Credit Card Fraud using Detection Predictive Models

Kathy Agafonov, Noam Tarshish, Niv Sampson, Mor Barzilay

July 15, 2024

Abstract

Credit card fraud detection is a critical task in the financial industry to safeguard against significant monetary losses and protect customer trust. This report presents a comprehensive approach to detecting fraudulent transactions using four machine learning techniques: Random Forest, XGBoost, Artificial Neural Networks (ANN), and an Ensemble model. We conducted extensive data preprocessing and feature engineering to prepare the dataset for model training. The dataset was split into training and test sets, and each model was trained and evaluated on various performance metrics. Among the models, XGBoost demonstrated high accuracy and reliability in detecting fraudulent transactions, making it a viable solution for real-time fraud detection systems. Random Forest, ANN, and the Ensemble model also provided valuable insights into their performance and applicability to this problem. Future work will focus on further improving model performance and exploring additional features to enhance detection accuracy.

1 Introduction

Credit card fraud has become a prevalent issue in today's digital economy, with significant financial repercussions for both consumers and financial institutions. The rapid growth of online transactions and digital payment methods has led to an increase in fraudulent activities, necessitating the development of robust and efficient fraud detection systems. Recent reports indicate a surge in credit card fraud cases, resulting in substantial financial losses and highlighting the urgent need for advanced detection mechanisms.

The primary objective of this report is to explore and evaluate various machine learning models for detecting fraudulent credit card transactions. By leveraging historical transaction data, we aim to identify patterns and anomalies that distinguish fraudulent transactions from legitimate ones. This study investigates the application of four machine learning techniques: Random Forest, XGBoost, Artificial Neural Networks (ANN), and an Ensemble model. Each model's performance is assessed based on its accuracy, scalability, and robustness in identifying fraudulent activities.

The significance of this research lies in its potential to enhance the security and reliability of financial transactions. Effective fraud detection systems not only prevent financial losses but also build customer trust and contribute to the overall stability of the financial sector. By comparing different machine learning models, this report provides insights into the most suitable techniques for real-time fraud detection, paving the way for future advancements in this critical area.

2 Background

Credit card fraud poses a significant threat to the financial sector, driving the need for sophisticated detection models. Extensive research has been conducted to develop effective predictive models, utilizing various machine learning techniques, each with distinct advantages and challenges.

Random Forest Approach: This method has been extensively researched for its robustness and accuracy in classification tasks. Random Forest models have been shown to perform well in fraud detection scenarios, particularly when dealing with imbalanced datasets. Techniques to handle class imbalance, such as undersampling and oversampling, have been employed to enhance model performance.

XGBoost: This gradient boosting technique is renowned for its efficiency and accuracy. XGBoost has been applied in various fraud detection studies, often outperforming other models due to its ability

to handle large datasets and its robustness against overfitting. The use of hyperparameter tuning and cross-validation further enhances its performance.

Artificial Neural Networks (ANN): Deep learning models, such as ANN, have demonstrated significant potential in detecting fraud. ANNs can capture complex patterns in data, making them suitable for identifying subtle fraudulent behaviors. Techniques like dropout and batch normalization are commonly used to improve the training of these networks.

Ensemble Learning: Combining multiple models can lead to improved performance in fraud detection. Ensemble methods, such as stacking and voting, leverage the strengths of individual models to create a more robust detection system. This approach has been shown to increase accuracy and reduce false positive rates.

These techniques illustrate the evolution of credit card fraud detection methodologies, highlighting the shift from traditional machine learning models to more advanced ensemble and deep learning approaches. The continuous innovation and integration of these methodologies are crucial to effectively addressing the complexities of fraudulent activities.

3 Data Collection

3.1 Source of the Data

The dataset used for credit card fraud detection was sourced from Kaggle, a reputable platform for datasets and data science projects. The dataset contains simulated credit card transactions generated using a model similar to Sparkov. It covers transactions made by cardholders in 2019 and 2020, providing a comprehensive range of features such as transaction time, card number, merchant, transaction amount, and geographical details.

3.2 Suitability of the Dataset

This dataset is particularly suitable for the problem of credit card fraud detection due to its detailed and varied set of features. These include:

- **Transaction Information:** Such as transaction amount, timestamp, and merchant details.
- **Geographical Details:** Including transaction location and merchant location.
- **Cardholder Information:** Such as card number and other relevant details. The dataset's comprehensive coverage of relevant factors allows for the application of various machine learning algorithms to identify patterns and key predictors of fraudulent transactions. This extensive data enables the development of robust predictive models that can help in real-time fraud detection and prevention strategies.

