

Artificial Intelligence and Machine Learning

Project Report

Semester-IV (Batch-2022)

Fraud Sleuth – Online Payment Fraud Detection

CHITKARA
UNIVERSITY



Supervised By:

Faculty Name – Ms. Shagun

Submitted By:

Name – Group 4 – Fraud Sleuth

Roll Number – 2210990365

2210990366

2210990421

2210990422

Group – G-9

**Department of Computer Science and Engineering
Chitkara University Institute of Engineering & Technology,
Chitkara University, Punjab**

1) Introduction

1.1 Background

In recent years, the global economy has undergone a profound transformation, characterized by a remarkable surge in digital transactions. This shift has been propelled by the rapid proliferation of e-commerce platforms, mobile payment systems, and online banking services, which have fundamentally altered the way consumers and businesses engage in financial transactions. With the convenience of shopping, banking, and conducting business online, individuals worldwide have embraced the digital economy, leading to an unprecedented growth in online commerce.

This digital revolution has undoubtedly brought about numerous benefits, offering unparalleled convenience, accessibility, and efficiency to users. Gone are the days of cumbersome physical transactions and lengthy queues; instead, consumers can now make purchases, transfer funds, and manage their finances with just a few clicks or taps on their devices. Moreover, businesses have capitalized on the global reach of the internet to expand their customer base, streamline operations, and drive revenue growth.

However, amid the widespread adoption of digital payment methods, a dark shadow looms large: the escalating threat of online payment fraud. Online payment fraud encompasses a range of illicit activities perpetrated during electronic transactions, including the unauthorized use of credit/debit card information, identity theft, account takeover, and various forms of cybercrime. Fraudsters, leveraging sophisticated techniques and exploiting vulnerabilities in payment systems, pose a significant risk to both consumers and businesses, resulting in substantial financial losses and damage to trust and reputation.

The rise of online payment fraud can be attributed to several factors, including the growing sophistication of cybercriminals, the increasing volume of digital transactions, and the inherent vulnerabilities in online payment infrastructure. Fraudsters constantly evolve their tactics to evade detection, employing tactics such as phishing attacks, malware infections, and social engineering schemes to compromise user accounts and steal sensitive information. Moreover, the anonymity and global reach of the internet provide fertile ground for criminals to operate with impunity across borders, making it challenging for law enforcement agencies to combat online fraud effectively.

In response to these challenges, there is a pressing need for innovative solutions to enhance the security and integrity of digital payment ecosystems. Traditional fraud prevention measures, such as rule-based systems and manual review processes, are no longer sufficient in the face of increasingly sophisticated threats. Instead, there is a growing recognition of the potential of machine learning algorithms and data analytics techniques to detect and prevent online payment fraud in real-time.

The proposed project seeks to leverage these advanced technologies to develop a proactive solution for online payment fraud detection. By analysing vast amounts of transaction data, including user behaviour, transaction history, device fingerprinting, and other relevant variables, machine learning models can identify patterns, trends, and anomalies indicative of fraudulent activity. Moreover, by continuously learning from new data and adapting to evolving threats, these models can improve their accuracy and effectiveness over time, minimizing false positives and enhancing the overall security of digital payment systems.

In tandem with the surge in digital transactions, the sophistication of online payment fraud techniques has grown exponentially. Fraudsters continuously adapt their tactics to exploit weaknesses in payment systems, often leveraging advanced technologies and exploiting human vulnerabilities through social engineering. From sophisticated phishing emails to malware-infected websites and fake mobile apps, the methods employed by cybercriminals are increasingly deceptive and difficult to detect. As a result, the detection and prevention of online payment fraud require a multi-faceted approach that combines technological innovation with user education and awareness.

One significant challenge in combating online payment fraud is the balance between security and user experience. While stringent security measures such as multi-factor authentication and transaction monitoring enhance protection against fraud, they can also introduce friction into the user experience, potentially leading to frustration and abandonment of transactions. As such, there is a delicate equilibrium to be struck between implementing robust security measures and maintaining a seamless and frictionless payment experience for users. Finding this balance is essential to ensure that security measures do not impede the growth of digital commerce or compromise user trust and satisfaction.

In summary, the rise of online payment fraud presents a significant challenge to the digital economy, threatening the trust, security, and stability of online transactions. By harnessing the power of machine learning and data analytics, however, it is possible to develop proactive solutions that can mitigate these risks and safeguard the integrity of digital payment ecosystems.

1.2 Objectives

1. Develop a Machine Learning Model:

Utilize historical transaction data: Historical transaction data serves as the foundation for training the machine learning model. This data provides valuable insights into patterns and trends associated with both legitimate and fraudulent transactions, enabling the model to learn distinguishing features.

Explore various machine learning algorithms: Experiment with a range of machine learning algorithms, including supervised learning techniques such as logistic regression, decision trees, random forests, support vector machines (SVM), and deep learning approaches like artificial neural networks. This exploration allows for the selection of the most suitable algorithm based on the characteristics of the data and the complexity of the fraud detection task.

Perform data preprocessing: Preprocess the transaction data to handle missing values, outliers, and categorical variables. Data preprocessing techniques such as normalization, standardization, and feature scaling help ensure the quality and consistency of the input data, facilitating effective model training.

Conduct feature selection: Identify the most relevant features or attributes that contribute to distinguishing between legitimate and fraudulent transactions. Feature selection techniques such as recursive feature elimination, principal component analysis (PCA), or correlation analysis help reduce dimensionality and improve model performance.

Evaluate model performance: Assess the performance of the machine learning model using appropriate evaluation metrics such as accuracy, precision, recall, and F1-score. Conduct cross-validation and use techniques such as confusion matrices and ROC curves to evaluate the model's ability to generalize to unseen data and its trade-offs between false positives and false negatives.

2. Achieve High Accuracy:

Strike a balance between false positives and false negatives: Aim to minimize both false positives (legitimate transactions incorrectly flagged as fraudulent) and false negatives (fraudulent transactions incorrectly classified as legitimate) to achieve high accuracy in fraud detection. This balance ensures effective identification of fraudulent activities while minimizing disruptions to legitimate transactions.

Optimize performance metrics: Focus on optimizing performance metrics such as precision, recall, and the F1-score to achieve high accuracy. Precision measures the accuracy of positive predictions, recall measures the ability to capture all positive instances, and the F1-score provides a harmonic mean of precision and recall, offering a balanced assessment of model performance.

Fine-tune model parameters: Fine-tune the parameters of the machine learning model, such as regularization parameters, learning rates, and thresholds, to optimize performance metrics. This iterative process involves experimenting with different parameter settings and evaluating their impact on model accuracy and stability.

Incorporate ensemble methods: Explore ensemble learning techniques such as bagging, boosting, and stacking to combine multiple models and improve overall predictive performance. Ensemble methods leverage the diversity of individual models to enhance accuracy and robustness, particularly in complex fraud detection scenarios.

Continuously monitor and update the model: Implement mechanisms for ongoing model monitoring and updates to adapt to changing patterns of fraudulent activity. Regularly retrain the model with new data and incorporate feedback from real-world performance to maintain high accuracy over time.

3. Implement Scalable Solution:

Design modular and adaptable software components: Develop a fraud detection system comprising modular software components that can be easily integrated into existing online payment systems. Modular design facilitates scalability and enables seamless updates and enhancements to the system architecture.

Leverage distributed computing and parallel processing: Utilize distributed computing and parallel processing techniques to handle large volumes of transactional data efficiently. Technologies such as Hadoop, Spark, and MapReduce enable parallel processing of data across multiple nodes, improving scalability and performance.

Utilize cloud infrastructure: Harness the scalability and flexibility of cloud computing platforms to deploy and scale the fraud detection system dynamically. Cloud services such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform offer on-demand resources and elastic scaling capabilities, facilitating efficient processing of transaction data.

Implement batch and real-time processing: Support both batch processing and real-time processing of transaction data to accommodate different use cases and requirements. Batch processing allows for the analysis of large historical datasets, while real-time processing enables immediate detection and response to fraudulent activities as they occur.

Monitor system performance and scalability: Continuously monitor the performance and scalability of the fraud detection system to identify bottlenecks and optimize resource utilization.

1.3 Significance

The significance of this project extends to multiple stakeholders within the digital payment ecosystem, including consumers, businesses, and financial institutions:

Certainly, let us delve deeper into the significance of the project across different dimensions:

1. Consumer Protection:

Financial Security: The project directly addresses consumers' concerns regarding the security of their financial transactions in the digital realm. By deploying advanced fraud detection techniques, it acts as a shield against unauthorized access to sensitive information and fraudulent activities, thus safeguarding consumers from financial losses and identity theft.

Trust and Confidence: When consumers perceive online payment systems as secure and reliable, they are more likely to embrace digital transactions with confidence. This trust not only encourages continued engagement with e-commerce platforms but also fosters a positive perception of digital payment methods, ultimately driving the growth of the digital economy.

