MACHINE LEARNING

PHASE 2

Team 17

Name	Sec.	B.N	ID
Mohamed Yasser Mohamed	2	12	9211066
Mustafa Tarek Salah	2	22	9211178
Ahmed Bassem ELKady	1	7	9210048
Daniel Nabil Khalil	1	16	9210386

TABLE OF CONTENTS

CONTRIBUTION

Name	Contribution
Mohamed Yasser Mohamed	Validation & Model Selection
Mustafa Tarek Salah	
Ahmed Bassem ELKady	Analyzing & Preprocessing Data
Daniel Nabil Khalil	

PROBLEM DEFINITION

CREDIT CARD FRAUD DETECTION

Credit card fraud detection is a critical problem in the financial sector, as fraudulent transactions result in significant financial losses and security threats for both consumers and financial institutions.

The primary challenge in fraud detection is the highly imbalanced nature of the data, where fraudulent transactions constitute only 0.172% (or approximately 1 in 581) of all transactions. Traditional classification models struggle to correctly identify fraudulent cases due to this imbalance, often leading to a high number of false negatives, which can be costly.

The dataset collected by Worldline and the Machine Learning Group of ULB consists of 30 numerical features, 28 of which are the result of a Principal Component Analysis (PCA) transformation, while the remaining two, 'Time' and 'Amount,' retain their original values. The use of PCA-transformed features makes it challenging to interpret the underlying patterns directly.

The goal of this project is to develop a robust machine learning model that accurately detects fraudulent transactions while minimizing false positives and false negatives.

MOTIVATION

With the increasing reliance on digital payment methods, the risk of credit card fraud has grown significantly, posing severe financial and security threats to both consumers and financial institutions.

Fraud detection is, therefore, a crucial application of machine learning, as an effective system can safeguard users and mitigate risks in the financial sector.

This project aims to develop a high-performing fraud detection model by evaluating and comparing various machine learning approaches. By systematically analyzing different models and optimization strategies, we aim to identify the best-performing approach that balances accuracy, precision, recall, and computational efficiency.

The primary goal of this project is not only to enhance fraud detection accuracy but also to provide insights into the trade-offs of different machine learning methods when applied to highly imbalanced datasets. Through rigorous experimentation with undersampling, oversampling, feature selection, and hyperparameter tuning, we seek to develop a model that financial institutions can effectively use to detect fraudulent transactions in real-time.

EVALUATION METRICS

- **Accuracy:** measures the overall correctness of the model by calculating the proportion of correctly classified transactions out of all transactions. However, since fraud detection is a highly imbalanced classification problem, relying solely on accuracy may not provide a meaningful evaluation. So, We will evaluate with other metrics such that precision, recall, and F1-Score.

- **Precision:** Measures the proportion of transactions classified as fraud that are actually fraudulent.
  - **True Positives / (True Positives + False Positives)**

- **Recall:** Measures the proportion of actual fraudulent transactions correctly identified.
  - **True Positives / (True Positives + False Negatives)**

- **F1-Score:** The harmonic mean of precision and recall, providing a balanced evaluation metric when dealing with imbalanced data.
  - **2 × (Precision × Recall) / (Precision + Recall)**

- **ROC-AUC:** ROC curve plots the True Positive Rate against the False Positive Rate at various threshold settings. AUC quantifies the overall ability of the model to discriminate between fraudulent and non-fraudulent transactions. AUC ranges from 0 to 1, where:
  - AUC = 0.5 indicates no discriminative ability (equivalent to random guessing)
  - AUC ≈ 1 indicates excellent discriminative ability

		Actual class	
		P	**N**
Predicted class	**P**	**TP**	FP
	N	FN	**TN**

DATASET LINKS

- **Dataset Link:**
  - Credit Card Fraud Detection
- **Dataset Resources:**
  - Resources

RESULTS: ORIGINAL

Decision Tree

Classification Report

	precision	recall	f1-score	support
0	1.00	1.00	1.00	56864
1	0.82	0.72	0.77	98
accuracy			1.00	56962
macro avg	0.91	0.86	0.88	56962
weighted avg	1.00	1.00	1.00	56962

Confusion Matrix

```
[[56848    16]
 [   27    71]]
```

K-Nearest Neighbors (KNN)

Classification Report

	precision	recall	f1-score	support
0	1.00	1.00	1.00	56864
1	0.93	0.77	0.84	98
accuracy			1.00	56962
macro avg	0.96	0.88	0.92	56962
weighted avg	1.00	1.00	1.00	56962

Confusion Matrix

```
[[56858     6]
 [   23    75]]
```

XGBoost

Classification Report

	precision	recall	f1-score	support
0	1.00	1.00	1.00	56864
1	0.93	0.77	0.84	98
accuracy			1.00	56962
macro avg	0.96	0.88	0.92	56962
weighted avg	1.00	1.00	1.00	56962

Confusion Matrix

```
[[56858     6]
 [   23    75]]
```

Logistic Regression

Classification Report

	precision	recall	f1-score	support
0	1.00	1.00	1.00	56864
1	0.79	0.61	0.69	98
accuracy			1.00	56962
macro avg	0.89	0.81	0.84	56962
weighted avg	1.00	1.00	1.00	56962

Confusion Matrix

```
[[56848    16]
 [   38    60]]
```

Random Forest

Classification Report

	precision	recall	f1-score	support
0	1.00	1.00	1.00	56864
1	0.81	0.70	0.75	98
accuracy			1.00	56962
macro avg	0.91	0.85	0.88	56962
weighted avg	1.00	1.00	1.00	56962

Confusion Matrix

```
[[56848    16]
 [   29    69]]
```

RESULTS: ORIGINAL_FEATURES

Decision Tree

Classification Report

	precision	recall	f1-score	support
0	1.00	1.00	1.00	56864
1	0.71	0.82	0.76	98
accuracy			1.00	56962
macro avg	0.86	0.91	0.88	56962
weighted avg	1.00	1.00	1.00	56962

Confusion Matrix

```
[[56832    32]
 [   18    80]]
```

K-Nearest Neighbors (KNN)

Classification Report

	precision	recall	f1-score	support
0	1.00	1.00	1.00	56864
1	0.71	0.88	0.79	98
accuracy			1.00	56962
macro avg	0.86	0.94	0.89	56962
weighted avg	1.00	1.00	1.00	56962

Confusion Matrix

```
[[56829    35]
 [   12    86]]
```

XGBoost

Classification Report

	precision	recall	f1-score	support
0	1.00	1.00	1.00	56864
1	0.71	0.83	0.76	98
accuracy			1.00	56962
macro avg	0.86	0.91	0.88	56962
weighted avg	1.00	1.00	1.00	56962

Confusion Matrix

```
[[56831    33]
 [   17    81]]
```

Logistic Regression

Classification Report

	precision	recall	f1-score	support
0	1.00	1.00	1.00	56864
1	0.83	0.76	0.79	98
accuracy			1.00	56962
macro avg	0.92	0.88	0.9	56962
weighted avg	1.00	1.00	1.00	56962

