Credit Card Fraud Detection

Anders Bjelland

Maaz bin alim

Usman Ali Khan

Project Overview

We tried to clean and evaluate the data set of credit card fraud. Then we used three machine learning models to accurately predict credit card fraud. First, we have tried to do it by Random Forest and Logistic. By using the different approach for the second time, we also tried doing our project by using LSTM machine learning model.

Evaluate ML accuracy of credit card fraud prediction using RandomForest and Logistic classifiers

In [1]:
```
#pip install imbalanced-learn
```

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

1 Import and Clean Data

In [3]:
```
df=pd.DataFrame(data = pd.read_csv('creditcard.csv'))
```

In [4]:
```
df.head()
```

Out[4]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817

5 rows × 31 columns

In [5]:
```
df.describe()
```

Out[5]:

	Time	V1	V2	V3	V4	V5
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05
mean	94813.859575	3.918649e-15	5.682686e-16	-8.761736e-15	2.811118e-15	-1.552103e-15

	Time	V1	V2	V3	V4	V5
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01

8 rows × 31 columns

In [6]:
```
# class by df_fraud and df_normal
dff=df.loc[df['Class']==1]
dfn=df.loc[df['Class']==0]
print('Fraction of fraid vs valid transactions', len(dff)/len(dfn))
```

```
Fraction of fraid vs valid transactions 0.0017304750013189597
```

In [7]:
```
y0=df['Class']
X0=df.drop(['Class'],axis=1)
```

In [8]:
```
np.bincount(df['Class']) # Valid vs Fraud transactions
```

Out[8]:
```
array([284315,    492], dtype=int64)
```

In [9]:
```
from matplotlib import gridspec
# distribution of anomalous features
features = df.iloc[:,0:30].columns

plt.figure(figsize=(12,28*4))
gs = gridspec.GridSpec(30, 1)
for i, j in enumerate(df[features]):
    ax = plt.subplot(gs[i])
    sns.distplot(df[j][df.Class == 1], bins=50)
    sns.distplot(df[j][df.Class == 0], bins=50)
    ax.set_xlabel('')
    ax.set_title('histogram of features: ' + str(j))
plt.show()
```

```
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
```

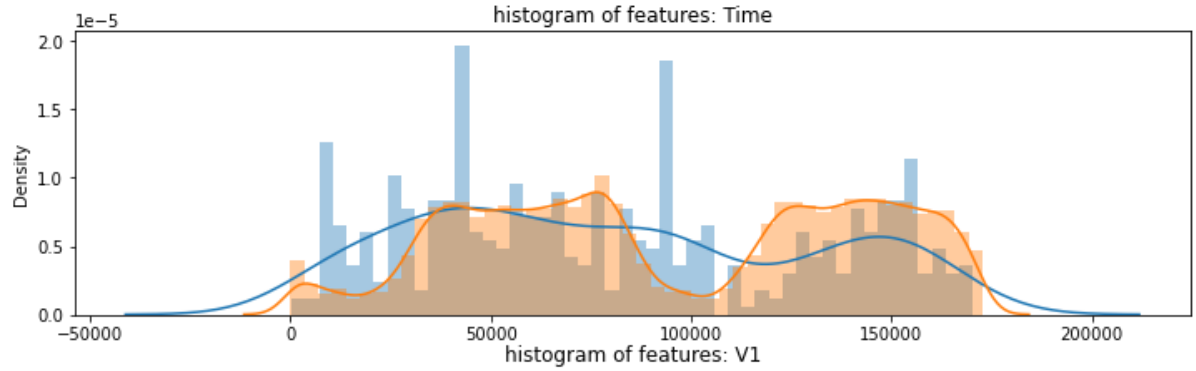
```
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
```

