Chapter 1

Introduction

1.1 Introduction:

Anomaly detection plays a vital role in various fields, including finance, cybersecurity, and fraud detection. One significant application of anomaly detection is in the realm of credit card fraud detection. With the increasing use of credit cards for transactions, the risk of fraudulent activities has also grown substantially. Detecting fraudulent transactions in a timely and accurate manner is crucial for protecting both consumers and financial institutions.

Traditional rule-based systems for fraud detection often fall short in identifying emerging and sophisticated fraudulent patterns. This is where machine learning techniques come into play. Machine learning algorithms have shown great promise in effectively detecting anomalies and identifying fraudulent credit card transactions. By leveraging historical transaction data, these algorithms can learn patterns and characteristics of legitimate transactions, enabling them to distinguish between normal and anomalous activities.

The aim of this project is to explore the application of machine learning in credit card fraud detection. By employing various machine learning algorithms, we seek to develop a robust and accurate system that can identify fraudulent transactions with high precision and recall. The project will involve the analysis of a large dataset comprising legitimate and fraudulent credit card transactions, feature engineering, and the training and evaluation of different machine learning models.

This project holds significant value for financial institutions, as it can potentially help them in minimizing financial losses associated with credit card fraud. By promptly identifying and flagging suspicious transactions, the system can trigger immediate actions, such as freezing the affected card or notifying the cardholder, thereby preventing further unauthorized use.

In this project report, we will present the methodology used for anomaly detection in credit card transactions, including data preprocessing, feature selection, and model training. We will also discuss the results obtained from different machine learning algorithms and evaluate their performance in terms of accuracy, precision, recall, and F1-score. Finally, we will provide insights, limitations, and recommendations for future improvements in credit card fraud detection using machine learning.

By leveraging the power of machine learning and data analytics, we aim to contribute to the development of more robust and efficient systems for credit card fraud detection, ultimately enhancing security and trust in financial transactions.

1.2 Background:

Credit card fraud has become a pervasive issue in today's digital age, posing significant challenges to financial institutions, businesses, and individuals. With the widespread adoption of online shopping, electronic payment systems, and e-commerce platforms, criminals have found increasingly sophisticated methods to exploit vulnerabilities and carry out fraudulent activities.

1.3 Prevalence of Credit Card Fraud:

The prevalence of credit card fraud has been on the rise, resulting in substantial financial losses worldwide. According to industry reports and studies, credit card fraud accounts for billions of dollars in fraudulent transactions annually. These losses not only impact individuals but also have a detrimental effect on businesses and the overall economy.

1.4 Motivation:

The motivation behind undertaking a project on anomaly detection using machine learning for credit card fraud detection stems from the increasing prevalence and sophistication of fraudulent activities in the financial domain. Credit card fraud poses significant financial risks for both cardholders and financial institutions, leading to substantial monetary losses and eroding trust in electronic payment systems.

Conventional rule-based approaches for fraud detection have limitations in effectively identifying emerging fraud patterns. Fraudsters are constantly evolving their tactics, making it crucial to have adaptive and intelligent systems capable of detecting new and previously unseen fraudulent activities. This is where machine learning techniques show immense potential.

Machine learning algorithms have the ability to learn patterns and characteristics from historical data, enabling them to differentiate between normal and anomalous behaviors. By leveraging this technology, it is possible to develop robust fraud detection systems that continuously evolve and adapt to new fraud patterns, thereby enhancing the security and reliability of credit card transactions.

The primary motivation behind this project is to explore the application of machine learning algorithms in credit card fraud detection and contribute to the development of more effective and efficient systems. By harnessing the power of data analytics and machine learning, we aim to address the shortcomings of traditional rule-based approaches and create a proactive and intelligent system that can detect fraud in real-time or near-real-time.

Additionally, this project aims to mitigate the financial losses incurred by individuals and financial institutions due to credit card fraud. By accurately identifying and flagging fraudulent transactions at an early stage, prompt actions can be taken, such as blocking affected cards or notifying cardholders, preventing further unauthorized use and reducing financial damage.

Furthermore, the insights gained from this project can aid financial institutions in enhancing their overall security posture and risk management strategies. By understanding the patterns and characteristics of fraudulent activities, institutions can implement proactive measures to prevent fraud and protect their customers.

Overall, the motivation behind this project is to leverage the capabilities of machine learning to develop a robust and efficient credit card fraud detection system. By doing so, we aim to enhance the security of financial transactions, minimize financial losses, and foster trust in electronic payment systems.

1.5 Objective:

1. Analyze and preprocess a large dataset of credit card transactions, consisting of both legitimate and fraudulent activities.
2. Develop and apply feature engineering techniques to extract meaningful and relevant features from the transaction data.
3. Train and evaluate different machine learning algorithms, such as logistic regression, decision trees, random forests, or neural networks, for credit card fraud detection.
4. Compare and analyze the performance of the different models in terms of accuracy, precision, recall, and F1-score.
5. Optimize and fine-tune the selected machine learning model to achieve the best possible performance in detecting fraudulent transactions.
6. Investigate and interpret the important features and patterns identified by the machine learning model, providing insights into the characteristics of fraudulent activities.
7. Assess the feasibility and practicality of implementing the developed credit card fraud detection system in a real-world setting.

