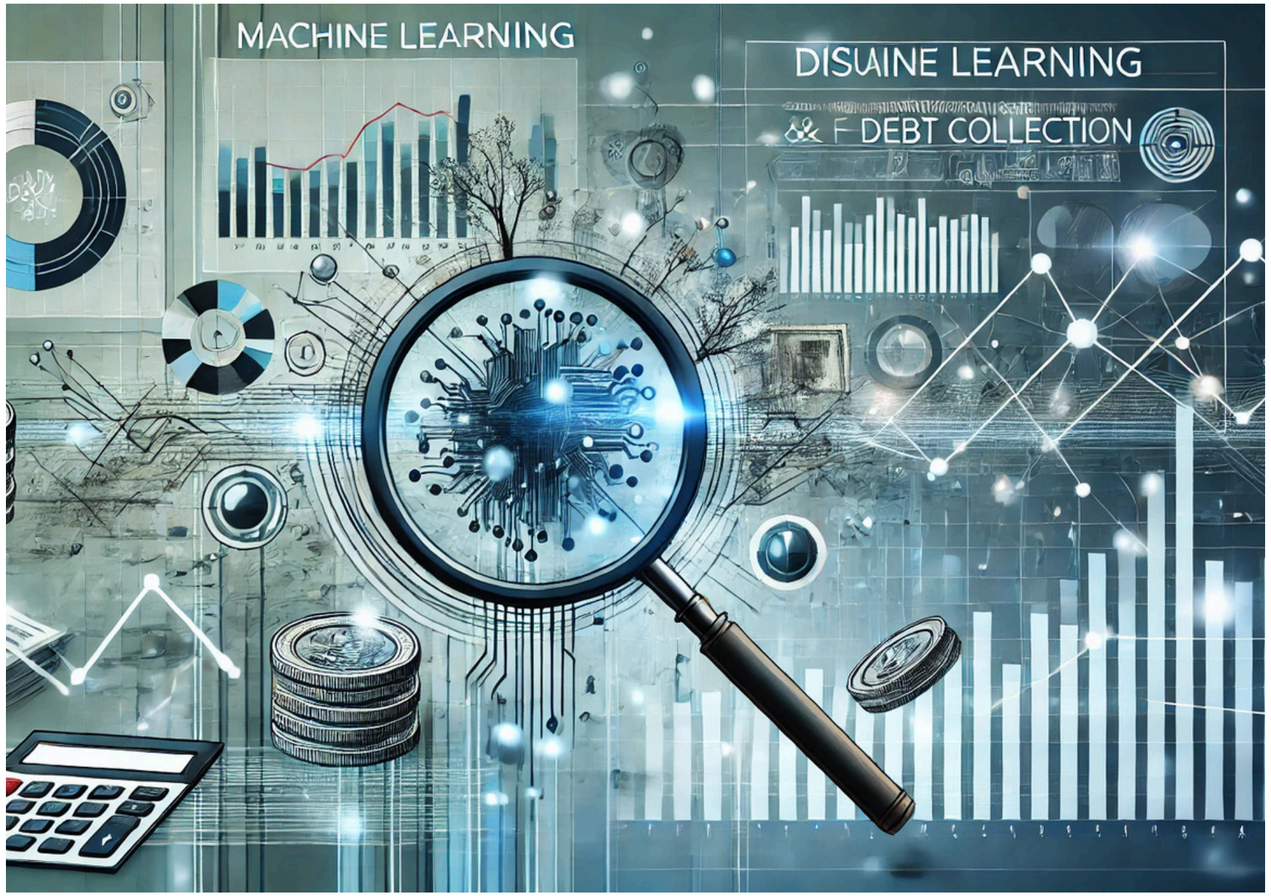
Machine Learning *Project*

Forecasting Debt Collection Trends



About the Title

The project aims to develop a machine learning model to predict the probability of successfully collecting debts by analyzing statute-barred status. Using features like balances, purchase price, debt type, and more, the goal is to identify accounts where statute-barred status impacts recovery likelihood.

