Fraud Detection Using Machine Learning Algorithms

Sarah Hosseini 400222026

Shahid Beheshti University

Fraud Detection

1. Data collection and EDA

We load the dataset into a dataframe. It has 1296675 rows and 23 columns. We notice that the first column is just the index of rows so we delete that column. The remaining columns are:

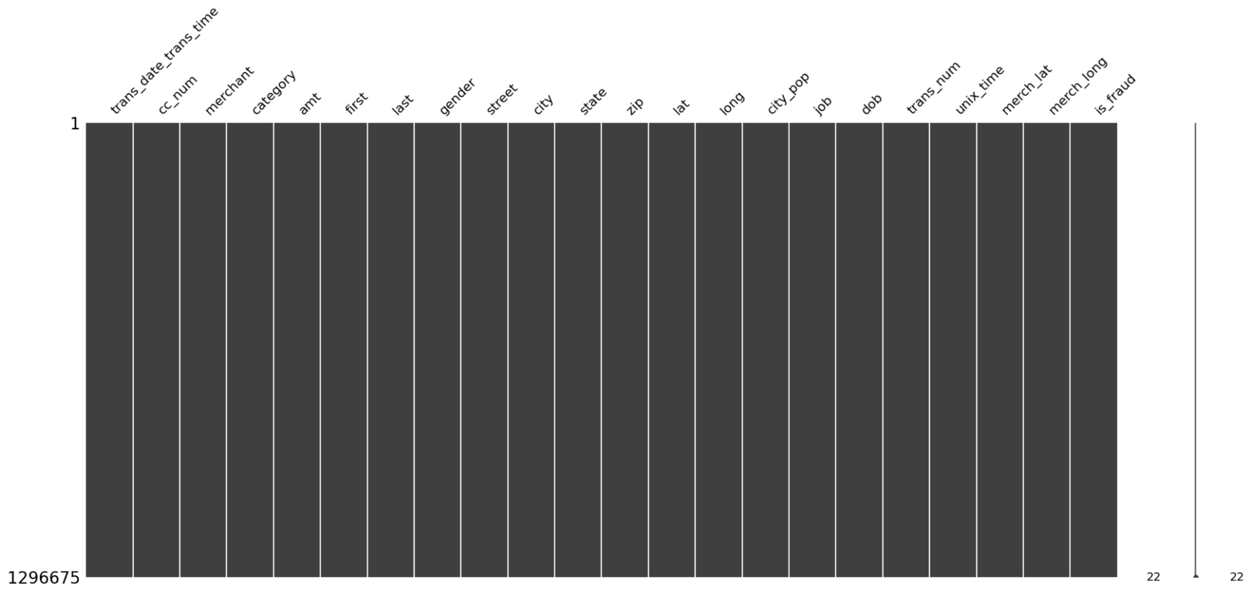
- Trans_date_trans_time: the time the transaction took place.
- cc_num: Credit card number of the customer.
- merchant: Name of the merchant or establishment where the transaction took place.
- category: Category or type of the transaction (e.g., retail, dining, travel).
- amt: Amount of the transaction.
- first: First name of the customer.
- last: Last name of the customer.
- gender: Gender of the customer.
- street: Street address of the customer.
- city: City of the customer's address.
- state: State or province of the customer's address.
- zip: ZIP or postal code of the customer's address.
- lat: Latitude coordinate of the customer's address.
- long: Longitude coordination of the customer's address.
- city_pop: Population of the city where the customer resides.
- job: Occupation or job title of the customer.
- dob: Date of birth of the customer.
- trans_num: Transaction number or identifier.

- unix_time: The transaction time recorded as a UNIX timestamp (number of seconds since January 1, 1970).
- merch_lat: Latitude coordinate of the merchant's location.
- merch_long: Longitude coordinate of the merchant's location.
- is_fraud: A binary variable indicating whether the transaction is flagged as fraudulent (1) or not (0).

We then convert the ‘trans_date_trans_time’ and’dob’ to datetime datatype.

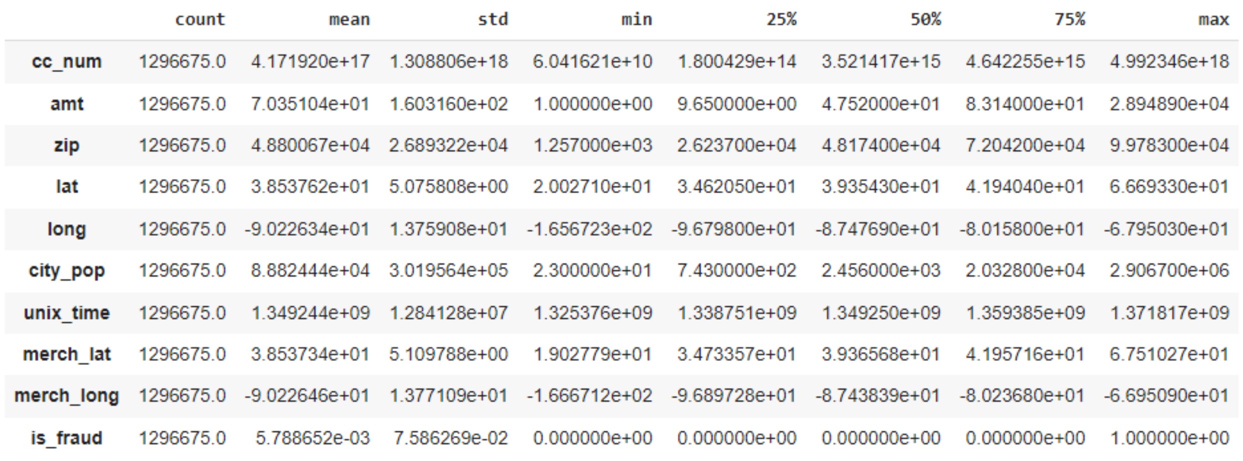
There are a few categorical (with less than 20 unique values) columns: ‘category’, ‘gender’, and ‘is_fraud’.

Also, there are no missing values in any columns:



Also, there were no identical columns so no column was deleted.

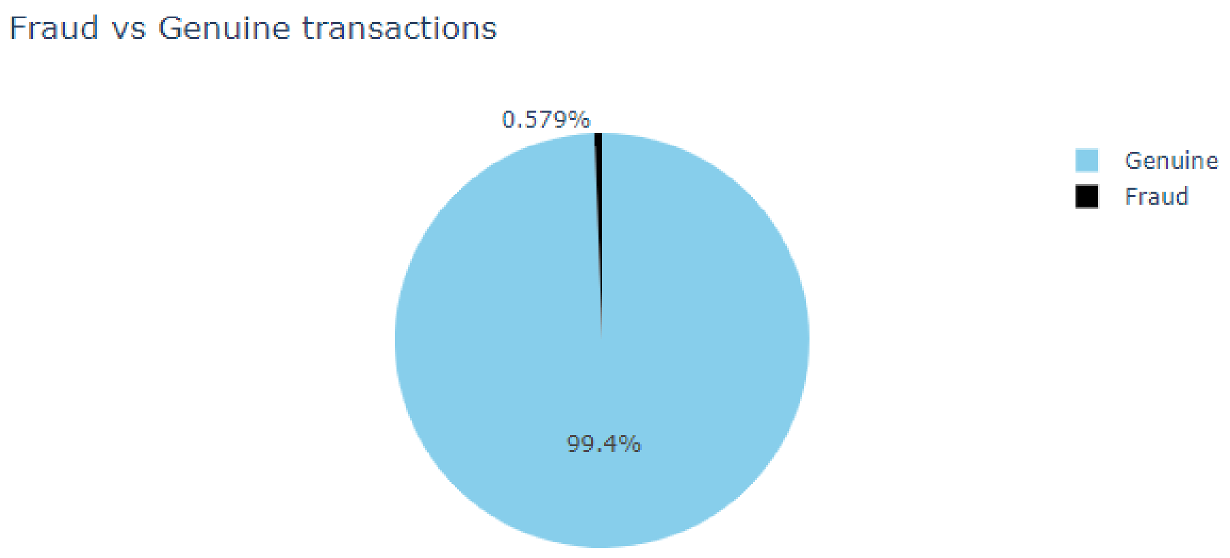
We also used the describe function to get a sense of the numerical data:



	count	mean	std	min	25%	50%	75%	max
cc_num	1296675.0	4.171920e+17	1.308806e+18	6.041621e+10	1.800429e+14	3.521417e+15	4.642255e+15	4.992346e+18
amt	1296675.0	7.035104e+01	1.603160e+02	1.000000e+00	9.650000e+00	4.752000e+01	8.314000e+01	2.894890e+04
zip	1296675.0	4.880067e+04	2.689322e+04	1.257000e+03	2.623700e+04	4.817400e+04	7.204200e+04	9.978300e+04
lat	1296675.0	3.853762e+01	5.075808e+00	2.002710e+01	3.462050e+01	3.935430e+01	4.194040e+01	6.669330e+01
long	1296675.0	-9.022634e+01	1.375908e+01	-1.656723e+02	-9.679800e+01	-8.747690e+01	-8.015800e+01	-6.795030e+01
city_pop	1296675.0	8.882444e+04	3.019564e+05	2.300000e+01	7.430000e+02	2.456000e+03	2.032800e+04	2.906700e+06
unix_time	1296675.0	1.349244e+09	1.284128e+07	1.325376e+09	1.338751e+09	1.349250e+09	1.359385e+09	1.371817e+09
merch_lat	1296675.0	3.853734e+01	5.109788e+00	1.902779e+01	3.473357e+01	3.936568e+01	4.195716e+01	6.751027e+01
merch_long	1296675.0	-9.022646e+01	1.377109e+01	-1.666712e+02	-9.689728e+01	-8.743839e+01	-8.023680e+01	-6.695090e+01
is_fraud	1296675.0	5.788652e-03	7.586269e-02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.000000e+00

1.2 univariate variable analysis

We visualize the percentage of fraud transactions:



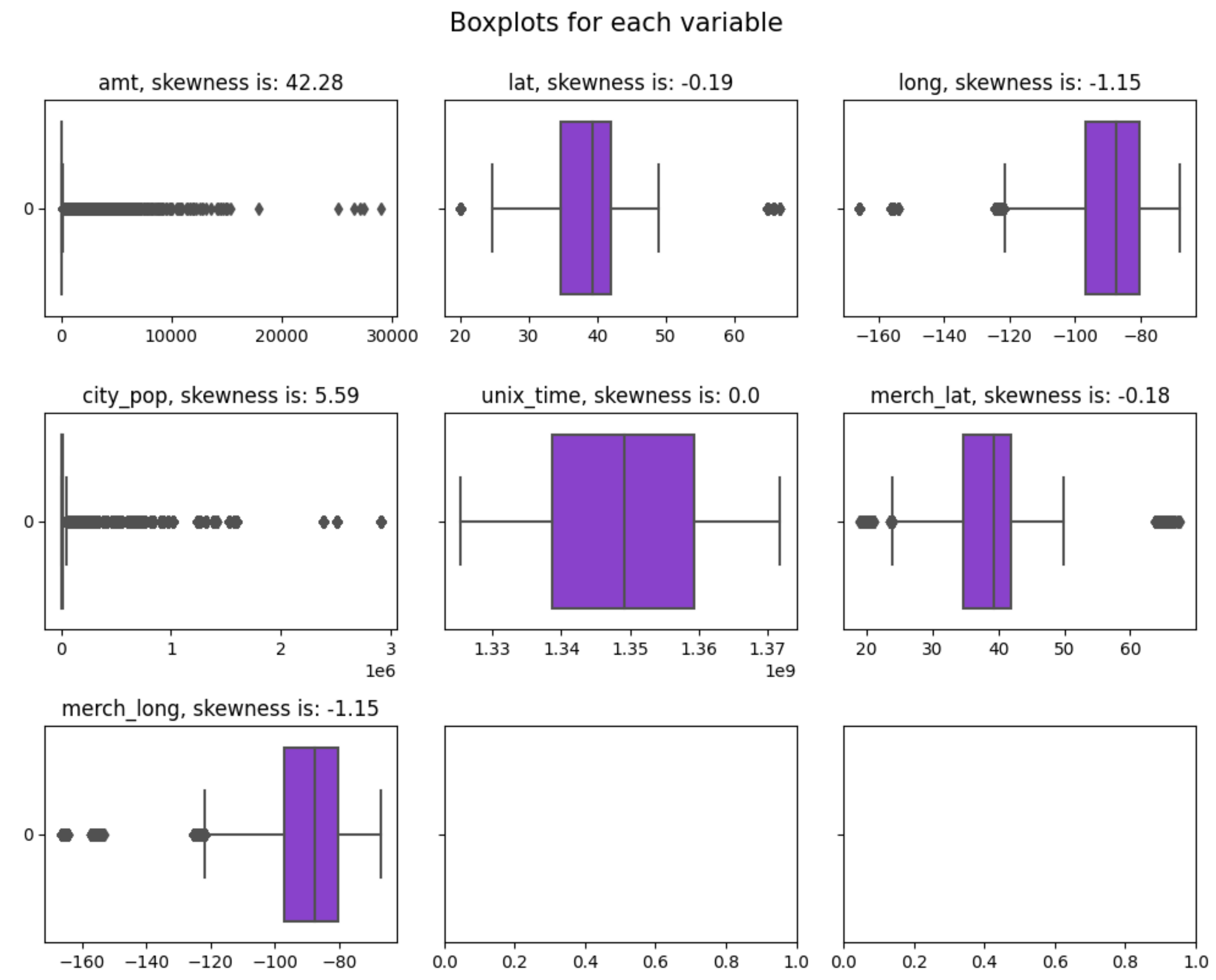
This dataset is imbalanced. Because in this dataset:

- we have **99,42%** of genuine transactions and only **0,58%** of fraud transactions;

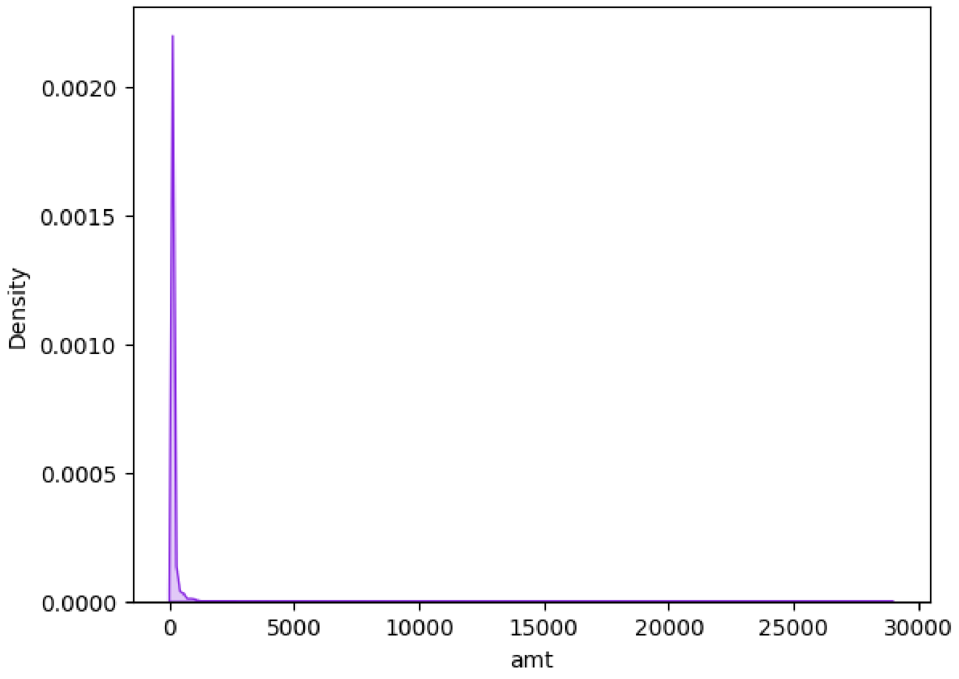
- this means that a blind guess would give us an accuracy of **99,42%**.

So, the accuracy score is not a good metric because it will most likely be a high number.

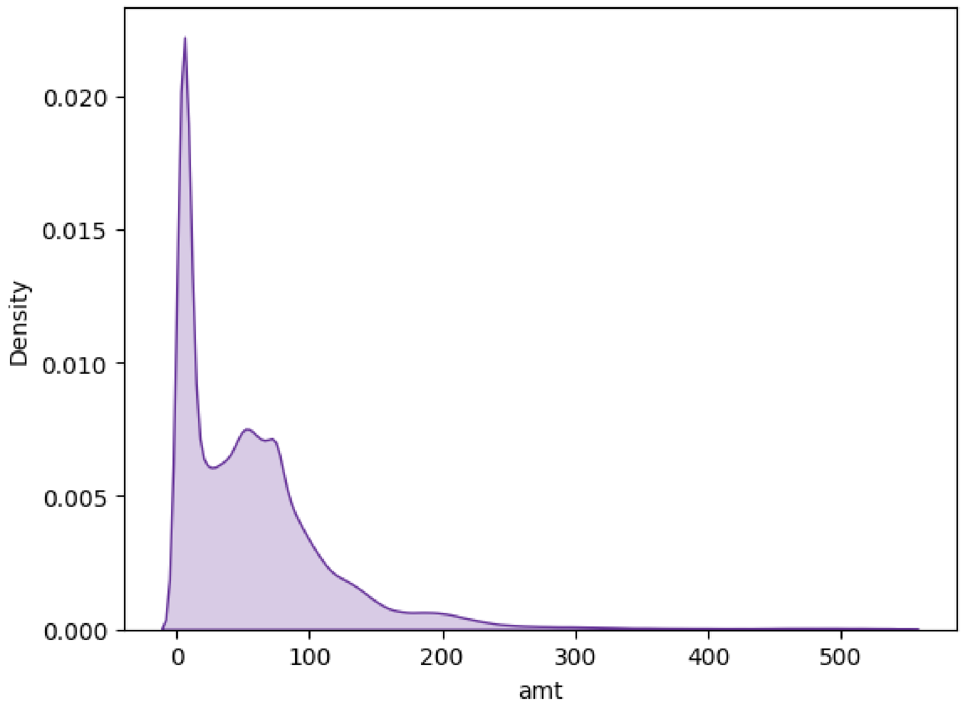
We visualized the distribution of numeric columns with these box plots:



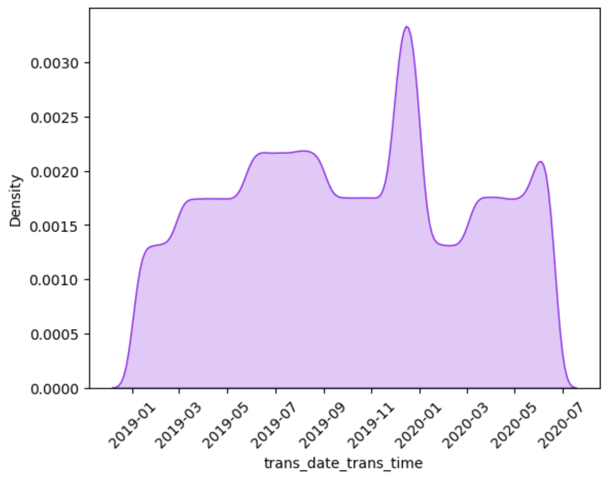
As seen above, the ‘amt’ variable is highly skewed. When we plot the density plot of this variable, we get this:

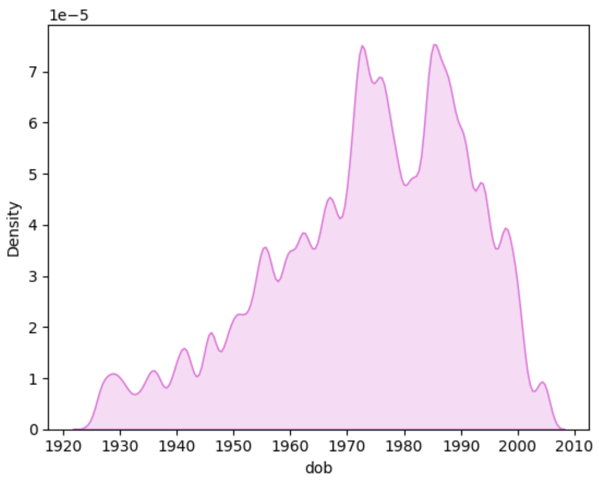


So, we plot again using only the subset of data that is less than its 99th percentile. It looks like this:



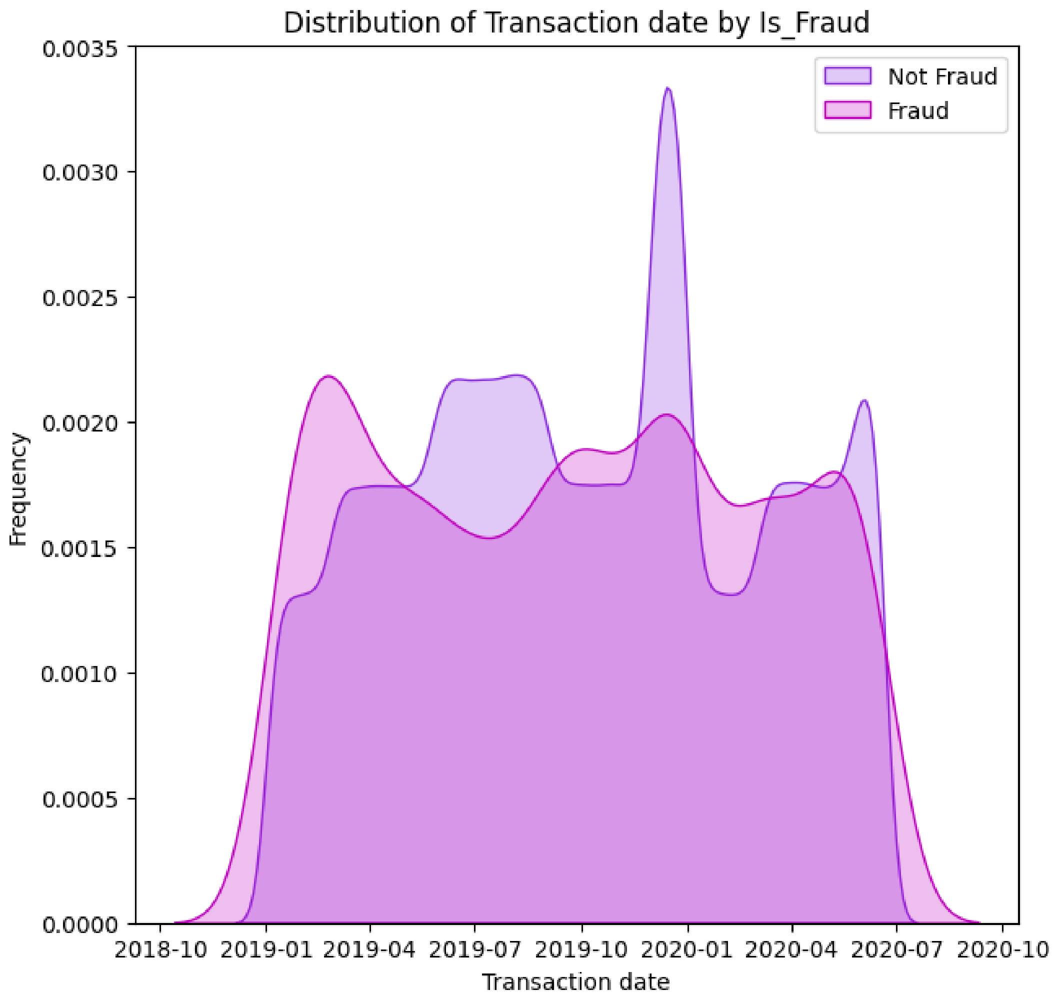
These plots demonstrate the frequencies of the transaction time and date of birth of the customers.



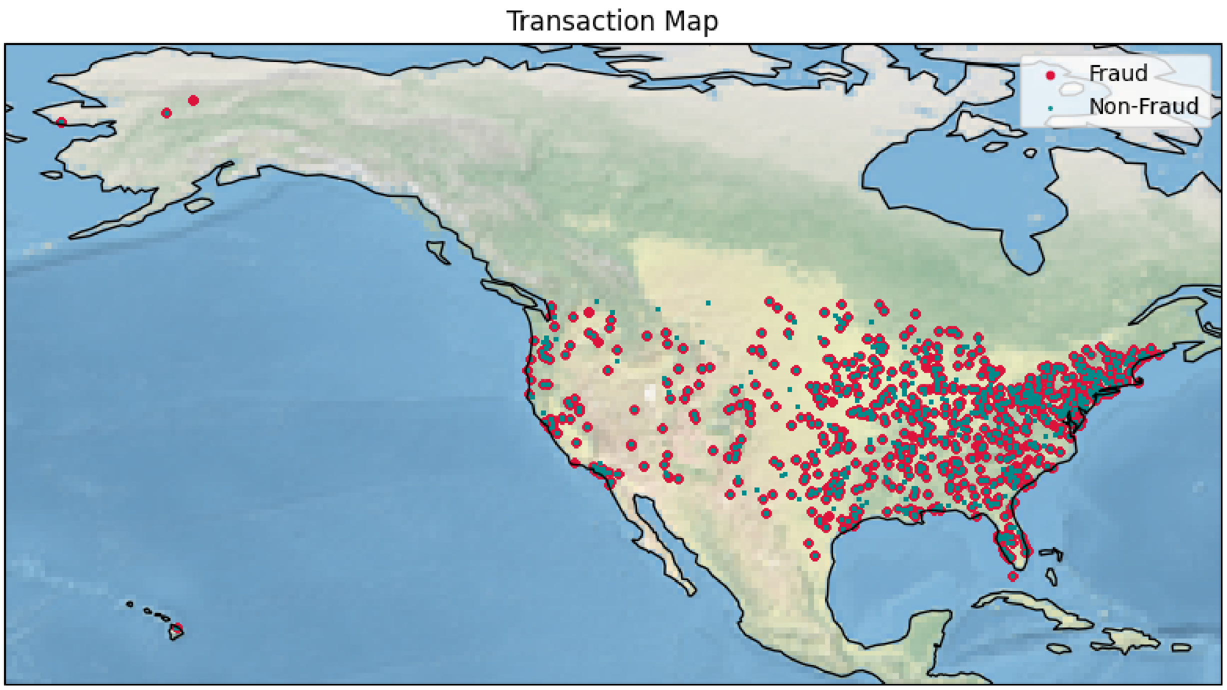


1.3. Multivariate variable analysis

This plot shows the distribution of transaction dates for both the "Not Fraud" and "Fraud" groups, allowing for a visual comparison of the two distributions. This can provide insights into potential patterns or differences between fraudulent and non-fraudulent transactions based on the transaction date.

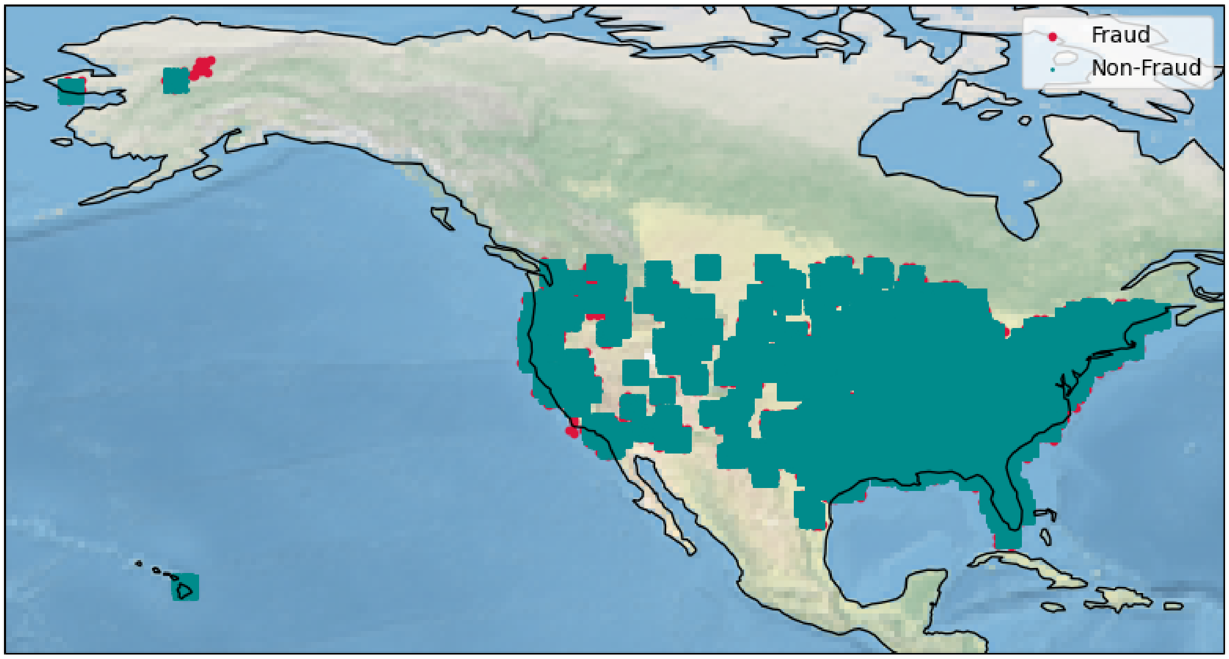


In the map below, we see the distribution of fraudulent versus non-fraudulent transactions based on where the customer was located. Note that even though the number of frauds (7506) is very small compared to the genuine ones (1289169), the frauds are distributed almost everywhere that genuine ones are.

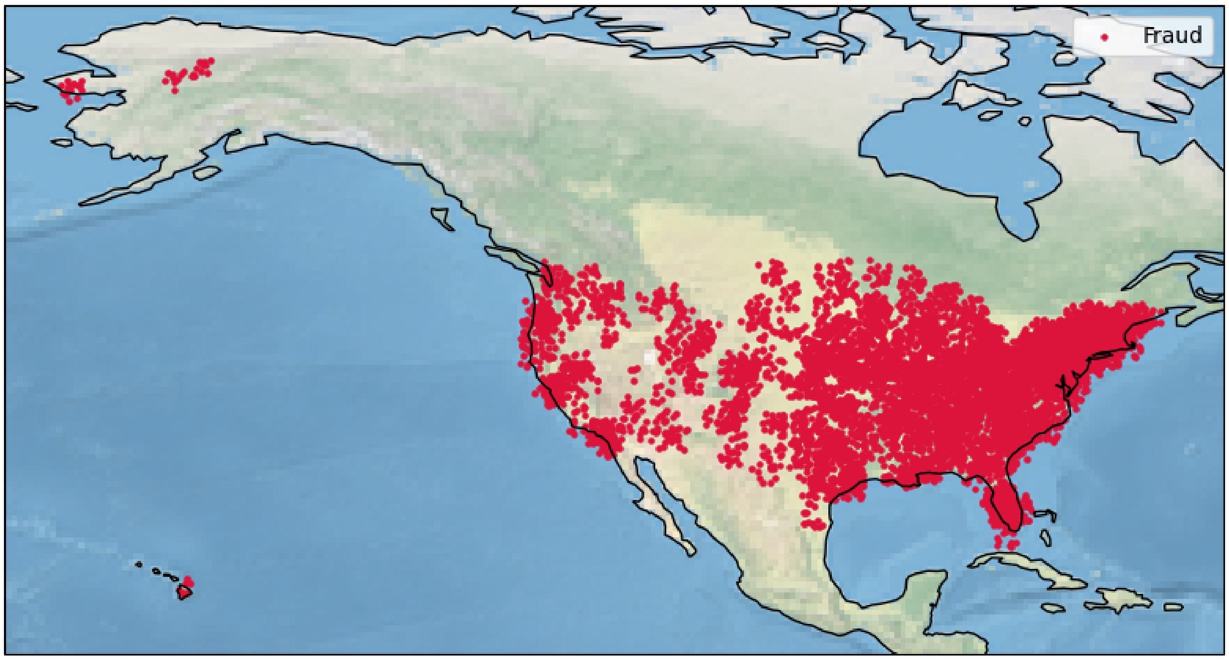


The following map shows the distribution of merchants' locations. The same thing happens here where the frauds are distributed everywhere that genuine ones are even though they are very fewer than the genuine ones.