Peace of Mind: Knowing that their transactions are being monitored and protected, consumers can enjoy greater peace of mind when conducting online purchases or managing their finances. This sense of security enhances the overall user experience and encourages sustained adoption of digital payment technologies.

2. Business Reputation:

Customer Trust: For businesses, maintaining the trust and confidence of their customer base is paramount for long-term success. By implementing effective fraud detection measures, companies demonstrate their commitment to protecting their customers' interests and ensuring the integrity of transactions. This proactive approach not only minimizes the financial impact of fraud but also helps preserve the reputation of the business as a trustworthy entity in the digital marketplace.

Brand Loyalty: A strong reputation for security and reliability fosters brand loyalty among customers, encouraging repeat purchases and referrals. Businesses that prioritize security and invest in fraud prevention measures are more likely to retain customers and attract new ones, thereby gaining a competitive edge in the digital landscape.

Regulatory Compliance: Compliance with industry regulations and standards is essential for maintaining business credibility and avoiding legal repercussions. By adhering to best practices in fraud detection and prevention, companies can demonstrate compliance with regulatory requirements, bolstering their reputation as responsible corporate citizens.

3. Industry Integrity:

Collaborative Efforts: The project's contribution to enhancing the integrity and security of the digital payment industry extends beyond individual businesses to the entire ecosystem. Collaboration among stakeholders, including businesses, financial institutions, technology providers, and regulatory bodies, is essential for addressing common challenges and setting industry-wide standards for security and compliance.

Innovation and Growth: A secure and trustworthy digital payment environment fosters innovation, investment, and growth within the industry. By mitigating the risks associated with online payment fraud, the project creates a conducive environment for businesses to explore new technologies, expand their operations, and capitalize on emerging opportunities in the digital economy.

Consumer Confidence: Ultimately, the success of the digital payment industry hinges on consumer confidence and trust. By implementing robust fraud detection measures and ensuring compliance with regulatory requirements, stakeholders collectively enhance consumer confidence in digital payment systems, driving continued adoption and growth.

In essence, the significance of the project transcends individual transactions or businesses, encompassing broader implications for consumer protection, business reputation, and the integrity of the digital payment ecosystem. Through collaborative efforts and a commitment to innovation and security, stakeholders can collectively foster a safer, more resilient, and more trustworthy environment for digital transactions.

2) Problem Definition and Requirements

2.1 Problem Definition

The challenge of online payment fraud detection using machine learning revolves around developing algorithms and systems capable of accurately identifying fraudulent transactions across various digital payment platforms. This task entails detecting a broad spectrum of fraudulent activities, ranging from unauthorized transactions and stolen card information to sophisticated account takeovers and other deceptive behaviours orchestrated by malicious actors. Unlike traditional fraud detection methods, which rely on predefined rules or patterns, machine learning-based approaches aim to learn and adapt from historical transaction data to discern subtle and evolving patterns indicative of fraud.

The complexity of online payment fraud detection stems from the sheer volume and intricacy of digital transactions occurring in real-time across diverse platforms and channels. With millions of transactions processed daily, distinguishing between legitimate and fraudulent transactions requires sophisticated algorithms capable of processing vast amounts of data efficiently. Moreover, fraudulent activities often exhibit subtle and nuanced patterns that can evade detection by conventional rule-based systems, necessitating the adoption of advanced machine learning techniques to uncover hidden fraud signals.

One of the primary challenges lies in swiftly and accurately identifying fraudulent transactions amidst the noise of legitimate transactions. This requires machine learning models to analyse multiple features and variables associated with each transaction, such as transaction amount, location, device information, and user behaviour, to discern anomalous patterns indicative of fraud. Additionally, the models must operate in real-time, making rapid decisions within milliseconds to prevent fraudulent transactions from being approved.

Furthermore, the effectiveness of online payment fraud detection systems hinges on their ability to continually adapt and refine their detection capabilities to keep pace with the evolving tactics employed by fraudsters. As fraudsters develop new techniques to circumvent detection, machine learning models must be agile and proactive in identifying emerging threats and adjusting their detection strategies accordingly. This necessitates continuous monitoring of model performance, retraining with updated data, and fine-tuning of algorithms to maintain high accuracy and efficacy in fraud detection.

In addition to algorithmic complexity, seamlessly integrating machine learning models within existing payment processing systems presents a significant implementation challenge. The integration process involves overcoming technical barriers, ensuring compatibility with existing infrastructure, and addressing regulatory compliance requirements. Moreover, deploying machine learning models in production environments requires robust monitoring and validation mechanisms to verify their performance and mitigate the risk of false positives or negatives.

2.2 Requirements

Software Requirements:

1. Python: Python is chosen as the primary programming language for its widespread adoption, robustness, and extensive support for data science and machine learning tasks. Its readability and ease of learning make it an ideal choice for both beginners and experienced developers.

2. Libraries:

Scikit-learn: This library provides simple and efficient tools for data mining and data analysis, making it a cornerstone for building machine learning models. It offers various algorithms for classification, regression, clustering, dimensionality reduction, and more.

Pandas: Pandas is used for data manipulation and analysis, offering data structures like DataFrame for easy handling of structured data. It provides functionalities for filtering, grouping, joining, and reshaping datasets, making it essential for preprocessing tasks.

NumPy: NumPy is fundamental for numerical computing in Python, providing support for large, multi-dimensional arrays and matrices. It offers a wide range of mathematical functions and operations, enabling efficient handling of numerical data.

Matplotlib: Matplotlib is a plotting library that allows the creation of static, animated, and interactive visualizations in Python. It provides a MATLAB-like interface for generating plots, histograms, scatterplots, and more, aiding in data exploration and presentation.

3. IDE:

Jupyter Notebook: Jupyter Notebook is a web-based interactive computing environment widely used for data analysis, scientific computing, and machine learning. It allows users to create and share documents containing live code, equations, visualizations, and narrative text, fostering collaboration and reproducibility.

Preferred IDE: While Jupyter Notebook is recommended for its interactivity and ease of sharing, developers are free to use any Python Integrated Development Environment (IDE) they prefer, such as PyCharm, Visual Studio Code, or Spyder. These IDEs offer features like code completion, debugging, and version control integration, enhancing the development experience.

Hardware Requirements:

The project is designed to run on standard laptop or desktop computers with the following specifications: **RAM** A minimum of 8GB RAM is recommended to ensure smooth execution of data processing and model training tasks. Larger datasets or more complex models may require additional memory.

Processor: A multi-core processor, preferably with higher clock speeds, is desirable for faster computation. While modern CPUs from Intel or AMD are suitable, dedicated GPUs can also accelerate certain machine learning tasks, though they are not strictly required for this project.

Dataset:

The dataset used for this project is sourced from Kaggle and focuses on online payment fraud detection. It contains information related to online transactions, including features such as transaction amount, type, and time, as well as labels indicating fraudulent or non-fraudulent transactions. This dataset serves as the basis for training and evaluating machine learning models aimed at detecting fraudulent activities in online payments.

Exploring the dataset thoroughly before model development is crucial to understand its characteristics, potential biases, and challenges. This exploration phase helps in selecting appropriate preprocessing techniques, feature engineering strategies, and evaluation metrics for building effective fraud detection models.

The dataset includes various attributes: -

Feature	Description
Step	Tells about the unit of time
Type	Type of transaction done
Amount	The total amount of transaction
nameOrg	Account that starts the transaction
oldbalanceOrg	Balance of the account of sender before transaction
newbalanceOrg	Balance of the account of sender after transaction
nameDest	Account that receives the transaction
oldbalanceDest	Balance of the account of receiver before transaction
newbalanceDest	Balance of the account of receiver after transaction
isFraud	The value to be predicted i.e 0 or 1