Problem Statement

KEY PROBLEM

The key problem is identifying which accounts in debt collection are statute-barred and, as a result, may not be recoverable. Specifically, the goal is to predict the likelihood of successfully collecting a debt based on the statute-barred status of each account. The challenge lies in analyzing a range of features—such as account balances, creditor information, purchase dates, etc.—and determining whether an account's statute-barred status significantly impacts its collectability.

WHY IS SOLVING THIS PROBLEM IMPORTANT?

Solving this problem is crucial because identifying statute-barred accounts early can help debt collectors avoid wasting resources on accounts that are unlikely to generate recovery. This leads to improved efficiency in debt collection processes, ensuring that efforts are focused on accounts with a higher probability of successful collection. Additionally, accurately predicting which debts are recoverable can help organizations minimize losses, optimize collections strategies, and enhance overall financial health.

PRACTICAL RELEVANCE

This problem is highly relevant in the financial services industry, particularly within debt collection agencies, banks, and credit institutions. It also has a significant impact on industries dealing with high volumes of outstanding debt, such as telecommunications, utilities, and consumer finance. By utilizing machine learning, these organisations can automate and streamline their decision-making process, making more informed choices about where to invest resources and which debts to prioritise. Furthermore, it plays a crucial role in regulatory compliance, as certain debts might have legal limitations on collection, affecting financial institutions' legal obligations.

About the Dataset

20% of data is selected for faster processing as dataset is large

DATA STRUCTURE

Rows- 81285(20%)
Columns- 22

BIFURCATION ON THE FEATURES ON BASIS OF DATA TYPE

Numerical Columns- 8
Categorical Columns- 13
Target Variable- 1

```
Data columns (total 22 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   EntityID                    81285 non-null  int64
 1   OriginalCreditor[Redacted]  81285 non-null  object
 2   AccountID                   81285 non-null  int64
 3   CurrentBalance              81285 non-null  object
 4   DebtLoadPrincipal           81285 non-null  object
 5   Balanaceatdebt_load         81285 non-null  object
 6   PurchasePrice               80762 non-null  float64
 7   ProductOrDebtType           81285 non-null  object
 8   CollectionStatus            81285 non-null  object
 9   ClosureReason               1754 non-null   object
 10  InBankruptcy                81285 non-null  object
 11  AccountInsolvencyType       62 non-null     object
 12  CustomerInsolvencyType      1662 non-null   object
 13  IsLegal                     81285 non-null  object
 14  LastPaymentAmount           20852 non-null  object
 15  LastPaymentMethod           20852 non-null  object
 16  NumLiableParties            81267 non-null  float64
 17  CustomerAge                 75504 non-null  float64
 18  NumPhones                   81285 non-null  int64
 19  NumEmails                   81285 non-null  int64
 20  NumAddresses                81285 non-null  int64
 21  IsStatBarred                81285 non-null  object
```

