Research Project Report
of
Fraud Transaction Detection in Mobile Payment Systems

Acknowledgment

I want to say how grateful I am to the teachers at the IIT, NSTU for giving me a chance to do this research on fraud transaction detection in mobile payment systems as part of the final software project lab. Their passion and cooperation have been critical to the success of our endeavor.

I would like to express my sincere gratitude to my project supervisor, **"Md. Eusha Kadir"** Sir for his direction, knowledge, and ongoing assistance throughout the study process. Their insightful opinions and helpful suggestions were extremely helpful in determining the course and results of this investigation.

I'm also appreciative of the individuals who shared information and perspectives for this study. Their participation and openness to offer assistance have significantly improved the research's findings.

Abstraction

Mobile payment solutions are becoming more and more common as technology develops. These technologies have shown to be quite practical for buying, using, and safely keeping tickets in the public transportation sector. But fraud is a serious problem that has emerged as a result of the growing use of mobile payments. Fraudulent activity not only lowers consumer satisfaction but also puts businesses operating in this area at financial risk and incurs significant expenditures. Therefore, it is crucial to effectively identify and stop fraudsters if mobile payment systems in the public transportation sector are to remain secure. In this study, we make use of highly unbalanced synthetic data from the Paysim Spanish Bank. We use three distinct strategies. Without balancing the data, utilizing Random under-sampling, SMOTE, or SMOTE followed by machine learning algorithms are the first two options. We demonstrate that the best method we presented, Random Forest with SMOTE, is the most effective at identifying fraudulent transactions.

Keywords: Fraud Detection, Machine Learning, Mobile Payment, Imbalance Data, Ensemble Learning.

Chapter 1: Introduction

1.1 Background and Motivation

Mobile payment solutions are incredibly popular across a variety of businesses due to the quick development of technology. When conducting business using mobile devices, these technologies provide efficiency, security, and convenience. The risk of fraudulent operations has grown along with the growth in the use of mobile payments, nevertheless. In many different businesses, detecting and preventing fraud in mobile transactions has become a top priority. Solving this problem is crucial for maintaining the dependability and security of mobile payment systems.

1.2 Problem Statement

Mobile payment systems are now in danger of fraud due to their broad adoption across several industries. Fraudulent transactions not only have a negative impact on consumer satisfaction, but they also put organizations at financial risk and add significantly to their costs. In order to identify and prevent fraud in mobile transactions across many businesses, it is crucial to create efficient solutions.

1.3 Research Objectives:

This study aims to achieve the following research objectives:

- Investigate the nature and patterns of fraud prevalent in mobile payment systems across multiple sectors.
- Evaluate and compare different approaches for fraud detection and prevention, including data balancing techniques (random under-sampling, SMOTE, etc.) and machine learning algorithms.
- Apply these approaches to diverse datasets representing various industries and transaction scenarios to assess their effectiveness.
- Identify the best-performing approach for fraud detection in mobile payment systems across different sectors.
- Explore the potential of new techniques and feature extraction methods to enhance fraud detection accuracy in mobile transactions.

1.4 Scope and Limitations

This study covers a variety of industries' mobile payment systems for fraud detection and prevention. To guarantee the generalizability of the suggested approaches, the study takes a variety of datasets from diverse industries into consideration. It is vital to understand that the limitations of the selected approaches as well as the quirks of the specific datasets used might affect the results and interpretations.

1.5 Significance of the Study

The effective detection and prevention of fraud in mobile payment systems will benefit several enterprises. This study helps by reducing financial risks, boosting customer confidence, and removing costs associated with fraud. The study's findings provide useful knowledge on the best strategies for preventing fraud in mobile transactions across a variety of businesses, ultimately boosting the dependability and security of mobile payment systems.

Chapter 2: Literature Review

2.1 Relevant Theories and Concepts

Exploring pertinent theories and concepts that have been covered in prior research is crucial when doing a literature review on fraud transaction detection. Key theories and ideas that are frequently discussed in this field include the following:

1. Fraud Detection Techniques: This includes a range of techniques and algorithms used to find fraudulent behavior in transactional data. Examples include statistical analysis tools, anomaly detection techniques, rule-based approaches, and machine learning algorithms (such as Random Forest, Support Vector Machines, and Neural Networks).

2. Class Imbalance Problem: The number of genuine transactions is frequently significantly higher than the number of fraudulent transactions in fraud detection databases, indicating a large class imbalance. To successfully balance the dataset, researchers have suggested strategies including under-sampling the majority class, oversampling the minority class (e.g., using SMOTE), or employing hybrid sampling approaches.

3. Feature Selection and Engineering: A key component of fraud detection is the identification and engineering of pertinent attributes. In order to find useful characteristics that may distinguish between fraudulent and genuine transactions, researchers have investigated a variety of feature extraction approaches, dimensionality reduction strategies, and domain-specific information.

4. Ensemble Methods: Ensemble approaches integrate many models to increase the performance of fraud detection. By combining the predictions of many base models, strategies including bagging, boosting, and stacking have been used to improve the accuracy and resilience of fraud detection models.

5. Evaluation Metrics: For evaluating the efficacy of fraud detection programs, choosing the right assessment criteria is essential. Common measurements include area under the receiver operating characteristic curve (AUC-ROC), area under the precision-recall curve (AUC-PR), accuracy, precision, recall, F1-score, and FNR. Taking into account the trade-offs between these measures in light of the application's particular needs is crucial.

6. Real-Time Fraud Detection: Real-time fraud detection methods have gained popularity as transaction speeds and volumes have accelerated. These methods, which frequently make use of parallel computing, scalable algorithms, and streaming data analysis, concentrate on swiftly and correctly identifying fraudulent transactions.

7. Domain-Specific Fraud Detection: There may be particular requirements and problems for fraud detection in various companies and sectors. In the literature, there may be discussions of methodologies that are specialized to certain industries, such as fraud detection in banking, e-commerce, healthcare, insurance, or mobile payment systems. These studies emphasize the distinctive traits, patterns, and methods particular to each domain.

2.2 Review of Related Research and Gaps

Table 1: Literature Review Summary

Study	**Data(Fraud/legimate**	**Method**	**Performance**
Rieke et al. (2013)	synthetic logs (20/5,297)	predictive security analyser	FNR=0.550
Coppolino et al. (2015)	synthetic logs	Dempster-Shafer theory	FNR=0.240
Xenopoulos (2017)	PaySim (492/284,315)	ensemble of deep belief networks	Acc=89.05, AUC =0.961
Choi and Lee (2017; 2018)	Korean payment data (2,402/274,670)	unsupervised (EM, K-means, FarthestFirst, X-means, MakeDen-sity), supervised (NB, SVM, LR, OneR, C4.5, RF)	Acc=99.97
Mubalaike and Adali (2018)	PaySim (8,213/6M)	restricted Boltzman machines	Acc=91.53
Du et al. (2018)	PaySim (8,213/6M)	SVM with LogDet regularization	Acc=97.57, AUC =0.978
Zhou et al. (2018)	Chinese bankcard enrolment (5,753/∼ 52M)	GB DT, LR, RF, rule-based expert	Precision=50.83, Recall=0.25
Pambudi et al. (2019)	PaySim (4,093/246,033)	RUS+SVM	F1=0.900, AUC =0.880
Misra et al. (2020)	PaySim (492/284,315)	Autoencoder+MLP	F1=0.900, AUC =0.880
Mendelson and Lerner (2020)	PaySim (492/284,315)	cluster drift detection	AUC=0.898
Schlör et al. (2021)	PaySim (8,213/6M)	deep MLP with ReLU and iNALU	F1=0.880, AUC =0.960
Buschjager et al. (2021)	PaySim (269/572K)	generalized Isolation Forest	AUC=0.821
Petr Hajek (2022)	PaySim (8,213/6M)	XGBoost + XGBOD	Accuracy 0.9955
This Study	PaySim (8,213/6M)	Random Forest +SMOTE	

Although fraud transaction detection has advanced, there are still a number of research gaps that require attention. To increase the effectiveness of fraud detection systems, more and more diversified real-world datasets are required. Second, the literature lacks a thorough assessment of cutting-edge machine learning-based techniques, particularly those that make use of under-sampling techniques. Additionally underutilized are hybrid semi-supervised techniques that combine supervised learning with unsupervised outlier identification. Last but not least, rather than depending simply on conventional performance indicators, the evaluation of fraud detection performance should take financial repercussions into account.

Chapter 3: Methodology

3.1 Research Design

The research design section's goal is to give a thorough and understandable explanation of the research process, including the approaches, methods, and resources used to look into and solve the research topic. It ensures transparency and reproducibility by allowing readers to comprehend the study's methodology. Additionally, it enables a critical assessment of the study design, determining the accuracy and dependability of the results. Here is the research design overview of fraud transaction detection.

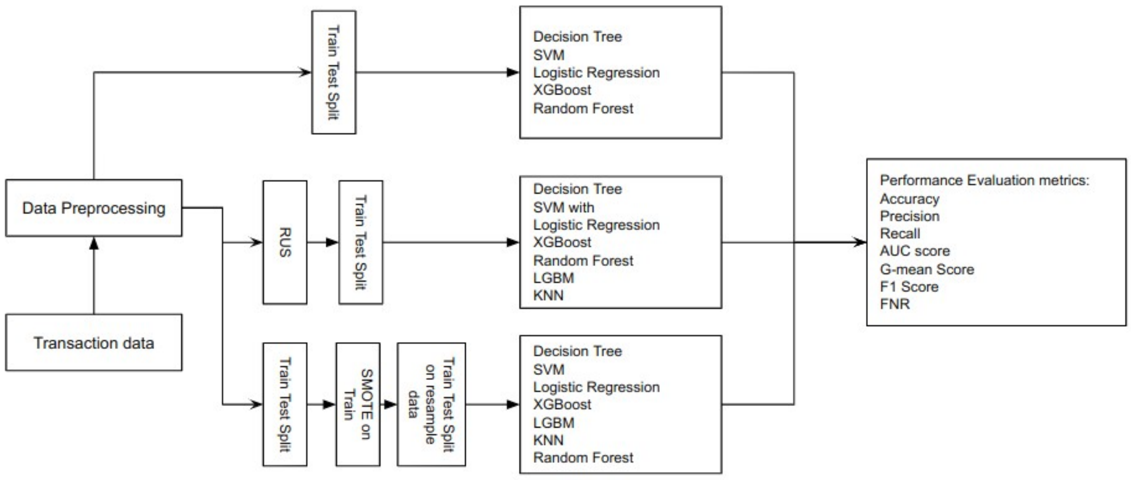


Figure 1: Research Design Diagram

3.1.1 Data Preprocessing

3.1.1.1 Data Transformation

- Encoding

We have three categorical features (nameOrig, type, nameDest) in our dataset. To transform the "type" feature into a suitable format for modeling, we use One-hot encoding. One-hot encoding is a technique that converts categorical variables into a binary vector representation. In this case, we can create separate binary columns for each unique value in the "type" feature (CASH-IN, CASH-OUT, DEBIT, PAYMENT, and TRANSFER).

For example, if the following data:

Table 2: "type" feature of dataset

type
CASH-IN
CASH-OUT
DEBIT
PAYMENT
TRANSFER

After applying one-hot encoding, the transformed data would look like this:

Table 3: One hot encoding representation of "type" feature

CASH-IN	CASH-OUT	DEBIT	PAYMENT	TRANSFER
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

Each value in the "type" feature has been transformed into a separate binary column. If a transaction belongs to a particular type, the corresponding binary column will have a value of 1, and all other columns will have a value of 0.

By applying one-hot encoding, we capture the categorical information of the "type" feature in a format that machine learning algorithms can effectively utilize for fraud transaction detection.

We have also two categorical features, 'nameOrig' and 'nameDest'. To transform those features:

1. Initialize a tokenizer object using tf.keras.preprocessing.text.Tokenizer(). The tokenizer is used to convert text into numerical sequences.
2. The tokenizer is fitted on the text data in the 'nameDest' and 'nameOrig' columns of the data. This step builds the vocabulary based on the unique words present in the

'nameDest' column. This ensures that the vocabulary contains all the unique words from both 'nameDest' and 'nameOrig' columns.

3. The 'nameOrig' and 'nameOrig' column of the data is converted into sequences of numerical indices using the tokenizer. Each word in each entry of the 'nameOrig' and 'nameOrig' columns is replaced with its corresponding index and create new column ‘customers_nameOrig’ and ‘customers_nameDest’.
4. The 'nameDest' and 'nameOrig' columns are dropped from the data since they have been encoded and are no longer needed.
5. The sequences in the 'customers_nameOrig' and ‘customers_nameDest’ lists are padded to a maximum length of 1 using tf.keras.preprocessing.sequence.pad_sequences. The padding ensures that all sequences have the same length, which is required when feeding data into a machine-learning model.

The purpose of these steps is to convert the text data in the 'nameOrig' and 'nameDest' columns into numerical sequences and pad them to a uniform length. This preprocessing is often performed when working with categorical features in machine learning models, as models typically require numerical inputs rather than text inputs. The encoding and padding ensure that the data is in a suitable format for training machine learning models.

