University of Science and Technology of Hanoi

REPORT: CREDIT CARD TRANSACTIONS FRAUD DETECTION

Machine Learning and Data Mining II

Group 20 - ICT

Name	ID Student
Lê Ngọc Phan Anh	BI12-010
Hồ Quang Anh	BI12-017
Trần Đức Anh	BI12-018
Trần Ngọc Ánh	BI12-040
Hoàng Minh Đức	BI12-093
Lê Nhật Minh	BI12-289
Nguyễn Hoàng Nam	BI12-308

Ha Noi, 01/2024

TABLE OF CONTENTS

Classify Credit Card Transactions Fraud

A. Introduction

1. Abstract

Credit card fraud poses a significant challenge for financial institutions, with detrimental effects on both customers and businesses. Fraudulent transactions can occur due to various factors, including unauthorized access, compromised card information, or sophisticated cyber-attacks. Detecting fraudulent activity is crucial to safeguarding financial systems and maintaining customer trust. Establishing robust methods to classify credit card transactions as either legitimate or fraudulent is imperative. Utilizing Machine Learning algorithms, this research aims to identify key features within a credit card transaction dataset that impact fraud classification. By developing a fraud prediction framework, this study seeks to enhance the effectiveness of fraud detection in the financial sector, ultimately contributing to the security and integrity of electronic payment systems.

2. Rationale for Selecting the Research Topic

Financial Impact: Credit card fraud represents a significant financial threat to both consumers and financial institutions. The ability to accurately detect and prevent fraudulent transactions can result in substantial savings for businesses and ensure the financial well-being of customers.

Technological Relevance: With the increasing reliance on electronic payment systems, the need for advanced and adaptive fraud detection mechanisms has never been more critical. Exploring the application of Machine Learning.

Customer Trust: Instances of credit card fraud can erode customer trust in financial systems. Developing an effective fraud detection framework demonstrates a commitment to customer security, fostering trust and loyalty among clients.

Regulatory Compliance: Financial institutions are often subject to regulatory standards regarding the protection of customer data and the prevention of fraudulent activities. Research in credit card fraud detection can help institutions stay compliant with industry regulations.

Data Availability: The prevalence of credit card transaction data provides a rich source for analysis. Access to a diverse dataset allows for a comprehensive exploration of patterns and features that contribute to the identification of fraudulent transactions.

Real-World Impact: Successful implementation of a fraud detection system can have tangible real-world benefits by reducing financial losses, enhancing the efficiency of financial operations, and contributing to the overall stability of the financial ecosystem.

Research Gap: If there is a current gap in the existing literature or industry practices regarding credit card fraud detection, your research can fill this void and contribute valuable insights to the field.

3. Objective

- Implement advanced anomaly detection techniques to identify subtle patterns indicative of potential credit card fraud, contributing to the continuous improvement of fraud detection capabilities.
- Investigate and incorporate innovative features or data sources that can provide additional insights into the evolving tactics of fraudsters, thereby staying ahead of emerging fraud patterns in credit card transactions.
- Optimize model interpretability and explainability to provide actionable insights for fraud analysts and investigators, facilitating a more transparent and understandable fraud detection process in credit card transactions.

B. Experimental protocols

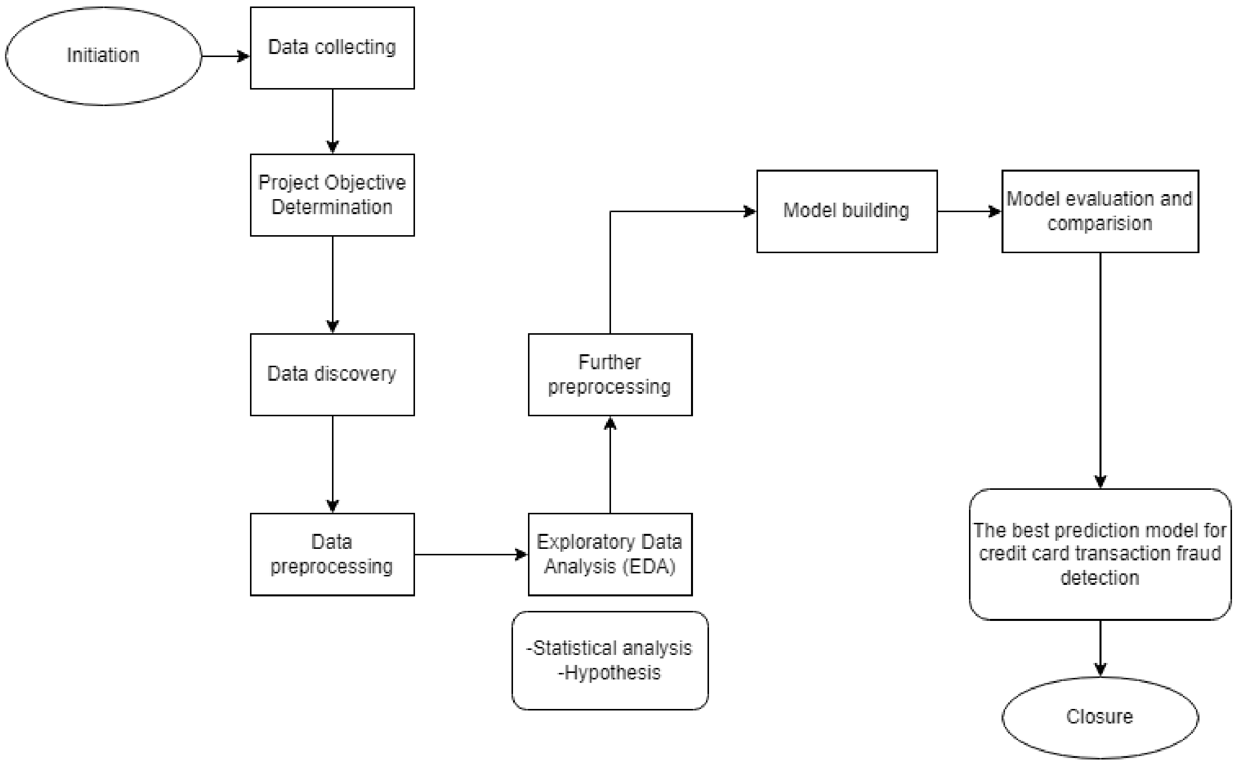

Initiation
Data collecting
Project Objective Determination
Data discovery
Data preprocessing
Exploratory Data Analysis (EDA)
-Statistical analysis
-Hypothesis
Further preprocessing
Model building
Model evaluation and comparision
The best prediction model for credit card transaction fraud detection
Closure

Figure 1. An overview of how the project will be conducted

1. Data Preprocessing

In order to perform further analysis or build models for classification, we have to preprocess the dataset in a proper way that we can utilize it. The process may consist of deleting unnecessary features, generating new useful features from the old and unuseful features or transforming data of a feature.

In this project we will split Data Preprocessing into 2 parts. One part is before the EDA and the other part after the EDA.

2. Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is crucial in credit card fraud detection to understand data patterns, identify relevant features, and detect outliers. It helps in cleaning and preprocessing data, ensuring accurate model training. Visualizations aid in communicating findings, and EDA results may suggest improvements for model performance. It validates assumptions, assesses risk, and contributes to a comprehensive understanding of transaction data, supporting effective fraud detection strategies.

3. Model Building and Evaluation

3.1. General steps explanation

Following data preprocessing and exploratory analysis, the next stage involves training and testing the dataset using three distinct models: Logistic Regression, Decision Tree, and Random Forest. Subsequently, we will evaluate the performance of these models through in-depth analysis, utilizing metrics such as the Confusion Matrix and Area Under the Curve (AUC), along with other widely recognized evaluation metrics.

Algorithm	Description	Reasons of choosing
Logistic Regression	Logistic regression is used for classification and predictive analytics. Logistic regression predicts a dependent data variable by analyzing the relationship between one or more existing given dataset of independent variables. And the outcome is predicted as binary outcome	Logistic Regression provides probabilistic outputs, enabling a nuanced understanding of prediction confidence, and is particularly effective when dealing with linearly separable data. It serves as an efficient baseline model, offering insights into feature importance and being easily interpretable, making it an excellent option for scenarios where a clear understanding of predictive factors is crucial.
P-values	P - value is calculated to validate a hypothesis on a dataset.In this study, p-value is used to determine whether a dataset feature has a significant impact on the output value. The lower p-value is, the more significant the features could be (Gelman, 2013) ● $p > 0.10 \rightarrow$ “not significant” ● $p \leq 0.10 \rightarrow$ “marginally significant”	P - values play a pivotal role in statistical hypothesis testing, quantifying evidence against a null hypothesis and aiding decision-making. They indicate whether observed results are likely due to chance, ensuring statistical significance in research findings. P-values contribute to the replicability of studies, offer a

	● $p \leq 0.05$ → “significant” ● $p \leq 0.01$ → “highly significant”	standardized communication method for evidence strength, and help control Type I error rates. Caution in interpretation and consideration alongside effect sizes enhance the reliability of research conclusions.
SMOTE	SMOTE is used for synthetic minority oversampling in machine learning. It generates synthetic samples to balance imbalanced datasets, specifically targeting the minority class. SMOTE should be used when dealing with imbalanced datasets to improve the performance of machine learning models on minority class predictions.	SMOTE prevents biases towards the majority class, enhances model performance on the minority class, and is applicable across various machine learning algorithms. It is particularly valuable in domains with imbalanced classes, such as fraud detection or rare event prediction, ensuring robust model performance and effective handling of critical instances.
Decision tree	A decision tree is a machine learning algorithm that uses a tree-like model for decision-making. It recursively splits data based on optimal conditions, leading to simple and interpretable outcomes. Despite their interpretability, decision trees can be prone to overfitting mitigated through techniques like pruning.	Decision trees are prized for their simplicity, interpretability, and adaptability. They efficiently handle diverse datasets, both numerical and categorical, without intricate preprocessing. Their visual clarity aids in easy comprehension and explanation, making them ideal for transparent decision-making. Notably versatile, decision trees excel in capturing complex patterns and are foundational in ensemble methods like Random Forests, ensuring robust performance across

		various tasks.
Random Forest	Random forest is the use of multiple classifier models for classifying the same data set, all the outcomes are combined to generate a single result. By following the large number principle, a random forest classifier is considered to reduce the error and give better performance, compared to a single classifier.	Random Forests stand out for their accuracy and adaptability in machine learning. By combining multiple decision trees, they deliver robust predictions, handling outliers and diverse data effortlessly. Versatile across classification and regression tasks, Random Forests provide reliable results with automatic feature importance assessment. Their reduced overfitting, efficient handling of missing data, and ease of use make them a powerful and accessible ensemble learning technique for a wide range of applications.
Area under the curve (AUC)	Area under the curve or area under the receiver operating characteristics is an evaluation metric used for measuring a classification model performance. The curve is plotted by the true positive rate as y-axis and the false positive rate as x axis. The higher AUC, the better the model.	AUC is a key metric in machine learning, offering a holistic assessment of a model's discriminative performance across various thresholds. Its threshold independence, insensitivity to class imbalance, and applicability to diverse scenarios make it invaluable. A higher AUC signifies superior performance, aiding in straightforward model comparison. With resilience to noise and suitability for both binary and multiclass scenarios, AUC provides a concise and effective measure for

		evaluating model performance.
Confusion matrix	A confusion matrix represents the prediction summary in matrix form. It shows how many predictions are correct and incorrect per class. It helps in understanding the classes that are being confused by models as other classes.	Confusion matrices are crucial in machine learning for a detailed performance breakdown, facilitating the calculation of key metrics like precision and recall. They aid in threshold adjustment, making informed decisions about model tuning, and are essential for handling class imbalances. With a clear visual representation, confusion matrices serve as a fundamental tool for evaluating and optimizing machine learning models.

Table 1. Detailed function of algorithms

C. Dataset discovery

1. Overview about the dataset

The dataset at hand encapsulates a simulated chronicle of credit card transactions, spanning the duration from January 1, 2019, to December 31, 2020. It has 23 features and 1852394 rows. These transactions involve a diverse array of 1000 customers, each engaging with a pool of 800 merchants. The dataset not only delineates legitimate transactions but also introduces the crucial facet of simulated fraudulent activities, amplifying its relevance for the development and evaluation of fraud detection models. Data scientists and researchers can leverage this rich dataset to uncover temporal patterns, discern subtle nuances in customer behaviors, and contribute meaningfully to the advancement of machine learning models tailored for credit card fraud detection. Despite being a simulated construct, this dataset emerges as a valuable asset for honing the efficacy and precision of fraud detection methodologies in the realm of electronic transactions.

	Feature	Data Format	Example
1	trans_date_trans_time	object(timestamp)	2019-01-01 00:00:18
2	cc_num	int64	2703186189652095
3	merchant	object	Rippin, Kub and Mann
4	category	object	misc_net
5	amt	float64	4.97
6	first	object	Jennifer
7	last	object	Banks
8	gender	object	F
9	street	object	561 Perry Cove
10	city	object	Moravian Falls
11	state	object	NC
12	zip	int64	28654
13	lat	float64	36.0788
14	long	float64	-81.1781
15	city_pop	int64	3495
16	job	object	Psychologist, counseling
17	dob	object	1988-03-09
18	trans_num	object	0b242abb623afc578575680df30655b9
19	unix_time	int64	1325376018
20	merch_lat	float64	36.011293

21	merch_long	float64	-82.048315
22	is_fraud	float64	0
23	split	object	train

Table 2. Overview about the dataset

To have more insights about the dataset we will divide the features into 2 set H1 and where:

H1: Customer information

The connections between individual characteristics and the occurrence of credit card fraud can have profound implications for fraud detection and prevention strategies. Within the credit card transaction dataset, particular attention is placed on customer information such as name, gender, address, job, and date of birth (DOB). The inclusion of personal details like name and address in the dataset underscores the importance of a granular understanding of customer demographics, providing essential insights for refining fraud detection models and implementing targeted measures to enhance the security of credit card transactions.

H2: Transactions information:

The transaction dataset delves into critical details that underpin the dynamics of credit card activities. Core transaction information includes the credit card number, enabling a distinct identification of financial interactions. The dataset further encapsulates essential parameters like merchant details, transaction amount, transaction date and time, unique transaction numbers, and Unix timestamps, providing a comprehensive view of the transactional landscape. Additionally, the inclusion of merchant latitude and longitude enriches the dataset, offering geographical insights into where transactions occur. This amalgamation of credit card specifics and transaction metadata forms a robust foundation for analyses, enabling a nuanced understanding of spending patterns, facilitating fraud detection, and contributing to the overall security and efficiency of electronic transactions.

