

NAME- Kshitijendra Pratap Nagar

COURSE- Bachelor of Technology CSEAI

ROLL NO- 202401100300142

DATE- 11/03/2025

***Credit Score Prediction***

***Introduction***

***Credit score prediction refers to the process of estimating a person's creditworthiness based on various financial and personal data points. A credit score is a numerical representation of a person's ability to repay loans or debts, and it is widely used by lenders, banks, and other financial institutions to assess the risk associated with lending money.***

***Credit scores are calculated using factors such as:***

* ***Age: The age of the individual, as it may influence financial habits and stability.***
* ***Income: The income level, which helps determine the borrower's ability to repay.***
* ***Loan Amount: The total amount of money the person has borrowed or the total loan amount they are applying for.***
* ***Credit History: A record of the individual's past borrowing and repayment behavior, including any defaults or late payments.***
* ***Credit Utilization: How much of the available credit is being used, which reflects the borrower’s reliance on credit.***
* ***Other factors: Depending on the model, this can include employment history, number of credit accounts, and outstanding debts.***

**Methodology of Credit Score Prediction**

1. **Data Collection**: Gather financial and personal data like age, income, loan amount, credit history, and credit score.
2. **Data Preprocessing**:
   * Handle missing values (either by removing or filling them).
   * Transform categorical data into numerical values using encoding techniques.
   * Scale features to ensure all contribute equally to the model.
   * Remove outliers that could skew the results.
3. **Feature Selection**: Identify and select the most relevant features using correlation matrices or feature importance techniques.
4. **Model Selection**: Choose an appropriate machine learning model, such as:
   * **Random Forest Regressor** for robust performance.
   * **Linear Regression** for simpler models.
   * **Support Vector Machines** (SVM) for non-linear relationships.
5. **Model Training and Tuning**:
   * Train the model on the dataset.
   * Tune hyperparameters (e.g., number of trees in Random Forest) using Grid Search or Random Search.
6. **Model Evaluation**: Evaluate the model using metrics like:
   * **Mean Absolute Error (MAE)** and **Mean Squared Error (MSE)** for prediction accuracy.
   * **R-squared (R²)** to measure model’s explanatory power.
7. **Model Testing**: Test the model on a separate test dataset to ensure it generalizes well to unseen data.

***CODE TYPED***

# Import necessary libraries

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

from sklearn.preprocessing import StandardScaler

# Load the dataset

df = pd.read\_csv('credit\_data.csv')

# Check the first few rows of the dataset to understand its structure

print(df.head())

# Check for missing values

print("\nMissing Values:")

print(df.isnull().sum())  # Check for missing data

# If there are missing values, handle them (e.g., drop rows with missing values)

df = df.dropna()  # You can also choose to fill missing values with df.fillna() if preferred

# Defining features (X) and target (y)

X = df[['Age', 'Income', 'LoanAmount']]  # Features (independent variables)

y = df['CreditScore']  # Target (dependent variable)

# Scaling the features (optional but can help for better model performance)

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# Split the data into training and testing sets (80% training, 20% testing)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)

# Initialize the Random Forest Regressor

rf\_regressor = RandomForestRegressor(n\_estimators=100, random\_state=42)

# Train the model

rf\_regressor.fit(X\_train, y\_train)

# Make predictions on the test data

y\_pred = rf\_regressor.predict(X\_test)

# Evaluate the model

mae = mean\_absolute\_error(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(f'\nMean Absolute Error: {mae}')

print(f'Mean Squared Error: {mse}')

print(f'R-squared: {r2}')

# Create a DataFrame to compare Actual vs Predicted values

comparison\_df = pd.DataFrame({'Actual': y\_test, 'Predicted': y\_pred})

# Display the first few rows of Predicted vs Actual values

print("\nPredicted vs Actual values:")

print(comparison\_df.head())

# Plotting the Predicted vs Actual values (using matplotlib)

plt.figure(figsize=(10, 6))

plt.scatter(comparison\_df['Actual'], comparison\_df['Predicted'], color='blue')

plt.plot([comparison\_df['Actual'].min(), comparison\_df['Actual'].max()],

         [comparison\_df['Actual'].min(), comparison\_df['Actual'].max()],

         color='red', linestyle='--')  # Line of perfect prediction

plt.title('Predicted vs Actual Credit Score')

plt.xlabel('Actual Credit Score')

plt.ylabel('Predicted Credit Score')

plt.grid(True)

plt.show()

# Plotting the residuals (Predicted - Actual)

residuals = y\_pred - y\_test

plt.figure(figsize=(10, 6))

plt.scatter(y\_pred, residuals, color='green')

plt.axhline(y=0, color='red', linestyle='--')  # Zero line for residuals

plt.title('Residual Plot')

plt.xlabel('Predicted Credit Score')

plt.ylabel('Residuals (Predicted - Actual)')

plt.grid(True)

plt.show()

***ScreenShots Output and graph.***