About the Target Variable

TARGET VARIABLE

IsStatBarred

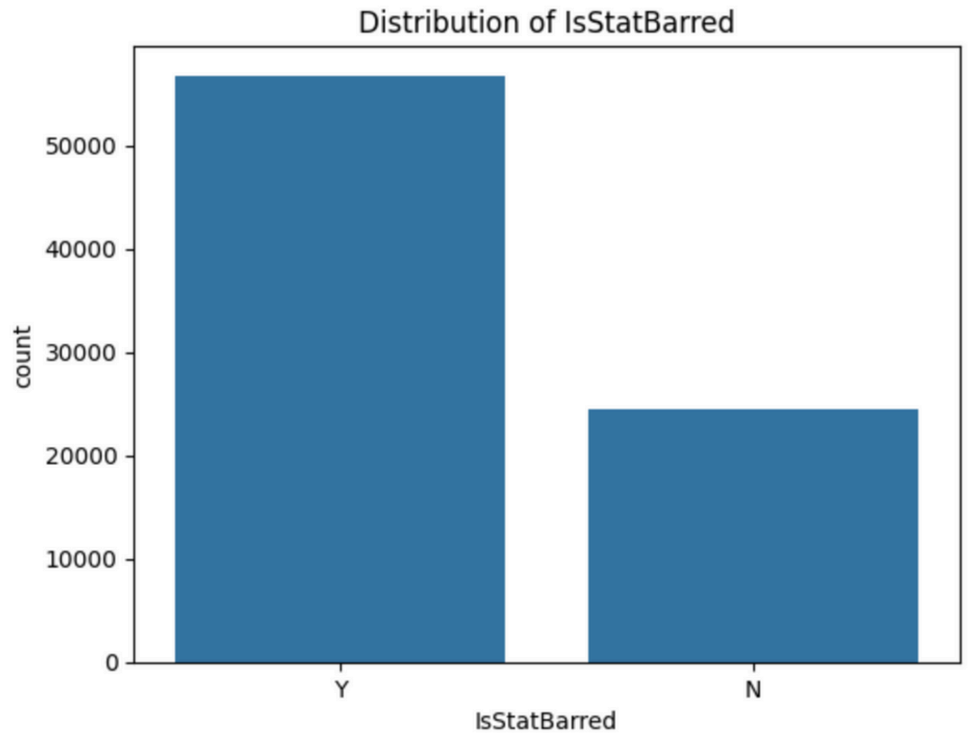
TARGET VARIABLE CLASSES

Y - 56777 (70%)
N - 24508 (30%)

DESCRIPTION

Target variable has 2 class
- Y mean unbarred
- N means unbarred

We can clearly see there is a class imbalance as 70% of the values belongs to class ‘Y’ and 30% to class ‘N’



EDA, Feature Engineering And Data Preprocessing Approaches

Categorical Feature Approach

CATEGORICAL FEATURES

- CurrentBalance
- DebtLoadPrincipal
- Balanaceatdebt_load
- ProductOrDebtType
- CollectionStatus
- ClosureReason
- InBankruptcy
- AccountInsolvencyType
- CustomerInsolvencyType
- IsLegal
- LastPaymentAmount
- LastPaymentMethod

SELECTED FEATURES

- CurrentBalance
- DebtLoadPrincipal
- Balanaceatdebt_load
- ProductOrDebtType
- CollectionStatus
- InBankruptcy
- IsLegal
- LastPaymentAmount
- LastPaymentMethod

REASON FOR SELECTION

- Every column should diverse distribution with both the classes of target varaible

APPROACH USED FOR FEATURE SELECTION

- Check Missing Value if more than 90% of the values are missing, drop the feature else use imputation techniques to handle missing values.
- Imputation techniques used are:
  - Random Sample Imputation
  - KNN Imputation
- Observe the count plot with target variable and based upon the diverse disribution select the feature.

DROPPED FEATURES

- ClosureReason
- AccountInsolvencyType
- CustomerInsolvencyType

REASON FOR DROP

- Every feature has more than 95% missing value

TOTAL FEATURES SELECTED

- So out of 12 features 9 are selected for model building

Numerical Feature Approach

NUMERICAL FEATURES

- PurchasePrice
- NumLiableParties
- CustomerAge
- NumPhones
- NumEmails
- NumAddresses

SELECTED FEATURES

- PurchasePrice
- CustomerAge
- NumPhones
- NumEmails

REASON FOR SELECTION

- Clear distinction and highlighed different distribution between 2 categories of target variable in box plot and good histogram
- Moderate correlation with target.

DROPPED FEATURES

- NumLiableParties
- NumAddresses

REASON FOR DROP

- Similar distinction and distribution in box plot and histogram between both categories of target variable.
- Week Correlation with target.