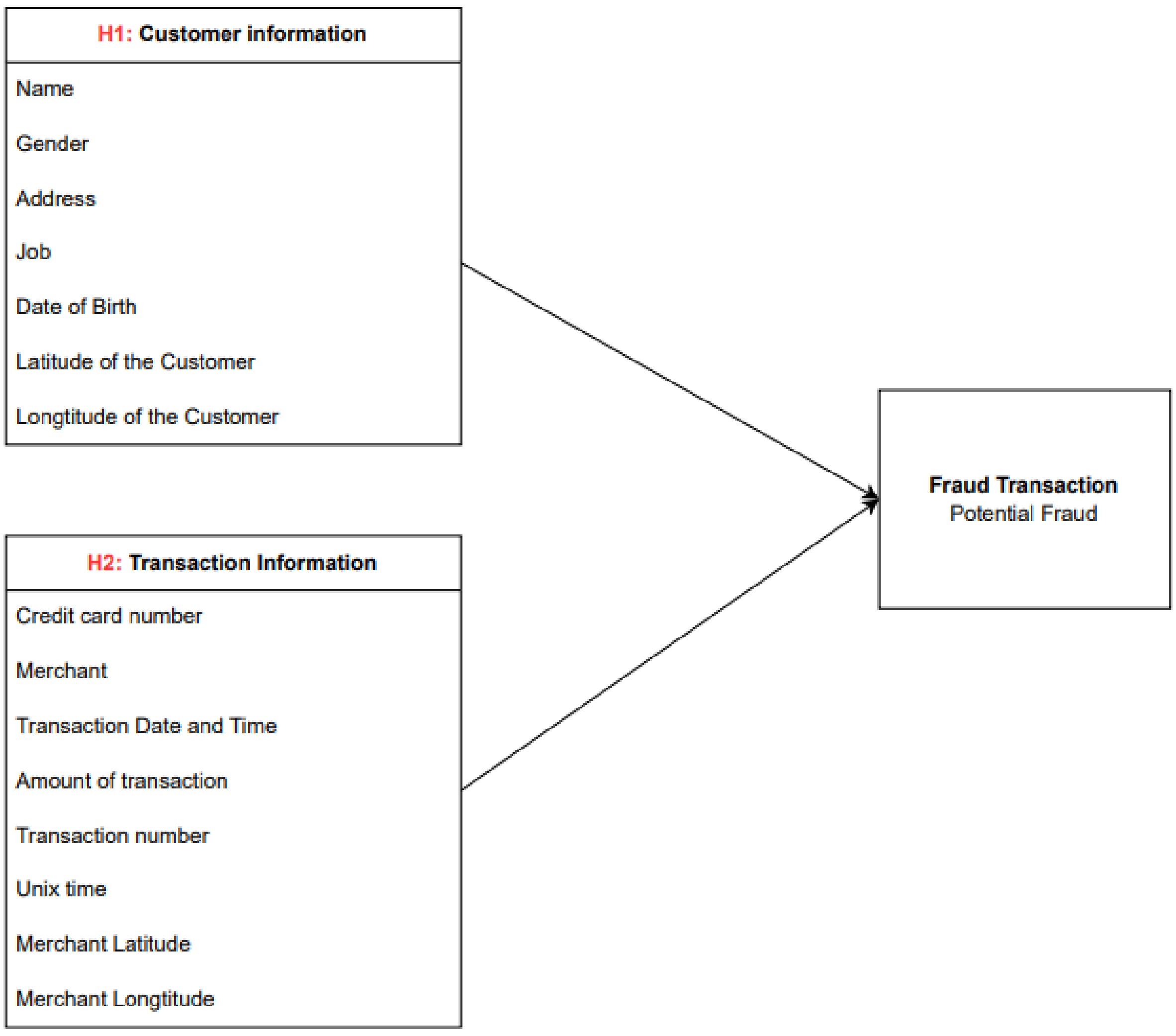


Figure 2. A conceptual model for the prediction of potential fraud transactions

	No	Column	Data type	Details
	1	first	qualitative	The customer's first name
	2	last	qualitative	The customer's last name
	3	gender	qualitative	The customer's gender(male or female)
	4	street	qualitative	The street where the customer lives
	5	state	qualitative	The state where the customer lives
	6	city	qualitative	The city where the customer lives

H1	7	job	qualitative	The customer’s job
	8	date of birth	qualitative	The customer’s date of birth
	9	lat	quantitative	The customer’s latitude
	10	long	quantitative	The customer’s longitude
	11	city_pop	quantitative	Population of the city the customer is living
	12	category	qualitative	The purpose for each transactions
H2	1	cc_num	qualitative	The customer’s credit card number
	2	merchant	qualitative	The merchant which the customer is paying to
	3	trans_date_trans_time	qualitative	The date and time of the transaction
	4	amt	quantitative	The amount of transaction
	5	trans_num	quantitative	Unique transaction number for each and every transaction
	6	unix_time	quantitative	Time of the transaction in Unix
	7	merch_lat	quantitative	The merchant’s latitude
	8	merch_long	quantitative	The merchant’s longitude
	9	zip	quantitative	Zip code of the transaction

Table 5. Dataset details

D. Data Preprocessing

We checked for null values and duplicated rows in a DataFrame using the **isnull()** and **duplicated()** function and then calculating the sum of null values for each column and the sum of duplicated rows using **sum()**.

Check Nulls: isnull()	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
	split	0
Check duplicate rows : duplicated().sum()		0

Table 6. Summarization of null data in each features and duplicated rows in the dataset

After that, we dropped unnecessary columns like "street", "state", "first", "last", "trans_num", "unix_time", "zip" as it's no longer useful for training models and detecting fraud. We already have an exact location to calculate distance later based on latitude and longitude.

Then for further analyses, we did some calculations:

- Based on the dob feature which is the customer's date of birth, we calculated the age of customers and created a new feature named "age".
- Based on the long, lat, merch_long, merch_lat features, we measured the distance between merchant and customer and created a new feature named "distance_km" Then we dropped the processed columns.
- We extracted new features: hour, month, day (of the week) from the trans_date_trans_time feature.
- All new features are put in the original dataframe.

Other features are remained originally, as there are many features in the dataset, we just put next here the table of representing new features:

gender	**hour**	**day**	**month**	**year**	**distance_km**
0	12	Tuesday	1	2019	127.61
0	8	Wednesday	1	2019	110.31
0	8	Wednesday	1	2019	21.79
0	12	Wednesday	1	2019	87.2
0	13	Wednesday	1	2019	74.21
...	...	...	...	...	...
1	2	Thursday	12	2020	44.89
1	5	Thursday	12	2020	81.49
1	11	Thursday	12	2020	36.05
1	11	Thursday	12	2020	81.77
1	13	Thursday	12	2020	22.08

Table 7. New dataset's features generated from the old ones

V. Exploratory Data Analysis

In order to inspect the dataset in a proper way, we will split the dataset's features into 2 groups. The first one is Category features and the other is Numerical features (A categorical feature is a variable with a set number of groups while a numeric feature is generally something that can be measured):

- Categorical features: 'cc_num', 'category', 'gender', 'city', 'job','day, 'is_fraud'.
- Numerical features: 'amt', 'city_pop', 'hour', 'month', 'age', 'distance_km'.
- Due to the fact that the feature 'trans_date_trans_time' is in timestamp format and has been extracted into 'hour,' 'day,' and 'month,' it will be excluded from the data exploration process. The same applies to the 'split' feature, as it merely indicates whether a row is from the 'train' or 'test' set and does not provide substantial information about the data.

1. Statistics summary

Obtain the processed DataFrame after the first preprocessing, the table below summarize all the statistical properties of Numerical features.

	count	mean	std	min	25%	50%	75%	max
amt	1852394.0	7.006357e+01	1.592540e+02	1.000000e+00	9.640000e+00	4.745000e+01	8.310000e+01	2.894890e+04
city_pop	1852394.0	8.864367e+04	3.014876e+05	2.300000e+01	7.410000e+02	2.443000e+03	2.032800e+04	2.906700e+06
hour	1852394.0	1.280612e+01	6.815753e+00	0.000000e+00	7.000000e+00	1.400000e+01	1.900000e+01	2.300000e+01
month	1852394.0	7.152067e+00	3.424954e+00	1.000000e+00	4.000000e+00	7.000000e+00	1.000000e+01	1.200000e+01
age	1852394.0	4.621138e+01	1.739545e+01	1.400000e+01	3.300000e+01	4.400000e+01	5.700000e+01	9.600000e+01
distance_km	1852394.0	7.611183e+01	2.911701e+01	2.000000e-02	5.532000e+01	7.822000e+01	9.851000e+01	1.521200e+02

Table 8. Statistic summary of Numerical features

- Count: The number of non-null values in the column. It indicates the completeness of the data.
- Mean: The average of all values in the column. It gives a measure of the central tendency.
- Std (Standard Deviation): A measure of the amount of variation or dispersion in the column values. It indicates how spread out the values are around the mean.
- Min: The minimum value in the column.
- 25% (Q1): The first quartile, or the 25th percentile. It represents the value below which 25% of the data falls.
- 50% (Median): The second quartile, or the 50th percentile. It is the middle value of the column. Half of the data falls below and half above this value.
- 75% (Q3): The third quartile, or the 75th percentile. It represents the value below which 75% of the data falls.
- Max: The maximum value in the column.

For example with the 'amt' feature we can extract some informations about the feature which is:

- Count: There are 1,852,394 data points for the 'amt' feature.
- Mean: The average amount is approximately $70.06.
- Standard Deviation (std): The amount varies around the mean by approximately $159.25 on average.
- Minimum (min): The smallest amount in the dataset is $1.00.
- 25th Percentile (25%): 25% of the amounts are less than $9.64. This represents the first quartile.
- 50th Percentile (50% or Median): The median amount is $47.45. This is the middle value when all amounts are sorted.
- 75th Percentile (75%): 75% of the amounts are less than $83.10. This represents the third quartile.
- Maximum (max): The largest amount in the dataset is $28,948.90.
- So we can conclude that the interquartile range is between $9.64 and $83.10.

After observing the Statistic summary of Numerical features, we will also try to do the same thing for the Categorical features.

	count	**unique**	**top**	**freq**
cc_num	1852394	999	6538441737335434	4392
merchant	1852394	693	Kilback LLC	6262
category	1852394	14	gas_transport	188029
gender	1852394	2	F	1014749
city	1852394	906	Birmingham	8040
job	1852394	497	Film/video editor	13898
is_fraud	1852394	2	0	1842743
day	1852394	7	Monday	369418

Table 9. Statistic summary of Categorical features

- Count: The number of non-null values in the column. It indicates the completeness of the data.
- Unique: The number of unique values in the column.
- Top: The most frequently occurring value in the column.
- Freq: The frequency of the most frequently occurring value (i.e., how many times the 'top' value appears in the column).

For example, the feature ‘merchant’ can be sum up as below:

- Count: There are 1,852,394 data points for the ‘merchant’ feature.
- Unique: There are 693 different merchants in the dataset.
- Top: The merchant that appear the most is ‘Killback LLC’
- Freq: The appearance of ‘Killback LLC’ is measured to be 6262 times in the dataset.

2. Univariate Analysis

In this context we will try to analyze and visualize the dataset by taking one variable at a time.

2.1. For Numerical features, we will use Histogram and Box Plot to indicate the distribution:

We will also calculate the skewness value for each feature so that we can evaluate those better. The skewness value is measured as below:

- Skewness Value: 0
- A skewness value around 0 suggests that the distribution is approximately symmetrical.

- Positive Skewness: > 0
    - If the skewness is positive, it indicates a right-skewed distribution.
    - In the box plot, the right tail will be longer than the left.
- Negative Skewness: < 0
    - If the skewness is negative, it indicates a left-skewed distribution.
    - In the box plot, the left tail will be longer than the right.

Amount of the transaction:

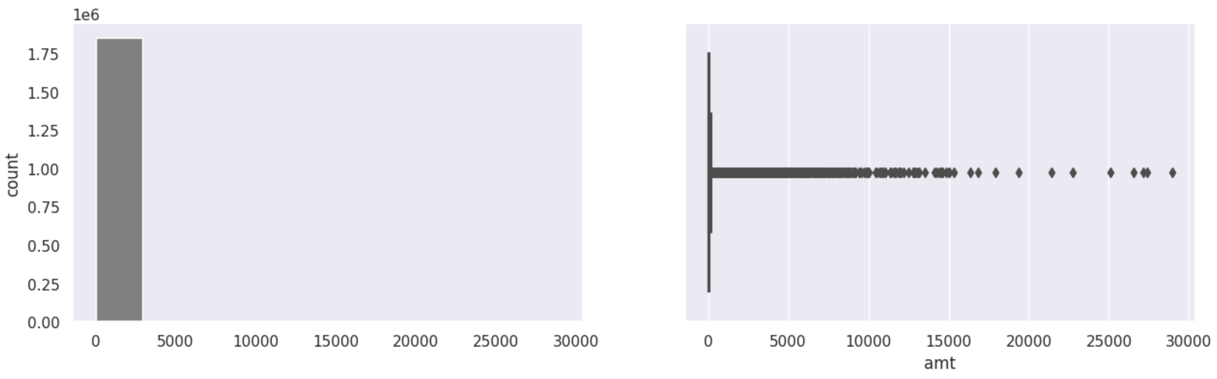


Figure 3. Histogram and Box plot of 'amt' features for all transactions (Skew: 40.81

Threshold for Outliers for amt: (-100.55, 193.29)

Number of Outliers for amt: 95054)

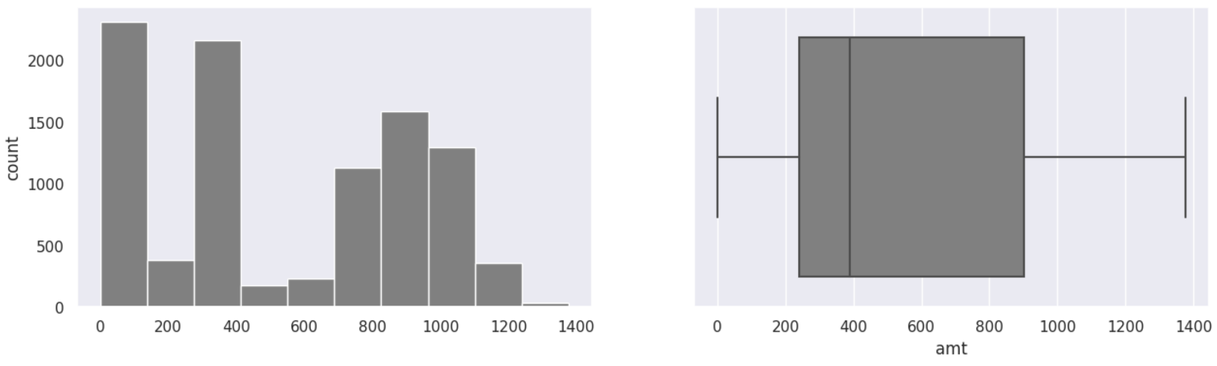


Figure 4. Histogram and Box plot of 'amt' features for fraudulent transactions (Skew: 0.03

Threshold for Outliers for amt: (-753.35, 1895.8)

Number of Outliers for amt: 0)

As we can see clearly from the first two charts above that there are many outliers in the right side of the Box plot, most of 'amt' data points are lying between the value of 0 and 5000 dollars and the rest (outliers) are lying outside the upper whisker value of the Box plot. So as a result for a bigger amount of money of the transaction the less frequent of that transaction appears.

The second two charts that show the distribution of 'amt' in fraudulent transactions indicate that the median is nearly 400 dollars and half of fraudulent transactions is below 400 dollars and the amount of transactions method varies between 0 and 1400 dollars. With the upper bound of the Box plot for all transactions equal to 193.29, we see that the majority of data points are outliers. Therefore, we need to handle the 'amt' in the next preprocessing part.

City population:

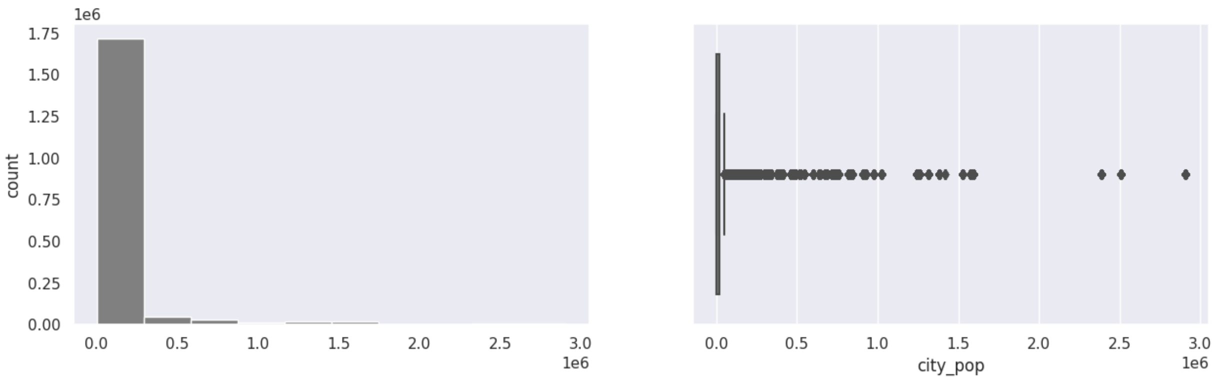


Figure 5. Histogram and Box plot of 'city_pop' for all transactions(Skew: 5.59
Threshold for Outliers for city_pop: (-28639.5, 49708.5)
Number of Outliers for city_pop: 346191)

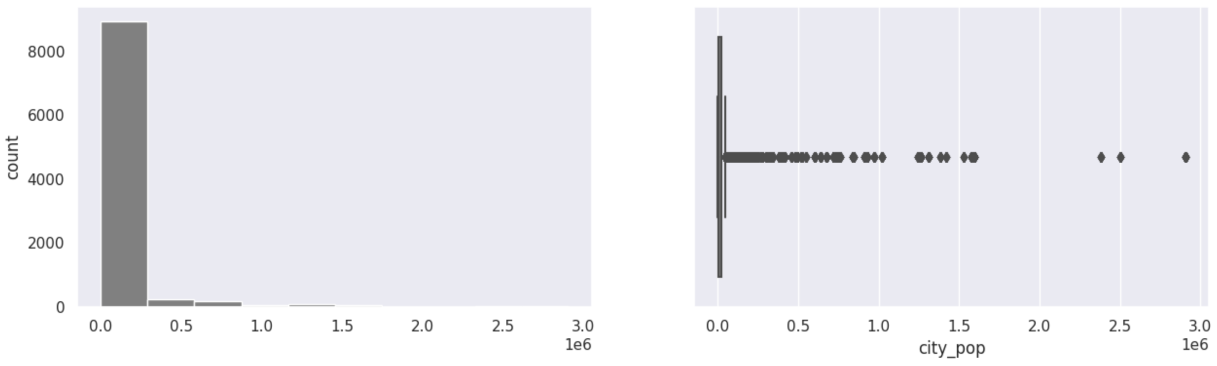


Figure 6. Histogram and Box plot of 'city_pop' for fraudulent transactions(Skew: 5.74
Threshold for Outliers for city_pop : (-26593.5, 46442.5)
Number of Outliers for city_pop: 1840)

Both the figures of 'city_pop' for all transactions and the 'city_pop' for fraudulent transactions look identical. We see that data points of 'city_pop' for all transactions have a gigantic number of outliers on the right hand side as well as the 'city_pop' for fraudulent transactions. The cities with the population that under 50000 tend to have the most made transactions and large numbers of fraudulent transactions also come from cities that are below 50000 citizens. Because the data point of 'city_pop' consists of a lot of outliers so the feature is also required to be handled in the next preprocessing.