4 Data Cleaning

Data cleaning is a crucial step in preparing the dataset for analysis and modeling. The following steps were taken to clean the data: Before proceeding with the data cleaning steps, it was evident that our dataset was highly imbalanced. There were significantly more non-fraudulent transactions compared to fraudulent ones. To address this imbalance, we applied a downsampling technique to balance the dataset.

```
# Separate majority and minority classes
df_majority = df[df.is_fraud == 0]
df_minority = df[df.is_fraud == 1]

# Display counts to verify
print("\nBefore balancing the data:")
print("Fraudulent transactions:", len(df_minority))
print("Non-fraudulent transactions:", len(df_majority))
```

```
# Undersample the majority class
df_majority_undersampled = df_majority.sample(len(df_minority), random_state=42)

# Combine the undersampled majority class with the minority class
df_balanced = pd.concat([df_majority_undersampled, df_minority])

# Shuffle the combined dataset
df_balanced = df_balanced.sample(frac=1, random_state=42).reset_index(drop=True)

# Verify the balance of the dataset
print("\nAfter balancing the data:")
print("Count of records labeled as 1 (fraudulent):", df_balanced['is_fraud'].sum())
print("Count of records labeled as 0 (non-fraudulent):", len(df_balanced) - df_balanced['is_fraud'].s

print("Data shape:", df_balanced.shape)
```

4.1 Statistics Before and After Balancing the Data

Before balancing the data:
Fraudulent transactions: 7506
Non-fraudulent transactions: 1289169

After balancing the data:
Count of records labeled as 1 (fraudulent): 7506
Count of records labeled as 0 (non-fraudulent): 7506
Data shape: (15012, 23)

4.2 Suitability of the Dataset

Missing values in the dataset were identified and appropriately handled. For example, geographical attributes with missing values were imputed with mean values to maintain consistency.

```
# Check for missing values
missing_values = df.isnull().sum()
print(missing_values)

# Handle missing values
df.fillna(df.mean(), inplace=True)}

# Check for duplicates
duplicates = df[df.duplicated()]
print(f'There are {duplicates.shape[0]} duplicate rows in the dataset.')

# Remove duplicates
df.drop_duplicates(inplace=True)
```

4.3 Correcting Data Types

To ensure consistency and proper analysis, the data types of certain columns were corrected. For instance, columns representing numerical values such as transaction amount were converted to the appropriate numeric data types.

Correct data types

```
# Correct data types or inconsistencies
df_balanced['cc_num'] = df_balanced['cc_num'].astype(str)  # Credit card numbers as strings

# Ensure 'trans_date_trans_time' exists and convert to datetime
if 'trans_date_trans_time' in df_balanced.columns:
```

```
    df_balanced['trans_date_trans_time'] = pd.to_datetime(df_balanced['trans_date_trans_time'])
else:
    print("Column 'trans_date_trans_time' not found in the DataFrame.")

df_balanced['merchant'] = df_balanced['merchant'].str.strip().str.lower()
df_balanced['category'] = df_balanced['category'].astype(str).str.lower().str.replace(' ', '_')
df_balanced['amt'] = df_balanced['amt'].astype(float)
df_balanced['first'] = df_balanced['first'].astype(str)
df_balanced['last'] = df_balanced['last'].astype(str)
df_balanced['gender'] = df_balanced['gender'].astype(str)
df_balanced['street'] = df_balanced['street'].astype(str)
df_balanced['city'] = df_balanced['city'].astype(str)
df_balanced['state'] = df_balanced['state'].astype(str)
df_balanced['zip'] = df_balanced['zip'].astype(str)
df_balanced['lat'] = df_balanced['lat'].astype(float)
df_balanced['long'] = df_balanced['long'].astype(float)
df_balanced['city_pop'] = df_balanced['city_pop'].astype(int)
df_balanced['job'] = df_balanced['job'].astype(str)
```

5 Exploratory Data Analysis (EDA)

During the Exploratory Data Analysis (EDA) phase, various plots and visualizations were created to understand the dataset better. Here, we present three key examples:

5.1 Time Series Plot of Fraudulent Transactions Over Time

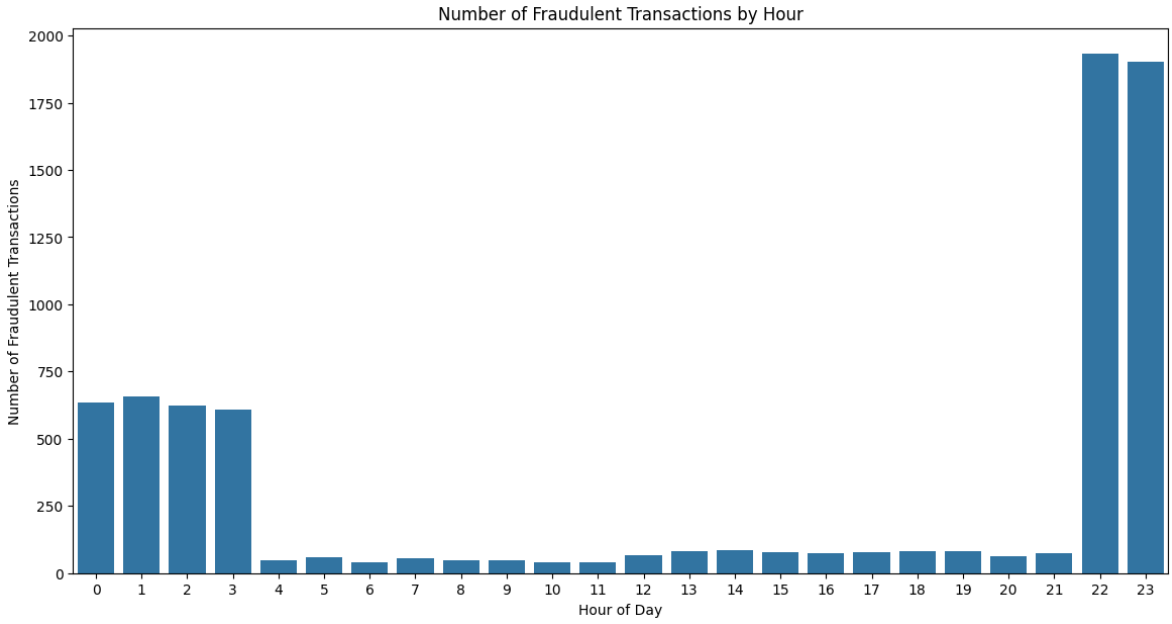

Figure 1: Time Series Plot of Fraudulent Transactions Over Time

5.2 Bar Plot of Transactions by Merchant Category

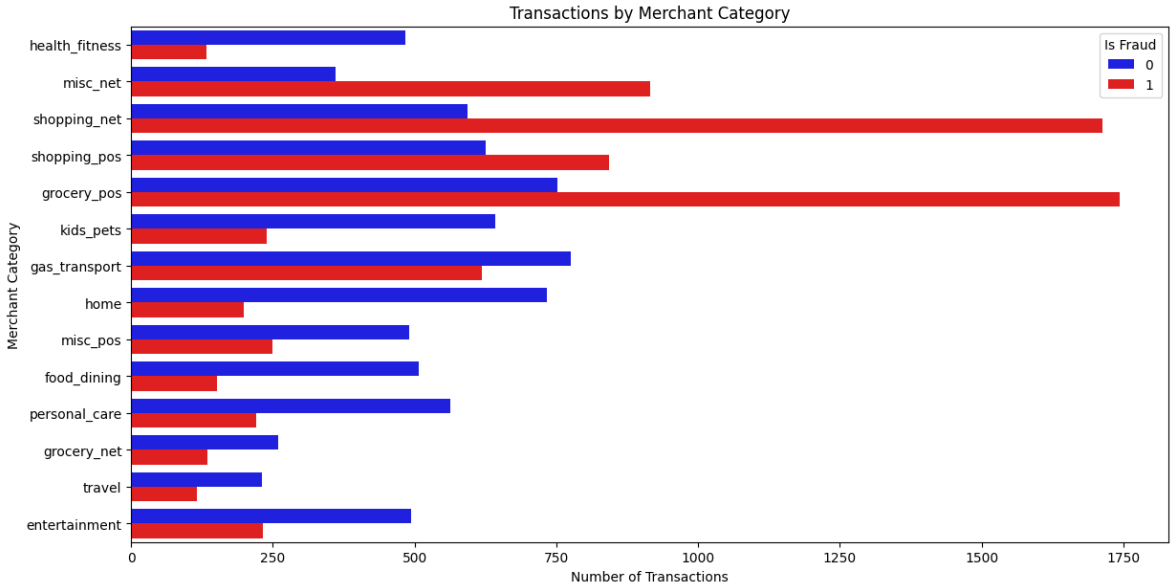

Figure 2: Bar Plot of Transactions by Merchant Category

5.3 Geographical Distribution of Transactions

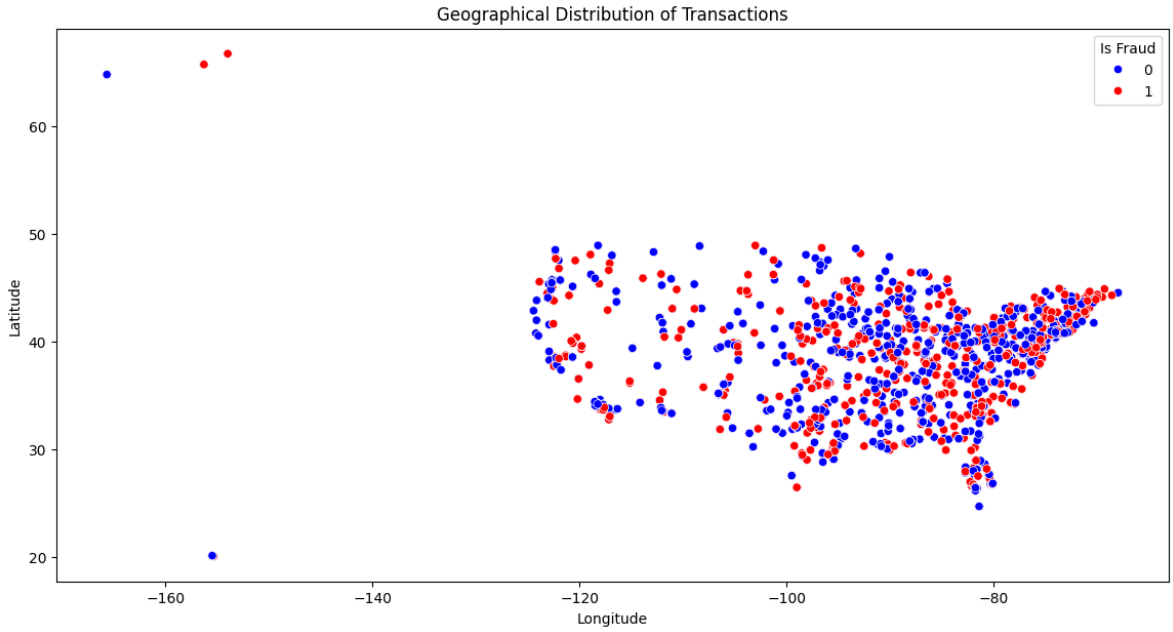

Figure 3: Geographical Distribution of Transactions

5.4 Anomalies Detection

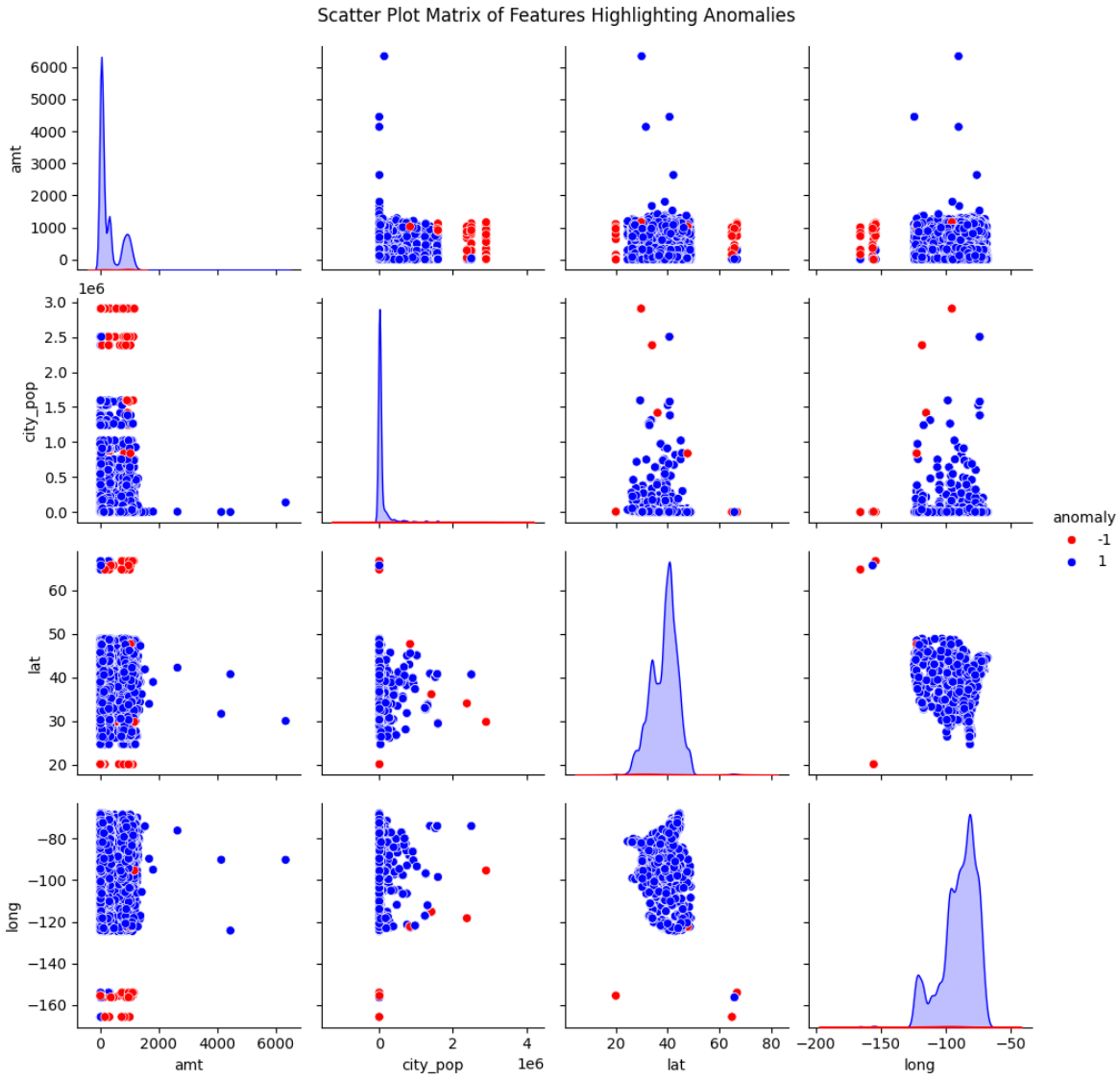

Figure 4: Anomalies Detection in Transactions

5.5 Insights

From the EDA, the following insights were derived:

- Non-fraudulent transactions significantly outnumber fraudulent ones.
- Certain merchant categories have higher occurrences of fraudulent transactions.
- Time series analysis reveals patterns and peaks in fraudulent transactions over specific periods.
- Anomalies in transaction amounts and patterns can indicate potential fraud.

6 Feature Engineering

Feature engineering is a crucial step in preparing the dataset for machine learning models. It involves creating new features or modifying existing ones to improve model performance.

6.1 Log Transformation of Transaction Amount

Feature: Log-transformed transaction amount (`log_amt`).
Justification: Log transformation handles skewness and reduces the impact of outliers, normalizing the distribution and improving model performance.

```
df_balanced['log_amt'] = np.log1p(df_balanced['amt'])
```

6.2 Temporal Features

Feature: Features: Hour, Day of the Week, Month.