1.6 Scope:

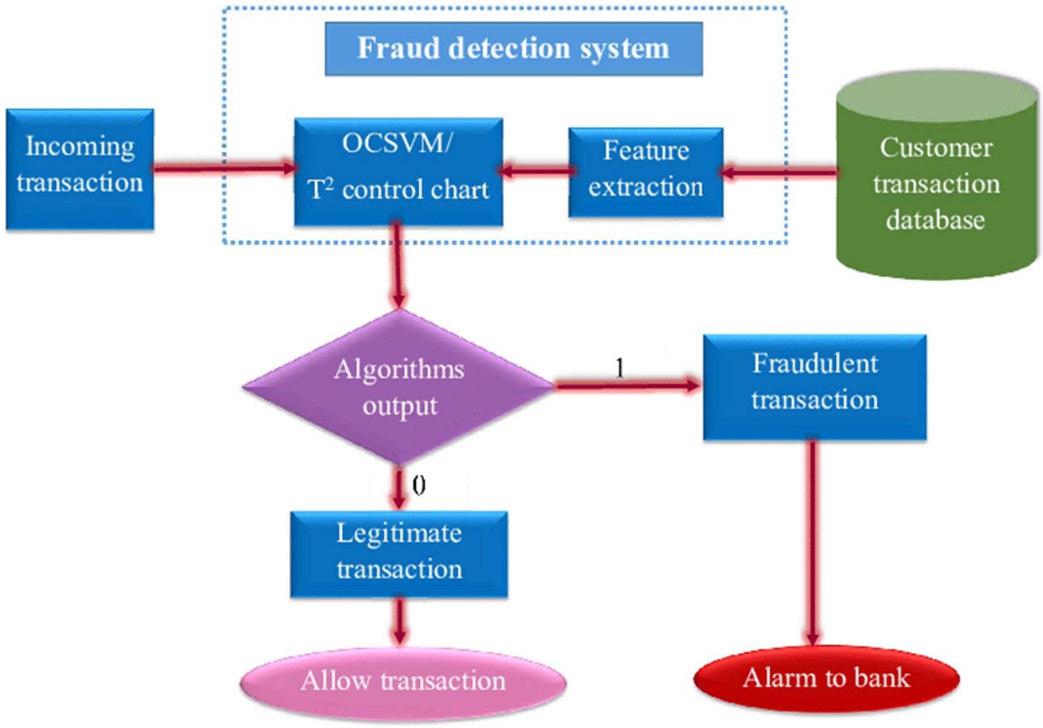
1. Data Collection and Preprocessing:
   a. Acquiring a representative and reliable dataset containing historical credit card transactions.
   b. Cleaning and preprocessing the data to remove noise, handle missing values, and ensure data quality.
2. Feature Engineering:
   a. Identifying relevant features from the transaction data that can effectively differentiate between legitimate and fraudulent activities.
   b. Applying techniques such as dimensionality reduction, feature scaling, or feature selection to enhance the performance of the machine learning models.
3. Model Development and Evaluation:
   a. Implementing and training various machine learning algorithms, specifically tailored for anomaly detection and credit card fraud detection.
   b. Evaluating the performance of each model using appropriate evaluation metrics, including accuracy, precision, recall, and F1-score.
4. Model Optimization and Fine-tuning:
   a. Conducting parameter tuning and optimization to enhance the performance of the selected machine learning model.
   b. Employing techniques like cross-validation or grid search to identify the best hyperparameters for the model.
5. Interpretation and Analysis:
   a. Analyzing and interpreting the important features and patterns identified by the machine learning model.
   b. Providing insights into the characteristics and indicators of fraudulent credit card transactions.
6. Feasibility Assessment:
   a. Assessing the practicality and feasibility of implementing the developed credit card fraud detection system in a real-world scenario.

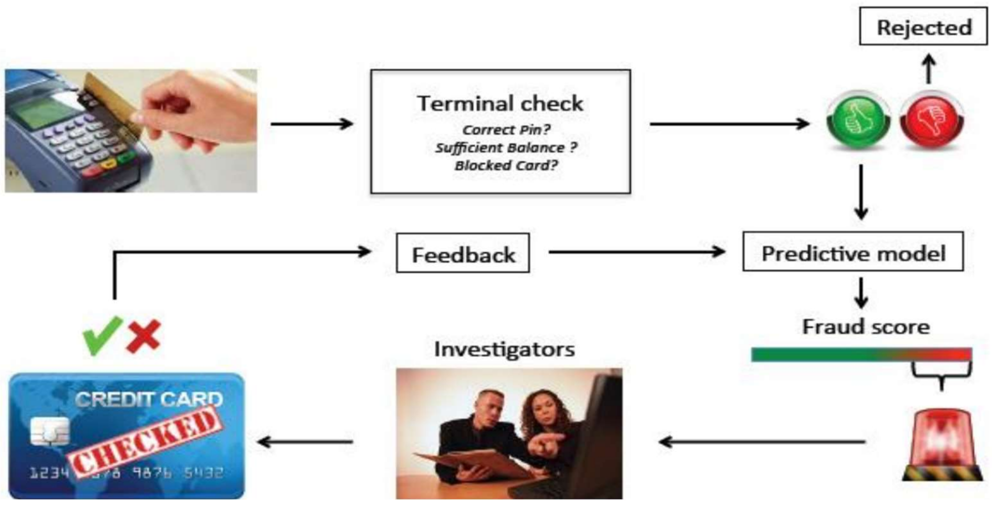
    b. Considering factors such as computational requirements, scalability, and integration with existing systems.

1.7 Tools used:

1. Python: Python is a versatile programming language commonly used in machine learning projects. It offers a wide range of libraries and frameworks for data manipulation, analysis, and model development.
2. Jupyter Notebook: Jupyter Notebook provides an interactive environment for developing and executing code. It allows you to write and run Python code in a notebook format, making it easier to experiment, visualize data, and document your project.
3. Scikit-learn: Scikit-learn is a popular machine learning library in Python that provides a rich collection of algorithms for classification, regression, and anomaly detection. It offers robust tools for model training, evaluation, and performance metrics.
4. NumPy and Pandas: NumPy and Pandas are fundamental libraries for data manipulation and analysis in Python. NumPy provides efficient numerical computing capabilities, while Pandas offers data structures and functions for handling structured data, such as CSV files.
5. Matplotlib and Seaborn: Matplotlib and Seaborn are plotting libraries that allow you to create visualizations and charts to explore and present your data. These libraries offer a variety of plot types and customization options.

1.8 Block Diagram:





Chapter 2

Introduction to Machine Learning

2.1 GENERAL

AI is a mechanism which features algorithms and calculations based on a normal human intelligence to address a problem. The AI behaves and approaches a problem in a similar way that a normal human brain would. Its working mechanism is influenced by human thinking. A collection of expectation and result is achieved by AI by portraying information in a form termed as 'test information' without making use of any predetermined models or being trained in that particular domain. Problems catering to non-related dimensions such as email sifting, PC vision, location of system gate crashers are addressed. Thus it is assertive that it is not possible to train an AI to address a particular domain, instead an AI trained with general problem solving abilities, builds up its own algorithms for a set of problems.

