IST707 Applied Machine Learning

CREDIT CARD FRAUD DETECTION AND DATA WRANGLING FOR DUPLICATE TRANSACTIONS

BASTIN BAJIYO JOB
KANISHK GUPTA
CHINMAY ASHOK MAGANUR

Introduction:

In our research we try to analyse the credit card transactions data, identify the structure and salient features, and use the knowledge to perform data wrangling, followed by building a model to detect fraudulent transactions.

Fraud is a problem for any bank. Fraud can take many forms, whether it is someone stealing a single credit card, to large batches of stolen credit card numbers being used on the web, or even a mass compromise of credit card numbers stolen from a merchant via tools like credit card skimming devices.

We try to build a predictive model to determine whether a given transaction will be fraudulent or not. We also perform extensive data analysis and attempt to identify multi-swipe transactions using data wrangling.

Exploratory Data Analysis (EDA):

The provided dataset is in the form of a .txt file, with records in the form of JSON objects. The file was read into a dataframe and exploratory data analysis was conducted using pandas library.

- Structure of the dataset:

  Once the data was loaded into a dataframe, examining its shape will give us the number of records and fields in the dataset.
  - 786363 records and 29 fields
  - Target Variable for Credit Card Fraud Classification: '**isFraud**'

- Description of variables/fields:

  account Number: The account number associated with the credit card used for the transaction.
  customerId: The customer ID associated with the account.
  creditLimit: The credit limit for the account.
  availableMoney: The amount of money available for the account after the transaction.
  transactionDateTime: The date and time when the transaction occurred.
  transactionAmount: The amount of money involved in the transaction.
  merchantName: The name of the merchant where the transaction took place.
  acqCountry: The country where the transaction was acquired (i.e., the country where the card was used).
  merchantCountryCode: The country where the merchant is located.
  posEntryMode: The method used to enter the payment information.
  posConditionCode: The condition code of the point of sale (POS) terminal used for the transaction.
  merchantCategoryCode: The category of the merchant where the transaction occurred.
  currentExpDate: The expiration date of the credit card.
  accountOpenDate: The date when the account was opened.
  dateOfLastAddressChange: The date when the address associated with the account was last changed.
  cardCVV: The card verification value (CVV) associated with the credit card.
  enteredCVV: The CVV entered during the transaction.
  cardLast4Digits: The last four digits of the credit card number used for the transaction.
  transactionType: The type of transaction (e.g., purchase, refund, withdrawal, etc.).
  echoBuffer: Reserved for future use.
  currentBalance: The current balance of the account after the transaction.
  merchantCity: The city where the merchant is located.
  merchantState: The state where the merchant is located.
  merchantZip: The zip code where the merchant is located.
  cardPresent: A boolean value indicating whether the physical card was present during the transaction.

posOnPremises: Reserved for future use.
recurringAuthInd: Reserved for future use.
expirationDateKeyInMatch: A boolean value indicating whether the expiration date of the credit card matched the date entered during the transaction.
isFraud: A boolean value indicating whether the transaction is fraudulent.

- Analysing the type of variable – categorical or numerical

Categorical variables	Numerical variables	Boolean Variables
'accountNumber' 'customerId' 'transactionDateTime' 'merchantName' 'acqCountry' 'merchantCountryCode' 'posEntryMode' 'posConditionCode' 'merchantCategoryCode' 'currentExpDate' 'accountOpenDate' 'dateOfLastAddressChange' 'cardCVV' 'enteredCVV' 'cardLast4Digits' 'transactionType' 'echoBuffer' 'merchantCity' 'merchantState' 'merchantZip' 'posOnPremises' 'recurringAuthInd'	'creditLimit' 'availableMoney' 'transactionAmount' 'currentBalance'	‘cardPresent' 'expirationDateKeyInMach' 'isFraud'