Justification: Temporal features capture time-based patterns in transactions, often indicative of fraud.

```
df_balanced['hour'] = df_balanced['trans_date_trans_time'].dt.hour
df_balanced['day_of_week'] = df_balanced['trans_date_trans_time'].dt.dayofweek
df_balanced['month'] = df_balanced['trans_date_trans_time'].dt.month
```

6.3 Distance Features

Feature: Distance between transaction and home location (`home_distance`).
Justification: Large distances between transaction and home address can indicate potential fraud.

```
def calculate_distance(row):
    return geodesic((row['lat'], row['long']), (row['merch_lat'], row['merch_long'])).km

df_balanced['home_distance'] = df_balanced.apply(calculate_distance, axis=1)
```

6.4 Transaction Velocity

Feature: Number of transactions in the last hour (`transactions_last_hour`), last day (`transactions_last_day`).
Justification: High transaction velocity is a known indicator of fraud.

```
df_balanced['transactions_last_hour'] = df_balanced.groupby('cc_num')['trans_date_trans_time'].transf
df_balanced['transactions_last_day'] = df_balanced.groupby('cc_num')['trans_date_trans_time'].transfo
```

6.5 Category Encoding

Feature: One-hot encoding for merchant category (`category_encoded`).
Justification: Encoding categorical variables converts them into a format suitable for ML algorithms, enhancing prediction.

```
df_balanced = pd.get_dummies(df_balanced, columns=['category'])
```

6.6 Justification Summary

