Identifying Loan Completion Status (Classification Task)

Supervised by: Dr. Manal Mohsen Tantawy

Dr. Sarah Nabil & Dr. Ahmed Talaat

Done by: AbdelRahman Mohamed Aly

Ahmed AbdelRahman & Bassel AbdelRahim

1. Data Exploration and Statistical Analysis

Before delving into any preprocessing techniques, it is crucial to thoroughly explore the data to understand its structure, characteristics, and any underlying patterns or issues. This initial exploration phase helps in determining which preprocessing techniques will be necessary for effective data cleaning and preparation. The following steps outline the process of data exploration and statistical analysis:

1.1 Understanding the Data

1. Loading the Data: begin by loading the dataset into a suitable environment for analysis, such as Python (using pandas), R, or any other data analysis tool.

2. Basic Information: Examine the basic structure of the dataset:

- **Shape:** Check the dimensions of the dataset to understand how many rows and columns it contains.
- **Columns:** Review the column names and data types to ensure they align with expectations.
- **Summary Statistics:** Generate summary statistics (mean, median, standard deviation, minimum, and maximum values) for numerical columns to get an initial sense of the data distribution.

1.2 Descriptive Statistics and Visualization

1. Descriptive Statistics: Perform a detailed statistical analysis:

- **Central Tendency:** Calculate mean, median, and mode for numerical variables.
- **Dispersion:** Assess the spread of the data using measures like range, variance, and standard deviation.

2. Data Visualization: Visualize the data to identify patterns, trends, and anomalies:

- **Histograms:** Plot histograms for numerical variables to observe their distribution.
- **Box Plots:** Use box plots to identify outliers and understand the distribution of the data.
- **Scatter Plots:** Create scatter plots to examine relationships between pairs of numerical variables.

By conducting a thorough data exploration and statistical analysis, you can make informed decisions about the necessary preprocessing techniques to apply. This approach ensures that the data is clean, consistent, and ready for further analysis or modeling, ultimately leading to more reliable and accurate results.

2. Data Preprocessing:

Before building predictive models, it is essential to preprocess the dataset to ensure it is clean and ready for use. Effective preprocessing can significantly enhance the performance of machine learning models by addressing various data issues. This section outlines key preprocessing steps, including handling missing values and categorical encoding, among other techniques.

2.1 Identifying and Handling Missing Values:

Missing values can lead to biased results and reduced model accuracy if not handled appropriately. The following steps detail how to manage missing data:

1. Identification:

- **Missing Data Report:** Generate a report to identify the percentage of missing values in each column.
- **Visualization:** Use heatmaps or bar plots to visualize the distribution and pattern of missing data across the dataset.

2. Handling Missing Values:

- Imputation: Mean /Mode (Most Frequent) Imputation: For numerical features, replace missing values with the mean, median, or mode of the column.

- Removal:

- **Dropping Rows:** Remove rows with missing values if the proportion of missing data is relatively low and their removal will not bias the analysis.

- **Dropping Columns:** Drop columns with a high percentage of missing values if they are deemed uninformative or if their imputation is not feasible.
-

2.2 Categorical Encoding:

Machine learning algorithms typically require numerical input, necessitating the conversion of categorical data into numerical formats. The following techniques can be used for categorical encoding:

- Label Encoding:

- **Implementation**: Assign a unique integer to each category in a categorical feature.
- **Usage:** Best for ordinal categorical variables where the categories have a meaningful order.

2.3 Additional Preprocessing Techniques:

Beyond handling missing values and categorical encoding, several other preprocessing steps may be necessary to ensure the dataset is in optimal condition for model building:

1. Feature Scaling:

- Standardization: Apply standardization to scale features to have zero mean and unit variance.

2. Inconsistencies:

- Check for inconsistencies in the data.
- Duplicate Entries: Identify and handle duplicate records (Drop Duplicates).
- Data Entry Errors: Look for typographical errors or inconsistencies in categorical data.

