Data Science

Assignment 4 – Fraud Detection

Farimah Rashidi – 99222040

1. Introduction

This is a simulated credit card transaction dataset containing legitimate and fraud transactions from the duration 1st Jan 2019 - 31st Dec 2020. It covers credit cards of 1000 customers doing transactions with a pool of 800 merchants.

Fraud detection is a critical component in safeguarding the financial integrity of businesses and organizations. The evolving landscape of technology has given rise to increasingly sophisticated fraudulent activities, necessitating the application of advanced methodologies for effective detection. Data science offers a promising avenue for building models capable of accurately predicting fraudulent transactions.

The primary objective is to develop models capable of predicting whether a given transaction is fraudulent or not.

link: Fraud Detection Dataset

This dataset has 22 columns and 1296675 rows.

trans_date_trans_time: transaction time

cc_num: credit card number

merchant: merchant location

category: transaction category

amt: transaction amount

first:

last:

gender: the gender of the person who did transaction.

Street: the street where the transaction occurs

City: the city where the transaction occurs

State: the state where the transaction occurs

zip

lat: transaction latitude

long: transaction longitude

city_pop: the population of the city where transaction occurs

job: the job of the person who did transaction

dob: date of birth

trans_num: the number of the transaction

unix_time

merch_lat: merchant latitude

merch_long: merchant longitude

is_fraud: is the transaction fraud or not?

2. Data preprocessing

Let's look at some first few rows of dataset:

	trans_date_trans_time	cc_num	merchant	category	amt	first	last	gender	street	city	...	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	...	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	...	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	...	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	...	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	...	38.4207	-79.4629	99	Dance movement psychotherapist	1986-03-28	a41d7549acf90789359a9aa5346dcb46	1325376186	38.674999	-78.632459	0
5	2019-01-01 00:04:08	4767265376804500	fraud_Stroman, Hudson and Erdman	gas_transport	94.63	Jennifer	Conner	F	4655 David Island	Dublin	...	40.3750	-75.2045	2158	Transport planner	1961-06-19	189a841a0a8ba03058526bcfe566aab5	1325376248	40.653382	-76.152667	0
6	2019-01-01 00:04:42	30074693890476	fraud_Rowe-Vandervort	grocery_net	44.54	Kelsey	Richards	F	889 Sarah Station Suite 624	Holcomb	...	37.9931	-100.9893	2691	Arboriculturist	1993-08-16	83ec1cc84142af6e2acf10c44949e720	1325376282	37.162705	-100.153370	0
7	2019-01-01 00:05:08	6011360759745864	fraud_Corwin-Collins	gas_transport	71.65	Steven	Williams	M	231 Flores Pass Suite 720	Edinburg	...	38.8432	-78.6003	6018	Designer, multimedia	1947-08-21	6d294ed2cc447d2c71c7171a3d54967c	1325376308	38.948089	-78.540296	0
8	2019-01-01 00:05:18	4922710831011201	fraud_Herzog Ltd	misc_pos	4.27	Heather	Chase	F	6888 Hicks Stream Suite 954	Manor	...	40.3359	-79.6607	1472	Public affairs consultant	1941-03-07	fc28024ce480f8ef21a32d64c93a29f5	1325376318	40.351813	-79.958146	0
9	2019-01-01 00:06:01	2720830304681674	fraud_Schoen, Kuphal and Nitzsche	grocery_pos	198.39	Melissa	Aguilar	F	21326 Taylor Squares Suite 708	Clarksville	...	36.5220	-87.3490	151785	Pathologist	1974-03-28	3b9014ea8fb80bd65de0b1463b00b00e	1325376361	37.179198	-87.485381	0

Our dataset has 22 columns and 1296675 rows.

Now let's see data types:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1296675 entries, 0 to 1296674
Data columns (total 22 columns):
 #   Column                 Non-Null Count    Dtype
---  ------                 --------------    -----
 0   trans_date_trans_time  1296675 non-null  object
 1   cc_num                 1296675 non-null  int64
 2   merchant               1296675 non-null  object
 3   category               1296675 non-null  object
 4   amt                    1296675 non-null  float64
 5   first                  1296675 non-null  object
 6   last                   1296675 non-null  object
 7   gender                 1296675 non-null  object
 8   street                 1296675 non-null  object
 9   city                   1296675 non-null  object
 10  state                  1296675 non-null  object
 11  zip                    1296675 non-null  int64
 12  lat                    1296675 non-null  float64
 13  long                   1296675 non-null  float64
 14  city_pop               1296675 non-null  int64
 15  job                    1296675 non-null  object
 16  dob                    1296675 non-null  object
 17  trans_num              1296675 non-null  object
 18  unix_time              1296675 non-null  int64
 19  merch_lat              1296675 non-null  float64
 20  merch_long             1296675 non-null  float64
 21  is_fraud               1296675 non-null  int64
dtypes: float64(5), int64(5), object(12)
memory usage: 217.6+ MB
```

Dataset has two parts: test and train. At the first we do some preprocessing on both parts. Then we concatenate these two parts.

Convert datetime columns:

I took a crucial step by converting date-related columns, "trans_date_trans_time" and "dob," into a datetime format for both the training and testing datasets.

I introduced a valuable feature, "hour_of_day," derived from the transaction time, offering insights into temporal patterns.

Dropping duplicated values:

This step ensures that the training and testing datasets maintain uniqueness, eliminating any duplicate records.

Convert dtypes:

For the training dataset, I converted 'cc_num,' 'is_fraud,' and 'hour_of_day' columns to the 'category' data type. This dtype conversion is beneficial for categorical variables, as it optimizes memory usage and facilitates categorical data manipulation. I repeated the same dtype conversions for the corresponding columns in the testing dataset.

Data Cleaning

I conducted a comprehensive check for missing values in both the training and testing datasets.

```
df_test.isnull().sum()
```

```
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
time                     0
hour_of_day              0
dtype: int64
```

```
df_train.isnull().sum()
```

```
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
time                     0
hour_of_day              0
dtype: int64
```

It seems that there is no missing value in dataset.

3. EDA & Feature Engineering

3.1 Numerical Variable Analysis

At the first I combined the training and testing datasets into a single DataFrame named ‘total’ using the concat function.

Then I created a new categorical column named "is_fraud_category" in the ‘total’ DataFrame based on the existing "is_fraud" column. The new column is assigned the value "T" when "is_fraud" is equal to 1 and "F" otherwise.

3.1.1. Finding numerical variable

I selected the numerical columns from the ‘total’ DataFrame and stored them in the ‘totalnum’ DataFrame.

	cc_num	amt	zip	lat	long	city_pop	unix_time	merch_lat	merch_long
0	2291163933867244	2.86	29209	33.9659	-80.9355	333497	1371816865	33.986391	-81.200714
1	3573030041201292	29.84	84002	40.3207	-110.4360	302	1371816873	39.450498	-109.960431
2	3598215285024754	41.28	11710	40.6729	-73.5365	34496	1371816893	40.495810	-74.196111
3	3591919803438423	60.05	32780	28.5697	-80.8191	54767	1371816915	28.812398	-80.883061
4	3526826139003047	3.19	49632	44.2529	-85.0170	1126	1371816917	44.959148	-85.884734
...	...	...	...	...	...	...	...	...	...
1296670	30263540414123	15.56	84735	37.7175	-112.4777	258	1371816728	36.841266	-111.690765
1296671	6011149206456997	51.70	21790	39.2667	-77.5101	100	1371816739	38.906881	-78.246528
1296672	3514865930894695	105.93	88325	32.9396	-105.8189	899	1371816752	33.619513	-105.130529
1296673	2720012583106919	74.90	57756	43.3526	-102.5411	1126	1371816816	42.788940	-103.241160
1296674	4292902571056973207	4.30	59871	45.8433	-113.8748	218	1371816817	46.565983	-114.186110

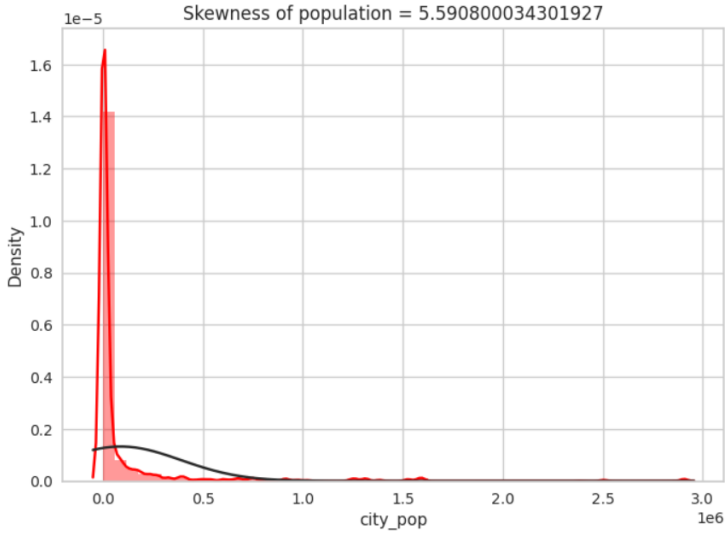
We have these 9 numerical columns.

3.1.2. cleaning numerical variable

Now we try to find missing values in numerical columns, but it seems that there is no missing value.

3.1.3. Skewness of population

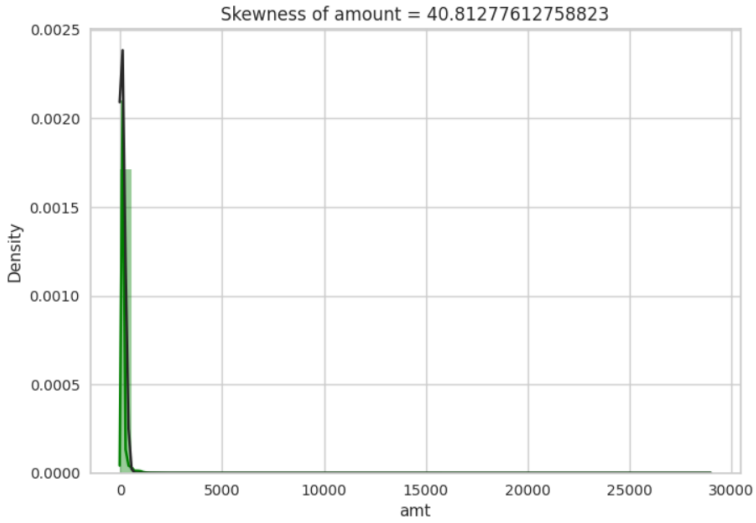
I calculated the skewness of the 'city_pop' and then created a distribution plot.



A skewness value of 5.59 indicates a right-skewed distribution.

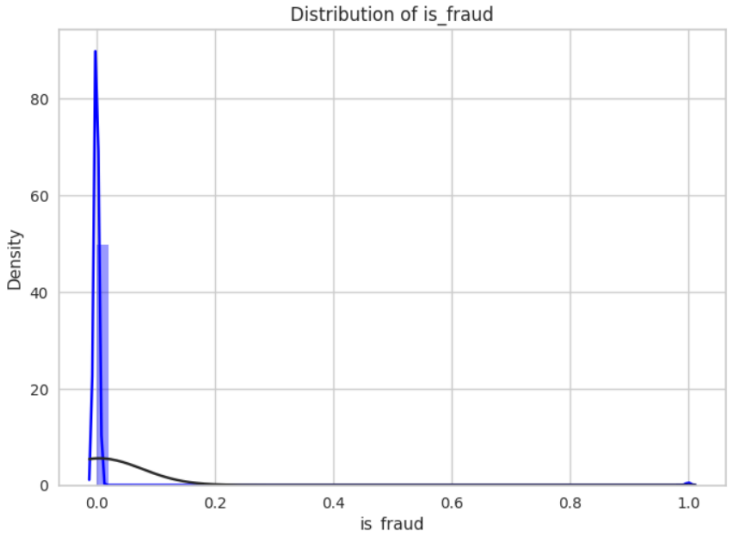
Here, the right tail is longer or fatter than the left tail, and the majority of the data points are concentrated on the left side. this implies that there are few cities with very large populations.

3.1.4. Skewness of amount



The skewness value of 40.81 for the amount variable indicates a highly right-skewed distribution. As you can see, a right-skewed distribution implies a long tail to the right, which may be indicative of a few transactions with significantly higher amounts.

3.1.5. Distribution of is_fraud



The distribution of the 'is_fraud' variable indicates a skewness of 13.75. In the context of fraud detection, this skewness might be attributed to a smaller number of fraudulent transactions compared to non-fraudulent ones.

A lot of the data is highly skewed.

Also, we can understand that proportion of non-fraud transactions are much much larger than fraud transactions so we are looking at an imbalanced dataset.

3.1.6. Remove numerical columns

Certain numerical columns are not needed for modeling and so they can be removed.

So, we remove these columns:

'cc_num','merchant','first','last','street','zip','trans_num','unix_time'

Here is the result:

