Predicción de incumplimiento de pagos de crédito en una entidad financiera utilizando chats de servicio al cliente

Proyecto de Grado- Maestría en Ciencia de Datos y Analítica – Universidad EAFIT

Autor: Javier Patiño Serna – jpatinos@eafit.edu.co
Director: Juan David Martínez Vargas - jdmartinev@eafit.edu.co-
Co-Directora: Paola Andrea Vallejo Correa - pvallej3@eafit.edu.co

Octubre 2023

Contenido

1 Resumen

Las entidades financieras enfrentan día a día el reto de identificar los clientes de alto riesgo que potencialmente pueden incumplir sus acuerdos de pago. De esta manera pueden buscar estrategias para optimizar la recuperación de cartera y que la recaudación no se vea afectada significativamente. La mayoría de las aproximaciones para resolver este problema se centran en analizar las características de los solicitantes antes de que les sea otorgado el crédito. En este trabajo se propuso una aproximación desde la minería de texto, a través de interacciones de servicio al cliente para predecir el no pago en la siguiente cuota del crédito. El modelo de procesamiento de lenguaje natural tomó como datos de entrada textos de las interacciones de la línea de servicio al cliente. Se implementaron tres algoritmos de aprendizaje profundo: *RandomForest*, Red Neuronal Artificial y un modelo pre-entrenado basado en BERT, con el objetivo de identificar los clientes que no pagaron su crédito después de la interacción. Los resultados arrojaron que el modelo que obtuvo un mejor desempeño fue el de red neuronal, logrando un valor de *recall* del 64% para la clase “NO PAGÓ”, de esta manera el modelo logró clasificar correctamente el 64% de los registros que fueron etiquetados con el incumplimiento de pago.

Keywords: Procesamiento de lenguaje natural, entidad financiera, aprendizaje profundo, clasificación, minería de texto.

2 Introducción

Los equipos de análisis de riesgos en las entidades financieras cumplen un rol fundamental en la sostenibilidad de estas compañías. Son los encargados de determinar qué personas son idóneas para prestarles dinero, y cuales probablemente van a incumplir sus compromisos financieros. Típicamente, la manera de abordar este problema ha sido mediante la evaluación de un conjunto de características que se recopilan antes de otorgar el préstamo, tales como los ingresos del solicitante, su nivel de estudios, si presenta o no un trabajo estable,

entre otras. En este trabajo, se propone abordar el problema después de que el crédito ha sido aprobado, para predecir el no pago de una futura cuota del crédito, mediante la minería de texto y haciendo uso de algoritmos de aprendizaje profundo. De esta manera, después de una interacción en la línea de servicio al cliente, esta podría ser evaluada para determinar si el cliente al otro lado de la línea podría llegar a incumplir con el pago de su cuota. Esta aproximación permite obtener información relevante de los clientes que no se toma en cuenta en la recopilación de características como por ejemplo sus actitudes, miedos o preocupaciones. Otra ventaja de este enfoque es que la información obtenida de los textos corresponde a información actualizada en tiempo real que puede revelar pistas importantes sobre la salud financiera y la estabilidad económica de los usuarios.

Al abordar la predicción de incumplimiento de pagos desde la perspectiva de los

chats de servicio al cliente, se busca no solo complementar, sino también enriquecer los métodos tradicionales de evaluación de riesgos crediticios. La comunicación escrita proporciona un vasto conjunto de datos no estructurados que, mediante técnicas avanzadas de procesamiento de lenguaje natural y análisis de sentimientos, pueden ser transformados en indicadores predictivos relevantes.

2.1 Planteamiento del Problema

La gestión adecuada de riesgos es crucial para garantizar la rentabilidad y la sostenibilidad de las instituciones crediticias. Uno de los riesgos más relevantes es el no pago de las cuotas de crédito, que implica la falta de cumplimiento de los compromisos de pago por parte de los clientes. La detección temprana de patrones o indicios de posibles incumplimientos podría ayudar a las instituciones a tomar medidas preventivas, minimizando las pérdidas y optimizando la gestión de la cartera de créditos (Poitras, 2002).

El problema de predecir si un cliente va a entrar en *default* (incumplimiento de pago) ha sido ampliamente estudiado, siendo los métodos más populares para abordarlo, las redes neuronales *feedforward*, convolucionales y recurrentes (Li, 2022) así como también los árboles de decisión y los modelos de ensamble (Zhu et al., 2023).

Actualmente, las entidades financieras tienen un alto flujo en sus líneas de atención al cliente a través de sus diferentes medios como son chat, llamada o correo electrónico, es decir que constantemente se están generando grandes volúmenes de textos que proporcionan información sobre los hábitos de pago de las personas. Esta información pocas veces es aprovechada en su totalidad, desperdiciando así interacciones que pueden otorgar indicios sobre el cumplimiento o incumplimiento de una cuota de pago.

Es así como el procesamiento de lenguaje natural (NLP por sus siglas en inglés) se presenta como una solución tecnológica para analizar y extraer información útil de las interacciones de servicio al cliente, entre ellas la posibilidad del incumplimiento en los acuerdos de pago por parte de un cliente (Piris & Gay, 2021). Sin embargo, se presentan varios retos en esta implementación, tales como realizar el pre-procesamiento de los textos, la representación de los textos, la selección del modelo y el ajuste de sus respectivos parámetros para obtener una predicción acertada.

Las aproximaciones desde la minería de texto y el procesamiento de lenguaje natural de varios autores se han enfocado en los textos generados por los usuarios en las aplicaciones a los créditos (Stevenson et al., 2021) (Kriebel & Stitz, 2022).

El enfoque de este trabajo parte principalmente de la perspectiva de generar valor sobre la información que se genera a partir de las observaciones de las líneas de servicio al cliente, y de esta manera predecir el incumplimiento en el pago por parte de un usuario.

2.2 Justificación

Para las entidades financieras es importante desarrollar estrategias para la gestión del riesgo, de esta manera se puede garantizar la rentabilidad y sostenibilidad de la compañía (Arora et al., 2022). Estas estrategias incluyen, entre otras cosas, el análisis de riesgo crediticio de sus clientes, es decir, cuáles serán más propensos a incumplir sus acuerdos de pago. Identificar con anticipación clientes que potencialmente pueden incumplir sus pagos, proporciona un margen de acción para que la empresa tome medidas y que la recuperación de cartera no se vea impactada negativamente. Entre las medidas que se pueden tomar tenemos, por ejemplo, establecer contratos más robustos con los clientes de alto riesgo con clausulas claras para los acuerdos de pago, gestionar seguros de crédito ante la insolvencia e incumplimiento de pagos y mejorar los canales de comunicación con los clientes para que estos dispongan de instrucciones correctas y oportunas sobre cómo realizar sus pagos (Kumar, 2017).

El desarrollo de este proyecto busca robustecer la gestión de riesgo con una herramienta para determinar la posibilidad de incumplimiento en los pagos, aprovechando para ello la información generada durante las interacciones de servicio al cliente. Específicamente busca desarrollar una manera de extraer información de los textos generados en estas interacciones para que alimenten posteriormente a modelos de aprendizaje profundo que puedan realizar una adecuada clasificación.

2.3 Objetivos

2.3.1 Objetivo General

Desarrollar un modelo de analítica de textos que permita predecir el incumplimiento de pago de un crédito, utilizando los datos de las interacciones de servicio al cliente en una entidad financiera, mediante la aplicación de modelos de aprendizaje profundo.

