Machine Learning

Project Report

Team 10

Name	Sec	B.N	Code
Ahmed Osama Helmy	1	5	9213061
Omar Mahmoud	1	29	9210758
Abdallah Ahmed	1	25	9210652
Aliaa Gheis	1	27	9210694

Contents

Problem Statement:

Credit card fraud poses a significant threat to both consumers and financial institutions, resulting in substantial financial losses and eroding customer trust. To counter this, banks increasingly rely on machine learning to analyze transaction patterns and detect anomalies indicative of fraudulent activity. Given the constantly evolving tactics of fraudsters, these models must be continuously updated and refined to remain effective.

The objective is to build a **robust machine learning model** that can accurately identify fraudulent credit card transactions. This task is driven by the dual need to:

- **Protect consumers** from unauthorized charges.
- **Safeguard financial institutions** from revenue loss and reputational damage.

Credit card fraud detection is challenging due to the **class imbalance**—fraudulent transactions are rare compared to legitimate ones. To address this, a mix of **supervised learning algorithms** (e.g., logistic regression, decision trees, random forests, and gradient boosting).

Dataset

Dataset Name: IEEE-CIS Fraud Detection

Link: https://www.kaggle.com/competitions/ieee-fraud-detection/data

Description:

- Contains 590540 transaction records with 433 columns

Project Pipeline:

Data Ingestion

This stage involves loading the dataset and defining its schema. The column names are anonymized and not descriptive, as is typical in fraud detection datasets to preserve confidentiality.

Data Cleaning

Handling Missing Values and Nulls

- We began by calculating the percentage of missing values in each column and sorting them in descending order.
- Columns with a very high proportion of missing values (e.g., over 96%) were dropped, while others with moderate missingness were imputed appropriately.Feature Engineering

Drop Low-Information Columns

- Removed columns with:
    - High missing values (default threshold: >96%)
    - Very low variance (i.e., one value dominates)

Encoding Categorical Data

- All object or category-type features were label encoded.
- • This ensured compatibility with machine learning models by converting strings to numeric values.

Grouping Rare Categories

- In selected categorical columns, rare categories with frequency below a threshold (e.g., 500) were grouped under the label 'Rare'.

Transaction Amount Feature Engineering

- Created new features by dividing TransactionAmt by the mean and standard deviation of groups (e.g., by card or user).
- Helps capture relative transaction size behavior.
- Applied log (1 + x) transformation to TransactionAmt to reduce skewness and improve model stability.

Time-Based Feature Extraction

- Derived temporal features from TransactionDT, such as:
    - o Transaction hour
    - o Weekday
    - o Is weekend (Sat/Sun)
    - o Is nighttime (0–5 AM)

Dropping Unused or Redundant Columns

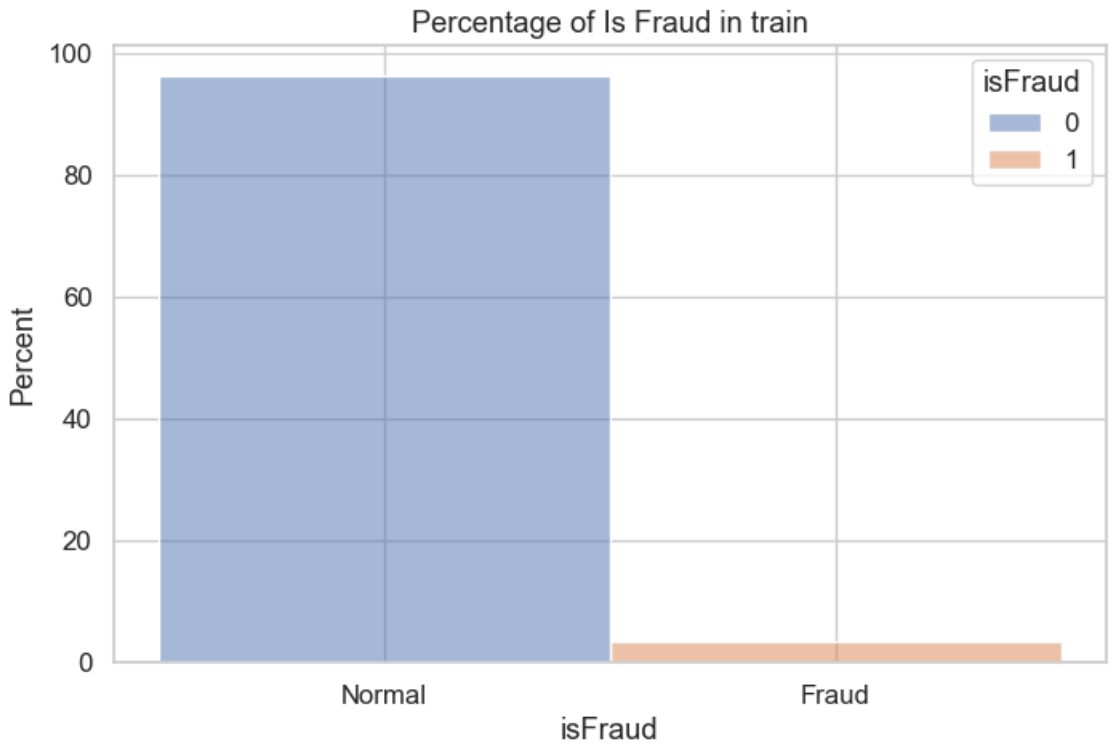
- Removed columns known to have little value based on domain expertise, such as some id_* features and TransactionID.

Dimensionality Reduction

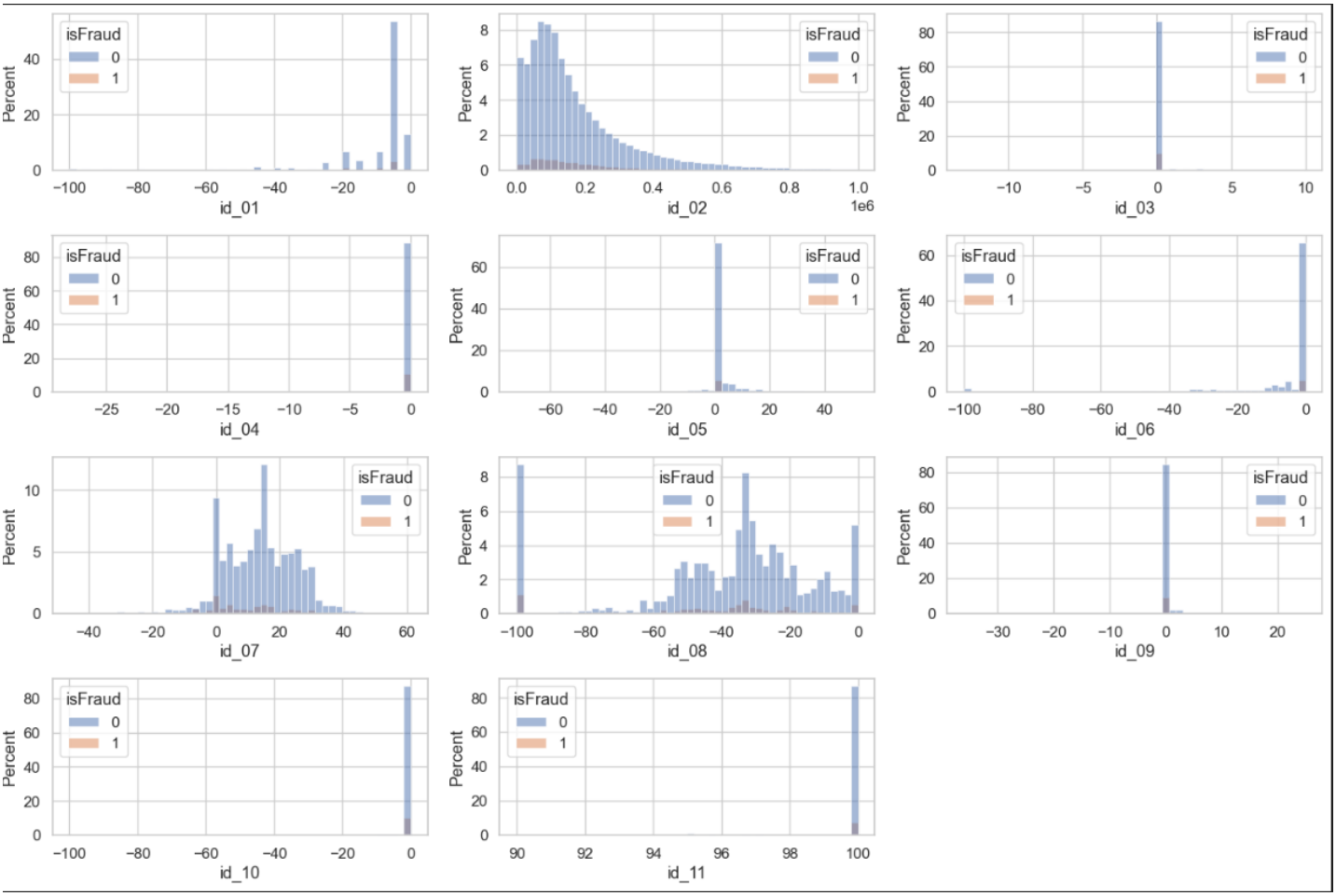
- Principal Component Analysis (PCA) was used to reduce feature space while retaining most variance.
- Useful for improving model performance and reducing noise

Data Visualization & EDA:

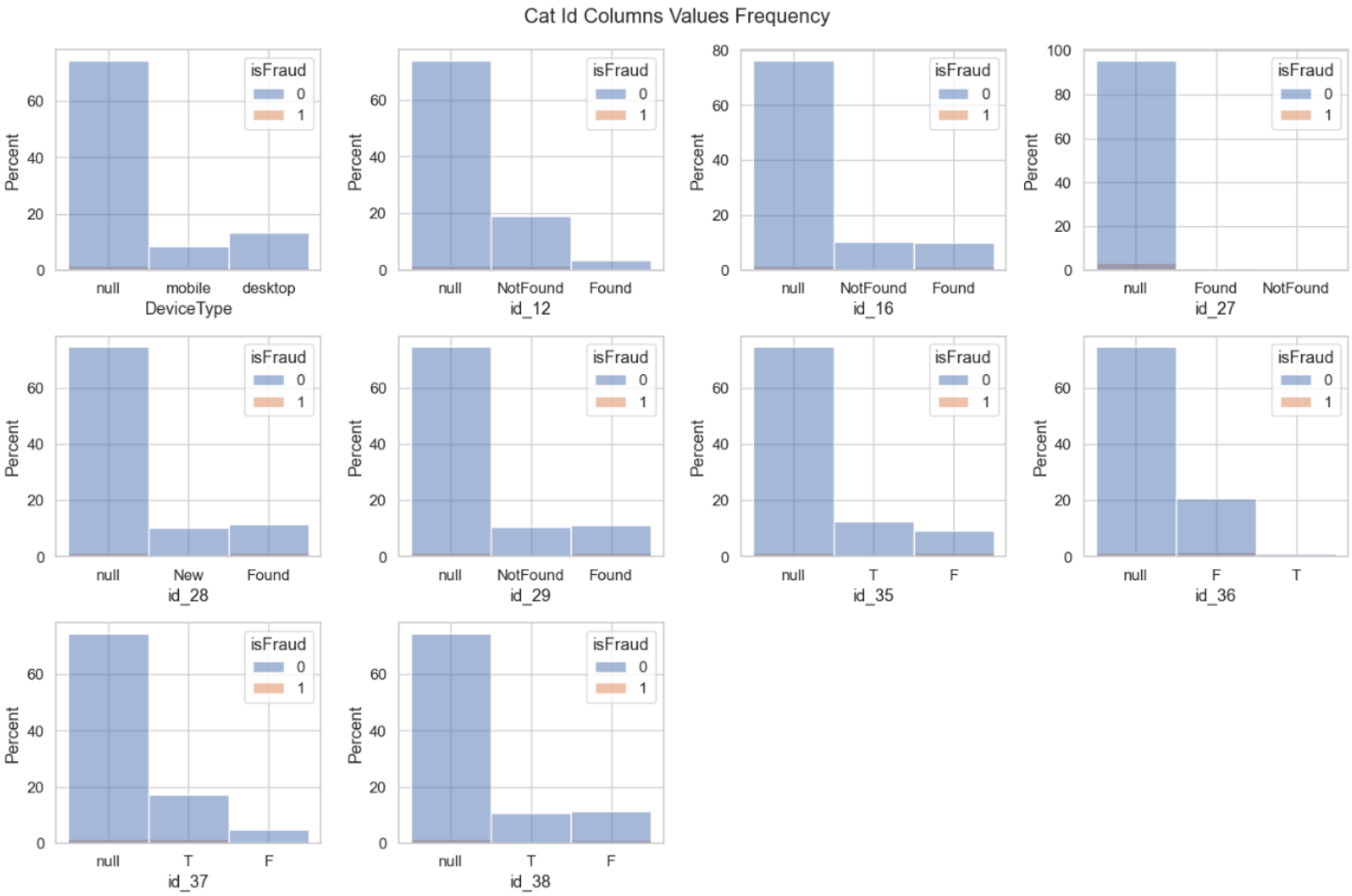
- The dataset is highly skewed — 96.5% of transactions are normal, while only 3.5% are fraudulent. This imbalance can bias the model toward predicting the majority class.
- Several features show very low variance, with values dominated by a single class or category. These may contribute little to model performance and could potentially be removed or transformed.



- High Many features (id_03, id_04, id_06, id_09, id_10, id_11) are dominated by a single value, especially in non-fraud cases, indicating low variance.
- Fraudulent transactions are sparsely distributed and often hidden under the bulk of normal transaction distributions.
- Features like id_07 and id_08 show more spread across fraud and non-fraud, suggesting higher potential for separating the classes.
- Columns such as id_03, id_09, and id_11 have extremely uniform distributions, possibly making them redundant or weak predictors.
- Some features (id_01, id_02, id_05) display subtle distribution shifts for fraud, which may still carry signal despite imbalance.



- Most id_ columns have a high proportion of missing values (often >75%).
- Several columns (e.g., id_08, id_09) show signs of placeholder values like -100, 0, or 100.
- id_02 has extremely large and variable values, likely needs normalization.
- Many categorical id_ columns are dominated by a single value or nulls.
- Common values in categorical features are "T", "F", "Found", "NotFound", "New", and "null".
- Low variation across many features reduces their predictive power.
- Fraud instances slightly deviate from normal in select categories, indicating potential signal.



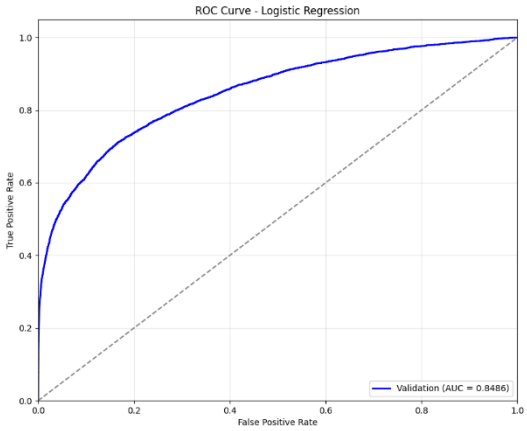
Evaluation Metrics:

- Precision: To minimize false positives.
- Recall: To catch as many frauds as possible.
- F1 Score: For balanced performance.
- AUC-ROC: To measure overall discriminative ability.
- Confusion Matrix

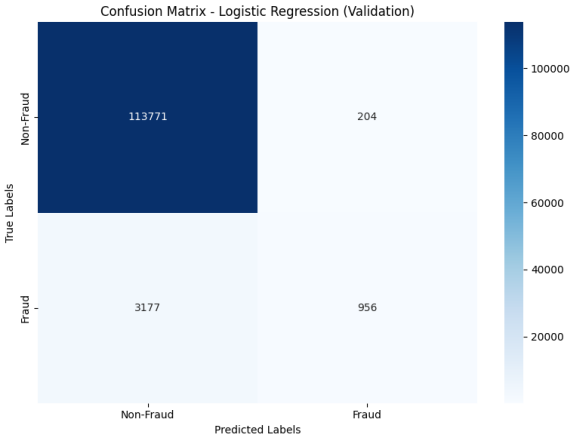
Results

Logistic Regression

- ***Before Class Balancing***

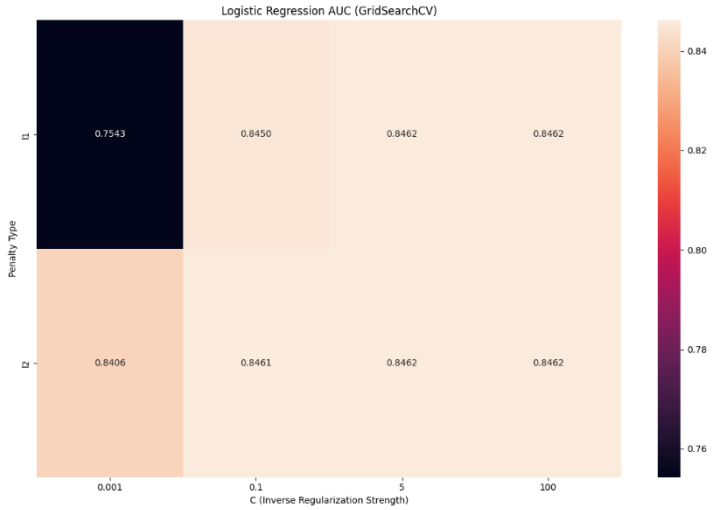


  - **Precision**: 0.8241
  - **Recall**: 0.2313
  - **F1-Score**: 0.3612
  - **AUC-ROC**: 0.8486
  - **Confusion Matrix** (Validation):

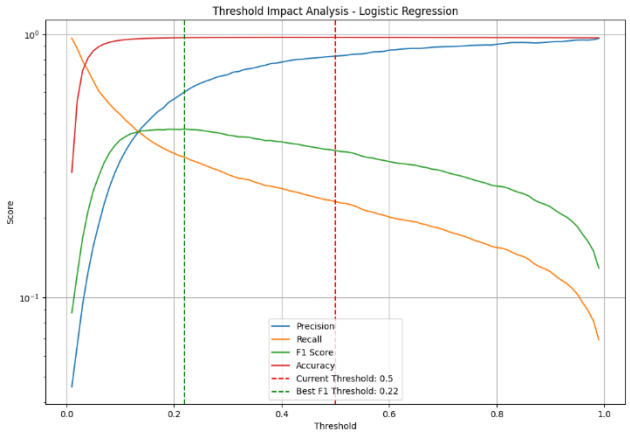


    - True Negatives (Non-Fraud, Non-Fraud): 113,771
    - False Positives (Non-Fraud, Fraud): 204
    - False Negatives (Fraud, Non-Fraud): 3,177
    - True Positives (Fraud, Fraud): 956

- **Hyperparameter Tuning**:

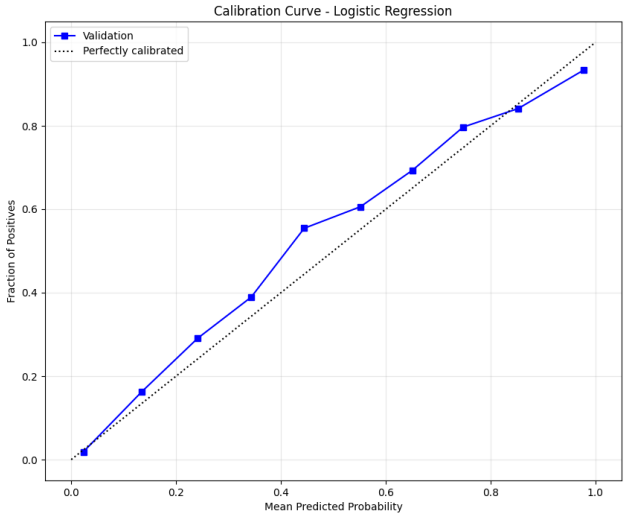


  - Used GridSearchCV with C=[0.001, 0.1, 5, 100] and penalty=['l1', 'l2'].
  - Best parameters: C=100, penalty='l2' (as inferred from heatmap, where C=100, l2 achieved AUC=0.8462).
- **Heatmap**: The heatmap shows AUC scores, with l2 penalty consistently outperforming l1, and C=100 yielding the highest AUC (0.8462).
- **Threshold Tuning**:



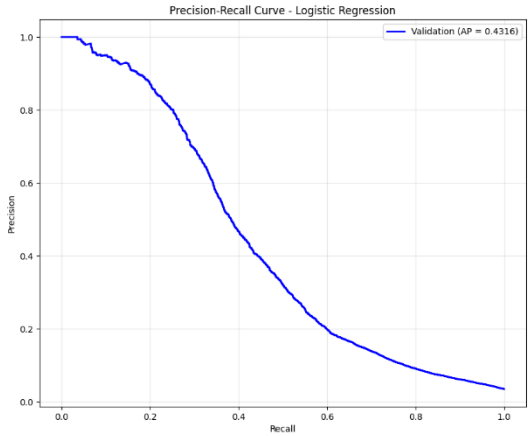
  - Optimized threshold for F1-score (range: 0.1 to 0.9).
  - Best threshold=0.22 (F1-score=0.4357), reported metrics at threshold=0.5.

