Predicting Credit Card Fraud using PySpark and Machine Learning

Leland Ly, Shawn Oyer, Ahmad Javed

DSCI 632, Drexel University, August 2024

1 Abstract

Credit card fraud detection is a critical concern for e-commerce platforms and financial institutions, given the rise of fraudulent transactions posing as substantial financial and security risks. This project utilizes PySpark and machine learning techniques to develop models for predicting fraudulent transactions in an e-commerce setting. A synthetic dataset was acquired from kaggle containing a variety of features commonly found in transactional data, with additional attributes specifically engineered to support the development and testing of fraud detection algorithms. We employed both traditional and deep learning models, including Logistic Regression (LR) and a Recurrent Neural Network (RNN) model called Gated Recurrent Units (GRU), to address the challenge of class imbalance inherent in fraud detection tasks. Initially, both models achieved high accuracy but struggled with fraud detection due to class imbalance. Predicting fraud detection improved when optimizing strategies were implemented, to include class weighting. The results underscore the importance of model tuning and feature engineering in enhancing the predictive performance of fraud detection systems and the need for continuous refinement in fraud detection methodologies.

2 Introduction

Occurrence of fraudulent transactions has been growing rapidly with the booming development of e-commerce. It costs consumers and financial institutions billions of dollars annually [6]. Credit card fraud is a form of identity theft in which an individual has their credit card information stolen by someone else, in which purchases are made without the owner's knowledge and approval. Credit card fraud can be carried out in a wide variety of ways, both in person and online. While credit card companies have made advances in their attempts to mitigate credit card fraud, it is still a crime that affects a lot of people [1]. The purpose of this project is to be able to classify and predict fraudulent transactions vs non-fraudulent transactions, as well as to determine if there is a specific group that is more susceptible to fraud (i.e. older vs younger people, category of products, etc) through the use of machine learning (ML).

3 Dataset

This synthetic dataset is designed to simulate transaction data from an e-commerce platform with a focus on fraud detection. It contains a variety of features commonly found in transactional data, with additional attributes specifically engineered to support the development and testing of fraud detection algorithms. The data was generated using Python's Faker library and custom logic to simulate realistic transaction patterns and fraudulent scenarios. The dataset is not based on real individuals or transactions and is created for educational and research purposes [5]. The data can be downloaded here: `https://www.kaggle.com/datasets/shriyashjagtap/fraudulent-e-commerce-transactions`

1. Overview of the dataset:
    (a) Data Type: .CSV

(b) Number of Transactions in Version 1: 1,472,952

(c) Number of Transactions in Version 2: 23, 634

(d) Features: 16 (Transaction ID, Customer ID, Transaction Amount, Transaction Date, Payment Method, Product Category, Quantity, Customer Age, Customer Location, Device Used, IP Address, Shipping Address, Billing Address, Is Fraudulent, Account Age Days, and Transaction Hour)

(e) Fraudulent Transactions: 5

2. License:

   (a) Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sub license, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so.

4 Goals

These are the proposed project goals:

1. What demographic of consumer is most likely to charge a fraudulent transaction?
2. What category of products have the highest rate of fraudulent transactions?
3. How much revenue was lost to fraudulent transactions, month over month?
4. What are the most important features in determining fraudulent transactions?
5. Can we build an accurate machine learning model to predict whether a transaction is fraudulent or not?

5 Feature Engineering and Selection:

Before building the machine learning models to be able to predict fraudulent and non-fraudulent transactions, we performed a few feature engineering steps to better prepare the data for model testing. The dataset contained a column called Customer Age, but we thought it would be more beneficial to create age buckets (1-14, 15-24, 25-34, 45-54, 55-64, and 65+) as a new column instead. We then created a function that performs multiple steps to properly clean the data. Within the function, we created a new column called Does Address Match, that returns a 1 if the Shipping Address and Billing Address match and a 0 if not. Next, we used the Transaction Date field to extract the Transaction Day (1-31), Transaction Day of the Week (1-7), Transaction Month (1-4 since the data only goes through April), and Transaction Year (2024). Then, we dropped irrelevant columns (Transaction ID, Customer ID, Customer Location (all fake names due to being a synthetic dataset), IP Address, Transaction Date, Shipping Address, Billing Address, and Customer Age). And lastly, the function casts the integer and float columns to Integer Type and Float Type respectively. Figure 1 shows the new cleaned data frame ready for model building and testing. The data was split into 63% Training, 30% Testing and 7% Validation using a random seed for consistency and comparability. All columns except for the "Is Fraudulent" column were used as the input and the "Is Fraudulent" column was used as the target feature. Since there were categorical and numeric features, we created a function to identify categorical and numeric and returned a list of categorical and numeric columns to be used with the pipeline. Next, we implemented a string indexer and used One Hot