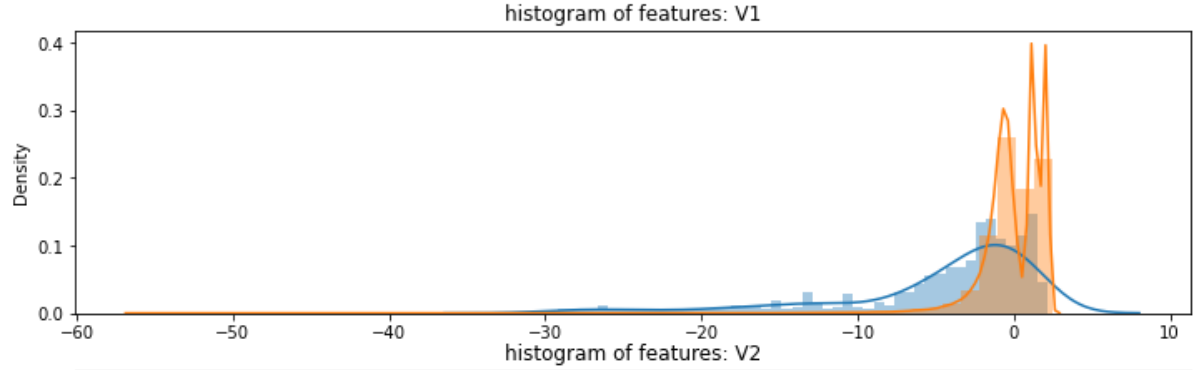
```
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
```

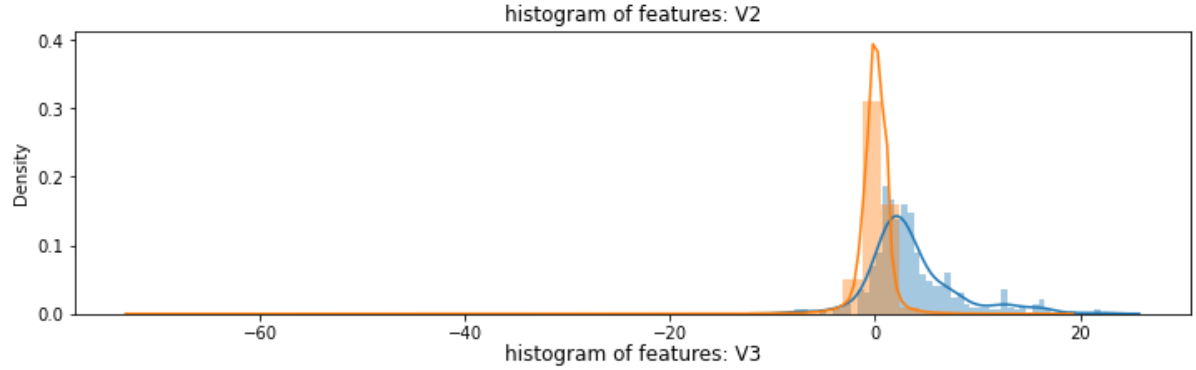
```
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
```

