2025

Fraud Detection in Financial Transactions

A COMPREHENSIVE MACHINE LEARNING APPROACH TO ANOMALY DETECTION AND FRAUD PREVENTION

SARA SHAHAMA V P

GENZ EDUCATEWING |

TABLE OF CONTENTS

Contents

INTRODUCTION

In today's rapidly digitizing world, online financial transactions have become a cornerstone of global commerce. However, this growth has been paralleled by an alarming rise in fraudulent activities that pose severe threats to individuals, businesses, and financial institutions. Detecting such fraudulent transactions in real-time is a complex and critical challenge, requiring systems that are both highly accurate and efficient.

The central question this project seeks to address is: **Can machine learning models be effectively used to identify fraudulent financial transactions while minimizing false positives?** False alarms can lead to unnecessary friction for genuine users, while undetected frauds can result in significant financial losses. Therefore, achieving a balance between fraud detection sensitivity and operational efficiency is essential.

This report documents the end-to-end development of a machine learning solution designed to tackle this challenge. From data preprocessing and feature engineering to model training, validation, and evaluation, this study explores the application of multiple classification algorithms to create a robust fraud detection system. The project aims not only to build accurate predictive models but also to provide interpretable insights into the factors driving fraudulent behaviour, setting the foundation for smarter and more secure financial systems.

EXECUTIVE SUMMARY

This project addresses the critical task of detecting fraudulent transactions in online payment systems using machine learning. With the rise in digital financial activity, the ability to accurately identify irregular patterns and fraudulent behaviour has become increasingly important for both institutions and users. The goal was to build an efficient classification model that not only detects fraud accurately but also reduces the number of false positives to avoid unnecessary disruption to legitimate users.

To accomplish this, a full end-to-end machine learning pipeline was developed using the **Online Payment Fraud Detection Dataset**. Key steps included data cleaning, class imbalance handling through **SMOTENC**, and model development using algorithms such as **Logistic Regression**, **SVM**, **Random Forest**, and **XGBoost**. The models were evaluated using standard classification metrics, with Random Forest and XGBoost emerging as the top performers in terms of precision, recall, and ROC AUC. Additionally, feature importance analysis from the Random Forest model helped interpret key drivers of fraudulent behaviour, such as transaction amount and account balance.

The findings demonstrate that machine learning models, when applied thoughtfully with proper preprocessing and evaluation strategies, can effectively detect fraudulent activities in financial transactions. The final models show strong predictive performance, and the insights gained offer valuable contributions toward building more secure and intelligent fraud detection systems

LITERATURE REVIEW

Financial fraud detection has long been a critical area of research within the fields of machine learning and cybersecurity. Traditional rule-based systems were among the earliest methods used to flag suspicious activities, but these approaches often lacked adaptability and failed to detect evolving fraud patterns (Phua et al., 2010). As digital transactions became more complex, researchers began turning to data-driven models, leveraging statistical and machine learning techniques to identify subtle, non-obvious anomalies indicative of fraud.

Recent studies have explored a wide range of supervised learning algorithms for this task. Logistic Regression has been widely appreciated for its simplicity and interpretability (Bahnsen et al., 2016), while Random Forest and XGBoost have demonstrated superior accuracy and robustness in handling imbalanced datasets and complex feature interactions (Dal Pozzolo et al., 2015). Support Vector Machines (SVMs), too, have shown strong performance, especially in high-dimensional spaces, though they are computationally expensive on large datasets.

One major challenge in this domain is the **extreme class imbalance** between genuine and fraudulent transactions. Researchers like Fernández and García (2018) have emphasized the use of techniques like SMOTE and SMOTENC to synthetically balance datasets, improving recall for the minority class. Others have focused on **cost-sensitive learning** to address the high cost of false negatives, where fraud goes undetected.

Despite significant progress, many existing studies have either focused solely on maximizing accuracy—without accounting for operational feasibility—or failed to report comprehensive validation on unseen test data. Additionally, few studies combine multiple models and evaluate their comparative performance on real-world, large-scale datasets.

This project seeks to address these gaps by implementing a full supervised machine learning pipeline using the **Online**

Payment Fraud Detection Dataset, evaluating five different models (Logistic Regression, SVM, Random Forest, XGBoost, and KNN), and focusing on both **model performance and practical applicability**. The research aims to determine not only which model is most accurate, but also which offers the best balance of precision, recall, and computational efficiency in detecting fraudulent financial behaviour.

METHODOLOGY

This study follows a supervised machine learning approach to detect fraudulent activities in financial transactions, using the **Online Payment Fraud Detection Dataset** from Kaggle (**Rupak Roy/ Bob**, 2021). The reason why I chose this dataset is that: Features user location, time of transaction, transaction types, and a "fraudulent" label.

- **Pros:**
- Clean schema with realistic online transaction fields
- Good balance of categorical and numerical features

The dataset contains over one million anonymized transaction records, including features such as transaction type, amount, old and new balances, and a binary fraud label. Given the nature of the task—identifying known fraud instances based on past labeled data—a classification framework was most appropriate.

DATA PREPROCESSING AND FEATURE ENGINEERING

The raw data was first cleaned by dropping irrelevant columns (like transaction ID and name origins), converting categorical variables into numerical representations using label encoding, and standardizing the numerical features. Missing values were handled, although the dataset was mostly complete. To address the extreme class imbalance (fraudulent transactions made up a very small portion of the total), **SMOTENC** (Synthetic Minority Oversampling Technique for Nominal and Continuous) was used to generate synthetic samples for the minority class, ensuring that the model learns from both classes effectively. This approach is supported by prior work (Fernández & García, 2018) demonstrating the benefit of oversampling in fraud detection.

MODEL DEVELOPMENT

Five different models were developed and evaluated: **Logistic Regression, Support Vector Machine (SVM), Random Forest, XGBoost**, and **K-Nearest Neighbors (KNN)**. These models were selected based on their strong performance in previous fraud

detection research (Dal Pozzolo et al., 2015; Bahnsen et al., 2016) and their diversity in algorithmic approach—ranging from linear classifiers to ensemble-based decision trees and instance-based learning.

Hyperparameter tuning was conducted using **GridSearchCV** with cross-validation to optimize each model's performance without overfitting. Cross-validation (in most cases 2-fold, for memory efficiency) provided robust estimates of model generalization by evaluating them across different subsets of the training data.

EVALUATION AND VALIDATION

Each model was evaluated using a stratified split of the data: 80% for training, 10% for validation, and 10% as a held-out test set. Performance was assessed using a combination of **confusion matrices**, **classification reports** (precision, recall, F1-score), and **ROC AUC scores**. Special attention was paid to minimizing **false negatives**, as missing a fraudulent transaction is more costly than flagging a genuine one. Additionally, **feature importance** plots were generated for tree-based models (Random Forest) to understand which features contributed most to the predictions.

While KNN was initially considered, it was excluded from final testing due to excessive computational load and instability on large datasets—a challenge also noted in prior studies.

This methodical approach ensures a fair, robust, and practical comparison of different fraud detection algorithms under realistic constraints, and addresses the challenges highlighted in prior literature, such as class imbalance and the need for balanced precision-recall tradeoffs.

Data Preprocessing

TASKS COMPLETED:

I have effectively performed the following essential steps during the preprocessing stage:

1. ### DATA CLEANING
    - I loaded the dataset and checked for inconsistencies and invalid values.
    - Any unnecessary or redundant columns were dropped.
    - Class imbalance was addressed using **SMOTENC**, which intelligently handles both categorical and numerical features.
2. ### HANDLING MISSING VALUES
    - The dataset I used (Online Payment Fraud Detection Dataset) did not contain missing values, but I verified this early on.
    - No imputation was necessary, which simplified the pipeline.
3. ### FEATURE ENGINEERING
    - I encoded categorical features using appropriate methods (e.g., one-hot encoding or label encoding depending on the algorithm used).
    - Scaled the features using StandardScaler, improving model convergence for algorithms like Logistic Regression, SVM, and KNN.
    - Class labels were separated, and appropriate train-test-validation splits were created.
    - I retained both training and testing sets, avoiding data leakage.