Hour:

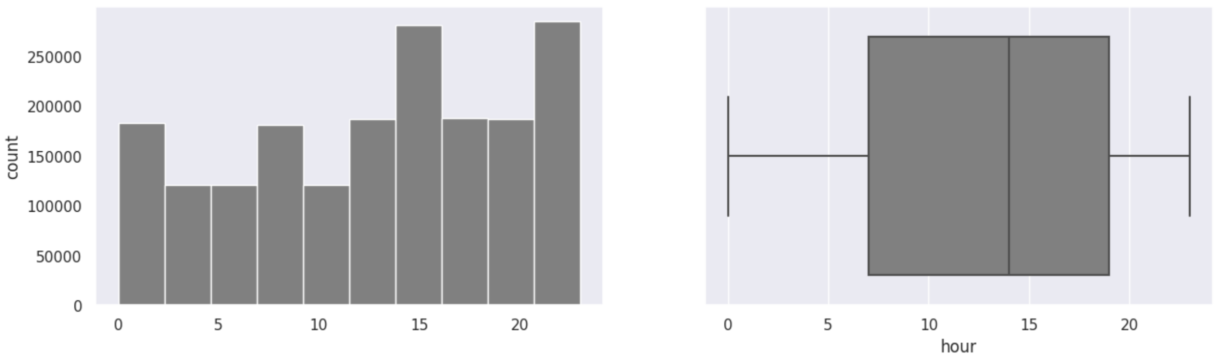


Figure 7. Histogram and Box plot of 'hour' for all transactions(Skew: -0.28
Threshold for Outliers for hour: (-11.0, 37.0)
Number of Outliers for hour: 0)

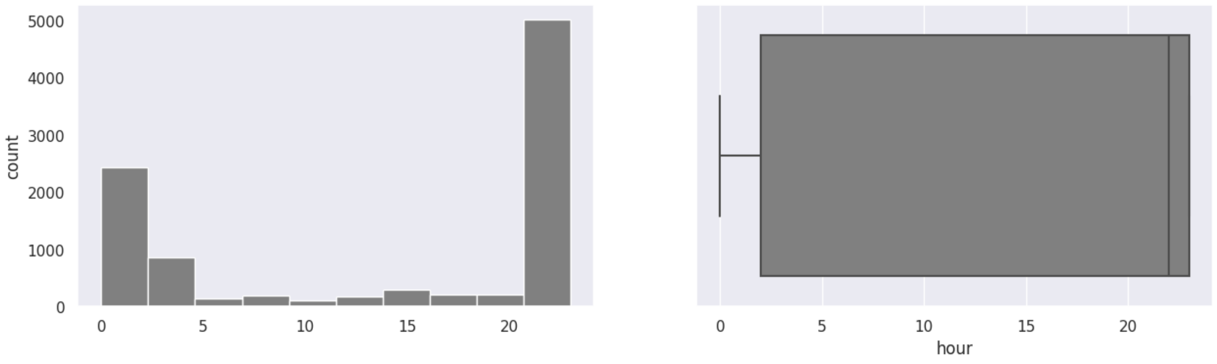


Figure 8. Histogram and Box plot of 'hour' for fraudulent transactions(Skew: -0.42
Threshold for Outliers for hour : (-29.5, 54.5)
Number of Outliers for hour: 0)

From the first figure along with the skew is negative and the left tail is longer we can conclude that the data of 'hour' is spread out more on the left side of the median which is 14h (2pm) and there are no outliers observed on the figure above. So half of the transactions are conducted before 14h and the other half are quickly fired up after 14h. So more people tend to make transactions from afternoon to midnight.

From the second figure, we can see that almost all of the fraudulent transactions are conducted during 20h until 5h of the next day.

Month:

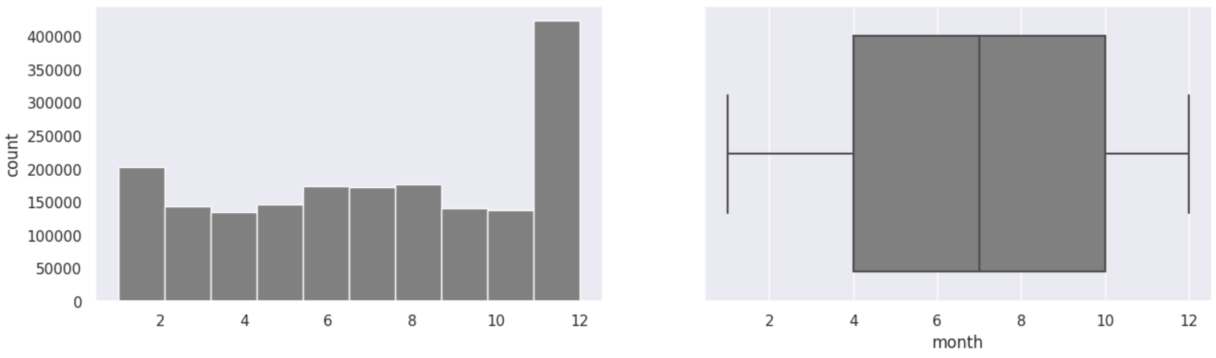


Figure 9. Histogram and Box plot of 'month' for all transactions(Skew: -0.13
Threshold for Outliers for month: (-5.0, 19.0)
Number of Outliers for month: 0)

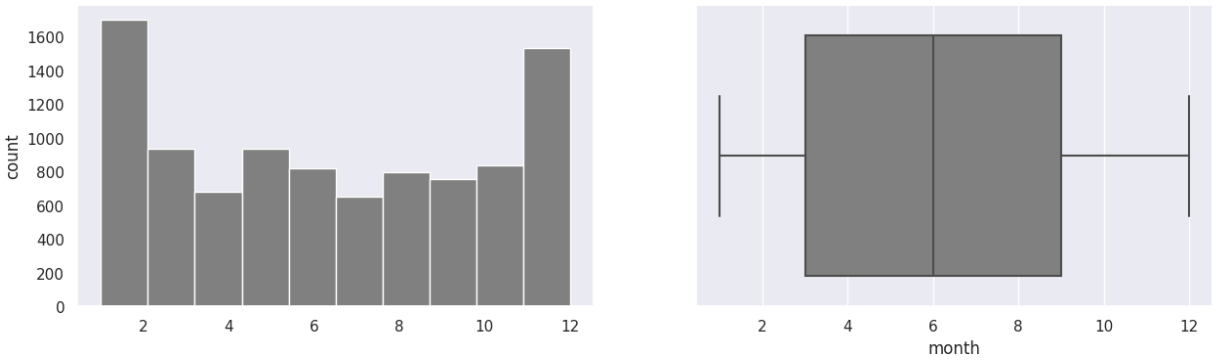


Figure 10. Histogram and Box plot of 'month' for fraudulent transactions(Skew: 0.06
Threshold for Outliers for month : (-6.0, 18.0)
Number of Outliers for month: 0)

Last months of the year receive an extensive amount of transactions (400000 transactions) and other months have similar amounts of transactions (around 200000 transactions). But for the fraudulent cases, the time of peak fraudulent activities are in the first few months of the year and the second highest fraudulent transactions are happening in the last few months. So we can conclude that the fraudulent cases appear most in the early months then go down to below 1000 cases in the middle months of the year and then rise up in those late months.

Age:

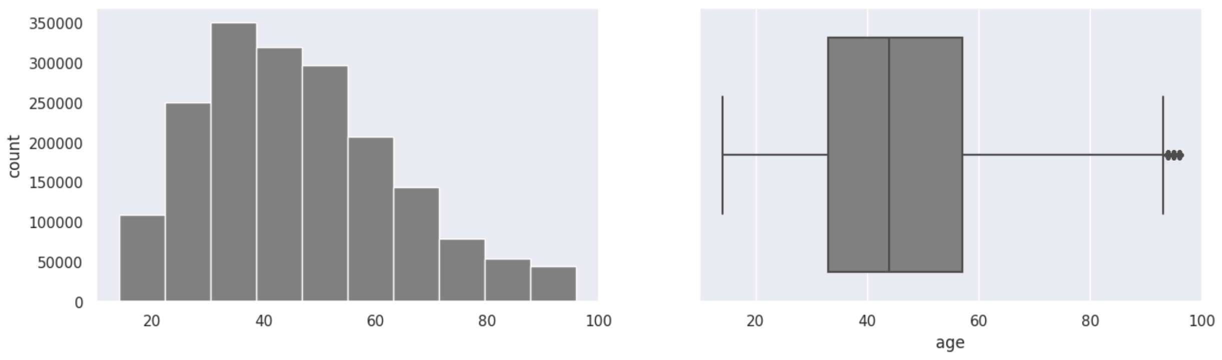


Figure 11. Histogram and Box plot of 'age' for all transactions(Skew: 0.61
Threshold for Outliers for age: (-3.0, 93.0)
Number of Outliers for age: 5177)

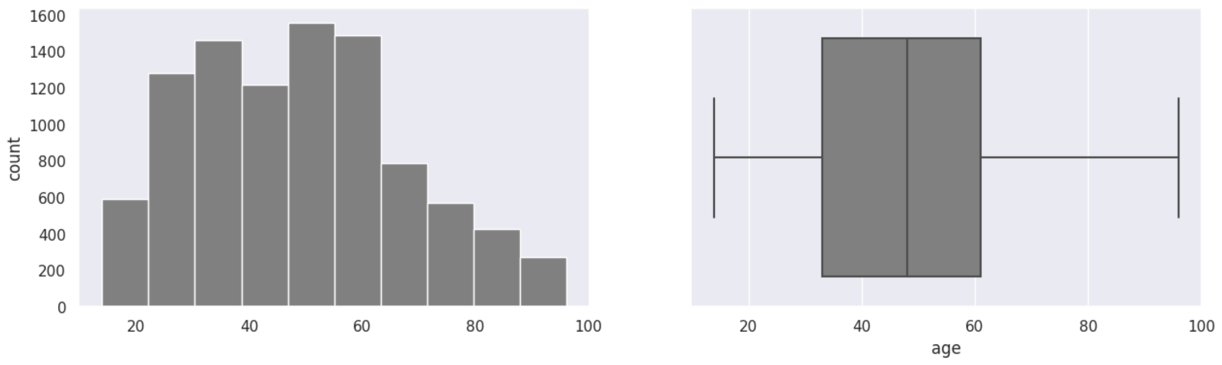


Figure 12. Histogram and Box plot of 'age' for fraudulent transactions(Skew: 0.38
Threshold for Outliers for age : (-9.0, 103.0)
Number of Outliers for age: 0)

The age of all customers obtained from all transactions is a right skewed distribution when the range of age spreads out more on the right side of the median. The age that has the most transactions is 30s (35000 transactions), the amount of transactions increase from 10s age to its peak at 30s and then decrease as the age gets older. Also we detect a small amount of outliers on the right side of the upper bound but because it just has 5177 points (which is small compared to the dataset) so we will not consider it as a problem.

For the fraudulent cases, most of the counterfeits are conducted with the customer who is under 60 years old. The age that associates with fraudulent transactions begin at 10s and rapidly rise up to its peak at 50s (which is 1600 transactions) and then decrease as the age gets older.

Distance between the customer and merchant:

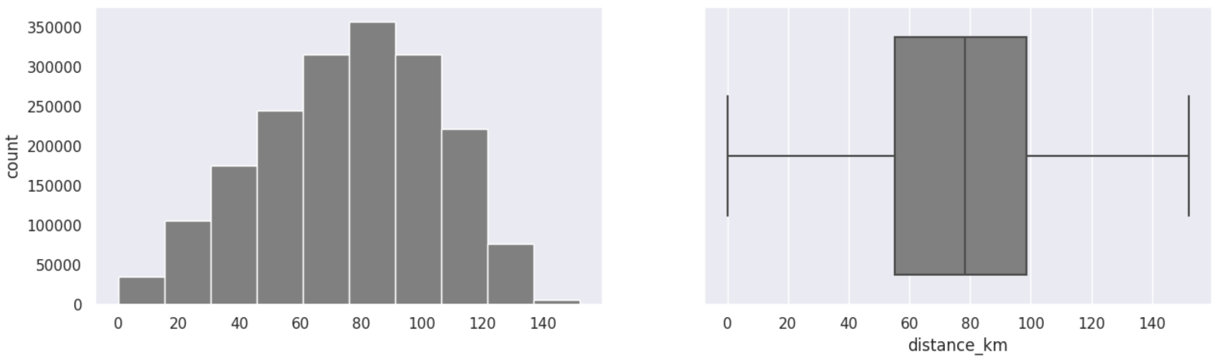


Figure 13. Histogram and Box plot of 'distance_km' for all transactions(Skew: -0.24
Threshold for Outliers for distance_km: (-9.46, 163.29)
Number of Outliers for distance_km: 0)

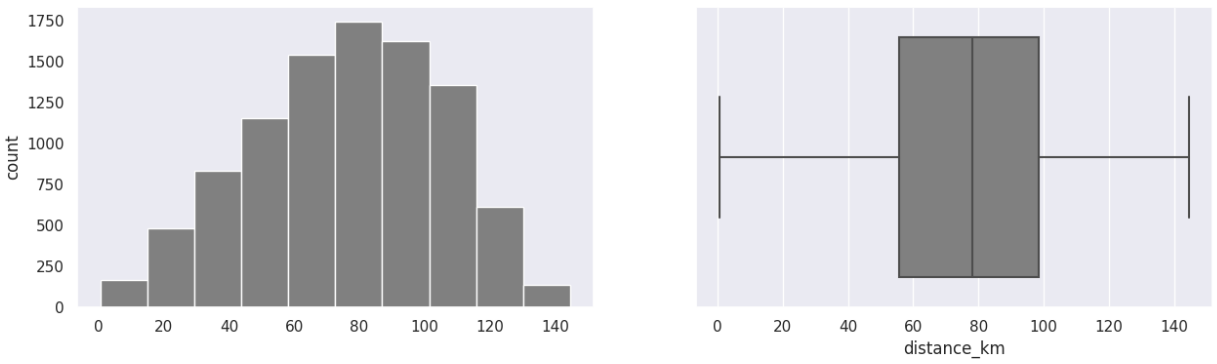


Figure 14. Histogram and Box plot of 'distance_km' for fraudulent transactions(Skew: -0.22
Threshold for Outliers for distance_km : (-8.69, 162.68)
Number of Outliers for distance_km: 0)

The distance between the customers and the merchants that those customers place orders with varies from 0 to 140 km. The frequency of distance goes up from 0 up to its peak at around 80 km (35000 transactions) and then decreases until 140 km. The median of the feature is also around 80 km which means half of the orders with the distance extend 80 km and respectively for the other half.

About the distance of fraudulent cases, the distribution of it looks analogous to the distribution of distance for all transactions. Most of the counterfeit transactions are conducted with the value of distance equal to around 80 km (1750 counterfeit transactions) and decrease as the value of distance gets bigger.

2.2. For Category features, we will use Pie Plot and Count Plot.

For a feature which has a lot of attributes (above 10 attributes) we will only use Count Plot to visualize the top 10 of it and the feature which has under 10 features we will visualize it both by Pie Plot and Count Plot.