- Checking for Null/Missing values :
Null values associated with each field were counted and displayed below. Some variables have entirely null values, and some have a large number of null values. Fields with entirely null values will be dropped since they are practically useless for data analysis.

```
accountNumber                  0
customerId                     0
creditLimit                    0
availableMoney                 0
transactionDateTime            0
transactionAmount              0
merchantName                   0
acqCountry                  4562
merchantCountryCode          724
posEntryMode                4054
posConditionCode             409
merchantCategoryCode           0
currentExpDate                 0
accountOpenDate                0
dateOfLastAddressChange        0
cardCVV                        0
enteredCVV                     0
cardLast4Digits                0
transactionType              698
echoBuffer                786363
currentBalance                 0
merchantCity              786363
merchantState             786363
merchantZip               786363
cardPresent                    0
posOnPremises             786363
recurringAuthInd          786363
expirationDateKeyInMatch       0
isFraud                        0
dtype: int64
```

- Descriptive Statistics of Numerical Variables:

  The basic descriptive statistics of applicable fields, like minimum, maximum, mean etc were calculated and displayed.

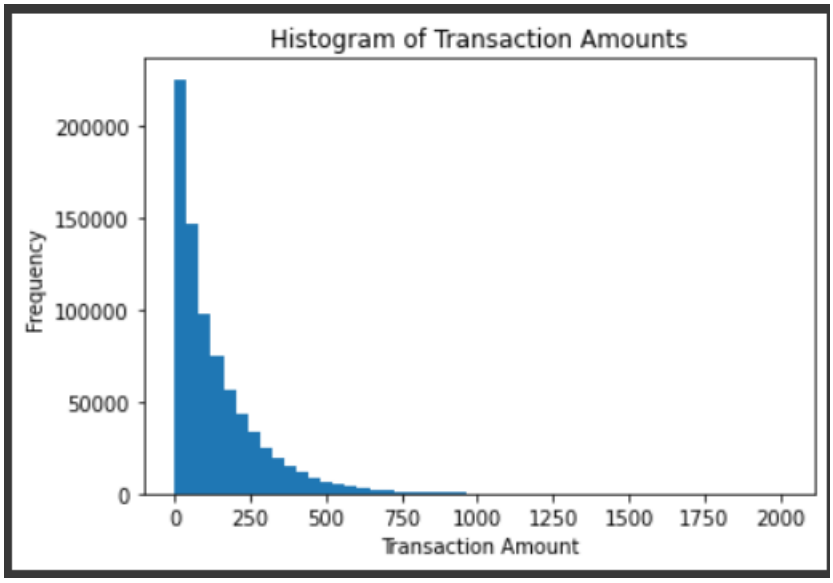
	creditLimit	availableMoney	transactionAmount	currentBalance
count	786363.000000	786363.000000	786363.000000	786363.000000
mean	10759.464459	6250.725369	136.985791	4508.739089
std	11636.174890	8880.783989	147.725569	6457.442068
min	250.000000	-1005.630000	0.000000	0.000000
25%	5000.000000	1077.420000	33.650000	689.910000
50%	7500.000000	3184.860000	87.900000	2451.760000
75%	15000.000000	7500.000000	191.480000	5291.095000
max	50000.000000	50000.000000	2011.540000	47498.810000

- Unique Values associated with applicable categorical variables:
  Unique values of categories variables with a countable number of unique values are displayed below.

Field Name	Unique Values
acqCountry	'US' ' 'CAN' 'MEX' 'PR'
merchantCountryCode	'US' 'CAN' ' 'PR' 'MEX'
posEntryMode	'02' '09' '05' '80' '90'
posConditionCode	'01' '08' '99'
merchantCategoryCode	'rideshare' 'entertainment' 'mobileapps' 'fastfood' 'food_delivery' 'auto' 'online_retail' 'gym' 'health' 'personal care' 'food' 'fuel' 'online_subscriptions' 'online_gifts' 'hotels' 'airline' 'furniture' 'subscriptions' 'cable/phone'