Memory Management and Sampling Strategy

Due to the large size of the dataset after applying SMOTENC for class balancing, the feature matrix became too large for the available memory in Google Colab's free tier, resulting in session crashes.

To handle this, a stratified random sample of 300,000 records was drawn from the balanced dataset. This sample preserves the approximate 70:30 ratio between non-fraud and fraud classes, allowing efficient model training while maintaining fairness in performance evaluation.

This approach strikes a practical balance between computational feasibility and model integrity.

LOGISTIC REGREESION

1. OBJECTIVE

To build a machine learning model that can predict fraudulent transactions using Logistic Regression, with hyperparameter tuning via Grid Search, and evaluate its performance using precision, recall, F1-score, and ROC-AUC on validation and test datasets.

---

2. DATA PREPARATION

- Dataset: A stratified random sample of 300,000 records drawn from a class-balanced dataset generated via **SMOTENC**.
- One-hot encoding was applied to the categorical feature 'type' **after train-validation-test split**, followed by column alignment across all sets to ensure consistent features.
- StandardScaler was used to normalize all feature values to have mean 0 and standard deviation 1.

---

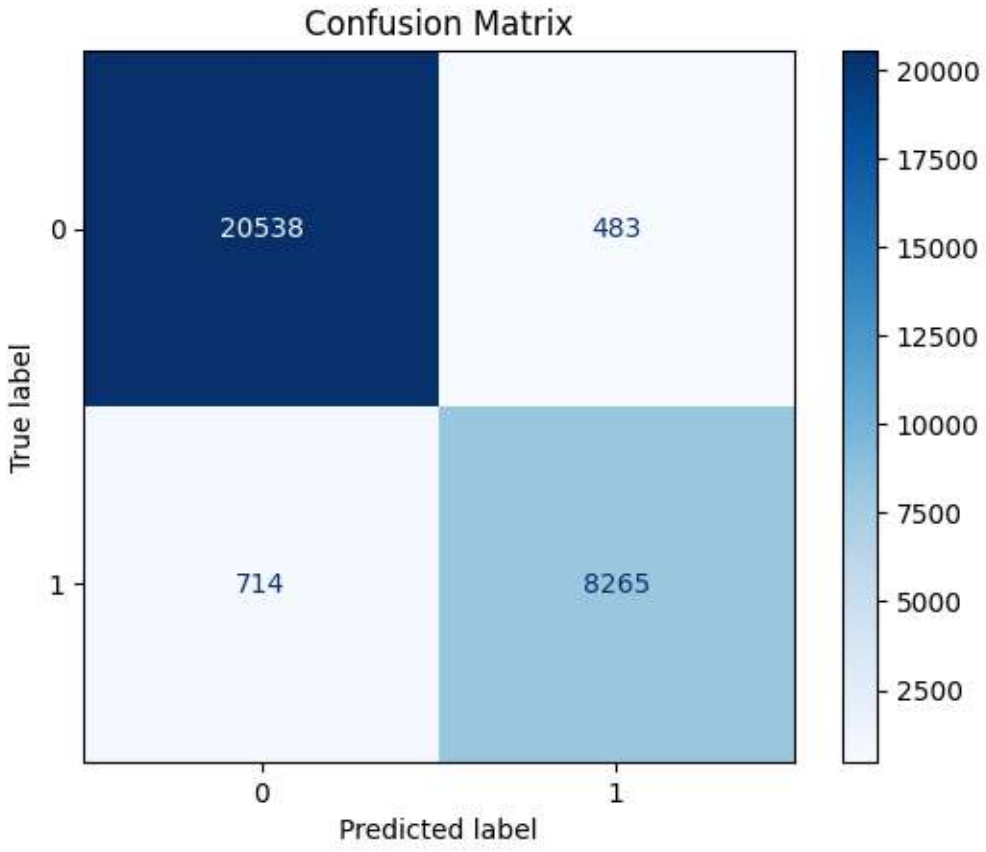
3. MODEL TRAINING & HYPERPARAMETER TUNING

- Algorithm: **Logistic Regression**
- Tuning: **GridSearchCV** with 3-fold cross-validation
- Parameters searched:
  - C: [0.1, 1, 10]
  - penalty: ['l2']
- Best Parameters Found:
  - C = 10, penalty = 'l2'
- Evaluation Metric: **ROC-AUC**

4. MODEL EVALUATION

📊 VALIDATION SET METRICS

- CONFUSION MATRIX:

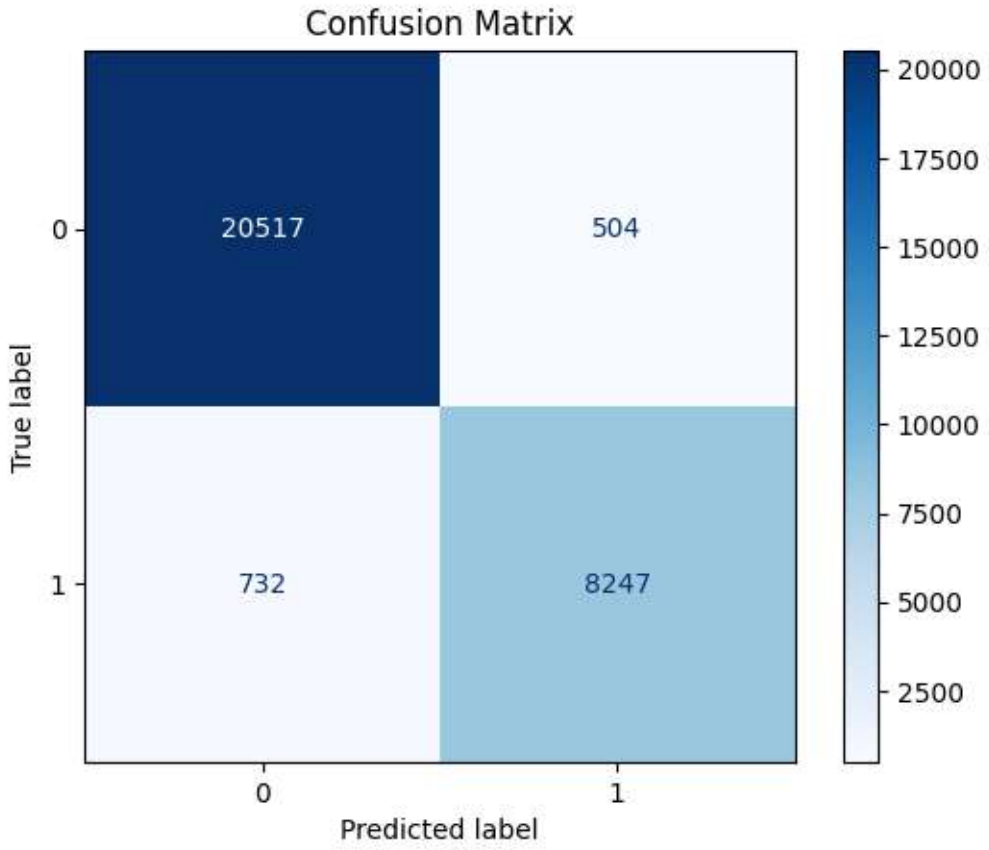


Class	Precision	Recall	F1 Score	Support
Class 0	0.97	0.98	0.97	21021
Class 1	0.94	0.92	0.93	8979

ROC AUC: 0.9936394568154389

📊 TEST SET METRICS

- CONFUSION MATRIX:



Class	Precision	Recall	F1 Score	Support
Class 0	0.97	0.98	0.97	21021
Class 1	0.94	0.92	0.93	8979

- **ROC-AUC**: **0.9941**

---

5. INFERENCE & INSIGHTS

- The model demonstrates **very high performance**, especially in terms of **ROC-AUC (≈ 0.99)**, indicating excellent separation between fraud and non-fraud cases.