APPROACH USED FOR FEATURE SELECTION

- Check Missing Value if more than 90% of the values are missing, drop the feature else use imputation techniques to handle missing values.
- Check the individual distribution with KDE plot to select the imputation techniques to fill missing values.
- Imputation techniques used is Median Imputation as most of the individual distributions are right skewed
- Additionally for feature **CustomerAge** for negative age, absolute values are used and then ages which are less than 20 are replace with median and missing values are also filled with median
- Relationship with target variable has been observe using 3 things:
  - BOX PLOT
  - HISTOGRAM
  - CORRELATION
- Based upon the combine observation of all 3 properties feature selection is done.

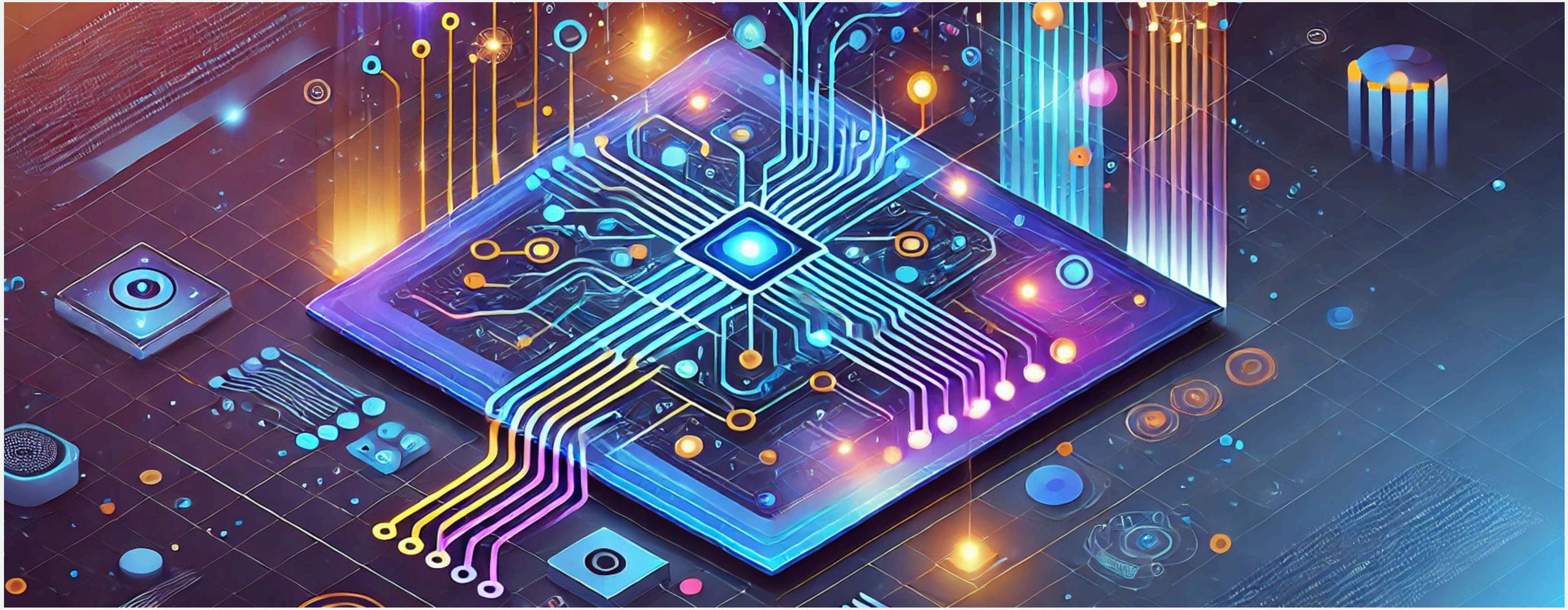
TOTAL FEATURES SELECTED

- So out of 6 features, 4 are selected for model building

Selected Features for Model Building

FOLLOWING ARE THE 14 FEATURES SELECTED FOR MODEL BUILDING INCLUDING 1 TARGET VARIABLE

- CurrentBalance
- DebtLoadPrincipal
- Balanaceatdebt_load
- ProductOrDebtType
- CollectionStatus
- InBankruptcy
- IsLegal
- LastPaymentAmount
- LastPaymentMethod
- PurchasePrice
- CustomerAge
- NumPhones
- NumEmails
- IsStatBarred **(Target)**