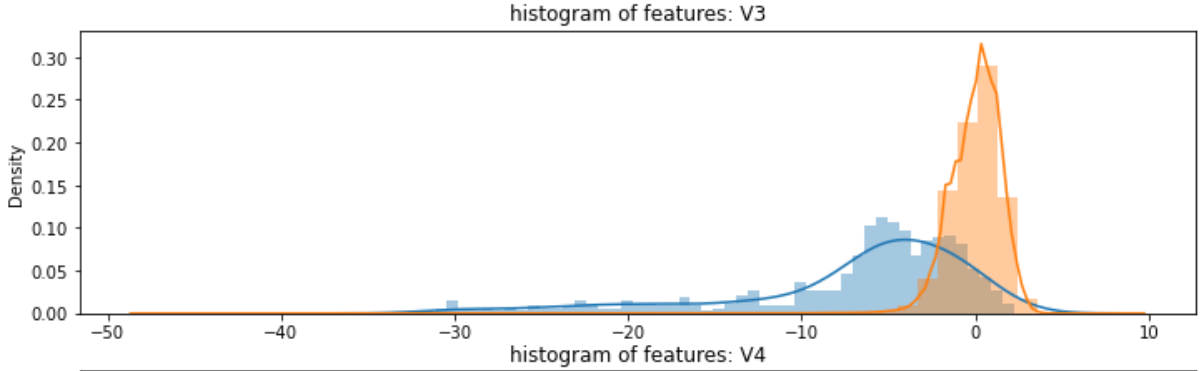
```
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

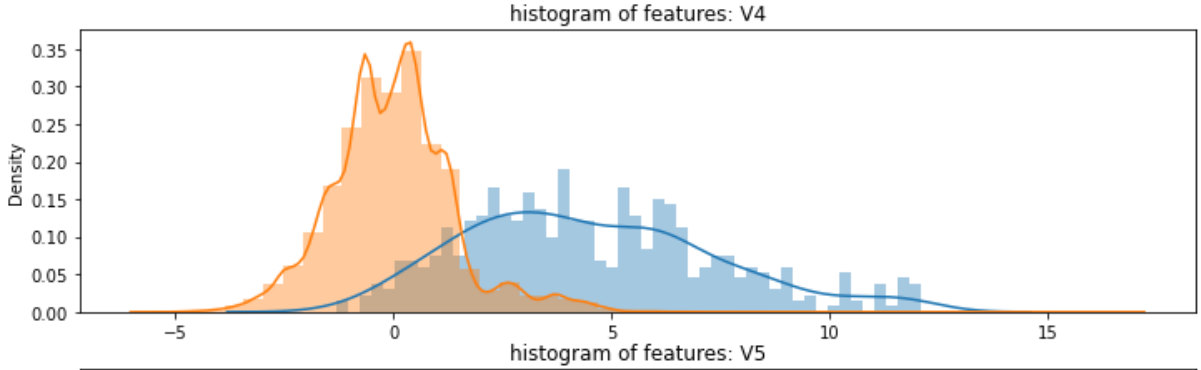
```
C:\Users\maaz2\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

