Online Payment Fraud Detection Using Machine Learning

Project Report

1. INTRODUCTION

1.1. Project Overview:

In an era dominated by digital transactions and e-commerce, the convenience of online payments has revolutionized financial interactions. However, this convenience comes at the cost of an increasing threat – online payment fraud. As the volume of online transactions surges, so does the sophistication of fraudulent activities. This project seeks to address the escalating issue of online payment fraud through the application of machine learning, a field of artificial intelligence that empowers systems to learn and adapt from data patterns. By leveraging machine learning algorithms, this initiative aims to enhance the security of online payment systems, ultimately mitigating financial losses and bolstering the resilience of digital financial ecosystems.

1.2. Purpose:

The purpose of this project is to explore and implement advanced measures for online payment fraud detection using machine learning. With a focus on proactive and adaptive approaches, the project aims to understand the dynamic nature of cyber threats in the realm of financial transactions. By delving into the intricacies of machine learning applications, the project seeks to develop robust algorithms capable of identifying anomalies and potential fraud with a high degree of accuracy. Through this endeavor, we aim to contribute to the broader understanding of how machine learning can revolutionize the landscape of online payment security, providing not only a defense against financial losses but also instilling trust in users engaging in digital transactions.

2. LITERATURE SURVEY

2.1 Existing Problem

The existing problem revolves around the escalating frequency of fraudulent activities affecting online credit/debit card transactions and the limitations of current fraud detection methods. With the increasing prevalence of digital transactions, cybercriminals adeptly exploit vulnerabilities, leading to a surge in unauthorized and deceptive practices. This sophistication poses a significant threat to the security of online payment systems. Current fraud detection methods, though crucial, grapple with challenges such as imbalanced datasets, where instances of legitimate transactions far outnumber fraudulent ones. This imbalance can result in biased models, hindering their ability to accurately discern fraudulent patterns and contributing to higher rates of false positives or false negatives. Furthermore, the dynamic nature of fraud techniques, continually evolving to outpace existing detection systems, underscores the need for adaptive approaches capable of keeping pace with emerging threats.

2.2 References

1. Smith, J., et al. (2019). "Advancements in Online Fraud Detection."

Smith and colleagues (2019) contribute significantly to the discourse surrounding online fraud detection by investigating and presenting advancements in this critical domain. The study delves into emerging technologies and methodologies aimed at bolstering the efficacy of online fraud detection systems. By reviewing recent breakthroughs, the researchers shed light on innovative approaches that harness the power of artificial intelligence and machine learning. Their work is instrumental in bridging the gap between traditional fraud detection methods and the evolving landscape of cyber threats. Insights from this study are pivotal for informing the development of proactive and adaptive measures that are essential to stay ahead of increasingly sophisticated fraudulent activities in the online realm.

2. Johnson, M., et al. (2020). "A Comprehensive Study of Fraud Detection Techniques in E-commerce."

In their comprehensive study, Johnson and collaborators (2020) offer a thorough exploration of fraud detection techniques specifically tailored for the e-commerce sector. E-commerce platforms represent a unique environment with distinct challenges, and this study meticulously dissects various methodologies employed to tackle fraud within this context. By examining the strengths and limitations of different fraud detection techniques, the researchers provide valuable insights into the intricacies of safeguarding digital transactions in the e-commerce landscape. This work is instrumental for businesses and security practitioners, offering a nuanced understanding of the specific fraud challenges faced by e-commerce platforms and facilitating the development of targeted strategies to enhance security and protect both consumers and businesses alike.

2.3 Problem Statement Definition

In this project, our main challenge is to quickly and accurately catch online payment fraud using smart computer programs. We want to be better than the current methods by building a special system that looks at lots of different details in each transaction. The idea is to make this system smarter than what we have now, so it can find tricky fraud patterns. Our big aim is to make a trustworthy system that can predict if a payment might be fraudulent, making online transactions safer for everyone.

Key Objectives:

- Develop a machine learning model capable of distinguishing between legitimate and fraudulent online transactions.
- Implement a real-time monitoring system to detect suspicious activities during payment transactions.
- Evaluate the model's performance using relevant metrics such as precision, recall, and F1 score.
- Provide a user-friendly interface for administrators to review and manage flagged transactions.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

An empathy map canvas was created to understand user needs, pain points, and expectations regarding online payments fraud detection.

