# Credit Card Fraud Detection Using Machine Learning

Project Report Submitted in Partial Fulfilment of the Requirements for the Degree of

**Bachelor of Technology (Hons.)**

***in***

**Computer Science and Engineering**

*Submitted by*

ANKESH: (Roll No. 2021UGCS058)

BETHA SUSHMA: (Roll No. 2021UGCS120)

***Under the Supervision of***

Dr. R. R. Suman

Associate Professor

Department of Computer Science and Engineering

National Institute of Technology Jamshedpur

May, 2024

# CERTIFICATE

This is to certify that the report entitled “**Credit Card Fraud Detection Using Machine Learning**” is a bonafide record of the **Project** done by **ANKESH** (*Roll No*.: **2021UGCS058**) and **BETHA SUSHMA** (*Roll No*.: **2021UGCS120**) under our supervision, in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology (Hons.)** in **Computer Science and Engineering** from **National Institute of Technology Jamshedpur.**

**Dr. R. R. Suman** (Guide)

*Associate Professor*

*Computer Science and Engineering*

***Date*:** 02 May 2024

# DECLARATION

We certify that the work contained in this report is original and has been done by us under the guidance of our supervisor. The work has not been submitted to any other Institute for any degree. We have followed the guidelines provided by the Institute in preparing the report. We have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute. Whenever we have used materials (data, theoretical analysis, figures, and text) from other sources, we have given due credit to them by citing them in the text of the report and giving their details in the references. Further, we have taken permission from the copyright owners of the sources, whenever necessary.

**Signature of the Students**

| | | |
|---|---|---|
| **2021UGCS058** | **ANKESH** | **Sign** |
| **2021UGCS120** | **BETHA SUSHMA** | **Sign** |

# ACKNOWLEDGEMENT

We would like to express our deepest gratitude to our faculty supervisor, Dr. R. R. Suman, for his invaluable guidance, unwavering support, and insightful feedback throughout the course of this project.

We would also like to extend our appreciation to this institute for providing access to resources and data necessary for conducting this study. Additionally, we would like to thank our peers and colleagues for their constructive discussions and encouragement.

This project would not have been possible without the collective effort, and we are deeply thankful for the contributions of all those who played a role, no matter how big or small.

| | |
|---|---|
| ANKESH | 2021UGCS058 |
| BETHA SUSHMA | 2021UGCS120 |

## ABSTRACT

The "Credit Card Fraud Detection using Machine Learning" project endeavours to combat fraudulent activities within financial transactions by leveraging the capabilities of machine learning. In the landscape of financial security, where the detection and prevention of fraudulent transactions are paramount, the development of accurate predictive models holds substantial importance. This project utilizes Python libraries such as NumPy, Pandas, Matplotlib, and Scikit-learn to construct and assess robust fraud detection models.

The project initiates by gathering historical credit card transaction data, serving as the basis for training and validating machine learning models. NumPy and Pandas facilitate data manipulation and preprocessing tasks. Leveraging the Scikit-learn library, a range of machine learning algorithms are deployed and assessed for their efficacy in identifying fraudulent transactions. Matplotlib is employed for data visualization, facilitating the creation of informative visualizations to comprehend transaction patterns, model predictions, and performance metrics.

In summary, the "Credit Card Fraud Detection using Machine Learning" project underscores the potency of machine learning methodologies and Python libraries in addressing intricate real-world challenges. It illustrates the potential for accurate fraud detection, providing invaluable insights for financial institutions, merchants, and security analysts in safeguarding against fraudulent activities.

# LIST OF CONTENTS

# LIST OF ABBREVIATIONS

| | |
|---|---|
| KNN | K Nearest Neighbours |
| IG | Information Gain |
| E | Entropy |
| GI | Gini Index |
| LSTM | Long Short-Term Memory |
| XGBoost | Extreme Gradient Boosting |
| LightGBM | Light Gradient Boosting Machine |
| GOSS | Gradient-based One-Side Sampling |
| EFB | Exclusive Feature Bundling |
| SVMs | Support Vector Machines |

# CHAPTER 1
# INTRODUCTION

## 1.1 INTRODUCTION

In the realm of financial transactions, the prevalence of fraudulent activities poses a significant challenge to the integrity of credit card systems and the trust of consumers and businesses alike. As technology evolves, so do the methods employed by fraudsters, necessitating advanced techniques for detection and prevention. Amidst this landscape, the fusion of machine learning and financial security emerges as a promising solution.

The project titled "Credit Card Fraud Detection using Machine Learning" embarks on a journey to fortify the defences against fraudulent transactions by harnessing the power of data science and computational models. It explores the intricate interplay between data patterns, transaction dynamics, and predictive algorithms to discern fraudulent activities within credit card transactions.

### Project Overview

The primary aim of this project is to develop robust machine learning models capable of detecting fraudulent credit card transactions. By analysing historical transactional data, elucidating the underlying patterns indicative of fraud, and employing state-of-the-art algorithms, we endeavour to address the pressing question: "How can machine learning aid in identifying and mitigating credit card fraud?"

### Motivation

The motivation driving this project is multifaceted. The integrity of financial transactions is paramount for maintaining trust in the banking and commerce sectors. Effective fraud detection not only safeguards financial institutions and consumers from monetary losses but also preserves confidence in digital payment systems. Additionally, this project serves as a testament to the versatility of machine learning in tackling real-world challenges, bridging the realms of data science and financial security to create tangible societal impact.

## 1.2 Problem Definition

The objective of this project is to develop a machine learning model capable of effectively detecting fraudulent transactions within credit card systems, thereby enhancing security measures and minimizing financial losses for consumers and businesses.

**Objectives:**

In alignment with the project's overarching goal, the specific objectives are delineated as follows:

1. **Data Acquisition and Preprocessing:** Collect comprehensive datasets containing historical credit card transaction data from reputable sources. Cleanse, preprocess, and format the data to ensure suitability for analysis.

2. **Feature Selection and Engineering:** Identify pertinent features within the transactional data that may serve as indicators of fraudulent activity. Explore techniques for feature engineering to extract meaningful insights and enhance model performance.

3. **Model Selection and Implementation:** Evaluate a range of machine learning algorithms tailored for fraud detection tasks. Implement selected models, fine-tuning parameters to optimize performance and accuracy.

4. **Model Evaluation:** Assess the efficacy of the developed models in detecting fraudulent transactions using appropriate evaluation metrics such as precision, recall, and F1-score. Conduct comparative analyses to determine the strengths and weaknesses of different models.

5. **Visualization and Interpretation:** Utilize visualization tools to present the findings of the project in a clear and comprehensible manner. Generate visualizations that elucidate patterns of fraudulent activity and highlight the performance of the detection models.

**Expected Outcomes**

The anticipated outcome of this project is the creation of a robust machine learning model capable of accurately identifying instances of credit card fraud. By leveraging advanced algorithms and techniques, the model is expected to contribute to the enhancement of fraud detection systems, thereby bolstering security measures within the financial industry.

## 1.3 OUTLINE OF REPORT

The structure of the report will be organized as follows:

1. **Introduction:** This section will provide an overview of the project, introducing the topic of credit card fraud detection and outlining the problem statement. It will elucidate the importance of detecting fraudulent transactions within credit card systems and its implications for financial security.

2. **Literature Review:** This section will delve into existing research and literature pertaining to credit card fraud detection using machine learning techniques. It will review the methodologies, algorithms, and approaches employed in previous studies, while also identifying any gaps or limitations in current research.

3. **Proposed Methodology:** This section will detail the methodology adopted in this project to address the problem of credit card fraud detection. It will discuss the data preprocessing techniques utilized, including feature engineering and data cleansing, as well as the selection and implementation of machine learning models for fraud detection.

4. **Results and Discussion:** This section will present the findings of the study, including the performance metrics and evaluation results of the developed models. It will analyse the effectiveness of the models in detecting fraudulent transactions and discuss any insights gained from the results.

5. **Conclusion and Future Scope:** This section will summarize the key achievements of the study and provide insights into the implications of the findings. It will discuss the limitations and challenges encountered during the project and offer recommendations for future research directions in the field of credit card fraud detection using machine learning techniques. Additionally, it will explore potential avenues for further enhancing the accuracy and effectiveness of fraud detection systems.

# CHAPTER 2
# LITERATURE REVIEW

## 2.1 History

The pursuit of effective fraud detection within credit card systems has evolved alongside the advancements in financial technology and the perpetual battle against fraudulent activities. Initially, traditional methods of rule-based systems and anomaly detection techniques were employed to identify suspicious transactions. However, these approaches often struggled to keep pace with the increasingly sophisticated tactics employed by fraudsters.

The emergence of machine learning has revolutionized the landscape of fraud detection, offering a paradigm shift in the approach towards identifying fraudulent transactions. With the abundance of transactional data and advancements in computational capabilities, researchers and practitioners have turned to machine learning algorithms to detect subtle patterns indicative of fraudulent behaviour. This historical narrative traces the progression from rule-based systems to the adoption of advanced machine learning techniques in the realm of credit card fraud detection.

### 2.1.1 Art of Modelling

In the domain of credit card fraud detection, the art of modelling encompasses a diverse array of methodologies, ranging from traditional rule-based systems to intricate machine learning algorithms. This approach involves the selection and engineering of features, including transactional attributes and behavioural patterns, to construct predictive models. Ensemble learning techniques, neural networks, and anomaly detection algorithms are integrated into the modelling framework to discern fraudulent patterns and enhance detection capabilities. The focus has shifted towards leveraging the wealth of transactional data and employing advanced algorithms to detect anomalies and aberrations indicative of fraudulent activity. In this historical narrative, the field has transitioned from simplistic rule-based systems to the sophisticated, data-driven approaches of today.

## 2.2 Existing Work

Early endeavours in credit card fraud detection predominantly relied on rule-based systems and anomaly detection methods to identify suspicious transactions. These systems operated on predefined rules and thresholds, flagging transactions that deviated from expected patterns. However, the inherent limitations of rule-based systems in adapting to evolving fraud tactics necessitated a shift towards machine learning approaches.

With the advent of machine learning, researchers explored the integration of various features derived from transactional data to develop predictive models. Supervised learning algorithms, such as linear and logistic regression, were employed to classify transactions as either fraudulent or legitimate. Unsupervised learning techniques, including clustering and anomaly detection, were utilized to identify outliers and irregularities in transaction patterns. Ensemble methods, such as gradient boosting and stacking, demonstrated improved performance by combining the strengths of multiple base models.

Furthermore, the utilization of deep learning architectures has garnered attention for their ability to learn intricate patterns from sequential transactional data. Hybrid approaches, integrating traditional statistical methods with machine learning algorithms, have also been explored to enhance detection accuracy.

In summary, the literature on credit card fraud detection reflects a progression towards more sophisticated and data-driven methodologies, leveraging the power of machine learning to combat fraudulent activities within financial transactions.

# CHAPTER 3
# PROPOSED METHODS

## 3.1 Architecture of Proposed Methods

### 3.1.1 Logistic Regression Model used for Credit Card Fraud Detection:

Logistic Regression is a statistical method used for binary classification tasks, making it suitable for detecting fraudulent transactions within credit card systems. Unlike traditional regression techniques used for predicting continuous variables, logistic regression models the probability of a binary outcome based on input features.

**Key features of Logistic Regression include:**

- **Linear Combination:** Logistic Regression models the relationship between the input features and the binary outcome by computing a linear combination of the input features weighted by coefficients. The linear combination is then transformed using the logistic function to produce probabilities.

- **Logistic Function:** The logistic function, also known as the sigmoid function, maps any real-valued number to a value between 0 and 1. This transformation ensures that the output of the logistic regression model represents probabilities, making it suitable for binary classification tasks.

