ESCUELA DE NEGOCIOS

MAESTRÍA EN INTELIGENCIA DE NEGOCIOS Y CIENCIA DE DATOS

ANÁLISIS DE CALIFICACIÓN DE CRÉDITO PARA CLIENTES DE INSTITUCIONES FINANCIERAS

Profesor
Manuel Eugenio Morocho Cayamcela

Autor
Carlos Javier Madrid Borja

2024

RESUMEN

Este proyecto aborda el problema que tienen que solucionar las instituciones financieras para clasificar y calificar clientes idóneos previo al desembolso de créditos.

Se propone hacer uso de inteligencia artificial y algoritmos de ciencia de datos para encontrar la mejor solución; para esto, se enfrentan a tres modelos de clasificación candidatos: Regresión Logística Multinomial, Random Forest Multiclase y Red Neuronal Artificial Multiclase para la construcción de sendos modelos cuyos resultados serán evaluados de acuerdo a su métrica de precisión al clasificar correctamente clientes dentro de las tres opciones posibles: malo, bueno y estándar.

Luego de comparar los resultados obtenidos se seleccionó al último como la mejor opción debido al porcentaje de precisión alcanzado.

Finalmente, se concluye que cualquier institución financiera que lo implemente puede tener la confianza de que logrará los resultados esperados ayudando a aumentar su ventaja competitiva e innovación tecnológica frente a la competencia.

ABSTRACT

This project addresses the problem that financial institutions have to solve to classify and qualify suitable clients prior to the disbursement of loans.

It is proposed to use artificial intelligence and data science algorithms to find the best solution; for this, there are three candidate classification models: Multinomial Logistic Regression, Multiclass Random Forest and Multiclass Artificial Neural Network for the construction of several models whose results will be evaluated according to their precision metric when correctly classifying clients within the three possible options: bad, good and standard.

After comparing the results obtained, the last one was selected as the best option due to the percentage of precision achieved.

Finally, it is concluded that any financial institution that implements it can trust that it will achieve the expected results, helping to increase its competitive advantage and technological innovation over the competition.

ÍNDICE DEL CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El crédito es un préstamo de dinero que una institución financiera otorga a su cliente para cumplir sus objetivos personales y de negocios, con el compromiso de que, en el futuro, el cliente devolverá dicho préstamo en forma gradual (mediante el pago de cuotas) o en un solo pago (mediante el prepago) y con un interés adicional que compensa a quien presta el dinero, por todo el tiempo que no tuvo ese monto.

Los más importantes tipos de crédito son los Créditos de Consumo, Créditos Comerciales y Créditos Hipotecarios.

La calificación de crédito (credit scoring) es una técnica previa al desembolso de préstamos, ésta permite a las instituciones financieras evaluar y clasificar a los clientes frente a la probabilidad de incumplimiento.

La palabra “score” en inglés se traduce como “calificación” y en el contexto financiero se refiere a un indicador que estima la probabilidad de que un consumidor de un producto de crédito pueda o no pagar un préstamo determinado.

En este contexto, es de vital importancia para todas las instituciones financieras el hecho de poder contar con modelos de clasificación lo más exactos posibles al momento de calificar a los clientes previo a la emisión de una operación de crédito.

Por esta razón, el apoyo de la tecnología y el uso de algoritmos de business intelligence sirven para dar solución a este problema.

REVISIÓN DE LITERATURA

La regresión logística es una técnica estadística popular que se puede utilizar para modelar la probabilidad de un resultado binario, como incumplimiento o no incumplimiento, basándose en un conjunto de variables predictivas, como ingresos, puntaje crediticio, edad, etc. En el análisis la regresión logística puede ayudar a los prestamistas a evaluar la solvencia de los prestatarios potenciales y asignarles una probabilidad de incumplimiento (PD), que es un componente clave del cálculo de la pérdida esperada (EL). La regresión logística también puede ayudar a identificar los factores más importantes que influyen en el riesgo de incumplimiento y proporcionar información sobre la relación entre los predictores y el resultado Aplicación De La Regresión Logística En El Análisis Del Riesgo Crediticio – FasterCapital (2024).

Según Támara-Ayús et al. (2019) en los resultados y conclusiones publicados en el año 2019 en su estudio, menciona que “para ambos modelos regresión logística y red neuronal, se logra una precisión del 71% en la base de entrenamiento y del 72% en la base de comprobación, sin embargo, a pesar de obtener resultados similares, la regresión logística arrojó la menor tasa de malos en la zona de aceptación.

Las dos técnicas utilizadas son adecuadas para el estudio y predicción de la probabilidad de incumplimiento de un cliente correspondiente a una cartera de consumo, lo anterior, respaldado por el alto índice de eficacia predictiva en ambos modelos”. En este sentido, para el presente trabajo se validará cuál de los dos modelos logra una mayor precisión.

Otros autores como Oswaldo et al. (2019) en sus estudios afirman que el modelo de red neuronal artificial es más robusto comparado con el modelo de regresión logística: “El modelo de redes neuronales se ajusta de mejor forma a los datos, pues se obtiene un criterio de información Akaike menor al de regresión logística, con una diferencia de 8.029,2 puntos. De igual forma, los estadísticos KS, coeficiente de Gini y curva ROC evidencia que hacer uso de las redes neuronales

logra una mejor clasificación de los clientes, con 5,19, 5,84 y 2,92 puntos porcentuales por encima de los estadísticos del modelo de regresión logística, respectivamente. Finalmente, la matriz de confusión muestra un menor error con el modelo de redes neuronales, al compararlos con un mismo punto de corte óptimo. Los resultados obtenidos evidencian que la metodología de redes neuronales proporciona un modelo scoring más robusto que al usar una regresión logística, pudiendo corroborar que la hipótesis planteada es verdadera, bajo el proceso de modelización empleado". Lo interesante de la revisión bibliográfica es la variedad de opiniones y resultados encontrados para solucionar el mismo problema, como se puede observar se tienen argumentos contrarios que serán confirmados o rechazados de acuerdo a los resultados propios de esta investigación.

Un estudio realizado en Perú por Alejandro & Flores (2021) que pretende comparar el éxito de la implementación de soluciones de credit score en países desarrollados vs países emergentes señala que "para lograr replicar el éxito que tienen los modelos de credit scoring en los países desarrollados, es necesario reunir suficiente información y que ésta sea relevante y concreta, mientras que en los países emergentes mucha de esta información suele ser cualitativa e informal. Sin embargo, esto no impide que la calificación crediticia tenga lugar en las operaciones diarias de las entidades financieras. Si bien hay consenso entre los investigadores de que ésta no llega a tener la eficacia que tiene en los países desarrollados y que no reemplazará a la evaluación cualitativa de los analistas. Aun así, el puntaje obtenido puede ser útil en la estimación y predicción del riesgo (y por lo tanto en la estimación del costo), incluso si la entidad basa su decisión en el criterio del analista de créditos. Por lo tanto, en los países emergentes, la calificación complementa, aunque no reemplaza las metodologías financieras actuales"

Un factor importante para el éxito de la implementación efectiva de soluciones y modelos de score crediticio está determinado por las herramientas tecnológicas que utilizan las instituciones financieras para lograr ventaja competitiva frente a

sus adversarios de negocios, según los autores Bülbül et al. (2019) en su estudio publicado en 2019 mencionan que “los niveles de sofisticación de las herramientas usadas en los sistemas de administración de riesgo crediticio en los bancos estudiados, son fundamentalmente consecuencia de los niveles de competencia y concentración del mercado de préstamos” haciendo cada vez más importante la inversión tecnológica en software especializado para este fin.

Un estudio de Freire López (2021) realizado en Ecuador en la Universidad Internacional SEK en el año de 2021 habla acerca del uso de la metodología CRISP-DM también propuesta para este proyecto utilizando el modelo de Random Forest en el que se menciona lo siguiente: “En la creación de modelo de clasificación se utiliza la base de datos de clientes de crédito de la organización con un histórico desde el año 2017 al 2020, este conjunto de datos cuenta con 18 variables y 63.896 registros. Adicionalmente, para la implementación del modelo de clasificación se utiliza la metodología CRISP-DM. Posteriormente, se prepara la información con el preprocesamiento de datos, en este paso se utiliza la técnica de eliminación de datos atípicos, con lo cual el conjunto de datos se reduce de 63.896 a 58.247. Finalmente, se seleccionan las variables con mayor importancia con el método de chi cuadrado, en este caso son 8 variables seleccionadas. El modelo es implementado con Random Forest, el cual arroja una precisión mayor al 97%, el porcentaje de error es del 2,8%, con el 2,1% falsos positivos y 11,1% falsos negativos para predecir. Finalmente, la creación del modelo ayuda a contar con una herramienta adicional que sirve para clasificar automáticamente a los clientes como “buenos” y “malos” pagadores, lo que puede ser utilizado para entregar créditos con más rapidez y con un menor grado de riesgo. Sin embargo, este modelo necesita ser desplegado como lo indica la metodología CRISP-DM, para que el conocimiento obtenido sea aprovechado por el cliente”, lo que llama la atención y se propone comprobar en este proyecto si el alto valor de la precisión mencionada es superior al 97%.

De igual manera un estudio realizado en el año 2021 en la Escuela Politécnica del Ejército E.S.P.E. Ingeniería En Software (2021) de Ecuador muestra resultados comparativos precisamente de los tres modelos que se pretende enfrentar en el presente proyecto Capstone utilizando la metodología CRISP-DM:

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

Modelo/Métrica	Exactitud	Tasa de Error	Sensibilidad	Especificidad	Precisión	Predicción Negativa
Random Forest (ML1)	0.90	0.10	0.99	0.06	0.91	0.58
Regresión Logística (ML2)	0.90	0.10	0.99	0.003	0.90	0.33
Red Neuronal (ML3)	0.90	0.10	0.99	0.06	0.91	0.53

Ingeniería En Software (2021)

La información mostrada en el gráfico anterior, será sumamente útil para comparar posteriormente los resultados encontrados en la presente investigación.

Otro material interesante publicado en 2022 por Martínez Fernández (2022) avala la amplia variedad de métodos de resolución disponibles para resolver el problema planteado de análisis de score crediticio en el que se mencionan dos de los algoritmos que serán utilizados para este proyecto: “De acuerdo al marco regulatorio que rige a las instituciones financieras, es necesario que a la hora de evaluar el riesgo de crédito las empresas establezcan de forma clara modelos que estimen la probabilidad de que un cliente falle con el objetivo de constituir provisiones necesarias que permitan cubrir eventuales pérdidas. Comúnmente la técnica estadística adoptada para este propósito en la industria financiera corresponde a la regresión logística, sin embargo, en los últimos años se ha prestado una atención creciente a los algoritmos de aprendizaje automático (Machine Learning) para desafiar y explorar nuevas soluciones a la modelación

de la probabilidad de incumplimiento. Es por esto que el objetivo de la presente memoria de título consiste en comparar la capacidad predictiva de siete algoritmos de Machine Learning para la clasificación de deudores según su probabilidad de incumplimiento. Específicamente los algoritmos estudiados fueron **regresión logística**, análisis discriminante lineal, árboles de decisión, **random forest**, gradient boosting, extreme gradient boosting y support vector machines".

En otro estudio realizado en Colombia en el año 2020 por Borrero-Tigreros & Bedoya-Leiva (2020) se realiza la comparación entre tres técnicas de aprendizaje supervisado: "se proponen modelos para la predicción de riesgo crediticio en Colombia utilizando diferentes técnicas de inteligencia artificial. Estos modelos se pueden usar como apoyo por el área de gestión de riesgo en los bancos y tienen como objetivo identificar clientes que podrían incurrir en un estado de mora generando un posible riesgo de crédito para las entidades financieras. En particular, se proponen modelos basados en tres técnicas de aprendizaje supervisado (redes neuronales, árboles de decisión y máquinas de soporte vectorial) para predecir el próximo pago de la cuota de un cliente a partir de datos básicos de la operación, del cliente y de pagos de cuotas anteriores registradas. De acuerdo con los resultados obtenidos, los árboles de decisión resultan ser más exactos que las otras técnicas utilizadas para la predicción de riesgo crediticio con un área bajo la curva ROC de 88.29%. Los modelos propuestos alcanzan exactitudes similares y en algunos casos superan las exactitudes reportadas en algunos trabajos del estado del arte".

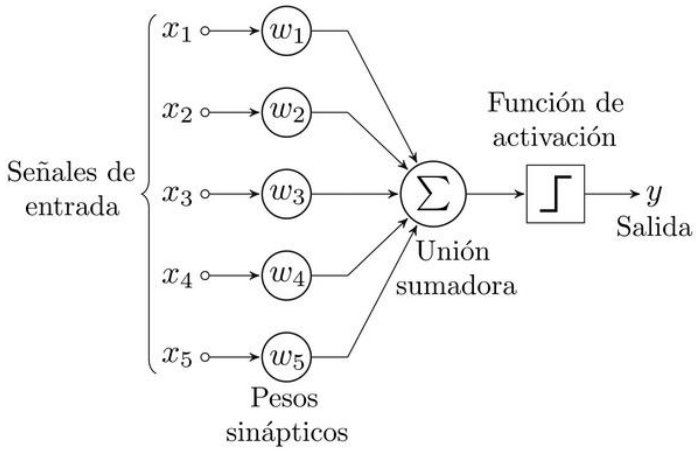
Algoritmo	Sn	Sp	Exact.	Precisión	AUC
Red neuronal	0.7183	0.6834	0.6362	0.4358	0.6991
C4.5	0.6428	0.8837	0.8245	0.6428	0.7824
Random Forest	0.5714	0.9302	0.8421	0.7272	0.8829
C5.0	0.6428	0.4651	0.6315	0.2812	0.8056
SVM	0.4444	0.8461	0.7192	0.5714	0.5930

Borrero-Tigreros & Bedoya-Leiva (2020)

Este estudio proporcionará para el presente proyecto la oportunidad de validar o refutar la premisa de que los árboles de decisión resultan ser más exactos frente a las otras técnicas propuestas.

En su estudio realizado en el año 2020 Gloria et al. (2020) menciona que: “Existen varios de tipos de RNA que se diferencian en su arquitectura, número de capas y su proceso de aprendizaje. Entre los tipos de RNA se encuentran los siguientes:

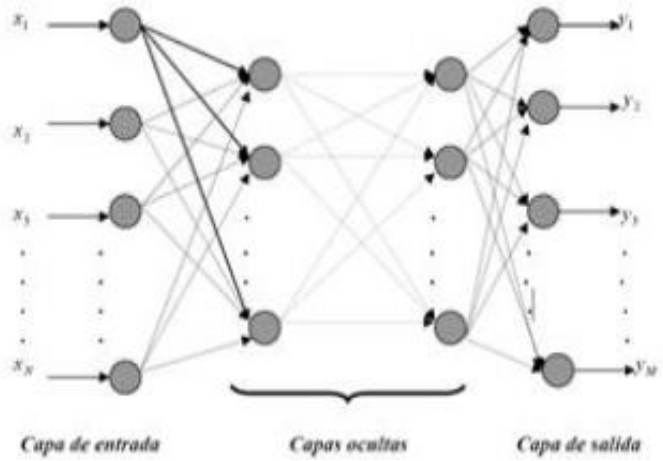
Perceptrón Simple: De acuerdo con Munt, A.M. (2018). Perceptrón simple es un modelo de Red Neuronal Artificial implementado en 1958 por el psicólogo Frank Rosenblatt y es considerado el modelo más simple y sencillo de las RNA debido a que consta de un conjunto de valores de entrada y una única neurona de salida la cual clasifica los valores en binarios, (1) si la neurona es activada o (0) desactivada.



Gloria et al. (2020)

Perceptrón Multicapa: Según Munt, A. M. (2018). Debido a que el Perceptrón simple era capaz de solucionar problemas de clasificación e implementar funciones lógicas en su procesamiento como las funciones AND y OR pero al mismo tiempo era incapaz de implementar la función XOR y con ello limitándose a no poder solucionar problemas de tipo no lineales, en 1969 Minsky y Papert plantean un modelo de RNA que introducía capas de neuronas artificiales

intermedias entre la capa de entrada y capa de salida permitiendo la implementación de cualquier función con el grado de precisión deseado, a este modelo de RNA se le llama Perceptrón Multicapa.



Gloria et al. (2020)

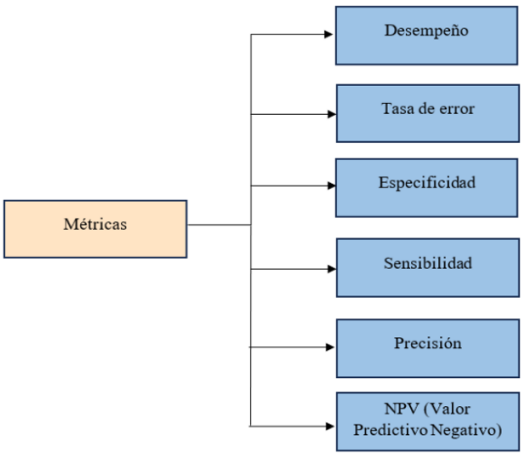
El método de aprendizaje de las redes neuronales se divide en dos tipos:

Aprendizaje supervisado: Según Quesada, F.J.G., Graciani, M.A.F., Bonal, M. T.L., & Díaz-Mata, M.A. (1994)., el aprendizaje supervisado se basa en la utilización de información ya existente donde se suministra a la RNA dos vectores, uno con los valores de entrada y otro con los valores de salida deseados. Con base a la salida generada por la red se realiza una comparación con la salida deseada según la entrada suministrada a la misma, actualizando los pesos sinápticos de la red con el fin de reducir el nivel de error generado. Esto se realiza en un número de iteraciones que permitan que el error o coste que existe entre la salida generada y la salida deseada sea el mínimo aceptable.