2.3.2 Objetivos Específicos

1 Desarrollar un modelo de procesamiento de lenguaje natural utilizando aprendizaje profundo que permita predecir el no pago del crédito utilizando como entrada la información de las interacciones en las líneas de servicio al cliente.

2 Evaluar el desempeño del modelo con datos reales de interacciones de servicio al cliente.

3 Marco Teórico

El marco teórico se dividirá en cuatro secciones para profundizar en los conceptos claves que conformarán el desarrollo de este proyecto. La primera sección proporcionará una visión general de la minería de texto y detallará los pasos necesarios para preparar los textos antes de utilizarlos en modelos de aprendizaje. La segunda sección presentará la arquitectura *transformer* y los modelos más utilizados basados en esta arquitectura para realizar tareas de clasificación. Por último, la tercera y cuarta sección abarcarán las generalidades de los bosques aleatorios y las redes neuronales respectivamente.

3.1 Minería de texto y procesamiento de lenguaje natural

La analítica de textos, también conocida como minería de texto, es el proceso mediante el cual se extrae información valiosa a partir de información escrita (Chen & Liu, 2008). Por otro lado, el procesamiento de lenguaje natural (NLP por sus siglas en inglés) es el campo de las ciencias de la computación que estudia la manera en que las computadoras entienden el lenguaje de los seres humanos (Stent & Bangalore, 2014). Para efectos de este trabajo, tanto la analítica de textos como el procesamiento de lenguaje natural son una parte fundamental dado que la información disponible consiste en textos de una línea de servicio al cliente, de la cual se busca extraer información valiosa, mediante el entendimiento de los textos por parte de modelos de aprendizaje profundo.

Los mensajes de texto pertenecen a la categoría de datos no estructurados, existen técnicas de preprocesamiento que transforman el texto en datos semi estructurados. Una vez el texto es llevado a forma semi estructurada es posible aplicar técnicas analíticas de clasificación, agrupamiento y predicción. La manera de convertir el texto en información semi-estructurada es a través de un pre-procesamiento de la información. Existen diferentes etapas en el pre-procesamiento de texto, entre las más comunes podemos encontrar la remoción de *stop words*, la tokenización, la lematización y el *stemming* (Haddi et al., 2013). Adicionalmente, para poder realizar operaciones analíticas con las palabras de un texto se debe encontrar una representación numérica de estas. Para lograr este objetivo tenemos a nuestra disposición la representación de los textos mediante *embeddings*. Los *embeddings* son representaciones vectoriales de palabras o textos en un espacio multidimensional continuo (Hassanien et al., 2022). En el contexto del NLP, los *embeddings* se utilizan para capturar la semántica y contexto de las palabras y frases, así como la similaridad de unas frases con otras (Zhu, 2021). Estos modelos han revolucionado las tareas de NLP, permitiendo una representación más eficiente y rica del lenguaje. Existen diferentes técnicas para crear *embeddings*, como Word2Vec, GloVe, FastText, etc. Estos modelos utilizan algoritmos como Skip-gram, Continuous Bag of Words (CBOW) o matrices de co-ocurrencia para aprender las representaciones. (Jatnika et al., 2019).

Adicionalmente, compañías como Open AI han desarrollado modelos propios para

extraer *embeddings* de textos, estos modelos pueden ser consumidos mediante una API y ser fácilmente agregados a un *pipeline* de desarrollo.

3.2 Arquitectura Transformer

En la actualidad, la principal arquitectura para el desarrollo de modelos de procesamiento de lenguaje natural es la arquitectura *Transformer* (Vaswani et al., 2017). Los modelos basados en *Transformers* tienen sus fundamentos en dos pilares del aprendizaje profundo: i) pre-entrenamiento y ii) atención. El pre-entrenamiento intensivo no supervisado, permite formar vectores de palabras y oraciones bastante robustos, y los mecanismos de atención permiten encontrar relaciones entre las diferentes palabras, no necesariamente de manera secuencial, esto mejora considerablemente la capacidad de capturar relaciones a largo plazo (Casola et al., 2022).

En la arquitectura *Transformer*, la parte más importante es el mecanismo de atención, el cual aprende cuales son las palabras más relevantes y se enfoca en estas para tener los aspectos más específicos del texto. En la ecuación 1 se puede observar la expresión formal para el mecanismo de atención. Esta fórmula nos muestra que el mecanismo de atención computa la relevancia de una secuencia de entrada (Key) para una salida (Query) esta importancia se calcula multiplicando una puntuación aprendida dinámicamente por cada elemento de la secuencia de entrada (Value). Q, K y V son las matrices del Key, Query y Value respectivamente y dk es la dimensión del vector Key y su respectiva Query (Casola et al., 2022).

$$Attention\ (Q, K, V) = softmax\ (\frac{QK^T}{\sqrt{d_k}})\ V \quad (1)$$

Adicionalmente, la arquitectura *Transformer* utiliza un codificador (*encoder*) y un decodificador (*decoder*). El *encoder* es un conjunto de N capas compuestas por un mecanismo de atención y una subcapa de alimentación hacia adelante (*feed-forward*). El mecanismo de atención es utilizado para calcular una representación del *input* de la siguiente capa sin utilizar recurrencia ni convolución. El decodificador tiene una arquitectura similar, con una subcapa de atención adicional sobre la salida del codificador. En la Figura 1 tomada de (Vaswani et al., 2017) se puede observar un esquema gráfico de la arquitectura *Transformer*.

Observamos en la figura al lado izquierdo el esquema de la arquitectura, con su codificador compuesto por una capa de atención, en este caso coloreada de anaranjado, y una capa *feedforward*, de color celeste, seguido del decodificador el cual tiene dos capas de atención y solo una capa *feedforward*, con los mismos colores. En la siguiente figura se muestra el esquema de cómo funciona la capa deatención, el cual recibe como entradas las variables V, K y Q, y a través de la función softmax calcula la importancia de una secuencia.

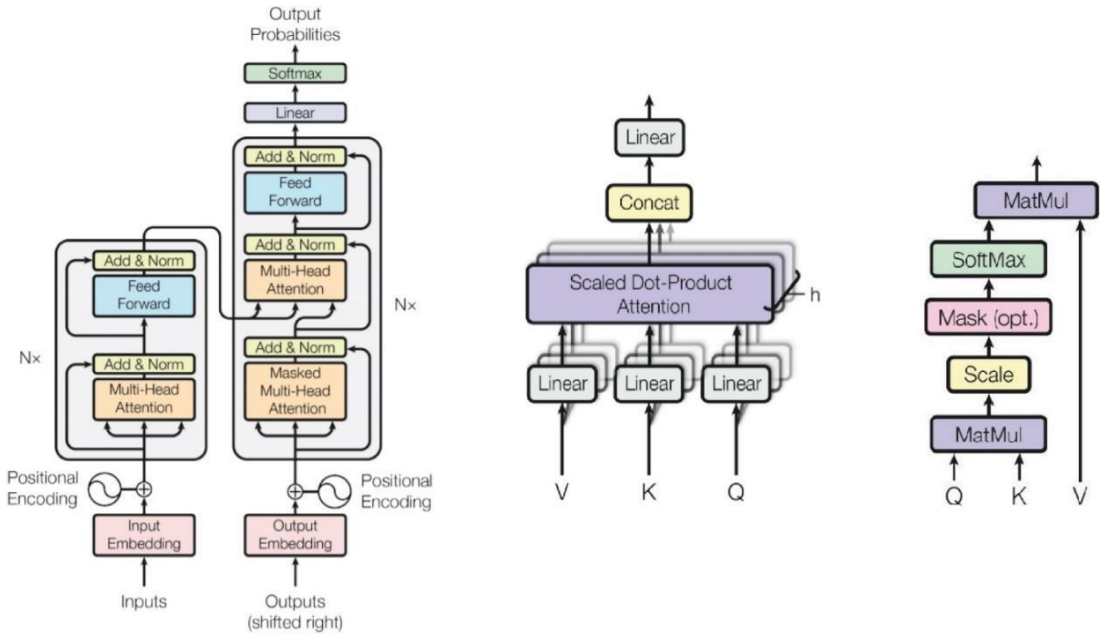

Figura 1. *Representación gráfica de la arquitectura Transformer*
(Vaswani et al., 2017)

