###### Department of CSE(DS)

**CONTENTS**

**CHAPTERS PAGES**

**CHAPTER-1: INTRODUCTION**

* 1. Credit card fraud detection system 4
  2. Credit card fraud detection using machine learning algorithm 4
  3. Credit card fraud detection predictive modelling 5

**CHAPTER-2: PROBLEM STATEMENT**

2.1 Overview 6

2.2 Class imbalance 6

2.3 Real time requuirment 6

2.4 Manual transaction monitoring 6

**CHAPTER-3: OBJECTIVES OF THE PROJECT**

3.1 Primary objective 7

3.2 Detailed objectives 7

3.3 Broader vision 8

**CHAPTER-4: SYSTEM REQUIREMENTS**

4.1 Hardware requirements 9

4.2 Software requirements 9

4.3 Dataset requirements 9

**CHAPTER-5: TECHNOLOGIES USED**

5.1 Python 10

5.2 Jupyter notebook 10

5.3 Scikit learn 10

5.4 XG boost 10

5.5 Imbalanced learn 10

5.6 Matplotlib and Seaborn 10

**CHAPTER-6: IMPLEMENTATION**

6.1 Data source and Description 11

6.2 Data preprocessing steps 11

6.3 Model building 11

6.4 Model evaluation metrics 12

**CHAPTER-7: RESULT**

7.1 Evaluation metrics 13

**CHAPTER-8: CONCLUSION** 15

**CHAPTER-9: REFERENCES**  16

**ABSTRACT**

The purpose of this project is to detect the fraudulent transactions made by credit cards by the use of machine learning techniques, to stop fraudsters from the unauthorized usage of customers’ accounts. The increase of credit card fraud is growing rapidly worldwide, which is the reason actions should be taken to stop fraudsters. Putting a limit for those actions would have a positive impact on the customers as their money would be recovered and retrieved back into their accounts and they won’t be charged for items or services that were not purchased by them which is the main goal of the project.

Detection of the fraudulent transactions will be made by using three machine learning techniques KNN, SVM and Logistic Regression, those models will be used on a credit card transaction dataset. Keywords: Credit Card Fraud Detection, Fraud Detection, Fraudulent Transactions, K Nearest Neighbors, Support Vector Machine, Logistic Regression, Naïve Bayes*.*

Credit card fraud is a significant concern in today's digital financial environment. This project focuses on developing a machine learning model to detect fraudulent transactions efficiently. The dataset contains labeled transaction records, which are used to train and evaluate the model. The system applies various classification algorithms to determine fraudulent activities, ensuring high accuracy and minimal false positives. The results indicate that advanced machine learning techniques can effectively detect credit card fraud

**CHAPTER-1**

**INTRODUCTION**

* 1. **Credit Card Fraud Detection System**

The proposed model involves pre-processing the credit card transaction data and then apply- ing various machine learning algorithms, such as Decision Trees, Random Forest, K-Nearest Neighbour, Naive Bayes, and Artificial Neural Networks, to classify transactions as either fraudulent or non-fraudulent. The authors evaluate the performance of their model using a dataset of credit card transactions and compare it with other existing models, such as Logistic Regression, Support Vector Machine, and Gradient Boosting Machine. The results show that the proposed model outperforms the other models in terms of accuracy, precision, recall, and F1-score. The paper also discusses the challenges associated with credit card fraud detection, such as the need for real-time detection, the challenges of handling imbalanced datasets, and the importance of feature selection for improving the performance of the model.

* 1. **Credit Card Fraud Detection using Machine Learning Algorithms**

The authors present their proposed model, which involves pre-processing the transaction data and then applying a CNN for feature extraction and classification. To improve the performance of the model, the authors also use feature selection techniques to identify the most relevant features for fraud detection. They evaluate the performance of their model using a dataset of credit card transactions and compare it to other models, such as logistic regression and decision trees. The results show that the proposed model performs better than the other models in terms of accuracy, precision, and recall. The authors also analyse the contribution of different features to fraud detection and discuss the limitations of the proposed model, such as the need for a large amount of data and the challenges associated with handling imbalanced datasets.