Transaction_Amount	Payment_Method	Product_Category	Quantity	Device_Used	Is_Fraudulent	Account_Age_Days	Transaction_Hour	Age_Group	Does_Address_Match	Transaction_Day	Transaction_DOW	Transaction_Month	Transaction_Year
58.09	bank transfer	electronics	1	tablet	0	30	5	15-24	1	20	3	2	2024
389.96	debit card	electronics	2	desktop	0	72	8	35-44	1	25	1	2	2024
134.19	PayPal	home & garden	2	tablet	0	63	3	15-24	1	18	2	3	2024
226.17	bank transfer	clothing	5	desktop	0	124	20	25-34	1	16	7	3	2024
121.53	bank transfer	clothing	2	tablet	0	158	5	45-54	1	15	2	1	2024
166.41	bank transfer	toys & games	2	tablet	0	38	10	25-34	0	30	3	1	2024
92.88	PayPal	toys & games	2	tablet	0	119	19	1-14	1	4	1	2	2024
318.14	credit card	health & beauty	4	desktop	0	251	13	35-44	1	20	3	2	2024
47.92	bank transfer	home & garden	4	desktop	0	190	19	35-44	1	3	1	3	2024
121.78	bank transfer	health & beauty	4	mobile	0	343	21	35-44	1	16	3	1	2024
633.39	PayPal	toys & games	5	tablet	0	285	22	15-24	0	20	7	1	2024
56.31	PayPal	home & garden	3	tablet	0	259	21	35-44	1	16	6	2	2024
275.87	credit card	home & garden	5	mobile	0	307	11	45-54	0	15	5	2	2024
178.94	debit card	clothing	4	mobile	0	200	15	25-34	1	29	2	1	2024
374.04	bank transfer	home & garden	5	tablet	0	22	13	45-54	1	14	4	2	2024
169.04	debit card	home & garden	1	tablet	0	60	10	45-54	1	4	2	3	2024
254.48	bank transfer	electronics	4	desktop	0	238	8	15-24	1	8	2	1	2024
266.06	debit card	electronics	1	tablet	0	264	14	35-44	1	26	6	1	2024
263.28	debit card	toys & games	3	tablet	0	159	22	25-34	1	7	5	3	2024
475.76	PayPal	clothing	2	mobile	0	143	10	25-34	1	17	4	1	2024

Figure 1: Cleaned Dataset

Encoding for all the categorical columns to be able to use them in the model. We implemented a vector assembler and a Min Max Scaler for all of the numeric columns to ensure normalization, then combined all features into a single vector that we used as the input into the pipeline. After fitting and transforming the pipeline on the training data, we extracted the features and labels to be used for training, testing and validation sets.

6 EDA

6.1 Pyspark Data Aggregation

After the data has been cleaned, we created some Pyspark commands to compute aggregates and averages of our dataset, which achieves some of our initial goals.

To satisfy our first goal of Consumer Demographics, we created the age bins as specified in the section above. We can then run a aggregation mean to find the fraud percentage across the groups.

```
# Group by Age Group

# Filter out negative ages
combined_df = combined_df.filter(col("Customer_Age") >= 0)

# Define age buckets
combined_df = combined_df.withColumn("Age_Group",
    when(col("Customer_Age") <= 14, "1-14") \
    .when((col("Customer_Age") >= 15) & (col("Customer_Age") <= 24), "15-24") \
    .when((col("Customer_Age") >= 25) & (col("Customer_Age") <= 34), "25-34") \
    .when((col("Customer_Age") >= 35) & (col("Customer_Age") <= 44), "35-44") \
    .when((col("Customer_Age") >= 45) & (col("Customer_Age") <= 54), "45-54") \
    .when((col("Customer_Age") >= 55) & (col("Customer_Age") <= 64), "55-64") \
    .otherwise("65+")
)

# Group by 'Age Group' and calculate statistics (e.g., count, average fraud percentage)
age_group_stats = combined_df.groupBy("Age_Group").agg(
    mean('Is_Fraudulent').alias("fraud_percentage")
).orderBy("fraud_percentage", ascending=False)

age_group_stats.show()
```

```
+---------+--------------------+
|Age_Group|    fraud_percentage|
+---------+--------------------+
|      65+| 0.05574912891986063|
|     1-14|0.050743968225739525|
|    15-24| 0.05071250401323751|
|    25-34| 0.05015059857797366|
|    55-64| 0.05004662729250855|
|    35-44| 0.04999178474466204|
|    45-54| 0.04988466257668712|
+---------+--------------------+
```

