Clasificación de riesgo crediticio en clientes bancarios

**Autor:** Kevin Gómez Villanueva

**Fecha:** 10 de diciembre de 2024

**Resumen ejecutivo:**

Este informe presenta la implementación de un pipeline de Machine Learning usando los servicios de AWS Sagemaker y AWS Bedrock para la clasificación de riesgo crediticio de clientes bancarios. Se aborda el etiquetado de datos usando Modelos Grandes de Lenguaje (LLM), el entrenamiento de un modelo de Machine Learning usando Support Vector Machine y el despliegue del modelo en un endpoint de Sagemaker para las inferencias.

**Índice**

[1. Introducción 1](#_Toc184735544)

[2. Objetivos 2](#_Toc184735545)

[3. Metodología 2](#_Toc184735546)

[4. Desarrollo 4](#_Toc184735547)

[4.1. Análisis Exploratorio de Datos 4](#_Toc184735548)

[4.2. Etiquetado automático de datos 6](#_Toc184735549)

[4.3. Entrenamiento 10](#_Toc184735550)

[4.4. Despliegue 15](#_Toc184735551)

[4.5. Inferencia 16](#_Toc184735552)

[5. Conclusiones 17](#_Toc184735553)

# Intr**oducción**

La clasificación de riesgo crediticio en clientes bancarios es un proceso analítico que evalúa la probabilidad de que un cliente incumpla con el pago de sus obligaciones financieras. Su objetivo es asignar una calificación que refleje el nivel de riesgo asociado con otorgar crédito a un cliente.

La importancia de este proceso radica en su capacidad para ayudar a las instituciones financieras a gestionar el riesgo de manera eficiente, minimizando pérdidas por incumplimientos y optimizando la asignación de recursos.

# Objetivos

Objetivo general: Determinar el nivel de riesgo crediticio de un cliente a partir de sus datos.

Objetivos específicos:

* Utilizar los servicios de AWS Bedrock para elaborar una descripción del perfil del cliente y el etiquetado automático de datos.
* Entrenar y desplegar un modelo de Machine Learning eficiente usando AWS Sagemaker.

# Metodología

**Base de datos:** Se obtiene la base de datos inicial del archivo CSV credit\_risk\_reto.csv. Esta base de datos contiene 1000 entradas con 9 columnas categóricas y numéricas donde cada entrada representa a un cliente. El detalle de las columnas se presenta a continuación.

* Age: Edad de la persona
* Sex: Sexo de la persona
* Job: ( 0 - unskilled and non-resident, 1 - unskilled and resident, 2 - skilled, 3 - highly skilled)
* Housing: Tipo de alojamiento
* Saving accounts: Tipo de cuenta de ahorro
* Checking account: Tipo de cuenta corriente
* Credit amount: Monto de crédito
* Duration (meses): Tiempo de préstamo
* Purpose: Motivo del préstamo

**Exploración de datos:** Se realiza una revisión del dataset para encontrar datos faltantes, datos incorrectos, cantidad de datos, datos numéricos, datos categóricos.

**Etiquetado de datos:** Se realiza la limpieza de datos en base a lo analizado en la exploración y se procede a utilizar AWS Bedrock para el etiquetado automático de datos.

**Preprocesamiento de datos:** Se acondicionan los datos para ser utilizados en el entrenamiento de modelos de Machine Learning.

**Entrenamiento de modelo:** Se utilizan los datos preprocesados para entrenar diferentes modelos y determinar el más adecuado que será utilizado en producción.

**Despliegue del modelo:** El modelo seleccionado se aloja en un endpoint de AWS Sagemaker para realizar inferencias.

Debido a que el proceso de llevar desde la base de datos hasta un modelo en producción requiere de una construcción por etapas, se plantean 3 etapas.

En la Etapa 1 se desarrolla el análisis de los datos, el etiquetado, preprocesamiento, entrenamiento de modelos candidatos con hiperparametrización y la evaluación.

En la Etapa 2 se lleva el modelo a producción tomando en cuenta que se pueden sumar nuevos datos para mantener el modelo eficiente.

En la Etapa 3 se integra la inferencia con el modelo actualizado y todo el pipeline se mantiene en constante monitoreo.

