Fraud Detection On Bank Transactions

Abstract:
Our research aims to develop targeted classification models such as SVM, decision trees, and random forest to assess their efficacy in distinguishing between fraudulent and non-fraudulent transactions. We commenced our investigation with a thorough analysis of the dataset, and feature engineering, followed by efforts to address data imbalances. Subsequently, we meticulously prepared the data for model training. To gauge the performance of these models, we conducted comprehensive evaluations using various techniques. Our overarching objective is to optimize the models and achieve superior results throughout this iterative process.

Introduction:

In the dynamic landscape of financial transactions, the imperative to detect and combat fraudulent activities has led to the exploration of advanced models. This research focuses on the classification of fraudulent transactions, utilizing diverse models such as Support Vector Machines, decision trees, and random forests. these included deriving new features from the dataset, encoding, scaling, resampling, feature selection methods, and pipelines. The output is reported in various ways including decision boundary, f1 score, accuracy score, and confusion matrix, a table for the final accuracy of the models is represented in the conclusion section.

Methods:
First step into knowing the dataset is loading it into a dataframe a getting its shape:
(1296675, 23)
Meaning we have 23 features and 1296675 data samples.
Now let's see what features we have:

```
Index(['Unnamed: 0', 'trans_date_trans_time', 'cc_num', 'merchant', 'category',
       'amt', 'first', 'last', 'gender', 'street', 'city', 'state', 'zip',
       'lat', 'long', 'city_pop', 'job', 'dob', 'trans_num', 'unix_time',
       'merch_lat', 'merch_long', 'is_fraud'],
      dtype='object')
```

ed: 0	trans_date_trans_time	cc_num	merchant	category	amt	first	last	gender	street	city	state	zip	lat	long	city_pop	job	dob	trans_num	unix_time	merch_lat	merch_long	is_fraud
0	2019-01-01 00:00:18	2703186189652095	fraud_Rippin, Kub and Mann	misc_net	4.97	Jennifer	Banks	F	561 Perry Cove	Moravian Falls	NC	28654	36.0788	-81.1781	3495	Psychologist, counselling	1988-03-09	0b242abb623afc578575680df30655b9	1325376018	36.011293	-82.048315	0
1	2019-01-01 00:00:44	630423337322	fraud_Heller, Gutmann and Zieme	grocery_pos	107.23	Stephanie	Gill	F	43039 Riley Greens Suite 393	Orient	WA	99160	48.8878	-118.2105	149	Special educational needs teacher	1978-06-21	1f76529f8574734946361c461b024d99	1325376044	49.159047	-118.186462	0
2	2019-01-01 00:00:51	38859492057661	fraud_Lind-Buckridge	entertainment	220.11	Edward	Sanchez	M	594 White Dale Suite 530	Malad City	ID	83252	42.1808	-112.2620	4154	Nature conservation officer	1962-01-19	a1a22d70485983eac12b5b88dad1cf95	1325376051	43.150704	-112.154481	0
3	2019-01-01 00:01:16	3534093764340240	fraud_Kutch, Hermiston and Farrell	gas_transport	45.00	Jeremy	White	M	9443 Cynthia Court Apt. 038	Boulder	MT	59632	46.2306	-112.1138	1939	Patent attorney	1967-01-12	6b849c168bdad6f867558c3793159a81	1325376076	47.034331	-112.561071	0
4	2019-01-01 00:03:06	375534208663984	fraud_Keeling-Crist	misc_pos	41.96	Tyler	Garcia	M	408 Bradley Rest	Doe Hill	VA	24433	38.4207	-79.4629	99	Dance movement psychotherapist	1986-03-28	a41d7549acf90789359a9aa5346dcb46	1325376186	38.674999	-78.632459	0

To start our preprocessing let's drop any duplicated data and get the shape again.
After getting the shape of the data we can observe number of samples remained the same meaning we didn't had any duplicated data.
Next let's check for any missing values.

```
Unnamed: 0              0
trans_date_trans_time    0
cc_num                   0
merchant                 0
category                 0
amt                      0
first                    0
last                     0
gender                   0
street                   0
city                     0
state                    0
zip                      0
lat                      0
long                     0
city_pop                 0
job                      0
dob                      0
trans_num                0
```

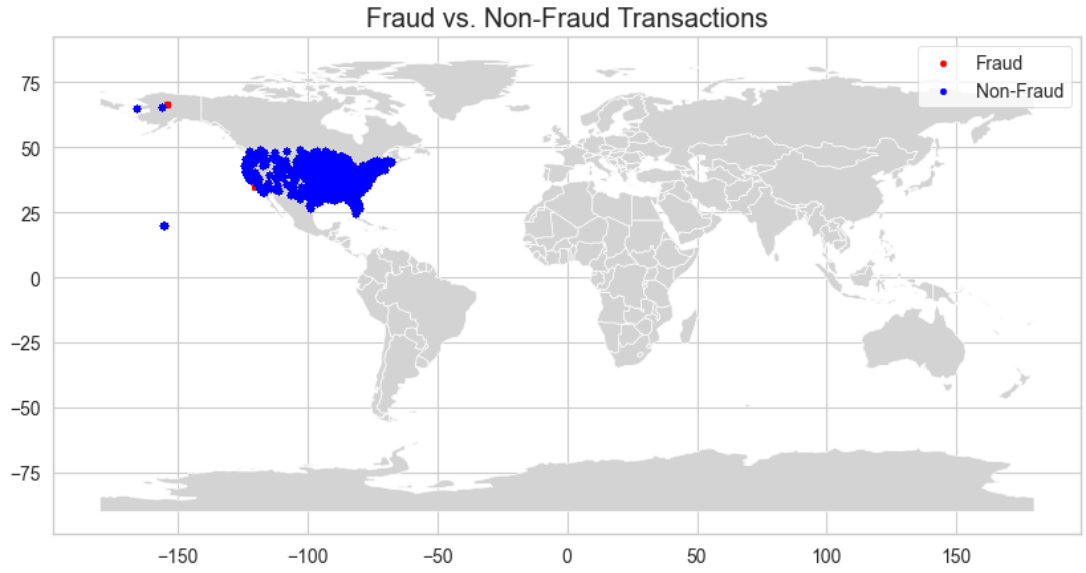
unix_time 0
merch_lat 0
merch_long 0
is_fraud 0
dtype: int64