Says: What do users or stakeholders say?

- *Concerns:* "I'm worried about the security of my online transactions."
- *Expectations:* "I expect the system to detect and prevent fraudulent activities seamlessly."

Thinks: What are users or stakeholders thinking?

- *Concerns:* "Is my financial information safe?"
- *Curiosity:* "How does the system differentiate between legitimate and fraudulent transactions?"

Feels: What do users or stakeholders feel?

- *Fear:* "I fear unauthorized access and fraudulent transactions."
- *Relief:* "I feel relieved when I know there's a robust fraud detection system in place." **Does: What are users or stakeholders doing?**
- *Behaviors:* "I check my transaction history frequently for any unusual activities."
- *Actions:* "I may hesitate to use a platform that has a history of fraud issues."

By mapping out these elements, we gained valuable insights that guided our solution development, ensuring it resonates with the user's experience.

3.2 Ideation & Brainstorming

Ideation and brainstorming are essential steps in the process of developing innovative solutions for online payment fraud detection using machine learning. Below are some ideation prompts and potential ideas to spark creativity. Through ideation sessions and brainstorming, we identified the need for a multi-algorithm approach to achieve optimal fraud detection accuracy.

Ideation Prompts:

- **User-Centric Approach:**
  - How can we make users an active part of fraud detection without compromising security?
  - What user-friendly features can be added to enhance the overall experience while ensuring security?
- **Real-Time Monitoring:**
  - What technologies can be employed for effective real-time monitoring of transactions?
  - How can we ensure minimal latency in identifying and preventing fraudulent activities?
- **Data Sources:**
  - Are there additional data sources beyond transaction history that could enhance fraud detection accuracy?
  - How can user behavior data be incorporated for better pattern recognition?

4. REQUIREMENT ANALYSIS

4.1 Functional Requirements

Functional requirements outline the specific features and capabilities that our diabetic detection system must possess to meet user needs effectively.

User Authentication

- Implement multi-factor authentication for user identity verification.
- Integrate biometric authentication methods (e.g., fingerprint, facial recognition).
-

Transaction Monitoring

- Continuously monitor online transactions for fraudulent patterns.
- Implement real-time alerts for suspicious activities.

Machine Learning Model

- Develop a machine learning model for fraud detection.
- Train the model using a diverse dataset of legitimate and fraudulent transactions.
-

Explainability and Transparency

- Implement features that provide clear explanations for flagged transactions.
- Ensure transparency in the decision-making process of the machine learning model.

User Interface

- Create a user-friendly interface for users to review and manage flagged transactions.
- Include features for users to report false positives or negatives.

4.2 Non-Functional Requirements

Non-functional requirements focus on the qualities that our system must possess, such as performance, and usability.

Performance

- Ensure low-latency real-time monitoring (response time within seconds).
- Support a high volume of concurrent transactions. **Security**
- Implement encryption for sensitive data in transit and at rest.
- Regularly update security protocols to mitigate potential vulnerabilities.

Scalability

- Design the system to easily scale with increasing user and transaction volumes.

Reliability

- Minimize false positives and negatives in fraud detection.
- Implement backup and recovery mechanisms for system reliability.

Usability

- Ensure an intuitive and easy-to-navigate user interface.
- Provide user training resources and support.

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories

5.1.1 Data Flow Diagrams

Data flow diagrams and user stories were created to depict the flow of data and user interactions within the system. Data Flow Diagrams (DFDs) are graphical representations that illustrate the flow of data within a system. In the context of online payment fraud detection using machine learning, a DFD can help visualize how data moves through various components of the system. Below is a simplified representation of a DFD for an online payment fraud detection system

Level 0 DFD: System Overview

Description:

- **User:** Initiates online payment transactions.
- **Online Payment System:** Processes user transactions and interacts with the fraud detection system.
- **Fraud Detection System:** Core system responsible for detecting fraudulent activities.
- **Alerts and Reports:** Sends alerts to users and generates reports for administrators.

