TRABAJO DE GRADO
Opción Seminario-Diplomado.

ESTRATEGIA COMPUTACIONAL PARA ESTIMAR LA APROBACION DE PRESTAMOS A PARTIR DE DATOS DE SOLICITANTES, UTILIZANDO ALGORITMOS DE MACHINE LEARNING

Corporación Universitaria Remington.

Ingenierías

Ingeniería Industrial

Estudiante:

Mickell Santiago Paredes Martínez

Tutor: Juan Carlos Briñez de León

Opción de Trabajo de grado Seminario-Diplomado.

2025.

Dedicatoria

A mis padres, por quienes todo soy, por ser mi ejemplo de esfuerzo y constancia. Gracias por cada sacrificio, cada palabra de aliento y por creer en mí incluso cuando yo dudaba. Cada logro en mi vida lleva su huella, porque sin su apoyo incondicional, nada de esto habría sido posible.

Y a ti, mi amor, que has vivido esta etapa a mi lado desde el primer día. Has sido testigo de mis desvelos, Por tu apoyo silencioso, Gracias por tu paciencia infinita, por esperarme cuando el tiempo no alcanzaba.

Este logro no es solo mío, también es suyo. Los llevo cada paso que doy, con todo mi corazón.

Agradecimientos

A DIOS, A LA UNIVERSIDAD. AL PROFE DEL SEMINARIO, MERO TALENTO; y a todas aquellas personas que suman en mi proceso de formación.

Contenido

Resumen

Este trabajo de grado tiene como objetivo aplicar técnicas de aprendizaje automático para mejorar el proceso de toma de decisiones en la aprobación de préstamos. Para ello, se utilizó un conjunto de datos reales con información de clientes, incluyendo variables como género, ingresos, historial crediticio, monto solicitado, entre otros.

A lo largo del proyecto, se realizó un análisis completo que incluyó la limpieza de los datos, transformación de variables y selección de características relevantes. Posteriormente, se entrenaron distintos algoritmos de clasificación supervisada como regresión logística, árboles de decisión, Random Forest, SVM, entre otros, con el fin de predecir si un préstamo debía ser aprobado o rechazado.

El desempeño de los modelos se evaluó con métricas como la exactitud (accuracy), la matriz de confusión, el informe de clasificación y la curva ROC, obteniendo resultados muy positivos. El modelo con mejor desempeño fue validado con datos nuevos mediante una simulación práctica, donde un usuario puede ingresar los datos de un cliente y recibir una predicción inmediata sobre la aprobación del préstamo. Además, se aplicó una estrategia de votación por mayoría entre varios modelos, lo cual fortaleció la confiabilidad del sistema.

Este proyecto demuestra cómo, a través del uso de herramientas de Machine Learning, se puede construir una solución práctica y automatizada para apoyar procesos financieros. El sistema desarrollado es capaz de adaptarse a distintos

tipos de usuarios, entregar resultados confiables y facilitar decisiones más objetivas y rápidas.

En conclusión, se logró implementar una estrategia computacional útil, funcional y con potencial de aplicación en contextos reales, permitiendo no solo optimizar recursos, sino también mejorar la experiencia tanto para las entidades financieras como para los usuarios.

Palabras clave

Machine Learning; Aprendizaje supervisado; Clasificación; Aprobación de préstamos; Algoritmos de predicción; Análisis de datos; Modelos de decisión; Regresión logística; Random Forest; Validación del modelo.

Marco conceptual y contextual

Contexto:

clasificación de datos con aprendizaje supervisado

Hoy en día, el machine learning está cambiando la manera en que muchas empresas usan la información para tomar decisiones. Dentro de este campo, el aprendizaje supervisado es una de las herramientas más útiles, ya que permite hacer predicciones basadas en datos que ya tienen una respuesta conocida. Uno de los casos más comunes de este tipo de aprendizaje es la clasificación, que consiste en enseñarle a un modelo a diferenciar entre dos o más grupos. En temas financieros, por ejemplo, se usa mucho para decidir si aprobar o no un préstamo, ya que normalmente solo hay dos opciones: aprobado o no aprobado. Para esto, existen varios algoritmos que permiten entrenar modelos con información histórica de clientes. Algunos de los más conocidos son:

Regresión logística

Árboles de decisión

Random Forest

Máquinas de vectores de soporte (SVM)

K-Nearest Neighbors (KNN)

Redes neuronales

XGBoost

Estos modelos aprenden a identificar patrones a partir de datos como el ingreso del solicitante, su edad, historial crediticio, el valor del préstamo, su tipo de empleo, entre otros. Con eso, logran predecir si un préstamo es viable o no. Usar este tipo de herramientas ayuda a que las entidades financieras tomen decisiones más rápidas, justas y confiables, además de ahorrar tiempo y reducir errores humanos.

La gran ventaja de aplicar este tipo de soluciones es que se pueden actualizar fácilmente con nuevos datos, y permiten analizar el rendimiento del modelo con métricas como la precisión, el recall, la especificidad, el F1 score y la curva ROC-AUC. Eso hace posible comparar modelos y elegir el que mejor se adapte a la realidad de la organización.