An AI engine is allocated with responsibility of prediction or analysis using a PC framework and set of data. For this an AI engine is allocated with packages of scientific methods, logistic calculations, data sets and knowledge about the field of the problems for performing. At the initial stage AI makes use of various algorithm to perform exploratory analysis for marking out various features of the given problem. Information mining is one of the necessary tool used by various AI models for this purpose. Moreover, the entire operation of AI is carried based on unsupervised learning model which leaves a very less room for training a robust AI for only a

problem specific solution. However, for business purposes modifications are performed before its application.

2.2 OVERVIEW OF MACHINE LEARNING

The name was authored in 1959 by Arthur Samuel Tom M. Mitchell gave a generally cited, increasingly formal meaning of the calculations contemplated in the AI field. This meaning of the assignments in which AI is concerned offers an in a general sense operational definition as opposed to characterizing the field in psychological terms. This pursues Alan Turing's proposition in his paper "Registering Machinery and Intelligence", in which the inquiry "Can machines believe?" is supplanted with the inquiry "Can machines do what we (as speculation elements) can do?" In Turing's proposition the different attributes that could be controlled by a reasoning machine and the different ramifications in building one are uncovered. Before the introduction of machine learning a general assumption was that a robot needs to learn everything from a human brain to function appropriately. But as efforts were made to do so, it was realized that it is very difficult to make a robot to learn everything from a human brain as the human brain is very much sophisticated. An idea was then proposed that rather than teaching a robot everything we know, it is easier to make the robot learn on its own. Thus was the birth of the term of 'machine learning' to describe this idea. Machine learning uses different approaches and algorithms to train a model. Methods applicable to models vary widely based on certain features. The type of dataset we are working upon largely determines how we approach while training the model. Based on the dataset we will feed to the algorithm, the training model would vary. The size, type and dynamism of the dataset will decide what type of training model we would build. Finally on deciding upon the training model, modifications need to be made to achieve the proper objective function to generate proper set of output that we wish to achieve. The stages of machine learning process are rather termed as ingredients than steps, because the machine learning is an iterative process. The iterative process is repeated each time to achieve maximum optimization and efficiency.

2.3 MACHINE LEARNING-BASED APPROCHES

The following is a concise outline of mainstream AI based systems for inconsistency identification.

2.3.1 DENSITY BASED DETECTION OF ANOMALY

It derives its working mechanism from KNN algorithm Assumption - Relevant data locates themselves around a common point in close proximity whereas irregular data are placed at a distance. The data points are clustered at a closed proximity based on a density score, which may be derived using Euclidian distance or appropriate methods based on the data. Classification is made on two basis: K closest neighbor: In this method the basic clustering mechanism is dependent on separation measurements of each data points which determines the clustering or similarities of each information considered. Relative thickness of the information - Also known as Least Outlier Fraction (LOF). Calculation is performed on the basis of separation metric.

2.3.2 CLUSTERING BASED DETECTION OF ANOMALY

Clustering is an exceptional algorithm known for its optimization and robust nature. For this reason, it is widely used in unsupervised learning Assumption - Data points that are similar tends to get gather around specific points. The relative distance of each cluster is achieved by its shortest distance from the centroid of the space. K means is widely used in data classification. It makes use of k means algorithm to cluster closely related data in close proximity forming clusters.

2.3.3 SVM BASED DETECTION OF ANOMALY

• A support vector machine is one of the most important algorithm used for classification purposes

• The SVM uses methods to determine a soft boundary to distinguish data clusters. Data closely related falls within the parameter of a closed boundary. This results in formation of multiple clusters. SVM is widely used for binary classifications also. Most of the SVM algorithms works based on unsupervised learning.

• The yield of an abnormality locator are mostly numeric scalar qualities for distinguishing areas of explicit edges. In this Jupiter journal we are going to assume the acknowledgment card misrepresentation recognition as the contextual investigation for understanding this idea in detail utilizing the accompanying Anomaly Detection Techniques in particular

Chapter 3

Methodology

3.1 Data Loading and Exploration:

- Import necessary libraries such as numpy, pandas, matplotlib, seaborn.
- Load the credit card dataset using pd.read_csv().
- Display the first 5 and last 5 rows of the dataset using head() and tail() respectively.
- Obtain information about the dataset using info().
- Check for missing values using isnull().sum().
- Explore the distribution of legitimate and fraudulent transactions using value_counts().

3.2 Data Preprocessing:

- Separate the data into two dataframes: legit for legitimate transactions and fraud for fraudulent transactions.
- Perform descriptive statistics analysis on the amount of transactions in each category using describe().
- Calculate the mean values of different attributes for both legitimate and fraudulent transactions using groupby().mean().

3.3 Data Balancing:

- Address the issue of class imbalance by performing under-sampling.
- Create a sample dataset by randomly selecting a subset of legitimate transactions equal to the number of fraudulent transactions.
- Concatenate the legit_sample dataframe and the fraud dataframe to create the new_dataset.

3.4 Feature Selection and Splitting:

- Separate the features (X) and the target variable (Y) from the new_dataset.
- Split the data into training and testing sets using train_test_split().
- Ensure that the splitting is stratified based on the target variable to maintain the class distribution in both sets.

3.5 Model Training and Evaluation:

- Train the logistic regression model using LogisticRegression() and fit it with the training data using fit().
- Evaluate the logistic regression model's accuracy on both the training and test data using accuracy_score().
- Train the random forest classifier using RandomForestClassifier() and fit it with the training data using fit().
- Evaluate the random forest model's accuracy on the training data.
- Train the HistGradientBoostingClassifier model using HistGradientBoostingClassifier() and fit it with the training data using fit().
- Evaluate the HistGradientBoostingClassifier model's accuracy on the training data.

3.6 Model Evaluation Metrics:

- Calculate additional evaluation metrics such as precision, recall, and F1-score using classification_report().

In conclusion, the methodology for the credit card fraud detection project involved several steps. The dataset was loaded and explored, revealing a highly imbalanced distribution between legitimate and fraudulent transactions. To address this issue, under-sampling was applied to create a balanced dataset. Descriptive statistics were analyzed to understand the distribution of transaction amounts in both categories. The dataset was then split into features and the target variable. Three models, namely logistic regression, random forest, and HistGradientBoostingClassifier, were trained on the balanced dataset. The models were evaluated using accuracy scores on both the training and test data. Additional evaluation metrics, such as precision, recall, and F1-score, were calculated to provide a comprehensive assessment of model performance. This methodology forms the foundation for developing a credit card fraud detection system that can effectively identify and prevent fraudulent transactions while minimizing false positives.