La arquitectura *Transformer* se pre-entrena en grandes cantidades de texto sin etiquetar utilizando tareas de autodenominación o de predicción de palabras enmascaradas (Radford et al., 2019). Luego, se realiza un ajuste fino (*fine-tuning*) en tareas específicas utilizando datos etiquetados para adaptar el modelo a una tarea particular (Mosin et al., 2023). La arquitectura *Transformer* ha sido fundamental en el desarrollo de modelos de lenguaje pre-entrenados, entre los más utilizados tenemos BERT, su variante en español BERTO y GPT.

El modelo BERT (*Bidirectional Encoder Representations from transformers*), fue desarrollado y publicado por Google en 2018. Se pre-entrena utilizando dos modelos: modelo de lenguaje enmascarado (MLM) y predicción de la siguiente oración (NSP) (Devlin et al., 2019). MLM consiste en enmascarar u ocultar aleatoriamente un porcentaje de los tokens de la entrada del modelo, de manera que a la salida se presente una distribución de probabilidad para predecir cuales son los tokens ocultos (Liu, Y. *et al.* 2020). Por otro lado, NSP es una tarea de clasificación binaria cuyo objetivo es determinar la secuencialidad entre diferentes oraciones de un texto. BooksCorpus (Zhu et al., 2015) y la Wikipedia en inglés (16 GB en total) se utilizaron como corpus de capacitación para el modelo BERT. El modelo BERTO es similar al modelo BERT, con la diferencia de haber sido pre-entrenado en español.

Finalmente, el modelo GPT (*Generative Pre Training*) con sus diferentes variantes (GPT-2, GPT-3 y GPT-4) ha mostrado un muy buen desempeño en tareas de generación de texto. Su arquitectura de *Transformer* es la estándar y, a diferencia de BERT y BETO, su pre-entrenamiento se basa en un enfoque auto-regresivo. El modelo toma una secuencia de palabras como entrada y genera una distribución de

probabilidad condicional sobre el vocabulario para la siguiente palabra (Radford et al., 2018). Luego, la siguiente palabra se elige basándose en la distribución de probabilidad.

Estos modelos pre-entrenados se utilizan bajo un concepto llamado *Transfer Learning*, el cual consiste en tomar modelos existentes y realizar el ajuste fino (*fine tunning*) de sus parámetros para una tarea específica (Zhu, 2021). En nuestro caso esta tarea específica consiste en predecir el no pago del crédito con la información disponible en los chats de servicio al cliente.

3.3 Bosques aleatorios

Los bosques aleatorios (*Random Forests*) son un poderoso algoritmo de aprendizaje automático que se basa en la combinación de múltiples árboles de decisión para realizar predicciones. Este enfoque se basa en dos conceptos clave: ensamble y aleatorización. Los ensambles combinan las predicciones de varios modelos para mejorar la precisión y la estabilidad del modelo, mientras que la aleatorización implica la introducción de variabilidad durante el proceso de entrenamiento para evitar el sobreajuste (Breiman, 2001). En la Figura 2 se ilustra gráficamente el algoritmo. Podemos observar como a partir del *dataset* inicial se construyen diversos árboles de decisión, los cuales se conforman con muestras aleatorias del *dataset* inicial. En la última capa del árbol se determina la categoría asignada, ya sea por voto mayoritario o realizando una ponderación.

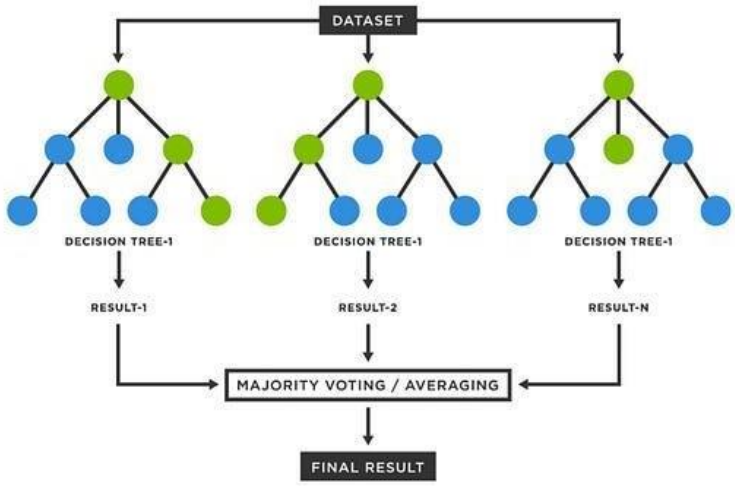

Figura 2. *Algoritmo de Bosque Aleatorio* (V, 2023)

El concepto central detrás de los bosques aleatorios es la construcción de múltiples árboles de decisión y luego combinar sus predicciones para obtener un resultado más preciso y estable. Cada árbol en el bosque se entrena en una submuestra aleatoria del conjunto de datos, y durante la predicción los resultados se promediano se ponderan (Liu et al., 2018). En nuestro caso existe un problema de clasificación binaria por lo que finalmente se tomaría el resultado mayoritario entre todos los

resultados obtenidos por cada árbol. Los bosques aleatorios tienen muchas ventajas, como su capacidad para manejar datos faltantes, su resistencia al sobreajuste y su capacidad para manejar conjuntos de datos grandes con muchas características, y adicionalmente han sido ampliamente utilizados en tareas de clasificación binaria de textos (Jalal et al., 2022). Sin embargo, también presentan desafíos en términos de interpretación y poder computacional.

3.4 Redes Neuronales

Por último, describiremos las redes neuronales artificiales, las cuales también son utilizadas para resolver problemas de clasificación binaria como el que nos compete en este trabajo. Las redes neuronales artificiales son modelos computacionales inspirados en el funcionamiento del cerebro humano. Estos modelos están compuestos por unidades llamadas neuronas, organizadas en capas, que procesan información y aprenden patrones complejos a través de algoritmos de optimización (Goodfellow et al., 2017). La aplicación de estas redes en diversos campos de la inteligencia artificial ha llevado a avances significativos en el reconocimiento de patrones, procesamiento del lenguaje natural y otras tareas complejas como el reconocimiento de imágenes (Hemanth et al., 2019). Estas estructuras permiten la extracción automática de características complejas a partir de datos sin requerir una ingeniería de características manual. Existen varias arquitecturas de redes neuronales dependiendo de las aplicaciones, para clasificar imágenes en la mayoría de los casos se utiliza la arquitectura de red neuronal convolucional, mientras que para tareas de clasificación de texto es más común el uso de redes neuronales recurrentes o pre-alimentadas (LeCun et al., 2015). En general, las redes neuronales se componen de una capa de entrada y varias capas ocultas, cada una con diferentes tamaños dependiendo de la aplicación.