As it seems like. We don't have any missing value so our data is clean.
The features obviously nedd some feature engineering methods. So first we divided the date and time from the trans_date_trans_time column then also derived month and year from the date column.
We can also introduce new features such as:
Trans_date_trans_time : number of transactions belonging to a unique costumer
Distance_to_merchant : derived from the difference of longitude and latitude of costumer and the merchant.
Is_weekend: to see if the day of the week is > 5.
Days_since_last_transaction : count the number of days past the last transaction of the costumer. First transactions of a costumer is returned as Nan, which we handle later.
The number of these nan datas are 983. Also we obtained a new column which is the categorical version of this column. Values <= 2 are labeled as recent, values <= 6 are labeled moderate, values <= 10 are labeled long and values <= 15 are labeled very long ago.
Birth_month : derived from dob column

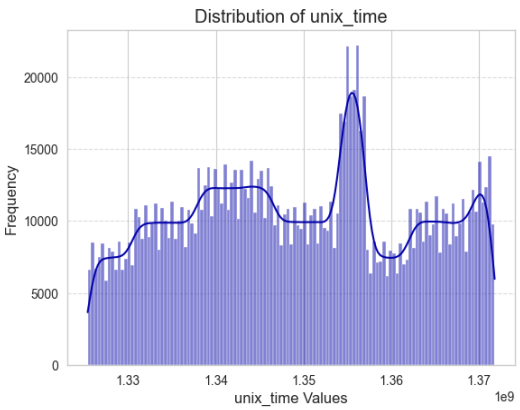
Now let's drop the unneccary columns including :
["days_since_last_transaction", "trans_num", "dob", "date", "time", "trans_date_trans_time", "cc_num", "Unnamed: 0", "first", "last"]

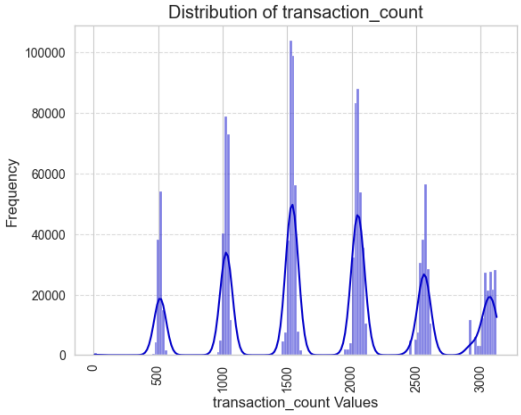
Visualization:

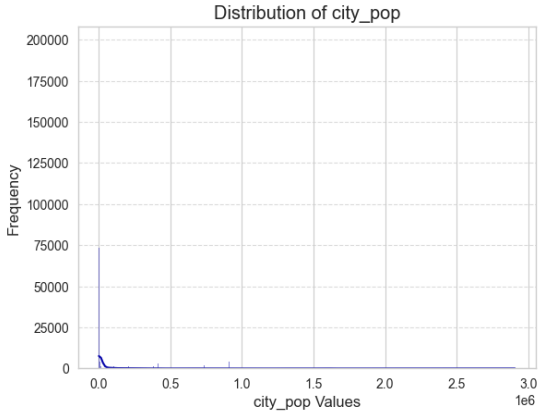
Let's plot the sparsity of fraud and non fraud data samples in a world map:



What this plot tells us is that the data belong to the US only with some outliers. And as it shows the data is unbalanced.
Now let's take a look at some univariable distribution plots:

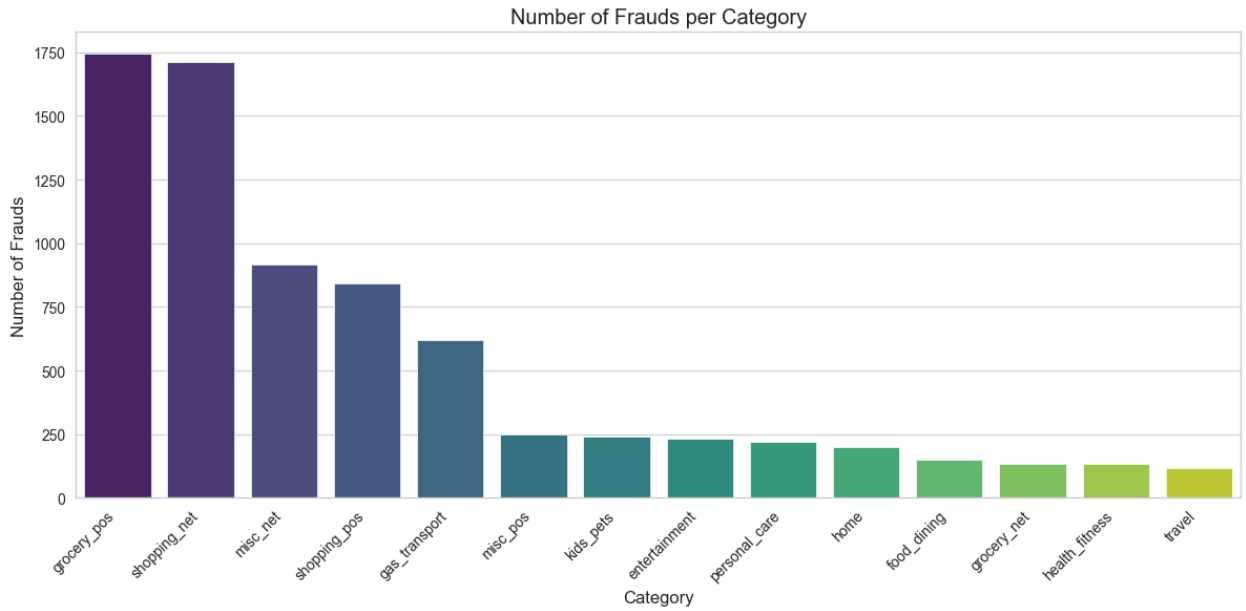




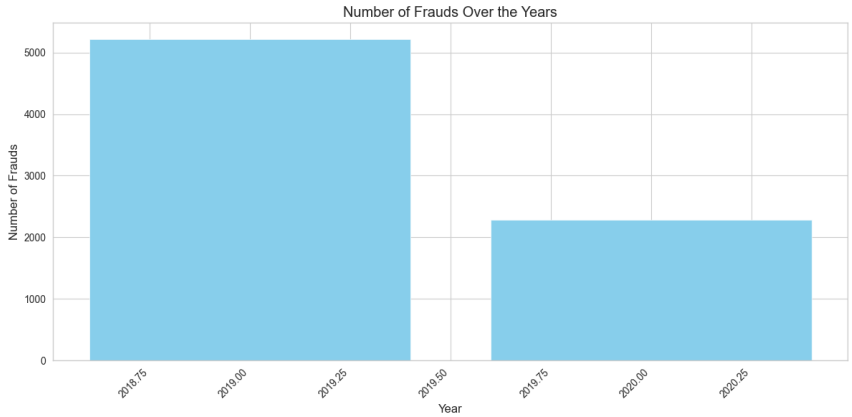


As it shows, none follow a normal distribution.

This plot which is fraud frequency per state shows most transactions happening in DE are the most fraudulent with a very high difference.

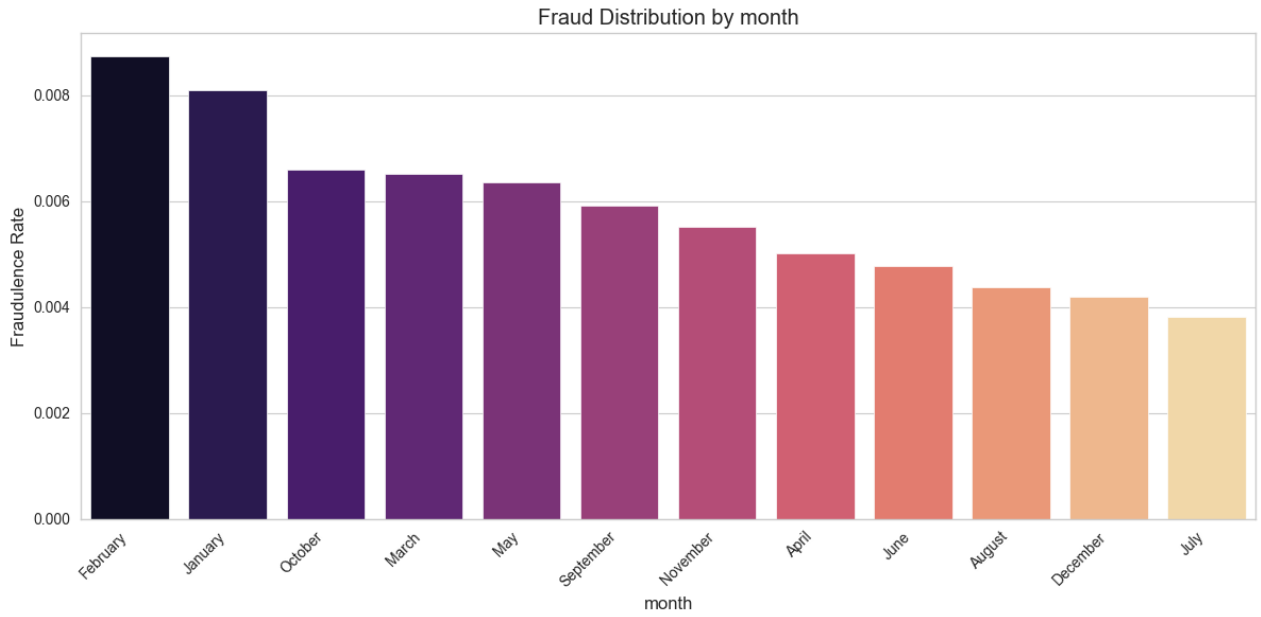


This plot is also fraud == 1 frequency per each category which indicates most frauds happen in the grocery_pos category.

