Credit Risk Analysis - Model Implementation Report

# Documentation and Commented Scripts

from google.colab import files  
uploaded = files.upload()

import pandas as pd  
import numpy as np  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler  
from sklearn.ensemble import RandomForestClassifier  
from xgboost import XGBClassifier  
from sklearn.metrics import classification\_report, confusion\_matrix  
from imblearn.over\_sampling import SMOTE  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
import warnings  
warnings.filterwarnings('ignore')

# Load training data  
df = pd.read\_csv('cs-training.csv')  
df = df.rename(columns={'SeriousDlqin2yrs': 'target'}) # Rename target column  
df.drop(columns=['Unnamed: 0'], inplace=True) # Remove index column if exists  
  
df.head()

# Show missing value counts  
print(df.isnull().sum())  
  
# Replace missing monthly income with median  
df['MonthlyIncome'] = df['MonthlyIncome'].fillna(df['MonthlyIncome'].median())  
  
# Fill NumberOfDependents NA with 0  
df['NumberOfDependents'] = df['NumberOfDependents'].fillna(0)

df['DebtRatio'] = df['DebtRatio'].clip(upper=1) # cap at 100%  
df['MonthlyDebt'] = df['DebtRatio'] \* df['MonthlyIncome']  
df['DebtPerPerson'] = df['MonthlyDebt'] / (df['NumberOfDependents'] + 1)  
  
df['RevolvingUtilizationOfUnsecuredLines'] = df['RevolvingUtilizationOfUnsecuredLines'].clip(upper=1)

X = df.drop('target', axis=1)  
y = df['target']

sm = SMOTE(random\_state=42)  
X\_resampled, y\_resampled = sm.fit\_resample(X, y)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_resampled, y\_resampled, test\_size=0.2, random\_state=42)  
  
scaler = StandardScaler()  
X\_train = scaler.fit\_transform(X\_train)  
X\_test = scaler.transform(X\_test)

rf = RandomForestClassifier(random\_state=42)  
rf.fit(X\_train, y\_train)  
y\_pred\_rf = rf.predict(X\_test)  
  
xgb = XGBClassifier(use\_label\_encoder=False, eval\_metric='logloss')  
xgb.fit(X\_train, y\_train)  
y\_pred\_xgb = xgb.predict(X\_test)

print("Random Forest Report:\n", classification\_report(y\_test, y\_pred\_rf))  
print("XGBoost Report:\n", classification\_report(y\_test, y\_pred\_xgb))  
  
sns.heatmap(confusion\_matrix(y\_test, y\_pred\_xgb), annot=True, fmt='d')  
plt.title('XGBoost Confusion Matrix')  
plt.show()

test\_df = pd.read\_csv('cs-test.csv')  
test\_df.drop(columns=['Unnamed: 0'], inplace=True)  
  
# Fill missing values as before  
test\_df['MonthlyIncome'] = test\_df['MonthlyIncome'].fillna(df['MonthlyIncome'].median())  
test\_df['NumberOfDependents'] = test\_df['NumberOfDependents'].fillna(0)  
  
# Same feature engineering  
test\_df['MonthlyDebt'] = test\_df['DebtRatio'] \* test\_df['MonthlyIncome']  
test\_df['DebtPerPerson'] = test\_df['MonthlyDebt'] / (test\_df['NumberOfDependents'] + 1)  
test\_df['RevolvingUtilizationOfUnsecuredLines'] = test\_df['RevolvingUtilizationOfUnsecuredLines'].clip(upper=1)  
  
# Drop the target column from the test set before scaling  
test\_df.drop(columns=['SeriousDlqin2yrs'], inplace=True)  
  
# Scale test set  
X\_test\_final = scaler.transform(test\_df)  
  
# Predict  
test\_predictions = xgb.predict(X\_test\_final)  
print(test\_predictions[:10]) # Sample predictions

def credit\_risk\_system(input\_data, model, scaler):  
 # Make a copy to avoid modifying original  
 df = input\_data.copy()  
  
 # Clean: Handle missing values  
 df['MonthlyIncome'] = df['MonthlyIncome'].fillna(df['MonthlyIncome'].median())  
 df['NumberOfDependents'] = df['NumberOfDependents'].fillna(0)  
  
 # Feature Engineering  
 df['MonthlyDebt'] = df['DebtRatio'] \* df['MonthlyIncome']  
 df['DebtPerPerson'] = df['MonthlyDebt'] / (df['NumberOfDependents'] + 1)  
 df['RevolvingUtilizationOfUnsecuredLines'] = df['RevolvingUtilizationOfUnsecuredLines'].clip(upper=1)  
  
 # Drop the target column ('SeriousDlqin2yrs') before scaling,  
 # as the scaler was fitted on data without this column.  
 if 'SeriousDlqin2yrs' in df.columns:  
 df.drop(columns=['SeriousDlqin2yrs'], inplace=True)  
  
 # Scaling  
 X\_scaled = scaler.transform(df)  
  
 # Predict  
 risk\_flags = model.predict(X\_scaled)  
  
 # Append predictions (append to the original df, as scaling is done on features)  
 # If you want the predictions appended to the dataframe after scaling,  
 # you need to handle the index alignment, or simply add it to the original df.  
 # Appending to the original df might be more useful if you want to see the  
 # original data alongside the prediction. Let's append to the original input\_data copy.  
 df['RiskFlag'] = risk\_flags # Add the risk flags to the processed dataframe before returning  
  
  
 return df  
  
# Rest of the code remains the same  
test\_df = pd.read\_csv('cs-test.csv')  
test\_df.drop(columns=['Unnamed: 0'], inplace=True)  
  
# Use XGBoost (or RandomForest) to flag high-risk customers  
flagged\_customers = credit\_risk\_system(test\_df, xgb, scaler)  
  
# View the first few flagged predictions  
flagged\_customers[['RiskFlag']].head(10)

print("High-Risk (1) vs Low-Risk (0):\n")  
print(flagged\_customers['RiskFlag'].value\_counts())

flagged\_customers[['RiskFlag']].to\_csv("flagged\_customers.csv", index=False)  
  
from google.colab import files  
files.download("flagged\_customers.csv")

import seaborn as sns  
import matplotlib.pyplot as plt  
  
# Bar chart of RiskFlag counts  
sns.countplot(x='RiskFlag', data=flagged\_customers, palette='coolwarm')  
plt.title('Distribution of Predicted Customer Risk')  
plt.xlabel('Risk Flag (0 = Low Risk, 1 = High Risk)')  
plt.ylabel('Number of Customers')  
plt.show()

# Markdown Notes (Contextual Descriptions)

Report Summary

# Dataset Preprocessing Steps

Describe the cleaning, normalization, encoding, and any other preprocessing steps used.

# Model Selection and Rationale

Explain the reasoning behind model selection for this specific task.

# Challenges Faced and Solutions

Discuss key challenges and how they were overcome during model development.

# Results with Visualizations and Interpretations

Summarize results and explain insights gained from visualizations and performance metrics.