CREDIT CARD FRAUD DETECTION USING MACHINE LEARNING

A Capstone Phase-II project report submitted

In partial fulfilment of requirement for the award of degree

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

By

D. Shantan Srivatsa (2203A51171)

N. Nagaraju (2203A51183)

K. Sree Chandana (2203A51299)

S. Akhila (2203A51619)

Under the guidance of

Dr.E.L.N.KiranKumar

Associate Professor, CS&AI.

SR UNIVERSITY

Ananthasagar, Warangal

CERTIFICATE

This is to certify that this project entitled **“CREDIT CARD FRAUD DETECTION** " is the bonafied work carried out by **D. Shantan Srivatsa ,N. Nagaraju , K. Sree Chandana , S. Akhila** as a Capstone Phase II project for the partial fulfilment toaward the degree **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE & ENGINEERING** during the academic year 2022-2026 under our guidance and Supervision.

Dr.E.L.N.KiranKumar	Dr.M.Sheshikala
Assoc.Prof.CS&AI	Assoc.Prof.&HOD(CSE),
SRUniversity	SRUniversity,
Ananthasagar,Warangal.	Ananthasagar,Warangal.

ACKNOWLEDGEMENT

We owe an enormous debt of gratitude to our project guide **Dr.E.L.N.KiranKumar, Assoc.Prof.CS and AI** as well as Head of the CSE Department **Dr.M.Sheshikala, Associate Professor** for guiding us from the beginning through the end of the Capstone Phase–II project with the irintellectual advices and insightful suggestions. We truly value their consistent feedback on our progress ,which was always constructive and encouraging and ultimately drove us to the right direction.

Finally, we express our thanks to all the teaching and non-teaching staff of the department for their suggestions and timely support.

CONTENTS

ABSTRACT

Credit card fraud detection is a critical task in the financial industry to safeguard customers from unauthorized transactions. This paper addresses the main challenges involved in this domain, including the processing of enormous data volumes, imbalanced data distributions where fraudulent transactions are rare, data privacy concerns, potential misclassifications, and adaptive techniques employed by fraudsters. To tackle these challenges, the proposed approach emphasizes the development of simple yet fast models capable of quickly identifying anomalies. Techniques for handling imbalanced data are discussed, along with methods for preserving user privacy through dimensionality reduction. Furthermore, the importance of utilizing trustworthy data sources for model training is highlighted. The paper also advocates for model simplicity and interpretability to facilitate rapid adaptation to evolving fraud tactics.

KEYWORDS

numpy, pandas, matplotlib, plt, seaborn, gridspec, sklearn, Random Forest Classifier

INTRODUCTION

Credit card fraud detection is crucial for protecting customers from unauthorized transactions in the financial industry. With the rise of digital transactions, the challenge lies in swiftly identifying fraudulent activities to prevent financial losses. However, this task is complicated by enormous data volumes processed daily, imbalanced datasets where fraudulent transactions are rare, and the sensitive nature of private transaction data. Moreover, misclassifications and adaptive fraud tactics further exacerbate the challenge. Effective fraud detection requires innovative machine learning approaches that are fast, accurate, and capable of handling data privacy concerns while remaining adaptable to evolving fraud tactics

In credit card fraud detection, machine learning models play a crucial role in identifying fraudulent transactions amidst vast datasets. Techniques like Random Forest Classifier and ensemble methods are employed to enhance accuracy and speed in classifying transactions. Keywords such as classification_report, accuracy_score, precision_score, recall_score, f1_score, and confusion_matrix are utilized to evaluate model performance. These models must contend with challenges like imbalanced data distributions and adaptive fraud tactics, necessitating innovative approaches such as oversampling techniques and real-time model adaptation. Ultimately, the development of robust and interpretable models is essential to effectively mitigate credit card fraud while ensuring customer trust and privacy.

Credit card fraud can be influenced by a variety of factors, including as shoen in fig(1):

Stolen Card Information: Fraudsters can obtain credit card information through data breaches, skimming devices, phishing scams, or hacking into databases.

Weak Security Measures: Inadequate security measures by merchants, financial institutions, or cardholders can make it easier for fraudsters to exploit vulnerabilities.

Online Transactions: The rise of e-commerce has increased the risk of credit card fraud as transactions occur remotely, making it easier for fraudsters to conceal their identity.

Card-Not-Present Transactions: Transactions where the physical card is not required (e.g., online purchases) are more susceptible to fraud since the card's presence cannot be verified.

Identity Theft: Fraudsters may use stolen personal information to open new credit card accounts or take over existing accounts, leading to fraudulent activity.

Counterfeit Cards: Criminals may create counterfeit credit cards using stolen card information or by physically altering legitimate cards.

Lack of Awareness: Consumers and businesses may not be fully aware of the latest fraud tactics and prevention measures, making them more vulnerable to scams.

Weak Authentication Processes: Inadequate verification methods, such as simple passwords or lack of multi-factor authentication, make it easier for fraudsters to gain unauthorized access to accounts.

Fig(1)

PROBLEM STATEMENT

The challenge is to develop an efficient and accurate credit card fraud detection system that can reliably distinguish between legitimate and fraudulent transactions in real-time. This system must contend with several obstacles, including the processing of vast amounts of transaction data, the highly imbalanced nature of fraudulent transactions, the sensitivity of private customer information, potential misclassifications, and the adaptive tactics employed by fraudsters to evade detection. The goal is to devise machine learning algorithms and techniques that can effectively address these challenges, ensuring timely and precise identification of fraudulent activities while minimizing false positives and preserving user privacy.

OBJECTIVE

The objective of a credit card fraud detection system is multifaceted, aiming to safeguard financial transactions by swiftly identifying and thwarting unauthorized activities. It is designed to operate with precision, ensuring that genuine transactions proceed smoothly while effectively flagging potentially fraudulent ones. Through real-time monitoring capabilities, the system continually assesses transactions, promptly detecting any irregularities or suspicious patterns. Its scalability enables it to handle high transaction volumes efficiently, adapting to evolving fraud tactics through advanced algorithms and machine learning. By assessing risk factors such as transaction details and cardholder behavior, the system provides a comprehensive defense against fraud, employing multiple layers of security measures. Ultimately, its goal is to minimize financial losses, maintain regulatory compliance, and enhance the overall customer experience by balancing effective fraud prevention with seamless transaction processing.

APPROACH

The approach of a credit card fraud detection system is multifaceted, involving the collection of transactional data, rigorous analysis through feature engineering, and the application of advanced algorithms and machine learning models. Real-time monitoring plays a pivotal role, enabling the system to promptly identify anomalies and flag suspicious transactions for investigation. Additionally, rule-based engines enforce predefined fraud detection rules, providing an added layer of security. Collaboration and information sharing among financial institutions enhance the system's effectiveness by leveraging collective insights and intelligence to combat evolving fraud tactics. Overall, the system's approach combines data-driven analysis, advanced technology, and collaborative efforts to proactively detect and prevent fraudulent activities in real-time, safeguarding financial transactions and minimizing potential losses.

MOTIVATION AND SCOPE OF WORK

MOTIVATION:

The motivation behind this project stems from the critical need to protect consumers and financial institutions from the escalating threat of credit card fraud. With the proliferation of digital transactions, fraudsters exploit vulnerabilities in existing systems, causing substantial financial losses and undermining trust in the banking industry. By developing an advanced fraud detection system, we aim to mitigate these risks, safeguarding both customers and businesses from fraudulent activities. Moreover, enhancing fraud detection capabilities not only strengthens security measures but also bolsters confidence in electronic payment systems, fostering a more secure and resilient financial ecosystem.

SCOPE OF WORK:

The scope of this project encompasses the design, development, and implementation of a comprehensive credit card fraud detection system. This involves:

1. **Data Collection and Preprocessing:** Gathering transactional data from various sources and cleaning, preprocessing, and transforming it into a suitable format for analysis.
2. **Model Development:** Exploring and implementing state-of-the-art machine learning algorithms, such as Random Forest, Support Vector Machines, and Neural Networks, to build robust fraud detection models.
3. **Evaluation and Validation**: Assessing the performance of the developed models using appropriate evaluation metrics and validation techniques to ensure accuracy and reliability.
4. **Deployment and Integration:** Integrating the trained models into existing banking systems or developing standalone applications for real-time fraud detection.

5. **Continuous Improvement:** Implementing mechanisms for monitoring model performance, collecting feedback, and iteratively improving the system to adapt to emerging fraud patterns and evolving threats.
6. **Documentation and Reporting:** Documenting the entire development process, including methodologies, algorithms used, and results obtained, to facilitate knowledge sharing and future enhancements.

LITERATURE REVIEW

The literature on credit card fraud detection underscores the critical role of machine learning techniques in mitigating financial losses and preserving consumer trust. Studies have extensively explored the application of algorithms like logistic regression, decision trees, random forests, and neural networks to effectively distinguish between legitimate and fraudulent transactions.