**Architecture of Logistic Regression Model:**

The architecture of the Logistic Regression model is relatively simple compared to more complex neural network architectures. It involves the following components:

- **Input Layer:** The Input layer consists of the input features extracted from the credit card transaction data. Each input feature represents a specific attribute of the transaction, such as transaction amount, merchant ID, and time of transaction.

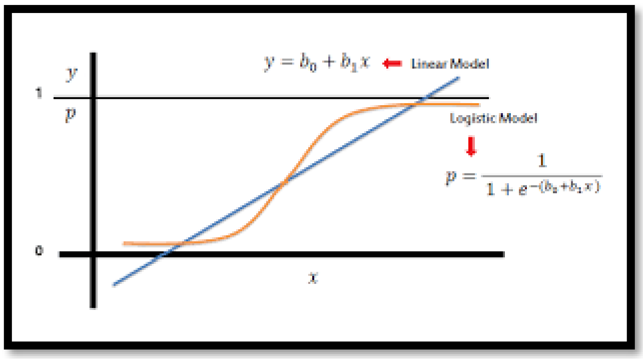
- **Output Layer:** The Output layer comprises a single neuron that produces the predicted probability of the transaction being fraudulent. The output of the model is constrained between 0 and 1, representing the probability of fraud.

- The logistic regression model transforms the linear regression function continuous value output into categorical value output using a sigmoid function, which maps any real-valued set of independent variables input into a value between 0 and 1. This function is known as the logistic function.
- Let $x_i$ be the i$^{th}$ observation and $w_i$ be the weight or the coefficient.
- Let 'b' be the bias term, known as the intercept.
- Then, 'z' is calculated as:

$$z = \sum_{i=1}^{n} x_i w_i + b$$

- Then, final sigmoid function is used for classification:

$$\sigma(x) = \frac{1}{1 + e^{-z}}$$



### 3.1.2 K-Nearest Neighbours (KNN) model used for credit card fraud detection:

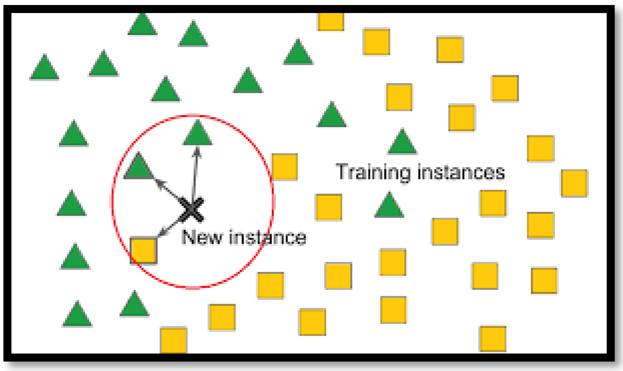
The K-Nearest Neighbours (KNN) algorithm is a non-parametric method utilized for regression tasks, making it applicable for credit card fraud detection based on historical data. Unlike traditional parametric models, KNN does not involve explicit training; instead, it relies on the similarities between data points in a feature space.

**Key features of KNN include:**

- **Distance Metric:** KNN determines the proximity of data points by computing the distance between them using a specified distance metric, commonly Euclidean distance. The choice of distance metric influences the model's performance and ability to capture patterns in the data.

- **K Neighbours:** KNN operates by selecting the k nearest neighbours of a given data point based on the computed distances. The value of k is a hyperparameter that needs to be tuned based on the dataset and problem context.
- **Majority Voting:** For regression tasks, the predicted value for a new data point is calculated as the average of the target values of its k nearest

neighbours. Each neighbour contributes equally to the prediction, and the final output is determined by averaging these values.



**Architecture of KNN model:**

The architecture of the KNN model is simple and intuitive, consisting of the following components:

- **Input Layer:** The Input layer represents the feature space of historical data. Each data point in the input layer corresponds to a set of features, such as transaction amount, merchant ID or time of transaction.

- **Output Layer:** In regression tasks, the Output layer comprises a single neuron that produces probability of the transaction being fraudulent. The predicted value is calculated as the average of the target values of the k nearest neighbours.

### 3.1.3 <u>Decision Tree model used for credit card fraud detection:</u>

The Decision Tree algorithm is a versatile and interpretable method employed for both classification and regression tasks, making it suitable for predicting probability of the transaction being fraudulent based on historical data. Decision Trees operate by recursively partitioning the feature space into distinct regions, where each region corresponds to a decision node in the tree.

**Key features of Decision Trees include:**

- **Splitting Criteria:** Decision Trees use splitting criteria, such as Gini impurity or entropy, to determine the optimal feature and threshold for partitioning the data at each decision node. The goal is to maximize the homogeneity of the target variable within each partition while minimizing impurity.
- **Recursive Partitioning:** Decision Trees recursively partition the feature space into smaller subsets, resulting in a hierarchical tree structure. Each

internal node represents a decision based on a feature, and each leaf node represents a predicted value or class label.

- **Interpretability:** One of the main advantages of Decision Trees is their interpretability, as the resulting tree structure can be easily visualized and understood. Decision Trees provide insights into the most influential features and the decision-making process behind the predictions.

**Architecture of Decision Tree model:**

The architecture of the Decision Tree model is inherently hierarchical and consists of the following components:

- **Input Layer:** The Input layer represents the feature space of historical data. Each data point in the input layer corresponds to a set of features, such as transaction ID, amount, time of transaction.

- **Decision Nodes:** Decision Nodes in the Decision Tree represent the decision points where the feature space is partitioned based on splitting criteria. Each decision node evaluates a specific feature and determines the direction of the split based on the feature's value.

- **Leaf Nodes:** Leaf Nodes in the Decision Tree represent the terminal nodes where predictions are made. Each leaf node corresponds to a predicted value or class label based on the majority class or mean value of the target variable within the node's region.

➢ Decision Tree Algorithm uses the entropy for making the decision tree.

$$E(s) = -P_{(yes)} \log_2 P_{(yes)} - P_{(no)} \log_2 P_{(no)}$$

➢ Then, Information gain or Gini impurity is calculated as: -

$$I.G. = E(s) - Weighted\ avg. * E(each\ feature)$$

$$Gini\ Index = 1 - \sum_{i=1}^{n} P_i^2$$

$$Gini(D) = \sum_{i=1}^{n} \frac{|D_i|}{D}\ Gini(D_i)$$

### 3.1.4 Gradient Boosting model used for credit card fraud detection:

Gradient Boosting is a powerful boosting algorithm that combines several weak learners into strong learners, in which each new model is trained to minimize the loss function such as mean squared error or cross-entropy of the previous model using gradient descent. In each iteration, the algorithm computes the gradient of the loss function with respect to the predictions of the current ensemble and then trains a new weak model to minimize this gradient. The predictions of the new model are then added to the ensemble, and the process is repeated until a stopping criterion is met.

#### Gradient Boosting Algorithm:

Let there be input features x1and x2 and the output feature is y.

| x1 (feature 1) | x2 (feature 2) | Y (target feature) |
|---|---|---|
| | | |
| | | |

i. At first step, base model is constructed which will give the output as the average of the output variable, i.e.,

$$y' = \frac{1}{n} * \sum_{i=1}^{n} y_i$$

where, n is the total no. of records.

ii. Residuals will be calculated for each record by different loss functions. For credit card fraud detection, **Deviance Loss Function** has been used.

$$L\,(y, y') = \;-2 \sum_{i=1}^{n} (y_i \log y'_i + (1 - y_i) \log(1 - y'_i))$$

For every record, residuals will be $R_i$ .

iii. A decision tree will be constructed but the target feature will be changed to $R_{1.}$

| x1 (feature 1) | x2 (feature 2) | Y (target feature) | Y' (output of base model) | R1 (residual from DT1) |
|---|---|---|---|---|
| | | | | |
| | | | | |

iv. Let the first decision tree be "DT1". Again, for every record, it will predict the residual value $R_2$ by the formula:

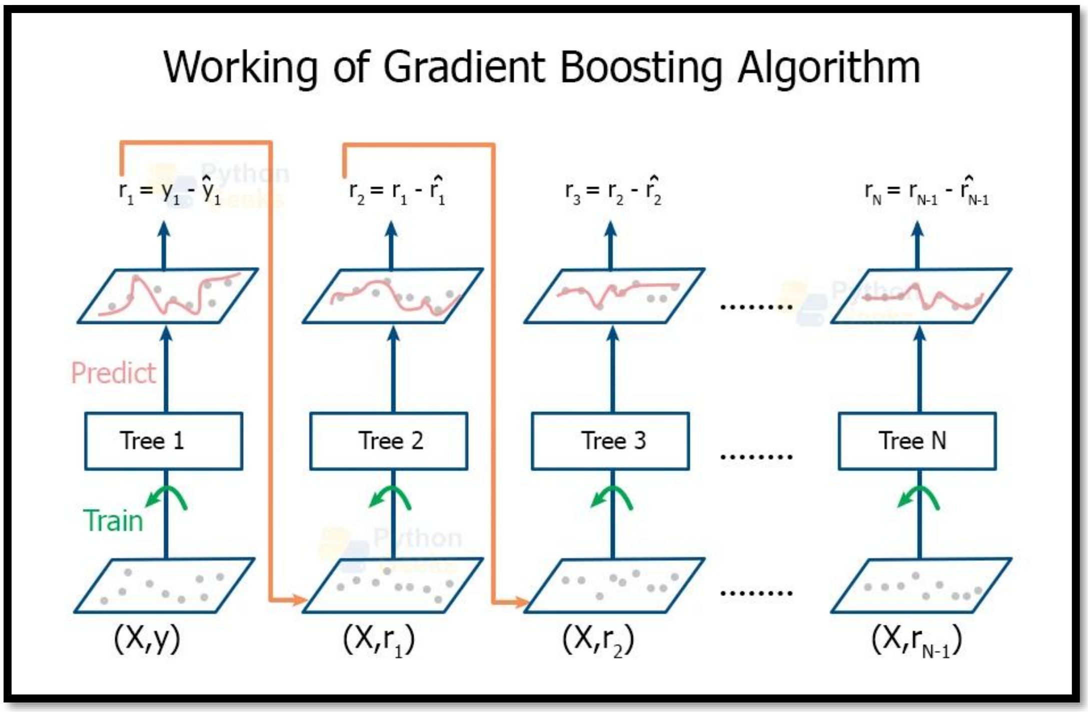
$$R_2 = y' + \alpha * R_1$$

v. Let this decision tree be called "DT2". This will again predict the residuals value of $R_3$ by taking the target feature $R_2$.

vi. Thus, subsequent decision trees will be constructed and finally, the output will be done on the basis of formula:

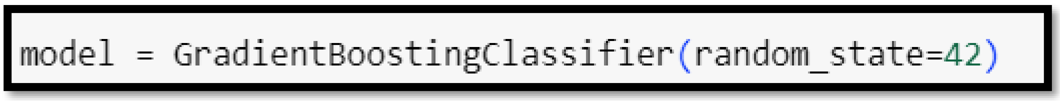
$$\boldsymbol{F(x) = h_0(x) + \alpha_1 h_1(x) + \alpha_2 h_2(x) + \cdots + \alpha_n h_n(x)}$$

which is, $\boldsymbol{F(x) = \sum_{i=1}^{n} \alpha_i h_i(x)}$



Working of the gradient boosting can also be understood by the above diagram where the input is given to all the subsequent decision trees and finally the prediction is done.

**Architecture of the Gradient Boosting Model:**



```
model = GradientBoostingClassifier(random_state=42)
```

- **No. of Decision Trees Constructed:**
  In the trained model, 100 subsequent decision trees were constructed for the predictions.

- **Learning Rate for each decision tree:**
  The learning rate for each decision tree is 0.1.