Despite the growing use of machine learning and deep learning models in fraud detection, there are still challenges that need to be addressed. One of the primary challenges is the ability to handle imbalanced datasets, where fraudulent transactions are rare compared to legitimate transactions. Another challenge is the need for interpretability of the models, which is crucial for financial institutions to understand how the models make decisions and ensure compliance with regulations..

* 1. **Credit Card Fraud Detection Predictive Modelling**

The paper covers different machine learning techniques such as supervised, unsupervised, semi-supervised, and deep learning, and how they are applied in credit card fraud detection. The authors provide a detailed explanation of each technique, including its advantages and limitations, and also present a comparative analysis of various machine learning techniques in terms of their performance metrics. The paper also discusses the challenges associated with credit card fraud detection, such as imbalanced datasets, the need for real-time detection, and the importance of feature selection for improving the performance of the model. The authors conclude the paper by highlighting the potential of machine learning techniques in credit card fraud detection and the need for further research in this area.

**CHAPTER-2**

**PROBLEM STATEMENT**

**2.1 Overview**  
Credit card fraud detection presents a multifaceted challenge. On one hand, frauds constitute a very small fraction of total transactions, making the dataset highly imbalanced. On the other hand, new fraud strategies keep emerging, making it difficult for static systems to keep up.

**2.2 Class Imbalance**  
Genuine transactions far outnumber fraudulent ones, causing traditional classifiers to lean toward majority class predictions. This leads to low sensitivity in detecting rare fraudulent activities.

**2.3 Real-Time Detection Requirement**  
The solution must work fast enough to stop fraud before the transaction is processed, which demands high-speed computation and low-latency algorithms. Fraudsters continually develop new strategies to bypass existing systems. Static models cannot cope with such adaptability unless continuously retrained.

**2.4** **Manual Transaction Monitoring**

Manual transaction monitoring refers to the traditional approach of reviewing and validating transactions by human analysts. This method involves evaluating transaction patterns, user behavior, and contextual details to determine whether a transaction is suspicious or genuine. Although time-consuming, manual monitoring remains an important step in many fraud detection workflows, especially for high-value or borderline transactions.

While manual monitoring can be effective in identifying nuanced fraud patterns, it suffers from several key limitations:

* **Scalability Issues**: Human analysts cannot keep up with the enormous volume of daily credit card transactions.
* **Time Delay**: Fraudulent transactions need to be stopped in real-time, but manual review introduces delay.
* **Human Error**: Judgments can vary between individuals, leading to inconsistencies in fraud detection.
* **High Operational Costs**: Hiring and training analysts incurs significant cost, making the approach unsustainable at scale.

**CHAPTER-3**

**OBJECTIVES OF THE PROJECT**

**3.1 Primary Objective**  
To design and develop a robust fraud detection model using machine learning algorithms that can identify fraudulent credit card transactions with high accuracy.

**3.2 Detailed Objectives**

**3.2.1 Research and Understanding**

* Gain a deep understanding of various types of credit card fraud and their characteristics.
* Review current fraud detection systems and identify their limitations.

**3.2.2 Data Collection and Preprocessing**

* Acquire and understand the publicly available credit card transaction dataset.
* Perform exploratory data analysis (EDA) to uncover insights.
* Preprocess data: handle missing values, normalize numerical values, encode categorical data if any.

**3.2.3 Handling Imbalanced Dataset**

* Analyze class distribution.
* Apply resampling techniques like SMOTE or ADASYN to balance the dataset.

**3.2.4 Model Implementation**

* Implement various classification algorithms: Logistic Regression, Decision Tree, Random Forest, and XGBoost.
* Use cross-validation for fair model evaluation.

**3.2.5 Performance Evaluation**

* Evaluate using metrics like Accuracy, Precision, Recall, F1-Score, ROC-AUC.
* Analyze confusion matrix to study true/false positive and negative rates.

**3.2.6 Optimization and Comparison**

* Tune hyperparameters using GridSearchCV or RandomizedSearchCV.
* Compare all models and select the best one for deployment.

**3.2.7 Integration and Usability**

* Design a user-friendly interface (CLI or web app prototype).
* Make the model usable for real-time or batch transaction detection.

**3.2.8 Report and Documentation**

* Document the project methodology, tools, and results clearly.
* Prepare presentations, reports, and visualizations for stakeholders.