Source: tf.keras.preprocessing.sequence.pad_sequences

- Normalization

StandardScaler is a data normalization technique commonly used in machine learning and statistics. It is used to transform numerical data in a dataset by scaling it to have a mean of 0 and a standard deviation of 1. This process ensures that the features are on a similar scale, preventing some features from dominating the learning process due to their larger values.

Mathematically, StandardScaler normalizes each feature (column) of a dataset by subtracting the mean of that feature and dividing it by its standard deviation. The formula for StandardScaler is as follows:

Standardized value = (x - mean) / standard deviation

Where:

"x" represents the original value of a data point in a specific feature (column).
"mean" represents the mean of that feature (the average value of all data points in that feature)."standard deviation" represents the standard deviation of that feature (a measure of how spread out the data points are from the mean).

By applying this formula to each data point in every feature, the resulting dataset will have a mean of 0 and a standard deviation of 1. This transformation is performed independently for each feature, allowing the data to be centered around zero and have a uniform scale.

The StandardScaler transformation can be applied using various programming libraries and frameworks, such as scikit-learn in Python.

source: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

3.1.2 Experimental Setup

Step 1: Data Partitioning:

- ○ Randomly create training and testing data with a 3:1 ratio (75% training data, 25% validation data).
- ○ Repeat this process 10 times to ensure reliable performance evaluation.

Step 2: Hyperparameter Selection:

- ○ Perform an optimal selection of hyperparameters for fraud detection methods using 10-fold cross-validation on the training data.
- ○ Use the list of hyperparameters and their values provided.

Step 3: Supervised Learning Implementation:

- ○ Python library Scikit-Learn (version 1.2.2) for supervised learning methods.
- ○ LightGBM (version 3.3.2) for LGBM (Light Gradient Boosting Machine).
- ○ Imbalanced-Learn (version 0.3.0.dev0) for data balancing.

Step 4: Fraud Detection:

- ○ Perform fraud detection on mobile payment transactions using the supervised learning methods mentioned above.
- ○ Use the training data for training the models and the testing data for evaluating their performance.

Step 5: Evaluation and Analysis:

- ○ Evaluate the performance of the fraud detection methods using appropriate metrics (e.g., Accuracy, G-mean Score, F1 Score, AUC, Recall, Precision, FNR) Compare the results obtained from different methods and identify the most effective approach.

- ❖ Without Balance Dataset
  - Step 1-5

- ❖ Balance Dataset using Random Under Sampling
  - First Balance the dataset using RUS
  - Step 1-5

- Balance Dataset using SMOTE
    - First Split the data into train and test
    - Balance the train data Using SMOTE(In this way there is no chance to data redundancy or leakage)
    - Step 1-5 (Evaluate performance based on test data)

3.2 Experimental Approach

In our fraud detection study, we investigated three distinct approaches. The first approach focused on applying machine learning algorithms to an unbalanced dataset, without employing any data-balancing techniques. The second approach involved utilizing a Random Under Sampling technique to balance the data before applying machine learning algorithms. Lastly, the third approach utilized Synthetic Minority Over-sampling Technique (SMOTE) to balance the data and then evaluated the performance of various machine learning algorithms.

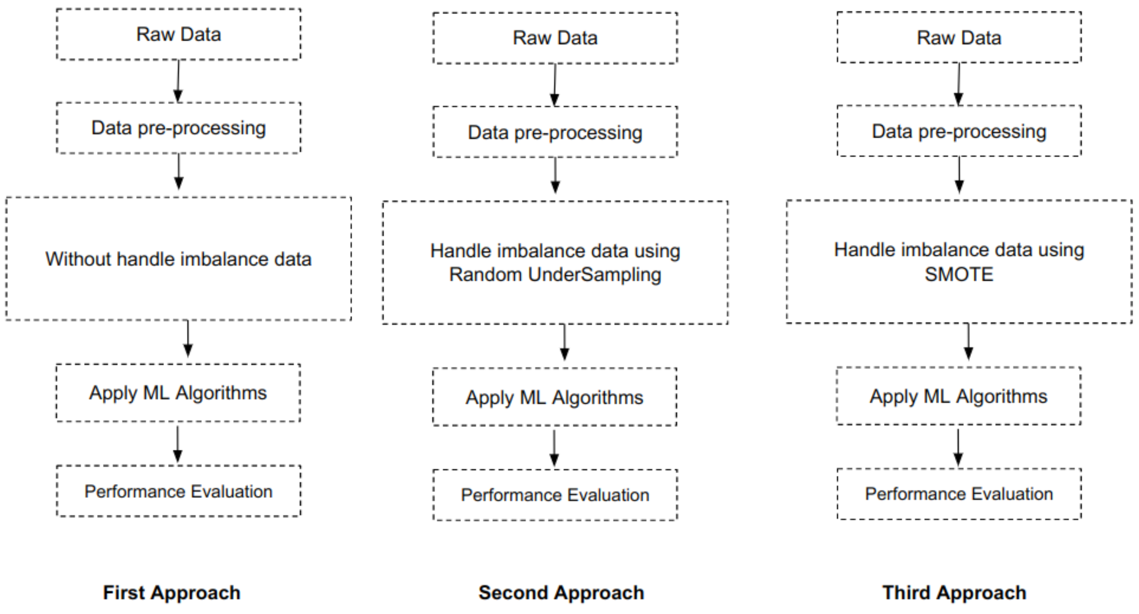


Figure 2: Experimental Approach

Among the three approaches we examined, the third approach, which involved balancing the data using SMOTE and applying different machine learning algorithms, yielded the best results when using the Random Forest algorithm. This combination demonstrated superior performance in detecting fraud compared to the other approaches we explored in our study

3.3 Data Analysis Techniques

In our research, we employed various data analysis techniques to extract meaningful insights and prepare the data for model development and prediction. The following techniques were applied:

Data Cleaning and Preprocessing: Before performing any analysis, we conducted data cleaning and preprocessing to ensure the data quality and consistency. This involved handling missing values, checking for duplicates, and addressing any data inconsistencies or errors. Additionally, we performed data encoding to transform categorical variables such as the transaction type (CASH-IN, CASH-OUT, DEBIT, PAYMENT, and TRANSFER) into numerical representations suitable for analysis.

Exploratory Data Analysis (EDA): Understanding the traits and patterns found in the dataset requires the use of EDA. To understand the distribution, connections, and trends of various variables, we used a variety of statistical and visualization tools. This enabled us to find probable outliers, comprehend the significance of the features, and investigate any possible links between the variables. We were able to better comprehend the data's nature and make wise judgments during the next phases thanks to EDA.

Feature Scaling: Feature scaling is essential to ensure that all features contribute equally to the model training process. We employed the Standard Scaler, a commonly used technique, to standardize the numerical features in the dataset. This transformation brings the features to a similar scale, preventing any particular feature from dominating the model's learning process. By applying feature scaling, we aimed to enhance the performance and convergence of the model during training.