Este trabajo de grado propone usar algoritmos de clasificación supervisada para predecir si un préstamo será aprobado o no, usando un conjunto de datos de solicitudes anteriores. El objetivo es comparar varios modelos de machine learning y ver cuál ofrece mejores resultados, manteniendo siempre una buena interpretación y buscando que esto pueda usarse de verdad en un entorno real, como el de una entidad financiera.

Algoritmos de Machine learning en Aprendizaje supervisado (Clasificación).

Varios autores han estudiado el desempeño de distintos algoritmos de clasificación. Por ejemplo, Lessmann et al. (2015) realizaron un estudio comparativo de 15 algoritmos de clasificación para tareas de scoring crediticio, concluyendo que algunos modelos basados en árboles, como Random Forest y XGBoost, ofrecían una mejor precisión que los modelos clásicos como la regresión logística.

A continuación, se describen los algoritmos más representativos del aprendizaje supervisado, destacando su uso y utilidad en problemas financieros como la aprobación de préstamos:

- Regresión Logística

Es uno de los modelos más antiguos y usados para clasificación binaria. Su ventaja principal es que permite interpretar fácilmente la relación entre las variables independientes y la probabilidad de que ocurra un evento. Por eso ha sido ampliamente usado en modelos de riesgo crediticio (Siddiqi, 2006).

- Árboles de Decisión

Ofrecen una representación gráfica de las decisiones, dividiendo los datos con base en criterios simples. Son fáciles de interpretar, lo cual los hace útiles para contextos donde se requiere justificar la decisión tomada. Sin embargo, pueden

ser propensos al sobreajuste (Overfitting) si no se podan adecuadamente (Quinlan, 1986).

- Random Forest

Es una técnica basada en la combinación de muchos árboles de decisión. Al generar múltiples modelos y tomar una “votación”, mejora la estabilidad y precisión de la predicción. Según Breiman (2001), Random Forest es resistente al sobreajuste y funciona bien incluso con datos faltantes.

- Support Vector Machines (SVM)

Buscan encontrar el hiperplano que mejor separe las clases. Se han utilizado con éxito en problemas financieros donde los datos presentan límites no lineales. Son potentes, pero menos interpretables y más sensibles al escalado de variables (Cortes & Vapnik, 1995).

- K-Nearest Neighbors (KNN)

Clasifica nuevas observaciones basándose en la mayoría de sus vecinos cercanos. Es fácil de implementar, pero su rendimiento disminuye con grandes volúmenes de datos. Aun así, es útil como modelo base para comparaciones (Cover & Hart, 1967).

- XGBoost

Es una técnica de boosting que ha demostrado excelentes resultados en tareas de clasificación. Combina muchos árboles simples para crear un modelo muy preciso. Ha sido usado con éxito en problemas de scoring crediticio, y destaca por su velocidad y precisión (Chen & Guestrin, 2016).

En este trabajo se pretende aplicar y comparar varios de estos algoritmos para resolver un problema real: estimar la probabilidad de que un préstamo sea aprobado, a partir de los datos de solicitantes registrados en una entidad financiera. El conjunto de datos contiene variables como género, casado, dependientes, ingresos, entre otros.

1.2 Descripción de caso de estudio.

El caso se enfoca en el análisis y desarrollo de una estrategia computacional basada en algoritmos de machine learning, con el objetivo de predecir si una solicitud de préstamo será aprobada o no. Para ello, se empleará un conjunto de datos de aprobación de préstamos, que simula o representa los registros históricos de una entidad financiera o concesionario de crédito.

Este tipo de problema es común en el sector financiero, donde miles de personas solicitan créditos, pero no todas cumplen con los criterios necesarios para ser aprobadas. Tomar estas decisiones de forma rápida y precisa es fundamental para reducir el riesgo de cartera y garantizar la sostenibilidad de la

operación crediticia. Sin embargo, muchas organizaciones aún utilizan procesos manuales o reglas rígidas que pueden ser mejorados con el uso de tecnologías como el machine learning.

El conjunto de datos incluye las siguientes variables:

- ✓ Genero
- ✓ Casado
- ✓ Dependientes
- ✓ Educación
- ✓ Trabajador independiente
- ✓ Ingresos del solicitante
- ✓ Ingresos del cosolicitante
- ✓ Monto del préstamo
- ✓ Cuotas
- ✓ Historial crediticio
- ✓ Área

La variable objetivo indica si la solicitud fue aprobada (1) o rechazada (0), por lo que se trata de un problema de clasificación binaria. El enfoque del caso de estudio consiste en aplicar varios algoritmos de aprendizaje supervisado para comparar su rendimiento en términos de precisión y utilidad práctica.

1.3 Pregunta problema:

¿Cómo desarrollar una estrategia computacional para estimar la aprobación de préstamos a partir de los datos de los solicitantes, utilizando algoritmos de aprendizaje supervisado de machine learning?

1.4 Hipótesis:

El análisis computacional de los datos de los solicitantes de préstamo, utilizando algoritmos de aprendizaje supervisado, permitirá predecir con precisión la aprobación o rechazo de sus solicitudes, mejorando así la eficiencia del proceso crediticio y reduciendo el riesgo financiero.

Objetivos

Objetivo general.

Implementar una estrategia computacional para predecir la aprobación de préstamos, a partir de los datos de los solicitantes, utilizando algoritmos de aprendizaje supervisado de machine learning.

