Credit Card Fraud Detection

Praneeth Sai Rachuri, Sravani Neelapala

Department of Computer Engineering

San Jose State University

San Jose, CA

Abstract— **Detecting credit card fraud has become essential as online transactions grow rapidly. This project uses both supervised and unsupervised machine learning methods to identify fraud. We analyzed a dataset from Kaggle, containing transaction details and fraud labels, and improved it through feature engineering. The Isolation Forest algorithm was used to detect unusual patterns (anomalies) without needing labeled data, while the Random Forest model identified fraud in labeled data. The models were evaluated using precision, recall, F1-score, and ROC-AUC to ensure accuracy and reliability. Although there were challenges like handling imbalanced data and high computation time, the results successfully detected fraudulent transactions. Future plans include using advanced deep learning methods, such as LSTM, to predict fraud over time.**

Keywords — **Fraud detection, online transactions, supervised learning, unsupervised learning, Isolation Forest, Random Forest, machine learning, anomaly detection, LSTM, feature engineering.**

I. INTRODUCTION

Online shopping and credit cards have made life so much easier, letting us buy things quickly and conveniently. But along with this convenience comes a big issue: credit card fraud. Fraud not only causes financial losses for businesses and customers but also makes people worry about trusting digital payments. Detecting fraud is tough because it happens less often than regular transactions, making it harder to catch [4]. On top of that, scammers are always finding new ways to cheat the system, which means older detection methods aren't always reliable.

This project works on solving these problems by using machine learning to make fraud detection better and smarter. It uses tools like Isolation Forest, which finds unusual behavior without needing labeled examples, and Random Forest, which analyzes known cases to spot fraud [4]. These methods are built to handle tricky data and keep up with changing fraud tactics.

The goal of this project is to make digital payments safer for everyone. It helps reduce financial losses, protects customers' money, and builds trust in online payments. With better fraud detection, businesses and customers can feel more confident using credit cards without worrying about security risks.

II. DATASETS

We used an imbalanced dataset consisting of five files in JSON and CSV formats, sourced from Kaggle [3]. For our analysis, we selected transactions_data.csv, which contains user transaction histories, including transaction amounts, merchant details, swipe or online transaction types, and timestamps from 2010 to 2020. This file has 13,305,915 rows and 12 columns. We also used a JSON file, train_fraud_labels.json, which contains IDs and labels indicating whether each transaction is fraudulent or non-fraudulent, consisting of 8,914,963 rows and 2 columns. To prepare the data for supervised learning, we merged the two files based on the id feature. This merged dataset was then analyzed to identify fraudulent transactions.

III. DATA LOADING AND FORMATTING

The process of loading and formatting the data involves two datasets: one in CSV format and the other in JSON. The transactions_data.csv file is imported into a DataFrame using pd.read_csv, while the train_fraud_labels.json file is loaded using Python's json module. The JSON file contains two fields: id and status, indicating whether a transaction is fraudulent or not. The transaction IDs and fraud labels are extracted into two separate lists, id and status, and then combined into a new DataFrame. To ensure consistency, the id column in both datasets is converted to a numeric format. Finally, the two Data Frames are merged based on the id column using an inner join, which ensures that only records present in both datasets are included. The resulting merged dataset is now ready for supervised learning, where transaction data is paired with fraud labels to train the model.

IV. LIBRARIES USED

When we worked on this fraud detection project, we needed some helpful tools to make our job easier. Think of these libraries as special assistants that help us handle different parts of our project.

First up, NumPy and Pandas were our data helpers. They're like super-organized assistants who can quickly clean up messy information, fill in missing pieces, and arrange our data neatly. Imagine having someone who can sort through a huge pile of papers and make everything make sense - that's what these libraries do for our data.

