Credit Card Fraudulent Transaction Detection

Rohith Reddy Bairi (A20526972)
Pavitrasai Vegiraju (A20525304)
Brijeshkumaryadav Bandi (A20528371)

The Problem

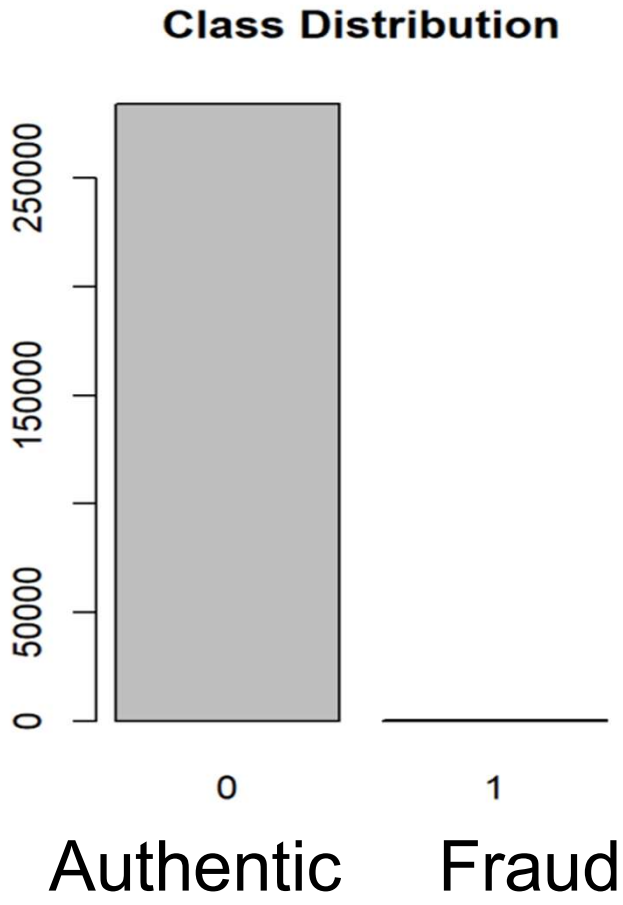
- Global credit card fraud losses reached $34.36 billion in 2022, up from $21.84 billion in 2015.
- U.S. card fraud losses are forecast to reach $165.1 billion over the next 10 years.
- Fraudulent transactions account for a small fraction of all transactions, typically less than 1%.
- Our sample dataset shows:
  - 284,315 Authentic Transactions
  - 492 Fraudulent Transactions
  - 0.172% of Total Transactions were Fraudulent

How Frauds are Recognized

- Predefined rules to identify suspicious transactions
- Advanced machine learning models trained on historical data
- Analyzing transaction patterns to detect anomalies
- Personalized models based on cardholder's spending habits
- Real-time monitoring to quickly flag potential fraud