Credit card number:

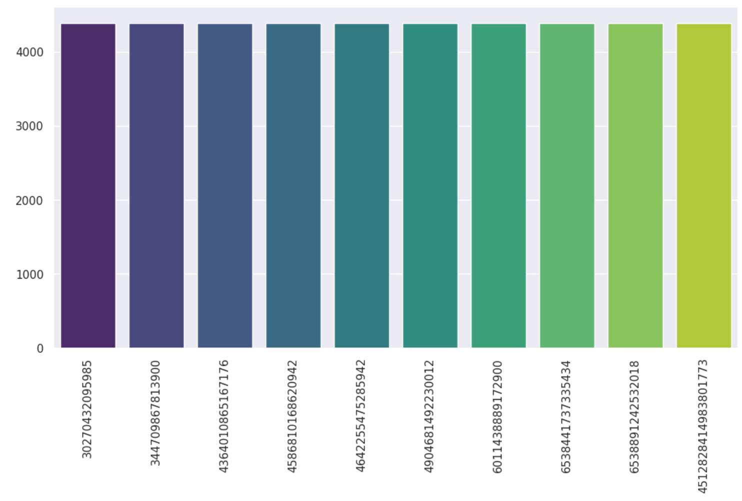


Figure 15. Count Plot of 10 most frequent credit card numbers

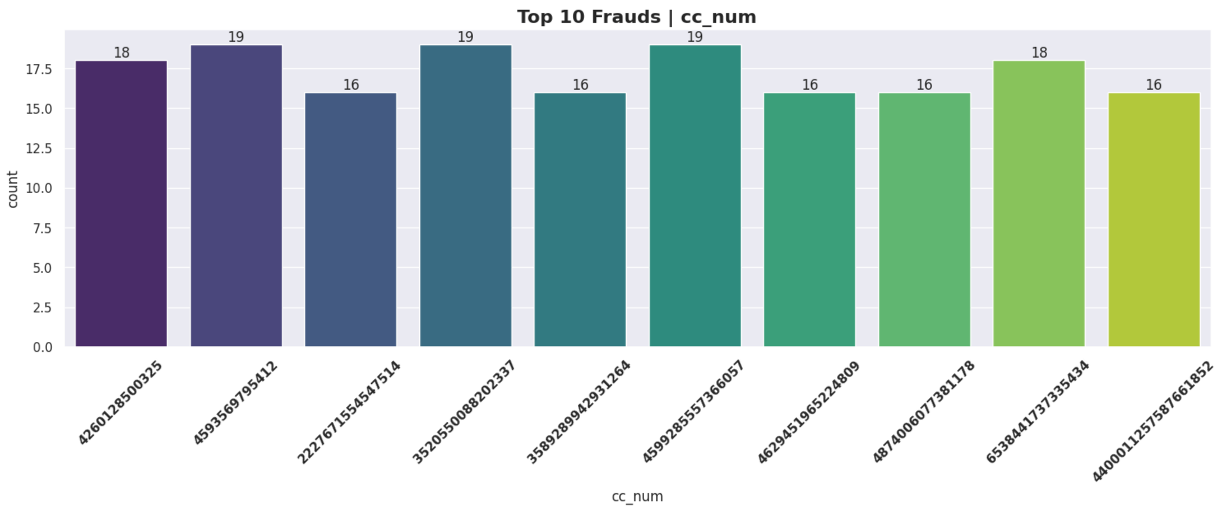


Figure 16. Count Plot of 10 most frequent fraud credit card numbers

Observing the two figures above we can see that the credit card that makes the most transactions is less likely to be associated with fraudulent circumstances. But the observation may incorrected because we only observe the top 10 credit cards in both cases.

Merchant:

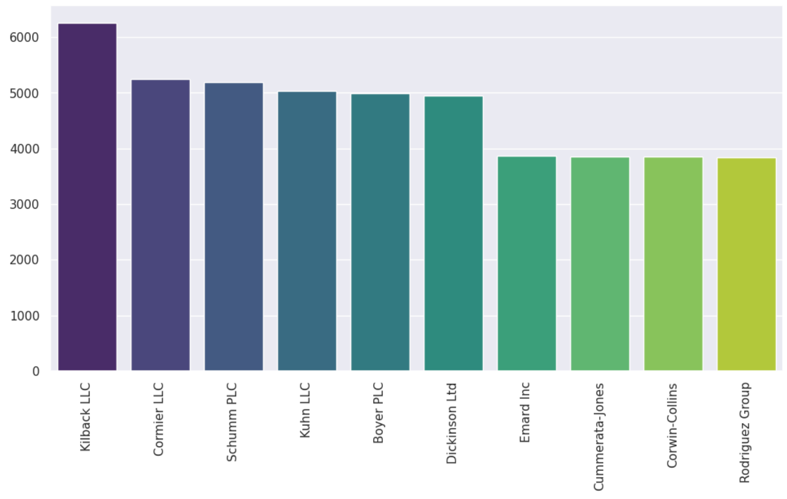


Figure 17. Count Plot of 10 most frequent merchants

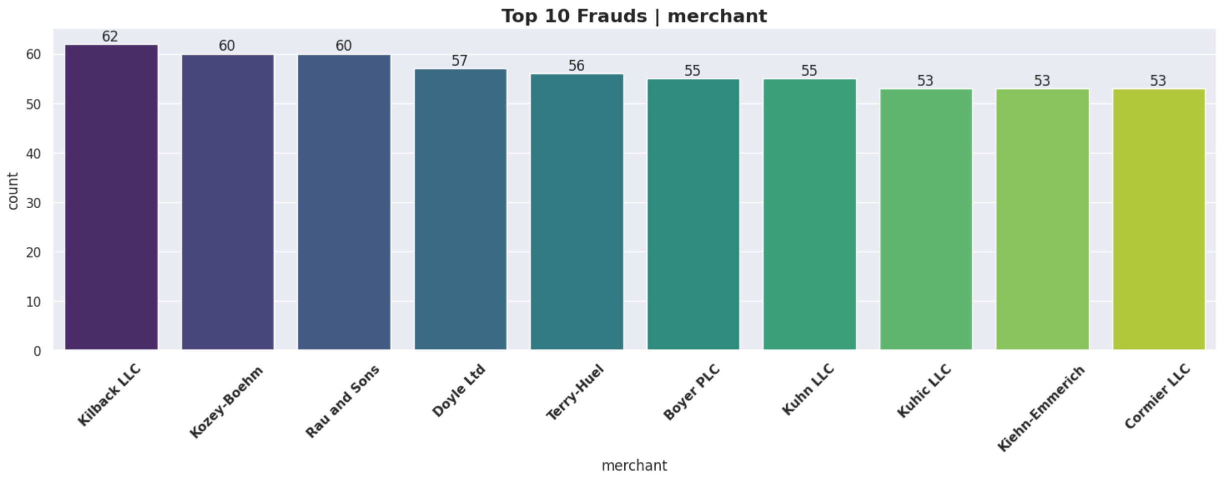


Figure 18. Count Plot of 10 most frequent fraud merchants

Just based on the two figures above we first see that the ‘Killback LLC’ merchant that has the most transactions (above 6000 transactions) also has the most counterfeit transactions (62 transactions). We also have ‘Cormier LLC’ merchant (above 5000 transactions), ‘Kuhn LLC’ merchant (approximately 5000 transactions) and ‘Boyer PLC’ merchant (approximately 5000 transactions) appearing in both figures. So there are 4/10 top merchants that have the most transactions and also have the most fraudulent transactions.

Category:

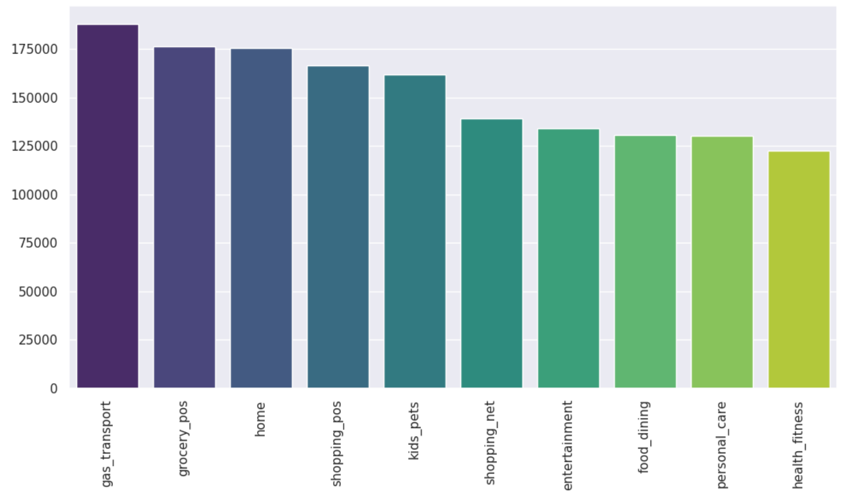


Figure 19. Count Plot of 10 most frequent categories

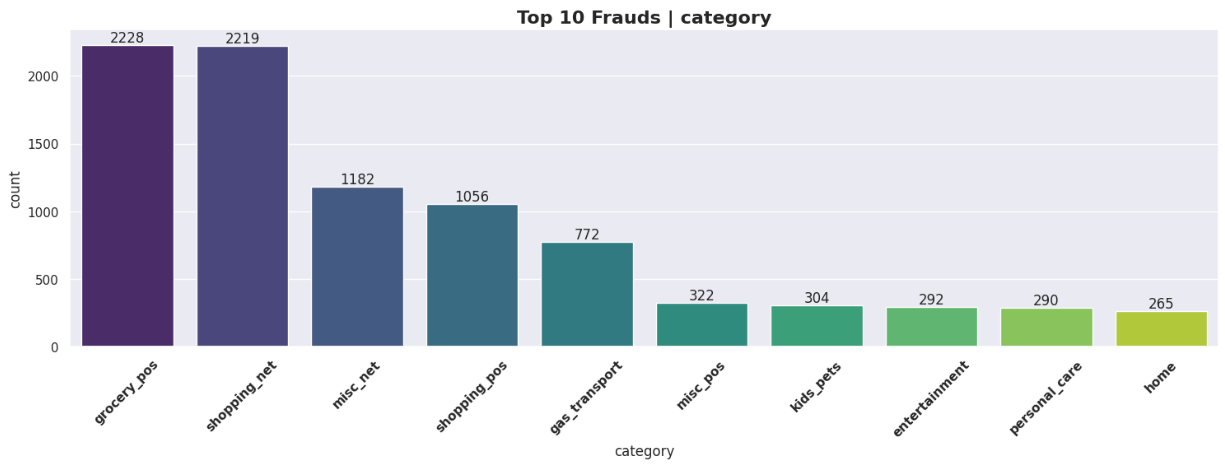


Figure 20. Count Plot of 10 most frequent fraud categories

Two figures show that 8/10 categories which have the most transactions are related to the most fraudulent transactions. Only two categories which are 'misc_net' and 'misc_pos' appear on the top 10 fraudulent categories but do not exist on the top 10 categories for transactions. Two most fraudulent categories are 'grocery_pros' and 'shopping_net' which have approximately 2200 transactions.

Gender:

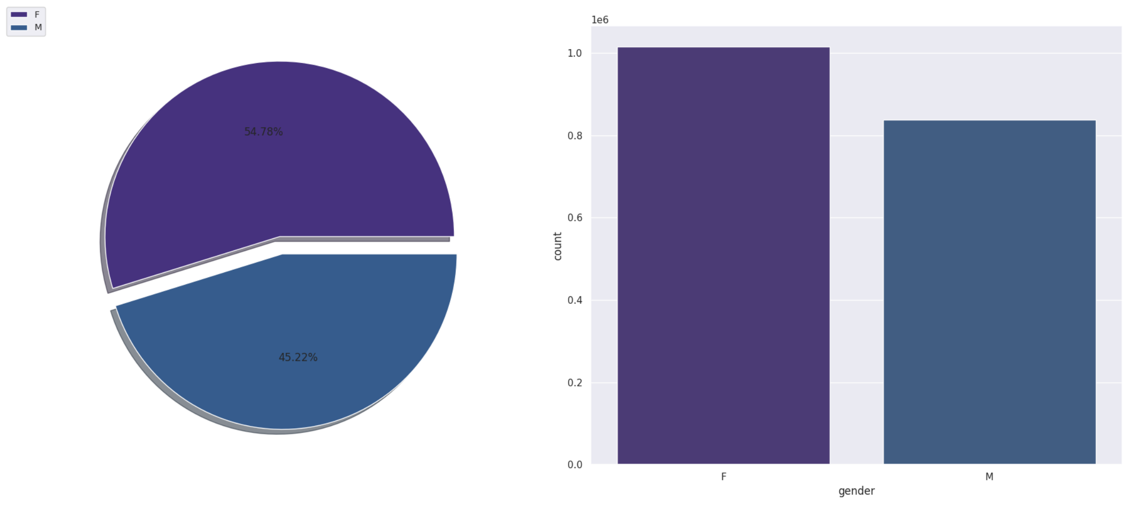


Figure 21. Pie Plot and Count Plot of 'gender' for all transactions (Purple is Female, Blue is Male)

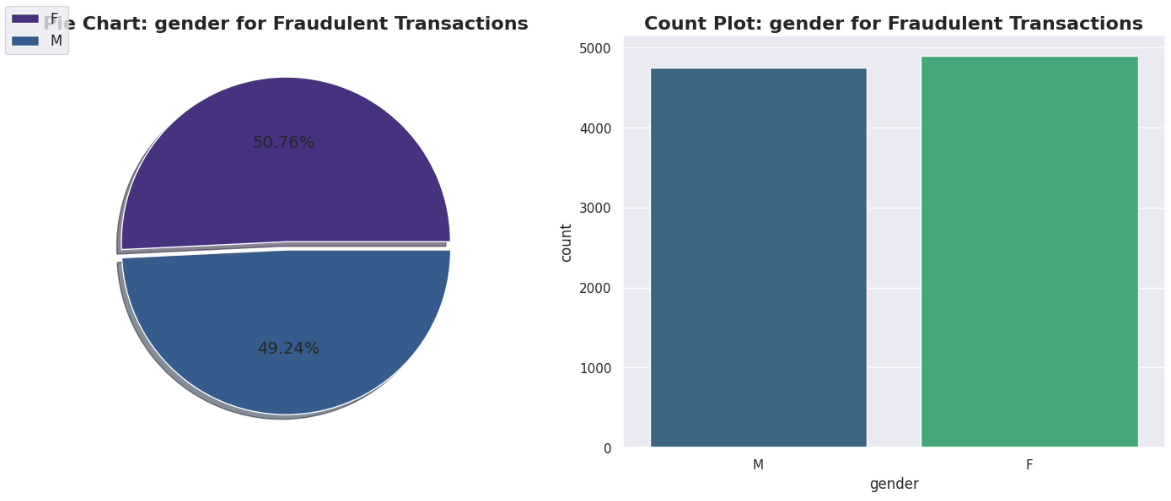


Figure 22. Pie Plot and Count Plot of 'gender' for fraudulent transactions(Purple is Female, Blue is Male)

Overall, the number of female customers (approximately 1000000) are greater than male customers (approximately 800000). But for fraudulent transactions, the amount of two genders is generally equal, which is close to 5000 transactions.