Scikit-learn was our machine learning expert [4]. This library is fantastic because it helped us build our smart fraud detection models. It's like having a detective who can spot unusual patterns and decide whether a transaction looks suspicious. Not only does it help create the models, but it also gives us ways to check how good our detective is - measuring things like how accurate the model is at catching fraud.

For making things look pretty and understandable, we used Matplotlib and Seaborn. These are our graphic design experts. They turn all our complex data and results into clear, easy-to-read charts and graphs. Instead of looking at boring numbers, we can now see colorful visuals that tell a story.

JSON was our data translator. It helped us work with structured information, making it easy to load and process our data files. Think of it like a universal language that helps different parts of our project talk to each other smoothly.

Finally, train_test_split was like our project's training coach. It helped us divide our data into two groups - one for teaching our model (training) and another for testing how well it learned. This is super important because we want to make sure our fraud detection system can handle new, unseen situations.

Each of these libraries played a crucial role in making our fraud detection project work smoothly and effectively. They're like a team of specialists, each bringing their own unique skills to solve a complex problem.

V. Feature engineering

The preprocessing steps involved for both supervised and unsupervised learning.
As we are considering two files: transactions_csv and train_fraud_labels.json.

A. Un Supervised Learning

To analyze transaction data for fraud detection on a daily basis, the first step is to convert the date column into a datetime format, ensuring proper handling of time-based operations. Transactions are then grouped by day, allowing the calculation of total transaction amounts for each date. To ensure accuracy, any non-numeric characters in the amount field are removed, and the values are converted to numeric format. After this, the day-to-day changes in transaction amounts (amount_diff) are calculated using the diff () function, which highlights shifts in transaction patterns from one day to the next. Any missing values created during this process are removed to keep the dataset clean and reliable. This feature engineering process in unsupervised learning helps to identify the unusual patterns in the data like sharp spikes or sudden drops in transactions showing fraudulent behavior [4]. It showed key details for fraud detection models using supervised learning. The patterns reveal anomalies without the use of labelled data.

B. Supervised Learning

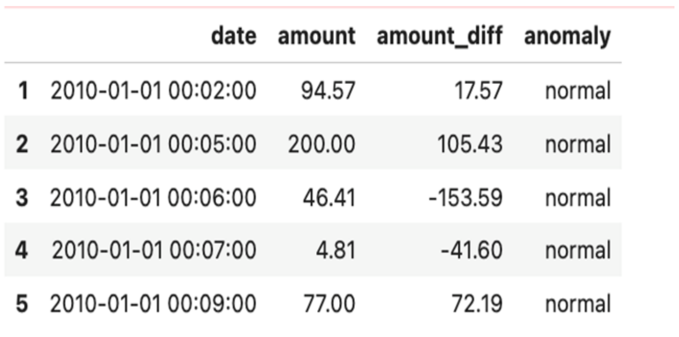
For supervised learning, the first step is to make sure the data is consistent, starting with converting the id column to a numeric format in both datasets before merging them. The two datasets are then joined based on the common id column, keeping only the records that match. Once merged, any missing values in important columns like merchant_state, zip, and errors are filled in with the most common value (the mode). The amount column is cleaned by removing special characters like $ and, and then converted into a numeric format for accurate calculations. The date column is also changed into a proper datetime format, allowing us to extract useful time-related features such as the year, month, day, hour, and weekday. These steps like cleaning the data, being consistent, considering useful features make the data suitable for performing supervised learning model.

VI. Models

A. Unsupervised Learning (Isolation Forest)

Unsupervised learning was implemented, as it does not require labeled training data. An unsupervised decision-tree-based algorithm was used for outlier detection, focusing on anomaly detection through a binary tree structure [4]. The Isolation Forest algorithm was applied, tuning parameters like contamination, and the amount feature was incorporated for time-series analysis. Anomalies were detected by identifying sudden positive or negative changes in transaction amounts, as drastic fluctuations often signal fraudulent activities. The Isolation Forest model was trained on the amount and amount_diff features of the time-series data to identify anomalies. With a contamination setting of 0.05, indicating that 5% of the data is expected to be anomalous, the model was applied to predict anomalies. Predictions were stored in a new column labeled "anomaly," with the values -1 for anomalies and 1 for normal transactions. These values were then remapped to 0 for normal transactions and 1 for anomalies, helping to identify suspicious transactions that may indicate fraud. Below is a screenshot of the first five rows based on time-series analysis as shown in "Table. 1".