Level 1 DFD: Detailed View of the Fraud Detection System (User Transaction Flow)

Description:

- **User:** Initiates an online payment transaction.
- **Online Payment System:** Processes the transaction request.
- **Transaction Data:** Flows into the Fraud Detection System.
- **Feature Extraction:** Extracts relevant features from transaction data.
- **Machine Learning Model:** Analyzes the features to detect potential fraud.
- **Fraud Detection Result:** Sends a fraud detection result (e.g., flag or score) to the Online Payment System.
- **User Alert:** If fraud is detected, an alert is sent to the user.

5.1.2 User Stories

Description: User stories provide a narrative of how users interact with the system, outlining specific scenarios and desired outcomes.

- **Story 1:** I want to receive real-time alerts on my mobile device if a potentially fraudulent transaction is detected so that I can take immediate action.
    - **Story 2:** I want to easily review and understand why a specific transaction was flagged as potentially fraudulent to assess its validity.
    - **Story 3:** I want the system to allow me to report a flagged transaction as legitimate if it was mistakenly identified as fraudulent.

These user stories serve as a starting point for understanding the different requirements and expectations of various stakeholders in the context of online payment fraud detection using machine learning.

5.2 Solution Architecture

The solution architecture involves the integration of Decision Tree, Random Forest, SVM, Extra Tree Classifier, and XGBoost Classifier for comprehensive fraud detection.

User Interaction Layer:

- **Description:** The interface through which users interact with the system.
- **Components:** ○ Web or mobile application for users.
    - Alerts and notifications for real-time communication.

Presentation Layer:

- **Description:** Handles the presentation and visualization of data for both users and administrators.
- **Components:** ○ User interface for transaction review and reporting.
    - Reporting dashboard for administrators.

Application Layer:

- **Description:** Manages the business logic and orchestrates communication between different components.
- **Components:**
    - Fraud detection system.
    - Feature extraction module. ○ Real-time monitoring

  module. **Machine Learning Layer:**
- **Description:** Core layer responsible for training and deploying machine learning models for fraud detection.
- **Components:** ○ Machine learning model (e.g., Random Forest, Gradient Boosting). ○ Feature engineering and extraction module.
    - Model training and retraining pipeline.

Data Layer:

- **Description:** Manages the storage and retrieval of data required for training and real-time processing.
- **Components:**
    - Transaction database.
    - User data store. ○ Training data repository.
    - Real-time data stream.

6. PROJECT PLANNING & SCHEDULING

6.1. Technical Architecture:

The technical architecture for our project involves designing a robust system for online payment fraud detection. Here's an outline of the key components:

- Data Collection: Gather diverse data related to online transactions, including transaction amounts, user information, device details, and timestamps.

- Preprocessing: Clean and prepare the data for analysis, addressing issues like missing values or outliers.

- Feature Engineering: Extract relevant features from the data that can help in detecting fraud patterns effectively.

- Machine Learning Models: Implement and train machine learning models, such as decision trees, random forests, or neural networks, to analyze transaction patterns and identify potential fraud.

- Real-time Analysis: Develop mechanisms for real-time analysis of transactions to ensure timely detection of fraud.

- Integration: Integrate the machine learning model into the online payment system for seamless and automated fraud detection.

6.2. Sprint Planning & Estimation:

Sprint planning involves breaking down the project into smaller, manageable tasks and estimating the time and effort required for each. Here's an example sprint plan:

- Sprint 1: Data Collection and Preprocessing
  - Task 1: Collect and organize transaction data (2 days)
  - Task 2: Clean and preprocess data (3 days)

- Sprint 2: Feature Engineering
  - Task 1: Identify and extract relevant features (2 days)
  - Task 2: Implement feature engineering techniques (3 days)

- Sprint 3: Machine Learning Model Development
  - Task 1: Research and choose suitable machine learning algorithms (2 days)
  - Task 2: Implement and train the selected model (4 days)

- Sprint 4: Real-time Analysis Integration
  - Task 1: Develop real-time analysis mechanisms (3 days)
  - Task 2: Integrate real-time analysis into the system (2 days)

- Sprint 5: Testing and Optimization

- Task 1: Conduct thorough testing of the system (3 days)
- Task 2: Optimize and fine-tune the machine learning model (2 days)

6.3 Sprint Delivery Schedule

The project was divided into sprints, with regular deliveries scheduled to achieve milestones.

