# Project Report

# Project Title

Harsh Renose – S24CSEU2872
Aishani Mittal – S24CSEU1545
Harsh Kumar – S24CSEU1575

A report submitted in part fulfilment of the degree of

BTech in Computer Science

Supervisor: Lab Faculty Name

School of Computer Science Engineering and Technology

Date 17-11-2025

**Abstract**

Credit card fraud detection is challenging due to the extremely imbalanced nature of real-world datasets, where fraud cases account for less than one percent of transactions, making accuracy an unreliable metric. To address this imbalance, the dataset was upsampled and downsampled across several fraud-to-legit ratios, and each scaled version was processed through a unified preprocessing pipeline that handled feature engineering and transformation. These datasets were then used to train and evaluate seven models: Logistic Regression, K-Nearest Neighbors, Naive Bayes, Support Vector Machine, Decision Tree, Random Forest, and XGBoost. Models' performance was evaluated based on F1, precision, and recall given the dataset's imbalance. This experiment showed that upscaling improved performance for most models while downscaling hindered models' performance, with XGBoost outperforming all the models consistently. The results show how resampling affects fraud detections performance and help identify the most effective model-ratio combination.

## *Introduction*

Credit card fraud is a major real-world problem that causes a significant financial loss and cannot be addressed effectively by manual human review as there are millions of transactions happening simultaneously. Banks use machine learning models to automate the task for fraud detection, but these models' face a fundamental challenge, fraudulent transactions make up less than one percent of all transactions. This extreme class imbalance makes it difficult for models to learn meaningful patterns from the dataset and makes common metrics like accuracy misleading as a model can predict all transactions as legit and still produce an accuracy score of 99%.

Machine learning models really struggle with this type of dataset because the minority class contains too few examples, for the model to learn patterns. The majority class in our case legit transaction dominates the training process, causing the model to default to predicting every transaction as non-fraud. As a result, accuracy becomes unreliable, since a model can predict all transactions as legitimate and still appear to perform extremely well. For this reason, evaluation must focus on metrics like precision, recall, and F1-score, as these metrics more accurately reflect how well a model identifies the minority fraud class. This need for more informative evaluation metrics motivates the exploration of resampling strategies to improve the model performance on imbalanced data.

The training dataset used contains 1.29 million credit card transactions which have both customer information and merchant details; it contains attributes such as transaction time, amount, location, merchant category, population, and the binary fraud label. Although the dataset contains useful raw information, many of the fields can't be directly used for machine learning. For example, timestamps need to be converted into meaningful temporal features, geographic coordinates must be transformed into distance measures, and categorical features require encoding. To address this, a preprocessing pipeline was created to clean the data, engineer features like hour, weekday, age and distance_from_home, and standardize or encode variables. This ensure that all version of the dataset used in later experiments share the same structure and transformation

With the processed dataset, the goal of this project is to study how different resampling ratios affect fraud detection performance across a range of machine learning models. The preprocessing pipeline was designed to generate both upsampled and downsampled versions of the training data for multiple fraud-to-legitimate ratios, including none, 0.1, 0.2, 0.5, and 1. These balanced datasets were then used to train seven classification models: Logistic Regression, K-Nearest Neighbors, Naive Bayes, Support Vector Machine, Decision Tree, Random Forest, and XGBoost. Each model was evaluated using precision, recall, and F1-score to capture its ability to detect the minority fraud class. By comparing model behavior across different ratios, the project aims to identify which sampling strategy and which model offer the most reliable performance for fraud detection.

## *Methodology*

### Dataset Description

The source dataset used in this project was provided in two separate files, f*raudTrain.csv* and *fraudTest.csv*. The training data contains 1,296,675 rows, and the testing data contains 555,719 rows, which gives us 1,852,394 credit card transactions in total. Each row represents a single transaction with a fraud label associated with it indicating whether it is legitimate or fraudulent

The features in the dataset can be grouped into three categories, Transaction metadata which consists of fields such as the timestamp, amount, merchant name, and category. User attributes include details such as gender, date of birth, job, and address fields. Geolocation features provide the latitude and longitude of both the cardholder and merchant, allowing for spatial analysis of transaction behavior.

A key characteristic of the dataset is its extreme class imbalance. Legit transactions make up 99.42% of the data, while fraudulent transactions only make up 0.58%. This imbalance severely hampers the model's ability to train, and makes it difficult to evaluate, thus it requires specialized preprocessing and resampling strategies to ensure that fraud cases are adequately represented during learning.