TABLE 1: REPRESENTATION OF GANTT CHART



	date	amount	amount_diff	anomaly
1	2010-01-01 00:02:00	94.57	17.57	normal
2	2010-01-01 00:05:00	200.00	105.43	normal
3	2010-01-01 00:06:00	46.41	-153.59	normal
4	2010-01-01 00:07:00	4.81	-41.60	normal
5	2010-01-01 00:09:00	77.00	72.19	normal

B. *Supervised Learning (Random Forest Classifier)*

Supervised learning is applied to verify the relevance of discovered patterns, and hyperparameter tuning is performed to optimize model performance. The Random Forest classifier, using multiple decision trees, handles complex patterns and avoids overfitting, making it suitable for imbalanced datasets [4]. Before we can use a dataset for machine learning, we need to get it ready first, so categorical columns are encoded into numeric columns with Label Encoder, so now we have use_chip, merchant_city, merchant_state and errors as labels. This enables the model to deal with these non-numeric features. Next the feature set, X is built by removing the target variable (Status) and non-numeric columns like date. The Status column yields the target variable y, resulting in 1 (fraud) for 'Yes' and 0 (non-fraud) for 'No'. The next step is to divide the data into 80% training and 20% testing sets. A Random Forest Classifier with 100 estimators, balanced class. Once the model is trained on the training set, predictions are carried out on the test set. However, a limitation of this approach is the significant computation time required for training.

VII. Evaluation Metrics

A. *Unsupervised Learning (Isolation Forest)*

In unsupervised learning, to evaluate the model, but we can visually assess the results. The image below illustrates anomaly detection on a time series from 2010 to 2020, where the red dots in “Fig. 1” represent the identified anomalies. The red dots highlight the points in time where unusual patterns were detected, helping to identify potential issues in the data.

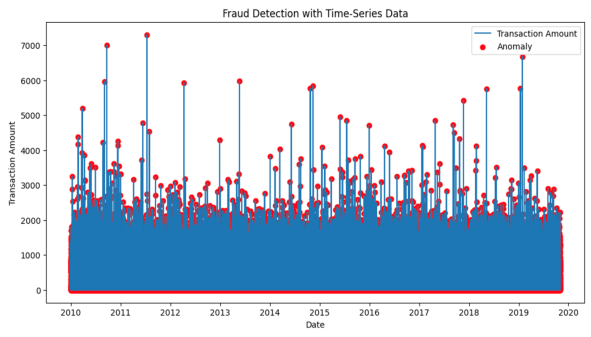
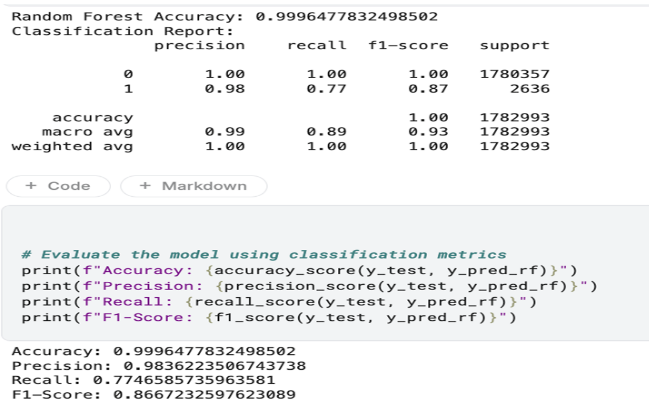


