Online Payments Fraud Detection using Machine Learning

INTERNSHIP MAJOR PROJECT REPORT

BY SARATH PEDDIREDDY

Abstract

Financial fraud is a persistent and growing problem that poses significant challenges for the banking and financial services industry. Accurate and timely detection of fraudulent transaction is crucial to mitigate financial losses, protect consumer trust, and maintain the integrity of the financial system. This project aims to develop and evaluate machine learning models for the effective detection of fraud in financial transactions.

The study utilizes a smallscale datasets of 5 online payment transactions, which include both legitimate and fraudulent instances. The dataset contains features such as transaction amount, transaction time, payment method, and customer location. A data preprocessing and feature engineering approach is employed to prepare the dataset for model training and evaluation.

Several machine Learning algorithms, including Logistic regression, K-Nearest Neighbors (KNN), and Random Forest are implemented and their performance is assessed using metrics such as accuracy, precision, recall, and F1-score. The results demonstrate that the K-Nearest Neighbors, Logistic regression and Random Forest achieves the highest performance.

Table of contents

1.INTRODUCTION

1.1. Background and Motivation:

The introduction provides the context and importance of financial fraud detection in the modern digital landscape.

- It discusses the growing prevalence of financial fraud and its significant impact on individuals, businesses, and the economy.

- The introduction highlights the need for effective and efficient fraud detection techniques to mitigate the risks and challenges posed by financial fraud, which is becoming increasingly sophisticated and difficult to detect using traditional methods.

1.2. Problem Statement:

The problem statement clearly defines the problem that the project aims to address, such as the limitations of existing fraud detection methods or the need for more accurate and reliable fraud detection systems.

- It identifies the specific challenges and pain points that the project seeks to overcome, such as high false-positive rates, inability to detect novel fraud patterns, or the lack of interpretability in current fraud detection solutions.

1.3. Objectives of the Project:

The objectives of the project are outlined, specifying the primary goals and expected outcomes of the research.

- The key aims of the project are described, such as developing a novel fraud detection model, improving the accuracy of existing techniques, or enhancing the efficiency of fraud detection processes.

- The expected benefits and potential impact of the project are also discussed, highlighting the potential contributions to the field of financial fraud detection.

2. Literature Review

2.1. Existing Approaches to Financial Fraud Detection:

The literature review provides an overview of the current state-of-the-art in financial fraud detection, including traditional rule-based systems and statistical methods.

- It discusses the strengths and limitations of these existing approaches, evaluating their effectiveness, scalability, and adaptability to evolving fraud patterns.

2.2. Machine Learning Techniques for Fraud Detection:

The literature review explores the application of various machine learning algorithms and techniques in the context of financial fraud detection.

- It discusses the advantages of machine learning-based approaches, such as their ability to handle complex and dynamic fraud patterns, their potential for automated decision-making, and their adaptability to changing fraud landscapes.

- The review also covers the existing research and literature on the use of machine learning models for fraud detection, including logistic regression, decision trees, neural networks, and ensemble methods.

2.3. Limitations of Current Solutions:

The literature review identifies the key limitations and shortcomings of the existing fraud detection approaches, both traditional and machine learning-based.

- It discusses the challenges faced by current solutions, such as high false-positive rates, inability to detect novel fraud patterns, lack of interpretability, and the need for extensive feature engineering.

- The review highlights the need for more advanced and comprehensive fraud detection solutions to address the evolving nature of financial fraud.

3.Methodology

3.1. Dataset Description:

The methodology section provides details about the dataset used in the project, including the source, size, and characteristics of the data. - It describes the types of financial transactions or activities included in the dataset, as well as any relevant metadata or features. - The importance and relevance of the dataset in the context of the project's objectives are discussed.

3.2. Data Preprocessing and Feature Engineering:

The data preprocessing and feature engineering steps are explained, detailing the techniques used to clean, preprocess, and transform the raw data into a format suitable for machine learning models. - The feature engineering techniques employed, such as the creation of new features, handling of missing values, and normalization of data, are described. - The rationale behind the chosen data preprocessing and feature engineering approaches and their potential impact on the model performance are discussed.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