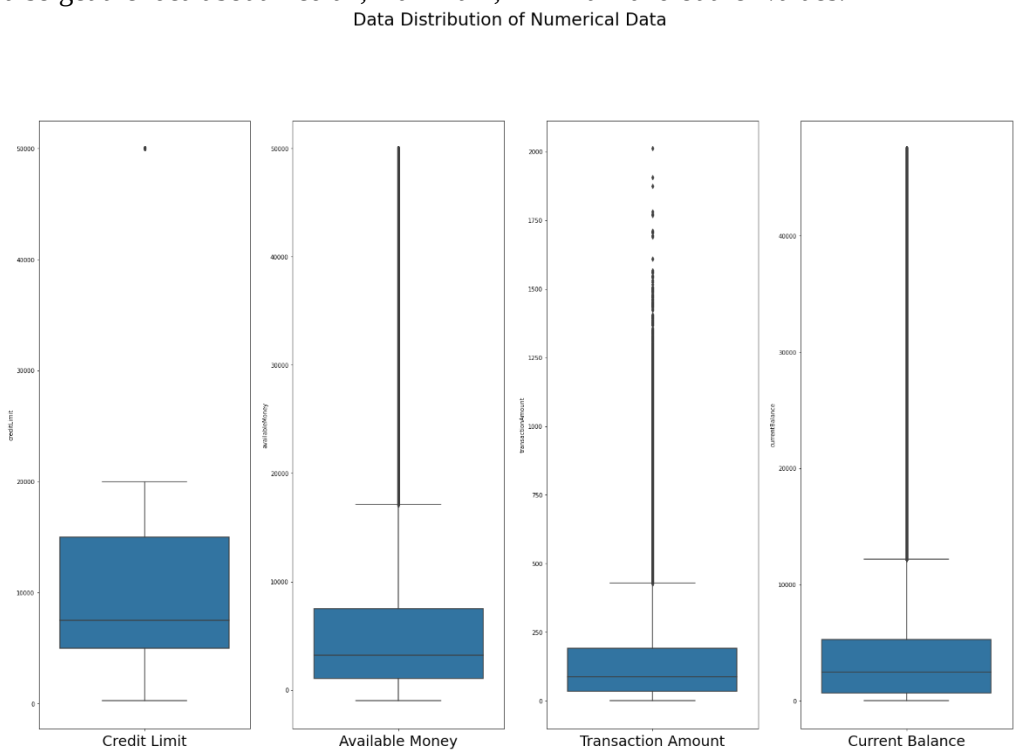
EDA – Visualizations:

- Histogram for Transaction Amounts



  - Distribution of Transaction Amounts: The histogram of transaction amounts shows a long-tailed distribution, where most transactions are small in value but there are a few very large transactions. This is a common pattern in financial data, where a small number of high-value transactions can skew the overall distribution.
  - The distribution of transaction amounts may be useful in identifying fraudulent transactions. For example, very high-value transactions may be more likely to be fraudulent.

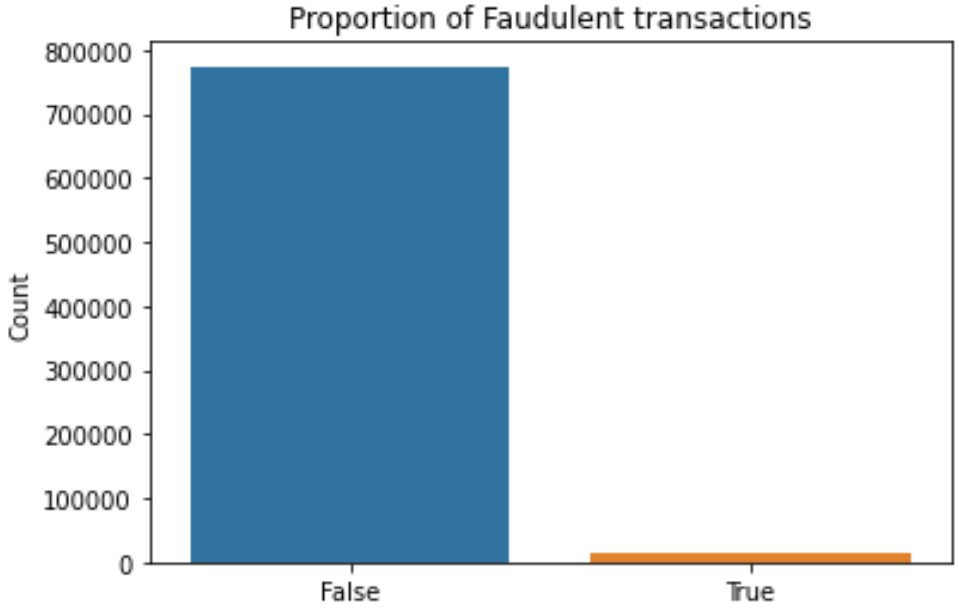
- Analysing numerical variables using boxplots:
  The distribution of the numerical variables is best explained by box plots. Along with the range of values, we also get the idea about median, maximum, minimum and outlier values.



