Task 1

Credit Scoring Model

Develop a credit scoring model to predict the creditworthiness of individuals based on historical financial data. Utilize classification algorithms and assess the model's accuracy.

Credit scoring: It is a scoring system used by lenders to help decide whether to extend or deny credit. Your credit score has the potential to influence your eligibility for various financial products, including mortgages, auto loans, credit cards, and personal loans.

FICO score, is the most widely used model for credit scoring system in the financial industry, employed by more than 90% of top lenders. It mainly depends on 5 important categories:

- Payment history (35%)
- Amounts owed (30%)
- Length of credit history (15%)
- New credit (10%)
- Credit mix (10%)

1.Preprocessing step

1.1 Check for special characters that mean nothing in my data and should be dropped

- remove the '-' and '_' for the columns like Age ,Annual_Income,Num_of_Loan ,Changed_Credit_Limit,Outstanding_Debt , Monthly_Balance ,Delay_from_due_date ,Num_of_Delayed_Payment ,Amount_invested_monthly and Credit_Mix.

1.2 Check for null values , we find that Monthly_Inhand_salary contain 15,002 null values which is a big number , Type of loan contain 11408 null values , and num_of_delayed_payment contain 7002 , and credit_history_age contain 9030.

Handling null values differently : Solving the problem of missing values in various columns of the `train_data` DataFrame using group-wise imputation based on the 'Customer_ID' grouping. For the columns 'Monthly_Inhand_Salary', 'Num_of_Delayed_Payment', 'Credit_History_Age', 'Credit_Mix', and 'Amount_invested_monthly', the missing values are filled with the mean or mode of the corresponding column within each 'Customer_ID' group. Specifically, the lambda functions within the `transform` method are applied to each group, ensuring that missing values are replaced with the mean for numeric columns or the mode for categorical ones.

Subsequently, missing values in columns 'Monthly_Balance', 'Num_Credit_Inquiries', and 'Monthly_Inhand_Salary' are dropped from the DataFrame, likely due to their relatively low count in the data.

2. Impute outliers with the mean

Replace the outliers in the numeric columns of the **train_data** DataFrame with the mean values of their respective columns and then displaying the DataFrame with the imputed values. This imputation strategy is a common technique to handle outliers and ensure that they do not unduly influence statistical analyses or machine learning models.

3.Handle data types

check for datatypes to be handled correctly throughout each column

- The 'Age' column is converted to integer type, 'Annual_Income', 'Changed_Credit_Limit', 'Outstanding_Debt', 'Monthly_Balance', 'Num_of_Delayed_Payment', and 'Amount_invested_monthly' columns are converted to float type, while 'Num_of_Loan' is converted to integer type. Additionally, 'Credit_History_Age' and 'Credit_Mix' columns are converted to string type.

- This process ensures that each column has the desired data type for subsequent analyses or modeling, addressing any inconsistencies in the original data types.

4. Handle Cateogorical values

Handle the cateogorical data to encode them

- the 'Credit_Mix' column is processed to replace categorical labels 'Bad', 'Standard', and 'Good' with numerical values '0', '1', and '2' respectively. Subsequently, the column is converted to the float data type.

- Similarly, the 'Credit_Score' column undergoes a similar transformation, replacing 'Poor' with '0', 'Standard' with '1', and 'Good' with '2', and then converting the column to the integer data type.

- The 'Payment_of_Min_Amount' column is modified by replacing 'Yes' with '1', 'NM' with '0', and 'No' with '0', followed by a conversion to the integer data type.

- Lastly, the 'Payment_Behaviour' column is processed by mapping categorical values to corresponding numerical representations and subsequently converting the column to the integer data type.

- The 'Type_of_Loan' column in the 'train_data' DataFrame is first converted to a string data type. Subsequently, the LabelEncoder from scikit-learn is employed to encode the categorical labels in the 'Type_of_Loan' column with numerical values.

- extract_year is defined to extract the number of years from a string representing a duration.

The extracted numeric value is then converted to an integer and returned. This function is then applied to the 'Credit_History_Age' column in the 'train_data' DataFrame, effectively extracting the number of years from each entry in that column and transforming the data accordingly.

- Fillna method is used to fill missing values (NaN) in the 'train_data' DataFrame with the value 0.

These transformations are typically performed to facilitate numerical computations and analyses on categorical data in machine learning models.

5. Standard Scaling to features

By applying this standardization process, the features in both the training and test datasets are transformed to have zero mean and unit variance, which is a common preprocessing step in machine learning workflows.

6. Building the model

6.1 Random Forest

Random Forest often performs well in credit scoring problems for several reasons:

1. **Handling Non-Linearity:**
    - Credit scoring problems are often non-linear, meaning the relationship between features (financial variables) and creditworthiness may not be a simple straight line. Random Forest can capture complex non-linear relationships between features and the target variable.
2. **Ensemble Learning:**
    - Random Forest is an ensemble learning method that builds multiple decision trees and combines their predictions. This helps to reduce overfitting and improve generalization to new, unseen data.
3. **Feature Importance:**
    - Random Forest provides a measure of feature importance. In credit scoring, understanding which financial variables contribute most to the prediction of creditworthiness is crucial for interpretability. Random Forest can highlight important features.

4. **Robustness to Outliers:**
    - Credit datasets may contain outliers or noisy data. Random Forest is robust to outliers and noise, as it builds multiple trees and averages their predictions.

5. **Handling Imbalanced Data:**
    - Credit scoring datasets are often imbalanced, meaning there are fewer instances of bad credit compared to good credit. Random Forest handles imbalanced data well and can prevent biases towards the majority class.
6. **Tuning Flexibility:**
    - Random Forest has hyperparameters that can be tuned for optimal performance. Parameters like the number of trees, tree depth, and the number of features considered at each split provide flexibility to optimize the model.
7. **Reducing Overfitting:**
    - Random Forest introduces randomness during the training process, both in terms of the features considered at each split and the data used to build each tree. This randomness helps prevent overfitting and improves the model's ability to generalize.

The model gives a good accuracy which is 0.78.

6.2 Gradient Boosting

is a powerful and flexible algorithm that performs well in a variety of settings, including credit scoring, due to its ability to handle complex relationships, avoid overfitting, and provide accurate predictions.

The accuracy is 0.69.