El diagrama se muestra en la Figura 1.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

**Figura 1.** Diagrama de procesos del pipeline de Machine Learning.

# Desarrollo

Los notebooks y archivos resultantes se encuentran en mi Github personal:

<https://github.com/kevinGmezIoT/credit-risk-reto>

## Análisis Exploratorio de Datos

El desarrollo inicia con el Análisis Exploratorio de Datos (EDA) para conocer las características del dataset.

Realizando una revisión de las primeras entradas notamos que contiene datos en texto, numéricos y que pueden faltar algunos datos ya que se encuentran NaN.

Imagen que contiene Texto

Descripción generada automáticamente

Al mostrar la información del dataset, se obtiene que son 1000 entradas, donde las columnas Saving accounts y Checking account contienen datos nulos.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

La descripción estadística muestra que los datos de Credit amount presentan una elevada desviación estándar, también el rango de duración es amplia y el rango de edad va de 19 a 75 años.

Texto

Descripción generada automáticamente

La distribución de edad muestra que la mayoría se encuentra entre 25 a 30 años.

Gráfico, Histograma

Descripción generada automáticamente

Las frecuencias de las variables categóricas muestran lo desbalanceado del dataset.

Gráfico, Gráfico de barras

Descripción generada automáticamenteGráfico, Gráfico de barras

Descripción generada automáticamente

Gráfico, Gráfico de barras

Descripción generada automáticamenteGráfico, Gráfico de barras

Descripción generada automáticamente

## Etiquetado automático de datos

Antes de etiquetar los datos, se decidió limpiar los datos rellenando los valores nulos con la moda de sus categorías. El resultado es almacenado en 'data/processed/credit\_risk\_reto\_preprocessed.csv'

def clean\_data(data):  
 # Calcular el número de valores faltantes por columna  
 missing\_values = data.isnull().sum()  
   
 # Filtrar solo las columnas con valores faltantes  
 columns\_with\_missing = missing\_values[missing\_values > 0]  
   
 # Imprimir el resultado  
 print("Valores faltantes por columna:")  
 print(columns\_with\_missing)  
   
 # Mostrar el porcentaje de valores faltantes para tener más contexto  
 percent\_missing = (columns\_with\_missing / len(data)) \* 100  
 print("\nPorcentaje de valores faltantes por columna:")  
 print(percent\_missing)  
   
 categorical\_cols = ['Sex', 'Housing', 'Saving accounts', 'Checking account', 'Purpose']  
 cat\_imputer = SimpleImputer(strategy='most\_frequent')  
 data[categorical\_cols] = cat\_imputer.fit\_transform(data[categorical\_cols])  
   
 # Calcular el número de valores faltantes por columna  
 missing\_values = data.isnull().sum()  
   
 # Filtrar solo las columnas con valores faltantes  
 columns\_with\_missing = missing\_values[missing\_values > 0]  
   
 # Imprimir el resultado  
 print("Valores faltantes por columna:")  
 print(columns\_with\_missing)  
   
 # Mostrar el porcentaje de valores faltantes para tener más contexto  
 percent\_missing = (columns\_with\_missing / len(data)) \* 100  
 print("\nPorcentaje de valores faltantes por columna:")  
 print(percent\_missing)  
   
 data.to\_csv('data/processed/credit\_risk\_reto\_preprocessed.csv', index=False)  
   
 print("\nData guardada en credit\_risk\_reto\_preprocessed.csv")  
   
 return data

Luego se utilizó langchain para formar el prompt que se rellena con los datos de las entradas y con ello obtener una descripción. Se utilizó el Modelo amazon.titan-text-premier-v1:0 debido a su precisión en descripción. Se probaron otros modelos como Llama3 y Amazon Titan Express pero, en algunos casos, no brindaban buenas descripciones. El prompt utilizado es:

“

Eres un experto en riesgos crediticio bancario.

Se te proveerá una serie de datos descritos a continuación:

Edad: Edad de la persona

Sexo: Sexo de la persona

Trabajo: ( 0 - unskilled and non-resident, 1 - unskilled and resident, 2 - skilled, 3 - highly skilled)

Alojamiento: Tipo de alojamiento

Cuentas de ahorro: Tipo de cuenta de ahorro

Cuenta corriente: Tipo de cuenta corriente

Monto del crédito: Monto de crédito

Duración (meses): Tiempo de préstamo

Finalidad: Motivo del préstamo

Tu tarea es describir los datos presentados en un máximo de 30 palabras con su relación con el riesgo crediticio.