```
<class 'pandas.core.frame.DataFrame'>
Index: 1852394 entries, 0 to 1296674
Data columns (total 16 columns):
 #   Column                 Dtype         
---  ------                 -----         
 0   trans_date_trans_time  datetime64[ns]
 1   category               object        
 2   amt                    float64       
 3   gender                 object        
 4   city                   object        
 5   state                  object        
 6   lat                    float64       
 7   long                   float64       
 8   city_pop               int64         
 9   job                    object        
 10  dob                    datetime64[ns]
 11  merch_lat              float64       
 12  merch_long             float64       
 13  is_fraud               category      
 14  time                   datetime64[ns]
 15  hour_of_day            category      
dtypes: category(2), datetime64[ns](3), float64(5), int64(1), object(5)
memory usage: 215.5+ MB
```

3.2. Categorical Variable Analysis

At the first we select categorical columns and put them in ‘totalcategory’.

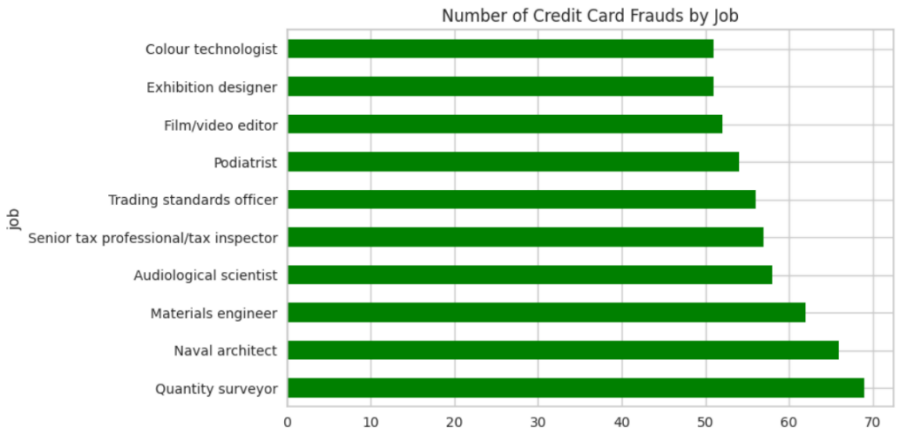
	merchant	category	first	last	gender	street	city	state	job	trans_num
0	fraud_Kirlin and Sons	personal_care	Jeff	Elliott	M	351 Darlene Green	Columbia	SC	Mechanical engineer	2da90c7d74bd46a0caf3777415b3ebd3
1	fraud_Sporer-Keebler	personal_care	Joanne	Williams	F	3638 Marsh Union	Altonah	UT	Sales professional, IT	324cc204407e99f51b0d6ca0055005e7
2	fraud_Swaniawski, Nitzsche and Welch	health_fitness	Ashley	Lopez	F	9333 Valentine Point	Bellmore	NY	Librarian, public	c81755dbbbea9d5c77f094348a7579be
3	fraud_Haley Group	misc_pos	Brian	Williams	M	32941 Krystal Mill Apt. 552	Titusville	FL	Set designer	2159175b9efe66dc301f149d3d5abf8c
4	fraud_Johnston-Casper	travel	Nathan	Massey	M	5783 Evan Roads Apt. 465	Falmouth	MI	Furniture designer	57ff021bd3f328f8738bb535c302a31b
...	...	...	...	...	...	...	...	...	...	...
1296670	fraud_Reichel Inc	entertainment	Erik	Patterson	M	162 Jessica Row Apt. 072	Hatch	UT	Geoscientist	440b587732da4dc1a6395aba5fb41669
1296671	fraud_Abernathy and Sons	food_dining	Jeffrey	White	M	8617 Holmes Terrace Suite 651	Tuscarora	MD	Production assistant, television	278000d2e0d2277d1de2f890067dcc0a
1296672	fraud_Stiedemann Ltd	food_dining	Christopher	Castaneda	M	1632 Cohen Drive Suite 639	High Rolls Mountain Park	NM	Naval architect	483f52fe67fabef353d552c1e662974c
1296673	fraud_Reinger, Weissnat and Strosin	food_dining	Joseph	Murray	M	42933 Ryan Underpass	Manderson	SD	Volunteer coordinator	d667cdcbadaaed3da3f4020e83591c83
1296674	fraud_Langosh, Wintheiser and Hyatt	food_dining	Jeffrey	Smith	M	135 Joseph Mountains	Sula	MT	Therapist, horticultural	8f7c8e4ab7f25875d753b422917c98c9

1852394 rows × 10 columns

We have 10 categorical columns.

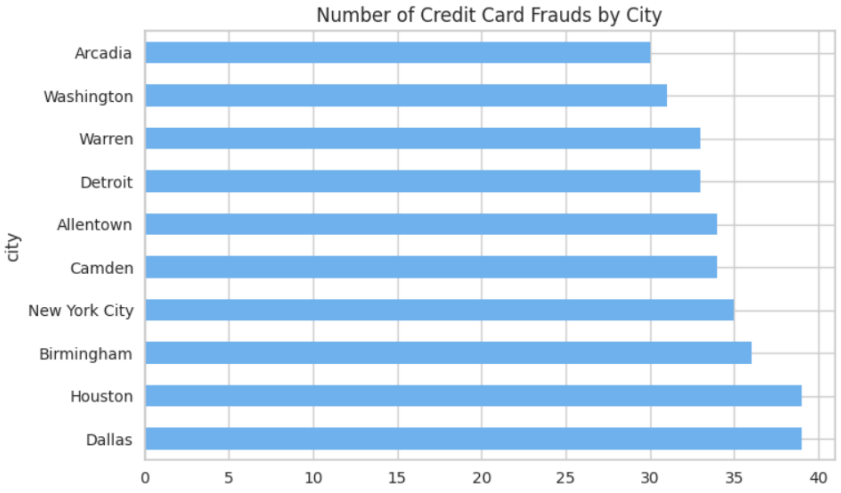
3.2.1. Number of Credit Card Frauds by Job

We visualize the number of credit card frauds based on the 'job' attribute for the category 'T' in the 'is_fraud_category' column. This bar chart displays the top 10 jobs associated with credit card fraud, with the count of occurrences.



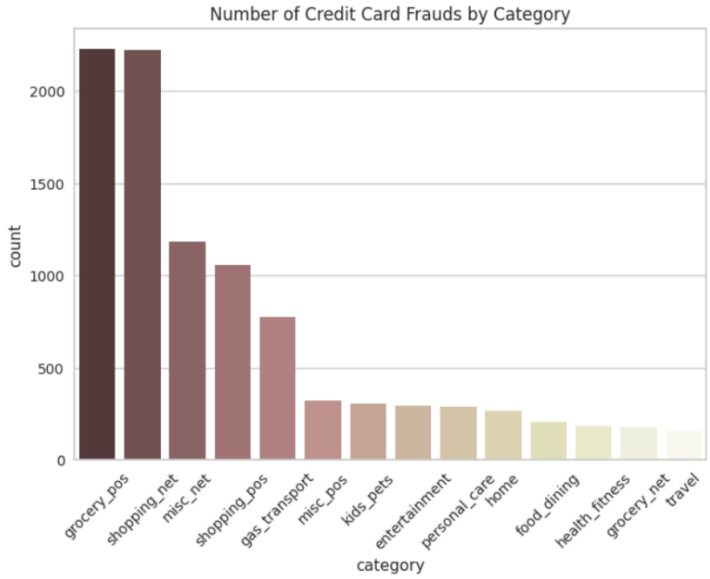
Most frauds occured in jobs of quantity surveyor followed by naval architect and materials engineer.

3.2.2. Number of Credit Card Frauds by City



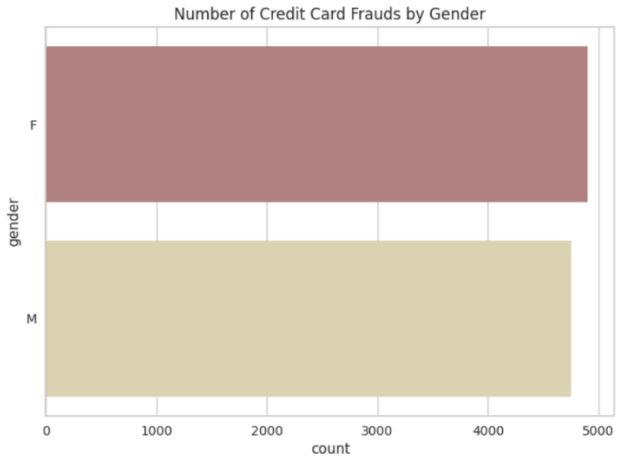
Most frauds occured in cities of dallas followed by Houston architect and Birmingham.

3.2.3. Number of Credit Card Frauds by Category



Most frauds happened in categories of shopping_net, grocery_pos and misc_net.

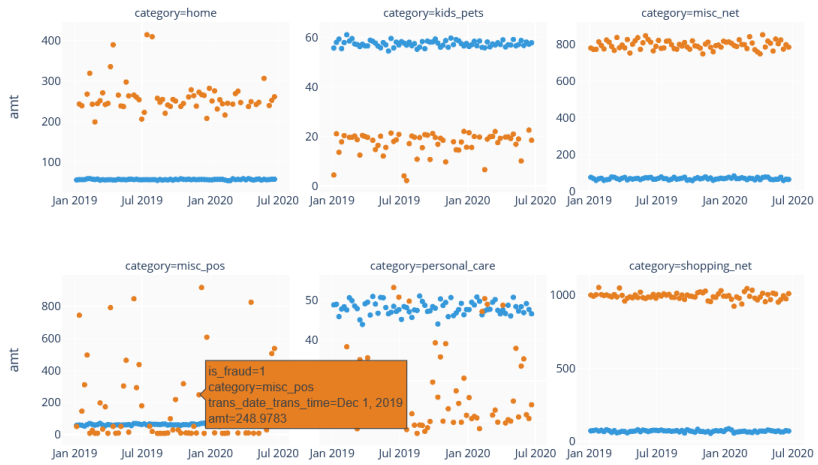
3.2.4. Number of Credit Card Frauds by Gender



Most cases of fraud occurred with female customers, but the number is almost the same for both Males and Females.

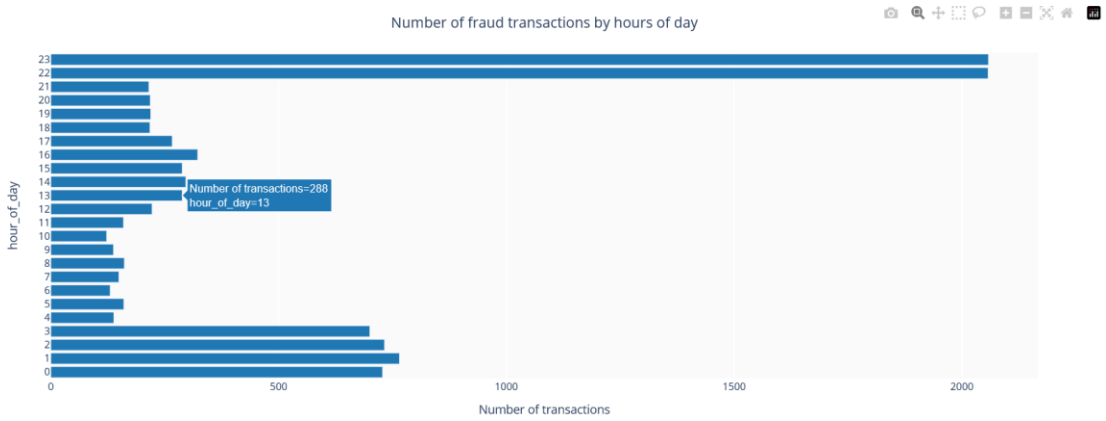
3.2.5. Summarize using pandas_profiling

In this part, we are creating a scatter plot. The plot visualizes the average transaction amount (amt) over time (trans_date_trans_time). The data is grouped by weekly intervals (freq="1W") and further faceted by the 'category' variable. Each facet represents a different category.



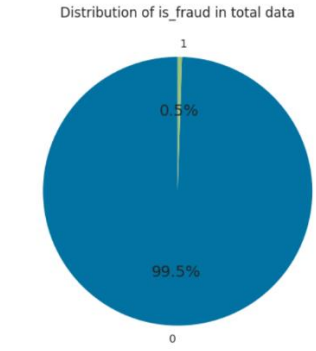
Each point on each category plot, shows the type of transaction and its time and its amount.

Number of fraud transactions by hours of day:



Most of the fraud transactions are happening at 23. You can also see number of transactions in each hour in plot.

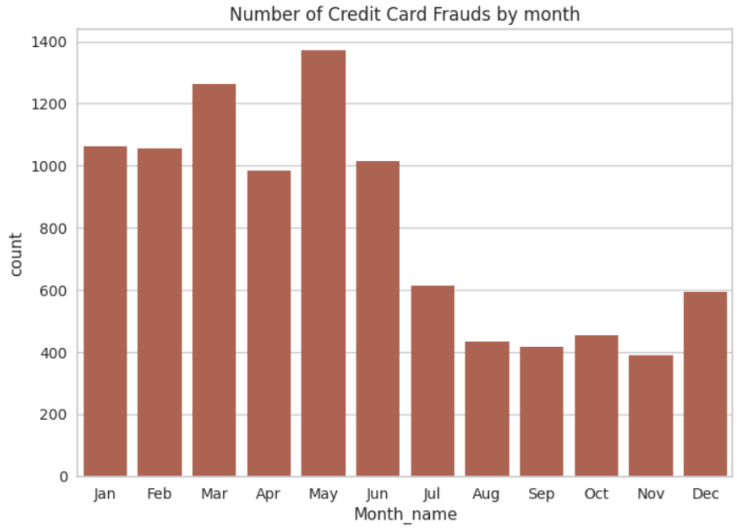
3.3 Distribution of is_fraud in total data