3. Data Splitting:

- Train-Test Split: Divide the dataset into training and testing sets to evaluate model performance.
- Cross-Validation: Use k-fold cross-validation to ensure the model's robustness and generalizability.

By meticulously preprocessing the data, including handling missing values, encoding categorical variables, and applying additional techniques like scaling and outlier treatment, you ensure that the dataset is clean and ready for effective model building. This careful preparation is crucial for achieving accurate and reliable model performance.

3. Feature Selection:

- We used the train_test_split method from pandas to separate the training dataset into features (X_train) and the target variable (y_train), with X_test representing the test features. To standardize the feature values, we employed StandardScaler from scikit-learn, which resulted in scaled versions of X_train and X_test.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest, f_classif

X_train = train.drop(columns=['completion_status'])
y_train = train['completion_status']

X_test = test

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

- We applied the SelectKBest method with the f_classif function to select the top 28 features most relevant to predicting the target variable. This process resulted in X_train_selected and X_test_selected, containing only the chosen features. We then transformed these selected features back into DataFrame format for easier interpretation and further analysis.

```
k = 28
selector = SelectKBest(score_func=f_classif, k=k)
X_train_selected = selector.fit_transform(X_train_scaled, y_train)
X_test_selected = selector.transform(X_test_scaled)

selected_features = selector.get_support(indices=True)
print("Selected features:", X_train.columns[selected_features])

X_train_selected_df = pd.DataFrame(X_train_selected, columns=X_train.columns[selected_features])
X_test_selected_df = pd.DataFrame(X_test_selected, columns=X_test.columns[selected_features])
```

- as a result, the selected features were identified and printed, which included important variables like 'owner_1_score', 'years_in_business', 'fsr', 'funded_last_30', among others. This selection process helped in focusing on the most impactful features for our machine learning model.

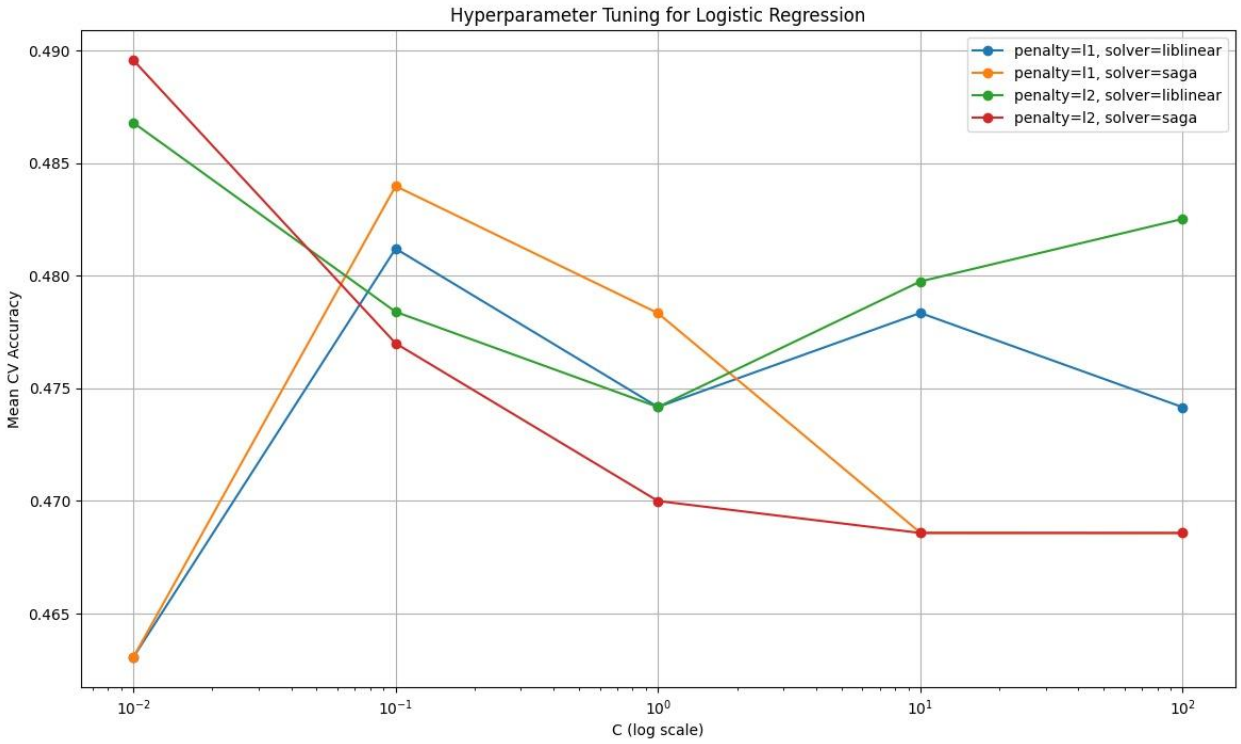
- We initially attempted to use Principal Component Analysis (PCA) for feature reduction; however, upon evaluation, we found that the SelectKBest method with the f_classif function provided significantly better results. This method enabled us to select the top 28 most relevant features, leading to more accurate and interpretable outcomes in our machine learning model.