Una parte muy importante en la construcción de una red neuronal son las funciones de activación ya que son cruciales en el proceso de entrenamiento. La función ReLU ha ganado popularidad debido a su capacidad para mitigar el problema del desvanecimiento del gradiente y acelerar el entrenamiento (Theodoridis, 2015). Además de ReLU, existen otras funciones de activación como sigmoide y tangente hiperbólica, cada una con sus propias ventajas en diferentes contextos. La función sigmoide limita la salida de las capas para los valores 0 y 1, esto puede ser interpretado como la probabilidad de que una neurona este activa o inactiva. Por otro lado, la función tangente hiperbólica limita la salida en un rango de -1 a 1 lo cual es útil cuando se tienen salidas numéricas negativas (Fávero et al., 2023). Sin embargo, el problema de estas dos funciones de activación es el desvanecimiento de gradiente, el cual consiste en que cuando se están trabajando datos de regiones extremas los gradientes tienden a ser pequeños, dificultando de esta manera el proceso de entrenamiento.

Algoritmos de optimización como el descenso del gradiente y sus variantes, como

el descenso del gradiente estocástico (SGD) y el descenso del gradiente con momento, son fundamentales en este proceso (Sever et al., 2023). Estos algoritmos determinan cómo la red aprende de los datos y converge hacia soluciones óptimas, durante el entrenamiento de la red se ajustan los pesos para minimizar una función de pérdida.

Para evitar el sobreajuste en las redes neuronales tenemos métodos como el *dropout*, el cual consiste en desactivar aleatoriamente neuronas durante el entrenamiento, de esta manera ayudan a prevenir el sobreajuste al mejorar la generalización del modelo (Sanjar et al., 2020). Además, técnicas como la regularización L1 y L2 también son empleadas para controlar la complejidad del modelo y evitar el sobreajuste.

4 Metodología

La metodología seleccionada para llevar a cabo este proyecto es CRISP-DM. En la Figura 3 se muestra un esquema de la metodología. La elección del modelo CRISP-DM para la estructuración de la metodología de este estudio se fundamenta en su reconocida aplicabilidad y versatilidad en proyectos de minería de datos, especialmente en contextos donde la exploración y análisis de datos son esenciales para la consecución de los objetivos de investigación (Schröer et al., 2021).

La elección de CRISP-DM como marco metodológico proporciona una estructura sistemática y ordenada para abordar los desafíos específicos de este estudio, asegurando un enfoque coherente y efectivo en todas las etapas del proceso de minería de datos aplicado a la predicción de incumplimientos de pagos en el sector financiero.

A continuación, se listan los pasos que se aplican para este proyecto (Schröer et al., 2021).

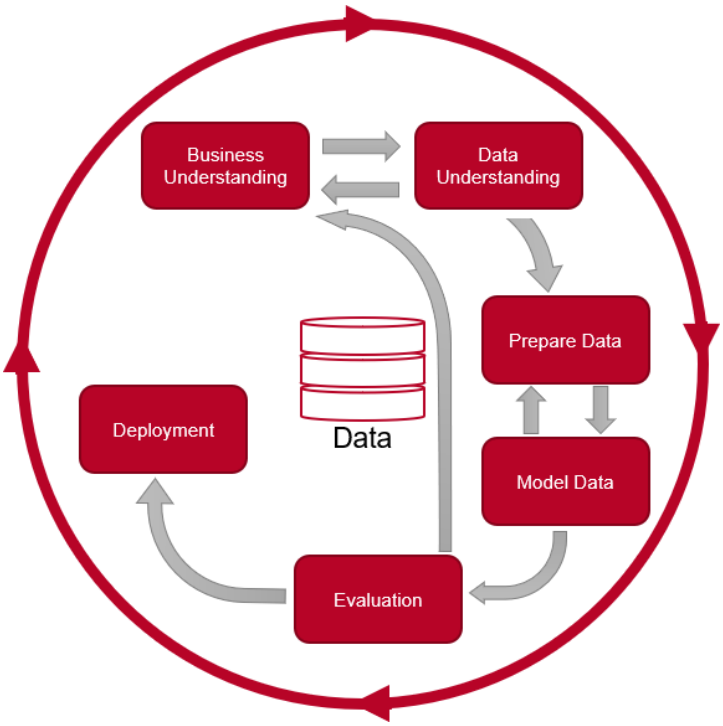

Figura 3. *Esquema Metodología CRIPS-DM*

4.1 Comprensión del negocio

Las entidades financieras para garantizar su sostenibilidad deben minimizar factores de riesgo, uno de los principales factores de riesgo para este tipo de empresas es el incumplimiento de los acuerdos de pago por parte de los clientes. De esta manera se pueden implementar diversas estrategias para mitigar el incumplimiento en los pagos. El objetivo general es predecir el no pago a partir de los chats de la línea de servicio al cliente.

Este proyecto puede ayudar a la compañía a mejorar la gestión del riesgo, al identificar posibles incumplimientos en los pagos de algunos clientes, mejorando de esta manera su rentabilidad y sostenibilidad financiera.

4.2 Entendimiento de los datos

Los datos disponibles son registros de texto de interacciones en la línea de servicio al cliente. Esta información se encuentra en una base de datos del software de gestión de relación con el cliente, los textos se pueden consumir por medio de una API. Se dispone de un histórico de datos desde enero del 2023 hasta julio de 2023. En total se tienen aproximadamente 75.000 registros de texto los cuales tienen en promedio una longitud de 500 caracteres.

4.3 Preparación de los datos

En esta etapa se realizará el pre-procesamiento de los textos, de manera que pasen de ser información no estructurada a ser información semi-estructurada. El pre-procesamiento de los datos incluye la estandarización del texto (convertirlo todo a minúsculas, eliminar caracteres especiales y signos de puntuación), la tokenización del texto, eliminación de *stopwords* y aplicación de las técnicas de *stemming* y lematización. Posteriormente, se realiza un proceso de representación numérica de las palabras mediante la técnica de extracción de *embbedings*. En este paso se utilizará la API de OpenAI para realizar las consultas y extraer los *embeddings* de las palabras. Adicionalmente se codificará la variable binaria de respuesta, se asignará el valor 0 a la categoría “NO PAGÓ” y el valor 1 a la categoría “PAGÓ”.

Los datos que se utilizarán en este proyecto corresponden a interacciones que contienen información sensible acerca de la situación financiera de los clientes de la compañía, por esta razón toda la información será anonimizada previamente. El proceso de anonimización incluye la eliminación de información personal identificable, el reemplazo de nombres y referencias, y la eliminación de identificadores únicos tales como documentos de identidad (Toom & Miller, 2018). Adicionalmente, únicamente el estudiante, el director y co-director de tesis tendrán acceso a los mismos y estos serán utilizados únicamente con fines académicos. Los resultados finales del proyecto no contendrán ningún tipo de información personal que permita la identificación de los titulares de los chats.

