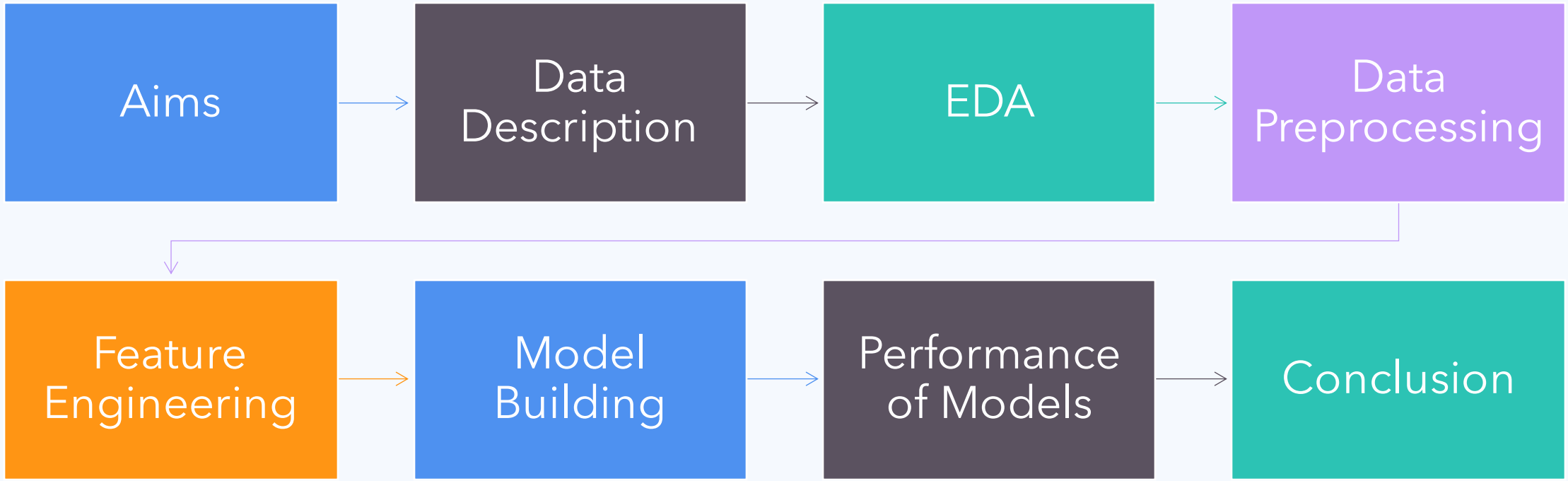
OGTIP Final PROJECT FINANCIAL FRAUD DETECTION