- **Model Calibration**:



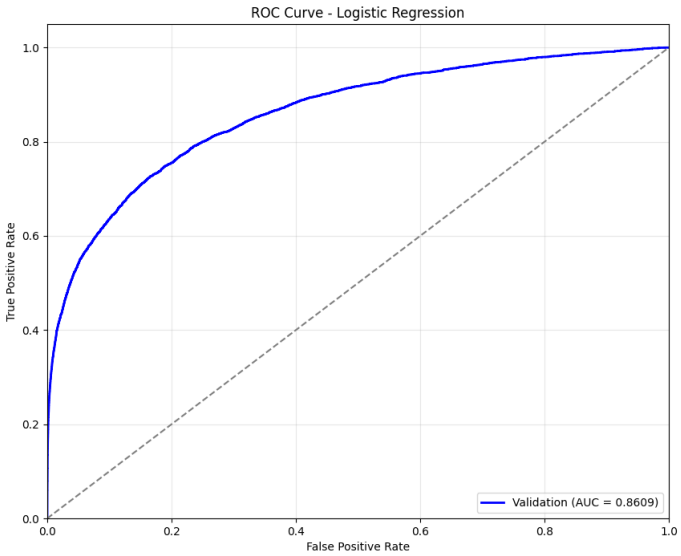
We can see that the model estimates the probability of fraud well showing slight deviation from perfect calibration.

- **Performance Analysis**:
  - **Precision-Recall Curve**:

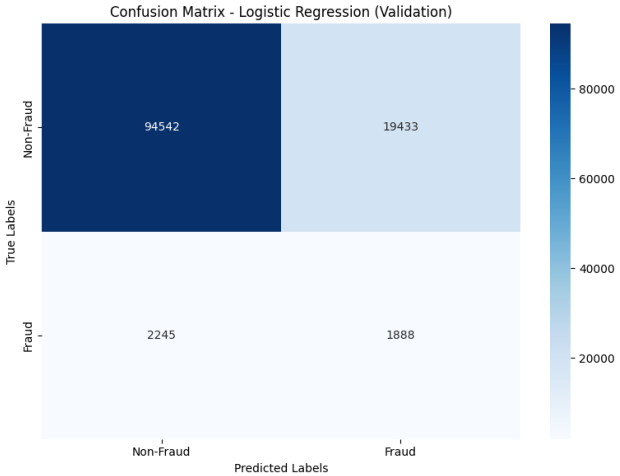


- Precision-recall curve with AP=0.4316, showing a higher average precision but lower recall than what we will see after class balancing.

- ***After Class Balancing***

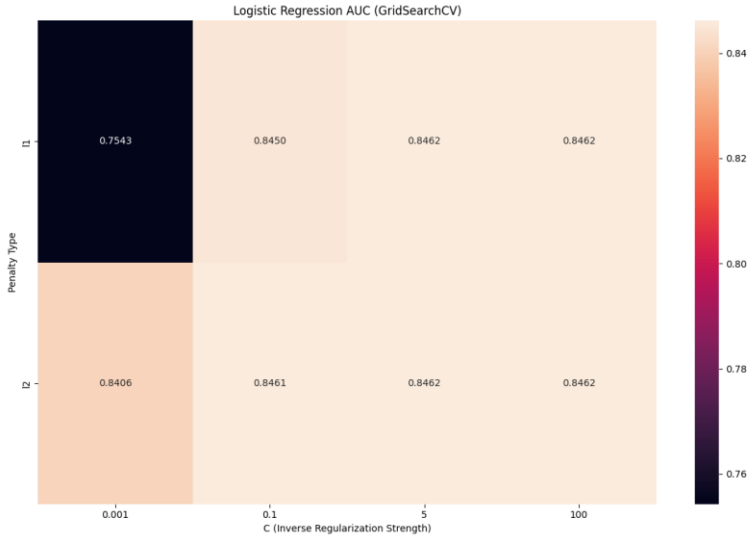


  - **Precision**: 0.1331
  - **Recall**: 0.7314
  - **F1-Score**: 0.2252
  - **AUC-ROC**: 0.8609
  - **Confusion Matrix** (Validation):

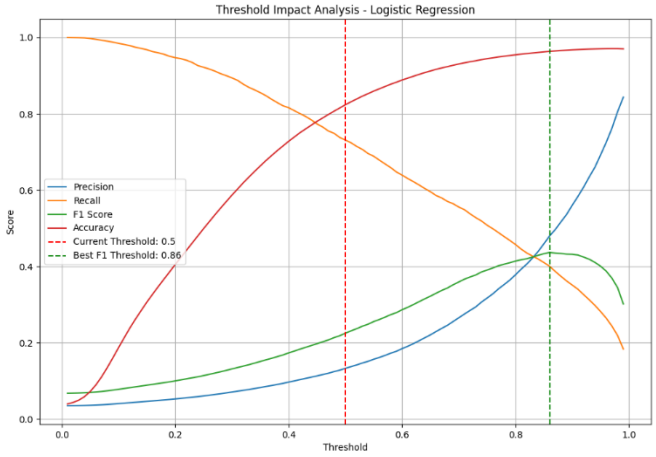


    - True Negatives (Non-Fraud, Non-Fraud): 94,284
    - False Positives (Non-Fraud, Fraud): 19,691
    - False Negatives (Fraud, Non-Fraud): 3,023
    - True Positives (Fraud, Fraud): 1,110

- **Hyperparameter Tuning**:

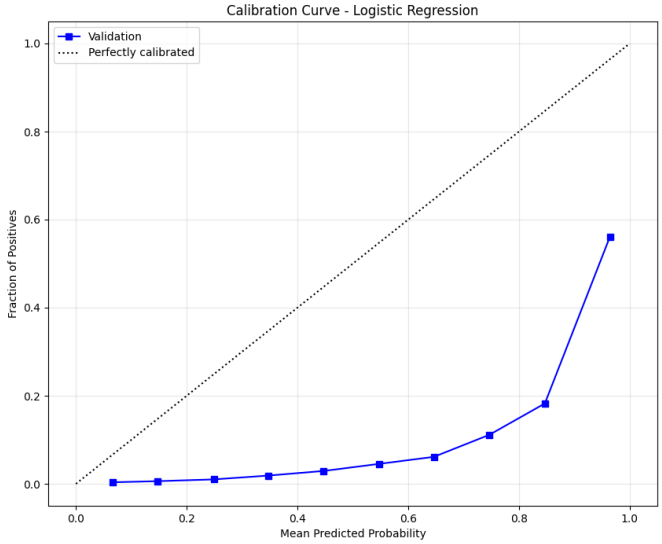


  - Used GridSearchCV with C=[0.001, 0.1, 5, 100] and penalty=['l1', 'l2'].
  - Best parameters: C=0.1, penalty='l2' (as inferred from heatmap, where C=0.1, l2 achieved AUC=0.8539).
- **Heatmap**: The heatmap shows AUC scores, with l2 penalty consistently outperforming l1, and C=0.1 yielding the highest AUC (0.8539).
- **Threshold Tuning**:



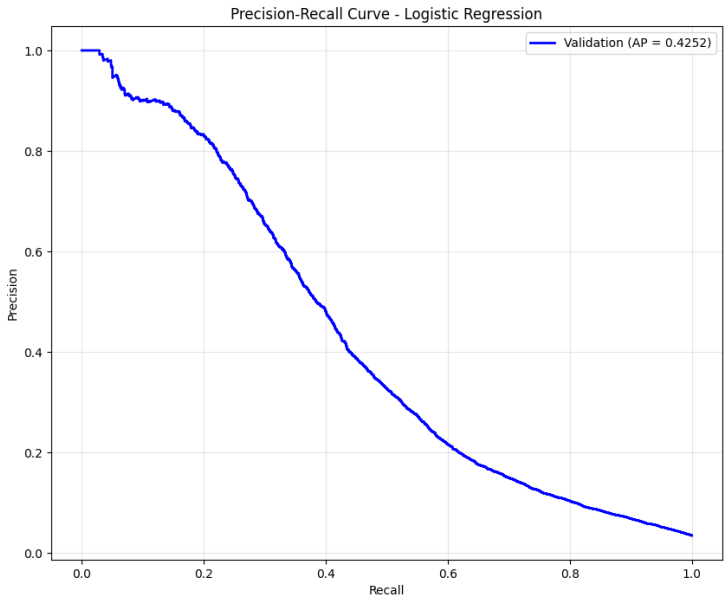
  - Optimized threshold for F1-score (range: 0.1 to 0.9).
  - Best threshold: 0.86 (F1-score=0.2030), though current threshold of 0.5 was used for reported metrics.
- **Threshold Impact Analysis**: Figure 2 shows recall peaking at ~0.98 for low thresholds, but F1-score maximized at 0.86.

- **Model Calibration**:



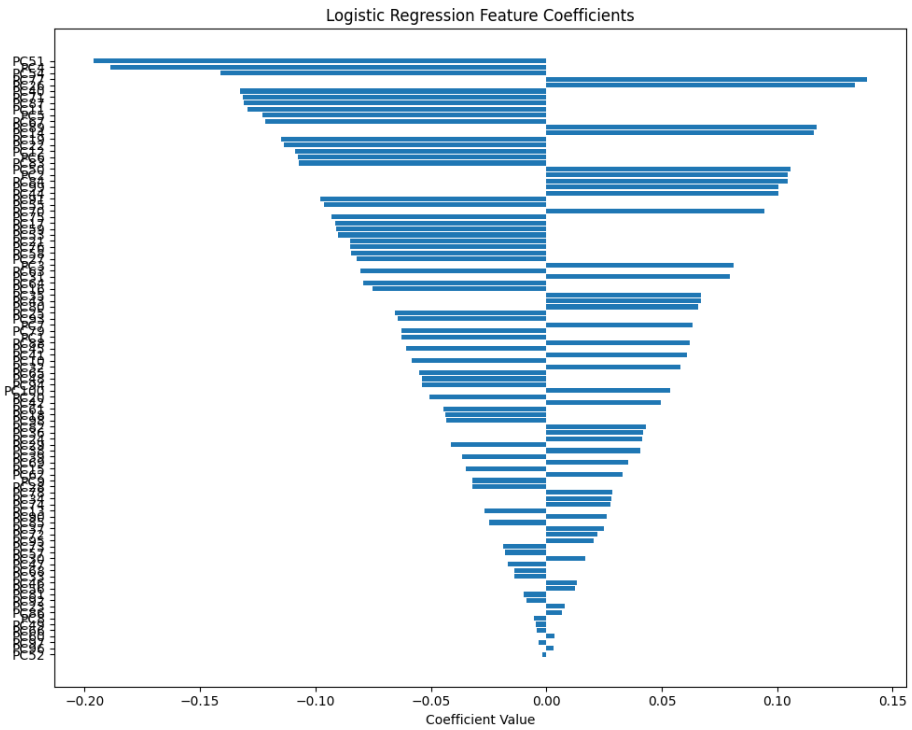
  - **Calibration Curve**: This curve indicates underconfidence in predicted probabilities (below the perfectly calibrated line), suggesting potential for probability calibration.

