Project 1

“Finance”

Credit Card Fraud Detection

Project Report

SUBMITTED IN PARTIAL FULFILLMENT REQUIREMENT FOR THE AWARD OF DEGREE OF

BACHELOR OF TECHNOLOGY

SUBMITTED BY :

Group 3

Ashish Saharia

Harshita Rupani

Khushi

Robin Yadav

UNDER THE SUPERVISION OF

Dr. Hirdesh Pharasi

SCHOOL OF ENGINEERING AND TECHNOLOGY

BML MUNJAL UNIVERSITY Gurugram, Haryana - 122413

BML MUNJAL UNIVERSITY™

FROM HERE TO THE WORLD

October, 2023

Abstract

This study focuses on enhancing transaction security in credit card transactions by efficiently detecting fraudulent activities while minimizing false positives. Leveraging a dataset comprising credit card transactions in September 2013 by European cardholders, the primary goal is to address the imbalance in fraudulent and non-fraudulent transactions. The aim is to achieve a more accurate and balanced approach to identifying unauthorized charges, critical for both consumers and financial institutions.

Introduction

Title: A Data-Driven Approach to Credit Card Fraud Detection

In today's digital landscape, the proliferation of financial transactions has heightened the risk of credit card fraud, posing significant challenges to consumers and financial institutions alike. This study focuses on the imperative task of fortifying transaction security by swiftly identifying unauthorized charges while minimizing false positives, which can lead to customer dissatisfaction and financial losses. The dataset utilized in this analysis comprises credit card transactions conducted by European cardholders in September 2013, encapsulating a snapshot of two days' worth of transactions.

Data Source Links: The dataset used for this analysis found at Kaggle.
Link: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud
Paper URL:
Research Paper 1: Credit Card Fraud Detection in e-Commerce: An Outlier Detection Approach.
Link: https://arxiv.org/abs/1811.02196
Research paper 2: Credit Card Fraud Detection - Machine Learning methods.
Link: https://ieeexplore.ieee.org/abstract/document/8717766/references
Research paper 3: Credit Card Fraud Detection Based on Machine and Deep Learning.
Link: https://ieeexplore.ieee.org/abstract/document/9078935

Within this dataset, there are 492 instances of fraudulent transactions out of a total of 284,807 transactions, resulting in a highly imbalanced dataset where fraudulent transactions account for only 0.172% of the total. The data features are predominantly numerical and have undergone a Principal Component Analysis (PCA) transformation to ensure confidentiality, thereby masking the original features' details.

The features 'V1' through 'V28' represent principal components derived from PCA, while 'Time' signifies the elapsed time in seconds between each transaction and the first transaction in the dataset. Additionally, the 'Amount' feature denotes the transaction value. The 'Class' attribute serves as the response variable, distinguishing between fraudulent transactions (Class 1) and legitimate transactions (Class 0).

The primary objective of this study is to develop robust fraud detection methodologies that strike a balance between accurately identifying fraudulent activities and minimizing the occurrence of false positives. Achieving this balance is paramount for enhancing customer trust, reducing revenue losses, and ensuring the sustained integrity of financial services amidst the burgeoning digital transaction landscape.

This project aims to explore advanced machine learning algorithms, possibly incorporating feature engineering techniques and specialized models that account for imbalanced data. The ultimate goal is to create a more resilient and adaptable fraud detection system that remains effective in identifying fraudulent transactions while minimizing disruptions to legitimate cardholders' experiences.

Discussion of Dataset

The dataset encompasses credit card transactions conducted by European cardholders in September 2013, spanning two days. Notably unbalanced, fraudulent transactions constitute only 0.172% of the 284,807 transactions recorded. The dataset primarily features principal components derived from PCA, with 'Time' (denoting the time gap between transactions) and 'Amount' (representing transaction value) being the sole non-transformed variables. The 'Class' variable distinguishes between fraud (Class 1) and non-fraud (Class 0) transactions.

Due to confidentiality constraints, the original features and additional background information are unavailable. Features V1 to V28 represent the principal components obtained through PCA. 'Time' measures the seconds elapsed between each transaction and the initial one, while 'Amount' indicates the transaction amount. The response variable 'Class' assumes a value of 1 for fraud and 0 for non-fraud. The dataset provides a valuable context for addressing the crucial task of recognizing fraudulent credit card transactions, essential for protecting customers from unauthorized charges.

Results and Discussion

- Importing necessary libraries including NumPy, Pandas, Matplotlib, Seaborn, and scikit-learn modules

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import gridspec
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score, matthews_corrcoef
from sklearn.metrics import confusion_matrix
```

- Loading the credit card dataset from a CSV file

```
data = pd.read_csv('/content/creditcard - dataset.csv')
```

- Displaying the first few rows, shape and summary statistics of the dataset

```
data.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	0.277
1	0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638
2	1	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771
3	1	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005
4	2	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798

5 rows × 31 columns

```
data.shape
```

```
(23323, 31)
```

```
data.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Time	23323.0	17878.150667	11286.921435	0.000000	6307.500000	19915.000000	28769.000000	32697.000000
V1	23323.0	-0.242770	1.900322	-30.552380	-0.959617	-0.293051	1.164536	1.960497
V2	23323.0	0.205955	1.536833	-40.978852	-0.368901	0.197782	0.850538	16.713389
V3	23323.0	0.728501	1.734579	-31.103685	0.289977	0.879225	1.509068	4.101716
V4	23323.0	0.250949	1.442832	-5.172595	-0.656219	0.216866	1.122700	11.927512
V5	23323.0	-0.187743	1.444970	-42.147898	-0.764524	-0.216723	0.324452	34.099309
V6	23323.0	0.081674	1.329037	-23.496714	-0.663843	-0.180224	0.474047	22.529298
V7	23323.0	-0.134119	1.343897	-26.548144	-0.595766	-0.066809	0.450587	36.677268
V8	23323.0	0.020994	1.389895	-41.484823	-0.167740	0.027044	0.284926	20.007208

- Seperating the dataset into fraud and valid transactions and displaying the information about fraud cases

```
fraud = data[data.Class == 1]
valid = data[data.Class == 0]

print(fraud)
```

```
        Time        V1        V2        V3        V4        V5        V6  \
541      406 -2.312227  1.951992 -1.609851  3.997906 -0.522188 -1.426545
623      472 -3.043541 -3.157307  1.088463  2.288644  1.359805 -1.064823
4920    4462 -2.303350  1.759247 -0.359745  2.330243 -0.821628 -0.075788
6108    6986 -4.397974  1.358367 -2.592844  2.679787 -1.128131 -1.706536
6329    7519  1.234235  3.019740 -4.304597  4.732795  3.624201 -1.357746
...      ...       ...       ...       ...       ...       ...       ...
30442  35926 -3.896583  4.518355 -4.454027  5.547453 -4.121459 -1.163407
30473  35942 -4.194074  4.382897 -5.118363  4.455230 -4.812621 -1.224645
30496  35953 -4.844372  5.649439 -6.730396  5.252842 -4.409566 -1.740767
31002  36170 -5.685013  5.776516 -7.064977  5.902715 -4.715564 -1.755633
33276  37167 -7.923891 -5.198360 -3.000024  4.420666  2.272194 -3.394483
```

```
              V7        V8        V9  ...       V21       V22       V23  \
541    -2.537387  1.391657 -2.770089  ...  0.517232 -0.035049 -0.465211
623     0.325574 -0.067794 -0.270953  ...  0.661696  0.435477  1.375966
4920    0.562320 -0.399147 -0.238253  ... -0.294166 -0.932391  0.172726
6108   -3.496197 -0.248778 -0.247768  ...  0.573574  0.176968 -0.436207
6329    1.713445 -0.496358 -1.282858  ... -0.379068 -0.704181 -0.656805
...          ...       ...       ...  ...       ...       ...       ...
30442 -6.805053  2.928356 -4.917130  ...  1.691042  0.920021 -0.151104
30473 -7.281328  3.332250 -3.679659  ...  1.550473  0.614573  0.028521
30496 -6.311699  3.449167 -5.416284  ...  1.194888 -0.845753  0.190674
31002 -6.958679  3.877795 -5.541529  ...  1.128641 -0.962960 -0.110045
33276 -5.283435  0.131619  0.658176  ... -0.734308 -0.599926 -4.908301

             V24       V25       V26       V27       V28  Amount  Class
541     0.320198  0.044519  0.177840  0.261145 -0.143276    0.00    1.0
623    -0.293803  0.279798 -0.145362 -0.252773  0.035764  529.00    1.0
4920   -0.087330 -0.156114 -0.542628  0.039566 -0.153029  239.93    1.0
6108   -0.053502  0.252405 -0.657488 -0.827136  0.849573   59.00    1.0
6329   -1.632653  1.488901  0.566797 -0.010016  0.146793    1.00    1.0
...          ...       ...       ...       ...       ...     ...    ...
30442   0.011007  0.080303  0.412191  0.635789  0.501050    4.56    1.0
30473   0.013704 -0.149512 -0.131687  0.473934  0.473757   14.46    1.0
30496  -0.216443 -0.325033 -0.270328  0.210214  0.391855  111.70    1.0
31002  -0.177733 -0.089175 -0.049447  0.303445  0.219380  111.70    1.0
33276   0.410170 -1.167660  0.520508  1.937421 -1.552593   12.31    1.0

[103 rows x 31 columns]
```

- Calculating and printing the fraction of fraud transactions compared to valid transactions

```python
print(f'Fraud Cases: {len(fraud)}')
print(f'Valid Transactions: {len(valid)}')
```

```
Fraud Cases: 103
Valid Transactions: 37762
```

```python
outlierFraction = len(fraud) / float(len(valid))
outlierFraction
```

```
0.0027276097664318626
```

- Displaying descriptive statistics for the ‘Amount’ feature in both fraud and valid transactions