Aprendizaje no supervisado: Según Quesada, F.J.G., Graciani, M.A.F., Bonal, M.T.L., & Díaz-Mata, M.A. (1994)., el aprendizaje no supervisado es el más aproximado al funcionamiento del cerebro humano. Este tipo de aprendizaje no hace uso de las salidas deseadas, sino que a partir de las similitudes de las entradas suministradas la red compute, actualice los pesos sinápticos y genere

una misma salida.”. En base a esta información, se define que para el presente desarrollo se probarán con ambos tipos de redes neuronales: Perceptrón Simple y Perceptrón Multicapa para observar la diferencia en cuanto a los resultados obtenidos. Así mismo se define que el Aprendizaje Supervisado será el elegido.

Con respecto a las métricas seleccionadas para la predicción de riesgo crediticio, los autores De Computación et al. (2024) en su trabajo publicado el año 2024 sugieren las siguientes:



Nota. Métricas relevantes para el algoritmo de predicción de riesgo de crédito. Fuente: Elaboración Propia.

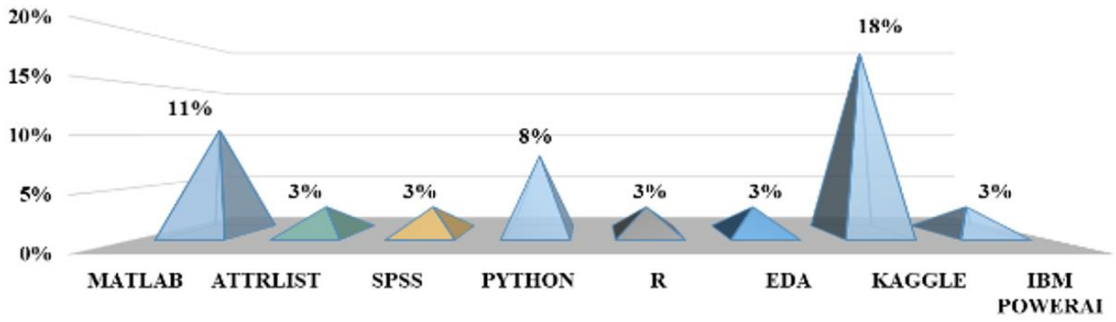
Dado que el tema de las métricas es importante para poder medir y cuantificar el grado de éxito de la implementación de los diferentes modelos, se tomará en cuenta como guía las métricas mencionadas en el gráfico anterior para seleccionar las que mejor se ajusten al presente proyecto.

Finalmente, y no menos importante, se busca información relevante acerca del software utilizado para realizar los entrenamientos, pruebas y la implementación de los algoritmos y modelos de credit score.

Los autores De Computacion et al. (2024) en su trabajo del año 2024 plantearon la siguiente pregunta y respuesta: “¿Qué software se utiliza para realizar las pruebas o implementaciones de los algoritmos? En los artículos seleccionados se hallaron un total de 8 herramientas de software utilizados para entrenamiento

y pruebas de los algoritmos IA, están distribuidos de la siguiente manera. Matlab en 11%, AttrList en 3%, SPSS en 3%, Python en 8%, R en 3%, EDA en 3%, Kaggle en 18%, IBM PowerAI en 3%. Las cifras indican que Kaggle es la opción más utilizada en entrenamiento-pruebas en 7 artículos; le sigue Matlab como software utilizado en 4 artículos. Una tercera herramienta es Python utilizada en 3 artículos, ver figura 10. Kaggle es una plataforma que ofrece conjunto de datos, competiciones, modelos y código para comunidades que trabajan en AI dirigido a algoritmo Machine Learning (Kaggle, 2023).

Matlab es una plataforma de computación y programación numérica que se utiliza para desarrollar algoritmos, análisis de datos y creación de modelos basados en AI Llerena Izquierdo (2023); Matlab (2023). Python permite integrar otros sistemas en forma efectiva Llerena Izquierdo (2023); Python (2023).”



De Computación et al. (2024)

En lo que respecta al tema del software, se propone trabajar con Python debido a que fue la herramienta de mayor utilización para el desarrollo de clases y tareas durante la presente maestría. Adicionalmente, el dataset elegido para esta investigación fue precisamente descargado de Kaggle.

IDENTIFICACIÓN DEL OBJETO DE ESTUDIO

El objeto de estudio se refiere al foco principal o el tema específico que se está investigando. Dicho de otra manera, es el fenómeno o la entidad sobre la cual se requiere obtener más conocimiento.

En el contexto del análisis de credit scoring, el objeto de estudio se centra en los aspectos y factores que influyen en la capacidad de los clientes solicitantes de crédito para cumplir con sus obligaciones financieras y no caer en mala calificación o clasificación.

Para este caso, el objeto de estudio son el comportamiento y las características de pago de los clientes.

Las entidades o variables involucradas en el objeto de estudio son los datos demográficos, el historial crediticio, los ingresos, las deudas y demás factores que pueden influir en la probabilidad de incumplimiento.

Este proyecto se justifica porque es necesario resolver el problema identificado en todas las organizaciones financieras que otorgan créditos debido a la importancia que tiene para éstas el contar con una herramienta confiable que emita predicciones basadas en los diferentes modelos matemáticos disponibles y así lograr ventaja competitiva con respecto a sus competidores.

PLANTEAMIENTO DEL PROBLEMA

El problema del análisis de calificación crediticia (credit scoring) es netamente de naturaleza económica y es fundamental en el ámbito financiero ya que permite evaluar la solvencia de los solicitantes de crédito.

Mediante la calificación y clasificación de clientes se puede predecir la probabilidad de incumplimiento del pago de obligaciones al momento de ingresar las solicitudes de crédito para su posterior emisión en cualquiera de las instituciones autorizadas del sistema financiero.

Para la resolución del problema se utilizarán datos históricos recolectados y proporcionados por las mismas instituciones financieras que contienen variables como el historial de crédito, ingresos, egresos, deudas anteriores, deudas actuales y demás características demográficas de los solicitantes.

Este problema es crítico para las organizaciones financieras porque de una buena implementación de la propuesta de solución depende mejorar la gestión del riesgo crediticio y ayudar a reducir la cartera vencida. De esta manera, se permitirá a las empresas tomar decisiones más informadas, reducir los riesgos y elevar la eficiencia del proceso de colocación de créditos en el mercado.

La justificación para adoptar un enfoque analítico del problema se basa en la facilidad y utilidad que brinda el análisis de datos aplicando técnicas estadísticas y de machine learning en el manejo de datos complejos.

OBJETIVO GENERAL

El objetivo general se refiere a la meta principal que se busca alcanzar mediante el análisis o la investigación del objeto de estudio.

En este caso el objetivo general es desarrollar un modelo predictivo que determine con precisión la probabilidad de que un solicitante de crédito incumpla con sus pagos con el propósito de mejorar la gestión de riesgos y la toma de decisiones de otorgamiento de créditos y minimizar el riesgo de incumplimiento para las instituciones financieras.

OBJETIVOS ESPECÍFICOS

De toda la variedad disponible de opciones para resolver el problema de análisis crediticio, se seleccionan tres modelos: Regresión Logística Multinomial, Random Forest Multiclase y Red Neuronal Artificial (ANN) Multiclase para cumplir con los siguientes objetivos específicos:

1. Elaborar un modelo de Regresión Logística Multinomial aplicado al problema de score crediticio para comparar los resultados obtenidos frente a los modelos de Random Forest Multiclase y Red Neuronal Artificial Multiclase.

2. Elaborar un modelo de Random Forest Multiclase aplicado al problema de score crediticio para comparar los resultados obtenidos frente a los modelos de Regresión Logística Multinomial y Red Neuronal Artificial Multiclase.

3. Elaborar un modelo de Red Neuronal Artificial Multiclase aplicado al problema de score crediticio para comparar los resultados obtenidos frente a los modelos de Regresión Logística Multinomial y Random Forest Multiclase.

JUSTIFICACIÓN Y APLICACIÓN DE LA METODOLOGÍA

1. Recolección de datos

La etapa de recolección es importante ya que permite cumplir con la fase de entendimiento mencionada en la metodología CRISP-DM utilizada para este proyecto.

Los datos fueron descargados desde el sitio web www.kaggle.com Credit score classification EDA and ANN Model (2024).

El archivo descargado “train.csv” contiene cien mil registros con datos recopilados del comportamiento crediticio de clientes que han solicitado créditos de todo tipo en una institución financiera.

La información está distribuida en 28 columnas, incluida la columna “Credit_Score” que contiene la variable objetivo con la calificación crediticia otorgada a cada uno de los clientes (Good, Poor, Standard).

2. Limpieza, pre-procesamiento y/o transformación de datos

Según la metodología CRIPS-DM, esta etapa corresponde a la preparación de datos e implica realizar las correcciones necesarias para acondicionar y estandarizar la información antes de seleccionar las variables para la implementación y ejecución de los modelos.

Previo a comenzar con las actividades de limpieza y transformación propiamente dichas, se inicia con el análisis exploratorio de datos EDA como sigue a continuación, cabe señalar que todas las tablas y figuras de esta sección son de creación propia:

1. Visualizar los tipos de datos:

#	Column	Non-Null Count	Dtype
1	ID	100000 non-null	object
2	Customer_ID	100000 non-null	object
3	Month	100000 non-null	object
4	Name	90015 non-null	object
5	Age	100000 non-null	object
6	SSN	100000 non-null	object
7	Occupation	100000 non-null	object
8	Annual_Income	100000 non-null	object
9	Monthly_Inhand_Salary	84998 non-null	float64
10	Num_Bank_Accounts	100000 non-null	int64
11	Num_Credit_Card	100000 non-null	int64
12	Interest_Rate	100000 non-null	int64
13	Num_of_Loan	100000 non-null	object
14	Type_of_Loan	88592 non-null	object
15	Delay_from_due_date	100000 non-null	int64
16	Num_of_Delayed_Payment	92998 non-null	object
17	Changed_Credit_Limit	100000 non-null	object
18	Num_Credit_Inquiries	98035 non-null	float64
19	Credit_Mix	100000 non-null	object
20	Outstanding_Debt	100000 non-null	object
21	Credit_Utilization_Ratio	100000 non-null	float64
22	Credit_History_Age	90970 non-null	object
23	Payment_of_Min_Amount	100000 non-null	object
24	Total_EMI_per_month	100000 non-null	float64
25	Amount_invested_monthly	95521 non-null	object
26	Payment_Behaviour	100000 non-null	object
27	Monthly_Balance	98800 non-null	object
28	Credit_Score	100000 non-null	object

Tabla 1 Visualización de tipos de datos

2. Validar filas duplicadas: No se hallaron filas duplicadas en el dataset.

3. Mostrar los estadísticos iniciales de los campos numéricos del dataset:

	count	mean	std	min	25%	50%	75%	max
Monthly_Inhand_Salary	84998	4194.2	3183.7	303.7	1625.6	3093.7	5957.5	15204.6
Num_Bank_Accounts	100000	17.09	117.4	-1	3	6	7	1798
Num_Credit_Card	100000	22.47	129.06	0	4	5	7	1499
Interest_Rate	100000	72.47	466.42	1	8	13	20	5797
Delay_from_due_date	100000	21.07	14.86	-5	10	18	28	67
Num_Credit_Inquiries	98035	27.75	193.18	0	3	6	9	2597
Credit_Utilization_Ratio	100000	32.29	5.12	20	28.05	32.31	36.5	50
Total_EMI_per_month	100000	1403.1	8306	0	30.31	69.25	161.22	82331

Tabla 2 Estadísticos iniciales de campos numéricos del dataset

4. Incluir las estadísticas iniciales de los campos no numéricos del dataset:

	count	unique	top	freq
ID	100000	100000	0x1602	1
Customer_ID	100000	12500	CUS_0xd40	8
Month	100000	8	January	12500
Name	90015	10139	Langep	44
Age	100000	1788	38	2833
SSN	100000	12501	#F%$D@*&8	5572
Occupation	100000	16	_______	7062
Annual_Income	100000	18940	36585.12	16
Num_of_Loan	100000	434	3	14386
Type_of_Loan	88592	6260	Not Specified	1408
Num_of_Delayed_Payment	92998	749	19	5327
Changed_Credit_Limit	100000	3635	_	2091
Credit_Mix	100000	4	Standard	36479
Outstanding_Debt	100000	13178	1360.45	24
Credit_History_Age	90970	404	15 Years and 11 Months	446
Payment_of_Min_Amount	100000	3	Yes	52326
Amount_invested_monthly	95521	91049	__10000__	4305
Payment_Behaviour	100000	7	Low_spent_Small_value_payments	25513
Monthly_Balance	98800	98790	__-333333333333333333333333333__	9
Credit_Score	100000	3	Standard	53174

Tabla 3 Estadísticas iniciales de campos no numéricos del dataset

5. Conteo de elementos nulos por cada variable:

ID	0
Customer_ID	0
Month	0
Name	9985
Age	0
SSN	0
Occupation	0
Annual_Income	0
Monthly_Inhand_Salary	15002
Num_Bank_Accounts	0
Num_Credit_Card	0
Interest_Rate	0
Num_of_Loan	0
Type_of_Loan	11408
Delay_from_due_date	0
Num_of_Delayed_Payment	7002
Changed_Credit_Limit	0
Num_Credit_Inquiries	1965
Credit_Mix	0
Outstanding_Debt	0

Credit_Utilization_Ratio	0
Credit_History_Age	9030
Payment_of_Min_Amount	0
Total_EMI_per_month	0
Amount_invested_monthly	4479
Payment_Behaviour	0
Monthly_Balance	1200
Credit_Score	0

Tabla 4 Conteo de elementos nulos por cada variable

6. Consolidar por separado solamente las variables con datos nulos:

Name	9985
Monthly_Inhand_Salary	15002
Type_of_Loan	11408
Num_of_Delayed_Payment	7002
Num_Credit_Inquiries	1965
Credit_History_Age	9030
Amount_invested_monthly	4479
Monthly_Balance	1200

Tabla 5 Conteo de nulos consolidado

7. Visualizar el volumen de datos nulos comparados con el total:

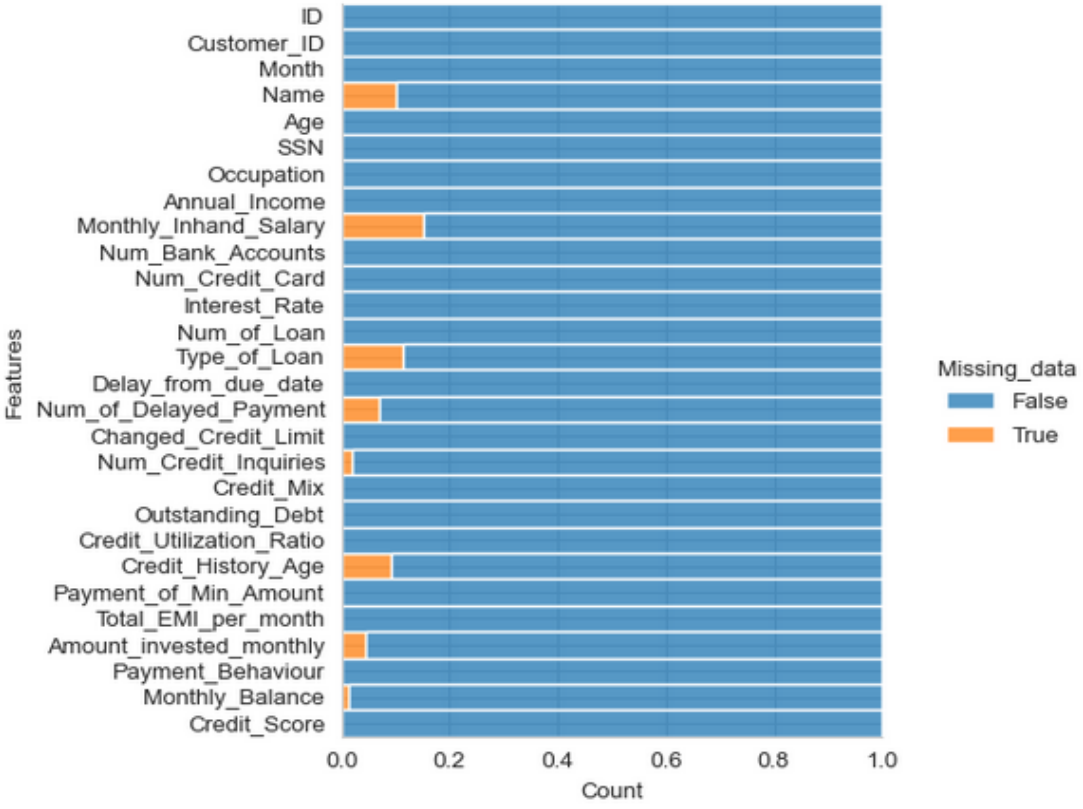


Figura 1 Volumen de datos nulos vs. total

Finalizada la etapa de análisis exploratorio de datos EDA se da inicio a la fase de limpieza, conversión y transformación de datos y variables que así lo requieren.