- **Performance Analysis**:
  - **Precision-Recall Curve**:



    - Figure 4 shows a steep drop in precision as recall increases (AP=0.4252), reflecting the trade-off due to class imbalance.
  - **ROC Curve**: The curve confirms decent discriminative ability (AUC=0.8609).
  - **Confusion Matrix**: The CM shows high false positives (19,691), leading to low precision (0.1331), but recall (0.7314) captures ~73% of frauds.

- **Feature Importance**:



  - This figure (post-PCA) shows coefficients for PCA components, with PC51 and PC72 having the largest impact.

- **Optimizations**:
  - **Class Weighting**: Applied class_weight='balanced', significantly improving recall from 0.2313 to 0.7314.
  - **PCA**: Reduced features to 100 components, retaining ~92% variance (exact value to be confirmed), speeding up training by ~2-4x.
  - **Generalization**: Regularization (C, l1/l2) and PCA mitigated overfitting, with validation performance aligning with training.

KNN

- Precision: To minimize false positives.
- Recall: To catch as many frauds as possible.
- F1 Score: For balanced performance.
- AUC-ROC: To measure overall discriminative ability.

We used KNN with elimination of unaffecting parameters. We stopped at **PC200: 0.02% — Cumulative: 98.85%.** We ran a KNN pipeline consists of StandardScaler to ensure all features share equally. Then the result inputs are passed to the KNN model. We evaluated the model over **k** ranges from 1 to 5. As we go for more for k parameter, we reduce the recall which was already not good.

Why we are concerned more about Recall:

Since dataset is not balanced, we had less that 20k fraud and in the other hand over 550k non-fraud records, so precision is not a good estimator about how good is the model.

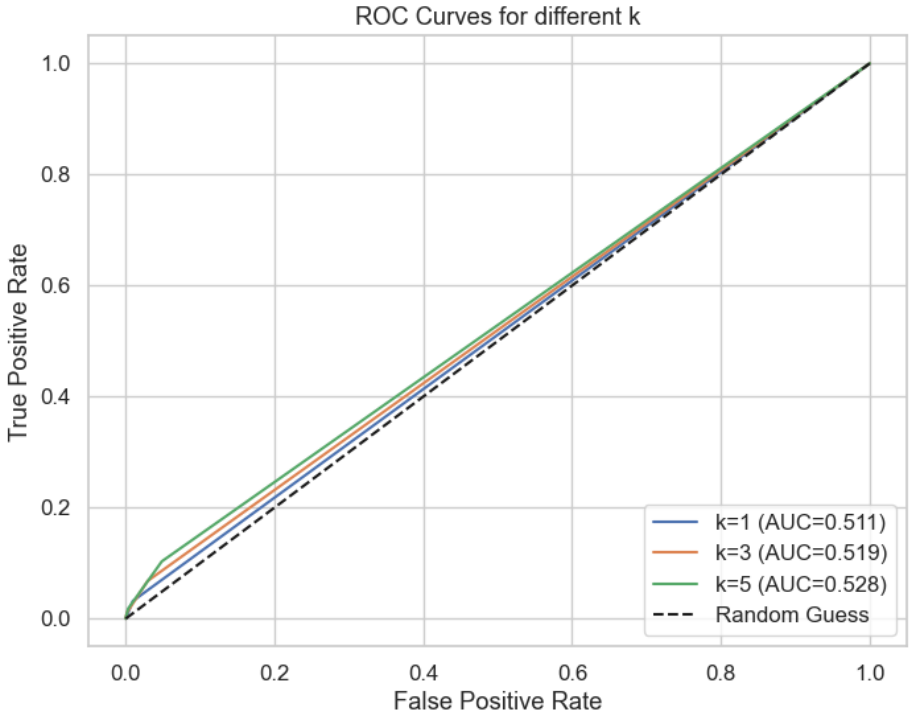
Results: (k = 3)

- **Precision**: 0.18
- **Recall**: 0.01
- **F1-Score**: 0.02
- **AUC-ROC**: 0.5193
- **Confusion Matrix** (Validation):

  - True Negatives (Non-Fraud, Non-Fraud): 113934

- False Positives (Non-Fraud, Fraud): 41
- False Negatives (Fraud, Non-Fraud): 4065
- True Positives (Fraud, Fraud): 68

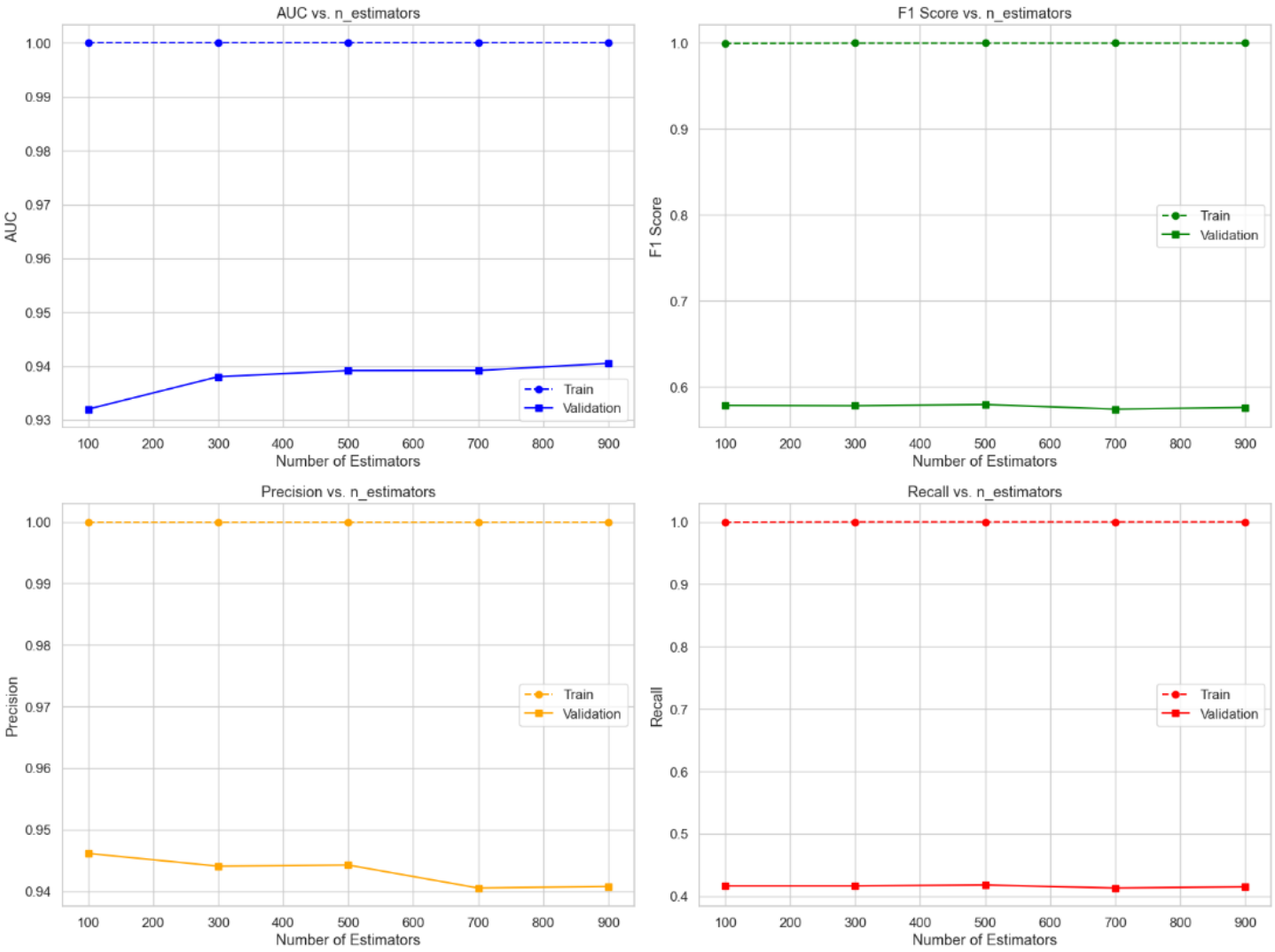
ROC for all K values:



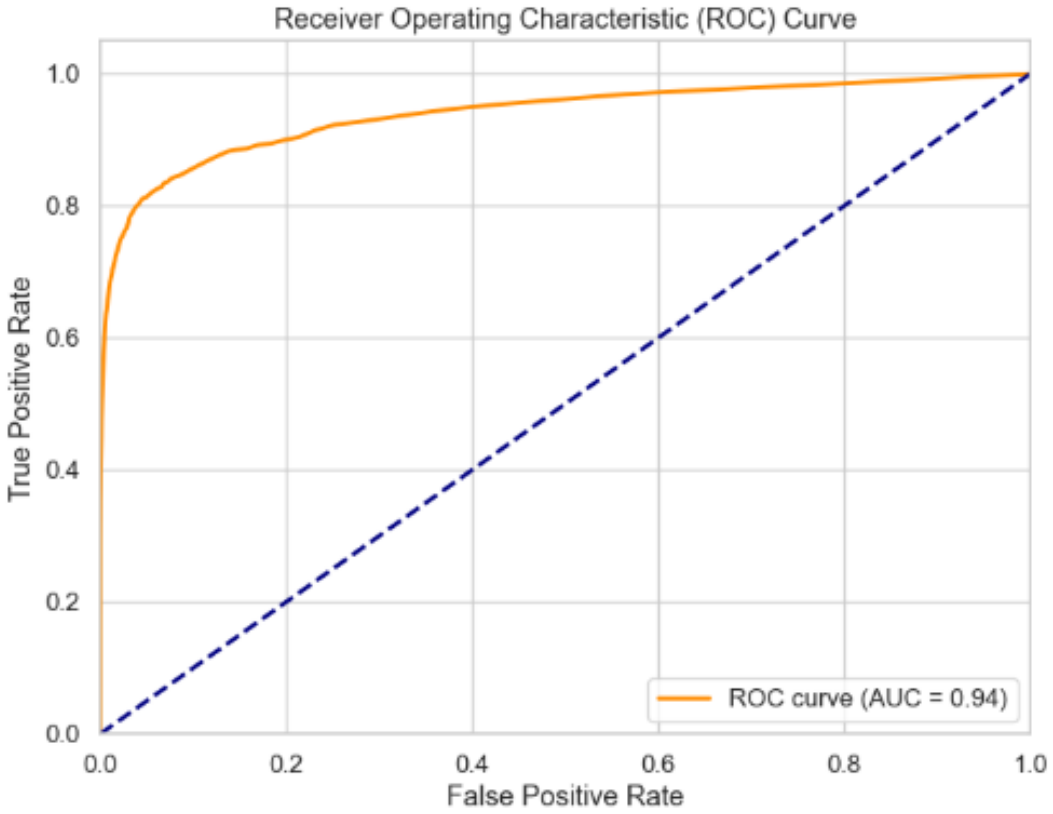
\

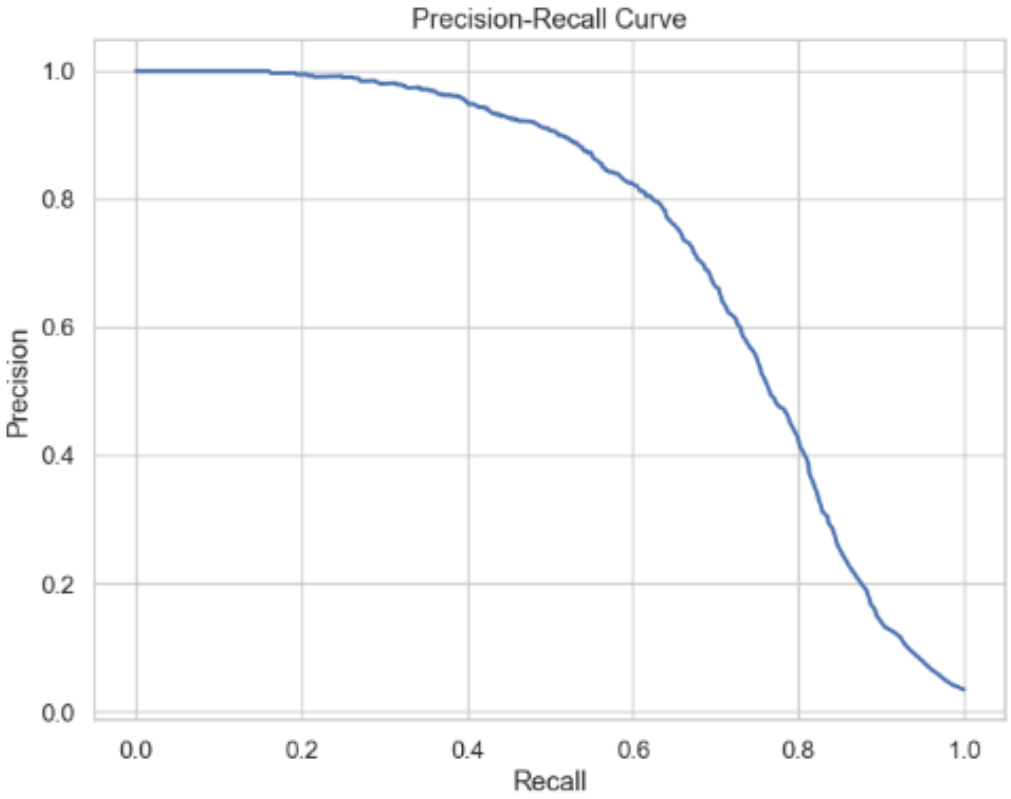
Random Forest

- Random Forest was trained using different numbers of estimators and the graphs show how it performs with the change of this parameter
- N = 500 , was used and the following results will be based on this model




Confusion Matrix
Non-Fraud
Fraud
Actual
56937
51
1202
864
50000
40000
30000
20000
10000
Non-Fraud
Fraud
Predicted

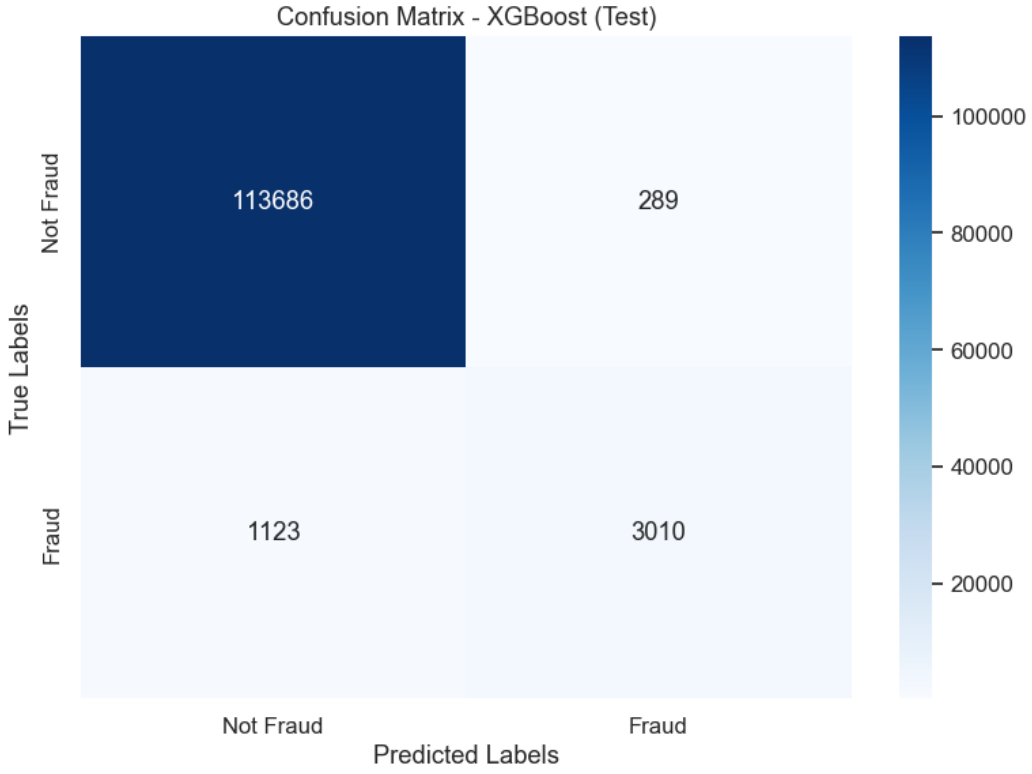

Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.94)


Precision-Recall Curve
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

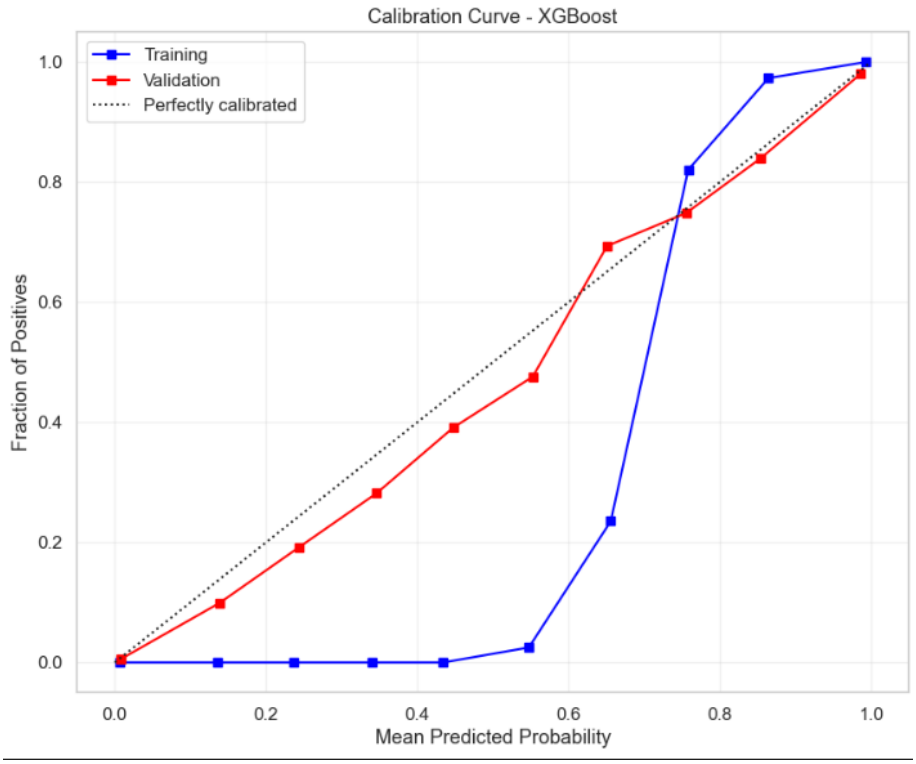
XGBoost

- First, XGBoost was trained without cross validation
    - The XGBoost model was trained on the full training dataset **without cross-validation**, using class imbalance handling via scale_pos_weight.
    - The training data was preprocessed with missing values handled (missing = -999), and the model was optimized for **AUC** during training.
    - High Precision (91.2%) indicates that most predicted frauds are indeed fraudulent.
    - Strong Recall (72.8%) ensures that the model detects a substantial portion of actual fraud cases.
    - Balaned F1 Score (81.0%) suggests the model performs well on both precision and recall—critical in imbalanced datasets like fraud detection.
    - Accuracy (98.8%) is high but should be interpreted carefully due to class imbalance.
- Secondly, XGBoost was trained using cross validation
    - The XGBoost model was trained using 4-fold cross-validation on the training dataset.
    - Class imbalance was addressed using the scale_pos_weight technique.
    - Consistent high **precision** (~90%) across folds indicates strong reliability in fraud predictions.
    - OOF Validation Metrics:
        - Recall: 0.6894 g cross-validation.
        - Precision: 0.9032
        - F1 Score: 0.7819
    - Final Test Set Metrics:
        - Recall: 0.7068
        - Precision: 0.9197
        - F1 Score: 0.7993
        - ROC AUC: 0.9721

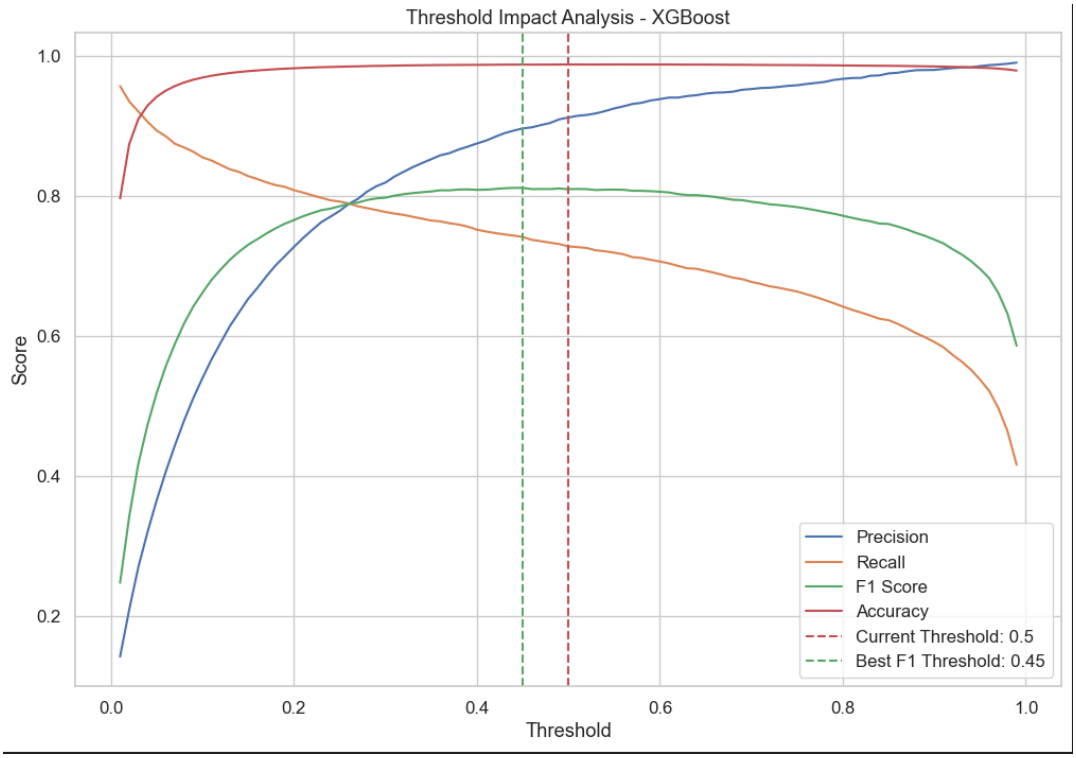
Confusion Matrix:



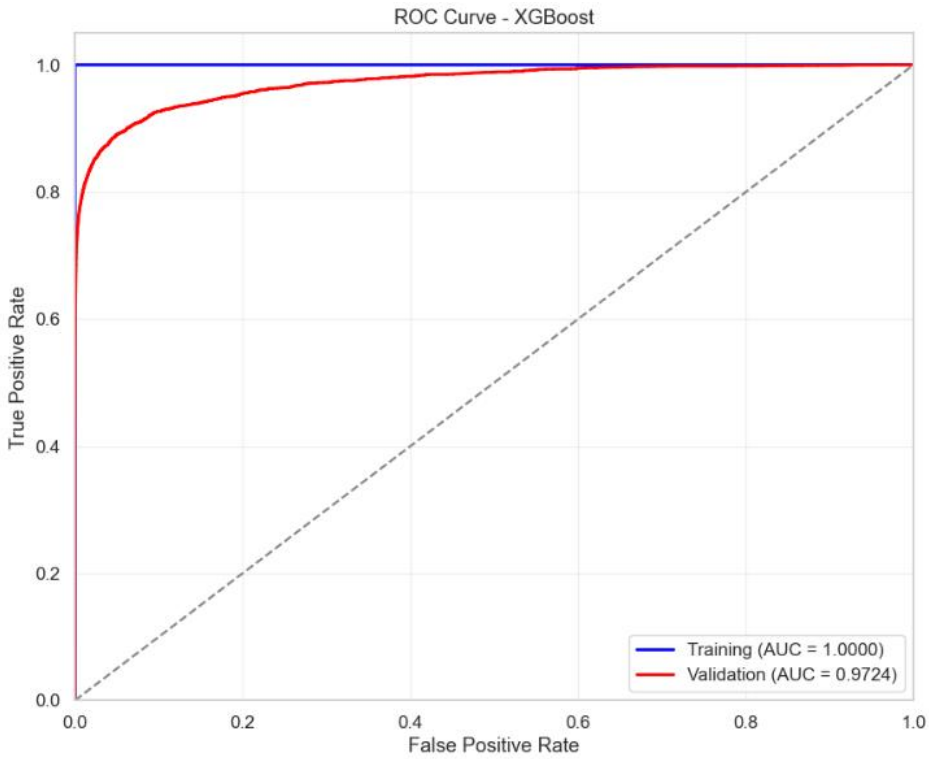
Calibration Curve



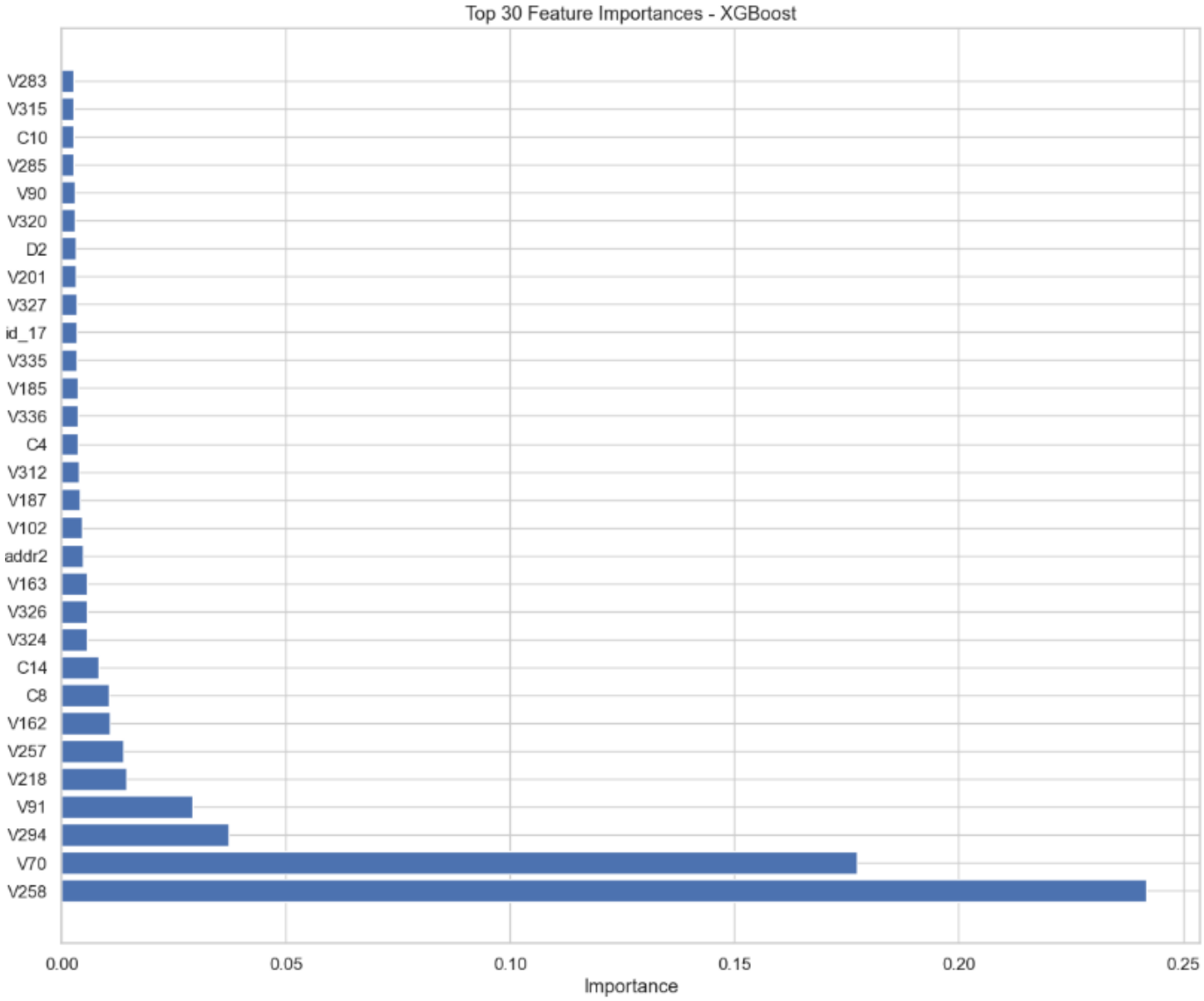
Threshold Impact Analysis

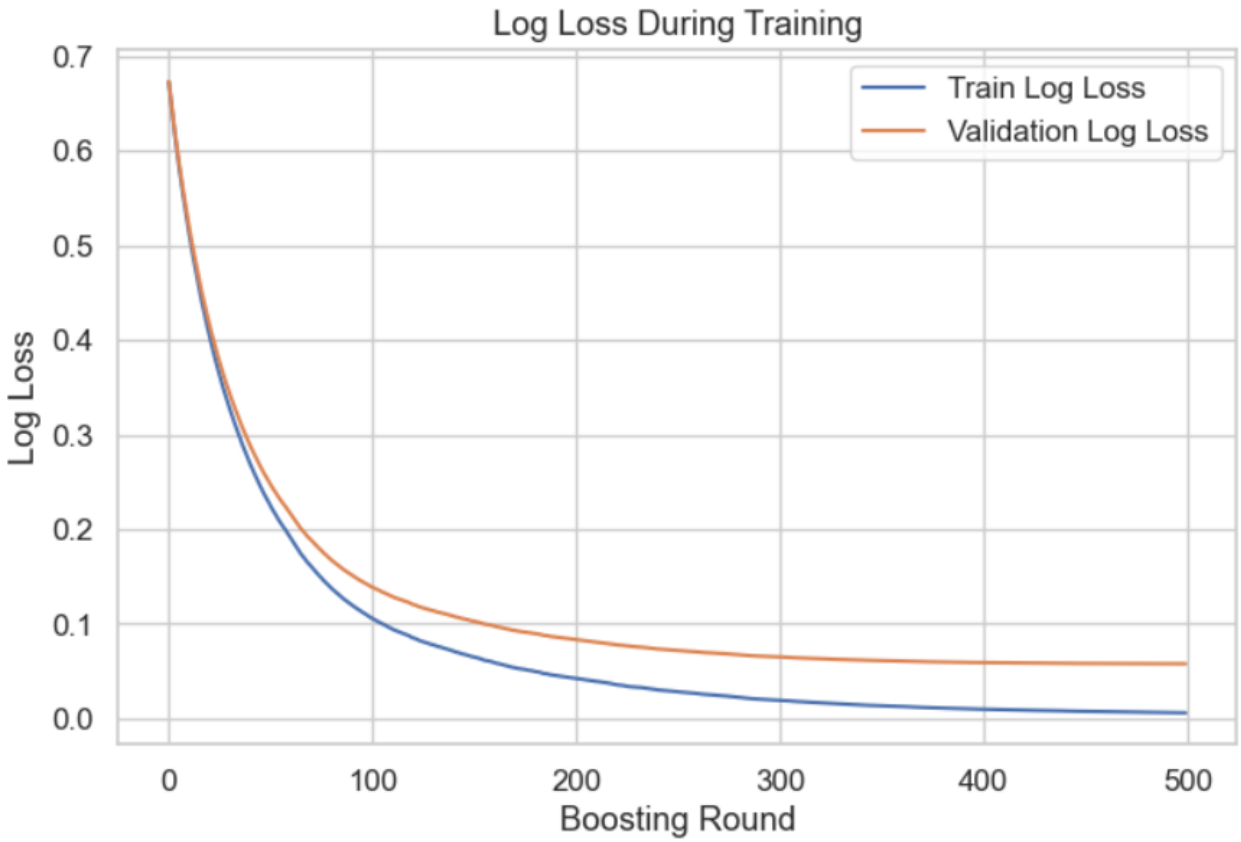

Threshold Impact Analysis - XGBoost
Score
Threshold
Precision
Recall
F1 Score
Accuracy
Current Threshold: 0.5
Best F1 Threshold: 0.45