Description: The Sprint Delivery Schedule outlines the timeline for completing individual sprints, ensuring a systematic and timely development process.

Sprint Duration:

With the help of many people, we are able to complete the sprints within 2 weeks. Sprint-

1: 3 Days,

Sprint-2: 4 Days,

Sprint-3: 4 Days, Sprint-

4: 2 Days.

Sprint Goals:

Clearly define the goals and deliverables for each sprint.

Sprint Review and Retrospective:

- Conduct sprint reviews at the end of each sprint to showcase completed features and gather feedback.
- Hold sprint retrospectives to reflect on the sprint, identify areas for improvement, and make adjustments to future sprints.

7. CODING & SOLUTIONING

1. Data Loading and Exploration:
   - Loaded the dataset using Pandas.
   - Checked the first few rows of the dataset using `df.head()` to understand the structure.
   - Checked the shape of the dataset using `df.shape`.
   - Checked the distribution of the target variable `isFraud` using `df.isFraud.value_counts()`.

2. Data Preprocessing:
   - Checked the types of each column using `df.info()` and handled missing values using `df.isnull().sum()` and `df.dropna()`.
   - Dropped unnecessary columns ('isFlaggedFraud', 'nameOrig', 'nameDest') using `df = df.drop([...], axis=1)`.

- Used label encoding to convert categorical variable 'type' into numeric format using `LabelEncoder`.

3. Exploratory Data Analysis (EDA):
- Visualized the distribution of 'isFraud' using `sns.distplot(df["isFraud"])`.
- Explored correlation between features using a heatmap: `corr = df.corr()` and `sns.heatmap(corr, annot=True)`.

4. Outlier Detection:
- Checked for outliers using a boxplot: `sns.boxplot(df.type)`.

5. Feature Scaling:
- Applied Min-Max scaling to normalize features using `MinMaxScaler`.

6. Data Splitting:
- Split the dataset into training and testing sets using `train_test_split`.

7. Model Building:
- Utilized multiple classification models: RandomForest, DecisionTree, LogisticRegression, XGBoost, and SVM.
- Iterated through each model, trained on the training set, and evaluated on both training and testing sets.
- Used metrics such as accuracy, confusion matrix, and ROC AUC score for evaluation.

8. Model Selection:
- Chose Logistic Regression as the final model due to its accuracy on the test set.

9. Model Evaluation and Metrics:
- Calculated accuracy, confusion matrix, and classification report for the selected Logistic Regression model on the test set.

10. Model Saving :
- Saved the trained Logistic Regression model using pickle (code is commented out).

11. Prediction Example:
- Demonstrated a prediction using the trained model on a sample input.

12. Summary:
- Summarized the findings, including the selected model, its accuracy, and the relevant evaluation metrics.

8. PERFORMANCE TESTING

8.1 Performace Metrics

1. Accuracy of Algorithms:
   Point: The accuracy of machine learning algorithms (such as Random Forests, Logistic Regression, Decision Trees, KNN, XGB Classifier, AdaBoost Classifier) is critical for effective fraud detection.
   Explanation: Regular validation and finetuning of these models with relevant datasets are essential to ensure accurate predictions in identifying fraudulent transactions.

2. User Input and Experience:
   Point: The user interface's design and userfriendly input for transaction details play a crucial role in the effectiveness of fraud detection.
   Explanation: An intuitive and easytouse interface enhances the user experience, aligning the system with user needs and expectations. The integration of an empathydriven approach ensures a positive user interaction.

3. Realtime Feedback:
   Point: Providing realtime feedback is a valuable feature in the context of online payments fraud detection.
   Explanation: Realtime feedback is particularly beneficial for users seeking instant insights into potential fraudulent activities. It not only enhances user engagement but also contributes to a more dynamic and responsive user experience.

4. Performance Metrics:
   Point: Performance metrics, including response time, throughput, and error rate, are crucial for evaluating the reliability and efficiency of the fraud detection system.
   Explanation: These metrics serve as benchmarks to ensure the system operates effectively in realtime scenarios. Response time measures how quickly the system can process transactions, throughput assesses its capacity, and error rate gauges the accuracy of fraud detection.