City:

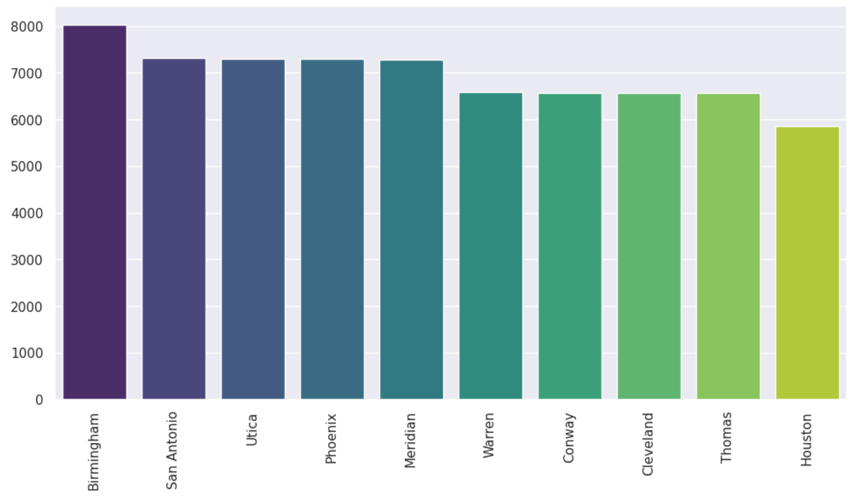


Figure 23. Count Plot of 10 most frequent cities

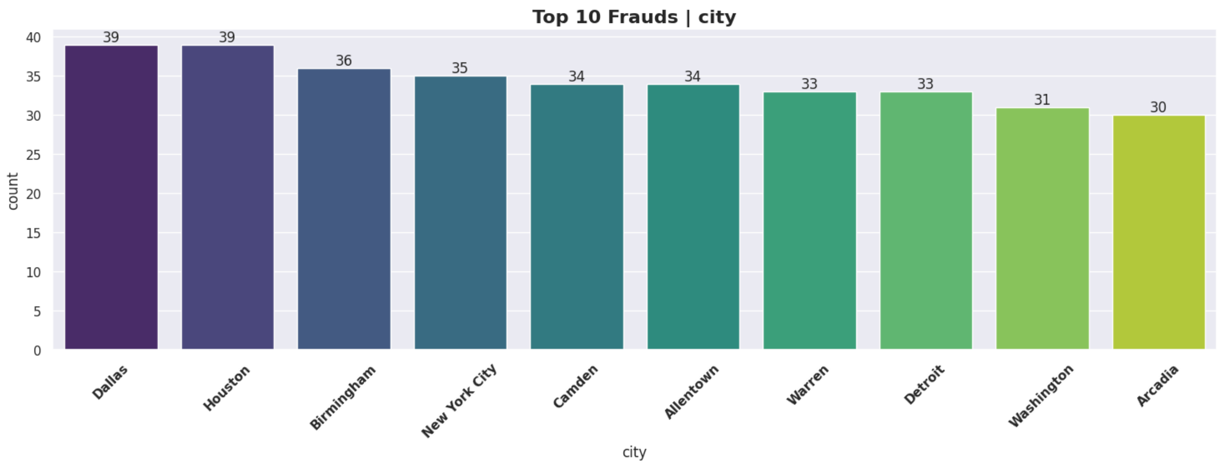


Figure 24. Count Plot of 10 most frequent fraud cities

There are only 3 cities that are associated with two figures which are ‘Houston’, ‘Birmingham’ and ‘Warren’. ‘Birmingham’ has the most number of transactions (8000 transactions) and resulted in 36 fraudulent transactions. ‘Houston’ although has less transactions than ‘Birmingham’ but resulted in a higher amount of fraudulent transactions (39 transactions).

Job:

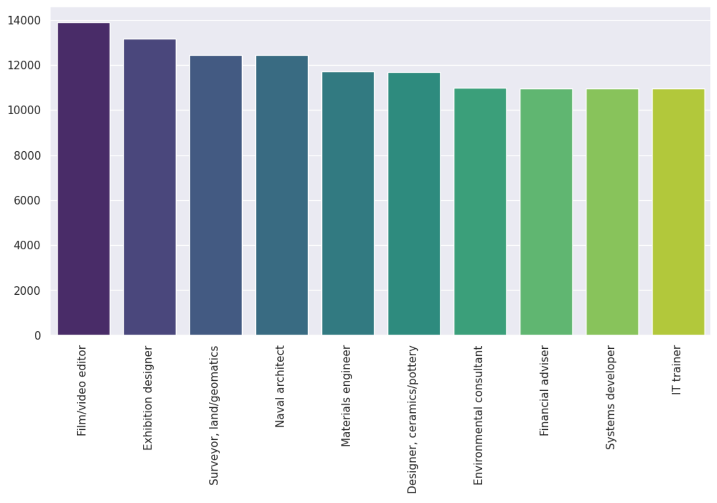


Figure 25. Count Plot of 10 most frequent job

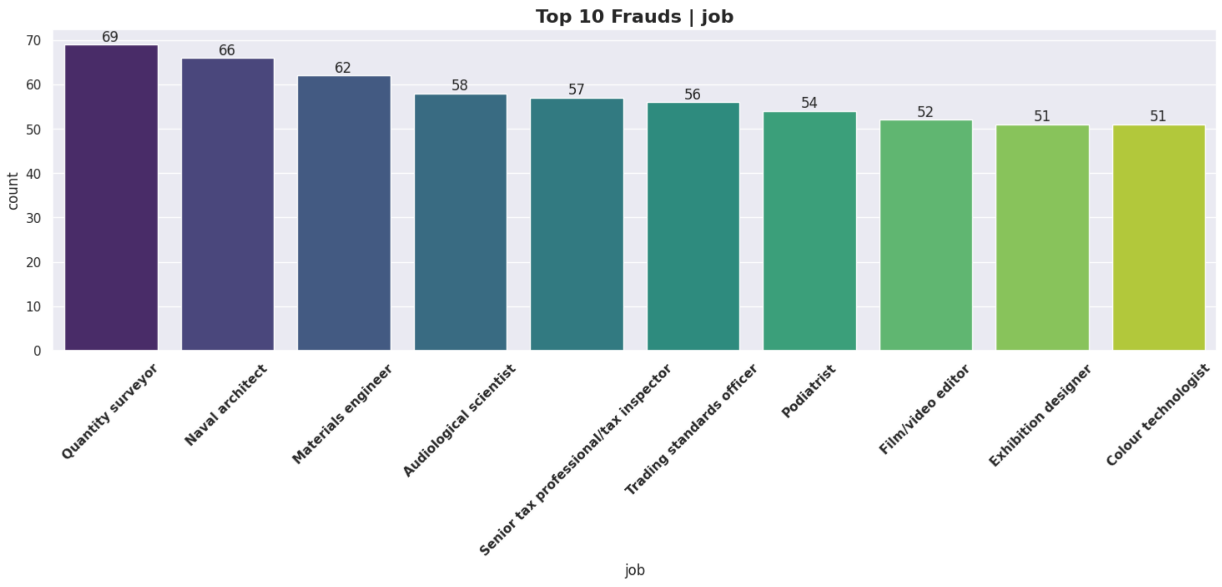


Figure 26. Count Plot of 10 most frequent fraud jobs

Two out of ten jobs in the fraudulent transactions appear also on the most 10 frequent jobs among the transactions. Two of them are ‘naval architect’ (over 12000 transactions) and ‘material engineer’ (also over 12000 transactions) and resulted in 66 fraudulent transactions and 62 fraudulent transactions respectively.

Fraud or not fraud:

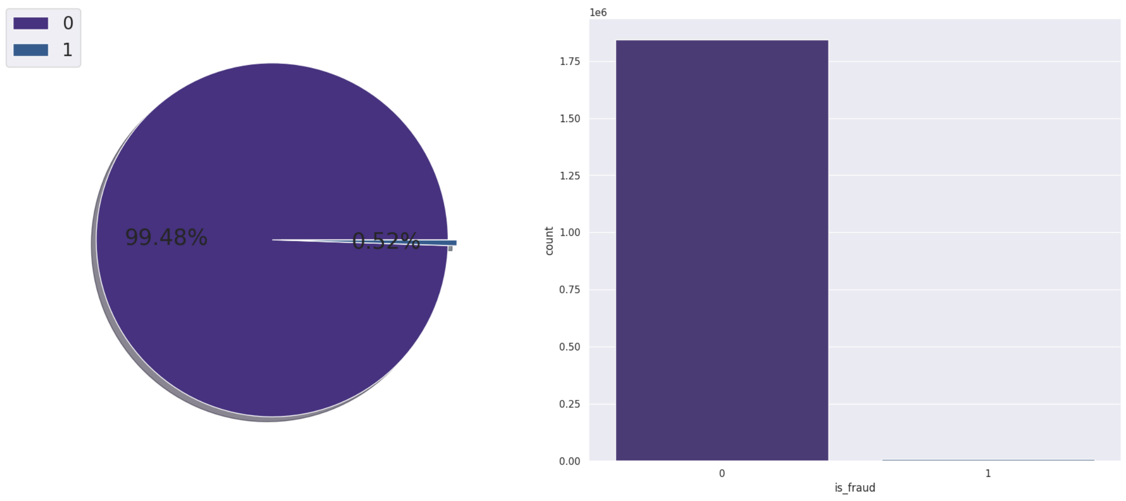


Figure 27. Pie Plot and Count Plot for 'is_fraud' (Purple is Not Fraud, Blue is Fraud)

As we can see from the figure above, the target value of the dataset which is 'is_fraud' is highly imbalanced when the amount of non-fraud transactions overwhelms the amount of fraudulent transactions. So as a result in the model building part we will try to implement the training before and after applying a method to treat the imbalance state of the data.

Day:

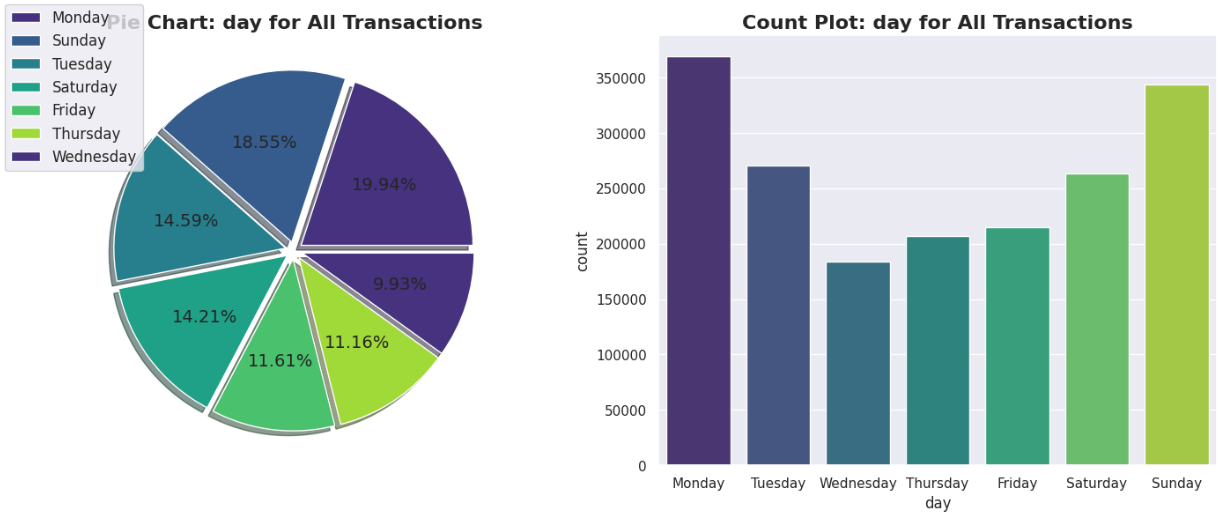
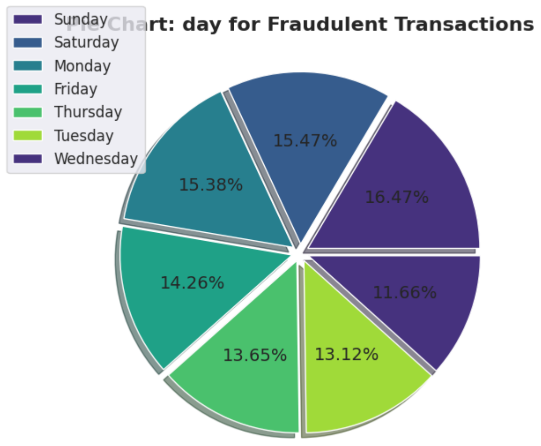


Figure 28. Pie Plot and Count Plot of 'day' for all transactions



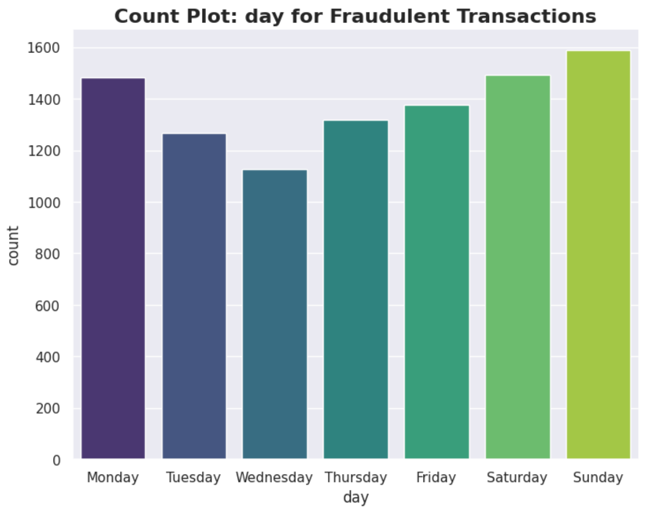


Figure 29. Pie Plot and Count Plot of ‘day’ for fraudulent transactions

From the first figure we can see that the number of transactions has the lowest value on Wednesday (close to 200000 transactions) and continuously rises up until the next Monday (over 350000 transactions). The distribution of ‘day’ for all transactions is pretty similar to fraudulent transactions when the lowest value is on Wednesday (close to 1200 transactions) and then climbs up to the next Monday (over 1400 transactions). But one slight difference is that in all transactions, Monday has the highest amount of transactions but in fraudulent transactions, Sunday is the one who claims the most transactions.

3. Multivariate Analysis

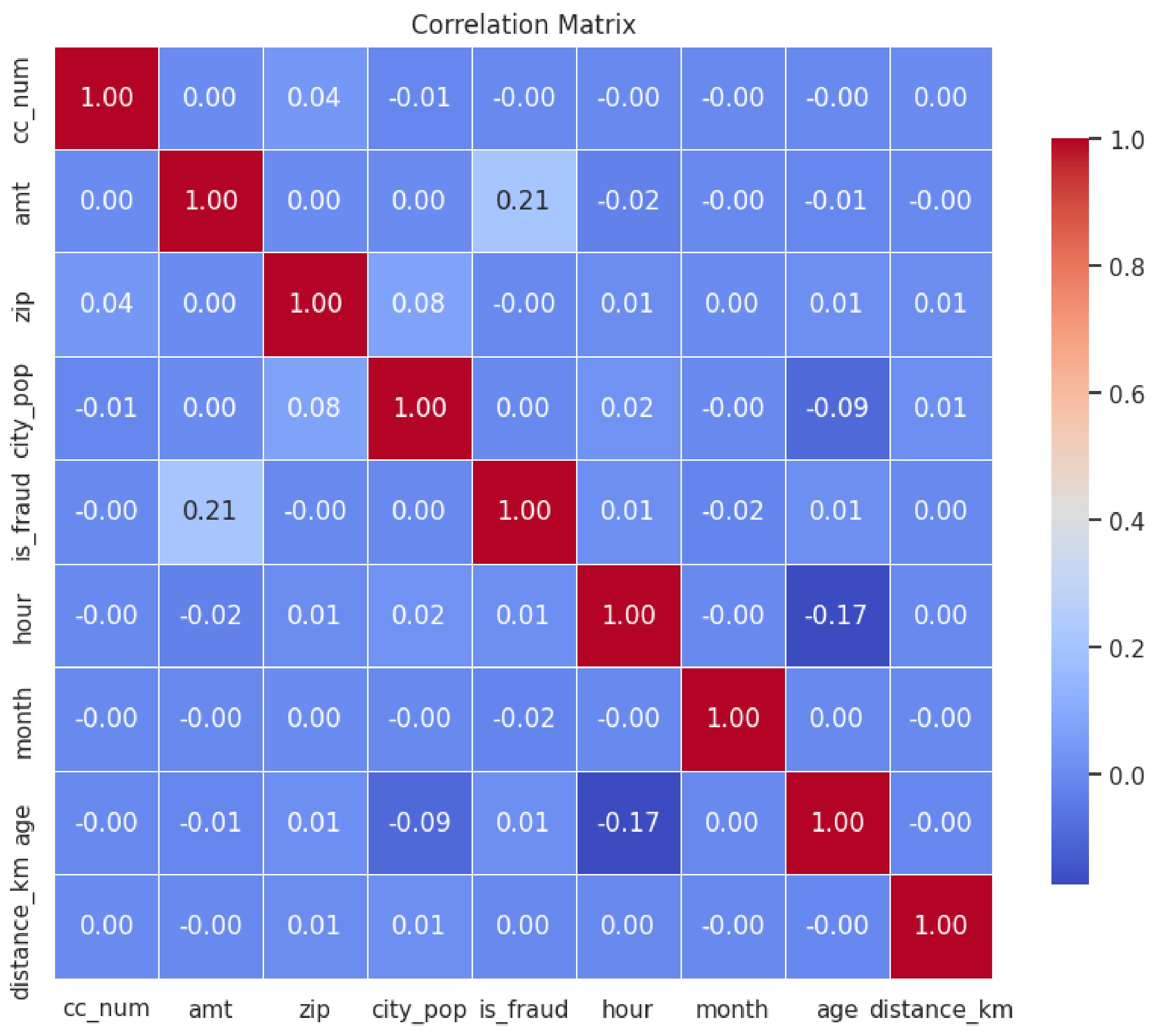


Figure 30. The heat map represents the correlation among attributed features which are numerical

The correlation matrix provides insights into the linear relationships between pairs of variables. Here are some observations based on the correlation coefficients:

- Amount (amt) and Fraud (is_fraud):
    - There is a moderate positive correlation (0.21) between the transaction amount (amt) and the likelihood of fraud (is_fraud). This suggests that higher transaction amounts are somewhat associated with a higher probability of fraud.
- City Population (city_pop) and Age:
    - There is a negative correlation (-0.09) between city population and age. This could indicate that larger cities tend to have a slightly younger population, although the correlation is not very strong.