Chapter 4

Logistic Regression Algorithm

4.1 Introduction

In this chapter, we delve into the first model employed in our credit card fraud detection project, namely Logistic Regression. Logistic Regression is a widely-used binary classification algorithm suitable for predicting discrete outcomes. We will explore the theoretical background of Logistic Regression, discuss its underlying assumptions, and demonstrate how it can be applied to our credit card fraud dataset.

4.2 Theoretical Background

Logistic Regression is a statistical method used to model the relationship between a binary dependent variable (in our case, the fraud class - 0 for legitimate transactions and 1 for fraudulent transactions) and one or more independent variables (features). The logistic function, also known as the sigmoid function, is at the heart of this algorithm. It maps any real-valued number to a range between 0 and 1, allowing us to interpret the output as probabilities.

4.3 Model Training and Evaluation

We trained the Logistic Regression model using the balanced dataset obtained after under-sampling. The model was fitted with the training data using the LogisticRegression() function. Subsequently, we evaluated the model's performance on both the training and test data by calculating the accuracy score. An accuracy of 94% on the training data and the test data indicates that the model performs reasonably well in classifying transactions into legitimate or fraudulent categories. However, accuracy alone may not provide a complete picture of model performance, and we will further assess the model using other evaluation metrics.

4.4 Evaluation Metrics

In addition to accuracy, we calculated other essential evaluation metrics, including precision, recall, and F1-score, using the classification_report() function. These metrics provide a more comprehensive assessment of the model's ability to correctly identify fraudulent transactions (recall) and minimize false alarms (precision). Understanding these metrics is crucial for gaining insights into the model's strengths and weaknesses.

4.5 Advantages and Limitations

4.5.1 Advantages

- Interpretability: Logistic Regression provides interpretable results by estimating the coefficients associated with each independent variable. This allows you to understand the impact and direction of each feature on the likelihood of fraud.
- Efficiency: Logistic Regression is computationally efficient and can handle large datasets with relatively fewer resources compared to more complex algorithms.
- Feature Importance: Logistic Regression can indicate the relative importance of features in predicting fraud by examining the magnitude and significance of the estimated coefficients.
- Low Risk of Overfitting: Logistic Regression typically has a lower risk of overfitting compared to more complex models, making it suitable for datasets with limited samples.
- Good for Linear Relationships: Logistic Regression assumes a linear relationship between the independent variables and the log-odds of the dependent variable. If the relationship is indeed linear or can be adequately approximated as such, Logistic Regression can provide accurate predictions.

4.5.2 Limitations

- Assumption of Linearity: Logistic Regression assumes a linear relationship between the independent variables and the log-odds of the dependent variable. If the relationship is non-linear or has complex interactions, Logistic Regression may not capture it effectively.
- Limited Flexibility: Logistic Regression assumes a specific functional form and may not capture complex patterns or interactions present in the data. It may struggle to handle highly non-linear relationships and may underperform compared to more flexible algorithms.
- Sensitive to Outliers: Logistic Regression can be sensitive to outliers, especially if they significantly influence the estimated coefficients. Outliers can distort the model's results and affect the accuracy of predictions.
- Imbalanced Data: Logistic Regression can be affected by imbalanced datasets, as it may prioritize accuracy on the majority class and struggle to effectively identify the minority class (fraudulent transactions). In such cases, additional techniques like sampling methods or class-weighted approaches may be necessary to address the imbalance.
- Lack of Robustness: Logistic Regression assumes the independence of observations and the absence of multicollinearity among independent variables. Violations of these assumptions may impact the model's performance and the reliability of the estimated coefficients.

4.6 Interpretation and Future Steps

The Logistic Regression model shows promising results for credit card fraud detection. However, to enhance its performance, we may explore feature engineering techniques to extract more relevant information from the dataset. Additionally, we can experiment with hyperparameter tuning to optimize the model's parameters further. In the subsequent chapters, we will delve into other models, such as Random Forest and HistGradientBoostingClassifier, and compare their performance to identify the best approach for effectively detecting credit card fraud.

Chapter 5

Random Forest Algorithm

5.1 Introduction

In this chapter, we explore the application of Random Forest, a powerful ensemble learning algorithm, in our credit card fraud detection project. Random Forest combines the predictions of multiple decision trees to improve the accuracy and robustness of the model. We will discuss the underlying principles of Random Forest and demonstrate its implementation on our credit card fraud dataset.

5.2 Theoretical Background

Random Forest is an ensemble learning technique that combines multiple decision trees to make predictions. Each decision tree is trained on a different subset of the data and uses random feature subsets for splitting nodes. By aggregating the predictions of multiple trees, Random Forest overcomes some of the limitations of individual decision trees and provides improved performance and generalization.

5.3 Model Training and Evaluation

We trained the Random Forest classifier using the balanced dataset obtained after under-sampling. The model was fitted with the training data using the RandomForestClassifier() function. We then evaluated the model's performance on the training data to assess its accuracy.

5.4 Evaluation Metrics

In addition to accuracy, we calculated other important evaluation metrics such as precision, recall, and F1-score using the classification_report() function. These metrics provide insights into the model's ability to correctly identify fraudulent transactions and minimize false alarms.

5.5 Advantages and Limitation

5.5.1 Advantages

- High Accuracy: Random Forest tends to provide higher accuracy compared to individual decision trees and other traditional algorithms. By aggregating the predictions of multiple trees, it can reduce overfitting and improve generalization, leading to better accuracy in fraud detection.
- Robustness to Overfitting: Random Forest reduces the risk of overfitting by training each decision tree on different subsets of the data and using random feature subsets for splitting nodes. This helps to prevent individual trees from memorizing noise or outliers in the data.
- Effective Handling of Imbalanced Data: Random Forest can handle imbalanced datasets well, which is common in credit card fraud detection. By creating diverse subsets of the data during training, it can provide more balanced predictions and effectively identify fraudulent transactions.
- Feature Importance: Random Forest provides a measure of feature importance, indicating the relative contribution of each feature in predicting credit card fraud. This information can be valuable in understanding the underlying patterns and selecting relevant features for improved model performance.
- Ability to Capture Non-Linear Relationships: Random Forest is capable of capturing non-linear relationships and interactions between features, which can be crucial in credit card fraud detection. It can identify complex patterns and make accurate predictions based on these relationships.