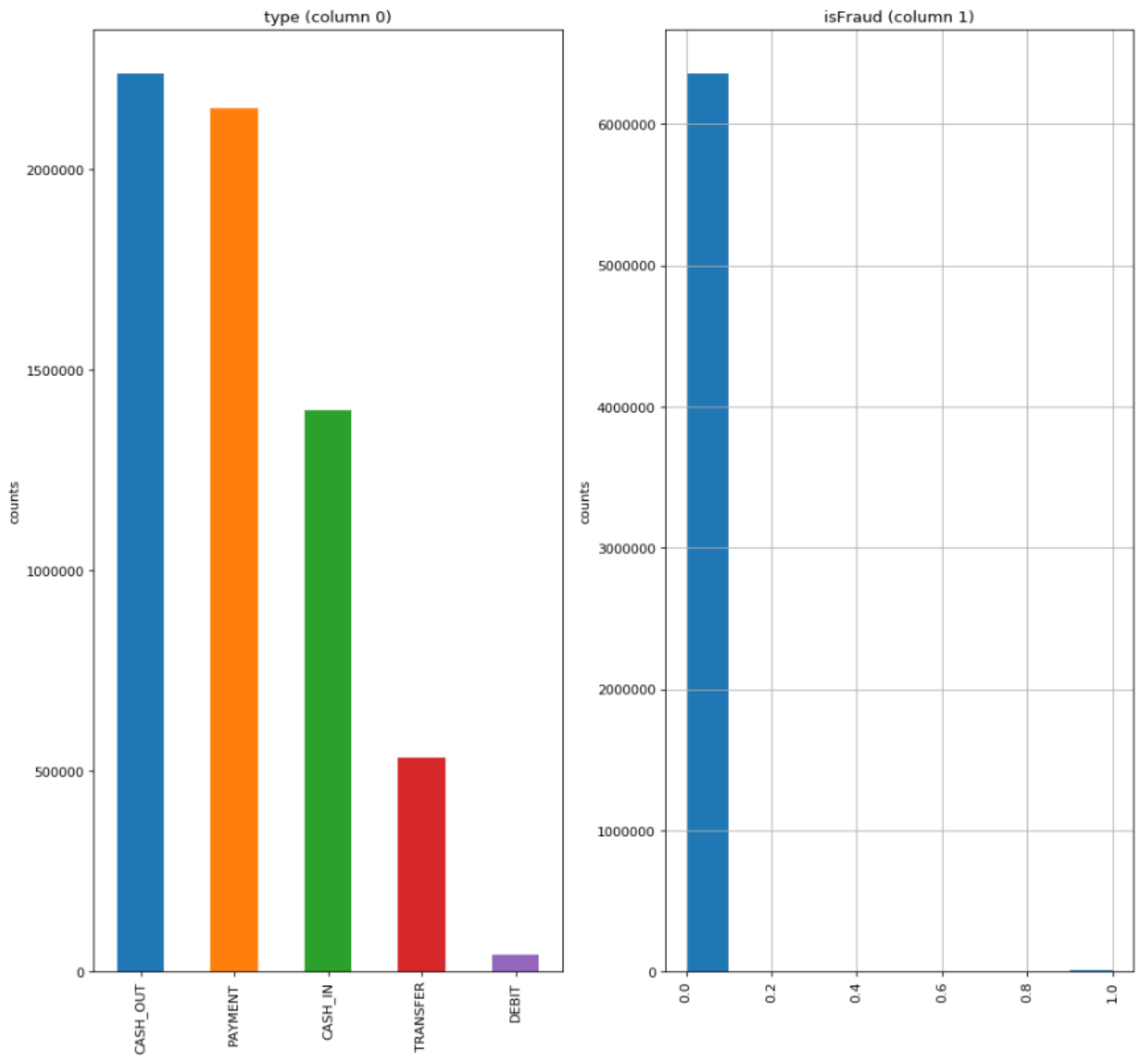


Figure 3: Transaction "type" and "isFraud" Feature

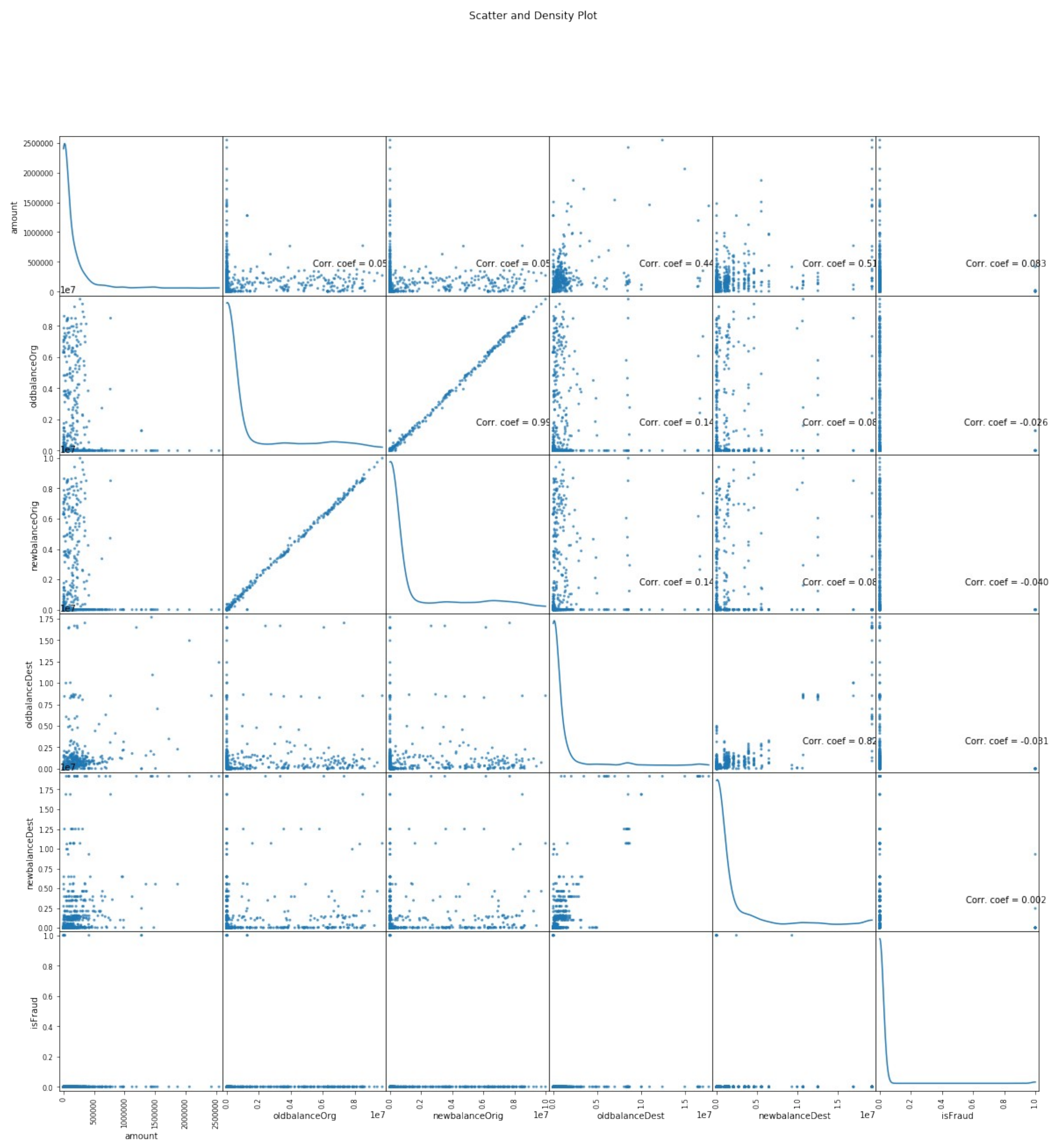


Figure 4: Scatter and Density plot of PaySim data

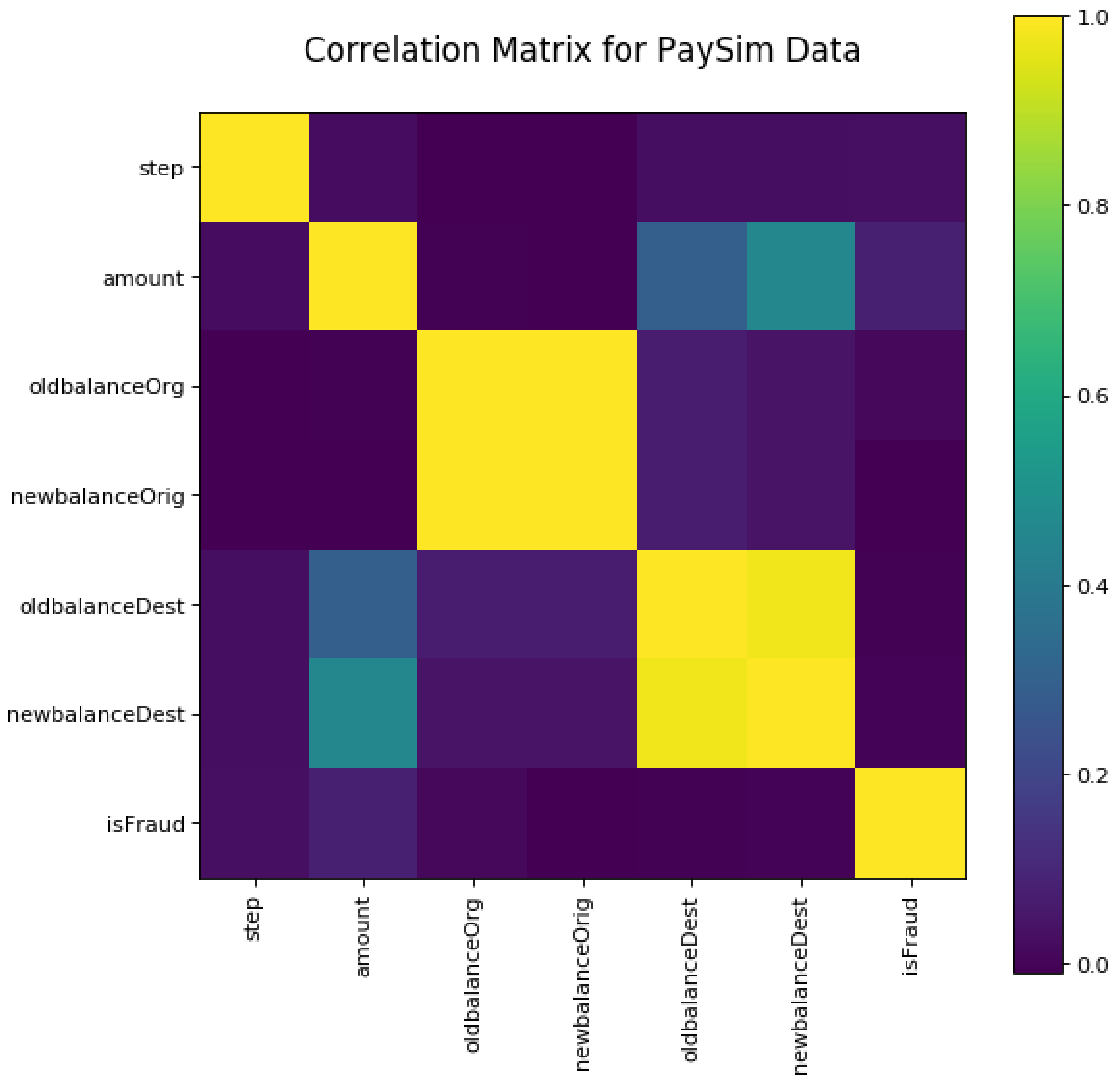


Figure 5: Correlation Matrix for PaySim data

These data analysis techniques collectively enabled us to clean, preprocess, and explore the dataset, making it suitable for model development. By understanding the data distribution and relationships between features, we could effectively identify patterns and correlations that are crucial for accurate fraud transaction detection. Moreover, feature scaling ensured that all features contributed optimally to the model's predictions.

3.4 Proposed Techniques

3.4.1 Random Forest with SMOTE

Random Forest

Random Forest is a machine learning algorithm that belongs to the ensemble learning family. It combines the predictions of multiple decision trees to make more accurate and robust predictions.

Here is a simplified step-by-step process of how Random Forest works:

Step 1: Data Sampling

- Randomly select a subset of the training data (with replacement) to create a random subset of data called a bootstrap sample.
- This random sampling allows different decision trees to be trained on different subsets of the data.

Step 2: Tree Construction

- Build a decision tree using the bootstrap sample created in Step 1.
- At each node of the tree, randomly select a subset of features to consider for splitting.
- Split the node based on the selected feature that provides the best separation of data.

Step 3: Repeat Steps 1 and 2

- Repeat Steps 1 and 2 to create multiple decision trees.
- Each decision tree is constructed using a different bootstrap sample and a random subset of features.

Step 4: Voting or Averaging

- For classification: When making predictions, each tree in the random forest independently predicts the class of a data point. The final prediction is determined by majority voting. The class with the most votes across all trees is chosen.
- For regression: When making predictions, each tree in the random forest independently predicts the target value of a data point. The final prediction is the average (or weighted average) of the predictions from all trees.

Step 5: Prediction

- Use the ensemble of decision trees to make predictions on new, unseen data.
- For classification, the class with the majority vote is chosen as the final prediction.
- For regression, the average (or weighted average) of the predictions is taken as the final prediction.

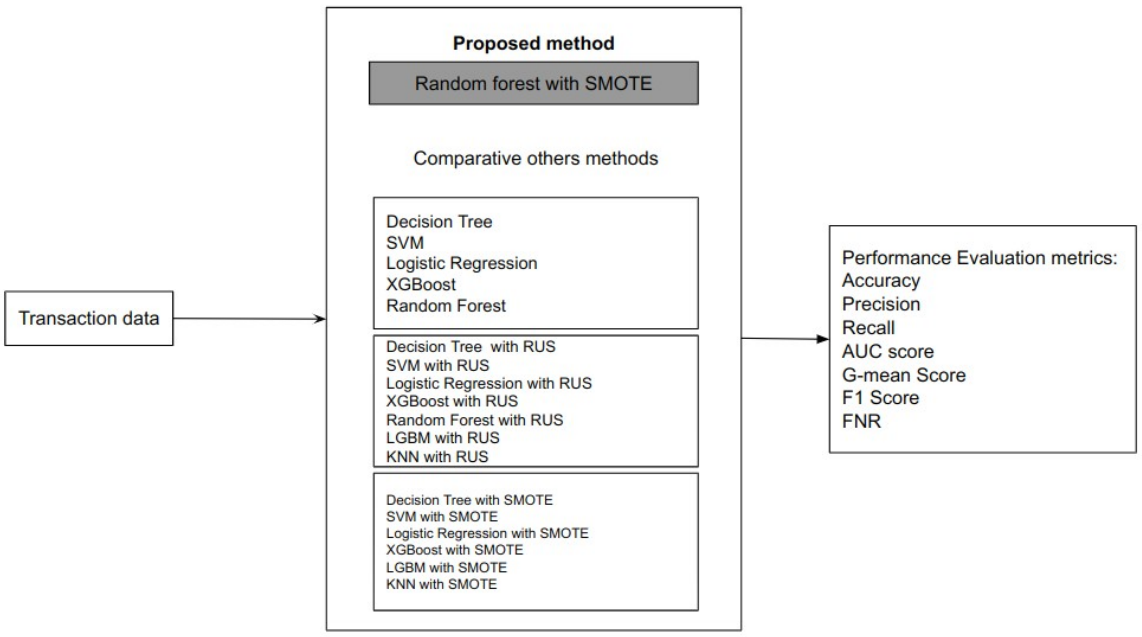


Figure 6: Proposed Fraud transaction detection technique

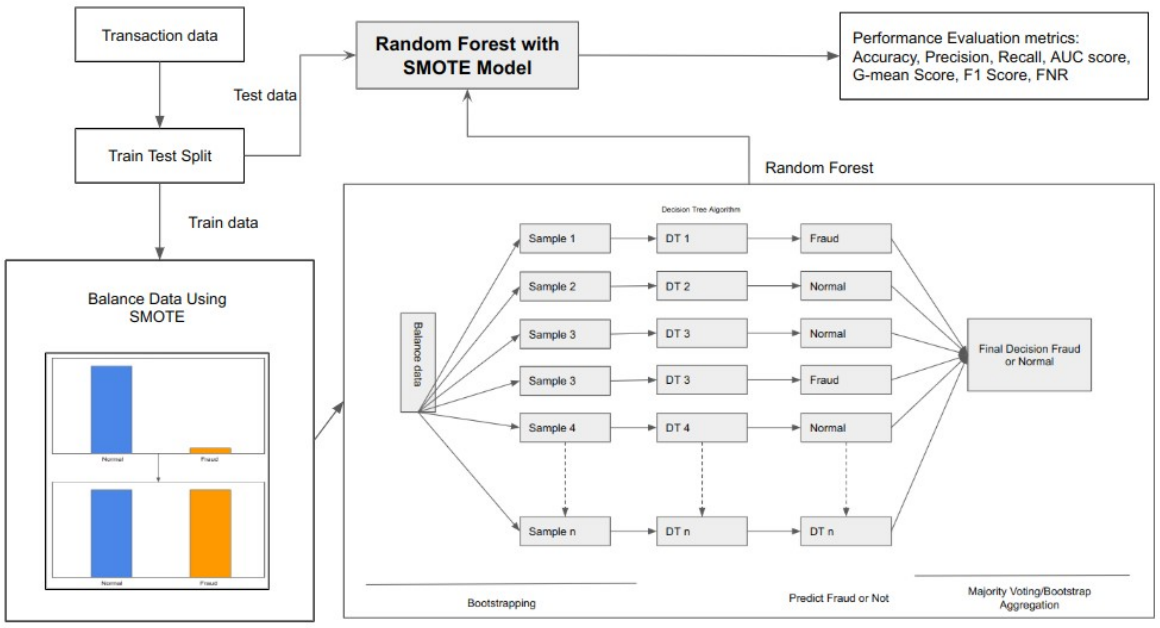


Figure 7: How the proposed fraud transaction detection technique works.

SMOTE

In order to overcome the issue of class imbalance in fraud transaction detection, notably in mobile payment systems, SMOTE (Synthetic Minority Over-sampling approach) is a data augmentation approach that is frequently utilized. To balance the dataset and enhance the effectiveness of classification models, it is particularly made to produce synthetic samples of the minority class (fraudulent transactions).

Here's an overview of how SMOTE works for fraud transaction detection:

- Finding the minority class in the dataset: The majority class in the dataset corresponds to genuine transactions, whereas the minority class in the dataset reflects fraudulent transactions.

- Choosing a minority sample: SMOTE chooses a minority sample at random from the dataset to serve as the basis for creating synthetic samples.

- Finding nearest neighbors: For the selected minority sample, SMOTE identifies its k nearest neighbors in the feature space. The value of k is a user-defined parameter.

- Creating synthetic samples:SMOTE creates synthetic samples by randomly choosing one of the k closest neighbors, and then constructing a new sample along the line segment that connects the original sample with the selected neighbor. The synthetic sample is a combination of the attributes of the original sample and the selected neighbor.

- Repeating the process: Steps 2-4 are repeated until the desired level of imbalance is achieved, or the dataset is sufficiently balanced.

SMOTE successfully improves the representation of the minority class in the dataset by creating fake samples. This lessens the bias toward the dominant class and gives the classification model more evenly distributed training data.