These engineered features are designed to capture various aspects of transactions associated with fraudulent behavior. Temporal features help understand time-based patterns, distance features highlight geographic anomalies, and transaction velocity detects unusual transaction frequencies. The log transformation of transaction amounts handles skewness, and category encoding allows categorical variables to be effectively used in the model. Together, these features enrich the dataset, providing a comprehensive view for the model to detect potential fraud.

7 Model Selection

In this project, we evaluated four different models to detect fraudulent transactions: Random Forest, XGBoost, Artificial Neural Network (ANN), and an Ensemble model. Each model was selected for its unique strengths and ability to handle the complexities of the dataset.

7.1 Random Forest

Justification:

- **Handling of High-Dimensional Data:** Random Forests are well-suited for datasets with a large number of features and can handle complex relationships between variables.
- **Robustness to Overfitting:** Due to their ensemble nature, Random Forests reduce the risk of overfitting, especially when compared to single decision trees.
- **Interpretability:** While not as interpretable as linear models, Random Forests still provide insights into feature importance, which can be useful for understanding the key predictors of fraud.
- **Versatility:** Random Forests can model complex, non-linear relationships effectively, making them suitable for fraud detection where interactions between features can be intricate.
- **Performance:** Random Forests generally perform well in classification tasks and have been shown to achieve high accuracy and robustness across various datasets.