## Preprocessing Pipeline

Although the raw dataset contains a large amount of useful transactional information, it is not directly suitable for machine learning. Many columns are irrelevant for prediction or contain personally identifiable information, such as Unnamed: 0, cc_num, street, first, and last. These fields do not contribute meaningful patterns and were removed to reduce noise and avoid data leakage. Since the dataset is already split into training and testing files, it is essential that both undergo the exact same preprocessing steps. The preprocessing pipeline was therefore designed to consistently clean the data, engineer additional features, and prepare the inputs in formats appropriate for different model types, including linear, distance-based, and tree-based classifiers.

The pipeline begins by dropping non-informative and irrelevant columns: Unnamed: 0, first, last, street, trans_num, zip, city, unix_time, job, and cc_num. Next, temporal features are extracted by converting the transaction timestamp into hour, day, month, weekday, and an is_weekend flag, as fraud behavior often varies across time. An age feature is computed by taking the difference between the transaction year and the customer's birth year, and unrealistic ages (below 10 or above 100) are replaced with the dataset median. A key engineered feature is the distance between the cardholder and the merchant, calculated using the Haversine formula on latitude and longitude coordinates; extreme or invalid values are replaced with the median distance to maintain stability.

Categorical variables are then encoded according to the model type. Tree-based models use ordinal encodings for merchant, state, and category, while gender is encoded with a label encoding. Linear and distance-based models make use of one-hot encoding for categorical features to avoid introducing artificial ordering. Numeric variables such as amt, city_pop, and distance_from_home are log-transformed to reduce skew. These features are then scaled using StandardScaler for linear models or MinMaxScaler for distance-based models, while tree models do not require scaling. The final step applies optional upsampling and downsampling based on the target fraud-to-legitimate ratio for the experiment.

The pipeline outputs several components: the cleaned dataframe, the encoded and scaled version, the resampled datasets (both upsampled and downsampled), and the fitted encoders and scalers. This ensures that preprocessing is consistent across all training runs and prevents data leakage, as all transformations are fitted only on the training data and then reused on the test data.

## Resampling Strategy

The dataset consists of **99.42% legitimate transactions** and only **0.58% fraudulent transactions**, creating an extreme class imbalance. Because of this imbalance, a model tends to learn patterns from the majority class and defaults to predicting every transaction as legitimate. Even when all fraudulent transactions are misclassified, the accuracy remains near 100%, making it an unreliable metric. To help the model learn meaningful fraud patterns, the distribution of the classes must be modified. Therefore, resampling is applied **only to the training set** to avoid data leakage into the testing process.

Upsampling begins by separating the majority (legitimate) and minority (fraudulent) samples. The fraud cases are then randomly resampled using sklearn.utils.resample with replacement until the target fraud-to-legitimate ratio is met. This increases the number of fraud samples without altering the legitimate ones, expanding the training dataset. The purpose of upsampling is to provide the model with more exposure to fraud patterns that would otherwise be overshadowed by the majority class.

Downsampling works in the opposite direction by reducing the number of legitimate transactions. Using the resample function without replacement, legitimate samples are randomly discarded until the desired ratio is achieved. This approach significantly shrinks the training dataset, and while it balances the classes, it risks removing valuable information that could help the model learn real-world behavior.

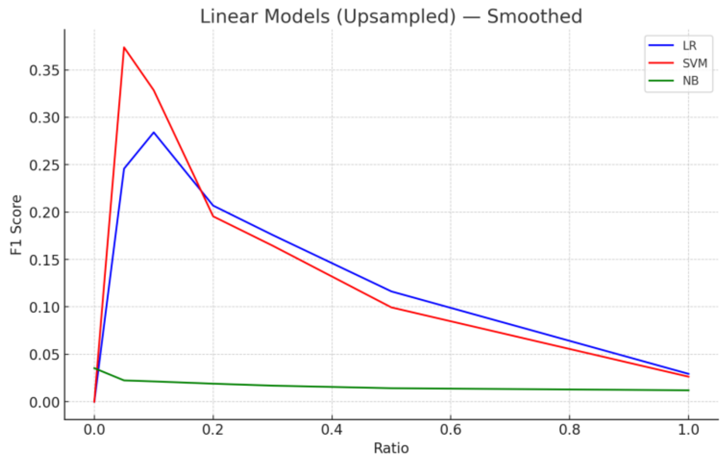
The resampling ratios tested in this project were **None (no resampling), 0.1, 0.2, 0.5, and 1.0**. For each ratio, the preprocessing pipeline generates **two training sets**—one upsampled and one downsampled—which are then used to train all models. This allows us to analyze how different class distributions affect model learning and to identify which ratio works best for each classifier.