Challenges

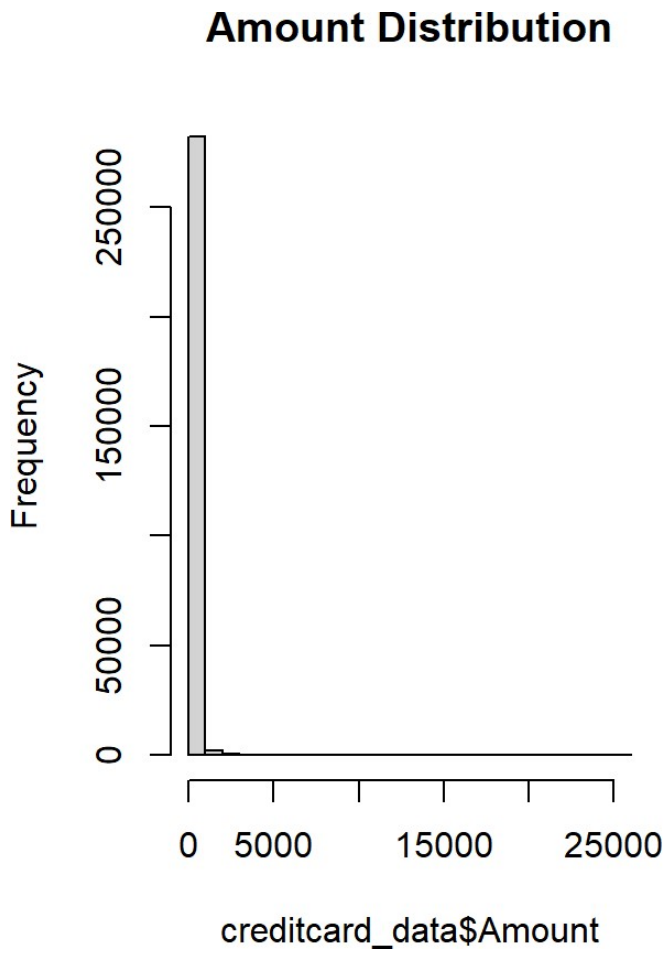
Since the data is highly skewed

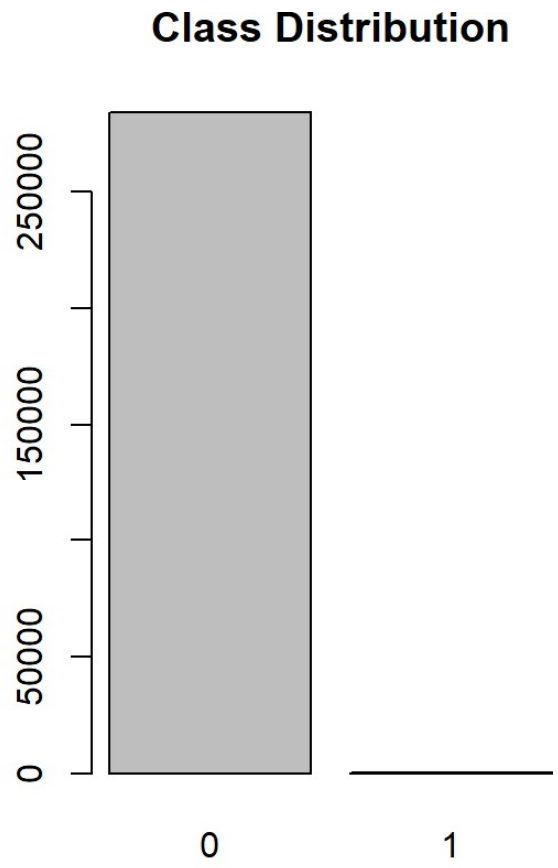


Data Exploration

- Dataset : creditcard.csv
- Features: Time, V1, V2,…., V28, Amount, Class.
- Class Imbalance:
  - Majority of the transactions are non-fraudulent
  - SMOTE oversampling required to balance the classes.

Data Exploration





Resampling Methods

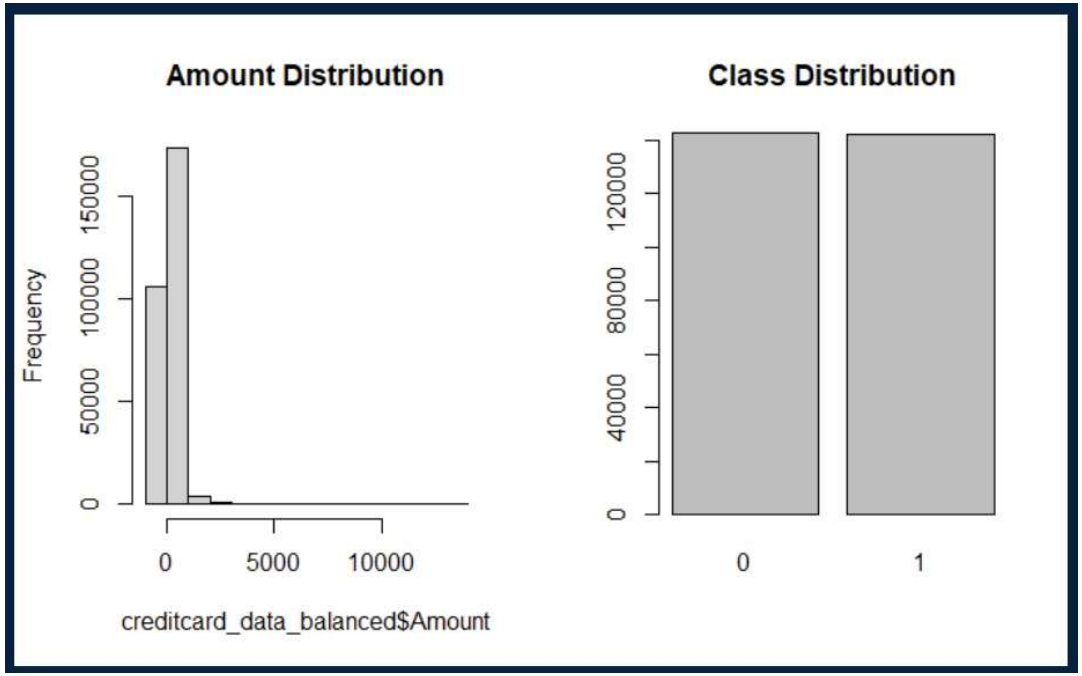
- The resampling methods are used to adjust the class distribution of the data as the minority class is not equally represented.
- There are two methods to perform the resampling.
  - Oversampling
  - Undersampling