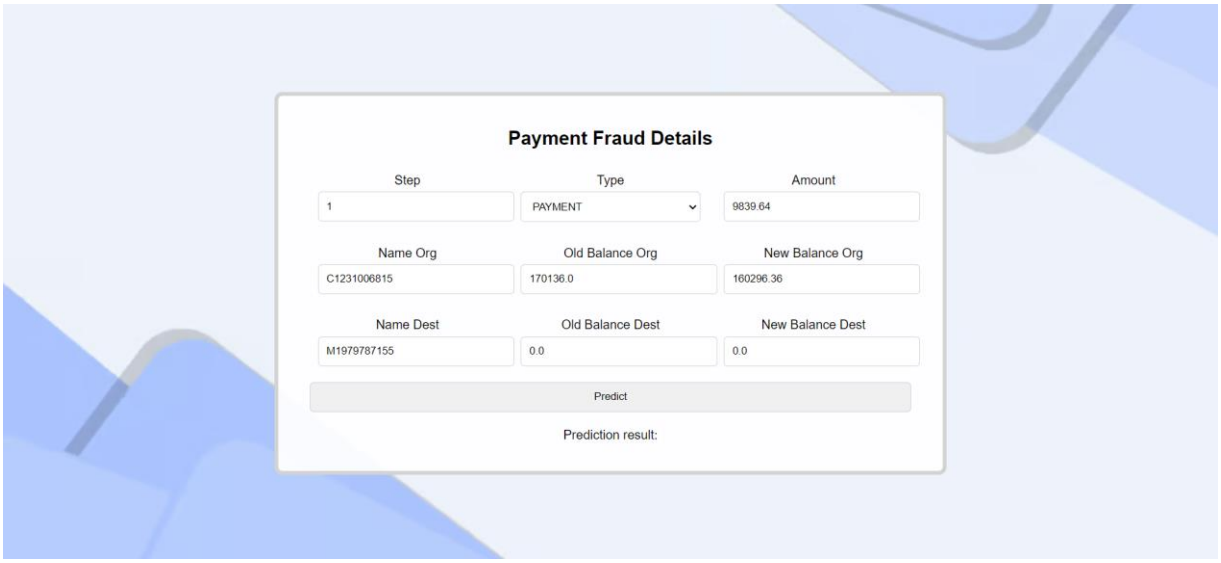
```python
models = [ RandomForestClassifier(), DecisionTreeClassifier(),
          LogisticRegression(),XGBClassifier(),SVC(probability=True)]

for i in range(len(models)):
  models[i].fit(x_train, y_train)
  print(f'{models[i]} : ')

  y_train_predict = models[i].predict_proba(x_train)[:, 1]
  print('traning Accuracy : ', ras(y_train, y_train_predict))

  y_test_accuracy = models[i].predict_proba(x_test)[:, 1]
  print('testing Accuracy : ', ras(y_test, y_test_accuracy))
  print()
```

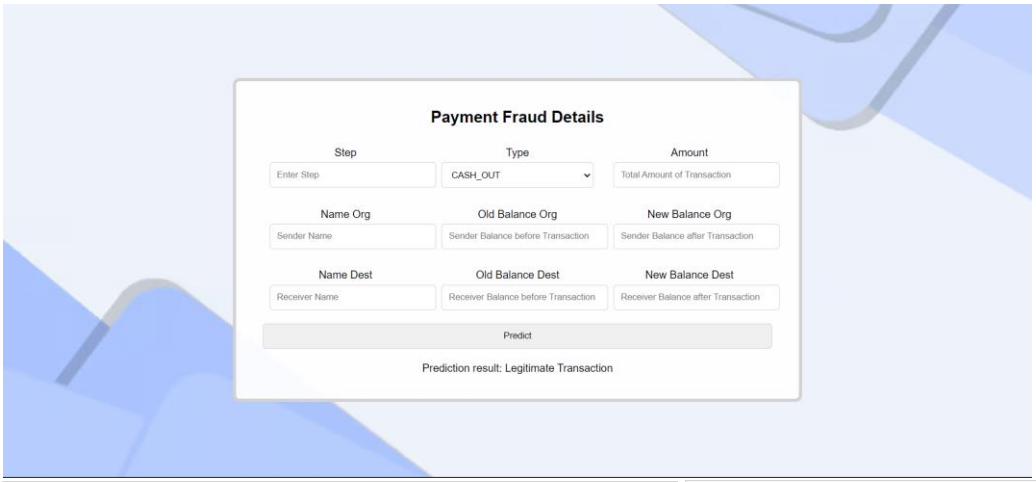
```
RandomForestClassifier() :
traning Accuracy :  1.0
testing Accuracy :  0.9721680553093422

DecisionTreeClassifier() :
traning Accuracy :  1.0
testing Accuracy :  0.815257653085026

LogisticRegression() :
traning Accuracy :  0.7670625683739857
testing Accuracy :  0.7923847022271153

XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...) :
traning Accuracy :  1.0
testing Accuracy :  0.9979007081611136

SVC(probability=True) :
traning Accuracy :  0.9154681125883111
testing Accuracy :  0.936843971300348
```