Las conversiones aseguran que los valores estén en un formato numérico limpio y consistente para su posterior análisis y modelización.

Ver en la sección de anexos el código y funciones utilizados para las tareas mostradas a continuación:

1. **Columna Age**, eliminar caracteres especiales y luego convertir a tipo entero manejando los casos en los que la conversión no es posible devolviendo None.
   Se define una función que trunca los dos últimos dígitos de la edad si la edad es mayor que 99 y luego aplica esta función a la columna Age, conservando las demás edades sin cambios.

2. **Columna Annual_Income**, eliminar los guiones bajos y convertir a float.

3. **Columna Monthly_Inhand_Salary**, llenar los valores faltantes con la moda de los salarios mensuales agrupados por cada ID de cliente. No se imputa utilizando la media anual porque los valores anuales son netos, en cambio los valores mensuales están sujetos a descuentos por concepto de seguros médicos, alimentación, parqueadero, etc.

4. **Columna Occupation**, limpiar caracteres extraños y reemplazar vacíos con la moda de las profesiones de cada cliente agrupadas por su número de seguridad social SSN, ya que cada cliente puede registrar bajo su único número SSN varios registros, uno por cada profesión reportada.

5. **Columna Num_of_Loan**, limpiar caracteres no deseados, convertir valores específicos a NaN y luego reemplazar los valores faltantes con el valor verosímil más común.

6. **Columna Num_of_Delayed_Payment**, llenar los valores vacíos con ceros y limpiar caracteres extraños.

7. **Columna Changed_Credit_Limit**, limpiar caracteres extraños, cambiar a tipo numérico y colocar valor promedio en vacíos.

8. **Columna Num_Credit_Inquiries**, identificar valores únicos y reemplazar vacíos por ceros

9. **Columna Credit_Mix**, limpiar caracteres extraños, calcular moda grupal y moda global para reemplazar valores null con modas calculadas.

10. **Columna Credit_History_Age**, unificar en una nueva columna los datos de años y meses a solamente meses, luego eliminar la columna original.

11. **Columna Amount_Invested_Monthly**, eliminar caracteres extraños y reemplazar con los valores de la moda de cada grupo de cliente.

12. **Columa Monthly_Balance**, eliminar caracteres extraños, cambiar el tipo de dato a float y reemplazar los valores nulos por la media.

13. **Columna Payment_Behaviour**, eliminar caracteres extraños y reemplazar valores nulos con los valores no nulos inmediatos anteriores de cada dato.

14. **Columna Interest_Rate**, convertir datos a float.

15. **Eliminar columnas innecesarias**, se eliminan las columnas ID, Customer_ID, Month, Name, SSN y Type_of_Loan porque ya fueron

utilizadas para procesar las demás y no aportan información significativa para los modelos.

16. **Validar persistencia de datos perdidos**, en tal caso continuar aplicando limpieza y transformaciones.

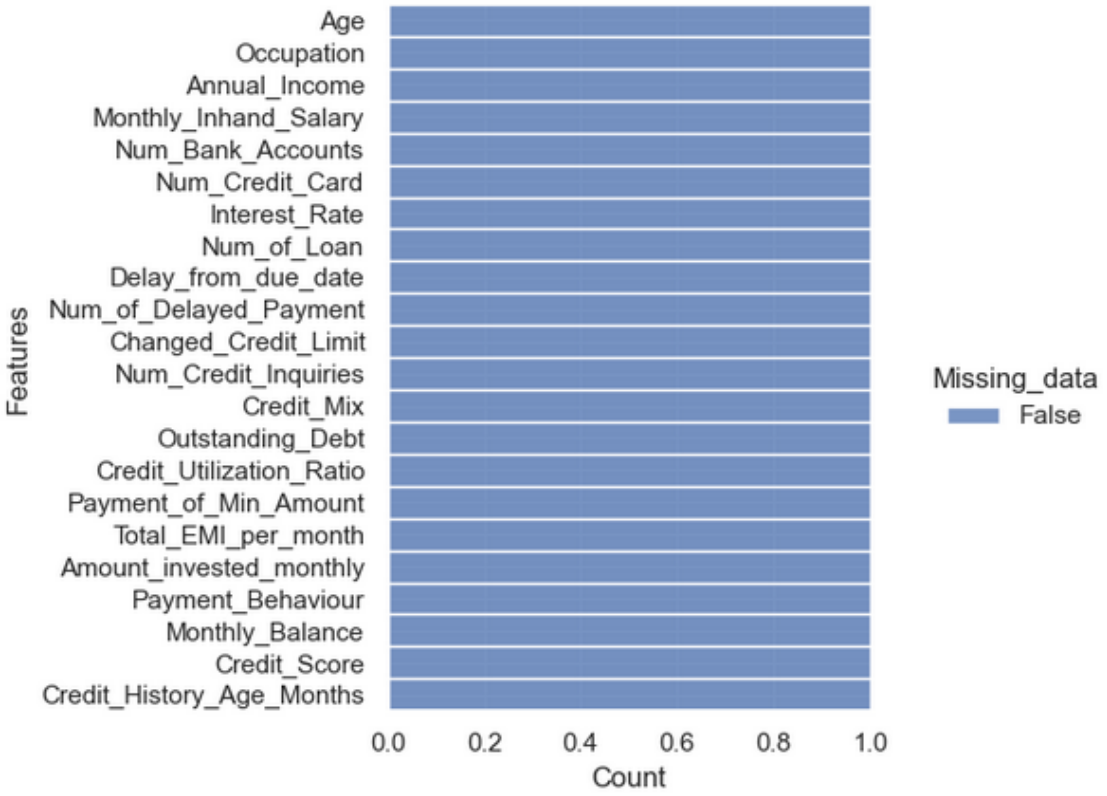


Figura 2 Validación de persistencia de datos perdidos

17. **Conversión de tipos de datos**, en las columnas seleccionadas convertir a float para acondicionar los datos para los modelos.

18. **Aplicar Label Encoding,** los modelos de machine learning a menudo requieren que las características sean numéricas para la modelización.

19. **Eliminar datos atípicos**, para mejorar la calidad de los datos y hacer que los análisis y modelos sean más precisos. Los valores atípicos extremos pueden sesgar los resultados del análisis estadístico y los modelos predictivos.

3. Identificación y descripción de variables

La base de datos objeto de esta investigación consta de un total de 28 columnas de las que posteriormente se seleccionarán a las variables dependiente e independientes respectivamente.

A continuación, se muestra una matriz con la información de las variables que pertenecen al archivo “train.csv”:

VARIABLE	TIPO	DESCRIPCIÓN
ID	object	Identificación única de cada entrada
Customer_ID	object	Identificación única de una persona
Month	object	Mes del año
Name	object	Nombre de una persona
Age	object	Edad de la persona
SSN	object	N° de seguro social de una persona
Occupation	object	Ocupación de la persona
Annual_Income	object	Ingreso anual de la persona
Monthly_Inhand_Salary	float64	Salario base mensual de una persona
Num_Bank_Accounts	int64	N° de cuentas bancarias que posee una persona
Num_Credit_Card	int64	N° de tarjetas de crédito que posee una persona
Interest_Rate	int64	Tasa de interés de la tarjeta de crédito
Num_of_Loan	object	N° de préstamos tomados en el banco
Type_of_Loan	object	Tipos de préstamos tomados por una persona
Delay_from_due_date	int64	N° prom. de días de retraso desde la fecha de pago
Num_of_Delayed_Payment	object	N° promedio de pagos retrasados por una persona
Changed_Credit_Limit	object	Cambio porcentual en el límite de la tarjeta de crédito
Num_Credit_Inquiries	float64	N° de solicitudes de tarjetas de crédito
Credit_Mix	object	Clasificación del mix de créditos
Outstanding_Debt	object	Deuda restante por pagar (en USD)
Credit_Utilization_Ratio	float64	Índice de utilización de la tarjeta de crédito
Credit_History_Age	object	Edad del historial crediticio de la persona
Payment_of_Min_Amount	object	Si la persona solo pagó el monto mínimo
Total_EMI_per_month	float64	Pagos mensuales EMI Emergencia Médica Integral
Amount_invested_monthly	object	Monto mensual invertido por el cliente (en USD)
Payment_Behaviour	object	Comportamiento de pago del cliente (en USD)
Monthly_Balance	object	Monto del saldo mensual del cliente (en USD)
Credit_Score	object	Rango de puntaje crediticio (Pobre, Estándar, Bueno)

Tabla 6 Matriz de variables de BDD

La naturaleza financiera de los datos contenidos en estas variables es relevante para poder cumplir exitosamente el objetivo general y específicos del proyecto debido a que a partir de los resultados de analizar sus relaciones y dependencias se podrán seleccionar, elaborar y configurar los modelos de clasificación de clientes correctamente.

Para efectos de establecer resultados numéricos del grado de relación y dependencia que existe entre las variables numéricas, se muestra a continuación el resultado del cálculo de la matriz de correlación del dataset:

	Age	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Interest_Rate	Num_of_Loan	Delay_from_due_date	Num_of_Delayed_Payment	...	Credit_Mix	Outstanding_Debt	Credit_Utilization_Ratio	Payment_of_Min_Amount	Total_EMI_per_month	Amount_invested_monthly	Payment_Behaviour	Monthly_Balance	Credit_Score	Credit_History_Age_Months
Age	1	-0	0.02	0.08	-0	-0	-0	-0.2	-0.2	-0.1	...	0.07	-0.2	0.02	-0.2	0	0.06	0.03	0.11	-0.1	0.22
Occupation	-0	1	-0	0	0	0	0	0	0.01	0	...	0.01	0.01	0	0	0	0	0	0	0.01	-0
Annual_Income	0.02	-0	1	0.21	-0	0	0	-0.1	-0.1	-0	...	0.02	-0.1	0.03	-0	0.01	0.1	0.07	0.15	-0	0.06
Monthly_Inhand_Salary	0.08	0	0.21	1	-0	-0	-0	-0.2	-0.2	-0.1	...	0.09	-0.3	0.16	-0.2	0.02	0.45	0.31	0.68	-0.1	0.25
Num_Bank_Accounts	-0	0	-0	-0	1	0.01	0.01	0.07	0.08	0.02	...	-0	0.08	-0	0.05	0	-0	-0	-0.1	0.03	-0.1
Num_Credit_Card	-0	0	0	-0	0.01	1	0	0.03	0.04	0.01	...	-0	0.04	-0	0.02	0	-0	-0	-0	0.01	-0
Interest_Rate	-0	0	0	-0	0.01	0	1	0.05	0.06	0.01	...	-0	0.07	-0	0.04	0	-0	-0	-0	0.01	-0.1
Num_of_Loan	-0.2	0	-0.1	-0.2	0.07	0.03	0.05	1	0.49	0.12	...	-0.4	0.62	-0.1	0.31	0.02	-0.1	-0.1	-0.4	0.08	-0.6
Delay_from_due_date	-0.2	0.01	-0.1	-0.2	0.08	0.04	0.06	0.49	1	0.15	...	-0.4	0.57	-0.1	0.31	0	-0.1	-0.1	-0.3	0.12	-0.5
Num_of_Delayed_Payment	-0.1	0	-0	-0.1	0.02	0.01	0.01	0.12	0.15	1	...	-0.1	0.14	-0	0.1	0	-0.1	-0	-0.1	0.07	-0.1
Changed_Credit_Limit	-0.1	0	-0	-0.2	0.05	0.01	0.04	0.35	0.28	0.09	...	-0.1	0.45	-0	0.31	0	-0.1	-0.1	-0.2	0.21	-0.4
Num_Credit_Inquiries	-0	0	0	-0	0.01	0	0.01	0.06	0.06	0.01	...	-0	0.07	-0	0.04	0	-0	-0	-0	0.01	-0.1
Credit_Mix	0.07	0.01	0.02	0.09	-0	-0	-0	-0.4	-0.4	-0.1	...	1	-0.6	0.03	-0	0	0.01	0.05	0.16	0.22	0.35
Outstanding_Debt	-0.2	0.01	-0.1	-0.3	0.08	0.04	0.07	0.62	0.57	0.14	...	-0.6	1	-0.1	0.3	0	-0.1	-0.1	-0.3	0.06	-0.6
Credit_Utilization_Ratio	0.02	0	0.03	0.16	-0	-0	-0	-0.1	-0.1	-0	...	0.03	-0.1	1	-0	0	0.06	0.11	0.23	-0	0.07
Payment_of_Min_Amount	-0.2	0	-0	-0.2	0.05	0.02	0.04	0.31	0.31	0.1	...	-0	0.3	-0	1	0	-0.1	-0.1	-0.2	0.19	-0.4
Total_EMI_per_month	0	0	0.01	0.02	0	0	0	0.02	0	0	...	0	0	0	0	1	0.01	0.01	0.01	-0	0
Amount_invested_monthly	0.06	0	0.1	0.45	-0	-0	-0	-0.1	-0.1	-0.1	...	0.01	-0.1	0.06	-0.1	0.01	1	0.07	0.25	-0.2	0.14
Payment_Behaviour	0.03	0	0.07	0.31	-0	-0	-0	-0.1	-0.1	-0	...	0.05	-0.1	0.11	-0.1	0.01	0.07	1	0.51	-0	0.09
Monthly_Balance	0.11	0	0.15	0.68	-0.1	-0	-0	-0.4	-0.3	-0.1	...	0.16	-0.3	0.23	-0.2	0.01	0.25	0.51	1	-0.1	0.32
Credit_Score	-0.1	0.01	-0	-0.1	0.03	0.01	0.01	0.08	0.12	0.07	...	0.22	0.06	-0	0.19	-0	-0.2	-0	-0.1	1	-0.1
Credit_History_Age_Months	0.22	-0	0.06	0.25	-0.1	-0	-0.1	-0.6	-0.5	-0.1	...	0.35	-0.6	0.07	-0.4	0	0.14	0.09	0.32	-0.1	1

Tabla 7 Matriz de correlación de variables numéricas

4. Visualización de variables

Luego de eliminar las variables innecesarias y limpiar/transformar las variables restantes, finalmente tenemos las visualizaciones de las variables de interés definitivas que servirán como insumo para el entrenamiento de los modelos las cuales son mostradas a continuación a través de herramientas de visualización de datos:

1. Lista general de variables numéricas de interés definitivas:

	0	1	2	3	4
Age	23	23	5	23	23
Occupation	12	12	12	12	12
Annual_Income	19114.1	19114.1	19114.1	19114.1	19114.1
Monthly_Inhand_Salary	1824.84	1824.84	1824.84	1824.84	1824.84
Num_Bank_Accounts	3	3	3	3	3
Num_Credit_Card	4	4	4	4	4
Interest_Rate	3	3	3	3	3
Num_of_Loan	4	4	4	4	4
Delay_from_due_date	3	-1	3	5	6
Num_of_Delayed_Payment	7	0	7	4	0
Changed_Credit_Limit	11.27	11.27	10.39	6.27	11.27
Num_Credit_Inquiries	4	4	4	4	4
Credit_Mix	1	1	1	1	1
Outstanding_Debt	809.98	809.98	809.98	809.98	809.98
Credit_Utilization_Ratio	26.82	31.94	28.61	31.38	24.8
Payment_of_Min_Amount	1	1	1	1	1
Total_EMI_per_month	49.57	49.57	49.57	49.57	49.57
Amount_invested_monthly	118.28	118.28	118.28	118.28	118.28
Payment_Behaviour	3	2	1	0	4
Monthly_Balance	312.49	284.63	331.21	223.45	341.49
Credit_Score	0	0	0	0	0
Credit_History_Age_Months	265	265	267	268	269