SMOTE Oversampling

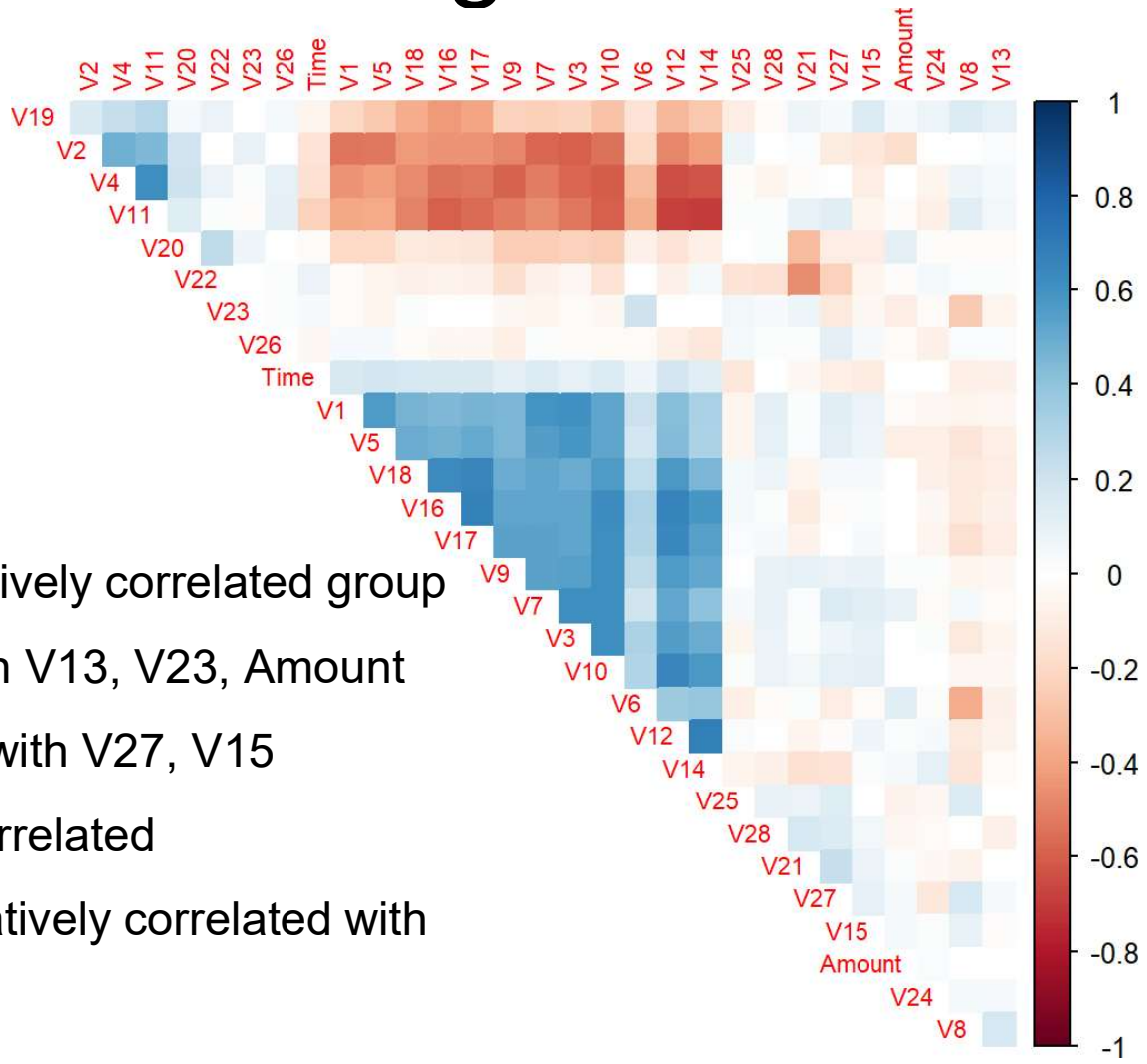
SMOTE (Synthetic Minority Over-Sampling Technique)

- Generates synthetic examples for minority class
- Balances the dataset by oversampling the minority class

Balanced dataset: creditcard_data_balanced



Correlation Factor Among Features



Observations:

- V4, V11, V20, V22 highly positively correlated group
- V19 moderately correlated with V13, V23, Amount
- Amount negatively correlated with V27, V15
- V8, V24 strongly negatively correlated
- V12, V14, V25, V28, V21 negatively correlated with positive group

Dividing the dataset into Test & Train

To validate our model's prediction

- Divide dataset into
  - 80% for Training
  - 20% for Testing

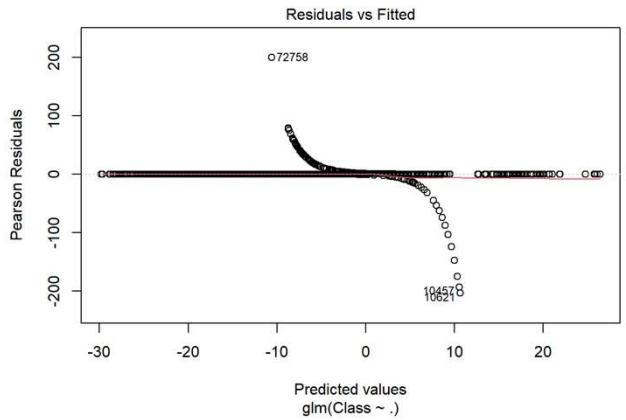
Logistic Regression

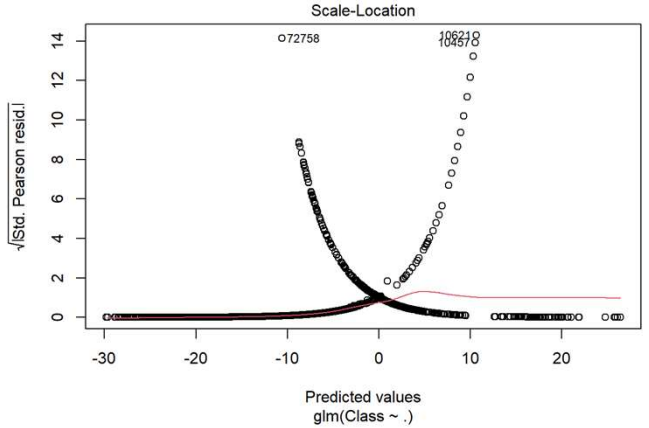
- Logistic regression is a binary classifier based on the regression model, where the dependable variable is categorical.