- Hour and Age:
  - There is a negative correlation (-0.17) between the hour of the transaction and the age of the account holder. This implies that, on average, transactions that occur later in the day are associated with slightly younger account holders.
- Amount (amt) and Hour:
  - There is a negative correlation (-0.02) between the transaction amount and the hour of the transaction. This suggests a weak tendency for higher transaction amounts to occur earlier in the day.

4. P-value for each feature

Feature	P-Value	Significance
cc_num	1.17e-01	not significant
merchant	0.00e+00	significant
category	0.00e+00	significant
amt	0.00e+00	significant
gender	1.97e-15	significant
city	0.00e+00	significant
city_pop	6.63e-01	not significant
job	0.00e+00	significant
hour	2.62e-36	significant
day	7.809807e-60	significant
month	2.82e-104	significant
age	1.21e-43	significant
distance_km	6.22e-02	significant

Table 10. P-values of features

Observing the table illustrate the p_values of each feature, we see that there 2 insignificant features which are 'cc_num' and 'city_pop' so we will take into account for the next preprocessing step.

E. Further Preprocessing

First we will try to delete the ‘city_pop’ column because the p-value of it shown in table 10 indicates that this feature is not significant for classifying fraudulent transactions and we can not extract any information about it.

Next, grouping transactions by credit card number using the groupby**()** function. Within each group, only the 'trans_date_trans_time' column, which presumably contains the transaction times, is selected for calculating time differences using **diff()** function.

We converted the time differences into numerical values representing the hours between transactions applying the **timedelta64()** function.

We generated a feature named “hours_diff_bet_trans” that captures the time intervals between transactions for each credit card, then converted the gender to binary classification: 1 for Male, 0 for Female.

Using **pingouin library** to investigate whether there are statistically significant differences in the time intervals between transactions (“hours_diff_bet_trans”) for fraudulent and non-fraudulent cases:

	T	**p-value**
T-test	27.266899	7.200860e-158

Table 11. T-test and p_value for new generated ‘hours_diff_bet_trans’ feature

The t-test yielded a very high T-statistic (27.27) and an extremely small p-value (7.20e-158), which is essentially zero. This strongly indicates that the difference in the mean hours between transactions for fraudulent and non-fraudulent cases is highly statistically significant.

The bar plot confirms this difference. The bar representing non-fraudulent transactions (the left one) is noticeably higher than the one for fraudulent transactions (the right one).

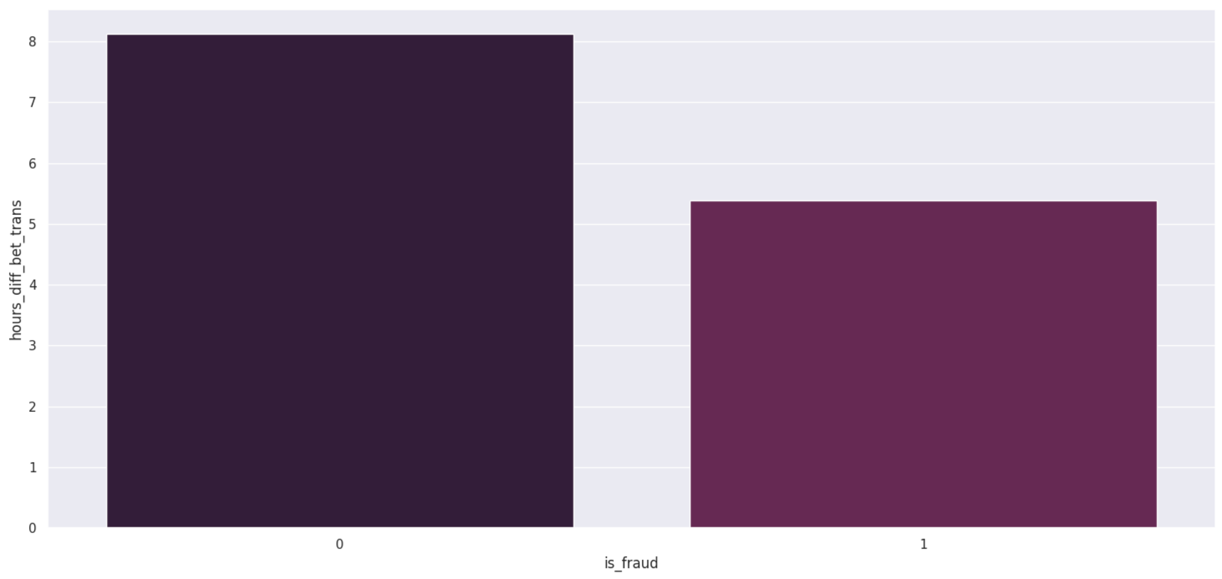


Figure 31. Visualization of the distribution of hours between transactions for both groups

We counted the number of transactions for each card then created a new feature named "cc_freq" that captures the transaction frequency for each credit card. This feature can be valuable for fraud detection, as unusual transaction frequencies might signal potential fraud.

index	**cc_num**	**cc_freq**
1017	60416207185	2196
2724	60416207185	2196
2726	60416207185	2196
2882	60416207185	2196
2907	60416207185	2196

Table12 . Example of the transaction frequency for a single credit card

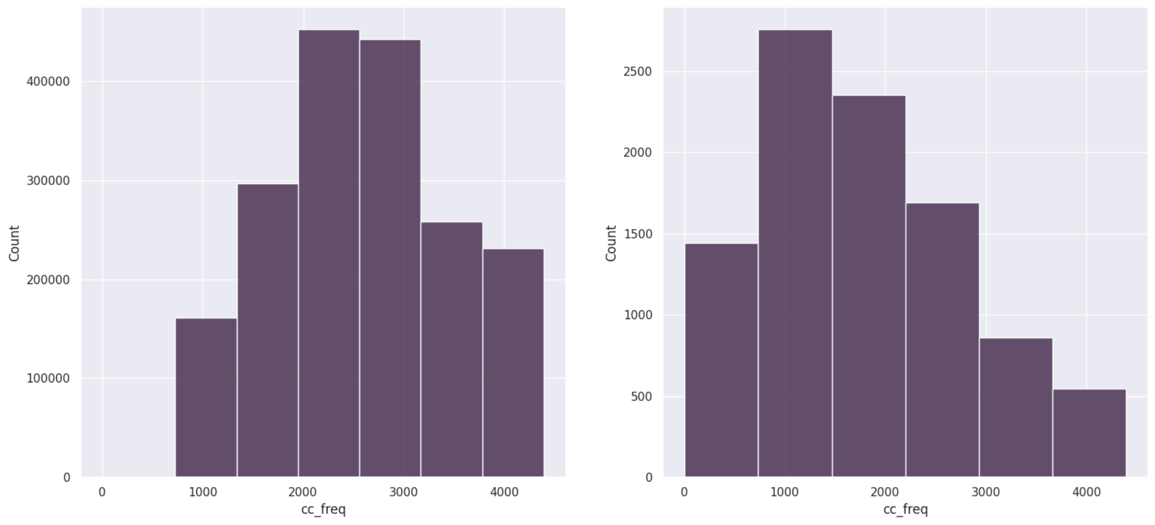


Figure 32. The distribution of the credit card number frequency for non-fraud (left side) and fraud(right side)

The first subplot shows the histogram of the specified column for non-fraud instances. The second subplot shows the histogram of the specified column for fraud instances. The left histogram is roughly symmetrical while the right one skewed to the right, with a longer "tail" extending towards. Clearly frauds occur more in credit cards with less use (new ones) and for genuine transactions, it follows a normal distribution

We categorized transaction frequencies into discrete classes based on defined thresholds which are in the range of 800 to 5000 with the step of 800. The class label are defined as follow:

[1 2 3 4 5 6]

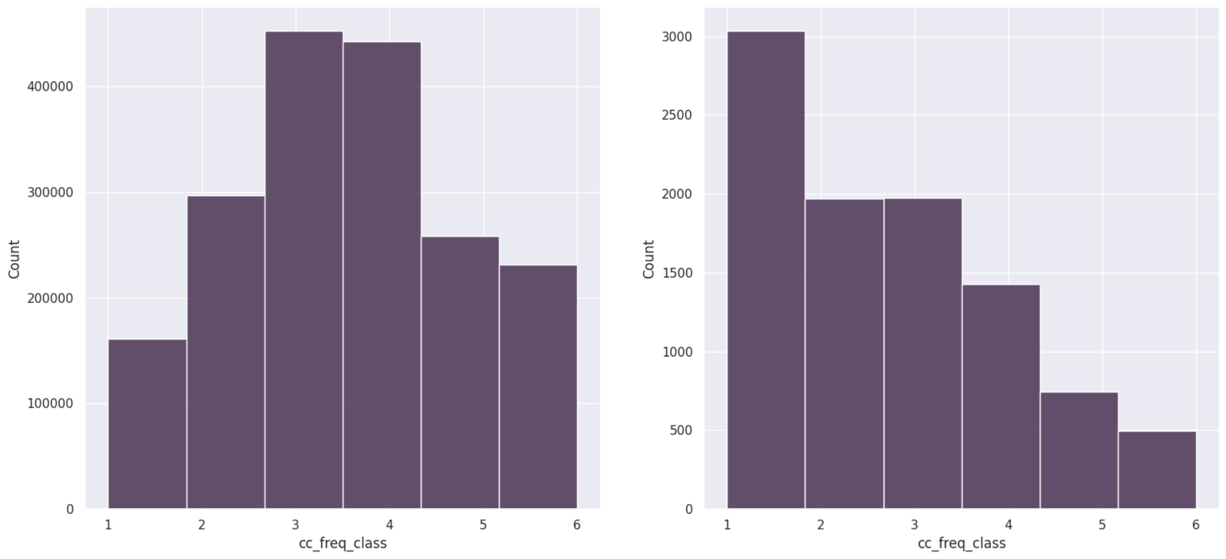


Figure 32. The distribution of the credit card number frequency classes for non-fraud (left side) and fraud(right side)

So in this figure 32, we clearly see that the amount of fraudulent credit cards is the credit card that belongs to the lower class(which are not used usually) and the credit cards of class 5 or 6 are less likely to be fraudulent credit cards.

We applied WOE encoding (Weight of Evidence) for encoding categorical variables in a way that captures their relationship with the target variable (whether fraud or not). The transformed values are then assigned back to the same column in the original dataset.

- WOE > 0: The category is more likely associated with fraud.
- WOE < 0: The category is more likely associated with non-fraud.

	cc_freq	**cc_freq_class**	**city**	**job**	**age**	**gender**	**merchant**
1017	2196	3	-0.193426	0.163804	33	0	0.472059
2724	2196	3	-0.193426	0.163804	33	0	-0.499771
2726	2196	3	-0.193426	0.163804	33	0	-0.132212
2882	2196	3	-0.193426	0.163804	33	0	-0.620062
2907	2196	3	-0.193426	0.163804	33	0	-1.29619
...	...	...	...	...	...	...	...
1849259	2922	4	-0.528658	0.253157	64	1	0.301146
1849567	2922	4	-0.528658	0.253157	64	1	1.088405
1850234	2922	4	-0.528658	0.253157	64	1	-0.164096
1850235	2922	4	-0.528658	0.253157	64	1	1.013009
1850558	2922	4	-0.528658	0.253157	64	1	-0.860991

Table 13. Dataset after applying calculations

category	**distance_km**	**month**	**day**	**hour**	**hours_diff_bet_trans**	**amt**	**is_fraud**	**split**
0.92586	127.61	1	Tuesday	12	0	7.27	0	train
-0.238221	110.31	1	Wednesday	8	19	52.94	0	train
-0.238221	21.79	1	Wednesday	8	0	82.08	0	train

-1.019735	87.2	1	Wednesday	12	3	34.79	0	train
-1.238421	74.21	1	Wednesday	13	0	27.18	0	train
...	...	...	...	...	...	...	...	...
-0.238221	44.89	12	Thursday	2	2	66.11	0	test
0.92586	81.49	12	Thursday	5	3	4.58	0	test
-0.238221	36.05	12	Thursday	11	6	95.96	0	test
0.894448	81.77	12	Thursday	11	0	149.48	0	test
-1.019735	22.08	12	Thursday	13	2	20.55	0	test

Table 14. Dataset after applying calculations

After we have the dataset filled with numerical values. We will split the dataset into a train set and test set based on the value of 'split' feature. The proportion of the dataset is represented in the figure 33 below.

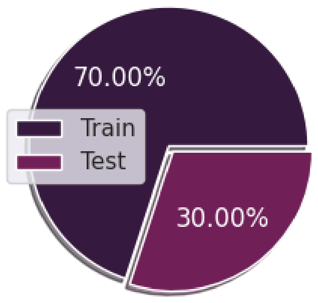


Figure 33. The proportion of the training and testing sets

But there is just one more thing before we can go to the model building step that we will treat the features that have a lot of outliers data points. After EDA, we can see from figure 3 that the feature 'amt' has an enormous amount of outliers and also we will try to plot the data point in newly generated 'hours_diff_bet_trans' features.

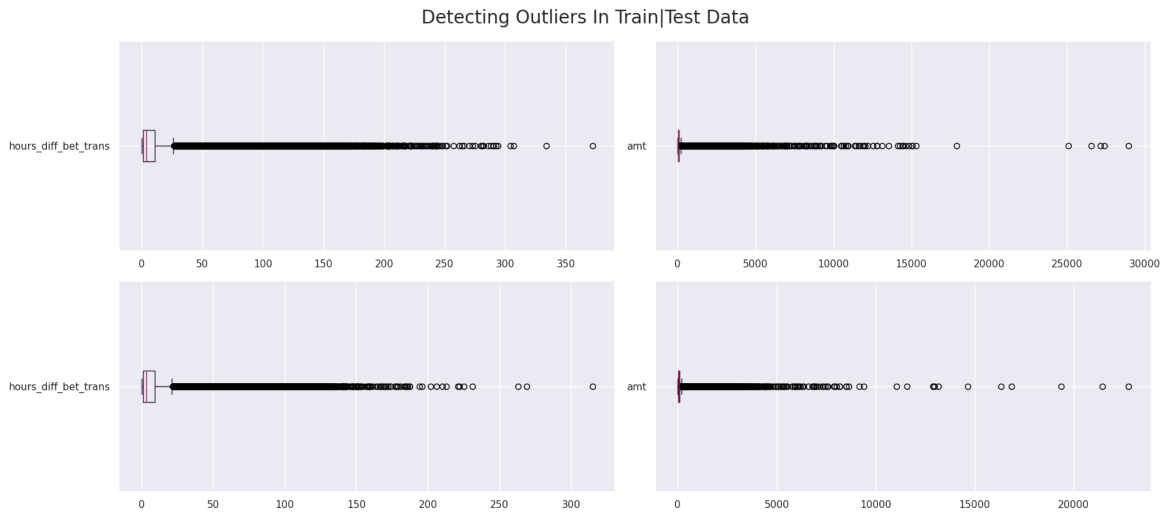


Figure 34. Box plot for ‘hours_diff_bet_trans’ and ‘amt’ features on the train set (row above) and test set (row below)

Figure 34 illustrates the presence of numerous outliers concentrated on the right side of the upper whisker. The skewed distributions, particularly prominent in fraudulent transactions, exhibit a substantial weighting toward typical amounts, punctuated by a few extreme values (outliers). To address this distributional imbalance and enhance the suitability of the data for analysis, a log scale transformation will be applied. The **log1p()** function, which computes the natural logarithm of (1 + number), is chosen for this purpose. This approach is a recognized technique for mitigating the impact of outliers, reducing skewness, and gracefully handling zero values in the data.