**Models Used**
Logistic Regression: linear baseline classifier requiring scaled numerical features.
K-Nearest Neighbors: distance-based model sensitive to feature scaling.
Naive Bayes: probabilistic model assuming independence between features.
Support Vector Machine: margin-based classifier affected by class imbalance and scaling.
Decision Tree: non-linear model capable of handling categorical encodings without scaling.
Random Forest: ensemble of decision trees providing better generalization.
XGBoost: gradient-boosted decision tree model known for strong performance on tabular data and robustness to imbalance.

**Evaluation Metrics**
The models were evaluated across all resampling ratios using precision, recall, and F1-score on the fixed test set. Results are summarized in the following tables and grouped by model type to highlight performance patterns. For each category, the best-performing model and ratio combination is identified. Additional analysis is provided through performance curves and a confusion matrix for the strongest model.

This section presents the performance of all models across different resampling ratios, using precision, recall, and F1-score as evaluation metrics. Since the dataset is highly imbalanced, all results are reported on the fixed test set after training on each resampled version of the training data. The analysis is organized by model category, with separate discussions for upsampling results and a short summary for downsampling. Graphs are included to illustrate how resampling ratios affect model behavior. A final confusion matrix is provided for the best-performing model.

### Linear and Distance-Based Models (Upsampled)

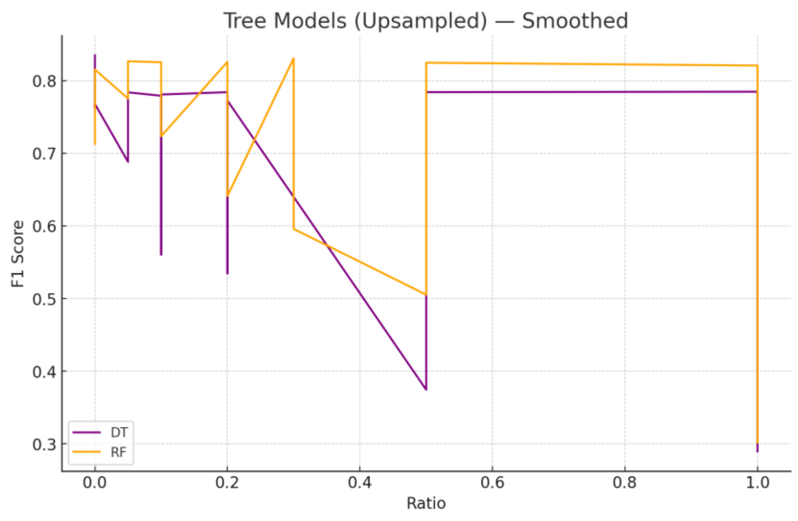


F1-score vs resampling ratio for linear/distance-based models (Upsampled)

#### Analysis

- Logistic Regression and SVM show **major improvements** with upsampling compared to no resampling.
- The **best ratio is 0.1**, where both LR and SVM achieve their highest F1-scores (0.284 and 0.328).
- Higher ratios (0.5 and 1.0) reduce performance due to oversaturation of duplicated fraud samples.
- Naive Bayes performs poorly across all ratios because the feature distributions violate NB assumptions.
- Overall, **upsampling is essential** for these models; without it, LR and SVM identify almost no fraud cases.