```
is_fraud
0    1842743
1       9651
Name: count, dtype: int64
```

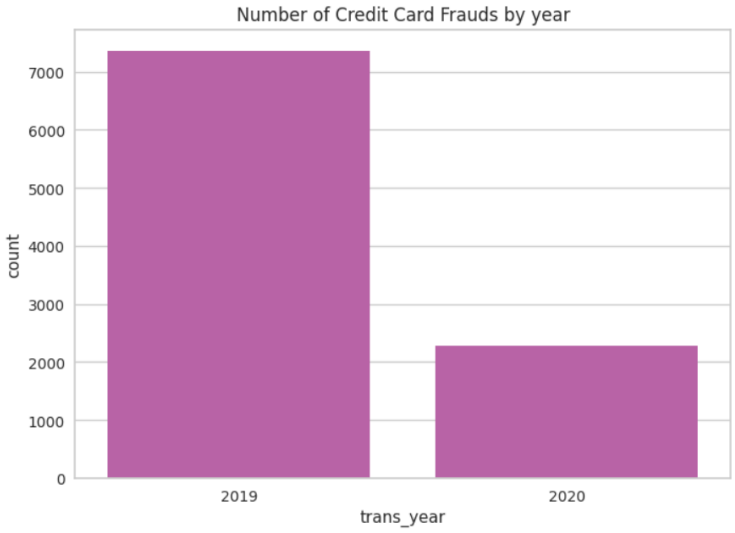
It seems that our dataset is imbalanced. Because only around 10,000 entries represent fraud transactions out of nearly 1.8 million entries.

3.4 creating transaction month and transaction year columns (feature engineering)



At the first we make a new column for transaction month.

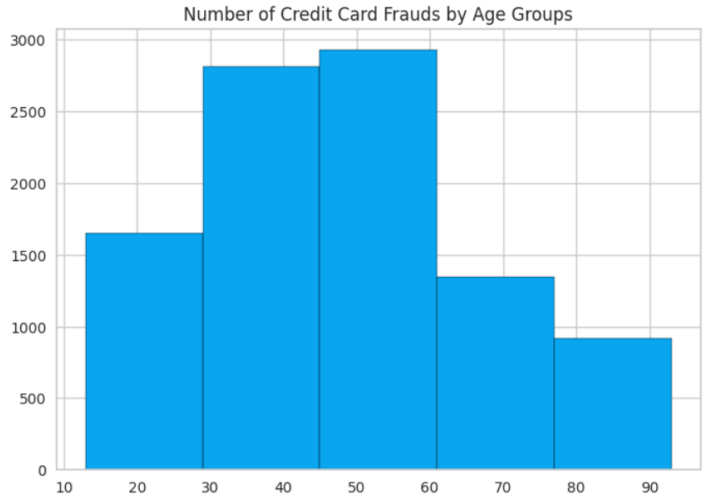
This plot shows us the number of credit cards which have frauds by month. As you can see, most of the frauds are in May and then March.



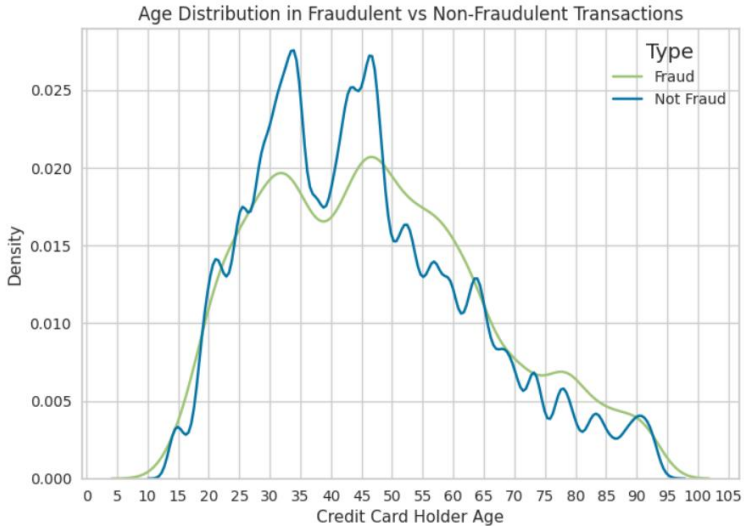
We can understand that most of the fraud transactions are in 2019.

3.5 creating age variable (feature engineering)

We calculate the age of individuals based on their date of birth ('dob') and the transaction date and time ('trans_date_trans_time').



It seems that most fraud transactions are done by people between 50 and 60.



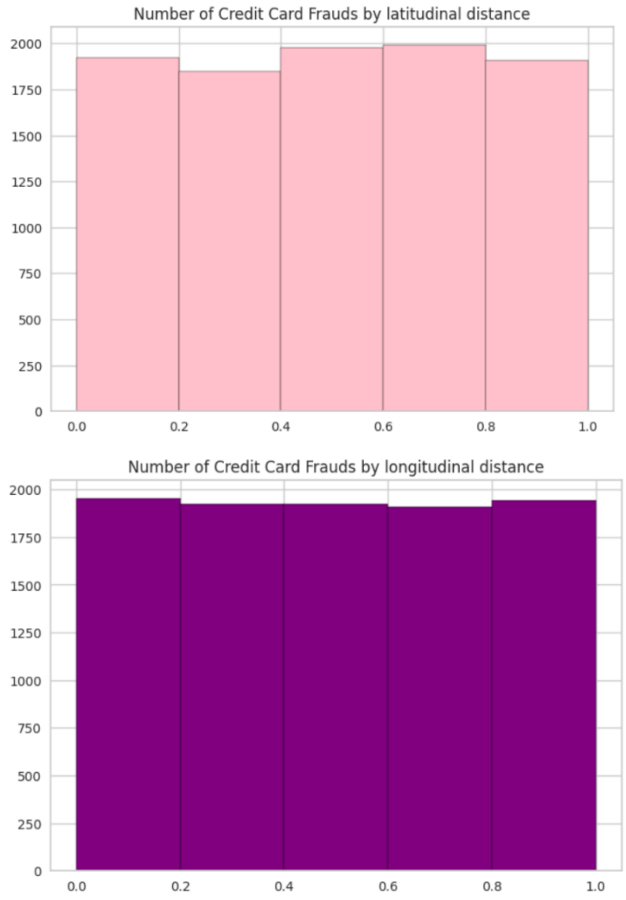
The age distribution is visibly different between 2 transaction types. In normal transactions, there are 2 peaks at the age of 37-38 and 49-50, while in fraudulent transactions, the age distribution is a little smoother and the second peak does include a wider age group from 50-65. This does suggest that older people are potentially more prone to fraud.

3.6 Finding distance

We find distance from customer location to merchant location in degrees latitude and degrees longitude.

The latitudinal distance is calculated as the absolute difference between the 'merch_lat' (merchant latitude) and 'lat' (transaction latitude) columns.

The longitudinal distance is calculated similarly for the 'merch_long' (merchant longitude) and 'long' (transaction longitude) columns.



3.7 gender values to binary values

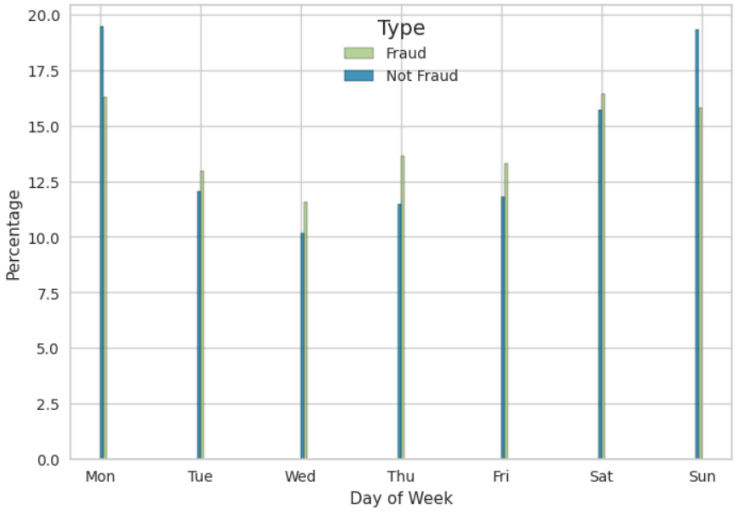
We convert gender values from categorical (F and M) to binary values (0 and 1).