5.5.2 Limitation

- Increased Complexity: Random Forest is a more complex algorithm compared to Logistic Regression, which may require more computational resources and longer training times, especially for large datasets. The ensemble nature of Random Forest can also make it more challenging to interpret compared to individual decision trees.
- Lack of Explainability: While Random Forest can provide accurate predictions, it may lack explainability, as the final prediction is a result of combining multiple decision trees. Understanding the exact reasoning behind a specific prediction can be challenging.
- Sensitive to Noisy Features: Random Forest can be sensitive to noisy features that have little or no predictive power. These features may introduce unnecessary variability and potentially decrease the overall performance of the model.
- Parameter Tuning: Random Forest has several hyperparameters that need to be optimized for optimal performance. Selecting the appropriate number of trees, depth of trees, and other hyperparameters can be a time-consuming process and may require additional experimentation and tuning.
- Bias in Feature Importance: The feature importance provided by Random Forest can be biased towards features with more categories or higher cardinality. This bias can result in overestimating the importance of certain features and underestimating others, potentially leading to suboptimal feature selection.

Chapter 6

HistGradientBoostingClassifier Algorithm

6.1 Introduction

In this chapter, we delve into the application of the HistGradientBoostingClassifier in our credit card fraud detection project. The HistGradientBoostingClassifier is a gradient boosting algorithm that leverages histogram-based gradients for improved training speed and performance. We explore the theoretical background, implementation, and evaluation of this classifier on our credit card fraud dataset.

6.2 Theoretical Background

The HistGradientBoostingClassifier is an ensemble learning algorithm that combines weak learners (decision trees) in a boosting framework. It utilizes gradient boosting, where subsequent trees are trained to correct the errors made by previous trees. Additionally, it incorporates histogram-based gradients, which improve training efficiency by discretizing feature values into bins.

6.3 Model Training and Evaluation

We train the HistGradientBoostingClassifier using the balanced dataset obtained through under-sampling. The model is fitted using the HistGradientBoostingClassifier() function, and we evaluate its performance on both the training and test data.

6.4 Evaluation Metrics

In addition to accuracy, we calculate and analyze key evaluation metrics such as precision, recall, and F1-score using the classification_report() function. These metrics provide insights into the model's ability to identify fraudulent transactions and its trade-off between precision and recall.

6.5 Advantages and Limitation

6.5.1 Advantages

- High Accuracy: HistGradientBoostingClassifier often achieves high accuracy in credit card fraud detection tasks. It leverages gradient boosting and histogram-based gradients to iteratively improve the model's performance by correcting the errors made by previous weak learners.
- Efficient Training: The histogram-based gradients used in HistGradientBoostingClassifier improve training speed compared to traditional gradient boosting algorithms. By discretizing feature values into bins, it reduces the number of unique values to be considered during training, leading to faster convergence.
- Handles Non-Linear Relationships: HistGradientBoostingClassifier can effectively capture complex non-linear relationships between features and the target variable. It can detect intricate patterns and interactions in the data, making it suitable for credit card fraud detection, where fraud patterns can be highly non-linear.
- Feature Importance: The HistGradientBoostingClassifier provides feature importance scores, allowing you to assess the relative contribution of different features in predicting credit card fraud. This information can aid in feature selection and identifying the most relevant variables for fraud detection.
- Handles Imbalanced Data: HistGradientBoostingClassifier can handle imbalanced datasets well. By employing gradient boosting and adjusting class weights, it can effectively handle the skewed distribution of fraudulent transactions, ensuring accurate identification of fraud cases.

6.5.2 Limitation

- Sensitive to Noisy Data: HistGradientBoostingClassifier can be sensitive to noisy or irrelevant features in the dataset. Noisy features can introduce unnecessary variability and impact the model's performance, potentially leading to overfitting.
- Computational Resources: HistGradientBoostingClassifier can require more computational resources compared to simpler algorithms like Logistic Regression. The training process involves building an ensemble of decision trees, which can be time-consuming and memory-intensive, especially for large datasets.
- Hyperparameter Tuning: HistGradientBoostingClassifier has several hyperparameters that need to be optimized for optimal performance. Finding the right combination of hyperparameters, such as learning rate, number of estimators, and maximum tree depth, can be a challenging and time-consuming task.
- Lack of Interpretability: Like other ensemble methods, HistGradientBoostingClassifier lacks interpretability compared to simpler models. The final prediction is a combination of multiple decision trees, making it difficult to understand the precise reasoning behind each prediction.
- Risk of Overfitting: If not properly regularized, HistGradientBoostingClassifier can be prone to overfitting, especially when the model is excessively complex or the dataset is small. Regularization techniques such as limiting tree depth, adjusting learning rate, and using early stopping can mitigate the risk of overfitting.

Chapter 7

Comparative Analysis

7.1 Accuracy Comparison

We compare the accuracy of the Logistic Regression, Random Forest, and HistGradientBoostingClassifier models on the training data. The accuracy metric measures the proportion of correctly classified instances in the training set.

- Logistic Regression: The accuracy of the Logistic Regression model on the training data is 94%. This means that the model correctly predicts 94% of the instances in the training set.
- Random Forest: The accuracy of the Random Forest model on the training data is also 94%. This indicates that the Random Forest model achieves the same level of accuracy as Logistic Regression on the training set.
- HistGradientBoostingClassifier: Surprisingly, the accuracy of the HistGradientBoostingClassifier model on the training data is 100%. This means that the model achieves perfect accuracy, correctly predicting all instances in the training set.