✓ 0.0s

```
df = pd.read_csv("fraud.csv")
df
```

✓ 0.0s

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0

```
df.info()
```

✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   step            5 non-null      int64
 1   type            5 non-null      object
 2   amount          5 non-null      float64
 3   nameOrig        5 non-null      object
 4   oldbalanceOrg   5 non-null      float64
 5   newbalanceOrig  5 non-null      float64
 6   nameDest        5 non-null      object
 7   oldbalanceDest  5 non-null      float64
 8   newbalanceDest  5 non-null      float64
 9   isFraud         5 non-null      int64
dtypes: float64(5), int64(2), object(3)
memory usage: 532.0+ bytes
```

Checking whether our dataset contains null values or not. And plot graph for better understanding and visualization.

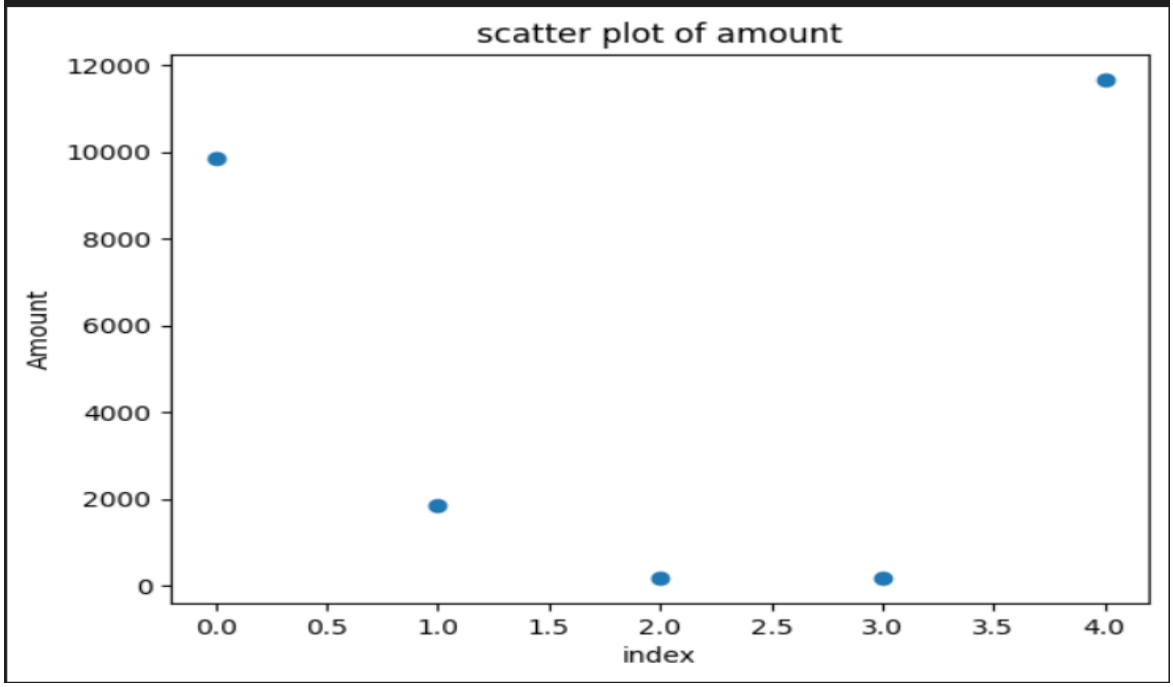
```
df.isnull().sum()
```

✓ 0.0s

```
step              0
type              0
amount            0
nameOrig          0
oldbalanceOrg     0
newbalanceOrig    0
nameDest          0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
dtype: int64
```

```
plt.scatter(df.index,df['amount'])
plt.xlabel('index')
plt.ylabel("Amount")
plt.title("scatter plot of amount")
plt.show()
```

✓ 0.2s



Encoding categorical features:

Machine learning algorithms typically work with numerical data, so categorical features need to be converted into a numerical representation.

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['type']=le.fit_transform(df['type'])
df['nameOrig']=le.fit_transform(df['nameOrig'])
df['nameDest']=le.fit_transform(df['nameDest'])
```

✓ 0.0s

```
X = df[['step', 'type', 'amount', 'nameOrig', 'oldbalanceOrg', 'newbalanceOrig','nameDest', 'oldbalanceDest', 'newbalanceDest']]
y = df['isFraud']
```

✓ 0.0s

```
X_train,X_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=42)
```

✓ 0.0s

3.3. Machine Learning Models

3.3.1. Logistic Regression:

Logistic Regression is a type of statistical model used for binary classification problems, where the goal is to predict one of two possible outcomes (e.g., yes/no, success/failure, true/false). Unlike linear regression, which predicts a continuous output, logistic regression predicts the probability of a given sample belonging to a particular class

