wi}^{(t+1)}}{1 - \beta_2^t} \quad (\text{B.16})$$

$$w_i^{(t+1)} = w_i^{(t)} - \eta \frac{\hat{m}_{wi}}{\sqrt{\hat{v}_{wi}} + \epsilon} \quad (\text{B.17})$$

en donde ϵ es un escalar pequeño (e.g. 10^{-8}) usado para prevenir la división por 0. Los parámetros β_1 y β_2 son las tasas de aprendizaje para el primer y segundo momento de los gradientes respectivamente. η es un valor arbitrario (usualmente cercano a cero) que es otra tasa de aprendizaje. Los valores iniciales $m_{wi}^{(0)}$ y $v_{wi}^{(0)}$ son cero (Fuentes, 2020).

B.3. Sobre Redes Neuronales

En un modelo multi capa podrá apreciarse mejor el funcionamiento de una red neuronal, para ello, es preciso definir la función $g(\cdot)$ de flujo de información:

Definición B.3.1 (Función $g(\cdot)$ de flujo de información) *Un modelo de red neuronal puede ser descrito como el paso de información de una capa a otra en un conjunto de capas con características predefinidas. Sea k una capa con m neuronas a la cuál se le ha asignado la función de activación φ_k , y sea $(k-1)$ la capa predecesora que cuenta con n neuronas y que le proporcionará información a la capa k , la función g_k de traspaso de información de una capa a otra se define como:*

$$g_k(g_{k-1}) = g_k(g_{1(k-1)}, \dots, g_{n(k-1)}) = (\varphi_k(\sum_{i=1}^n w_{i1}g_{i(k-1)} + \theta_1), \dots, \varphi_k(\sum_{i=1}^n w_{im}g_{i(k-1)} + \theta_m)) \quad (\text{B.18})$$

en donde g_k es una función de $\mathbb{R}^n \rightarrow \mathbb{R}^m$, ya que $g_k(g_{1(k-1)}, \dots, g_{n(k-1)}) = (g_{1(k)}, \dots, g_{m(k)})$, θ_j es el bias de la neurona j en la capa k y $g_0 = (x_1, \dots, x_l)$ en donde $(x_1, \dots, x_l)$ es la información de entrada (l columnas de entrada de información) del modelo.

Para que una Red Neuronal pueda aprender se puede utilizar el algoritmo de Back Propagation como se mencionó anteriormente y consiste en distribuir el error entre cada uno de los parámetros proporcionalmente desde la última capa a la primera. Para calcular los nuevos parámetros, se puede utilizar alguno de los algoritmos de optimización mencionados (GD o ADAM), con la peculiaridad de que el gradiente se le calcula a una función compuesta $L(g_k, y)$ para una red de k capas, es por ello que se hace uso de la regla de la cadena:

$$\frac{\partial p(q)}{\partial a} = \frac{\partial p}{\partial q} \frac{\partial q}{\partial a} \quad (\text{B.19})$$

en donde p y q son funciones y a es una variable de la función q . Con esta fórmula será posible calcular el gradiente $\nabla E_{t(z_{ij})}$:

$$\nabla E_{t(z_{ij})} = \frac{\partial L(g_k \circ g_{k-1} \circ \dots \circ g_z \circ \dots \circ g_0, y)}{\partial w_{(z_{ij})}} \quad (\text{B.20})$$

para el parámetro $w_{(zij)}$ que representa el peso sináptico que conecta a la neurona j en la capa z con la neurona i que se encuentra en la capa $z-1$. Usando la regla de la cadena recursivamente se pueden calcular las derivadas parciales para cada parámetro de la siguiente forma:

$$\nabla E_{t(zij)} = \frac{\partial L}{\partial g_k} \frac{\partial g_k}{\partial g_{k-1}} \dots \frac{\partial g_{z+1}}{\partial g_z} \frac{\partial g_z}{\partial w_{(zij)}} \quad (\text{B.21})$$

Se puede notar que si se selecciona la función de activación ReLU, su derivada será uno o cero dependiendo del valor de la variable independiente facilitando el cálculo del gradiente.