4.4 Implementación de Modelos

Durante esta etapa se realizará el entrenamiento de los modelos a utilizar. Las opciones que se implementarán serán las siguientes:

- Modelo de lenguaje pre-entrenado basado en la arquitectura *Transformer* (BERT en español)
- *Random Forest* para clasificación
- Red Neuronal para clasificación

4.5 Evaluación de Modelos

Dado que nuestro problema consiste en predecir el incumplimiento en el pago de un cliente, se utilizarán métricas clásicas para evaluar un problema de clasificación, entre las cuales tenemos: exactitud, sensibilidad y el *f-1 score* (Al-jabery et al., 2020). La exactitud se define como la proporción de predicciones correctas realizadas por el modelo con respecto al total de predicciones. Se calcula dividiendo el número de predicciones correctas entre el número total de predicciones. La sensibilidad mide la proporción de instancias positivas que el modelo es capaz de

identificar. Se calcula dividiendo el número de verdaderos positivos (TP) entre la suma de los verdaderos positivos y los falsos negativos (FN) (Fränti & Mariescu-Istodor, 2023). Finalmente, *el f1 score* es una medida que combina la precisión y la exhaustividad en un solo valor. Es útil cuando se desea tener un equilibrio entre ambas métricas. Se calcula como el promedio armónico entre la precisión y la exhaustividad.

Hasta acá llega nuestra implementación de la metodología CRISP-DM, este proyecto no abarcará la sección de despliegue de los modelos. A continuación se listan los productos esperados al finalizar el proyecto.

4.6 Productos esperados:

- Código fuente del modelo implementado para la resolución del problema
- Reporte escrito del desarrollo del proyecto.

5 Presentación de los Resultados

En esta sección inicialmente realizaremos la descripción del conjunto de datos con el que trabajamos. Posteriormente indicaremos el pre-procesamiento que se llevó a cabo junto con la implementación de los modelos y los resultados obtenidos. Para finalizar, se discutirán los resultados obtenidos en el paso previo.

5.1 Descripción del dataset

El *dataset* dispuesto para este trabajo consiste de seis columnas y aproximadamente 75.000 registros, nuestras columnas de interés son “Observación Gestión” la cual contiene el texto que refleja la interacción de la línea de servicio al cliente y “Pago” que simplemente es la etiqueta que identifica el cumplimiento o incumplimiento del pago de la cuota por parte del cliente después de la interacción. Este etiquetado se realizó cruzando con otra base de datos mediante el número de requerimiento, el cual en nuestro *dataset* se llama “Obligacion”. La temporalidad de los datos va desde enero del 2023 hasta julio del 2023.

5.2 Pre-Procesamiento

Inicialmente se realizó un pipeline de pre-procesamiento de texto que incluyó la remoción de los caracteres especiales utilizando expresiones regulares, en este paso se eliminan todos los caracteres especiales y los números y adicionalmente se estandariza todo el texto en minúscula. Posteriormente, se utilizó la biblioteca spacy para tokenizar los textos sin caracteres especiales, de esta manera se dividió el texto en entidades compactas para su posterior uso por parte de los modelos. Para terminar, los tokens se unieron en una nueva columna del dataframe que obtuvo el nombre de texto procesado. En la Figura 4 se puede observar un diagrama de flujo con el *pipeline* de pre-procesamiento.

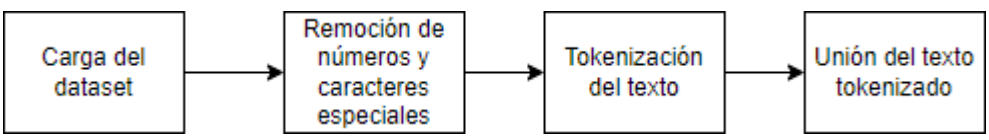

Figura 4. *Pipeline de pre-procesamiento.*

Una vez realizado el pre-procesamiento de los textos, el siguiente paso consistió en la codificación de la variable de respuesta. Al tratarse de una variable binario se utilizó un *OneHotEncoding* y de esta manera al valor de “PAGÓ” se le asignó el número 1 y al valor de “NO PAGÓ” se le asignó el valor de 0.

5.3 Obtención de embbedings

Una vez realizado el pre-procesamiento de los textos y la codificación de nuestra variable objetivo el siguiente paso fue representar los textos a través de *embeddings*, de esta manera se hacían entendibles para los modelos de aprendizaje automático que se deseaban implementar. Para la extracción de los *embeddings* se realizaron solicitudes al *endpoint* de la API de OpenAI. Se utilizó el modelo “text-embedding-ada-002” para obtener las representaciones numéricas. La longitud de los *embeddings* para cada texto fue de 1516.

5.4 Entrenamiento de Modelos

Después de obtener los *embeddings* para los textos procesados, se realizó un gráfico para observar el conteo de las categorías de nuestra variable objetivo el cual se muestra en la Figura 5.

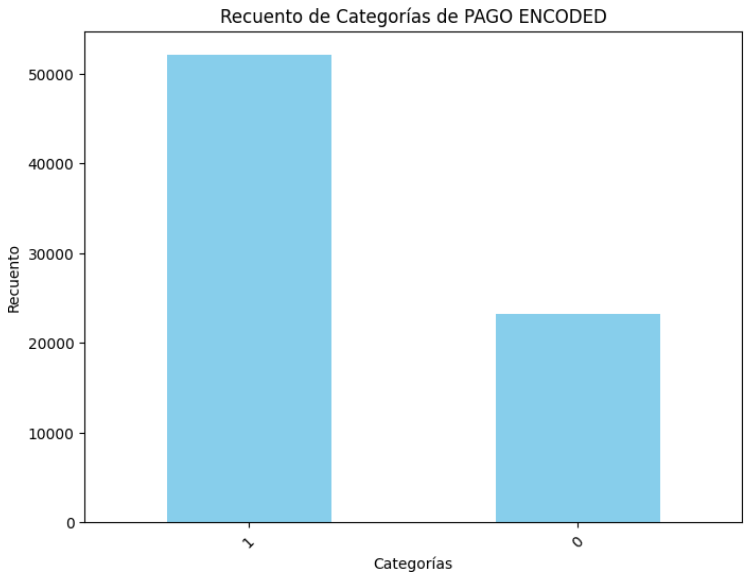

Figura 5. *Conteo de categorías variable objetivo.*

Podemos observar que tenemos un desbalance en el conteo de las categorías. La variable 1 que corresponde a la etiqueta “PAGÓ” tiene aproximadamente el doble de registros que la variable 0 que corresponde a la etiqueta “NO PAGÓ”. Por esta razón se decidió realizar un proceso de *UnderSampling* para equilibrar las categorías de la variable de respuesta. En la Figura 6 se muestra el conteo de las categorías del *dataset* después de realizar el balanceo de categorías.

