Credit Ranking

Bijeet Basak(IMT2022510)
Dikshant Mahawar(IMT2022549)

Github: https://github.com/bijeet1221/Credit-Ranking

Preprocessing:

1. **Loan_Type**: Transformed column, which contained comma-separated strings, into a numerical feature **`Loan_Count`** by counting the number of loan types using a custom function. This operation converted a categorical text field into a meaningful numerical representation for the model.
2. **Credit_History_Age**: Converted the **`Credit_History_Age`** column, which contained values in the format "X Years and Y Months," into a numerical feature representing the total number of months. This was achieved by parsing the string to extract years and months, multiplying years by 12, and adding the months.
3. **numeric_cols** = ['Age', 'Income_Annual', 'Base_Salary_PerMonth','Total_Bank_Accounts', 'Total_Credit_Cards', 'Rate_Of_Interest', 'Total_Current_Loans', 'Delay_from_due_date','Total_Delayed_Payments', 'Credit_Limit','Total_Credit_Enquiries', 'Current_Debt_Outstanding','Ratio_Credit_Utilization', 'Per_Month_EMI','Monthly_Investment', 'Monthly_Balance', 'Loan_Count']

   We cleaned and converted the **`numeric_cols`** columns, which contained numeric data stored as strings, into proper numeric types by removing non-numeric characters and coercing the data into numerical format. Missing values were then imputed using the median value of each column, grouped by **`Customer_ID`**, and further filled with the overall column median if needed.
4. **Credit_History_Age:** We handled missing values in the column by imputing NaN values within each **`Customer_ID`** group with the mode of that group. If the group mode was unavailable, the overall mode of the column was used. Any remaining NaN values were filled with the overall mode, ensuring no missing values remained in the dataset.
5. We resolved issues in categorical columns (**`Credit_Mix`, `Payment_Behaviour`, `Payment_of_Min_Amount`, `Profession`**) where certain placeholder values (e.g., **`_`, `!@9#%8`, `NM`, `_______`**) needed replacement. Each problematic value was replaced by the mode within its

`Customer_ID` group, with a fallback to the overall column mode if the group mode was unavailable. This ensured consistency and accuracy in categorical data imputation.

For handling missing values in categorical columns by imputing NaN values within each **`Customer_ID`** group using the mode of that group. If the group mode was unavailable, missing values were filled with the overall column mode, ensuring all categorical columns were complete and ready for analysis.
Label encoding is performed to the categorical columns, converting them into numerical values suitable for machine learning models.

6. **Profession**: Applied label encoding to the column,converting them into numerical values suitable for machine learning models.
7. The same preprocessing steps, including handling missing values, imputing categorical columns, label encoding, and dropping unnecessary columns, were consistently applied to the test dataset to ensure compatibility.
8. **Credit_Score**: This column was mapped to numerical values using the dictionary. This conversion transformed the categorical credit score labels into numerical representations for use in the machine learning model.

Models:

1. **Decision Tree**:
    a. Training Accuracy: 0.71
    b. Test Accuracy: 0.68569
    c. Hyperparameter list:
        {'max_depth': [None] + list(range(3, 21)),
        'min_samples_split': range(2, 21),
        'min_samples_leaf': range(1, 21),
        'criterion': ['gini', 'entropy'],
        'max_features': [None, 'sqrt', 'log2'],
        'splitter': ['best', 'random'],
        'class_weight': [None, 'balanced'],
        'min_impurity_decrease': [0.0, 0.01, 0.1],
        'ccp_alpha': [0.0, 0.01, 0.1]}
    d. Best parameters found by **RandomizedSearchCV**:
    {'splitter': 'best', 'min_samples_split': 8, 'min_samples_leaf': 2, 'min_impurity_decrease': 0.0, 'max_features': None, 'max_depth': 8, 'criterion': 'gini', 'class_weight': None, 'ccp_alpha': 0.0}

2. **Gradient Boost:**
    a. Training Accuracy: 0.80
    b. Test Accuracy: 0.77770
    c. Hyperparameter list:
    { 'n_estimators': (50, 300), 'learning_rate': (0.01, 1.0), 'max_depth': (3, 10), 'min_samples_split': (2, 20),'min_samples_leaf': (1, 20), 'max_features': ['sqrt', 'log2', None], 'subsample': (0.6, 1.0), 'loss': ['log_loss']}

    d. Best parameters found by **Bayes_Search**:
    'learning_rate', 0.25661936734235113), ('loss', 'log_loss'), ('max_depth', 10), ('max_features', None), ('min_samples_leaf', 2), ('min_samples_split', 3), ('n_estimators', 300), ('subsample', 0.8302611906636985)]
3. **Random Forest**:
    a. Training Accuracy: 0.713975
    b. Test Accuracy: 0.68677
    c. Hyperparameter list:
    {'n_estimators': trial.suggest_int('n_estimators', 50, 200),
    'max_depth': trial.suggest_int('max_depth', 3, 15),
    'max_features': trial.suggest_categorical('max_features', ['sqrt','log2', None]),
    'max_leaf_nodes': trial.suggest_int('max_leaf_nodes', 10, 50),
    'min_samples_split': trial.suggest_int('min_samples_split', 2, 10),
    'min_samples_leaf': trial.suggest_int('min_samples_leaf', 1, 5),
    'bootstrap': trial.suggest_categorical('bootstrap', [True, False]),
    'random_state': 42}
    d. Best parameters found by **Optuna**:
    {'n_estimators': 176, 'max_depth': 15, 'max_features': None, 'max_leaf_nodes': 46, 'min_samples_split': 10, 'min_samples_leaf': 5, 'bootstrap': True}