Objetivos específicos.

- ✓ Caracterizar y procesar los datos de interés, con miras a la toma de decisiones informadas.
- ✓ Entrenar y evaluar diferentes algoritmos de aprendizaje supervisado
- ✓ Evaluar y analizar el desempeño de los algoritmos implementados para la toma de decisiones.
- ✓ Proponer una estrategia computacional aplicable en escenarios reales, que permita apoyar el proceso de toma de decisiones crediticias en entidades financieras o comerciales.

Desarrollo e implementación del aprendizaje

se diseñó una estrategia computacional basada en algoritmos de machine learning, específicamente del tipo aprendizaje supervisado orientado a clasificación binaria. Esta metodología fue aplicada sobre un conjunto de datos reales que simula las solicitudes de crédito realizadas por diferentes personas, y contiene variables clave:

Loan ID (ID del préstamo)

Nos indica el código asignado a la solicitud del prestamo

Gender (Género)

Indica si el solicitante es hombre**(male)** o mujer**(female)**

influye en el análisis si históricamente se observa una diferencia en aprobación por género

Married (Estado civil, Casado)

Indica si el solicitante está casado o no.

Estar casado podría relacionarse con mayor estabilidad financiera o respaldo económico.

Dependents (Dependientes)

Número de personas que dependen económicamente del solicitante (hijos, adultos mayores, etc.).

Un solicitante con 3 dependientes puede tener más gastos mensuales que alguien sin dependientes.

Education (Educación)

Nivel educativo del solicitante. dividido en graduado**(graduate)** y no graduado **(not graduate).**

Un nivel educativo más alto puede estar relacionado con mayores ingresos y estabilidad laboral.

Self employed (Trabajador independiente)

Indica si el solicitante trabaja por cuenta propia**(true)** o es empleado**(false).**

Un trabajador independiente puede tener ingresos menos predecibles, lo cual influye en la decisión de préstamo.

Applicant income (Ingresos del solicitante)

Es el ingreso mensual del solicitante principal.

A mayor ingreso, mayor capacidad de pago y, por tanto, más probabilidad de aprobación.

Coapplicant income (Ingresos del cosolicitante)

Si hay una segunda persona respaldando el, aquí se registra su ingreso mensual.

Loan amount (Monto del préstamo)

Es la cantidad total de dinero que se está solicitando en el préstamo.

Term (Cuotas)

Es el número de cuotas mensuales a pagar (max 480).

Cuotas muy altas en relación al ingreso pueden ser señales de alto riesgo.

Credit history (Historial crediticio)

Indica si el solicitante tiene buen historial crediticio.

1 = buen historial

0 = mal historial o sin historial

Este es uno de los factores más importantes para aprobar un préstamo.

Property Area (Área de propiedad)

Tipo de zona donde vive el solicitante: Urbana, Semiurbana o Rural.

Loan Status (estado del préstamo)

Indica si el préstamo fue o no aprobado

Preparación y análisis de los datos

Cargando y visualizando datos en Python (Dataframes)

Se cargó el conjunto de datos (conjunto de datos de préstamos) en formato csv importado de kaggle y se revisó su estructura.

figura 1

```
#Cargando datos
import pandas as pd
from google.colab import files
uploaded = files.upload()
for filename in uploaded.keys():
    Datos_Loan = pd.read_csv(filename,sep=',')

Datos_Loan.head(7)
```

```
Elegir archivos Ningún archivo seleccionado Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving Dataset_loan_transanction (1).csv to Dataset_loan_transanction (1).csv
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
5	LP001011	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0	360.0	1.0	Urban	Y
6	LP001013	Male	Yes	0	Not Graduate	No	2333	1516.0	95.0	360.0	1.0	Urban	Y

Analizando datos:

Como parte del análisis inicial usamos el algoritmo **Conjunto_Datos.info()** el cual es una función de Pandas que es una librería de Python para análisis de datos

Figura 2

```
Datos_Loan.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            614 non-null    object 
 1   Gender             601 non-null    object 
 2   Married            611 non-null    object 
 3   Dependents         599 non-null    object 
 4   Education          614 non-null    object 
 5   Self_Employed      582 non-null    object 
 6   ApplicantIncome    614 non-null    int64  
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object 
 12  Loan_Status        614 non-null    object 
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

Aquí podemos observar cuántas filas y columnas tiene el conjunto de datos, en este caso tenemos 614 registros que van desde el 0 hasta las 613 y 13 columnas, observamos si hay valores faltantes, Si una columna tiene menos valores que el total como se observa en la columna de Gender que tiene 601 en vez de 614, eso indica que tiene 13 datos nulos, también identificamos los tipos de datos, si las columnas son de tipo object son de texto,si son int64 son de números enteros o float64 números decimales.

Luego usamos el algoritmo **Conjunto_Datos.describe()** Este genera un resumen estadístico de las columnas numéricas del conjunto de datos lo que te permite entender la distribución de los datos rápidamente.