- All four distributions seems to be right skewed, implying that majority of values are on the lower side (<10000).
- Rarely high credit limits (~50000) can be observed, implying that such high credit limits are achieved by very few people.

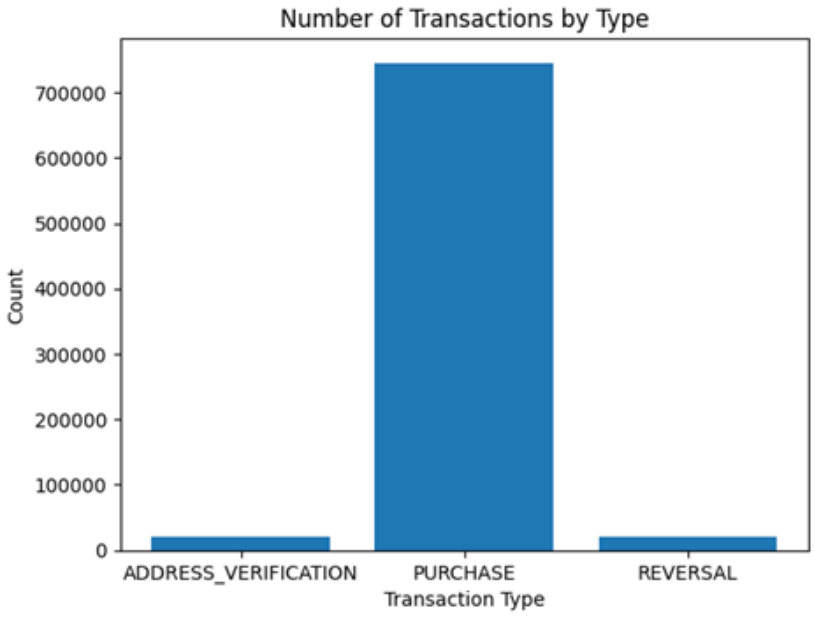
- Transaction amounts and current balance are predominantly low, denoting a positive correlation. This is intuitive, since a low current balance allows low transaction amounts.

- Proportion of Fraudulent transactions:
  We now look at how many transactions are fraudulent relative to normal transactions.



  - As we observe a very small proportion of transactions are fraudulent.
  - From a business point of view, it definitely is a good thing to have as less fraudulent transactions as possible.
  - For a data scientist, the dataset is severely imbalanced!

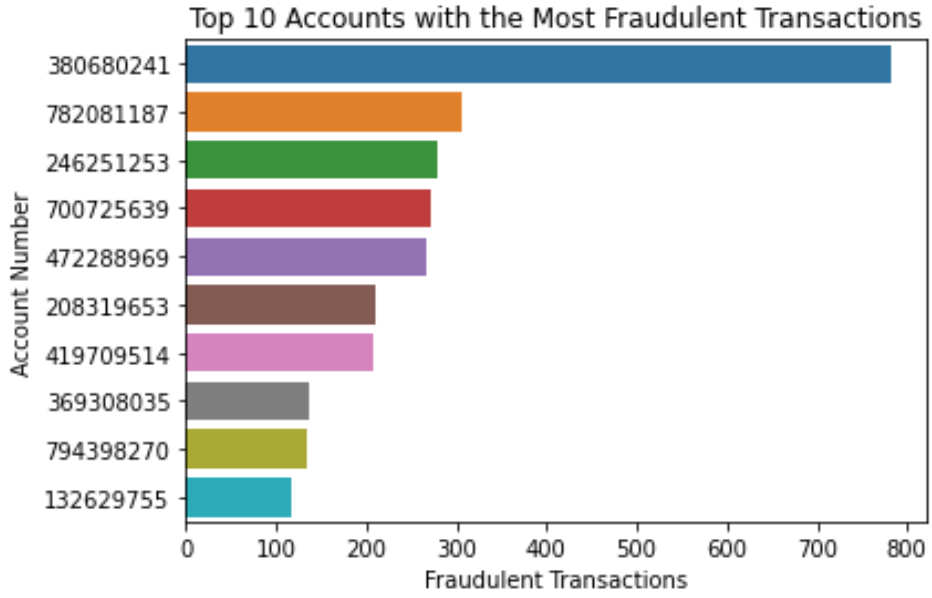
- Proportion of different transaction types:



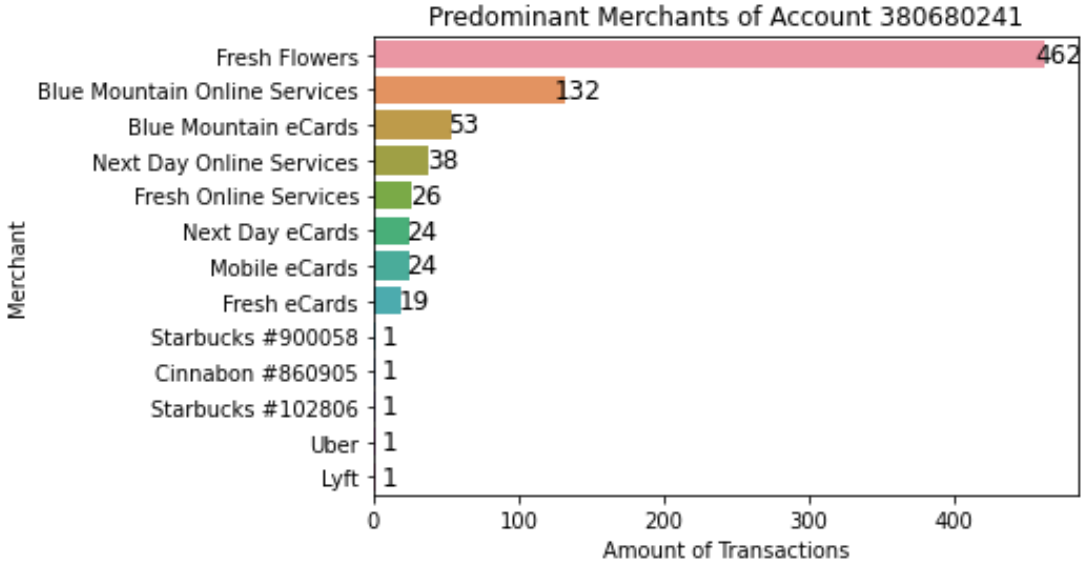

```
transactionType
ADDRESS_VERIFICATION     20169
PURCHASE                745193
REVERSAL                 20303
dtype: int64
```

  - We can observe that huge majority of transactions are purchases with very few reversals and address verifications.

- Taking a closer look at fraudulent transactions:
  Now we focus on the fraudulent transactions with respect to account numbers and merchant types.



- The account number 380680241 shows suspiciously high number of fraudulent transactions.
- Lets take a look at the spending habits of this account number.



- This account number does a lot of Fresh Flower shopping – again raising suspicion as its not common for people to regularly buy flowers, unless they have splendid dating life!
- Overall, this particular account number shows abnormal activity and unusual transactions. There is a possibility of a scam happening through this particular merchant, which this account holder may be a victim.

Data Wrangling (Reversal and Multi-Swipes)

To find out reversed and multi-swipe transactions, the following steps were followed sequentially:

- Initialize an empty list called "multi_swipes" to store the indices of the transactions that are flagged as potential multiple swipes.
- Loop through the rows of the "data" DataFrame using the range function, starting from the second row (i.e., index 1) up to the last row (i.e., len(data) - 1).
- Check if the current transaction (at index "i") satisfies all of the following conditions:
- The transaction amount, account number, merchant name, merchant category code, and POS entry mode are the same as the previous transaction (at index "i-1").
- The current transaction type is "PURCHASE" and the previous transaction type is also "PURCHASE".
- The time difference between the current transaction datetime and the previous transaction datetime is less than 60 seconds.
- If all of these conditions are satisfied, add the indices of both transactions (i-1 and i) to the "multi_swipes" list.
- The transaction amount, account number, merchant name, merchant category code, and POS entry mode are the same as the previous transaction (at index "i-1").
- The current transaction type is "REVERSAL" and the previous transaction type is "PURCHASE".
- The transaction datetime of the current transaction is within 1 second of the transaction datetime of the previous transaction.
- If all of these conditions are satisfied, add the indices of both transactions (i-1 and i) to the "reversals" list.