Confusion Matrix

```
[[56849    15]
 [   24    74]]
```

Random Forest

Classification Report

	precision	recall	f1-score	support
0	1.00	1.00	1.00	56864
1	0.80	0.76	0.78	98
accuracy			1.00	56962
macro avg	0.90	0.88	0.89	56962
weighted avg	1.00	1.00	1.00	56962

Confusion Matrix

```
[[56846    18]
 [   24    74]]
```

RESULTS: OVERSAMPLED_FEATURES

Decision Tree

Classification Report

	precision	recall	f1-score	support
0	1.00	0.99	0.99	56864
1	0.12	0.85	0.21	98
accuracy			0.99	56962
macro avg	0.56	0.92	0.60	56962
weighted avg	1.00	0.99	0.99	56962

Confusion Matrix

```
[[56246   618]
 [   15    83]]
```

K-Nearest Neighbors (KNN)

Classification Report

	precision	recall	f1-score	support
0	1.00	0.99	1.00	56864
1	0.22	0.89	0.36	98
accuracy			0.99	56962
macro avg	0.61	0.94	0.68	56962
weighted avg	1.00	0.99	1.00	56962

Confusion Matrix

```
[[56559   305]
 [   11    87]]
```

XGBoost

Classification Report

	precision	recall	f1-score	support
0	1.00	1.00	1.00	56864
1	0.48	0.89	0.62	98
accuracy			1.00	56962
macro avg	0.74	0.94	0.81	56962
weighted avg	1.00	1.00	1.00	56962

Confusion Matrix

```
[[56770    94]
 [   11    87]]
```

Logistic Regression

Classification Report

	precision	recall	f1-score	support
0	1.00	0.97	0.99	56864
1	0.05	0.92	0.10	98
accuracy			0.97	56962
macro avg	0.53	0.94	0.54	56962
weighted avg	1.00	0.97	0.98	56962

Confusion Matrix

```
[[55249  1615]
 [   8    90]]
```

Random Forest

Classification Report

	precision	recall	f1-score	support
0	1.00	0.98	0.99	56864
1	0.08	0.90	0.15	98
accuracy			0.98	56962
macro avg	0.54	0.94	0.57	56962
weighted avg	1.00	0.98	0.99	56962

Confusion Matrix

```
[[56897   967]
 [   10    88]]
```

RESULTS: UNDERSAMPLED_FEATURES

Decision Tree

Classification Report

	precision	recall	f1-score	support
0	1.00	0.95	0.97	56864
1	0.03	0.90	0.05	98
accuracy			0.95	56962
macro avg	0.51	0.92	0.51	56962
weighted avg	1.00	0.95	0.97	56962

Confusion Matrix

```
[[53754  3110]
 [   10    88]]
```

K-Nearest Neighbors (KNN)

Classification Report

	precision	recall	f1-score	support
0	1.00	0.97	0.99	56864
1	0.05	0.91	0.10	98
accuracy			0.97	56962
macro avg	0.53	0.94	0.54	56962
weighted avg	1.00	0.97	0.98	56962

Confusion Matrix

```
[[55287  1577]
 [   9    89]]
```

XGBoost

Classification Report

	precision	recall	f1-score	support
0	1.00	0.95	0.98	56864
1	0.03	0.93	0.06	98
accuracy			0.95	56962
macro avg	0.52	0.94	0.52	56962
weighted avg	1.00	0.95	0.97	56962

Confusion Matrix

```
[[54108  2756]
 [   7    91]]
```

Logistic Regression

Classification Report

	precision	recall	f1-score	support
0	1.00	0.96	0.98	56864
1	0.03	0.93	0.07	98
accuracy			0.96	56962
macro avg	0.52	0.94	0.52	56962
weighted avg	1.00	0.96	0.98	56962

Confusion Matrix

```
[[54348  2516]
 [   7    91]]
```

Random Forest

Classification Report

	precision	recall	f1-score	support
0	1.00	0.97	0.98	56864
1	0.05	0.89	0.09	98
accuracy			0.97	56962
macro avg	0.52	0.93	0.54	56962
weighted avg	1.00	0.97	0.98	56962

Confusion Matrix

```
[[55185  1679]
 [   11   87]]
```

SVM

Classification Report

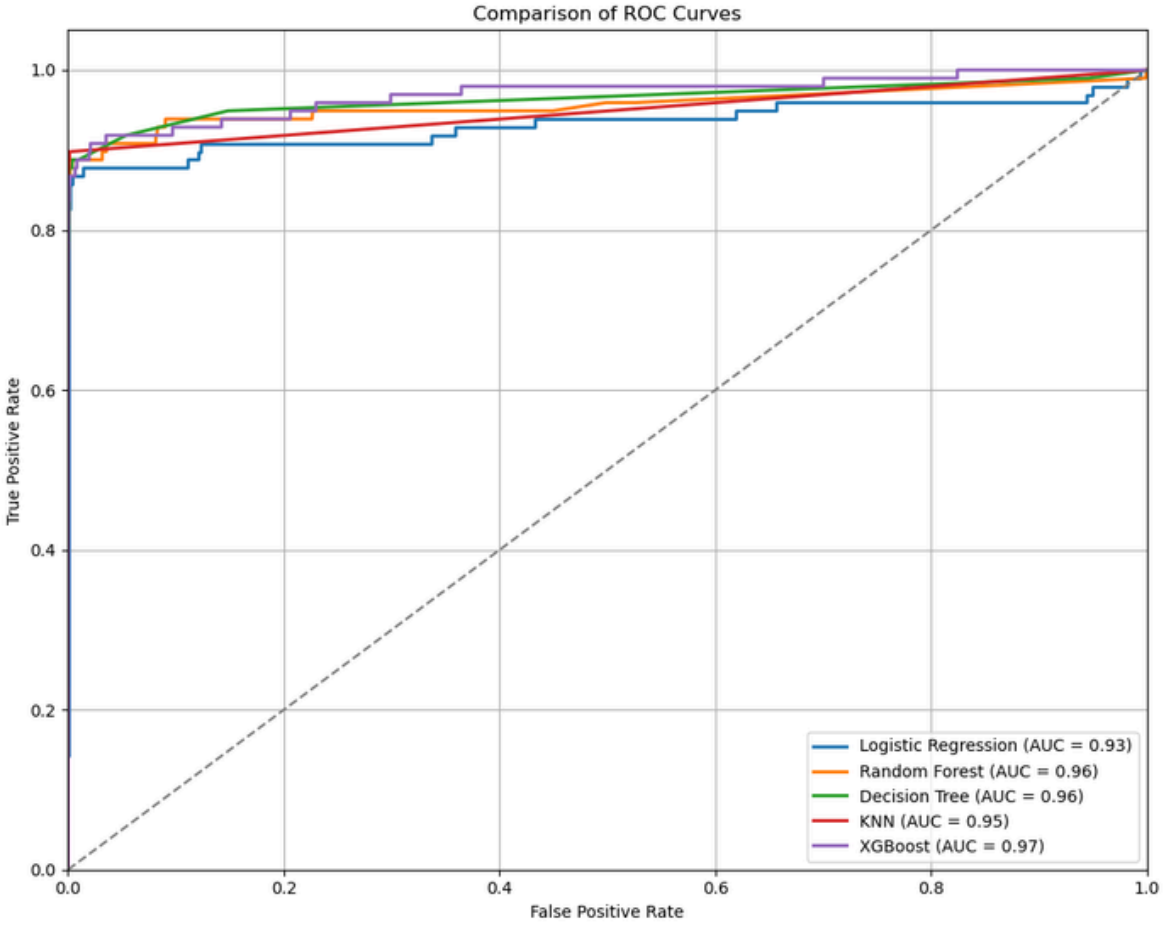
	precision	recall	f1-score	support
0	1.00	0.97	0.99	56864
1	0.05	0.92	0.10	98
accuracy			0.97	56962
macro avg	0.53	0.94	0.54	56962
weighted avg	1.00	0.97	0.98	56962