**3.3 Broader Vision**

* Ensure the system is scalable and adaptable to real-world financial environments.
* Explore future integrations with blockchain for secure audit trails or real-time fraud alerting systems.
* Encourage open-source collaboration and future research extensions.

**CHAPTER-4**

**SYSTEM REQUIREMENTS**

**4.1 Hardware Requirements**

* Processor: Intel i5 or above
* RAM: 8 GB minimum
* Hard Disk: 500 GB or more
* GPU (optional): NVIDIA for faster model training

**4.2 Software Requirements**

* Operating System: Windows/Linux
* Programming Language: Python 3.8 or higher
* Libraries: NumPy, Pandas, Matplotlib, Scikit-learn, XGBoost, imbalanced-learn
* IDE: Jupyter Notebook/Google Colab/VS Code

**4.3 Dataset Requirements**

* Publicly available credit card fraud dataset from Kaggle
* Features: anonymized 28 features + 'Time', 'Amount', and 'Class'

**CHAPTER-5**

**TECHNOLOGIES USED**

**5.1 Python Programming Language**  
Python is the core language used for implementation due to its simplicity and rich ecosystem for machine learning.

**5.2 Jupyter Notebook**  
Used for prototyping, visualization, and interactive analysis.

**5.3 Scikit-learn**  
Used for data preprocessing, model training, and evaluation of traditional ML algorithms.

**5.4 XGBoost**  
An advanced gradient boosting technique used for better classification performance.

**5.5 Imbalanced-learn**  
A Python package offering tools like SMOTE for addressing class imbalance.

**5.6 Matplotlib and Seaborn**  
Visualization libraries used to understand data distribution and model performance.

**CHAPTER-6**

**IMPLEMENTATION**

**6.1 Data Source and Description**  
The dataset used in this project is the publicly available credit card fraud detection dataset from Kaggle. It contains transactions made by European cardholders in September 2013. The dataset includes 284,807 transactions, of which only 492 are fraudulent. Each record has 30 features: 'Time', 'Amount', 28 anonymized PCA components (V1–V28), and the 'Class' label (0 for genuine, 1 for fraud).

**6.2 Data Preprocessing Steps**

**6.2.1 Data Cleaning**

* No missing values were found in the dataset.
* Unnecessary features were not present as data is already anonymized.

**6.2.2Train-Test Split**

* The dataset was split into 80% training and 20% testing using stratified sampling to preserve the class ratio.

**6.3 Model Building**  
Multiple classification algorithms were implemented to compare their effectiveness:

**6.3.1 Logistic Regression**

* A baseline model.
* Performs well on linearly separable data but lacks performance on complex patterns.

**6.3.2 Decision Tree**

* Simple tree-based classifier.
* Interpretable but prone to overfitting.

**6.3.3 Random Forest**

* An ensemble of decision trees.
* Handles overfitting better and improves accuracy.

**6.3.4 XGBoost**

* An optimized gradient boosting algorithm.
* Provides high accuracy, handles imbalance, and is robust against overfitting.

**6.4 Model Evaluation Metrics**  
To assess model performance, we used:

* **Accuracy**: Proportion of total correct predictions.
* **Precision**: Ratio of true fraud predictions to total predicted frauds.
* **Recall** (Sensitivity): Ratio of detected frauds to total actual frauds.
* **F1-Score**: Harmonic mean of Precision and Recall.
* **ROC-AUC Score**: Area under the ROC curve, indicates model’s ability to distinguish between classes.

**6.5 Hyperparameter Tuning**

* GridSearchCV and RandomizedSearchCV were used for tuning hyperparameters in Random Forest and XGBoost models.
* Parameters such as n\_estimators, max\_depth, learning\_rate, and subsample were optimized.