Addressing the challenge of imbalanced data distribution, researchers have devised various approaches such as oversampling, undersampling, and ensemble methods to enhance model performance. Additionally, feature engineering techniques focusing on transaction characteristics like amount, frequency, and time have been instrumental in improving the discriminatory power of fraud detection models.

Moreover, the emergence of real-time detection systems leveraging stream processing frameworks has facilitated prompt identification of suspicious activities, while ensuring user privacy through techniques like differential privacy and data anonymization. Continued research efforts in these areas are imperative to stay ahead of evolving fraud tactics and bolster the security of electronic payment systems.

DATASET

Dataset Description for Credit Card Fraud Detection:

1. Attributes:

The dataset contains attributes related to credit card transactions, including features such as transaction amount, time, merchant information, location, type of transaction (online or in-person), and potentially additional factors like cardholder demographics or previous transaction history.

These attributes serve as input features for machine learning models to learn patterns and relationships indicative of fraudulent activity.

2. Fraud Label:

The primary target variable in the dataset is a binary label indicating whether a transaction is fraudulent or genuine.

This label is crucial for training machine learning models to distinguish between legitimate and fraudulent transactions.

3. Data Preprocessing:

Before model training, preprocessing steps are applied to ensure data quality and suitability for analysis.

Preprocessing tasks may include handling missing values, removing outliers, scaling or normalizing features, and encoding categorical variables (e.g., merchant category codes or transaction types).

4. Exploratory Data Analysis (EDA):

Exploratory data analysis techniques are utilized to understand the dataset's characteristics and identify patterns indicative of fraud.

EDA may involve visualizing distributions of transaction attributes, examining temporal patterns, exploring correlations, and detecting anomalies.

5. Dataset Size and Composition:

The dataset may vary in size, with a larger number of transactions facilitating more robust model training and evaluation.

It may include a mix of fraudulent and genuine transactions, with the class distribution typically highly imbalanced, reflecting the rarity of fraudulent activity.

6. Data Collection:

The dataset may have been collected from various sources, such as financial institutions' transaction records, industry databases, or publicly available datasets.

Data collection methods aim to capture a representative sample of credit card transactions while ensuring data accuracy and privacy compliance.

Creditcard.csv:

A	B	C	D	E	F	G	H	I	J
Time	V1	V2	V3	V4	V5	V6	V7	V8	V9
0	-1.35981	-0.07278	2.536347	1.378155	-0.33832	0.462388	0.239599	0.098698	0.363787
0	1.191857	0.266151	0.16648	0.448154	0.060018	-0.08236	-0.0788	0.085102	-0.25543
1	-1.35835	-1.34016	1.773209	0.37978	-0.5032	1.800499	0.791461	0.247676	-1.51465
1	-0.96627	-0.18523	1.792993	-0.86329	-0.01031	1.247203	0.237609	0.377436	-1.38702
2	-1.15823	0.877737	1.548718	0.403034	-0.40719	0.095921	0.592941	-0.27053	0.817739
2	-0.42597	0.960523	1.141109	-0.16825	0.420987	-0.02973	0.476201	0.260314	-0.56867
4	1.229658	0.141004	0.045371	1.202613	0.191881	0.272708	-0.00516	0.081213	0.46496
7	-0.64427	1.417964	1.07438	-0.4922	0.948934	0.428118	1.120631	-3.80786	0.615375
7	-0.89429	0.286157	-0.11319	-0.27153	2.669599	3.721818	0.370145	0.851084	-0.39205
9	-0.33826	1.119593	1.044367	-0.22219	0.499361	-0.24676	0.651583	0.069539	-0.73673
10	1.449044	-1.17634	0.91386	-1.37567	-1.97138	-0.62915	-1.42324	0.048456	-1.72041
10	0.384978	0.616109	-0.8743	-0.09402	2.924584	3.317027	0.470455	0.538247	-0.55889
10	1.249999	-1.22164	0.38393	-1.2349	-1.48542	-0.75323	-0.6894	-0.22749	-2.09401
11	1.069374	0.287722	0.828613	2.71252	-0.1784	0.337544	-0.09672	0.115982	-0.22108
12	-2.79185	-0.32777	1.64175	1.767473	-0.13659	0.807596	-0.42291	-1.90711	0.755713
12	-0.75242	0.345485	2.057323	-1.46864	-1.15839	-0.07785	-0.60858	0.003603	-0.43617
12	1.103215	-0.0403	1.267332	1.289091	-0.736	0.288069	-0.58606	0.18938	0.782333
13	-0.43691	0.918966	0.924591	-0.72722	0.915679	-0.12787	0.707642	0.087962	-0.66527
14	-5.40126	-5.45015	1.186305	1.736239	3.049106	-1.76341	-1.55974	0.160842	1.23309
36	-1.16942	1.158314	1.4068	0.860189	-0.10381	0.122035	0.264451	-0.10877	-0.18198	0.659593
36	1.095525	-0.11609	1.397912	1.497547	-1.04912	0.072839	-0.7238	0.287532	0.996327	-0.14914
37	1.295668	0.341483	0.081505	0.566746	-0.11046	-0.76632	0.073155	-0.1683	0.071837	-0.28104
38	0.158332	0.872687	0.965525	1.802127	0.139022	0.110863	0.559079	-0.02907	-0.39491	0.032674
39	-0.55406	0.215728	0.844265	-0.75307	0.034848	-0.5586	1.090401	-0.20387	0.262394	-1.35471
39	-1.33088	0.26754	-0.16847	-0.70123	3.281972	3.21639	-0.05939	0.89543	-0.01778	-0.16575
40	1.110692	0.081942	0.406063	1.29074	-0.16981	0.120955	-0.08085	0.174373	0.152071	0.065267
41	1.154312	0.265462	0.384871	0.575007	-0.21748	-0.39152	-0.08149	0.062789	-0.26058	-0.16168
41	0.986063	-0.20296	-0.49277	0.407691	0.30566	-0.23053	0.585028	-0.20822	-0.2475	-0.19225
41	1.138759	-1.19295	1.407131	-0.33007	-2.0695	-0.24218	-1.30664	0.10451	0.134628	0.493931
41	1.145524	0.575068	0.194008	2.598192	-0.09221	-1.04443	0.531588	-0.24189	-0.89629	0.757952
42	-0.24936	0.399227	0.068009	-1.06062	2.410399	3.736574	0.316446	0.672296	0.01514	-0.2695
42	-0.52267	1.009923	0.27647	1.475289	-0.70701	0.355243	1.559849	-0.39958	-0.47981	0.516352
44	-0.89999	0.136255	1.883665	-0.209	1.051441	1.905241	0.241423	0.647631	-0.05347	-0.50417
44	-0.71476	0.514969	1.821676	0.616434	0.848776	-0.11194	1.505617	-0.79817	0.244757	0.264554
44	-0.9489	0.248414	2.956914	2.81375	0.145539	-0.02735	0.133702	-0.30753	-0.12524	1.03494
44	0.92706	-0.32368	0.387585	0.544474	0.246787	1.650358	-0.42758	0.615371	0.226278	-0.2255
46	-1.92321	-0.87048	2.32017	1.988776	0.417091	-0.38001	0.472139	-0.55733	-0.64908	1.410889
46	1.006589	-0.07111	0.347614	1.329684	-0.19324	0.155418	0.008574	0.146538	0.103844	0.03183
46	-0.37824	0.732925	-0.12015	0.185755	2.594269	3.797183	0.059088	0.976768	-0.41266	0.006754

PROPOSED METHODOLOGIES:

The proposed methodology for a credit card fraud detection system involves a systematic approach that integrates various techniques and processes to effectively identify and prevent fraudulent transactions. Here's an outline of the methodology:

1.Data Collection and Preprocessing:

- Gather transactional data from multiple sources, including financial institutions' records, industry databases, or publicly available datasets.
- Preprocess the data to ensure quality and suitability for analysis, which may involve handling missing values, removing outliers, and encoding categorical variables.

2. Exploratory Data Analysis (EDA):

- Conduct exploratory data analysis to understand the characteristics of the dataset and identify patterns indicative of fraudulent activity.
- Visualize distributions of transaction attributes, examine temporal patterns, explore correlations, and detect anomalies.

3. Feature Engineering:

- Engineer relevant features from the transactional data that can enhance the detection of fraudulent activity.
- Create features that capture transaction frequency, amount, location, time of day, and other relevant factors.