Figure 2: Fraudulent Percentage by Age Bucket

Its interesting to notice that here, the top two percentages are 1-14 and 65+. This intuitively makes sense since they are the most extreme age groups. We had filtered out the ages less than 0, but its useful to know that when we're given an age that looks like an outlier, then we should be more careful handling the transaction

Another demographic we can briefly categorize is device type. Simply aggregating the unique device types and averaging their fraudulent rates, we can learn more about our transactions. The results are within normal deviations, so we closed this query on device type, Citing that there is no correlation between device used and fraudulent transactions.

```
from pyspark.sql.functions import mean

# Group by 'Device Used' and calculate the percentage of fraudulent transactions
device_fraud_percentage = combined_df.groupBy("Device_Used").agg(
    mean('Is_Fraudulent').alias("fraud_percentage")
).orderBy("fraud_percentage", ascending=False)
device_fraud_percentage.show()
```

```
+-----------+--------------------+
|Device_Used|    fraud_percentage|
+-----------+--------------------+
|     mobile|0.05063017471934404|
|     tablet| 0.0499484633247372|
|    desktop|0.04988466379848587|
+-----------+--------------------+
```

Figure 3: Fraudulent Percentage by Device Type

The last simple feature engineering we performed was on the category of product purchased. Like the above two features, we aggregated the 5 unique categories: Toys and Games, Clothing, Health and Beauty, Home and Garden, Electronics. and queried to see if any one category had more fraudulent transactions than the rest. All of the results landed within .1 percentage points, so we concluded no evidence of association.

```
# Group by 'Category' and calculate the percentage of fraudulent transactions
category_fraud_percentage = combined_df.groupBy("Product_Category").agg(
    mean('Is_Fraudulent').alias("fraud_percentage")
).orderBy("fraud_percentage", ascending=False)
category_fraud_percentage.show()
```

```
+----------------+--------------------+
|Product_Category|    fraud_percentage|
+----------------+--------------------+
|    toys & games|0.05044889548963727|
|        clothing|0.05040541082968731|
| health & beauty|0.05021085090424202|
|    home & garden|0.05020372331157604|
|     electronics|0.04950732676435686|
+----------------+--------------------+
```

Figure 4: Fraudulent Percentage by Product Category

6.2 Pyspark Exploratory Data Analysis

Now that the data has been cleaned and prepped for analysis, more data exploration can occur to satisfy the third and forth goals. The data used encompasses four months of 2024 (January, February, March, April). When looking at the fraudulent transactions in the data, approximately 25,000 transactions occurred in January and March, 23,000 occurred in February, and roughly 2000 transactions occurred during April as shown in figure 5. The low amount of transaction occurring in April is due to the small amount of April transactions as a whole in the data, rather than a drop fraud.

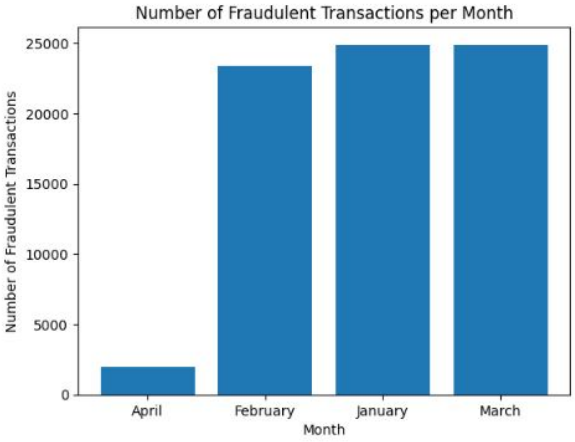

Figure 5: Number of Fraud Transactions per Month

When looking at the how much a fraud transaction per month cost, January through March had an average of five hundred and fifty dollars per fraud transaction, with April having am average of five hundred and thirty three dollars per transaction.