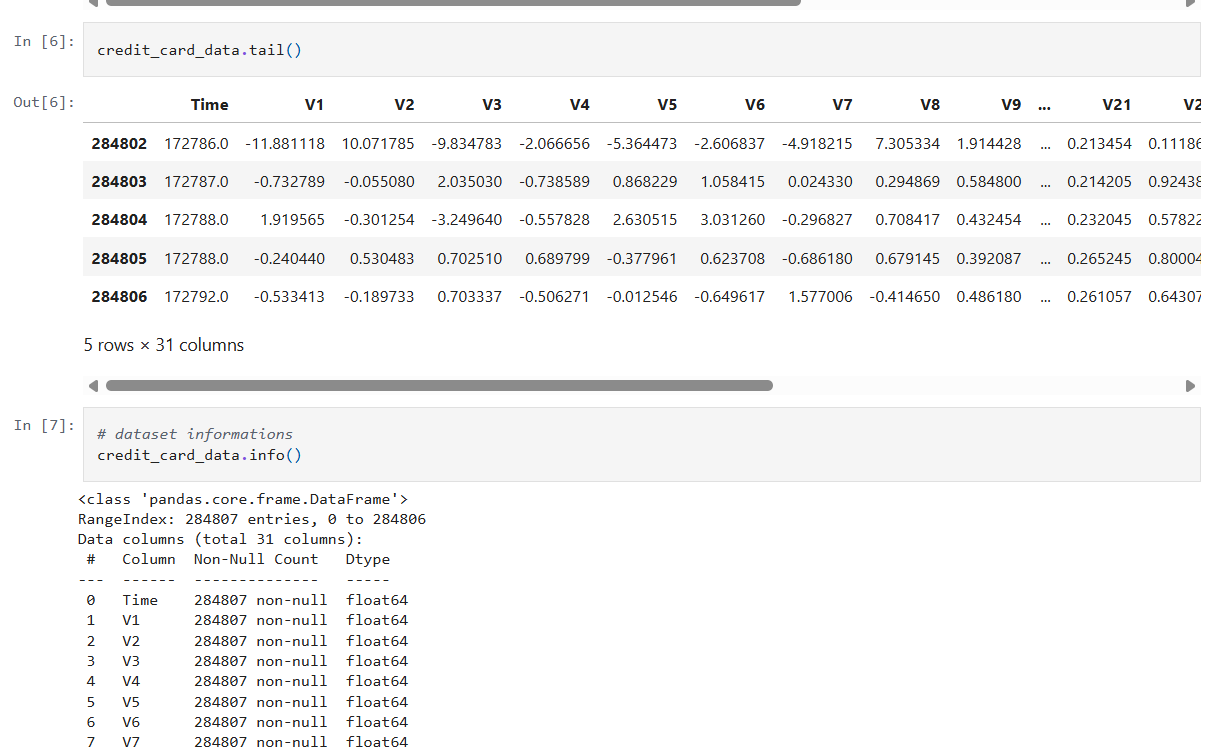
**6.5.1Final Model Selection**

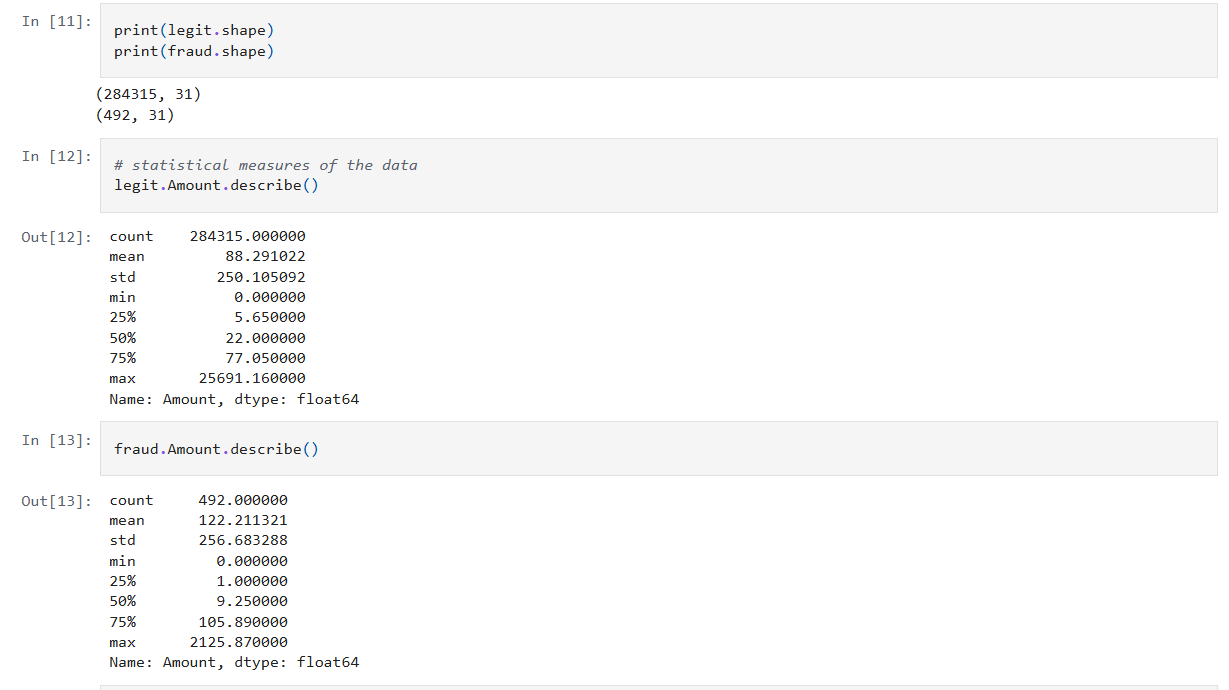
* XGBoost emerged as the best-performing model with:
  + High Recall (important for detecting most frauds)
  + Good Precision and F1-score
  + Minimal overfitting and fast prediction time