Results:

Number of reversed transactions	25
Total amount of reversed transactions	$5654.74
Number of multi-swipe transactions	332
Total amount of multi-swipe transactions	$92100.98

Inferences and Insights:

- Based on the identified number of reversed and multi-swipe transactions, we can infer that there might be issues with the payment processing system or with certain vendors.
- The presence of reversed transactions indicates that some purchases might have been made in error, or there might be some fraudulent activity occurring.
- The occurrence of multi-swipe transactions suggests that some vendors might not have robust payment processing systems in place, leading to accidental multiple charges to customers' cards.
- Both of these issues can have negative consequences for customers and businesses, and thus should be carefully investigated and resolved.

Predictive Modelling for Credit Card Fraud

How does credit card fraud happen?

- Lost/Stolen card
- Card details overseen by another person
- Hacked bank details

Main challenges involved in credit card fraud detection are:

- Enormous size of Data: the model build must be fast enough to respond to the scam in time.
- Imbalanced Data: most of the transactions are not fraudulent which makes it really hard for detecting the fraudulent ones.
- Data availability as the data is mostly private.
- Misclassified Data can be another major issue, as not every fraudulent transaction is caught and reported.

How to find a way around?

- The model used must be **simple and fast** enough to detect the anomaly and classify it as a fraudulent transaction as quickly as possible.
- Use AUPRC metric to evaluate performance
- Since a synthetic dataset is readily provided, the issue of availability is solved beforehand.

Plan of Action:

- **Random Forest** and **Logistic Regression** algorithms offers the required amount of complexity, while keeping the algorithm simple and transparent without compromising on speed. Ensemble methods like XGBoost will also be tested along with a simpler decision tree, for comparison.
- The data will be pre-processed, omitting unwanted columns, adding new derived features, normalisations and encodings.
- The model will be trained, tested and evaluation metrics examined.

Data Pre-Processing:

The following are the steps I followed to pre-process the data :

- Remove unwanted fields : The dataset had some fields which was entirely NAN. In addition to that some variables are irrelevant for model training. The following fields are omitted : 'echoBuffer', 'cardLast4Digits','merchantName','accountOpenDate','transactionDateTime','currentExpDate','customerId','dateOfLastAddressChange','accountNumber','enteredCVV','cardCVV'.
- Adding derived features : In case of a brute-force attempt to crack the card-CVV, incorrect CVV may be entered initially. This is an indicator of attempted fraud. So, I am creating a new variable: matchingCVV which denotes whether the real CVV and entered CVV are same.
- Boolean variables are converted to binary integer (0:False, 1:True) values.
- Categorical values are One-Hot encoded.
- Numerical values, along with the other converted values are normalised using standard scaler.
- The data set is split into features and labels : labels being the isFraud column.
- The dataset is split into 80% for training and 20% for testing.
- Random forest and Logistic regression models are trained on the training data and tested on the testing data.
- Accuracy, Precision, Recall, F1-score are examined, and results are visualised using a confusion matrix.

Feature Selection: The following features are selected for training.

creditLimit: Credit Limit is the most important predictor, since a higher credit limit invites a higher fraudulent transaction.
availableMoney: Just like credit limit, higher available money makes room for more fraudulent transactions.
transactionAmount: Anomaly in transaction amounts can be used to spot usual transactions.
acqCountry: There might be geographical correlation with fraudulent transactions, since some regions maybe prone to higher rates of financial fraud.
merchantCountryCode: Same reason as acqCountry, but for the merchant side.
posEntryMode: The method used to enter payment information may have associated vulnerabilities.

posConditionCode: Certain patterns in Condition codes may show anomaly when a fraud transaction occurs.
merchantCategoryCode: Is there certain type of merchants associated more with fraudulent transactions ?
transactionType: A particular pattern of transaction types maybe more prone to fraud.
currentBalance: Same reason as credit limit and available money.
cardPresent: Does presence of card affect the possibility of fraud ?
expirationDateKeyInMatch: Similar to matchingCVV, unmatched expiration dates may hint attempt of fraud.
matchingCVV: Incorrect CVV entered repeatedly may be a hint to a brute force attack.