The classification model may learn from a more representative dataset by using SMOTE to detect fraud transactions in mobile payment systems, which can increase the model's capacity to detect fraud with greater accuracy. The SMOTE-generated synthetic samples capture the traits and patterns of fraudulent transactions, improving the model's capacity for fraud detection.

SMOTE, which ensures a more thorough and accurate examination of mobile payment data, is a useful approach for tackling the class imbalance problem in fraud transaction detection.

It benefits from the diversity of the individual trees and reduces the risk of overfitting.
Random Forest is an ensemble learning algorithm that combines multiple decision trees. Each tree is trained on a random subset of the data and features. The trees work together to make predictions, and their individual predictions are combined to obtain the final prediction. Random Forest is known for its ability to handle complex datasets, reduce overfitting, and provide robust predictions.

3.5 Comparative Machine learning methods

XGBoost

XGBoost (Extreme Gradient Boosting) is an ensemble machine learning algorithm used for classification and regression tasks. It is particularly effective in detecting fraudulent transactions.

Ensemble Technique: XGBoost is an ensemble technique, which means it combines multiple weak prediction models to create a stronger and more accurate model. In the case of XGBoost, these weak models are decision trees. Here is a simplified step-by-step process of how XGBoost works:

Working process of XGBoost:

Step 1. Initialization:

- Set the hyperparameters for the XGBoost model, such as the number of trees, learning rate, maximum depth of each tree, and regularization parameters.
- Define the objective function based on the specific problem (classification or regression).

Step 2. Initial Prediction:
- Assign an initial prediction value to all the training examples.
- For classification, this can be the log odds or probabilities of the classes.
- For regression, this can be the mean or median of the target variable.

Step 3. Residual Calculation:
- Calculate the residuals by taking the difference between the actual target values and the initial predictions.

Step 4. Decision Tree Construction:
- Build a decision tree by recursively partitioning the data based on feature values.
- Determine the best splits that minimize the loss function, which measures the discrepancy between the actual target values and the predicted values.

Step 5. Leaf Output Calculation:
- Calculate the output value for each leaf of the decision tree.
- For regression, this is typically the mean of the residuals within the leaf.
- For classification, this can be the log odds or probabilities of the class within the leaf.

Step 6. Update Predictions:
- Update the predictions for each training example by adding the output value of the corresponding leaf for each decision tree.
- This step incorporates the predictions from all the decision trees constructed so far.

Step 7. Update Residuals:

- Calculate the new residuals by taking the difference between the actual target values and the updated predictions.
- These new residuals become the target values for the next iteration.

Step 8. Repeat Steps 4-7:

- Iterate the process by constructing additional decision trees to further reduce the residuals.
- Each new decision tree is built to correct the mistakes made by the previous trees.
- The number of iterations depends on the specified number of trees or until a stopping criterion is met. The criterion can be a specified number of trees or when the residuals no longer show a significant reduction.

Step 9. Final Prediction:

- Use the ensemble of decision trees to make predictions on new, unseen data.
- For regression, the final prediction is the sum of the initial predictions and the predictions from all the decision trees.
- For classification, the final prediction can be the class with the highest probability or the class with the highest sum of log odds.

XGBoost starts by making an initial prediction and then constructs decision trees based on the residuals. Each tree aims to minimize the residuals by finding the best feature splits. The output values of the trees are used to update the residuals, and the process iterates to build a strong ensemble of trees. The final prediction is obtained by combining the outputs of all the trees with a learning rate.

Decision Tree

Decision Tree recursively splits the data based on the best features and values at each node to create a tree structure. The final predictions are made by traversing the tree from the root node to a leaf node based on the true/false questions at each node. Decision Trees are intuitive, easy to understand, and provide interpretable results. Here is a simplified step-by-step process of how Decision Treet works:

Step 1: Selecting the Root Node

- Start by selecting a feature that will become the root node of the decision tree.
- The goal is to choose the feature that best classifies or predicts the target variable.

Step 2: Splitting the Nodes

- Calculate the impurity of each feature to determine how well it separates the data into

different classes.
- Common methods to measure impurity are Gini impurity, entropy, and information gain.
- Select the feature with the least impurity as the node at the current level.

Step 3: Creating Child Nodes

- Repeat Steps 1 and 2 for each child node (internal node) until the data is fully classified.
- At each level, select the feature and value that minimize impurity to create child nodes.
- For categorical features, create child nodes for each unique value of the selected feature.
- For numerical features, select a value to split the data into two groups (e.g., less than or greater than the selected value).

Step 4: Assigning Leaf Nodes

- Once all the data is classified, assign leaf nodes as the final classification categories or real values.
- Each leaf node represents a specific class or value.

Step 5: Making Predictions

- To make a prediction for a new data point: Start at the root node and follow the conditions at each node based on the feature values.
- Traverse down the tree until reaching a leaf node.
- The leaf node provides the final classification or value prediction for the data point

LightGBM

LightGBM is a gradient-boosting ensemble method based on decision trees, used for classification and regression tasks. It is optimized for high performance with distributed systems. It uses a gradient-boosting approach with decision trees. It grows trees leaf-wise, processes data using histograms, and employs exclusive feature bundling to enhance performance. Additionally, it utilizes gradient-based one-side sampling (GOSS) to improve training efficiency. By iteratively adding trees and adjusting their predictions, LightGBM gradually improves the model's performance. Here is a simplified step-by-step process of how LightGBM works:

Step 1: Initialization

- Set hyperparameters, such as the learning rate, number of trees, and maximum tree depth.
- Initialize the model and define the objective function based on the problem type (classification or regression).

Step 2: Tree Construction

- Start with an empty ensemble of decision trees.
- For each iteration, a new decision tree is added to the ensemble to correct the errors made by the previous trees.

Step 3: Leaf-wise Tree Growth

- LightGBM grows the decision trees leaf-wise rather than level-wise.
- It selects the leaf with the highest gain (reduction in the objective function) to split at each step.
- This approach can sometimes lead to overfitting, so limiting the tree depth is important to avoid overfitting.

Step 4: Histogram-based Data Processing

- LightGBM uses a histogram-based method to process the data.
- Instead of considering each data point individually, it bins the data into histograms and operates on the bins.
- This approach improves the efficiency of the calculations, especially for sparse datasets.

Step 6: Exclusive Feature Bundling

- LightGBM performs exclusive feature bundling, which combines exclusive features to reduce dimensionality.
- By reducing the number of features, it makes the training process faster and more efficient.

Step 7: Gradient-based One Side Sampling (GOSS)

- GOSS is a sampling technique used in LightGBM.
- It assigns higher weights to data points with larger gradients, focusing on instances that have not been well trained.
- Data points with smaller gradients are randomly removed or retained to maintain accuracy.
- This sampling method improves the efficiency and effectiveness of the model.

Step 8: Ensemble Prediction

- Combine the predictions from all the trees in the ensemble to make the final prediction.
- For classification, it may use a voting or averaging scheme to determine the predicted class.
- For regression, it may take the average or weighted average of the predicted values.

Logistic Regression

Logistic regression is a supervised machine learning algorithm used for binary classification tasks. It models the relationship between input features and the probability of a binary outcome. It works by applying a Sigmoid function to a linear combination of the input features, which transforms the output into a probability between 0 and 1. By setting a decision threshold (typically 0.5), instances with probabilities above the threshold are classified as one class, while those below it are classified as the other class. The model is trained by minimizing the difference between the predicted probabilities and the actual class labels using optimization techniques like gradient descent. Here is a simplified step-by-step process of how Logistic Regression works:

Step 1. Initialize the model parameters

- Start by setting the initial values for the weights (w) and the bias (b).

Step 2. Compute the Sigmoid function

- Define the Sigmoid function, which takes a linear combination of the input features (X) multiplied by the weights (w) and added with the bias
- The Sigmoid function transforms this value into a probability between 0 and 1.

Step 3. Training the model:

- Forward pass: Calculate the predicted probabilities by applying the Sigmoid function to the linear combination.
- Compute the cost/loss: Measure the discrepancy between the predicted probabilities and the actual class labels (y) using the loss function. The commonly used loss function in logistic regression is the binary cross-entropy loss.
- Backward pass (Gradient Descent): Calculate the gradients of the weights (dw) and bias (db) with respect to the loss. This step involves finding the derivatives of the loss function with respect to the model parameters.
- Update parameters: Adjust the weights and biases by subtracting the learning rate multiplied by the gradients. This step aims to minimize the loss and improve the model's predictions.

Step 4. Repeat steps 2 and 3 for a specified number of iterations or until convergence.

Step 5.Prediction:

- Compute the linear combination of the input features (X_new) using the learned weights (w) and bias (b).
- Apply the Sigmoid function to obtain the predicted probability.
- Classify based on the decision boundary: If the predicted probability is greater than 0.5, assign the positive class label; otherwise, assign the negative class label.

SVM

SVM, which stands for Support Vector Machines, is a method used in machine learning to separate things into different groups. It looks at a set of examples that are already labeled and tries to find the best line or boundary to separate them. The goal is to have the biggest gap possible between the groups. SVM can also handle situations where things are not in a straight line by using tricks to transform the data. Once the separation line is found, SVM can then classify new things based on which side of the line they fall on.

To understand how SVM works, it's important to grasp some key terms and concepts. SVM is a machine learning algorithm used for classification tasks. It finds the best way to draw a line or plane that separates different classes of data points. Terms like support vectors, margin, and kernel function are used in SVM. The goal is to maximize the distance between the decision boundary and the nearest data points, and SVM achieves this by finding the optimal decision boundary that separates the classes. Once the boundary is established, new data points can be classified based on which side of the boundary they fall on.

1. Decision function: The decision function in SVM is used to determine the class of a data point based on its features. For a binary classification problem, it can be represented as

   f(x) = sign(w^T * x + b)

   where f(x) is the decision function, x represents the input features, w is the weight vector, b is the bias term and sign() is the sign function.

2. Margin: The margin is the distance between the decision boundary (hyperplane) and the support vectors. It is denoted as γ in mathematical equations.

3. Objective function: The objective of SVM is to find the hyperplane that maximizes the margin while minimizing the classification error. This can be formulated as an optimization problem, typically using the following objective function:

   min 1/2 * ||w||^2 + C * Σ(max(0, 1 - y_i * (w^T * x_i + b)))

   where ||w||^2 represents the squared norm of the weight vector, C is the regularization parameter, Σ represents the sum over all training examples, y_i is the class label of the i-th example, x_i is the feature vector of the i-th example, and b is the bias term.

4. Support vectors: Support vectors are the data points that lie closest to the decision boundary. They have non-zero values for the Lagrange multipliers (α_i) in the optimization problem. The support vectors influence the position and orientation of the decision boundary.

5. Kernel function: The kernel function allows SVM to operate in a higher-dimensional feature space without explicitly computing the transformation. It calculates the similarity between two data points in the original feature space or the transformed space. The commonly used kernel functions include the linear kernel, polynomial kernel, and radial basis function (RBF)

kernel.

Here is a simplified step-by-step process of how SVM works:

- Step 1. Start with a labeled dataset: SVM requires a dataset with labeled examples, where each example belongs to one of the two classes (binary classification).
- Step 2. Select the optimal hyperplane: SVM aims to find the best hyperplane that separates the two classes. This hyperplane should maximize the margin, which is the distance between the hyperplane and the nearest data points of each class.
- Step 3. Transform the data: If the data is not linearly separable in its original feature space, SVM can transform it into a higher-dimensional space using a technique called the kernel trick. This transformation helps to find a linear decision boundary in the transformed space.
- Step 4. Find the support vectors: Support vectors are the data points that are closest to the decision boundary. These points play a crucial role in determining the position and orientation of the hyperplane.
- Step 5. Optimize the hyperplane: SVM solves an optimization problem to find the optimal hyperplane. It aims to minimize the classification error while maximizing the margin. The optimization process involves finding the best values for the hyperplane parameters (weights and biases).
- Step 6. Handle non-linear boundaries: In cases where the classes are not linearly separable, SVM can use different types of kernels (e.g., polynomial, radial basis function) to create non-linear decision boundaries. These kernels allow SVM to handle complex patterns and improve classification accuracy.
- Step 7. Classify new data points: Once the optimal hyperplane is determined, SVM can classify new, unseen data points by checking which side of the hyperplane they fall on. Points on one side of the hyperplane belong to one class, while points on the other side belong to the other class

KNN

KNN classifier operates by finding the k nearest neighbors to a given data point, and it takes the majority vote to classify the data point. This is a machine-learning method for classification that aims to label new, unseen objects based on their similarity to known objects with given labels. It falls under the supervised learning category, meaning it requires training data with labeled examples. The basic idea is to assign the label of the closest sample object in the training set to the query object. KNN can be extended to consider a set of k nearest neighbors instead of just one, and the predicted label is determined through majority voting or weighted schemes based on distances or frequencies. Mean-based nearest neighbor classifiers work on the means of classes rather than individual data points. Here is a simplified step-by-step process of how KNN works:

- Step 1. Choose the number K of neighbors
    - How to select the number of K: calculate the error or performance for a different number of k and choose the best one.
- Step 2. Take the K nearest neighbors of the new data point, according to any distance.
    - Euclidean distance:

      Distance: $\sqrt{(\sum(x_i - y_i)^2)}$

    - Manhattan distance:

      Distance: $\sum|x_i - y_i|$

    - Minkowski Distance:

      Distance: $(\sum(|x_i - y_i|)^p)^{(1/p)}$

      The p parameter of the Minkowski Distance metric of SciPy represents the order of the norm. When the order(p) is 1, it will represent Manhattan Distance and when the order in the above formula is 2, it will represent Euclidean Distance. Xi and Yi are the coordinates.