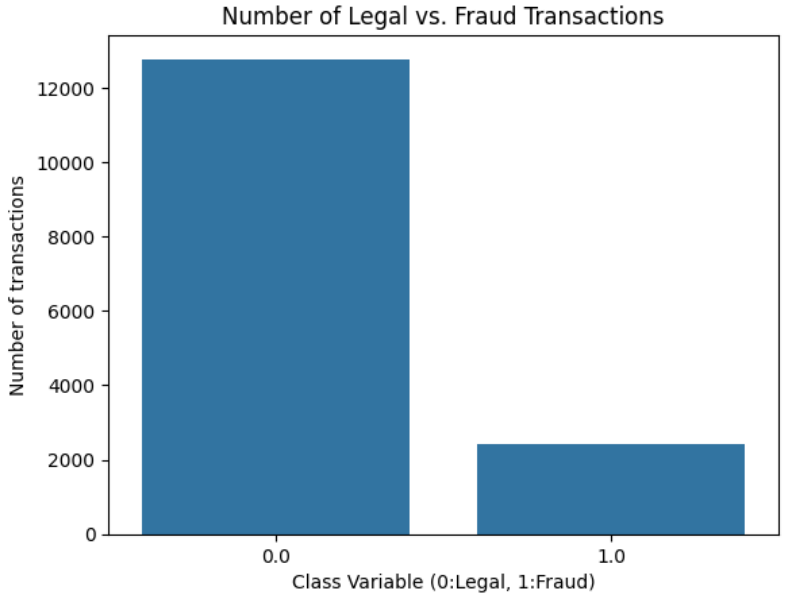
MICHAEL RANTISI

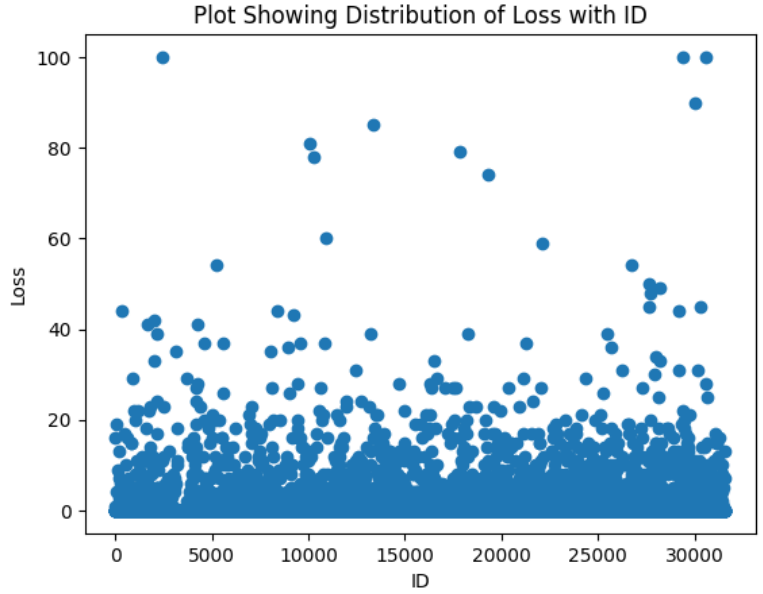
TABLE OF CONTENT



Aims

- We have received a dataset from XYZ Bank that includes the loan records of their clients. The task is to conduct a comprehensive data analysis on this dataset and develop a machine learning model. This model will predict potential defaulters among future loan applicants, allowing the bank to devise a prudent approach for managing credit risk.



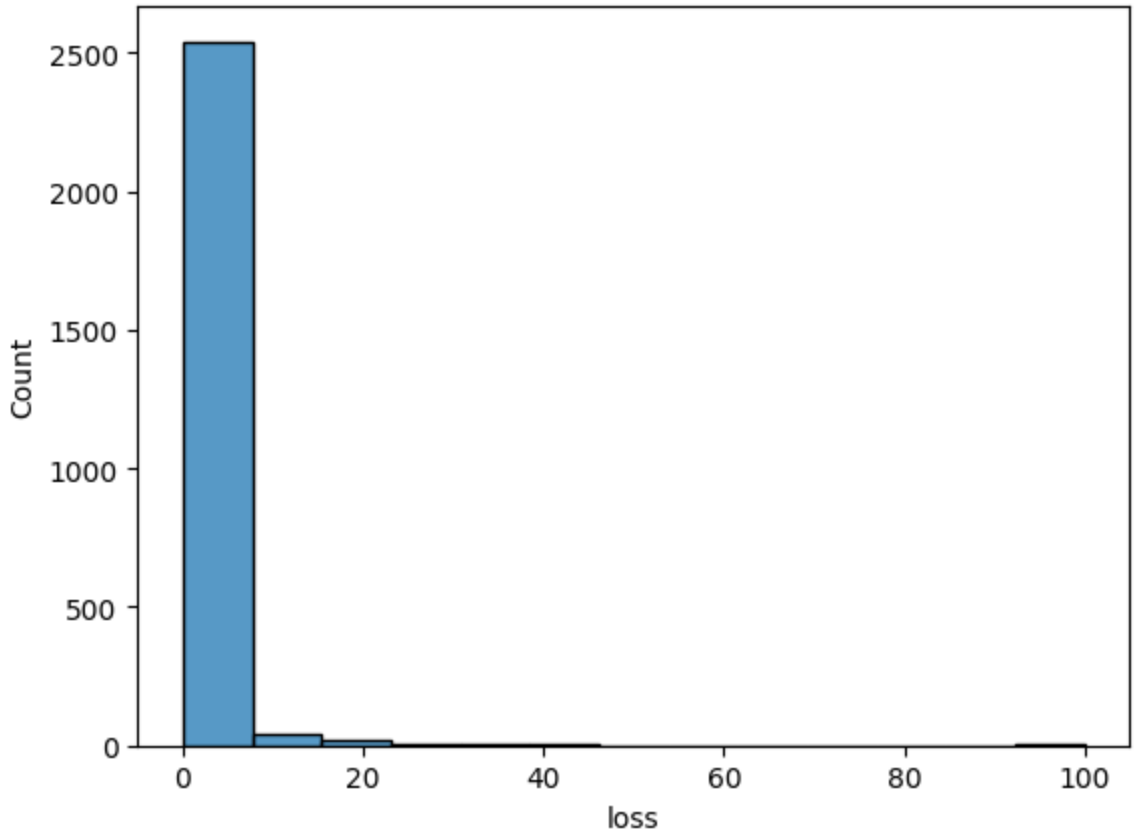


Data Description

- The dataset consists of 771 columns for the train, containing various features related to loan records, the specifics of which are unknown. There is a particular variable labeled as 'loss', which indicates whether a client has defaulted on a loan — a value of 0 means no default, while a value of 1 or higher signifies default, with higher values potentially reflecting the severity of the default leading to establishing the number of legal vs fraud transactions that can be plotted

EDA

The distribution of the 'loss' variable in the dataset is positively skewed, indicating that a greater number of clients are non-defaulters. This is also evident from the skewness metric calculated from the data, which has a value of approximately 199.6 .