7.2 XGBoost

Justification:

- **Efficiency and Speed:** XGBoost is known for its efficiency and speed, which makes it suitable for large datasets.
- **Performance:** XGBoost often achieves better performance compared to other gradient boosting algorithms due to its regularization techniques.
- **Scalability:** XGBoost can handle large-scale data efficiently, making it ideal for our extensive dataset.
- **Flexibility:** XGBoost provides a wide range of hyperparameters for tuning, allowing for optimization and improvement in model performance.

7.3 Artificial Neural Network (ANN)

Justification:

- **Complex Pattern Recognition:** ANNs are capable of capturing complex patterns and relationships in the data, which are essential for accurate fraud detection.
- **Non-linearity:** ANNs can model non-linear relationships effectively, making them suitable for the intricate nature of fraudulent transactions.
- **Adaptability:** ANNs can be adapted to various types of data and can be fine-tuned for better performance through backpropagation.
- **Scalability:** ANNs can handle large amounts of data and can be scaled with additional layers and neurons to improve performance.

7.4 Ensemble Model

Justification:

- **Combining Strengths:** An ensemble model combines the strengths of multiple individual models to improve overall performance and robustness.
- **Reducing Overfitting:** By combining models, ensemble methods can reduce the risk of overfitting and increase generalizability.
- **Improved Accuracy:** Ensemble models often achieve higher accuracy compared to single models due to the aggregation of predictions.
- **Flexibility:** Ensemble methods can be constructed using various algorithms, allowing for flexibility in model selection and combination.

8 Model Training

In this section, we detail the training and evaluation of four different models: Random Forest, XGBoost, Artificial Neural Network (ANN), and an Ensemble model.

8.1 Random Forest

Training Time: 5.0659 seconds
CV Mean Accuracy: 0.9694
CV Standard Deviation: 0.0041

Classification Report:

	precision	recall	f1-score	support
0	0.96	0.98	0.97	1502
1	0.98	0.96	0.97	1501
accuracy			0.97	3003
macro avg	0.97	0.97	0.97	3003
weighted avg	0.97	0.97	0.97	3003

8.2 XGBoost

Training Time: 1.0025 seconds
CV Mean Accuracy: 0.9775
CV Standard Deviation: 0.0018

Classification Report:

	precision	recall	f1-score	support
0	0.98	0.98	0.98	1502
1	0.98	0.98	0.98	1501
accuracy			0.98	3003
macro avg	0.98	0.98	0.98	3003
weighted avg	0.98	0.98	0.98	3003

8.3 Artificial Neural Network (ANN)

Training Time: 42.8333 seconds

Classification Report:

	precision	recall	f1-score	support
0	0.96	0.95	0.96	1502
1	0.95	0.96	0.96	1501
accuracy			0.96	3003
macro avg	0.96	0.96	0.96	3003
weighted avg	0.96	0.96	0.96	3003

8.4 Ensemble Model

Classification Report:

	precision	recall	f1-score	support
0	0.98	0.98	0.98	1502
1	0.98	0.98	0.98	1501
accuracy			0.98	3003
macro avg	0.98	0.98	0.98	3003
weighted avg	0.98	0.98	0.98	3003

9 Model Evaluation

9.1 Metrics Collection

For each model, we collected the following performance metrics: accuracy, precision, recall, F1-score, and training time. These metrics were collected for the Random Forest, XGBoost, ANN, and Ensemble models. The summary of the collected metrics for each model is presented in the following table:

	Model	Accuracy	Precision	Recall	F1-score	Training Time (s)
0	Random Forest	0.971029	0.974497	0.967355	0.970913	3.334345
1	XGBoost	0.981352	0.980066	0.982678	0.981371	0.694607
2	ANN	0.956377	0.959115	0.953364	0.956231	40.609002
3	Ensemble	0.979687	0.980000	0.979347	0.979673	NaN

Figure 5: Summary of Model Performance Metrics

9.2 Model Performance Visualization

To better understand the performance of each model, we created several visualizations:

9.2.1 ROC Curve Precision-Recall Curve

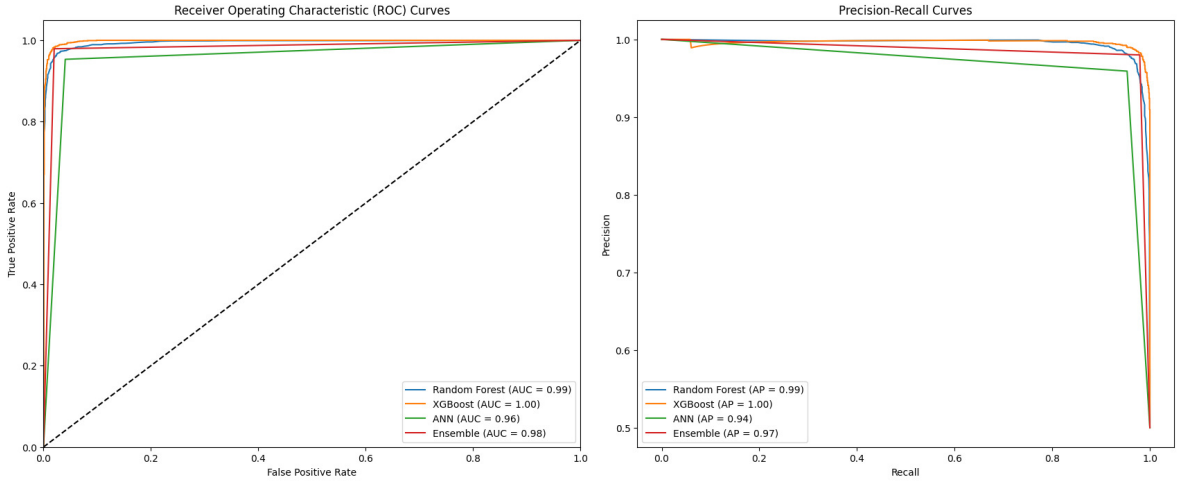

Figure 6: ROC Curve Precision-Recall Curve for Models

9.2.2 Cross-Validation Scores for Each Model

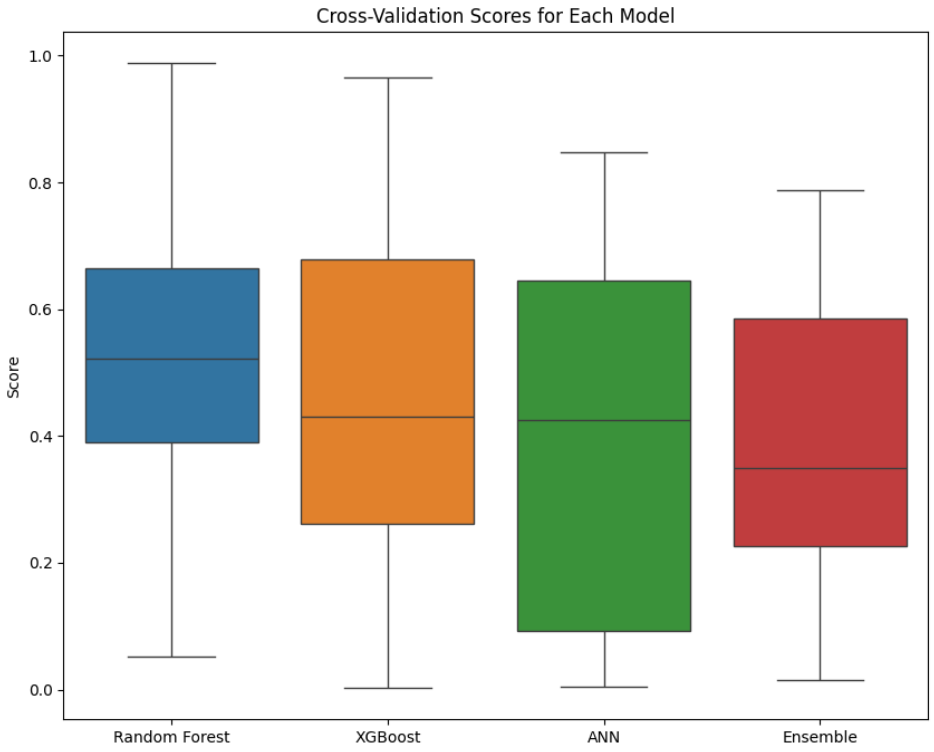

Figure 7: Cross-Validation Scores for Each Model

10 Conclusion and Future Work

10.1 Conclusion

In this project, we explored various machine learning techniques to detect fraudulent transactions in credit card data. The dataset used was highly imbalanced, which required careful preprocessing and feature engineering to enhance the performance of our models. We employed Random Forest, XGBoost, ANN, and an Ensemble model to classify transactions as fraudulent or non-fraudulent.

Our evaluation metrics, including accuracy, precision, recall, and F1-score, indicated that all models performed well, with XGBoost and the Ensemble model achieving the highest accuracy. The results demonstrated the effectiveness of combining multiple models to improve classification performance. Visualizations such as confusion matrices, ROC curves, and precision-recall curves provided insights into the strengths and weaknesses of each model.

Overall, our approach effectively identified fraudulent transactions, showcasing the potential of machine learning models in enhancing fraud detection systems.