```
gender
F    1014749
M     837645
Name: count, dtype: int64
```

The output shows that there are 1,014,749 instances of "F" (female) and 837,645 instances of "M" (male).

3.8 Weekly Trend

Now we visualize the distribution of fraud and non-fraud transactions across different days of the week.



We see that Monday and Sunday have the most not fraud transactions and Saturday and Monday have the most fraud transactions.

4. Model Building

Dropping some variables

We drop some variables which are not useful for model building. These variables can be deleted:

['cc_num','merchant','first','last','street','zip','trans_num','unix_time','time','is_fraud_category', 'trans_date_trans_time','city','lat','long','job','dob','merch_lat','merch_long','state']

The we convert categorical variables in ‘category’ column into dummy variables. The result is that new binary columns are added to the dataset, each representing a category from the original 'category' column.

Here is the final result:

```
<class 'pandas.core.frame.DataFrame'>
Index: 1852394 entries, 0 to 1296674
Data columns (total 24 columns):
 #   Column                  Dtype
---  ------                  -----
 0   amt                     float64
 1   gender                  int64
 2   city_pop                int64
 3   is_fraud                category
 4   hour_of_day             category
 5   trans_month             int32
 6   trans_year              int32
 7   age                     int64
 8   lat_distance            float64
 9   long_distance           float64
 10  day                     int32
 11  category_food_dining    bool
 12  category_gas_transport  bool
 13  category_grocery_net    bool
 14  category_grocery_pos    bool
 15  category_health_fitness bool
 16  category_home           bool
 17  category_kids_pets      bool
 18  category_misc_net       bool
 19  category_misc_pos       bool
 20  category_personal_care  bool
 21  category_shopping_net   bool
 22  category_shopping_pos   bool
 23  category_travel         bool
dtypes: bool(13), category(2), float64(3), int32(3), int64(3)
memory usage: 146.6 MB
```

Now we will implement 2 different methods:

1. Split to train and test, then handle imbalanced data, then run model.
2. handle imbalanced data, then split to train and test, then run model.

Then according to results, we will decide which method is suitable for which model.

4.1. split, handle imbalanced data, run model:

So, at the first we split our dataset to test and train and then we decide about oversampling/under sampling techniques for different models.

Split to train and test:

We extract the target variable 'is_fraud' and features from the 'total' dataset, converting them into NumPy arrays. Subsequently, the dataset is split into training and testing sets. The resulting arrays, namely X_train, X_test, Y_train, and Y_test, are ready for use in training.

20% of the data will be used for testing, and the remaining 80% will be used for training.

1. Logistic Regression Classifier:

Logistic regression may benefit from oversampling, especially when dealing with imbalanced datasets. It helps to balance the class distribution and improve the model's ability to predict the minority class.

So, we use random oversampling on data for this model. And then we run model.

Here are our results:

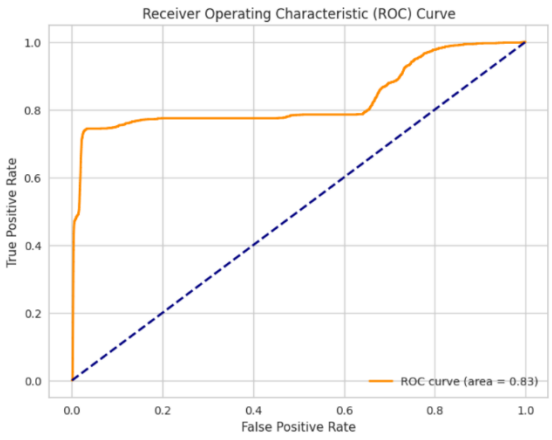
```
Accuracy: 0.9524534454044629

Confusion Matrix:
 [[351453  17129]
 [   486   1411]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.95      0.98    368582
           1       0.08      0.74      0.14      1897

    accuracy                           0.95    370479
   macro avg       0.54      0.85      0.56    370479
weighted avg       0.99      0.95      0.97    370479
```

The Logistic Regression model trained on the resampled data achieved an accuracy of approximately 95.25%. The confusion matrix reveals that there are 351,453 true negatives (non-fraudulent transactions correctly identified), 17,129 false positives (non-fraudulent transactions incorrectly identified as fraudulent), 486 false negatives (fraudulent transactions incorrectly identified as non-fraudulent), and 1,411 true positives (fraudulent transactions correctly identified).

Then we can see precision, recall, and F1-score metrics for both classes (0 for non-fraud, 1 for fraud). The precision for class 1 is notably low, indicating that among the instances predicted as fraud, only a small proportion are true fraud cases. The recall for class 1 is higher, suggesting that the model is better at capturing actual fraud cases.

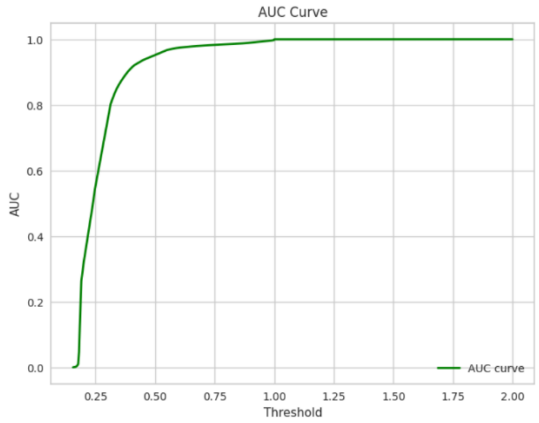
Overall, these results highlight the trade-off between precision and recall in imbalanced datasets.

ROC curve:



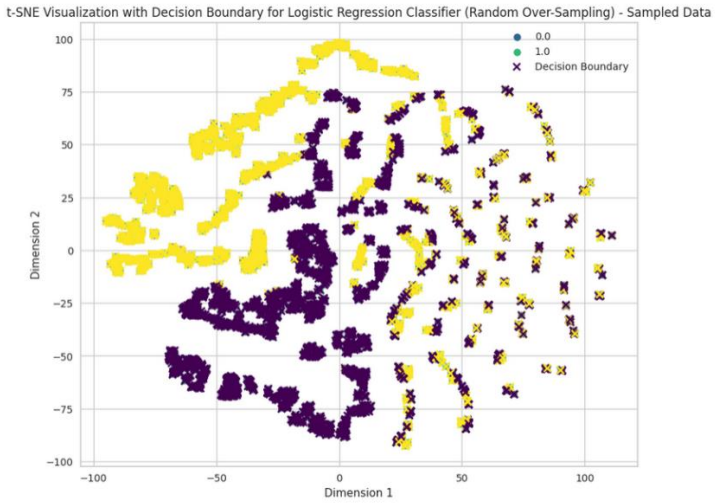
If a curve is closer to the upper-left corner, it indicates better performance. So, we can see that our model is good based on ROC results.

AUC curve:



The increasing AUC value signifies good overall model performance across different thresholds. AUC values closer to 1.0 indicate better model discrimination. And here we can see that AUC is close to 1.

t-sne:



2. Support Vector Machine (SVM) Classifier:

SVMs can benefit from oversampling, particularly when the dataset is imbalanced. Oversampling can improve the decision boundary, especially when the positive class is underrepresented.

So, we use random oversampling on data for this model. And then we run model. We just use 100,000 of data as sample for svm.

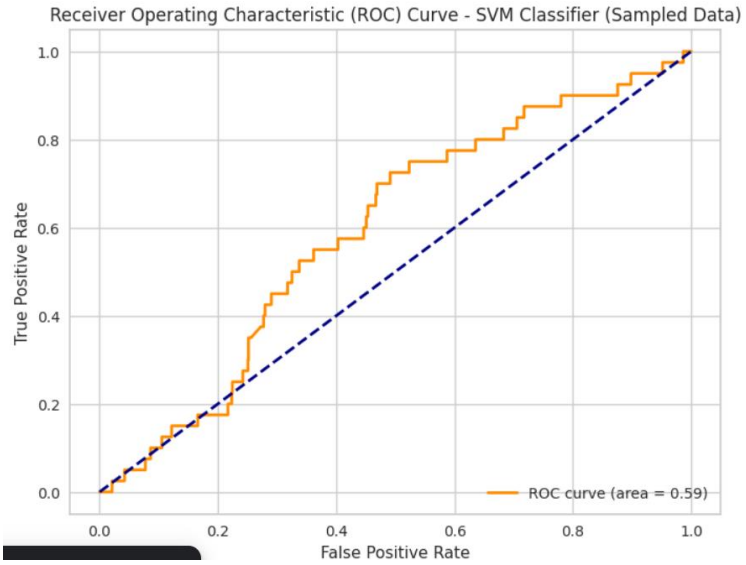
Here are our results:

```
Accuracy: 0.819

Confusion Matrix:
 [[8183 1777]
 [  33    7]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.82      0.90      9960
           1       0.00      0.17      0.01        40