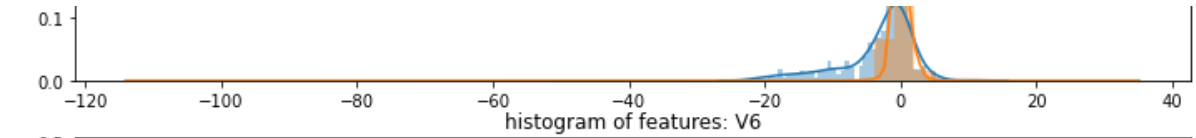




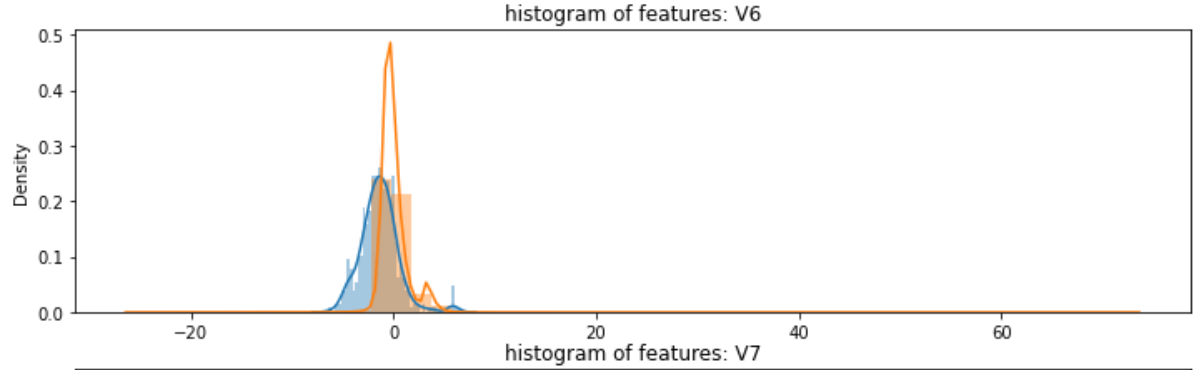




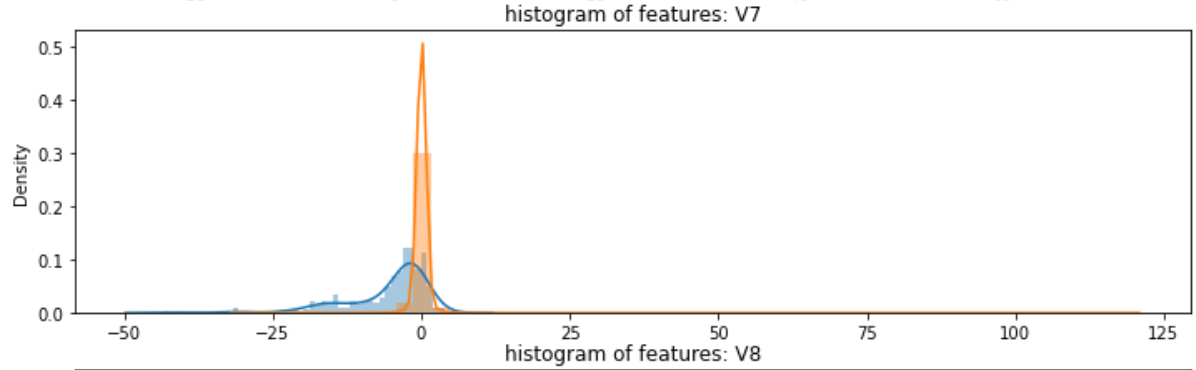




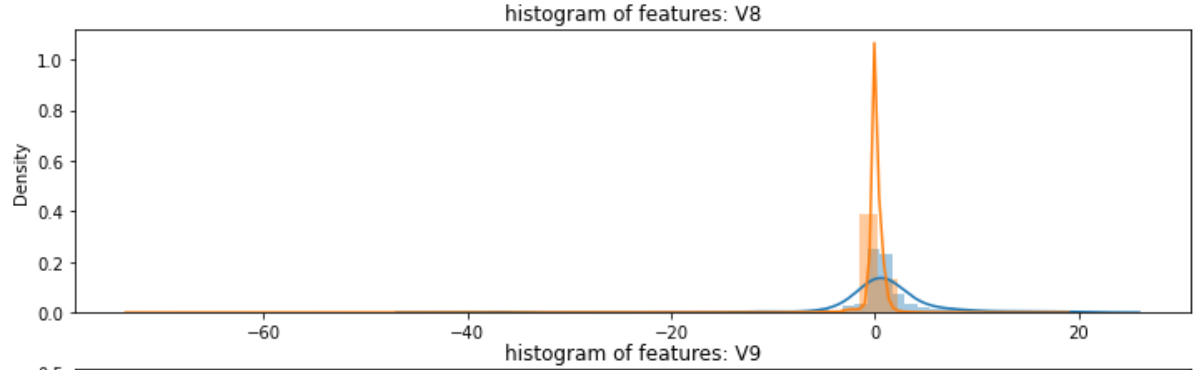
histogram of features: V6