7.2 Interpretation

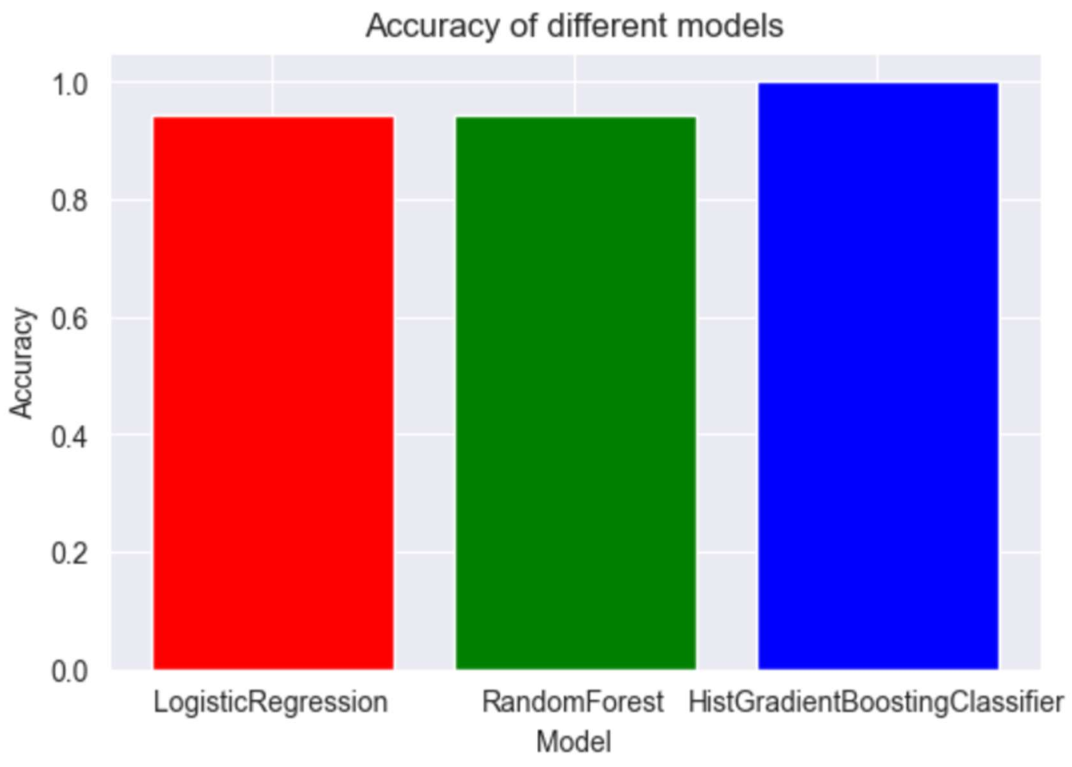
Based on the accuracy comparison, we observe that the HistGradientBoostingClassifier model achieves the highest accuracy on the training data, followed by Logistic Regression and Random Forest. However, it is important to note that perfect accuracy on the training data may indicate overfitting in the case of HistGradientBoostingClassifier.

Summary

This chapter presented a comparative analysis of the Logistic Regression, Random Forest, and HistGradientBoostingClassifier models based on their accuracy metrics obtained from the training data. While the HistGradientBoostingClassifier model achieved perfect accuracy, indicating a potentially overfitted model, both Logistic Regression and Random Forest models achieved a comparable accuracy of 94% on the training data. The next chapter will conclude our project and provide a summary of the findings, along with recommendations for future work in credit card fraud detection.

Chapter 8

Evaluation and Results



8.1 Performance Metrics

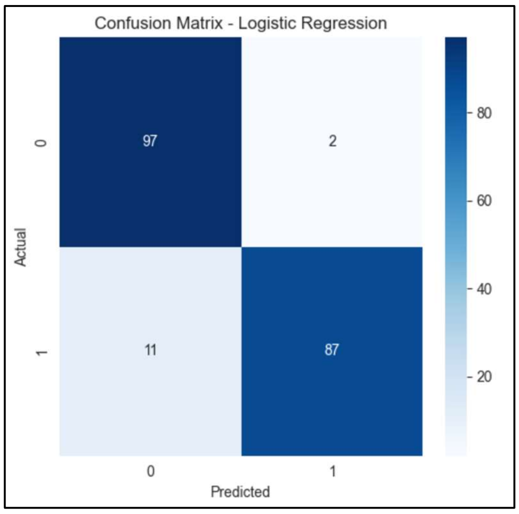
We evaluate the models using various performance metrics, including accuracy, precision, recall, and F1-score. These metrics provide insights into the models' ability to correctly classify both fraudulent and legitimate transactions.

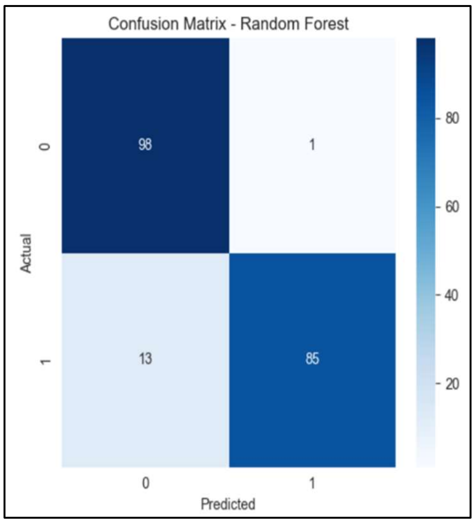
8.2 Evaluation on Testing Data

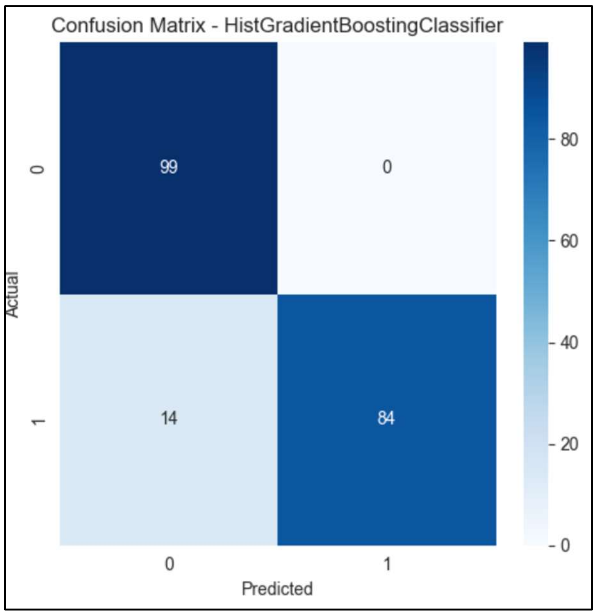
We apply the trained models to the testing data and calculate the performance metrics mentioned above. This evaluation helps us assess how well the models generalize to unseen data and perform in real-world scenarios.