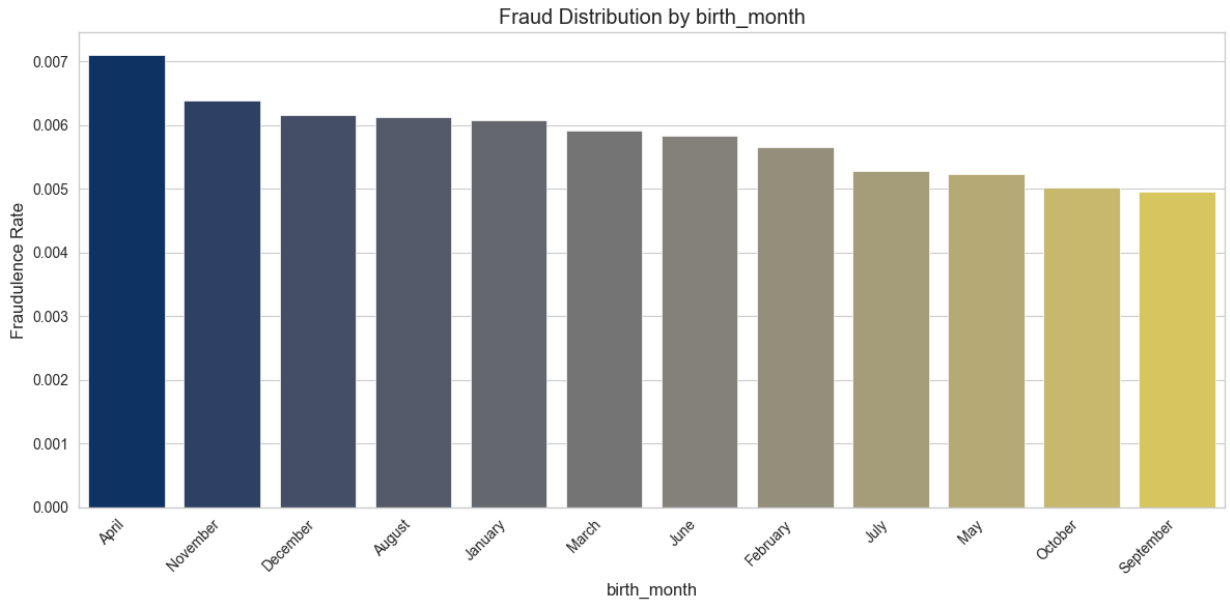


Since our data are only from 2019 and 2020. We only have two bars for frauds == 1 per year. Which itself indicates a high reduced amount of frauds in 2020 in comparison to 2019.

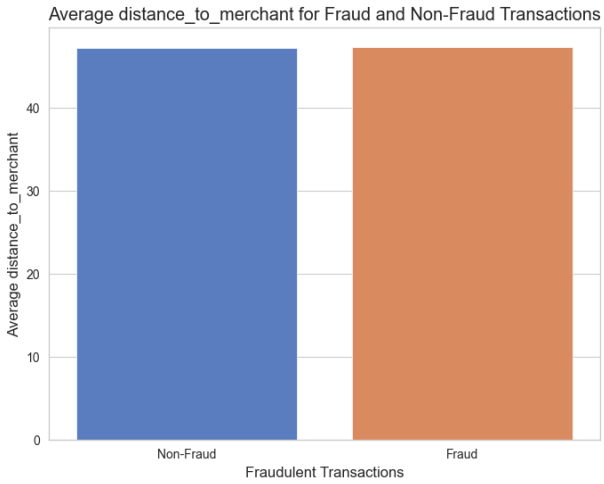
By calculating frequency of Fraud transaction per whole Transaction for male and Female, it shows more fraud Transactions belong to males.

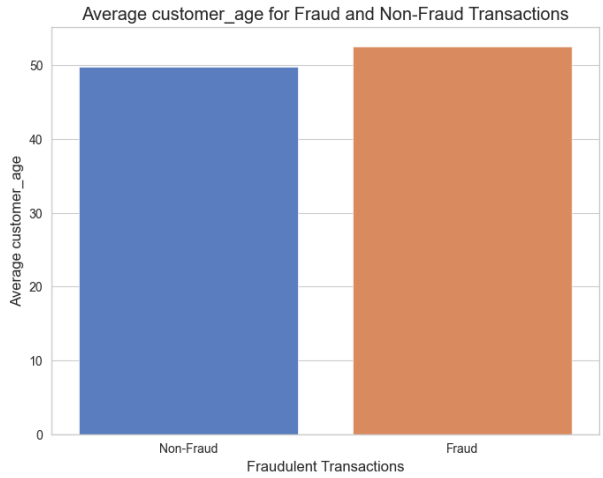


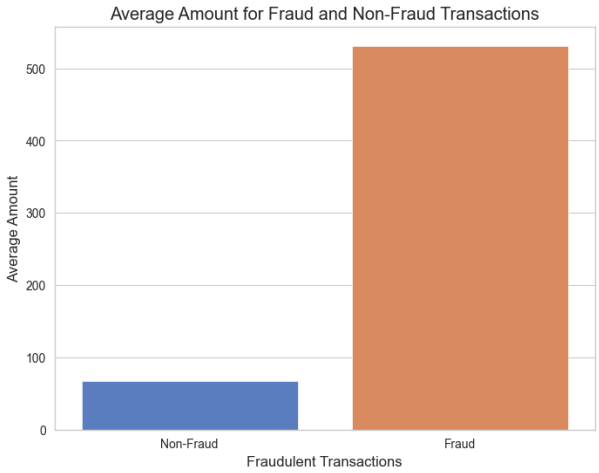
Also most frauds happened in February.