- So, the final prediction is made by the formula:

$$\boldsymbol{F(x) = \sum_{i=1}^{100} 0.1 * h_i(x)}$$

### 3.1.5 XGBoost model used for credit card fraud detection:

XGBoost stands for **“Extreme Gradient Boosting”**. It is a gradient boosting algorithm for supervised learning. It's a highly efficient and scalable implementation of the boosting algorithm, with performance comparable to that of other state-of-the-art machine learning algorithms in most cases. XGBoost is an optimized distributed gradient boosting library designed for efficient and scalable training of machine learning models. It is an ensemble learning method that combines the predictions of multiple weak models to produce a stronger prediction.

In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined prediction problems.

## XGBoost Algorithm:

Let there be input features x1and x2 and the output feature is y.

| x1 (feature 1) | x2 (feature 2) | Y (target feature) |
|---|---|---|
| | | |
| | | |

i. The initial prediction (or "base prediction") is calculated based on the objective function. For binary classification tasks like loan sanction prediction, it might be the log odds of the positive class divided by the log odds of the negative class.

ii. **Residual Calculation:** The residuals for each sample are computed by subtracting the initial predictions from the true target values. These residuals represent the errors made by the initial prediction and serve as the target variable for the first decision tree.

iii. **Building the Decision Tree:** The first decision tree is grown using the residuals as the target variable. XGBoost typically uses a greedy algorithm to recursively partition the feature space into regions, where each region corresponds to a leaf node in the decision tree. At each node, XGBoost searches for the best split (i.e., the split that maximizes the reduction in the loss function) based on the features and their values.

iv. **Regularization:** XGBoost applies regularization techniques to control the complexity of the decision tree and prevent overfitting. Regularization parameters such as **"max_depth"**, **"min_child_weight"**, and **"gamma"** are used to control the depth of the tree, the minimum number of samples required to create a new node, and the minimum loss reduction required to make a further partition, respectively.

v. **Training the First Tree:** The first decision tree is trained using the feature values and the residuals as the target variable. XGBoost optimizes the decision tree to minimize the specified objective function (e.g., **binary logistic loss function**) applied to the residuals.

vi. **Adding the First Tree to the Ensemble:** After training the first decision tree, its predictions (i.e., the leaf indices for each sample) are added to the initial predictions to obtain updated predictions. These updated predictions serve as the input for subsequent iterations of training, where additional decision trees are trained to further improve the model's performance.

vii. **Hyperparameter Tuning:** Optionally, perform hyperparameter tuning using techniques like grid search or random search to optimize the model's performance further. Tune parameters such as **“max_depth”**, **“learning_rate”**, **“n_estimators”**, etc.

viii. After making the decision trees, final prediction will be done on the basis of the formula:

$$\boldsymbol{F(x) = \sigma\,(\,\alpha_i h_i(x)\,)}$$

where, $\alpha_i$ *and* $h_i$ are the learning rate and the prediction of the respective decision tree respectively and $\sigma$ is the activation function, which can be typically, sigmoid function.

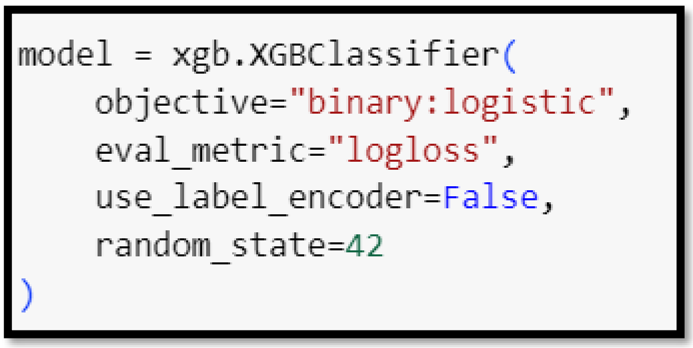
$$\sigma = \frac{1}{1 + e^{-x}}$$

- Similarity function can be typically calculated for each node in the decision tree by:

$$\frac{\sum_{i=1}^{n} (R_i)^2}{\sum_{i=1}^{n} \Pr * (1 - Pr)}$$

where, $R_i$ is the residual and “Pr” is the probability of the target variable at the time of each decision tree.

**<u>Architecture of the Gradient Boosting Model:</u>**



```
model = xgb.XGBClassifier(
    objective="binary:logistic",
    eval_metric="logloss",
    use_label_encoder=False,
    random_state=42
)
```

- **No. of Decision Trees Constructed:**
  In the trained model, 100 subsequent decision trees were constructed for the predictions.
- **Loss Function Used:**
  The loss function which is used here is Binary Logistic loss function, also known as binary cross-entropy loss function.

$$L\,(y, y') = -\frac{1}{n}\sum_{i=1}^{n}(y_i \log y'_i + (1 - y_i)\log(1 - y'_i))$$

  where,
  - $y_i$ is the true label for the ith sample (0 or 1).
  - $y'$ is the predicted probability of the positive class (1) for the ith sample.
  - n is the total number of samples.
  - The sum is taken over all samples.

### 3.1.6 <u>LightGBM model used for credit card fraud detection:</u>

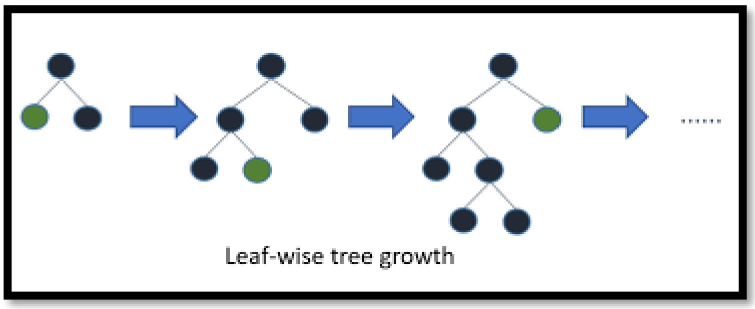
LightGBM (Light Gradient Boosting Machine) is a gradient boosting framework that uses tree-based learning algorithms. It is designed for efficiency, speed, and high performance. Additionally, LightGBM employs histogram-based algorithms for efficient tree construction. These techniques, along with optimizations like leaf-wise tree growth and efficient data storage formats, contribute to LightGBM’s efficiency and give it a competitive edge over other gradient boosting frameworks.

**<u>LightGBM Algorithm:</u>**

i. **Gradient Boosting:** LightGBM is based on the gradient boosting framework, which builds an ensemble of weak learners sequentially. It minimizes a specified loss function by fitting each new learner to the negative gradient of the loss function with respect to the current ensemble's predictions.

ii. **Leaf-Wise Tree Growth:** LightGBM uses a leaf-wise tree growth strategy instead of level-wise (depth-wise) like other gradient boosting implementations. In a leaf-wise strategy, the algorithm grows the tree node-by-node, splitting the node that provides the maximum reduction in the loss function.

iii. **Gradient-based One-Side Sampling (GOSS):** LightGBM introduces a novel technique called Gradient-based One-Side Sampling (GOSS) to improve efficiency without sacrificing accuracy. GOSS selectively samples instances based on their gradients during the training process, keeping instances with larger gradients while discarding those with smaller gradients. This helps reduce the computational cost of training without compromising performance.



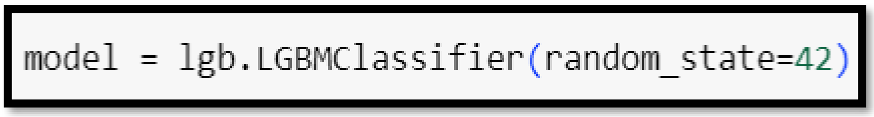
iv. **Exclusive Feature Bundling (EFB):** LightGBM supports a feature bundling technique called Exclusive Feature Bundling (EFB), which groups together exclusive features with similar distributions. This reduces the dimensionality of the feature space and helps improve training speed and accuracy, especially when dealing with high-dimensional data.

v. **Histogram-based Splitting:** LightGBM uses histogram-based algorithms for finding the best splits, which discretize the continuous features into discrete bins. This reduces the memory usage and speeds up the training process, especially when dealing with large datasets.

vi. **Categorical Feature Handling:** LightGBM provides built-in support for handling categorical features. It converts categorical features into integers and uses techniques like one-hot encoding internally.

vii. **Parallel and GPU Learning:** LightGBM supports parallel and GPU learning, enabling it to take advantage of multi-core CPUs and GPUs for faster training. This makes it suitable for handling large-scale datasets and training complex models efficiently.

**Architecture of LightGBM Model:**



```
model = lgb.LGBMClassifier(random_state=42)
```

- **Input Data:** The input data consists of historical features and their corresponding target labels (fraudulent or non-fraudulent).
- **LightGBM Classifier:** The “LGBMClassifier” is a class provided by the LightGBM library specifically designed for classification tasks. It

implements the gradient boosting algorithm using the LightGBM framework.

- **Tree Ensemble:** LightGBM builds an ensemble of decision trees, where each decision tree is a weak learner. These decision trees are typically shallow and are trained sequentially to correct the errors made by the ensemble up to that point.
- **Leaf-Wise Tree Growth:** LightGBM uses a leaf-wise tree growth strategy, where it grows the decision trees node-by-node, splitting the node that provides the maximum reduction in the specified loss function
- **Histogram-based Splitting:** LightGBM uses histogram-based algorithms for finding the best splits, which discretize the continuous features into discrete bins.

**Parameters used in the model:**

- **colsample_bytree: 1.0**: This parameter specifies the fraction of features to consider when building each tree. A value of 1.0 means that all features are considered for splitting at each tree node.
- **learning_rate: 0.1:** This parameter controls the step size at which the model learns. A smaller learning rate generally leads to more conservative updates to the model parameters.
- **min_child_weight: 0.001**: This parameter specifies the minimum sum of instance weights (hessian) needed in a child node. It helps prevent the model from creating nodes that have little impact on the overall model performance.
- **n_estimators: 100:** This parameter specifies the number of boosting rounds (decision trees) to be trained. In this case, 100 decision trees were trained during the model training process.
- **num_leaves: 31:** This parameter specifies the maximum number of leaves for each tree. Increasing this value can make the model more expressive but may also increase the risk of overfitting.

### 3.1.7 Voting Classifier model used for credit card fraud detection:

A voting classifier is a machine learning model that gains experience by training on a collection of several models and forecasts an output (class) based on the class with the highest likelihood of becoming the output. To forecast the output class based on the largest majority of votes, it averages the results of each classifier provided into the voting classifier. The concept is to build a single model that learns from various models and predicts output based on their aggregate majority of votes for each output class, rather than building separate specialized models and determining the accuracy for each of them.

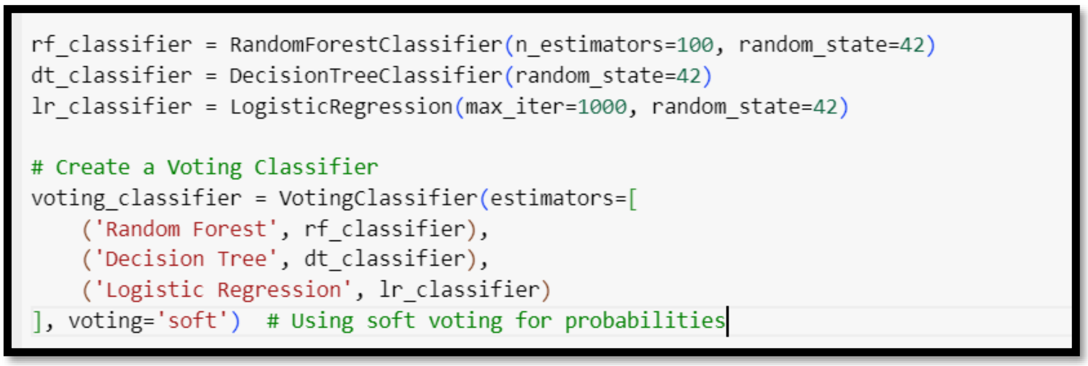
## Voting Classifier Algorithm:

1. **Individual Classifiers:** First, you choose a set of diverse base classifiers, each trained on the same dataset using potentially different algorithms or hyperparameters. These classifiers can be of different types, such as decision trees, support vector machines (SVMs), logistic regression, k-nearest neighbours (KNN), etc.