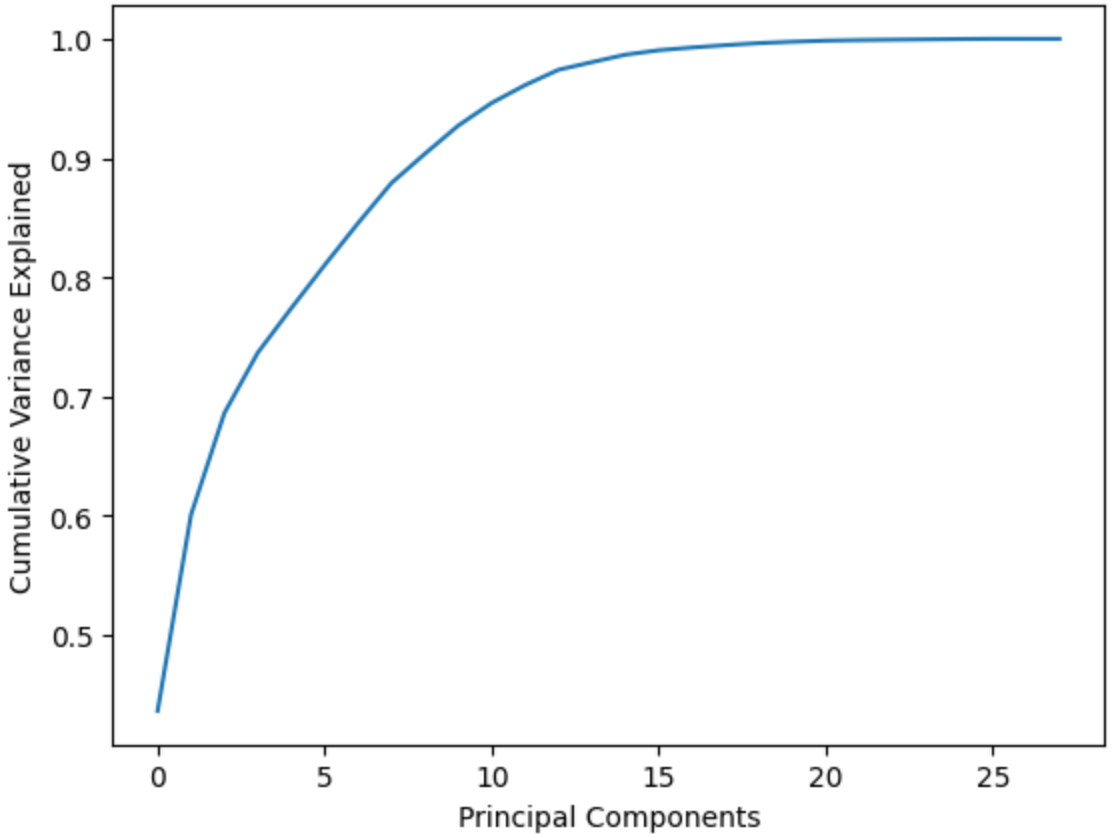


Data Preprocessing

- The dataset contains some missing values; however, these represent a small fraction of the total data. To address this, we will impute the missing values and eliminate any rows still containing gaps. Furthermore, we will modify the 'loss' column by assigning a value of 1 to any entries that are greater than or equal to 1, transforming it into a binary indicator. This change will simplify the dataset into a binary classification format, enabling us to distinguish between defaulters and non-defaulters for prediction purposes.

Feature Engineering

- I will apply Principal Component Analysis (PCA) to reduce the dimensionality of our data, maintaining the most significant statistical features. PCA simplifies the complexity of high-dimensional data while preserving 95% of the variance, which we determine as our threshold for the number of principal components to retain. This process allows us to focus on the most informative aspects of the data for our analysis.



Model Building

For our binary classification problem, we will employ the following predictive models:

1. Logistic Regression
2. Linear Regression
3. Gaussian Naive Bayes
4. Random Forest Classifier
5. XGBoost.

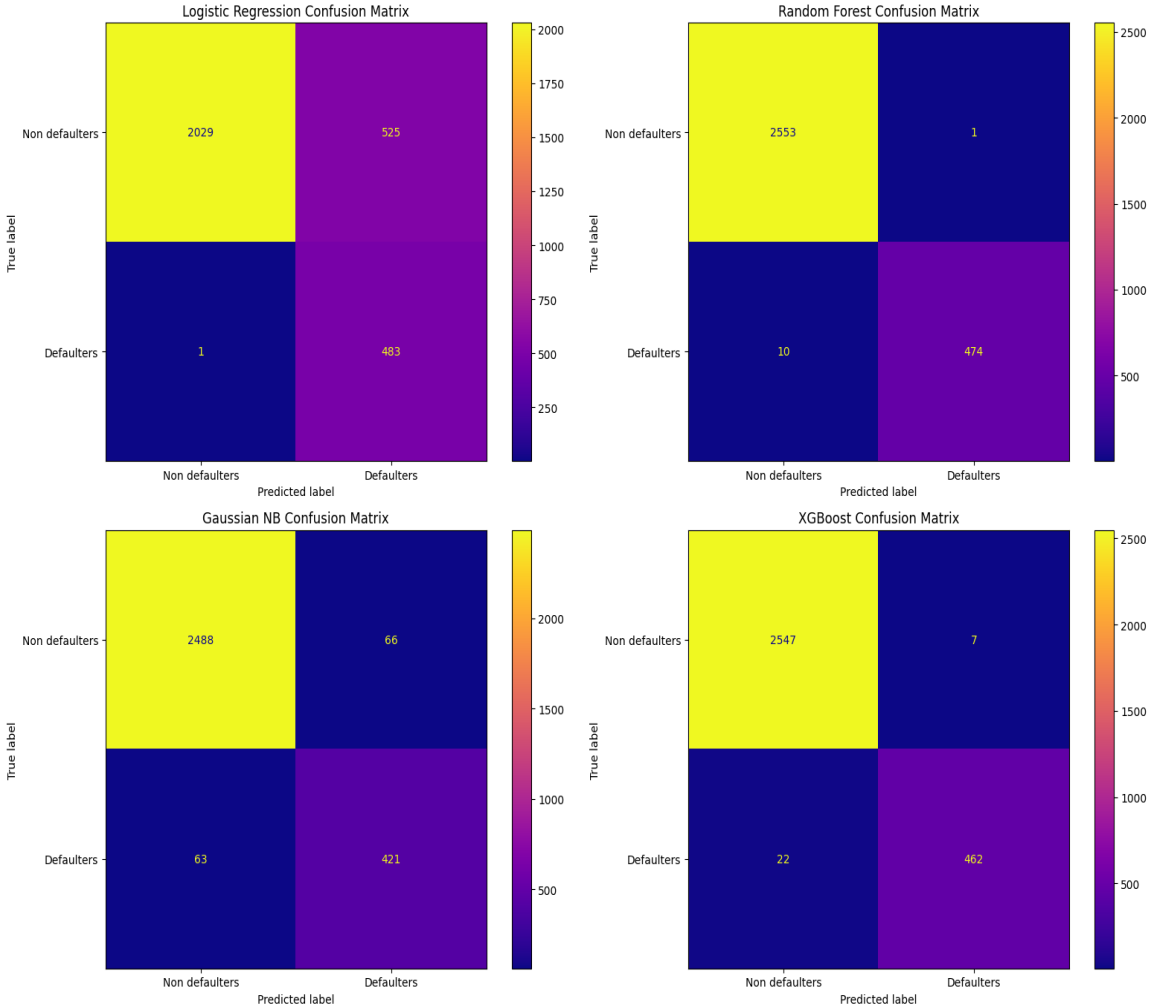
Performance Metrics Of The Models 1/3

- The R2 scores suggest that the Linear Regression and Random Forest models perform very well on the given dataset, with scores close to 1. The Logistic Regression model shows moderate predictive ability, which is expected given that it's typically used for classification. Gaussian Naive Bayes appears to be less effective in this scenario. XGBoost's perfect score might indicate overfitting.

Machine Learning models	R2 Score
Linear Regression	0.96
Logistical Regression	0.79
Random Forest Classifier	0.97
Gaussian Naive Bayes	0.61
XGBoost	1.0

Performance Metrics Of The Models 2/3

- Logistic Regression and Gaussian Naive Bayes have some false positives and false negatives, indicating room for improvement. The Random Forest model performs very well with very few misclassifications. XGBoost has a high true positive rate and very few errors, suggesting it might be the best performing model among the four, but the perfection here should be cross-checked for overfitting.



Performance Metrics Of The Models 3/3 after Hyper tunning

Machine Learning models	Accuracy before	Accuracy after
Linear Regression	Train: 97% Test:97%	Train: 97% Test:97%
Logistical Regression	Train: 97.53% Test:97.43%	94.4%
Random Forest Classifier	Train: 100% Test:100%	99.4%
Gaussian Naive Bayes	Train: 94.39% Test:94.34%	95%
XGBoost	Train: 100% Test:99.7%	96.9%

Conclusion

The process improved accuracy for Logistic Regression and Gaussian Naive Bayes, indicating a better generalization on the unseen test data, it slightly reduced the accuracy for Random Forest and XGBoost. This could suggest that the initial models may have been overfitting to the training data, and tuning helped to mitigate this