Model Building

Model Building Approach

ENCODING

- Label Encode to maintain the same feature count and avoid adding more feature that makes the model more complex by increasing its dimension

FEATURE SCALING

- Standard Scaler is used for feature scaling

TRAIN AND TEST SPLIT RATIO

- Train- 80%
- Test- 20%

MODEL USED

- Logistic Regression
- SVM
- Naive Bayes
- Random Forest
- Adaboost
- Gradient Boosting

APPROACH

- Firstly train the models individually on the given dataset (imbalanced) and evaluate their performance.
- For Evaluation **ACCURACY** and **ROC_AUC_SCORE** metrics are used because on an imbalanced data, accuracy alone is not reliable measure as it may get biased towards the majority class, where as ROC_AUC score is generally a better evaluation metric than accuracy because it evaluates the model's ability to distinguish between the classes without being biased by class imbalances.
- Once the evaluation is done then use different techniques to improve the model performance.
- Techniques which are used to improve performance are:
  - Ensemble technique(Stacking)
  - Hyper Parameter Tuning using GridSearchCV.
  - SMOTE to handle imbalance date and applying it to each model and evaluate their performance before SMOTE and after SMOTE.
- Give the final conclusion after applying all the technique and observing the results.

Model Performance on Imbalance Data

LOGISTIC REGRESSION

Accuracy- ***86.38***
ROC_AUC_Score- **90.95**

SVM

Accuracy- **93.08**
ROC_AUC_Score- **97.28**

NAIVE BAYES

Accuracy- **81.60**
ROC_AUC_Score- **88.79**

ADABOOST

Accuracy- **94.55**
ROC_AUC_Score- **98.87**

RANDOM FOREST

Accuracy- **97.20**
ROC_AUC_Score- **99.54**

GRADIENT BOOSTING

Accuracy- **96.54**
ROC_AUC_Score- **99.40**

Observations on Model Performance

- Random Forest and Gradient Boosting show the **highest accuracy and ROC AUC**, indicating strong predictive performance on the imbalanced dataset. They seem to be the best-performing individual models out of those tested.
- Logistic Regression has a **lower accuracy** compared to ensemble methods (Random Forest, Gradient Boosting, AdaBoost). Its performance is decent, but it lags behind the ensemble models.
- SVM performs reasonably well, with a ROC AUC score that is not significantly lower than the top-performing models. Its accuracy might be lower, but it still shows good discrimination between the two classes. Its absence of a predict_proba method made it harder to evaluate the ROC_AUC.
- Adaboost shows decent performance, showing promise but not outperforming Random Forest and Gradient Boosting.
- Naive Bayes underperforms compared to other models. Its simplicity and assumptions may not align well with the characteristics of this dataset, leading to lower accuracy and ROC_AUC. The large discrepancy between Naive Bayes's performance and the others suggests that more complex models capture the patterns in the data better.

Techniques to improved Model Performance

Ensemble Method- *StackingClassifier*

BASE ESTIMATORS USED

- Random Forest
- Gradient Boosting
- SVM

META CLASSIFIER USED

- Logistic Regression

PERFORMANCE

- *Accuracy*- **97.14**
- *ROC_AUC_SCORE*- **99.57**

OBSERVATION

- Comparing to other models Stacking Classifier performed really well on both parameters Accuracy and ROC_AUC_Score because of the simple fact that we have used best models like Random Forest, Gradient Boosting and SVM which performed well individually as base estimators for Stacking Classifier.