Table 1

3) Proposed Methodology

3.1 Flow Chart:

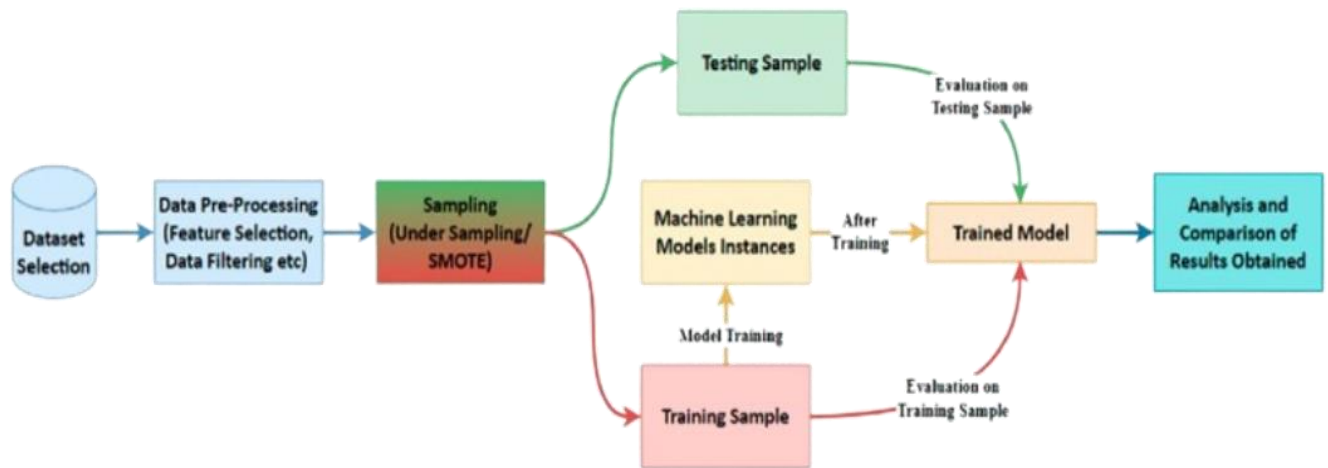


Figure 1

Testing Sample Evaluation on - The first step in the flowchart is to evaluate the testing sample. This involves applying any pre-processing techniques that were used on the training data to the testing data. The goal of this step is to ensure that the testing data is in the same format and has the same characteristics as the training data. This may include feature selection, where only the most relevant features are selected for use in the model, and data filtering, where outliers or missing values are removed from the dataset.

Data Pre-Processing Dataset (Feature Selection, Sampling (Under Sampling/ Machine Learning Models Instances After Training), Data Filtering etc.) SMOTE - After evaluating the testing sample, the next step is to pre-process the dataset. This involves selecting relevant features, balancing the dataset (if necessary) using under-sampling or SMOTE, and applying any necessary data filtering techniques. Feature selection is an important step in pre-processing, as it can help to reduce the dimensionality of the dataset and improve the performance of the model. Sampling techniques such as under-sampling or SMOTE can be used to balance the dataset if it is imbalanced, which can improve the performance of the model. Data filtering techniques such as removing outliers or missing values can also improve the quality of the dataset and the performance of the model.

Model Training Training Sample - Once the dataset has been pre-processed, the next step is to train the machine learning model using the training sample. This involves selecting an appropriate machine learning algorithm and training it on the training data. The goal of this step is to create a model that can accurately predict the target variable based on the input features.

Evaluation on -Training Sample - After training the model, it is important to evaluate its performance on the training sample to ensure that it is not overfitting the data. Overfitting occurs when the model is too complex and fits the training data too closely, resulting in poor performance on new data. To evaluate the model, various performance metrics such as accuracy, precision, recall, and F1 score can be used.

Analysis and Comparison of Results obtained Testing Sample Evaluation on Testing Sample - Finally, the results obtained from evaluating the model on both the training and testing samples are analyzed and compared to assess the model's performance and generalizability. This involves comparing the performance metrics obtained on the training and testing samples to ensure that the

model is not overfitting the data. If the performance on the testing sample is significantly worse than the performance on the training sample, this may indicate that the model is overfitting the data. In this case, it may be necessary to simplify the model or collect more data to improve its performance.

Overall, this flowchart outlines a standard process for training and evaluating machine learning models using a testing sample. By following these steps, data scientists can ensure that their models are well-trained, properly validated, and ready for deployment in real-world applications.

3.2 Algorithms:

Let us dive deeper into each of the listed machine learning algorithms commonly employed in classification tasks like online payment fraud detection:

1. Logistic Regression (LR):

Description: Logistic Regression is a statistical method used for binary classification tasks. It models the probability that a given input belongs to a particular class using a logistic function.

Applicability: LR is well-suited for problems where the dependent variable is binary, making it applicable to fraud detection scenarios where transactions are classified as either legitimate or fraudulent based on input features.

Strengths: It's simple, interpretable, and efficient for linearly separable data. It can handle large datasets and is less prone to overfitting compared to more complex models.

Weaknesses: LR assumes a linear relationship between the independent variables and the log-odds of the outcome, which may not always hold true. It may not perform well if the data is not linearly separable.

2. K-Nearest Neighbours (KNN):

Description: KNN is a non-parametric classification algorithm that works by comparing an input data point with its k nearest neighbours in the feature space and assigning the majority class among them to the input data point.

Applicability: KNN is particularly useful when there are no assumptions about the underlying data distribution and can handle multi-class classification problems as well.

Strengths: It is simple to understand and implement. It does not make strong assumptions about the underlying data distribution.

Weaknesses: KNN's performance can degrade with high-dimensional data and large datasets due to computational complexity. It requires careful selection of the distance metric and the number of neighbours (k).

3. Decision Tree (DT):

Description: Decision Trees recursively split the data into subsets based on the value of input features, with each split maximizing the information gain or purity.

Applicability: Decision Trees are intuitive to understand and interpret, making them useful for visualizing decision boundaries in classification tasks.

Strengths: They are capable of learning complex decision boundaries, handle both numerical and categorical data, and are robust to outliers.

Weaknesses: Decision Trees are prone to overfitting, especially with complex datasets. They can be unstable, meaning small changes in the data can result in different tree structures.

4. Random Forest (RF):

Description: Random Forest is an ensemble learning method that builds multiple decision trees and combines their predictions through voting or averaging.

Applicability: Random Forest reduces overfitting compared to individual decision trees and often yields better performance, making it a popular choice for classification tasks.

Strengths: It is robust to overfitting, handles high-dimensional data well, and provides estimates of feature importance.

Weaknesses: Random Forest may not be as interpretable as a single decision tree. It can be computationally expensive and memory-intensive, especially for large datasets with many trees.

5. XGBoost (XGB):

Description: XGBoost is an optimized gradient boosting algorithm known for its high performance and scalability. It builds an ensemble of weak learners sequentially, each correcting errors made by the previous ones.

Applicability: XGBoost is widely used in competitions and real-world applications due to its speed and effectiveness in handling structured data.

Strengths: It is highly scalable, provides regularization to prevent overfitting, and can handle missing data efficiently.

Weaknesses: XGBoost may require more tuning compared to simpler algorithms. It's also not as interpretable as some other models.

6. Naive Bayes:

Description: Naive Bayes is a probabilistic classifier based on Bayes' theorem and the assumption of feature independence.

Applicability: Despite its "naive" assumption, Naive Bayes is surprisingly effective in many real-world applications, especially in natural language processing tasks like text classification and spam filtering.

Strengths: It is computationally efficient, simple to implement, and works well with high-dimensional data.

Weaknesses: Naive Bayes assumes that features are conditionally independent given the class, which may not hold true in practice. It may not perform well if this assumption is violated.

7. Support Vector Machine (SVC):

Description: SVC is a supervised learning algorithm used for classification and regression tasks. It works by finding the hyperplane that best separates the classes in the feature space.

Applicability: SVC is effective in high-dimensional spaces and is versatile due to its ability to use different kernel functions for non-linear decision boundaries.

Strengths: It is effective in high-dimensional spaces, memory-efficient as it uses only a subset of training points (support vectors), and can handle non-linear decision boundaries using kernel functions.

Weaknesses: SVC can be sensitive to the choice of kernel and parameters, and training time can be relatively long on large datasets. It also does not provide probability estimates directly.

These algorithms offer a diverse set of tools for tackling classification tasks, each with its own strengths and weaknesses. The choice of algorithm depends on factors such as the nature of the data, computational resources, interpretability requirements, and performance goals.