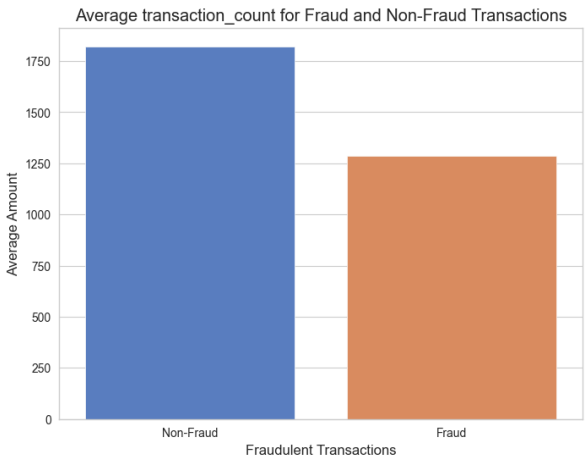


This also shows april-borned people had the most fraudulent transactions.



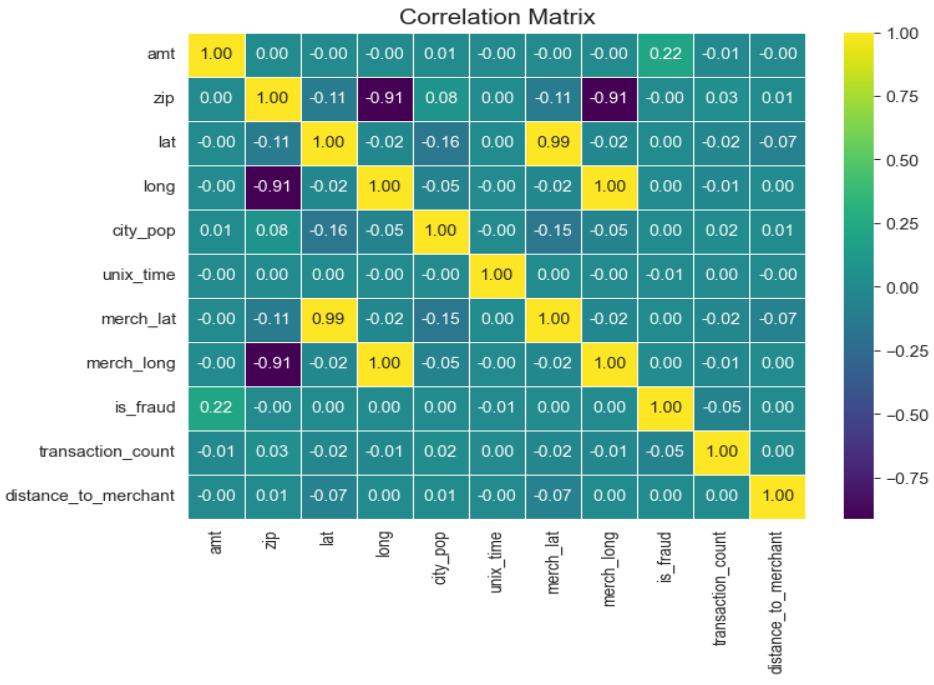






Above plots only show a difference in amount and transaction_count over fraud and non-fraud data. This means usually higher amount of transactions have more chance of being fraudulent and people who have fraud transactions have less transactions recorded.

Also corr matrix for this df looks like
This plot suggests a high correlation between amt and is_fraud.



In this section after finishing all preprocessing steps, it's time to encode categorical data. For most columns we choose one hot encoding including : gender, state, job and category. For month and birth_month we choose cycle encoding and for gap_from_last_transaction, ordinal encoding (we assigned -1 to Nan values). Other features containing 'merchant', 'city', 'street' were dropped from the dataframe.
(features like job which had multiple values were separated at first.)
Now that our data is properly encoded, it's tile to separate X and y and then separate them into train and test using scikit-learn train_test_split.
Test data was considered 20 percent of the whole data.
Next step is to perform a kind of scaling method to the train and test X. which is in this case min max scaler.
Now in order to use a feature selection method after remaining with 651 features. We chose PCA only to receive the 50 most important components.
And now the last step is to handle imbalancement of the data:
(is_fraud
0 1289169
1 7506
Name: count, dtype: int64)
is to use a resampling method. Here we chose SMOTE.

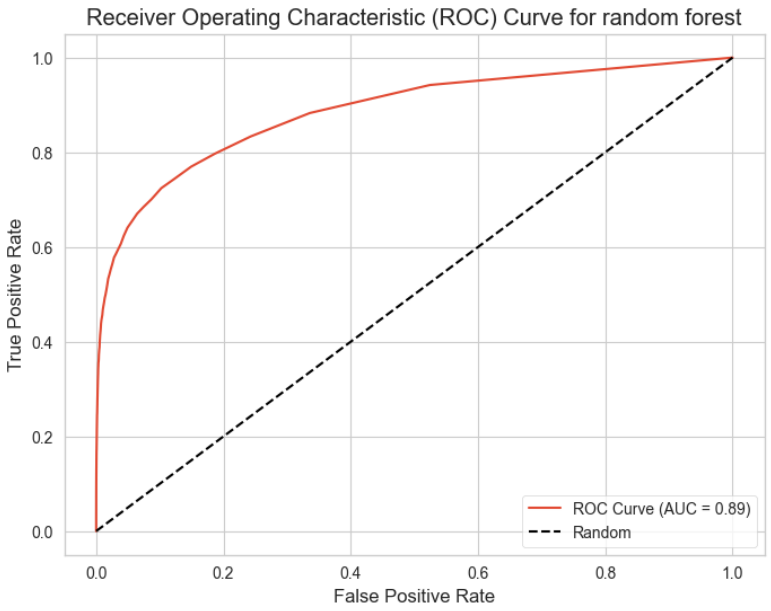
Now that our data is ready. Next step is to choose various classifiers to chose the best classification model for this dataset. Below is the result of training each model:

Random forest:

For test data:
Random Forest Accuracy: 0.9935990128598146

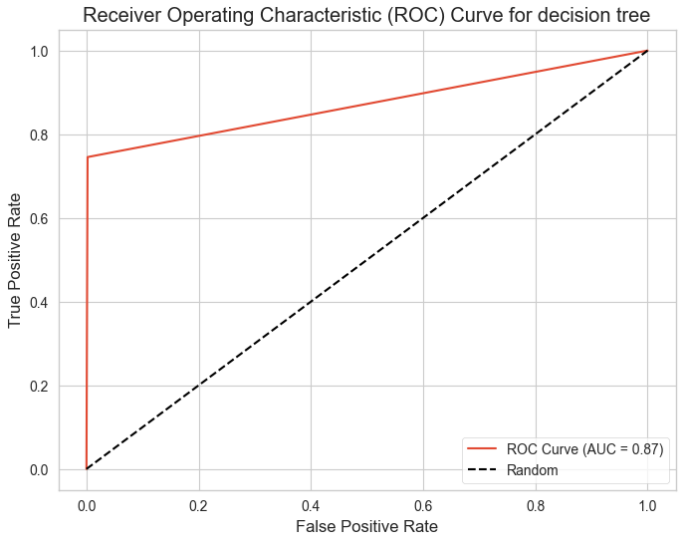
Confusion matrix:
[[257220 595]
[1065 455]]

	precision	recall	f1-score	support
0	1.00	1.00	1.00	257815
1	0.43	0.30	0.35	1520
accuracy			0.99	259335
macro avg	0.71	0.65	0.68	2593
weighted avg	0.99	0.99	0.99	259



Decision tree:

For test data:
Accuracy on the test set: 0.9964987371546455



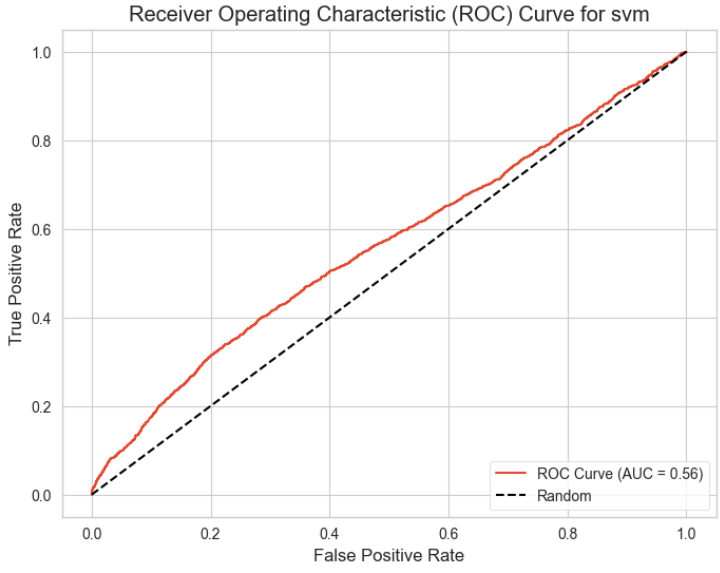
Confusion Matrix:
[[257294 521]
[387 1133]]

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	257815
1	0.69	0.75	0.71	1520
accuracy			1.00	259335
macro avg	0.84	0.87	0.86	259335
weighted avg	1.00	1.00	1.00	259335

Accuracy on the training set subset: 1.0

SVM:
Model was trained with rbf kernel and max_iter=500
accuracy svm: 0.9233115468409586