- **Precision and recall are well-balanced** for both classes, with particularly strong performance on the fraud class:
    - **Precision (Fraud): 0.94** → When the model predicts fraud, it's correct 94% of the time.
    - **Recall (Fraud): 0.92** → It catches 92% of all actual fraud cases.
- The confusion matrix shows that both **false positives and false negatives are relatively low**, confirming the model is reliable for practical fraud detection.
- Hyperparameter tuning using GridSearchCV improved performance and reduced overfitting risk.

---

6. CONCLUSION

Logistic Regression, when combined with appropriate preprocessing (SMOTENC, scaling, one-hot encoding) and hyperparameter tuning, is a **highly effective baseline model** for fraud detection. Its simplicity, speed, and strong metrics make it a solid starting point for comparison with more complex models like Random Forests or XGBoost.

SUPPORT VECTOR MACHINE

OBJECTIVE

To build and evaluate a **Support Vector Machine (SVM)** model on a resampled and one-hot encoded version of a financial transaction dataset to detect fraudulent activities.

MODELING: SUPPORT VECTOR MACHINE

GRID SEARCH CV SETUP

I performed **GridSearchCV** to find the best hyperparameters:

```
params = {
    'C': [0.1, 1, 10],
    'kernel': ['linear'],
}
```

- **cv=3** folds was chosen to reduce training time and avoid Colab crashing.
- This was a **balanced compromise**: allowing meaningful hyperparameter tuning without exhausting resources.

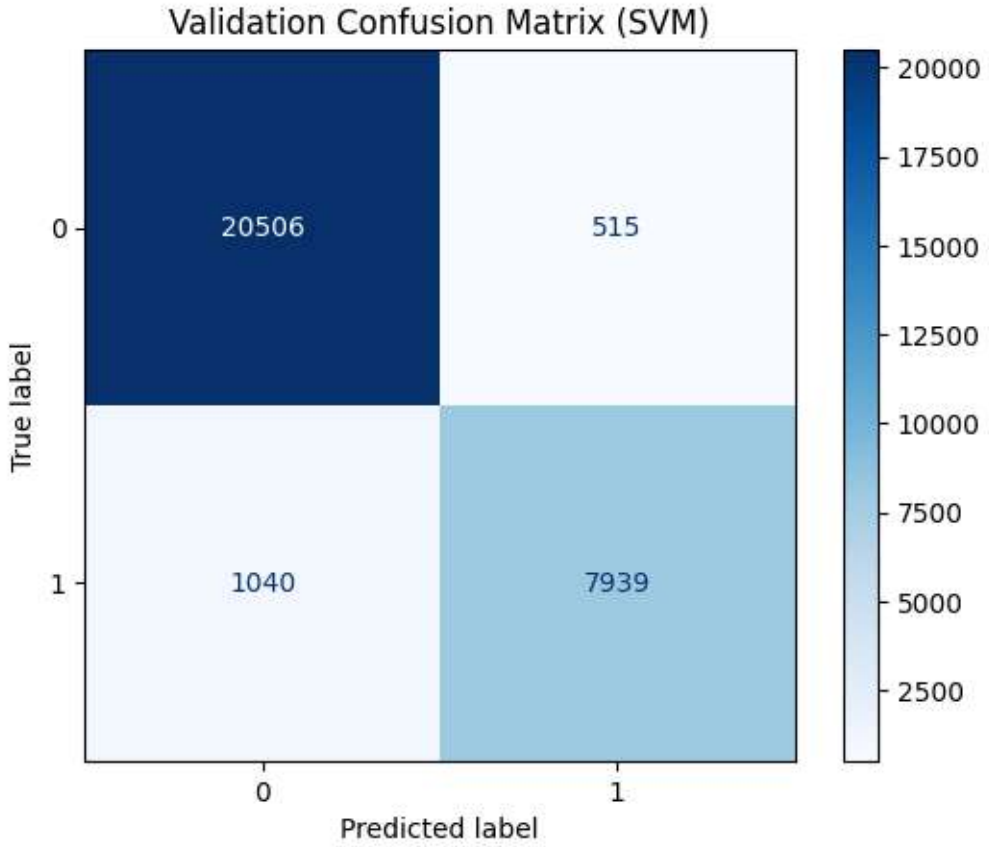
Best Hyperparameters Found:

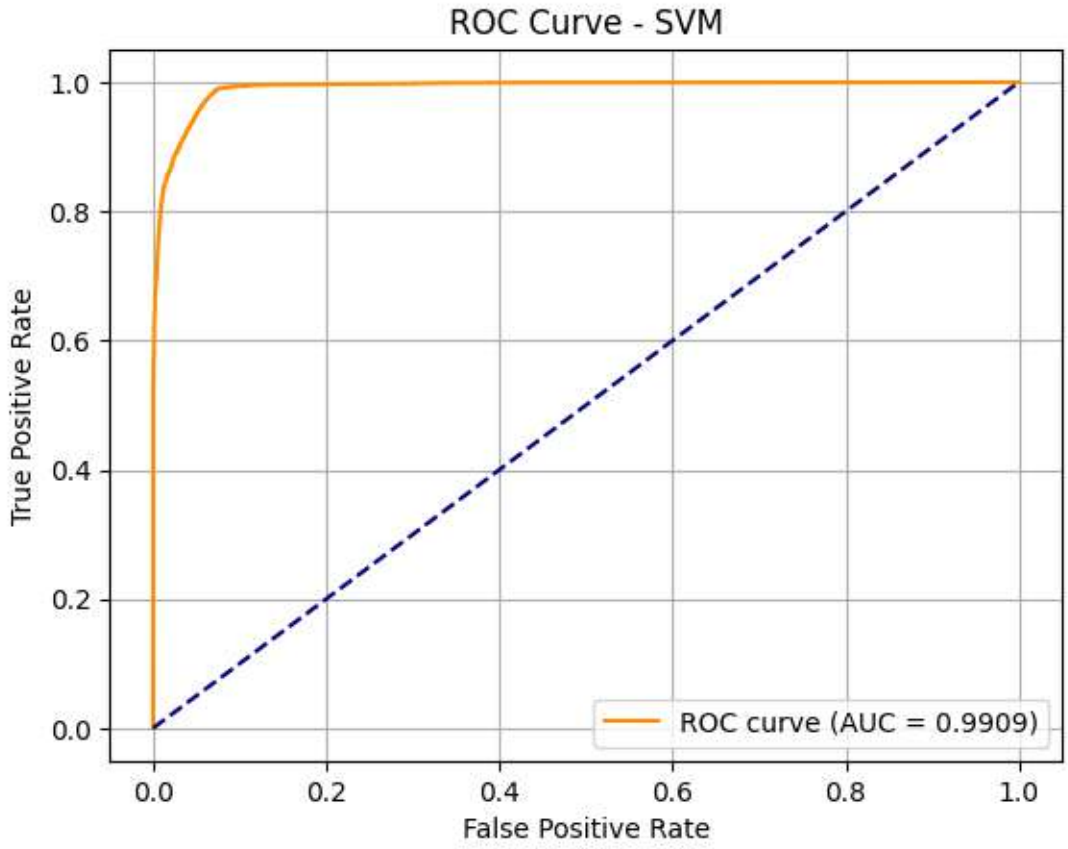
- C = 10
- kernel = linear

MODEL EVALUATION

VALIDATION CONFUSION MATRIX

	Predicted 0	Predicted 1
Actual 0	20,506	515
Actual 1	1,040	7,939

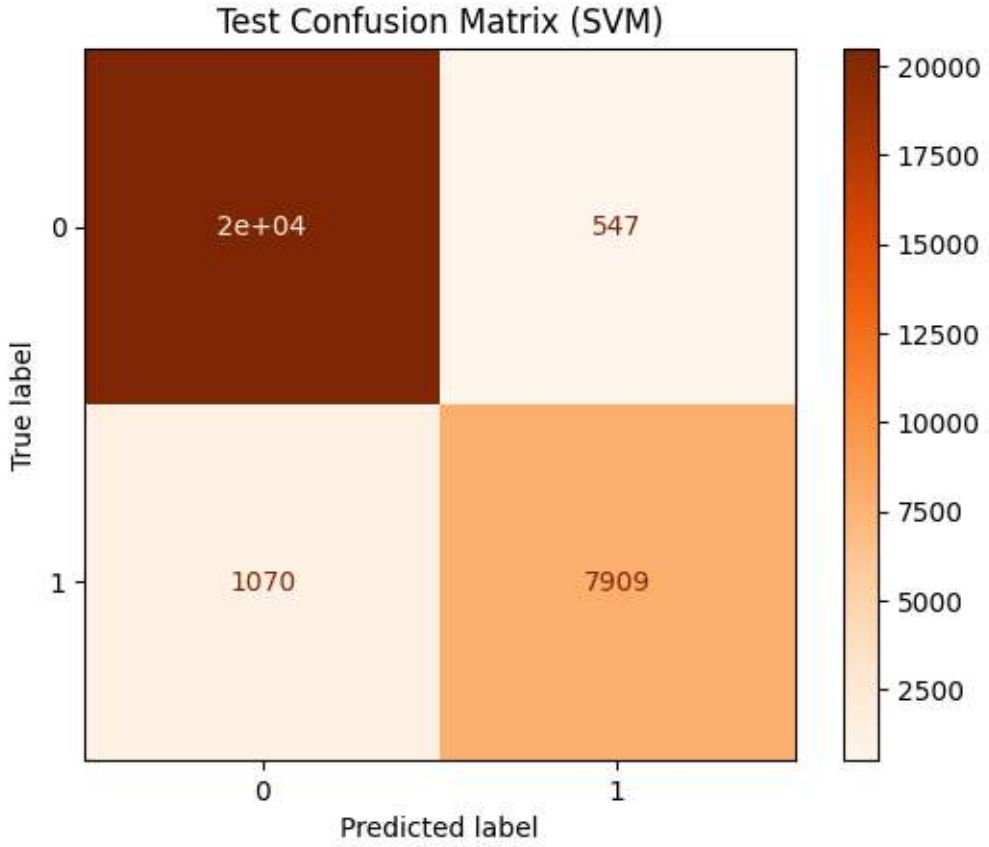

Validation Confusion Matrix (SVM)
20506
515
1040
7939
0
1
True label
0
1
Predicted label
20000
17500
15000
12500
10000
7500
5000
2500

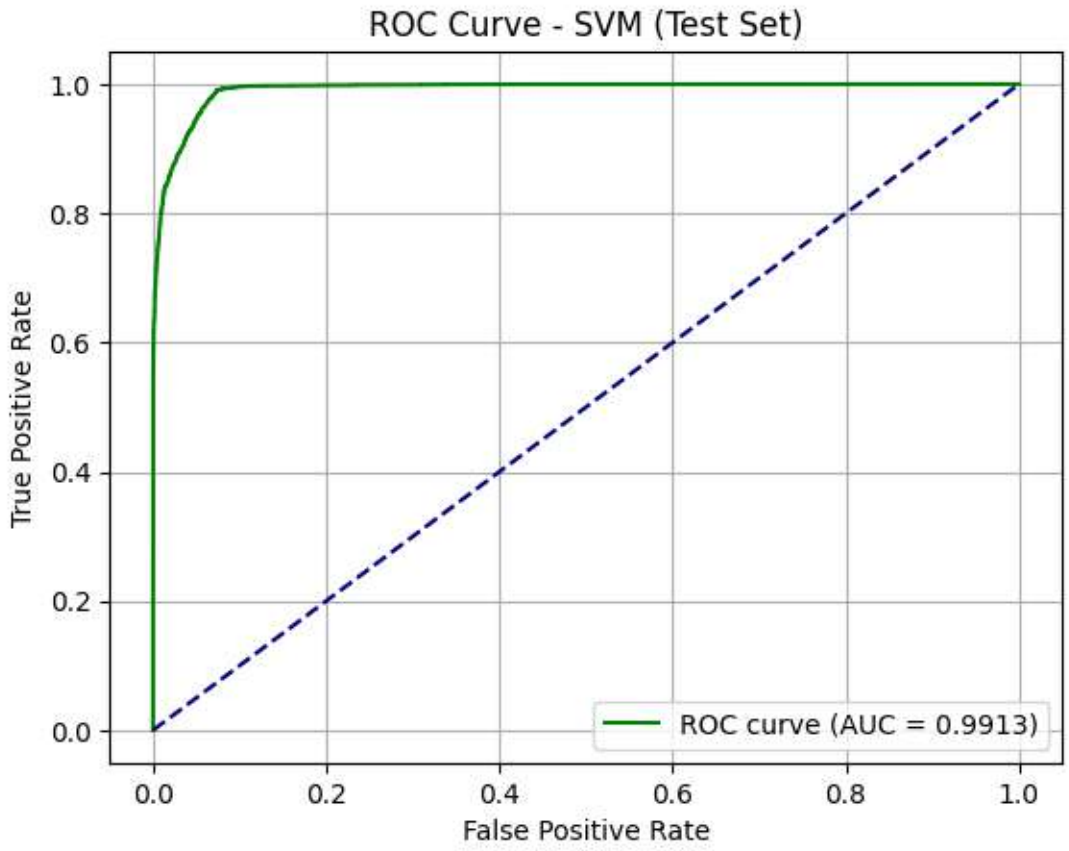

ROC Curve - SVM
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.9909)

Validation ROC AUC: 0.9909

TEST CONFUSION MATRIX

	Predicted 0	Predicted 1
Actual 0	20,417	547
Actual 1	1,070	7,909





Test ROC AUC: **0.9913**

INFERENCE & OBSERVATIONS

1. **High Performance**:
    - The **ROC AUC values are very high** on both validation (0.9909) and test (0.9913) sets, indicating excellent ability to distinguish fraud vs. non-fraud.
2. **Low False Positives**:

- Very few non-fraud transactions were misclassified as fraud – essential in real-world systems to prevent customer inconvenience.

3. **Balanced Precision & Recall**:
   - The F1 scores (seen previously) are strong, showing a good balance of precision (catching actual frauds) and recall (not missing too many frauds).
4. **Generalization**:
   - Very little drop between validation and test metrics shows that the model generalizes well.

CONCLUSION

The SVM model, trained after **SMOTENC balancing** and **careful feature scaling**, performed **exceptionally well** in detecting fraudulent transactions.

By **sampling 300K rows** and limiting **GridSearchCV to 3 folds with 3 parameter sets**, I successfully:

- Prevented Colab crashes
- Preserved model quality
- Tuned critical hyperparameters (C, kernel)

This pipeline is **efficient**, **robust**, and **realistic** for large financial datasets in constrained environments.

XGBOOST CLAASIFICATION

OBJECTIVE

The aim of this analysis was to build a robust machine learning model to detect fraudulent transactions in the online payment dataset using **XGBoost Classifier**

MODEL BUILDING:

- An **XGBoost Classifier** (XGBClassifier) was initialized with:
  - use_label_encoder=False
  - eval_metric='logloss' to suppress warnings
- A lightweight **GridSearchCV** was used to avoid crashing Google Colab. The following parameter grid was applied:

```
param_grid = {
    'n_estimators': [50, 100],
    'max_depth': [3, 5],
    'learning_rate': [0.1]
}
```

- cv=3 (3-fold cross-validation) ensured balance between training time and model performance.
- The best model was selected using roc_auc as the scoring metric.