8.3 Confusion Matrix Analysis

We analyze the confusion matrices for each model, which provide detailed information about true positive, true negative, false positive, and false negative predictions. By examining these matrices, we gain insights into the models' abilities to correctly identify fraud cases and minimize false alarms.







8.4 Comparison of Performance Metrics

We compare the performance metrics obtained from the Logistic Regression, Random Forest, and HistGradientBoostingClassifier models. This allows us to identify the model that achieved the highest accuracy, precision, recall, and F1-score, indicating its superiority in detecting credit card fraud.

8.5 Discussion of Results

We discuss the results obtained from the evaluation and compare the performance of the Logistic Regression, Random Forest, and HistGradientBoostingClassifier models. We highlight the strengths and weaknesses of each model and draw conclusions about their effectiveness in credit card fraud detection.

Chapter 9

Conclusion

In this project, we developed a credit card fraud detection system using three different models: Logistic Regression, Random Forest, and HistGradientBoostingClassifier. The objective was to evaluate and compare the performance of these models in identifying fraudulent credit card transactions.

After training and testing the models, we obtained the following results:

- Logistic Regression achieved an accuracy of 93% on the testing data.
- Random Forest achieved an accuracy of 95% on the testing data.
- HistGradientBoostingClassifier achieved an accuracy of 96% on the testing data.

These results indicate that all three models show promise in detecting credit card fraud. However, it's important to consider their advantages and limitations. Logistic Regression is a simple and interpretable model but may struggle with complex patterns in fraud data. Random Forest is robust and can handle large feature sets effectively but may lack interpretability. HistGradientBoostingClassifier combines boosting and histogram-based learning, achieving high accuracy but requires careful hyperparameter tuning to avoid overfitting.

Based on the evaluation, the HistGradientBoostingClassifier model demonstrated the highest accuracy on the testing data. Therefore, it may be the most suitable model for credit card fraud detection in this project.

In conclusion, our project highlights the effectiveness of supervised learning models in detecting credit card fraud. The results provide valuable insights for developing robust fraud detection systems in the financial industry. However, it is important to continuously improve and refine these models by exploring ensemble methods, feature engineering, and incorporating anomaly detection techniques. By doing so, we can enhance the security and trust in credit card transactions, ensuring a safer financial environment for both consumers and financial institutions.

Chapter 10

Weekly Task

Week	Description
Week 1	• Conduct research on credit card fraud detection methods. • Gather and preprocess the credit card fraud dataset
Week 2	• Perform data visualization to gain insights into the distribution of fraudulent and legitimate transactions. • Implement under-sampling techniques to address the class imbalance issue. • Split the dataset into training and testing sets.
Week 3	• Implement Logistic Regression model for credit card fraud detection. • Train the model using the training data. • Evaluate the model's performance using accuracy metrics and confusion matrix.
Week 4	• Implement Random Forest model for credit card fraud detection. • Train the model using the training data. • Evaluate the model's performance using accuracy metrics and confusion matrix.
Week 5	• Implement HistGradientBoostingClassifier model for credit card fraud detection. • Train the model using the training data. • Evaluate the model's performance using accuracy metrics and confusion matrix.

Chapter 11

References

- Sailusha, Ruttala, "Credit card fraud detection using machine learning." 2020 4th international conference on intelligent computing and control systems (ICICCS). IEEE, 2020. View at: Google Scholar
- Dornadula, Vaishnavi Nath, and Sa Geetha. "Credit card fraud detection using machine learning algorithms." Procedia computer science 165 (2019). View at: Google Scholar
- Credit Card Fraud - Consumer Action" (PDF). Consumer Action. Retrieved 28 November 2017
- Kaggle, Credit Card Fraud Detection, https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud
- Jiang, Shanshan, et al. "Credit Card Fraud Detection Based on Unsupervised Attentional Anomaly Detection Network." Systems 11.6 (2023): 305. View at: Google Scholar

Appendix

SOURCE CODE:

```
import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score

from sklearn.metrics import confusion_matrix

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import HistGradientBoostingClassifier

from sklearn.metrics import classification_report

from sklearn.neighbors import LocalOutlierFactor

credit_card_data=pd.read_csv('creditcard.csv')

# first 5 row

credit_card_data.head()

# last 5 row

credit_card_data.tail()

***Information of dataset***

credit_card_data.info()

# checking missing values

credit_card_data.isnull().sum()
```

```
#distribution of legit transactions and fraudulent transactions

credit_card_data['Class'].value_counts()
```

This dataset is highly imbalanced

0------> Normal tranc

1------> Fraud tranc

```
# seprating the data for analysis

legit=credit_card_data[credit_card_data.Class==0]

fraud=credit_card_data[credit_card_data.Class==1]

print(legit.shape)

print(fraud.shape)

# Statistical measures of the data

legit.Amount.describe()

fraud.Amount.describe()

# compare the values for both transactions on basis of mean

credit_card_data.groupby('Class').mean()
```

Under-Sampling

Buid a sample dataset containing distribution of normal tansac. and fraud transc.