4. Model Selection and Training:

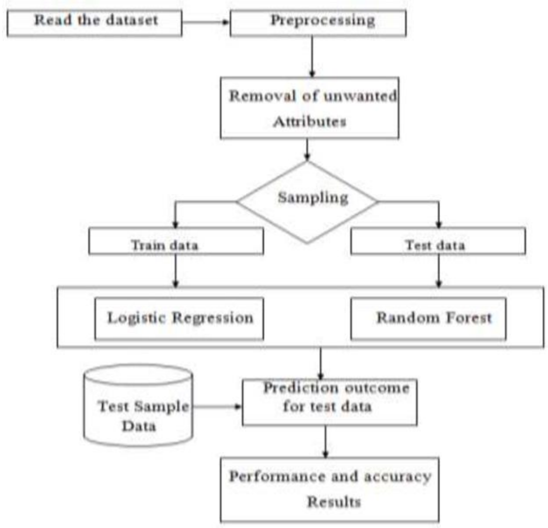
- Select appropriate machine learning algorithms for fraud detection, considering factors such as interpretability, scalability, and performance.
- Train multiple models, including supervised learning algorithms like logistic regression, decision trees, random forests, gradient boosting, and anomaly detection algorithms like Isolation Forest or One-Class SVM.
- Tune hyperparameters and evaluate model performance using suitable metrics such as accuracy, precision, recall, F1-score, and ROC-AUC.

5. Ensemble Methods:

- Explore ensemble methods such as bagging, boosting, or stacking to combine the predictions of multiple models and improve overall performance.

The workflow of the project looks like as shown in fig[2]-

1. **Data Collection:** Gather transactional data from various sources, including financial institutions' records, industry databases, or publicly available datasets.
2. **Data Preprocessing:** Handle missing values, outliers, and encode categorical variables to ensure data quality and suitability for analysis.
3. **Exploratory Data Analysis (EDA):** Conduct exploratory data analysis to understand dataset characteristics, visualize attribute distributions, examine temporal patterns, and detect anomalies.
4. **Feature Engineering:** Engineer relevant features from transactional data, such as transaction frequency, amount, location, time of day, and other relevant factors.
5. **Model Selection and Training:** Choose appropriate machine learning algorithms and train multiple models, including supervised learning algorithms and anomaly detection algorithms.
6. **Real-time Monitoring and Alert Generation:** Implement models into a real-time monitoring system to analyze transactions as they occur and develop alerts to flag suspicious transactions for further investigation.



fig(2)

RESULTS &DISCUSSION :

Importing libraries:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
%matplotlib inline

import warnings
warnings.filterwarnings("ignore")
```

Background

The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

```
df = pd.read_csv('/content/creditcard.csv')
```

```
df.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21
0	0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307
1	0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775
2	1	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998
3	1	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300
4	2	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431

5 rows × 31 columns

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11959 entries, 0 to 11958
Data columns (total 31 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Time    11959 non-null  int64  
 1   V1      11959 non-null  float64
 2   V2      11959 non-null  float64
 3   V3      11959 non-null  float64
 4   V4      11959 non-null  float64
 5   V5      11959 non-null  float64
 6   V6      11959 non-null  float64
 7   V7      11959 non-null  float64
 8   V8      11959 non-null  float64
 9   V9      11959 non-null  float64
 10  V10     11959 non-null  float64
 11  V11     11959 non-null  float64
 12  V12     11959 non-null  float64
 13  V13     11959 non-null  float64
 14  V14     11959 non-null  float64
 15  V15     11959 non-null  float64
 16  V16     11959 non-null  float64
 17  V17     11959 non-null  float64
 18  V18     11959 non-null  float64
 19  V19     11959 non-null  float64
 20  V20     11958 non-null  float64
 21  V21     11958 non-null  float64
 22  V22     11958 non-null  float64
 23  V23     11958 non-null  float64
 24  V24     11958 non-null  float64
 25  V25     11958 non-null  float64
 26  V26     11958 non-null  float64
 27  V27     11958 non-null  float64
 28  V28     11958 non-null  float64
 29  Amount  11958 non-null  float64
 30  Class   11958 non-null  float64
dtypes: float64(30), int64(1)
```

```
df.describe()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8
count	11959.000000	11959.000000	11959.000000	11959.000000	11959.000000	11959.000000	11959.000000	11959.000000	11959.00000
mean	8009.996822	-0.216230	0.277097	0.889505	0.282606	-0.086585	0.139986	-0.121943	-0.04872
std	6204.332248	1.583914	1.308884	1.331824	1.478162	1.191776	1.306285	1.153899	1.24682
min	0.000000	-27.670569	-34.607649	-22.804686	-4.657545	-32.092129	-23.496714	-26.548144	-23.63250
25%	2542.000000	-0.978944	-0.261503	0.417186	-0.622456	-0.688114	-0.622521	-0.591335	-0.18524
50%	6662.000000	-0.340742	0.256346	0.951223	0.213029	-0.183847	-0.146903	-0.094876	0.01361
75%	12382.000000	1.161273	0.883626	1.613678	1.159141	0.346298	0.508432	0.431657	0.26756
max	20642.000000	1.960497	9.092123	4.101716	11.927512	34.099309	21.393069	34.303177	5.49996

8 rows × 31 columns

UNIVARIETE ANALYSIS

Univariate analysis generally refers to the data analysis where there is only one dependent variable. The main goal of the univariate analysis is to summarize the data. We can easily identify measures of central tendency like mean, median, mode, the quartiles, and the standard deviation.

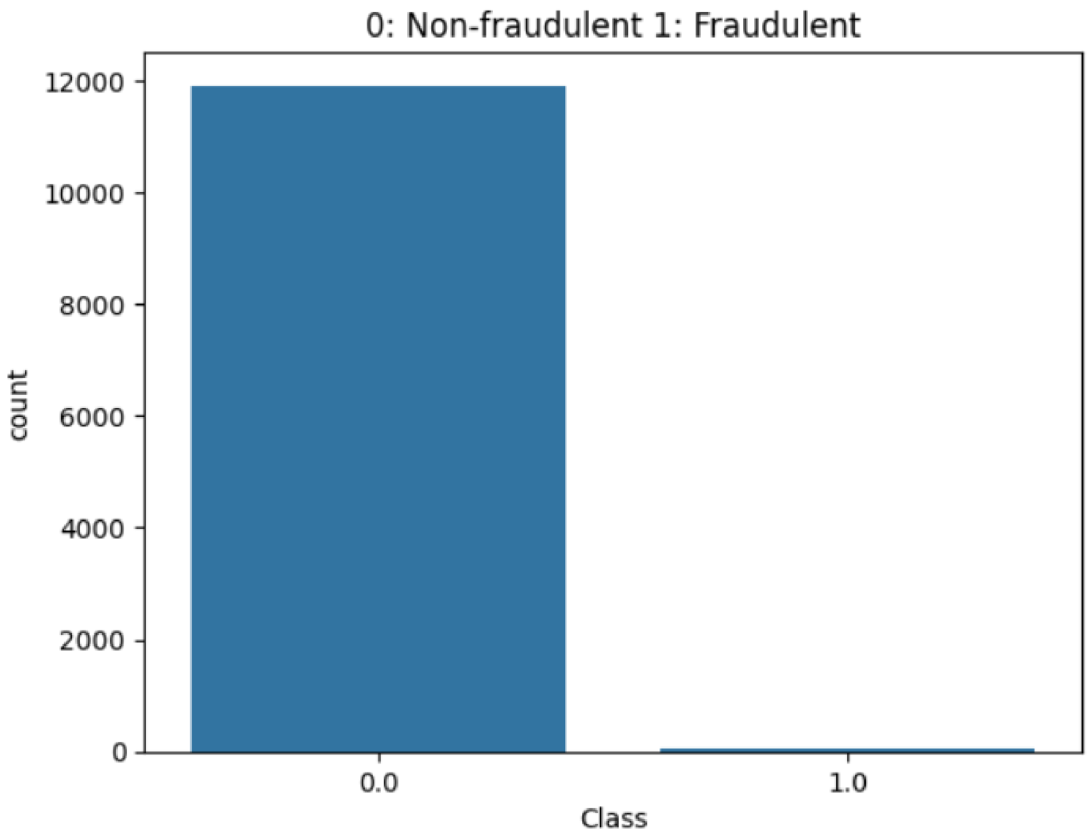
BIVARIETE ANALYSIS

Bivariate analysis happens between 2 variables to identify the relationship between them.

```
fraud = round(len(df[df['Class']==1])/len(df)*100,2)
nofraud = round(len(df[df['Class']==0])/len(df)*100,2)

print("No fraud transactions are:",str(nofraud)+'%',"of the dataset")
print("Fraud transactions are:",str(fraud)+'%',"of the dataset")
```

```
No fraud transactions are: 99.56% of the dataset
Fraud transactions are: 0.43% of the dataset
```