Estos son los datos:

Edad: {age}

Sexo: {sex}

Trabajo: {job}

Alojamiento: {Housing}

Cuentas de ahorro: {Saving\_accounts}

Cuenta corriente: {Checking\_account}

Monto del crédito: {Credit\_amount}

Duración: {Duration}

Finalidad: {Purpose}

Escribe tu respuesta a continuación:

”

Finalmente, el código se muestra debajo y el resultado se guarda en 'data/processed/output\_description.csv'

def add\_description(data):  
 llm = ChatBedrock(  
 credentials\_profile\_name="bedrock-user-admin", model\_id="amazon.titan-text-premier-v1:0"  
 )  
   
 text = """  
 Eres un experto en riesgos crediticio bancario.  
 Se te proveerá una serie de datos descritos a continuación:  
 Edad: Edad de la persona  
 Sexo: Sexo de la persona  
 Trabajo: ( 0 - unskilled and non-resident, 1 - unskilled and resident, 2 - skilled, 3 - highly skilled)  
 Alojamiento: Tipo de alojamiento  
 Cuentas de ahorro: Tipo de cuenta de ahorro  
 Cuenta corriente: Tipo de cuenta corriente  
 Monto del crédito: Monto de crédito  
 Duración (meses): Tiempo de préstamo  
 Finalidad: Motivo del préstamo  
   
 Tu tarea es describir los datos presentados en un máximo de 30 palabras con su relación con el riesgo crediticio.  
   
 Estos son los datos:  
 Edad: {age}  
 Sexo: {sex}  
 Trabajo: {job}  
 Alojamiento: {Housing}  
 Cuentas de ahorro: {Saving\_accounts}  
 Cuenta corriente: {Checking\_account}  
 Monto del crédito: {Credit\_amount}  
 Duración: {Duration}  
 Finalidad: {Purpose}  
   
 Escribe tu respuesta a continuación:  
 """  
   
 prompt\_template = PromptTemplate.from\_template(text)  
 results = []  
 for index, row in data.iterrows():  
 age = row['Age']  
 sex = row['Sex']  
 job = row['Job']  
 Housing = row['Housing']  
 Saving\_accounts = row['Saving accounts']  
 Checking\_account = row['Checking account']  
 Credit\_amount = row['Credit amount']  
 Duration = row['Duration']  
 Purpose = row['Purpose']  
 result = prompt\_template.invoke({"age": age,  
 "sex": sex,  
 "job": job,  
 "Housing": Housing,  
 "Saving\_accounts": Saving\_accounts,  
 "Checking\_account": Checking\_account,  
 "Credit\_amount": Credit\_amount,  
 "Duration": Duration,  
 "Purpose": Purpose  
 })  
 classification = llm.invoke(input=result)  
   
 results.append(classification.content)  
   
 print(classification.content)  
 time.sleep(5)  
   
 data['description'] = results  
 data.to\_csv('data/processed/output\_description.csv', index=False)  
   
 print("DataFrame guardado en 'output\_description.csv'")  
 return data

El siguiente paso es clasificar las descripciones como “Good risk” o “bad risk”, para ello se utilizó el modelo Claude 3.5 sonnet ya que se requiere que el modelo siga lo mejor posible las indicaciones y minimizar sus alucinaciones, además de una gran precisión y por eso se utiliza un prompt en inglés.

Para la clasificación se utiliza el siguiente prompt:

“

You are an expert in bank credit risk.

Your task is to classify the credit risk as 'good risk' or 'bad risk'.

Examples:

Description: A 67 year old man requested a loan of 1169 for a TV, he is skilled, has little savings, his own home and has requested a loan for 6 months.

Answer: bad risk

Description: A 22 year old woman requested a loan of 5951 euros for 48 months to buy a radio or television. She is a skilled worker and owns her own home. She has a small savings account and a moderate current account.

Answer: good risk

The following is the description you must classify and is important your answer should be only 'good risk' or 'bad risk':

Description: {description}

Answer:

”

El código se muestra debajo y el resultado se guarda en 'data/processed/output\_target.csv'

def add\_target(data):  
 llm = ChatBedrock(  
 credentials\_profile\_name="bedrock-user-admin", model\_id="anthropic.claude-3-5-sonnet-20240620-v1:0"  
 )  
   
 text = """  
 You are an expert in bank credit risk.  
 Your task is to classify the credit risk as 'good risk' or 'bad risk'.  
   