Figure 1. Fraud Detection with Time-Series Data

B. *Supervised Learning (Random Forest Classifier)*

Accuracy isn't always the best measure of success, especially in cases where the data is imbalanced. For example, in applications like fraud detection or intrusion detection, we're more concerned with identifying rare events—like fraudulent transactions or security breaches—rather than the majority of normal cases. In these situations, relying on accuracy can be misleading, because a model that simply predicts "normal" all the time might still appear accurate, even though it's failing to spot the fraud or intrusion. IN addition to accuracy, one should focus on other metrics like precision, recall, and F1 score, which give a clearer picture
of how well the model is identifying those important, but rare, positive cases.

The evaluation metrics like accuracy and a classification report are two ways we evaluate our model's performance which gives us more idea into the precision, recall and F1 score of the model and thus lets us know how efficient our model is at detecting fraud[4] as shown in “Fig. 2” and “Table 2”



```
Random Forest Accuracy: 0.9996477832498502
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00   1780357
           1       0.98      0.77      0.87      2636

    accuracy                           1.00   1782993
   macro avg       0.99      0.89      0.93   1782993
weighted avg       1.00      1.00      1.00   1782993
```

```
# Evaluate the model using classification metrics
print(f"Accuracy: {accuracy_score(y_test, y_pred_rf)}")
print(f"Precision: {precision_score(y_test, y_pred_rf)}")
print(f"Recall: {recall_score(y_test, y_pred_rf)}")
print(f"F1-Score: {f1_score(y_test, y_pred_rf)}")
```

```
Accuracy: 0.9996477832498502
Precision: 0.9836223506743738
Recall: 0.7746585735963581
F1-Score: 0.8667232597623089
```

Figure 2: Classification Report and evaluation metrics

TABLE 1. EVALUATION METRICS

Accuracy	99%
Precision	98%
Recall	76%
F1 score	86%

VIII. Visualization

A. *Confusion matrix*

The confusion matrix was plot by comparing the actual labels with the predicted labels as shown in “Fig. 3”. This matrix is then displayed using a visualization tool that labels the axes based on the classes predicted by the random forest model. To understand the model performance confusion matrix is shown in a plot with title ‘Confusion Matrix’.

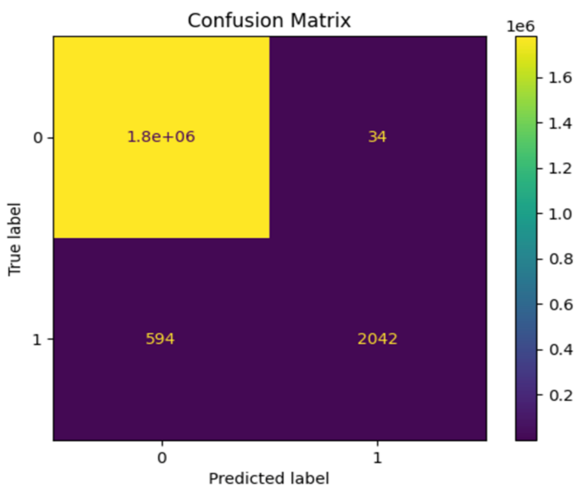


Figure 3: Confusion Matrix

B. *ROC-AUC Curve*

Initially, calculated the predicted probabilities of the random forest model and selected the positive class probabilities. The next step is to calculate the ROC curve comparing the true labels to these predicted probabilities [4]. We can use the roc_curve function to get the false positive rate (FPR), true positive rate (TPR), and thresholds. Next, to summarize a models ability to distinguish between classes we can compute the Area Under the Curve (AUC) score. The trade-off plot was shown by ROC curve with AUC score in “Fig. 4” .