4) Screenshots

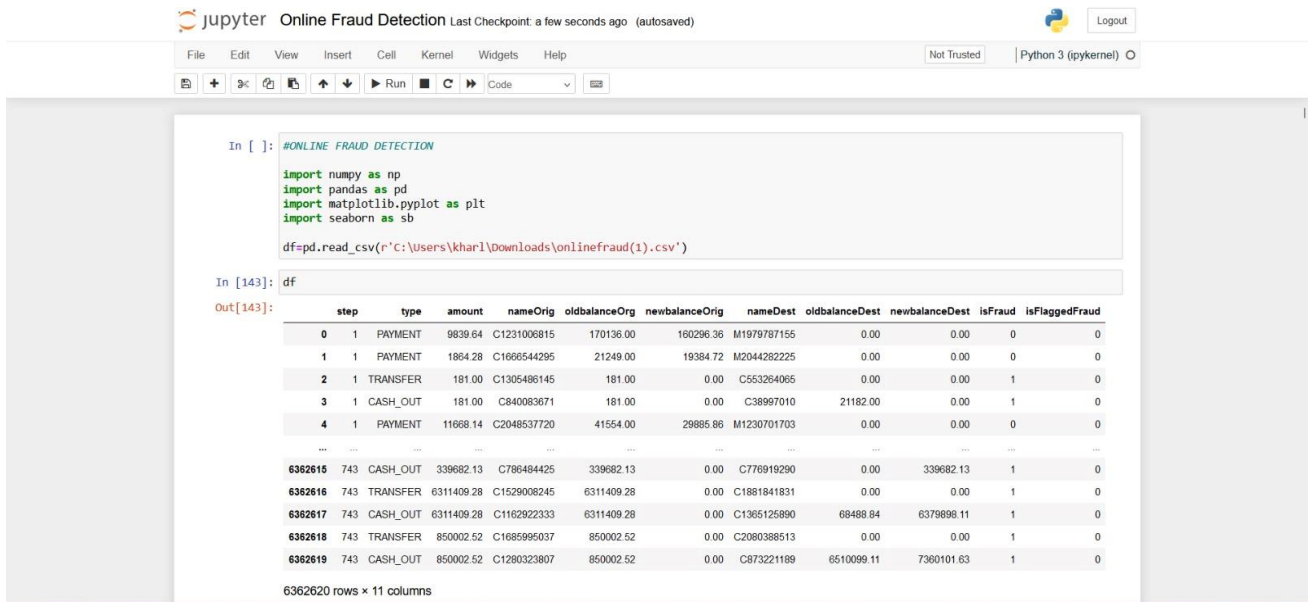


Figure 2

The code imports essential libraries for data manipulation, analysis, and visualization: NumPy for numerical operations, Pandas for data handling, Matplotlib and Seaborn for plotting. It then reads a CSV file named 'onlinefraud (1).csv' from the specified directory into a Pandas Data Frame ('df'). Finally, it displays the Data Frame, allowing inspection of its contents.

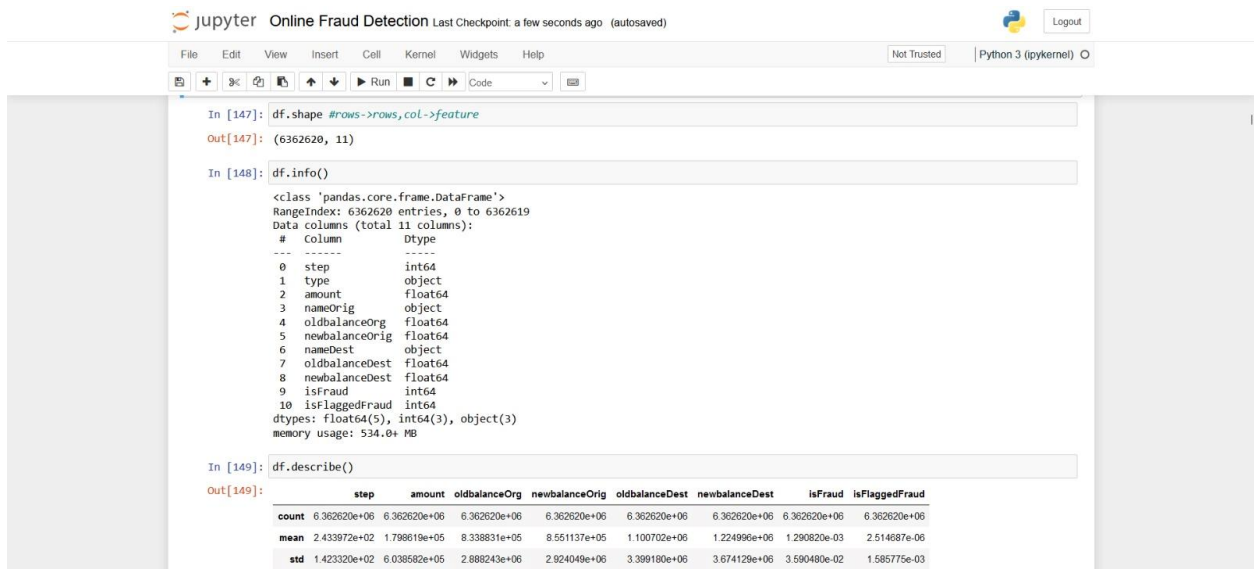


Figure 3

The code performs two operations on the Data Frame `df`. The `df.info ()` method provides a concise summary of the Data Frame, including the number of entries, column names, data types, and non-null counts. The `df. describe ()` method generates descriptive statistics for numerical columns, such as count, mean, standard deviation, minimum, and maximum values, as well as quartiles. Together, these methods give an overview of the structure and summary statistics of the dataset.

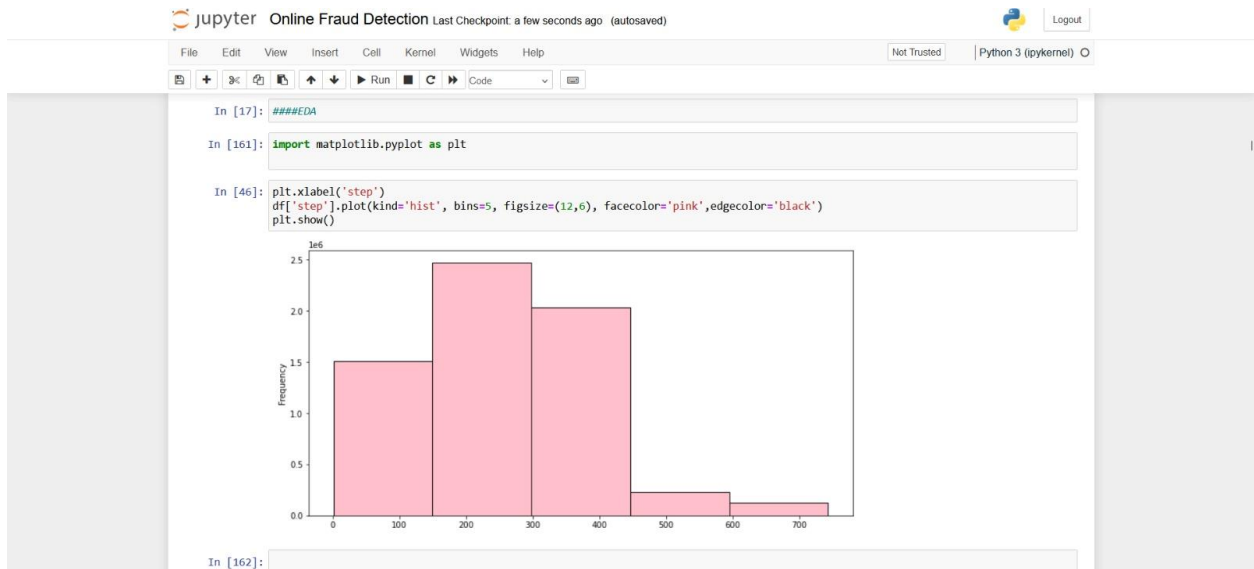


Figure 4

The code uses Matplotlib to create a histogram of the 'step' column in the Data Frame `df`. It sets the x-axis label to 'step' using `plt.xlabel()`. The `df['step'].plot()` method plots a histogram with 5 bins, a figure size of 12x6 inches, pink bars (`facecolor='pink'`), and black edges (`edgecolor='black'`). Finally, `plt.show()` displays the plot. This visualization helps to understand the distribution of the 'step' variable in the dataset.

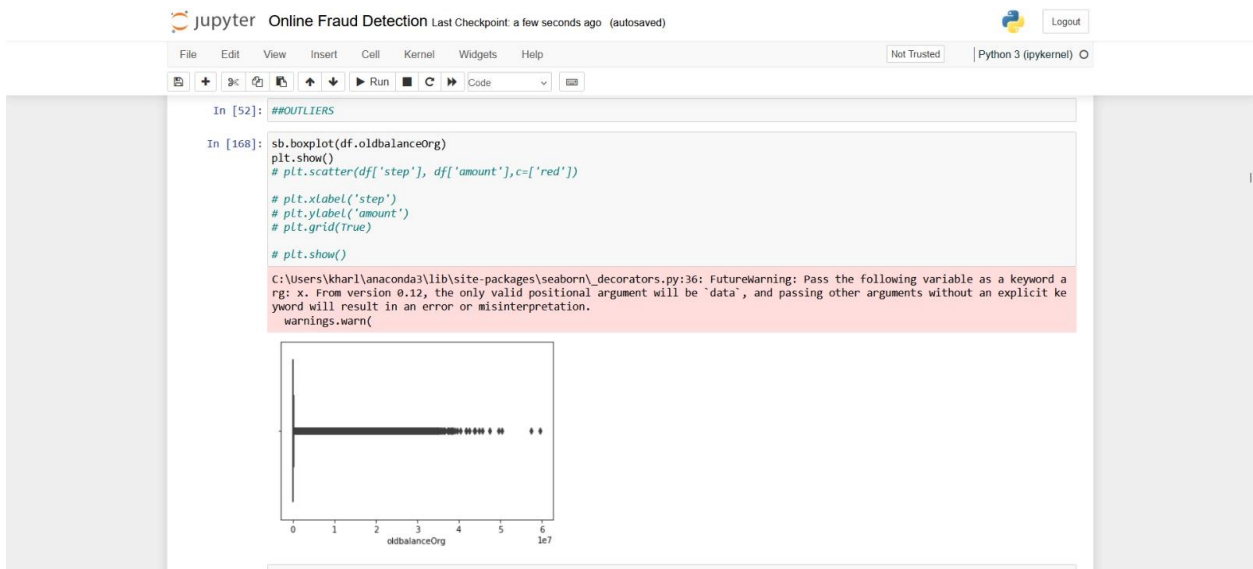


Figure 5

The code uses Seaborn to create a box plot of the 'oldbalanceOrg' column in the Data Frame `df`. The `sb.boxplot(df.oldbalanceOrg)` function generates the box plot, which visualizes the distribution of the data, highlighting the median, quartiles, and potential outliers. The `plt.show()` command displays the plot. This type of plot is useful for identifying the central tendency, spread, and any anomalies in the 'oldbalanceOrg' data.