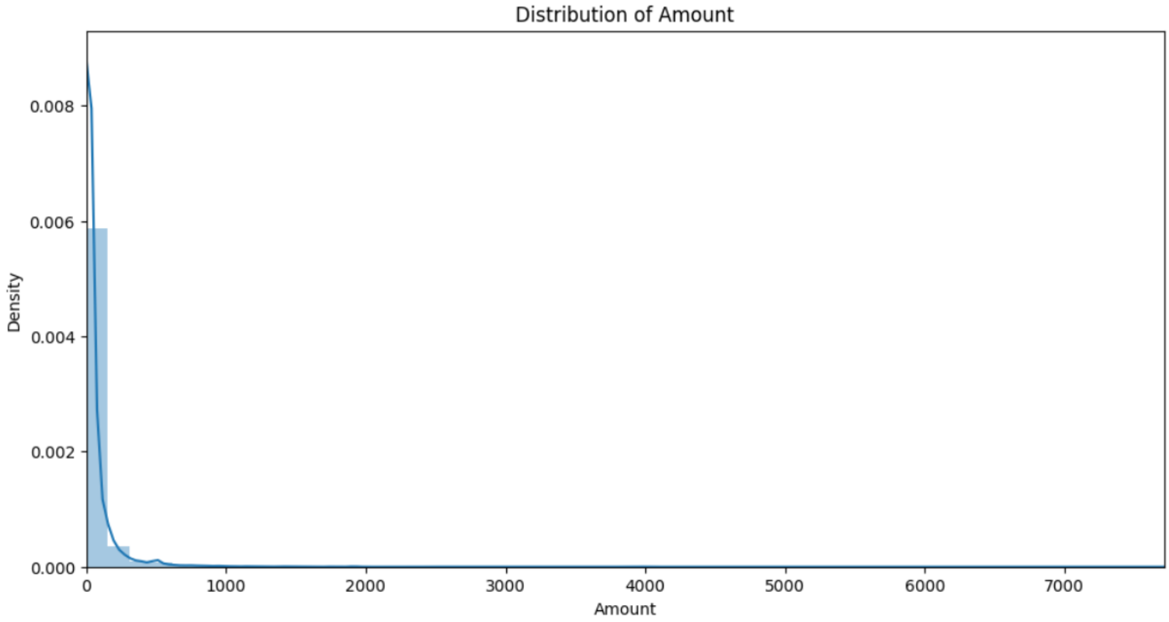
```
sns.countplot(x='Class',data=df)
plt.title("0: Non-fraudulent 1: Fraudulent")
```



- The amount variable is mostl dense around the samllar amount regions.

```
amount_val = df['Amount'].values
plt.figure(figsize = (12,6))
sns.distplot(df['Amount'])
plt.xlim(min(amount_val),max(amount_val))
plt.title("Distribution of Amount")
```

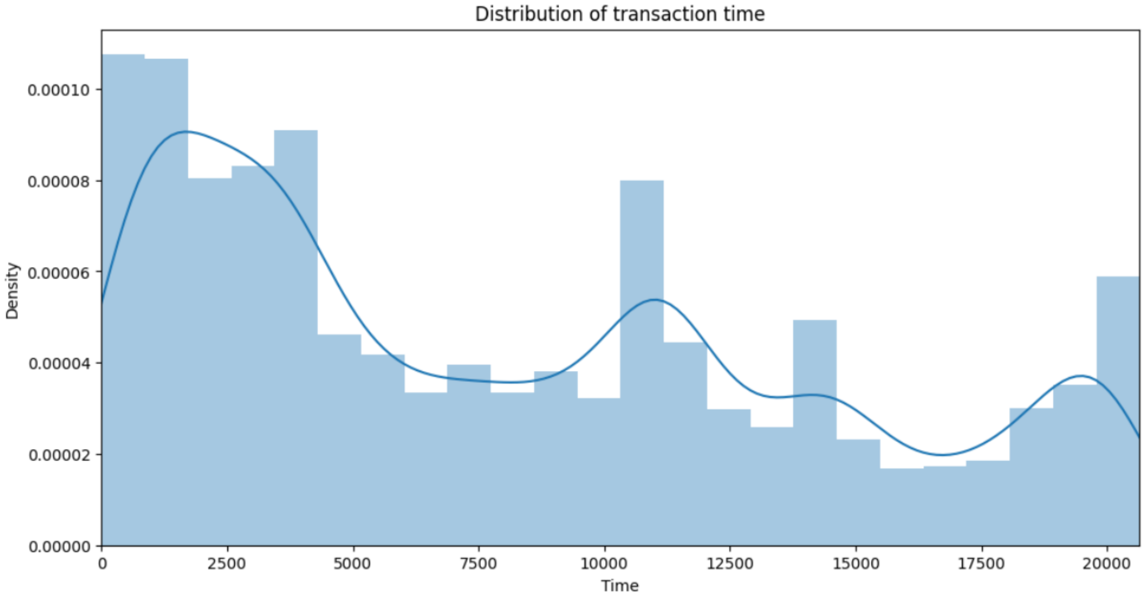
```
Text(0.5, 1.0, 'Distribution of Amount')
```



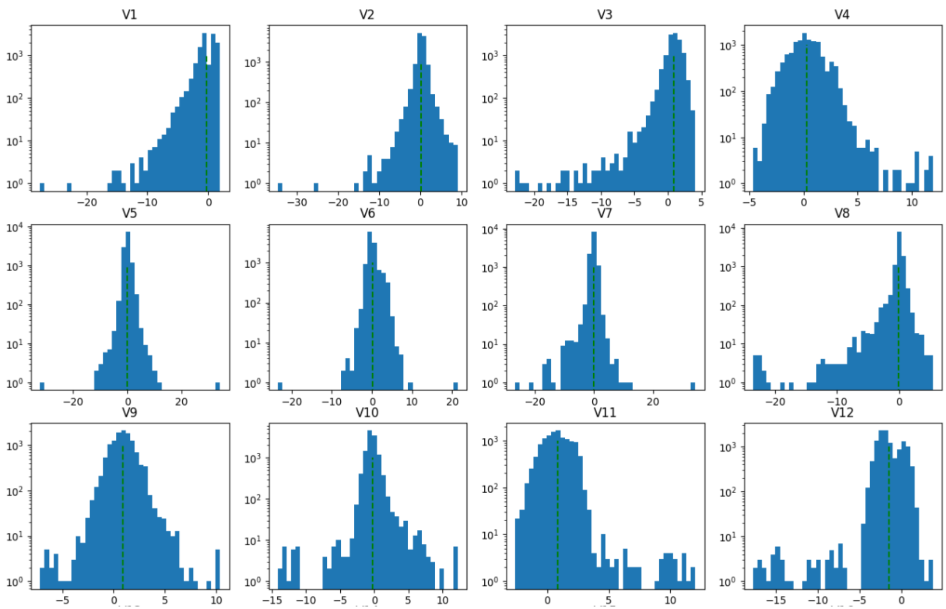
Time has **bimodal distribution** i.e. the peak rises and falls down and rises again. The fall might happen due to night time.

```python
time_val = df['Time'].values
plt.figure(figsize=(12,6))
sns.distplot(df['Time'])
plt.xlim(min(time_val),max(time_val))
plt.title("Distribution of transaction time")
```

```
Text(0.5, 1.0, 'Distribution of transaction time')
```

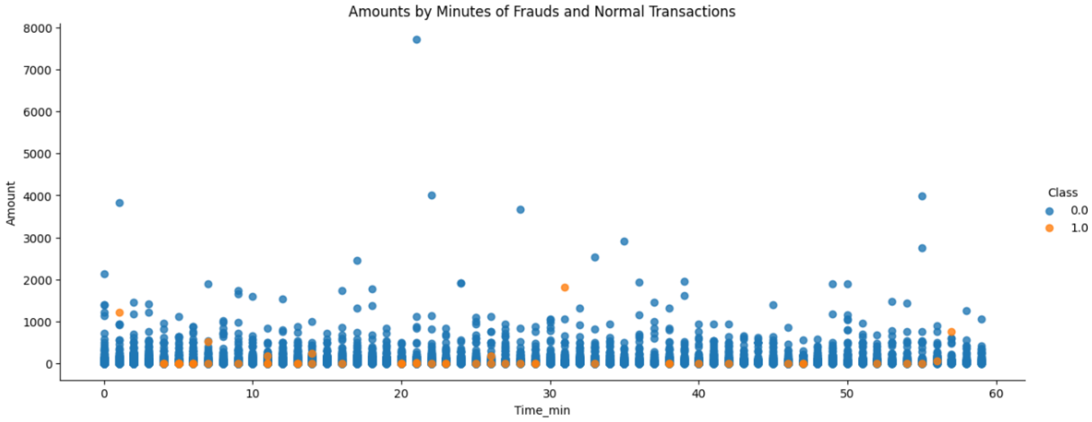



```
#distribution of the different features

fig, ax = plt.subplots(nrows=7,ncols=4,figsize=(16,24))

for i in range(1,29):
  m = (i-1)//4
  n = (i-1)%4
  col = 'V' + str(i)
  ax[m,n].hist(df[col],bins=40)
  ax[m,n].set_title(col)
  ax[m,n].vlines(x=df[col].mean(),ymin=0,ymax=10**3,linestyle='dashed',colors='g')
  ax[m,n].set_yscale('log')
```