AUPRC Metric:
We have already seen that the dataset is highly imbalanced and the positive class is rare. So we use an appropriate metric that can handle this kind of data and provide appropriate comparison of performance.

The AUPRC (Area Under the Precision-Recall Curve) is a performance metric used to evaluate the quality of binary classification models. It is particularly useful when the positive class is rare, i.e., when the proportion of positive instances in the dataset is low.

- The AUPRC score measures the trade-off between precision (the fraction of correctly classified positive instances among all instances classified as positive) and recall (the fraction of correctly classified positive instances among all positive instances). The AUPRC is calculated by integrating the precision-recall curve, which is generated by plotting precision as a function of recall for different classification thresholds.

- The main advantages of AUPRC is that it provides a comprehensive evaluation of a binary classification model that is robust to class imbalance. Unlike other evaluation metrics like accuracy, which can be misleading when the class distribution is skewed, AUPRC takes into account the trade-off between precision and recall, and it reflects the ability of a model to correctly identify the rare positive instances.

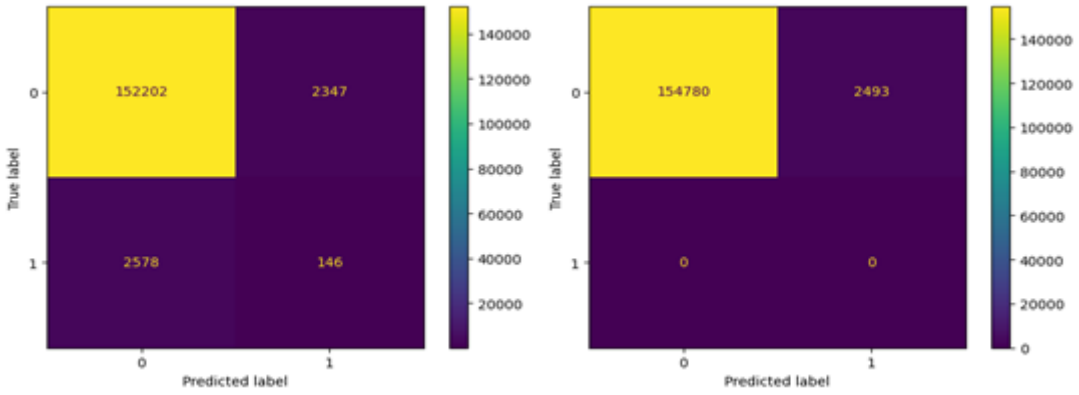
- One of the main drawbacks is that it only evaluates the quality of a model at a single operating point, which is the threshold used to classify instances as positive or negative. Therefore, AUPRC may not fully capture the performance of a model across different decision thresholds. Additionally, AUPRC can be sensitive to class distribution and may not be appropriate for datasets where the proportion of positive instances is extremely low or high.

- In conclusion, AUPRC is a useful metric for evaluating binary classification models, especially when the positive class is rare. However, it should be used in conjunction with other evaluation metrics, such as accuracy or F1-score, to obtain a comprehensive understanding of the model's performance.

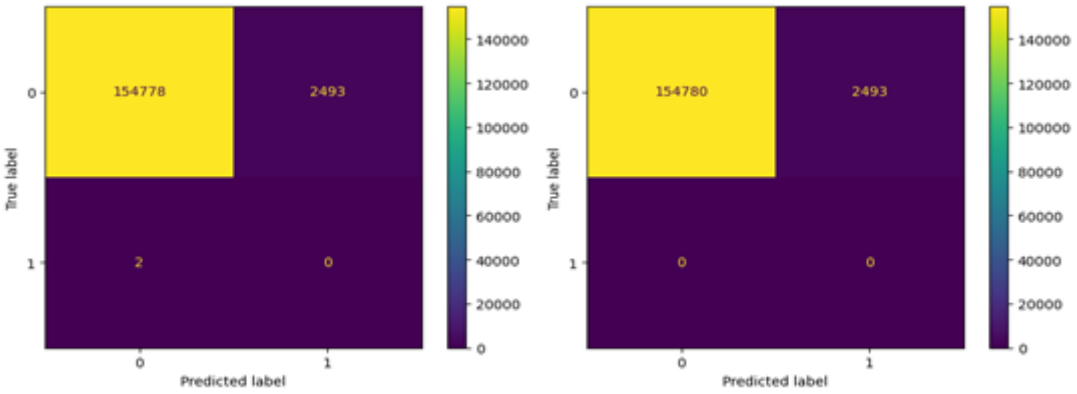
Results:

We trained and tested the performance of 4 models – decision tree, random forest, logistic regression, and XG-Boost. Metrics like accuracy, precision, recall an F1 score were compared, along with AUPRC score. The resulting comparison table and confusion matrices are given below.