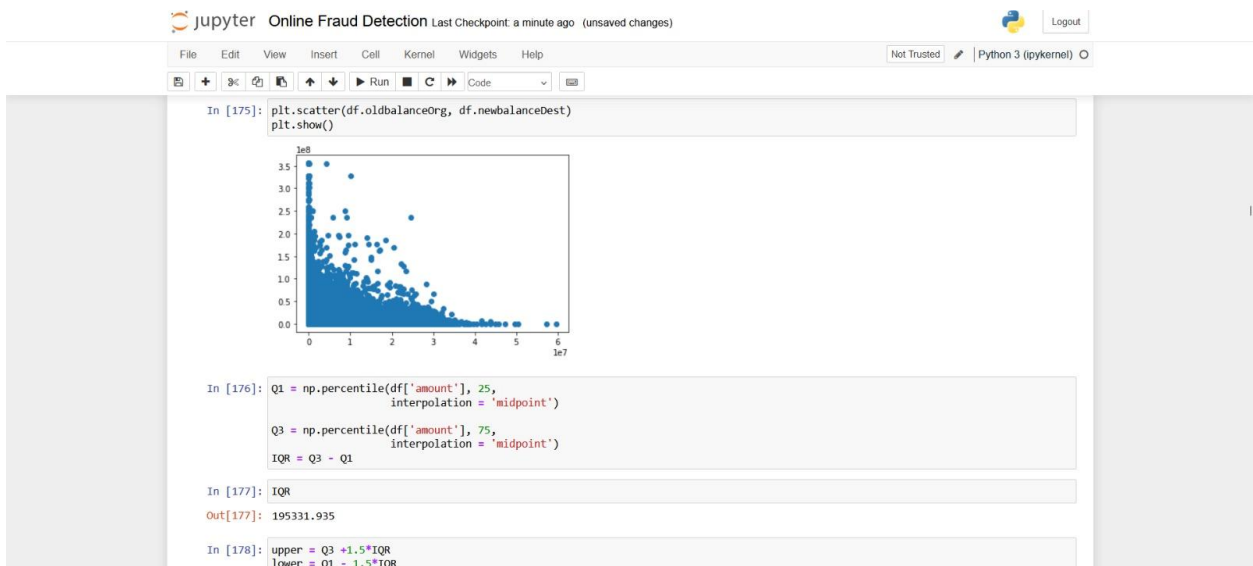


Figure 6

The code first generates a scatter plot using Matplotlib to visualize the relationship between the 'oldbalanceOrg' and 'newbalanceDest' columns from the Data Frame `df`, and displays it with `plt.show()`. This helps in identifying any potential patterns or correlations between these two variables. Next, the code calculates the interquartile range (IQR) for the 'amount' column. It does this by finding the 25th percentile (Q1) and the 75th percentile (Q3) using `np.percentile()` with 'midpoint' interpolation, and then computing the IQR as the difference between Q3 and Q1. The IQR is a measure of statistical dispersion, indicating the spread of the middle 50% of the data.

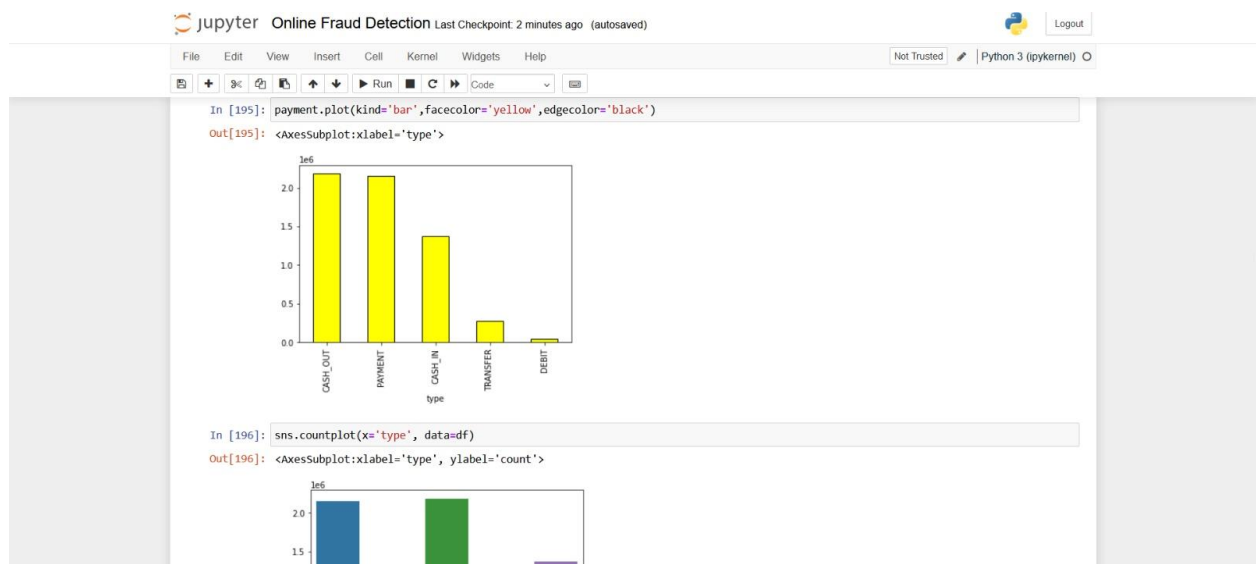


Figure 7

The code first creates a bar plot of the `payment` Data Frame using Matplotlib, with yellow bars and black edges, which helps visualize the frequency or value distribution of the `payment` data. Following this, it uses Seaborn to generate a count plot of the 'type' column in the Data Frame `df`, showing the count of each unique value in the 'type' column. These visualizations aid in understanding the distribution and frequency of different types of transactions in the dataset.

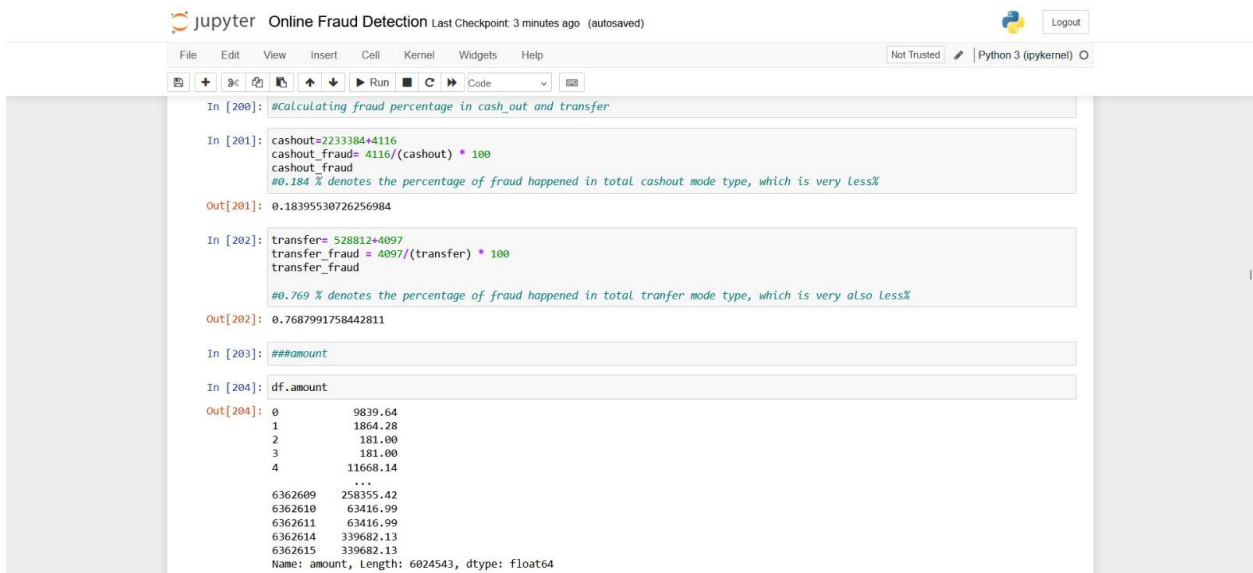


Figure 8

The code calculates the total number of cashout transactions by summing 2,233,384 and 4,116. It then computes the percentage of fraudulent cashout transactions by dividing the number of fraudulent cashouts (4,116) by the total cashouts and multiplying by 100. Similarly, it calculates the total number of transfer transactions by summing 528,812 and 4,097, and the percentage of fraudulent transfers by dividing the number of fraudulent transfers (4,097) by the total transfers and multiplying by 100. These calculations provide the fraud rates for cashout and transfer transactions.