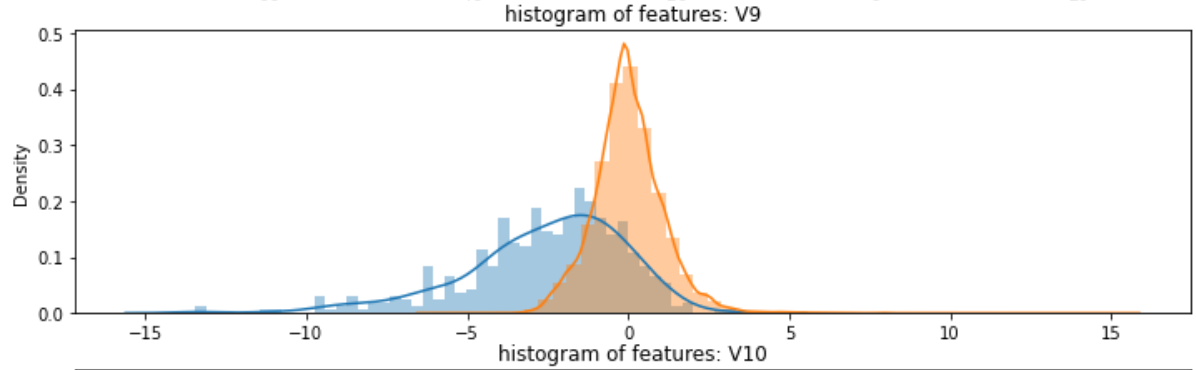
histogram of features: V7



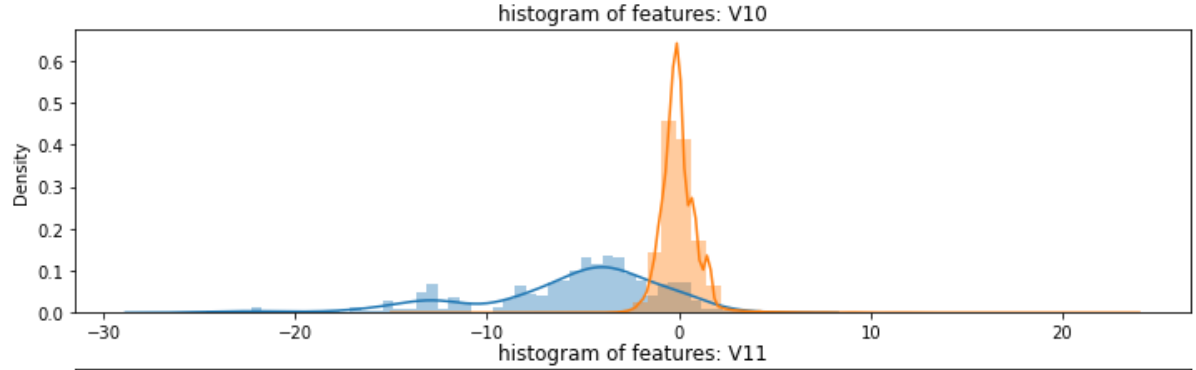
histogram of features: V8



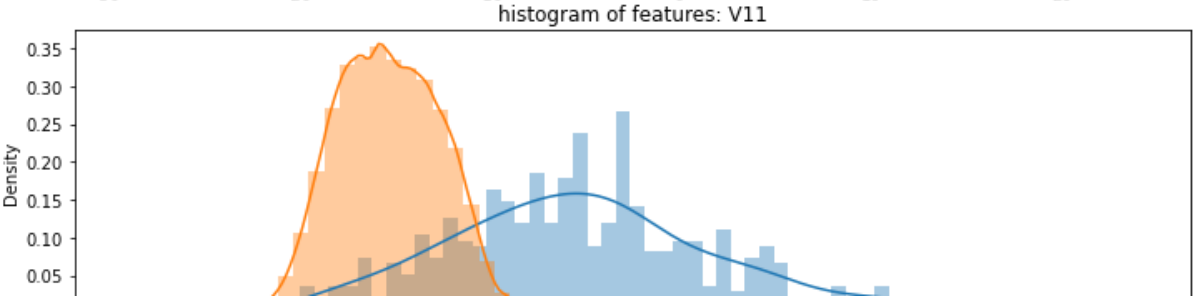