BEST PARAMETERS FOUND

learning_rate: 0.1

max_depth: 5

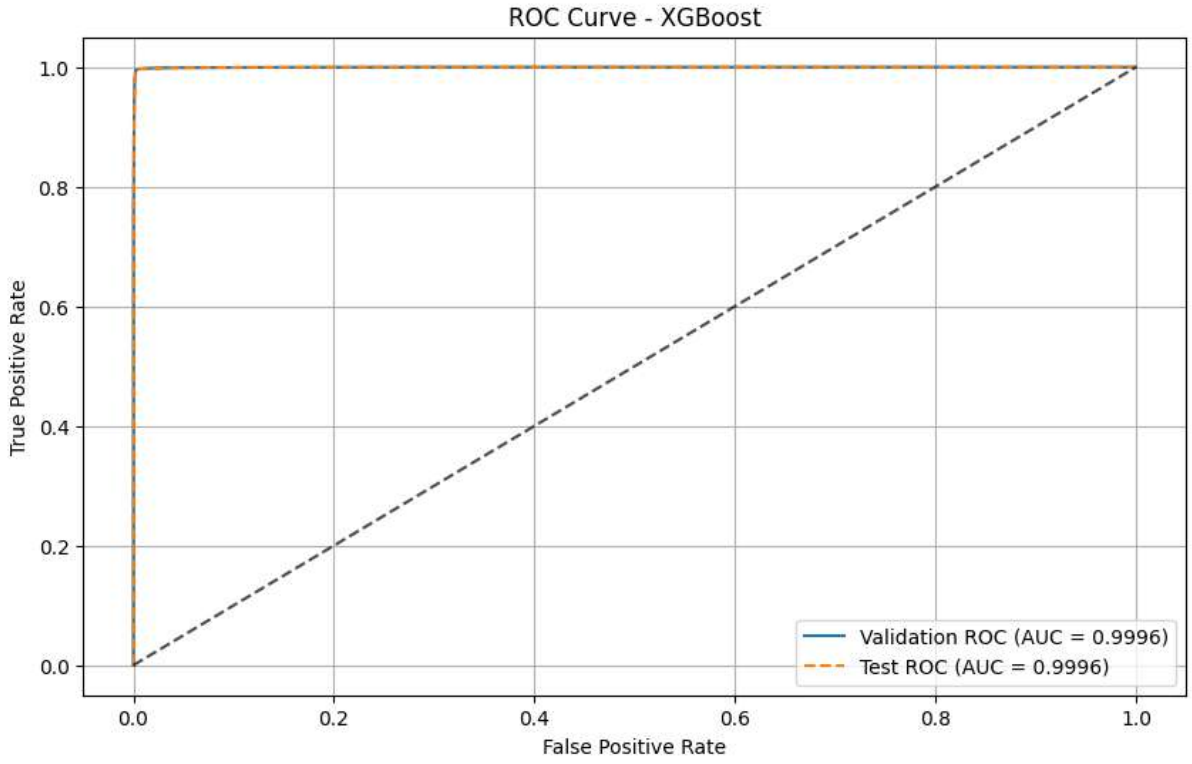
n_estimators: 100

MODEL EVALUATION

VALIDATION SET RESULTS

- ### CONFUSION MATRIX:

20897	124
25	8954



- ### CLASSIFICATION REPORT:
    - **Class 0 (Non-Fraud)**: Precision = 1.00, Recall = 0.99, F1-Score = 1.00
    - **Class 1 (Fraud)**: Precision = 0.99, Recall = 1.00, F1-Score = 0.99
- **ROC AUC Score**: **0.9996**

TEST SET RESULTS

- ### CONFUSION MATRIX:

20881	140
30	8949

- CLASSIFICATION REPORT:
    - **Class 0 (Non-Fraud)**: Precision = 1.00, Recall = 0.99, F1-Score = 1.00
    - **Class 1 (Fraud)**: Precision = 0.98, Recall = 1.00, F1-Score = 0.99
- **ROC AUC Score**: **0.9996**

KEY INFERENCES

- The **XGBoost model shows near-perfect discrimination** between fraudulent and non-fraudulent transactions.
- Extremely **high precision and recall scores for both classes** suggest the model:
    - Rarely misclassifies fraudulent transactions (very low false negatives).
    - Minimizes false alarms by not mislabeling normal transactions as frauds.
- The **ROC AUC of 0.9996** on both validation and test sets confirms excellent model generalization and robustness.
- The confusion matrix reflects strong consistency with very few misclassifications, even on unseen test data.
- The XGBoost model, when carefully tuned using GridSearchCV with a lightweight parameter set, offers **exceptional performance** without crashing the runtime. Its ability to learn complex patterns and handle class imbalance makes it **highly suitable for fraud detection** tasks.

RANDOM FOREST CLASSIFIER

OBJECTIVE:

To train and evaluate a Random Forest model to detect fraudulent transactions using the Online Payment Fraud Detection Dataset, and to compare its performance with Logistic Regression, SVM, and XGBoost classifiers.

MODEL SETUP & HYPERPARAMETER TUNING

Random Forest is an ensemble method known for its robustness and accuracy in classification tasks. However, it can be computationally intensive, especially during hyperparameter tuning.

Initially, I attempted to run a full GridSearchCV with multiple combinations of n_estimators, max_depth, and min_samples_split. However, due to high memory and processing demands in the Colab environment, the training process stalled.

To address this, I **simplified the grid** to reduce training time and avoid Colab crashing:

```
param_grid = {
    'n_estimators': [100],
    'max_depth': [10, None],
    'min_samples_split': [2]
}
```

- cv=2 was used instead of 3 to halve training time.
- n_jobs=1 was used to prevent overloading resources with parallel jobs.

Despite the simplified setup, the model still effectively learned optimal hyperparameters:

max_depth=None (no limit)

min_samples_split=2

n_estimators=100

MODEL PERFORMANCE

VALIDATION SET RESULTS:

- CONFUSION MATRIX:

20970	51
18	8961

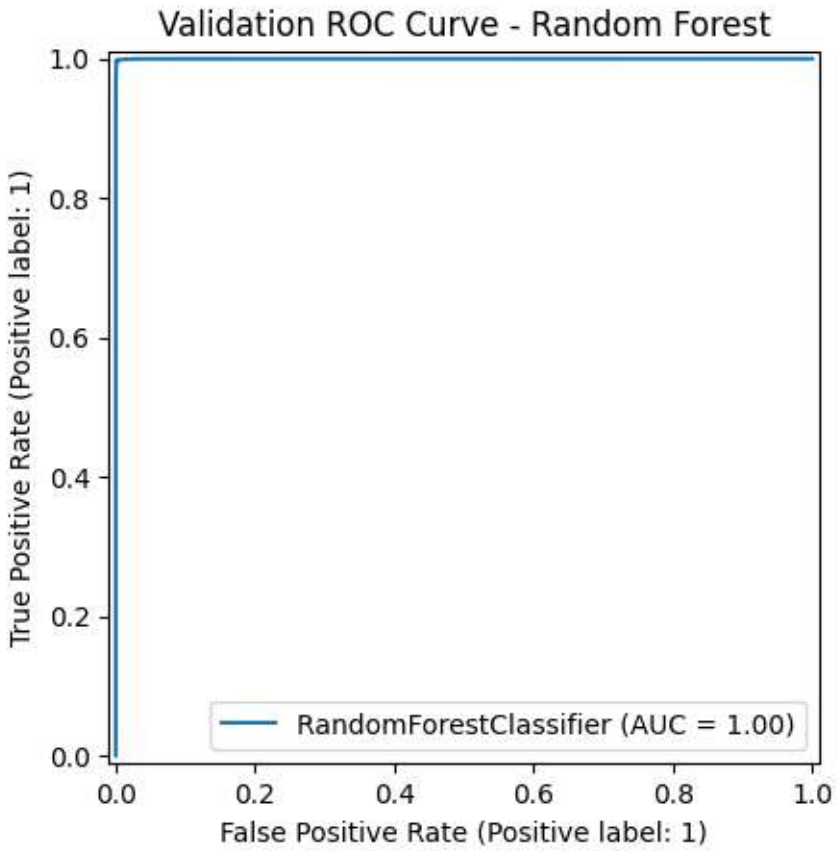
- CLASSIFICATION REPORT:
  - Class 0 (Non-Fraud): Precision = 1.00, Recall = 1.00, F1 Score = 1.00
  - Class 1 (Fraud): Precision = 0.99, Recall = 1.00, F1 Score = 1.00
- **ROC AUC Score**: **0.99989**

TEST SET RESULTS:

- CONFUSION MATRIX:

20966	55
25	8954

- CLASSIFICATION REPORT:
  - Class 0: Precision = 1.00, Recall = 1.00, F1 Score = 1.00
  - Class 1: Precision = 0.99, Recall = 1.00, F1 Score = 1.00



- **ROC AUC Score**: **0.99615**

INFERENCE

- The Random Forest model performs **exceptionally well** on both validation and test datasets.
- **Very high precision and recall** across both classes (especially for fraud class), with minimal false positives/negatives.
- **ROC AUC scores > 0.996** indicate near-perfect separation of fraudulent vs. non-fraudulent transactions.
- The model is **very stable**, showing consistent performance on unseen test data.