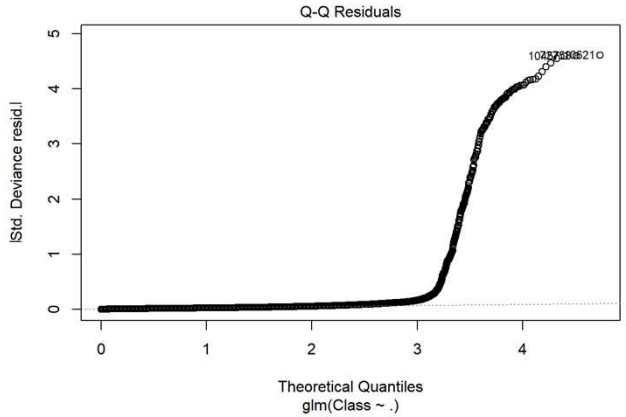
- In logistic regression, we find

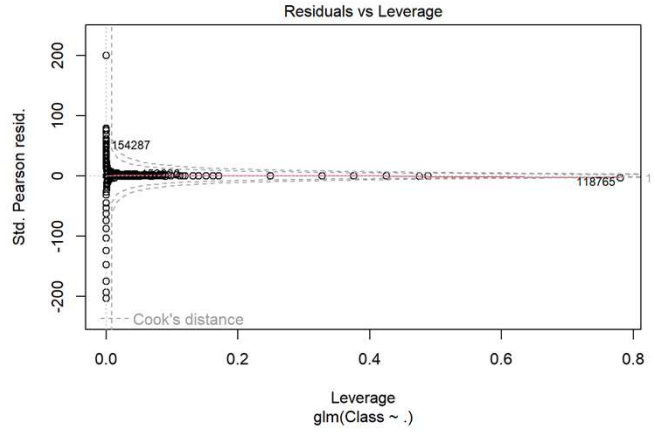
  $\text{logit}(P) = a + b\,X,$

Model Implementation Before SMOTE - Logistic Regression





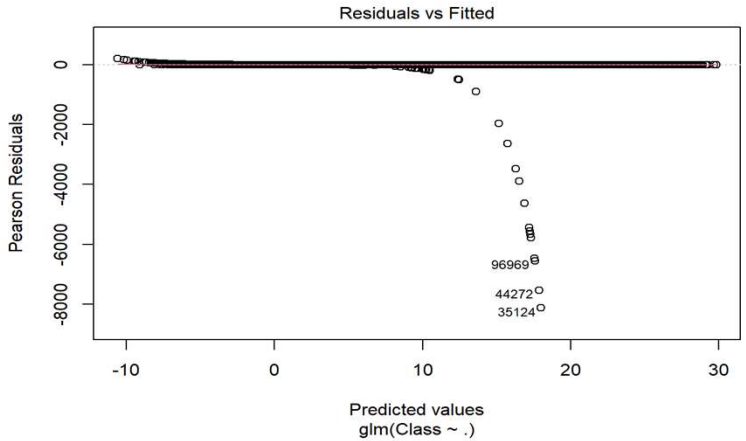


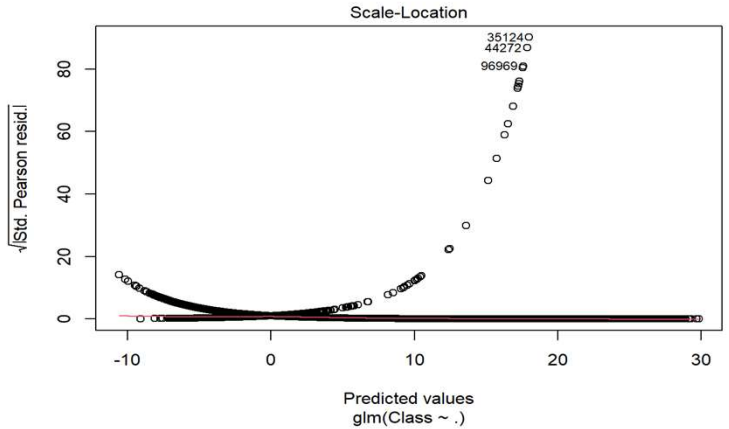


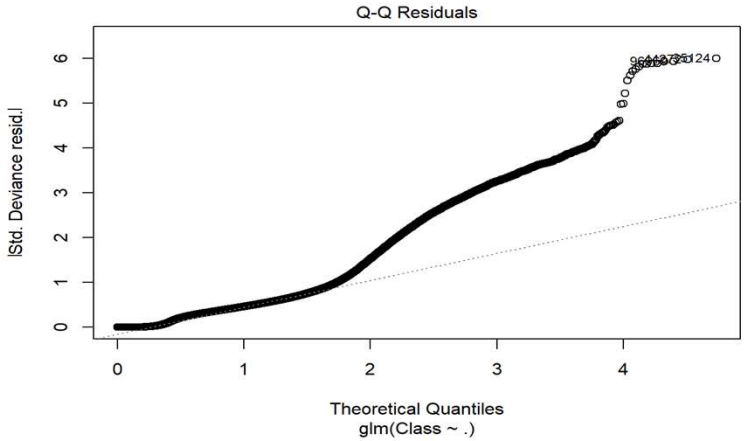
LOGISTIC REGRESSION WITH SMOTE

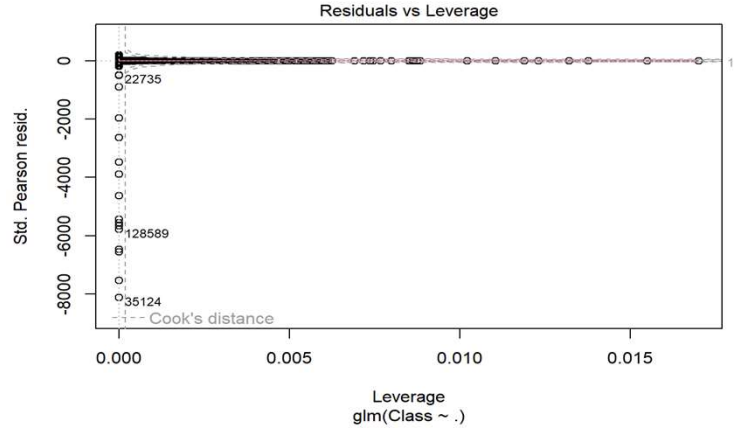
- SMOTE stands for Synthetic Minority Oversampling Technique to an input dataset
- This is a statistical technique to increase the number of samples in minority class in the dataset to make it balanced.
- It works by generating new instances of data from the existing by taking feature space of each target class and its nearest neighbors