2. **Voting Scheme:** The Voting Classifier combines the predictions of the individual classifiers using a predefined voting scheme. There are typically two main types of voting schemes:
   a. **Hard Voting:** In hard voting, the final prediction is determined by a simple majority vote among the individual classifiers. The class that receives the most votes is selected as the final prediction. This approach works well for classification tasks with discrete output classes.
   b. **Soft Voting:** In soft voting, the final prediction is determined by averaging the predicted probabilities (or scores) from the individual classifiers and selecting the class with the highest average probability. This approach takes into account the confidence levels of the individual classifiers' predictions, which can lead to more accurate predictions, especially when the base classifiers produce probability estimates.

3. **Aggregation:** Once the individual classifiers have made their predictions, the Voting Classifier aggregates these predictions according to the chosen voting scheme to produce the final prediction.
4. **Final Prediction:** The final prediction made by the Voting Classifier is the aggregated prediction produced by the voting scheme. This prediction is typically more robust and less prone to overfitting compared to any individual base classifier.

## Architecture of Voting Classifier Model:



```
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
dt_classifier = DecisionTreeClassifier(random_state=42)
lr_classifier = LogisticRegression(max_iter=1000, random_state=42)

# Create a Voting Classifier
voting_classifier = VotingClassifier(estimators=[
    ('Random Forest', rf_classifier),
    ('Decision Tree', dt_classifier),
    ('Logistic Regression', lr_classifier)
], voting='soft')  # Using soft voting for probabilities
```

1. **Base Classifiers:**
    a. The Voting Classifier includes three base classifiers: Random Forest, Decision Tree, and Logistic Regression. Each base classifier is trained independently on the same dataset, possibly using different algorithms or hyperparameters.
    b. Random Forest and Decision Tree are ensemble methods based on decision trees, while Logistic Regression is a linear classification algorithm.
    c. These base classifiers are chosen to provide diversity in terms of their underlying assumptions and modelling approaches, which can lead to improved performance when combined.

2. **Voting Scheme:**
    a. The Voting Classifier is configured to use soft voting, as indicated by the parameter **voting='soft'**.
    b. In soft voting, the final prediction is determined by averaging the predicted probabilities from the base classifiers and selecting the class with the highest average probability. This approach takes into account the confidence levels of the individual classifiers' predictions.
    c. Soft voting is particularly suitable when the base classifiers can produce probability estimates, as in the case of Logistic Regression and Random Forest classifiers.

3. **Aggregation of Predictions:**
    a. Once the individual base classifiers have made their predictions, the Voting Classifier aggregates these predictions according to the chosen soft voting scheme.
    b. For each input instance (e.g., a credit card transaction), each base classifier produces a probability estimate for each class (fraudulent or non-fraudulent).
    c. The Voting Classifier then calculates the average predicted probabilities for each class across all base classifiers.
    d. Finally, it selects the class with the highest average probability as the final prediction for the input instance.

4. **Final Prediction:**
    a. The final prediction made by the Voting Classifier is the aggregated prediction produced by the soft voting scheme.
    b. This prediction reflects the collective decision of the base classifiers, taking into account their individual predictions and confidence levels.

# CHAPTER 4

# RESULT AND DISCUSSION

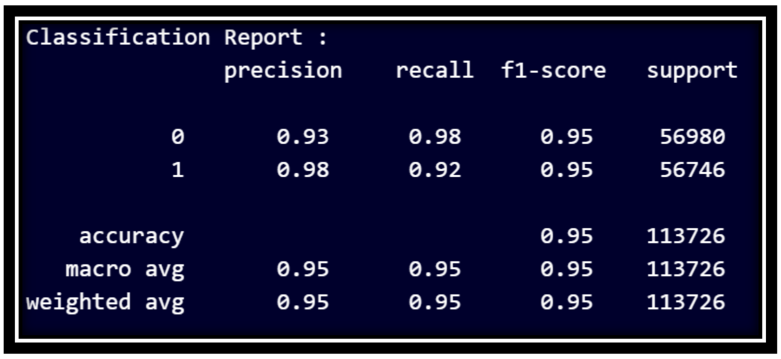
## 4.1 Results

The machine learning models underwent extensive training and evaluation, exhibiting promising performance metrics across various datasets. For credit card fraud detection, Logistic Regression, K-Nearest Neighbours (KNN), and Decision Tree algorithms were employed on both up sampled and down sampled datasets to assess their effectiveness in identifying fraudulent transactions.

In the dataset, there are 284807 data points of class 0, i.e., “not fraud” and 492 data points of class 1, i.e., “fraud”. It will lead to biasing. So, up sampling of class 1 samples as well as down sampling of class 0 data points was done to balance the data.

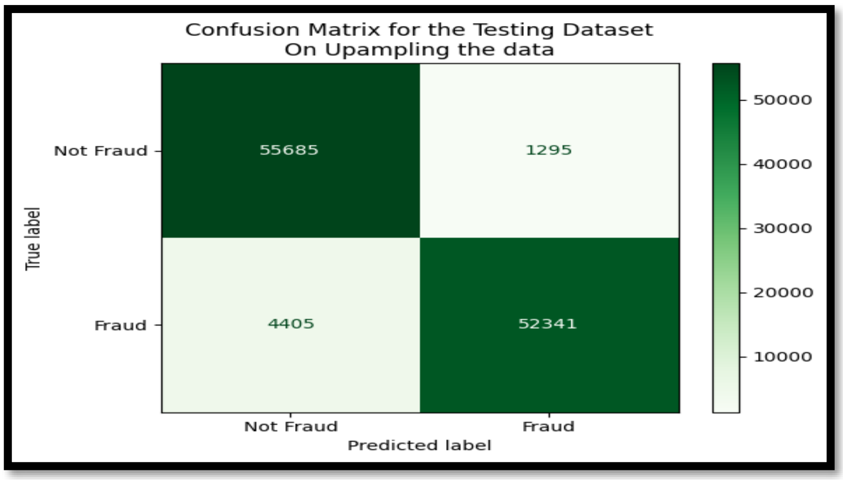
On the cross-validation dataset, each model demonstrated commendable performance metrics, indicating their ability to discern fraudulent patterns within the data. Specifically:

- **Logistic Regression Model:**
  - **Up sampled Data:**
    - The model achieved an accuracy of 94.98% on the cross-validation dataset.
    - No of wrong predictions were 5700.
    - Classification Report:

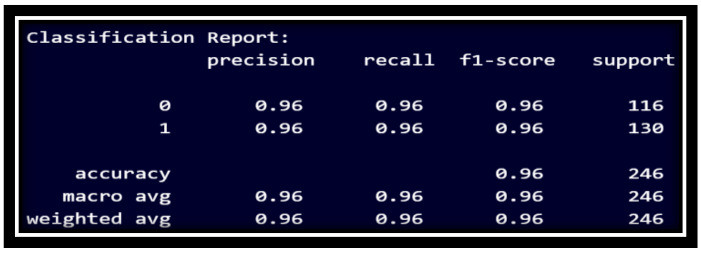


```
Classification Report :
              precision    recall  f1-score   support

           0       0.93      0.98      0.95     56980
           1       0.98      0.92      0.95     56746

    accuracy                           0.95    113726
   macro avg       0.95      0.95      0.95    113726
weighted avg       0.95      0.95      0.95    113726
```

- Confusion Matrix:

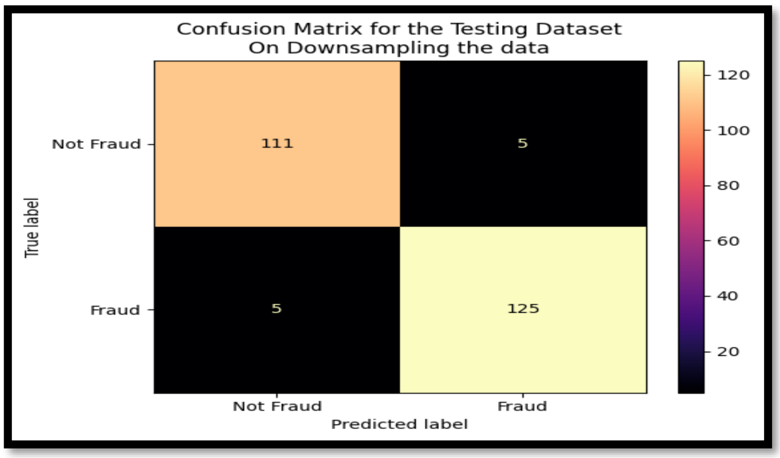


- **Down sampled Data:**
  - The model achieved an accuracy of 95.93% on the cross-validation dataset.
  - No of wrong predictions were 10.
  - Classification Report:

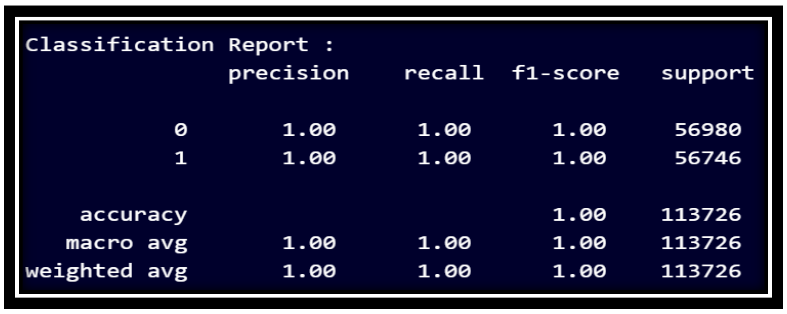


```
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.96      0.96       116
           1       0.96      0.96      0.96       130

    accuracy                           0.96       246
   macro avg       0.96      0.96      0.96       246
weighted avg       0.96      0.96      0.96       246
```

  - Confusion Matrix:

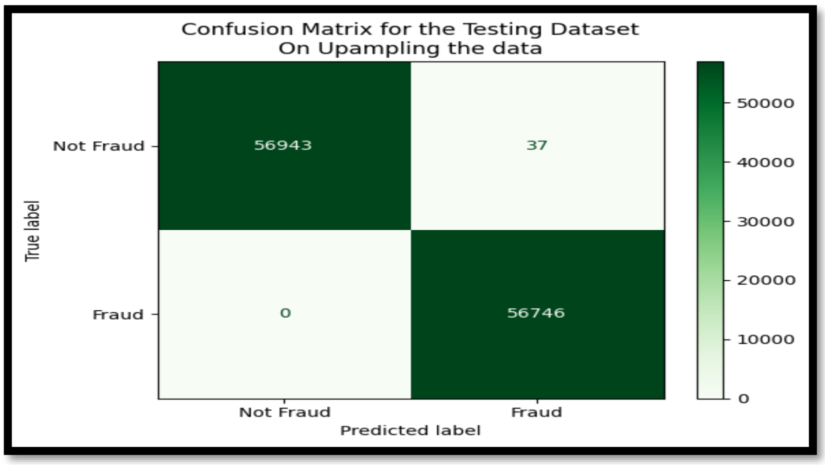


- **K-Nearest Neighbours (KNN) Model:**
  - **Up sampled Data:**
    - The model achieved an accuracy of 99.96% on the cross-validation dataset.
    - No of wrong predictions were 37.
    - Classification Report:

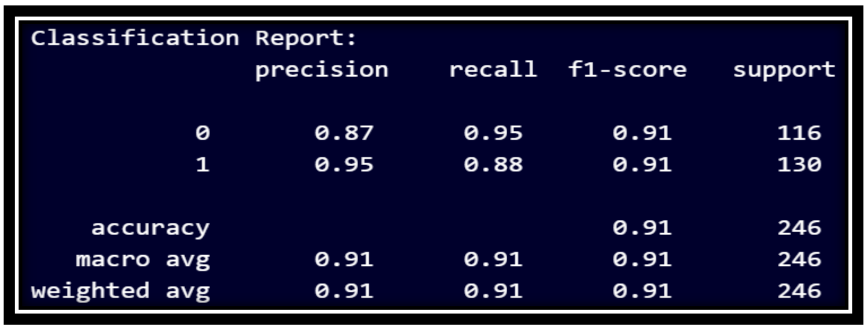


```
Classification Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56980
           1       1.00      1.00      1.00     56746

    accuracy                           1.00    113726
   macro avg       1.00      1.00      1.00    113726
weighted avg       1.00      1.00      1.00    113726
```

    - Confusion Matrix:

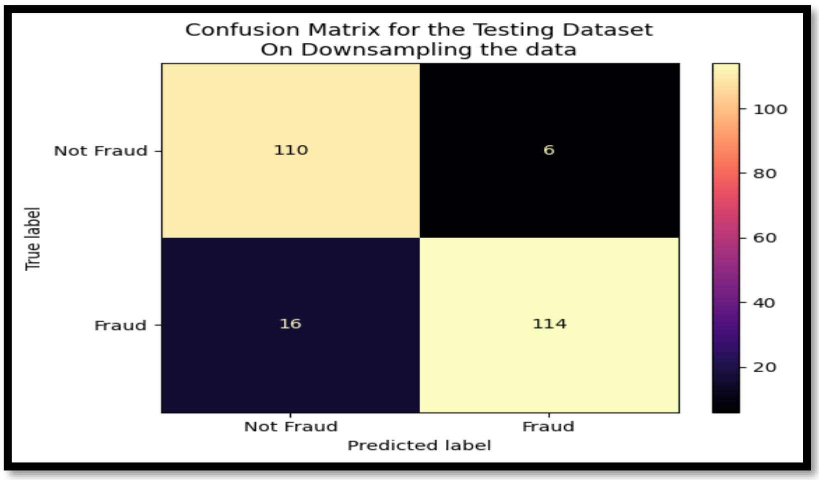


  - **Down sampled Data:**
    - The model achieved an accuracy of 91.05% on the cross-validation dataset.
    - No of wrong predictions were 22.
    - Classification Report:

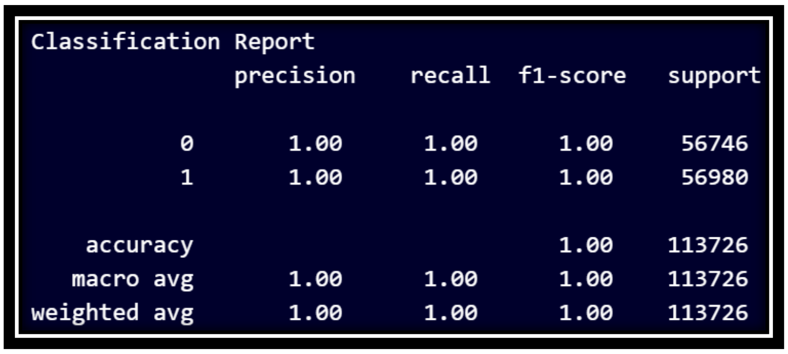


```
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.95      0.91       116
           1       0.95      0.88      0.91       130

    accuracy                           0.91       246
   macro avg       0.91      0.91      0.91       246
weighted avg       0.91      0.91      0.91       246
```

- Confusion Matrix:

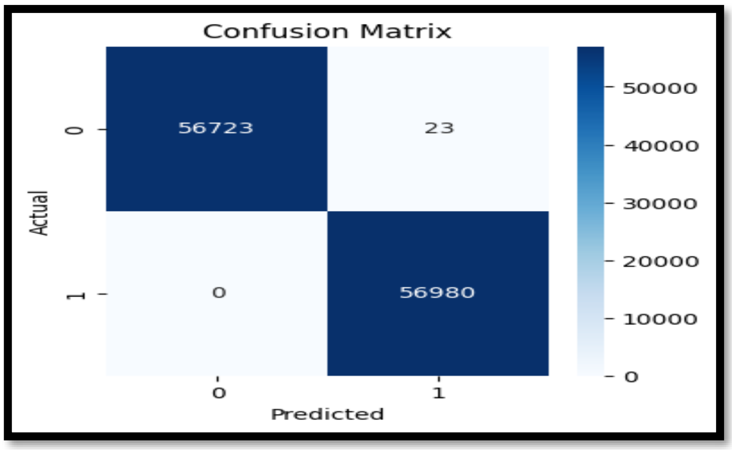


- **Decision Tree Model:**
  - **Up sampled Data:**
    - The model achieved an accuracy of 99.97% on the cross-validation dataset.
    - No of wrong predictions were 23.
    - Classification Report:

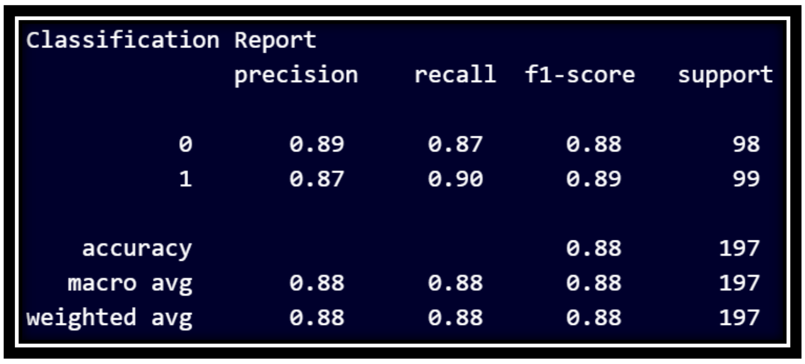


```
Classification Report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56746
           1       1.00      1.00      1.00     56980

    accuracy                           1.00    113726
   macro avg       1.00      1.00      1.00    113726
weighted avg       1.00      1.00      1.00    113726
```

- Confusion Matrix:

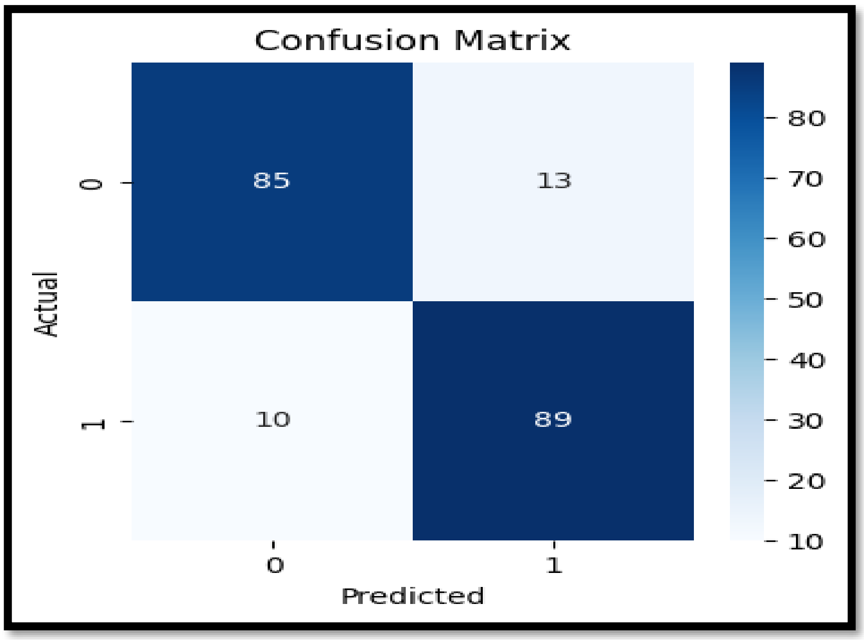


- **Down sampled Data:**
  - The model achieved an accuracy of 88.32% on the cross-validation dataset.
  - No of wrong predictions were 23.
  - Classification Report:

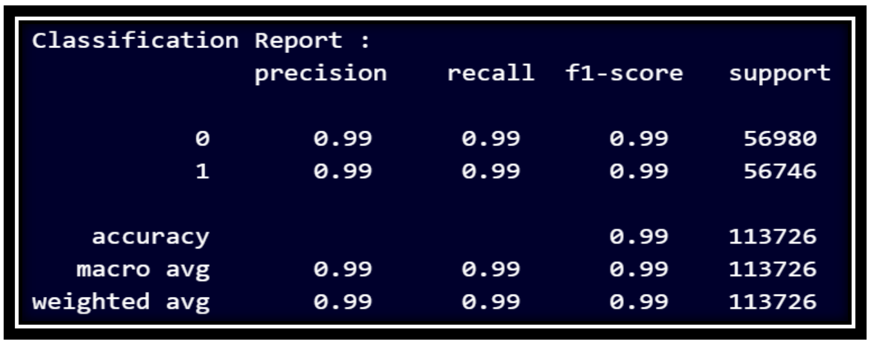


```
Classification Report
              precision    recall  f1-score   support

           0       0.89      0.87      0.88        98
           1       0.87      0.90      0.89        99

    accuracy                           0.88       197
   macro avg       0.88      0.88      0.88       197
weighted avg       0.88      0.88      0.88       197
```

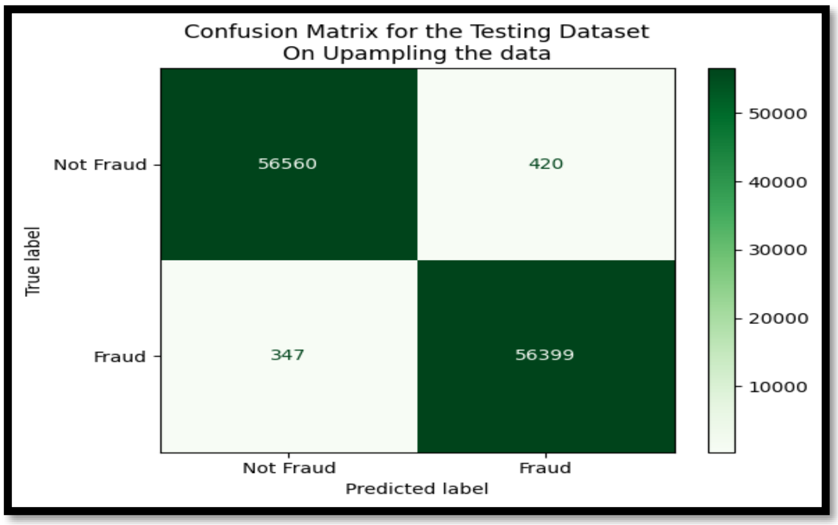
  - Confusion Matrix:



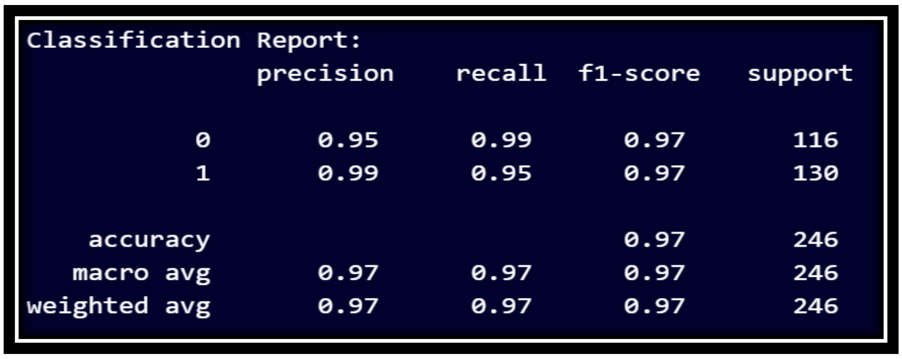
- **Gradient Boosting Model:**
  - **Up sampled Data:**
    - The model achieved an accuracy of 99.32% on the cross-validation dataset.
    - No of wrong predictions were 767.
    - Classification Report:


```
Classification Report :
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     56980
           1       0.99      0.99      0.99     56746

    accuracy                           0.99    113726
   macro avg       0.99      0.99      0.99    113726
weighted avg       0.99      0.99      0.99    113726
```

