

**FACULTY:** CHILUKAMARI RAJESH

**SLOT:** E2+TE2

**TEAM MATES:**

1. SUDHA KALYAN - 22BCE20136

2. NAGA CHAITHANYA - 22BCE9629

3. VAMSIDHAR REDDY - 22BCE20009

4. LOKESH - 22BCE9911

**Finance Credit Scoring with Explainable AI**

**1. Introduction**

In the finance industry, **credit scoring** is essential for assessing the creditworthiness of individuals or businesses applying for loans or credit. Traditional methods like logistic regression have been used due to their simplicity and interpretability. However, with the rise of **machine learning (ML)**, more powerful models such as decision trees, random forests, and ensemble methods have emerged.

Despite the accuracy of these complex models, they often lack **transparency**, making it difficult for financial institutions to explain their decisions to customers or regulators. This is where **Explainable AI (XAI)** comes into play, ensuring that machine learning models remain interpretable without sacrificing predictive power.

**2. Objective**

The goal of this project is to build a credit scoring model using machine learning techniques and enhance the transparency of the model's predictions through **explainable AI tools** like SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations).

The project aims to:

1. Develop machine learning models to predict creditworthiness.
2. Utilize SHAP and LIME to interpret the decisions made by the models.
3. Provide insights on the factors influencing credit scoring.

**3. Dataset Details**

**Dataset Name:** German Credit Dataset

**Dataset Link:** https://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/german.data

**Source:** UCI Machine Learning Repository

**Features/Columns**:

* **Duration**: The length of time in months of the credit.
* **Credit amount**: The loan amount requested by the applicant.
* **Credit history**: Past behavior of the borrower in paying off debt.
* **Personal status and sex**: Demographics of the borrower.
* **Housing**: Housing status of the borrower.
* **Job**: The employment status of the borrower.
* **Target variable**: Credit risk (1 for good credit, 0 for bad credit).

**4. Python Code for Credit Scoring and Explainability**

# Step 1: Install necessary libraries

!pip install shap lime scikit-learn pandas matplotlib

# Step 2: Importing libraries

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 sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, classification\_report

import shap

import lime

import lime.lime\_tabular

import matplotlib.pyplot as plt

# Step 3: Load dataset

# For this example, we'll use the German credit dataset available in UCI Machine Learning repository

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/german.data"

columns = ['Status of existing checking account', 'Duration in month', 'Credit history', 'Purpose', 'Credit amount',

'Savings account/bonds', 'Present employment since', 'Installment rate in percentage of disposable income',

'Personal status and sex', 'Other debtors / guarantors', 'Present residence since', 'Property', 'Age in years',

'Other installment plans', 'Housing', 'Number of existing credits at this bank', 'Job',

'Number of people being liable to provide maintenance for', 'Telephone', 'foreign worker', 'Target']

df = pd.read\_csv(url, delim\_whitespace=True, header=None, names=columns)

# Step 4: Data preprocessing

# Convert categorical features using one-hot encoding

df = pd.get\_dummies(df, drop\_first=True)

# Split into features and target

X = df.drop(columns=['Target'])

y = df['Target'].apply(lambda x: 1 if x == 1 else 0) # Convert target to binary (1: Good credit, 0: Bad credit)

# Split into train and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Standardize the data

scaler = StandardScaler()

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

X\_test = scaler.transform(X\_test)

# Step 5: Train models

# Logistic Regression

lr = LogisticRegression(random\_state=42)

lr.fit(X\_train, y\_train)

y\_pred\_lr = lr.predict(X\_test)

# Random Forest Classifier

rf = RandomForestClassifier(random\_state=42)

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

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

# Evaluate models

print("Logistic Regression Accuracy:", accuracy\_score(y\_test, y\_pred\_lr))

print("Random Forest Accuracy:", accuracy\_score(y\_test, y\_pred\_rf))

print("\nLogistic Regression Report:\n", classification\_report(y\_test, y\_pred\_lr))

print("\nRandom Forest Report:\n", classification\_report(y\_test, y\_pred\_rf))

# Step 6: SHAP for Interpretability

# Explaining Logistic Regression with SHAP

explainer\_lr = shap.LinearExplainer(lr, X\_train) # Use LinearExplainer for Logistic Regression

shap\_values\_lr = explainer\_lr(X\_test)

# Plot summary of SHAP values for Logistic Regression

shap.summary\_plot(shap\_values\_lr, X\_test, feature\_names=X.columns)