```
fraud.Amount.describe()
```

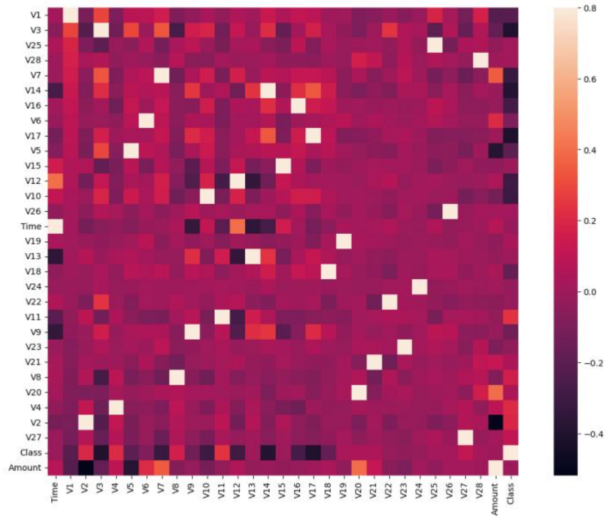
```
count     103.000000
mean       90.471165
std       247.173335
min         0.000000
25%         1.000000
50%         3.760000
75%        99.990000
max      1809.680000
Name: Amount, dtype: float64
```

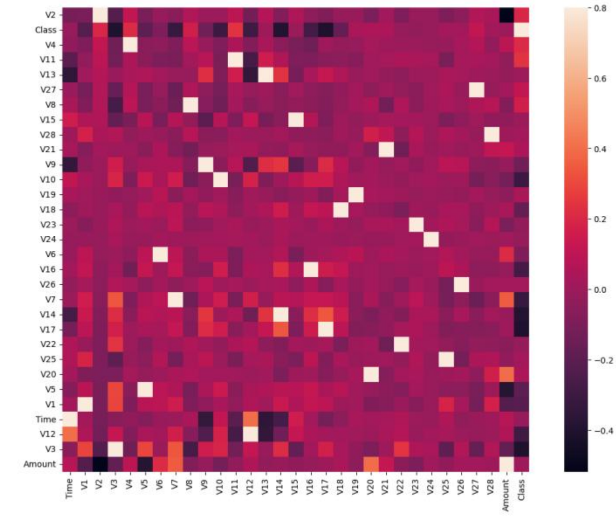
```
valid.Amount.describe()
```

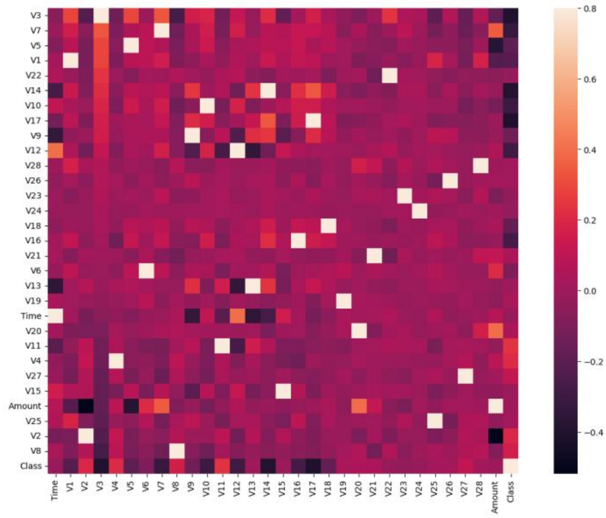
```
count    37762.000000
mean        86.279875
std        234.019053
min          0.000000
25%          7.300000
50%         22.900000
75%         77.787500
max       7879.420000
Name: Amount, dtype: float64
```

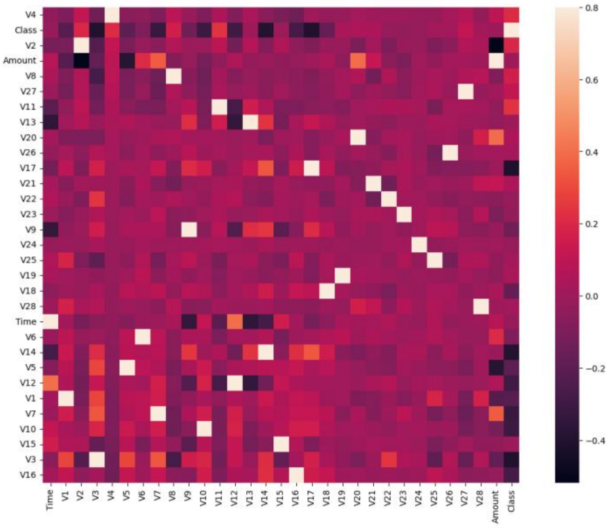
- Generating the heatmaps for 28 features (‘V1’ to ‘V28’) to visualize the relationships

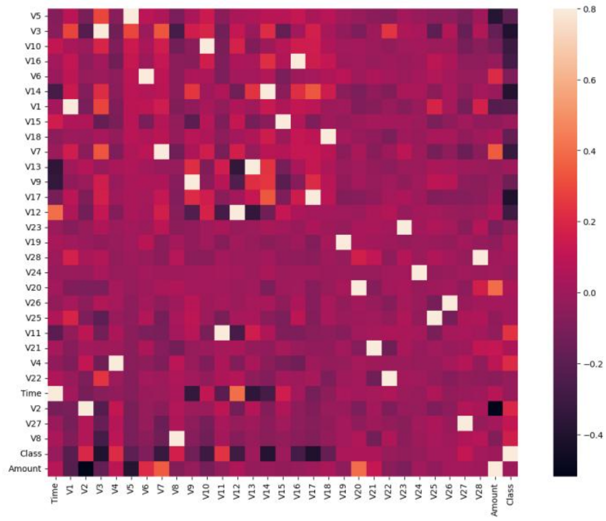
```
for a in range(1,26):
  corrmat = data.corr()
  sorted_columns = corrmat['V'+str(a)].sort_values(ascending=False).index
  corrmat = corrmat.reindex(sorted_columns).loc[sorted_columns]