 Examples:  
 Description: A 67 year old man requested a loan of 1169 for a TV, he is skilled, has little savings, his own home and has requested a loan for 6 months.  
 Answer: bad risk  
   
 Description: A 22 year old woman requested a loan of 5951 euros for 48 months to buy a radio or television. She is a skilled worker and owns her own home. She has a small savings account and a moderate current account.  
 Answer: good risk  
   
 The following is the description you must classify and is important your answer should be only 'good risk' or 'bad risk':  
   
 Description: {description}  
 Answer:  
 """  
   
 prompt\_template = PromptTemplate.from\_template(text)  
   
 results = []  
 for index, row in data.iterrows():  
 description = row['description']  
 result = prompt\_template.invoke({"description": description})  
 classification = llm.invoke(input=result)  
   
 results.append(classification.content)  
   
 print(classification.content)  
 time.sleep(5)  
   
 data['target'] = results  
 data.to\_csv('data/processed/output\_target.csv', index=False)  
   
 print("DataFrame guardado en 'output\_target.csv'")  
 return data

## Entrenamiento

Se comparó el entrenamiento de los modelos:

Regresión logística, K vecinos cercanos, Support Vector Machine, Árboles de decisión, Random Forest.

El modelo con mayor precisión resultó Support Vector Machine.

Texto

Descripción generada automáticamente

Con el modelo definido y evaluado, se procede a la Etapa 2, en donde todo lo realizado se implementará en un pipeline para tener los datos preparados y luego se creará un training job para realizar el entrenamiento con sagemaker.

El procesamiento se encuentra en el archivo ‘sagemaker\_preprocessing\_pipeline.ipynb’. De este preprocesamiento se crean los archivos ‘train-V-1.csv’ y ‘test-V-1.csv’ que contiene el dataset limpio y normalizado listo para el entrenamiento.

El entrenamiento se realiza en el archivo ‘sagemaker\_train\_pipeline.ipynb’. Primero se define un cliente sagemaker para utilizar los servicios de AWS Sagemaker y también seleccionamos un bucket donde se llevará a cabo el almacenamiento de archivos de entrenamiento, en la cuenta de AWS se creó el bucket ‘retocreditriskkgv’.

Texto

Descripción generada automáticamente

Luego se almacenan los CSV de entrenamiento y prueba en el bucket:

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Lo siguiente es crear un script que será ejecutado en Sagemaker para realizar el entrenamiento.

%%writefile script.py  
  
import argparse  
  
from sklearn.svm import SVC  
from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix, precision\_score  
import sklearn  
import joblib  
import os  
import numpy as np  
import pandas as pd  
  
def model\_fn(model\_dir):  
 clf = joblib.load(os.path.join(model\_dir, "model.joblib"))  
 return clf  
  
if \_\_name\_\_== "\_\_main\_\_":  
 print("[INFO] Extracting arguments")  
 parser = argparse.ArgumentParser()  
   
 parser.add\_argument("--kernel", type=str, default="rbf")  
 parser.add\_argument("--C", type=float, default=1.0)  
 parser.add\_argument("--gamma", type=str, default="scale")  
   
 parser.add\_argument("--model-dir", type=str, default=os.environ.get("SM\_MODEL\_DIR"))  
 parser.add\_argument("--train", type=str, default=os.environ.get("SM\_CHANNEL\_TRAIN"))  
 parser.add\_argument("--test", type=str, default=os.environ.get("SM\_CHANNEL\_TEST"))  
 parser.add\_argument("--train-file", type=str, default="train-V-1.csv")  
 parser.add\_argument("--test-file", type=str, default="test-V-1.csv")  
   
 args, \_ = parser.parse\_known\_args()  
   
 print("SKLearn Version: ", sklearn.\_\_version\_\_)  
 print("Joblib Version: ", joblib.\_\_version\_\_)  
   
 print("[INFO] Reading data")  
 print()  
 train\_df = pd.read\_csv(os.path.join(args.train, args.train\_file))  
 test\_df = pd.read\_csv(os.path.join(args.test, args.test\_file))  
   
 features = list(train\_df.columns)  
 label = features.pop(-1)  
   
 print("Building training and testing datasets")  
 print()  
   