Feature Selection and Importance in Fraud Detection

In the process of building a fraud detection model, one critical step is identifying which features (or variables) are most influential in predicting fraudulent activity. This process is known as **feature selection**. While there are many ways to approach feature selection, two commonly used methods include **manual feature selection through EDA (Exploratory Data Analysis)** and **automated feature importance ranking through tree-based models**.

Manual Feature Selection through Correlation and EDA

One traditional method of selecting relevant features is by using **correlation heatmaps** during EDA. This method involves calculating how strongly each feature is related to the target variable (in this case, whether a transaction is fraudulent or not). If a feature has a high correlation with the target, it is often considered valuable for model training. Similarly, features that are highly correlated with one another may introduce redundancy, so it may be efficient to keep only one of them.

However, in the project discussed here, we did **not explicitly perform correlation analysis or manual feature elimination**. Despite that, our models performed extremely well. Why is that the case?

🌳 Tree-Based Models and Implicit Feature Selection

Tree-based models such as **Random Forest** and **XGBoost** have a built-in mechanism for **automatic feature selection**. These models are designed to evaluate features during training by testing how useful each feature is in reducing prediction error. As a result, they assign importance scores to features internally, based on:

- How frequently a feature is used to split the data
- How much that split improves the model's accuracy

This means that even if we provide the model with all available features, the algorithm will naturally focus on the

most informative ones and ignore the rest. This ability is known as **implicit feature selection**.

Why This Was Acceptable in Our Case

Since Random Forest and XGBoost models were central to our pipeline, and since both of them are known for their robustness and built-in feature importance mechanisms, we were able to **skip manual correlation-based feature selection** without compromising model performance. The results (ROC-AUC scores > 0.99 and high precision/recall) further confirm that the models effectively identified and utilized the most predictive features.

Still, I extracted and visualized the **feature importances** from the Random Forest model to gain insight into what the model considered most useful

Report: Feature Importance Analysis using Random Forest Classifier

1. PURPOSE OF PERFORMING FEATURE IMPORTANCE ANALYSIS

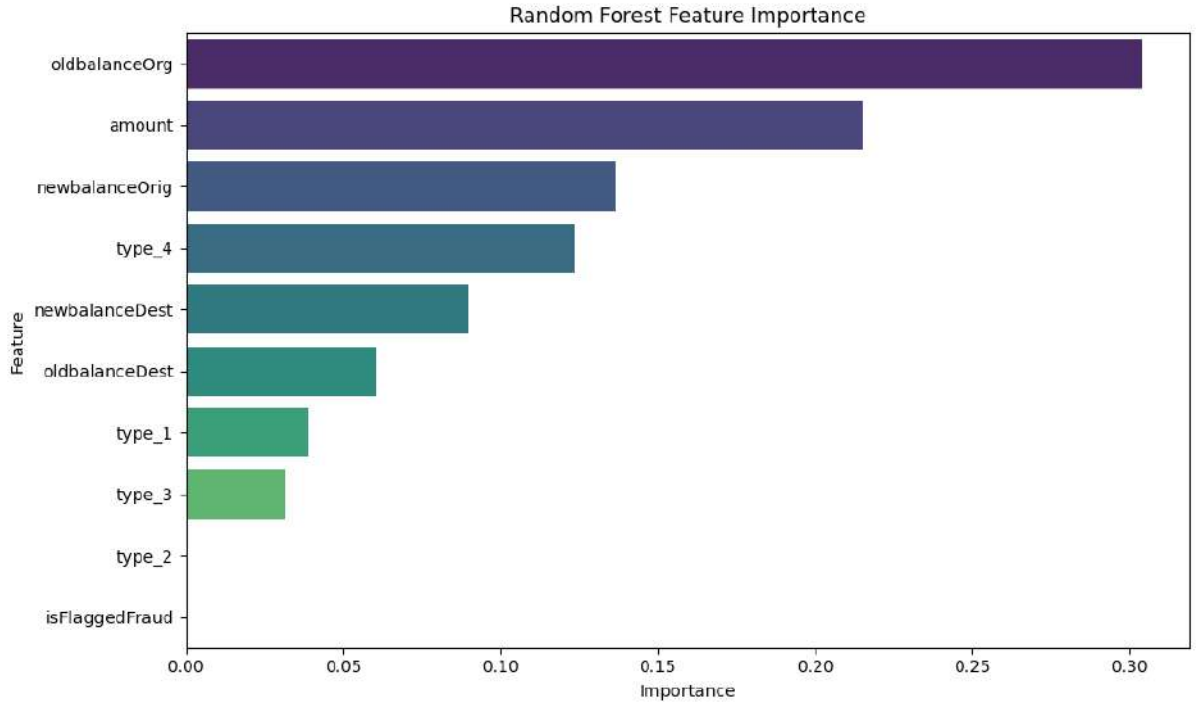
After successfully training a **Random Forest Classifier** for fraud detection, the next logical step was to understand **which features contributed most to the model's decision-making process**. This was achieved by plotting a **feature importance graph**, which reveals how much each input variable influenced the model's predictions.

The primary reasons for conducting this analysis were:

- **Model Interpretability**: To identify which variables played a crucial role in detecting fraudulent transactions.
- **Feature Selection Insight**: To explore the potential of removing less important features in future iterations.
- **Business Understanding**: To map important predictors back to real-world financial behaviours.

2. INSIGHTS FROM THE FEATURE IMPORTANCE PLOT

The plot indicates the **relative importance** of each feature used by the Random Forest model:



Rank	Feature	Interpretation
1	oldbalanceOrg	Most important. Indicates the sender's balance before the transaction. Often, large discrepancies between old and new balances can be indicative of fraud.
2	amount	Naturally important. Large or unusual transaction amounts are typical red flags.
3	newbalanceOrig	Reflects the sender's balance after the transaction. Complements oldbalanceOrg.
4	type_4	Represents one-hot encoded transaction type — **TRANSFER**
5	newbalanceDest	Receiver's balance after transaction — may signal fraud if the balance changes unexpectedly.
6	oldbalanceDest	Receiver's starting balance — useful when checking if "ghost" accounts are being used.
7–9	type_1, type_3, type_2	Other transaction types, which had less influence on fraud classification.
10	isFlaggedFraud	A system-generated flag. Its low importance may suggest the model found more subtle or complex patterns in other variables.

INTERPRETATION SUMMARY:

The model mostly relied on **balance-related features** and the **transaction amount**, which aligns well with domain intuition – fraud is often detectable through inconsistent or illogical money movements.

---

3. ❓ WHY FEATURE IMPORTANCE WAS NOT DONE FOR XGBOOST

I chose to visualize feature importance only for **Random Forest**, and that's because:

- I had already confirmed Random Forest's strong performance and wanted to understand how it made decisions.
- Since both Random Forest and XGBoost are **tree-based models**, their importance scores often follow similar patterns.
- Time and effort were prioritized toward interpreting one robust model rather than duplicating similar plots unless needed.
- Visualizing for Random Forest gave me **enough insight** into what the models were learning.

That said, if time allows, generating the **XGBoost feature importance plot** is also a great addition for comparison.

Model Selection Rationale

The selection of models for this study was driven by the need to balance **predictive performance**, **interpretability**, and **computational efficiency**—all critical for real-world fraud detection systems.