  fig = plt.figure(figsize=(15, 10))
  sns.heatmap(corrmat, vmax=0.8, square=True)
  plt.show()
```

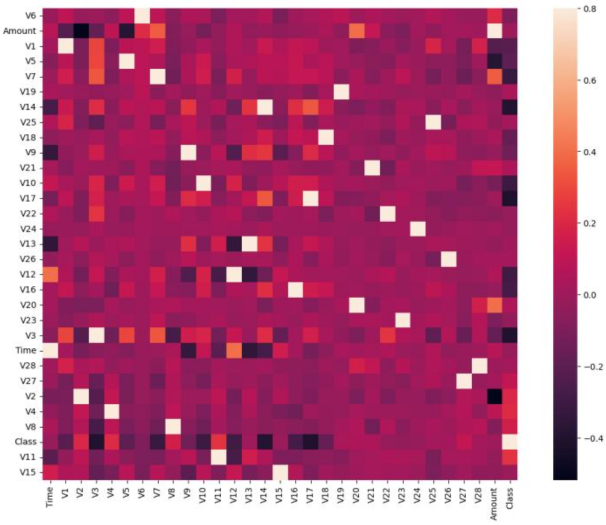


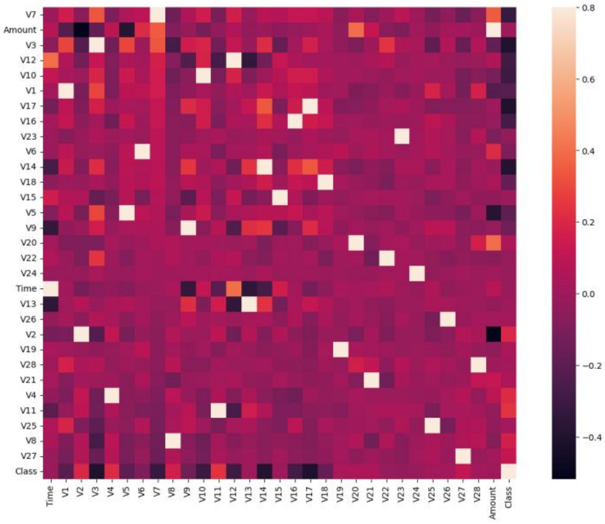


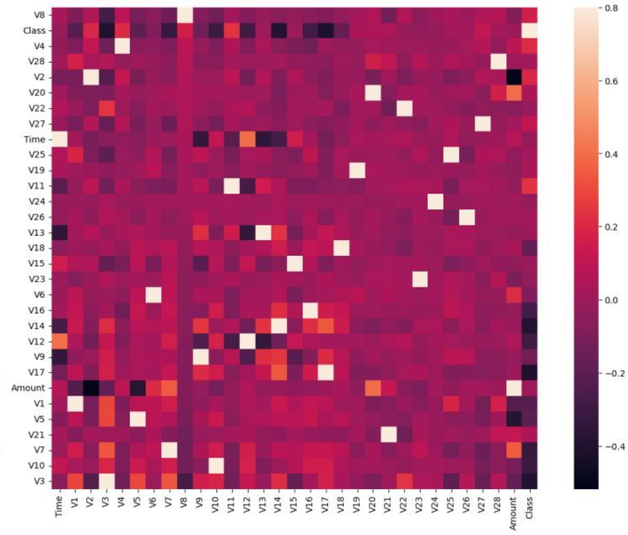


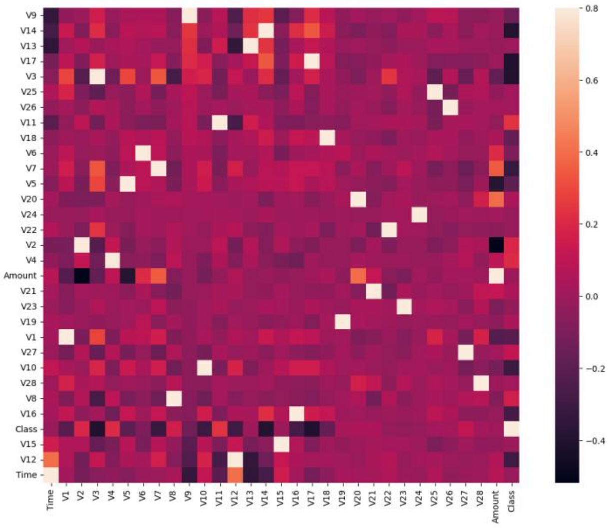


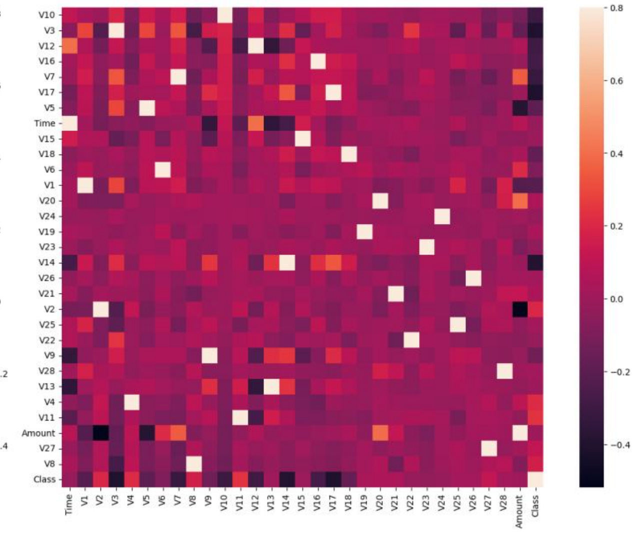


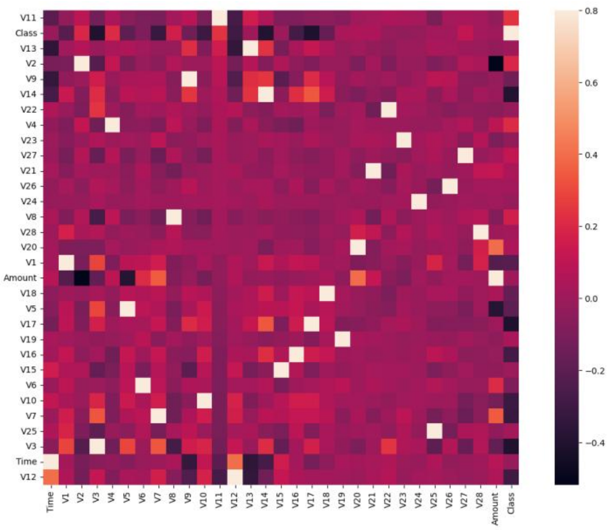


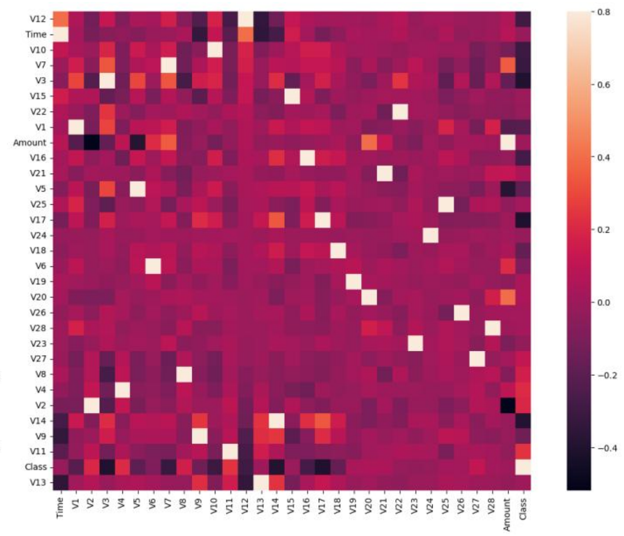

V7 Amount V3 V12 V10 V1 V17 V16 V23 V6 V14 V18 V15 V5 V9 V20 V22 V24 Time V13 V26 V2 V19 V28 V21 V4 V11 V25 V8 V27 Class
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 Amount Class
0.8 0.6 0.4 0.2 0.0 −0.2 −0.4

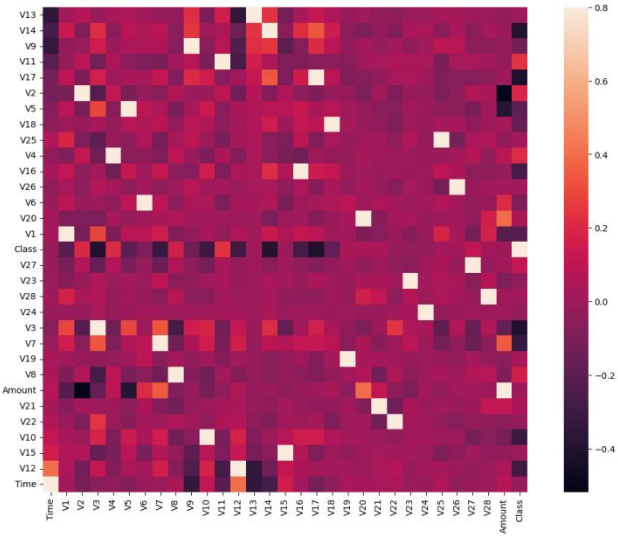

V8 Class V4 V28 V2 V20 V22 V27 Time V25 V19 V11 V24 V26 V13 V18 V15 V23 V6 V16 V14 V12 V9 V17 Amount V1 V5 V21 V7 V10 V3
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 Amount Class
0.8 0.6 0.4 0.2 0.0 −0.2 −0.4

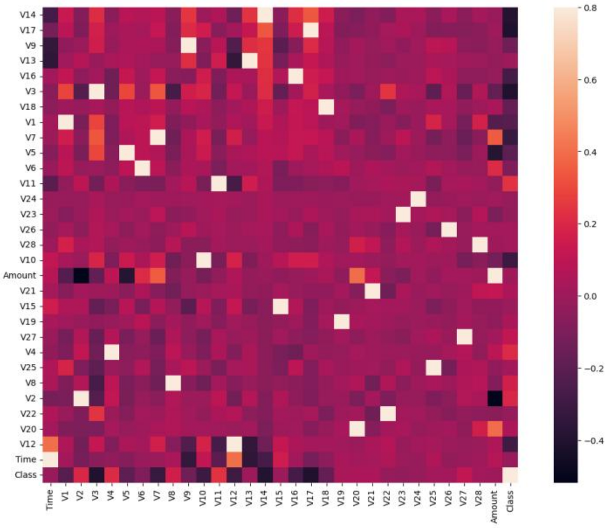

V9 V14 V13 V17 V3 V25 V26 V11 V18 V6 V7 V5 V20 V24 V22 V2 V4 Amount V21 V23 V19 V1 V27 V10 V28 V8 V16 Class V15 V12 Time
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 Amount Class
0.8 0.6 0.4 0.2 0.0 −0.2 −0.4

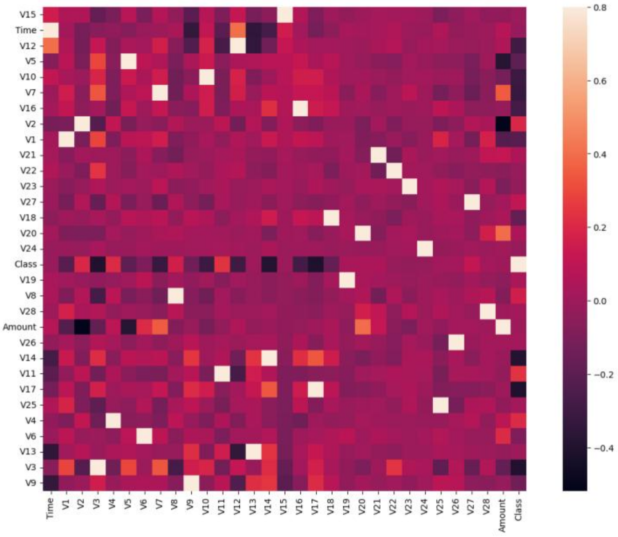

V10 V3 V12 V16 V7 V17 V5 Time V15 V18 V6 V1 V20 V24 V19 V23 V14 V26 V21 V2 V25 V22 V9 V28 V13 V4 V11 Amount V27 V8 Class
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 Amount Class
0.8 0.6 0.4 0.2 0.0 −0.2 −0.4

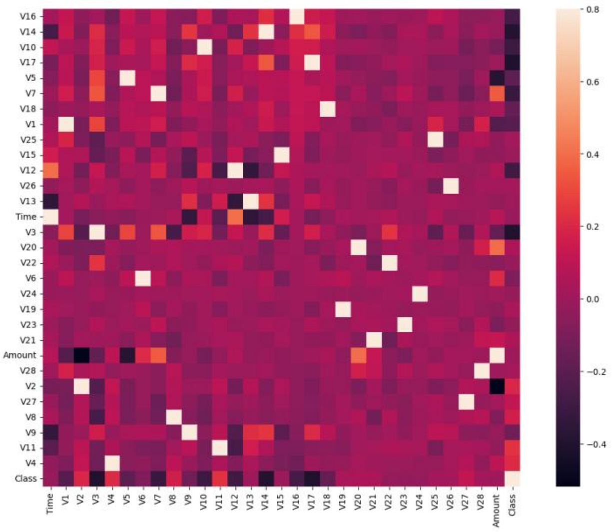

V11 Class V13 V2 V9 V14 V22 V4 V23 V27 V21 V26 V24 V8 V28 V20 V1 Amount V18 V5 V17 V19 V16 V15 V6 V10 V7 V25 V3 Time V12
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 Amount Class
0.8 0.6 0.4 0.2 0.0 −0.2 −0.4

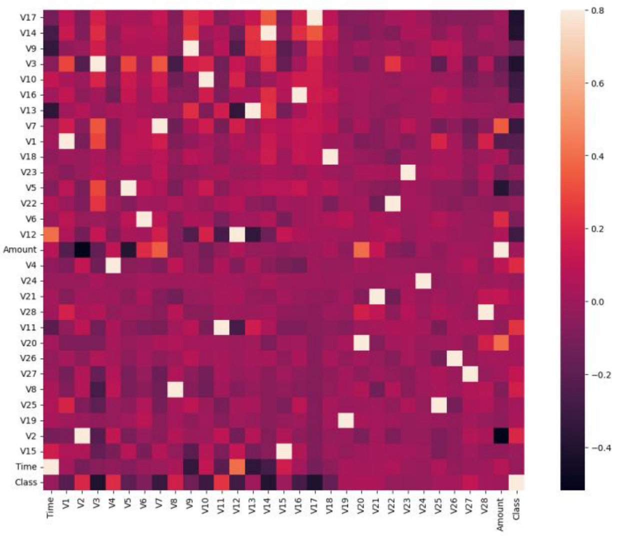

V12 Time V10 V7 V3 V15 V22 V1 Amount V16 V21 V5 V25 V17 V24 V18 V6 V19 V20 V26 V28 V23 V27 V8 V4 V2 V14 V9 V11 Class V13
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 Amount Class
0.8 0.6 0.4 0.2 0.0 −0.2 −0.4

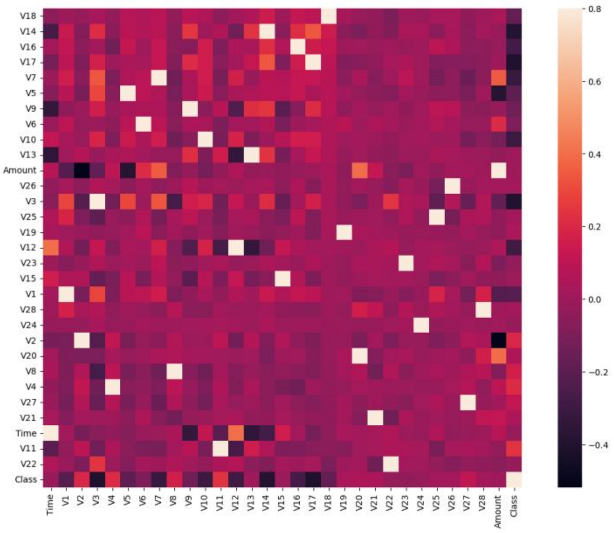

V13 V14 V9 V11 V17 V2 V5 V18 V25 V4 V16 V26 V6 V20 V1 Class V27 V23 V28 V24 V3 V7 V19 V8 Amount V21 V22 V10 V15 V12 Time