LOGISTIC REGRESSION AFTER SMOTE

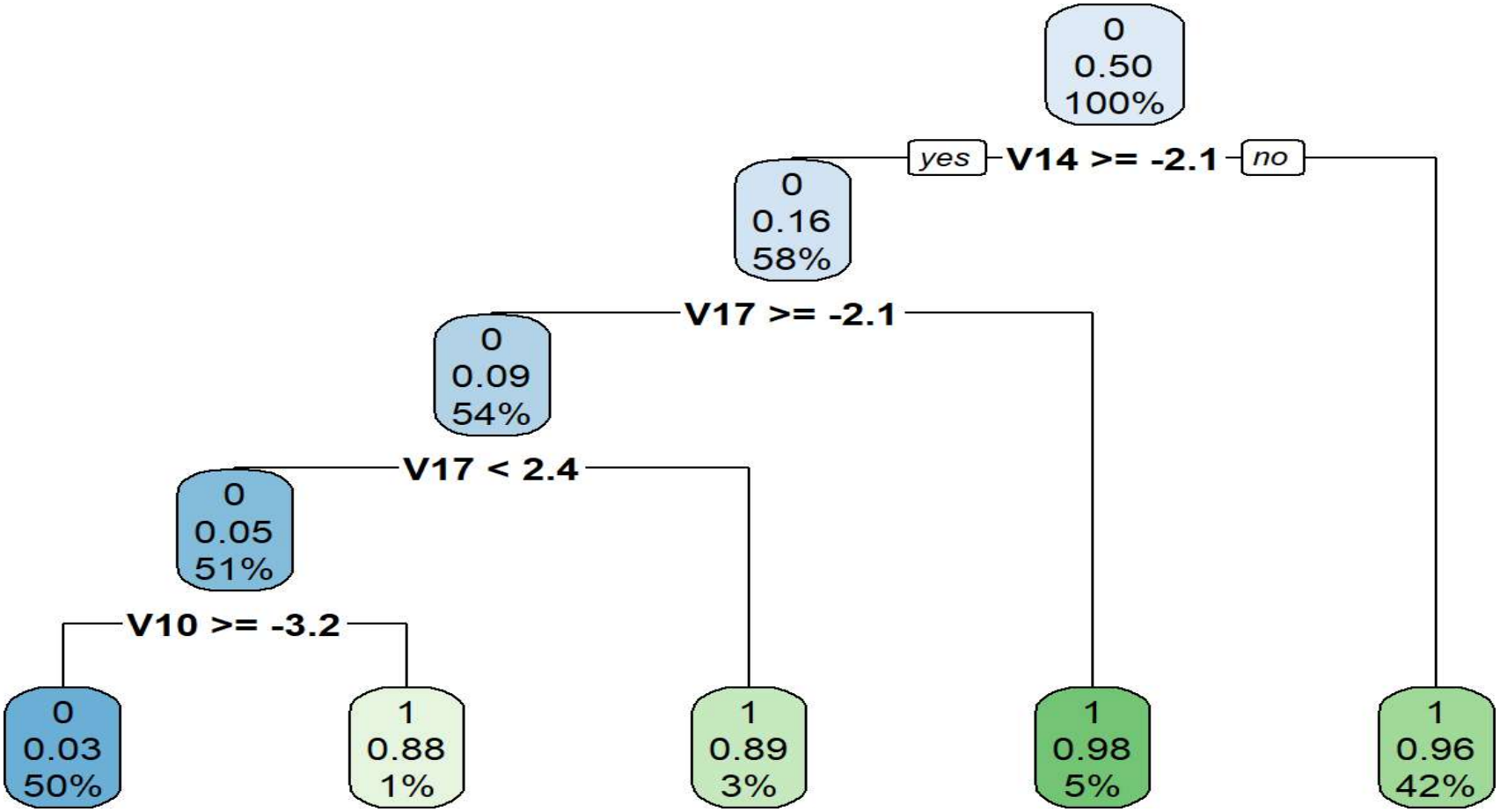




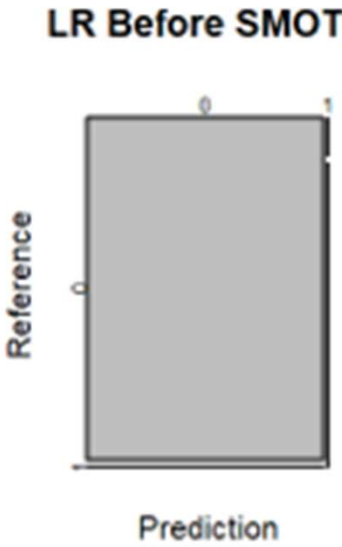


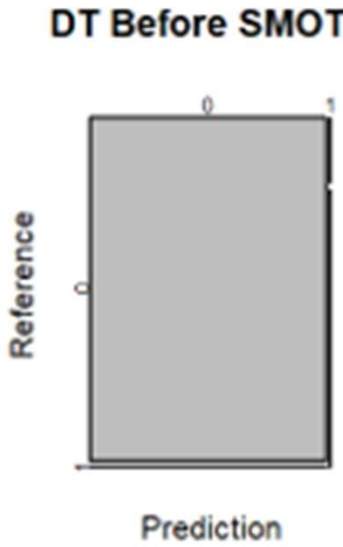