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
selected_columns = train[[
                        'INPUT_VALUE_ID_FOR_current_position',
                        'location',
                        'deal_application_thread_id',
                        'INPUT_VALUE_ID_FOR_fc_margin',
                        'RATE_ID_FOR_num_deposits',
                        'INPUT_VALUE_ID_FOR_average_ledger',
                        'RATE_ID_FOR_current_position',
                        'fsr',
                        'INPUT_VALUE_ID_FOR_num_deposits',
                        'RATE_ID_FOR_num_negative_days',
                        'RATE_ID_FOR_years_in_business',
                           'years_in_business',
                           'INPUT_VALUE_ID_FOR_num_negative_days',
                           'funded_last_30',
                           'RATE_ID_FOR_location']]
scaler = StandardScaler()
std = scaler.fit_transform(selected_columns)
pca = PCA(n_components=1)
pca.fit(std)
X_pca = pca.transform(std)
X_pca.shape
print("\nExplained Variance Ratio:", pca.explained_variance_ratio_)
```

4. Models and Results:

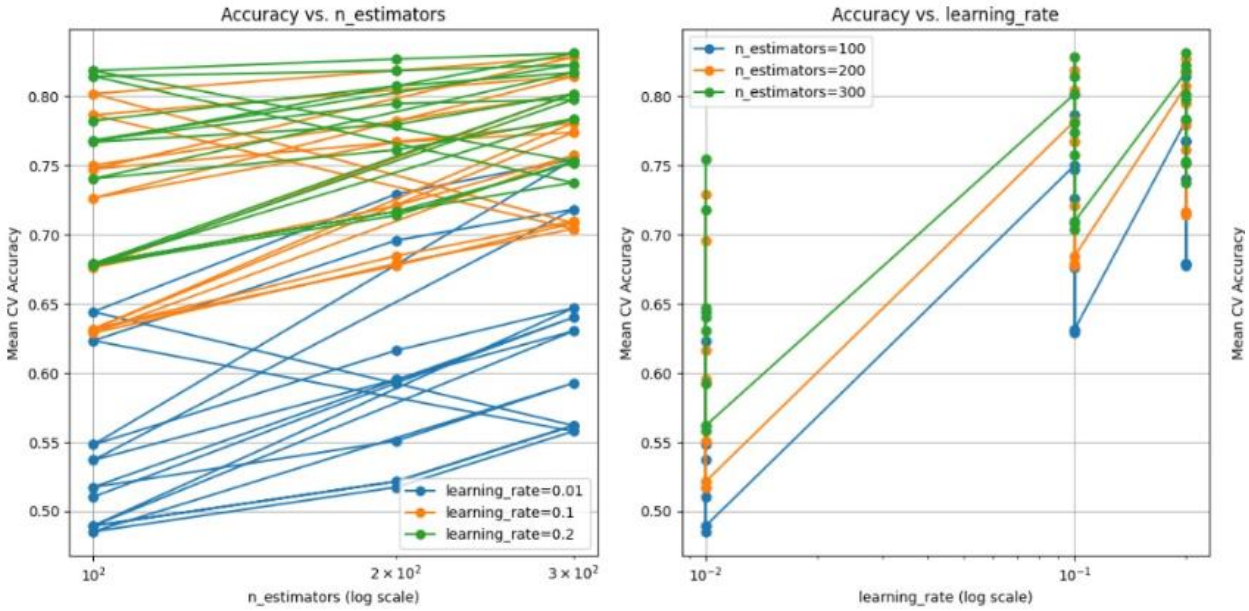
4.1 Logistic Regression:

- Logistic Regression is a popular statistical method used for binary classification problems, where the outcome or dependent variable is categorical and has only two possible values, typically labeled as 0 and 1. It's a type of regression analysis that estimates the probability that a given input belongs to a particular class.
- We applied it on our data set and that is the results:



4.2 LightGBM (Light Gradient Boosting Machine):

- Is a high-performance gradient boosting framework developed by Microsoft. It's designed for efficiency, scalability, and accuracy, making it one of the most popular machine learning algorithms for both regression and classification tasks.
- The Results:

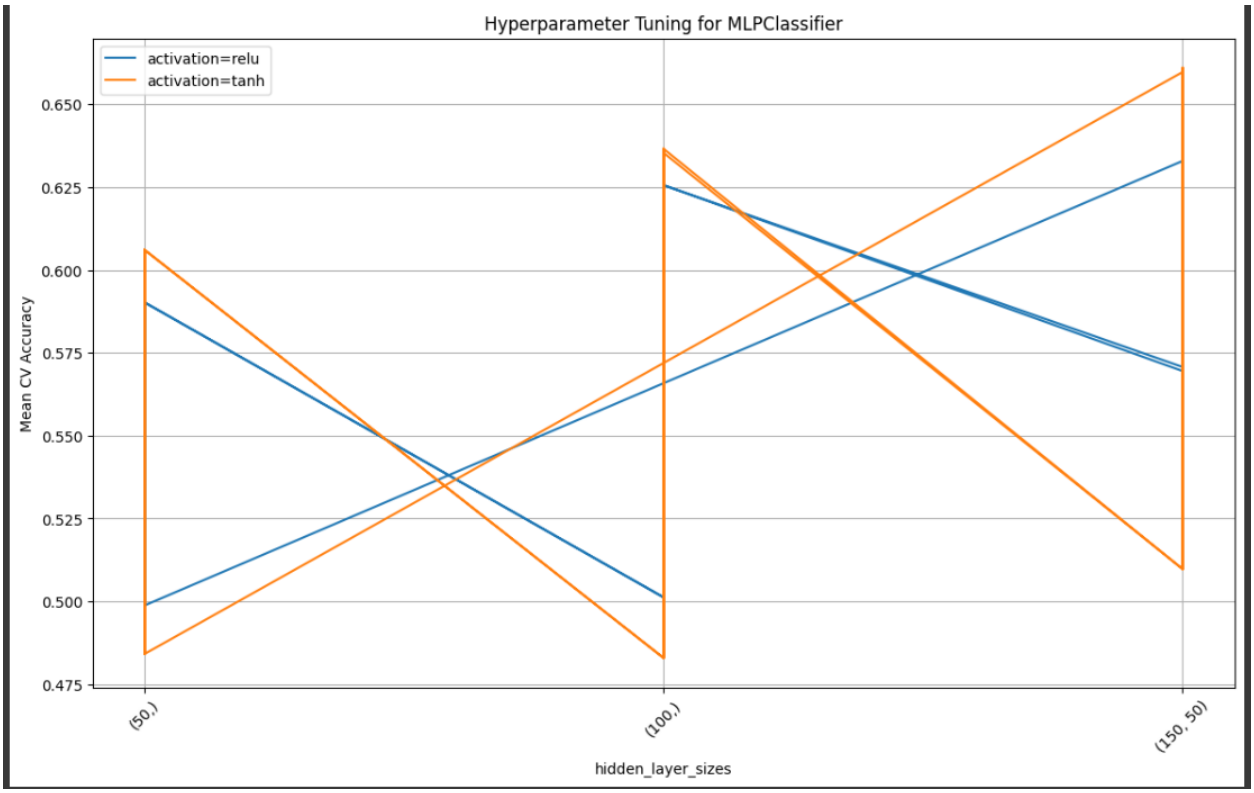