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 Amount Class
0.8 0.6 0.4 0.2 0.0 −0.2 −0.4

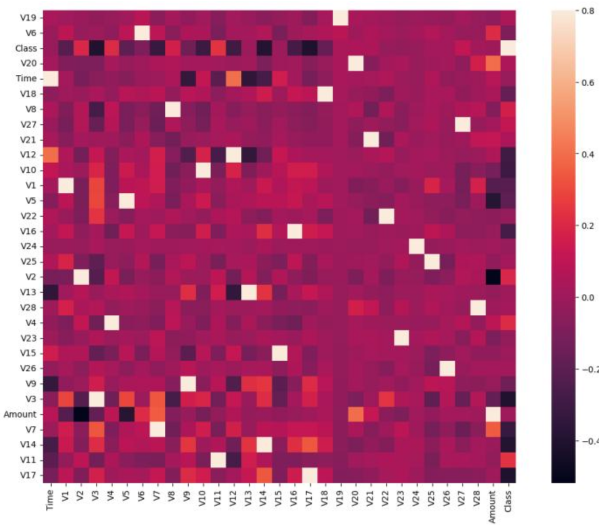

V14 V17 V9 V13 V16 V3 V18 V1 V7 V5 V6 V11 V24 V23 V26 V28 V10 Amount V21 V15 V19 V27 V4 V25 V8 V2 V22 V20 V12 Time Class
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 Amount Class
0.8 0.6 0.4 0.2 0.0 −0.2 −0.4

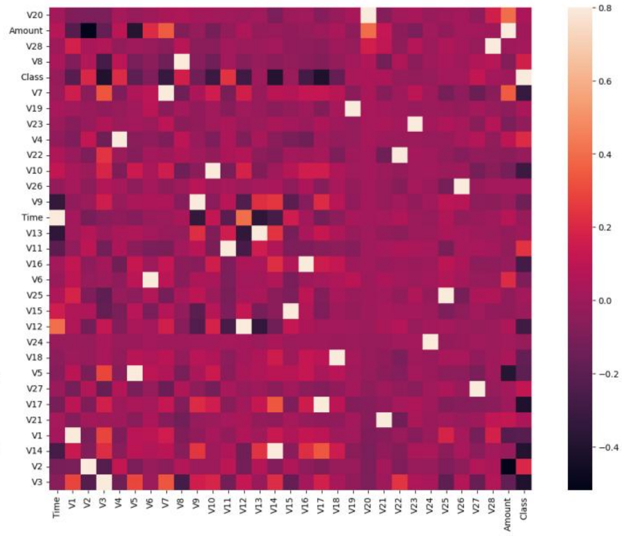

V15 Time V12 V5 V10 V7 V16 V2 V1 V21 V22 V23 V27 V18 V20 V24 Class V19 V8 V28 Amount V26 V14 V11 V17 V25 V4 V6 V13 V3 V9
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 Amount Class
0.8 0.6 0.4 0.2 0.0 −0.2 −0.4

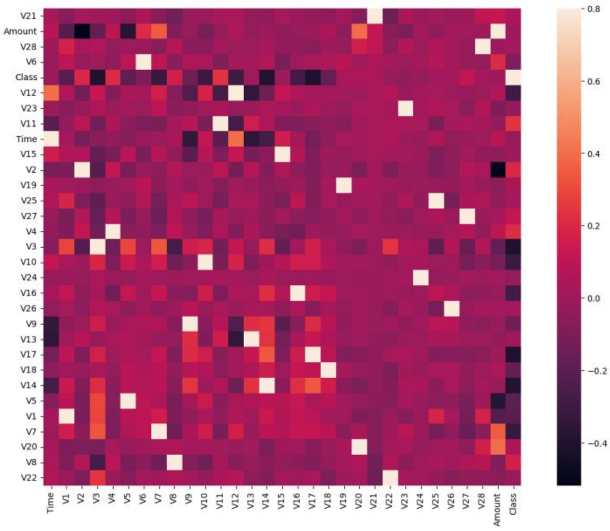

V16 V14 V10 V17 V5 V7 V18 V1 V25 V15 V12 V26 V13 Time V3 V20 V22 V6 V24 V19 V23 V21 Amount V28 V2 V27 V8 V9 V11 V4 Class
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 Amount Class
0.8 0.6 0.4 0.2 0.0 −0.2 −0.4

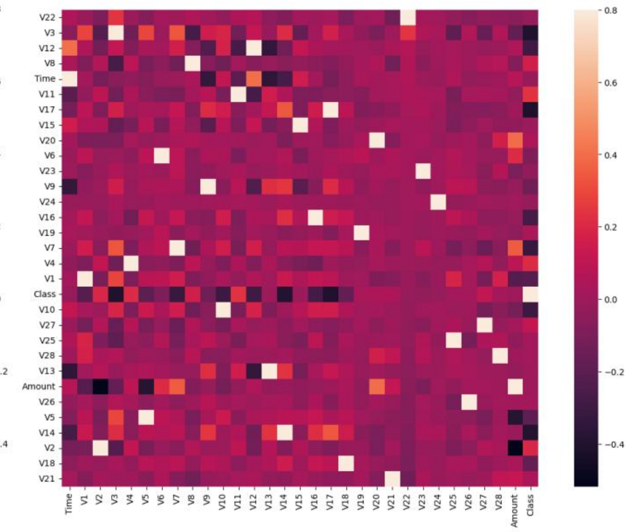

V17 V14 V9 V3 V10 V16 V13 V7 V1 V18 V23 V5 V22 V6 V12 Amount V4 V24 V21 V28 V11 V20 V26 V27 V8 V25 V19 V2 V15 Time Class
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 Amount Class
0.8 0.6 0.4 0.2 0.0 −0.2 −0.4

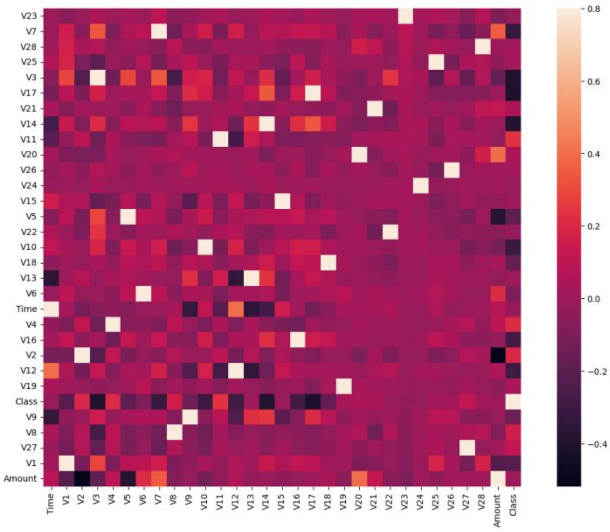

V18 V14 V16 V17 V7 V5 V9 V6 V10 V13 Amount V26 V3 V25 V19 V12 V23 V15 V1 V28 V24 V2 V20 V8 V4 V27 V21 Time V11 V22 Class
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 Amount Class
0.8 0.6 0.4 0.2 0.0 −0.2 −0.4

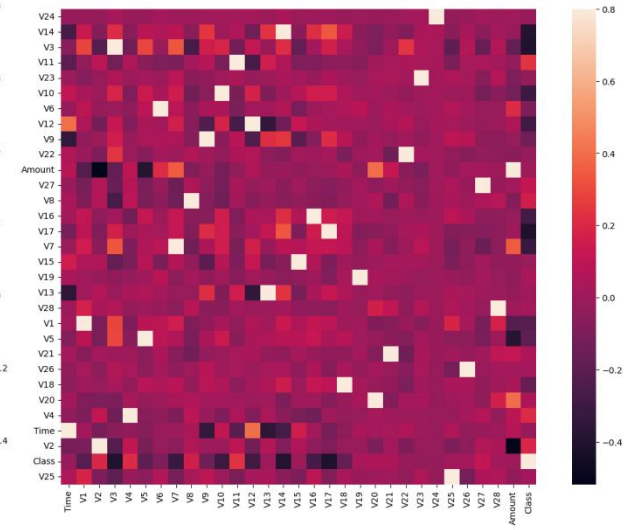

V19 V6 Class V20 Time V18 V8 V27 V21 V12 V10 V1 V5 V22 V16 V24 V25 V2 V13 V28 V4 V23 V15 V26 V9 V3 Amount V7 V14 V11 V17
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 Amount Class
0.8 0.6 0.4 0.2 0.0 −0.2 −0.4

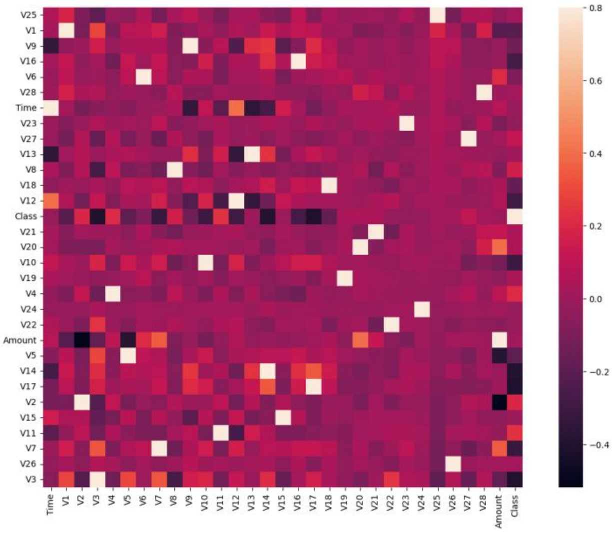

V20 Amount V28 V8 Class V7 V19 V23 V4 V22 V10 V26 V9 Time V13 V11 V16 V6 V25 V15 V12 V24 V18 V5 V27 V17 V21 V1 V14 V2 V3
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 Amount Class
0.8 0.6 0.4 0.2 0.0 −0.2 −0.4

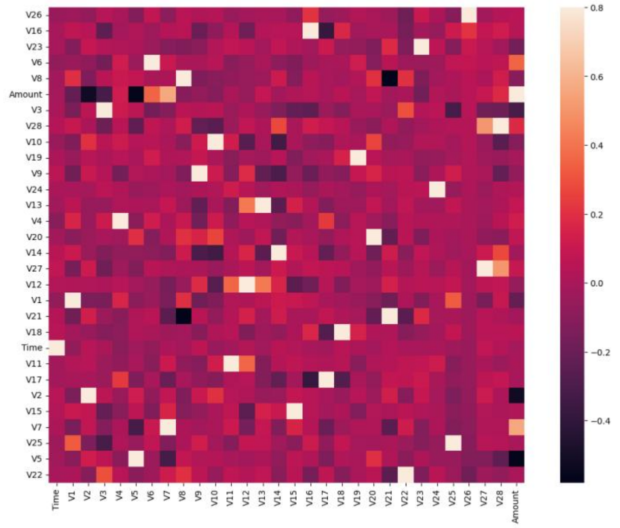

V21 Amount V28 V6 Class V12 V23 V11 Time V15 V2 V19 V25 V27 V4 V3 V10 V24 V16 V26 V9 V13 V17 V18 V14 V5 V1 V7 V20 V8 V22