Confusion Matrix

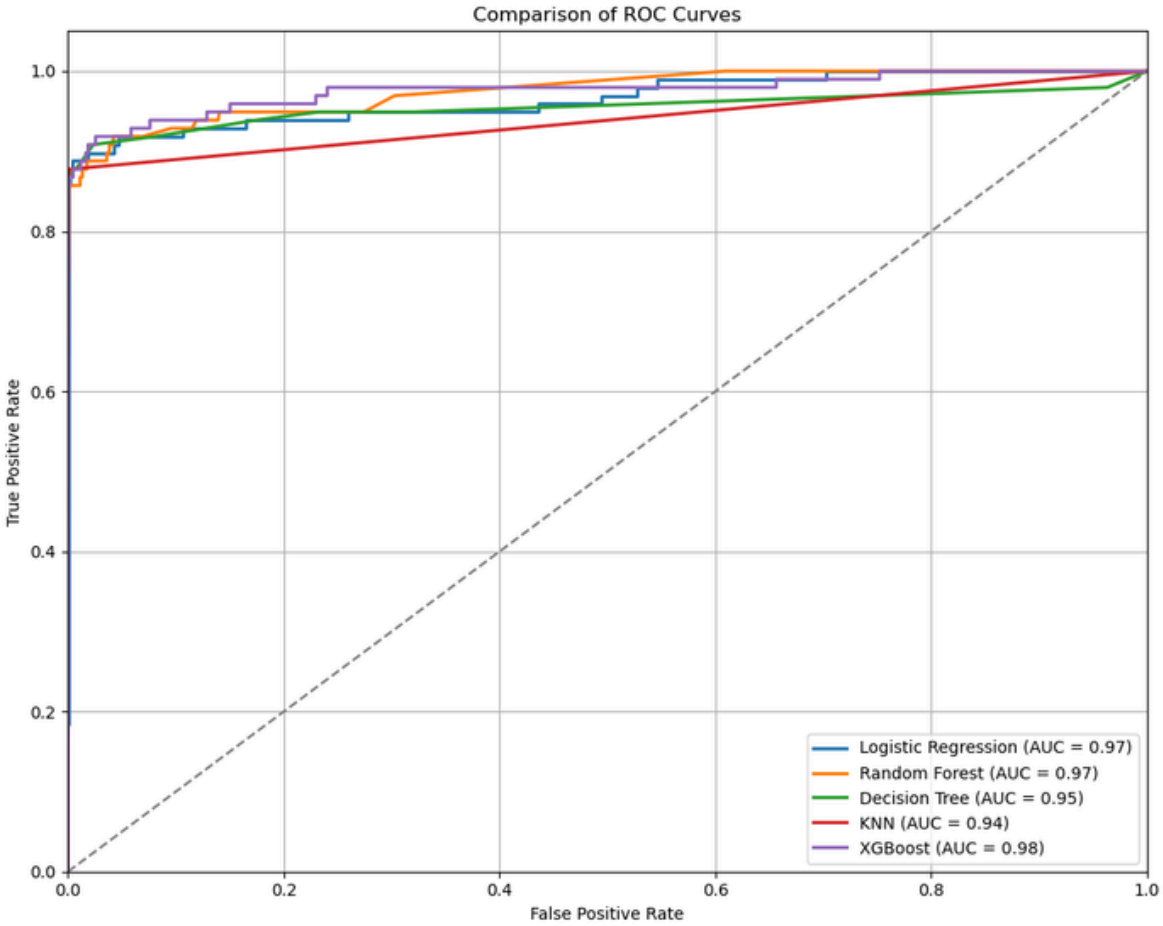
```
[[55234  1630]
 [ 8     90]]
```

RESULTS

ORIGINAL

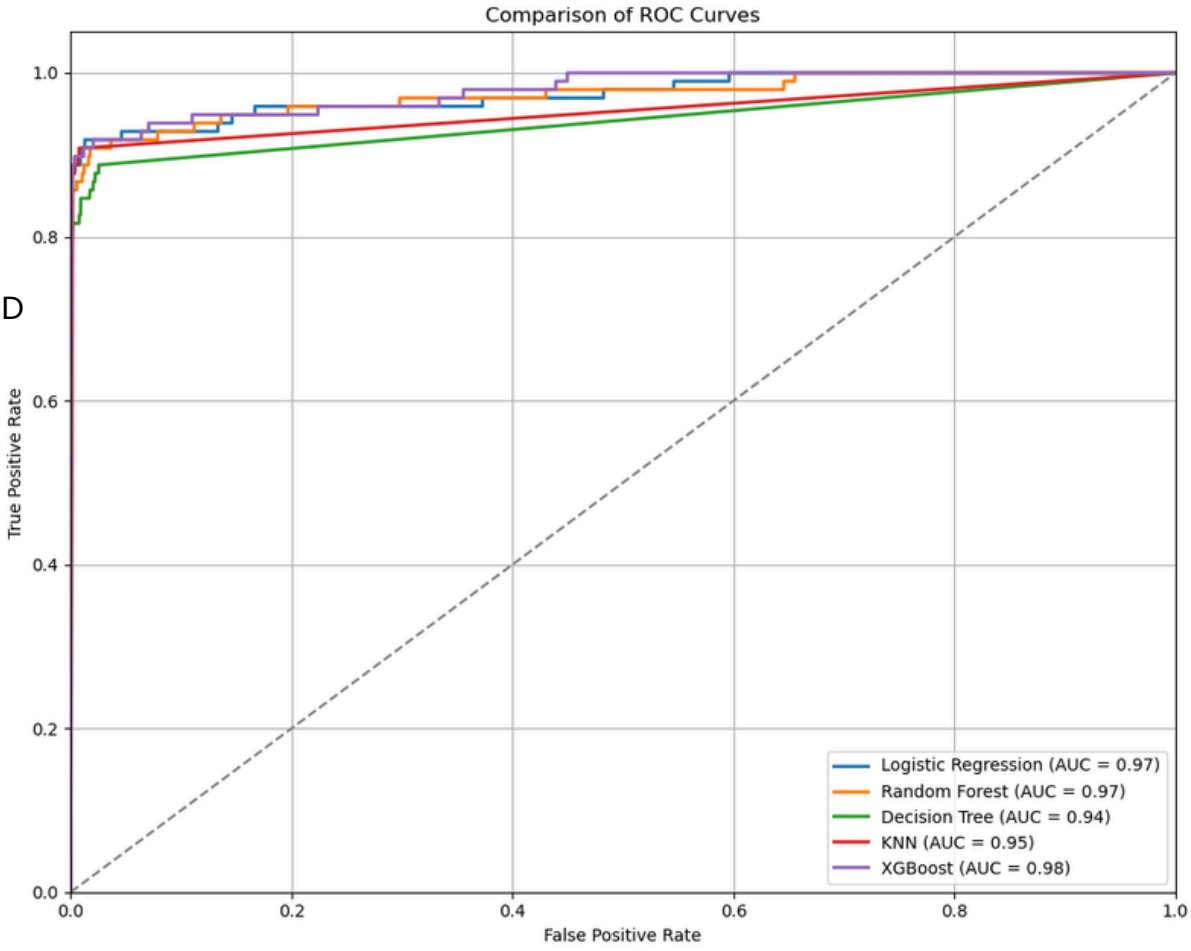


ORIGINAL FEATURES

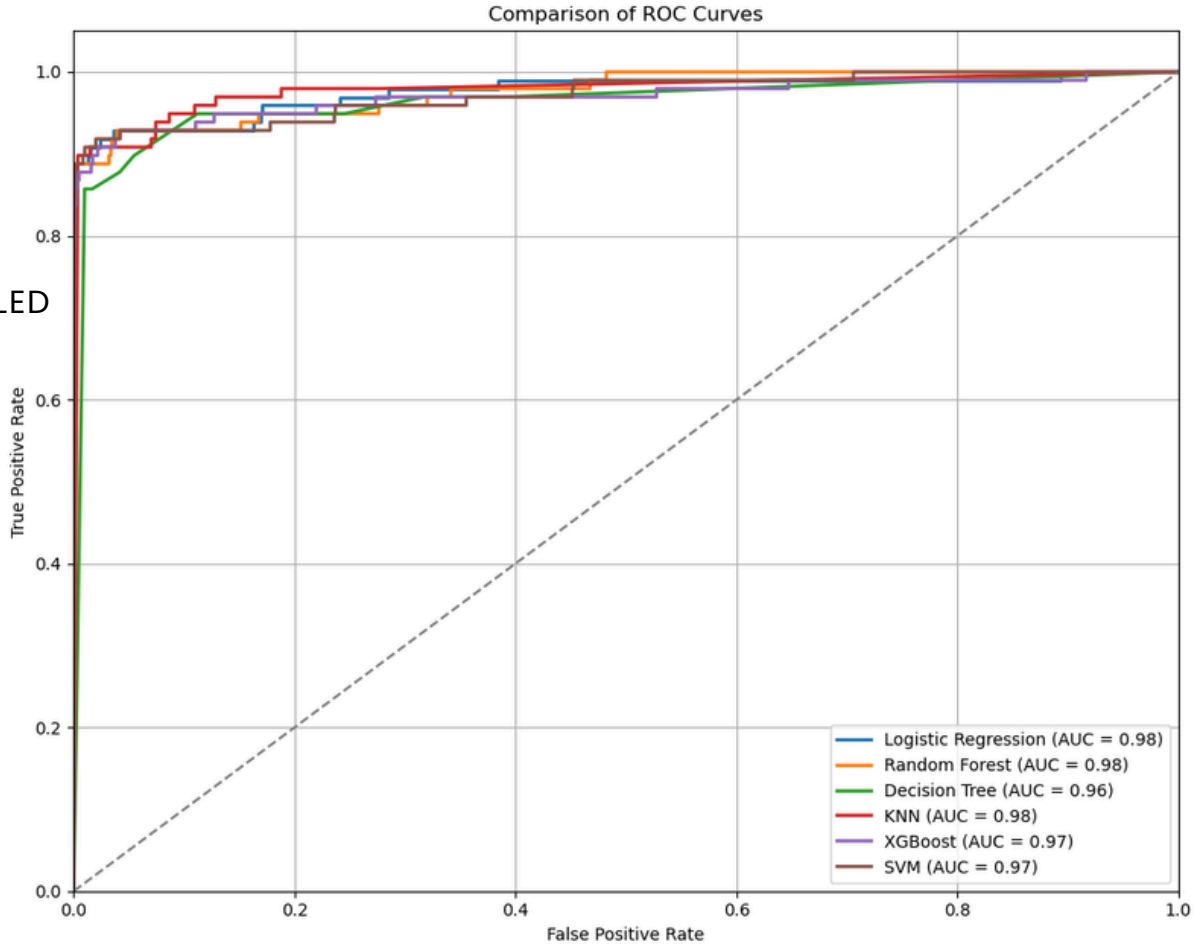


RESULTS

OVERSAMPLED FEATURES



UNDERSAMPLED FEATURES



EXPERIMENTAL RESULTS

Data	Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
All Dataset	KNN	0.9994	0.8949	0.7665	0.8253	0.9211
	XGBOOST	0.9995	0.9323	0.7691	0.8425	0.9819
	DTREE	0.9993	0.8756	0.7260	0.7930	0.9342
	RF	0.9992	0.8701	0.6017	0.7100	0.9651
	LOG REG	0.9990	0.8790	0.5202	0.6533	0.9841
	SVM	-	-	-	-	-
Undersampled Dataset	KNN	0.9366	0.9943	0.8782	0.9326	0.9717
	XGBOOST	0.9518	0.9741	0.9290	0.9506	0.9828
	DTREE	0.9264	0.9828	0.8681	0.9218	0.9685
	RF	0.9289	0.9971	0.8604	0.9237	0.9776
	LOG REG	0.9353	0.9887	0.8807	0.9311	0.9851
	SVM	0.960659	0.991869	0.819796	0.95937	0.99
Oversampled Dataset	KNN	0.9979	0.9957	1.0000	0.9979	0.9995
	XGBOOST	0.9997	0.9993	1.0000	0.9997	1.0000