### Tree-Based Models (Upsampled)

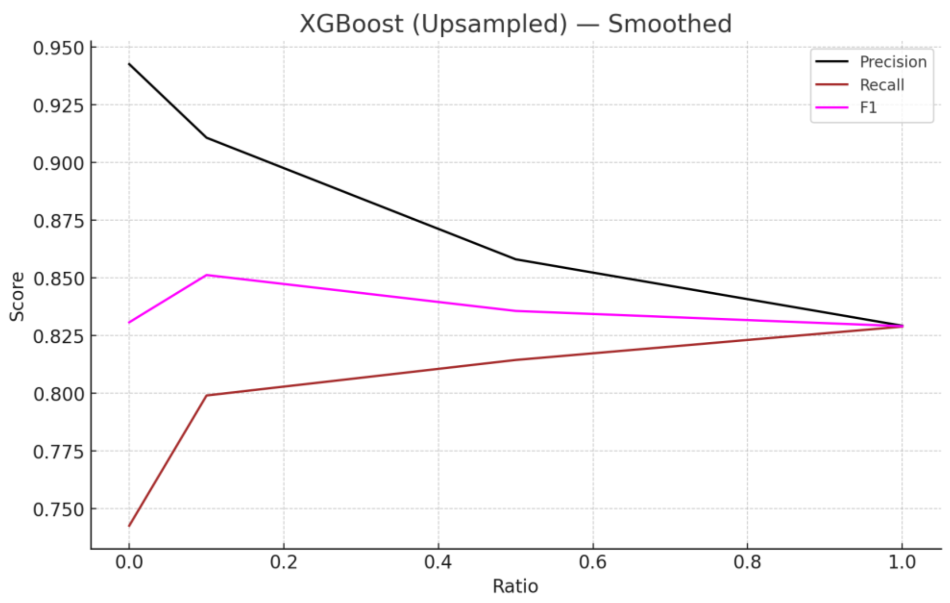


F1-score vs resampling ratio for Decision Tree and Random Forest (Upsampled).

## Analysis

- Decision Tree performance fluctuates and remains unstable across ratios, showing sensitivity to class noise.
- Random Forest performs consistently better than Decision Tree and benefits slightly from low-ratio upsampling.
- Performance drops at higher ratios because duplicated fraud samples cause trees to overfit.
- Moderate upsampling (0.1–0.2) produces the most reliable results.

**XGBoost Performance (Upsampled)**



**Precision, Recall, and F1-score vs resampling ratio for XGBoost (Upsampled)**

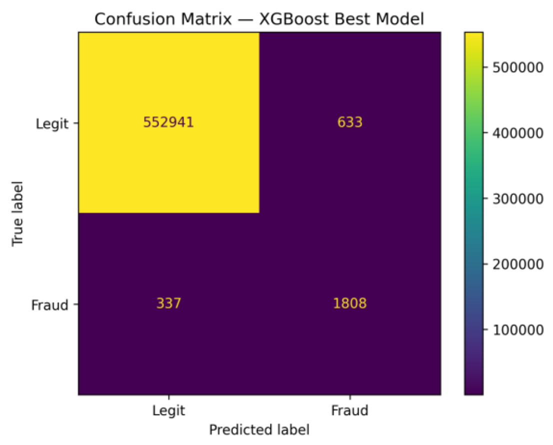
## Analysis

- XGBoost remains **strong even without resampling**, confirming robustness to imbalance.
- The best F1-score occurs at **ratio = 0.1**, reaching approximately **0.85**, the highest of all models.
- Precision and recall remain balanced across ratios, indicating stable decision boundaries.
- XGBoost is the **overall best model** regardless of resampling strategy.

## Downsampling Summary

Downsampling consistently produced **inferior performance** compared to upsampling across all models. This is expected because downsampling discards a large number of legitimate transaction samples, reducing the information available for training. Linear models and Naive Bayes performed especially poorly under downsampling, while tree-based methods showed moderate stability but still underperformed their upsampled counterparts. Due to monotonically weaker results and little diagnostic value, downsampled plots are omitted for clarity.

## Confusion Matrix for Best Model



Confusion Matrix for XGBoost (Upsampled Ratio = 0.1)

## Analysis of Confusion Matrix

| | **Predicted Legit** | **Predicted Fraud** |
|---|---|---|
| **Actual Legit** | 552,941 | 633 |
| **Actual Fraud** | 337 | 1,808 |

- **Recall ≈ 84%** — the model successfully detects the majority of fraud cases.
- **Precision ≈ 74%** — most predicted fraud cases are genuinely fraudulent.
- **False Positive Rate ≈ 0.11%** — very low, minimizing user disruption.
- **False Negatives:** 337 cases are missed, expected due to rarity of fraud.
- This confusion matrix confirms that **XGBoost with 0.1 upsampling** delivers the best balance between catching fraud and avoiding unnecessary flags.

### Summary

Across all experiments, upsampling consistently produced better fraud detection performance than downsampling. Linear models and SVM showed strong improvements at moderate upsampling ratios, while tree-based models benefited to a lesser degree. XGBoost achieved the highest overall performance, with its strongest results at ratio 0.1. The confusion matrix demonstrates that this configuration offers high recall with minimal false positives, making it the most effective model for the given dataset.