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 Amount Class
0.8 0.6 0.4 0.2 0.0 −0.2 −0.4

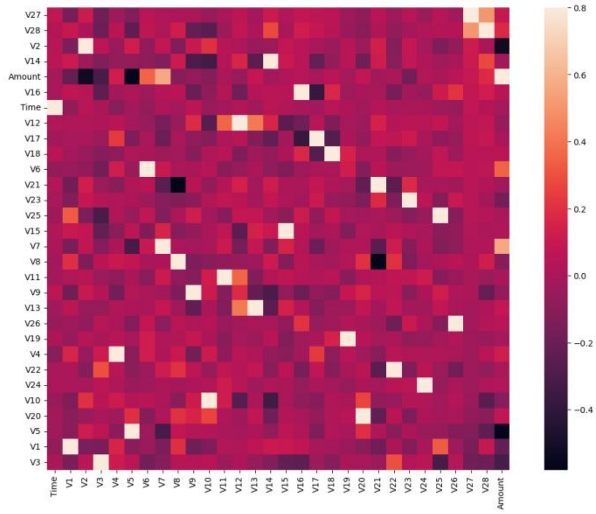

V22 V3 V12 V8 Time V11 V17 V15 V20 V6 V23 V9 V24 V16 V19 V7 V4 V1 Class V10 V27 V25 V28 V13 Amount V26 V5 V14 V2 V18 V21
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 Amount Class
0.8 0.6 0.4 0.2 0.0 −0.2 −0.4

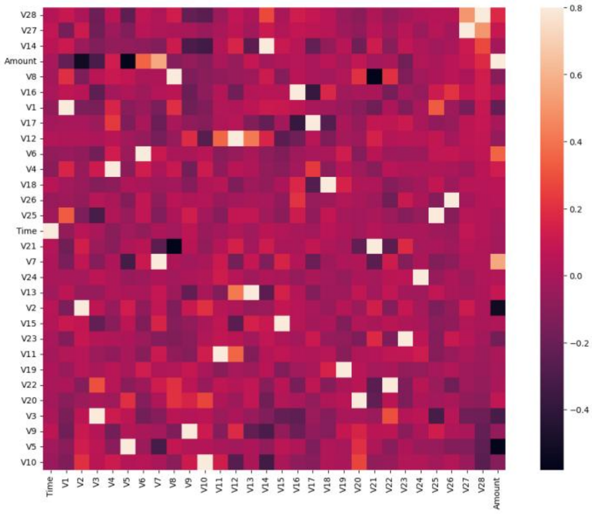

V23 V7 V28 V25 V3 V17 V21 V14 V11 V20 V26 V24 V15 V5 V22 V10 V18 V13 V6 Time V4 V16 V2 V12 V19 Class V9 V8 V27 V1 Amount
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 Amount Class
0.8 0.6 0.4 0.2 0.0 −0.2 −0.4


V24 V14 V3 V11 V23 V10 V6 V12 V9 V22 Amount V27 V8 V16 V17 V7 V15 V19 V13 V28 V1 V5 V21 V26 V18 V20 V4 Time V2 Class V25
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 Amount Class
0.8 0.6 0.4 0.2 0.0 −0.2 −0.4


V25 V1 V9 V16 V6 V28 Time V23 V27 V13 V8 V18 V12 Class V21 V20 V10 V19 V4 V24 V22 Amount V5 V14 V17 V2 V15 V11 V7 V26 V3
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 Amount Class
0.8 0.6 0.4 0.2 0.0 −0.2 −0.4


V26 V16 V23 V6 V8 Amount V3 V28 V10 V19 V9 V24 V13 V4 V20 V14 V27 V12 V1 V21 V18 Time V11 V17 V2 V15 V7 V25 V5 V22
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 Amount
0.8 0.6 0.4 0.2 0.0 −0.2 −0.4


V27 V28 V2 V14 Amount V16 Time V12 V17 V18 V6 V21 V23 V25 V15 V7 V8 V11 V9 V13 V26 V19 V4 V22 V24 V10 V20 V5 V1 V3
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 Amount
0.8 0.6 0.4 0.2 0.0 −0.2 −0.4


V28 V27 V14 Amount V8 V16 V1 V17 V12 V6 V4 V18 V26 V25 Time V21 V7 V24 V13 V2 V15 V23 V11 V19 V22 V20 V3 V9 V5 V10
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 Amount
0.8 0.6 0.4 0.2 0.0 −0.2 −0.4

- Calculating and displaying the accuracy,precision, recall and F1 score for model evaluation

✧ This is the result we found in Research Paper 2:

RF model obtained following results

- precision: 96.38%,
- recall: 81.63%,
- accuracy: 99.96%.

✧ This is our result:

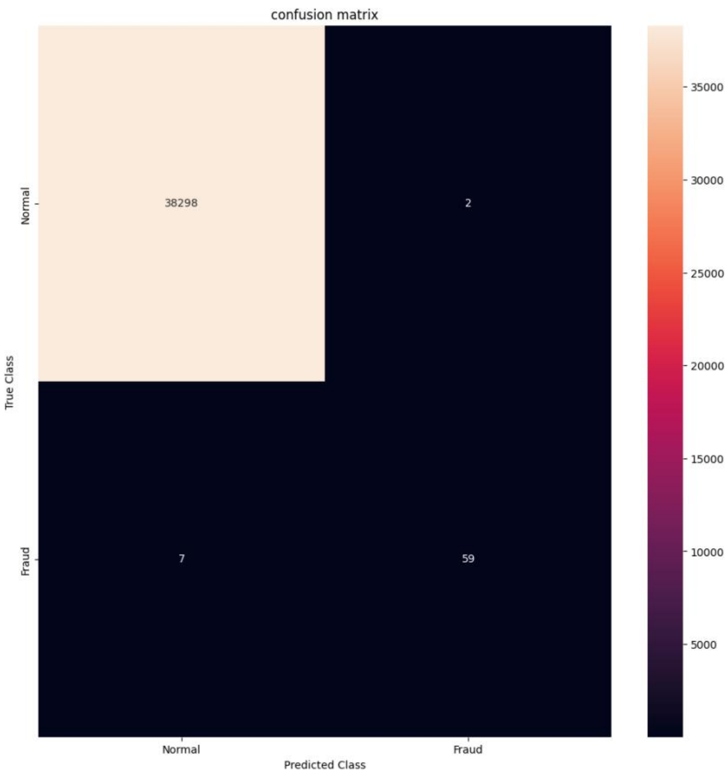
```
[ ] acc = accuracy_score(y_test, pred)
    acc