    - Hamming Distance:

      Distance: $\sum(x_i \neq y_i)$

      Primarily used for categorical or binary data, the Hamming distance counts the positions where two vectors differ. It measures dissimilarity based on the number of mismatches.

- Step 3. Among the K neighbors, count the number of data points in each category.
- Step 4. Assign the new data point to the category where you counted the most neighbors.

Chapter 4: Requirements Analysis and Specification

4.1 Functional Requirements:

Functional requirements list the particular capabilities or characteristics that a system or piece of software must have in order to carry out its intended function. These specifications outline what the system **must do** and how it must act in response to various inputs and events.Functional specifications frequently include in-depth explanations of system interactions, data processing, user behaviors, and system responses. On the basis of the provided inputs or circumstances, they indicate the predicted outputs or outcomes.

For instance, some functional requirements for a mobile payment system fraud detection application may be:

Table 4: Functional Requirement

Functional requirement	**Description**	**Stackholder**	**Priority**
User authentication	Users must authenticate themselves before accessing.	Mobile transaction service provider	High
Transaction Sample Upload	Users need to upload the transaction sample in a specific format, such as CSV, to the system.	Mobile transaction service provider	High
Data preprocessing	The system should preprocess the uploaded transaction data by performing encoding, feature scaling, etc before go to the model for prediction.	Developer	High

Fraud detection	The system should apply a fraud detection algorithm or model to the preprocessed data to identify the fraudulent or legitimate transactions.	Developer	High
Result Genaration	The system should generate a report or output file (e.g., CSV) containing the prediction results for each transaction.	Developer	High
Download functionality	Users should be able to download the generated output file to access the prediction results.	Mobile transaction service provider	High

4.2 Non-functional Requirements:

The characteristics or features of a system that indicate how it should behave rather than what it **should behave** are referred to as non-functional requirements. Performance, dependability, usability, security, maintainability, and other factors that affect the system's overall quality are the major emphasis of these criteria.

Non-functional requirements must be accurate and quantitative since they serve as the standards by which the system's efficacy is measured. How to formulate non-functional needs is demonstrated by the following:

1. Performance:

   - Response Time: For user interactions, the system should have a response time of under 2 seconds.
   - Throughput: The system must be able to handle at least 10,000 transactions per second.

2. Reliability:

    - Availability: With room for maintenance and updates, the system should operate at least 99% of the time each month.
    - Error Handling: The system must handle problems politely and give users clear error messages.

3. Usability:

    - User Interface: The system should have an easy-to-use interface that only takes a brief amount of training to allow users to navigate and complete activities.
    - Accessibility: The program should follow accessibility guidelines to guarantee that people with disabilities can use it.

4. Security:

    - Authentication and Authorization: To confirm user identities and limit access to authorized users exclusively, the system should utilize secure authentication procedures.
    - Data Encryption: To prevent unwanted access to sensitive user data during transmission and storage, the system should encrypt the data.

5. Maintainability:

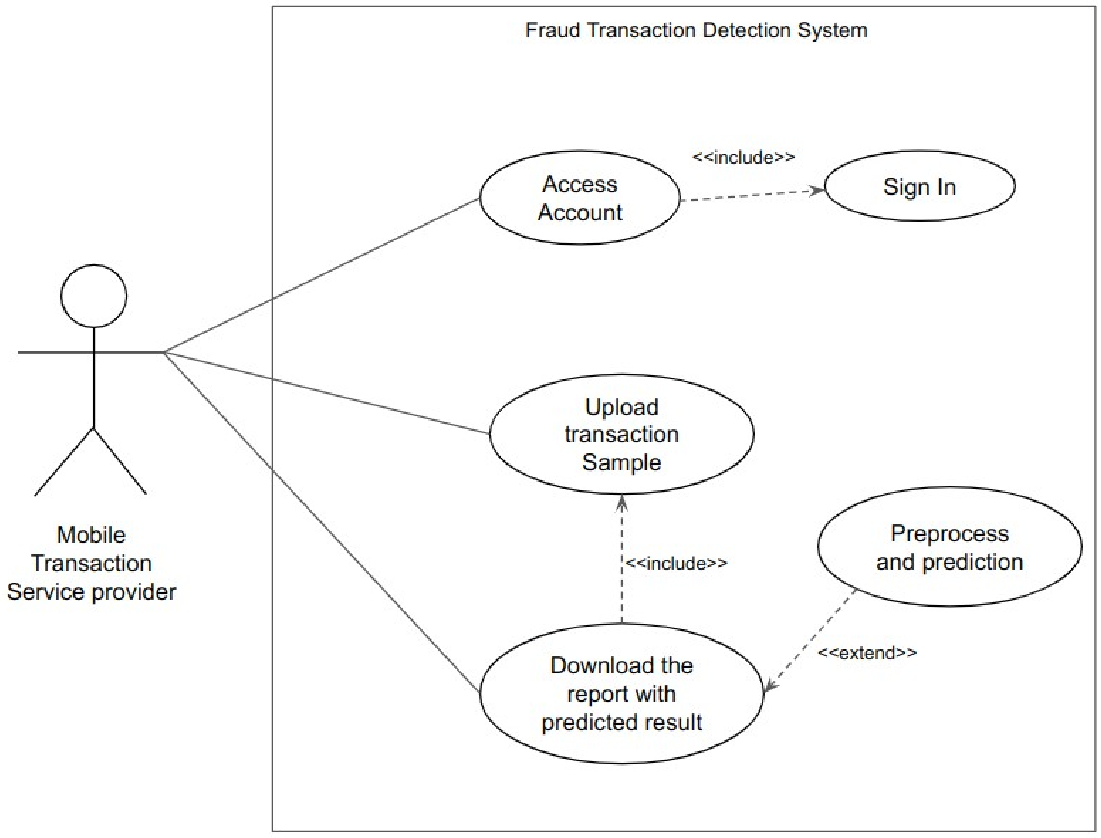
    - Modularity: The architecture of the system should be modular in order to provide simple maintenance and future improvements without affecting other parts.
    - Code Documentation: The system's code has to be well-documented to make it easier for other developers to understand and maintain it.

6. Scalability:

    - Concurrent Users: The system must be able to handle at least 1,000 concurrent users without noticeably degrading its performance.
    - Data Volume: Without sacrificing performance, the system should be able to manage databases with up to 1 million transactions.

4.3 Use case diagram

A use case diagram consists of actors and use cases, where actors can conduct one or more instances. Additionally, it demonstrates which actors have access to each use case. This use case graphic shows how mobile payment systems may identify fraudulent transactions.



Fraud Transaction Detection System
Access Account
<<include>>
Sign In
Upload transaction Sample
<<include>>
Preprocess and prediction
<<extend>>
Download the report with predicted result
Mobile Transaction Service provider

Figure 8: Use case Diagram

4.4 Use Cases and Scenarios

Use Case 1: User Uploads Transaction Sample

- Scenario: The registered user logs in, selects the option to upload a transaction sample, and provides the CSV file containing the transactions.
- Expected Outcome: The application receives and stores the uploaded transaction sample securely.

Use Case 2: Preprocessing and Prediction

- Scenario: The uploaded transaction sample is preprocessed to clean and prepare the data. The machine learning model predicts the fraud status of each transaction.
- Expected Outcome: The application successfully processes the data, applies the model for prediction, and generates the fraud prediction results.

Use Case 3: Downloading Prediction Results

- Scenario: The registered user requests to download the CSV file with the prediction results.
- Expected Outcome: The application provides a downloadable CSV file containing the fraud prediction results.

Chapter 5: System Design and Architecture

5.1 System Architecture Overview

The system architecture follows a simple and intuitive flow. Users upload their transaction samples, which go through preprocessing and are then fed into the deployed machine learning model for prediction. The model generates a CSV file with the prediction results, which users can download and utilize for their specific purposes. This architecture enables efficient and convenient fraud transaction detection for mobile transfer service providers.

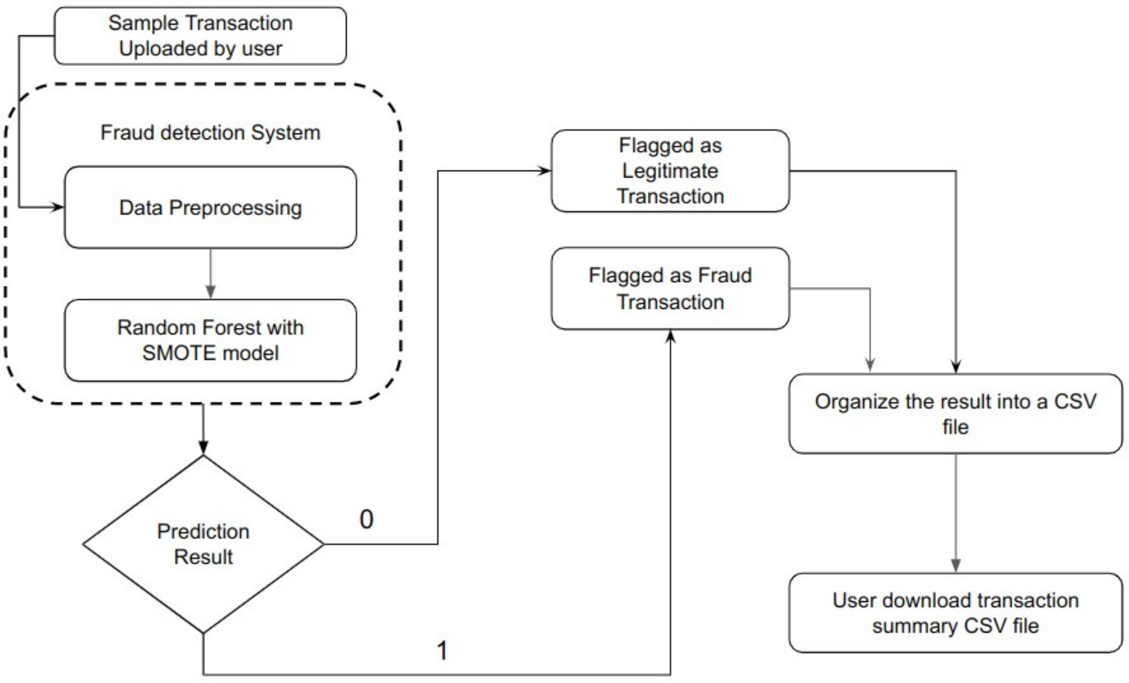

Sample Transaction Uploaded by user
Fraud detection System
Data Preprocessing
Random Forest with SMOTE model
Prediction Result
0
1
Flagged as Legitimate Transaction
Flagged as Fraud Transaction
Organize the result into a CSV file
User download transaction summary CSV file

Figure 9: System Architecture Overview

5.2 High-Level Design

In the development process, high-level design acts as a road map for the phases of detailed design and execution. It aids stakeholders in comprehending the general structure and operation of the system, including developers, project managers, and clients.

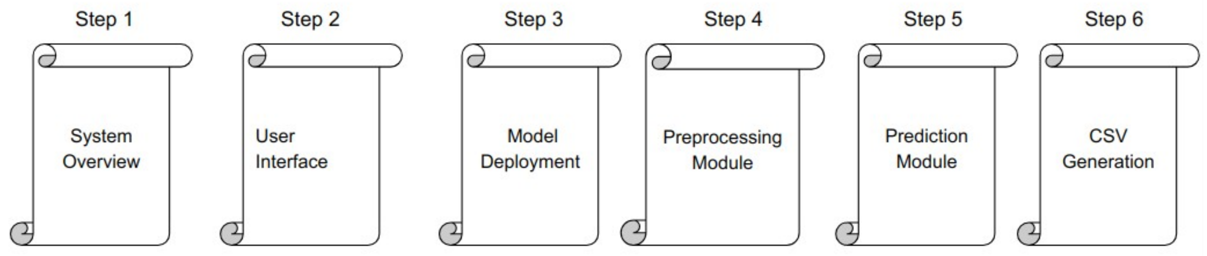


Figure 10: High level design

Step 1: System Overview : The system has a clear and user-friendly flow. The suppliers of mobile transfer services submit their transaction samples, which go through preprocessing procedures to guarantee the accuracy of the data. The deployed machine learning model then receives these preprocessed data and makes a prediction. The model evaluates the data and produces a CSV file with the outcome of the prediction. Users may easily download this CSV, giving them the information they need to decide how to handle transactions that have been flagged as fraudulent. For the advantage of service providers, this design places a strong emphasis on efficiency and effectiveness in identifying and combating fraud in mobile payment systems.

Step 2: User Interface:The system requires users to log in before they may utilize it. Users who don't already have accounts can create them. The system has a video that shows how it functions after logging in. With the help of this, the user may quickly comprehend how to utilize it. By clicking on upload your transaction file here, a user may normally upload their transaction file. Additionally, you may download the prediction report's summary in csv file.