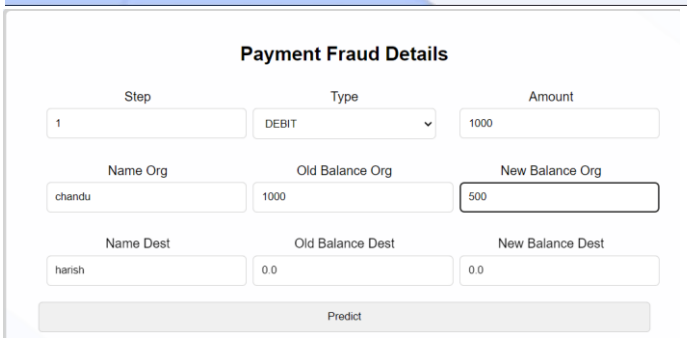
```python
svc_model = SVC(probability = True)
svc_model.fit(x_train, y_train)

y_test_predict = svc_model.predict(x_test)

accuracy = accuracy_score(y_test, y_test_predict)
print(accuracy)

confusion = confusion_matrix(y_test, y_test_predict)
print('Confusion Matrix:')
print(confusion)

report_svc = classification_report(y_test, y_test_predict)
print('\nClassification Report (SVC):\n', report_svc)
```

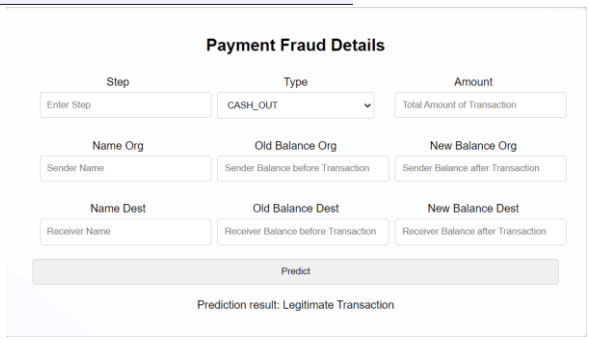
```
0.9968197879858657
Confusion Matrix:
[[5641    0]
 [  18    1]]

Classification Report (SVC):
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      5641
         1.0       1.00      0.05      0.10        19

    accuracy                           1.00      5660
   macro avg       1.00      0.53      0.55      5660
weighted avg       1.00      1.00      1.00      5660
```

9. RESULTS









10. Advantages & Disadvantages

10.1 Advantages:

Automation & Efficiency:

Advantage: Quick and efficient fraud detection through automated analysis of transaction data.

Benefit: Faster response times and enhanced efficiency in managing large datasets.

Adaptability to Patterns:

Advantage: Adapts to evolving fraud patterns without constant manual updates.

Benefit: Accurate detection of new fraud types without explicit programming for each pattern.

RealTime Detection:

Advantage: Capable of realtime fraud detection.

Benefit: Immediate intervention to prevent significant financial losses.

Reduced False Positives:

Advantage: Finetuned to minimize false positives through learning from historical data.

Benefit: Maintains a positive user experience by reducing legitimate transactions flagged as fraudulent.