Podemos observar, que para el método de aprendizaje de Back Propagation se ha utilizado una función de costo, un algoritmo de optimización basado en Gradiente Descendiente con una tasa de aprendizaje. Múltiples experimentos cambiando estos hiperparámetros podrán aportar mejoras en los resultados.

B.4. Máquinas de Soporte Vectorial

Sea el plano de la clase C_1 definido como:

$$\vec{w} \cdot \vec{x}_+ - b = 1 \quad (\text{B.22})$$

y el plano de la clase C_2 :

$$\vec{w} \cdot \vec{x}_- - b = -1 \quad (\text{B.23})$$

en donde $\vec{w}$ es un vector de valores $\langle w_1, \dots, w_p \rangle$ de los pesos para cada x_i en donde $i \in \{1, \dots, p\}$ para un problema de p variables de entrada. El escalar b orienta a los planos en una apropiada región del espacio y se buscará su valor en un problema de optimización. El vector $\vec{x}$ tiene subíndice $+$ o $-$ para denotar que pertenece a los puntos que resuelven las ecuaciones B.22 y B.23 respectivamente. Estos dos planos pueden ser representados por la siguiente función:

$$y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1 \quad (\text{B.24})$$

en donde y_i es igual a uno cuando $\vec{x}_i \in C_1$ y a menos uno cuando $\vec{x}_i \in C_2$ para cada $\vec{x}_1$ elemento de una muestra de tamaño n . Los planos B.22 y B.23 son paralelos entre sí, debido a que cuentan con los mismos valores $\vec{w}$ y b . Si se toma un vector $\vec{x}_+$ que pertenece a B.22 y otro vector $\vec{x}_-$ que pertenece a B.23, la distancia entre ambos vectores será:

$$\vec{x}_+ - \vec{x}_- \quad (\text{B.25})$$

Si se encuentran dos vectores con el mismo ángulo desde el origen, entonces se podrá saber la distancia entre ambos planos, que será la distancia entre ambos vectores. Para encontrar dos vectores con estas características, se puede encontrar la proyección de la resta B.25 con respecto a un plano definido por $\vec{w}$ y que se sabe que es paralelo a B.22 y B.23:

$$\text{Proy}(\vec{x}_+ - \vec{x}_-, \vec{w}) = \frac{(\vec{x}_+ - \vec{x}_-) \cdot \vec{w}}{\|\vec{w}\|} \quad (\text{B.26})$$

$$= \frac{\vec{x}_+ \cdot \vec{w} - \vec{x}_- \cdot \vec{w}}{\|\vec{w}\|} \quad (\text{B.27})$$

$$= \frac{1 + b + 1 - b}{\|\vec{w}\|} \quad (\text{B.28})$$

$$= \frac{2}{\|\vec{w}\|} \quad (\text{B.29})$$

y como se busca encontrar el plano que maximice la distancia entre los dos planos B.22 y B.23, el problema de optimización quedará de la siguiente forma:

$$\text{Máx}_{\vec{w}, b} \left\{ \frac{2}{\|\vec{w}\|} \right\} \quad (\text{B.30})$$

Sujeto a:

$$\vec{w} \cdot \vec{x} - b \geq 1 \quad \forall \vec{x} \in C_1 \quad (\text{B.31})$$

$$\vec{w} \cdot \vec{x} - b \leq -1 \quad \forall \vec{x} \in C_2 \quad (\text{B.32})$$

Maximizar $\frac{2}{\|\vec{w}\|}$ equivale a minimizar $\|\vec{w}\|$ y esto a su vez es equivalente a minimizar $\frac{1}{2}\|\vec{w}\|^2$ la cual se usará en lo sucesivo por ser una expresión matemáticamente conveniente para la determinación del gradiente.