Figura 3

```
[ ] Datos_Loan.describe()
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

Quitando columnas indeseadas y filas con datos nulos

Eliminamos columnas que no aportan información relevante al análisis o al modelo, en este caso el Loan_ID, ya que esta no influye en la predicción de la aprobación del préstamo, también borramos las filas que tienen valores faltantes porque los modelos no pueden trabajar con datos vacíos.

Eliminando columnas indeseadas

Figura 4

```
#Quitando columnas indeseadas
Datos_Loan=Datos_Loan.drop(columns=['Loan_ID'],axis=1)
Datos_Loan.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
1	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
5	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0	360.0	1.0	Urban	Y

Eliminando filas con datos nulos

Figura 5

```
#Elimina filas que tengan datos nulos
Datos_Loan=Datos_Loan.dropna()
Datos_Loan.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
5	LP001011	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0	360.0	1.0	Urban	Y

Se eliminaron o trataron los datos faltantes, dejando los datos en un estado limpio y adecuado para el análisis.

codificando las variables categóricas.

con el algoritmo

```
print('Analizando el género')

Datos_Loan['Gender'].unique()
```

Nos muestra las categorías que tiene cada columna, esto con el fin de codificarlas, esto los hacemos con todas las variables categoricas

Figura 6

```
[ ] #Verificación de las opciones de la variable
    print('Analizando el género')
    Datos_Loan['Gender'].unique()

    Analizando el género
    array(['Male', 'Female'], dtype=object)

    #Verificación de las opciones de la variable
    print('Analizando el estado')
    Datos_Loan['Married'].unique()

    Analizando el estado
    array(['Yes', 'No'], dtype=object)

[ ] #Verificación de las opciones de la variable
    print('Analizando los dependientes')
    Datos_Loan['Dependents'].unique()

    Analizando los dependientes
    array(['1', '0', '2', '3+'], dtype=object)

[ ] #Verificación de las opciones de la variable
    print('Analizando nivel de estudios')
    Datos_Loan['Education'].unique()

    Analizando nivel de estudios
    array(['Graduate', 'Not Graduate'], dtype=object)
```

figura 7

```
[ ] #Verificación de las opciones de la variable
    print('Analizando tipo de trabajo')
    Datos_Loan['Self_Employed'].unique()

    Analizando tipo de trabajo
    array(['No', 'Yes'], dtype=object)

[ ] #Verificación de las opciones de la variable
    print('Analizando lugar de residencia')
    Datos_Loan['Property_Area'].unique()

    Analizando lugar de residencia
    array(['Rural', 'Urban', 'Semiurban'], dtype=object)

[ ] #Verificación de las opciones de la variable
    print('Analizando la decisión')
    Datos_Loan['Loan_Status'].unique()

    Analizando la decisión
    array(['N', 'Y'], dtype=object)
```

Pasamos las variables de categóricas a numéricas

```
#Mapeando todas la variables categóricas a numéricas

Reemplazo_1={'Male':100,'Female':200}

Datos_Loan['Gender']=Datos_Loan['Gender'].map(Reemplazo_1)
```

Con este código se hace un reemplazo de valores categóricos por valores numéricos en la columna, esto se hace con todas las columnas categóricas, las cuales analizamos anteriormente

Figura 8

```
#Mapeando todas la variables categóricas a numéricas
Reemplazo_1={'Male':100,'Female':200}
Datos_Loan['Gender']=Datos_Loan['Gender'].map(Reemplazo_1)

Reemplazo_2={'Yes':100,'No':200}
Datos_Loan['Married']=Datos_Loan['Married'].map(Reemplazo_2)

Reemplazo_3={'0':0,'1':1,'2':2,'3+':3}
Datos_Loan['Dependents']=Datos_Loan['Dependents'].map(Reemplazo_3)

Reemplazo_4={'Graduate':100,'Not Graduate':0}
Datos_Loan['Education']=Datos_Loan['Education'].map(Reemplazo_4)

Reemplazo_5={'Yes':100,'No':200}
Datos_Loan['Self_Employed']=Datos_Loan['Self_Employed'].map(Reemplazo_5)

Reemplazo_5={'Urban':0,'Rural':10,'Semiurban':20}
Datos_Loan['Property_Area']=Datos_Loan['Property_Area'].map(Reemplazo_5)

Reemplazo_6={'Y':1,'N':0} #La salida
Datos_Loan['Loan_Status']=Datos_Loan['Loan_Status'].map(Reemplazo_6)

Datos_Loan.head()
```

Selección y entrenamiento de modelos

Se busca entrenar lo modelos con distintos enfoques para comparar su rendimiento en la tarea de clasificación. Convertimos los datos a una matriz NumPy y los dividimos en entrenamiento y testeo

Figura 9

```
[ ]  #Divide datos en entradas y salidas
     import numpy as np
     Datos_matriz=np.array(Datos_Loan)
     #Datos_matriz[np.isnan(Datos_matriz)] = 0
     X = Datos_matriz[:,0:-1] #datos de entrada (Todas las variables del cliente)
     Y = Datos_matriz[:,-1] #Datos de salida (La decisiòn del crédito)
```