    accuracy                           0.82     10000
   macro avg       0.50      0.50      0.45     10000
weighted avg       0.99      0.82      0.90     10000
```

The SVM model achieved an accuracy of approximately 81.9% on the sampled test set. The confusion matrix indicates 8,183 true negatives (non-fraudulent transactions correctly identified), 1,777 false positives (non-fraudulent transactions incorrectly identified as fraudulent), 33 false negatives (fraudulent transactions incorrectly identified as non-fraudulent), and 7 true positives (fraudulent transactions correctly identified).

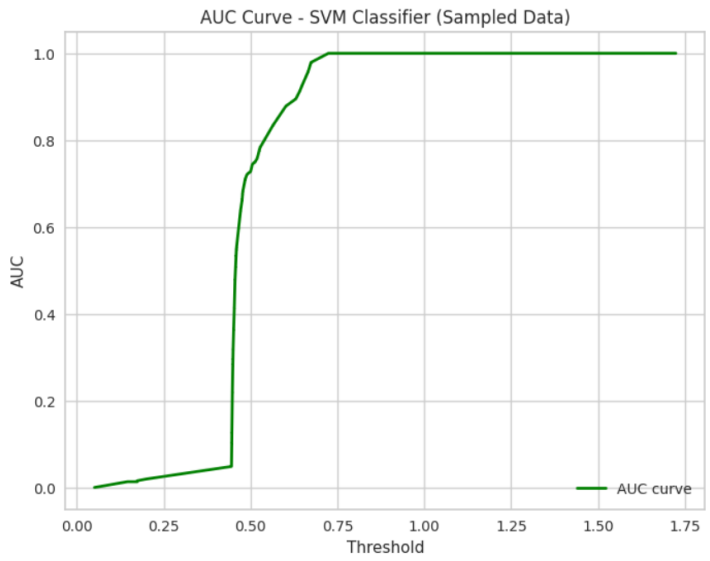
We present precision, recall, and F1-score metrics for both classes (0 for non-fraud, 1 for fraud). It's important to note that due to the imbalanced nature of the dataset, the precision for class 1 is low at 0.00%, indicating that among the instances predicted as fraud, none are true fraud cases. However, the recall for class 1 is 17%, suggesting that the model can capture some actual fraud cases. These results highlight the challenges posed by imbalanced datasets.

ROC curve:



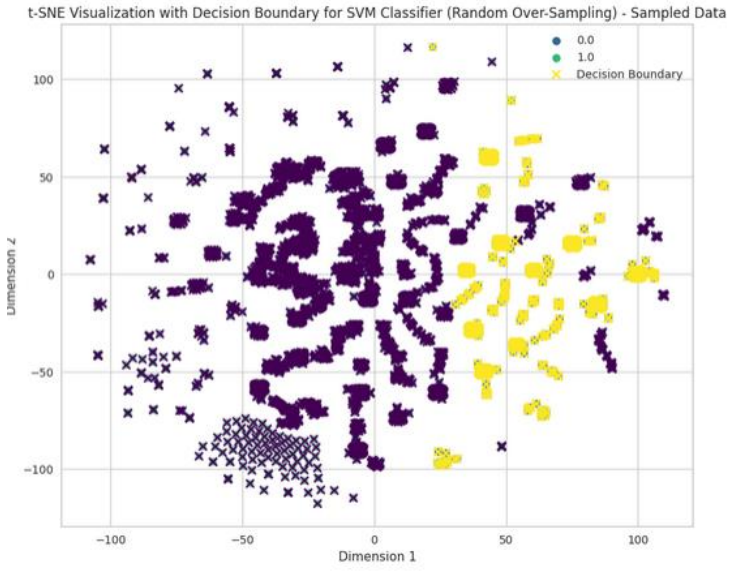
If a curve is closer to the upper-left corner, it indicates better performance. As you can see. Our model doesn’t have a good performance.

AUC curve:



The increasing AUC value signifies good overall model performance across different thresholds. AUC values closer to 1.0 indicate better model discrimination. Due to AUC curve, we can say that our model does not have good performance in this case.

t-sne:



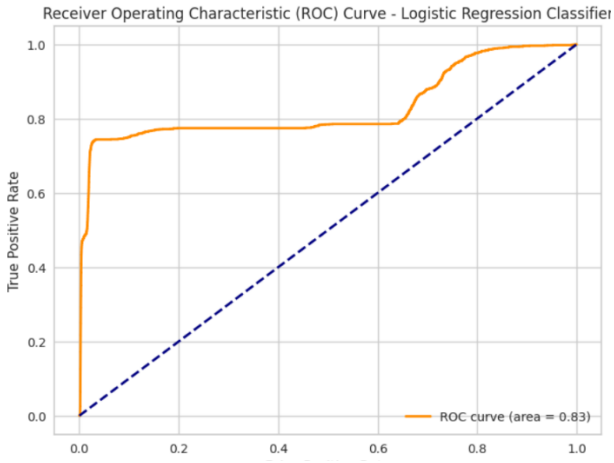
3. K-nearest neighbor (KNN) Classifier:

KNN can benefit from oversampling, especially if the class distribution is highly imbalanced. It helps prevent the model from being biased toward the majority class.

The KNN model achieved an accuracy of approximately 95.25% on the test set. The confusion matrix indicates 351,453 true negatives (non-fraudulent transactions correctly identified), 17,129 false positives (non-fraudulent transactions incorrectly identified as fraudulent), 486 false negatives (fraudulent transactions incorrectly identified as non-fraudulent), and 1,411 true positives (fraudulent transactions correctly identified).

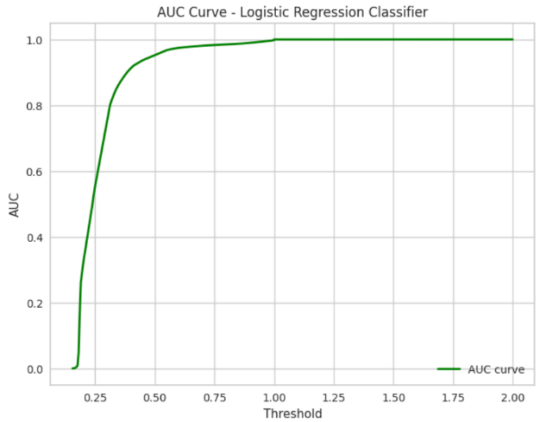
We show precision, recall, and F1-score metrics for both classes (0 for non-fraud, 1 for fraud). Notably, the precision for class 1 is relatively low at 8%, indicating that among the instances predicted as fraud, only a small proportion are true fraud cases. On the other hand, the recall for class 1 is higher at 74%, suggesting that the model is better at capturing actual fraud cases. These results emphasize the challenges posed by imbalanced datasets and the need for a balanced evaluation approach in fraud detection.

ROC curve:



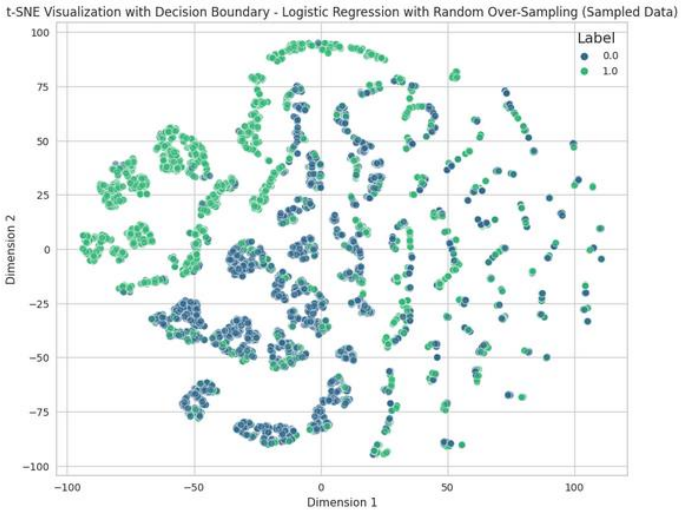
If a curve is closer to the upper-left corner, it indicates better performance. Based on this, our model is not good and not bad.

AUC curve:



The increasing AUC value signifies good overall model performance across different thresholds. AUC values closer to 1.0 indicate better model discrimination. As you can see, AUC is closer to 1.

t-sne:



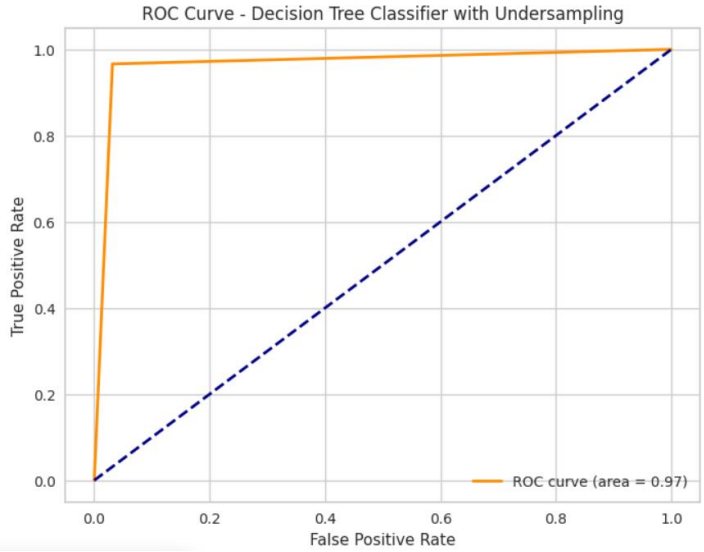
4. Decision Tree Classifier

Decision trees are less sensitive to imbalanced datasets compared to some other algorithms. However, if there is a significant class imbalance, undersampling the majority class may help.

```
Accuracy: 0.9684327586718815

Confusion Matrix:
 [[356951  11631]
 [    64   1833]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.97      0.98    368582
           1       0.14      0.97      0.24      1897

    accuracy                           0.97    370479
   macro avg       0.57      0.97      0.61    370479
weighted avg       1.00      0.97      0.98    370479
```

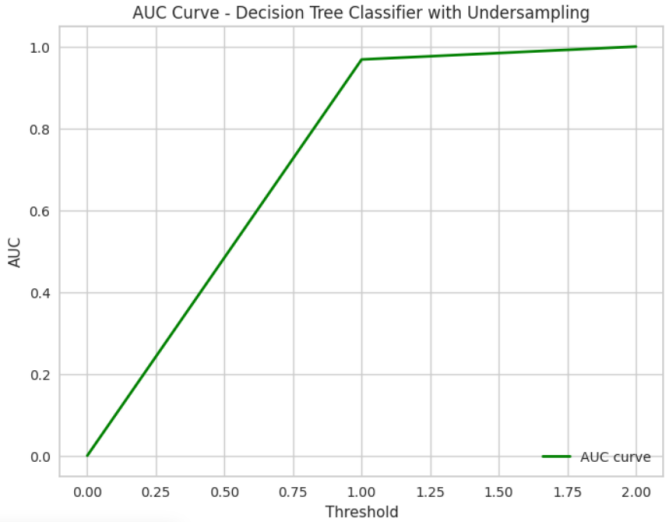
The Decision Tree Classifier trained on the resampled data achieved a high accuracy of approximately 96.84% on the test set. The confusion matrix reveals 356,951 true negatives (non-fraudulent transactions correctly identified) and 11,631 false positives (non-fraudulent transactions incorrectly identified as fraudulent). Additionally, there are 64 false negatives (fraudulent transactions incorrectly identified as non-fraudulent) and 1,833 true positives (fraudulent transactions correctly identified). Then we show precision, recall, and F1-score metrics for both classes (0 for non-fraud, 1 for fraud). Notably, the precision for class 1 is 14%, indicating that among the instances predicted as fraud, only a small proportion are true fraud cases. However, the recall for class 1 is high at 97%, suggesting that the model effectively captures most actual fraud cases. Overall, these results highlight the trade-off between precision and recall and emphasize the effectiveness of the Decision Tree Classifier in identifying fraudulent transactions.

ROC curve:



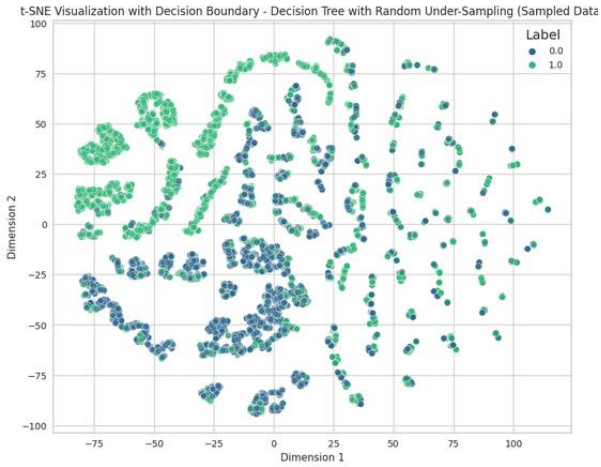
If a curve is closer to the upper-left corner, it indicates better performance. Based on this, our model is good.

AUC curve:



Based on this plot, our model is not good here.

t-sne:



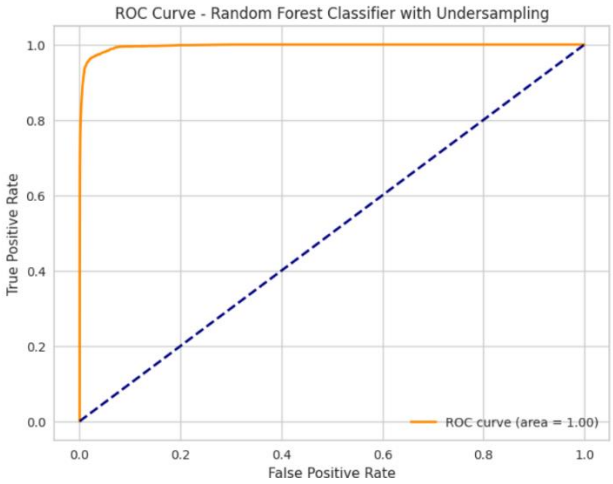
5. Random Forest Classifier:

Random Forests are generally robust to imbalanced datasets, but under sampling the majority class might be considered, depending on the nature of the problem.