Breaking down the data into a day by day basis, the first two days of the month had the highest number of fraud transactions per day, around 3000, with the remaining days of the month ranging from mid to low 2000 transactions per month. From a value standpoint, the average amount of a fraud transaction per day was the same as the month average (roughly five hundred and fifty dollars per month) with the fifth

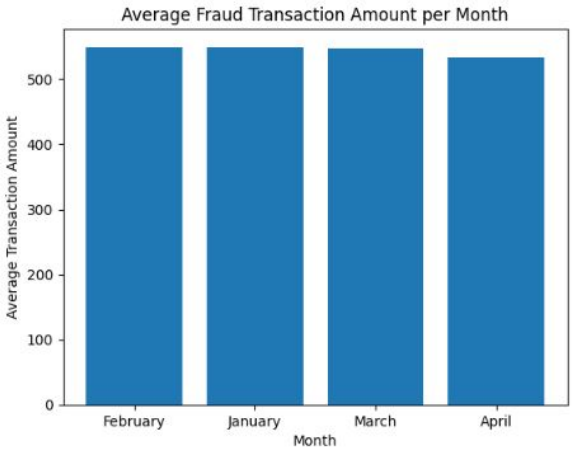

Figure 6: Average Amount of Fraud Transactions per Month

day having the highest (590) and the twenty third day having the lowest (500). The following two figures show the data in more detail.

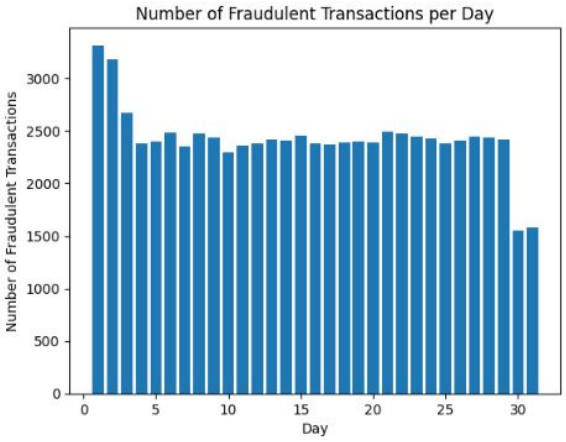

Figure 7: Number of Fraud Transactions per Day

A correlation matrix was created to find the relationship between different factors in the data, specifically focusing on the relationship each factor has with the "Is Fraud" factor. For this matrix all numerical factors were used as well as any categorical factors that can be transformed into numerical data (i.e. If

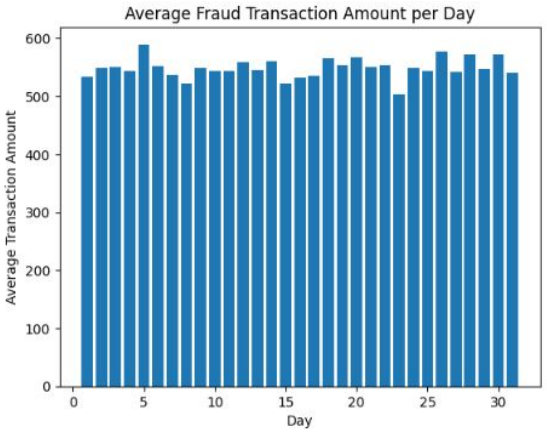

Figure 8: Average Amount of Fraud Transactions per Day

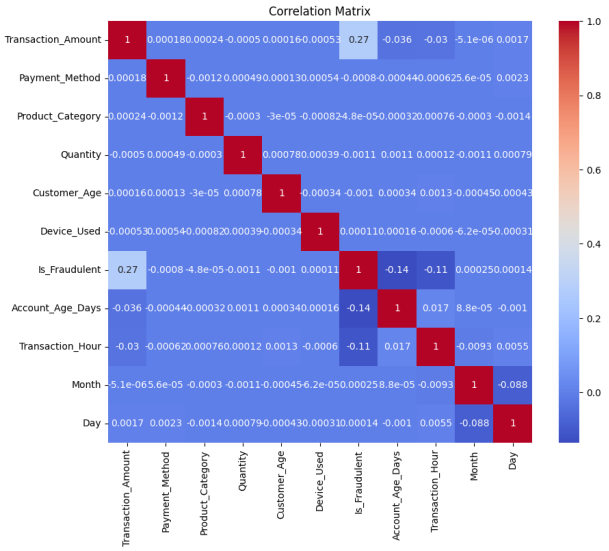

Figure 9: Correlation Matrix Showing Factors Related to Fraud Detection

a factor had 5 categories, each category was given a number 1 through 5). The three categorical factors used were Payment Used, Device Used, and Product Category with their transformation being showed in the following figure.

Once all the factors were set, using Pyspark, a correlation matrix was created where all the relationships were displayed along with a heat map to show how strongly related each factor is with each other. As apparent in the correlation matrix show in figure 6, the transaction amount had the strongest relationship with fraud detection at 0.27. This relationship was the strongest of the ones tested in this matrix, making transaction amount the most prominent feature when determining fraudulent transactions.