    0.9997654172965647

[ ] prec = precision_score(y_test, pred)
    prec

    0.9672131147540983

[ ] rec = recall_score(y_test, pred)
    rec

    0.8939393939393939

[ ] f1 = f1_score(y_test, pred)
    f1

    0.9291338582677166
```

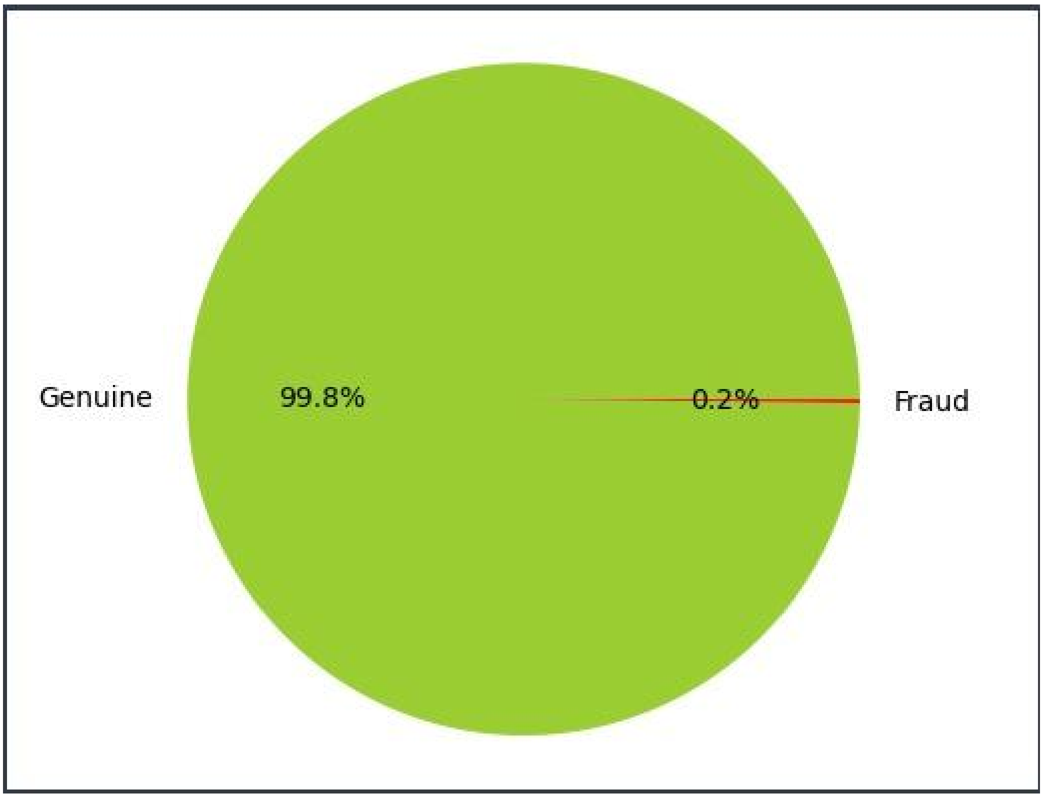
- Displaying the confusion matrix to visualize the model performance on classifying 'Normal' and 'Fraud' transactions. The matrix includes counts of true positive, true negative, false positive and false negative predictions

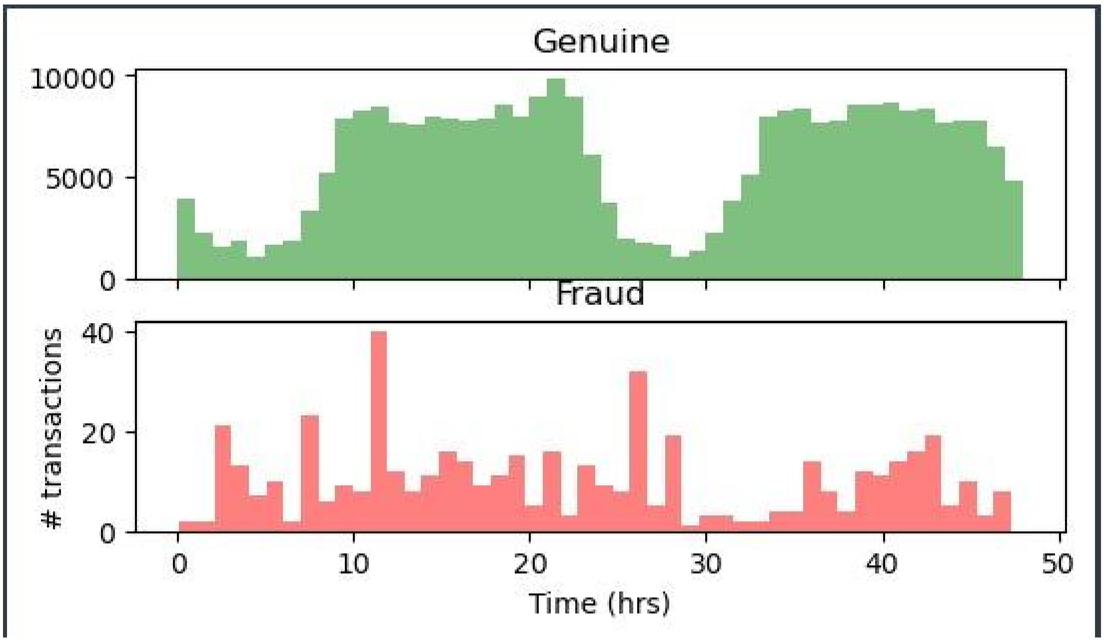
✧ This is the result we found in Research Paper 2:

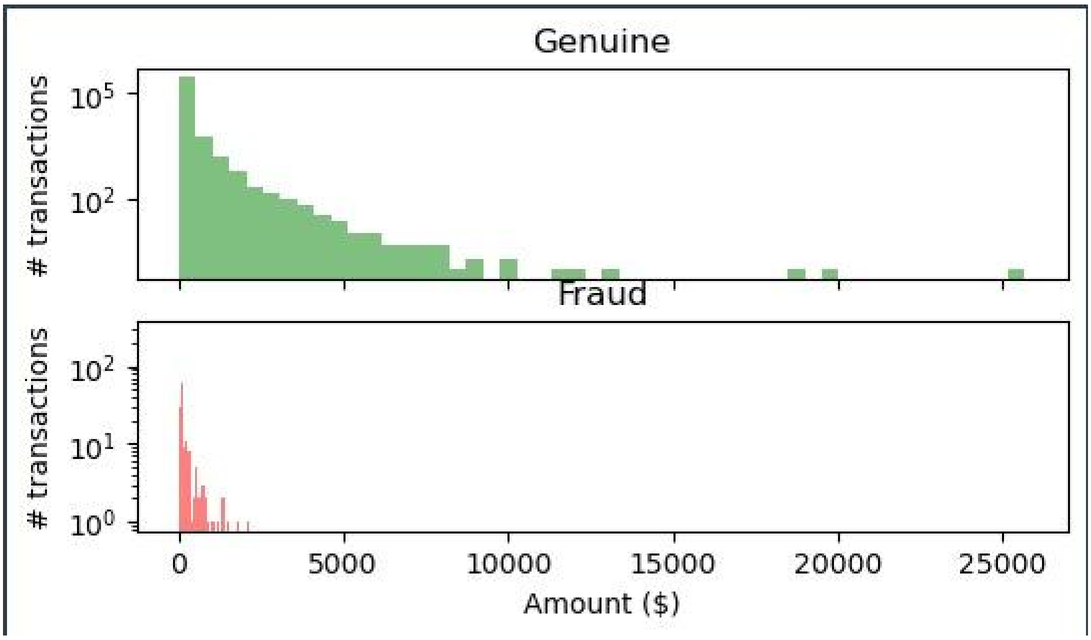
		Predicted	
		0	1
Actual	0	56861	3
	1	18	80

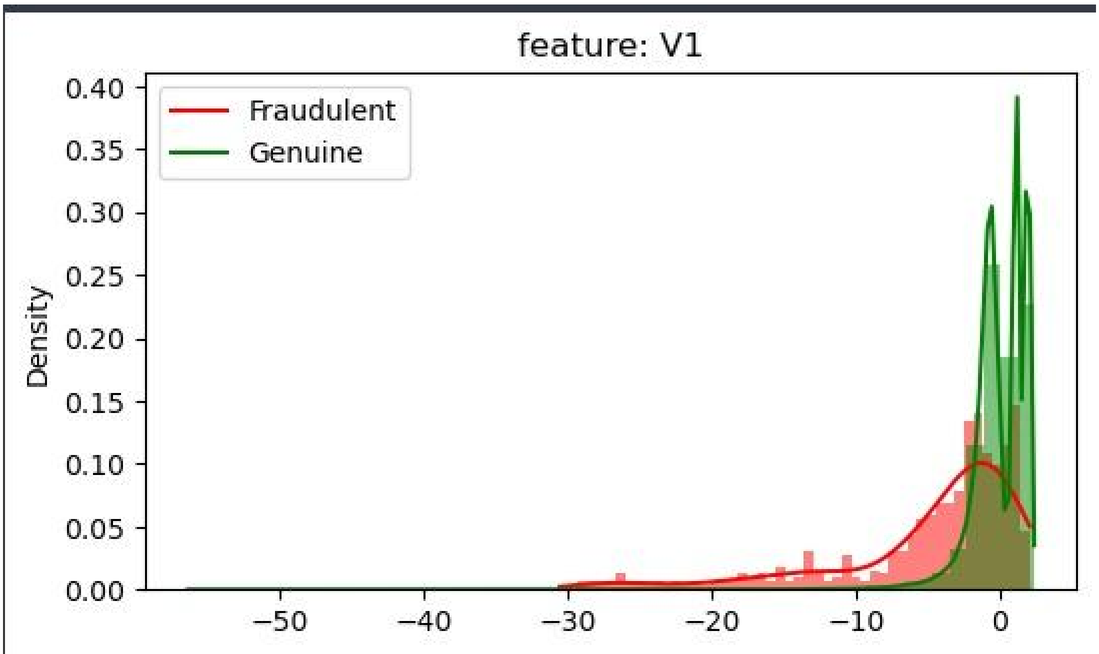
✧ This is our result:

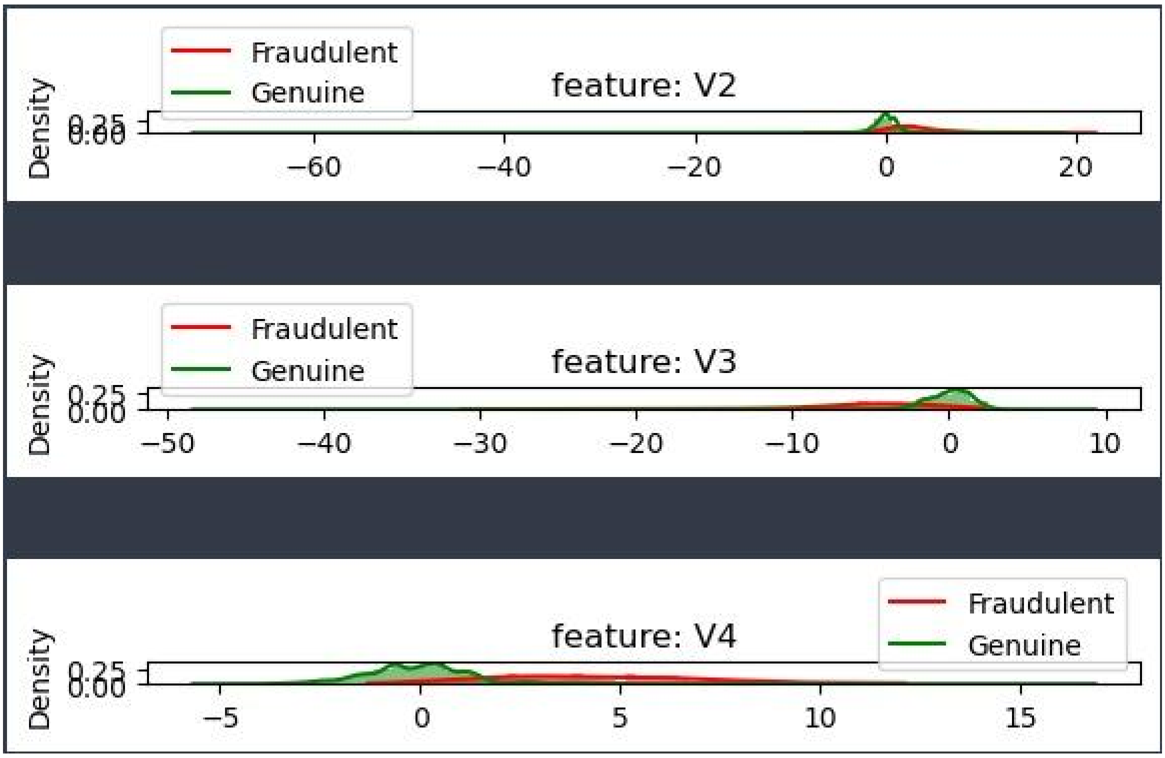


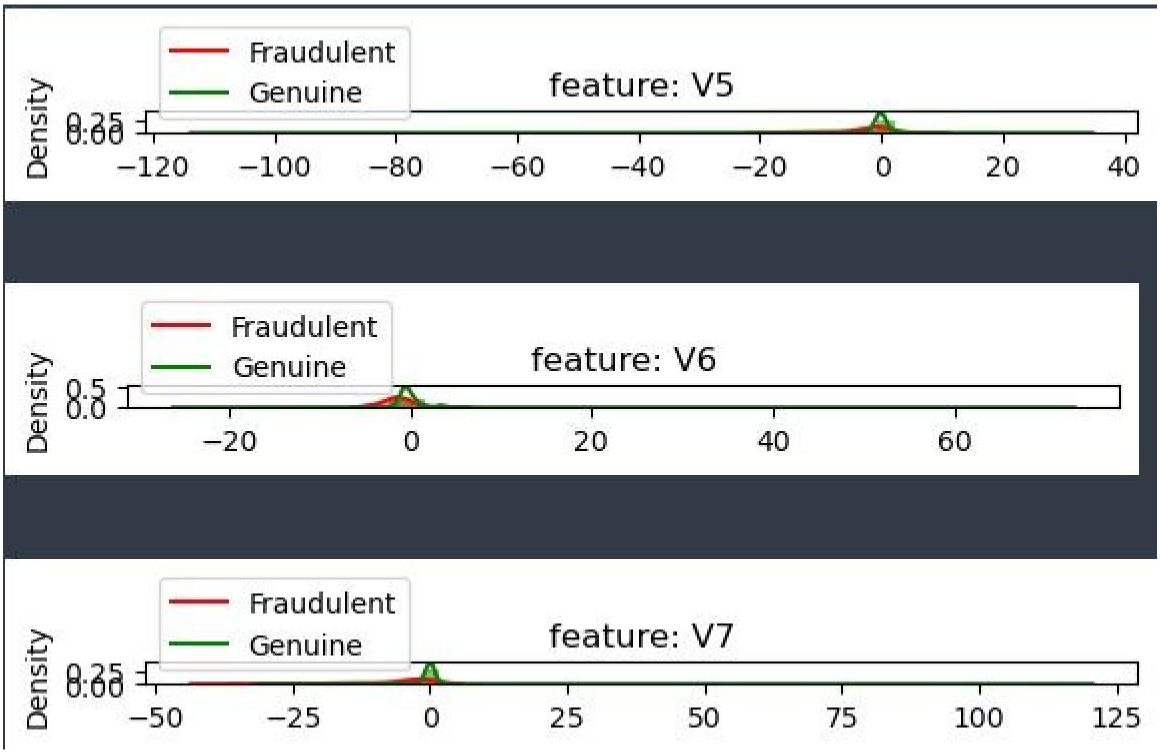

Genuine
99.8%
0.2%
Fraud

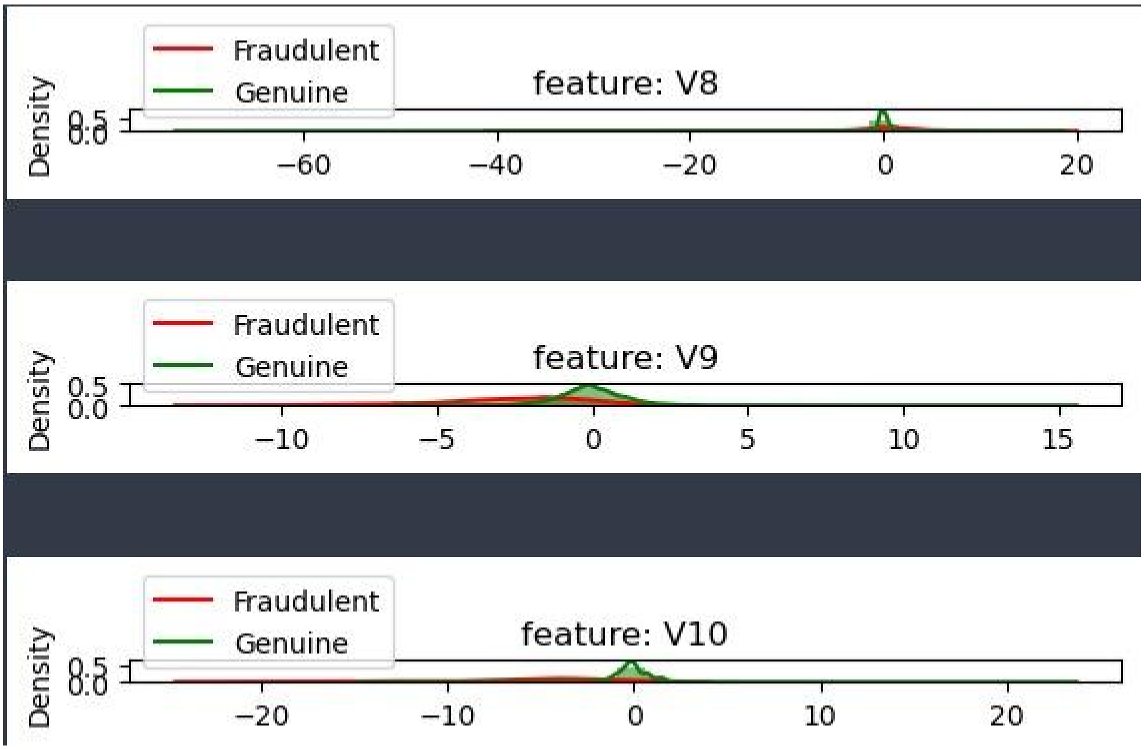

Genuine
10000
5000
0
Fraud
40
20
0
transactions
0
10
20
30
40
50
Time (hrs)

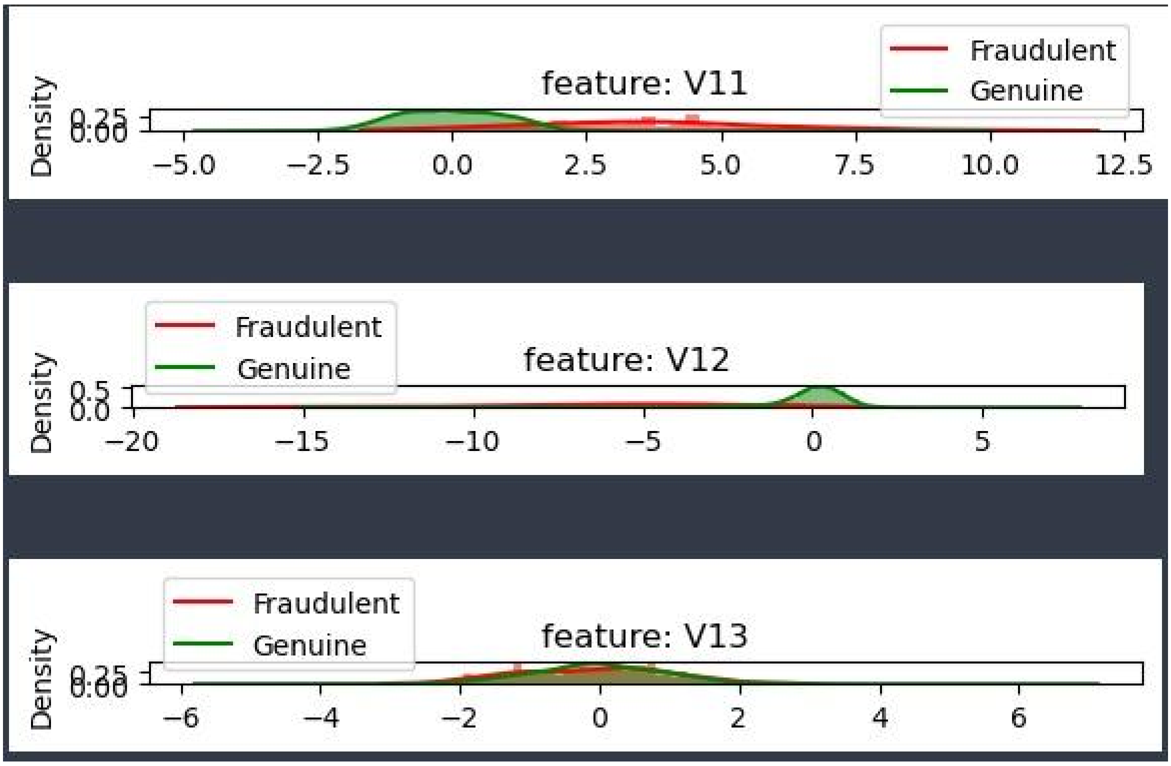

Genuine
transactions
10^5
10^2