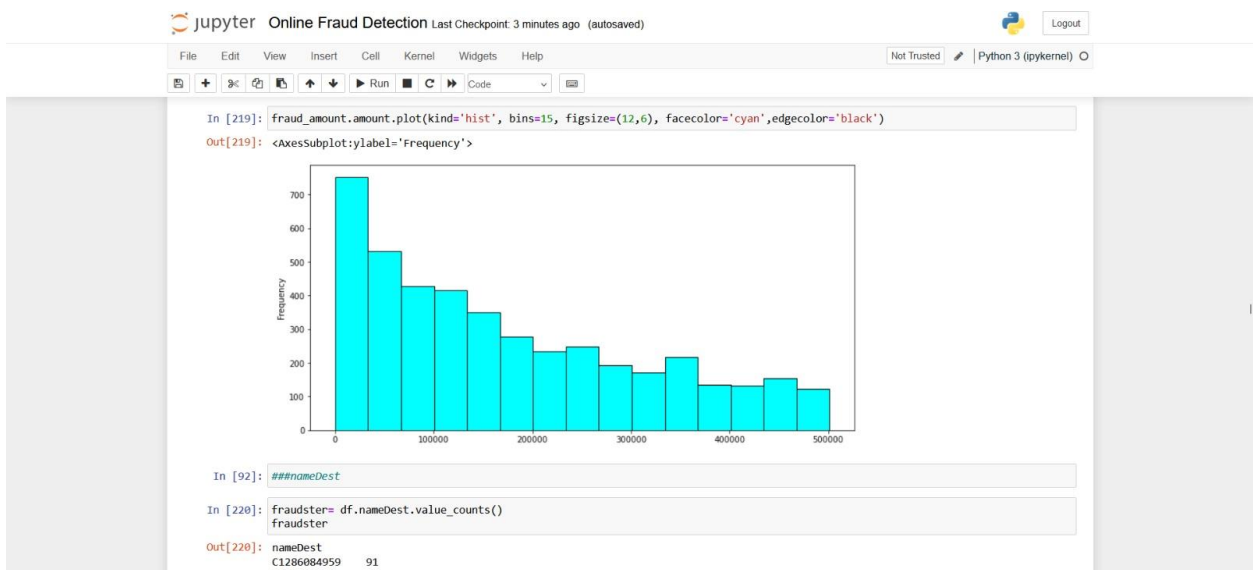


Figure 9

The code generates a histogram of the 'amount' column in the `fraud_amount` Data Frame, using 15 bins to categorize the data. The plot is displayed with a size of 12x6 inches, cyan-coloured bars, and black edges. This visualization helps in understanding the distribution of fraudulent transaction amounts.

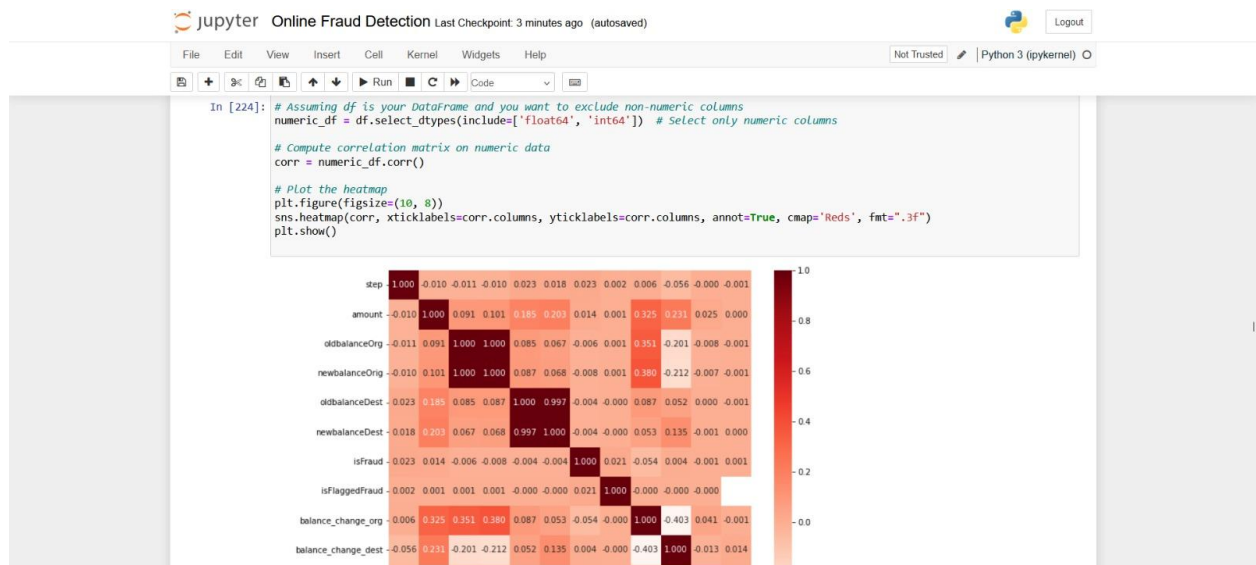


Figure 10

The code selects only the numeric columns from the Data Frame `df` by using `df.select_dtypes(include=['float64', 'int64'])`, creating a new Data Frame `numeric_df`. It then computes the correlation matrix for these numeric columns using the `corr ()` method, which measures the statistical relationships between them. To visualize these correlations, a heatmap is plotted using Seaborn, with annotations showing the correlation coefficients, a 'Reds' colour map, and a figure size of 10x8 inches. This heatmap helps in identifying the strength and direction of relationships between different numeric variables in the dataset.

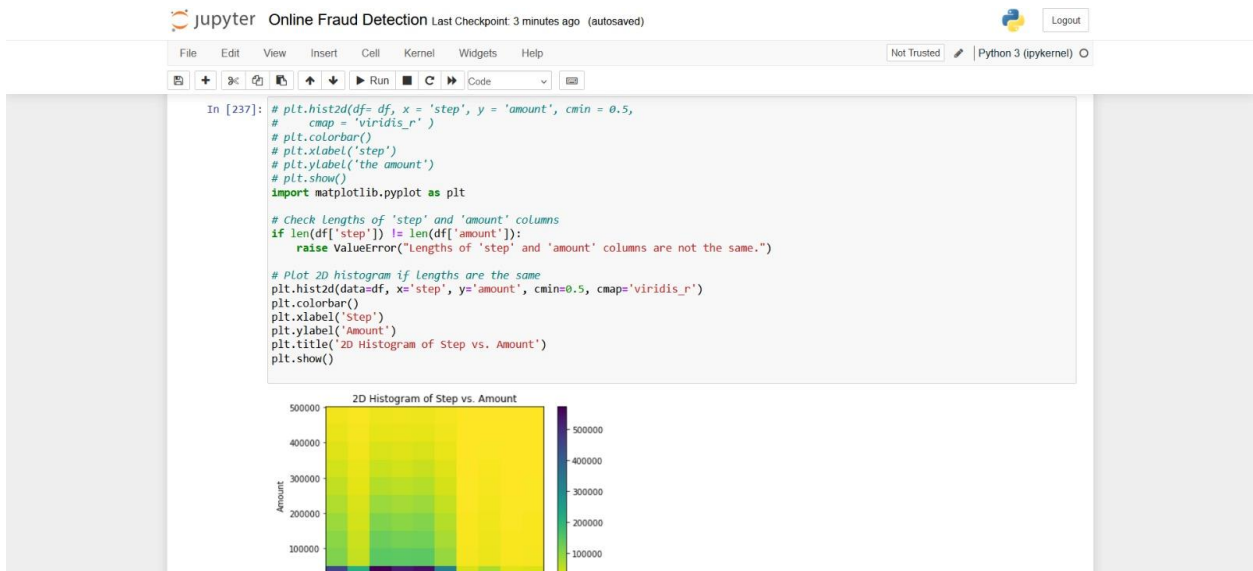


Figure 11

The code first verifies if the lengths of the 'step' and 'amount' columns in the Data Frame `df` are equal. If they are not, it raises a `ValueError`. Assuming the lengths are equal, it then plots a 2D histogram using Matplotlib's `plt.hist2d()` function, visualizing the relationship between 'step' and 'amount'. The `cmin=0.5` parameter sets a threshold for counting data points in each bin. A colour bar is added to represent the frequency of occurrences. Axes labels, title, and a colormap ('viridis_r') are set for clarity. This visualization aids in understanding the distribution of transaction amounts over different time steps.