```
Accuracy: 0.9775506843842701

Confusion Matrix:
 [[360335   8247]
 [    70   1827]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.98      0.99    368582
           1       0.18      0.96      0.31      1897

    accuracy                           0.98    370479
   macro avg       0.59      0.97      0.65    370479
weighted avg       1.00      0.98      0.99    370479
```

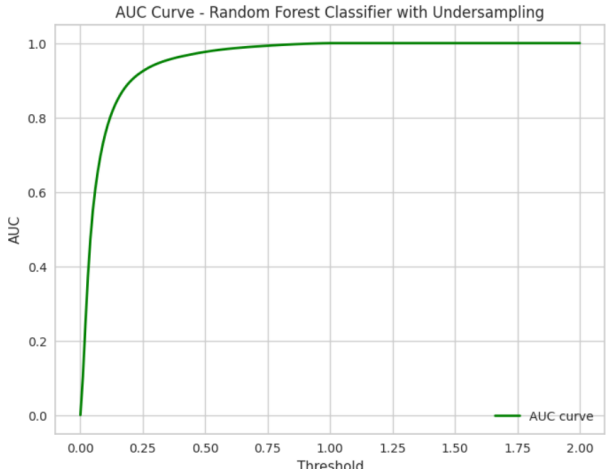
The Random Forest Classifier, trained on the resampled data using Random Under-Sampling, demonstrated excellent performance with an accuracy of approximately 97.76% on the test set. The confusion matrix reveals 360,335 true negatives (non-fraudulent transactions correctly identified) and 8,247 false positives (non-fraudulent transactions incorrectly identified as fraudulent). In addition, there are 70 false negatives (fraudulent transactions incorrectly identified as non-fraudulent) and 1,827 true positives (fraudulent transactions correctly identified). We can see precision, recall, and F1-score metrics for both classes (0 for non-fraud, 1 for fraud). Notably, the precision for class 1 is 18%, indicating that among the instances predicted as fraud, only a small proportion are true fraud cases. However, the recall for class 1 is high at 96%, suggesting that the model effectively captures most actual fraud cases. Overall, these results highlight the Random Forest Classifier's robust performance in identifying fraudulent transactions.

ROC curve:



It seems that our model is great based on ROC.

AUC curve:



Also, AUC curve shows that our model is great!

t-sne:

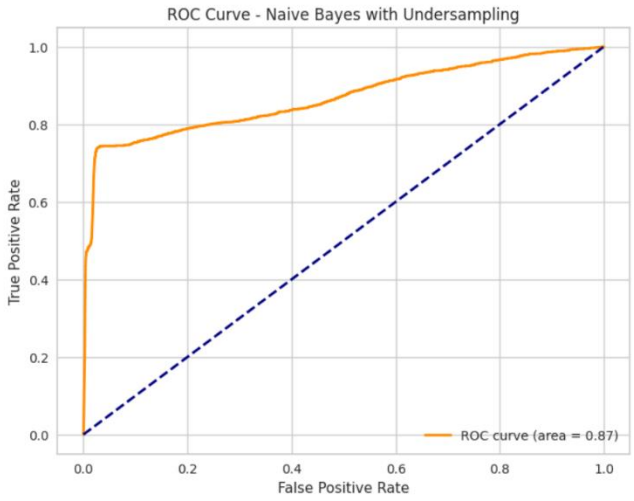
6. Naive Bayes:

Naive Bayes can work well with imbalanced datasets, and oversampling/undersampling might not be necessary. However, undersampling the majority class can be tried if the imbalance is severe.

```
Accuracy: 0.9750431198529471

Confusion Matrix:
 [[359854   8728]
 [   518   1379]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.98      0.99    368582
           1       0.14      0.73      0.23      1897

    accuracy                           0.98    370479
   macro avg       0.57      0.85      0.61    370479
weighted avg       0.99      0.98      0.98    370479
```

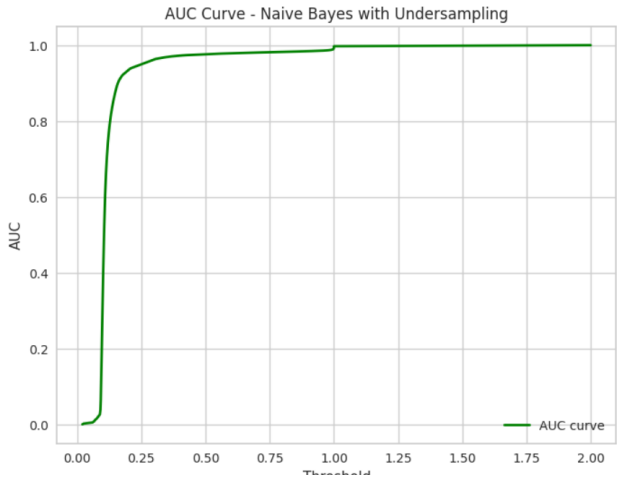
The Naive Bayes classifier, trained on the resampled data using Random Under-Sampling, exhibited strong performance with an accuracy of approximately 97.50% on the test set. The confusion matrix shows 359,854 true negatives (non-fraudulent transactions correctly identified) and 8,728 false positives (non-fraudulent transactions incorrectly identified as fraudulent). Additionally, there are 518 false negatives (fraudulent transactions incorrectly identified as non-fraudulent) and 1,379 true positives (fraudulent transactions correctly identified). We can see precision, recall, and F1-score metrics for both classes (0 for non-fraud, 1 for fraud). Notably, the precision for class 1 is 14%, indicating that among the instances predicted as fraud, only a small proportion are true fraud cases. However, the recall for class 1 is 73%, suggesting that the model is effective at capturing actual fraud cases. Overall, these results highlight the Naive Bayes classifier's ability to perform well in identifying fraudulent transactions, especially in the context of imbalanced datasets.

ROC curve:



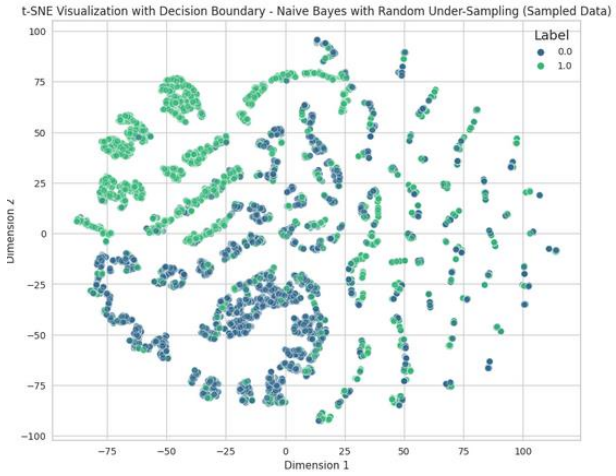
ROC curve shows that our model is a good performance but not great.

AUC curve:



AUC curve is close to 1 which is good.

t-sne:



4.2. handling imbalance, split, run model

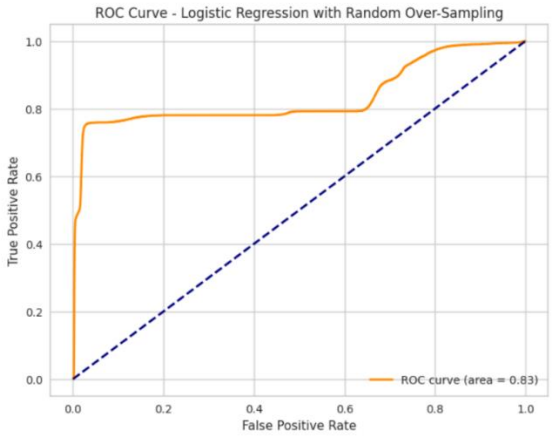
So, at the first we are handling imbalanced data using oversampling/under sampling techniques and then we split data to train and test.

1. Logistic Regression Classifier

```
Accuracy: 0.855939101720531

Confusion Matrix:
 [[351316  17233]
 [ 88954 279595]]

Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.95      0.87    368549
           1       0.94      0.76      0.84    368549

    accuracy                           0.86    737098
   macro avg       0.87      0.86      0.85    737098
weighted avg       0.87      0.86      0.85    737098
```

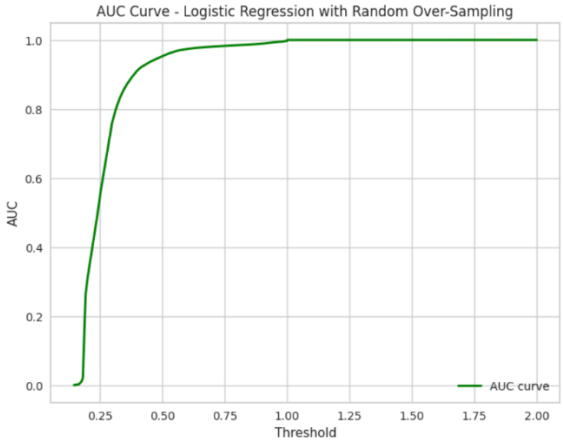
The Logistic Regression classifier, trained on the data after Random Over-Sampling, demonstrated an accuracy of approximately 85.59% on the test set. The confusion matrix reveals that there are 351,316 true negatives (non-fraudulent transactions correctly identified), 17,233 false positives (non-fraudulent transactions incorrectly identified as fraudulent), 88,954 false negatives (fraudulent transactions incorrectly identified as non-fraudulent), and 279,595 true positives (fraudulent transactions correctly identified). We see precision, recall, and F1-score metrics for both classes (0 for non-fraud, 1 for fraud). The precision for class 1 is 94%, indicating that among the instances predicted as fraud, a high proportion are true fraud cases. The recall for class 1 is 76%, suggesting that the model is somewhat effective at capturing actual fraud cases. Overall, these results highlight the trade-off between precision and recall in imbalanced datasets and emphasize the need for a comprehensive evaluation approach.

ROC curve:



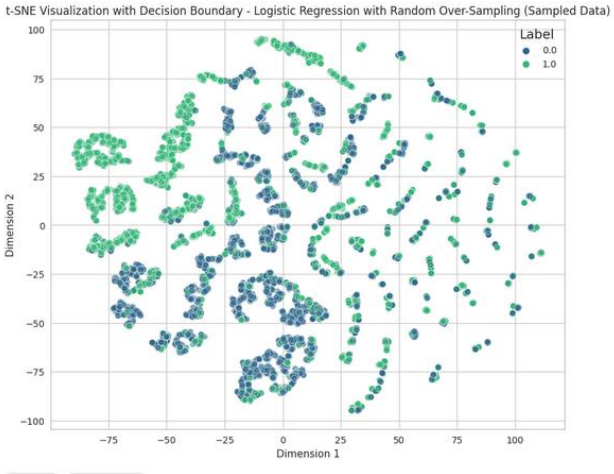
ROC shows that our model performance is not good and not bad.

AUC curve:



It shows that our model performance is good because curve is near 1.

t-sne:



2. Support Vector Machine (SVM) Classifier

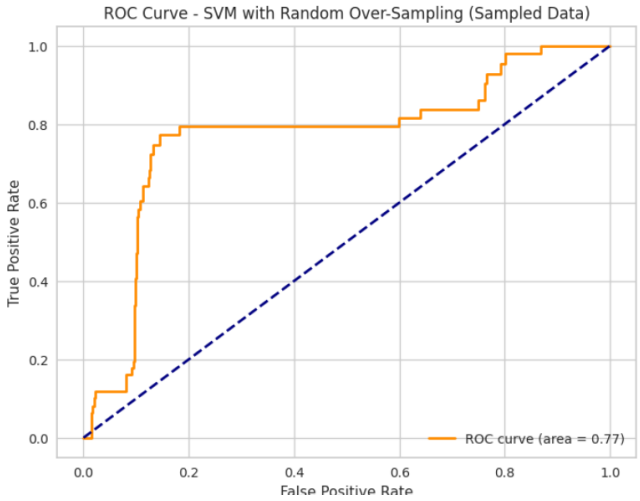
We just use 100,000 of data as sample for SVM.