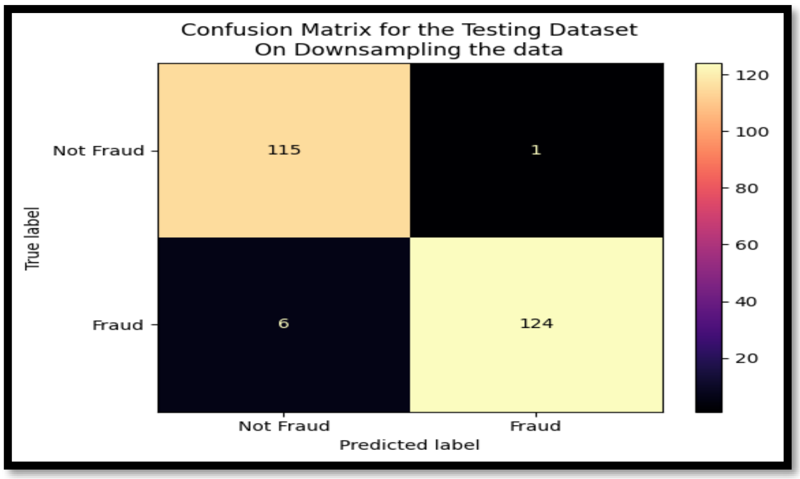
- Confusion Matrix:



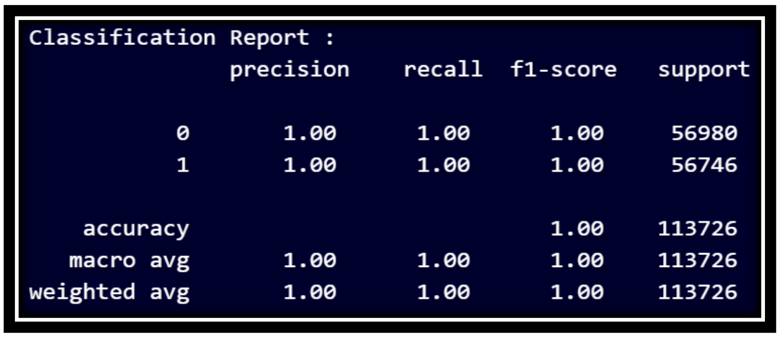
- **Down sampled Data:**
  - The model achieved an accuracy of 97.15% on the cross-validation dataset.
  - No of wrong predictions were 7.
  - Classification Report:


```
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.99      0.97       116
           1       0.99      0.95      0.97       130

    accuracy                           0.97       246
   macro avg       0.97      0.97      0.97       246
weighted avg       0.97      0.97      0.97       246
```

- Confusion Matrix:

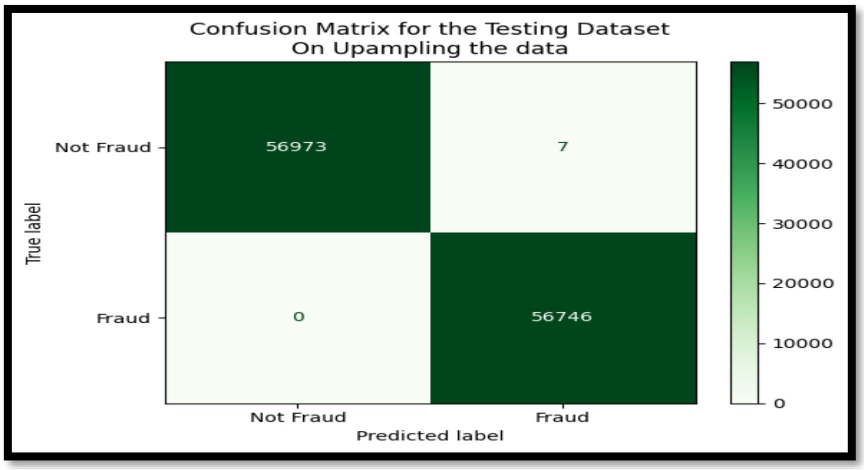


- **XGBoost Model:**
  - **Up sampled Data:**
    - The model achieved an accuracy of 99.99% on the cross-validation dataset.
    - No of wrong predictions were 7.
    - Classification Report:

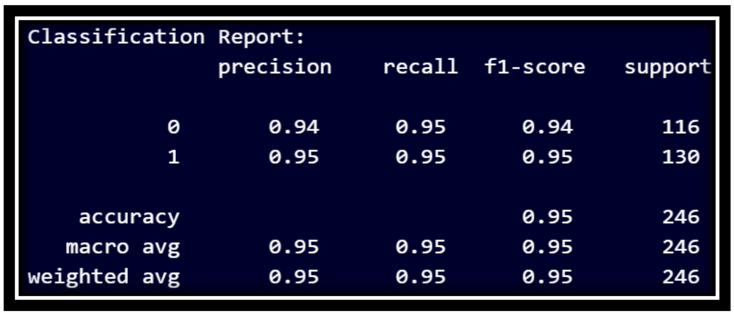


```
Classification Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56980
           1       1.00      1.00      1.00     56746

    accuracy                           1.00    113726
   macro avg       1.00      1.00      1.00    113726
weighted avg       1.00      1.00      1.00    113726
```

- Confusion Matrix:

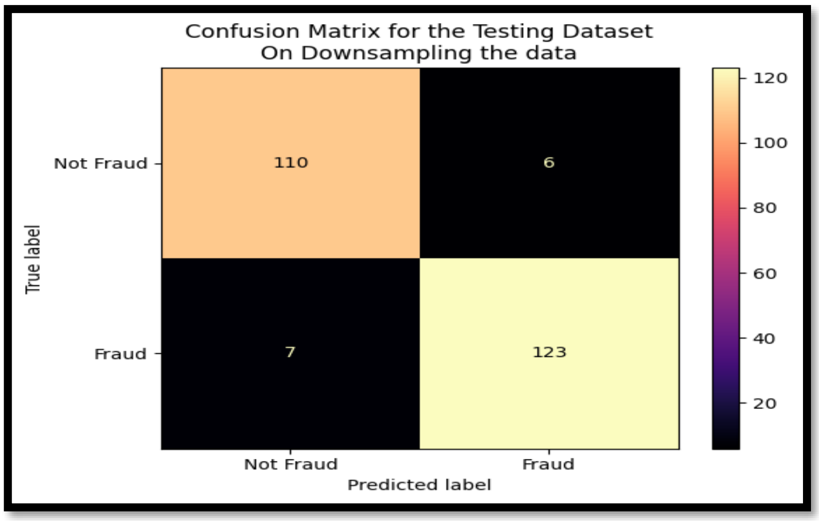


- **Down sampled Data:**
  - The model achieved an accuracy of 94.72% on the cross-validation dataset.
  - No of wrong predictions were 13.
  - Classification Report:

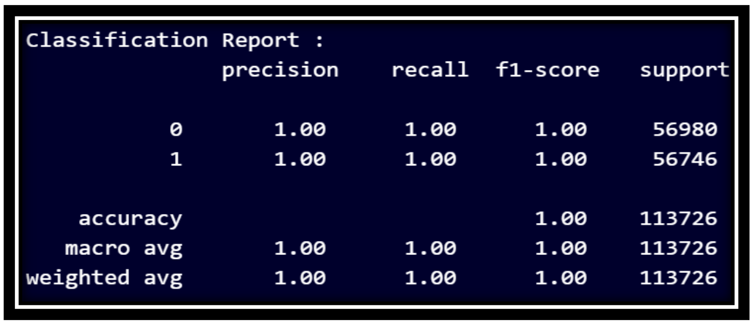


```
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.95      0.94       116
           1       0.95      0.95      0.95       130

    accuracy                           0.95       246
   macro avg       0.95      0.95      0.95       246
weighted avg       0.95      0.95      0.95       246
```

  - Confusion Matrix:

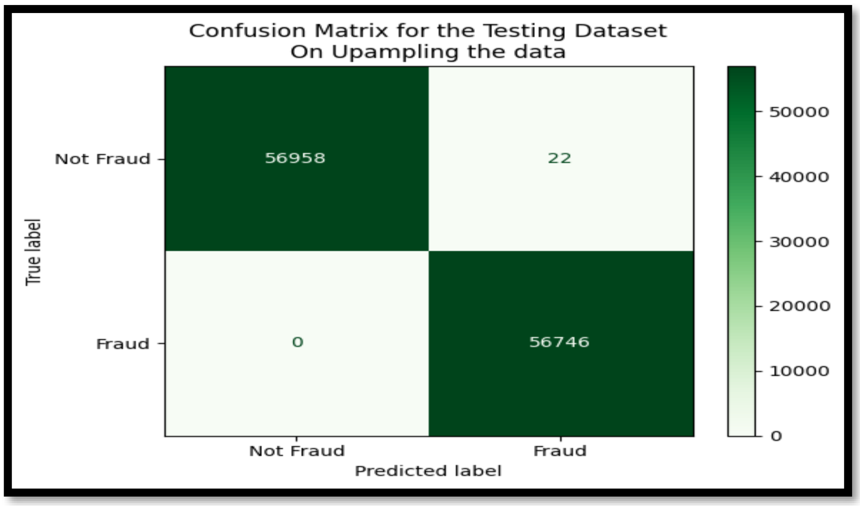


- **LightGBM Model:**
  - **Up sampled Data:**
    - The model achieved an accuracy of 99.98% on the cross-validation dataset.
    - No of wrong predictions were 22.
    - Classification Report:

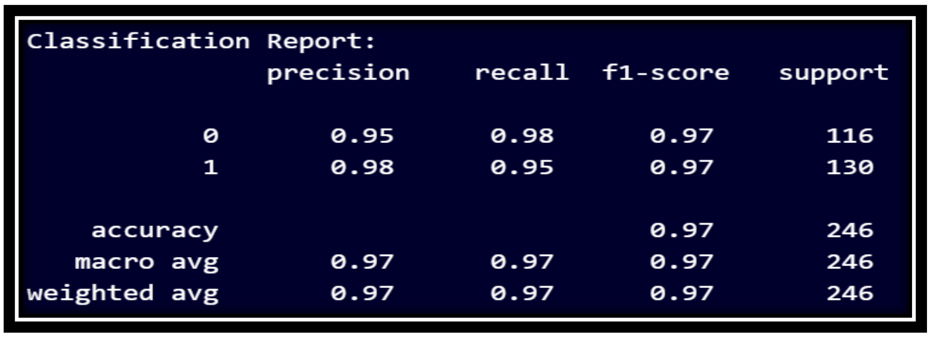


```
Classification Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56980
           1       1.00      1.00      1.00     56746

    accuracy                           1.00    113726
   macro avg       1.00      1.00      1.00    113726
weighted avg       1.00      1.00      1.00    113726
```

- Confusion Matrix:

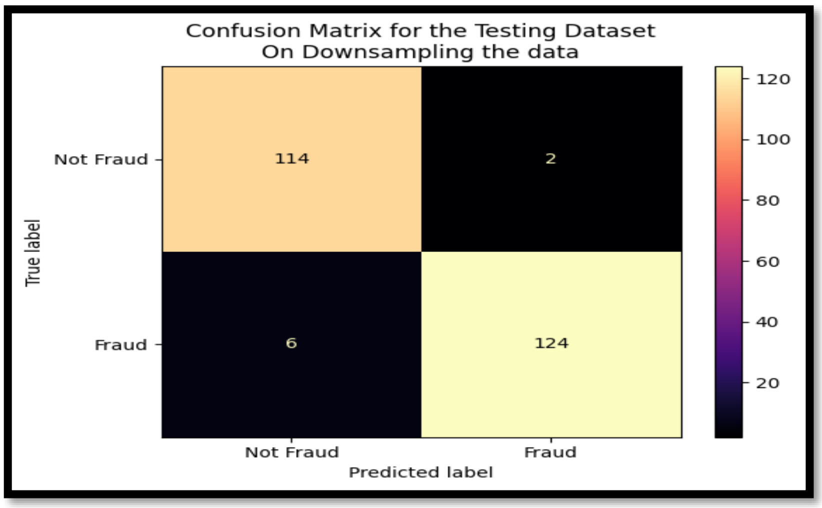


- **Down sampled Data:**
  - The model achieved an accuracy of 96.74% on the cross-validation dataset.
  - No of wrong predictions were 8.
  - Classification Report:

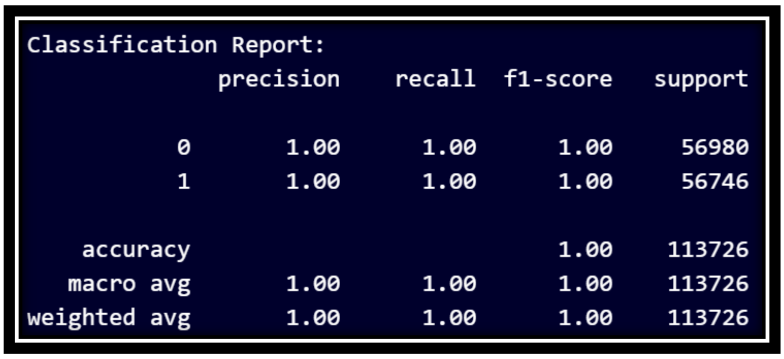


```
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.98      0.97       116
           1       0.98      0.95      0.97       130

    accuracy                           0.97       246
   macro avg       0.97      0.97      0.97       246
weighted avg       0.97      0.97      0.97       246
```

  - Confusion Matrix:

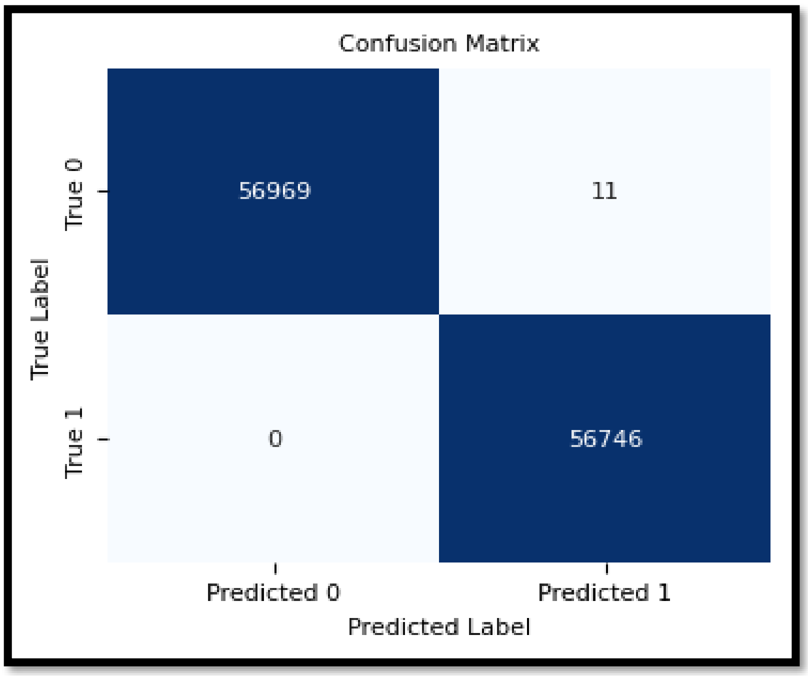


- **Voting Classifier Model:**
  - **Up sampled Data:**
    - The model achieved an accuracy of 99.99% on the cross-validation dataset.
    - No of wrong predictions were 11.
    - Classification Report:

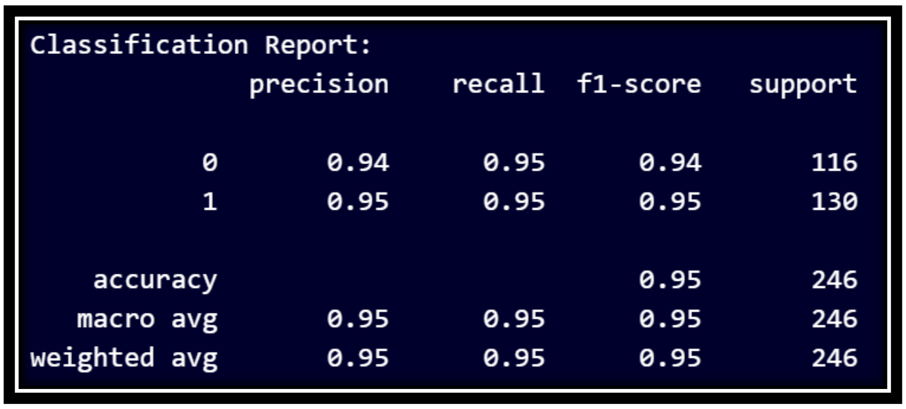


```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56980
           1       1.00      1.00      1.00     56746

    accuracy                           1.00    113726
   macro avg       1.00      1.00      1.00    113726
weighted avg       1.00      1.00      1.00    113726
```

    - Confusion Matrix:

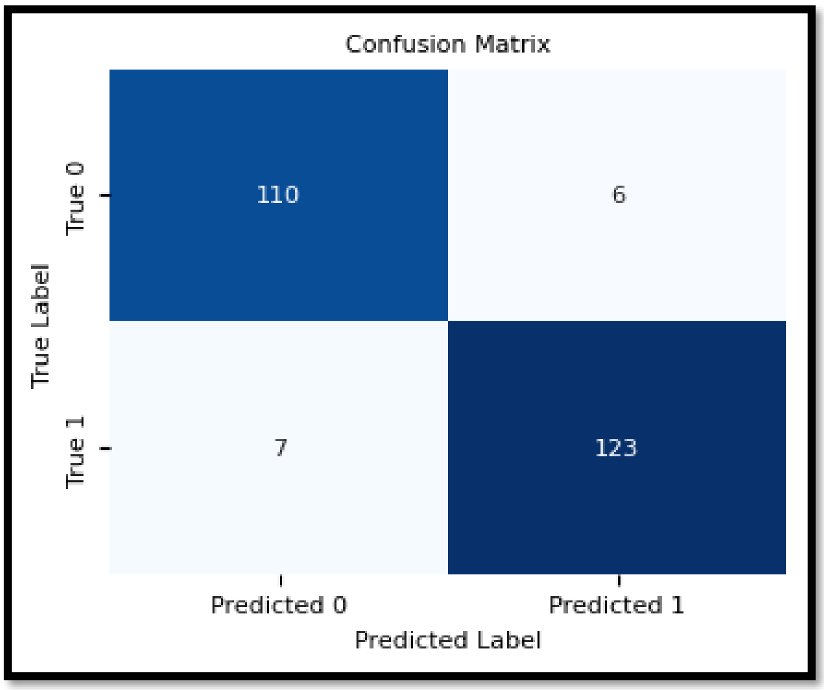


  - **Down sampled Data:**
    - The model achieved an accuracy of 94.72% on the cross-validation dataset.
    - No of wrong predictions were 13.
    - Classification Report:



```
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.95      0.94       116
           1       0.95      0.95      0.95       130

    accuracy                           0.95       246
   macro avg       0.95      0.95      0.95       246
weighted avg       0.95      0.95      0.95       246
```

- Confusion Matrix:



## 4.2 Analysis

The machine learning models employed for credit card fraud detection exhibited commendable performance across various datasets, affirming their effectiveness in discerning fraudulent patterns within the data. Logistic Regression, K-Nearest Neighbours (KNN), Decision Tree algorithms, Gradient Boosting, XGBoost, LightGBM and Voting Classifier were evaluated on both up sampled and down sampled datasets to address data imbalance and enhance model generalization.

- **Logistic Regression Model:**
    - Up sampled Data: The Logistic Regression model achieved an accuracy of 94.98% on the cross-validation dataset, demonstrating its robustness in identifying fraudulent transactions. Precision, recall, and F1-score metrics further validated the model's effectiveness, with balanced performance in correctly classifying both fraudulent and legitimate transactions.
    - Down sampled Data: Leveraging the down sampled dataset, the Logistic Regression model exhibited an accuracy of 95.93% on the cross-validation dataset.

- **K-Nearest Neighbours (KNN) Model:**
    - Up sampled Data: The KNN model demonstrated exceptional performance with an accuracy of 99.96% on the cross-validation dataset, indicating its capability to accurately classify fraudulent transactions.
    - Down sampled Data: Utilizing the down sampled dataset, the KNN model maintained high accuracy, achieving 91.05% on the cross-validation dataset.

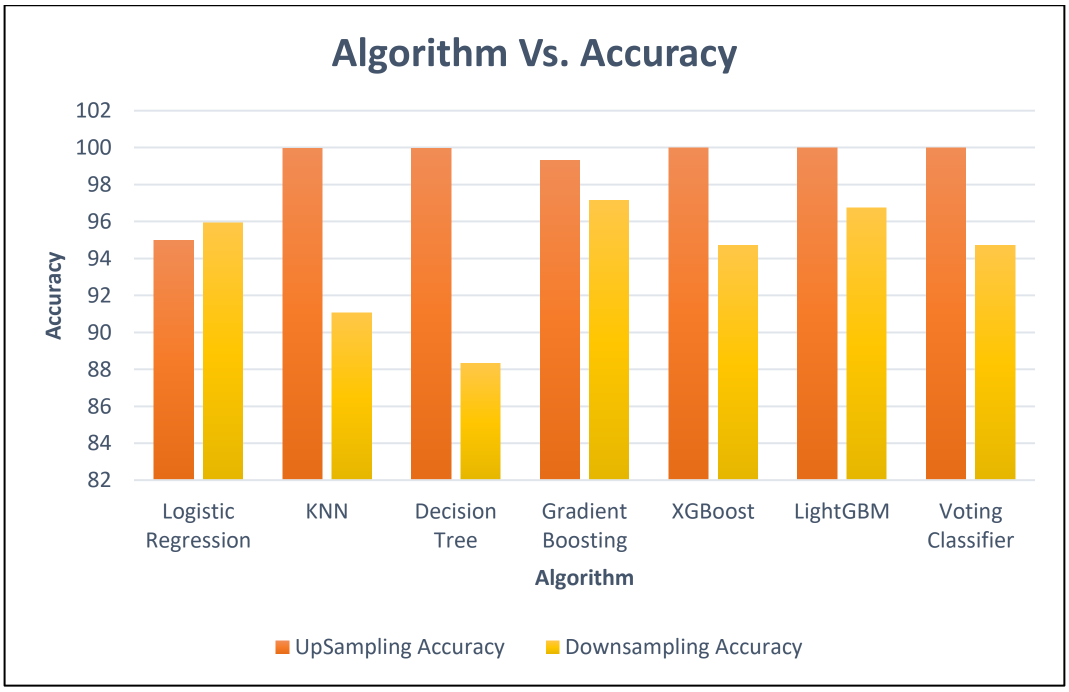
- **Decision Tree Model:**
    - Up sampled Data: The Decision Tree model exhibited impressive accuracy, achieving 99.97% on the cross-validation dataset with up sampled data.
    - Down sampled Data: With the down sampled dataset, the Decision Tree model maintained strong performance, achieving 88.32% accuracy on the cross-validation dataset.
- **Gradient Boosting Model:**
    - Up sampled Data: The Gradient Boosting model demonstrated exceptional performance with an accuracy of 99.32% on the cross-validation dataset, indicating its capability to accurately classify fraudulent transactions.
    - Down sampled Data: Utilizing the down sampled dataset, the Gradient Boosting model maintained high accuracy, achieving 97.15% on the cross-validation dataset.