This correlation between transaction amount and fraud detection is also in line with fraud detection measures used by certain banks and credit card companies today. Some companies will alert their customers that a fraud transaction may have occurred on their account if a transaction exceed a certain threshold set by either the user or the institution itself (i.e. if a transaction exceeds five hundred dollars, a credit card user may receive a text alerting them of this transaction and allowing them to approve or cancel the transaction).

7 ML Models:

We used two models in this project, a more traditional model in Logistic Regression (LR) and a Deep Learning (DL) model using a Recurrent Neural Network (RNN) called Gated Recurrent Unit (GRU). The LR model was used because it is known to be extremely efficient of detecting frauds based on its ability to isolate the data that belong to different binary classes [2].

DL is a subset of machine learning techniques that teaches computers to perform tasks which are natural to the human. A computer model learns to perform classification tasks directly from image, text or sound where it builds features automatically based on training data and provides state of the art results in various classification and prediction tasks [3]. The GRU technique was introduced in 2014, in the form of a new neural network model called RNN Encoder-Decoder which consists of two recurrent neural networks to make each unit recurrent, in order to adaptively capture the dependencies of different timescales [4]. GRU was used in this project

```
df_fraud = fraud_newdf.withColumn("Payment_Method",
                                  when(col("Payment_Method") == "debit card", 1)
                                  .when(col("Payment_Method") == "bank transfer", 2)
                                  .when(col("Payment_Method") == "PayPal", 3)
                                  .when(col("Payment_Method") == "credit card", 4)
                                  .otherwise(0))

df_fraud = df_fraud.withColumn("Device_Used",
                                  when(col("Device_Used") == "desktop", 1)
                                  .when(col("Device_Used") == "mobile", 2)
                                  .when(col("Device_Used") == "tablet", 3)
                                  .otherwise(0))

df_fraud = df_fraud.withColumn("Product_Category",
                                  when(col("Product_Category") == "health & beauty", 1)
                                  .when(col("Product_Category") == "toys & games", 2)
                                  .when(col("Product_Category") == "electronics", 3)
                                  .when(col("Product_Category") == "clothing", 4)
                                  .when(col("Product_Category") == "home & garden", 5)
                                  .otherwise(0))
df_fraud.show(10)
#
```

```
+------------------+--------------+----------------+--------+--------------+-------------+---------------+------------------+----------------+-----+---+
|Transaction_Amount|Payment_Method|Product_Category|Quantity|Customer_Age|Device_Used|Is_Fraudulent|Account_Age_Days|Transaction_Hour|Month|Day|
+------------------+--------------+----------------+--------+--------------+-------------+---------------+------------------+----------------+-----+---+
|            401.83|             3|               5|       3|            27|            3|              0|               354|              16|    3|  9|
|            768.11|             2|               1|       4|            51|            2|              0|               338|              22|    1| 20|
|            166.75|             1|               1|       4|            50|            1|              0|               308|               9|    2| 15|
|            159.26|             1|               2|       3|            39|            3|              0|               211|              17|    3| 20|
|            380.15|             3|               1|       1|            27|            2|              1|               309|              16|    1|  9|
|            318.69|             1|               2|       3|            24|            2|              0|               291|              18|    2| 12|
|             14.8 |             4|               1|       2|            26|            3|              0|               363|               7|    3| 13|
|             64.94|             4|               4|       2|            37|            2|              0|               334|              16|    3| 24|
|             48.05|             2|               4|       4|            23|            1|              0|               289|              19|    2| 16|
|            252.69|             1|               3|       3|            42|            1|              0|               262|              15|    1| 27|
+------------------+--------------+----------------+--------+--------------+-------------+---------------+------------------+----------------+-----+---+
only showing top 10 rows
```

Figure 10: Transformation of Categorical Data to Numerical Data

due to having strong impacts on the learning process during gradient back-propagation phase of basic RNN, which leads to situations called vanishing or exploding gradients. The GRU model modifies 3 gates (input/forget/output) of LSTM into 2 gates (update/reset), and merges the unit/output into one state, which reduces the matrix multiplication to some extent making it a more efficient model [6]. Figure 11 displays the GRU framework.