histogram of features: V9

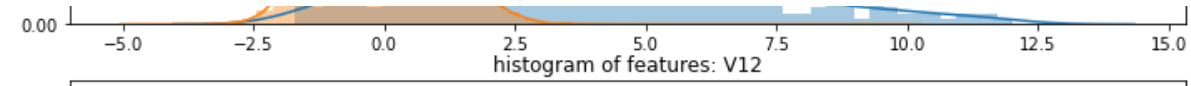


histogram of features: V10

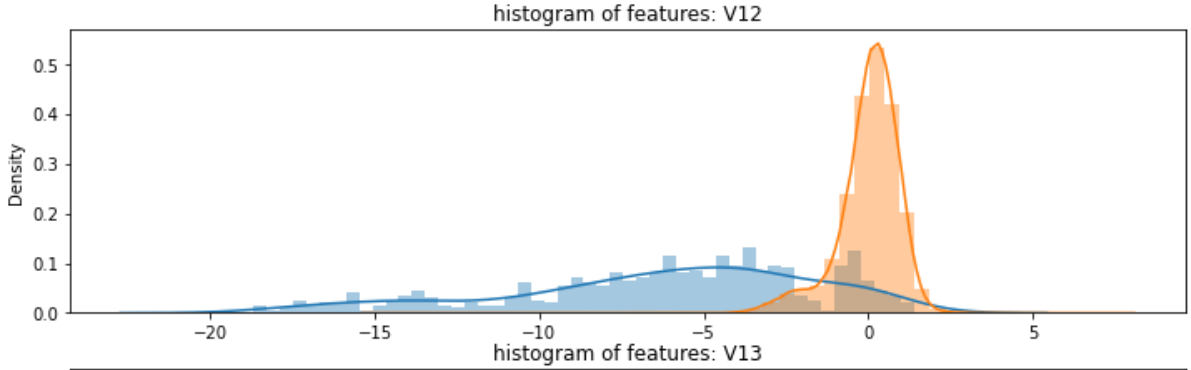


histogram of features: V11

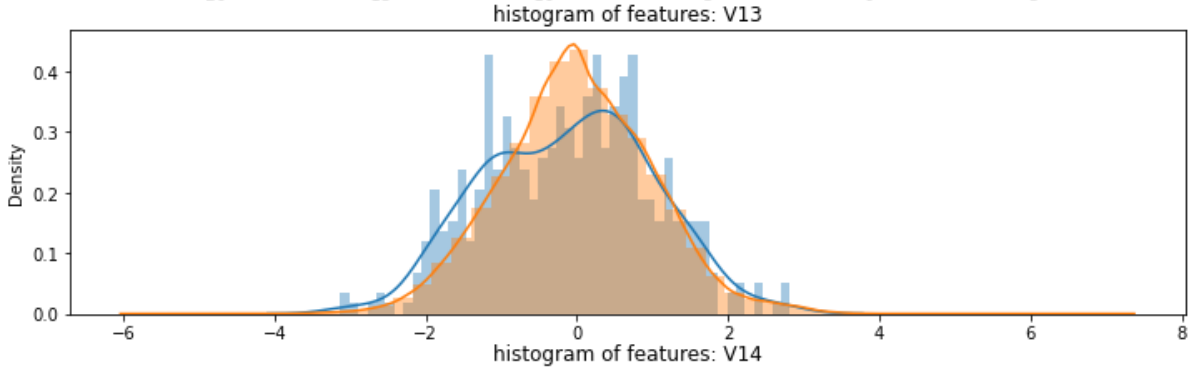