Tabla 8 Resultado final post limpieza y transformación

2. Variable objetivo “Credit_Score” (0:Good, 1:Poor, 2:Standard):

Figura 3 Distribución variable objetivo Credit_Score

3. Mapa de calor de la correlación de las variables de interés:

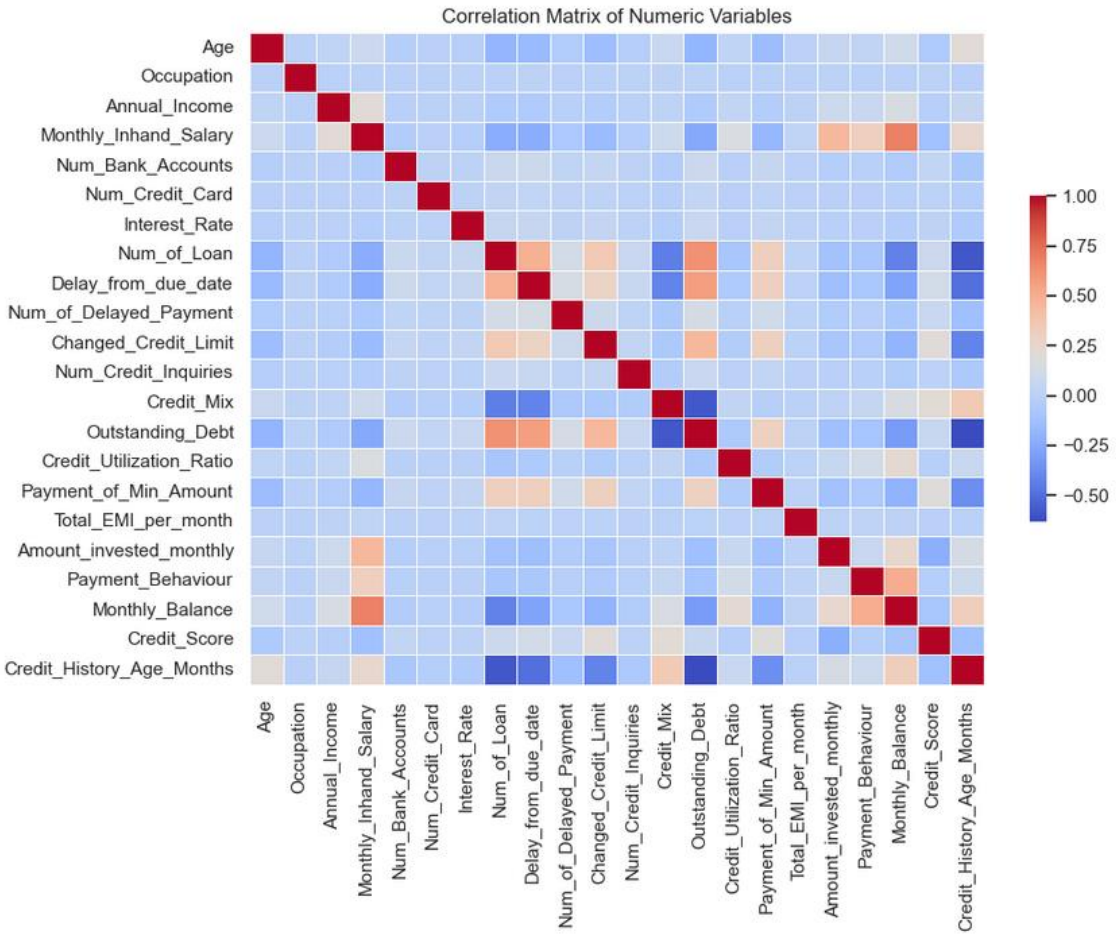


Figura 4 Mapa de calor correlación variables de interés

4. Relaciones entre las columnas numéricas y la columna objetivo:

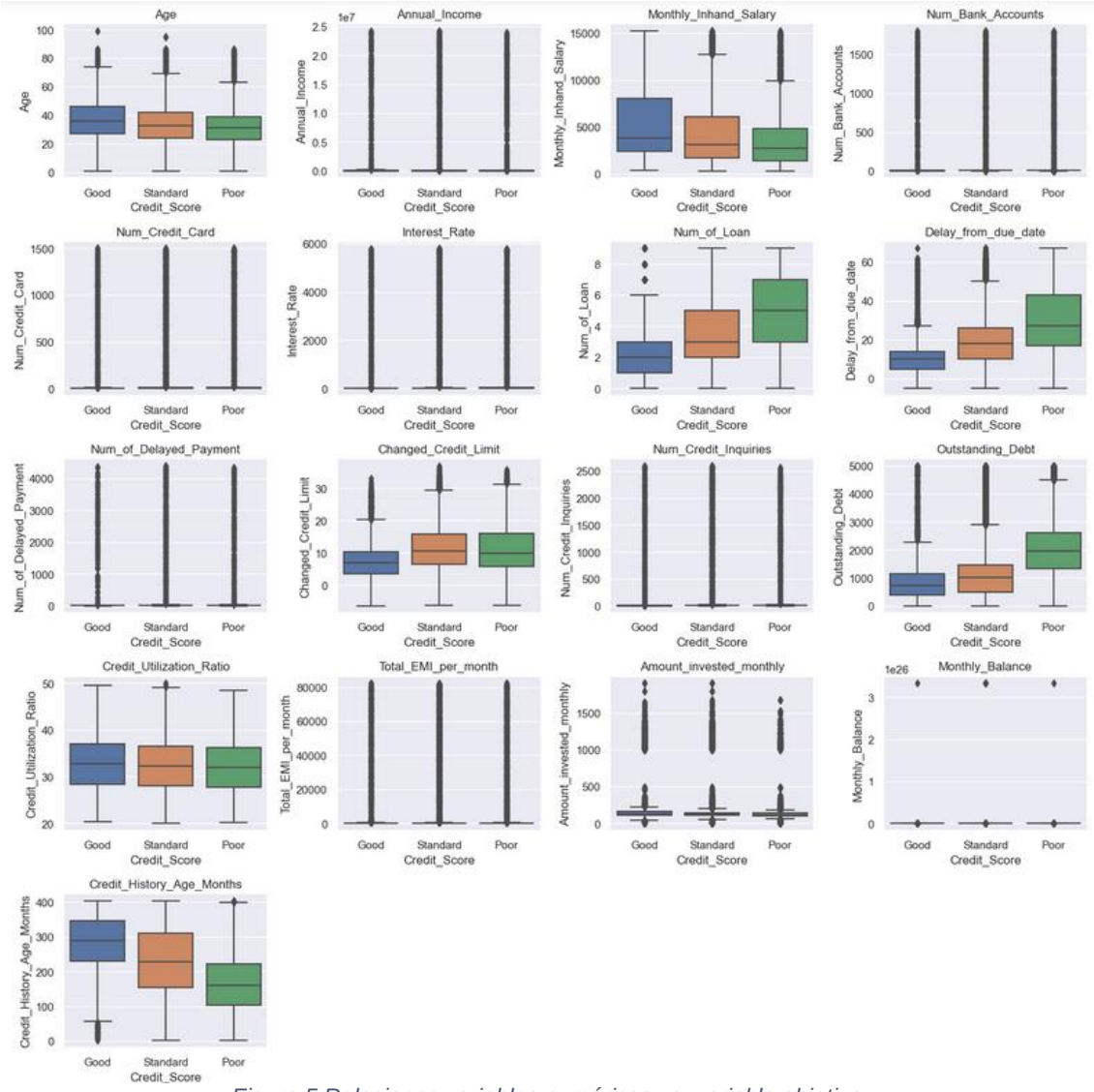


Figura 5 Relaciones variables numéricas vs. variable objetivo

5. Distribución de la variable edades de los clientes:

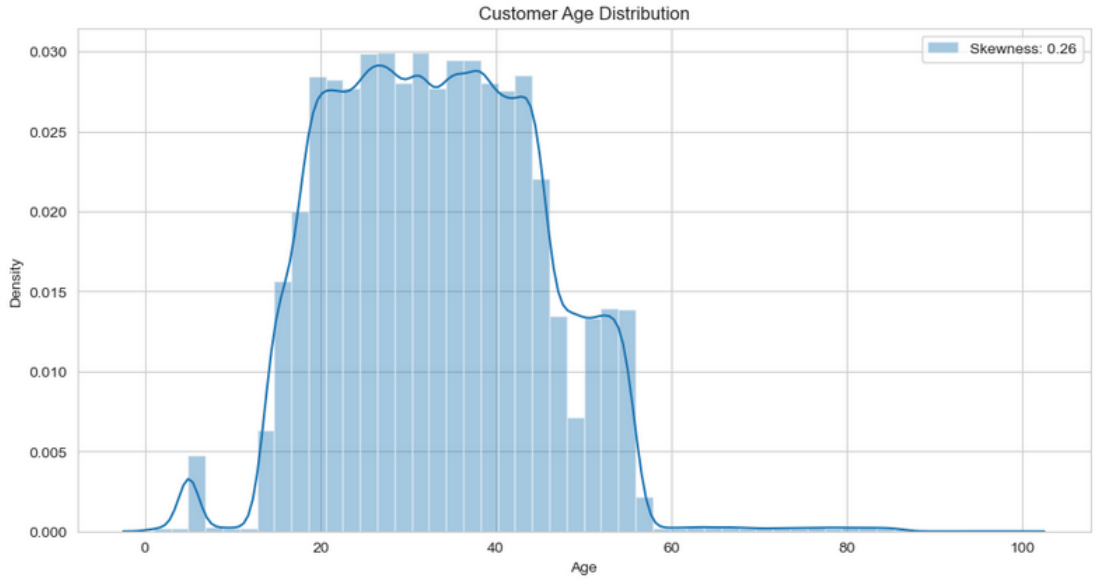


Figura 6 Distribución edades de los clientes

6. Distribución de la variable Monthly_Inhand_Salary de los clientes:

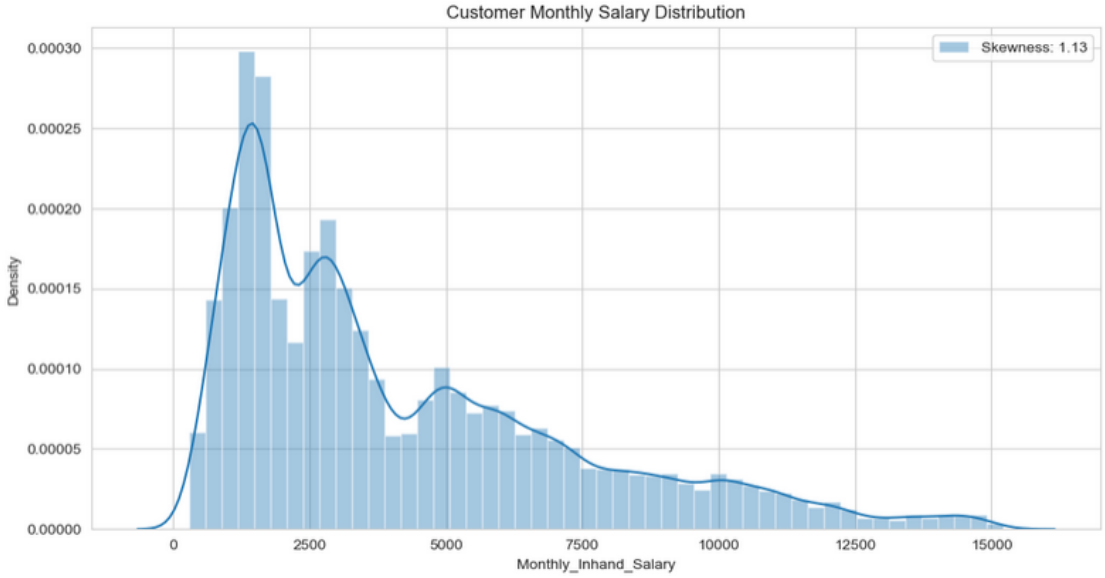


Figura 7 Distribución variable Monthly_Inhand_Salary

7. Distribución de profesiones de los clientes:

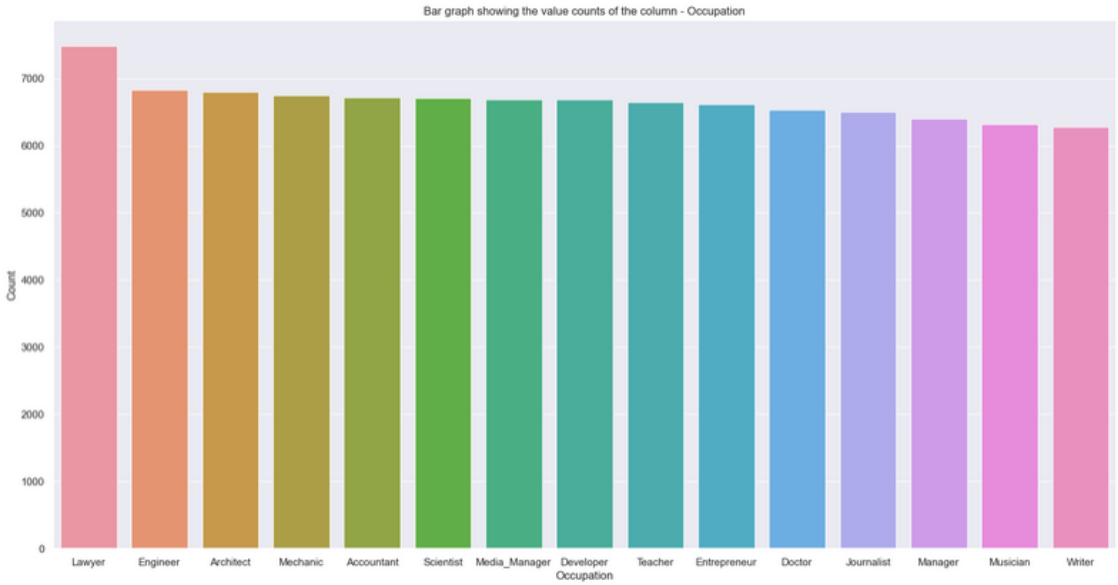


Figura 8 Distribución variable Occupation

8. Distribución de las profesiones de los clientes vs. variable objetivo:

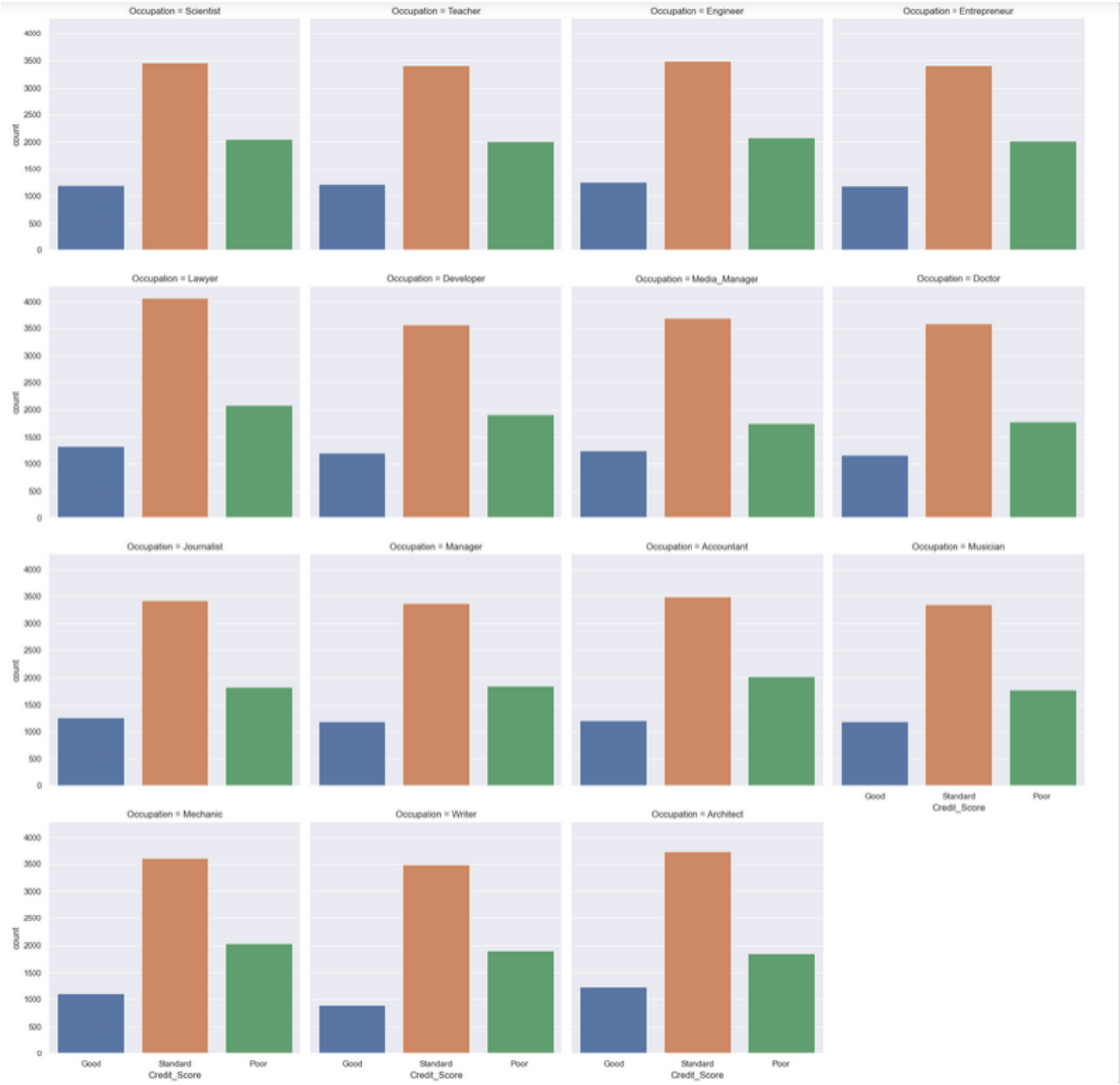


Figura 9 Distribución de profesiones vs. variable objetivo

9. Distribución de clientes que pagan el valor mínimo mensual de tarjeta de crédito:

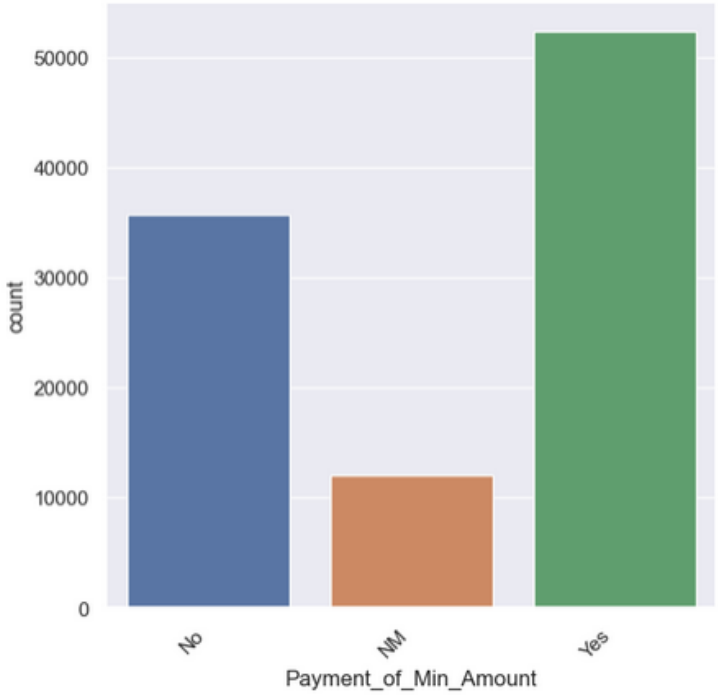


Figura 10 Distribución de clientes que pagan el mínimo de tarjeta de crédito

10. Distribución del comportamiento de pago de los clientes:

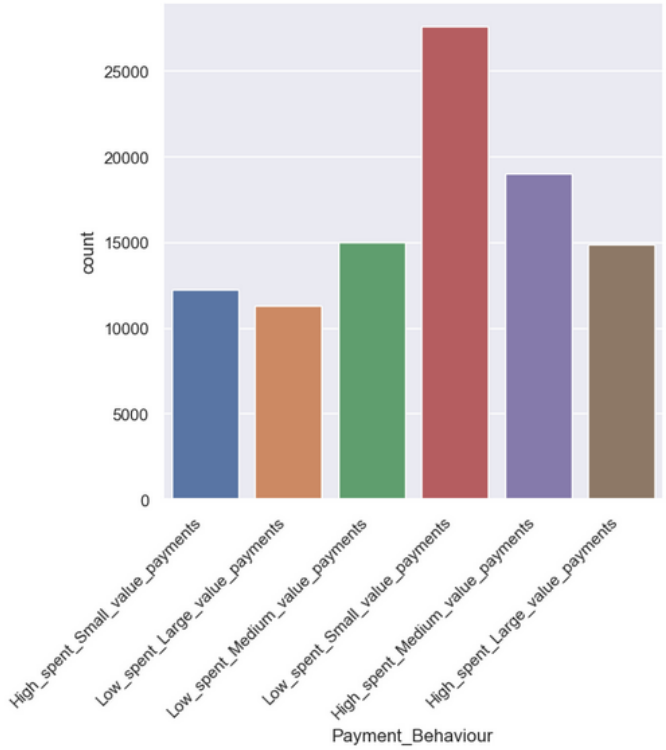


Figura 11 Distribución del comportamiento de pago de los clientes

5. Selección de modelos

Siguiendo la metodología CRISP-DM se ha llegado a la fase del modelado para el cumplimiento del objetivo general y los objetivos específicos propuestos, en este sentido se seleccionaron los siguientes modelos para solucionar el problema de clasificación de clientes que solicitan créditos en instituciones financieras:

1. Modelo de Regresión Logística Multinomial

La regresión logística multinomial es una extensión de la regresión logística que se utiliza cuando la variable dependiente es categórica con más de dos niveles (clases). A diferencia de la regresión logística binaria, que maneja dos clases, la regresión logística multinomial puede manejar tres o más clases.

La fórmula general para la regresión logística multinomial puede expresarse de la siguiente manera:

Para una variable dependiente Y con k categorías (clases), se selecciona una categoría como referencia (generalmente la primera). Para cada una de las otras k−1 categorías, el modelo estima un conjunto de coeficientes.

Supongamos que Y tiene k categorías y X es el conjunto de variables independientes. La probabilidad de que Y tome la clase k-1 se da por:

$$P(Y = j|X) = \frac{e^{\beta j0 + \beta j1X1 + \beta j2X2 + \cdots + \beta jpXp}}{1 + \sum_{m=1}^{k-1} e^{\beta m0 + \beta m1X1 + \beta m2X2 + \cdots + \beta mpXp}}$$

donde:

- j es una de las categorías de Y.
- X = (X1,X2,...,Xp) son las variables independientes.
- βj0 es la intersección (coeficiente constante) para la categoría j.
- βj1,βj2,...,βjp son coeficientes para las variables independientes para la categoría j.

2. Modelo de Random Forest Multiclase

El modelo de Random Forest para clasificación multiclase es una extensión del algoritmo Random Forest tradicional, el cual se basa en la creación de múltiples árboles de decisión y la combinación de sus predicciones para obtener una clasificación más robusta.
A diferencia de la regresión logística multinomial, no hay una única ecuación que describa el modelo de Random Forest, ya que este se basa en un conjunto de árboles de decisión en lugar de una fórmula matemática explícita.

Su funcionamiento se da de la siguiente manera:

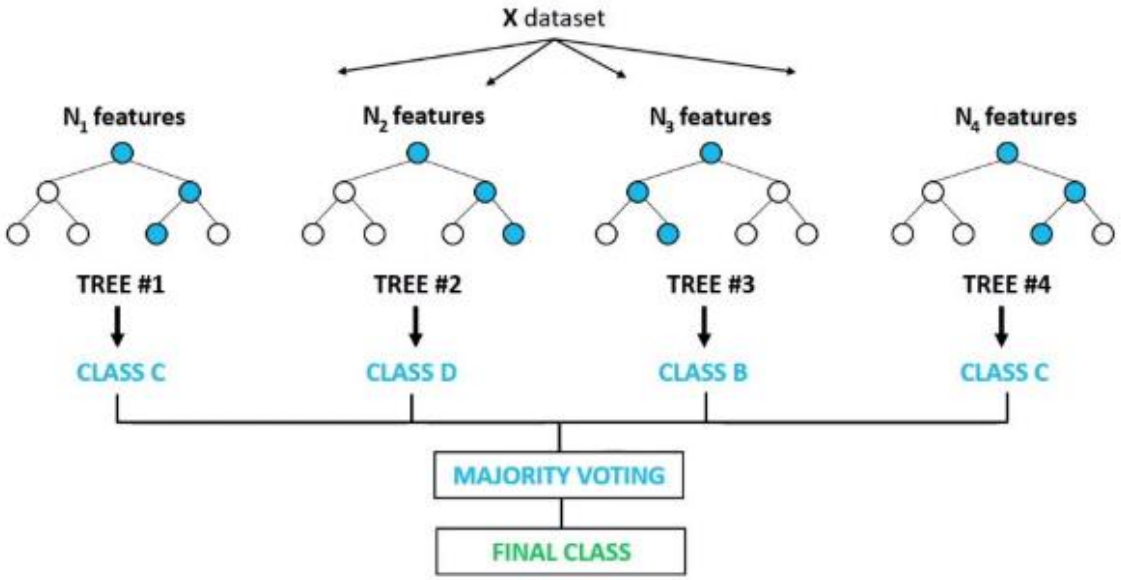


Figura 12 Modelo de Random Forest

1. Construcción del Bosque**:**

Se generan múltiples subconjuntos de datos a partir del conjunto de datos original mediante el muestreo con reemplazo (bootstrap sampling).

Para cada subconjunto, se construye un árbol de decisión. Cada nodo del árbol se divide utilizando el mejor subconjunto aleatorio de las características en lugar de todas las características, lo que introduce más diversidad en el modelo.

2. Predicción**:**

Para predecir la clase de una nueva instancia, cada árbol en el bosque realiza una predicción y la clase final se determina mediante votación mayoritaria entre todas las predicciones de los árboles.

3. Entrenamiento para B iteraciones:

- Se toma una muestra aleatoria (con reemplazo) del conjunto de entrenamiento.
- Se construye un árbol de decisión utilizando esta muestra. Durante la construcción, en cada nodo, se selecciona un subconjunto aleatorio de características y se elige la mejor división entre ellas.
- El resultado es un bosque de B árboles de decisión.

4. Predicción:

Para predecir la clase de un nuevo ejemplo:

- Se pasa el ejemplo por cada uno de los B árboles.
- Se toma la clase que obtuvo más votos (votación mayoritaria) como la predicción final.

3. Modelo de Red Neuronal Multiclase

Las redes neuronales son modelos computacionales inspirados en el cerebro humano, que se utilizan para resolver problemas de clasificación, regresión y otros. En el caso de una red neuronal para clasificación multiclase, la estructura típica consiste en una capa de entrada, una o más capas ocultas y una capa de salida con tantas neuronas como clases haya en el problema.

Para una red neuronal con una capa oculta y una capa de salida, la ecuación se puede expresar de la siguiente manera:

1. Capa de Entrada:

   - Recibe el vector de características $X = (x_1, x_2, \dots, x_n)$

2. Capas ocultas:

   - Una o más capas que procesan la información mediante una combinación lineal de las entradas seguida de una función de activación no lineal.
   - Por ejemplo, en una capa oculta h con m neuronas:

$$h = f(W^{(h)}X + b^{(h)})$$

   Donde:

   - $W^{(h)}$ es la matriz de pesos de la capa h.
   - $b^{(h)}$ es el vector de sesgos de la capa h.
   - f es la función de activación (por ejemplo, ReLu, Sigmoid, Tanh).

3. Capa de Salida:

   - La capa de salida tiene k neuronas (una por cada clase) y utiliza una función de activación Softmax para convertir las salidas en probabilidades.
   - La salida z de la capa anterior (última capa oculta) se convierte en probabilidades:

$$Y = softmax\ (W^{(0)}z + b^{(0)})$$