Step 3: Model Deployment: To minimize data redundancy or data leakage, first divide the data into train tests and balance the test data using SMOTE. After that, use balanced data to train the Random forest model, then test the model with test data. In the end, integrate the model into the system.

Step 4: Preprocessing Module: When user click “Download the transaction summary”, the user-uploaded transaction sample data are processed and predictions are made. Encoding is a sort of data transformation, and preprocessing involves feature scaling.

Step 5: Prediction Module: After entering a sample process, the model predicts which transaction is fraudulent and flags it accordingly.

Step 6: CSV Generation: Make a CSV file and add a column called "Prediction Result" with the value determined by model judgment for each transaction.

5.3 Detailed Design

The process of enhancing the high-level design and giving a more thorough and complete definition of the system or software application is known as detail design, often referred to as low-level design. It entails converting the general concept into precise implementation specifics, such as the design of particular components, algorithms, data structures, and interfaces. Detail design focuses on how the system functions internally and offers a step-by-step blueprint for constructing the system. It takes into account things like the programming languages, frameworks, libraries, and particular technologies to be employed.

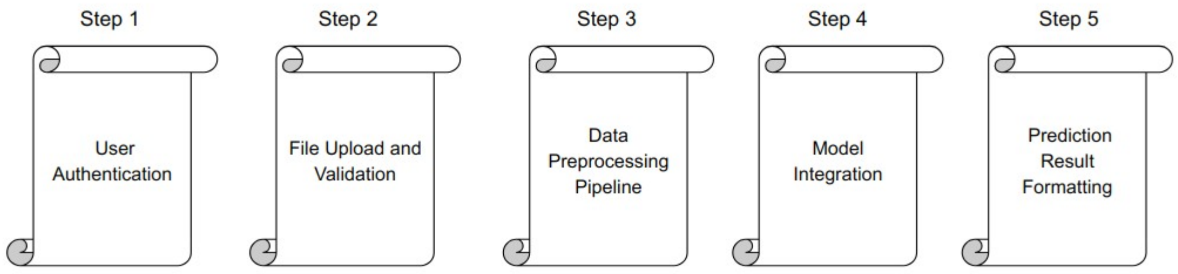

Step 1
User Authentication
Step 2
File Upload and Validation
Step 3
Data Preprocessing Pipeline
Step 4
Model Integration
Step 5
Prediction Result Formatting

Figure 11: Details Design

Step 1: User Authentication: The system allows only registered and authenticated users to access its functionalities. Users must log in with valid credentials to gain access.

Step 2: File Upload and Validation: Users can upload their transaction sample files by clicking on the "Upload your transaction here" button. The system requires the transaction sample file to be in CSV format. Without a valid CSV file upload, the system will not process the data.

Step 3: Data Preprocessing Pipeline: The uploaded transaction samples undergo a data preprocessing pipeline before being passed to the model. This preprocessing includes encoding categorical variables and applying feature scaling to ensure the data is suitable for analysis.

Step 4: Model Integration: The system utilizes a combined technique of Random Forest and SMOTE for fraud detection. This model has been integrated into the system. The preprocessed transaction samples are fed into the model for prediction.

Step 5: Prediction Result Formatting: The model analyzes each transaction and determines whether it is classified as a normal or fraudulent transaction. The system flags each transaction accordingly and creates a CSV file that contains the prediction results. Users can download this CSV file to view the predictions made by the system.

Chapter 6: Implementation and Development

6.1 Programming Languages and Technologies Used

The main programming language used to implement the application and model was Python. Python offers a flexible and effective platform for system development. Also used in the construction of the program was the Django framework, which facilitated efficient web development. Using the well-known CSS framework Bootstrap, which guarantees a responsive and user-friendly layout, the front-end interface was created. There were also used a number of Python libraries, including scikit-learn, imbalanced-learn, lightgbm, numpy, and pandas. These libraries provided crucial modeling, analytical, and data processing functions. The main components of the development environment were Google Colab, Kaggle, and Jupyter Notebook, which assisted in model construction, experimentation, and coding.

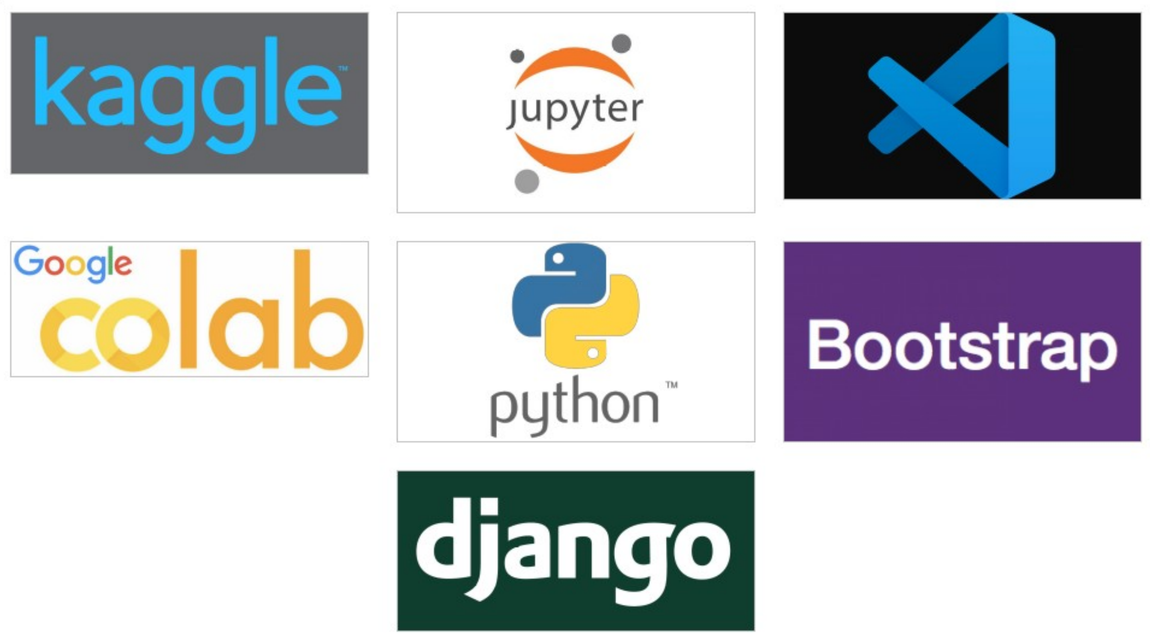


Figure 12: Tools

6.2 Implementation Details

There were several parts and phases involved in the application's implementation. Through the application's user interface, users who are mobile transfer service providers may upload transaction examples in CSV format. The system then underwent preprocessing on the uploaded samples to get them ready for additional analysis. The deployed model received the preprocessed data and used them to make predictions about which transactions were fraudulent and which were valid. Then, the system created the prediction findings and saved them in a CSV file format that consumers could download. The solution used the Django framework to manage workflows for file management, user authentication, and data processing. The user experience was improved by using Bootstrap to produce a simple and eye-catching user interface.

6.3 Testing and Debugging Approaches

In order to guarantee the operation and dependability of the designed system, testing, and debugging were extremely important. A sample of test data from the original PaySim dataset was randomly chosen in order to assess the system's performance. The test data covered a range of situations, including fraud and honest transactions. These test examples were entered into the system, and the predictions were then contrasted with the known results to evaluate the model's efficacy. The performance and dependability of the system were enhanced by carefully examining and debugging any anomalies or problems that surfaced during testing. The system offered accurate predictions and a flawless user experience thanks to this iterative testing and debugging approach, which helped discover and correct any flaws or anomalies.

Chapter 7: Results Analysis and Discussion

7.1 Data Collection and Analysis

In accordance with the majority of prior research (Buschjäger and Honysz 2021); Du et al., 2018; Xenopoulos, 2017), we employed the PaySim dataset for our study. Lopez-Rojas and their research team (Lopez-Rojas et al., 2016, 2018; Lopez-Rojas & Barneaud, 2019) conducted simulations with the primary goal of replicating common fraud scenarios that possess statistical characteristics similar to the original mobile payment transaction data. In order to achieve this objective, various forms of fraudulent transactions were introduced, encompassing activities such as cash-in (boosting account balance), cash-out (withdrawing cash), payment (making payments for goods or services), transfer (transferring funds to another user), and debit (sending money to a bank account).PaySim simulated 743-time steps, representing thirty days of real-time data. The dataset consisted of a total of 6,362,620 mobile transactions, out of which 8,213 were identified as fraudulent. Here is the description of the dataset.

Table 5: Dataset Summary

Feature	Description
step	Unit of time where 1 step equals 1 hour. Total steps 744 (30 days simulation)
type	CASH-IN, CASH-OUT, DEBIT, PAYMENT, and TRANSFER
amount	Amount of the transaction in local currency (180K)
nameOrgin	Customer who started the transaction (6.35M unique values)
oldbalanceOrg	Initial balance before the transaction (834K)
newbalanceOrig	New balance after the transaction (855K)
nameDest	Customer who is the recipient of the transaction (2.72M unique values)
oldbalanceDest	initial balance recipient before the transaction (1.1M)
newbalanceDest	new balance recipient after the transaction (1.22M)
isFraud	This is the transactions made by the fraudulent agents inside the simulation. 0 (legitimate 6.36M) / 1 (fraud 8.2K)

Source: *Synthetic Financial Datasets For Fraud Detection*

7.2 Evaluation Metrics and Criteria

The act of measuring the effectiveness and efficiency of a system, model, algorithm, or process in attaining its desired objectives or aims is referred to as performance evaluation. It entails assessing multiple performance measurements and indicators to determine the success, accuracy, robustness, and overall quality of the system or process being evaluated. By reviewing these metrics, one can assess how effectively the system or process meets its intended objectives and make educated decisions based on the evaluation results.

The following steps are frequently included in the performance evaluation process:

1. Selecting appropriate evaluation metrics: Choosing the proper performance indicators for an evaluation depends on the goals and specifications of the system or process being assessed.

2. Collecting representative data: To provide a thorough evaluation, it is important to collect a broad and representative dataset that includes a variety of scenarios and variations.

3. Preprocessing and preparing the data: Cleaning, normalizing, and transforming the data to ensure its quality and compatibility with the evaluation process.

4. Applying the system or model to the data: Implementing the system or model on the dataset and obtaining predictions or results.

5. Computing performance metrics: Calculating various performance metrics based on the predictions or results obtained, such as accuracy, precision, recall, F1-score, and AUC, G-mean score, FNR.

6. Analyzing the results: interpreting the performance indicators and assessing the system's or model's advantages, disadvantages, and constraints in light of the evaluation's findings.

7. Iterating and improving: Determining areas for improvement and modifying the system, model, or procedure in light of the evaluation's findings.

Several previous studies (Du et al., 2018; Misra et al., 2020; Mubalaike & Adali, 2018) have commonly utilized the accuracy metric, which represents the ratio of correctly classified transactions to the total number of transactions, as an evaluation measure. However, when dealing with imbalanced data where the minority class (fraudulent transactions) is underrepresented, accuracy may not accurately reflect the model's performance for the minority class. As highlighted in prior studies (Lopez-Rojas & Barneaud, 2019), an inherent challenge in detecting financial fraud is the uncertainty surrounding the distribution and impact of all fraudulent transactions. In the absence of a comprehensive metric for fraud detection performance, existing approaches often resort to conventional measures of classification performance. The most desirable performance measure is the ability to accurately identify fraudulent transactions, known as the true positive rate(Recall).

Performance evaluation in the context of fraud transaction detection tries to evaluate the capability of a fraud detection system to precisely identify and categorize fraudulent transactions. It entails the evaluation of a variety of parameters, including false negative rate, F1-score, G-mean score, recall, accuracy, precision, and area under the receiver operating characteristic curve (AUC).

1. Accuracy: It measures the overall correctness of the model's predictions by calculating the proportion of correctly classified transactions (both fraudulent and legitimate) out of the total number of transactions. The calculation for accuracy is as follows:

Accuracy = (TP + TN) / (TP + TN + FP + FN)

where TP denotes true positives (fraudulent transactions correctly identified), TN denotes true negatives (legitimate transactions correctly identified), FP denotes false positives (legitimate transactions incorrectly classified as fraudulent), and FN denotes false negatives (fraudulent transactions incorrectly classified as legitimate).

2. F1 Measure: Previous studies have also considered the F1 measure (Pambudi et al., 2019; Schlör et al. 2021), defined as the harmonic mean of precision and recall, providing a single metric that balances both metrics. It is particularly useful when precision and recall need to be considered together. The calculation for the F1 measure is as follows:

   F1 = 2 * (Precision * Recall) / (Precision + Recall)

   where Precision is calculated as TP / (TP + FP) and Recall is calculated as TP / (TP + FN).

   By considering the accuracy, a model's overall correctness can be evaluated, which is important for fraud detection systems. The F1 measure provides a balanced evaluation of precision and recall, ensuring that the model not only identifies fraudulent transactions accurately but also minimizes misclassifications of legitimate transactions as fraudulent

3. False Negative Rate (FNR): The false negative rate indicates the proportion of fraudulent transactions that are incorrectly identified as legitimate. It measures the failure of the fraud detection model to correctly classify fraudulent transactions.