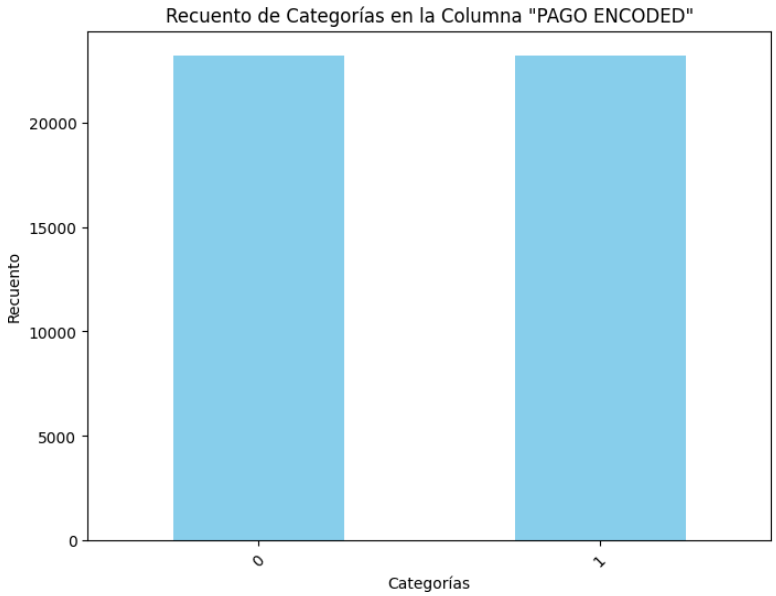

Figura 6. *Conteo de categorías variable objetivo después de UnderSampling.*

El *dataset* con las categorías balanceadas quedó con 44.000 registros aproximadamente. Una vez finalizado el proceso de *undersampling* se tomaron dos conjuntos de datos: uno con las categorías desbalanceadas, pero con más registros (75.000) y el otro con las categorías balanceadas, pero con muchos menos registros (44.000). Estos conjuntos de datos se dividieron posteriormente en conjuntos de entrenamiento y conjunto de testeo. Se entraron los modelos y posteriormente se realizó la evaluación con el conjunto de testeo. Las métricas de evaluación utilizadas para medir el desempeño de los modelos fueron el *accuracy score*, el *recall* y el *f-1 score*, las cuales se implementaron mediante la biblioteca sklearn, la librería metric y los métodos *classification report* y *accuracy score*. Los modelos que se implementaron se listan a continuación:

- *Random Forest*
- Red Neuronal
- Modelo pre-entrenado BERTO

5.5 Random Forest

Para la implementación del *Random Forest* se utilizó la biblioteca sklearn, la librería ensemble y el método de *RandomForest Classifier*. En la Tabla 1 se muestran los resultados de testeo para el modelo *RandomForest* con el *dataset* desbalanceado.

	Precisión	Recall	f-1 score
Categoría “NO PAGÓ”	0.31	0.03	0.06
Categoría “PAGÓ”	0.69	0.96	0.80

Tabla 1. *Resultados conjunto de prueba RandomForest sin Undersampling.*

En la Tabla 2 se muestran los resultados de testeo para el modelo *RandomForest* con el *dataset* balanceado.

	Precisión	Recall	f-1 score
Categoría “NO PAGÓ”	0.50	0.49	0.49
Categoría “PAGÓ”	0.51	0.52	0.51

Tabla 2. *Resultados conjunto de prueba RandomForest con Undersampling.*

5.6 Red Neuronal

Se implementaron dos variantes de red neuronal pre-alimentada. Para su construcción se utilizó la biblioteca PyTorch, se procesaron los *embeddings* y los *encodings* como tensores, de manera que se pudieran utilizar en la biblioteca, estos tensores se cargaron en *loaders* de manera que se combinan los tensores de los datos y las etiquetas para que fueran los inputs de las redes. La función de pérdida para ambas arquitecturas fue la entropía cruzada, debido a que es una métrica que penaliza fuertemente las predicciones incorrectas.

La primera arquitectura se construyó con dos capas lineales, a la primera capa se le asignó un total de 1536 nodos de entrada, lo cual corresponde con el tamaño del *data loader* de entrada, y 500 nodos de salida. Después de esta capa se adicionó una función de activación RelU. La salida de la función RelU ingresa a la segunda capa la cual tiene 500 nodos de entrada y 2 de salida, los cuales corresponden con las dos clases que se desean clasificar. De esta manera queda completa la primera arquitectura de red neuronal. Finalmente, se utilizó un optimizador ADAM (*Adaptive Moment Estimation*) para incluir la corrección de peso de decaimiento, lo que ayuda a prevenir ciertos problemas de sobreajuste.

En la Tabla 3 se muestran los resultados del conjunto de testeo para la primera arquitectura de red neuronal con el *dataset* desbalanceado.

	Precisión	Recall	f-1 score

Categoría “NO PAGÓ”	0.00	0.00	0.00
Categoría “PAGÓ”	0.69	1.00	0.81

Tabla 3. *Resultados conjunto de prueba Red Neuronal 1 sin Undersampling.*

En la Tabla 4 se muestran los resultados del conjunto de testeo para la primera arquitectura de red neuronal con el *dataset* balanceado.

	Precisión	Recall	f-1 score
Categoría “NO PAGÓ”	0.52	0.43	0.47
Categoría “PAGÓ”	0.52	0.60	0.56

Tabla 4. *Resultados conjunto de prueba Red Neuronal 1 con Undersampling.*

La segunda arquitectura consistió en 3 capas lineales, una de entrada y 2 ocultas, cada una con 1536 nodos de entrada, la capa de entrada y la primera capa oculta contaron con el mismo número de nodos de salida, mientras que la última capa oculta le fue asignado un total de 2 nodos de salida correspondientes a las categorías a clasificar. A la salida de la capa de entrada y de la primera capa oculta se aplicó una función de activación RelU y una capa de normalización por lotes. Finalmente, a la salida de la primera capa oculta se aplica también un *dropout* para evitar sobreajuste.

En la Tabla 5 se muestran los resultados del conjunto de testeo para la segunda arquitectura de red neuronal con el *dataset* desbalanceado.

	Precisión	Recall	f-1 score
Categoría “NO PAGÓ”	0.00	0.00	0.00
Categoría “PAGÓ”	0.69	1.00	0.81

Figura 5. *Resultados conjunto de prueba Red Neuronal 2 sin Undersampling.*

En la Tabla 6 se muestran los resultados del conjunto de testeo para la segunda arquitectura de red neuronal con el *dataset* balanceado.

	Precisión	Recall	f-1 score
Categoría “NO PAGÓ”	0.50	0.64	0.56
Categoría “PAGÓ”	0.50	0.36	0.42

Tabla 6. *Resultados conjunto de prueba Red Neuronal 2 con Undersampling.*

Para ambas redes neuronales se utilizó una tasa de aprendizaje de $2 * 10^{-3}$ y un total de 5 épocas de entrenamiento. Adicionalmente se utilizó un optimizador AdamW para hacer el entrenamiento más robusto.

5.7 Modelo pre-entrenado BERTO

El último modelo que se implementó para evaluar su desempeño en esta tarea de clasificación fue un modelo basado en BERT para el idioma español de la biblioteca *Transformers* de Hugging Face, el modelo puede encontrarse dentro de hugging face (https://huggingface.co/) como "dccuchile/bert-base-spanish-wwm-uncased". Este modelo se evaluó únicamente con el *dataset* sin el balanceo de categorías.

El primer paso para implementar este modelo fue la tokenización de los textos, para este fin se utilizó el AutoTokenizer del modelo pre-entrenado BERT. Este AutoTokenizer cumplió dos funciones al mismo tiempo: dividir el texto en entidades más pequeñas y adicionalmente representarlos numéricamente para que pudieran ser entendidos por el modelo de aprendizaje. Debido a las propias limitaciones del modelo, el número de tokens máximo para cada texto fue de 413. Posteriormente se utilizó la biblioteca pytorch para crear los tensores del *dataset* de entrenamiento y de testeo. Adicionalmente, se crearon tanto el *train* como el *test loader* para cargar los datos en lotes al modelo de entrenamiento. El tamaño de lote se estableció en 16, y se agregó una condición de aleatorización para que los datos se mezclen en cada época de entrenamiento, de esta manera el modelo no vio los mismos datos en el mismo orden.