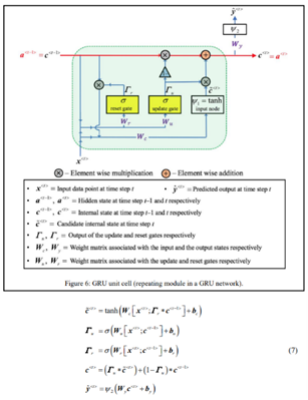

Figure 11: GRU Framework [7]

Even though there was a heavy class imbalance for fraudulent vs non-fraudulent transactions, we wanted to see how both the LR and GRU would perform without optimizing the parameters explicitly first. For the LR model, we trained the model on training data, we made predictions and evaluated the predictions on the test set using accuracy, precision, recall, and F1 scores. For the GRU model, we created a function to convert every column from a PySpark Dataframe to a TensorFlow Dataset to be able to input it into the GRU. We then defined the model by having dropouts to ensure normalization and used the sigmoid activation function (figure 12 below). We then compiled the model to use Adam as the optimizer, binary cross entropy as the loss since it is a binary classification, and accuracy, precision, recall, and F1 metrics.

```
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
reshape_1 (Reshape)	(None, 1, 23)	0
gru_2 (GRU)	(None, 1, 32)	5,472
dropout_2 (Dropout)	(None, 1, 32)	0
gru_3 (GRU)	(None, 16)	2,400
dropout_3 (Dropout)	(None, 16)	0
dense_1 (Dense)	(None, 1)	17