Fraud
transactions
10^2
10^1
10^0
0
5000
10000
15000
20000
25000
Amount ($)


feature: V1
Fraudulent
Genuine
Density
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−50
−40
−30
−20
−10
0


Fraudulent
Genuine
feature: V2
Density
0.25
0.00
−60
−40
−20
0
20
Fraudulent
Genuine
feature: V3
Density
0.25
0.00
−50
−40
−30
−20
−10
0
10
Fraudulent
Genuine
feature: V4
Density
0.25
0.00
−5
0
5
10
15


Fraudulent
Genuine
feature: V5
Density
0.25
0.00
−120
−100
−80
−60
−40
−20
0
20
40
Fraudulent
Genuine
feature: V6
Density
0.5
0.0
−20
0
20
40
60
Fraudulent
Genuine
feature: V7
Density
0.25
0.00
−50
−25
0
25
50
75
100
125


Fraudulent
Genuine
feature: V8
Density
0.5
0.0
−60
−40
−20
0
20
Fraudulent
Genuine
feature: V9
Density
0.5
0.0
−10
−5
0
5
10
15
Fraudulent
Genuine
feature: V10
Density
0.5
0.0
−20
−10
0
10
20


Fraudulent
Genuine
feature: V11
Density
0.25
0.00
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
Fraudulent
Genuine
feature: V12
Density
0.5
0.0
−20
−15
−10
−5
0
5
Fraudulent
Genuine
feature: V13
Density
0.25
0.00
−6
−4
−2
0
2
4
6