Los multiplicadores de Lagrange (Hansen, 1999) que cambian la función de optimización, permiten convertir el problema de optimización en uno más sencillo en el caso de que sea más fácil (una ecuación más sencilla) encontrar un óptimo local en las restricciones. Cuando se aplican los multiplicadores de Lagrange a un problema de optimización, se obtiene uno segundo llamado dual y al primer problema que se tenía se le denomina primal. Aplicando los multiplicadores de Lagrange al problema de optimización primal que se estaba tratando, se obtendrá el siguiente problema dual:

$$L(\vec{w}, b, \alpha) = \frac{1}{2}\|\vec{w}\|^2 - \sum_{i=1}^n \alpha [y_i(\vec{w} \cdot \vec{x}_i - b) - 1] \quad (\text{B.33})$$

para encontrar un máximo de la función $L(\vec{w}, b, \alpha)$ se requiere calcular los gradientes:

$$\frac{\partial L(\vec{w}, b, \alpha)}{\partial \vec{w}} = \vec{w} - \sum_{i=1}^n \alpha y_i \vec{x}_i = 0 \quad (\text{B.34})$$

y

$$\frac{\partial L(\vec{w}, b, \alpha)}{\partial b} = - \sum_{i=1}^n \alpha y_i = 0 \quad (\text{B.35})$$

De la ecuación B.34 se puede despejar $\vec{w}$ y sustituir en la ecuación B.33 obteniendo la siguiente ecuación:

$$\begin{aligned}
L(\vec{w}, b, \alpha) = & \\
& \frac{1}{2} \left(\sum_{i=1}^n \alpha_i y_i \vec{x}_i \right) \cdot \left(\sum_{j=1}^n \alpha_j y_j \vec{x}_j \right) \\
& - \left(\sum_{i=1}^n \alpha_i y_i \vec{x}_i \right) \cdot \left(\sum_{j=1}^n \alpha_j y_j \vec{x}_j \right) \\
& - b \sum_{i=1}^n \alpha_i y_i \vec{x}_i + \sum_{i=1}^n \alpha_i
\end{aligned} \tag{B.36}$$

La ecuación B.36 puede resumirse en la siguiente:

$$L(\vec{w}, b, \alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j \tag{B.37}$$

Se puede observar que el problema de optimización depende de $\vec{x}_i \cdot \vec{x}_j$ por lo que la elección de estos vectores deberá ser la apropiada. Para ello, se ideó el llamado Truco del Kernel. Se llama como Truco del Kernel o Truco del Núcleo cuando se aplica una de estas transformaciones a los datos lo cual permite mejorar la interpretación de estos en varias dimensiones debido a que crece su relación cuando se les aplican algunos tipos de transformaciones. Por ejemplo, $y = x$ implica que existe una correlación lineal de uno entre y y x , si se aplica la transformación $y = x^2$, se tendrá una relación no lineal, creciente a medida que x crece a diferencia de la relación (constante) de la ecuación inicial. Cuando se aplica una transformación (función) a un conjunto en $\mathbb{R}^n$ que resulta en un conjunto en $\mathbb{R}^m$, se dice que esta función es un Kernel que mapea al espacio $\mathbb{R}^n$ en el espacio $\mathbb{R}^m$. Dependiendo del Kernel elegido, se obtendrán resultados diferentes, es por ello que se recomienda realizar diferentes experimentos con diferentes núcleos. Dos de ellos se definen a continuación:

1. Kernel Polinomial

$$K(\vec{x}_i, \vec{x}_j) = (\vec{x}_i^T \cdot \vec{x}_j - \theta)^d \tag{B.38}$$

en donde θ y d son hiperparámetros que se pueden obtener a través de múltiples experimentos.

2. Kernel Gaussiano

$$K(\vec{x}_i, \vec{x}_j) = \exp \frac{1}{2\sigma^2} \|\vec{x}_i - \vec{x}_j\|^2 \tag{B.39}$$

en donde σ es un hiperparámetro que puede obtenerse por medio de múltiples experimentos o con el cálculo de la desviación estándar de los datos.