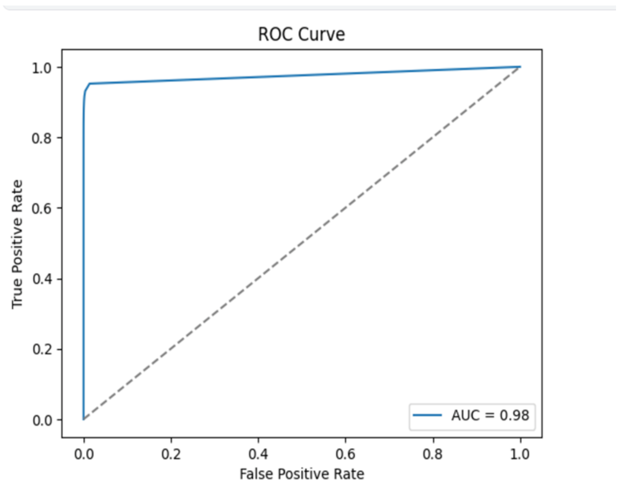


Figure 4: ROC-AUC curve

C. *Correlation matrix*

To understand the relationship between variables, we calculate the correlation between the numeric columns. This tells us how strongly each feature is connected to the others as shown in “Fig. 5”. To make this easier to understand, we use a heatmap, where the correlation values are shown in each cell. The color scale helps us see which features are positively or negatively related, warmer colors represent a stronger positive correlation, while cooler colors show a negative one.

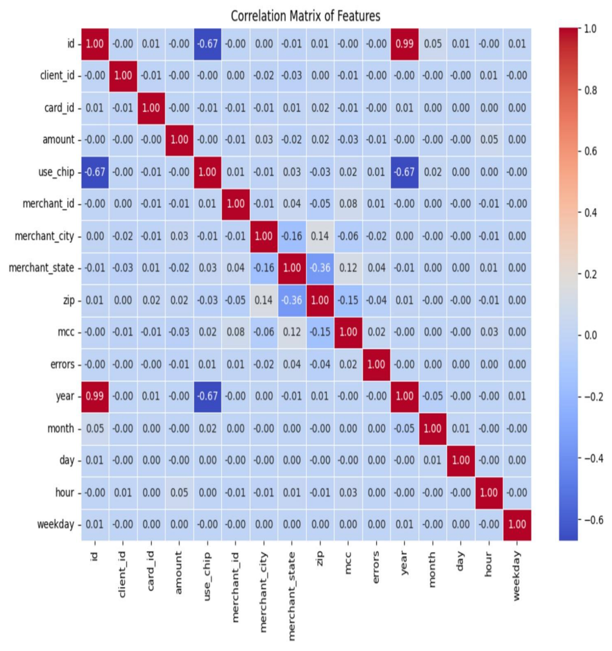


Figure 5: Correlation matrix of features

D. *Time-series*

To explore trends and seasonal patterns in fraud occurrences between 2010 and 2020, the data is first organized with a proper datetime index as shown in “Fig. 6”. This ensures the information is in the right format for time-based analysis. After sorting the data by date, a plot is created to show the number of fraud transactions recorded each day, with markers for each data point. The plot is titled "Fraud Occurrences Over Time," with the date on the x-axis and the number of fraud transactions on the y-axis. By looking at the plot, we can identify patterns where in 2014 there are more fraud cases happened when compared with other years.