   Our primary focus will be on minimizing the "False Negative Rate" in our fraud detection efforts. While the false positive rate is also important, the risk associated with false negatives is higher for financial transactions. If a transaction is actually fraudulent but the model incorrectly predicts it as not fraudulent, it poses a significant risk. The calculation for FNR is as follows:

   FNR = FN / (TP + FN)

   Alternatively, FNR can be expressed as 1 - Recall.

4. Precision: Measures the proportion of correctly identified fraudulent transactions out of all transactions flagged as fraudulent by the model. It quantifies the model's ability to avoid misclassifying legitimate transactions as fraudulent. The calculation for Precision is as follows:

   Precision = TP / (TP + FP)

5. Area Under the Receiver Operating Characteristic Curve (AUC): It represents the model's ability to rank fraudulent transactions higher than legitimate ones across different

classification thresholds. A higher AUC value indicates a better discriminatory power of the model. The calculation for AUC involves integrating the True Positive Rate (Recall) and False Positive Rate (FPR) curve, and it is typically obtained using numerical methods or built-in functions provided by libraries.

6. G-mean score: The geometric mean (G-mean) is the root of the product of class-wise sensitivity. This measure tries to maximize the accuracy of each of the classes while keeping these accuracies balanced. For binary classification, G-mean is the squared root of the product of the sensitivity and specificity.

   specificity = TN / (TN + FP)
   sensitivity = TP / (TP + FN)
   gmean = np.sqrt(sensitivity * specificity)

7. True Positive Rate (Recall): The true positive rate, also known as Recall, measures the proportion of correctly identified fraudulent transactions out of all actual fraudulent transactions. It is calculated as

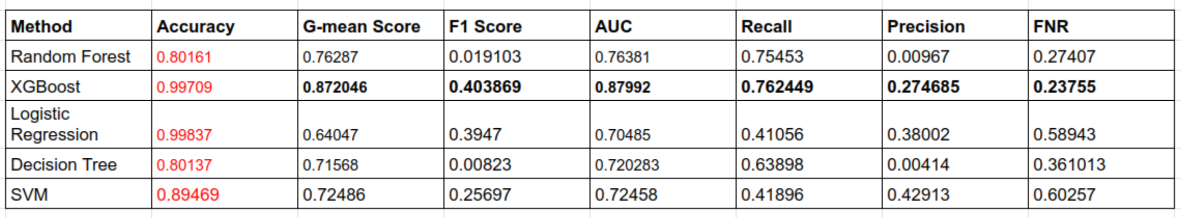
   Recall = TP / (TP + FN)

   where TP represents true positive (correctly identified fraudulent transactions) and FN represents false negative (fraudulent transactions misclassified as legitimate).

7.3 Interpretation of Results

7.3.1 Without balancing the dataset

In our study, we experimented with three different approaches to fraud detection. The first approach involved applying machine learning algorithms to the unbalanced dataset without any data balancing techniques. Interestingly, the XGBoost algorithm yielded promising results in this approach, demonstrating good performance in detecting fraudulent transactions.



Method	Accuracy	G-mean Score	F1 Score	AUC	Recall	Precision	FNR
Random Forest	0.80161	0.76287	0.019103	0.76381	0.75453	0.00967	0.27407
XGBoost	0.99709	**0.872046**	**0.403869**	**0.87992**	**0.762449**	**0.274685**	**0.23755**
Logistic Regression	0.99837	0.64047	0.3947	0.70485	0.41056	0.38002	0.58943
Decision Tree	0.80137	0.71568	0.00823	0.720283	0.63898	0.00414	0.361013
SVM	0.89469	0.72486	0.25697	0.72458	0.41896	0.42913	0.60257

Figure 13: Without balance dataset performance evaluation

The best results are in bold and skip accuracy is in red color.

Skip Accuracy: In the case of extremely imbalanced (He and Ma 2013) data, where the majority class (non-fraudulent transactions) significantly outweighs the minority class (fraudulent transactions), accuracy is not a suitable measurement for evaluating the performance of a fraud detection model. It doesn't make any sense.

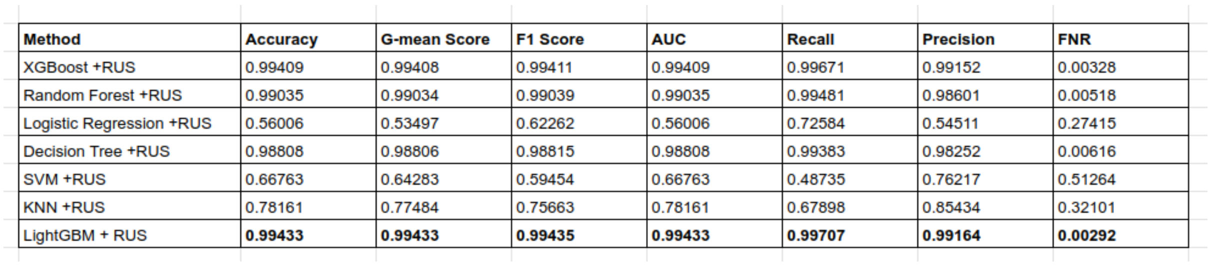
The reason is that accurately calculates the proportion of correctly classified instances (both fraud and non-fraud) out of the total instances. In highly imbalanced datasets, where the majority class dominates, a classifier that simply labels all transactions as non-fraudulent would achieve a high accuracy because it correctly predicts the majority class. However, such a model would fail to identify the minority class (fraudulent transactions) effectively.

Using accuracy alone as an evaluation metric can be misleading, as it does not provide insights into how well the model performs in detecting actual fraud cases. It tends to be biased towards the majority class and may lead to an inaccurate assessment of the model's effectiveness in identifying fraudulent transactions.

To overcome this issue, evaluation considerations for imbalanced datasets, such as precision, recall, F1-score, or area under the Receiver Operating Characteristic (ROC) curve utilized. These metrics provide a more comprehensive and reliable assessment of the model's performance, particularly in detecting the minority class (fraudulent transactions) accurately.

7.3.2 Balance the data using Random Undersampling

In the second approach, our first step was to balance the data using Random Under-Sampling (RUS). Once the data was balanced, we applied various machine-learning methods to detect fraudulent transactions. Among these methods, the combination of LightGBM (LGBM) and RUS demonstrated superior performance compared to other techniques. This finding suggests that utilizing LGBM with RUS resulted in improved accuracy in identifying fraudulent transactions.

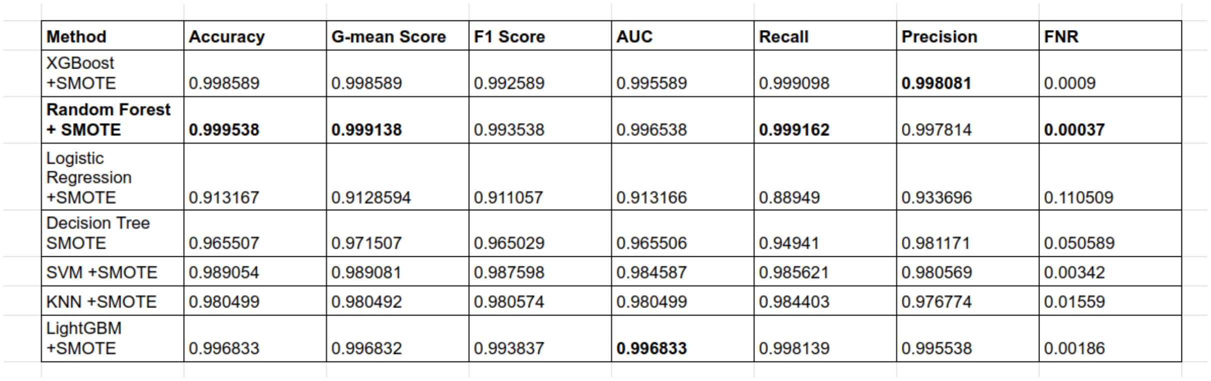


Method	**Accuracy**	**G-mean Score**	**F1 Score**	**AUC**	**Recall**	**Precision**	**FNR**
XGBoost +RUS	0.99409	0.99408	0.99411	0.99409	0.99671	0.99152	0.00328
Random Forest +RUS	0.99035	0.99034	0.99039	0.99035	0.99481	0.98601	0.00518
Logistic Regression +RUS	0.56006	0.53497	0.62262	0.56006	0.72584	0.54511	0.27415
Decision Tree +RUS	0.98808	0.98806	0.98815	0.98808	0.99383	0.98252	0.00616
SVM +RUS	0.66763	0.64283	0.59454	0.66763	0.48735	0.76217	0.51264
KNN +RUS	0.78161	0.77484	0.75663	0.78161	0.67898	0.85434	0.32101
LightGBM + RUS	**0.99433**	**0.99433**	**0.99435**	**0.99433**	**0.99707**	**0.99164**	**0.00292**

Figure 14: Balanced (RUS) Dataset performance evaluation (Best results are in bold)

7.3.3 Balance the data using SMOTE

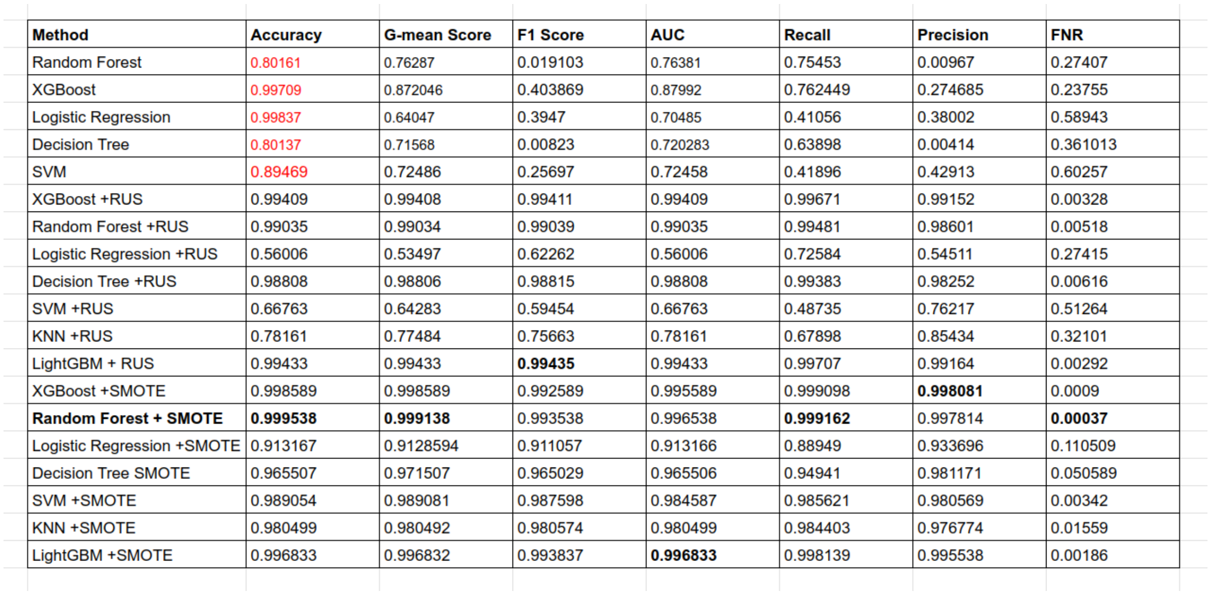
In the third approach, we employed Synthetic Minority Over-sampling Technique (SMOTE) to balance the data before applying various machine-learning methods. Notably, the Random Forest algorithm, when combined with SMOTE, yielded the best results compared to other methods. This finding suggests that utilizing Random Forest with SMOTE was particularly effective in detecting fraudulent transactions and outperformed other machine learning techniques in terms of performance.



Method	Accuracy	G-mean Score	F1 Score	AUC	Recall	Precision	FNR
XGBoost +SMOTE	0.998589	0.998589	0.992589	0.995589	0.999098	**0.998081**	0.0009
Random Forest + SMOTE	**0.999538**	**0.999138**	0.993538	0.996538	**0.999162**	0.997814	**0.00037**
Logistic Regression +SMOTE	0.913167	0.9128594	0.911057	0.913166	0.88949	0.933696	0.110509
Decision Tree SMOTE	0.965507	0.971507	0.965029	0.965506	0.94941	0.981171	0.050589
SVM +SMOTE	0.989054	0.989081	0.987598	0.984587	0.985621	0.980569	0.00342
KNN +SMOTE	0.980499	0.980492	0.980574	0.980499	0.984403	0.976774	0.01559
LightGBM +SMOTE	0.996833	0.996832	0.993837	**0.996833**	0.998139	0.995538	0.00186

Figure 15: Balanced (SMOTE) Dataset performance evaluation (Best results are in bold.)

We achieved performance metrics highlighting the effectiveness of this approach. The model demonstrated a high accuracy of 99.9538%, indicating a strong overall classification performance. Additionally, the G-mean score 99.9138% and F1 score also reached 99.3538%, reflecting the balanced performance of the model in capturing both the minority class (fraudulent transactions) and the majority class (non-fraudulent transactions). The AUC (Area Under the Curve) score of 99.6538% further confirms the model's excellent ability to distinguish between the two classes. The recall rate of 99.9162% indicates a high proportion of correctly identified fraudulent transactions, while the precision rate of 99.7814% signifies the model's capability in correctly classifying positive instances. Notably, the false negative rate was impressively low at 00.0037%, suggesting that the model effectively minimized the instances where fraudulent transactions were wrongly classified as non-fraudulent. Overall, these outstanding performance metrics establish Random Forest with SMOTE as the preferred approach for fraud detection in this study.