	DTREE	0.9888	0.9866	0.9910	0.9888	0.9989
	RF	0.9455	0.993	0.8970	0.9427	0.9923
	LOG REG	0.9536	0.9739	0.9322	0.9526	0.9916
	SVM	-	-	-	-	-
All Dataset (Important Features)	KNN	0.9996	0.9529	0.8148	0.8781	0.9250
	XGBOOST	0.9996	0.9658	0.7843	0.8653	0.9817
	DTREE	0.9993	0.8086	0.7741	0.7895	0.9397
	RF	0.9994	0.9576	0.6853	0.7989	0.9715
	LOG REG	0.9993	0.9956	0.6015	0.7493	0.9770
	SVM	-	-	-	-	-
Undersampled Dataset (Important Features)	KNN	0.9332	0.9804	0.8858	0.9306	0.9712
	XGBOOST	0.9435	0.9607	0.9264	0.9430	0.9836
	DTREE	0.9254	0.9828	0.8680	0.9218	0.9690
	RF	0.9370	0.9917	0.8832	0.9342	0.9788
	LOG REG	0.9345	0.9911	0.8782	0.9308	0.9802
	SVM	0.94473	0.986149	0.90355	0.943046	0.98
Oversampled Dataset (Important Features)	KNN	0.9967	0.9935	1.0000	0.9967	0.9992
	XGBOOST	0.9991	0.9983	0.9999	0.9991	1.0000
	DTREE	0.9852	0.9843	0.9864	0.9854	0.9987
	RF	0.9504	0.9835	0.9175	0.9494	0.9908
	LOG REG	0.9369	0.9690	0.9043	0.9355	0.9864
	SVM	-	-	-	-	-