```python
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
# Case-NB-1 : do not drop anything
drop_list = []
X_train, X_test, y_train, y_test = split_data(df, drop_list)
y_pred, y_pred_prob = get_predictions(GaussianNB(), X_train, y_train, X_test)
print_scores(y_test,y_pred,y_pred_prob)
```

```
Index(['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10', 'V11',
       'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20', 'V21',
       'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Class', 'Time_Hr',
       'scaled_Amount'],
      dtype='object')
train-set size:  227845
test-set size:  56962
fraud cases in test-set:  98
train-set confusion matrix:
 [[222480   4971]
 [    69    325]]
test-set confusion matrix:
 [[55535  1329]
 [   15    83]]
recall score:  0.8469387755102041
precision score:  0.058781869688385266
f1 score:  0.10993377483443707
accuracy score:  0.9764053228468101
ROC AUC: 0.963247971529636
```

```python
drop_list = ['V28','V27','V26','V25','V24','V23','V22','V20','V15','V13','V8']
X_train, X_test, y_train, y_test = split_data(df, drop_list)
y_pred, y_pred_prob = get_predictions(GaussianNB(), X_train, y_train, X_test)
print_scores(y_test,y_pred,y_pred_prob)
```

```
Index(['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V9', 'V10', 'V11', 'V12',
       'V14', 'V16', 'V17', 'V18', 'V19', 'V21', 'Class', 'Time_Hr',
       'scaled_Amount'],
      dtype='object')
train-set size:  227845
test-set size:  56962
fraud cases in test-set:  98
train-set confusion matrix:
 [[223967   3484]
 [    61    333]]
test-set confusion matrix:
 [[55935   929]
 [   12    86]]
recall score:  0.8775510204081632
precision score:  0.08472906403940887
f1 score:  0.15453728661275834
accuracy score:  0.9834802148800955
ROC AUC: 0.9622034097825962
```

```python
drop_list = ['Time_Hr','V28','V27','V26','V25','V24','V23','V22','V20','V15','V13','V8']
X_train, X_test, y_train, y_test = split_data(df, drop_list)
y_pred, y_pred_prob = get_predictions(GaussianNB(), X_train, y_train, X_test)
print_scores(y_test,y_pred,y_pred_prob)
```

```
Index(['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V9', 'V10', 'V11', 'V12',
       'V14', 'V16', 'V17', 'V18', 'V19', 'V21', 'Class', 'scaled_Amount'],
      dtype='object')
train-set size:  227845
test-set size:  56962
fraud cases in test-set:  98
train-set confusion matrix:
 [[223964   3487]
 [    60    334]]
test-set confusion matrix:
 [[55936   928]
 [   12    86]]
recall score:  0.8775510204081632
precision score:  0.08481262327416174
f1 score:  0.15467625899280577
accuracy score:  0.9834977704434535
ROC AUC: 0.9613612643988377
```

```python
drop_list = ['scaled_Amount','Time_Hr','V28','V27','V26','V25','V24','V23','V22','V20','V15','V13','V8'
X_train, X_test, y_train, y_test = split_data(df, drop_list)
y_pred, y_pred_prob = get_predictions(GaussianNB(), X_train, y_train, X_test)
print_scores(y_test,y_pred,y_pred_prob)
```

```
Index(['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V9', 'V10', 'V11', 'V12',
       'V14', 'V16', 'V17', 'V18', 'V19', 'V21', 'Class'],
      dtype='object')
train-set size:  227845
test-set size:  56962
fraud cases in test-set:  98
train-set confusion matrix:
 [[224025   3426]
 [    60    334]]
test-set confusion matrix:
 [[55954   910]
 [   12    86]]
recall score:  0.8775510204081632
precision score:  0.08634538152610442
f1 score:  0.15722120658135283
accuracy score:  0.9838137705838981
ROC AUC: 0.9611556179872063
```

```
# let us check recall score for logistic regression
# Case-LR-1
y_pred, y_pred_prob = get_predictions(LogisticRegression(C = 0.01)
                                      , X_train, y_train, X_test)
print_scores(y_test,y_pred,y_pred_prob)
```

```
train-set confusion matrix:
 [[227427     24]
 [   153    241]]
test-set confusion matrix:
 [[56851    13]
 [   39    59]]
recall score:  0.6020408163265306
precision score:  0.8194444444444444
f1 score:  0.6941176470588234
accuracy score:  0.9990871107053826
ROC AUC: 0.9725212608960297
```

Future Outlook

1. Continuous Model Improvement: The project's future involves a commitment to continuous improvement of the fraud detection models. This includes refining the existing machine learning algorithms and exploring new techniques to enhance the accuracy and efficiency of fraud detection.

2. Integration of Advanced Technologies: As technology evolves, the project should consider integrating advanced technologies such as deep learning to further enhance its ability to detect sophisticated fraud patterns that may evolve over time.

3. Real-time Monitoring and Detection: The future outlook involves transitioning towards real-time monitoring and detection. This would enable the system to identify and respond to potential fraudulent activities as they occur, minimizing the impact of fraudulent transactions.

4. Enhanced Data Security Measures: Given the sensitivity of financial data, future iterations of the project should prioritize and implement advanced data security measures to protect the integrity and confidentiality of the data used for training and testing the models.

5. User-Friendly Interfaces for Analysts: Developing user-friendly interfaces and dashboards for fraud analysts is key. This allows human experts to easily interpret model outputs, investigate flagged transactions, and provide necessary feedback for model improvement.

6. Scalability and Flexibility: The project will prioritize scalability and flexibility, ensuring that the fraud detection system can handle increasing volumes of transactions and adapt to changing data patterns without compromising performance.