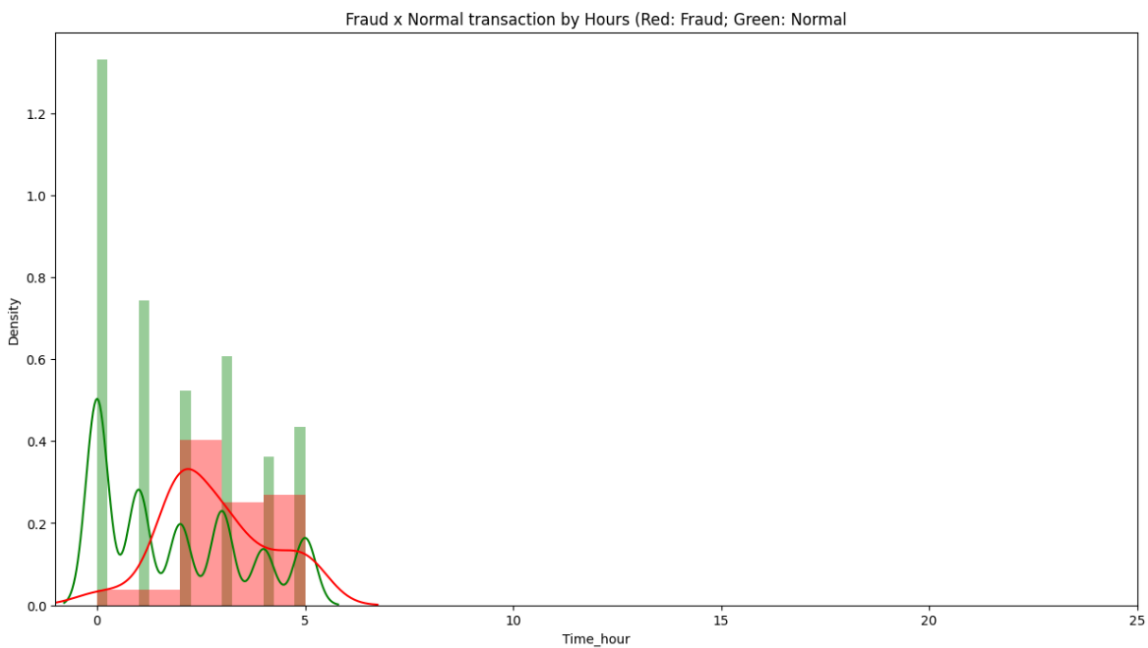
```python
ax = sns.lmplot(y='Amount',x='Time_min',fit_reg=False,aspect=2.5,data=data_new,hue='Class')
plt.title("Amounts by Minutes of Frauds and Normal Transactions",fontsize=12)
```

```
Text(0.5, 1.0, 'Amounts by Minutes of Frauds and Normal Transactions')
```



Now transactions show a more non uniform disribution accross Time_hour which makes sense but still we see no particular pattern to distinguish fraud and non-fraud from this analysis, non-fraud volume is more on active hour compared to lean hours.

```python
plt.figure(figsize=(15,8))
sns.distplot(data_new[data_new['Class']==0]['Time_hour'],color='g')
sns.distplot(data_new[data_new['Class']==1]['Time_hour'],color='r')
plt.title('Fraud x Normal transaction by Hours (Red: Fraud; Green: Normal',fontsize=12)
plt.xlim([-1,25])
```



CONCLUSIONS FROM EDA

1. The data consisted of around 2,85,000 data points, 30 features including time and amount, and the labeled class of whether a transaction is actually fraud or not.
2. There were no null values present in the original dataset but the data was highly skewed with 99.83% of the data points being non-fraudulent transactions.
3. The time feature had a bimodal distribution i.e. peaks falling and rising. I have concluded that the peaks might fall due to lesser transactions during nighttime.
4. Very small proportion of transactions had amounts > 10,000 hence they were eliminated from the dataset.
5. Most of the fraudulent transactions were of small amounts (<1000 units - since we don't know about the units about the currency).
6. The occurrence of fraudulent transactions was independent of the time of the day.

Data Preprocessing

SCALING

Standardization and Robust Scalar

Since fraud transactions which are also low in number have relatively smaller value(amount) so we need to have our data scaled, We are going to use robust scalar to scale our data.

Go through sklaern.preprocessing to know about all different kind of scalars present

```python
from sklearn.preprocessing import RobustScaler

rob_scaler = RobustScaler()

df['scaled_amount'] = rob_scaler.fit_transform(df['Amount'].values.reshape(-1,1))

df.drop(['Time','Amount'],axis=1,inplace=True)
```

```python
df.drop(['Time_min','Time_hour'],axis=1,inplace=True)
```

```python
#inserting these scaled columns at 0,1
scaled_amount = df['scaled_amount']

df.drop(['scaled_amount'],axis=1,inplace=True)
df.insert(0,'scaled_amount',scaled_amount)

df.head()
```

	scaled_amount	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V20	V2
0	2.970444	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	0.251412	-0.0183
1	-0.294667	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.069083	-0.2257
2	8.060222	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.524980	0.2479
3	2.390000	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.208038	-0.1083
4	1.200889	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	0.408542	-0.0094

5 rows × 30 columns

```python
from sklearn.model_selection import StratifiedKFold

X = df.drop('Class',axis=1)
y = df['Class']

sss = StratifiedKFold(n_splits=5,random_state=None,shuffle=False)

for train_index, test_index in sss.split(X,y):
  print("Train:", train_index, "Test:", test_index)
  original_Xtrain, original_Xtest = X.iloc[train_index], X.iloc[test_index]
  original_ytrain, original_ytest = y.iloc[train_index],y.iloc[test_index]

#converting it into an array
original_Xtrain = original_Xtrain.values
original_Xtest = original_Xtest.values
original_ytrain = original_ytrain.values
original_ytest = original_ytest.values

#check if both train and test distributions are similarly distributed
train_unique_label, train_counts_label = np.unique(original_ytrain, return_counts=True)
test_unique_label, test_counts_label = np.unique(original_ytest,return_counts=True)

print("Label dstributions: \n")
print(train_counts_label/len(original_ytrain))
print(test_counts_label/len(original_ytest))
```

```
Train: [ 2384  2385  2386 ... 11955 11956 11957] Test: [    0     1     2 ... 6336 6338 6427]
Train: [    0     1     2 ... 11955 11956 11957] Test: [2384 2385 2386 ... 6774 6820 6870]
Train: [    0     1     2 ... 11955 11956 11957] Test: [4765 4766 4767 ... 8617 8842 8845]
Train: [    0     1     2 ... 11955 11956 11957] Test: [ 7169  7170  7171 ... 10484 10497 10498]
Train: [    0     1     2 ... 10484 10497 10498] Test: [ 9563  9564  9565 ... 11955 11956 11957]
Label dstributions:

[0.99560991 0.00439009]
```

UNDERSAMPLING TO MAKE THE DATASET BALANCED

since our classes are highly skewed, we have to make them equivalent in occurence to have a normal distribution of the classes, shuffle the data before creating the sub-samples.

```
print('Distribution of the Classes in the subsample dataset')
print(new_df['Class'].value_counts()/len(new_df))
```

```
Distribution of the Classes in the subsample dataset
Class
0.0    0.904412
1.0    0.095588
Name: count, dtype: float64
```

CORRELATION HEAT MAP

Correlation is a term used to represent the statistical measure of linear relationship between two variables. It can also be defined as the measure of dependence between two different variables. If there are multiple variables and the goal is to find correlation between all of these variables and store them using appropriate data structure, the matrix data structure is used. Such matrix is called as correlation matrix.

We look at the correlation matrix in original distribution and later used balanced data it shows no much correlation between features or classes for orginal distribution but on balanced set we can visualize correlation more easily, so we find out features highly correlated(positively/negatively) and do outlier detection and removal from them for our data preperation.