```
[ ]  # Divide datos en Entrenamiento y testeo
     import sklearn
     from sklearn.model_selection import train_test_split
     X_train, X_test,Y_train, Y_test= train_test_split(X,Y,test_size=0.1,random_state=751)
```

Modelo de toma de decisiones

Es un modelo de toma de decisiones basado en algoritmos de aprendizaje supervisado, con el objetivo de predecir si una solicitud de préstamo será aprobada o no, a partir de los datos del solicitante.

Para esto, nos enfocamos en aplicar modelos de clasificación binaria, donde el resultado final es una etiqueta que representa la decisión del crédito:

1: Aprobado

0: No aprobado

El modelo analiza las variables y aprende de los datos históricos qué tipo de perfiles tienden a recibir aprobación, y luego es capaz de predecir con cierto grado de certeza si un nuevo solicitante será aprobado o no.

Lo que se hizo fue importar todos los algoritmos y herramientas necesarias para aplicar y comparar diferentes modelos de clasificación supervisada a nuestro conjunto de datos desde sklearn, una de las librerías más potentes para machine learning en Python.

Figura 10

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis, QuadraticDiscriminantAnalysis
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, AdaBoostClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import warnings
warnings.filterwarnings("ignore")

# Modelo 0: K-Nearest Neighbors
modelo_0 = KNeighborsClassifier(n_neighbors=5)
modelo_0.fit(X_train, Y_train)
Y_pred_0 = modelo_0.predict(X_test)
print("Accuracy KNN:", accuracy_score(Y_test, Y_pred_0))

# Modelo 1: Naive Bayes
modelo_1 = GaussianNB()
modelo_1.fit(X_train, Y_train)
Y_pred_1 = modelo_1.predict(X_test)
print("Accuracy Naive Bayes:", accuracy_score(Y_test, Y_pred_1))

# Modelo 2: Linear Discriminant Analysis
modelo_2 = LinearDiscriminantAnalysis()
modelo_2.fit(X_train, Y_train)
Y_pred_2 = modelo_2.predict(X_test)
print("Accuracy LDA:", accuracy_score(Y_test, Y_pred_2))
```

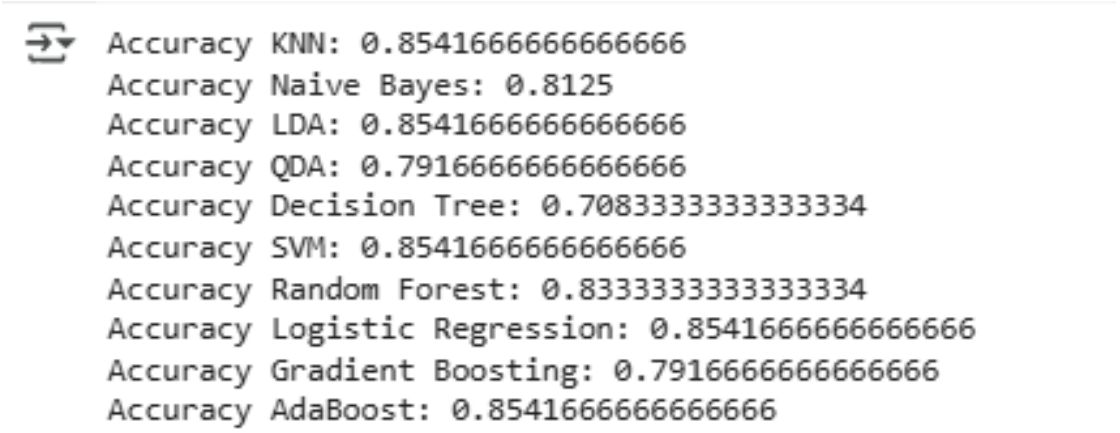
Los modelos que importamos fueron los siguientes:

- ✓ KNeighborsClassifier
- ✓ GaussianNB
- ✓ LinearDiscriminantAnalysis
- ✓ QuadraticDiscriminantAnalysis
- ✓ DecisionTreeClassifier

- ✓ SVC (Support Vector Classifier)
- ✓ RandomForestClassifier
- ✓ GradientBoostingClassifier
- ✓ AdaBoostClassifier
- ✓ LogisticRegression

También importamos, **accuracy_score** para evaluar qué tan preciso es cada modelo qué porcentaje de veces acertó en su predicción.

Figura 11



Los modelos que mejor desempeño mostraron, con una precisión del 85.41%, fueron:

KNN

LDA

SVM

Regresión Logística

AdaBoost

La estrategia computacional implementada permitió evaluar diferentes algoritmos de machine learning para la predicción de aprobación de préstamos. Gracias a esto, se logró identificar los modelos más adecuados para este tipo de problema

Análisis de desempeño

Se evaluó el rendimiento del modelo seleccionado aplicando métricas clave que permiten medir su precisión, capacidad de generalización y calidad en la toma de decisiones.

Precisión global (Accuracy)

```
# Mostrar resultados
print(f"Accuracy: {accuracy:.2f}")
print('==================')
print(' ')
```

```
Accuracy: 0.85
```