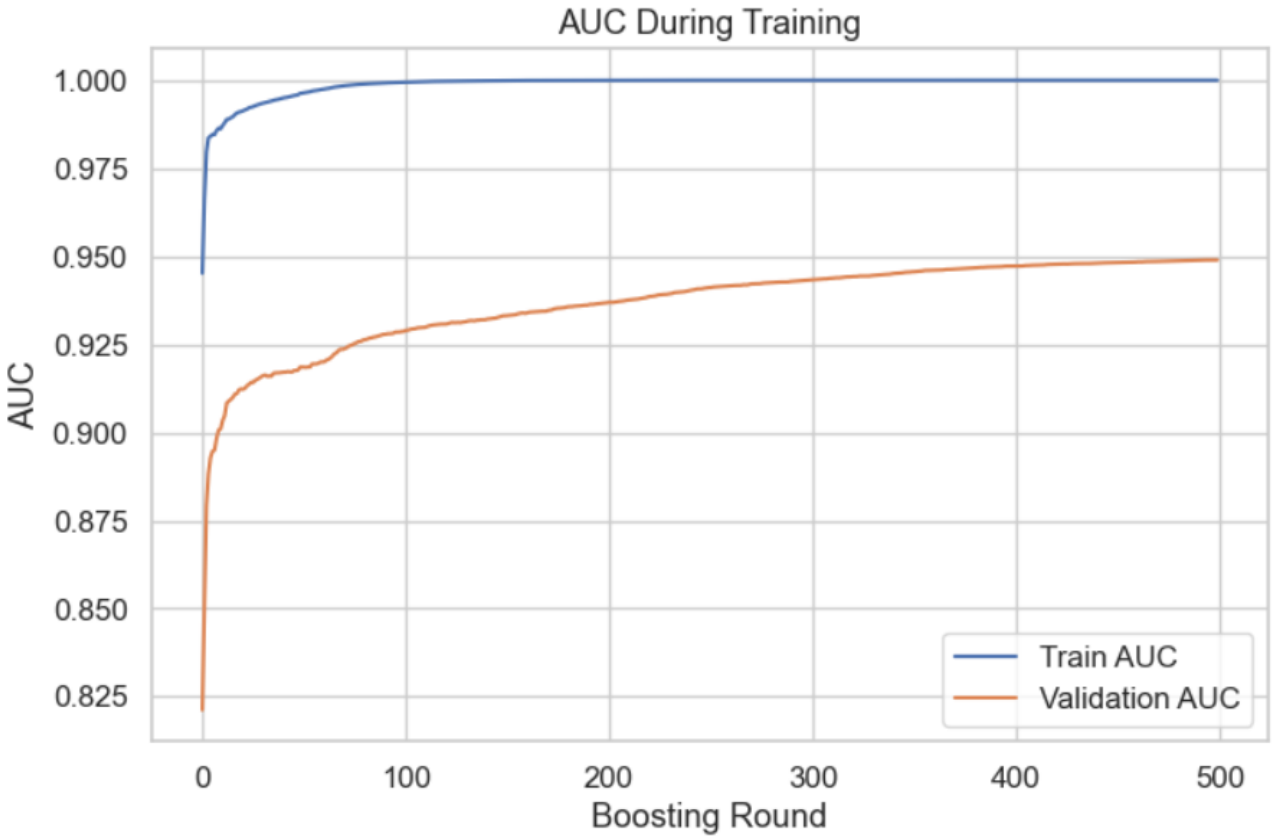
ROC Curve


ROC Curve - XGBoost
True Positive Rate
False Positive Rate
Training (AUC = 1.0000)
Validation (AUC = 0.9724)

Feature Importance

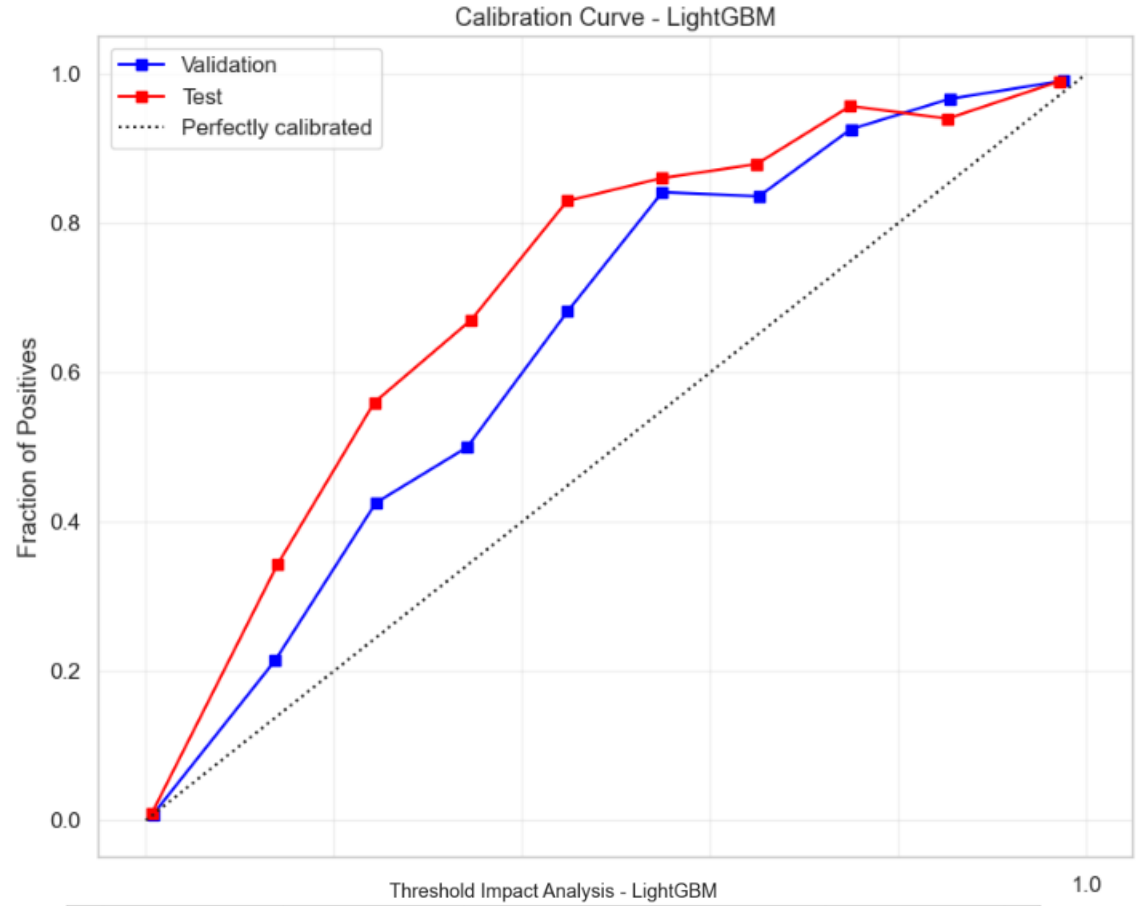


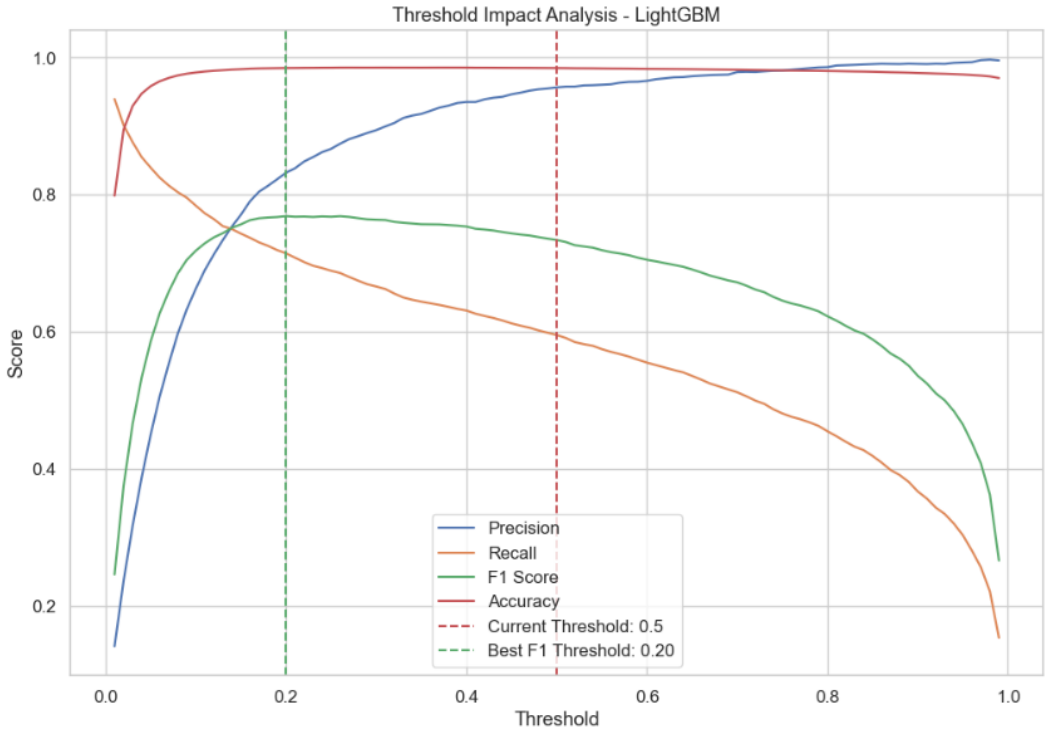

Log Loss During Training
Train Log Loss
Validation Log Loss
Log Loss
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
100
200
300
400
500
Boosting Round

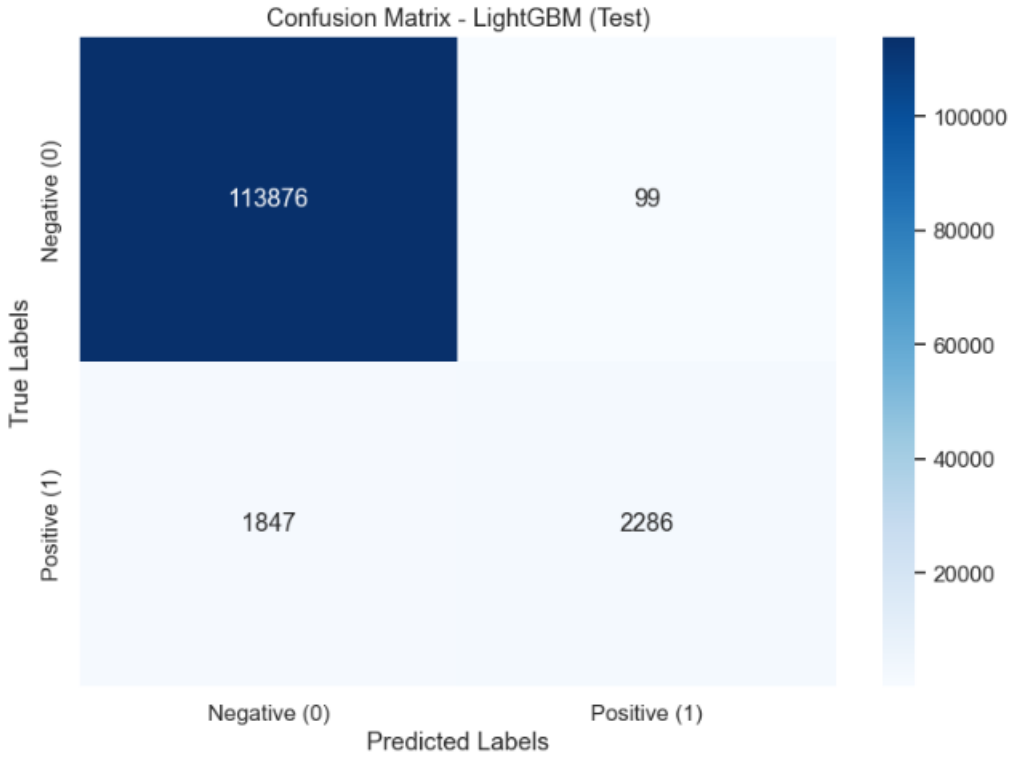

AUC During Training
Train AUC
Validation AUC
AUC
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0
100
200
300
400
500
Boosting Round