   Donde:

   - $W^{(0)}$ es la matriz de pesos de la capa de salida.
   - $b^{(0)}$ es el vector de sesgos de la capa de salida.
   - La función softmax se define como:

$$softmax\left(z_j\right) = \frac{e^{z_j}}{\sum_{i=1}^{k} e^{z_i}}$$

RESULTADOS

Análisis de modelos

Para esta etapa, se utilizó el software Anaconda Navigator versión 2.6.1 como entorno de ejecución de Jupyter Notebook versión 6.5.4 y el lenguaje de programación usado fue Python versión 3.11.7.

A continuación, se muestra el detalle de la ejecución del análisis de los datos para cada uno de los tres modelos seleccionados:

1. Ejecución del modelo de regresión logística multinomial

El código fue diseñado para entrenar el modelo utilizando “LogisticRegression” de “scikit-learn” y evaluar su desempeño tanto en el conjunto de datos de prueba como en el de entrenamiento.

1. **Configuración:**

   - **multi_class = ´multinomial´:** indica que el modelo debe manejar múltiples clases en lugar de binario.
   - **solver = ´lbfgs´:** Es un método de optimización eficiente y especifica el algoritmo de optimización utilizado para encontrar los parámetros del modelo.
   - **max_iter = 900:** Establece el número máximo de iteraciones para el algoritmo de optimización.

   Predicciones:

   - **predict(X_test):** Realiza predicciones sobre el conjunto de datos de prueba.
   - **predict(X_train):** Realiza predicciones sobre el conjunto de datos de entrenamiento.

2. **Evaluación:**

   - confusión_matrix(y_true, y_pred): Calcula la matriz de confusión que muestra el número de predicciones correctas e incorrectas para cada clase.

   -**classification_report(y_true, y_pred):** Muestra un informe detallado con precisión, recall, f1-score y soporte para cada clase.

3. **Resultados obtenidos:**

```
RESULTADOS MODELO DE REGRESIÓN LOGÍSTICA MULTINOMIAL:

TEST SET:

MATRIZ DE CONFUSIÓN - CON DATOS DE PRUEBA:

 [[5988  122  913]
 [ 913 4398 1712]
 [1597 1549 3878]]

REPORTE DE CLASIFICACIÓN - CON DATOS DE PRUEBA:

              precision    recall  f1-score   support

           0       0.70      0.85      0.77      7023
           1       0.72      0.63      0.67      7023
           2       0.60      0.55      0.57      7024

    accuracy                           0.68     21070
   macro avg       0.68      0.68      0.67     21070
weighted avg       0.68      0.68      0.67     21070


TRAIN SET:

MATRIZ DE CONFUSIÓN - CON DATOS DE ENTRENAMIENTO:

 [[34022   726  5051]
 [ 5405 24869  9525]
 [ 8661  8690 22447]]

REPORTE DE CLASIFICACIÓN - CON DATOS DE ENTRENAMIENTO:

              precision    recall  f1-score   support

           0       0.71      0.85      0.77     39799
           1       0.73      0.62      0.67     39799
           2       0.61      0.56      0.58     39798

    accuracy                           0.68    119396
   macro avg       0.68      0.68      0.68    119396
weighted avg       0.68      0.68      0.68    119396
```

Figura 13 Resultados - Regresión logística multinomial

2. Ejecución del modelo de random forest multiclase

El código fue diseñado para entrenar el modelo para clasificación multiclase y evaluar su desempeño tanto en el conjunto de datos de prueba como en el de entrenamiento.

1. **Configuración:**

- random_state = 42: Asegura la reproducibilidad del modelo.

- **n_estimators = 900:** Especifica el número de árboles en el bosque. Aumentar el número de árboles generalmente mejora el desempeño, pero también incrementa el tiempo de entrenamiento.

Predicciones:

- **predict(X_test):** Realiza predicciones sobre el conjunto de datos de prueba.

-**predict(X_train):** Realiza predicciones sobre el conjutno de datos de entrenamiento.

2. **Evaluación:**

- confusión_matrix(y_true, y_pred): Calcula la matriz de confusión que muestra el número de predicciones correctas e incorrectas para cada clase.

-**classification_report(y_true, y_pred):** Muestra un informe detallado con precisión, recall, f1-score y soporte para cada clase.

3. Resultados obtenidos:

```
RESULTADOS MODELO DE RANDOM FOREST MULTICLASE:

TEST SET:

MATRIZ DE CONFUSIÓN - CON DATOS DE PRUEBA:

 [[6752    8  263]
 [ 145 6452  426]
 [ 687  923 5414]]

REPORTE DE CLASIFICACIÓN - CON DATOS DE PRUEBA:

              precision    recall  f1-score   support

           0       0.89      0.96      0.92      7023
           1       0.87      0.92      0.90      7023
           2       0.89      0.77      0.82      7024

    accuracy                           0.88     21070
   macro avg       0.88      0.88      0.88     21070
weighted avg       0.88      0.88      0.88     21070


TRAIN SET:

MATRIZ DE CONFUSIÓN - CON DATOS DE ENTRENAMIENTO:

 [[39799     0     0]
 [    0 39799     0]
 [    0     0 39798]]

REPORTE DE CLASIFICACIÓN - CON DATOS DE ENTRENAMIENTO:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00     39799
           1       1.00      1.00      1.00     39799
           2       1.00      1.00      1.00     39798

    accuracy                           1.00    119396
   macro avg       1.00      1.00      1.00    119396
weighted avg       1.00      1.00      1.00    119396
```

Figura 14 Resultados - Random forest multiclase

3. Ejecución del modelo de red neuronal multiclase

El código fue diseñado para entrenar el modelo de clasificación con una red neuronal simple y evaluar su desempeño tanto en el conjunto de datos de prueba como en el de entrenamiento.

1. Configuración:

- **Sequential:** Utilizado para crear un modelo secuencial.

- **Dense:** Capa densa(fully connected layer) con 16, 8 y 3 neuronas respectivamente.
- **input_dim:** Número de características de entrada.
- **activation = 'relu':** Función de activación ReLU.
- **activation = 'softmax':** Función de activación para la capa de salida, adecuada para clasificación multiclase.
- **loss = 'sparse_categorical_crossentropy':** Función de pérdida para clasificación multiclase con etiquetas enteras.
- **optimizer = 'adam':** Optimizador Adam.
- **metrics = ['accuracy']:** Métrica de evaluación de precisión.

2. **Entrenamiento:**
   - **epochs = 900:** Número de épocas para entrenar el modelo.
   - **batch_size = 512:** Número de muestras por lote.
   - **validation_split = 0.1:** Porcentaje de datos de entrenamiento utilizado para validación.
   - **verbose = 1:** Modo de salida detallada.