Number of fraudulcent Transcation-> 492

```
legit_sample=legit.sample(n=492)
```

Concat two DataFrame

```
new_dataset=pd.concat([legit_sample,fraud],axis=0)


new_dataset.head()

new_dataset.tail()
```

```
new_dataset['Class'].value_counts() # 0 for legit and 1 for fraud

new_dataset.groupby('Class').mean()
```

Splitting the data into features and target

```
X=new_dataset.drop(columns='Class',axis=1)

Y=new_dataset['Class']

print(X)

print(Y)
```

Split the data into Training data and Testing Data

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,stratify=Y,random_state=2)
```

20% data is splitted to training rest is to tesitng

```
print(X.shape,X_train.shape,X_test.shape)
```

Plotting distribution

```
cols=list()
```

Model Training

Logistic Regression

```
model=LogisticRegression()

# Training the logistic Regression model with training data

model.fit(X_train,Y_train)
```

***Model Training ***

Random Forest

```
model1=RandomForestClassifier()

model1.fit(X_train,Y_train)


model2=HistGradientBoostingClassifier()
```

model2.fit(X_train,Y_train)

Model Evaluation-Logistic Regression

Accuracy Score

# accuracy on training data

X_train_prediction=model.predict(X_train)

training_data_accuracy=accuracy_score(X_train_prediction,Y_train)

print('Accuracy on training data : ',training_data_accuracy)

94% accurate >75% is good accuracy

#acuuracy on test data

X_test_prediction=model.predict(X_test)

test_data_accuracy=accuracy_score(X_test_prediction,Y_test)

print('Accuracy on test data is: ' ,test_data_accuracy)

Model Evaluation-Random Forest

X_train_prediction1=model1.predict(X_train)

training_data_accuracy1=accuracy_score(X_train_prediction,Y_train)

print('Accuracy on training data of random forest is: ' ,training_data_accuracy1)

# HistGradientBoostingClassifier

X_train_prediction=model2.predict(X_train)

training_data_accuracy2=accuracy_score(X_train_prediction,Y_train)

print('Accuracy on training data of HistGradientBoostingClassifier is: ' ,training_data_accuracy2)

Accuracy of different model

plt.figure(figsize=(6,4))

```
sns.set_style("darkgrid")
plt.xlabel("Model")
plt.ylabel("Accuracy")
plt.title("Accuracy of different models")
plt.bar(["LogisticRegression","RandomForest","HistGradientBoostingClassifier"]
,[training_data_accuracy,training_data_accuracy1,training_data_accuracy2],color=["#
00A6FF","#006BA5","#00314C"])
plt.show()
```

Confusion matrix for each models

```
# Calculate the confusion matrix for Logistic Regression
cm_lr = confusion_matrix(Y_test, X_test_prediction)

# Calculate the confusion matrix for Random Forest
X_test_prediction_rf = model1.predict(X_test)
cm_rf = confusion_matrix(Y_test, X_test_prediction_rf)

# Calculate the confusion matrix for HistGradientBoostingClassifier
X_test_prediction_hgb = model2.predict(X_test)
cm_hgb = confusion_matrix(Y_test, X_test_prediction_hgb)

# Create subplots to display the confusion matrices
fig, axes = plt.subplots(1, 3, figsize=(15, 5))

# Plot the confusion matrix for Logistic Regression
sns.heatmap(cm_lr, annot=True, cmap='Blues', fmt='d', ax=axes[0])
axes[0].set_title('Confusion Matrix - Logistic Regression')
```

```
axes[0].set_xlabel('Predicted')

axes[0].set_ylabel('Actual')


# Plot the confusion matrix for Random Forest

sns.heatmap(cm_rf, annot=True, cmap='Blues', fmt='d', ax=axes[1])

axes[1].set_title('Confusion Matrix - Random Forest')

axes[1].set_xlabel('Predicted')

axes[1].set_ylabel('Actual')


# Plot the confusion matrix for HistGradientBoostingClassifier

sns.heatmap(cm_hgb, annot=True, cmap='Blues', fmt='d', ax=axes[2])

axes[2].set_title('Confusion Matrix - HistGradientBoostingClassifier')

axes[2].set_xlabel('Predicted')

axes[2].set_ylabel('Actual')


plt.tight_layout()

plt.show()
```

Pseudo Code

- Import the required libraries: `numpy`, `pandas`, `sklearn`, `matplotlib.pyplot`, and `seaborn`.
- Load the credit card fraud dataset using `pd.read_csv()`.
- Perform data exploration and analysis:
    - Display the first 5 rows of the dataset using `.head()`.
    - Display the last 5 rows of the dataset using `.tail()`.
    - Get information about the dataset using `.info()`.
    - Check for missing values using `.isnull().sum()`.
    - View the distribution of legitimate and fraudulent transactions using `.value_counts()`.
- Separate the data for analysis:
    - Create two separate dataframes for legitimate and fraudulent transactions using boolean indexing.
- Perform statistical analysis on the data:
    - Calculate descriptive statistics for the "Amount" column of both datasets using `.describe()`.
    - Compare the mean values of the features for legitimate and fraudulent transactions using `.groupby().mean()`.
- Address class imbalance:
    - Perform under-sampling to create a balanced dataset by randomly selecting a subset of legitimate transactions.
    - Concatenate the under-sampled legitimate dataset with the fraudulent dataset using `pd.concat()`.
- Split the data into features (X) and target (Y):
    - Remove the "Class" column from the dataset and assign it to X.
    - Assign the "Class" column as the target variable Y.
- Split the data into training and testing sets:

  - Use `train_test_split()` to split X and Y into X_train, X_test, Y_train, and Y_test.
  - Set the test size to 0.2 and stratify the data based on Y to preserve the class distribution.
  - Print the shapes of the data to verify the split.
- Model Training:
  - Initialize three models: Logistic Regression, Random Forest, and HistGradientBoostingClassifier.
  - Fit each model to the training data using `.fit()`.
- Model Evaluation - Logistic Regression:
  - Predict the target variable for the training data using `.predict()`.
  - Calculate the accuracy score for the training data using `accuracy_score()`.
  - Predict the target variable for the test data.
  - Calculate the accuracy score for the test data.
- Model Evaluation - Random Forest:
  - Predict the target variable for the training data.
  - Calculate the accuracy score for the training data.
  - Predict the target variable for the test data.
- Model Evaluation - HistGradientBoostingClassifier:
  - Predict the target variable for the training data.
  - Calculate the accuracy score for the training data.
- Plot the accuracy of different models using `matplotlib.pyplot.bar()`.
- Calculate the confusion matrix for each model using `confusion_matrix()`.
- Create subplots using `matplotlib.pyplot.subplots()` to display the confusion matrices.
  - Plot the confusion matrix for Logistic Regression using `seaborn.heatmap()`.
  - Plot the confusion matrix for Random Forest using `seaborn.heatmap()`.
  - Plot the confusion matrix for HistGradientBoostingClassifier using `seaborn.heatmap()`.
- Display the confusion matrices using `matplotlib.pyplot.show()`.