Syntax to calculate and see features below or above certain threshold

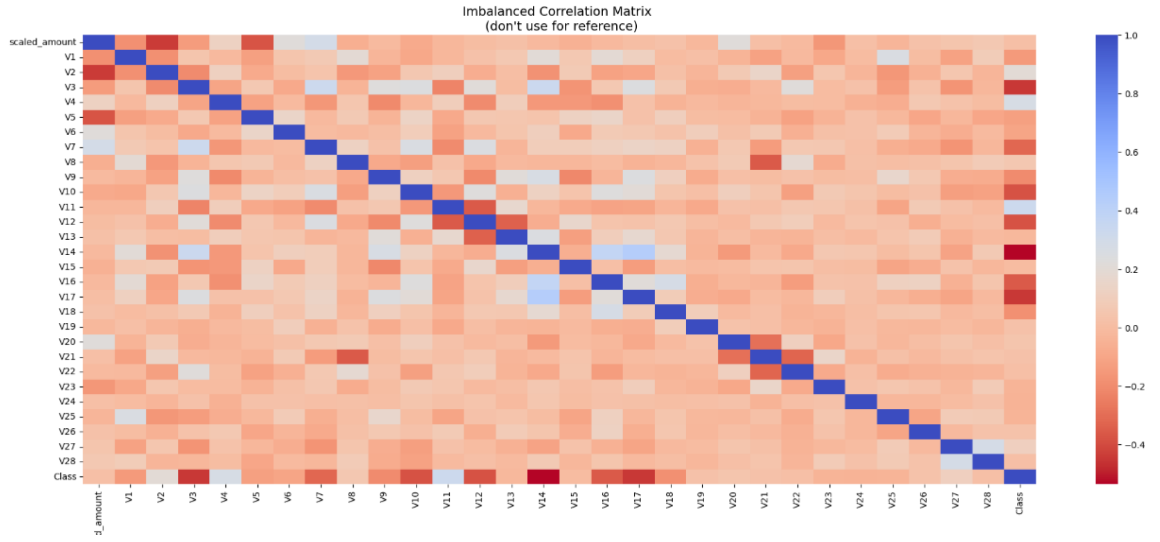
corr = new_df.corr()

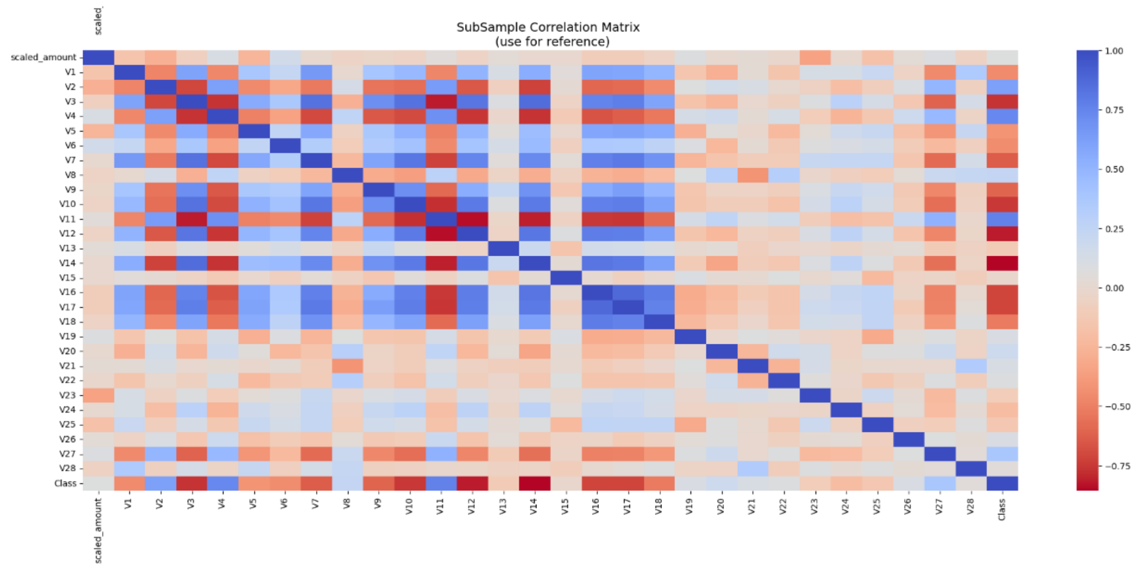
corr[['Class']]

corr[corr.Class<-0.6]['Class']

```
f, (ax1, ax2) = plt.subplots(2, 1, figsize=(24,20))

# comparing correlation between dataset
# Entire DataFrame
corr = df.corr()
sns.heatmap(corr, cmap='coolwarm_r', annot_kws={'size':20}, ax=ax1)
ax1.set_title("Imbalanced Correlation Matrix \n (don't use for reference)", fontsize=14)

# new_df
sub_sample_corr = new_df.corr()
sns.heatmap(sub_sample_corr, cmap='coolwarm_r', annot_kws={'size':20}, ax=ax2)
ax2.set_title('SubSample Correlation Matrix \n (use for reference)', fontsize=14)
```

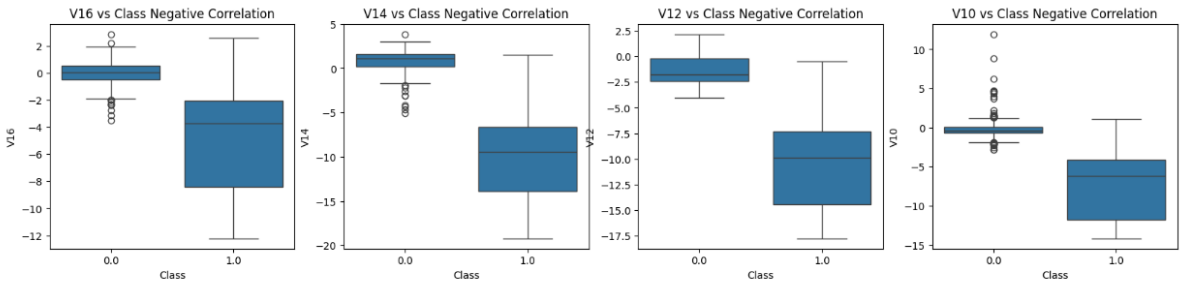
```
Text(0.5, 1.0, 'SubSample Correlation Matrix \n (use for reference)')
```





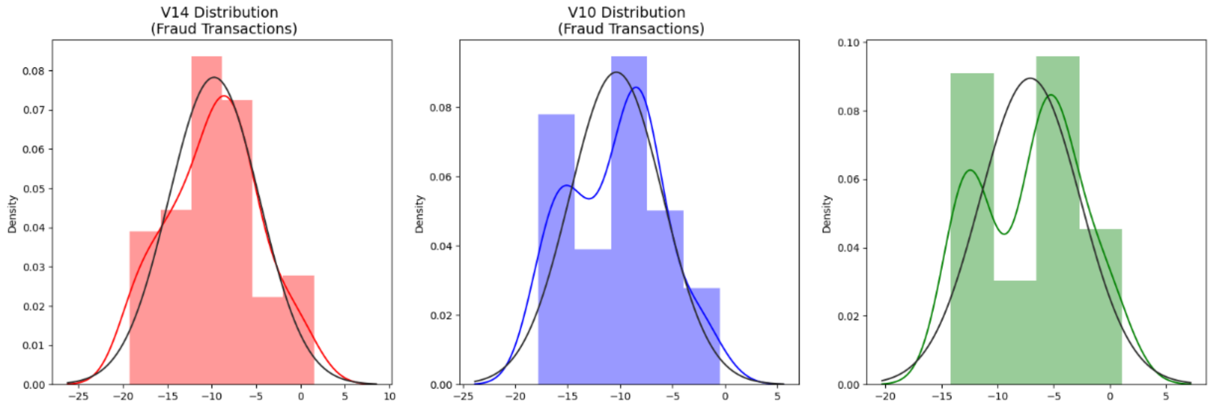
```python
f, axes = plt.subplots(ncols=4,figsize=(20,4))

#Negative correlations with our class (the lower the feature value, higher the chances of it being a fraud transaction)
sns.boxplot(x='Class',y='V16',data=new_df,ax=axes[0])
axes[0].set_title('V16 vs Class Negative Correlation')

sns.boxplot(x='Class',y='V14',data=new_df,ax=axes[1])
axes[1].set_title('V14 vs Class Negative Correlation')

sns.boxplot(x='Class',y='V12',data=new_df,ax=axes[2])
axes[2].set_title('V12 vs Class Negative Correlation')

sns.boxplot(x='Class',y='V10',data=new_df,ax=axes[3])
axes[3].set_title('V10 vs Class Negative Correlation')
```

```
Text(0.5, 1.0, 'V10 vs Class Negative Correlation')
```




```python
from scipy.stats import norm

f, (ax1,ax2,ax3) = plt.subplots(1,3, figsize=(20,6))

v14_fraud_dist = new_df['V14'].loc[new_df['Class']==1].values
sns.distplot(v14_fraud_dist, ax=ax1,fit=norm, color='red')
ax1.set_title('V14 Distribution \n (Fraud Transactions)',fontsize=14)

v12_fraud_dist = new_df['V12'].loc[new_df['Class']==1].values
sns.distplot(v12_fraud_dist, ax=ax2,fit=norm, color='blue')
ax2.set_title('V12 Distribution \n (Fraud Transactions)',fontsize=14)

v10_fraud_dist = new_df['V10'].loc[new_df['Class']==1].values
sns.distplot(v10_fraud_dist, ax=ax3,fit=norm, color='green')
ax2.set_title('V10 Distribution \n (Fraud Transactions)',fontsize=14)
```

```
Text(0.5, 1.0, 'V10 Distribution \n (Fraud Transactions)')
```



REMOVING OUTLIERS

We will use interquatile range to remove outliers from highly correlated features

IQR is used to measure variability by dividing a data set into quartiles. The data is sorted in ascending order and split into 4 equal parts. Q1, Q2, Q3 called first, second and third quartiles are the values which separate the 4 equal parts.