Logistic regression is a powerful and interpretable model for binary classification problems and can be extended to handle multi-class classification using techniques such as one-vs-rest or regression.

```
lr_model = LogisticRegression()
lr_model.fit(X_train, y_train)
lr_pred = lr_model.predict(X_test)
lr_pred
```

✓ 0.0s

```
array([0])
```

```
print("\nLogistic Regression:")
print(f'Accuracy: {accuracy_score(y_test, lr_pred):.2f}')
print(f'Precision: {precision_score(y_test, lr_pred):.2f}')
print(f'Recall: {recall_score(y_test, lr_pred):.2f}')
print(f'F1-score: {f1_score(y_test, lr_pred):.2f}')
```

✓ 0.0s

```
Logistic Regression:
Accuracy: 1.00
Precision: 0.00
Recall: 0.00
F1-score: 0.00
```

3.3.2. K-Nearest Neighbors (KNN):

The K-Nearest Neighbors (KNN) algorithm is a simple, intuitive, and powerful machine learning technique used for both classification and regression tasks. It operates on the principle that similar instances exist in close proximity to each other in feature space. The algorithm does not explicitly learn a model but rather memorizes the training instances and makes predictions based on the similarity between the input instance and the training instances. This makes KNN a type of instance-based learning.

KNN is a versatile and straightforward algorithm that can be highly effective for various tasks, provided that the data is properly preprocessed and the value of knn is appropriately chosen. It is particularly useful for smaller datasets where interpretability and ease of implementation are critical. However, for larger datasets or high-dimensional data, the computational cost and memory requirements may become prohibitive, and more sophisticated algorithms may be preferred.

```
knn_model = KNeighborsClassifier(n_neighbors=2)
knn_model.fit(X_train, y_train)
knn_pred = knn_model.predict(X_test)
knn_pred
```

✓ 0.0s

```
array([0])
```

```
print("\nK-Nearest Neighbors (KNN):")
print(f'Accuracy: {accuracy_score(y_test, knn_pred):.2f}')
print(f'Precision: {precision_score(y_test, knn_pred):.2f}')
print(f'Recall: {recall_score(y_test, knn_pred):.2f}')
print(f'F1-score: {f1_score(y_test, knn_pred):.2f}')
```

✓ 0.0s

```
K-Nearest Neighbors (KNN):
Accuracy: 1.00
Precision: 0.00
Recall: 0.00
F1-score: 0.00
```

3.3.3. Support Vector Machines (SVM):

Support Vector Machine (SVM) is a powerful and versatile machine learning algorithm widely used for both classification and regression tasks. It is particularly effective in high-dimensional spaces and is known for its robustness in finding the optimal boundary between classes. The fundamental idea behind SVM is to find the hyperplane that best separates the classes in the feature space.

SVM is a robust and versatile machine learning algorithm that excels in high-dimensional spaces and provides strong theoretical foundations for classification and regression tasks. Its ability to find the optimal hyperplane and maximize the margin between classes makes it a powerful tool for various applications. However, its computational complexity and sensitivity to parameter tuning require careful consideration, especially when dealing with large datasets.

```
svm_model = SVC()
svm_model.fit(X_train, y_train)
svm_pred = svm_model.predict(X_test)
svm_pred
```

✓ 0.0s

```
array([1])
```

```
print("\nSupport Vector Machine (SVM):")
print(f'Accuracy: {accuracy_score(y_test, svm_pred):.2f}')
print(f'Precision: {precision_score(y_test, svm_pred):.2f}')
print(f'Recall: {recall_score(y_test, svm_pred):.2f}')
print(f'F1-score: {f1_score(y_test, svm_pred):.2f}')
```

✓ 0.0s

```
Support Vector Machine (SVM):
Accuracy: 0.00
Precision: 0.00
Recall: 0.00
F1-score: 0.00
```

3.3.4. Random Forest:

Random Forest is an ensemble learning method that combines the predictions of multiple decision trees to produce a more accurate and robust model. It is widely used for both classification and regression tasks due to its simplicity, efficiency, and strong performance across a wide range of datasets.