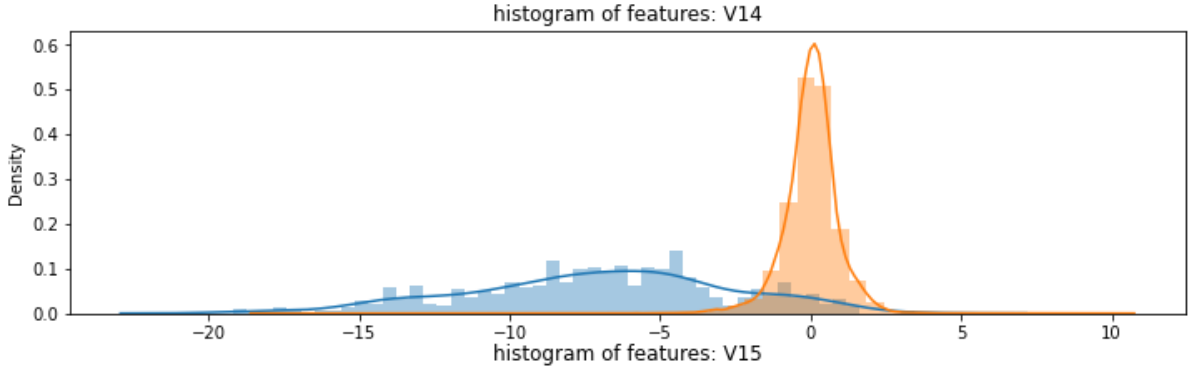
histogram of features: V12



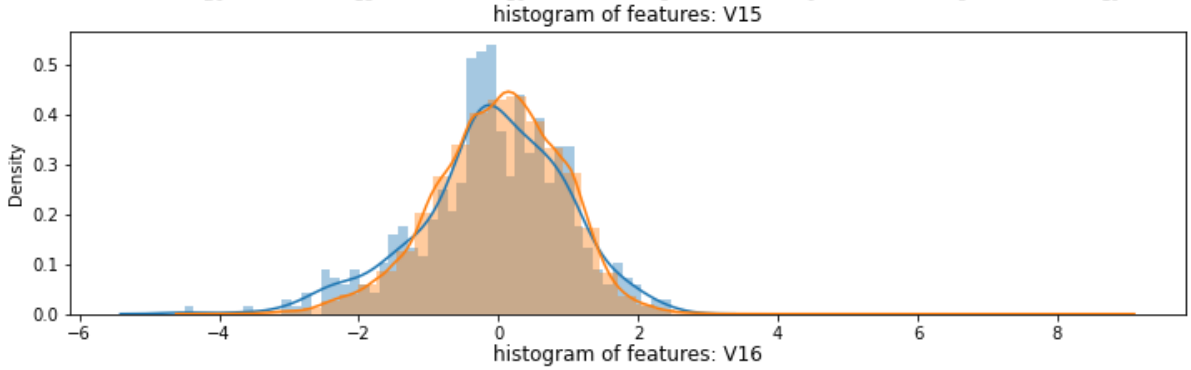
histogram of features: V13



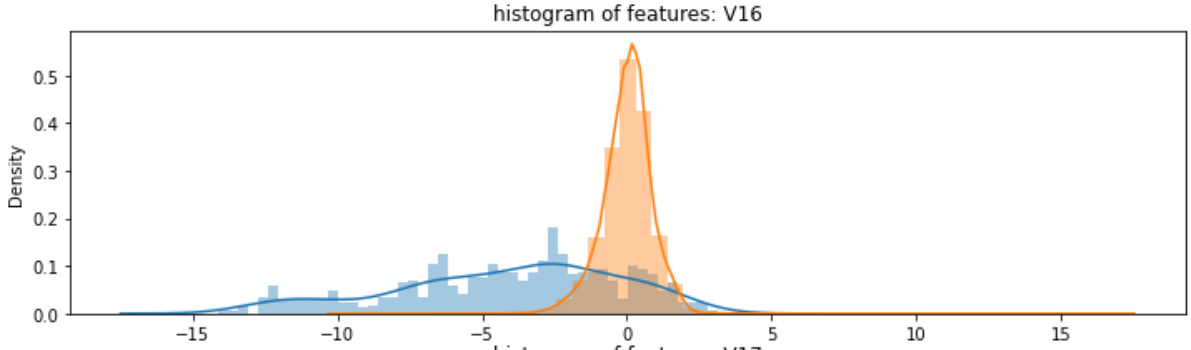