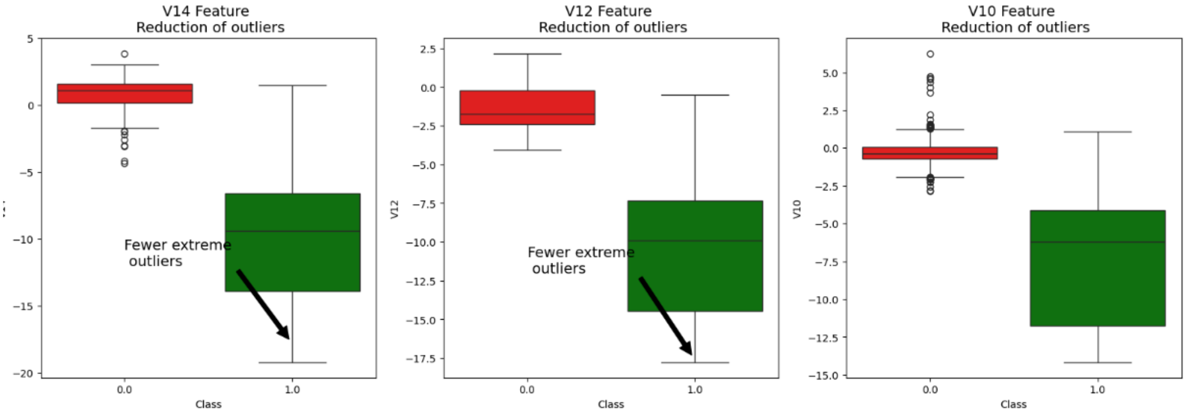
- Q1 represents the 25th percentile of the data.
- Q2 represents the 50th percentile of the data.
- Q3 represents the 75th percentile of the data.

IQR is the range between the first and the third quartiles namely Q1 and Q3: IQR = Q3 – Q1. The data points which fall below Q1 – 1.5 IQR or above Q3 + 1.5 IQR are outliers.

- To read about other outlier detection and removal techniques follw the link
- Here is how we are going to find and remove outliers from data using interquartile range method
- Go through the example to complete the cell below and run for yourself

```
f, (ax1,ax2,ax3) = plt.subplots(1,3, figsize=(20,6))

colors = ['red','green']

#feature V14
sns.boxplot(x='Class',y='V14',data=new_df,ax=ax1,palette=colors)
ax1.set_title("V14 Feature \n Reduction of outliers", fontsize=14)
ax1.annotate('Fewer extreme \n outliers', xy=(0.98, -17.5), xytext=(0, -12),arrowprops=dict(facecolor='black'),fontsize=14)

#feature v12
sns.boxplot(x='Class',y='V12',data=new_df,ax=ax2,palette=colors)
ax2.set_title("V12 Feature \n Reduction of outliers", fontsize=14)
ax2.annotate('Fewer extreme \n outliers', xy=(0.98, -17.3), xytext=(0, -12),arrowprops=dict(facecolor='black'),fontsize=14)

#feature V10
sns.boxplot(x='Class',y='V10',data=new_df,ax=ax3,palette=colors)
ax3.set_title("V10 Feature \n Reduction of outliers", fontsize=14)
ax3.annotate('Fewer extreme \n outliers', xy=(0.95, -16.5), xytext=(0, -12),arrowprops=dict(facecolor='black'),fontsize=14)
```

```
Text(0, -12, 'Fewer extreme \n outliers')
```



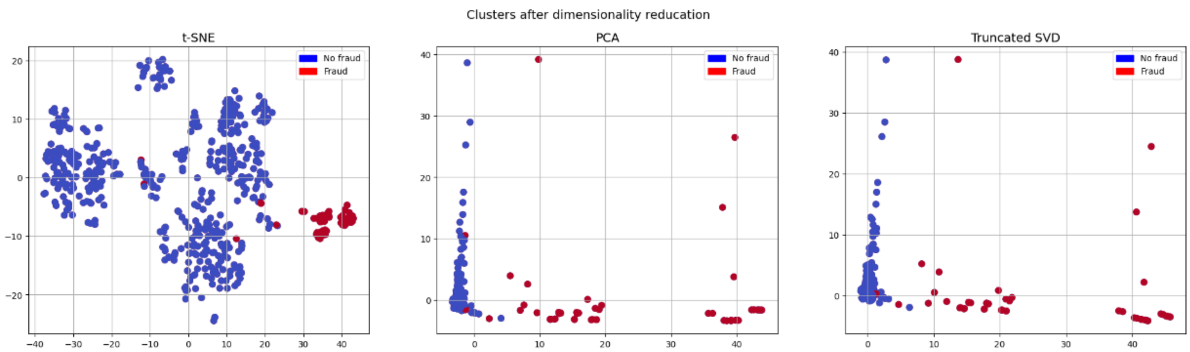
DIMENSIONALITY REDUCTION VISUALIZATION

We have a data that is high dimensional and visualising any patterns in higher than 3 is not possible so to see how our data would look like we are going to use dimensionality reduction techniques for visualisation of our data, we are going to use.

- TSNE
- Principal Component Analysis(PCA)

```python
f, (ax1,ax2,ax3) = plt.subplots(1,3,figsize=(24,6))
f.suptitle('Clusters after dimensionality reducation',fontsize=14)

blue_patch = mpatches.Patch(color='blue',label = 'No fraud')
red_patch = mpatches.Patch(color='red',label='Fraud')

# t-SNE scatter plot
ax1.scatter(X_reduced_tsne[:,0], X_reduced_tsne[:,1], c=(y == 0), cmap='coolwarm', label='No Fraud', linewidths=2)
ax1.scatter(X_reduced_tsne[:,0], X_reduced_tsne[:,1], c=(y == 1), cmap='coolwarm', label='Fraud', linewidths=2)
ax1.set_title('t-SNE', fontsize=14)
ax1.grid(True)
ax1.legend(handles=[blue_patch, red_patch])


# PCA scatter plot
ax2.scatter(X_reduced_pca[:,0], X_reduced_pca[:,1], c=(y == 0), cmap='coolwarm', label='No Fraud', linewidths=2)
ax2.scatter(X_reduced_pca[:,0], X_reduced_pca[:,1], c=(y == 1), cmap='coolwarm', label='Fraud', linewidths=2)
ax2.set_title('PCA', fontsize=14)
ax2.grid(True)
ax2.legend(handles=[blue_patch, red_patch])

# TruncatedSVD scatter plot
ax3.scatter(X_reduced_svd[:,0], X_reduced_svd[:,1], c=(y == 0), cmap='coolwarm', label='No Fraud', linewidths=2)
ax3.scatter(X_reduced_svd[:,0], X_reduced_svd[:,1], c=(y == 1), cmap='coolwarm', label='Fraud', linewidths=2)
ax3.set_title('Truncated SVD', fontsize=14)
ax3.grid(True)
ax3.legend(handles=[blue_patch, red_patch])
```

```
<matplotlib.legend.Legend at 0x79d50219f610>
```



IMPLEMENTING MODELS

As we have a classification problem on our hands we'll apply classification models and calculate the cross validation score to identify the best fit for our data.

Some of the models that we are going to use are.

- LogisticRegression
- KNeighborsClassifier
- Support Vector Classifier
- DecisionTreeClassifier

```python
from sklearn.model_selection import cross_val_score


for key, classifier in Models.items():
    classifier.fit(X_train, y_train)
    training_score = cross_val_score(classifier, X_train, y_train, cv=5)
    print("Classifier: ", classifier.__class__.__name__, "has a training score of", round(training_score.mean(), 2) * 100, '
```

```
Classifier:  LogisticRegression has a training score of 99.0 % accuracy score
Classifier:  KNeighborsClassifier has a training score of 99.0 % accuracy score
Classifier:  SVC has a training score of 99.0 % accuracy score
Classifier:  DecisionTreeClassifier has a training score of 99.0 % accuracy score
```

VALIDATION

Now we have our best set models trained using gridsearch now we move onto testing them on different metrices so as to know which one works best overall for us.