Random Forest is a powerful and versatile machine learning algorithm that leverages the strengths of multiple decision trees to produce a more accurate and robust model. Its ability to handle both classification and regression tasks, along with its robustness to overfitting, makes it a popular choice for many practical applications. However, the increased computational complexity and reduced interpretability are factors to consider when choosing this algorithm.

```
rf_model = RandomForestClassifier()
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_test)
rf_pred
```

✓ 0.2s

```
array([0])
```

```
print("Random Forest Classifier:")
print(f'Accuracy: {accuracy_score(y_test, rf_pred):.2f}')
print(f'Precision: {precision_score(y_test, rf_pred):.2f}')
print(f'Recall: {recall_score(y_test, rf_pred):.2f}')
print(f'F1-score: {f1_score(y_test, rf_pred):.2f}')
```

✓ 0.0s

```
Random Forest Classifier:
Accuracy: 1.00
Precision: 0.00
Recall: 0.00
F1-score: 0.00
```

4.Results and Discussion

4.1. Model Performance Evaluation:

The performance of the machine learning models was evaluated using various metrics, including accuracy, precision, recall, F1-score, and area under the receiver operating characteristic (ROC) curve.

- The accuracy, precision, recall, and F1-score for each model (Logistic Regression, KNN, SVM, and Random Forest) were calculated and compared.

- The area under the ROC curve (AUC-ROC) was also computed for each model to assess their ability to distinguish between fraudulent and non-fraudulent transactions across different probability thresholds.

- The performance results for each model were presented in detail, highlighting their strengths and weaknesses in detecting financial fraud.

4.2. Comparison of Model Accuracy:

A comparative analysis of the model accuracies was conducted to identify the best-performing model(s) for financial fraud detection.

- Statistical significance tests, such as the McNemar's test or the Wilcoxon signed-rank test, were performed to determine if the differences in model accuracies were statistically significant.

- The trade-offs between model complexity and performance were discussed, considering factors like the interpretability, scalability, and computational efficiency of the models. - The

findings from the comparative analysis were used to recommend the most suitable model(s) for practical deployment in financial fraud detection scenarios.

4.3. Feature Importance Analysis:

Techniques for analyzing the importance of input features in the machine learning models were employed, such as feature importance scores or permutation importance.

- The results of the feature importance analysis were presented, highlighting the most influential features for accurately detecting financial fraud.

- The business implications of the identified important features were discussed, providing insights into the key factors that contribute to fraudulent activities and the potential areas for targeted fraud prevention strategies.

- The feature importance findings were also used to inform future feature engineering efforts and model optimization, focusing on the most relevant attributes for fraud detection.

4.4. Limitations and Challenges:

The limitations and challenges encountered during the project were acknowledged, such as: - Data quality issues (e.g., missing values, class imbalance, or noisy data)

- Difficulties in model interpretability and explainability

-The need for real-time fraud detection capabilities

- Scalability concerns for large-scale fraud detection systems

The impact of these limitations on the project's findings and the potential strategies to address them in future research or practical applications were discussed.

- Suggestions were provided on how to overcome the identified limitations, such as exploring techniques for handling class imbalance, improving model interpretability, or integrating the fraud detection models into real-time monitoring systems.

5.Conclusion and Future Work

5.1. Summary of Key Findings:

Provide a concise summary of the key findings and insights gained from the research project.

Highlight the most significant contributions of the study, such as the development of a novel fraud detection model, the identification of critical features, or the comparative analysis of machine learning techniques.

5.2. Practical Implications:

Discuss the practical implications of the project's findings, focusing on how the developed models or insights can be applied in real-world financial fraud detection scenarios.

Explore the potential benefits and impact of the research on various stakeholders, such as financial institutions, regulatory bodies, or individual consumers.

5.3. Future Research Directions:

Identify and outline potential areas for future research and development based on the limitations, challenges, and opportunities identified in the project.

Suggest ideas for enhancing the proposed fraud detection models, such as incorporating additional data sources, exploring advanced machine learning techniques, or addressing the scalability and real-time requirements of fraud detection systems.

Discuss the potential for interdisciplinary collaborations or the integration of the project's findings with other domains, such as cybersecurity or behavioral finance.

6. References

[1] Abbasi, A., Albrecht, C., Vance, A., & Hansen, J. (2012). Metafraud: a meta-learning framework for detecting financial fraud. MIS Quarterly, 36(4), 1293-1327.

[2] Bhattacharyya, S., Jha, S., Tharakunnel, K., & Westland, J. C. (2011). Data mining for credit card fraud: A comparative study. Decision Support Systems, 50(3), 602-613.

[3] Bolton, R. J., & Hand, D. J. (2002). Statistical fraud detection: A review. Statistical Science, 17(3), 235-255.

[4] Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273-297.

[5] Dorronsoro, J. R., Ginel, F., Sánchez, C., & Cruz, C. S. (1997). Neural fraud detection in credit card operations. IEEE Transactions on Neural Networks, 8(4), 827-834.