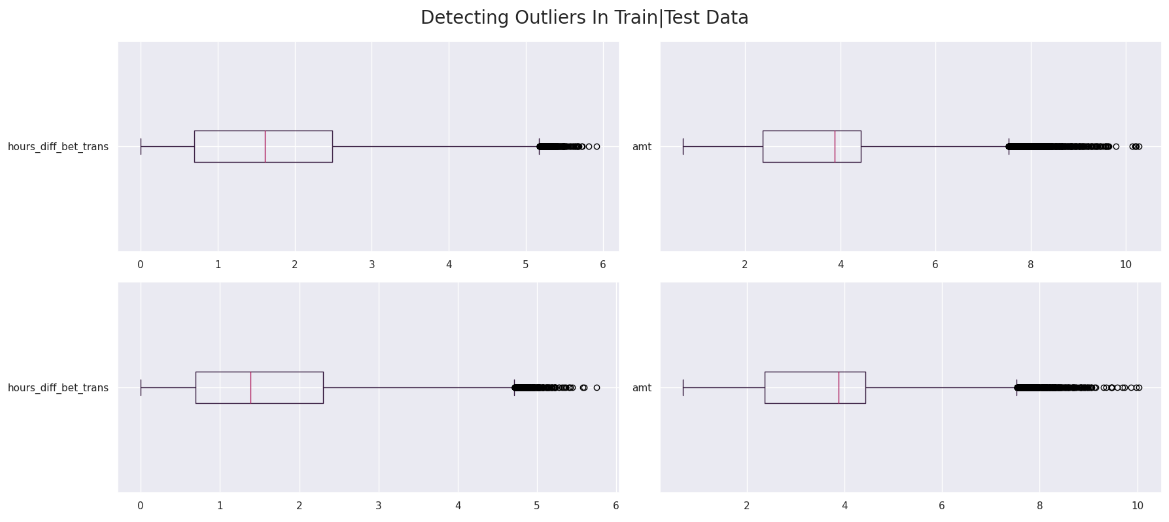


Figure 35. Box plot for ‘hours_diff_bet_trans’ and ‘amt’ features on the train set (row above)

and test set (row below) after applied ***log1p()*** *function*

It’s evident that the application of a log transformation results in a more symmetrical distribution and lessens the influence of outliers (which we have not eliminated). We could process these data separately, but for now, log functions are frequently employed to manage right-skewed distributions. Once our model is applied, we can use the exp function to revert the data back to its original form.

F. Model Building and Evaluation

In the EDA process we know that our dataset is imbalanced so we will try to implement models and training and classifying data before we apply a method to treat the imbalance circumstance and then we obtain the metrics then we apply a method called SMOTE which will generate synthetic samples for the minority class by interpolating between existing instances, thereby addressing class imbalance in a dataset.

1. Specification of models

1.1. Logistic Regression.

The logistic regression model is built using the LogisticRegression class from scikit-learn library and takes several parameters that control the behavior of the logistic regression model. And here is some important parameters and their explanation

- **Penalty** = L2 regularization: This parameter specifies the type of regularization to be applied. Regularization helps prevent overfitting by adding a penalty term to the loss function. And in this dataset ridge regularization is used by adds the squared values of the coefficients as a penalty.
- **C** = 1: The inverse of the regularization strength. It controls the amount of regularization applied to the model. Smaller values of C increase the regularization strength, while larger values decrease it. It's worth noting that C is the inverse of the regularization parameter (1 / lambda).
- **solver** = L-BFGS : This parameter specifies the algorithm to be used for optimization. Different solvers are suitable for different types of problems. Because this dataset is medium size so Limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm is used.
- **max_iter** = 125: The maximum number of iterations for the solver to converge. It determines the maximum number of iterations the optimization algorithm will run

before stopping. If the algorithm does not converge within max_iter iterations, it will terminate without reaching the optimal solution.

In this logistic regression model, the max iteration was set to 125, 125 iterations may be too low for a certain dataset but for this dataset, 125 is a decent value because the model converges within these values.

1.2. Decision Tree

The Decision Tree model is built using the DecisionTreeClassifier from scikit-learn library and takes several parameters that control the behavior of the Decision Tree model. There are some important parameters and their explanation:

- **criterion** = ‘gini’ **:** The function to measure the quality of a split. In our model, we use the default value “gini” which is Gini impurity.
- **splitter** = ‘best’ **:** The strategy used to choose the split at each node. The value of the splitter is “best” which will choose the best split.
- **Max_depth** = 13 **:** The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.After optimizing and testing we decided to choose the value as 13.

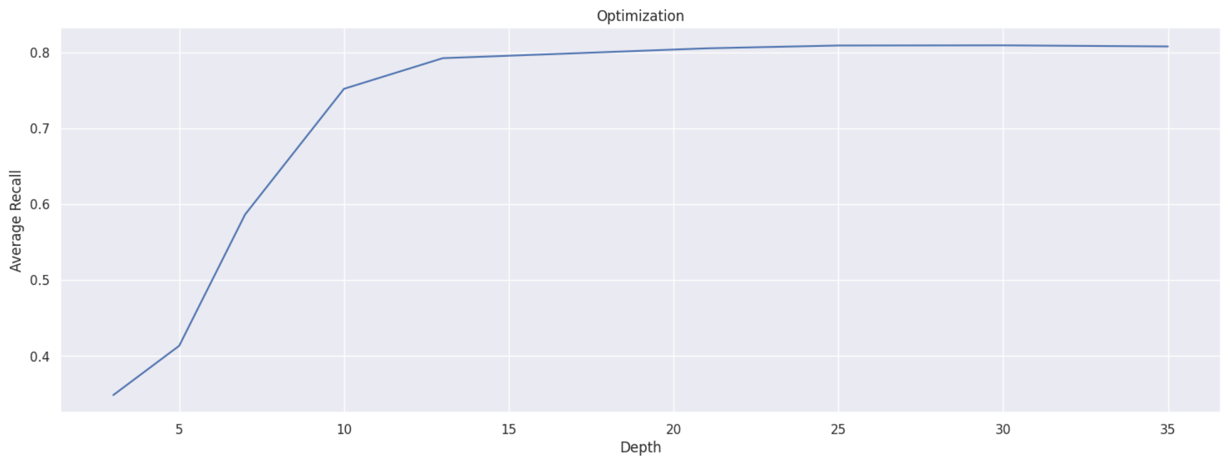


Figure 36. Graph to find the best depth for the Decision Tree model. The graph measures the value of Average Recall in each depth of the model

As you can see in the figure 36 above, at the value 13 of Depth, the value of “ Average Recall ” has slowly changed, the slope is decreasing rapidly. From 13 to 35, the slope is nearly a linear line. So we decided to use the value 13 for max_depth parameter in the class DecisionTreeClassifier imported from sklearn.tree.

- **random_state** = 10**:** Controls the randomness of the estimator. In our model we set this parameter’s value to 10

- **max_leaf_nodes** = None **:** The maximum number of leaf nodes. Best nodes are defined as relative reduction in impurity. The value is None then unlimited number of leaf nodes.

1.3. Random Forest

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the **max_samples** parameter if **bootstrap=True** (default), otherwise the whole dataset is used to build each tree.

- **n_estimators** = 100**:** The number of trees in the forest.
- **Criterion** = 'gini'**:** The function to measure the quality of a split in this case is Gini impurity.
- **max_depth** = 13**:** The maximum depth of the trees. Controls the maximum depth of each decision tree in the forest.The value '13' obtained from the figure 36.
- **Min_samples_split** = 2 **:**The minimum number of samples required to split an internal node. It prevents the tree from splitting nodes with a small number of samples.
- **Bootstrap** = True**:** To make bootstrap samples are used when building trees.
- **random_state** = 10**:** Seed for random number generation. Ensures reproducibility of results.
- **max_leaf_nodes** = None **:** The maximum number of leaf nodes. Best nodes are defined as relative reduction in impurity. The value is None then unlimited number of leaf nodes.

Random Forest algorithm was also trained, it optimized the number of trees hyper parameter. The model was built by changing the values of this parameter every time in 100 and 200. The proper results show that the best number of trees was 100 trees.

2. Evaluation

In the context of classifying card transactions for fraud detection, the primary focus is on minimizing false negatives to avoid missing actual fraud cases. Recall, also known as sensitivity or the true positive rate, takes precedence in this use case. This emphasis on recall is justified due to imbalanced data, where the number of non-fraudulent transactions significantly outweighs fraudulent ones. False negatives, denoting fraudulent transactions incorrectly classified as non-fraudulent, carry a high cost in credit card fraud detection, leading to potential financial losses for cardholders and issuers. Prioritizing sensitivity over precision aims to identify the majority of fraud cases, even at the expense of a higher false positive rate. The choice of evaluation metrics aligns with business goals, and while recall is crucial, it's essential to consider other metrics like precision, F1 score, and AUC-ROC based on specific requirements and trade-offs. Ultimately, the selection of metrics should be in harmony with business objectives and the associated costs of false positives and false negatives.

2.1. Before the application of SMOTE

2.1.1. Logistic regression

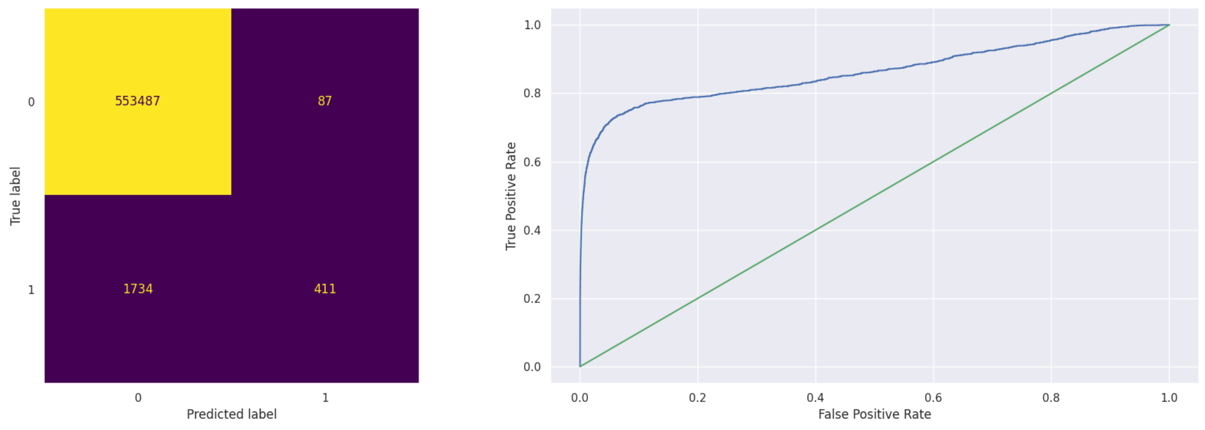


Figure 37. Confusion matrix and AUC-ROC curve of imbalanced data for Logistic Regression model.

Recall score	0. 1916083916083916
Precision	0.8253012048192772
Accuracy score	0.9967231640451379
AUC score	0.8631322009132382
F1-score	0.3110102156640182
Running time	0.19 Mins

Table 15. Evaluation metrics of Logistic Regression model

Conclusions (from imbalanced data), since we focus on the best accuracy for the model to predict the real fraud transaction and don't predict non-fraud and we detect it from the (True Positive Rate 'Recall') the model logistic Regression gives a very weak recall accuracy (about almost 0.2). The AUC score (0.8631) gives us a quality result which means it defines the number of True Positives and True Negatives more than False Positives and False Negatives. The ROC curve also looks good.

2.1.2. Decision Tree

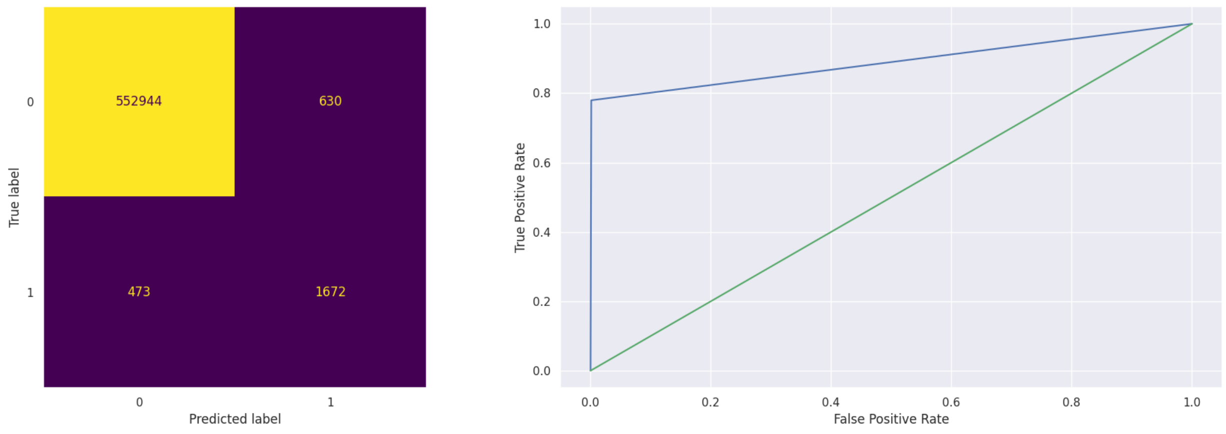


Figure 38. Confusion matrix and AUC-ROC curve of imbalanced data for Decision Tree model.

Procedure	**Score**
Recall Score	0.77948
Precision	0.72632
F1-Score	0.75196
Accuracy Score	0.99801
AUC Score	0.88917
Running Time	0.23 Mins

Table 16. Evaluation metrics of Decision Tree model

Overall, each metric score has an average score around 0.75 except Accuracy Score and AUC Score. Accuracy is the metric that we will significantly evaluate because it does not play an important role in our topic. AUC score seems to be satisfactory but the ROC curve is creating a triangle with the Random classifier so that indicates that the model has been overfitted.

2.1.3. Random Forest

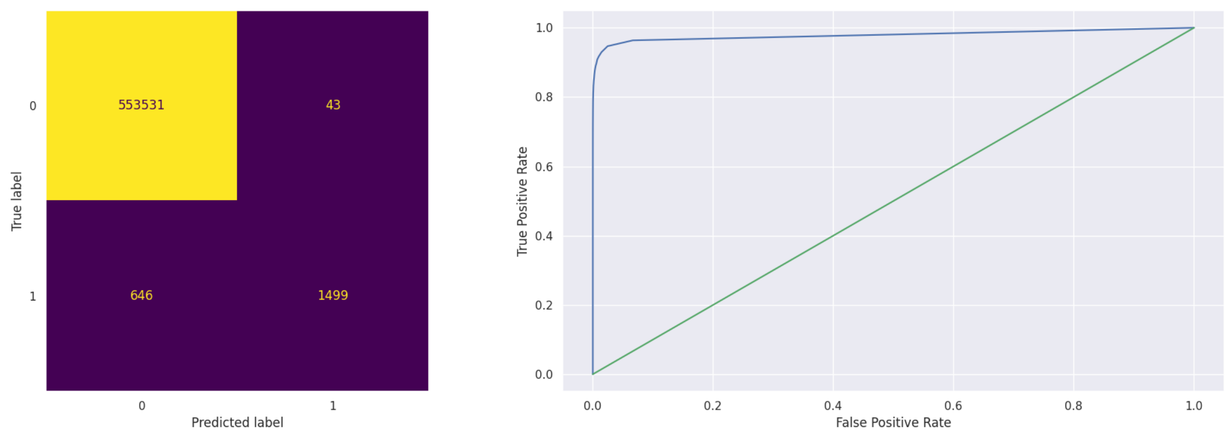


Figure 39. Confusion matrix and AUC-ROC curve of imbalanced data for Random Forest model.

Procedure	**Score**
Recall Score	0.6988
Precision	0.972
F1-Score	0.813
Accuracy Score	0.998
AUC Score	0.978
Running Time	5.5 Mins

Table 17. Evaluation metrics of Random Forest model

The metrics above indicate that the recall score is equal to 0.7 which is quite good for detecting a fraudulent transaction and also to avoid predicting a false fraudulent transaction.

The precision score is extensively high so there are less percentage of missing the fraudulent transaction. The AUC score gives us an outstanding result with 0.978 that means the model can correctly distinguish between all the Positive and the Negative class points.

2.2. After the application of SMOTE

After applying SMOTE to our dataset, the data point of non-fraud and fraud will be equal.