CONCULSION

- After testing the model trained on the full set of original features, we achieved high F1 scores and recall across all models on the test data.
- When we reduced the input to the top 10 most important features, the performance remained nearly identical to that of the full feature set, while significantly improving computational efficiency.
- We also explored oversampling and undersampling techniques. Both led to slightly lower overall scores compared to the original distribution, as it increased recall substantially but at the cost of reduced precision.
- In contrast, undersampling achieved similarly high recall but resulted in nearly zero precision.
- Despite this, we consider undersampling to reflect a more realistic scenario, as it relies solely on actual data rather than synthetic samples."

PIPELINE: PREPROCESSING

1. Data Loading and Initial Inspection:

- The dataset was loaded from a CSV file containing credit card transactions, with the goal of identifying fraudulent behavior. Each transaction is represented by anonymized numerical features (V1 to V28), along with Time, Amount, and the Class label indicating whether a transaction is fraudulent (1) or not (0).

2. Feature Categorization:

- The dataset contains 30 numerical features:
  - Time: Seconds elapsed between this transaction and the first transaction.
  - Amount: The transaction amount.
  - V1–V28: Anonymized features resulting from PCA transformation.
- The categorical target label
  - Class: (0 = non-fraudulent, 1 = fraudulent).

3. Target Variable Exploration:

- The dataset is found to be highly imbalanced, with fraudulent transactions accounting for only ~0.17% of the data. This class imbalance is a central challenge in the modeling stage.
- Percentages:
  - Non-fraudulent transactions: ~99.83%
  - Fraudulent transactions: ~0.17%

4. Data Scaling & Splitting:

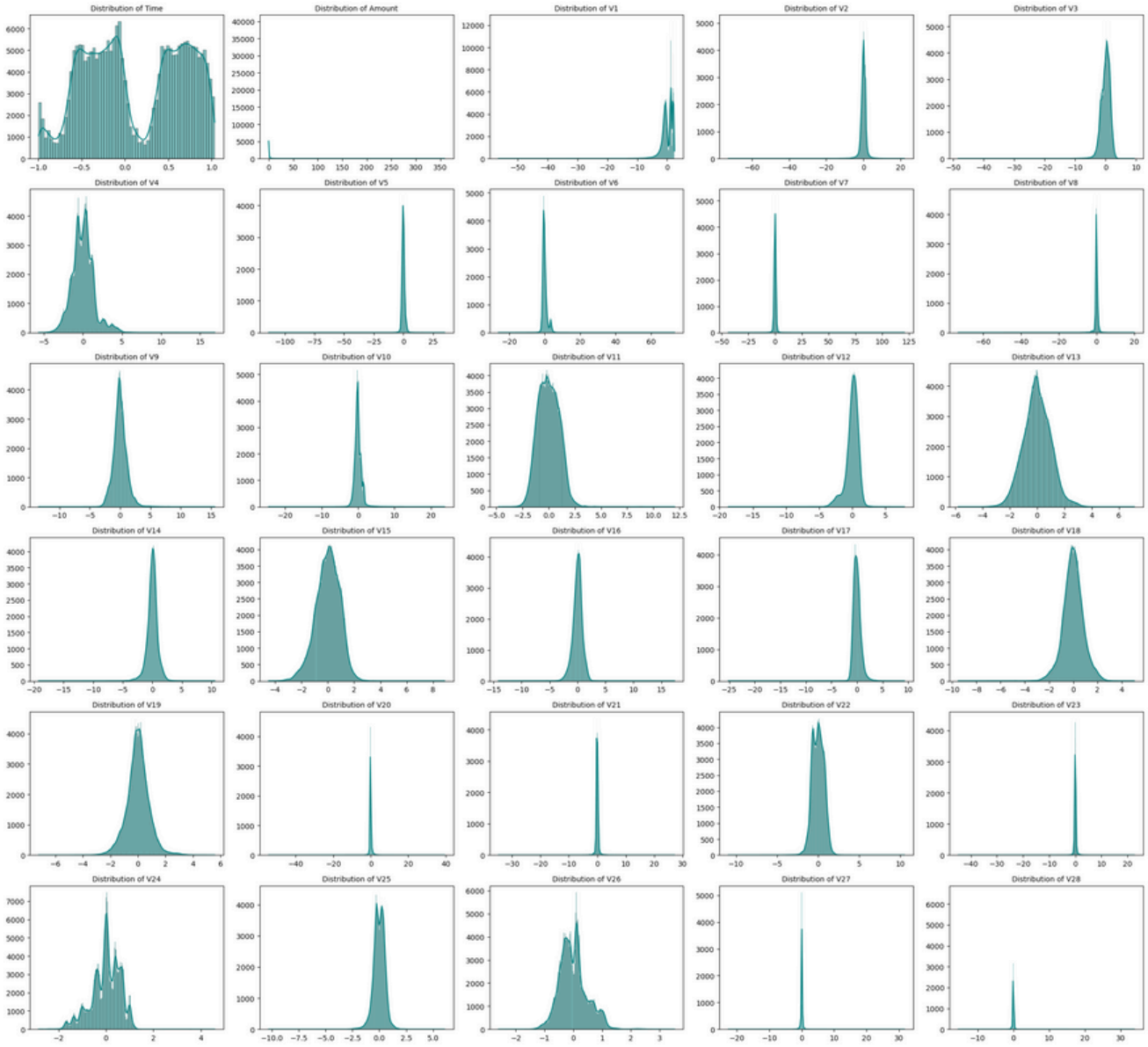
- The Amount and Time features were scaled using RobustScaler (less sensitive to outliers).
- The dataset was split into training (80%) and testing (20%) subsets using stratified sampling, preserving the proportion of fraud vs. non-fraud cases in both sets. This ensures the model is exposed to a similar distribution during training and evaluation.