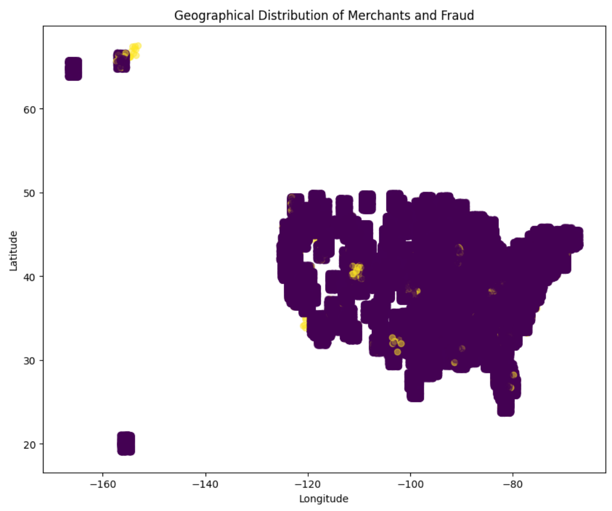
Merchent Map

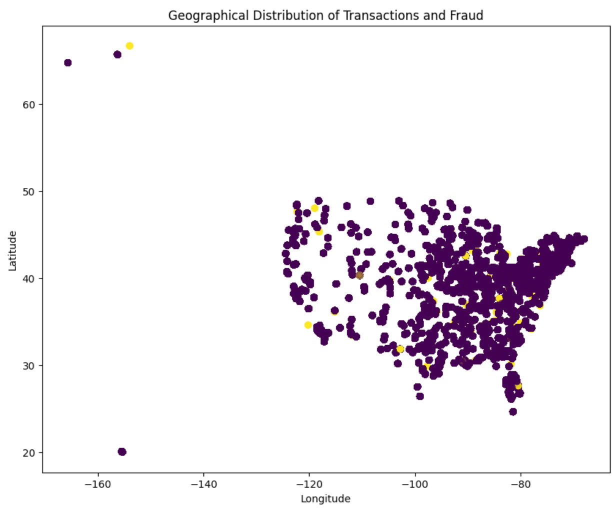


Merchent Map

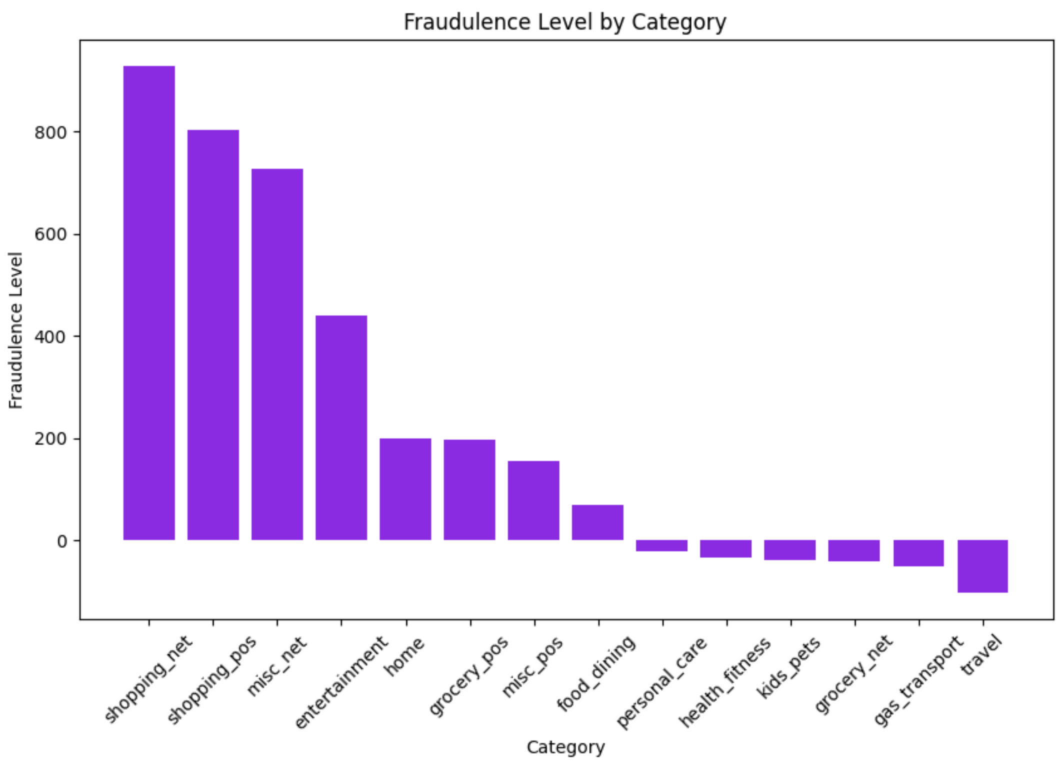


Fraud Detection



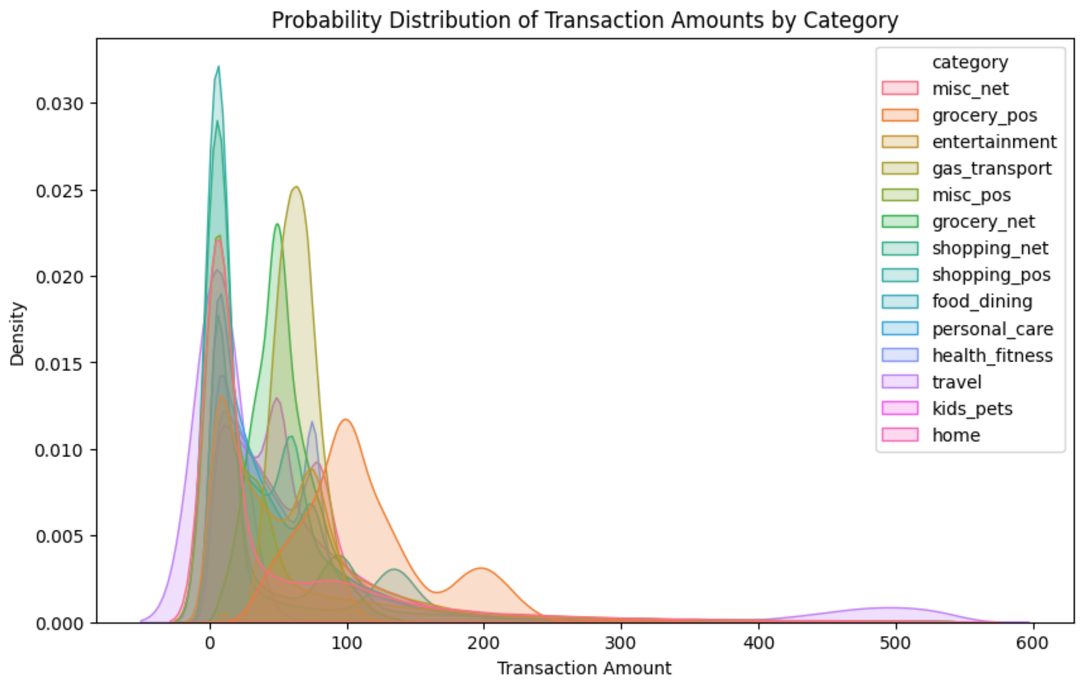


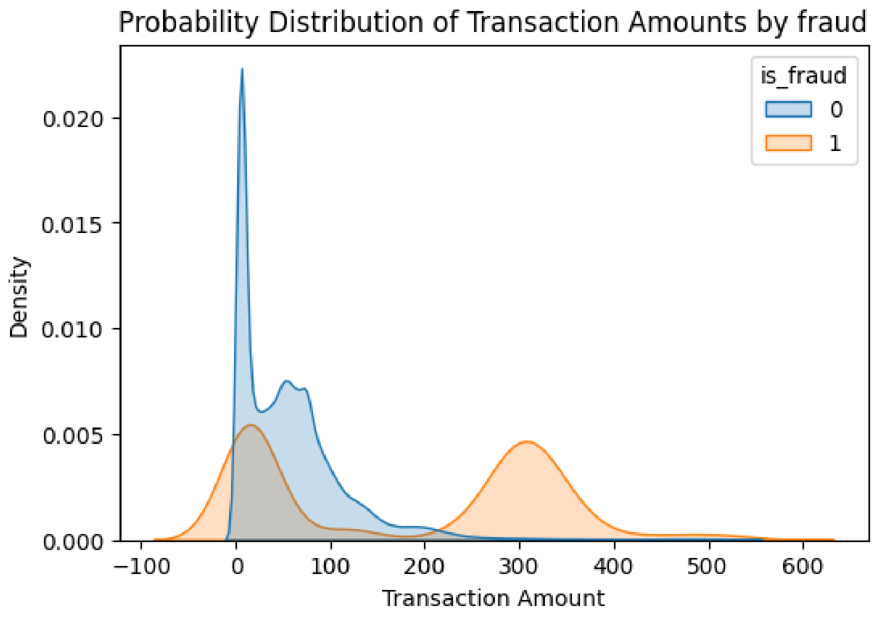
After that, we came up with a measurement to see which categories have a larger amount of fraudulent money transactions. Positive values of fraudulence level in this plot indicate that the average transaction amount is higher for fraudulent transactions in that category compared to non-fraudulent transactions:



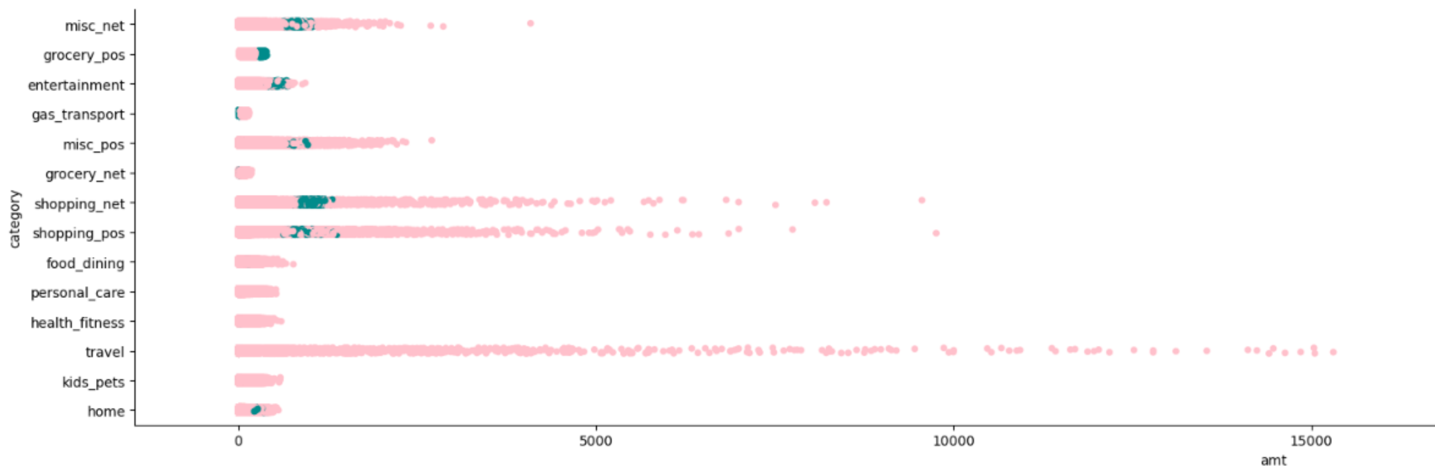
Fraud Detection

These plots show the distribution of the money amount of transactions based on their category and their fraudulence state:

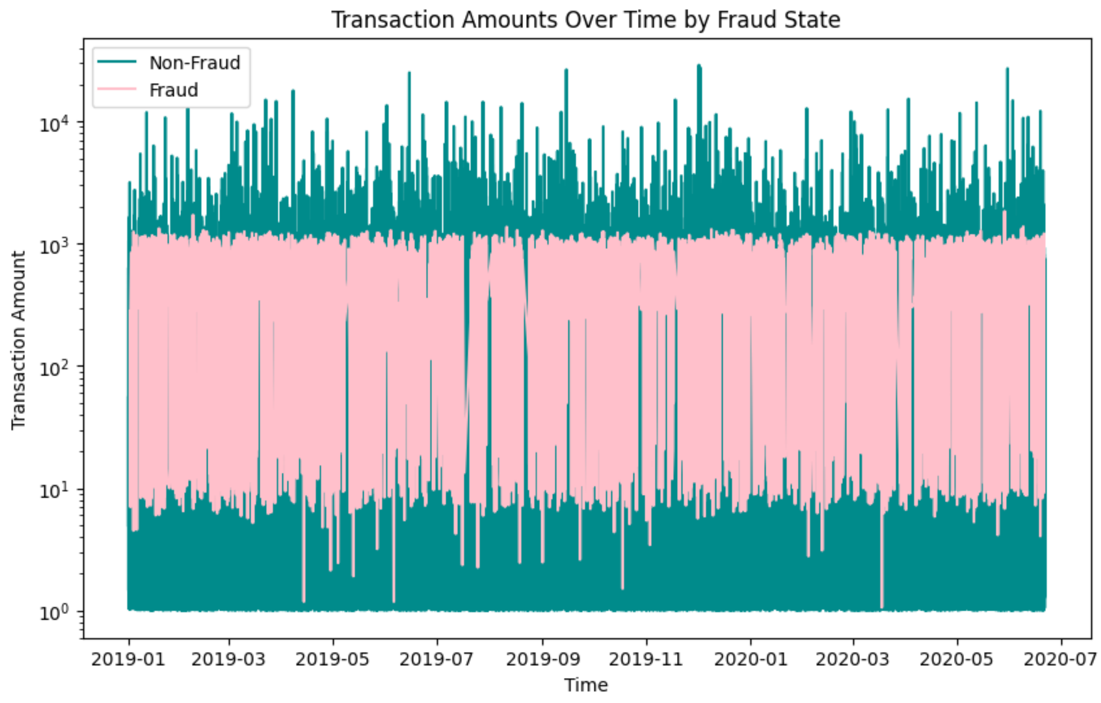




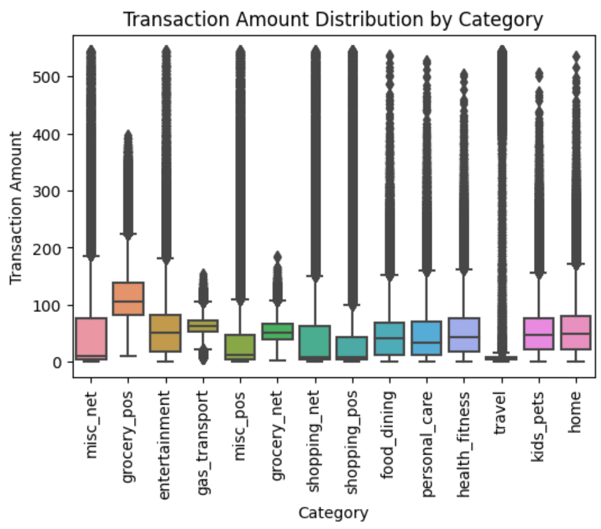
The catplot below illustrates that the most frauds happened for the categories: [Shopping_net, Shopping_pos, misc_pos, entertainment, grocery_pos]. (Compare with the list of the highest fraudulent_level we calculated earlier: [Shopping_net, Shopping_pos, misc_net, entertainment, home]). Pink circles are genuine and green ones are fraud:

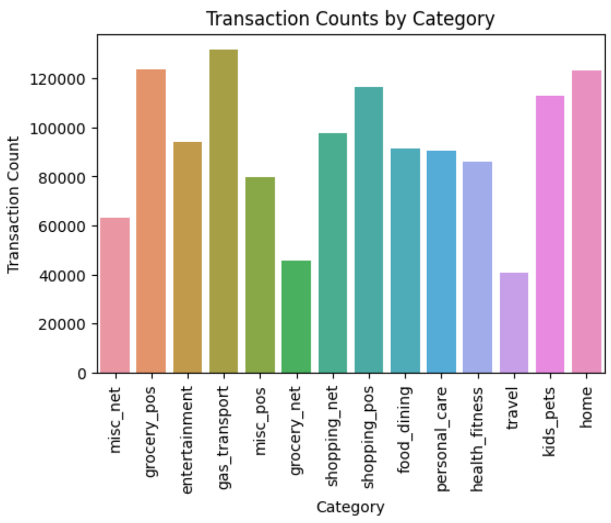


We can also demonstrate the amount of transactions in time. Here, the y-axis is logarithmic to help magnify the visual representation of smaller values, such as the transaction amounts for fraud cases.



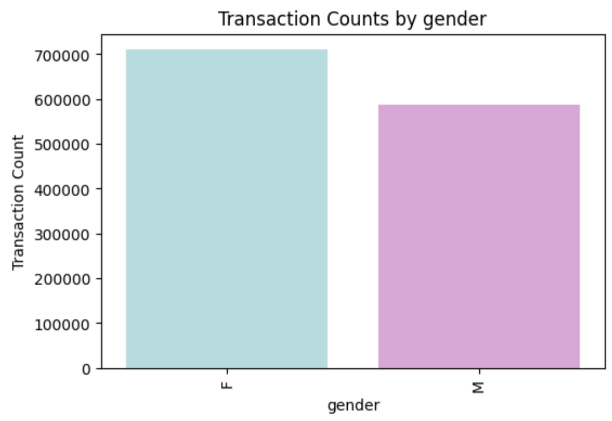
To analyze the transactions based on their categories, we have these two plots:

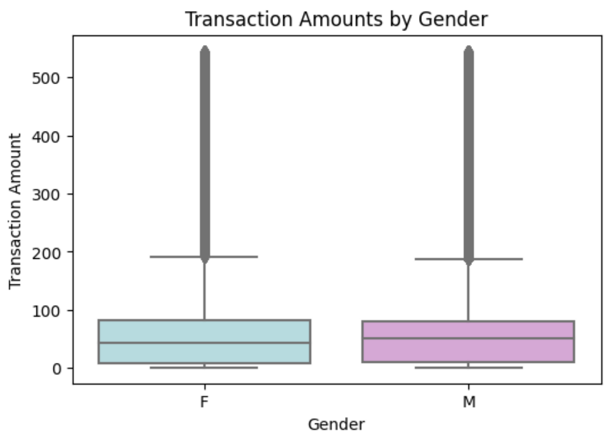




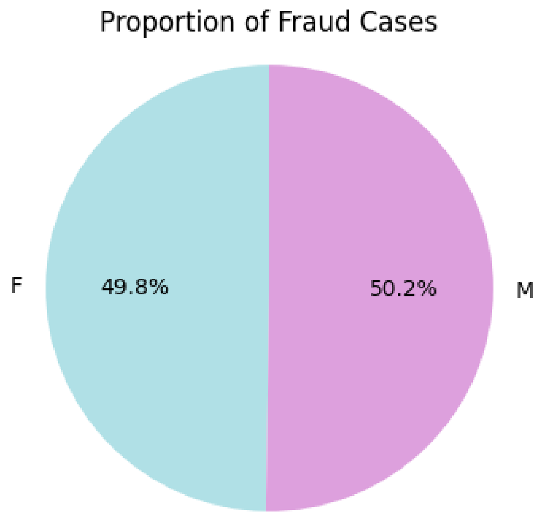
This shows that ‘grocery_pos’ has the highest amount and a high transaction count. So, this category is probably very important.

Then, we have these plots about the gender column. They show that most transactions are from female customers, and the distribution of the amount of money is somewhat similar between the two genders.

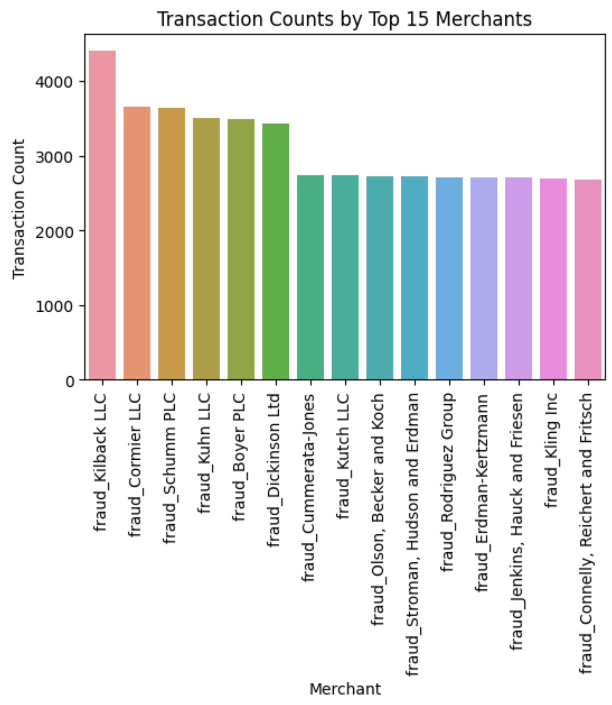


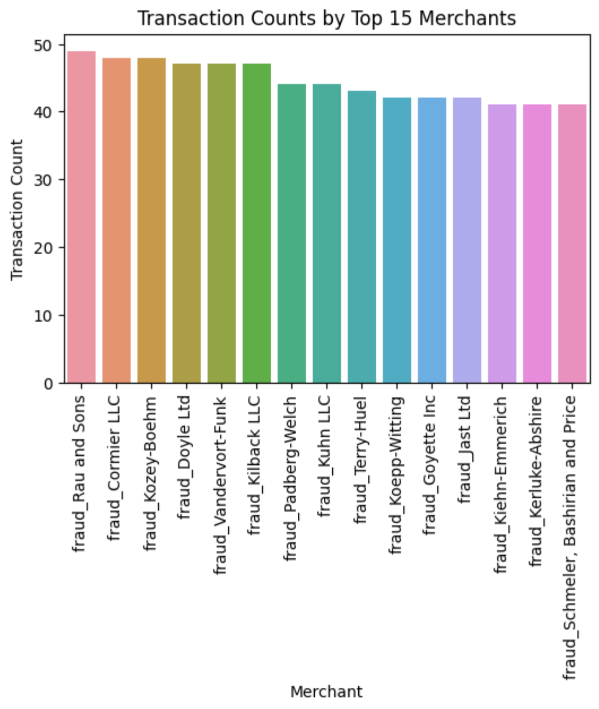


However, in fraud cases, about half of the frauds are by male customers and half females:



Finally, we have these two plots that show which merchant has the most transactions and which has the most fraud transactions:





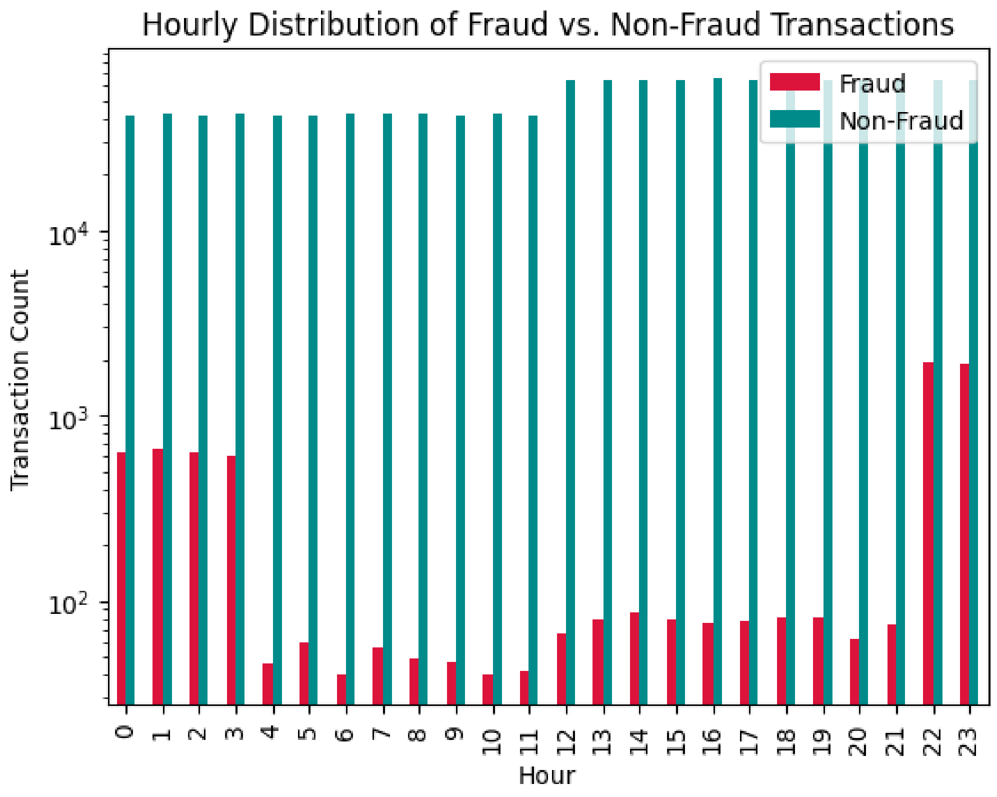
2. Feature transformation and feature engineering

We don’t want to have leakage from test to train data, so we start by splitting the data into train and test sets with a ratio of 80:20. Then we apply the following changes to both sets separately. Since all of the values in the merchant column start with ‘fraud_’, we remove that from them. Then, using MinMaxScaler from scikit-learn, we scale the numerical variable ‘amt’.

The date of birth column might not be as informative by itself. However, we can derive additional features from it, such as the customer's age or age group, which can be useful for fraud detection. The data on the temporal aspects of the transaction, such as the hour of the day, day of the week, or month is useful too. Fraudulent activity can exhibit patterns based on specific periods or temporal dynamics, and encoding these features can help capture such patterns.

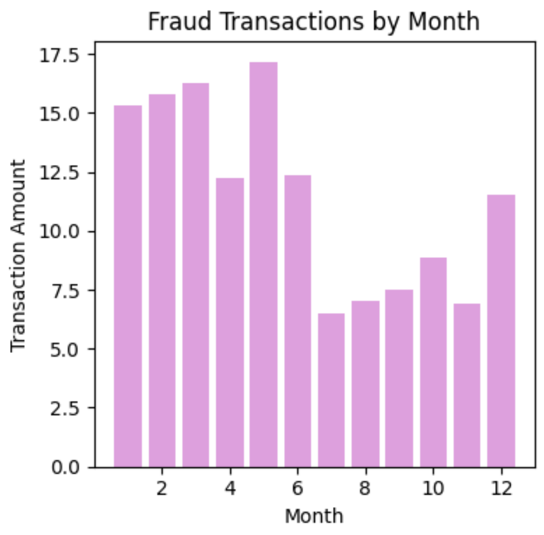
Therefore, we create the new columns ‘year’, ‘month’, ‘day’, ‘hour’, and ‘weekday’ from the transaction time that we had, as well as ‘age’ from the date of birth of the customer.

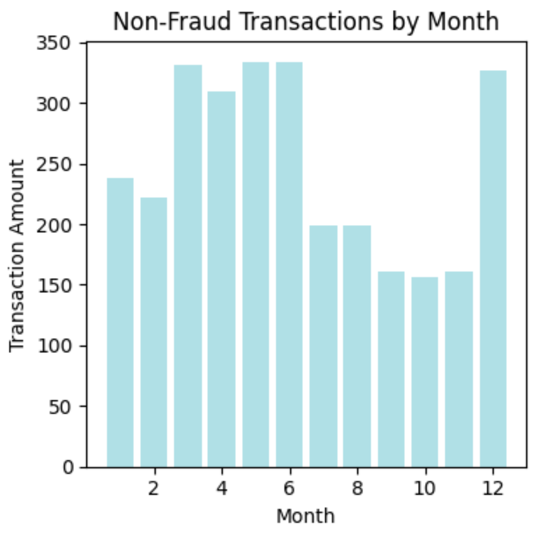
This plot shows the hourly distribution of transactions with a logarithmic y-scale to capture the frauds better. It shows that most fraud activities happen overnight after 10 pm and before 4 am:



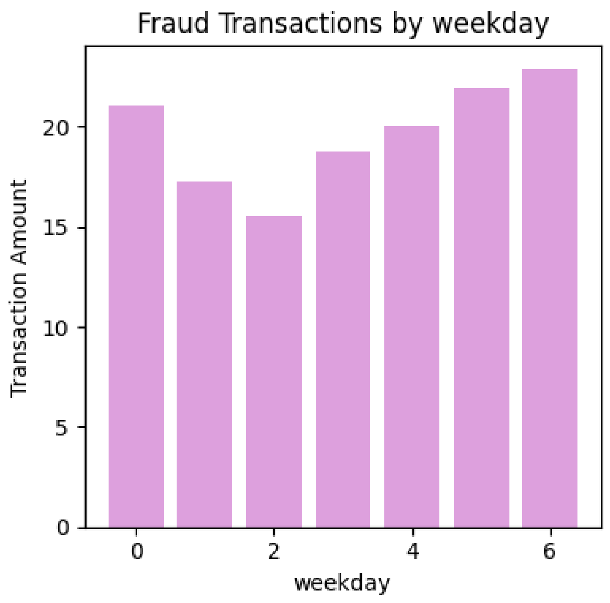
Fraud Detection

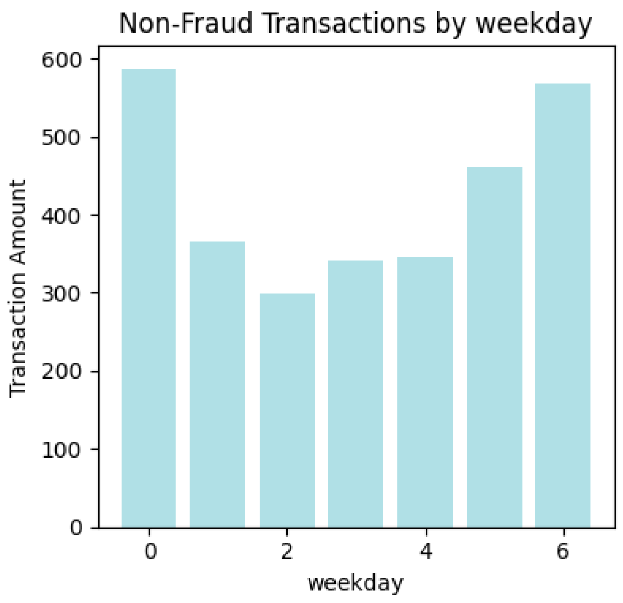
These plots show the transaction amounts of fraud and non-fraud transactions based on month:



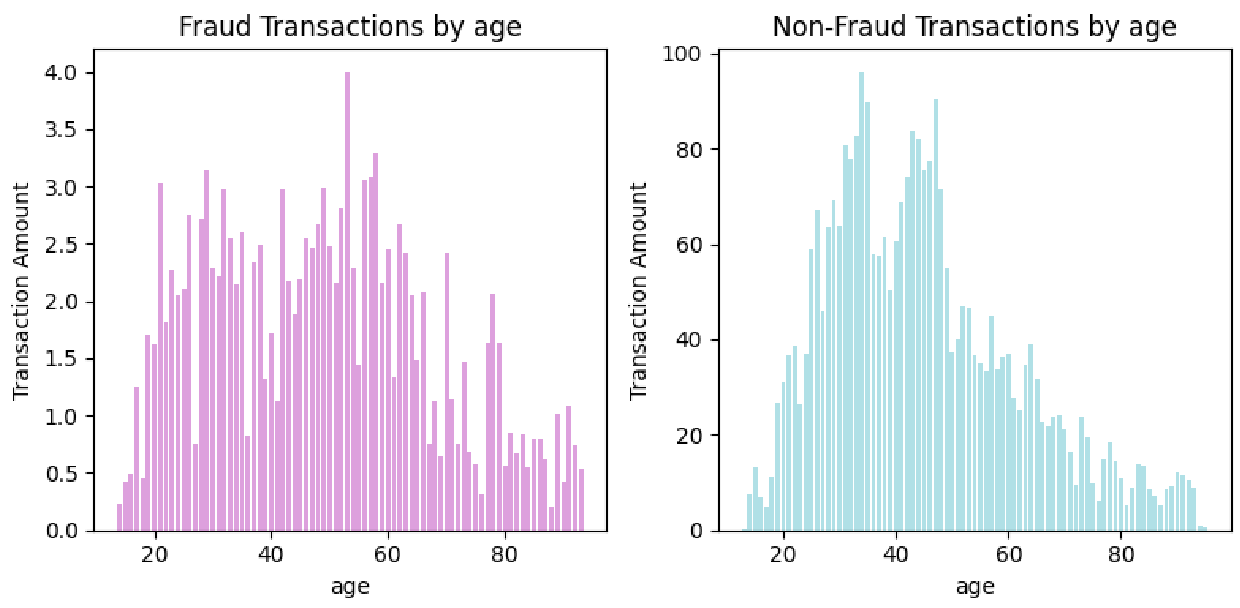


And then analyzing based on the weekday:



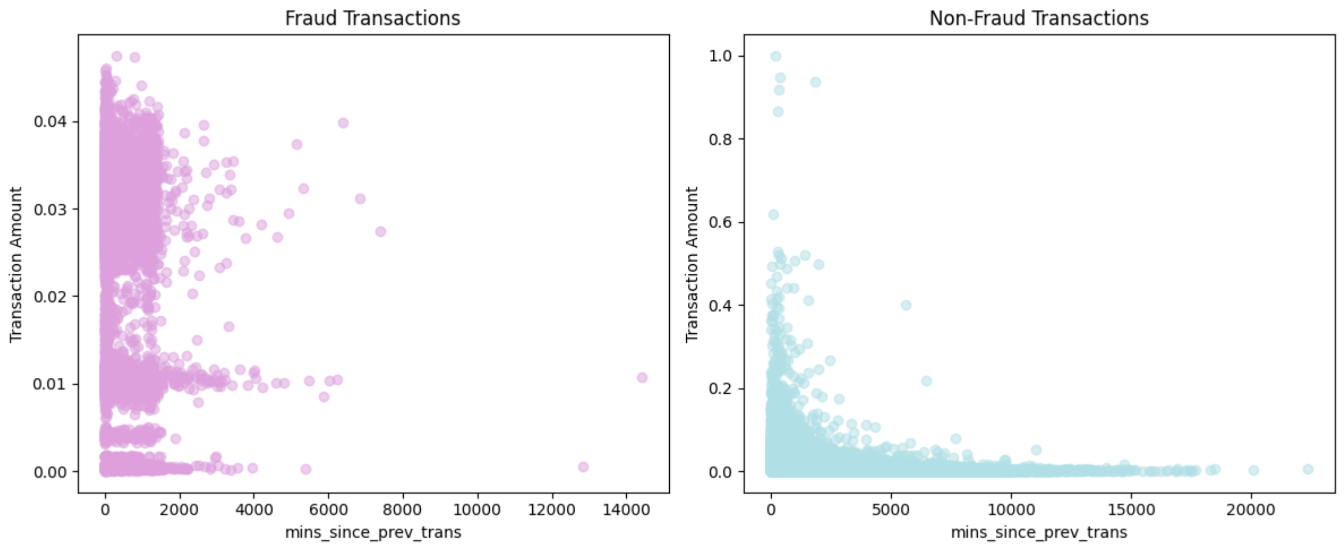


The same plots for the age of the customer which show the distributions are very different:



We also checked to get the cc numbers used by different people. Since this dataset was empty, we ensured that each cc_num was unique to each person and could be the customer's identifier. Then, using cc_num as an id, we created ‘total_trans_num’: the total number of transactions per customer, ‘avg_amt’: average transaction amount per customer, and ‘mins_since_prev_trans’: time passed after the previous transaction of each customer, in minutes. Note that we put -1 for the first transactions without previous transactions.

Then we plotted the minutes since the previous transaction of the data for fraud versus non-frauds. The plots show that the distributions are very different and fraudulent transactions with the same credit cards happen within shorter intervals.

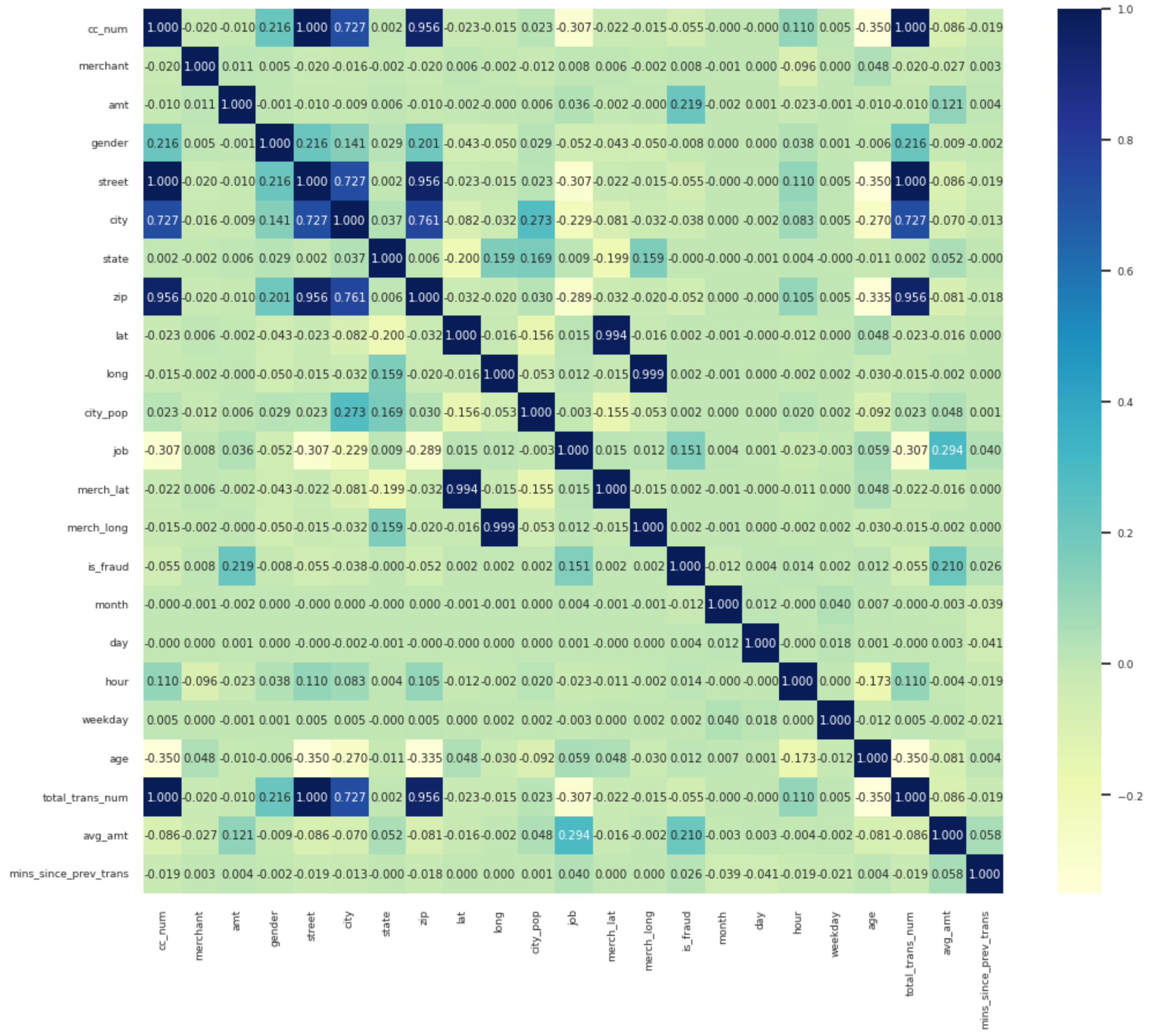


After that, we encode the non-numeric variables. We replace ‘F’ in gender with 1 and ‘M’ with 0. Since we are going to use several classifiers and each of them works better with a certain way of encoding, we should pay close attention to the classifiers’s requirements. As for one-hot encoding, all of the classifiers can understand and work with it properly. But out of all the features, only ‘category’ that has 14 unique values is suitable for one-hot encoding, and applying this encoding to other features will radically add to the dimension of our data. So we apply it to ‘category’ only. For ‘job’, we use target encoding, and for 'cc_num', 'merchant', 'city', 'street', 'state', and 'zip', we use frequency encoding. Because the frequency of these values can be meaningful, all classifiers can derive insightful information from it.

Then, we drop the useless columns: "trans_date_trans_time", "first", "last", "dob", "trans_num", "unix_time", "year", and "date".

Here, we decided to replace the -1 that was a filler for null values in ‘mins_since_prev_trans’ with the maximum value in this column to the power of 2. This was because we wanted the first transaction of each customer (initial nulls) to be significantly larger than all other data and have a large difference from the transactions that happened shortly after another transaction of the same customer (small ‘mins_since_prev_trans’ value). Now that the firsts are larger, even after scaling this column later, they will be far from those immediate sequential transactions.

For the final analysis, we plot the correlation matrix. This matrix shows that ‘ merch_long’ and ‘merch_lat’ are highly correlated with ‘lat’ and ‘long’ so they are probably useless variables and we drop them later. ‘Total_trans_num’ and ‘street’ are also useless variables since they are probably identical to ‘cc_num’ as they are attributes unique to each customer.

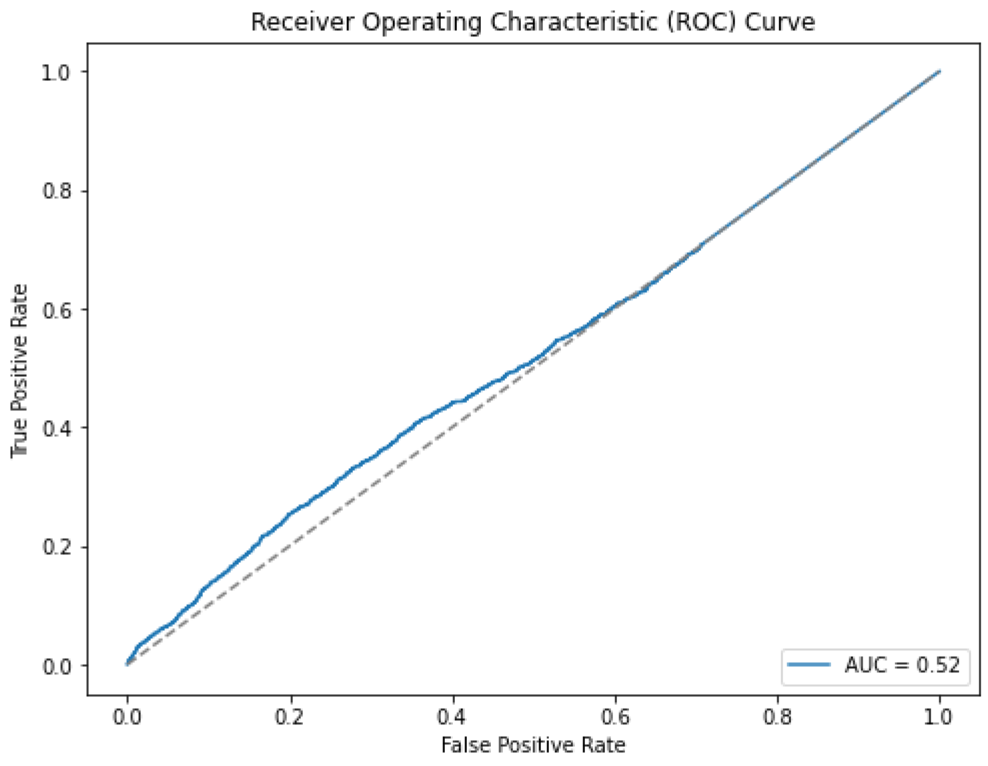


3. Model Selection and Training

In our training function, we train each of the ML models with our train data and test their accuracy with accuracy score, F1 score, AUC, and confusion matrix.