HyperParameter Tuning- *Random Forest*

MODEL SELECTED

- Random Forest

REASON

- Out of all the models, Random Forest performance is best on accuracy and ROC_AUC_Score

PERFORMANCE BEFORE TUNING

- *Accuracy*- **97.20**
- *ROC_AUC_SCORE* - **99.54**

PERFORMANCE AFTER TUNING

- *Accuracy*- **98.20**
- *ROC_AUC_SCORE*- **96.99**

OBSERVATION

- Accuracy has been improved from 97.20 to 98.20 howevr the ROC_AUC_Score is slightly reduced to 96.99 from 99.54 it could be due to:
  - Class Imbalance Effects
  - Change in Decision Threshold
  - Overfitting to the Training Data
  - Improper Weighting of Classes.

Handling Class Imbalance by applying *SMOTE*

SYNTHETIC MINORITY OVERSAMPLING TECHNIQUE (SMOTE)

- Synthetic Minority Over-Sampling Technique (SMOTE) is a powerful method used to handle class imbalance in datasets. SMOTE handles this issue by generating samples of minority classes to make the class distribution balanced.

CLASS DISTRIBUTION BEFORE APPLYING SMOTE

- Y OR 1 - 56777(70%)
- N OR 0 - 24508(30%)

CLASS DISTRIBUTION AFTER APPLYING SMOTE

- Y OR 1 - 45455(50%)
- N OR 0 - 45455(50%)

MODEL PERFORMANCE POST APPLYING SMOTE

LOGISTIC REGRESSION

ACC : 86.03
ROC_AUC: 91.16

SVM

ACC : 93.51
ROC_AUC: 97.34

NAIVE BAYES

ACC : 82.89
ROC_AUC: 88.82

ADABOOST

ACC : 95.58
ROC_AUC: 98.97

RANDOM FOREST

ACC : 97.09
ROC_AUC: 99.52

GRADIENT BOOSTING

ACC : 96.29
ROC_AUC: 99.36

Comparison of Model Performance before and after *SMOTE*

Comparison of Individual Models Before and After SMOTE on the basis of Highest ROC_AUC Score Post SMOTE

	Acc_Imb	Acc_SMOTE	ROC_AUC_Imb	ROC_AUC_SMOTE
Random Forest	0.972074	0.970966	0.995446	0.995288
Gradient Boosting	0.965430	0.962970	0.994010	0.993632
Adaboost	0.945562	0.955834	0.988752	0.989773
SVM	0.930861	0.935105	0.972891	0.973480
Logistic Regression	0.863874	0.860306	0.909558	0.911654
Naive Bayes	0.816018	0.828935	0.887943	0.888242

Obsevations based upon the comparison

- *Logistic Regression, SVM, Naive Bayes*, and *Adaboost* performed better with SMOTE in terms of ROC AUC, which is an essential metric for imbalanced data because it captures the trade-off between true positive rate (TPR) and false positive rate (FPR).
- *Random Forest* and *Gradient Boosting* did not benefit from SMOTE, likely because these models already have mechanisms to handle imbalanced data effectively (e.g., weighted splits and ensemble techniques).
- Overall Performance: While SMOTE improved the ROC AUC for some models, it did not significantly benefit Random Forest, and in some cases, it even slightly decrease the performance. The Random Forest model after hyperparameter tuning already demonstrated good performance on the imbalanced dataset. Stacking provides a marginal increase in performance.
- Robustness: Random Forest generally handles class imbalance better than many other models. Its internal mechanisms help to address the issue without the need for oversampling.
- Interpretability: Random Forest models offer good interpretability compared to more complex ensemble techniques, such as Stacking. This can be beneficial for explaining model predictions to stakeholders.
- Computational Cost: Avoid SMOTE unless strictly necessary. It increases the size of the dataset, which can lead to longer training times without guaranteed improvement in performance.
- Tuning: Hyperparameter tuning has already been applied to the Random Forest, showing a potential for improvement over the base model.

*In summary, the **Random Forest model with hyperparameter tuning** provides a good balance between performance, robustness, and computational efficiency for this imbalanced dataset.*


THANK
YOU !