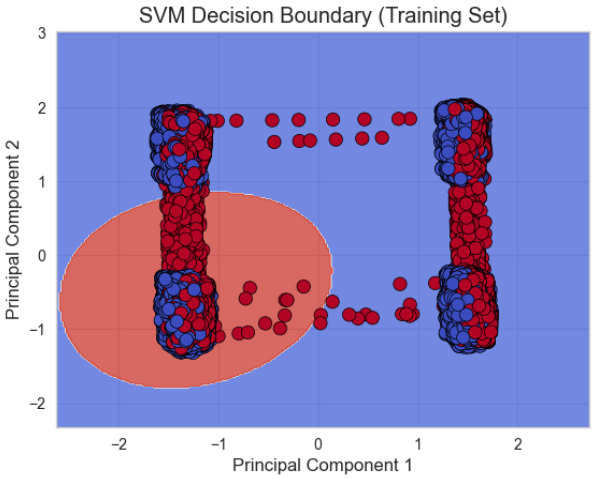


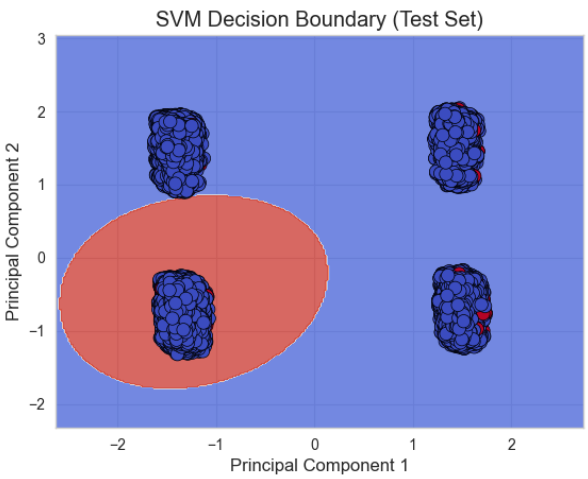
Confusion matrix:
[[239257 18558]
[1330 190]]

	precision	recall	f1-score	support
0	0.99	0.93	0.96	257815
1	0.01	0.12	0.02	1520
accuracy			0.92	259335
macro avg	0.50	0.53	0.49	259335
weighted avg	0.99	0.92	0.95	259335

accuracy svm on train: 0.5728060968485557

Data was transformed into 2 components so we can observe the decision boundary of svm:





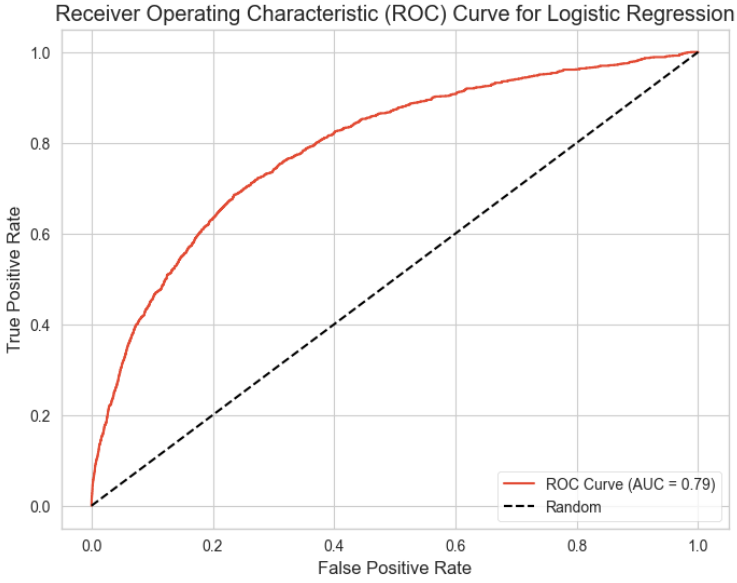
Test accuracy with this 2 featured model:
accuracy svm: 0.6086336206065514

Logistic regression:

Accuracy: 0.7311315479977635

Confusion matrix:
[[188522 69293]
[434 1086]]

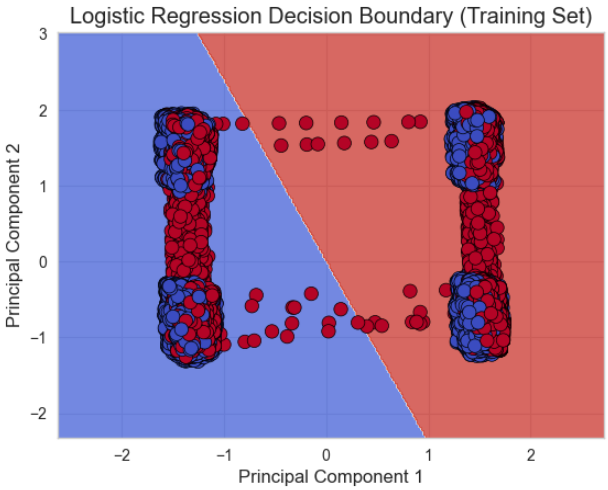
	precision	recall	f1-score	support
0	1.00	0.73	0.84	257815
1	0.02	0.71	0.03	1520
accuracy			0.73	259335
macro avg	0.51	0.72	0.44	259335
weighted avg	0.99	0.73	0.84	259335

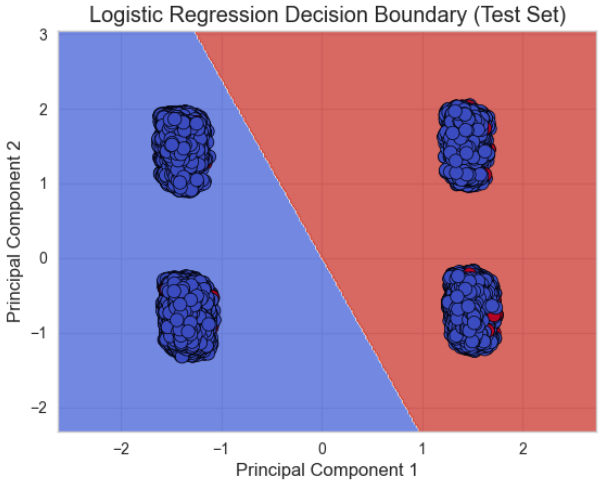


Accuracy on the train set: 0.7390974472840275
Confusion Matrix:
[[150806 55499]
[52134 154102]]
Classification Report:

	precision	recall	f1-score	support
0	0.74	0.73	0.74	206305

1	0.74	0.75	0.74	206236
accuracy			0.74	412541
macro avg	0.74	0.74	0.74	412541
weighted avg	0.74	0.74	0.74	412541