3. **Predicciones:**
   - **predict(X_test):** Realiza predicciones sobre el conjunto de datos de prueba.
   -**predict(X_train):** Realiza predicciones sobre el conjutno de datos de entrenamiento.

4. **Evaluación:**
   - **confusión_matrix(y_true, y_pred):** Calcula la matriz de confusión que muestra el número de predicciones correctas e incorrectas para cada clase.
   -**classification_report(y_true, y_pred):** Muestra un informe detallado con precisión, recall, f1-score y soporte para cada clase.

5. Resultados obtenidos:

```
RESULTADOS MODELO DE REDES NEURONALES MULTICLASE:

TEST SET:

MATRIZ DE CONFUSIÓN - CON DATOS DE PRUEBA:

 [[6254  141  628]
 [ 863 5444  716]
 [1295 1630 4099]]

REPORTE DE CLASIFICACIÓN - CON DATOS DE PRUEBA:

              precision    recall  f1-score   support

           0       0.74      0.89      0.81      7023
           1       0.75      0.78      0.76      7023
           2       0.75      0.58      0.66      7024

    accuracy                           0.75     21070
   macro avg       0.75      0.75      0.74     21070
weighted avg       0.75      0.75      0.74     21070


TRAIN SET:

MATRIZ DE CONFUSIÓN - CON DATOS DE ENTRENAMIENTO:

 [[35468   865  3466]
 [ 4915 30958  3926]
 [ 7104  9182 23512]]

REPORTE DE CLASIFICACIÓN - CON DATOS DE ENTRENAMIENTO:

              precision    recall  f1-score   support

           0       0.75      0.89      0.81     39799
           1       0.75      0.78      0.77     39799
           2       0.76      0.59      0.67     39798

    accuracy                           0.75    119396
   macro avg       0.75      0.75      0.75    119396
weighted avg       0.75      0.75      0.75    119396
```

Figura 15 Resultados - Red neuronal multiclase

Además, se experimentó con dos ejecuciones adicionales de sendos modelos de redes neuronales **"perceptrón multicapa"** y una **"red neuronal secuencial"** pero esta última configurando hiperparámetros mejorados adicionales para comprobar si los resultados de éstos son más óptimos que la **"red neuronal multiclase"** inicial:

4. Ejecución del modelo de red neuronal perceptrón multicapa

Resultados obtenidos:

```
RESULTADOS MODELO DE RED NEURONAL PERCEPTRÓN MULTICAPA MLP2:

TEST SET:

MATRIZ DE CONFUSIÓN - CON DATOS DE PRUEBA:

 [[6538   95  390]
 [ 223 5807  993]
 [1030 1115 4879]]

REPORTE DE CLASIFICACIÓN - CON DATOS DE PRUEBA:

              precision    recall  f1-score   support

           0       0.84      0.93      0.88      7023
           1       0.83      0.83      0.83      7023
           2       0.78      0.69      0.73      7024

    accuracy                           0.82     21070
   macro avg       0.82      0.82      0.81     21070
weighted avg       0.82      0.82      0.81     21070


TRAIN SET:

MATRIZ DE CONFUSIÓN - CON DATOS DE ENTRENAMIENTO:

 [[39067   132   600]
 [  588 36168  3043]
 [ 3239  3599 32960]]

REPORTE DE CLASIFICACIÓN - CON DATOS DE ENTRENAMIENTO:

              precision    recall  f1-score   support

           0       0.91      0.98      0.94     39799
           1       0.91      0.91      0.91     39799
           2       0.90      0.83      0.86     39798

    accuracy                           0.91    119396
   macro avg       0.91      0.91      0.91    119396
weighted avg       0.91      0.91      0.91    119396
```

Figura 16 Resultados - Red neuronal perceptrón multicapa

5. Ejecución modelo de red neuronal secuencial con hiperparámetros mejorados

Resultados obtenidos:

```
RESULTADOS MODELO DE RED NEURONAL ARTIFICIAL ANN1:

TEST SET:

MATRIZ DE CONFUSIÓN - CON DATOS DE PRUEBA:

 [[6615   39  369]
 [ 109 6232  682]
 [ 813 1224 4987]]

REPORTE DE CLASIFICACIÓN - CON DATOS DE PRUEBA:

              precision    recall  f1-score   support

           0       0.88      0.94      0.91      7023
           1       0.83      0.89      0.86      7023
           2       0.83      0.71      0.76      7024

    accuracy                           0.85     21070
   macro avg       0.85      0.85      0.84     21070
weighted avg       0.85      0.85      0.84     21070


TRAIN SET:

MATRIZ DE CONFUSIÓN - CON DATOS DE ENTRENAMIENTO:

 [[38969    57   773]
 [  129 37300  2370]
 [ 2690  5447 31661]]

REPORTE DE CLASIFICACIÓN - CON DATOS DE ENTRENAMIENTO:

              precision    recall  f1-score   support

           0       0.93      0.98      0.96     39799
           1       0.87      0.94      0.90     39799
           2       0.91      0.80      0.85     39798

    accuracy                           0.90    119396
   macro avg       0.90      0.90      0.90    119396
weighted avg       0.90      0.90      0.90    119396
```

Figura 17 Resultados - Red neuronal secuencial con hiperparámetros mejorados

DISCUSIÓN DE LOS RESULTADOS Y PROPUESTA DE SOLUCIÓN

Luego de enfrentar los diferentes modelos propuestos se obtuvieron los siguientes resultados consolidados:

MODELO	CREDIT SCORE	ACCURACY %	RECALL %	F1 - SCORE %	TOTAL ACCURACY %
REGRESIÓN LOGÍSTICA MULTINOMIAL	0: Good	70	85	77	68
	1: Poor	72	63	67	
	2: Standard	60	55	57	
RANDOM FOREST MULTICLASE	0: Good	89	96	92	**88**
	1: Poor	87	92	90	
	2: Standard	89	77	82	
RED NEURONAL MULTICLASE	0: Good	74	89	81	75
	1: Poor	75	78	76	
	2: Standard	75	58	66	
RED NEURONAL PERCEPTRÓN MULTICAPA	0: Good	84	93	88	82
	0: Good	83	83	83	
	1: Poor	78	69	73	
RED NEURONAL MULTICLASE CON HIPERPARÁMETROS MEJORADOS	2: Standard	88	94	91	**85**
	1: Poor	83	89	86	
	2: Standard	83	71	76	

Tabla 9 Resultados finales consolidados

1. Interpretación de resultados

Según estos resultados, el modelo de Random Forest Multiclase obtuvo el mejor valor de precisión, seguido en segundo lugar por el modelo de Red Neuronal Multiclase configurado con hiperparámetros mejorados.

Cabe mencionar que, según la matriz de resultados mostrada a continuación, en el proceso de entrenamiento de Random Forest se detectó sobreajuste ya que con los datos de entrenamiento se obtuvieron métricas perfectas:

```
RESULTADOS MODELO DE RANDOM FOREST MULTICLASE:

TEST SET:

MATRIZ DE CONFUSIÓN - CON DATOS DE PRUEBA:

 [[6752    8  263]
 [ 145 6452  426]
 [ 687  923 5414]]

REPORTE DE CLASIFICACIÓN - CON DATOS DE PRUEBA:

              precision    recall  f1-score   support

           0       0.89      0.96      0.92      7023
           1       0.87      0.92      0.90      7023
           2       0.89      0.77      0.82      7024

    accuracy                           0.88     21070
   macro avg       0.88      0.88      0.88     21070
weighted avg       0.88      0.88      0.88     21070


TRAIN SET:

MATRIZ DE CONFUSIÓN - CON DATOS DE ENTRENAMIENTO:

 [[39799     0     0]
 [    0 39799     0]
 [    0     0 39798]]

REPORTE DE CLASIFICACIÓN - CON DATOS DE ENTRENAMIENTO:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00     39799
           1       1.00      1.00      1.00     39799
           2       1.00      1.00      1.00     39798

    accuracy                           1.00    119396
   macro avg       1.00      1.00      1.00    119396
weighted avg       1.00      1.00      1.00    119396
```

Figura 18 Sobreajuste - Modelo random forest multiclase

En este sentido, para corregir el sobreajuste del modelo se recomienda a futuro modificar varios hiperparámetros e implementar estrategias y técnicas adicionales como:

- Reducir la profundidad máxima del árbol **(max_depth)** para prevenir que los árboles se ajusten demasiado a los datos de entrenamiento:
  rf_model = RandomForestClassifier(random_state=42, n_estimators=900, **max_depth=10**).
- Aumentar el número mínimo de muestras por hoja **(min_samples_leaf)** para reducir la complejidad del modelo:

rf_model = RandomForestClassifier(random_state=42, n_estimators=900, **min_samples_leaf=4**).

- Aumentar el número mínimo de muestras para dividir un nodo **(min_samples_split)**:
  rf_model = RandomForestClassifier(random_state=42, n_estimators=900, **min_samples_split=10**).
- Reducir el número de características consideradas para dividir en cada nodo **(max_features)** para ayudar a que los árboles sean más diversos:
  rf_model = RandomForestClassifier(random_state=42, n_estimators=900, **max_features='sqrt'**).
- Aumentar el número de árboles en el bosque **(n_estimators)** para mejorar la capacidad del modelo para generalizar, aunque se incrementa el tiempo de entrenamiento:
  rf_model = RandomForestClassifier(random_state=42, **n_estimators=1200**).
- Utilizar **cross-validation** para seleccionar los mejores hiperparámetros y evitar el sobreajuste:
  grid_search = GridSearchCV(estimator=RandomForestClassifier(random_state=42), param_grid=param_grid, **cv=5**, n_jobs=-1, verbose=2).

2. Implicaciones para la organización

Por lo tanto, después de las consideraciones del punto anterior se elige como el mejor candidato a implementarse en la organización al modelo de Red Neuronal Artificial Multiclase configurado con hiperparámetros mejorados para solucionar el problema de clasificación de clientes para la colocación de créditos.

Esto implica que la institución financiera que adopte e implemente esta solución, estará en la capacidad de planificar a nivel gerencial nuevas y mejores estrategias para el Área de Riesgos, específicamente para el producto de Créditos de Consumo.

Gracias a la explotación de esta tecnología la organización incrementará sus niveles de innovación tecnológica lo que genera una clara ventaja competitiva frente al resto de instituciones financieras.

Finalmente, se mencionan algunos beneficios adicionales colaterales a la implementación de esta solución:

- Disminución de la cartera vencida, ya que se otorgarán los créditos solamente a los mejores clientes calculados por el algoritmo de clasificación.
- Aumento de los ingresos por utilidades anuales debido a que se generará un efecto de cascada por referencias positivas de los clientes satisfechos hacia nuevos clientes potenciales.
- Anualmente, los valores asignados como contingencia para cubrir los posibles incumplimientos serán reasignados al presupuesto general de la institución.
- El prestigio de la institución financiera debido al alto índice de recuperación de cartera de créditos igualmente incrementará en la calificación general que es otorgada por entes de control como la Superintendencia de Bancos en el caso de Ecuador.

CONCLUSIONES Y RECOMENDACIONES

- Se cumplieron el objetivo general y específicos propuestos para este proyecto, concluyendo que para solucionar el problema de clasificación de score crediticio las mejores opciones disponibles son los modelos de redes neuronales artificiales y el modelo de random forest.
- Se logró implementar dos modelos adicionales para contrastar los resultados entre redes neuronales.
- Como conclusión general, no existe un modelo ideal para solucionar el problema de análisis de crédito, todo depende entre otros de la calidad y volumen de los datos, del contexto social y económico de los clientes, del país y sus políticas económicas, del tamaño y necesidades de negocio de la institución financiera.
- Se recomienda explorar otras alternativas de modelos y herramientas de software disponibles para evaluar la diferencia en tiempos de análisis y resultados.
- Dependiendo de las políticas de seguridad de la información que maneje la organización, se recomienda el uso de software libre para iniciar en el análisis de datos para luego escalar al software de pago más robusto y con soporte técnico.

REFERENCIAS

Alejandro, P., & Flores, C. (2021). *Credit scoring aplicado a países emergentes*. Universidad de Piura. https://hdl.handle.net/11042/5086

Aplicación De La Regresión Logística En El Análisis Del Riesgo Crediticio - FasterCapital. (2024). Recuperado 8 de junio de 2024, de https://fastercapital.com/es/tema/aplicaci%C3%B3n-de-la-regresi%C3%B3n-log%C3%ADstica-en-el-an%C3%A1lisis-del-riesgo-crediticio.html

Borrero-Tigreros, D., & Bedoya-Leiva, O. F. (2020). Predicción de riesgo crediticio en Colombia usando técnicas de inteligencia artificial. *Revista UIS Ingenierías*, *19*(4), 37-52. https://doi.org/10.18273/REVUIN.V19N4-2020004

Bülbül, D., Hakenes, H., & Lambert, C. (2019). What influences banks' choice of credit risk management practices? Theory and evidence. *Journal of Financial Stability*, *40*, 1-14. https://doi.org/10.1016/j.jfs.2018.11.002

Credit score classification EDA and ANN Model. (2024). Recuperado 20 de julio de 2024, de https://www.kaggle.com/code/demettal/credit-score-classification-eda-and-ann-model

De Computacion, C., Bryan, C., & Andrade, Q. (2024). *Estado del arte de Modelos de IA en asignación de créditos bancarios*. http://dspace.ups.edu.ec/handle/123456789/27861

De Computación, C., De, A., De, S., De, M., Automático, A., La, E. N., De, P., De Crédito En, R., Banca, L. A., & El Ecuador, E. N. (2024). *Análisis de sesgo de modelos de aprendizaje automático en la predicción de riesgo de crédito en la banca en el Ecuador*. http://dspace.ups.edu.ec/handle/123456789/27860

Freire López, J. (2021). *Modelo de clasificación de riesgo crediticio utilizando Random Forest en financiera del Ecuador*. http://localhost:8080/xmlui/handle/123456789/4256

Gloria, A., Guerrero, C., Néstor, V., Gómez, A., Yair, R., Prieto, A., Especialización, A., De Proyectos, G., Aprobado Por Leidy, R. Y., & Restrepo, N. Z. (2020). *DISEÑO DE UN MODELO TEÓRICO DE ANÁLISIS CREDITICIO, USANDO REDES NEURONALES ARTIFICIALES APLICADAS A START-UPS*.

Ingeniería En Software, M. E. (2021). *VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y TRANSFERENCIA DE TECNOLÓGICA CENTRO DE POSGRADOS*.

Martínez Fernández, T. C. (2022). *Comparación de modelos machine learning aplicados al riesgo de crédito.* http://repositorio.udec.cl/jspui/handle/11594/9846

Oswaldo, C., Acaro, M., Alexander, F., & Cóndor, A. (2019). *Universidad Andina Simón Bolívar Sede Ecuador Área de Gestión Credit scoring, aplicando técnicas de regresión logística y redes neuronales, para una cartera de microcrédito*.

Támara-Ayús, A. L., Vargas-Ramírez, H., Cuartas, J. J., Chica-Arrieta, I. E., Támara-Ayús, A. L., Vargas-Ramírez, H., Cuartas, J. J., & Chica-Arrieta, I.