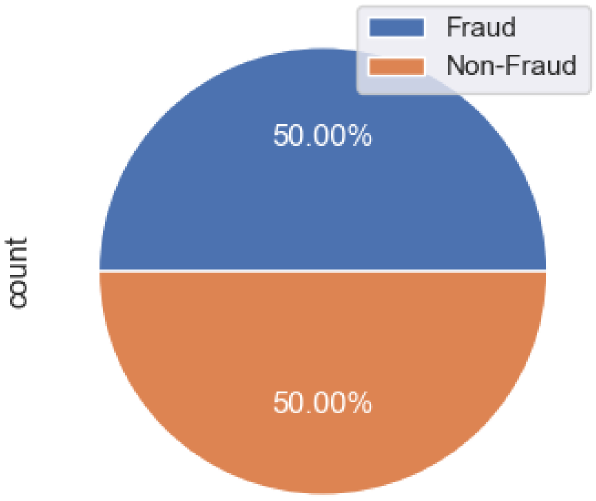


Figure 40. Balanced sampling after applying SMOTE

2.2.1. Logistic regression

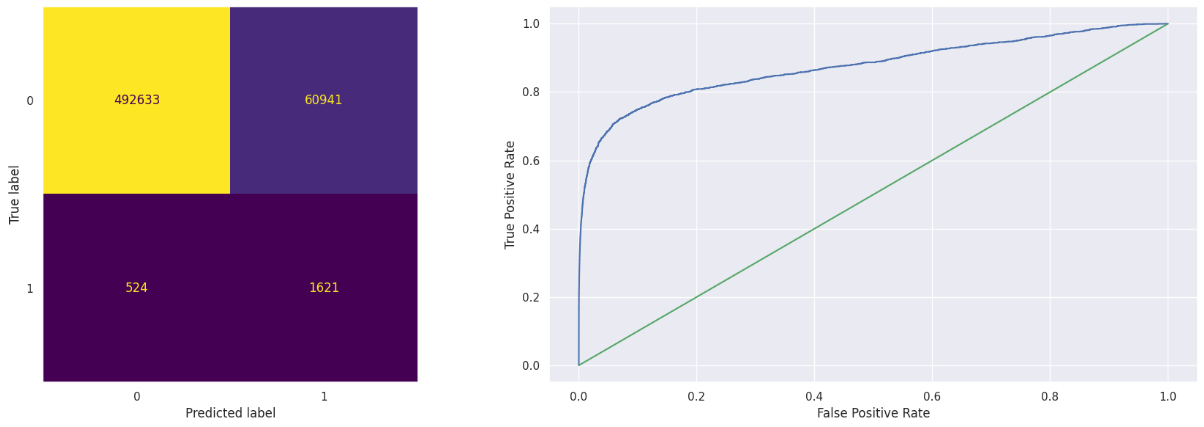


Figure 41. Confusion matrix and AUC-ROC curve of imbalanced data for Logistic Regression model after SMOTE

Recall score	0.7557109557109557
Precision	0.025910296985390494
Accuracy score	0.8093955398321813
AUC score	0.8748796679324486
F1-score	0.0501027709521381
Running time	0.92 Mins

Table 18. Evaluation metrics of Logistic Regression model after SMOTE

Compute the evaluation after applying SMOTE and before applying SMOTE we can see that: before data is balanced, the recall is very low (almost 0.2) but the precision and accuracy is too high about 0.82 – 0.99, that means that the model is overfitting. And after SMOTE was applied the recall score significantly increased to 0.7557, indicating that the model improved in its ability to capture more positive cases but precision dropped to 0.0259, indicating that the model generated a high number of false positives. Moreover, the accuracy also dropped from 0.99 to 0.80 and the AUC score stayed the same.

2.2.2. Decision Tree

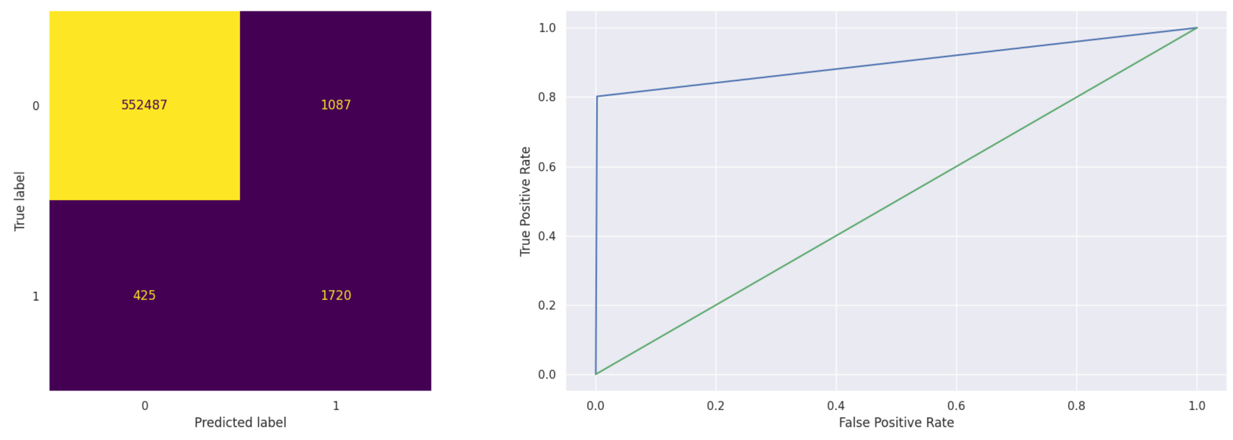


Figure 42. Confusion matrix and AUC-ROC curve of imbalanced data for Decision Tree model after SMOTE

Procedure	**Score**

Recall Score	0.80986
Precision	0.61275
F1-Score	0.69466
Accuracy Score	0.99727
AUC Score	0.89995
Running Time	1.15 Mins

Table 19. Evaluation metrics of Decision Tree model after SMOTE

Compare Before and after using smote

- **Recall Score Improvement**: The recall score has increased after applying SMOTE, suggesting a better ability to identify positive instances.
- **Precision, F1-Score Trade-off:** While recall improved, there is a trade-off with other metrics, which slightly decreased.
- **Accuracy, AUC Score**: There is a insignificant change in value of these two metrics and they still evaluate almost the same as before applying SMOTE.

2.2.3. Random Forest

And after we redo the process one more time:

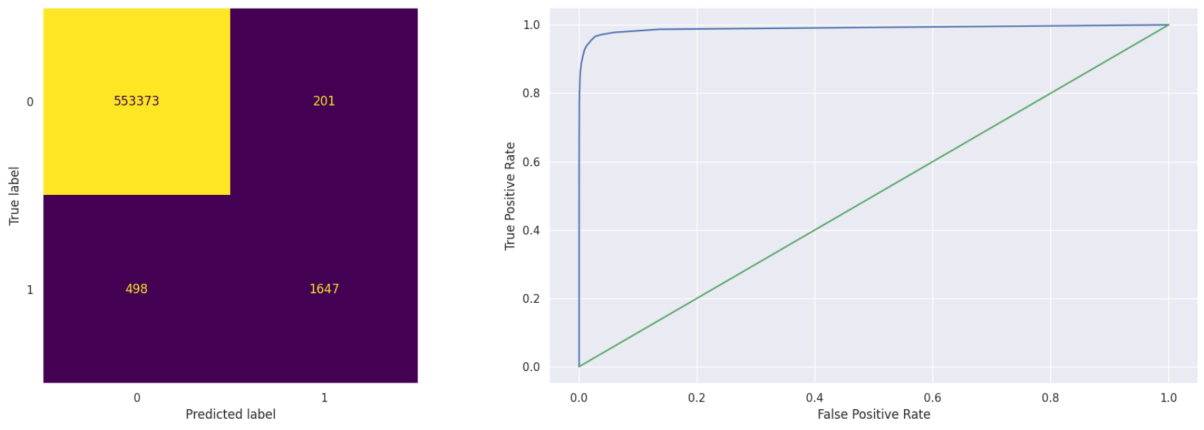


Figure 43. Confusion matrix and AUC-ROC curve of imbalanced data for Random Forest model after SMOTE

Procedure	**Score**

Recall Score	0.7678
Precision	0.8912
F1-Score	0.8249
Accuracy Score	0.9987
AUC Score	0.9901
Running Time	22.05 Min

Table 20. Evaluation metrics of Random Forest model

Compare Before and after using smote

- **Recall Score Improvement**: The recall score has increased after applying SMOTE, suggesting a better ability to identify positive instances.
- **Precision Trade-off:** While recall improved, there is a trade-off with precision, which slightly decreased.
- **F1-Score Improvement:** The F1-Score has improved, indicating a better balance between precision and recall.
- **Accuracy Stability:** The accuracy score remains high and stable before and after SMOTE.
- **AUC Score Improvement:** The AUC score has improved, indicating better overall model performance in distinguishing between classes.
- **Running Time Increase:** The running time has increased after applying SMOTE, as expected due to the additional steps involved in generating synthetic samples.

2.3. Calculate time complexity

2.3.1 Logistic Regression

Time complexity:

- Before SMOTE applied:

O (n x m x iter) = O (1299675x 15 x 125) = O (2436890625)

Whereas:

n: number of samples = 1299675

m: number of features = 15

iter: number of iterations until convergence = 125

- After SMOTE applied:

O (n x m x iter) = O (2578338x 15 x 125) = O (4834383750)

Whereas:

n: number of samples = 2578338

m: number of features = 15

iter: number of iterations until convergence = 125

The time complexity after applying SMOTE is significantly higher than before SMOTE due to the increased number of samples. SMOTE generates synthetic examples to address class imbalance, effectively increasing the number of minority class instances. As a result, the time required for training and evaluating the model increases proportionally to the number of samples.

2.3.2 Decision Tree

- Before SMOTE applied:

O(n × m × log(n)) = O(1296675 x 15 x log(1296675)) = O (118895329.732)

Whereas:

n = number of samples = 1296675

m = number of features = 15

- After SMOTE applied:

O (n x m x log(n)) = O (2578338 x 15 x log(2578338)) = O (247959017.468)

Whereas:

n: number of samples = 2578338

m: number of features = 15

2.3.3 Random Forest

- Before SMOTE applied:

O(n × m × k x n x log(n)) = O(1296675 x 15 x 100 x 1296675 x log(1296675))

= O (1.541686e+16)

Whereas:

n = number of samples = 1296675

m = number of features = 15

k = number of trees = 100

- After SMOTE applied:

O (n x m x k x n x log(n)) = O (2578338 x 15 x 100 x 2578338 x log(2578338))

= O (6.3932216e+16)

Whereas:

n: number of samples = 2578338

m: number of features = 15

k = number of trees = 100

2.4. Compare 3 models

	Logistic Regression	**Random Forest**	**Decision Tree**
Recall Score	0.1916	0.6988	0.77948
Precision	0.8253	0.972	0.72632
F1-Score	0.311	0.813	0.75196
Accuracy Score	0.9967	0.998	0.99801
AUC Score	0.8631	0.978	0.88917
Running Time(mins)	0.19	5.5	0.23

Table 21. Evaluation metrics of 3 models before applying SMOTE

Accuracy score indicates the ratio of correctly predicted observations (both true positives and true negatives) to the total observations.

- All models achieve high accuracy (above 0.996), but this can be misleading due to the imbalance between fraudulent and normal transactions.

Therefore, we focus on recall, precision, and AUC for better assessment in such scenarios:

Recall represents the ability to identify actual fraud cases, minimizing false negatives, high recall means fewer fraudulent transactions go undetected:

- Decision Tree excels with a recall of 0.7795, indicating its ability to identify most fraudulent cases, followed behind is Random Forest with 0.6988 point score.
- Logistic Regression lags significantly (0.1916), suggesting it misses many fraudulent transactions.

Precision measures the accuracy of positive predictions made by the model:

- Random Forest again leads with 0.972, ensuring most of its fraud predictions are accurate.

- Logistic Regression performs well (0.8253) but isn't as reliable as Random Forest.
- Decision Tree has lower precision (0.72632), potentially raising more false alarms.

AUC (Area Under the ROC Curve) quantifies the area under the receiver operating characteristic curve. The ROC curve is a plot of the true positive rate against the false positive rate at various thresholds.

- Random Forest maintains its lead with 0.978, indicating excellent class separation.
- Logistic Regression and Decision Tree are less effective (0.8631 and 0.88917, respectively).

	Logistic Regression	**Random Forest**	**Decision Tree**
Recall Score	0.7557	0.7678	0.80986
Precision	0.0259	0.8912	0.61275
F1-Score	0.0501	0.8249	0.69466
Accuracy Score	0.8094	0.9987	0.99727
AUC Score	0.8749	0.9901	0.89995
Running Time(mins)	0.92	22.05	1.15

Table 22. Evaluation metrics of 3 models after applying SMOTE

Applying SMOTE, we have some key changes and considerations:

1. With recall improvements:
- Significant recall gains across all models, indicating SMOTE's effectiveness in addressing class imbalance and improving detection of fraudulent cases.
- Recall score goes up in parallel with the Precision score goes down as a consequence, so we consider it as a trade-off between two metrics.
- Logistic Regression experienced the most dramatic improvement (from 0.1916 to 0.7557 in recall score).
2. Logistic Regression's precision decreased to 0.0259, suggesting overfitting or generation of excessively noisy synthetic data by SMOTE that’s why its F1-score dropped significantly.

3. Minor increases in running time due to SMOTE's computational overhead, most notably for Random Forest.
4. Random Forest remains the strongest overall performer, with balanced recall, high precision, excellent AUC, and higher running time. Decision Tree appears to perform well across recall , if we consider to minimize the number of false negatives (missed fraud cases) it will be the best solution, but it fails to make good predictions on non-fraud transactions that might lead to inconveniencing legitimate customers. Logistic Regression appears to perform well across recall, but gives a bad precision and accuracy so it's out of scope for our project. → Random Forest is the most optimal model.

G. Conclusion

1. Key findings and contributions

- **Model Development:**
  - The primary focus revolved around constructing robust predictive models capable of discerning fraudulent transactions from legitimate ones.
  - Through meticulous data preprocessing, including handling missing values, scaling features, and exploring the dataset's characteristics, we established a solid foundation for model development.
- **Feature Importance:**
  - The project delved into the identification of crucial features contributing significantly to the prediction of fraudulent activities.
  - This insight provides a deeper understanding of the variables pivotal in fraud detection, aiding in the refinement and enhancement of future models.
- **Validation Techniques:**
  - Rigorous validation techniques were employed to ensure the model's reliability and generalizability.
  - Cross-validation and performance metrics assessments were instrumental in validating the model's efficacy across various scenarios.
- **Contribution to Security Measures:**
  - The project's findings contribute substantively to enhancing transactional security measures, providing financial institutions and stakeholders with effective tools to combat and minimize the impact of fraudulent activities, fostering a more secure digital payment ecosystem.

2. Limitations and Challenges

- **Imbalanced Dataset:** The primary challenge encountered was dealing with the highly imbalanced nature of the dataset. Strategies implemented to address this, such as resampling techniques, might lead to potential biases or overfitting, affecting the model's generalization.
- **Privacy and Feature Engineering:** The dataset's anonymized nature, while essential for privacy preservation, presented challenges in comprehensively understanding and engineering features, potentially impacting the model's predictive capabilities.
- **Model Interpretability:** While achieving high accuracy in detecting fraud, some models lacked interpretability, hindering a clear understanding of the decision-making process behind identifying fraudulent transactions.
- **Overcoming Challenges:**
    - To address the imbalanced dataset issue, a balanced approach deploying ensemble methods like SMOTE (Synthetic Minority Over-sampling Technique) was employed.
    - Careful attention was paid to hyperparameter tuning and model selection to mitigate overfitting and biases.
    - Feature engineering techniques were utilized cautiously, considering privacy constraints, and efforts were made to strike a balance between model complexity and interpretability.

3. Conclusion & Future Directions

In conclusion, the project embarked upon the formidable task of addressing credit card fraud detection, an ever-evolving challenge in the landscape of financial security. The culmination of extensive data analysis, model development, and algorithmic exploration has contributed significantly to understanding and combating fraudulent activities in credit card transactions.

Moving forward, continuous research and development in machine learning techniques tailored for imbalanced datasets will be imperative. Exploring advanced algorithms, neural network architectures, and deep learning methodologies could potentially unlock new avenues for more accurate and interpretable fraud detection systems.

Moreover, collaborating with financial institutions to integrate these models into real-time transaction monitoring systems will be a crucial step towards proactive fraud prevention. Ongoing efforts in this domain will ensure a safer and more secure environment for financial transactions, safeguarding the interests of both financial institutions and consumers.

H. References

- Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16, 321-357.
- IBM Cloud Education. (n.d.). Decision Trees. IBM.
- Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5-32. Read PaperGoogle Developers. (n.d.). Receiver Operating Characteristic (ROC) and Area Under the Curve (AUC). Google Machine Learning Crash Course. Read More
- Powers, D. M. W. (2011). Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation. Journal of Machine Learning Technologies, 2(1), 37-63. Read Paper.