Question 1: Load

- Programmatically download and load into your favorite analytical tool the transactions data. This data, which is in line-delimited JSON format, can be found

→ Downloaded the zip file, and modified transactions.txt to transactions.json. Added the json file to my drive and connected a Google Colaboratory file to my drive to access the file

- Please describe the structure of the data. Number of records and fields in each record?

→ Converted the json data to pandas dataframe
→ (786363, 29). The number of records = 786363
The number of fields = 29

- Please provide some additional basic summary statistics for each field. Be sure to include a count of null, minimum, maximum, and unique values where appropriate.

→ Performed a basic statistical analysis using the describe function of pandas

```
#Basic Summary statistics
df.describe()
```

	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

→ *Calculated the null values using the isnull() function*

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

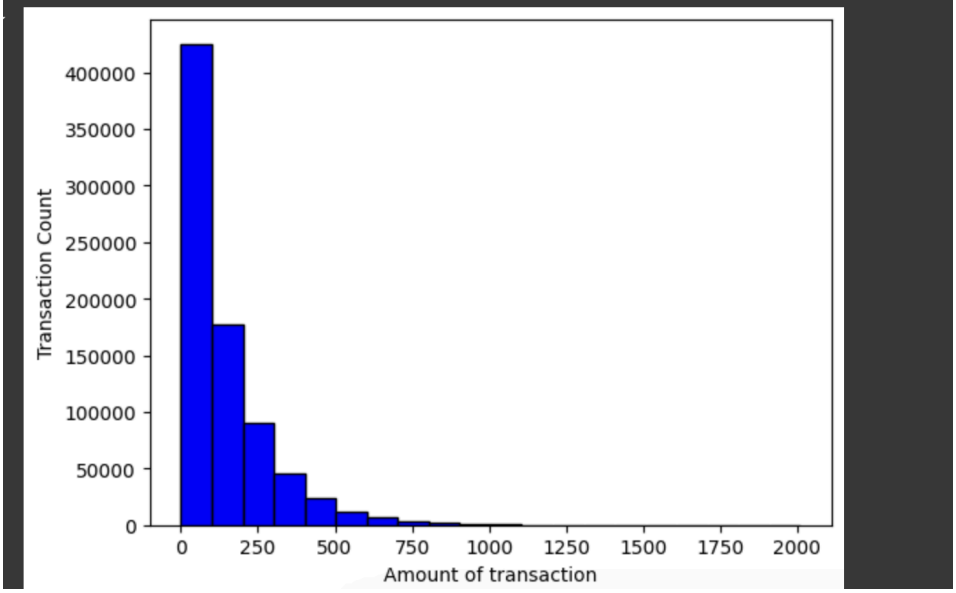
```
accountNumber                0
customerId                   0
creditLimit                  0
availableMoney               0
transactionDateTime          0
transactionAmount            0
merchantName                 0
acqCountry                   0
merchantCountryCode          0
posEntryMode                 0
posConditionCode             0
merchantCategoryCode         0
currentExpDate               0
accountOpenDate              0
dateOfLastAddressChange      0
cardCVV                      0
enteredCVV                   0
cardLast4Digits              0
transactionType              0
echoBuffer                   0
currentBalance               0
merchantCity                 0
merchantState                0
merchantZip                  0
cardPresent                  0
posOnPremises                0
recurringAuthInd             0
expirationDateKeyInMatch     0
isFraud                      0
```

Question 2: Plot

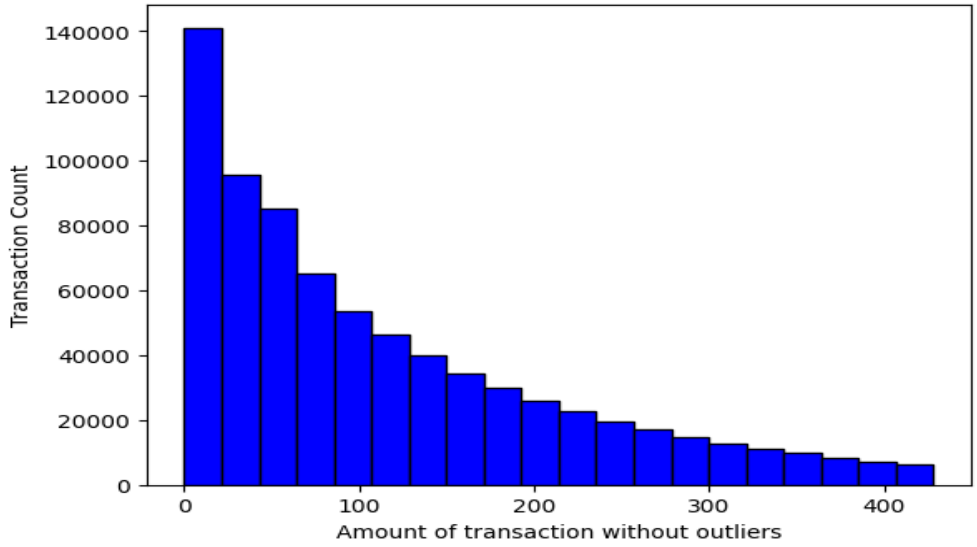
- Plot a histogram of the processed amounts of each transaction, the transactionAmount column.