E. (2019). Regresión logística y redes neuronales como herramientas para realizar un modelo Scoring. *Revista Lasallista de Investigación*, *16*(1), 187-200. https://doi.org/10.22507/RLI.V16N1A5

ANEXOS

Tabla 10 Matriz de investigaciones similares

Autor	Objetivo de la investigación	Problema abordado	Fuente de datos utilizados	Metodología implementada	Resultados	Implicaciones
Támara-Ayús et al. (2019)	Confrontar el poder de pronosticación de dos modelos *scoring* obtenidos a través de regresión logística y red neuronal.	El propósito de esta investigación es analizar el riesgo crediticio de una institución financiera no vigilada por la Superintendencia Financiera de Colombia en torno a un modelo *scoring* que permita determinar el incumplimiento de los clientes correspondiente a su cartera de consumo.	Los modelos se desarrollan con base en una muestra de 43.086 obligaciones correspondiente a una cartera de consumo.	Utilizando las técnicas estadísticas de regresión logística y red neuronal.	Para ambos modelos se logra una precisión del 71% en la base de entrenamiento y del 72 % en la base de comprobación, sin embargo, a pesar de obtener resultados similares, la regresión logística arrojó la menor tasa de malos en la zona de aceptación.	Las dos técnicas utilizadas son adecuadas para el estudio y predicción de la probabilidad de incumplimiento de un cliente correspondiente a una cartera de consumo, lo anterior, respaldado por el alto índice de eficacia predictiva en ambos modelos.
Oswaldo et al. (2019)	Comparar la metodología de credit scoring de regresión logística y redes neuronales en una cartera de microcrédito, de una institución financiera ecuatoriana.	Esta investigación pretende contrastar la hipótesis de sí el uso de redes neuronales, para la modelización del credit scoring de una cartera de microcrédito, logra un mejor performance que utilizar una metodología de regresión logística.	Para esto se hace uso de la información de una cartera del producto de microcrédito, proporcionada por una institución financiera ecuatoriana de mediano tamaño, desembolsada entre los periodos de enero 2014 a diciembre 2017. La información corresponde al momento del	Utilizando las técnicas estadísticas de regresión logística y red neuronal.	El modelo de redes neuronales se ajusta de mejor forma a los datos, pues se obtiene un criterio de información Akaike menor al de regresión logística, con una diferencia de 8.029,2 puntos. De igual forma, los estadísticos KS, coeficiente de Gini y curva ROC evidencia que hacer uso de las redes neuronales logra una mejor clasificación	Los resultados obtenidos evidencian que la metodología de redes neuronales proporciona un modelo scoring más robusto que al usar una regresión logística, pudiendo corroborar que la hipótesis planteada es verdadera, bajo el proceso de modelización empleado.

			desembolso de los créditos, por lo cual se construye modelos scoring de originación.		de los clientes, con 5,19, 5,84 y 2,92 puntos porcentuales por encima de los estadísticos del modelo de regresión logística, respectivamente. Finalmente, la matriz de confusión muestra un menor error con el modelo de redes neuronales, al compararlos con un mismo punto de corte óptimo.	
Alejandro & Flores (2021)	Analizar los distintos enfoques realizados para la aplicación de credit scoring para economías emergentes.	La literatura coincide en apuntar que el mayor desafío que se debe enfrentar para el desarrollo y masificación de estos modelos en economías como la nuestra es una limitación en la cantidad y calidad de los datos a los que las entidades pueden tener acceso.		Modelos Probit y Logit, Arboles de decisión, Random forest y Redes neuronales.	Se ha encontrado evidencia de que la pobre medición sobre los beneficios y ventajas de la implementación de estos sistemas sumado a la deficiencia en la capacidad técnica en la mayoría de las instituciones participantes y la alta concentración de mercado acarrea costos adicionales a las entidades de menor tamaño para el desarrollo de dichos modelos	Para lograr replicar el éxito que tienen los modelos de Credit Scoring en los países desarrollados es necesario reunir suficiente información y que ésta sea relevante y concreta, mientras que en los países emergentes mucha de esta información suele ser cualitativa e informal.
Freire López (2021)	Construir un modelo de clasificación de riesgo crediticio	Esta investigación, se enfoca en Insotec, organización que se	En la creación de modelo de clasificación se utiliza la base de	Modelo Random Forest.	El modelo es implementado con Random Forest, el	La creación del modelo ayuda a contar con una herramienta adicional

	utilizando Random Forest en financiera del Ecuador	encuentra potenciando los procedimientos de sus áreas, incluyendo la evaluación crediticia que actualmente se realiza manualmente, este procedimiento analiza caso por caso y se aprueban las solicitudes dependiendo de la calificación, esto genera muchos inconvenientes como alta carga operativa y probabilidad de cometer errores en las evaluaciones.	datos de clientes de crédito de la organización con un histórico desde el año 2017 al 2020, este conjunto de datos cuenta con 18 variables y 63.896 registros.		cual arroja una precisión mayor al 97%, el porcentaje de error es del 2,8%, con el 2,1% falsos positivos y 11,1% falsos negativos para predecir.	que sirve para clasificar automáticamente a los clientes como “buenos” y “malos” pagadores, lo que puede ser utilizado para entregar créditos con más rapidez y con un menor grado de riesgo. Sin embargo, este modelo necesita ser desplegado como lo indica la metodología CRISP-DM, para que el conocimiento obtenido sea aprovechado por el cliente.
Ingeniería En Software (2021)	Desarrollar un modelo predictivo para la evaluación del riesgo crediticio en la Cooperativa de Ahorro y Crédito Virgen del Cisne.	Cómo el modelo de predicción evaluará el riesgo crediticio en la Cooperativa de Ahorro y Crédito Virgen del Cisne.	BDD transaccional de riesgos Cooperativa Ahorro y Crédito Virgen del Cisne.	Metodología CRISP-DM utilizando los modelos de Regresión Logística, Random Forest y Red Neuronal.	El modelo de Random Forest utilizando el módulo de XGBoots, es el más acertado para predecir si un socio es o no sujeto a crédito.	
Martínez Fernández (2022)	Estudiar e implementar algoritmos de aprendizaje supervisado de clasificación que nos permitan determinar si un cliente entrará o no en incumplimiento.	Los estudios no obtienen el mismo modelo con mejor desempeño, esto se debe a que los algoritmos de Machine Learning dependen en gran medida de los datos con los que se entrenan, por lo que no se puede definir un mejor modelo globalmente.	Para implementar los algoritmos se utilizó un subconjunto de la base de datos del libro IFRS 9 and CECL Credit Risk Modelling and Validation que contiene la información de la cartera de n = 18135 clientes.	Regresión logística, Análisis discriminante lineal, Árboles de decición, Random Forest, Gradient Boosting, Extreme Gradient Boosting y Support Vector Machines.	Las metodologías más adecuadas para predecir la probabilidad de incumplimiento corresponden a la Regresión logística y Gradient Boosting, sin embargo, la elección del mejor modelo depende principlamente del punto de vista del negocio	Un problema de los modelos de calificación crediticia que debe enfatizarse es la falta de disponibilidad de datos crediticios del mundo real, ya que los datos crediticios de los clientes son confidenciales en la mayoría de las instituciones financieras.

					tomando en consideración las ventajas y desventajas de ambos algoritmos.	
Borrero-Tigreros & Bedoya-Leiva (2020)	Proponer modelos para la predicción de riesgo crediticio en Colombia utilizando diferentes técnicas de inteligencia artificial.	Se aborda el problema de riesgo de no pago.	Para la selección de los datos se cuenta con una base de datos relacional anónima la cual hace parte del ciclo de vida de los créditos registrados en una entidad financiera en Colombia.	En particular, se proponen modelos basados en tres técnicas de aprendizaje supervisado (redes neuronales, árboles de decisión y máquinas de soporte vectorial)	De acuerdo con los resultados obtenidos, los árboles de decisión resultan ser más exactos que las otras técnicas utilizadas para la predicción de riesgo crediticio con un área bajo la curva ROC de 88.29%. Los modelos propuestos alcanzan exactitudes similares y en algunos casos superan las exactitudes reportadas en algunos trabajos del estado del arte.	
Gloria et al. (2020)	Proponer un diseño teórico de modelo riesgo crediticio aplicado a Redes Neuronales Artificiales, enfocado en Start-ups (CleanTech).	¿De qué forma proponer el diseño teórico de un modelo de análisis crediticio usando redes neuronales, aplicadas a Start-ups?	Dadas las condiciones mundiales actuales causadas por el COVID-19, se parte una población en el sector empresarial de energías renovables particularmente en las Start-up. Dado que en Colombia no hay la suficiente representatividad de datos, la muestra se basará en 9 empresas europeas;	Perceptrón Simple y Perceptrón Multicapa.	El perceptrón multicapa es una RNA capaz de resolver el problema de clasificación que se presenta cuando una entidad financiera realiza un análisis de riesgo crediticio porque se adapta al problema de clasificación. Concluyendo que este tipo de red es capaz de llevar a cabo con resultados a un nivel de	

			considerando que es un estudio teórico, este no tendrá un muestreo o procedimiento aplicado, sino más bien búsqueda en la literatura existente y accesible.		confianza del 94% en el análisis de riesgo crediticio.	
De Computación et al. (2024)	Analizar el sesgo de modelos de aprendizaje automático para la predicción de riesgo de crédito en la Banca del Ecuador.	Predecir el potencial de sesgo en los algoritmos utilizados para evaluar la confiabilidad y sensibilidad de los datos necesarios en el proceso de calificación crediticia.	En primer lugar, se procedió a realizar encuestas en la ciudad de Guayaquil con el fin de obtener datos reales de personas que solicitaron crédito. Una vez recopilados los datos de las encuestas, se procedió a utilizar la base de datos real como punto de partida para generar datos ficticios adicionales. Estos datos ficticios se crearon de para simular la información histórica de instituciones financieras en Ecuador. La combinación de estos enfoques de recopilación de datos permitió obtener un dataset “datosCredito” más amplio y diverso.	Regresión Logística y Redes Neuronales.	Algoritmo de regresión logística presenta un rendimiento moderado, con un desempeño del 68.70% y una tasa de error del 31.28%. La sensibilidad es del 65.48% y su especificidad es del 69.19%. Mientras que la precisión y el valor predictivo negativo del modelo son similares, con un 68.92% y un 66.58% respectivamente. El algoritmo de redes neuronales tiene un desempeño del 72.36% y su tasa de error es del 27.64%. En cuanto a la sensibilidad del modelo, se encuentra en un 91.56%, lo que indica que tiene la capacidad de identificar la gran mayoría de los casos	

					positivos. Sin embargo, la especificidad del modelo es baja, situada en un 5.36%. En cuanto a la precisión, se encuentra en un 73.89%, lo que indica que el modelo tiene una alta proporción de predicciones positivas. 27 Por otro lado, el valor predictivo negativo del modelo se sitúa en un 32.54%.	

13.- AGE: LIMPIAR CARACTERES EXTRAÑOS Y TRANSFORMAR EDADES FUERA DE RANGOS LÓGICOS

Este código está diseñado para limpiar y convertir una columna de edades en un DataFrame df. Limpia los valores de la columna Age eliminando caracteres especiales y luego convierte los valores a enteros, manejando los casos en los que la conversión no es posible devolviendo None.

```
# La función clean_age toma un valor age y trata de convertirlo a un entero
def clean_age(age):
    try:
        return int(age) # Si la conversión es exitosa (try), devuelve el valor convertido a entero
    except ValueError:
        return None     # Si ocurre un ValueError (es decir, si el valor no puede convertirse a un entero), devuelve None.

# Limpieza y conversión de la columna Age en el DataFrame df:
df['Age'] = df['Age'].str.replace('_', '').str.replace('-', '')
df['Age'] = df['Age'].apply(clean_age)
```

Este código define una función que trunca los dos últimos dígitos de la edad si la edad es mayor que 99, y luego aplica esta función a la columna Age de un DataFrame df. Trunca los dos últimos dígitos de las edades que son mayores que 99, dejando las demás edades sin cambios.

```
def truncate_last_two_digits(age):
    if age > 99:
        return age // 100
    else:
        return age


df['Age'] = df['Age'].apply(truncate_last_two_digits)

df.Age
```

```
0        23
1        23
2         5
3        23
4        23
        ..
99995    25
99996    25
99997    25
99998    25
99999    25
Name: Age, Length: 100000, dtype: int64
```

14.- ANNUAL_INCOME: LIMPIAR CARACTERES EXTRAÑOS Y CONVERTIR DE TIPO OBJECT A FLOAT

Este código limpia la columna Annual_Income eliminando los guiones bajos finales, convierte los valores a tipo float y obtiene los valores únicos en la columna.

```
def remove_trailing_dash(value):
    if isinstance(value, str) and value.endswith('_'):
        return value[:-1]
    else:
        return value


df['Annual_Income'] = df['Annual_Income'].apply(remove_trailing_dash)

df['Annual_Income'] = df['Annual_Income'].astype(float)

df['Annual_Income'].unique()
```

```
array([ 19114.12,  34847.84, 143162.64, ...,  37188.1 ,  20002.88,
        39628.99])
```

15.- MONTHLY_INHAND_SALARY: AGRUPAR POR CADA CLIENTE Y COMPLETAR VACÍOS CON LA MODA DE C/U

Este código tiene como objetivo llenar los valores faltantes (NaN) en la columna Monthly_Inhand_Salary con la moda de los salarios mensuales en mano (Monthly_Inhand_Salary) para cada Customer_ID.

```
# Agrupa el DataFrame df por la columna Customer_ID y selecciona la columna Monthly_Inhand_Salary
# Aplica la función transform para calcular la moda de Monthly_Inhand_Salary para cada grupo (Customer_ID)
# La función lambda x: x.mode().iloc[0] calcula la moda de x
# y selecciona el primer valor de la moda en caso de múltiples modas

Customer_Mode_Salary = df.groupby('Customer_ID')['Monthly_Inhand_Salary'].transform(lambda x : x.mode().iloc[0])
df['Monthly_Inhand_Salary'] = np.where(df['Monthly_Inhand_Salary'].isnull(),Customer_Mode_Salary,df['Monthly_Inhand_Salary'])
```

16.- OCCUPATION: LIMPIAR CARACTERES EXTRAÑOS Y REEMPLAZAR VACÍOS CON LA MODA DE LAS OCUPACIONES DE CADA CLIENTE BASADOS EN SU NÚMERO DE SEGURIDAD SOCIAL SSN

Este código tiene como objetivo llenar los valores faltantes en la columna Occupation del DataFrame df, donde los valores faltantes se indican con '_______', usando el valor más común para cada número de seguro social (SSN).

Limpia la columna Occupation reemplazando los valores '_______' con la moda de Occupation para cada SSN, asegurando que los valores faltantes se completen con la información más frecuente disponible.

```
def fill_occupation_by_ssn(df):
    # Replace '_______' values ••in 'Occupation' column with NaN (empty) values
    df['Occupation'] = df['Occupation'].replace('_______', np.nan)

    # Find the most recurring 'Occupation' values ••for each SNN number
    most_common_occupation_by_ssn = df.groupby('SSN')['Occupation'].apply(lambda x: x.mode().iloc[0])

    # 'Populating '_______' values ••in 'Occupation' column
    for index, row in df.iterrows():
        if pd.isnull(row['Occupation']) and row['SSN'] in most_common_occupation_by_ssn:
            df.at[index, 'Occupation'] = most_common_occupation_by_ssn[row['SSN']]


fill_occupation_by_ssn(df)
```

```
occupation_count = df['Occupation'].value_counts()
occupation_count
```

```
Occupation
Lawyer           7489
Engineer         6837
Architect        6806
Mechanic         6752
Accountant       6717
Scientist        6713
Media_Manager    6689
Developer        6687
Teacher          6646
Entrepreneur     6621
Doctor           6537
Journalist       6502
Manager          6402
Musician         6322
Writer           6280
Name: count, dtype: int64
```

17.- NUM_OF_LOAN: LIMPIAR CARACTERES EXTRAÑOS Y REDUCIR EL NÚMERO DE CRÉDITOS A VALORES MÁS REALES

Este código tiene como objetivo limpiar y procesar la columna Num_of_Loan en el DataFrame df. Primero, limpia los caracteres no deseados y convierte los valores específicos, luego maneja los valores faltantes.

En resumen, este código limpia la columna Num_of_Loan eliminando caracteres no deseados, convirtiendo valores específicos a NaN, y luego reemplaza los valores faltantes con el valor más común, lo que facilita el análisis y la modelización de datos.

```
# num.strip("-_"): Elimina los caracteres "-" y "_" de los extremos de la cadena num

# if num == "100": Si el valor después de eliminar los caracteres es "100"
# lo convierte en NaN. Esto es útil si "100" es un marcador de datos faltantes o erróneos.

# elif len(num) > 1: Si el valor restante tiene más de un carácter, devuelve el primer carácter
# Esto puede ser útil si los valores en Num_of_Loan deben ser simplificados a un solo dígito.

# else: Si el valor tiene un solo carácter, se devuelve tal cual.

def clean_num(num):
    num = num.strip("-_")
    if num == "100":
        return np.nan
    elif len(num) > 1:
        return num[0]
    else:
        return num


df["Num_of_Loan"] = df["Num_of_Loan"].apply(clean_num)

# most_common_value = df["Num_of_Loan"].mode()[0]: Calcula la moda (valor más frecuente) de la columna Num_of_Loan
# y selecciona el primer valor de la moda.

# df["Num_of_Loan"].fillna(most_common_value): Rellena los valores faltantes (NaN) en Num_of_Loan
# con el valor más común calculado.

most_common_value = df["Num_of_Loan"].mode()[0]
df["Num_of_Loan"] = df["Num_of_Loan"].fillna(most_common_value)
```

```
df.Num_of_Loan.value_counts()
```

```
Num_of_Loan
3    19016
2    15076
4    14776
0    10930
1    10800
6     7839
7     7368
5     7231
9     3736
8     3228
Name: count, dtype: int64
```

19.- NUM_OF_DELAYED_PAYMENT: LLENAR LOS VALORES VACÍOS CON 0 Y LIMPIAR CARACTERES EXTRAÑOS

```
df['Num_of_Delayed_Payment'] = df['Num_of_Delayed_Payment'].fillna('0')

def remove_special_characters(value):
    if isinstance(value, str):

        value = value.strip('_').strip('-')
    return value

df['Num_of_Delayed_Payment'] = df['Num_of_Delayed_Payment'].apply(remove_special_characters)
```

20.- CHANGED_CREDIT_LIMIT: LIMPIAR CARACTERES EXTRAÑOS, LUEGO CAMBIAR A TIPO NUMÉRICO Y COLOCAR VALOR PROMEDIO EN VACÍOS

pd.to_numeric(): Función de pandas que convierte una serie o columna a tipo numérico (como int o float).

errors='coerce': Parámetro que especifica qué hacer con los valores que no se pueden convertir a numérico.

errors='coerce' convierte estos valores no válidos en NaN (valores faltantes). Esto es útil para manejar datos sucios o mal formateados.

```
df['Changed_Credit_Limit'] = df['Changed_Credit_Limit'].replace('-', np.nan)
df['Changed_Credit_Limit'] = df['Changed_Credit_Limit'].replace('_', np.nan)


df['Changed_Credit_Limit'] = pd.to_numeric(df['Changed_Credit_Limit'], errors='coerce')


mean_value = df['Changed_Credit_Limit'].mean()


df['Changed_Credit_Limit'].fillna(mean_value, inplace=True)
```

21.- NUM_CREDIT_INQUIRIES: IDENTIFICAR VALORES ÚNICOS Y REEMPLAZAR VACÍOS POR CEROS

```
df['Num_Credit_Inquiries'].unique()
```

```
array([   4.,    2.,    3., ..., 1361.,  310.,   74.])
```

```
df['Num_Credit_Inquiries'].fillna(0, inplace=True)
```

22.- CREDIT_MIX: LIMPIAR CARACTERES EXTRAÑOS COLOCANDO NULL, CALCULAR MODA POR GRUPO Y MODA GLOBAL PARA LUEGO REEMPLAZAR LOS VALORES NULL POR MODA GRUPAL O GLOBAL SEGÚN EL CASO NECESARIO

```
credit_mix_count = df['Credit_Mix'].value_counts()
credit_mix_count
```

```
Credit_Mix
Standard    36479
Good        24337
_           20195
Bad         18989
Name: count, dtype: int64
```

Este código está diseñado para limpiar y rellenar valores faltantes en columnas categóricas del DataFrame df, específicamente para la columna Credit_Mix.