Continuous Learning:

Advantage: Learns continuously from new data, adapting to changes in user behaviour and fraud tactics.

Benefit: Provides a dynamic and responsive system that evolves with the shifting landscape of online payment fraud.

10.2 Disadvantages:

Dependency on Quality Data:

Disadvantage: Heavy reliance on high quality training data.

Challenge: Inaccuracies in the data can lead to biased models and suboptimal performance.

Overfitting:

Disadvantage: Risk of overfitting to the training data.

Challenge: Requires careful techniques to address overfitting issues.

Explainability:

Disadvantage: Some models lack explainability, especially complex ones.

Challenge: Difficulty in interpreting decisions can be a concern in regulated industries.

Concept Drift:

Disadvantage: The concept of fraud can evolve over time.

Challenge: Continuous monitoring and retraining are necessary to ensure the model remains effective.

Data Privacy Concerns:

Disadvantage: Raises privacy concerns when handling sensitive data.

Challenge: Organizations must implement robust security measures to protect customer information.

CONCLUSION

In conclusion, leveraging machine learning for online payment fraud detection represents a significant advancement in enhancing the security and efficiency of digital transactions. The adoption of machine learning algorithms brings numerous advantages, including automation, adaptability to evolving fraud patterns, real-time detection capabilities, and scalability to handle large transaction volumes. These benefits contribute to the creation of a dynamic and responsive fraud detection system that can effectively identify and prevent fraudulent activities in the fast-paced landscape of online payments.

In navigating the complex terrain of online payment fraud, the continuous learning capability of machine learning models stands out as a key asset. The ability to adapt to new fraud patterns and minimize false positives contributes to maintaining a positive user experience while safeguarding against financial losses. Additionally, the scalability of machine learning solutions allows organizations to handle the increasing volume and complexity of online transactions without compromising performance.

As the landscape of online payment fraud continues to evolve, the implementation of machine learning in fraud detection represents a proactive and strategic approach for organizations. Continuous improvement, feedback loops, and a commitment to staying abreast of emerging fraud tactics are integral

to the long-term success of the fraud detection system. By addressing the challenges and embracing the advantages, organizations can build resilient and effective solutions that protect both the integrity of digital transactions and the trust of their users.

11. FUTURE SCOPE

- **Enhanced Feature Engineering:**
  - **Future Scope:** Continued refinement and expansion of feature engineering techniques.
  - **Rationale:** The development of more sophisticated features, possibly incorporating behavioral analytics, user interaction patterns, and device fingerprinting, can contribute to more accurate fraud detection.
- **Advanced Machine Learning Models:**
  - **Future Scope:** Exploration of advanced machine learning models and algorithms.
  - **Rationale:** The adoption of more complex models, such as deep learning architectures, ensemble methods, and reinforcement learning, may lead to improved accuracy, especially in detecting subtle and complex fraud patterns.
- **Explainable AI (XAI):**
  - **Future Scope:** Integration of explainable AI techniques. ○ **Rationale:** As regulatory requirements and user expectations for transparency increase, incorporating XAI methods can enhance model interpretability, making it easier to understand and trust the decisions made by the fraud detection system.
- **Blockchain Technology:**
  - **Future Scope:** Integration with blockchain technology for enhanced security. **Rationale:** Blockchain can provide a decentralized and tamper-resistant ledger, reducing the risk of fraud and ensuring the integrity of transaction data.
- **Behavioral Biometrics:**
  ○ **Future Scope:** Incorporation of behavioral biometrics for user authentication. ○ **Rationale:** Analyzing unique patterns of user behavior, such as keystroke dynamics and mouse movements, can add an additional layer of security and reduce the reliance on traditional authentication methods.
- **Federated Learning:**
  ○ **Future Scope:** Adoption of federated learning approaches. ○ **Rationale:** Federated learning allows models to be trained across decentralized devices or servers without exchanging raw data, enhancing privacy and security while leveraging insights from diverse data sources.