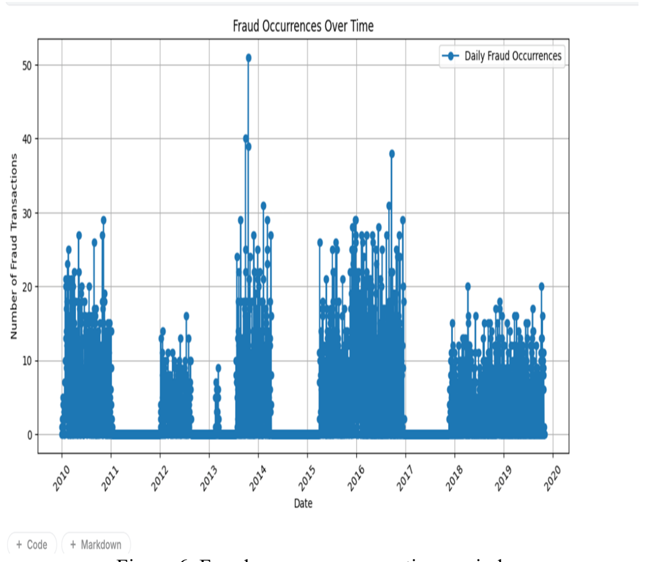


Figure 6: Fraud occurrences over time period

IX. GANTT CHART

TABLE 3: REPRESENTATION OF GANTT CHART

DATE	WORK DISTRIBUTION
November 1st	Project signup
November 8th	Selecting for proper dataset
November 15th	Feature engineering
November 22nd	Feature engineering, model selection
November 29th	Unsupervised, Supervised and deep learning techniques implementation and evaluation
December 7th	Paper presentation submission and working on LSTM model
December 14th	Report writing . Code and report submission

X. CONCLUSION

The implementation of unsupervised (Isolation Forest) and supervised (Random Forest classifier) learning effectively detected anomalies and fraudulent transactions. The enhanced feature engineering techniques such as time-based features and label encoding, converting raw time stamps to datetime objects, and imputation techniques significantly improved model performance.

Challenges and Limitations include finding imbalanced datasets, feature engineering process and longer computational time for modelling the data.

XI. FUTURE WORK

A. LSTM Based Deep Learning for Future Fraud Detection

Proposed future work for fraud detection is predicting future fraud events using LSTM-based deep learning [2]. Long Short-Term Memory (LSTM), a type of Recurrent Neural Network (RNN), is specifically designed for sequential data like time series. It is capable of learning long-term dependencies in sequential data, making it ideal for time-series forecasting.
The first step defines the dataset: in this phase, we group the transaction records by date and count the number of frauds per day. Finally, Minmax Scaler is used to clean and scale these counts into the range that is more feasible as input data to LSTM. Then, we create sequences of data when the model is trained with previous fraud counts (e.g., 7 days) to predict the next day's fraud occurrence. Next, the data is divided into training and test for evaluation. A bidirectional LSTM network is built, which contains dropout layers and batch normalization layers for the purpose of preventing overfitting and stabilizing training. The training set is used to train the model and avoid overfitting with early stopping. The model is evaluated by predicting fraud counts for test set and then inverse-scaled to actuals after the training. Fraud Classification Model: For classification purposes a binary threshold is applied on the model. The accuracy, precision, recall and F1-score are computed to measure performance of the model. This method can make accurate trends for future fraud forecasts, as well as anomalies transactions.

REFERENCES

[1] S. Abedin, A. Lahiri, and B. Mukherjee, "Credit card fraud detection using supervised and unsupervised machine learning algorithms," *Engineering Applications of Artificial Intelligence*, vol. 85, 2019. https://www.sciencedirect.com/science/article/abs/pii/S0952197618301520?fr=RR-2&ref=pdf_download&rr=8eb9987ca8682379

[2] M. A. J. Qasem and J. R. Moya, "A novel fraud detection system using deep learning and decision analytics," *Annals of Operations Research*, vol. 306, no. 1, pp. 345-368, 2021. https://link.springer.com/article/10.1007/s10479-021-04149-2.

[3] Kaggle, "Transactions fraud dataset," *Kaggle Datasets*. https://www.kaggle.com/datasets/computingvictor/transactions-fraud-datasets.

[4] GeeksforGeeks, "Supervised vs. unsupervised learning," *GeeksforGeeks*. https://www.geeksforgeeks.org/supervised-unsupervised-learning/

[5] OpenAI, "ChatGPT: AI model for natural language understanding," *OpenAI Chat*. https://chat.openai.com.