Los valores NaN en Credit_Mix se han rellenado con la moda por Customer_ID, y si aún hay NaN, se rellenan con la moda global de la columna.

En resumen, este código limpia y completa los valores faltantes en las columnas categóricas del DataFrame, asegurando que la columna Credit_Mix tenga datos completos y listos para el análisis.

```
df['Credit_Mix'] = df['Credit_Mix'].replace('_', np.nan)

def fill_na_cat(data, val):

    for col in data.select_dtypes(include='object').columns:
        mode_by_customer = data.groupby('Customer_ID')[col].transform(lambda x: x.mode()[0] if not x.mode().empty else np.nan)
        mode_global = data[col].mode()[0]
        data[col] = data[col].fillna(mode_by_customer.fillna(mode_global))
    return data

df = fill_na_cat(data=df, val="Credit_Mix")
```

```
credit_mix_count = df['Credit_Mix'].value_counts()
credit_mix_count
```

```
Credit_Mix
Standard    45848
Good        30384
Bad         23768
Name: count, dtype: int64
```

23.- CREDIT_HISTORY_AGE: UNIFICAR LOS DATOS DE AÑOS Y MESES A SOLAMENTE MESES, LUEGO ELIMINAR LA COLUMNA ORIGINAL

Este código convierte una columna que representa la edad del historial de crédito en años y meses a una representación total en meses. La función parse_years_and_months_to_months es usada para realizar esta conversión y luego se aplica a la columna Credit_History_Age del DataFrame df.

Con esta función se logra:

- Consistencia: Facilita el análisis al tener una sola unidad de medida (meses) para la edad del historial de crédito.
- Facilita cálculos: Permite realizar cálculos y análisis adicionales de manera más sencilla al trabajar con una sola métrica.

En resumen, este código transforma las descripciones de edad en años y meses a una cifra total en meses, lo que facilita su uso en análisis y modelización de datos.

```
def parse_years_and_months_to_months(age):
    if isinstance(age, str):
        age_parts = age.split(' Years and ')
        years = int(age_parts[0]) if 'Years' in age else 0
        months_str = age_parts[1].split(' Months')[0] if 'Months' in age_parts[1] else '0'
        months = int(months_str)
        total_months = years * 12 + months
        return total_months
    else:
        return 0


df['Credit_History_Age_Months'] = df['Credit_History_Age'].apply(parse_years_and_months_to_months)
```

```
df.drop(columns=['Credit_History_Age'], inplace=True)
```

```
df['Credit_History_Age_Months']
```

```
0        265
1        265
2        267
3        268
4        269
        ...
99995    378
99996    379
99997    380
99998    381
99999    382
Name: Credit_History_Age_Months, Length: 100000, dtype: int64
```

24.- AMOUNT_INVESTED_MONTHLY: ELIMINAR LOS CARACTERES EXTRAÑOS Y REEMPLAZAR CON LOS VALORES DE LA MODA DE CADA GRUPO DE CLIENTE

```
df['Amount_invested_monthly'] = df['Amount_invested_monthly'].replace(
    '__10000__', np.nan)

df['Amount_invested_monthly'] = df.groupby(
    'Customer_ID')['Amount_invested_monthly'].transform(
        lambda x: x.mode()[0] if not x.mode().empty else np.NaN)
```

```
df['Amount_invested_monthly'].isnull().sum()
```

```
0
```

25.- MONTHLY_BALANCE: ELIMINAR LOS CARACTERES EXTRAÑOS, TRANSFORMAR EL TIPO DE DATOS A FLOAT Y REEMPLAZAR LOS VALORES NULOS CON LA MEDIA

Beneficio de las transformaciones:

- Consistencia: Asegura que los valores estén en un formato numérico limpio y consistente.
- Preparación para análisis: Facilita la conversión de la columna a tipo numérico para análisis y modelización.

En resumen, este código prepara la columna Monthly_Balance para una conversión a valores numéricos limpios al eliminar caracteres no deseados y formatear los datos adecuadamente.

```
df['Monthly_Balance'] = df['Monthly_Balance'].astype(str)

df['Monthly_Balance'] = df['Monthly_Balance'].str.replace(r'[^0-9.-]+', '').str.replace('_', '').str.replace('-', '')
```

Este código convierte los valores de Monthly_Balance en valores flotantes, calcula el valor medio y utiliza ese valor para rellenar los valores faltantes en la columna. Esto asegura que la columna esté lista para el análisis y modelización.

```
df['Monthly_Balance'] = df['Monthly_Balance'].astype(float)

mean_value = df['Monthly_Balance'].mean()

df['Monthly_Balance'].fillna(mean_value, inplace=True)
```

```
df['Monthly_Balance'].value_counts()
```

```
Monthly_Balance
333333333333333314856026112.00    9
239.60                            6
123.09                            5
346.64                            5
192.66                            5
                                 ..
571.31                            1
510.54                            1
658.94                            1
1072.87                           1
393.67                            1
Name: count, Length: 98790, dtype: int64
```

26.- PAYMENT_BEHAVIOUR: ELIMINAR LOS CARACTERES EXTRAÑOS Y REEMPLAZAR LOS VALORES NULOS CON LOS VALORES NO NULOS INMEDIATOS ANTERIORES DE CADA DATO

```
df['Payment_Behaviour'].value_counts()
```

```
Payment_Behaviour
Low_spent_Small_value_payments      25513
High_spent_Medium_value_payments    17540
Low_spent_Medium_value_payments     13861
High_spent_Large_value_payments     13721
High_spent_Small_value_payments     11340
Low_spent_Large_value_payments      10425
!@9#%8                               7600
Name: count, dtype: int64
```

```
df['Payment_Behaviour'] = df['Payment_Behaviour'].replace('!@9#%8', np.nan)
```

```
df['Payment_Behaviour'].isna().sum()
```

```
7600
```

```
grouped_df = df.groupby('Payment_Behaviour').size()

print(grouped_df)
```

```
Payment_Behaviour
High_spent_Large_value_payments     13721
High_spent_Medium_value_payments    17540
High_spent_Small_value_payments     11340
Low_spent_Large_value_payments      10425
Low_spent_Medium_value_payments     13861
Low_spent_Small_value_payments      25513
dtype: int64
```

```
df['Payment_Behaviour'] = df['Payment_Behaviour'].fillna(method='ffill')
```

```
df['Payment_Behaviour'].isna().sum()
```

```
0
```

28.- INTEREST_RATE: CONVERTIR DATOS A FLOAT

```
df['Interest_Rate'] = df['Interest_Rate'].astype(float)
```

```
df.Interest_Rate.value_counts()
```

```
Interest_Rate
8.00       5012
5.00       4979
6.00       4721
12.00      4540
10.00      4540
           ...
4995.00       1
1899.00       1
2120.00       1
5762.00       1
5729.00       1
Name: count, Length: 1750, dtype: int64
```

29.- ELIMINAR COLUMNAS INNECESARIAS:

```
df.drop(['ID','Customer_ID', 'Month', 'Name','SSN', 'Type_of_Loan'], axis = 1, inplace = True)
```

32.- CONVERTIR LOS TIPOS DE DATOS A FLOAT EN COLUMNAS SELECCIONADAS:

```
columns_to_convert = ['Num_of_Delayed_Payment', 'Outstanding_Debt', 'Amount_invested_monthly','Num_of_Loan']
for col in columns_to_convert:
    df[col] = df[col].str.replace('_', '').astype(float)
```

```
df.dtypes
```

```
Age                         int64
Occupation                 object
Annual_Income             float64
Monthly_Inhand_Salary     float64
Num_Bank_Accounts           int64
Num_Credit_Card             int64
Interest_Rate             float64
Num_of_Loan               float64
Delay_from_due_date         int64
Num_of_Delayed_Payment    float64
Changed_Credit_Limit      float64
Num_Credit_Inquiries      float64
Credit_Mix                 object
Outstanding_Debt          float64
Credit_Utilization_Ratio  float64
Payment_of_Min_Amount      object
Total_EMI_per_month       float64
Amount_invested_monthly   float64
Payment_Behaviour          object
Monthly_Balance           float64
Credit_Score               object
Credit_History_Age_Months   int64
dtype: object
```

37.- LABEL ENCODING PARA LA COLUMNA OBJETIVO "CREDIT_SCORE" 0:GOOD 1:POOR 2:STANDARD

```
df["Credit_Score"] = LabelEncoder().fit_transform(df["Credit_Score"])
df["Credit_Score"]
```

```
0        0
1        0
2        0
3        0
4        0
        ..
88944    1
88945    1
88946    1
88947    2
88948    1
Name: Credit_Score, Length: 88949, dtype: int32
```

```
#  0:GOOD 1:POOR 2:STANDARD
df["Credit_Score"].value_counts()
```

```
Credit_Score
2    46822
1    26077
0    16050
Name: count, dtype: int64
```

39.- LABEL ENCODING PARA EL RESTO DE COLUMNAS CATEGÓRICAS

```
df.select_dtypes(include=['object']).columns
```

```
Index(['Occupation', 'Credit_Mix', 'Payment_of_Min_Amount',
       'Payment_Behaviour'],
      dtype='object')
```

COLUMNA PAYMENT_BEHAVIOUR:

Este código convierte la columna Payment_Behaviour del DataFrame df en valores numéricos utilizando la codificación ordinal (Ordinal Encoding). La columna original contiene categorías que representan diferentes comportamientos de pago, y la codificación ordinal asigna un número entero a cada categoría en función de un orden específico.

- Preparación de datos para el modelado: Los modelos de machine learning a menudo requieren que las características sean numéricas. La codificación ordinal convierte las categorías en valores numéricos manteniendo el orden inherente de las categorías.
- Mantener el orden: En algunos casos, las categorías tienen un orden natural (por ejemplo, de bajo a alto), y la codificación ordinal conserva esta relación de orden.

En resumen, este código transforma la columna Payment_Behaviour en valores numéricos utilizando la codificación ordinal, manteniendo el orden de las categorías, lo que es útil para preparar los datos para el análisis y la modelización.

```
payment_behaviour_categories = ['Low_spent_Small_value_payments',
                                'Low_spent_Medium_value_payments',
                                'Low_spent_Large_value_payments',
                                'High_spent_Small_value_payments',
                                'High_spent_Medium_value_payments',
                                'High_spent_Large_value_payments']

payment_behaviour_encoder = OrdinalEncoder(categories=[payment_behaviour_categories])

df['Payment_Behaviour'] = payment_behaviour_encoder.fit_transform(df[['Payment_Behaviour']])
```

```
df['Payment_Behaviour']
```

```
0       3.00
1       2.00
2       1.00
3       0.00
4       4.00
        ...
88944   4.00
88945   5.00
88946   4.00
88947   2.00
88948   2.00
Name: Payment_Behaviour, Length: 88949, dtype: float64
```

COLUMNA CREDIT_MIX:

Este código utiliza OrdinalEncoder para transformar la columna Credit_Mix del DataFrame df en valores numéricos.

- Preparación de datos para el modelado: Los modelos de machine learning a menudo requieren que las características sean numéricas. La codificación ordinal convierte las categorías en valores numéricos.
- Manejo de datos categóricos: La codificación ordinal es una forma de manejar datos categóricos cuando no se necesita una relación de orden específica entre las categorías, o cuando dicha relación es implícita en los datos.

En resumen, este código transforma la columna Credit_Mix en valores numéricos utilizando la codificación ordinal, lo que es útil para preparar los datos para el análisis y la modelización.

```
label_encoder = OrdinalEncoder()
df['Credit_Mix'] = label_encoder.fit_transform(df[['Credit_Mix']])
```

```
df['Credit_Mix']
```

```
0       1.00
1       1.00
2       1.00
3       1.00
4       1.00
        ...
88944   1.00
88945   1.00
88946   1.00
88947   1.00
88948   1.00
Name: Credit_Mix, Length: 88949, dtype: float64
```

COLUMNA PAYMENT_OF_MIN_AMOUNT:

Este código utiliza LabelEncoder para transformar la columna Payment_of_Min_Amount del DataFrame df en valores numéricos. LabelEncoder convierte las etiquetas categóricas en números enteros, asignando un número único a cada categoría.

- Preparación de datos para el modelado: Los modelos de machine learning a menudo requieren que las características sean numéricas. LabelEncoder convierte las categorías en valores numéricos de manera eficiente.
- Manejo de datos categóricos: La codificación de etiquetas es útil para convertir datos categóricos en números cuando no se necesita mantener una relación de orden específica entre las categorías.

En resumen, este código transforma la columna Payment_of_Min_Amount en valores numéricos utilizando LabelEncoder, lo que es útil para preparar los datos para el análisis y la modelización.

```
label_encoder = LabelEncoder()
df['Payment_of_Min_Amount'] = label_encoder.fit_transform(df['Payment_of_Min_Amount'])
```

```
df['Payment_of_Min_Amount']
```

```
0        1
1        1
2        1
3        1
4        1
        ..
88944    1
88945    1
88946    1
88947    1
88948    1
Name: Payment_of_Min_Amount, Length: 88949, dtype: int32
```

COLUMNA OCCUPATION:

Este código utiliza LabelEncoder para transformar la columna Occupation del DataFrame df en valores numéricos. LabelEncoder convierte las etiquetas categóricas en números enteros, asignando un número único a cada categoría.

- Preparación de datos para el modelado: Los modelos de machine learning a menudo requieren que las características sean numéricas. LabelEncoder convierte las categorías en valores numéricos de manera eficiente.
- Manejo de datos categóricos: La codificación de etiquetas es útil para convertir datos categóricos en números cuando no se necesita mantener una relación de orden específica entre las categorías.

En resumen, este código transforma la columna Occupation en valores numéricos utilizando LabelEncoder, lo que es útil para preparar los datos para el análisis y la modelización.

```
label_encoder = LabelEncoder()
df['Occupation'] = label_encoder.fit_transform(df['Occupation'])
```

```
df['Occupation']
```

```
0        12
1        12
2        12
3        12
4        12
         ..
88944     9
88945     9
88946     9
88947     9
88948     9
Name: Occupation, Length: 88949, dtype: int32
```

34.- VALIDAR Y ELIMINAR DATOS ATÍPICOS:

Este código tiene como objetivo identificar y eliminar los valores atípicos de las columnas numéricas del DataFrame df para cada grupo de Credit_Score. Los valores atípicos se definen como aquellos que están a más de cuatro desviaciones estándar de la media.

Beneficio de elliminar datos atípicos:

- Limpieza de datos: Eliminar los valores atípicos puede mejorar la calidad de los datos y hacer que los análisis y modelos sean más precisos.
- Reducción de sesgo: Los valores atípicos extremos pueden sesgar los resultados del análisis estadístico y los modelos predictivos.

En resumen, este código identifica y elimina los valores atípicos en las columnas numéricas del DataFrame df para cada grupo de Credit_Score, ayudando a limpiar y preparar los datos para análisis y modelización.

```
df_num = df.select_dtypes(include='number')
for column in df_num.columns:
    for i in df["Credit_Score"].unique():
        selected_i = df[df["Credit_Score"] == i]
        selected_column = selected_i[column]

        std = selected_column.std()
        mean= selected_column.mean()

# Estas líneas definen los límites superior (max) e inferior (min) para determinar qué valores se considerarán atípicos.
        max = mean + (4 * std)
        min =  mean - (4 * std)

        outliers = selected_column[((selected_i[column] > max) | (selected_i[column] < min))].index
        df.drop(index=outliers, inplace=True)
        print(column, i, outliers)
```