```
Total params: 7,889 (30.82 KB)
Trainable params: 7,889 (30.82 KB)
Non-trainable params: 0 (0.00 B)
```

Figure 12: GRU Model Params

To address the class imbalance, we would have liked to have used Synthetic Minority Over-Sampling Technique (SMOTE) since it has a great track record of improving prediction scores, but since SMOTE requires pandas to function, we instead used class weights that also works pretty well. We computed class weights using a balanced function and created a dictionary of all of the class weights that was fed to the model as an additional parameter.

8 Results

LR Results: In reviewing the LR results, the LR_1 (not optimized to address class imbalance) had an overall accuracy of 95% with all of the weighted metrics in the 90s for both testing and validation sets. However, in taking a closer look, the model did an incredible job on predicting non-fraudulent transactions (96% recall) but had only a 2% recall in ac-

	Model	Logistic Regression_1	Logistic Regression_2	RNN GRU_1	RNN GRU_2
	Data Source	Fraudulent E-Commerce Transactions	Fraudulent E-Commerce Transactions	Fraudulent E-Commerce Transactions	Fraudulent E-Commerce Transactions
	Optimized?	No	Used hyperparameters	No	Optimized to Address Class Imbalance
	#Train	938387	938387	938387	938387
	#Test	445899	445899	445899	445899
	#Validation	103976	103976	103976	103976
	Classification	Fraudulent, Non-fraudulent	Fraudulent, Non-fraudulent	Fraudulent, Non-fraudulent	Fraudulent, Non-fraudulent
	Accuracy	95.00%	68%	94.40%	58.80%
	Loss	N/A	N/A	0.17	0.89
Testing	Precision	95%	94%	91%	91%
	Recall	96%	69%	94%	59%
	F1	93%	78%	93%	70%
	Fradulent Predicted	2%	67%	1%	42%
	Non-Fraudulent Predicted	99%	69%	99%	60%

Figure 13: Results Table

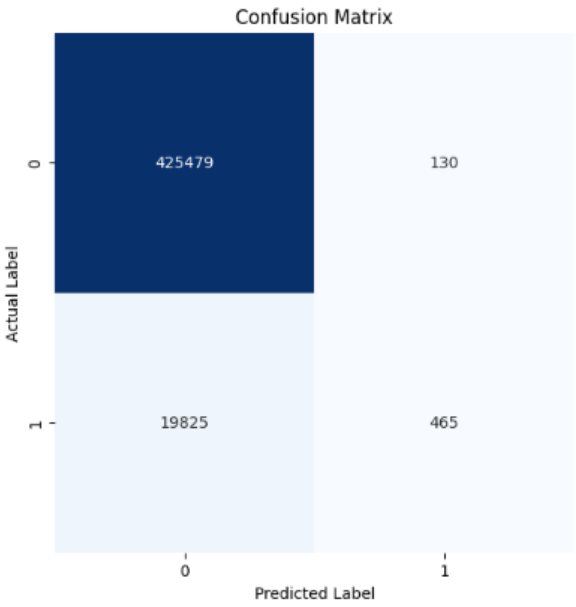

Figure 14: Confusion Matrix for LR_1

curately predicting fraudulent transactions (see figure 14). This is where the class imbalance really comes into play because it is clear that there were not enough fraudulent transactions in the training dataset.

One of our goals was to determine which features had most importance in predicting fraudulent vs non fraudulent transactions, and it was determined that Quantity, Transaction Hour and Transaction Day were the highest indicators of fraudulent activity (see figure 15). It's interesting because the correlation matrix noted the transaction hour as a potential correlated factor but didn't note quantity or transaction day as a potential indicator, which is why it's important to determine the feature importance after running the model to determine what is really driving the predictions.

GRU Results: In reviewing the GRU results, the GRU_1 (not optimized to address class imbalance) had an overall accuracy of 94% with a loss of 0.17 with all of the weighted metrics in the 90s for both testing and validation sets. However, in taking a closer look, the model did an incredible job on predicting non-fraudulent transactions (99% recall) but had only a 1% recall in accurately predicting fraudulent transactions (see figure 16). Similarly to the LR model, the class imbalance needs to be addressed.

8.1 Results after optimizing models for class imbalance

In reviewing the LR results, the LR_2 (optimized to address class imbalance) had an overall accuracy of 68% with precision having a weighted score of 94%, recall having a weighted score of 69% and F1 having a weighted score of 78%. Even though the model had overall worse weighted metrics than LR_1, the model outperformed the first one significantly in accurately predicting fraudulent transactions with 67% correctly predicted. The non-fraudulent transactions were less successful, however, with 69% accurately predicted (see figure 17). It's obvious there is a good trade-off between the models, and we believe with further fine-tuning to address the class imbalance a more successful model can be employed.

In reviewing the GRU results, the GRU_2 (optimized to address class imbalance) had an overall accuracy of 59% with a loss of 0.89 with precision having a weighted score of 92%, recall having a weighted score of 59% and F1 having a weighted score of 70%. The GRU model had overall worse weighted met-