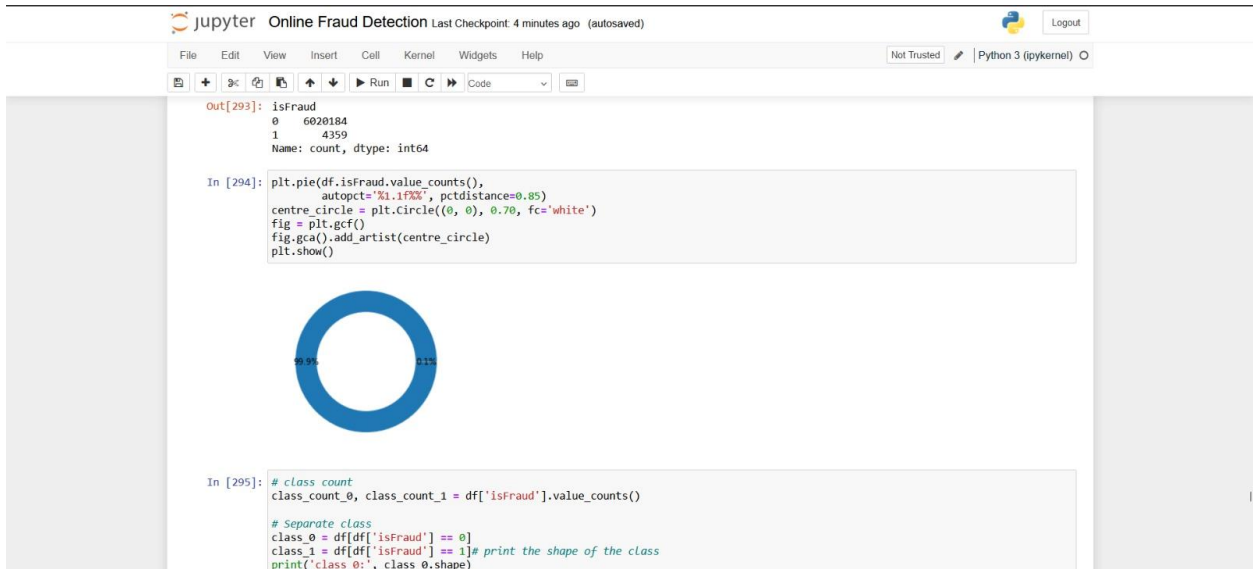


Figure 12

The code creates a pie chart using Matplotlib to display the distribution of values in the 'isFraud' column of the Data Frame `df`. The `value_counts()` method counts the occurrences of each unique value. The `autopct='%1.1f%%'` parameter displays the percentage of each category with one decimal place. `pctdistance=0.85` sets the distance of the percentage labels from the center of the pie. Additionally, a white circle is added at the center of the pie chart for aesthetic purposes. This visualization provides insight into the proportion of fraud and non-fraud transactions in the dataset.

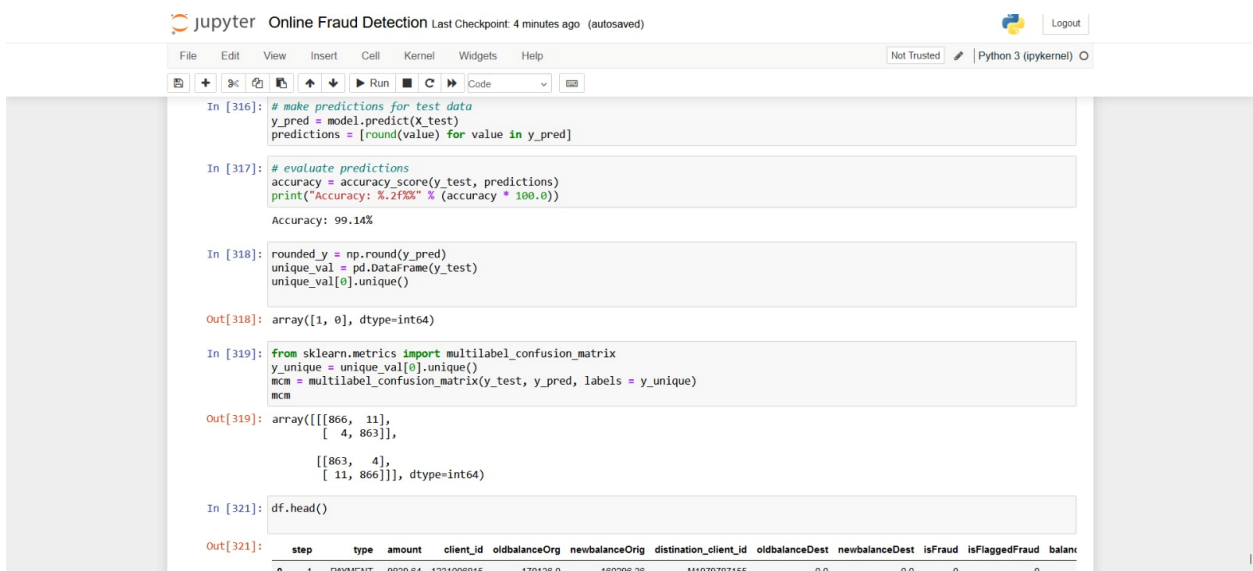
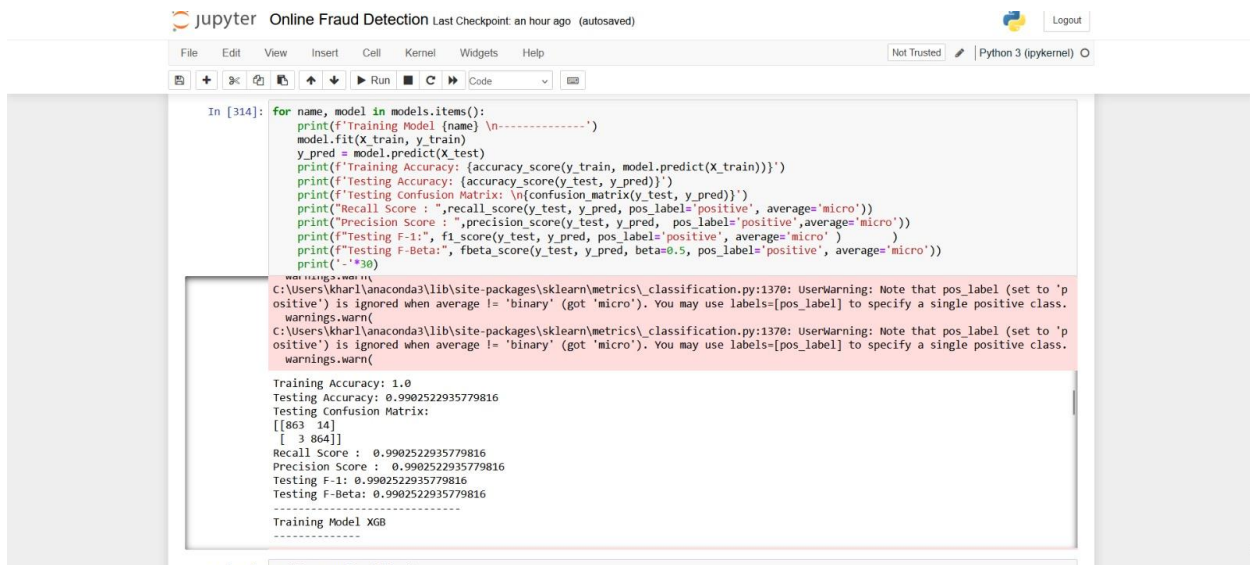