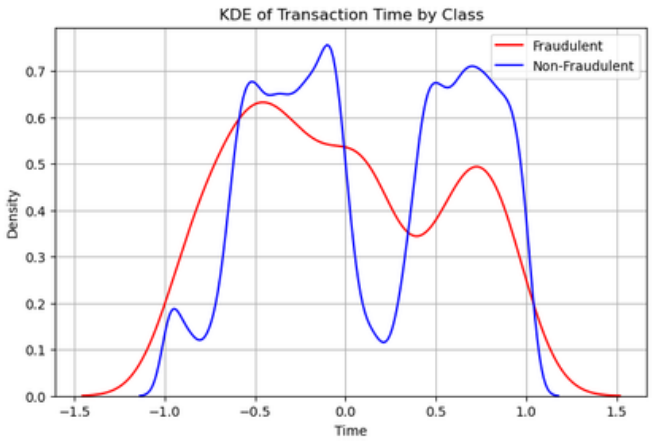
```
Original Distribution: [0.99827251 0.00172749]
Train Distribution   : [0.99827075 0.00172925]
Test Distribution    : [0.99827955 0.00172045]
```

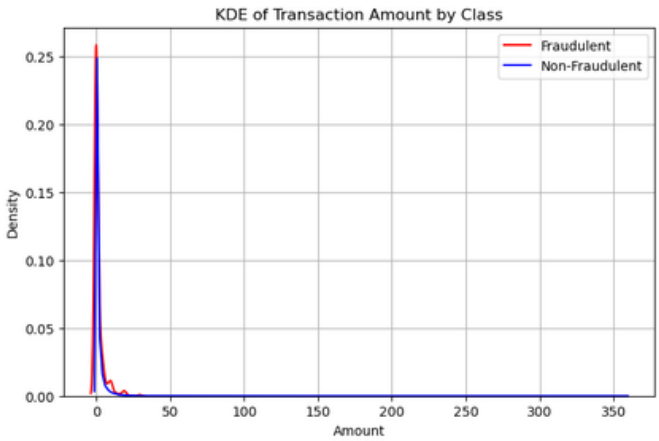
PIPELINE: PREPROCESSING

5. Feature Distribution Analysis:

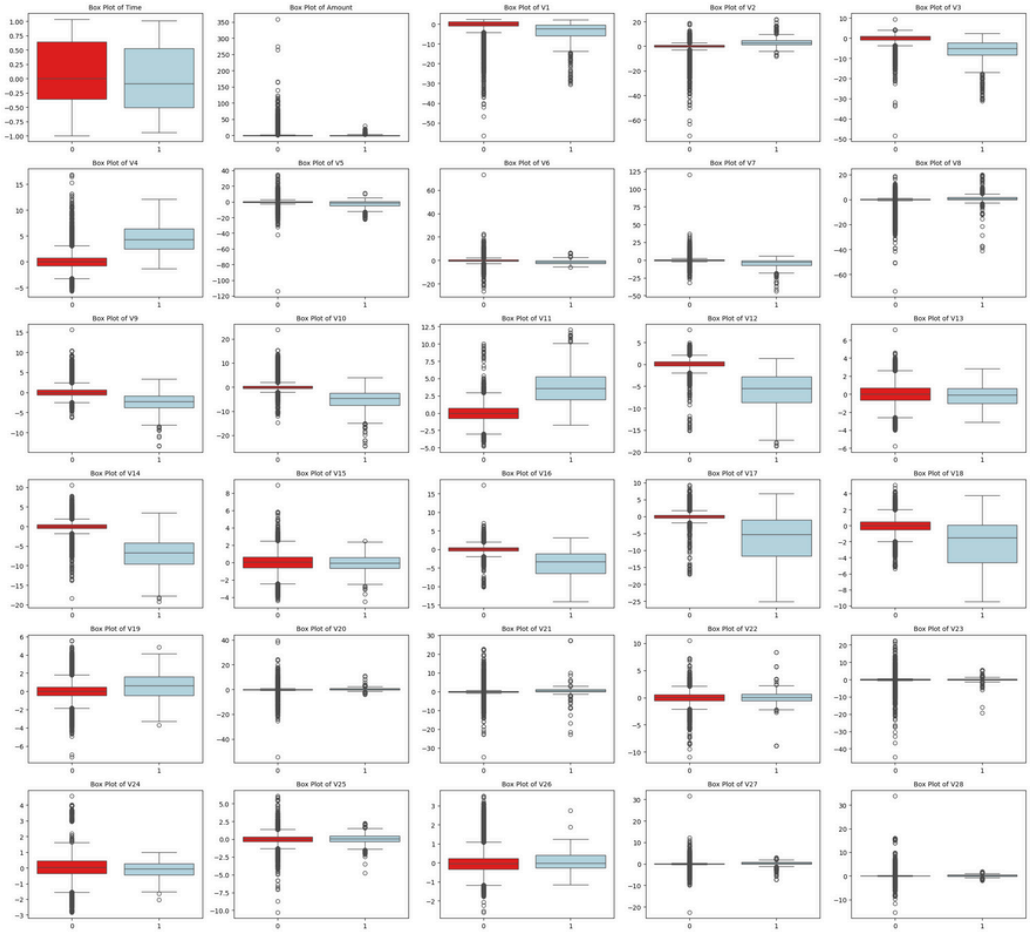
- Histograms
- KDE Plots by Class
- Boxplots
- Correlation Matrix Heatmap