```
Best Parameters: {'learning_rate': 0.2, 'max_depth': 7, 'min_child_samples': 20, 'n_estimators': 300, 'num_leaves': 31}
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
Training Score: 1.0
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
Validation Score: 0.8928571428571429
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
              precision    recall  f1-score   support

           0       0.90      0.89      0.90        82
           1       0.89      0.94      0.92       145
           2       0.88      0.85      0.86        67
           3       1.00      0.57      0.73        14

    accuracy                           0.89       308
   macro avg       0.92      0.81      0.85       308
weighted avg       0.89      0.89      0.89       308

Accuracy: 0.8928571428571429
Precision: 0.8949629455040711
Recall: 0.8928571428571429
F1 Score: 0.8908102645675089
Sensitivity: 0.8143115821828022
Specificity: 0.8143115821828022
Confusion Matrix:
[[ 73   8   1   0]
 [  4 137   4   0]
 [  2   8  57   0]
 [  2   1   3   8]]
```

4.3 Multilayer Perceptron (MLP):

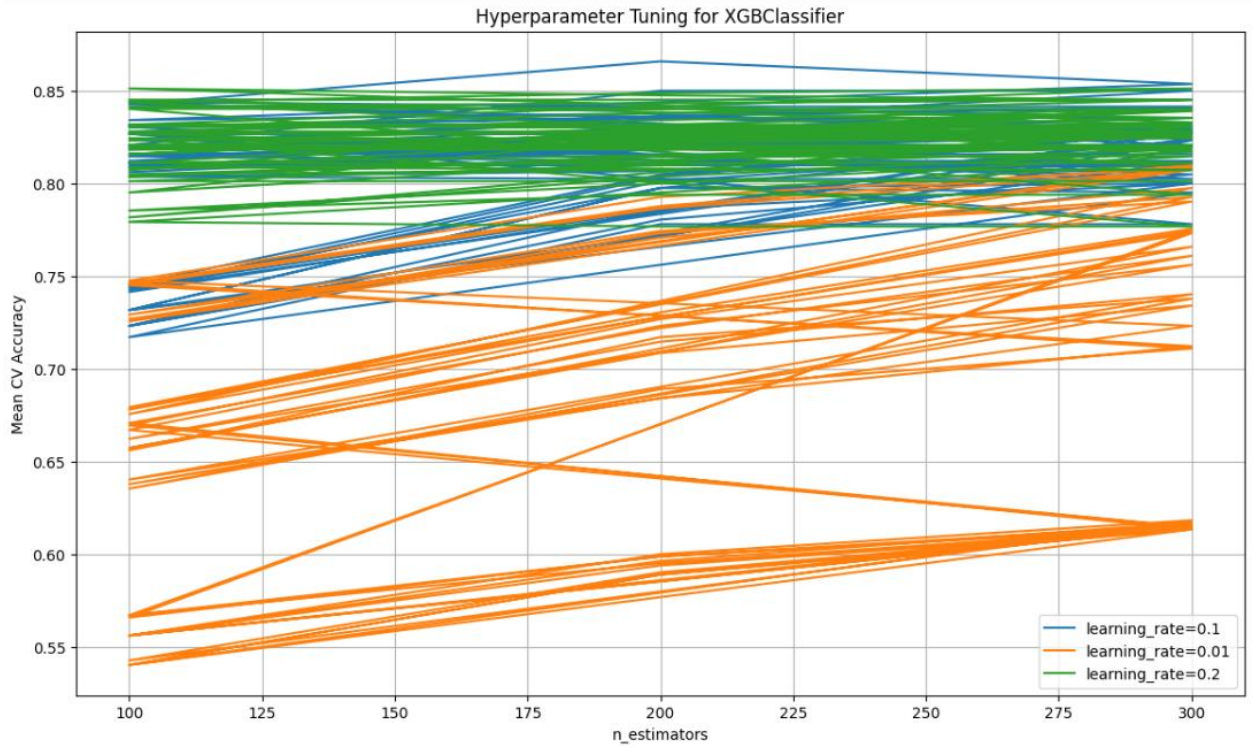
- is a type of artificial neural network (ANN) widely used in machine learning for both regression and classification tasks.
- The Results:

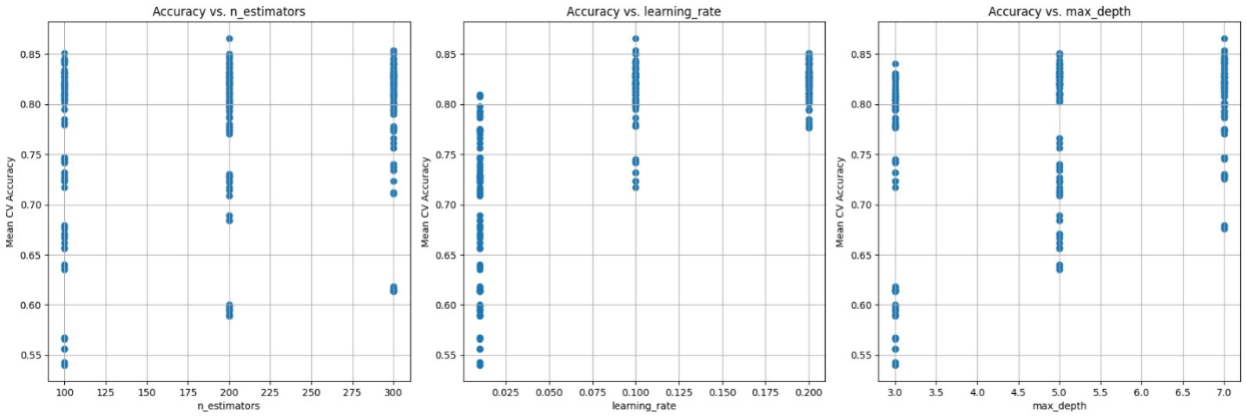
```