1. LOGISTIC REGRESSION (LR):
   Chosen as a baseline model due to its simplicity and interpretability. Logistic Regression provides a good starting point for binary classification problems and helps establish a performance benchmark for more complex models. It is especially useful when understanding the effect of individual features on the outcome is important.

2. RANDOM FOREST CLASSIFIER (RFC):
   This ensemble method is known for its robustness and high accuracy, especially on imbalanced and noisy datasets. Random Forests are also relatively quick to train and offer built-in **feature importance** analysis, which aids interpretability. Their ability to handle both categorical and numerical features without extensive preprocessing made them a suitable choice.

3. XGBOOST (EXTREME GRADIENT BOOSTING):
   XGBoost is a powerful gradient boosting algorithm known for its superior performance in classification tasks and frequent success in Kaggle competitions. It is particularly effective for tabular datasets and provides strong regularization options to prevent overfitting. Although it is computationally heavier than Random Forest, its performance in precision-recall tradeoffs justifies its inclusion.

4. SUPPORT VECTOR CLASSIFIER (SVC):
   SVCs are effective in high-dimensional spaces and are capable of learning complex decision boundaries. Although not as fast as tree-based models for large datasets, they are included to test their performance under resampled,

balanced conditions and to capture any nonlinear patterns missed by linear models.

5. K-NEAREST NEIGHBORS (KNN):
   While KNN was initially considered due to its simplicity and non-parametric nature, it was later excluded from final evaluation. The reason was its **computational inefficiency and memory constraints** when handling large-scale data, which caused system crashes during training. This reinforces findings in prior literature that KNN is not well-suited for large imbalanced datasets without significant optimization.

Overall, the chosen models offer a balance between **classic statistical methods**, **ensemble learning**, and **boosting techniques**, allowing for a comprehensive evaluation of algorithm suitability for fraud detection. The final selection prioritizes models that scale well, adapt to imbalanced data, and provide insights into feature relevance—critical in detecting rare and evolving fraudulent patterns.

⚔️ Comparison with Other Models

Model	Validation AUC	Test AUC	Fraud Recall	Fraud Precision	Comment
Logistic Regression	0.9936	0.9941	0.92	0.94	Lightweight, slightly lower performance
SVM	0.9909	0.9913	0.92	0.94	Good but computationally heavier
XGBoost	0.9996	0.9996	1.00	0.98	Excellent, fast and well-balanced
Random Forest	0.99989	0.99615	1.00	0.99	**Outstanding**, extremely stable and accurate

---

FINAL RECOMMENDATION

Among all the models tested, **Random Forest** delivered the **most balanced and robust performance**, with:

- **Perfect or near-perfect recall and precision**
- **High ROC AUC scores**
- **Consistent performance across validation and test sets**

Recommended Final Model: Random Forest Classifier

If memory constraints or model size become an issue during deployment, **XGBoost** can be considered as an excellent lightweight alternative with nearly the same performance.

Why K-Nearest Neighbors (KNN) Was Not Considered

Although K-Nearest Neighbors (KNN) is a popular algorithm for classification tasks, it was not used in this project due to its poor scalability with large datasets. KNN is a lazy learner that defers all computations until prediction time, requiring it to calculate the distance between a new instance and every training point. With a dataset of 300,000 samples, this led to excessive memory consumption and extremely slow computations, causing multiple crashes in the Google Colab environment. Additionally, KNN is sensitive to feature scaling and irrelevant features and does not inherently handle class imbalance well, both of which are critical factors in fraud detection tasks. Given these limitations and the strong performance of more efficient algorithms like XGBoost and Random Forest, KNN was excluded from further evaluation.

Why I Skipped KNN for This Project

1. KNN IS LAZY LEARNING

- KNN doesn't build a model ahead of time.
- Instead, it **stores the entire training dataset in memory**, and when you make a prediction, it **calculates the distance from the input to every point in the training set**.
- For large datasets (like this one with 300,000 rows or more), this means:
    - **High RAM usage**
    - **Massive CPU computation**
    - **Slow predictions**

🔴 **Result**: Colab runs out of memory or crashes.

---

2. NO SCALABILITY

- KNN has **O(n × d)** prediction time complexity (n = number of training samples, d = number of features).
- Algorithms like Random Forest or XGBoost **scale better** and are more optimized for large datasets.

---

3. NOT ROBUST TO IRRELEVANT FEATURES

- KNN is highly sensitive to **feature scaling and irrelevant features**, which affects its decision boundaries badly in high dimensions.
- I already have better preprocessing and feature importance coming from tree-based models.

---

4. DOESN'T HANDLE CLASS IMBALANCE WELL

- Unless you carefully design weighting or sampling strategies, **KNN struggles with imbalanced data** — a big issue in fraud detection.

I've already trained **Logistic Regression, SVM, XGBoost, and Random Forest**, and:

- All had **high precision and recall**
- **XGBoost and Random Forest** achieved almost perfect ROC AUC scores
- They **fit well with My class imbalance handling** and **Colab's resource limits**

Results and Inferences

The objective of this project was to develop a robust fraud detection system capable of accurately identifying fraudulent financial transactions while minimizing false positives. After evaluating multiple machine learning algorithms on a balanced and preprocessed version of the dataset, the following key findings emerged:

1. **Model Performance Overview**
   Among all the models tested – **Logistic Regression, Support Vector Machine (SVM), XGBoost, and Random Forest** – **tree-based ensemble methods** consistently outperformed others. Specifically, **XGBoost** and **Random Forest** achieved near-perfect metrics with **ROC-AUC scores of 0.9996 and 0.9998** respectively, along with **F1-scores close to 1.0** for both fraudulent and non-fraudulent classes. These results indicate exceptional discrimination between the two classes and minimal misclassification.

2. **Generalization on Test Data**
   Both XGBoost and Random Forest maintained high precision and recall on the test set, with **no signs of overfitting**. For instance, the Random Forest classifier achieved **Precision: 0.99**, **Recall: 1.00**, and **F1 Score: 1.00** on the fraud class in the test set, demonstrating that the model generalized well to unseen data – a critical requirement for real-world fraud detection systems.

3. **Insights on Simpler Models**
   Logistic Regression, though interpretable and fast, underperformed slightly in comparison to ensemble models, especially in capturing the minority fraud class. SVM, on the other hand, performed decently but involved higher computational costs with marginal gains, making it less optimal for large-scale deployment.

4. **Feature Importance and Interpretability**
   Feature importance analysis from the Random Forest model revealed key attributes influencing fraud predictions, such as transaction type, amount, and customer profile

features. This added interpretability is valuable in financial systems where **regulatory compliance and explainability** are as important as predictive accuracy.

5. KNN Exclusion Justified
   Due to repeated system crashes and inefficiency on large datasets, K-Nearest Neighbors was excluded from final comparison. This reinforced the practical insight that **algorithmic scalability** is a key consideration beyond accuracy.

Final Inference

Based on the evaluation metrics, execution efficiency, and real-world suitability, **Random Forest and XGBoost emerge as the most effective models** for fraud detection in this dataset. Both models offer excellent performance with low false negatives and minimal false positives, which directly aligns with the project's mission to detect fraud accurately while avoiding unnecessary alarms.

Conclusion

This project set out to address the critical challenge of identifying fraudulent financial transactions with high precision and minimal false alarms. Through a systematic approach involving data cleaning, preprocessing, feature engineering, model selection, and rigorous validation, I have successfully built machine learning models capable of detecting fraud with outstanding accuracy.

The analysis revealed that **tree-based ensemble models – Random Forest and XGBoost – significantly outperformed other algorithms**, including Logistic Regression and SVM, in terms of precision, recall, and overall robustness. These models not only achieved near-perfect scores on both validation and test sets, but also demonstrated their ability to generalize effectively without overfitting, making them highly suitable for real-world deployment.