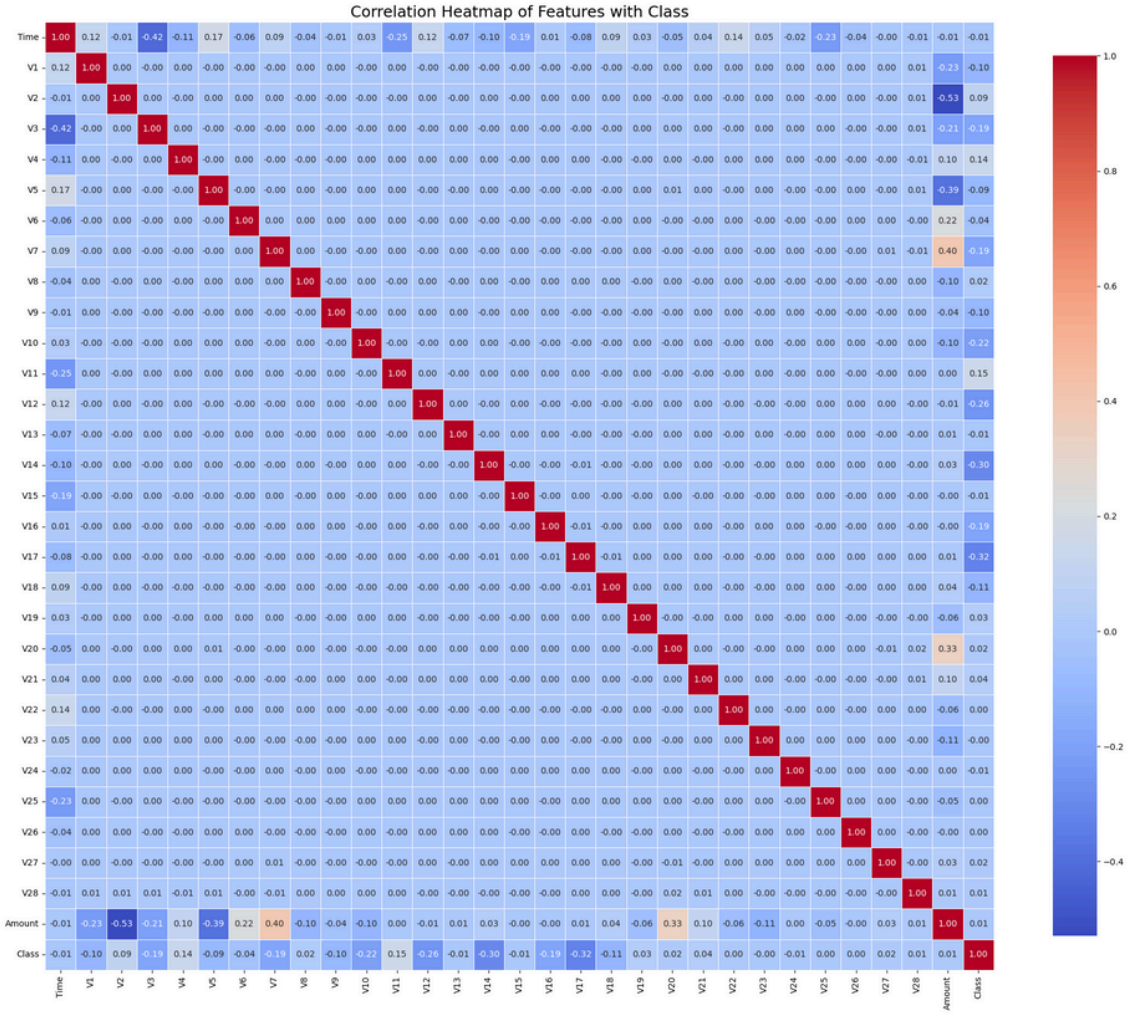






PIPELINE: PREPROCESSING





PIPELINE: PREPROCESSING

- Based on the box plots & correlation heatmap, we observe that features V4 and V11 exhibit a positive correlation with fraud — as the values of these features increase, the likelihood of a transaction being fraudulent also increases.
- In contrast, features V3, V7, V10, V12, V14, V16, V17, and V18 show a negative correlation with fraud, indicating that lower values of these features are more associated with fraudulent transactions.

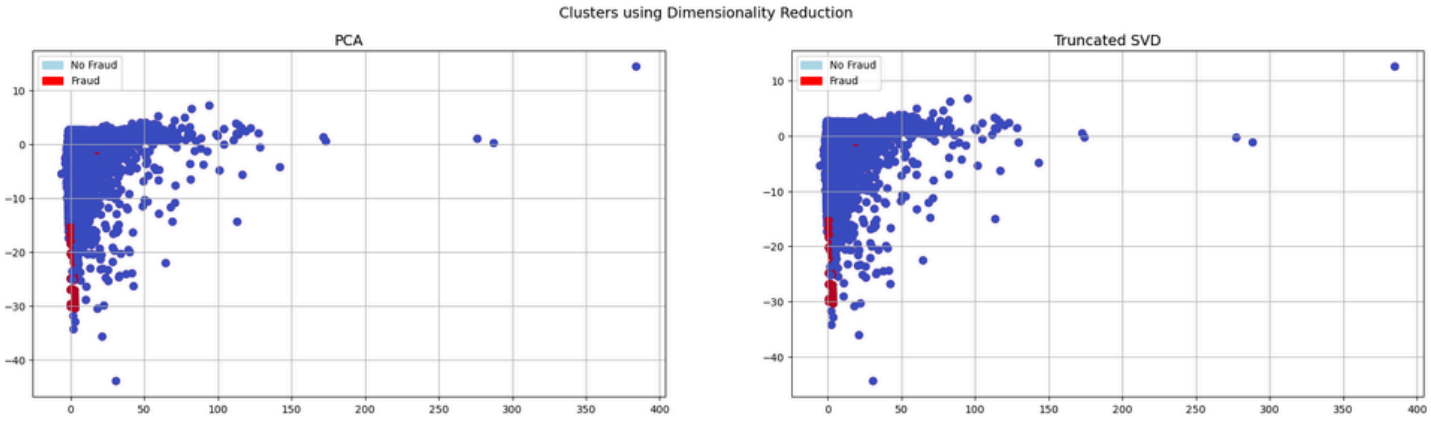
6. Outlier Removal:

- Since fraud data itself is inherently anomalous, outliers were removed only from non-fraudulent transactions. This approach prevents the accidental removal of legitimate fraud signals.