 X\_train = train\_df[features]  
 X\_test = test\_df[features]  
   
 y\_train = train\_df[label]  
 y\_test = test\_df[label]  
   
 print('Column order: ')  
 print(features)  
 print()  
   
 print('Label column is:', label)  
 print()  
   
 print("Data Shape: ")  
 print()  
 print("----SHAPE OF TRAINING DATA (80%)")  
 print(X\_train.shape)  
 print(y\_train.shape)  
 print()  
 print("----SHAPE OF TESTING DATA (20%)")  
 print(X\_test.shape)  
 print(y\_test.shape)  
 print()  
   
 print("Training SVM Model....")  
 print()  
 model = SVC(kernel=args.kernel, C=args.C, gamma=args.gamma, verbose=True)  
 model.fit(X\_train, y\_train)  
 print()  
   
 model\_path = os.path.join(args.model\_dir, "model.joblib")  
 joblib.dump(model, model\_path)  
 print("Model persisted at "+ model\_path)  
 print()  
   
 y\_pred\_test = model.predict(X\_test)  
 test\_acc = accuracy\_score(y\_test, y\_pred\_test)  
 test\_rep = classification\_report(y\_test, y\_pred\_test)  
   
 print()  
 print("--- METRICS RESULTS FOR TESTING DATA ---")  
 print()  
 print("Total Rows are:", X\_test.shape[0])  
 print("[TESTING] Model accuracy is: ", test\_acc)  
 print("[TESTING] Testing Report: ")  
 print(test\_rep)

Luego se utiliza el estimador de SKLearn de SageMaker para configurar y entrenar un modelo de clasificación SVM (Support Vector Machine), en una instancia gestionada de AWS SageMaker. El rol debe tener los permisos de Sagemaker y S3. La instancia seleccionada es ml.m5.large que es una opción razonable para entrenar modelos pequeños a medianos y además soporta instancias Spot lo que reduce el costo.

from sagemaker.sklearn.estimator import SKLearn  
  
FRAMEWORK\_VERSION = "0.23-1"  
  
sklearn\_estimator = SKLearn(  
 entry\_point="script.py",  
 role="arn:aws:iam::940583586544:role/sagemaker\_credit\_risk",  
 instance\_count=1,  
 instance\_type="ml.m5.large",  
 framework\_version=FRAMEWORK\_VERSION,  
 base\_job\_name = "RF-custom-sklearn",  
 hyperparameters={  
 "kernel": "rbf",  
 "C": 1.0,  
 "gamma": "scale"  
 },  
 use\_spot\_instances= True,  
 max\_wait = 7200,  
 max\_run = 3600  
)

Luego se procede a realizar el entrenamiento con los datos de los CSV:

sklearn\_estimator.fit({"train":trainpath, "test": testpath}, wait=True, logs=True)

Se obtuvieron las siguientes métricas:

Texto

Descripción generada automáticamente

Se observa que el recall, que es la sensibilidad y mide la capacidad del modelo para identificar correctamente todas las instancias positivas en un conjunto de datos, para la clase “bad risk” es baja, por lo que el modelo aún no resulta ser lo suficientemente eficiente.

Luego del entrenamiento, se genera un artifact en S3 (model.tar.gz). Este artifact será utilizado para el despliegue del modelo en un endpoint.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

## Despliegue

El modelo se despliega en el endpoint “Credit-risk-model-2024-12-10-19-46-04” usando una instancia ml.m5.large.

Texto

Descripción generada automáticamente

Se observa que el endpoint ha sido creado en AWS Sagemaker.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

## Inferencia

El notebook de inferencia es el archivo “sagemaker\_inference\_pipeline.ipynb” en donde se programa el pipeline de limpieza y normalización de datos utilizados en el entrenamiento y luego se llama al endpoint usando un Predictor de Sagemaker.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Finalmente, se realiza la predicción de datos. Para el ejemplo se utiliza como entrada todo el csv de datos en crudo inicial.

Texto

Descripción generada automáticamente

El monitoreo de todas las operaciones realizadas se lleva a cabo en AWS CloudWatch, tanto para el entrenamiento como las inferencias.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

# Conclusiones

Se desarrolló un pipeline simple de etiquetado automático, preprocesamiento, entrenamiento y despliegue de un modelo de Machine Learning para la clasificación de riesgo crediticio de clientes. Este enfoque es interesante dado su componente de etiquetado automático a partir de la potencia del LLM y del criterio de los prompts establecidos.