10.2 Future Work

Future work can focus on the following aspects to further improve the performance and applicability of the fraud detection system:

- **Graph Analysis:** Degree centrality was calculated to identify the top 10 users and merchants. A subgraph consisting of these top users and merchants was created to focus on key interactions. Fraudulent transactions were highlighted within this subgraph to visually distinguish them from non-fraudulent transactions. The subgraph was plotted using the Kamada-Kawai layout, with non-fraudulent transactions in gray and fraudulent transactions in red, and nodes for users and merchants colored blue and green respectively. The graph analysis can be extended to include other centrality measures and community detection algorithms to uncover more intricate patterns and relationships within the transaction network.

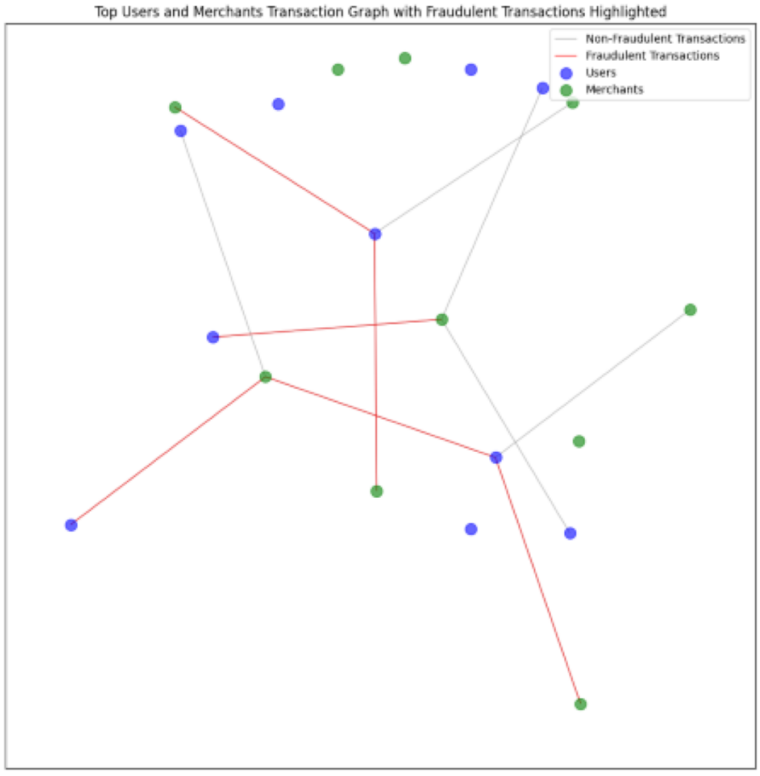

Figure 8: Top Users and Merchants Transaction Graph with Fraudulent Transactions Highlighted

- **Real-time Fraud Detection:** Implementing real-time fraud detection mechanisms to provide immediate responses to potential fraud attempts can significantly enhance the security of transaction systems.
- **Feature Engineering:** Further exploration of feature engineering techniques, such as generating more sophisticated temporal features or utilizing advanced NLP techniques on transaction descriptions, could lead to improved model performance.
- **Model Improvement:** Experimenting with other advanced machine learning models and techniques, such as deep learning architectures and transfer learning, to capture more complex patterns in the data.
- **Explainability:** Incorporating model explainability methods, such as SHAP values or LIME, to understand the decision-making process of the models and to build trust with stakeholders.
- **Scalability:** Ensuring that the models and techniques used are scalable and efficient for deployment in large-scale, real-world financial systems.

By addressing these areas, we can further enhance the robustness, accuracy, and applicability of fraud detection systems, contributing to more secure and reliable financial transactions.

11 References

1. Patil, S., Nemade, V., & Soni, P. (2018). Predictive Modelling for Credit Card Fraud Detection Using Data Analytics. *Procedia Computer Science, 132*, 385-395.
2. Tekkali, C. G., & Natarajan, K. (2024). Assessing CNN's Performance with Multiple Optimization Functions for Credit Card.
3. Mim, M. A., Majadi, N., & Mazumder, P. (2020). A Soft Voting Ensemble Learning Approach for Credit Card Fraud Detection. *Journal of Artificial Intelligence, 6*(1), 1-21.
4. Peng, C.-Y. J., Lee, K. L., & Ingersoll, G. M. (2002). An Introduction to Logistic Regression Analysis and Reporting. *Journal of Educational Research, 96*(1), 3-14.
5. O'Shea, K., & Nash, R. (2015). An Introduction to Convolutional Neural Networks.
6. Mohammed, A., & Kora, R. (2024). A Comprehensive Review on Ensemble Deep Learning: Opportunities and Challenges. *Journal of Artificial Intelligence, 6*(1), 1-21.