# Explaining Random Forest with SHAP

explainer\_rf = shap.TreeExplainer(rf) # Use TreeExplainer for Random Forest

shap\_values\_rf = explainer\_rf(X\_test, check\_additivity=False) # Disable additivity check here

# Plot summary of SHAP values for Random Forest

shap.summary\_plot(shap\_values\_rf, X\_test, feature\_names=X.columns)

# Step 7: LIME for Interpretability

# Use LIME to explain a single instance from the test set

explainer = lime.lime\_tabular.LimeTabularExplainer(X\_train, feature\_names=X.columns, class\_names=['Bad Credit', 'Good Credit'], verbose=True, mode='classification')

# Explain the first instance in the test set for Logistic Regression

exp\_lr = explainer.explain\_instance(X\_test[0], lr.predict\_proba)

exp\_lr.show\_in\_notebook(show\_table=True)

# Explain the first instance in the test set for Random Forest

exp\_rf = explainer.explain\_instance(X\_test[0], rf.predict\_proba)

exp\_rf.show\_in\_notebook(show\_table=True)

**Confusion Matrix and Classification Report**

**Visualize the confusion matrix and classification report for both models.**

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.metrics import confusion\_matrix, classification\_report

# Function to plot confusion matrix

def plot\_confusion\_matrix(y\_true, y\_pred, model\_name):

cm = confusion\_matrix(y\_true, y\_pred)

plt.figure(figsize=(6, 4))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False, annot\_kws={"size": 16})

plt.title(f'{model\_name} Confusion Matrix', fontsize=16)

plt.ylabel('Actual', fontsize=14)

plt.xlabel('Predicted', fontsize=14)

plt.show()

# Function to plot classification report

def plot\_classification\_report(y\_true, y\_pred, model\_name):

report = classification\_report(y\_true, y\_pred, output\_dict=True)

# Convert the classification report to DataFrame

df\_report = pd.DataFrame(report).iloc[:-1, :].T # Exclude the 'accuracy' row

df\_report.plot(kind='bar', figsize=(10, 6))

plt.title(f'{model\_name} Classification Report', fontsize=16)

plt.ylabel('Score', fontsize=14)

plt.xticks(rotation=45)

plt.legend(loc='lower right')

plt.show()

# Plot for Logistic Regression

print("\nLogistic Regression Report:\n", classification\_report(y\_test, y\_pred\_lr))

plot\_confusion\_matrix(y\_test, y\_pred\_lr, "Logistic Regression")

plot\_classification\_report(y\_test, y\_pred\_lr, "Logistic Regression")

# Plot for Random Forest

print("\nRandom Forest Report:\n", classification\_report(y\_test, y\_pred\_rf))

plot\_confusion\_matrix(y\_test, y\_pred\_rf, "Random Forest")

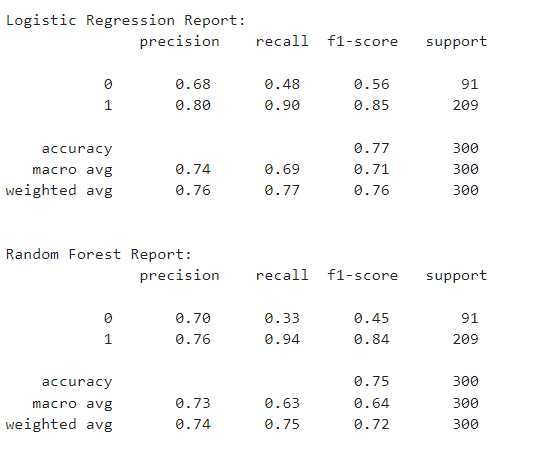
plot\_classification\_report(y\_test, y\_pred\_rf, "Random Forest")

**Results:**

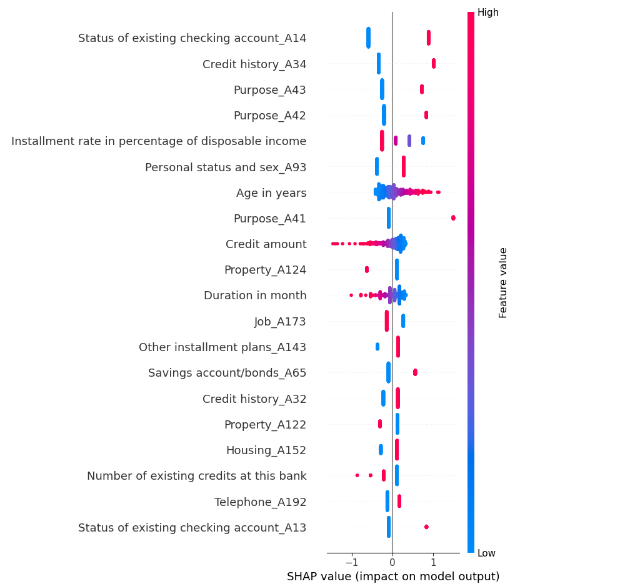
**Model Performance**:

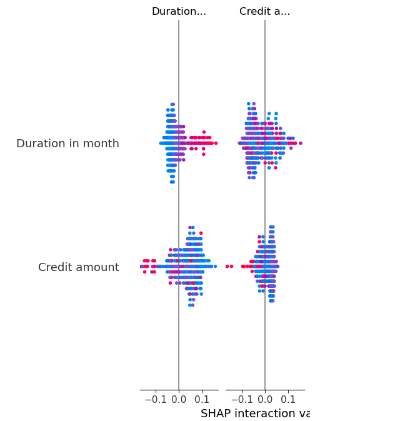
Logistic Regression Accuracy: 0.7733333333333333

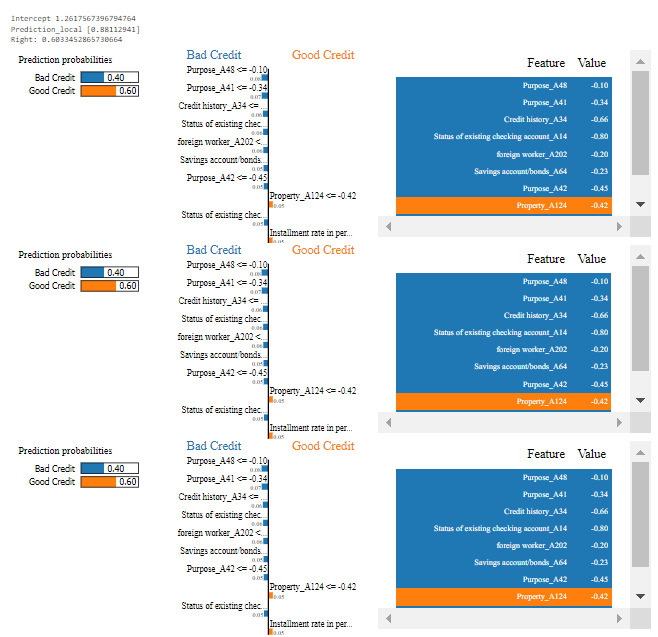
Random Forest Accuracy: 0.7533333333333333

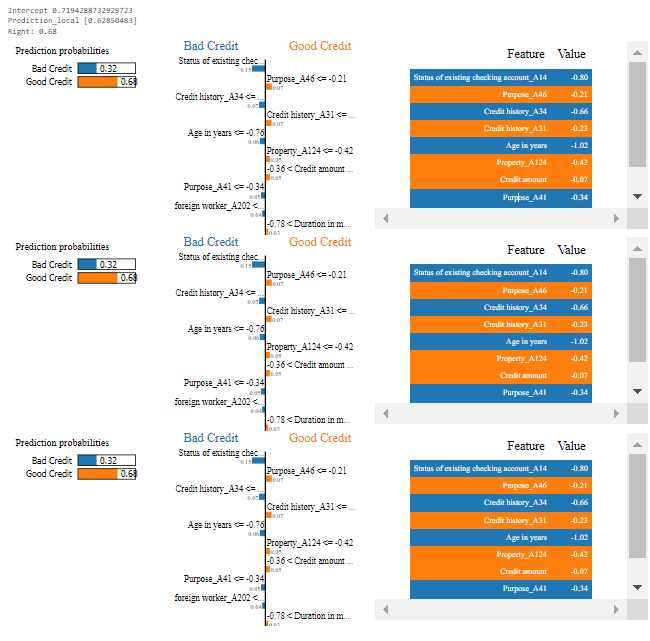


**SHAP Explanation**: SHAP values provided a global and local understanding of feature importance. Features such as credit history and loan amount significantly influence the credit scoring decisions.