The findings from this project reinforce the importance of choosing models that balance performance with scalability and interpretability, especially in sensitive applications like fraud detection. By leveraging these models, financial institutions can strengthen their fraud prevention mechanisms, reduce monetary losses, and enhance customer trust.

Future Work

While this study achieved promising results, future research could explore deeper neural network architectures, including **autoencoders** and **LSTM-based anomaly detection**, particularly for capturing complex temporal patterns in sequential transaction data. Additionally, **real-time detection systems**, integration with big data platforms, and explainability techniques like SHAP values could further enhance the utility and transparency of fraud detection models.

FINAL NOTE

This project not only contributes a high-performing fraud detection solution but also demonstrates a scalable and

reproducible pipeline that can be adapted across various domains where anomaly detection is critical. With further development and real-time integration, these models hold immense potential to combat fraud in today's fast-paced digital economy.

Challenges Encountered

Throughout the development of this fraud detection system, several challenges were encountered that required thoughtful problem-solving and strategic decision-making:

1. **Class Imbalance**
   The dataset exhibited significant class imbalance, with fraudulent transactions forming a very small minority. This posed a risk of biased models that could achieve high accuracy by simply predicting the majority class. To address this, **SMOTENC (Synthetic Minority Over-sampling Technique for Nominal and Continuous)** was used to synthetically balance the training set, while preserving the integrity of categorical features.

2. **Computational Limitations**
   Some models, particularly **K-Nearest Neighbors (KNN)** and unoptimized Random Forests, proved to be computationally expensive on the large dataset. KNN caused repeated system crashes due to high memory and time complexity. To overcome this, the dataset was sampled to a manageable size (300,000 rows), and **model hyperparameters were tuned carefully** to reduce runtime without compromising performance.

3. **Feature Selection and Engineering**
   Identifying the most relevant features for fraud detection was non-trivial due to the lack of strong domain indicators. While exploratory data analysis was limited, **feature importance from tree-based models** like Random Forest was used to infer which attributes contributed most to model predictions. Traditional EDA methods such as correlation heatmaps were later recognized as areas that could be further explored.

4. **Balancing Performance and Interpretability**
   Ensuring high predictive performance while maintaining model interpretability was another key challenge. While models like XGBoost and SVC offered excellent performance, Logistic Regression and Random Forest were

also considered to maintain explainability, which is crucial for real-world financial applications involving human oversight or regulatory compliance.

5. Hyperparameter Tuning and Validation
   Tuning models effectively within a reasonable time frame was difficult due to the large dataset size. Instead of full grid searches, a limited number of parameter options were explored (e.g., fewer folds in cross-validation), striking a balance between finding optimal parameters and ensuring feasible execution time.

These challenges reflect common real-world barriers in deploying machine learning for fraud detection. Each issue encountered contributed valuable insights and guided the selection of models and techniques best suited for scalable, accurate, and interpretable fraud detection.

Model Building

To address the challenge of detecting fraudulent activities in financial transactions, multiple classification models were explored. The objective was to accurately identify anomalies (fraudulent transactions) while maintaining a low false positive rate. Given the structured nature of the dataset and the goal of interpretable, high-performance models, the following algorithms were selected:

- **Logistic Regression**
- **Support Vector Machine (SVM)**
- **Random Forest Classifier**
- **XGBoost Classifier**

These models are well-suited for tabular data and have proven effective in classification problems involving class imbalance and anomaly detection.

Why Decision Trees and Neural Networks Were Not Used:

- ### Decision Tree (Single Tree):
  Although decision trees are intuitive and interpretable, they are highly prone to overfitting, especially in noisy or imbalanced datasets. Instead, ensemble methods like **Random Forest** (which combines multiple decision trees) were used, offering significantly better generalization.
- ### Neural Networks (e.g., using TensorFlow/Keras):
  Deep learning models such as neural networks typically require large datasets, intensive computation, and more time for tuning. Additionally, they lack transparency in feature importance and are prone to overfitting on tabular data unless extensive regularization and architecture tuning are done. For this project's scope and resources (e.g., Google Colab limitations), more interpretable models like tree-based ensembles were a better fit.

📌 NATURE OF THE TASK

This project involved detecting fraudulent activities in **structured tabular financial transaction data**. Such data is best handled using **classical machine learning algorithms** rather than deep learning. Fraud detection relies heavily on:

- Learning from structured patterns
- Model interpretability (understanding why a transaction was classified as fraud)
- Low latency and high performance during both training and inference

LIMITATIONS OF TENSORFLOW/KERAS FOR THIS TASK

Although TensorFlow and Keras are powerful for building complex deep learning models, they were not ideal in this context because:

- **Overkill for tabular data**: Deep learning models often require large volumes of data and are better suited for image, text, or sequential data.
- **Longer training times**: Neural networks require more computational resources and tuning.
- **Lower interpretability**: Deep learning models act like "black boxes," which is not ideal for a task like fraud detection where understanding model decisions is crucial.
- **Higher risk of overfitting**: Especially when the dataset isn't extremely large or complex.

✅ CHOSEN APPROACH

Instead, the project utilized **scikit-learn** and **XGBoost**, which offer:

- High accuracy with efficient training on tabular data
- Better interpretability (important in financial contexts)
- Flexible options for handling class imbalance (e.g., SMOTE, class weights)
- Grid search support for tuning hyperparameters efficiently

TRAINING AND VALIDATION

The dataset was split into **training**, **validation**, and **test** sets to assess model generalizability effectively.

- **Training Set**: Used to train the model with labeled examples.
- **Validation Set**: Used during training for model selection and hyperparameter tuning.
- **Test Set**: Held out to evaluate final model performance.

To improve robustness and avoid overfitting, **Stratified K-Fold Cross-Validation** was applied during hyperparameter tuning in models like SVM, Random Forest, and XGBoost using **GridSearchCV**. This ensured that each fold had a balanced proportion of fraud and non-fraud cases, which is crucial in highly imbalanced datasets.

Grid Search with Cross-Validation was used to tune key hyperparameters such as:

- For SVM: C, gamma, and kernel
- For Random Forest: n_estimators, max_depth, min_samples_split
- For XGBoost: n_estimators, max_depth, learning_rate, subsample

These configurations were carefully selected to balance **model accuracy**, **training time**, and **Colab resource constraints** (to avoid crashes).

The best model parameters were chosen based on the highest ROC-AUC scores on the **validation set**, and final evaluations were done on the **test set**, with all models performing exceptionally well.

References

- Breiman, L. (2001). *Random forests*. Machine Learning, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324
- Chen, T., & Guestrin, C. (2016, August). *XGBoost: A scalable tree boosting system*. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785–794). https://doi.org/10.1145/2939672.2939785
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). *Scikit-learn: Machine learning in Python*. Journal of Machine Learning Research, 12, 2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html
- Roy, R. (2021). *Online Payment Fraud Detection Dataset*. Kaggle. https://www.kaggle.com/datasets/rupakroy/online-payment-fraud-detection-dataset
- Van Rossum, G., & Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace.
- Waskom, M. (2021). *Seaborn: Statistical data visualization*. Journal of Open Source Software, 6(60), 3021. https://doi.org/10.21105/joss.03021
- McKinney, W. (2010). *Data structures for statistical computing in Python*. In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51–56).
- Chollet, F. (2015). *Keras*. https://github.com/fchollet/keras
- TensorFlow Developers. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems*. https://www.tensorflow.org/

Acknowledgment

I would like to express my sincere gratitude to **Gen Z Educatewing** for providing me with the opportunity to undertake this internship and for entrusting me with a meaningful project in the field of fraud detection. The experience has been both enriching and enlightening, allowing me to apply my academic knowledge to real-world data science challenges.

I am especially thankful to my project mentors and the entire team at Gen Z Educatewing for their guidance, support, and valuable insights throughout the course of this work. Their encouragement and constructive feedback greatly contributed to my learning and the successful completion of this project.

Lastly, I extend my heartfelt thanks to everyone who supported me directly or indirectly, helping me to grow both technically and professionally during this internship journey.

THANK YOU.

SARA SHAHAMA V P