Los parámetros de entrenamiento para el modelo fueron los siguientes: la tasa de aprendizaje se fijó en $2 * 10^{-3}$ y 3 épocas. Al igual que con los modelos de red neuronal, se utilizó un optimizador ADAM para evitar el sobreajuste. En la Tabla 7 se muestran los resultados obtenidos después de evaluar el modelo con el *dataset* de testeo.

	Precisión	Recall	f-1 score
Categoría “NO PAGÓ”	0.00	0.00	0.00
Categoría “PAGÓ”	0.69	1.00	0.81

Tabla 7. *Resultados dataset sin balancear modelo pre-entrenado BERT.*

6 Discusión de Resultados

Al comparar los resultados de las Tablas 1 y 2 podemos evidenciar que, al balancear las categorías, el modelo de *Random Forest* mejora la clasificación de la categoría “0” correspondiente a “NO PAGÓ”. Podemos evidenciarlo ya que el *accuracy score* pasa de un valor de 0.31, con el *dataset* sin balancear, a un valor de 0.50, con el *dataset* balanceado. El *recall* presenta una mejora desde un valor de 0.03 hasta 0.49, esto nos indica que el modelo pasa de identificar correctamente el 3% de los registros con categoría “NO PAGÓ” a identificar correctamente el 49% de los registros.

Al comparar los resultados de la Tabla 3 y la Tabla 4 podemos observar que la

arquitectura de red neuronal simple no identifica los elementos de la categoría “NO PAGÓ” al entrenarse con el *dataset* desbalanceado. Cuando se entrena con el *dataset* balanceado, se obtiene un *accuracy score* de 0.52 y un *recall* de 0.43. Al comparar los resultados de la Tabla 5 y la Tabla 6 podemos observar que la arquitectura de red neuronal mejorada no identifica los elementos de la categoría “NO PAGÓ” al entrenarse con el *dataset* desbalanceado. Cuando se entrena con el *dataset* balanceado, se obtiene un *accuracy score* de 0.50 y un *recall* de 0.64.

Para los tres modelos observamos que la mejora en las métricas para la categoría “NO PAGÓ” supone un desmejoramiento en las métricas para la categoría “PAGÓ”, adicionalmente se muestran mejores resultados después de realizar el *undersampling* y balancear las categorías de la variable respuesta. Desde la perspectiva del negocio es preferible identificar los clientes que puedan incurrir en el incumplimiento de pago, así que podemos afirmar que los modelos mejoran su desempeño al realizar el balanceo de categorías. Se observa que el balanceo de categorías mejora el desempeño de los modelos para detectar la clase de “NO PAGÓ”, esto debido a que cuando se entrena teniendo un desbalanceo, el modelo no tiene suficientes casos de la clase minoritaria para aprender correctamente los patrones.

Podemos evidenciar que la red neuronal no logra identificar la clase minoritaria al realizar el entrenamiento con el *dataset* desbalanceado, mientras que por otro lado el *RandomForest* logra identificar una parte de esta misma clase. Debido a que los árboles individuales están expuestos a diferentes subconjuntos de datos y tienen profundidad limitada, son capaces de aprender patrones de ambas clases de manera relativamente equitativa. La red neuronal tiene mayores problemas para manejar el desbalanceo de clases, por lo que no tiene implementados métodos de remuestreo durante el entrenamiento.

Comparando los resultados de la Tabla 6 con los resultados de la Tabla 4 podemos observar que la arquitectura de la segunda red neuronal mejora la métrica de *recall* para la categoría “NO PAGÓ”, esto significa que identifica una mayor cantidad de elementos etiquetados en esta categoría que la arquitectura de la primera red neuronal. Esta mejora para identificar la clase minoritaria se puede explicar mediante la incorporación de capas de *Batch Normalization* después de las capas ocultas. La normalización por lotes ayuda a estabilizar y acelerar el entrenamiento al normalizar las activaciones de cada capa. Esto es útil cuando hay desequilibrios en los datos, ya que puede ayudar a prevenir problemas como el desvanecimiento del gradiente, permitiendo así una mejor propagación de los gradientes durante el entrenamiento. Adicionalmente, la incorporación de *Dropout* ayuda a prevenir el sobreajuste al desactivar aleatoriamente un porcentaje de neuronas durante el entrenamiento. Esto puede ser beneficioso para evitar que la red se ajuste demasiado a los datos de entrenamiento, lo que podría hacer que se vuelva demasiado específica para la clase mayoritaria y no generalice bien para la clase minoritaria.

Finalmente, observamos que los resultados del entrenamiento para el modelo pre-entrenado BERT utilizando el *dataset* sin balancear son iguales a los obtenidos para las dos arquitecturas de redes neuronales cuando se entrenaron con el *dataset* sin balancear. Esto es debido a que finalmente el modelo de *transformer* en el fondo es una red neuronal con mecanismos de atención, por esta razón tiene problemas para identificar la clase minoritaria cuando el *dataset* está muy desbalanceado. Adicionalmente, el máximo de tokens por texto fueron 413, lo cual representa una cantidad muy pequeña del total de los textos, por lo que es posible que se haya perdido información importante para lograr la clasificación en el proceso de tokenización. Otra de las desventajas de implementar el modelo BERT fue el tiempo de entrenamiento, al ser un modelo tan complejo requirió de la utilización de GPU para poder agilizar el proceso, y aun así su tiempo de ejecución fue de 5 horas aproximadamente, siendo este mucho mayor en comparación con el de los demás modelos.

7 Conclusiones y Recomendaciones

La aplicación de la minería de texto en el campo de la gestión del riesgo financiero mediante la predicción del *default* es un área que está en desarrollo. Se han llevado a cabo algunas investigaciones extrayendo información sobre los textos de aplicación a los créditos. Este estudio pretende contribuir mostrando un enfoque orientado hacia el aprovechamiento de fuentes de datos dentro de las compañías, las cuales no son muy aprovechadas, como lo son las líneas de servicio al cliente. Adicionalmente, mostrando que la predicción de default no es una única tarea que se hace antes de otorgar el crédito, sino un proceso de seguimiento que puede hacerse cuota a cuota hasta la terminación del mismo.

Los resultados mostraron claramente que los modelos que se implementaron funcionan en la medida en que las categorías de la variable objetivo estén balanceadas. Por esta razón, se debe utilizar un método adecuado que equilibre ambas clases. En nuestro caso el método escogido fue realizar un *UnderSampling* de las muestras de la clase mayoritaria, para futuros trabajos resultaría interesante realizar el caso contrario: un *OverSampling*, generando datos sintéticos de la clase mayoritaria.

Adicionalmente, la métrica más importante para medir el desempeño de los modelos fue el *recall*, ya que esta métrica nos permite determinar cuántas observaciones de cada categoría fueron clasificadas correctamente. Los resultados muestran que la clasificación de ambas categorías es pareja, teniendo valores de *recall* muy cercanos a 0.5 para ambas categorías. Sin embargo, el modelo de la red neuronal modificada logró el mejor desempeño, llegando a identificar correctamente hasta el 64% de los clientes que incumplieron con el pago.

Se pudo evidencias también que la naturaleza del algoritmo de bosque aleatorio, resultó útil a la hora de identificar la clase minoritaria sin necesidad de realizar el balanceo de clases, aunque su desempeño no fue muy bueno.