→ *The plot below depicts the transactionAmount of each record*

```
plt.hist(df['transactionAmount'], color='blue', edgecolor='black', bins=20)
plt.xlabel('Amount of transaction', fontsize=10)
plt.ylabel('Transaction Count', fontsize=10)
plt.figure(figsize=(25,15))
plt.show()
```



→ The plot below is obtained after removing the outliers. Statistically, outliers can be handled by calculating the quartile ranges.



- Report any structure you find and any hypotheses you have about that structure.

→ Using the 25%(Q1), 50%(Q2) and 75%(Q3) quartile ranges and the formula: Q3 + 1.5 IQR(i.e Inter Quartile range). The outliers lie within this range. (IQR is the range between the third and first quartile).
→ Dealing with outliers is important to avoid skewed results.
→ This shows that the maximum transaction was around 500

Question 3: Data Wrangling - Duplicate Transactions

You will notice a number of what look like duplicated transactions in the data set. One type of duplicated transaction is a reversed transaction, where a purchase is followed by a reversal. Another example is a multi-swipe, where a vendor accidentally charges a customer's card multiple times within a short time span.

Can you programmatically identify reversed and multi-swipe transactions?
What total number of transactions and total dollar amount do you estimate for the reversed transactions? For the multi-swipe transactions? (please consider the first transaction to be "normal" and exclude it from the number of transaction and dollar amount counts)

→ I split this into two criteria. The first criterion was reversed transactions and the second was multi-swipe transactions.

→ For reversed transactions, I used the transactionType column with 'REVERSAL' and 'PURCHASE' values to create two separate data frames. The next step was to join these data frames in an inner join fashion such that only the values in common were in the new dataframe. This was done in order to obtain the exact reversed transaction immediately after the purchase. Finally, I gathered data where the purchase transaction time was less than the reversed transaction time. The total sum of dollar amount for reversed transactions obtained was 2701311.87. The total number of reversed transactions is 17999

→ For multi-swipe transactions, I grouped transactions that occurred back to back, sorted them based on transaction time and finally created a dataframe where the transactions occurred in less than an hour. The total sum of the dollar amount for multi-swipe transactions (after treating the first element of each of the duplicated transactions as a normal transaction) is 962268.78 and the total number of multi swipe transactions is 6269

→ As a future scope, I would try to understand the transaction times and then group them based on not just under an hour but different time intervals.

Did you find anything interesting about either kind of transaction?

→ *Most of the transactions were under 600 units of amount*

Question 4: Model

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.

- Each of the transactions in the dataset has a field called isFraud. Please build a predictive model to determine whether a given transaction will be fraudulent or not. Use as much of the data as you like (or all of it).

→ *The predictive model created was using Decision Trees and Random Forest.*

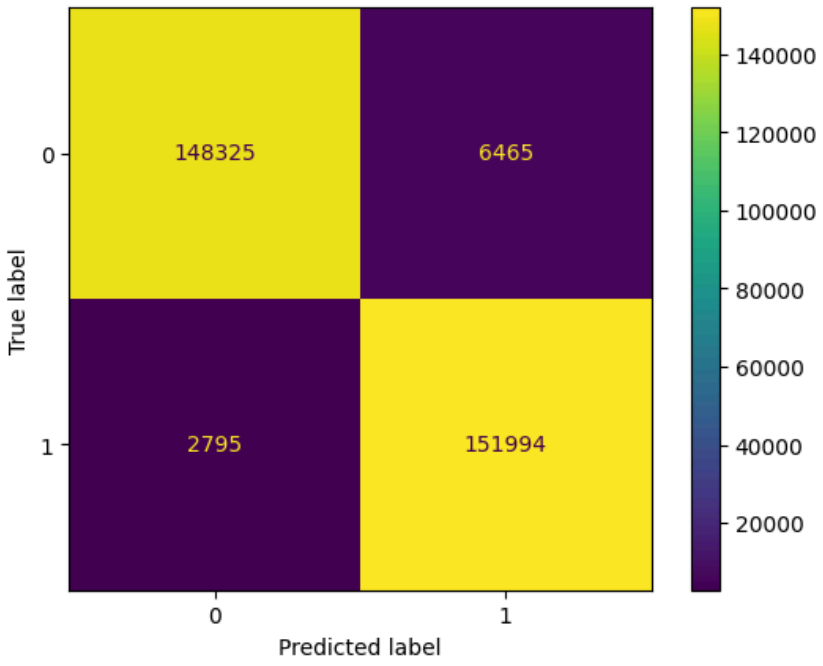
- Provide an estimate of performance using an appropriate sample, and show your work.