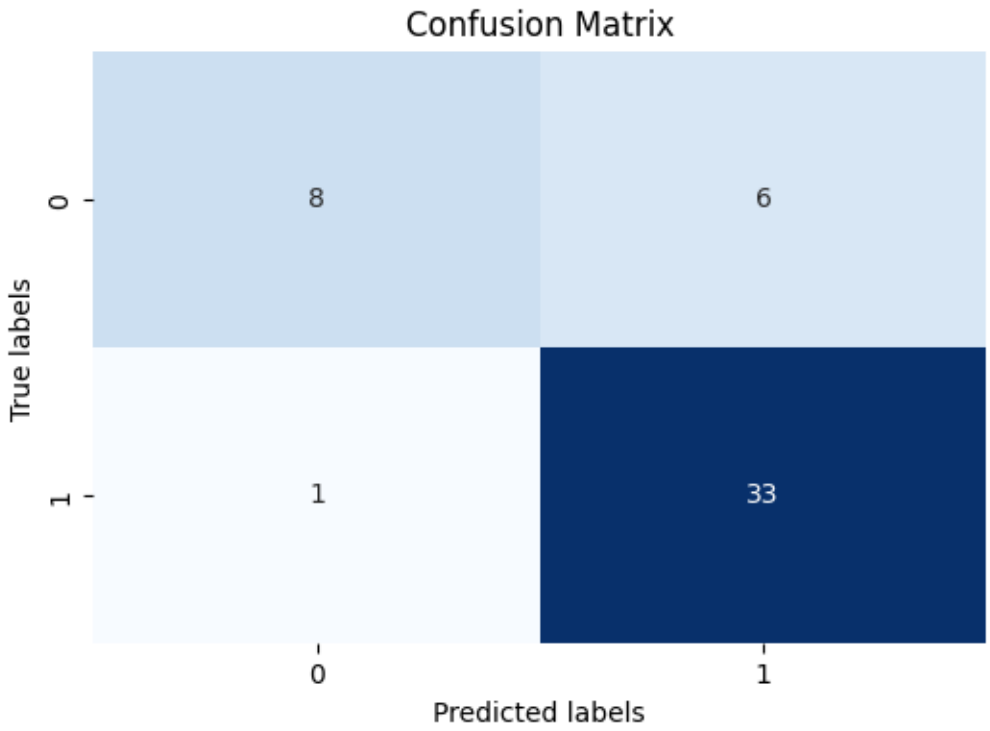
El modelo obtuvo una precisión del 85%, lo que indica que acertó en 85 de cada 100 predicciones realizadas. Este resultado sugiere que el modelo tiene un desempeño sólido en la clasificación binaria de solicitudes de préstamo como aprobadas o no aprobadas.

Matriz de confusión

Nos muestra la cantidad de verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos.

- Verdaderos Positivos (TP): solicitudes correctamente aprobadas por el modelo.
- Verdaderos Negativos (TN): rechazos correctamente clasificados.
- Falsos Positivos (FP): solicitudes que fueron aprobadas incorrectamente.
- Falsos Negativos (FN): rechazos incorrectos a personas que sí calificaban.

Figura 12



- El modelo acierta en la mayoría de los casos, especialmente en identificar aprobaciones, acertó 33 casos correctos frente a solo 1 error.

- El error más común es aprobar préstamos que no deberían aprobarse 6 falsos positivos. Esto puede representar riesgo financiero si se dieran préstamos a personas no aptas.

- solo 1 caso fue rechazado erróneamente

Esta matriz de confusión muestra que el modelo es altamente efectivo identificando clientes aptos para préstamos, este análisis permite entender en qué tipo de errores incurre más el modelo, lo cual es clave para mejorar su precisión y reducir el riesgo crediticio.

Classification Report

Figura 13

Classification Report:

	precision	recall	f1-score	support
0.0	0.89	0.57	0.70	14
1.0	0.85	0.97	0.90	34
accuracy			0.85	48
macro avg	0.87	0.77	0.80	48
weighted avg	0.86	0.85	0.84	48

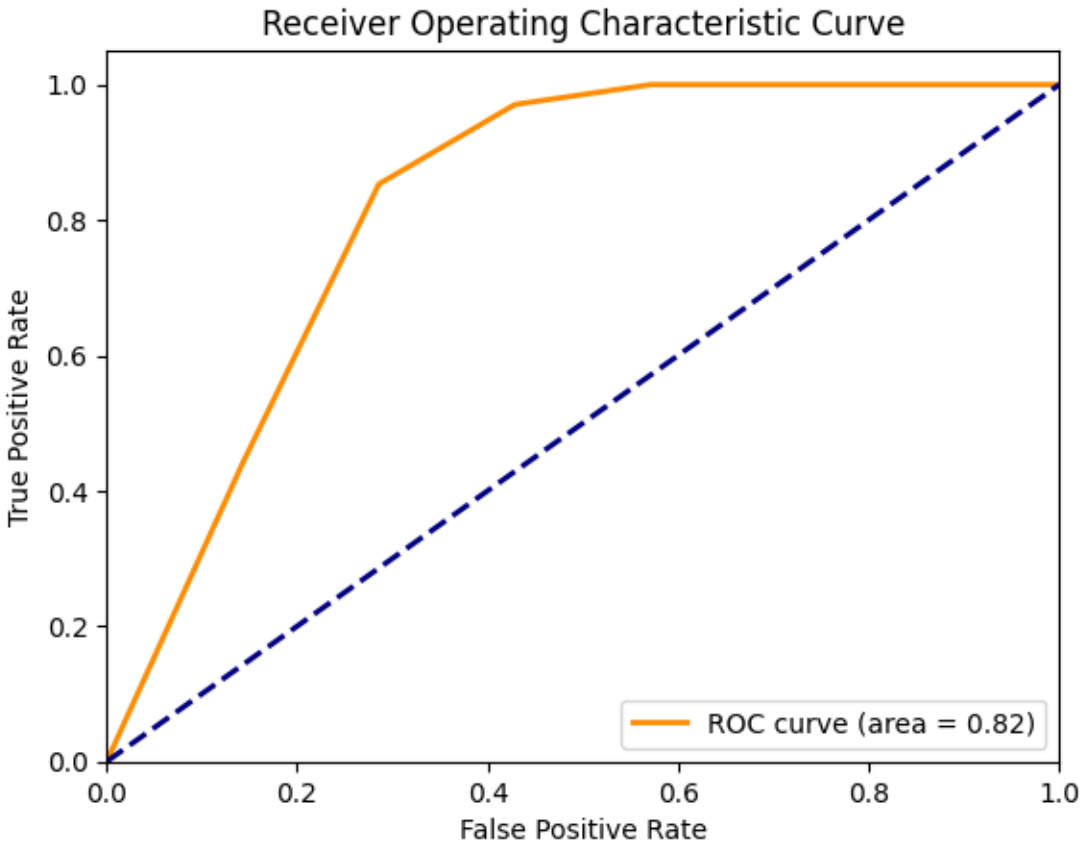
- El modelo es muy fuerte en detectar quién sí debe recibir un préstamo