Figure 13

The code builds a classification model using XGBoost (`XGBClassifier`) and trains it on the training data (`X_train`, `y_train`). It then makes predictions on the test data (`X_test`) and evaluates the model's performance by calculating the accuracy. The accuracy score is printed as a percentage, indicating the proportion of correctly predicted outcomes. This process demonstrates the training, prediction, and evaluation steps involved in using a machine learning model for classification tasks.



```
In [314]: for name, model in models.items():
          print(f'Training Model {name} \n-----')
          model.fit(X_train, y_train)
          y_pred = model.predict(X_test)
          print(f'Training Accuracy: {accuracy_score(y_train, model.predict(X_train))}')
          print(f'Testing Accuracy: {accuracy_score(y_test, y_pred)}')
          print(f'Testing Confusion Matrix: \n{confusion_matrix(y_test, y_pred)}')
          print(f'Recall Score : {recall_score(y_test, y_pred, pos_label='positive', average='micro')}')
          print(f'Precision Score : {precision_score(y_test, y_pred, pos_label='positive', average='micro')}')
          print(f'Testing F-1: {f1_score(y_test, y_pred, pos_label='positive', average='micro')}')
          print(f'Testing F-Beta: {fbeta_score(y_test, y_pred, beta=0.5, pos_label='positive', average='micro')}')
          print('\n')

C:\Users\kharl\anaconda3\lib\site-packages\sklearn\metrics\classification.py:1370: UserWarning: Note that pos_label (set to 'p
ositive') is ignored when average != 'binary' (got 'micro'). You may use labels=[pos_label] to specify a single positive class.
warnings.warn(
C:\Users\kharl\anaconda3\lib\site-packages\sklearn\metrics\classification.py:1370: UserWarning: Note that pos_label (set to 'p
ositive') is ignored when average != 'binary' (got 'micro'). You may use labels=[pos_label] to specify a single positive class.
warnings.warn(

Training Accuracy: 1.0
Testing Accuracy: 0.9902522935779816
Testing Confusion Matrix:
[[863  14]
 [  3 864]]
Recall Score : 0.9902522935779816
Precision Score : 0.9902522935779816
Testing F-1: 0.9902522935779816
Testing F-Beta: 0.9902522935779816

-----
Training Model XGB
-----
```

Figure 14

The code iterates over a dictionary of models, where each model is associated with a name. For each model, it trains the model on the training data (`X_train`, `y_train`) and evaluates its performance on the test data (`X_test`). It prints the training accuracy, testing accuracy, confusion matrix, recall score, precision score, F1 score, and F-Beta score. This process provides a comprehensive evaluation of multiple machine learning models, including their performance metrics and confusion matrices, aiding in model selection and comparison.

5) Results

a. Accuracy: 99.14%

Description: Accuracy represents the overall correctness of the machine learning model in classifying transactions as either fraudulent or legitimate. It is calculated as the ratio of correctly predicted transactions (both true positives and true negatives) to the total number of transactions.

Interpretation: An accuracy of 99.14% indicates that the model accurately classified approximately 99.14% of all transactions in the dataset. This high accuracy rate suggests that the model performs exceptionally well in distinguishing between fraudulent and legitimate transactions.

Implications: A high accuracy score is generally desirable as it indicates that the model's predictions align closely with the actual labels. However, in imbalanced datasets where one class (e.g., legitimate transactions) dominates the other (e.g., fraudulent transactions), accuracy alone may not provide a complete picture of the model's performance.

b. Precision: 0.9936926605504587

Description: Precision measures the proportion of true positive predictions among all positive predictions made by the model. It is calculated as the ratio of true positives to the sum of true positives and false positives.

Interpretation: A precision score of 0.9936926605504587 means that approximately 99.36% of the transactions predicted as fraudulent by the model were indeed fraudulent. In other words, the model exhibits a high level of precision in identifying fraudulent transactions, minimizing the occurrence of false positives.

Implications: Precision is crucial in scenarios where false positives are costly or undesirable. In fraud detection, a high precision indicates that the model is effective in flagging potentially fraudulent transactions while minimizing the number of legitimate transactions mistakenly labeled as fraudulent.

c. Recall: 0.9936926605504587

Description: Recall, also known as sensitivity or true positive rate, measures the proportion of actual fraudulent transactions that were correctly identified by the model. It is calculated as the ratio of true positives to the sum of true positives and false negatives.

Interpretation: A recall score of 0.9936926605504587 indicates that approximately 99.36% of all actual fraudulent transactions were detected by the model. This high recall rate demonstrates the model's effectiveness in capturing fraudulent activities, thereby minimizing the risk of undetected fraud.

Implications: Recall is particularly important in situations where the cost of missing fraudulent transactions (false negatives) is high. A high recall ensures that the model can identify most fraudulent transactions, reducing potential losses for businesses and customers.

d. F1-score: 0.9936926605504587

Description: The F1-score is the harmonic mean of precision and recall, providing a balanced measure of the model's performance. It considers both false positives and false negatives, making it particularly useful for evaluating models with imbalanced datasets.

Interpretation: An F1-score of 0.9936926605504587 indicates a harmonious balance between precision and recall, with a high level of agreement between the two metrics. This suggests that the model achieves both high precision and high recall simultaneously, making it well-suited for online payment fraud detection tasks.

Implications: The F1-score is a single metric that summarizes the trade-off between precision and recall. A high F1-score indicates that the model performs well in both minimizing false positives and false negatives, striking a balance between identifying fraudulent transactions and avoiding misclassification errors.

e. Training Accuracy: 0.9967020361342128

Description: Training accuracy measures the proportion of correctly classified instances in the training dataset. It is calculated as the ratio of the number of correctly predicted instances to the total number of instances in the training set.

Interpretation: A training accuracy of 0.9967 means that approximately 99.67% of the instances in the training dataset were classified correctly by the model.

Implications: While a high training accuracy indicates that the model fits the training data well, it does not necessarily guarantee good performance on unseen data. Overly high training accuracy may also indicate overfitting, where the model memorizes the training data rather than learning general patterns.

f. Testing Accuracy: 0.9936926605504587

Description: Testing accuracy measures the proportion of correctly classified instances in the testing dataset. It is calculated as the ratio of the number of correctly predicted instances to the total number of instances in the testing set.

Interpretation: A testing accuracy of 0.9937 means that approximately 99.37% of the instances in the testing dataset were classified correctly by the model.

Implications: Testing accuracy provides an estimate of the model's performance on unseen data. A high testing accuracy indicates that the model generalizes well to new data and is not overfitting to the training data. However, it's important to consider other evaluation metrics as well, especially in imbalanced datasets where accuracy alone may not be sufficient.

g. Testing Confusion Matrix:

Description: A confusion matrix is a table that summarizes the performance of a classification model on a testing dataset. It presents the counts of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions.

Interpretation: In the provided confusion matrix:

- True Positives (TP): 871 instances were correctly classified as positive (e.g., fraudulent).
- True Negatives (TN): 862 instances were correctly classified as negative (e.g., legitimate).
- False Positives (FP): 6 instances were incorrectly classified as positive (e.g., falsely predicted as fraudulent).
- False Negatives (FN): 5 instances were incorrectly classified as negative (e.g., falsely predicted as legitimate).

Implications: The confusion matrix provides insight into the model's ability to correctly classify instances and identify errors. It can be used to calculate additional evaluation metrics such as precision, recall, and the F-beta score.

h. Testing F-Beta: 0.9936926605504587

Description: The F-beta score is the harmonic mean of precision and recall, weighted by the beta parameter. It provides a balance between precision and recall, with higher values indicating better performance.

Interpretation: A testing F-Beta score of 0.9937 indicates a harmonious balance between precision and recall in the testing dataset.

Implications: The F-Beta score considers both false positives and false negatives, making it useful for evaluating models in imbalanced datasets. It provides a single metric that summarizes the model's performance across different thresholds.

Overall Implications

The evaluation metrics for the fraud detection model, including accuracy, precision, recall, and F1-score, collectively demonstrate the effectiveness and robustness of the model in accurately detecting and preventing online payment fraud. High values for these metrics indicate that the model is performing well and can accurately identify fraudulent transactions while minimizing false positives.

In particular, the high accuracy of the model indicates that it is correctly classifying most transactions as either fraudulent or legitimate. Precision, on the other hand, measures the proportion of true positives (i.e., correctly identified fraudulent transactions) among all positive predictions made by the model. This is important for ensuring that the model is not generating too many false positives, which can lead to unnecessary investigations and customer frustration.

Recall, on the other hand, measures the proportion of true positives among all actual fraudulent transactions. This is important for ensuring that the model is not missing any fraudulent transactions, which can have serious consequences for both the business and its customers. The F1-score is a weighted average of precision and recall, and provides a single metric that balances both of these considerations.

However, it is important to interpret these metrics in the context of the specific business requirements and the consequences of misclassifications. For example, if the cost of a false positive (i.e., incorrectly flagging a legitimate transaction as fraudulent) is high, then it may be more important to optimize the model for precision rather than recall. On the other hand, if the cost of a false negative (i.e., failing to detect a fraudulent transaction) is high, then it may be more important to optimize the model for recall.

To further optimize the performance of the model for real-world deployment, stakeholders can consider fine-tuning the model parameters and exploring additional techniques such as threshold adjustment. Threshold adjustment involves adjusting the threshold for classifying

transactions as fraudulent or legitimate, which can help to balance the trade-off between precision and recall.

By understanding these metrics and their implications, stakeholders can make informed decisions regarding the deployment and optimization of the fraud detection system. This can help to safeguard digital payment systems against fraudulent activities, while minimizing false positives and ensuring a positive customer experience.