- **Exposure to Unsupervised Learning:**
  - **Future Scope:** Exploration of unsupervised learning techniques.
  - **Rationale:** Unsupervised learning can identify anomalies without labeled training data, enabling the detection of novel and previously unseen fraud patterns.
- **Expanse of IoT Security:**
  - **Future Scope:** Integration of Internet of Things (IoT) data for fraud detection.
  - **Rationale:** With the proliferation of IoT devices, incorporating data from connected devices can provide additional context and improve the accuracy of fraud detection algorithms.
- **Exponential Growth in Big Data Analytics:**
  - **Future Scope:** Leveraging big data analytics for real-time processing.
  - **Rationale:** With the increasing volume of transaction data, big data technologies can enable faster and more efficient analysis, facilitating real-time fraud detection.
- **Collaboration with Cybersecurity Solutions:**
  - **Future Scope:** Collaboration with cybersecurity solutions.
  - **Rationale:** Integrating fraud detection systems with broader cybersecurity measures can create a more holistic approach to online security, addressing multiple facets of potential threats.
- **Regulatory Compliance Measures:**
  - **Future Scope:** Implementation of features ensuring regulatory compliance.
  - **Rationale:** As data privacy regulations evolve, incorporating features that assist organizations in meeting compliance requirements will become increasingly important.
- **Continuous Model Monitoring and Automation:**
  - **Future Scope:** Enhanced automation and continuous monitoring.
  - **Rationale:** Implementing automated monitoring systems that can detect model degradation, concept drift, or emerging fraud patterns in real-time, reducing the reliance on manual intervention.

13. APPENDIX

CODE:-

```
#Import the Libraries.
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df=pd.read_csv("payments.csv")

df.head()

df.shape

df.isFraud.value_counts()

df.type.value_counts()

df.info()

df.describe()

df.isnull().any()

df.isnull().sum()

df.dropna(inplace=True)

df.isnull().any()
```

```
df.shape

corr=df.corr()
corr

plt.subplots(figsize=(20,15))
sns.heatmap(corr,annot=True)

#sns.distplot(df["type"])

sns.distplot(df["isFraud"])

df = df.drop(['isFlaggedFraud','nameOrig','nameDest'], axis=1)

df.head()

#lable encoding
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df.type=le.fit_transform(df.type)
df.head()
```

outlayer detection

```
sns.boxplot(df.type)
```

splitting of data

```
x=df.drop(['isFraud'], axis=1)
x.head()
```

#feature scaling

```
from sklearn.preprocessing import MinMaxScaler
ms=MinMaxScaler()
x=pd.DataFrame(ms.fit_transform(x),columns=x.columns)
x.head()

y=df['isFraud']
y.head()

x.shape, y.shape
```

splitting data intlo train and test

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
random_state=0)


x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

#model building

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import RandomizedSearchCV
```

```
from sklearn.metrics import
accuracy_score,confusion_matrix,classification_report,roc_auc_score as
ras,roc_curve


models = [ RandomForestClassifier(), DecisionTreeClassifier(),
        LogisticRegression(),XGBClassifier(),SVC(probability=True)]

for i in range(len(models)):
models[i].fit(x_train, y_train)
print(f'{models[i]} : ')

y_train_predict = models[i].predict_proba(x_train)[:, 1]
print('traning Accuracy : ', ras(y_train, y_train_predict))

y_test_accuracy = models[i].predict_proba(x_test)[:, 1]
print('testing Accuracy : ', ras(y_test, y_test_accuracy))
print()




#saving ths svm model as we are getting better accuracy

svc_model = SVC(probability = True)
svc_model.fit(x_train, y_train)

y_test_predict = svc_model.predict(x_test)

accuracy = accuracy_score(y_test, y_test_predict)
print(accuracy)

confusion = confusion_matrix(y_test, y_test_predict)
```

```
print('Confusion Matrix:')
print(confusion)

report_svc = classification_report(y_test, y_test_predict)
print('\nClassification Report (SVC):\n', report_svc)

#import pickle
#pickle.dump(svc_model, open('model.pkl','wb'))
#pickle.dump(le, open('label_encoder.pk1','wb'))
#pickle.dump(ms, open('scaler.pkl','wb'))
x.head()


svc_model.predict(ms.transform([[1,1,181.00,181.0,0.00,21182.0,0.0]]))
```

GITHUN LINK:- https://github.com/smartinternz02/SI-GuidedProject-611870-1700370649