```
Accuracy: 0.5381717729784028

Confusion Matrix:
 [[1908   83]
 [1756  235]]

Classification Report:
              precision    recall  f1-score   support

           0       0.52      0.96      0.67      1991
           1       0.74      0.12      0.20      1991

    accuracy                           0.54      3982
   macro avg       0.63      0.54      0.44      3982
weighted avg       0.63      0.54      0.44      3982
```

The SVM model achieved an accuracy of approximately 53.8% on the sampled test set. The confusion matrix indicates 1,908 true negatives (non-fraudulent transactions correctly identified), 83 false positives (non-fraudulent transactions incorrectly identified as fraudulent), 1,756 false negatives (fraudulent transactions incorrectly identified as non-fraudulent), and 235 true positives (fraudulent transactions correctly identified).

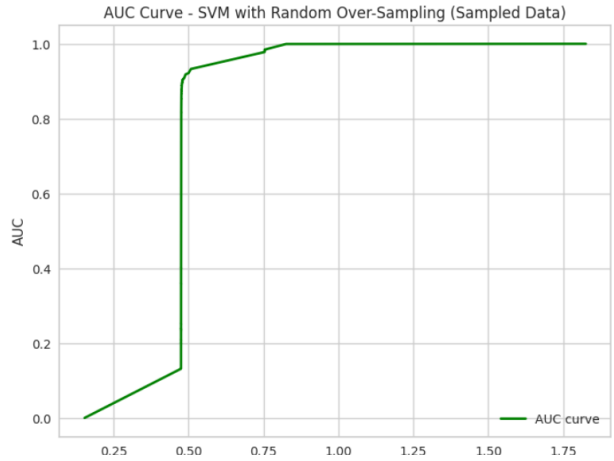
We present precision, recall, and F1-score metrics for both classes (0 for non-fraud, 1 for fraud). The precision for class 0 is 52%, indicating that among the instances predicted as non-fraud, 52% are true non-fraud cases. The precision for class 1 is 74%, suggesting that among the instances predicted as fraud, 74% are true fraud cases. However, the recall for class 1 is relatively low at 12%, indicating that the model is not effective at capturing all actual fraud cases. These results highlight the challenges posed by imbalanced datasets and the need for a balanced evaluation approach in fraud detection.

ROC curve:



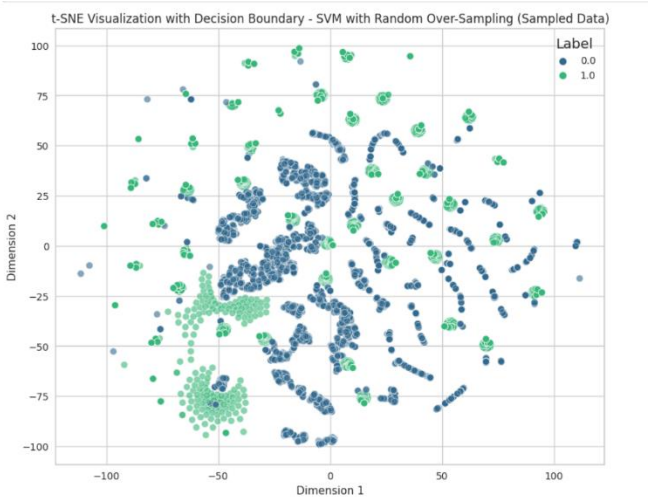
according to this plot, our model does not have good performance in this case.

AUC curve:



according to this plot, our model performance is not good and not bad.

t-sne:



3. K-nearest neighbor (KNN) Classifier

We just use 100,000 of data in this model.

```
Accuracy: 0.9925

Confusion Matrix:
 [[ 980   15]
 [   0 1005]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       995
           1       0.99      1.00      0.99      1005

    accuracy                           0.99      2000
   macro avg       0.99      0.99      0.99      2000
weighted avg       0.99      0.99      0.99      2000
```

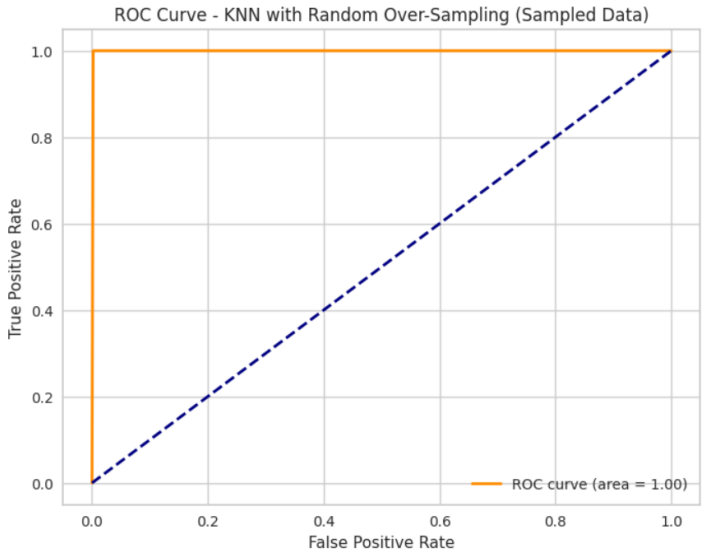
The KNN model achieved an accuracy of approximately 99.25% on the sampled test set. The confusion matrix indicates 980 true negatives (non-fraudulent transactions correctly identified), 15 false positives

(non-fraudulent transactions incorrectly identified as fraudulent), 0 false negatives (fraudulent transactions incorrectly identified as non-fraudulent), and 1005 true positives (fraudulent transactions correctly identified).

We present precision, recall, and F1-score metrics for both classes (0 for non-fraud, 1 for fraud). The precision for class 0 is 100%, indicating that among the instances predicted as non-fraud, all are true non-fraud cases. The precision for class 1 is 99%, suggesting that among the instances predicted as fraud, 99% are true fraud cases. The recall for both classes is 100%, indicating that the model effectively captures all actual cases, both non-fraud and fraud.

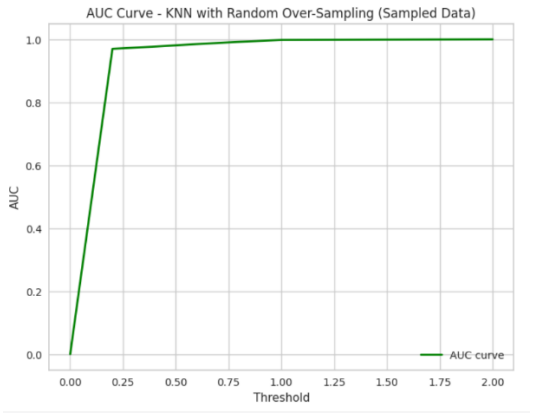
These results showcase the strong performance of the KNN model on the imbalanced dataset. The high precision and recall for both classes suggest that the model is effective in identifying both non-fraudulent and fraudulent transactions. The high accuracy further supports the model's overall performance.

ROC curve:

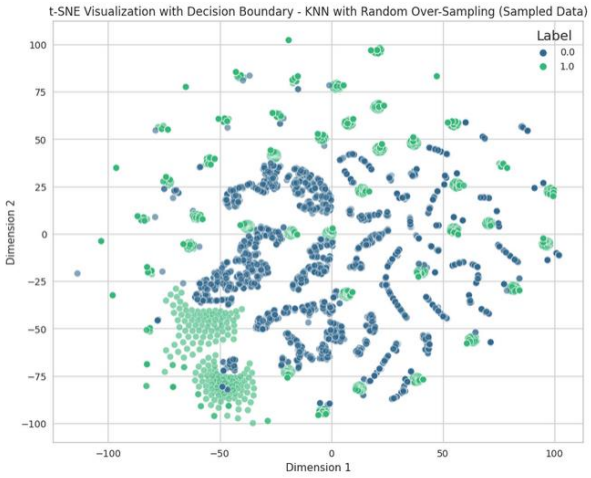


It seems that our model is great on just 100,000 of data.

AUC curve:



t-sne:

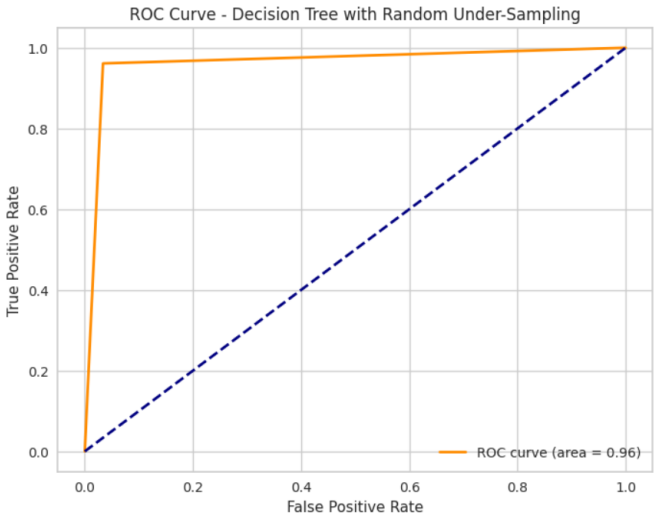


4. Decision Tree Classifier

```
Accuracy: 0.9637399637399637

Confusion Matrix:
 [[1866   65]
 [  75 1855]]