→ *The performance metrics for Random Forest Model with Randomised Search CV for hyperparameter optimisation are given below:*

→ *The training accuracy is : 0.9999725432907512*

→ *The test accuracy is : 0.9700884103895936*

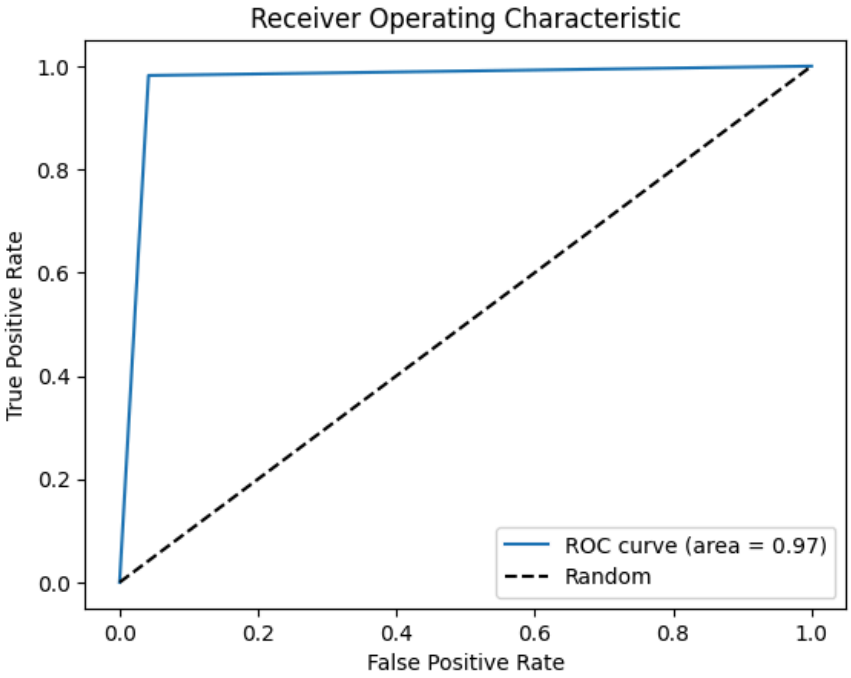
→ *The matrix below is that of a confusion matrix.*



→ *The True Positive Value is : 148325*
→ *The True Negative Value is : 151994*
→ *The False Positive Value is : 6465*
→ *The False Negative Value is : 2795*

→ *Precision Score = 0.9592008027313059*
→ *Recall Score = 0.9819431613357539*
→ *The f1 score value is : 0.9704387577893554*

→ The ROC-AUC curve is plotted below.



- Please explain your methodology (modeling algorithm/method used and why, what features/data you found useful, what questions you have, and what you would do next with more time)

  → *The features that I used for the model were:*
  [accountNumber, customerId, creditLimit, availableMoney, transactionAmount, currentBalance, isFraud]

  → *The issue with the data set was that it was imbalanced. So I used the technique of oversampling to create a balanced dataset. I used the Borderline smote technique because of its better performance compared to smote.*

  → *After creating a balanced data set, I decided to use a decision tree model. After running and testing the model, the results were indicative of the model overfitting the*

data. That is the training accuracy was 99% while the test accuracy was 95%. Overfitting is when the model picks up random fluctuations or noise and this affects its performance on a new set of data(test data)

→ To address this issue, I used Random Forest with Randomised Search CV to use optimized hyperparameters that aid in overcoming the issue of overfitting.

→ The test accuracy jumped to 97% with an F1 score (which is a harmonic mean of precision and recall) was 0.9704

→ The other method that I tried was using undersampling and Logistic regression and a simple random forest implementation(This is a separate ipynb file that I've added in the folder). The model performance was underwhelming. Clearly, the undersampled data set was not enough for the model to capture the underlying relations between the features.

→ The idea behind using decision tree and random forest for model building is that these models are better at making decisions, in this case, determining the nature of a transaction(fraudulent or legit). Decision tree makes a sequence of decisions in a tree like fashion and a random forest is an ensemble of such decision trees. The ability to reduce overfitting and reducing variance makes random forest a better model than decision trees.

→ The reason behind using Randomised Search CV was to use optimised hyperparameters so that the model gets trained for better performance. The CV is cross-validation which is performed during the optimization process. The number of iterations is specified in the parameters list of the Randomised Search CV. Each iteration samples a random combination of the hyperparameters, thereby increasing the chances of finding a good combination.

→ Questions I have: What was the echoBuffer column for?

→ Things I would have done differently:

a) *Performed feature engineering in a thorough manner. I would use PCA for dimensionality reduction. Most importantly, I would try and create a correlation between the transaction times and the amount to understand their influence on the nature of the transaction. I would also try to look into zip codes for consecutive transactions from the same card or account along with the transaction time.*
b) *I would use normalisation technique to improve the values of the features. Maybe min-max would be my first choice.*
c) *I would optimize the logistic regression model on the oversampled dataset and compare it with the results of the random forest.*