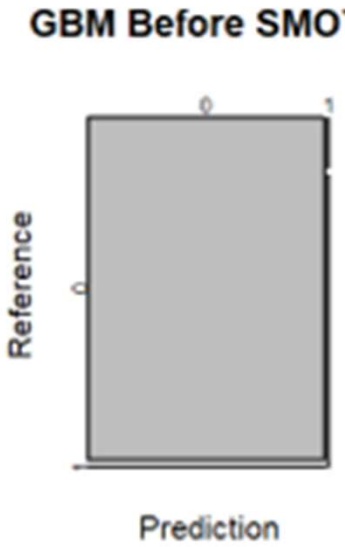
Decision Tree Model after SMOTE

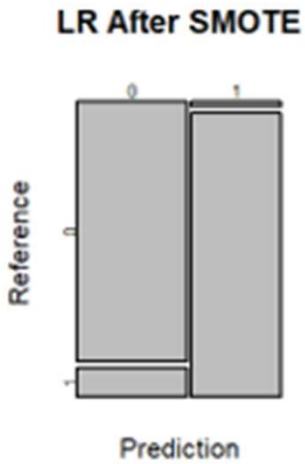


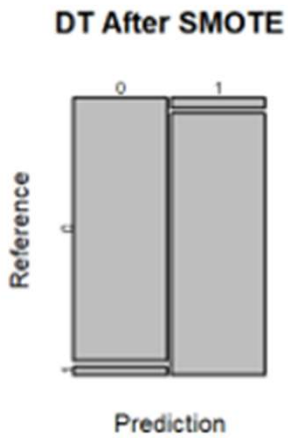
Confusion matrices Before and After smote