1. log_reg

2. knears_negihbors

3. svc

4. tree_clf

we'll evaluate model using:

- Cross_val_score
- ROC AUC Score
- ROC Curve
- Confusion Matrix, Classification report
- Average precison score, Area under precision recall curve

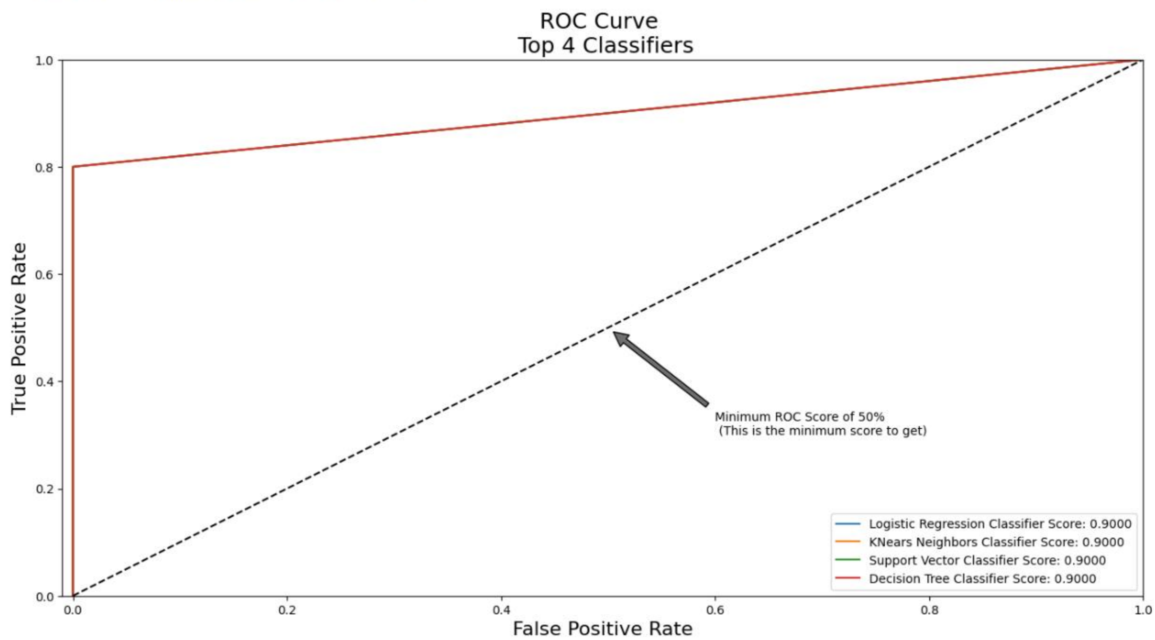
Now we find cross_validation scores for the models we got using GridsearchCV, use *log_reg*, *knears_negihbors*, *svc* and *tree_clf* in the below cell.

- Use the *cross_val_score* in the model training section to complete the code below.

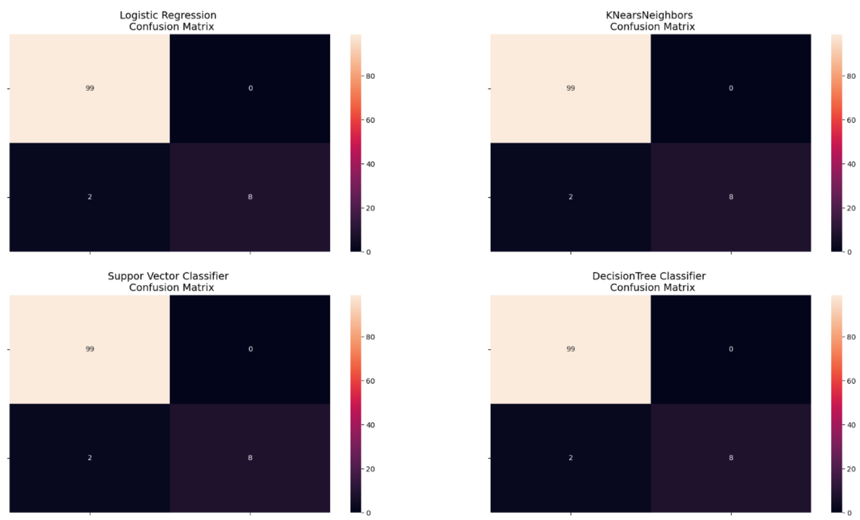
```
log_fpr, log_tpr, log_thresold = roc_curve(y_test, log_reg_pred)
knear_fpr, knear_tpr, knear_threshold = roc_curve(y_test, knears_pred)
svc_fpr, svc_tpr, svc_threshold = roc_curve(y_test, svc_pred)
tree_fpr, tree_tpr, tree_threshold = roc_curve(y_test, tree_pred)
```

```
plt.figure(figsize=(16,8))
plt.title('ROC Curve \n Top 4 Classifiers', fontsize=18)
plt.plot(log_fpr, log_tpr, label='Logistic Regression Classifier Score: {:.4f}'.format(roc_auc_score(y_test, log_reg_pred)))
plt.plot(knear_fpr, knear_tpr, label='KNears Neighbors Classifier Score: {:.4f}'.format(roc_auc_score(y_test, knears_pred)))
plt.plot(svc_fpr, svc_tpr, label='Support Vector Classifier Score: {:.4f}'.format(roc_auc_score(y_test, svc_pred)))
plt.plot(tree_fpr, tree_tpr, label='Decision Tree Classifier Score: {:.4f}'.format(roc_auc_score(y_test, tree_pred)))
plt.plot([0, 1], [0, 1], 'k--')
plt.axis([-0.01, 1, 0, 1])
plt.xlabel('False Positive Rate', fontsize=16)
plt.ylabel('True Positive Rate', fontsize=16)
plt.annotate('Minimum ROC Score of 50% \n (This is the minimum score to get)', xy=(0.5, 0.5), xytext=(0.6, 0.3),
            arrowprops=dict(facecolor='#6E726D', shrink=0.05))
plt.legend()
```

```
<matplotlib.legend.Legend at 0x79d501e59330>
```




```
[Text(0, 0.5, ''), Text(0, 1.5, '')]
```



TESTING

Now we'll test our models on original test data "***original_Xtest***".

First of all let's see confusion matrix for all our models.

- Taking predictions from logistic regression model, go ahead and complete the below cell to calculate these.

org_log_reg_pred = log_reg.predict(original_Xtest)

```
print('Logistic Regression:')
print(classification_report(original_ytest, org_log_reg_pred))

print('KNears Neighbors:')
print(classification_report(original_ytest, org_knears_pred))

print('Support Vector Classifier:')
print(classification_report(original_ytest, org_svc_pred))

print('Tree Classifier:')
print(classification_report(original_ytest, org_tree_pred))
```

```
Logistic Regression:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      2381
         1.0       0.67      1.00      0.80        10

    accuracy                           1.00      2391
   macro avg       0.83      1.00      0.90      2391
weighted avg       1.00      1.00      1.00      2391

KNears Neighbors:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      2381
         1.0       0.67      1.00      0.80        10

    accuracy                           1.00      2391
   macro avg       0.83      1.00      0.90      2391
weighted avg       1.00      1.00      1.00      2391

Support Vector Classifier:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      2381
         1.0       0.59      1.00      0.74        10

    accuracy                           1.00      2391
   macro avg       0.79      1.00      0.87      2391
weighted avg       1.00      1.00      1.00      2391

Tree Classifier:
              precision    recall  f1-score   support

         0.0       1.00      0.99      1.00      2381
         1.0       0.42      1.00      0.59        10

    accuracy                           0.99      2391
   macro avg       0.71      1.00      0.79      2391
weighted avg       1.00      0.99      1.00      2391
```

CONCLUSION:

In conclusion, a credit card fraud detection system is a vital component of financial security infrastructure, employing advanced data analysis techniques and machine learning algorithms to identify and mitigate fraudulent transactions. By leveraging extensive datasets and real-time monitoring capabilities, these systems can effectively detect anomalies and suspicious patterns, enabling prompt intervention to prevent financial losses and protect both cardholders and financial institutions. Continuous improvement and adaptation to evolving fraud tactics are essential to maintaining the system's effectiveness in safeguarding electronic payment systems and enhancing consumer trust in financial transactions.

FUTURE WORK REFERENCES:

1. "Credit Card Fraud Detection Using Machine Learning: A Review" by Shivanand Hulyal, Gururaj Hulyal, and Mahantesh Hanchinal.

2. "A Survey of Credit Card Fraud Detection Techniques: Data and Technique-Oriented Perspective" by A. L. M. Abdul Gafur, M. Shahjahan, and A. K. M. Jahangir Alam Majumder.

3. "Credit Card Fraud Detection Using Machine Learning Techniques: A Comparative Study" by Priyanka M. Wankhade and Prof. D. V. Jadhav.

4. "Credit Card Fraud Detection Using Convolutional Neural Networks" by Kiran R., Dhanya Pramod, and Sumithra Devi K. A.

5. "Fraud Detection in Credit Card Transactions Using Machine Learning Techniques: A Review" by Mohammad Golam Sohrab, Mohammed Nasser, and Mohammed Ibrahim AbuAlhaj.

6. "Credit Card Fraud Detection Using Supervised Machine Learning" by R. Fiondella, R. Martins, and M. Nappi.

7. "Detection of Credit Card Fraud Detection Using Machine Learning Techniques: A Review" by K. Renuka and Dr. K. Nirmala Devi.

8. "Comparative Study of Machine Learning Algorithms for Credit Card Fraud Detection" by R. L. Ukiwe, K. K. Agu, and E. U. Anyaoha.