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.97      0.96      1931
           1       0.97      0.96      0.96      1930

    accuracy                           0.96      3861
   macro avg       0.96      0.96      0.96      3861
weighted avg       0.96      0.96      0.96      3861
```

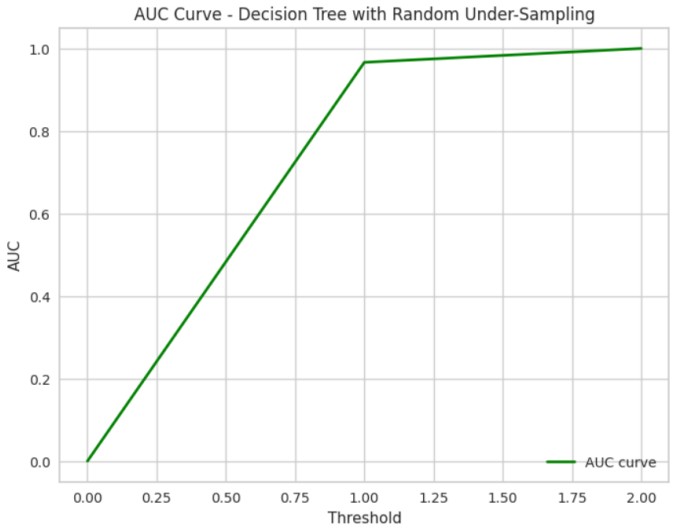
The Decision Tree classifier, trained on the data after Random Under-Sampling, achieved an accuracy of approximately 96.37% on the test set. The confusion matrix shows that there are 1,866 true negatives (non-fraudulent transactions correctly identified), 65 false positives (non-fraudulent transactions incorrectly identified as fraudulent), 75 false negatives (fraudulent transactions incorrectly identified as non-fraudulent), and 1,855 true positives (fraudulent transactions correctly identified). We can see precision, recall, and F1-score metrics for both classes (0 for non-fraud, 1 for fraud). The precision and recall for both classes are around 96%, indicating a balanced performance in terms of correctly identifying both non-fraudulent and fraudulent transactions. Overall, these results suggest that the Decision Tree classifier, trained on the under-sampled data, performs well in detecting fraud while maintaining a good overall accuracy.

ROC curve:



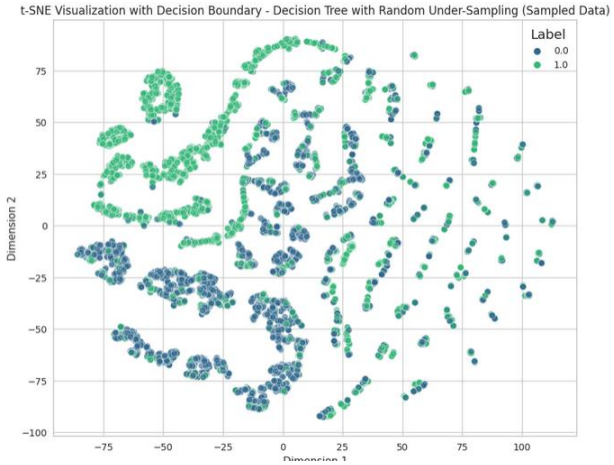
It seems that the performance is great due to the ROC curve.

AUC curve:



AUC curve shows model performance is not good and not bad.

t-sne:

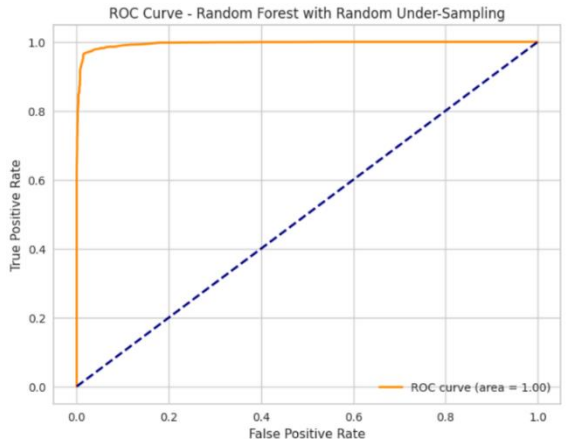


5. Random Forest Classifier

```
Accuracy: 0.9746179746179746

Confusion Matrix:
 [[1895   36]
 [  62 1868]]

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.98      0.97      1931
           1       0.98      0.97      0.97      1930

    accuracy                           0.97      3861
   macro avg       0.97      0.97      0.97      3861
weighted avg       0.97      0.97      0.97      3861
```

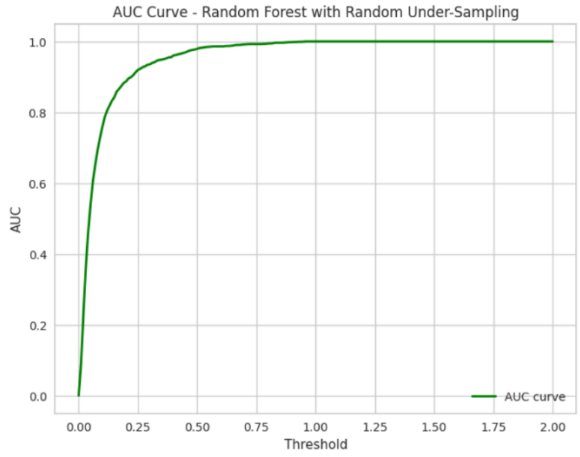
The Random Forest classifier, trained on the data after Random Under-Sampling, achieved an accuracy of approximately 97.46% on the test set. The confusion matrix reveals that there are 1,895 true negatives (non-fraudulent transactions correctly identified), 36 false positives (non-fraudulent transactions incorrectly identified as fraudulent), 62 false negatives (fraudulent transactions incorrectly identified as non-fraudulent), and 1,868 true positives (fraudulent transactions correctly identified). We see precision, recall, and F1-score metrics for both classes (0 for non-fraud, 1 for fraud). The precision and recall for both classes are around 97%, indicating a balanced performance in terms of correctly identifying both non-fraudulent and fraudulent transactions. Overall, these results suggest that the Random Forest classifier, trained on the under-sampled data, performs well in detecting fraud while maintaining a high overall accuracy.

ROC curve:



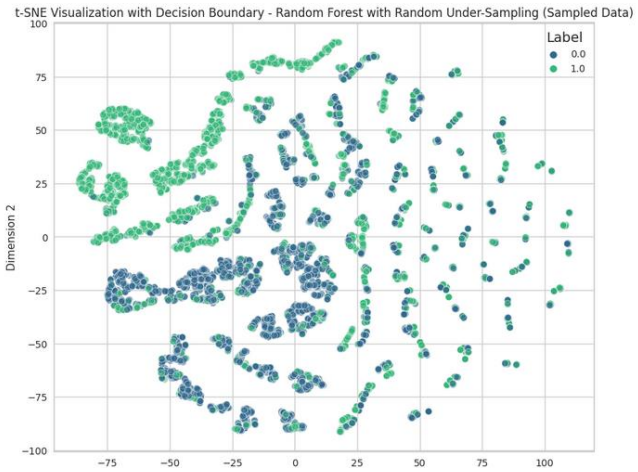
It seems that model performance is great.

AUC curve:



AUC curve is near to 1 so it is good.

t-sne:

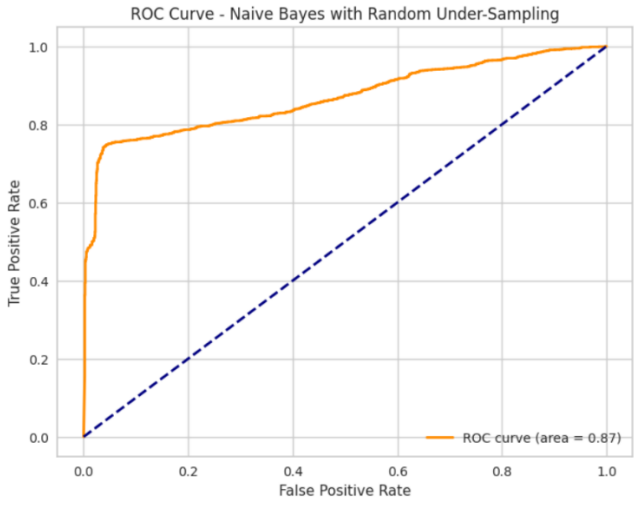


6. Naive Bayes

```
Accuracy: 0.8451178451178452

Confusion Matrix:
 [[1868   63]
 [ 535 1395]]

Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.97      0.86      1931
           1       0.96      0.72      0.82      1930

    accuracy                           0.85      3861
   macro avg       0.87      0.85      0.84      3861
weighted avg       0.87      0.85      0.84      3861
```

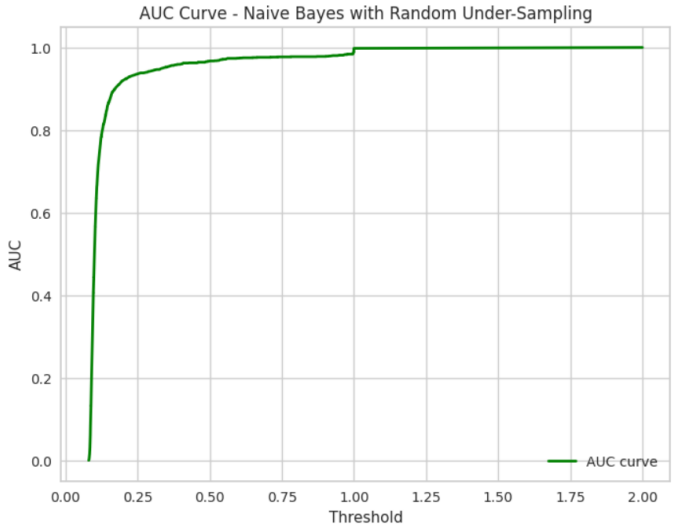
The Naive Bayes classifier, trained on the data after Random Under-Sampling, achieved an accuracy of approximately 84.51% on the test set. The confusion matrix shows 1,868 true negatives (non-fraudulent transactions correctly identified), 63 false positives (non-fraudulent transactions incorrectly identified as fraudulent), 535 false negatives (fraudulent transactions incorrectly identified as non-fraudulent), and 1,395 true positives (fraudulent transactions correctly identified). We see precision, recall, and F1-score metrics for both classes (0 for non-fraud, 1 for fraud). The precision for class 1 is notably high at 96%, indicating that among the instances predicted as fraud, a significant proportion are true fraud cases. However, the recall for class 1 is lower, suggesting that the model might miss some actual fraud cases. The ROC curve and AUC plot further illustrate the model's performance in terms of the trade-off between true positive rate and false positive rate, with an AUC of approximately 0.91. Overall, these results suggest that the Naive Bayes classifier, trained on the under-sampled data, performs well in detecting fraud but may benefit from further tuning to improve recall.

ROC curve:



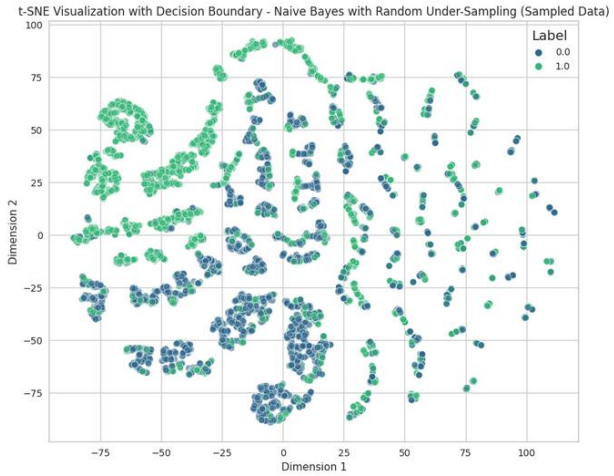
It shows that model performs well in this case.

AUC curve:



We can see that AUC curve is near to 1 so that's good.

t-sne:



We run all models with both methods and analyze all of them with metrics. Now we can compare each model with two methods easily and say which one is better!