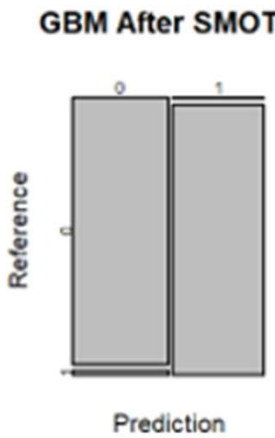




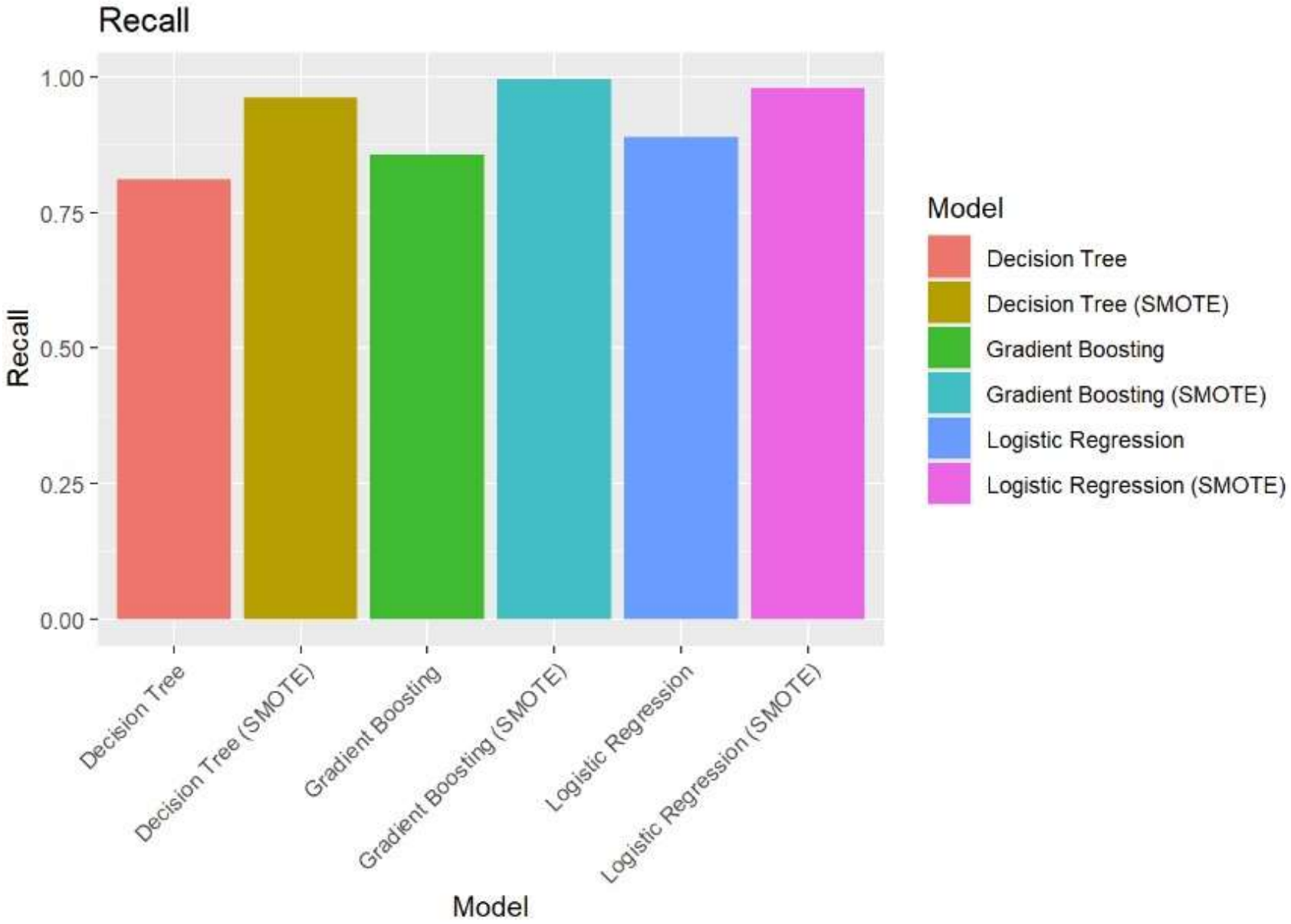


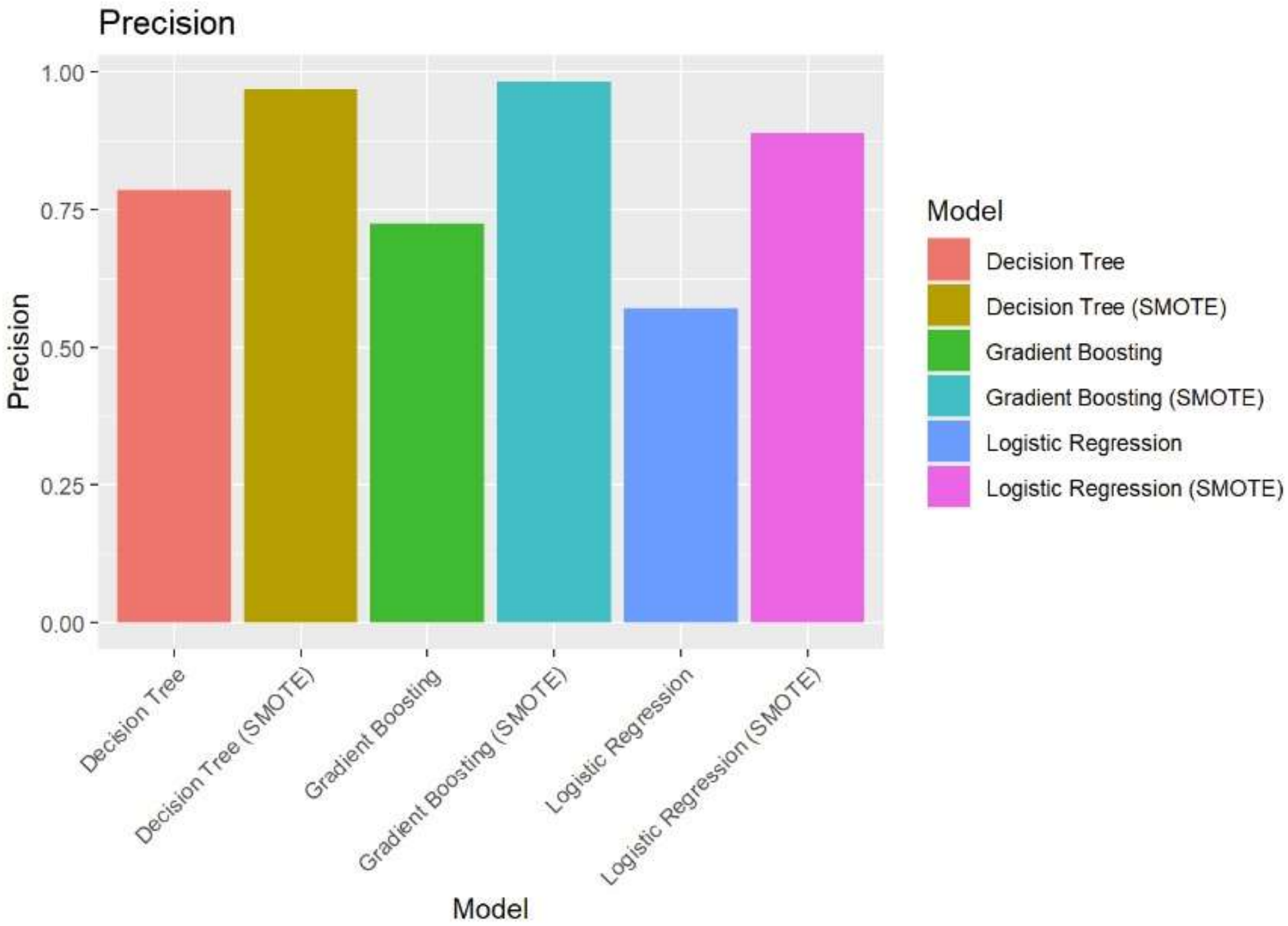






Precision and Recall




Precision
1.00
0.75
0.50
0.25
0.00
Precision
Decision Tree
Decision Tree (SMOTE)
Gradient Boosting
Gradient Boosting (SMOTE)
Logistic Regression
Logistic Regression (SMOTE)
Model
Model
Decision Tree
Decision Tree (SMOTE)
Gradient Boosting
Gradient Boosting (SMOTE)
Logistic Regression
Logistic Regression (SMOTE)

Performance Analysis

- **Logistic Regression (SMOTE):** Achieves high precision (0.889) and recall (0.980), effectively identifying positive cases while minimizing false positives.
- **Decision Tree (SMOTE):** Demonstrates robustness with remarkable precision (0.969) and recall (0.962), effectively classifying both positive and negative instances.
- **Gradient Boosting (SMOTE):** Performs exceptionally well with near-perfect precision (0.982) and recall (0.996), effectively identifying positive cases while minimizing false negatives.
- **Conclusion:** SMOTE significantly improves predictive capabilities, effectively addressing class imbalance and enhancing model performance.

APPLICATIONS

- These techniques of detecting the fraud transactions can be implemented to make the transactions through the credit cards more secure.
- The learned models can be deployed on high-performance computers to monitor all the transactions in real time and stop the ongoing transaction while also alerting the owner about the steps required to further eliminate the risk

CONCLUSION

SMOTE oversampling significantly elevated the performance of key algorithms:

- **Logistic Regression (After SMOTE):** Achieved top-tier performance, boasting exceptional precision (0.889) and recall (0.980).