4. **AdaBoost**:
    a. Training Accuracy: 0.703975
    b. Test Accuracy: 0.68534
    c. Hyperparameter list:
    {'n_estimators': trial.suggest_int('n_estimators', 50, 200),

    'max_depth': trial.suggest_int('max_depth', 3, 15),
    'max_features': trial.suggest_categorical('max_features', ['sqrt','log2', None]),
  'max_leaf_nodes': trial.suggest_int('max_leaf_nodes', 10, 50),
  'min_samples_split': trial.suggest_int('min_samples_split', 2, 10),
  'min_samples_leaf': trial.suggest_int('min_samples_leaf', 1, 5),
  'bootstrap': trial.suggest_categorical('bootstrap', [True, False]),
  'random_state': 42}

d. Best parameters found by **Bayes_Search**:
   {'n_estimators': 176, 'max_depth': 15, 'max_features': None, 'max_leaf_nodes': 46, 'min_samples_split': 10, 'min_samples_leaf': 5, 'bootstrap': True}

5. **Xgboost**:
   a. Training accuracy: 0.9988375
   b. Test accuracy: 0.79192
   c. Hyperparameter list:

```
param = {
        'n_estimators': trial.suggest_int('n_estimators', 50, 300),
        'max_depth': trial.suggest_int('max_depth', 3, 15),
        'learning_rate': trial.suggest_float('learning_rate', 0.01, 0.3),
        'subsample': trial.suggest_float('subsample', 0.5, 1.0),
        'colsample_bytree': trial.suggest_float('colsample_bytree', 0.5,
1.0),
        'gamma': trial.suggest_float('gamma', 0.0, 0.5),
        'min_child_weight': trial.suggest_int('min_child_weight', 1, 10),
        'reg_alpha': trial.suggest_float('reg_alpha', 1e-8, 1.0, log=True),
        'reg_lambda': trial.suggest_float('reg_lambda', 1e-8, 1.0,
log=True),
        'max_delta_step': trial.suggest_int('max_delta_step', 0, 10),
        'objective': 'multi:softmax',
        'num_class': 3,
        'eval_metric': 'mlogloss',
        'random_state': 42
    }
```

   d. Best parameters found by **Optuna**:

```
{'n_estimators': 299, 'max_depth': 14, 'learning_rate':
0.09029244155247403, 'subsample': 0.8621963214294011,
'colsample_bytree': 0.5094283190411915, 'gamma':
0.06544346239673754, 'min_child_weight': 1, 'reg_alpha':
1.589281578937299e-07, 'reg_lambda': 1.7694621194522904e-05,
'max_delta_step': 1}
```

MODELS LOW ACCURACY :

1- Decision Tree :

Potential Reasons:

I. Decision Trees can overfit the training data, especially with many features, leading to poor generalization to new data.

II. If the classes (Good, Standard, Poor) are imbalanced, DTs may bias predictions toward the majority class.

III. DTs may struggle to capture complex interactions between features, limiting their predictive power.

2- Adaboost Classification :

Potential Reasons:

I. AdaBoost relies on weak learners (e.g., decision stumps); if these base models are not effective for the problem, AdaBoost won't improve much.

II. If the features do not strongly correlate with the target variable, AdaBoost may not be able to effectively learn the underlying patterns.

3- Random Forest Classification:

Potential Reasons:

I. Imbalanced classes can bias Random Forest toward the majority class, reducing accuracy for minority classes.

II. Random Forest uses many features for splits. If irrelevant or highly correlated features dominate, performance may degrade.

III. `min_samples_split=10` and `min_samples_leaf=5` could make splits too restrictive, leading to underfitting.

4- Gradient Boost Classification:

Potential Reasons:

I. Gradient Boosting can overfit if the number of iterations (`n_estimators`) is too high or the trees are too deep.

II. Building sequential trees can be computationally expensive and slow on large datasets. Time taken to train model: 196 minutes.

III. Gradient Boosting may exhibit bias toward the majority class in imbalanced datasets, as it focuses on minimizing overall error, which can result in underperformance for minority classes.

BEST MODEL : XGBOOST CLASSIFICATION

Reasons:

I. Optuna effectively explored a wide range of hyperparameters (e.g., `n_estimators`, `max_depth`, `learning_rate`), enabling the model to find the optimal configuration.
II. The `multi:softmax` objective explicitly optimizes for multi-class classification, ensuring suitability for your problem.
III. Bayesian optimization by Optuna converges faster and more effectively on the best hyperparameter set compared to grid or random search.
IV. The inclusion of `reg_alpha` (L1) and `reg_lambda` (L2) terms prevents overfitting by penalizing overly complex models.

Additional Insights:

1- Transforming `Loan_Type` into `Loan_Count` assumes that the number of loans a customer has can reflect their financial behavior. Having multiple loans might suggest that the customer manages complex finances or faces higher financial risks.

2- Converting `Credit_History_Age` into months gave a clearer and more precise way to measure credit history length. This helped the model better understand small differences in how long a customer has had credit.

3- Features like `Credit_Limit`, `Ratio_Credit_Utilization`, and `Delay_from_due_date` are likely strong indicators of creditworthiness because they directly reflect a customer's financial stability and how well they manage payments.

4- Categorical columns like `Profession`, `Credit_Mix`, and `Payment_Behaviour` likely helped the model understand customers' spending patterns and assess their risk levels.

5- Features like `Rate_Of_Interest`, `Per_Month_EMI`, and `Monthly_Balance` were thought to indicate how much financial strain a customer faces and their available cash flow, which can directly influence their credit score and classification.

6- `Profession` and `Credit_Mix` likely influenced creditworthiness because different professions may have varying income levels and spending habits, while the balance between secured and unsecured credit affects financial stability and risk.