Credit Score Prediction

Name: Shashank Gautam
Roll No.: 202401100300225
Course: Introduction to AI
Date: 11-03-2025

Introduction

In this project, we aim to classify individuals based on their credit scores using logistic regression. A credit score is an important financial metric that helps banks and lenders assess a person's creditworthiness. We will categorize credit scores into two classes:

- **Good (1)** if the score is above 600
- **Bad (0)** if the score is 600 or below

The dataset used for this project contains customer details such as age, income, loan amount, and credit score. By applying machine learning, we will predict whether a person has a good or bad credit rating.

Methodology

1. **Data Collection:**
    a. A CSV file containing credit score data was uploaded using Google Colab.
2. **Preprocessing:**
    a. Checked for missing values and removed any incomplete entries.
    b. Converted the **CreditScore** column into a binary classification:
        i. **Good Credit (1):** Credit score > 600
        ii. **Bad Credit (0):** Credit score ≤ 600
    c. Selected relevant features (**Age, Income, LoanAmount**) for model training.
3. **Data Splitting:**
    a. The dataset was divided into two subsets:
        i. **Training Set (80%)** - Used to train the model.
        ii. **Testing Set (20%)** - Used to evaluate performance.
4. **Feature Scaling:**
    a. Applied **StandardScaler()** to normalize the feature values, ensuring uniform distribution and better model performance.
5. **Model Training:**
    a. Trained a **Logistic Regression** model with **balanced class weights** to handle potential class imbalances.
6. **Prediction & Evaluation:**
    a. Used the trained model to predict credit categories on the test dataset.
    b. Evaluated performance using **accuracy score** to measure the model's effectiveness.

CODE

```
# Import necessary libraries
import pandas as pd  # For handling data in DataFrame format
```

```
from sklearn.model_selection import train_test_split  # To split data into training and testing sets
from sklearn.preprocessing import StandardScaler  # For normalizing numerical data
from sklearn.linear_model import LogisticRegression  # Logistic Regression model for classification
from sklearn.metrics import accuracy_score  # To evaluate model performance
from google.colab import files  # To allow file upload in Google Colab

# Prompt user to upload a CSV file in Google Colab
print("Please upload your CSV file")
uploaded = files.upload()  # Opens file upload dialog
csv_filename = list(uploaded.keys())[0]  # Get uploaded file name
df = pd.read_csv(csv_filename)  # Load dataset into a pandas DataFrame

# Check and remove missing values (if any)
df.dropna(inplace=True)  # Drops any rows with missing values to ensure clean data

# Convert 'CreditScore' into a binary category: 'Good' (1) or 'Bad' (0)
df['CreditCategory'] = df['CreditScore'].apply(lambda x: 1 if x > 600 else 0)

# Display the distribution of 'Good' vs 'Bad' credit scores
print("Class distribution:\n", df['CreditCategory'].value_counts())

# Select independent variables (features) and dependent variable (target)
X = df[['Age', 'Income', 'LoanAmount']]  # Features used for prediction
y = df['CreditCategory']  # Target variable (1 for good credit, 0 for bad credit)

# Split the dataset into training (80%) and testing (20%) sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the feature values to have mean 0 and variance 1 (better model performance)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)  # Fit and transform training data
X_test = scaler.transform(X_test)  # Transform test data using the same scaler

# Train the Logistic Regression model with balanced class weights (to handle imbalanced data)
model = LogisticRegression(class_weight='balanced')  # Adjusts for class imbalance
```

```
        model.fit(X_train, y_train)  # Train the model on training data

        # Make predictions using the trained model on the test dataset
                        y_pred = model.predict(X_test)

                # Calculate and print the accuracy of the model
                    accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')  # Display accuracy as a percentage (e.g.,
                                    85%)
```

OUTPUT

```
Please upload your CSV file
 Choose Files  credit_data.csv
• credit_data.csv(text/csv) - 472 bytes, last modified: 3/11/2025 - 100% done
Saving credit_data.csv to credit_data (11).csv
Class distribution:
 CreditCategory
1    12
0     8
Name: count, dtype: int64
Accuracy: 0.75
```

References/Credits

1. ChatGpt.
2. Internet.
3. Used libraries : google.colab, pandas, skyline.