El modelo pre-entrenado BERT fue el que mostró los peores resultados, la sospecha principal es que debido al límite de tokens que tenía el tokenizador se perdió información fundamental para poder identificar la clase minoritaria.

Finalmente, para futuros trabajos se hace interesante explorar otras arquitecturas de red neuronal como redes recurrentes para evaluar su desempeño de clasificación, así como también la posibilidad de realizar una traducción de los textos y entrenarlos con un modelo pre-entrenado en inglés, ya que estos cuentan con ciertas ventajas en vocabulario sobre los modelos pre-entrenados en español.

Realizar la comparación de modelos tradicionales de scoring contra modelos que utilicen información de texto también es una tarea interesante para futuros trabajos. De esta manera se puede determinar con exactitud los pros y contras de cada aproximación.

También un enfoque en el cual se utilice análisis de supervivencia es útil para evaluar la duración hasta el incumplimiento y para comparar la supervivencia entre diferentes grupos de clientes o condiciones. También es valioso para identificar patrones de comportamiento a lo largo del tiempo y para comprender mejor la dinámica temporal de eventos críticos, lo que puede ser esencial para la toma de decisiones en la gestión de riesgos y la asignación de créditos.

8 Bibliografía

Al-jabery, K.K. *et al.* (2020) 'Data Analysis and machine learning tools in MATLAB and python', *Computational Learning Approaches to Data Analytics in Biomedical Applications*, pp. 231–290. doi:10.1016/b978-0-12-814482-4.00009-7.

Arora, S. et al. (2022) 'Prediction of credit card defaults through data analysis and Machine Learning Techniques', Materials Today: Proceedings, 51, pp. 110–117. doi:10.1016/j.matpr.2021.04.588.

Breiman, L. (2001) Machine Learning, 45(1), pp. 5–32. doi:10.1023/a:1010933404324.

Casola, S., Lauriola, I. and Lavelli, A. (2022) 'Pre-trained transformers: An empirical comparison', *Machine Learning with Applications*, 9, p. 100334. doi:10.1016/j.mlwa.2022.100334.

Chen, S.Y. and Liu, X. (2008) 'Data mining in practice', *Data Warehousing and Mining*, pp. 2273–2280. doi:10.4018/978-1-59904-951-9.ch134.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Fávero, L.P., Belfiore, P.P. and Rafael, D.F.S. (2023) Data Science, Analytics and machine learning with R. London: Academic Press.

Fränti, P. and Mariescu-Istodor, R. (2023) 'Soft precision and recall', *Pattern Recognition Letters*, 167, pp. 115–121. doi:10.1016/j.patrec.2023.02.005.

Goodfellow, I., Bengio, Y. and Courville, A. (2017) Deep learning. Cambridge, Mass: The MIT Press

Haddi, E., Liu, X. and Shi, Y. (2013) 'The role of text pre-processing in sentiment analysis', *Procedia Computer Science*, 17, pp. 26–32. doi:10.1016/j.procs.2013.05.005.

Hassanien, A.E., Chatterjee, J.M. and Jain, V. (2022) Artificial Intelligence and industry 4.0. London: Academic Press, an imprint of Elsevier.

Jalal, N. et al. (2022) 'A novel improved random forest for text classification using feature ranking and optimal number of trees', Journal of King Saud University - Computer and Information Sciences, 34(6), pp. 2733–2742. doi:10.1016/j.jksuci.2022.03.012.

Jatnika, D., Bijaksana, M.A. and Suryani, A.A. (2019) 'Word2Vec model analysis for semantic similarities in English words', *Procedia Computer Science*, 157, pp. 160–167. doi:10.1016/j.procs.2019.08.153.

Kriebel, J. and Stitz, L. (2022) 'Credit default prediction from user-generated text in peer-to-peer lending using Deep Learning', European Journal of Operational Research, 302(1), pp. 309–323. doi:10.1016/j.ejor.2021.12.024.

Kumar, R. (2017) Strategic Financial Management Casebook. Amsterdam, Netherlands: Elsevier.

LeCun, Y., Bengio, Y. and Hinton, G. (2015) 'Deep learning', Nature, 521(7553), pp. 436–444. doi:10.1038/nature14539.

Li, B. (2022) 'Online loan default prediction model based on Deep Learning Neural Network', Computational Intelligence and Neuroscience, 2022, pp. 1–9. doi:10.1155/2022/4276253.

Liu, Y. *et al.* (2020) 'Multilingual denoising pre-training for Neural Machine Translation', *Transactions of the Association for Computational Linguistics*, 8, pp. 726–742. doi:10.1162/tacl_a_00343.

Liu, X., Wu, Q. and Pan, W. (2018) 'Sentiment Classification of micro-blog comments based on Randomforest algorithm', Concurrency and Computation: Practice and Experience, 31(10). doi:10.1002/cpe.4746.

Miner, G. (2016) *Practical text mining and statistical analysis for non-structured text data applications*. Amsterdam: Academic Press.

Mosin, V. *et al.* (2023) ‘Fine-tuning transformers: Vocabulary Transfer’, *Artificial Intelligence*, 317, p. 103860. doi:10.1016/j.artint.2023.103860.

Piris, Y. and Gay, A.-C. (2021) ‘Customer satisfaction and natural language processing’, *Journal of Business Research*, 124, pp. 264–271. doi:10.1016/j.jbusres.2020.11.065.

Poitras, G. (2002) ‘Risk-management concepts’, *Risk Management, Speculation, and Derivative Securities*, pp. 99–163. doi:10.1016/b978-012558822-5.50003-6.

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, *1*(8), 9.

Sanjar, K. et al. (2020) ‘Weight dropout for preventing neural networks from overfitting’, 2020 8th International Conference on Orange Technology (ICOT) [Preprint]. doi:10.1109/icot51877.2020.9468799.

Schröer, C., Kruse, F. and Gómez, J.M. (2021) ‘A systematic literature review on applying CRISP-DM process model’, *Procedia Computer Science*, 181, pp. 526–534. doi:10.1016/j.procs.2021.01.199.

Sever, K., Golušin, L.M. and Lončar, J. (2023) ‘Optimization of gradient descent parameters in attitude estimation algorithms’, Sensors, 23(4), p. 2298. doi:10.3390/s23042298.

Stent, A. and Bangalore, S. (2014) *Natural language generation in interactive systems*. Cambridge: Cambridge University Press.

Stevenson, M., Mues, C. and Bravo, C. (2021) ‘The value of text for small business default prediction: A deep learning approach’, European Journal of Operational Research, 295(2), pp. 758–771. doi:10.1016/j.ejor.2021.03.008.

Theodoridis, S. (2015) ‘Neural networks and deep learning’, Machine Learning, pp. 875–936. doi:10.1016/b978-0-12-801522-3.00018-5.

Toom, K. and Miller, P.F. (2018) ‘Ethics and integrity’, *The European Research Management Handbook*, pp. 263–287. doi:10.1016/b978-0-12-805059-0.00013-4.

V, V. (2023) Thyroid disease detection using machine learning approach [Preprint]. doi:10.31219/osf.io/peha2.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Zhu, C. (2021) ‘Pretrained language models’, *Machine Reading Comprehension*, pp. 113–133. doi:10.1016/b978-0-323-90118-5.00006-0.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision* (pp. 19-27).

Zhu, X. et al. (2023) 'Explainable prediction of loan default based on machine learning models', Data Science and Management, 6(3), pp. 123–133. doi:10.1016/j.dsm.2023.04.003.