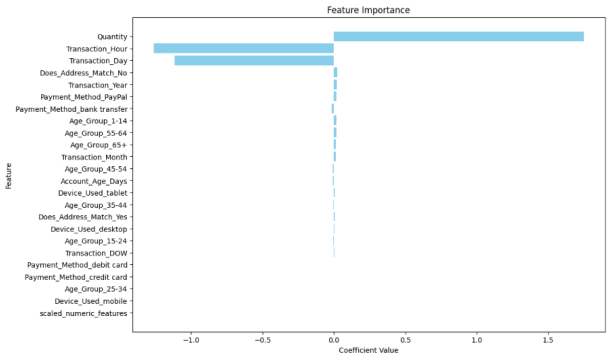

Figure 15: Feature Importance

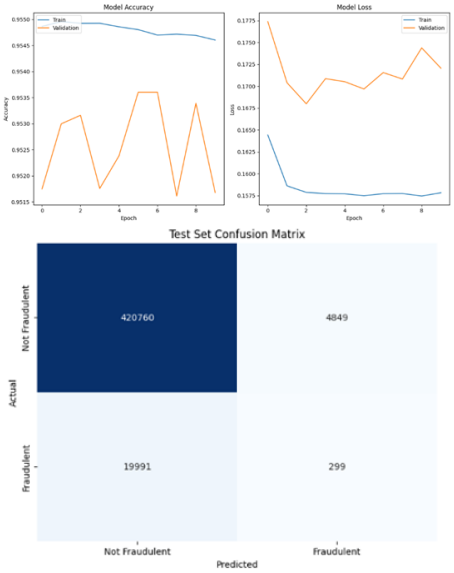

Figure 16: GRU Run 1

rics than GRU_1, and similarly to the LR model, the GRU model outperformed the first one significantly in accurately predicting fraudulent transactions with 42% correctly predicted. The non-fraudulent transactions were less successful, however, with 60% accurately predicted (see figure 18). It's obvious there is a good trade-off between the models, and we believe with further fine-tuning to address the class imbalance a more successful model can be employed.

9 Conclusion

This project aimed to tackle the complex problem of credit card fraud detection using a combination of machine learning models and feature engineering techniques. Through the application of LR and GRU on a synthetic e-commerce transactions dataset, we explored various aspects of fraud detection, from feature importance to model performance.

Our initial models revealed that while LR achieved high accuracy overall, it was significantly impaired in detecting fraudulent transactions due to class imbalance. Similarly, the GRU model demonstrated solid performance in predicting non-fraudulent transactions but struggled with fraud detection. These findings emphasize the impact of class imbalance on model performance and highlight the need for effective optimization strategies. After optimizing the models to address the class imbalance, we observed improvements in the detection of fraudulent transactions. The LR model, when optimized, showed a better balance between precision and recall for fraudulent transactions, while the GRU model's performance remained lower compared to the optimized Logistic Regression but still demonstrated valuable insights into handling imbalanced classes.

Our analysis also provided insights into feature importance, with Quantity, Transaction Hour, and Transaction Day emerging as significant predictors of fraudulent activity. These features, along with advanced data engineering and modeling techniques, contribute to a better understanding of fraud patterns. The results highlight that while traditional and deep learning models can be effective, addressing class imbalance and continuous model optimization are crucial for improving fraud detection capabilities. Future work should focus on further fine-tuning these models, exploring additional features, and leveraging more sophisticated techniques to enhance detection accuracy and reduce financial losses from fraudulent transactions.

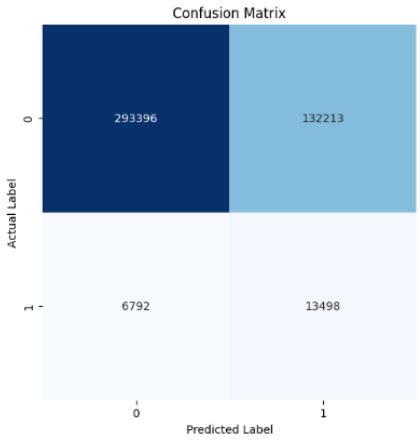

Figure 17: Confusion Matrix for LR_2

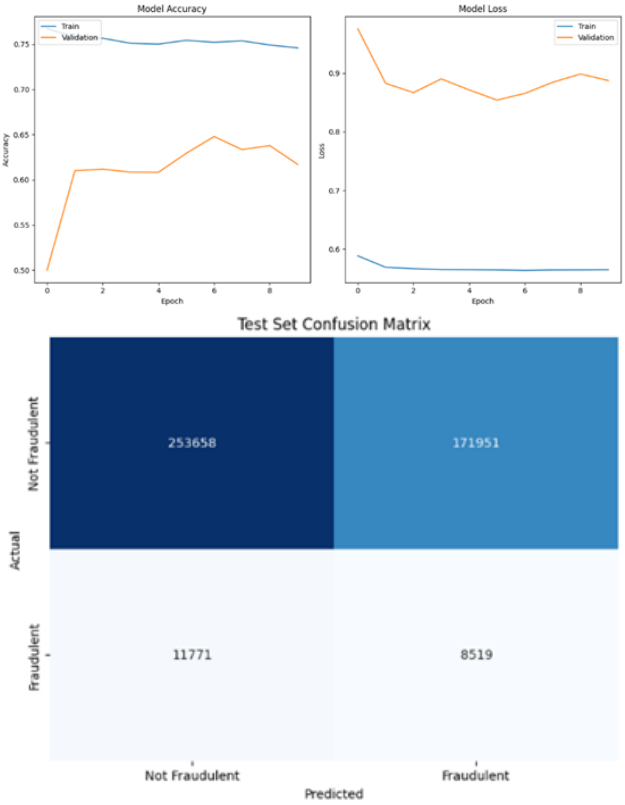

Figure 18: GRU Run 2

References

[1] Bandyopadhyay, Samir Kumar. "(PDF) Detection of Fraud Transactions Using Recurrent Neural Network during COVID-19." ResearchGate, 2020, Retrieved from url Accessed 16 August 2024.

[2] Hala, Alenzi, and Aljehane Nojood. "Fraud Detection in Credit Cards using Logistic Regression." The Science and Information (SAI) Organization, 2020. Retrieved from url Accessed 22 August 2024.

[3] Hassan, Najadat. "(PDF) Credit Card Fraud Detection Based on Machine and Deep Learning." ResearchGate, 26 May 2020. Retrieved from url Accessed 16 August 2024.

[4] Imane, Sadgali, et al. "Bidirectional gated recurrent unit for improving classification in credit card fraud detection." Indonesian Journal of Electrical Engineering and Computer Science, vol. 21, no. 3, 2021, pp. 1704-1712. Indonesian Journal of Electrical Engineering and Computer Science. Retrieved from url Accessed 16 August 2024.

[5] Jagtap, Shriyash. "Fraudulent E-Commerce Transactions." Kaggle, 2024. Retrieved from url Accessed 21 August 2024

[6] Xurui, Li, et al. "Paper Title (use style: paper title)." arXiv, 2018. Retrieved from url Accessed 16 August 2024

[7] Zargar, Sakib A. "Introduction to Sequence Learning Models: RNN, LSTM, GRU Sakib Ashraf Zargar Department of Mechanical and Aerospace Engineering." ResearchGate, 2021. Retrieved from url Accessed 16 August 2024.