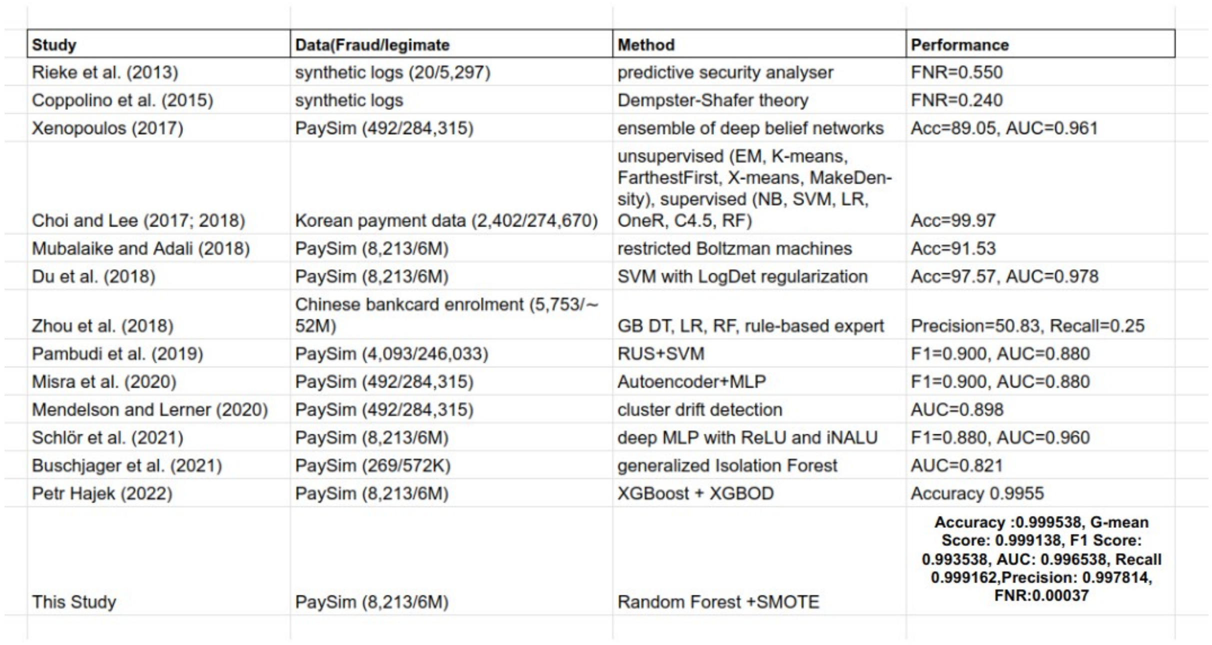
Method	Accuracy	G-mean Score	F1 Score	AUC	Recall	Precision	FNR
Random Forest	0.80161	0.76287	0.019103	0.76381	0.75453	0.00967	0.27407
XGBoost	0.99709	0.872046	0.403869	0.87992	0.762449	0.274685	0.23755
Logistic Regression	0.99837	0.64047	0.3947	0.70485	0.41056	0.38002	0.58943
Decision Tree	0.80137	0.71568	0.00823	0.720283	0.63898	0.00414	0.361013
SVM	0.89469	0.72486	0.25697	0.72458	0.41896	0.42913	0.60257
XGBoost +RUS	0.99409	0.99408	0.99411	0.99409	0.99671	0.99152	0.00328
Random Forest +RUS	0.99035	0.99034	0.99039	0.99035	0.99481	0.98601	0.00518
Logistic Regression +RUS	0.56006	0.53497	0.62262	0.56006	0.72584	0.54511	0.27415
Decision Tree +RUS	0.98808	0.98806	0.98815	0.98808	0.99383	0.98252	0.00616
SVM +RUS	0.66763	0.64283	0.59454	0.66763	0.48735	0.76217	0.51264
KNN +RUS	0.78161	0.77484	0.75663	0.78161	0.67898	0.85434	0.32101
LightGBM + RUS	0.99433	0.99433	**0.99435**	0.99433	0.99707	0.99164	0.00292
XGBoost +SMOTE	0.998589	0.998589	0.992589	0.995589	0.999098	**0.998081**	0.0009
Random Forest + SMOTE	**0.999538**	**0.999138**	0.993538	0.996538	**0.999162**	0.997814	**0.00037**
Logistic Regression +SMOTE	0.913167	0.9128594	0.911057	0.913166	0.88949	0.933696	0.110509
Decision Tree SMOTE	0.965507	0.971507	0.965029	0.965506	0.94941	0.981171	0.050589
SVM +SMOTE	0.989054	0.989081	0.987598	0.984587	0.985621	0.980569	0.00342
KNN +SMOTE	0.980499	0.980492	0.980574	0.980499	0.984403	0.976774	0.01559
LightGBM +SMOTE	0.996833	0.996832	0.993837	**0.996833**	0.998139	0.995538	0.00186

Figure 16: Overall performance evaluation

Overall, after considering multiple approaches for fraud detection, it can be concluded that the Random Forest algorithm in conjunction with SMOTE (Synthetic Minority Over-sampling Technique) emerges as the most effective and optimal approach. This combination proved to be highly successful in addressing the challenges posed by imbalanced data and achieving superior performance in detecting fraudulent transactions. Hence, Random Forest with SMOTE can be considered the best approach among the ones evaluated in this study.

All experiments were conducted using the Google Colab, Kaggle and Jupyter Notebook platforms. These platforms provide a convenient and efficient environment for running machine-learning experiments and analyzing the results. By leveraging the computational resources and collaborative features of Google Colab and Kaggle, we were able to carry out the experiments smoothly and effectively, facilitating the implementation and evaluation of different approaches for fraud detection.

7.4 Comparison with Related Work



Study	Data(Fraud/legimate	Method	Performance
Rieke et al. (2013)	synthetic logs (20/5,297)	predictive security analyser	FNR=0.550
Coppolino et al. (2015)	synthetic logs	Dempster-Shafer theory	FNR=0.240
Xenopoulos (2017)	PaySim (492/284,315)	ensemble of deep belief networks	Acc=89.05, AUC=0.961
Choi and Lee (2017; 2018)	Korean payment data (2,402/274,670)	unsupervised (EM, K-means, FarthestFirst, X-means, MakeDen-sity), supervised (NB, SVM, LR, OneR, C4.5, RF)	Acc=99.97
Mubalaike and Adali (2018)	PaySim (8,213/6M)	restricted Boltzman machines	Acc=91.53
Du et al. (2018)	PaySim (8,213/6M)	SVM with LogDet regularization	Acc=97.57, AUC=0.978
Zhou et al. (2018)	Chinese bankcard enrolment (5,753/~ 52M)	GB DT, LR, RF, rule-based expert	Precision=50.83, Recall=0.25
Pambudi et al. (2019)	PaySim (4,093/246,033)	RUS+SVM	F1=0.900, AUC=0.880
Misra et al. (2020)	PaySim (492/284,315)	Autoencoder+MLP	F1=0.900, AUC=0.880
Mendelson and Lerner (2020)	PaySim (492/284,315)	cluster drift detection	AUC=0.898
Schlör et al. (2021)	PaySim (8,213/6M)	deep MLP with ReLU and iNALU	F1=0.880, AUC=0.960
Buschjager et al. (2021)	PaySim (269/572K)	generalized Isolation Forest	AUC=0.821
Petr Hajek (2022)	PaySim (8,213/6M)	XGBoost + XGBOD	Accuracy 0.9955
This Study	PaySim (8,213/6M)	Random Forest +SMOTE	**Accuracy :0.999538, G-mean Score: 0.999138, F1 Score: 0.993538, AUC: 0.996538, Recall 0.999162,Precision: 0.997814, FNR:0.00037**

Figure 17: Comparison with related work

7.5 Discussion of Findings

The main goals of the discussion of findings section were to offer a thorough analysis and explanation of the study's major findings and conclusions. It involves a thorough analysis of each strategy's effectiveness, noting its advantages and disadvantages. The effectiveness of the suggested Random Forest with SMOTE technique in identifying fraudulent transactions was explicitly assessed. Also addressed were any noteworthy trends or patterns found during the investigation. An in-depth discussion of the relevance and importance of the findings in the context of mobile payment systems and fraud detection in the public transportation industry also helped to clarify the possible effect and practical ramifications of the study's findings.

Chapter 8: Conclusion and Future Work

8.1 Summary of Contributions

This research study aimed to address the financial impact of fraud detection in mobile payment systems by proposing a novel approach using Random Forest with SMOTE (Synthetic Minority Over-sampling Technique). The key contributions of this study are as follows:

Introducing a Combined Approach: We proposed the combination of Random Forest with SMOTE to effectively handle the challenge of extreme class imbalance and mitigate over-fitting issues in fraud detection. By integrating SMOTE, we were able to generate synthetic samples and balance the dataset, which improved the performance of the Random Forest classifier.

Comparative Analysis: Through a comprehensive comparative analysis against various state-of-the-art machine learning methods, our proposed Random Forest with SMOTE-based approach demonstrated cutting-edge performance in detecting fraudulent activities within mobile payment systems. The results highlight the superiority of ensemble methods over individual machine-learning techniques in the domain of fraud detection.

Addressing Fraud Detection Challenges: Our research contributes to addressing the challenges of fraud detection in mobile payment systems. By effectively detecting fraudulent transactions, we aim to protect customer satisfaction, mitigate financial risks, and reduce substantial costs for companies operating in this sector.

8.2 Limitation and Challenges

While our research offers valuable insights and advancements in fraud detection, it is important to acknowledge certain limitations and challenges that were encountered during the study:

Generalizability: The synthetic PaySim Spanish Bank dataset used to validate the suggested methodology may not accurately reflect the complexity and variety of actual mobile payment transactions. To verify the generalizability of the suggested technique, more research is required to test its efficacy utilizing a variety of datasets from other industries and geographical areas.

Data accessibility: Due to the sensitive nature of the data, obtaining big, labeled datasets for fraud detection might be difficult. It's still difficult to gather a sufficiently large amount of fraudulent cases in a varied and representative dataset. Future research should concentrate on investigating methods to get over this restriction and obtain larger, more trustworthy datasets.

Real-Time Implementation: While our study focuses on the development of a fraud detection approach, the implementation of a real-time application for fraud transaction identification is a direction for future work. The development of such an application would require addressing challenges related to data handling, scalability, performance, and resilience to ensure efficient and accurate detection of fraudulent transactions in real-time.

Continuous Improvement: Fraudsters are constantly evolving their tactics, making it necessary to regularly update and retrain the fraud detection system with new and representative data. Future research should focus on developing adaptive and dynamic models that can adapt to emerging fraud patterns and ensure sustained performance over time.

By acknowledging these limitations and challenges, future research can further enhance the proposed approach and contribute to the continuous improvement of fraud detection in mobile payment systems.

8.3 Recommendations for Future Work

Experiment with additional over- and under-sampling techniques: In addition to the ones that were covered in this study, further over-, under-, and combination techniques should be investigated in the future. To increase the variety and efficacy of the oversampling process, newer techniques like ADASYN (Adaptive Synthetic Sampling), K-means SMOTE, and Borderline-SMOTE may be taken into consideration. For comparison and better fraud transaction detection performance, under-sampling techniques like Tomek Links, Edited Nearest Neighbors, and Cluster Centroids can be investigated. Additionally, cutting-edge hybrid approaches like SMOTEENN and SMOTETomek that mix oversampling and under sampling techniques would be worth researching to make use of both methods' advantages at once.

Development of a Real-Time Application: The next stage of this research will be the development of a real-time application for fraud transaction identification. This application should be able to apply optimum sampling strategies, make use of the chosen feature extraction algorithms, and analyze incoming transaction data in real time. To reduce the danger of fraudulent transactions, the system should be built to handle huge volumes of data efficiently and to generate predictions quickly. The program needs to be able to handle high-speed data streams without compromising accuracy, therefore scalability, performance, and resilience should all be considered.

Diversity of Datasets and Dataset Generalizability: In order to ensure the generalizability and robustness of the fraud transaction detection system, it is essential to evaluate the suggested methodologies on a range of datasets from different sectors and geographical locations. This would demonstrate the usefulness of the strategies in diverse transactional scenarios and highlight potential difficulties unique to particular areas. It is also essential to regularly update and retrain the detection system with new and representative data due to the dynamic nature of fraud tendencies in order to sustain its performance over time.

Feature Extraction Enhancement: Further experimentation and exploration of new feature extraction techniques can be conducted to enhance the existing data analysis process. Investigate advanced feature extraction methods that can capture more intricate patterns and anomalies associated with fraudulent transactions. This can potentially improve the accuracy and effectiveness of the fraud detection system.

These recommendations aim to enhance the proposed approach and contribute to the ongoing development of fraud detection in mobile payment systems. By exploring additional sampling techniques, developing real-time applications, diversifying datasets, and enhancing feature extraction, future research can further advance the field of fraud detection and prevention in mobile transactions.