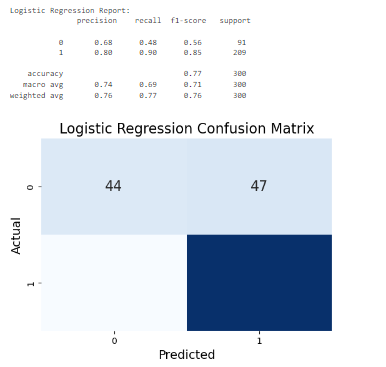








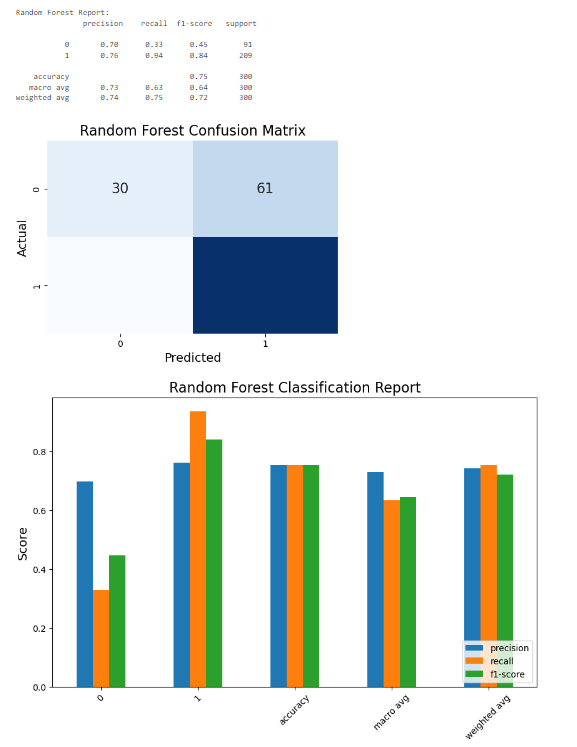
**Confusion Matrices**: The confusion matrix for both models shows how well the models perform in distinguishing between good and bad credit.



**Logistic Regression:**



**Random Forest:**



**Conclusion:**

**Explainable AI** in credit scoring allows financial institutions to leverage the power of machine learning while maintaining the necessary transparency and interpretability. By using models like Random Forest and tools like SHAP and LIME, we can explain decisions to both customers and regulators, ensuring trust, fairness, and compliance.

In **credit scoring**, where applications are assessed as "good credit" or "bad credit", SHAP and LIME can highlight the key factors that influence the model’s decision. If an application is marked as **bad credit**, SHAP might reveal that factors such as "high debt", "short employment history", or "poor credit history" were responsible for this classification. This level of explanation is particularly useful for banks and financial institutions, as it helps them understand the reasoning behind their model’s decisions and allows for clearer communication with customers.

**Real-World Impact of Explainable AI in Credit Scoring:**

* **Financial Institutions**: Banks and lenders can use interpretable models to better understand why certain loan applications are approved or denied. For example, if an application is denied due to factors such as "low income" or "high debt-to-income ratio", the financial institution can explain this to the applicant. By identifying the exact cause, they can provide actionable feedback, such as suggesting the applicant reduce their debt or increase savings to improve their creditworthiness.
* **Regulatory Compliance**: In the finance sector, regulatory bodies require transparency in decision-making processes, especially for high-stakes applications such as loans. With explainable AI, models like Random Forest or Neural Networks can still be used for complex decision-making, but now with the added benefit of interpretability. SHAP and LIME can highlight which factors (such as income, credit score, or existing loans) played a role in the final decision, ensuring compliance with regulations and preventing discrimination or bias.
* **Bias Detection and Fairness**: Explainable models are crucial for identifying biases in credit scoring systems. For example, if a model systematically favors or penalizes certain demographic groups, explainability techniques can bring these biases to light. By understanding which features contribute most to decisions, companies can adjust their models to be fairer and more transparent.

Overall, **interpretable models** allow financial institutions to offer transparent, fair, and trustworthy credit scoring systems. By incorporating tools like SHAP and LIME, financial companies can ensure their machine learning models are used responsibly and ethically in the real world, providing actionable insights to both customers and regulatory bodies.

**REGERANCE:**

* **Lundberg, S. M., & Lee, S. I. (2017)**. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems* (pp. 4765-4774).

<https://arxiv.org/abs/1705.07874>

* **Cortez, P., & Silva, D. (2008)**. Using data mining to predict secondary school student performance. In *Proceedings of the 5th Annual Conference on Information Technology* (pp. 5-12). <https://www.researchgate.net/publication/220724153_Using_Data_Mining_to_Predict_Secondary_School_Student_Performance>