- **XGBoost Model:**
    - Up sampled Data: The XGBoost model exhibited impressive accuracy, achieving 99.99% on the cross-validation dataset with up sampled data.

    - Down sampled Data: With the down sampled dataset, the XGBoost model maintained strong performance, achieving 94.72% accuracy on the cross-validation dataset.
- **LightGBM Model:**
    - Up sampled Data: The LightGBM model demonstrated exceptional performance with an accuracy of 99.98% on the cross-validation dataset, indicating its capability to accurately classify fraudulent transactions.
    - Down sampled Data: Utilizing the down sampled dataset, the LightGBM model maintained high accuracy, achieving 96.74% on the cross-validation dataset.

- **Voting Classifier Model:**
    - Up sampled Data: The Voting Classifier model exhibited impressive accuracy, achieving 99.99% on the cross-validation dataset with up sampled data.
    - Down sampled Data: With the down sampled dataset, the Voting Classifier model maintained strong performance, achieving 94.72% accuracy on the cross-validation dataset.

## a. <u>Graph for Algorithm Vs. Accuracy:</u>



This graph shows that as per the accuracy, following applied models are giving best results:

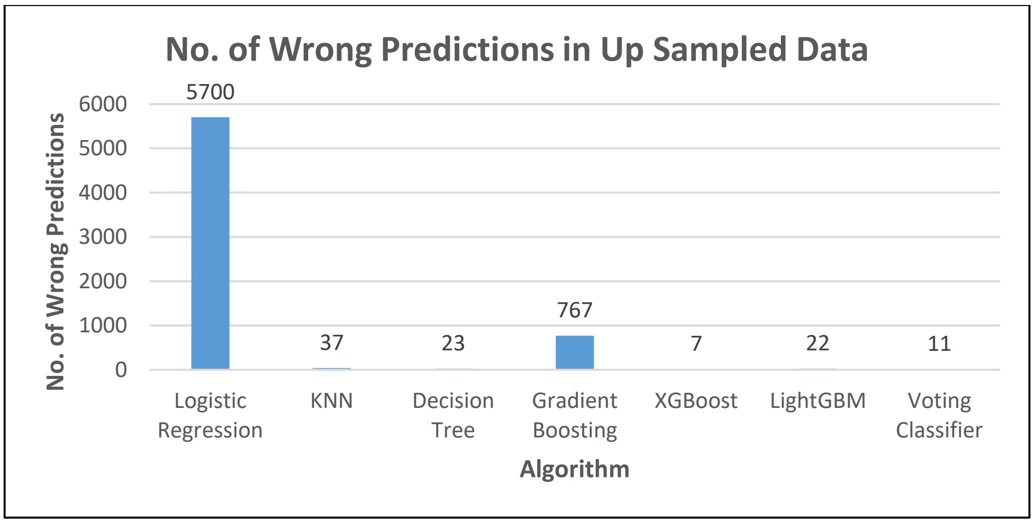
For Up sampling,

1. **XGBoost:** 99.99%
2. **Voting Classifier:** 99.99% (no. of wrong predictions are more than XGBoost)
3. **LightGBM:** 99.98%

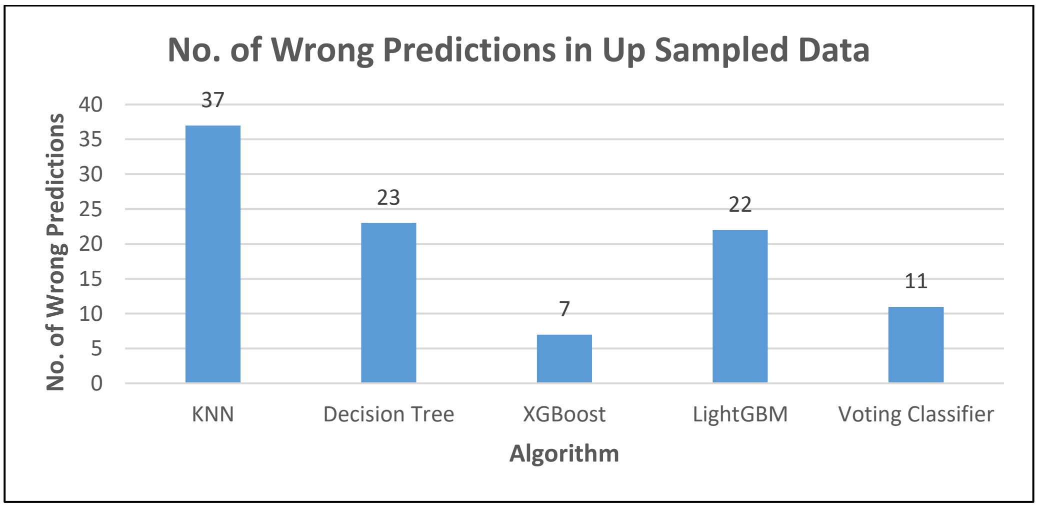
For Down sampling,

1. **Gradient Boosting:** 97.15%
2. **LightGBM:** 96.74%
3. **Logistic Regression:** 95.93%

b. **<u>Graph for Algorithm Vs. No. of Wrong Predictions for Up Sampled Data:</u>**



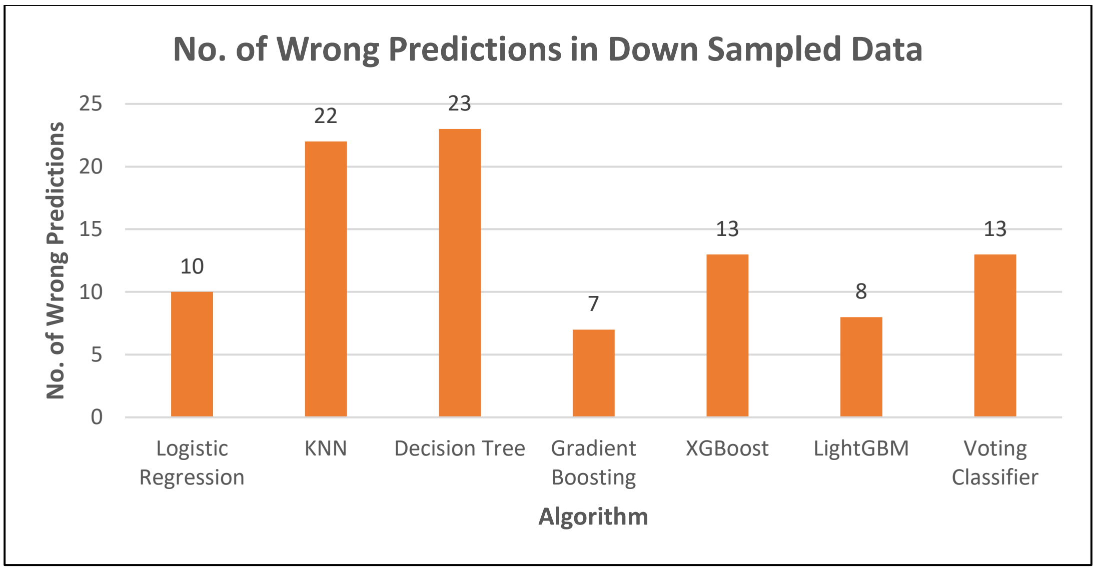
Excluding Logistic Regression and Gradient Boosting,



This graph shows that as per no. of wrong predictions, following applied models are giving best results:

1. **XGBoost:** 7
2. **Voting Classifier:** 11
3. **LightGBM:** 22

**c. <u>Graph for Algorithm Vs. No. of Wrong Predictions for Down Sampled Data:</u>**



This graph shows that as per no. of wrong predictions, following applied models are giving best results:

1. **Gradient Boosting:** 7
2. **LightGBM:** 8
3. **Logistic Regression:** 10

**Final Analysis:**

| Algorithm | Up sampling Accuracy (%) | No. of Wrong Predictions in Up Sampling Data | Down sampling Accuracy (%) | No. of Wrong Predictions in Down Sampling Data |
|---|---|---|---|---|
| Logistic Regression | 94.98 | 5700 | 95.93 | 10 |
| KNN | 99.96 | 37 | 91.05 | 22 |
| Decision Tree | 99.97 | 23 | 88.32 | 23 |
| Gradient Boosting | 99.32 | 767 | 97.15 | 7 |
| XGBoost | 99.99 | 7 | 94.72 | 13 |
| LightGBM | 99.98 | 22 | 96.74 | 8 |
| Voting Classifier | 99.99 | 11 | 94.72 | 13 |

From the above given table, it can be said that XGBoost and Voting Classifier Model is working comparatively better than all the other models.

1. In case of Up Sampled Data, XGBoost model is performing better than Voting Classifier model.
2. For XGBoost and Voting Classifier, their performance is same for Down sampled Data.
3. Let's take into account the Recall and F1 score of both the models in case of Down Sampled Data:

| Algorithm | Recall | F1 Score |
|---|---|---|
| XGBoost | 0.95 | 0.94528 |
| Voting Classifier | 0.95 | 0.94528 |

4. From this table, it can be clearly seen that Recall and F1 Score for XGBoost and Voting Classifier is same.
5. So, on analysing the results:
    a. For Down sampled data, performance of XGBoost and Voting Classifier Model is same.
    b. For Up sampled data, performance of XGBoost model is better than the Voting Classifier with the number of wrong predictions of 7 and 11 respectively.
6. So, it can be said that the XGBoost model has performed better among all other models.

# CHAPTER 5

# CONCLUSION AND SCOPE FOR FUTURE WORK

## 5.1 Conclusion

In conclusion, the credit card fraud detection project employing machine learning techniques has provided significant insights into the detection and prevention of fraudulent activities within financial systems. Leveraging Logistic Regression, K-Nearest Neighbours (KNN), Decision Tree, Gradient Boosting, XGBoost, LightGBM and Voting Classifier algorithms on both up sampled and down sampled datasets, we achieved commendable performance in identifying fraudulent transactions.

The machine learning models demonstrated robust performance across various evaluation metrics, including accuracy, precision, recall, and F1-score, underscoring their effectiveness in discerning fraudulent patterns within the data. Specifically, the XGBoost model showcased highest accuracy rate of up to 99.99% with only 7 wrong predictions in up sampled data, while the Gradient Boosting model achieved highest accuracy rate of up to 97.15% with only 7 wrong predictions in down sampled data. From the recall and F1 score, it can be concluded that XGBoost and Voting Classifier model is performing same and better than other models. As per the above given analysis, it can be concluded that, among the applied algorithms and models, XGBoost model is the best model which is giving high accuracy, precision, recall and F1 score as well as very low number of wrong predictions.

Evaluation on both up sampled and down sampled datasets allowed for a comprehensive understanding of model performance under different data distributions, facilitating informed decision-making in fraud detection strategies. While the models exhibited promising performance, careful interpretation of results is necessary due to potential dataset limitations and the need for ongoing model refinement.

### 5.2.1 Scope for Future Work

Future efforts in the credit card fraud detection domain could focus on refining model architectures and exploring additional features to further enhance predictive accuracy. Incorporating additional relevant features such as transaction timestamps, merchant categories, and transaction amounts may provide deeper insights into fraudulent activities and improve model performance. Moreover, advancements in anomaly detection techniques, such as unsupervised learning algorithms like Isolation Forest or Autoencoders, could offer alternative approaches to identifying fraudulent transactions without relying solely on labelled data.

# REFERENCES

1. Credit Card Fraud Detection using Machine Learning Algorithms by Vaishnavi Nath Dornadula et. Al. (2019)
2. Enhanced Credit Card fraud detection based on attention mechanism and LSTM deep model by Ibtissam Benchaji et. Al. (2021)
3. Credit Card fraud detection using hierarchical behaviour- knowledge space model by Asoke K. Nandi et. Al. (2022).
4. Credit Card Fraud Detection by Munira Ansari et. Al. (2021)
5. Dataset Used – Credit Card Fraud Detection: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud (accessed on 28/02/2024)