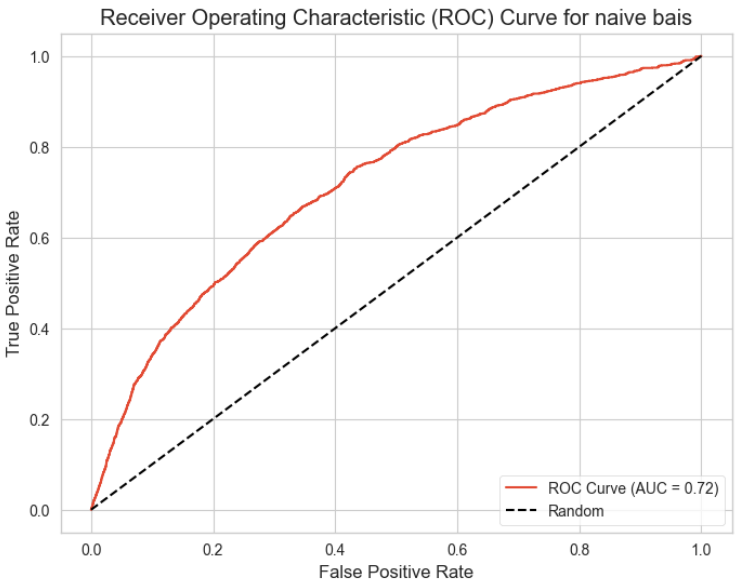
Test accuracy with this 2 featured model:
accuracy logreg: 0.5460658993194131

Naive bais:

Naive Bayes Accuracy: 0.6352362774018162

Confusion matrix:
[[163705 94110]
[486 1034]]

	precision	recall	f1-score	support
0	1.00	0.63	0.78	257815
1	0.01	0.68	0.02	1520
accuracy			0.64	259335
macro avg	0.50	0.66	0.40	259335
weighted avg	0.99	0.64	0.77	259335



On train:
Naive Bayes Accuracy: 0.6882225039450625
[[130872 75433]

[53188 153048]]

	precision	recall	f1-score	support
0	0.71	0.63	0.67	206305
1	0.67	0.74	0.70	206236
accuracy			0.69	412541
macro avg	0.69	0.69	0.69	412541
weighted avg	0.69	0.69	0.69	412541

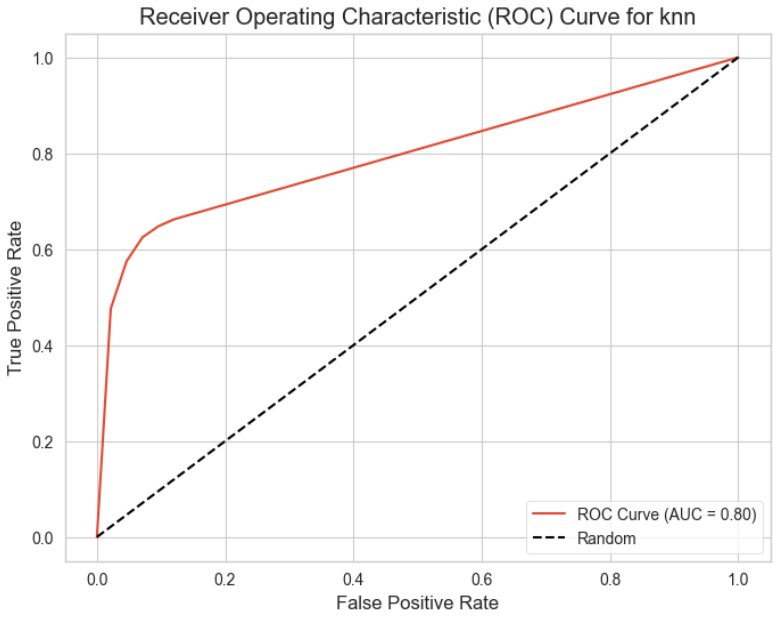
KNN:

accuracy knn is: 0.9270403146509342

Confusion Matrix:
[[239464 18351]
[570 950]]

Classification Report:

	precision	recall	f1-score	support
0	1.00	0.93	0.96	257815
1	0.05	0.62	0.09	1520
accuracy			0.93	259335
macro avg	0.52	0.78	0.53	2593
weighted avg	0.99	0.93	0.96	259



For train:
accuracy knn is: 0.9757745290771099

As we observed through this process decision tree was rather the best classifier for this task since it has very higher accuracy and the best precision. And among them, SVM performed the poorest.
We wanted to visualize decision boundaries for a few of them so we chose SVM and logistic regression.
But to do so we first transformed the data into a 2D feature space using the help of PCA. As it was seen the accuracy of the model trained with only 2 features is a lot less than the original one which is trained with 50 features.
After this training process, we created a pipeline available in the notebook which performs all the feature transformation steps needed to be run on the dataset before applying the classifier model. It contains four steps, preprocessor, PCA, smote, and classifier, in which preprocessor refers to the encoding process. And classifier is dt as the model containing the best result.

Conclusion:

	Log reg	svm	rf	dt	knn	nb
Test accuracy	0.73	0.92	0.99	0.99	0.92	0.63

However we achieved a good accuracy in some of these models >= 90. But since we observed a low precision for most of them except the decision tree, we can confidently state that our final best model trained on this dataset is the decision tree.