The results for the logistic regression classifier were:

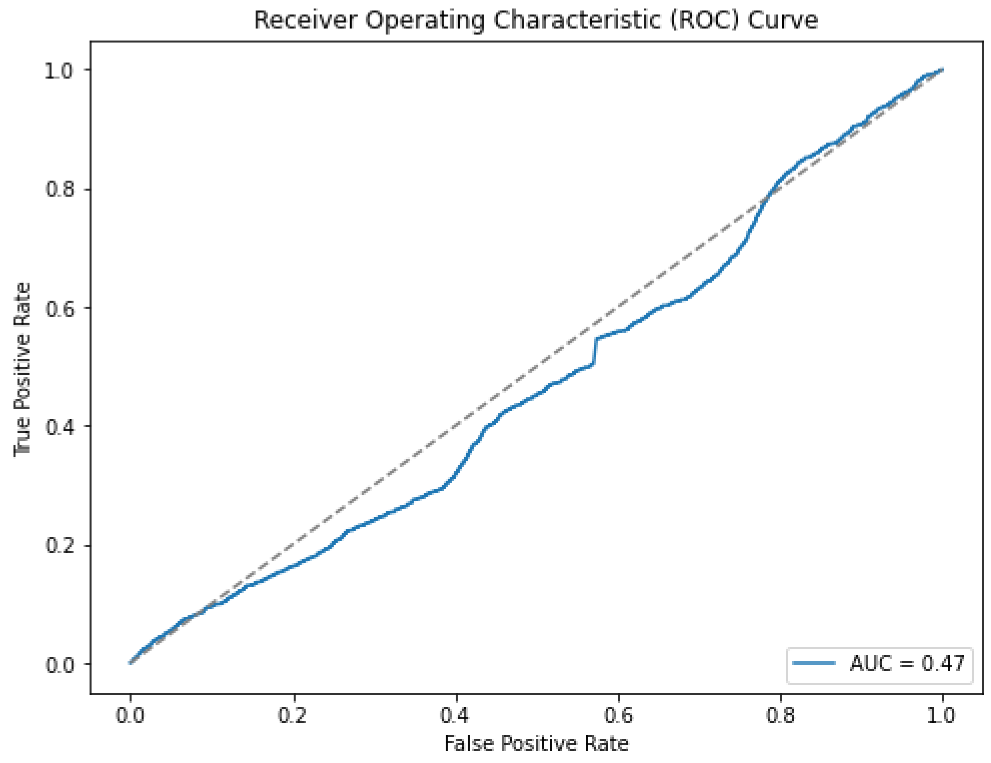
```
Training Set:
Accuracy: 0.8522553846690855
AUC: 0.8522553846690855
F1 Score: 0.8440816259706718
Confusion Matrix:
[[933044  98310]
 [206444 824910]]

Testing Set:
Accuracy: 0.9941388551487458
AUC: 0.5
F1 Score: 0.0
Confusion Matrix:
[[257815      0]
 [  1520      0]]
```



Fraud Detection

For SVM classifier:

```
Training Set:
Accuracy: 0.5000014543987806
AUC: 0.5000014543987806
F1 Score: 0.6666673130667515
Confusion Matrix:
[[       3 1031351]
 [       0 1031354]]
Testing Set:
Accuracy: 0.010145179015559025
AUC: 0.4988845658402547
F1 Score: 0.011627727895766275
Confusion Matrix:
[[   1121 256694]
 [     10   1510]]
 [     10   1510]]
```



For the KNN classifier we also did random resampling to reduce the dimension of data and we got:

```
Training Set:
Accuracy: 0.9847473676726283
AUC: 0.9847473676726283
F1 Score: 0.9848430105869298
Confusion Matrix:
```

Fraud Detection

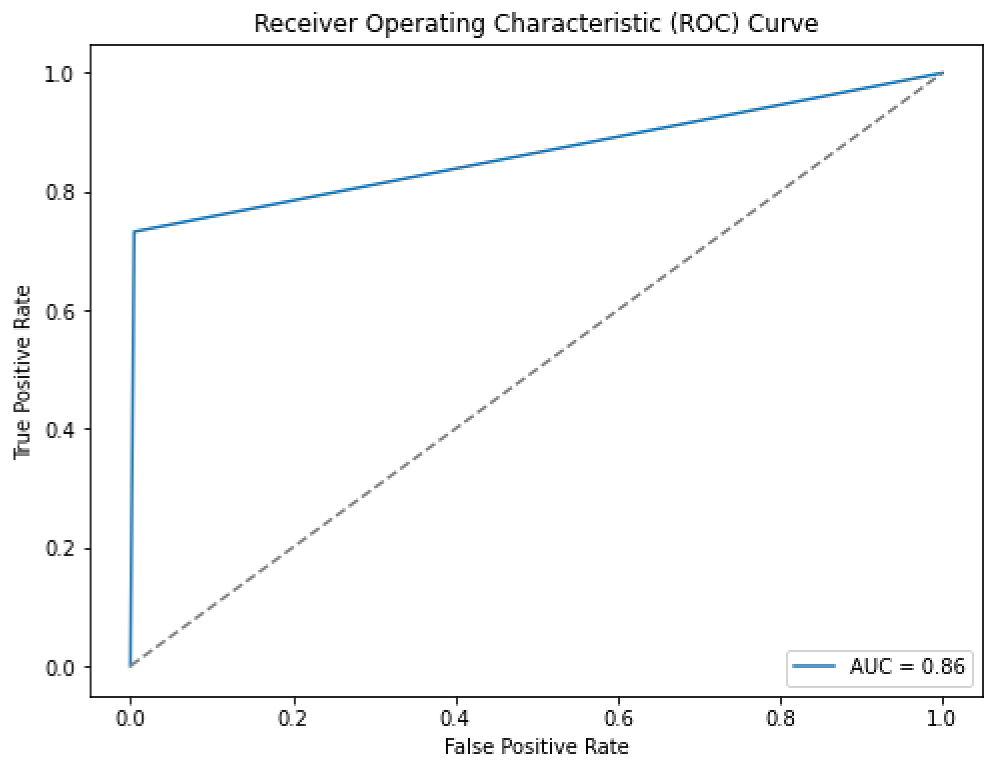
```
[[302750   6672]
 [  2767 306655]]
Testing Set:
Accuracy: 0.9492124086606127
AUC: 0.5542512187441627
F1 Score: 0.03445495198299245
Confusion Matrix:
[[245929  11886]
 [  1285    235]]
```

We did not plot the AUC curve but looking at the score of 0.5, we know that the the KNN model has a performance of a random classifier.

For decision tree:

```
Training Set:
Accuracy: 1.0
AUC: 1.0
F1 Score: 1.0
Confusion Matrix:
[[1031354       0]
 [      0 1031354]]
Testing Set:
Accuracy: 0.9938226618080861
AUC: 0.8638008677682998
F1 Score: 0.5815047021943573
Confusion Matrix:
[[256620   1195]
 [   407   1113]]
```

Fraud Detection



For random forest classifier:

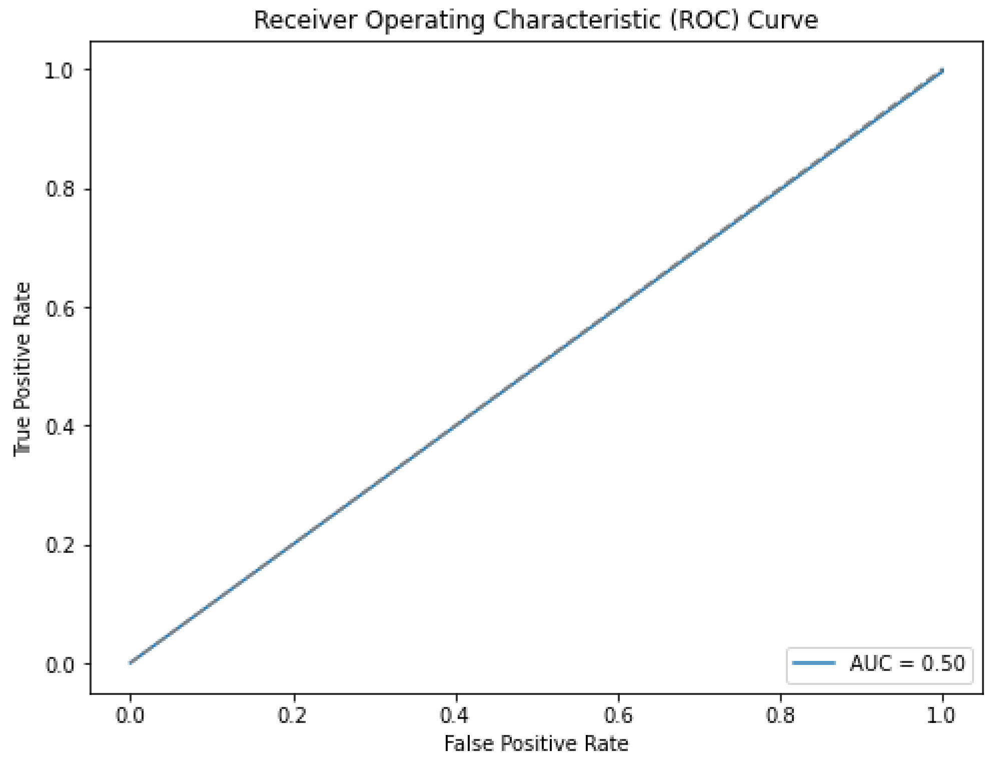
```
Training Set:
Accuracy: 1.0
AUC: 1.0
F1 Score: 1.0
Confusion Matrix:
[[1031354       0]
 [      0 1031354]]

Testing Set:
Accuracy: 0.9915630362272736
AUC: 0.5013205613572359
F1 Score: 0.007259528130671507
Confusion Matrix:
[[257139    676]
 [  1512      8]]
```

Fraud Detection

Even though the accuracy on the test set for random forest is high, the F1 score is very low, and the AUC score is 0.5 which means the model's performance was completely random. Thus, we know that the KNN model is not the best model for this task.

For naive Bayes classifier:

```
Training Set:
Accuracy: 0.789927609724692
AUC: 0.789927609724692
F1 Score: 0.7633502489279387
Confusion Matrix:
[[930523 100831]
 [332487 698867]]

Testing Set:
Accuracy: 0.006960109510864 3265
AUC: 0.4982636659599855 6
F1 Score: 0.011613626245413653
Confusion Matrix:
[[   292 257523]
 [     7   1513]]
```



After comparing the results, we can infer that the best model to classify fraud transactions is a decision tree, without using StandadScaler on all variables. In this particular problem, it was important to consider metrics beyond accuracy due to the inherent class imbalance in the dataset. Accuracy alone can be misleading in such cases. So, we had to look at F1, AUC, and confusion matrix to be able to select the best model.

After that, we create a pipeline using all the preprocessing steps and the model to easily use this workflow on new data.