Team Name: Lancelot

Team Members:

- IMT2022023 Ananthula Harshith Reddy
- IMT2022570 Mupparapu Koushik
- IMT2022052 Narayana Srimanth

1. Introduction

This project aims to develop a machine learning pipeline to predict the credit scores of customers based on financial, demographic, and behavioral data. Multiple classification algorithms, including **Decision Tree**, **Random Forest**, **XGBoost**, **K-Nearest Neighbors (KNN)**, and **Naive Bayes**, were evaluated. The pipeline leverages SMOTE for handling class imbalance, advanced hyperparameter tuning, and robust evaluation techniques.

2. Data Processing Steps

2.1 Handling Special Characters

- Numeric columns with irregular formats (e.g., non-numeric characters) were sanitized using regular expressions.
- Numeric columns such as `Age`, `Income_Annual`, and `Credit_Limit` were cleaned by removing special characters (e.g., `$`, `,`) to ensure consistency.

2.2 Missing Data

- **Numeric Columns**: Missing values were imputed using **K-Nearest Neighbors (KNN)** to predict values based on similar records.
- **Categorical Columns**: Missing values in `Profession`, `Credit_Mix`, and `Payment_Behaviour` were replaced using the mode within grouped categories.

2.3 Feature Transformation

- **Credit History Age**: Converted from `"X Years Y Months"` to total months for numerical analysis.
- **Loan Types**: Transformed into multiple binary columns using **MultiLabelBinarizer**, improving feature representation for machine learning models.

2.4 Feature Selection and Scaling

- **Numeric Features**: Included attributes such as Age, `Total_Current_Loans`, and `Monthly_Balance`.
- **Categorical Features**: Included variables like `Profession`, `Credit_Mix`, and `Payment_Behaviour`.
- **Scaling**: Features were scaled using **RobustScaler** to normalize data and reduce the effect of outliers.

2.5 Class Imbalance Handling

- **SMOTE (Synthetic Minority Oversampling Technique)** was applied to balance the dataset, addressing class imbalance in the target variable.

3. Models Used

3.1 Decision Tree Classifier

- A tree-based algorithm that divides data into hierarchical decision rules. It is interpretable but susceptible to overfitting.

3.2 Random Forest Classifier

- An ensemble of decision trees that averages predictions, offering improved generalization and reduced overfitting.

3.3 XGBoost Classifier

- A gradient boosting method known for handling complex data relationships and delivering high accuracy in classification tasks.

3.4 K-Nearest Neighbors (KNN) Classifier

- A distance-based algorithm that classifies based on the majority label of nearest neighbors.

3.5 Naive Bayes Classifier

- A probabilistic model based on Bayes' theorem, assuming independence between features for simplicity and speed.

4. Hyperparameter Tuning

GridSearchCV was used to fine-tune model parameters.

4.1 Decision Tree

- Parameters: `criterion`, `max_depth`, `min_samples_split`, `min_samples_leaf`.

4.2 Random Forest

- Parameters: `n_estimators`, `max_depth`, `min_samples_split`, `min_samples_leaf`, and `bootstrap`.

4.3 XGBoost

- Parameters: `n_estimators`, `max_depth`, `learning_rate`, and `gamma`.

4.4 KNN

- Parameters: `n_neighbors`, `weights` (uniform or distance), and distance metrics (`euclidean`, `manhattan`).

4.5 Naive Bayes

- Parameter: `var_smoothing` for managing small variance values in feature distributions.

5. Discussion on the Performance of Different Approaches

5.1 XGBoost

- Delivered the best overall performance due to its boosting mechanism and ability to handle complex relationships.

5.2 Random Forest

- Performed robustly with a balanced approach to bias and variance but slightly lagged behind XGBoost in accuracy.

5.3 Decision Tree

- Provided reasonable accuracy but was prone to overfitting, which limited its performance on unseen data.

5.4 KNN

- Performance was sensitive to hyperparameter selection (k and distance metrics) and struggled with higher-dimensional data.

5.5 Naive Bayes

- Performed adequately but underperformed compared to other models due to its independence assumption, which does not fully hold in the dataset.

6. Final Model Selection

XGBoost was selected as the final model for predictions due to its superior accuracy and generalization capabilities.

7. Test Set Prediction

The best-performing **XGBoost** model was used to predict `Credit_Score` for the test set. Predictions were mapped back to categorical values (`Poor`, `Standard`, `Good`) and saved in a file named `credit_score_predictions.csv`.

8. Conclusion

This project successfully implemented a machine learning pipeline for credit score prediction. By addressing class imbalance with SMOTE, applying robust feature engineering, and evaluating multiple models, the XGBoost classifier demonstrated the best performance. Future improvements could include additional feature engineering and the exploration of advanced ensemble techniques.

9. GitHub Link

The source code used to generate the predictions and train the models can be found in the following GitHub repository: https://github.com/Harshith2835/ml_course_proj