Confusion Matrices :



Decision Tree **Random Forest**

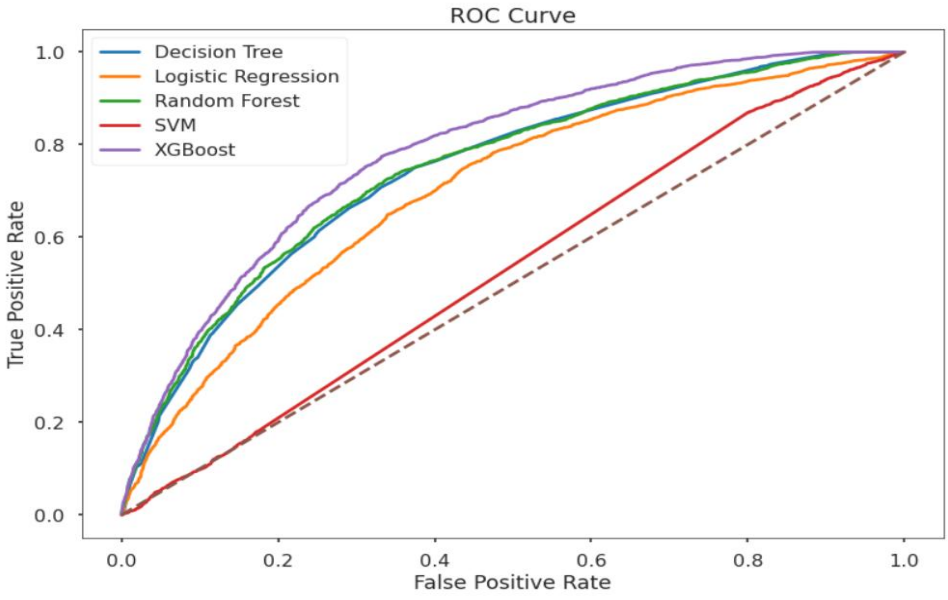


Logistic Regression **XG-Boost**

Comparison of metrics:

Model	Accuracy	Precision	Recall	F1-score	AUPRC
Decision Tree	**0.97**	**0.98**	**0.98**	**0.98**	**0.0079**
Logistic Regression	**0.98**	**0.98**	**1.0**	**0.98**	**0.066**
Random Forest	**0.98**	**0.98**	**1.0**	**0.98**	**0.507**
XGBoost	**0.98**	**0.98**	**1.0**	**0.99**	**0.714**

ROC-Curve:



Inferences and Insights:

- With respect to accuracy, precision, recall and F1-score, all models seemed to be heavily overfitting and a plausible comparison couldn't be drawn.
- However, AUPRC score clearly displayed the performance trend where XG-Boost algorithm performed the best, and decision tree the least.
- SVM however was not able to fit the data at all, as suggested by the ROC curve.
- We conclude that **a single classification algorithm alone may not be efficient enough to classify credit card fraud, where are ensemble methods show improved performance.**

Limitations and Future Work: What if we had more time ?

- The primary task up ahead would be to find a suitable method to deal with the excessive **imbalance** in data. There are methods like Undersampling/Oversampling that can be experimented with. This would require more time and research.
- **Neural Networks** and deep learning architectures are found to be more effective at present, and still is a vast field of study associated with fraud detection. I would love to try out the different architectures, although computational hardware might be a limitation.

References :

https://medium.com/analytics-vidhya/credit-card-fraud-detection-in-python-using-scikit-learn-f9046a030f50

https://github.com/jbofill10/C1_Transaction_Data

Dataset : https://github.com/CapitalOneRecruiting/DS