- **Decision Tree (After SMOTE):** Demonstrated a substantial leap forward, showcasing notable precision (0.969) and recall (0.962).
- **GBM (After SMOTE):** Presented enhanced results, with precision (0.982) and recall (0.996) nearing perfection.
- **Before SMOTE, algorithms showed:**
- **Logistic Regression (Before SMOTE):** Displayed commendable performance.
- **Decision Tree & GBM (Before SMOTE):** Presented moderate results.

References

[1] G. O. [1]Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson and Gianluca Bontempi. Calibrating Probability with Undersampling for Unbalanced Classification. In Symposium on Computational Intelligence and Data Mining (CIDM), IEEE, 2015

[2] [2]http://mlg.ulb.ac.be(URL).

[3] [3]http://www.businesswire.com/news/home/20150804007054/en/Global-Card-Fraud-Losses-Reach-16.31-Billion

[4] [4]http://www.kaggle.com(URL).

[5] [5]The Nilson Report

[6] [6]http://www.thinksaveretire.com/2015/09/14/how-credit-card-fraud-detection-works/(URL).

[7] [7]SMOTE: Synthetic Minority Over-sampling Technique. Nitesh V. Chawla chawla@csee.usf.edu. Department of Computer Science and Engineering

[8] [8]Area Under the Precision-Recall Curve: Point Estimates and Confidence Intervals Kendrick Boyd1 , Kevin H. Eng , and C. David Page

[9] [9]http://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html(URL).

[10] [11]Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[11] [10]Jones E, Oliphant E, Peterson P, et al. SciPy: Open Source Scientific Tools for Python, 2001-, http://www.scipy.org/ [Online; accessed 2017-03-30]

[12] [11]Stéfan van der Walt, S. Chris Colbert and Gaël Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation, Computing in Science & Engineering, 13, 22-30 (2011), DOI:10.1109/MCSE.2011.37