- Tiende a fallar más al predecir quién no debe recibirlo
- Este comportamiento favorece al cliente, pero podría implicar riesgo para la empresa, ya que aprueba algunas solicitudes que deberían ser rechazadas.
- A pesar de ello, mantiene una precisión global (accuracy) del 85%, lo cual muy bueno

Curva ROC

La curva ROC (Receiver Operating Characteristic) muestra la relación entre la tasa de verdaderos positivos (True Positive Rate o Sensibilidad) y la tasa de falsos positivos (False Positive Rate) para diferentes umbrales de decisión del modelo.

Figura 14



La línea curva naranja representa el desempeño real del modelo mientras la línea diagonal azul punteada representa el rendimiento de un modelo aleatorio, es decir, sin capacidad predictiva.

El área bajo la curva es de 0.82, lo cual indica un rendimiento muy bueno. Un AUC entre 0.8 y 0.9 se considera excelente, lo que demuestra que el modelo es consistente y confiable para distinguir entre clientes que deben o no ser aprobados.

Validación del modelo

Una vez entrenado y evaluado el modelo de clasificación para estimar la aprobación de préstamos, se procedió con la etapa de validación, que nos permite conocer cómo se comportaría el sistema ante nuevos datos reales que no han sido vistos previamente.

Para verificar cómo funciona el modelo ante clientes reales o hipotéticos que no han sido parte del entrenamiento, se realizó una simulación con datos nuevos. Se uso el siguiente código:

```
 import warnings
import numpy as np
from collections import Counter
```

```
warnings.filterwarnings("ignore")


# Vector de entrada para un solo sujeto
Target = np.zeros((1, 11))


# Inputs con instrucciones dentro del mismo input
Target[0, 0] = int(input("Género (100 = Male, 200 = Female): "))
Target[0, 1] = int(input("Estado civil (100 = Casado, 200 =
Soltero): "))
Target[0, 2] = int(input("Dependientes (0, 1, 2, 3 para '3+'): "))
Target[0, 3] = int(input("Educación (100 = Graduate, 0 = Not
Graduate): "))
Target[0, 4] = int(input("¿Trabajador independiente? (100 = Sí, 200
= No): "))
Target[0, 5] = float(input("Ingreso del solicitante (ej. 5000): "))
Target[0, 6] = float(input("Ingreso del codeudor (ej. 1000): "))
Target[0, 7] = float(input("Monto del préstamo (ej. 150): "))
Target[0, 8] = float(input("Plazo del préstamo en meses (ej. 360):
"))
Target[0, 9] = float(input("Historial crediticio (1.0 = Sí, 0.0 =
No): "))
Target[0, 10] = int(input("Área de la propiedad (0 = Urbana, 10 =
Rural, 20 = Semiurbana): "))


# Normalización
Target = scaler.transform(Target)
```

```
# Predicciones
predicciones = [
    modelo_0.predict(Target)[0],
    modelo_1.predict(Target)[0],
    modelo_2.predict(Target)[0],
    modelo_3.predict(Target)[0],
    modelo_4.predict(Target)[0],
    modelo_5.predict(Target)[0],
    modelo_6.predict(Target)[0],
    modelo_7.predict(Target)[0],
    modelo_8.predict(Target)[0],
    modelo_9.predict(Target)[0]
]

# Resultados individuales
nombres_modelos = [
    "KNN", "Bayes", "LDA", "QDA", "Árbol", "SVM",
    "Random Forest", "Logística", "Gradient Boosting", "AdaBoost"
]

for nombre, pred in zip(nombres_modelos, predicciones):
    print(f"Según {nombre}: {'✅ Aprobado' if pred == 1 else '❌
Rechazado'}")

# Mayoría
```