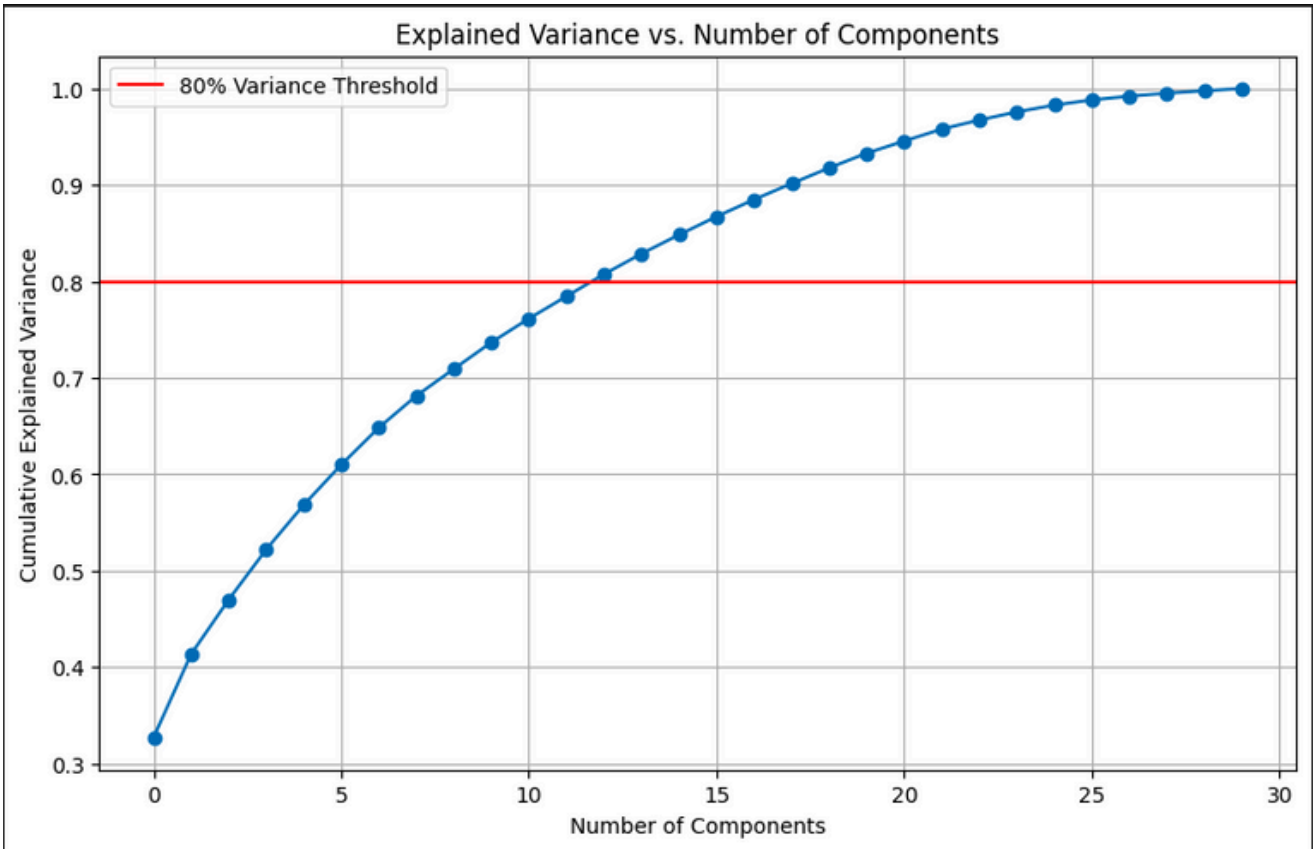
```
V3 Outliers: 2464
Removed 2464 rows from V3
-------------------------------
V4 Outliers: 8765
Removed 8765 rows from V4
-------------------------------
V7 Outliers: 6891
Removed 6891 rows from V7
-------------------------------
V10 Outliers: 7263
Removed 7263 rows from V10
-------------------------------
V11 Outliers: 382
Removed 382 rows from V11
```

```
V12 Outliers: 12022
Removed 12022 rows from V12
-----------------------------------
V14 Outliers: 11042
Removed 11042 rows from V14
-----------------------------------
V16 Outliers: 6290
Removed 6290 rows from V16
-----------------------------------
V17 Outliers: 5551
Removed 5551 rows from V17
-----------------------------------
V18 Outliers: 5909
Removed 5909 rows from V18
```

7.Dimensionality Reduction and Clustering:



PIPELINE: PREPROCESSING



We attempted to visualize fraud vs. non-fraud classes using 2 PCA components, but observed significant overlap with poor class separation. Our analysis revealed these 2 components captured only 41% of the total variance, meaning we lost approximately 60% of the dataset's information.

The explained variance curve showed we would need around 12 components to reach the standard 80% variance threshold. However, we didn't pursue this approach since 12-dimensional data cannot be effectively visualized and Interpretation would become challenging

Instead, we conducted feature importance analysis to identify the 9 most predictive features from the original dataset. This approach maintains interpretability while focusing specifically on the signals most relevant to fraud detection, enabling more meaningful visualization and analysis.

PIPELINE: PREPROCESSING

8. Under-Sampling:

- We used under-sampling technique to reduce the number of samples in the majority class (non-fraud) to match the number in the minority class (fraud), resulting in a balanced training dataset.
  - Before Under-Sampling: The class distribution was skewed toward non-fraudulent transactions.
  - After Under-Sampling: Both classes had equal representation, helping to reduce bias in machine learning models.

Original class distribution: Counter({0: 227451, 1: 394})
Undersampled class distribution: Counter({0: 394, 1: 394})

9. Over-Sampling:

- To further evaluate class balancing strategies, SMOTE (Synthetic Minority Over-sampling Technique) was applied. This technique generates synthetic samples for the minority class (fraud), rather than simply duplicating them.
  - Before Over-Sampling: The dataset was imbalanced.
  - After Over-Sampling: Both classes were equally represented.

Original class distribution: Counter({0: 227451, 1: 394})
Oversampled class distribution: Counter({0: 227451, 1: 227451})

10. Applying ZeroR-Baseline Evaluation:

Predicted Class: 0
Accuracy: 99.82795547909133%
Precision: 0.0000
Recall: 0.0000
F1 Score: 0.0000

PIPELINE: MODELING

In our study, we implemented six different models(Logistic Regression, KNN, XGBOOST, Decision Tree, SVM, Random Forest) to evaluate classification performance across various data preparation techniques. For each model, we conducted hyperparameter optimization using Grid Search and validated results with k-Fold Cross-Validation (k=5) to ensure robustness. The models were built on both raw and engineered feature sets, as well as on data with different sampling strategies (undersampling, oversampling).
Below, we outline the detailed pipeline applied consistently to each model variant.

Step 1: Data Loading

- Load the required CSV files for training and testing from the designated directory.
- These include raw data, undersampled, oversampled, and datasets with selected or engineered features.

Step 2: Data Preparation

- Separate features (X) and labels (y) from each dataset.
- Depending on the model variant, use either all original features or a selected subset of important features.

Step 3: Model Setup and Hyperparameter Tuning

- Define the logistic regression model.
- Set up a grid of hyperparameter values for parameters.
- Use GridSearchCV with roc_auc as the scoring metric and k-fold cross-validation to find the optimal parameters.

Step 4: Best Estimator Extraction

- Retrieve the best-performing model based on cross-validation results.
- This model is automatically configured with the best values.

Step 5: Model Training

- Train a new model using the best params identified.
- Fit the model on the entire training dataset.

PIPELINE: MODELING

Step 7: Performance Evaluation on Train Data

- Generate predictions on the training set.
- Compute and display key evaluation metrics:
  - Confusion Matrix
  - Accuracy
  - Precision
  - Recall (Sensitivity)
  - F1 Score
  - Specificity
  - Classification Report

Step 8: Cross-Validation Score

- Perform another round of 5-fold cross-validation using the final model.
- Report the average score to confirm model stability and generalization performance.

Step 9: Iterate Over All Training Variants

Repeat Steps 1 through 8 for each of the following training datasets to ensure a comprehensive evaluation across different preprocessing strategies:

- train_data.csv – Original training data without sampling or feature extraction
- undersampled_train_data.csv – Training data after undersampling the majority class
- oversampled_train_data.csv – Training data after oversampling the minority class
- extracted_features_train_data.csv – Training data with selected features
- extracted_features_undersampled_train_data.csv – Feature selected data with undersampling
- extracted_features_oversampled_train_data.csv – Feature selected data with oversampling

By systematically applying the full pipeline to each variant, we can fairly compare how sampling and feature engineering impact model performance.