**6.5.2Interface Design (Optional Phase)**

* A basic web interface was created using Streamlit for demo purposes.
* Users can input transaction details and get instant fraud prediction output.
* Model is hosted locally for testing; potential for cloud deployment in the future.

****

****

****

****

**CHAPTER -7**

**RESULTS**

**7.1 Evaluation Metrics**  
The following metrics were used for evaluation:

* **Accuracy**: Overall correctness
* **Precision**: True positives over predicted positives
* **Recall**: True positives over actual positives
* **F1-Score**: Harmonic mean of precision and recall
* **ROC-AUC**: Trade-off between true positive and false positive rates

****

**CHAPTER-8**

**CONCLUSION**

This project presented a machine learning-based approach to credit card fraud detection. It addressed key challenges like class imbalance, real-time requirements, and model interpretability. Several algorithms were tested, and XGBoost demonstrated the best performance.

**Key Takeaways**:

* Data preprocessing and resampling are crucial for handling class imbalance.
* Ensemble learning models like Random Forest and XGBoost provide high accuracy and recall.
* Real-time fraud detection is feasible with optimized models.

**Limitations**:

* The dataset is anonymized and based on a specific region/time period.
* The models are only as good as the data they are trained on.

**Future Work**:

* Explore deep learning techniques such as autoencoders for anomaly detection.
* Implement real-time streaming fraud detection using tools like Apache Kafka and Spark.
* Incorporate blockchain for secure transaction logging and model explainability frameworks like SHAP.

**CHAPTER-9**

**REFERENCES**

**Kaggle-CreditCardFraudDetectionDataset**  
This dataset is publicly available on the Kaggle website. It contains information about credit card transactions made by European customers. The dataset is anonymized for privacy and is often used for testing fraud detection models.  
*Link:* https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

**SMOTE:Synthetic Minority Oversampling Technique**  
This is a technique used to deal with imbalanced datasets, where fraud cases are much fewer than normal cases. SMOTE creates synthetic samples of the minority class (fraudulent cases) to help the model learn better. The technique was proposed by Chawla et al. in 2002.  
*Citation:* Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16, 321–357.

 **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow – Aurélien Géron**  
This book is a popular resource for beginners and professionals learning machine learning. It explains many concepts used in your project like classification, preprocessing, model tuning, and evaluation in simple language with code examples in Python.  
*Author:* Aurélien Géron  
*Publisher:* O'Reilly Media, 2nd Edition, 2019

**Scikit-learnDocumentation**   
Scikit-learn is one of the most commonly used machine learning libraries in Python. Its official documentation provides detailed information about the algorithms, evaluation metrics, and preprocessing techniques used in this project.  
*Website:* <https://scikit-learn.org/>

**Imbalanced-learnDocumentation**  
This library provides tools like SMOTE and others to handle imbalanced datasets. The documentation offers usage examples and explanations of different resampling strategies.  
*Website:* <https://imbalanced-learn.org/>