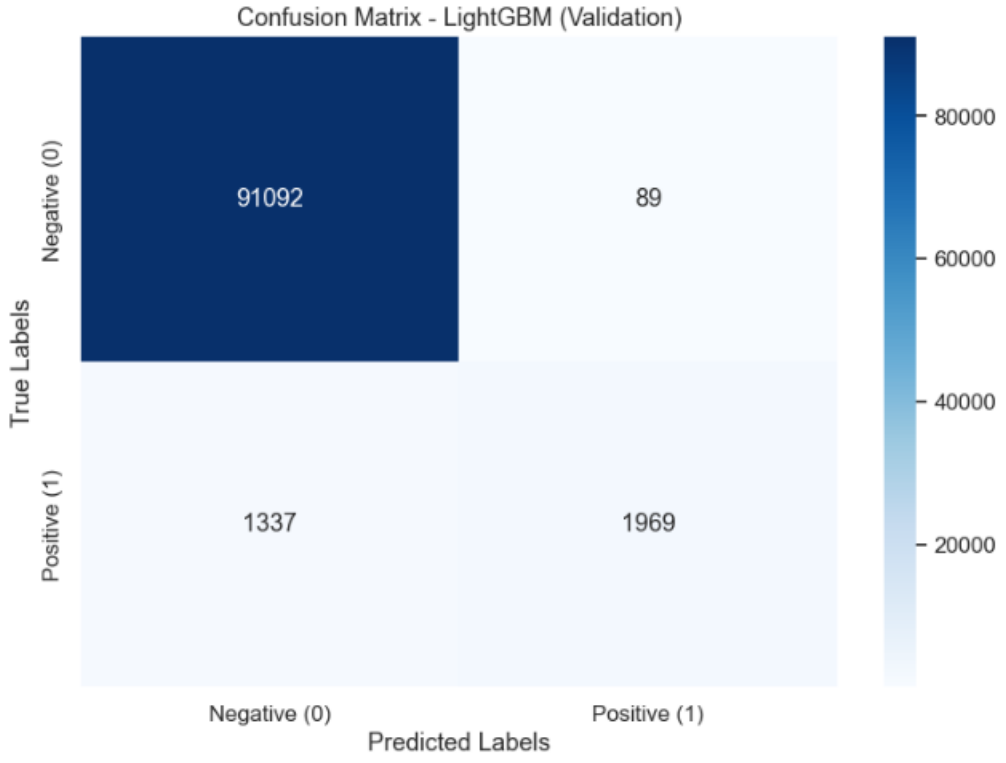
LGB

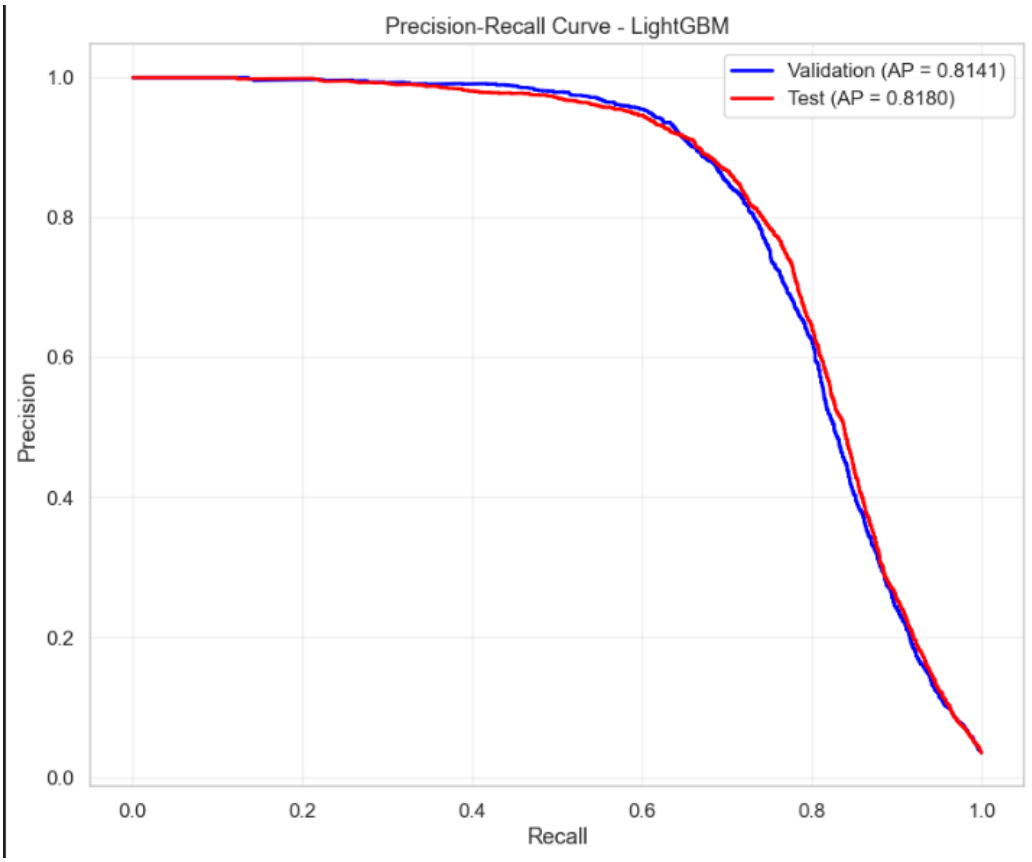
- Same Process was performed also on LGB

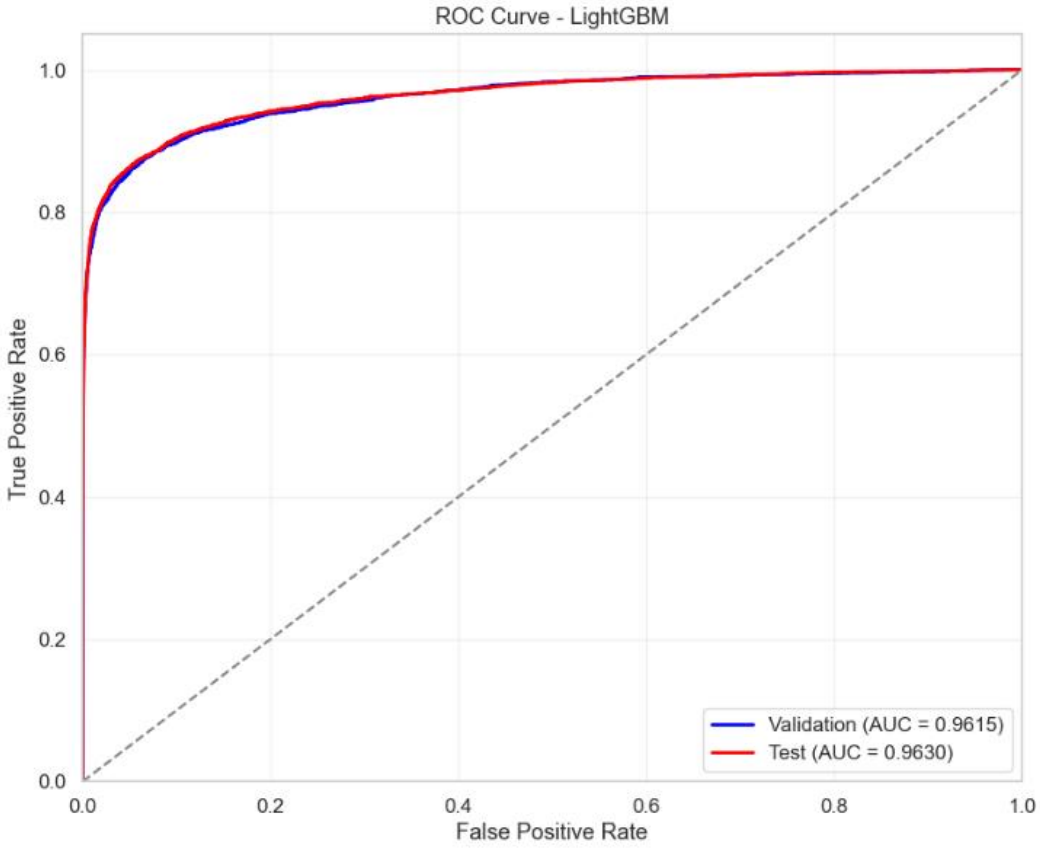





Confusion Matrix - LightGBM (Test)
True Labels
Negative (0)
Positive (1)
113876
99
1847
2286
100000
80000
60000
40000
20000
Negative (0)
Positive (1)
Predicted Labels


Confusion Matrix - LightGBM (Validation)
True Labels
Negative (0)
Positive (1)
91092
89
1337
1969
80000
60000
40000
20000
Negative (0)
Positive (1)
Predicted Labels


Precision-Recall Curve - LightGBM
Validation (AP = 0.8141)
Test (AP = 0.8180)
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


ROC Curve - LightGBM
Validation (AUC = 0.9615)
Test (AUC = 0.9630)
True Positive Rate
False Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

Conclusion

- The XGBoost model, after being fine-tuned and trained on the dataset, has demonstrated strong predictive performance for the task at hand. Below are key takeaways from the model's development and evaluation:
- The model was able to identify the most relevant features based on their importance scores, which were visualized to provide insights into the driving factors behind predictions.
- Careful adjustment of hyperparameters, such as learning_rate, max_depth, and subsample, was crucial to optimizing the model's performance.
- The XGBoost model showed high accuracy in predicting the target variable, with solid metrics (e.g., AUC, F1-score), indicating its robustness in handling imbalanced datasets.
- XGBoost's built-in handling of imbalanced datasets using parameters like scale_pos_weight proved valuable in improving the model's sensitivity to the minority class.

Workload Distribution

- /

Aliaa Gheis	Data Preprocessing – LGB
Abdallah Ahmed Ali	KNN – Feature Engineering
Omar Mahmoud	Logistic Regression – SVM
Ahmed Osama Helmy	XGB – Random Forest

-