Best Parameters: {'activation': 'tanh', 'alpha': 0.001, 'hidden_layer_sizes': (150, 50), 'learning_rate': 'constant', 'solver': 'adam'}
Training Score: 0.953170731707317
Validation Score: 0.9609756097560975
              precision    recall  f1-score   support

           0       1.00      0.96      0.98        46
           1       0.95      0.97      0.96       101
           2       0.94      0.98      0.96        48
           3       1.00      0.80      0.89        10

    accuracy                           0.96       205
   macro avg       0.97      0.93      0.95       205
weighted avg       0.96      0.96      0.96       205

Accuracy: 0.9609756097560975
Precision: 0.9620345725787355
Recall: 0.9609756097560975
F1 Score: 0.9607156033144965
Sensitivity: 0.9264963588750179
Specificity: 0.9264963588750179
Confusion Matrix:
[[44  2  0  0]
 [ 0 98  3  0]
 [ 0  1 47  0]
 [ 0  2  0  8]]
```



4.4 XGBoost (Extreme Gradient Boosting):

- Is a powerful and efficient implementation of gradient boosting machines, a popular machine learning algorithm known for its performance and flexibility.
- Results:





4.5 K-Nearest Neighbors (KNN):

- KNN makes predictions based on the similarity of data points. It assumes that similar things are close to each other in the feature space.
- When given a new data point, KNN finds the 'k' closest data points (neighbors) in the training dataset.