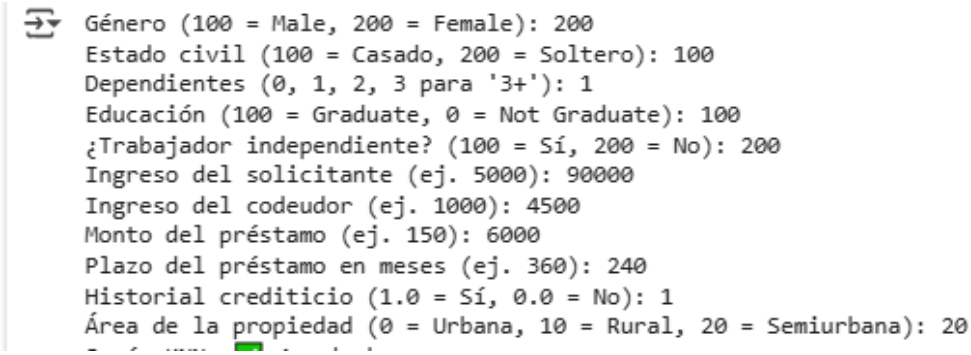
```
conteo = Counter(predicciones)

mayoria = conteo.most_common(1)[0][0]

total = conteo[mayoria]


print("\n--- Decisión por mayoría ---")

print(f"{'✅ Aprobado' if mayoria == 1 else '❌ Rechazado'}

({total}/10 modelos)")
```

El sistema pide los siguientes datos del cliente:

- Género (codificado como 100 para masculino, 200 para femenino)
- Estado civil
- Número de dependientes
- Nivel educativo
- Si es trabajador independiente
- Ingresos del solicitante y del codeudor
- Monto del préstamo solicitado
- Plazo del préstamo (en meses)
- Historial crediticio
- Tipo de zona de residencia

Los datos ingresados fueron los siguientes:

Figura 15



Una vez ingresados los datos se usa el entrenamiento, y se ejecutan las predicciones usando los 10 modelos entrenados:

KNN

Naive Bayes

LDA

QDA

Árbol de decisión

SVM

Random Forest

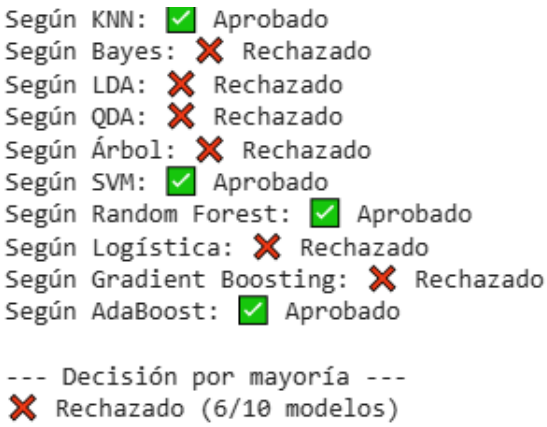
Regresión logística

Gradient Boosting

AdaBoost

Cada modelo da una respuesta: **1 = Aprobado** o **0 = Rechazado.**

Figura 16



Finalmente, se hace una votación por mayoría para decidir la recomendación general del sistema.

Este sistema no solo muestra que el modelo funciona bien desde lo técnico, sino que también se puede usar de forma sencilla en la vida real. Es decir, cualquier persona, como un asesor o encargado de aprobar préstamos, podría usarlo fácilmente. Además, con la validación interactiva pudimos ver que el modelo se adapta bien a distintos tipos de usuarios y sigue dando resultados confiables.

Conclusiones y trabajos futuros

El uso de Machine Learning sí mejora la toma de decisiones, especialmente en escenarios como la aprobación de préstamos. Se logro entrenar varios modelos capaces de predecir con buen nivel de acierto si una solicitud debía ser

aprobada o no. se trabajó con datos reales, hicimos limpieza, transformación y análisis, lo cual permitió entender cómo ciertas variables como el ingreso, el historial crediticio o el estado civil influyen en la aprobación de un préstamo.

Se probaron varios algoritmos de clasificación, como árboles de decisión, regresión logística, Random Forest, entre otros, y comparamos su rendimiento usando métricas como accuracy, matriz de confusión, curva ROC, etc. Se comprobó que el modelo responde bien a situaciones reales. También se diseño una simulación donde un usuario puede ingresar información y recibir una predicción inmediata.

Finalmente, este trabajo permitió afianzar lo aprendido en el curso, desarrollar habilidades en análisis de datos, programación en Python y aplicación de modelos predictivos, todo con una finalidad práctica que responde a necesidades reales del entorno.

Referencias bibliográficas

Han, J., Kamber, M., & Pei, J. (2012). *Data Mining: Concepts and Techniques*. Morgan Kaufmann.

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly.

Baesens, B., Setiono, R., Mues, C., & Vanthienen, J. (2003). Using neural network rule extraction and decision tables for credit-risk evaluation. *Management Science*, 49(3), 312-329.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273-297.

Siddiqi, N. (2006). *Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring*. Wiley.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5-32.

Lessmann, S., Baesens, B., Seow, H. V., & Thomas, L. C. (2015). Benchmarking classification algorithms for credit scoring: An update. *European Journal of Operational Research*, 247(1), 124–136.

Quinlan, J. R. (1986). Induction of decision trees. Machine Learning*, 1(1), 81–106.*

Bishop, C. M. (2006). Pattern Recognition and Machine Learning*. Springer.*

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning*. Springer.*

https://www.kaggle.com/datasets/mirzahasnine/loan-data-set/data