histogram of features: V14



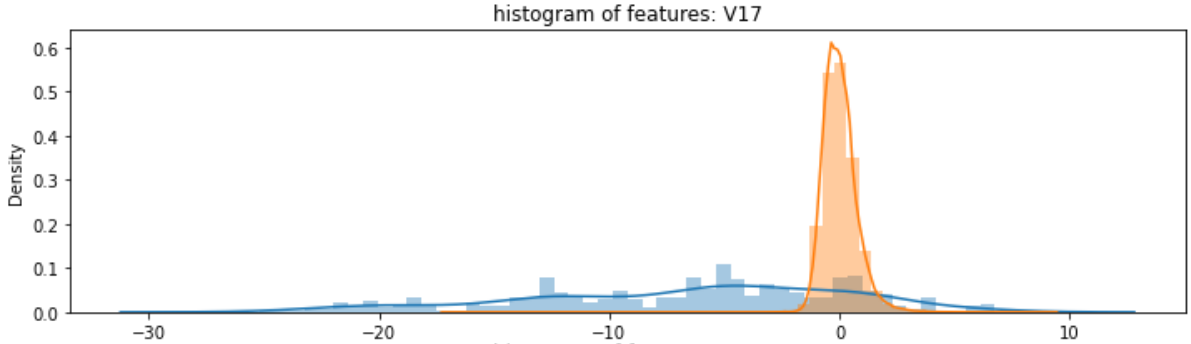
histogram of features: V15

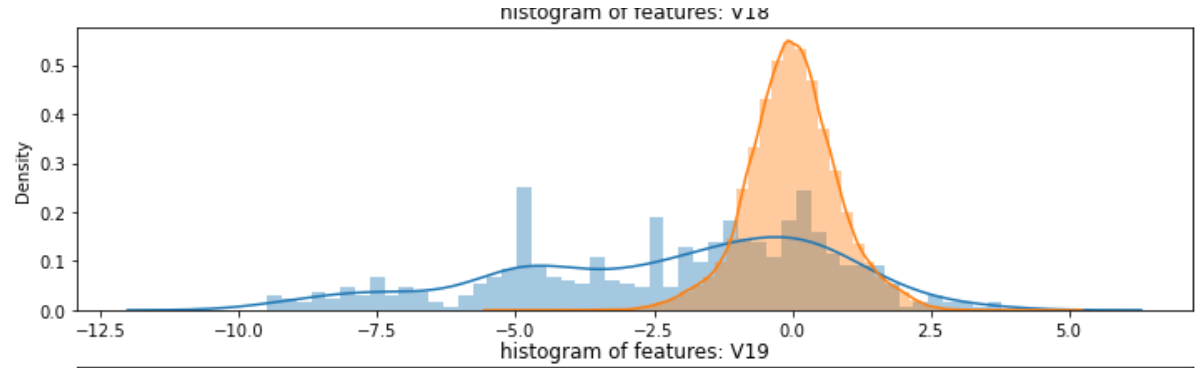


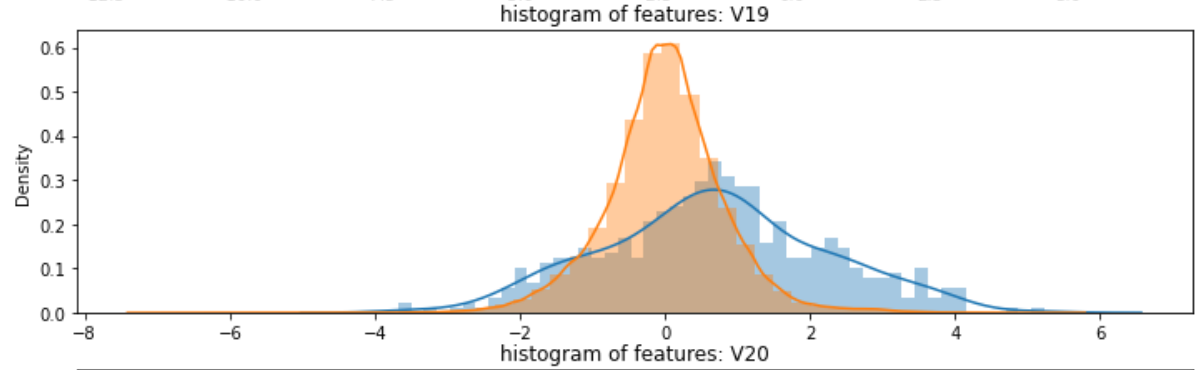
histogram of features: V16

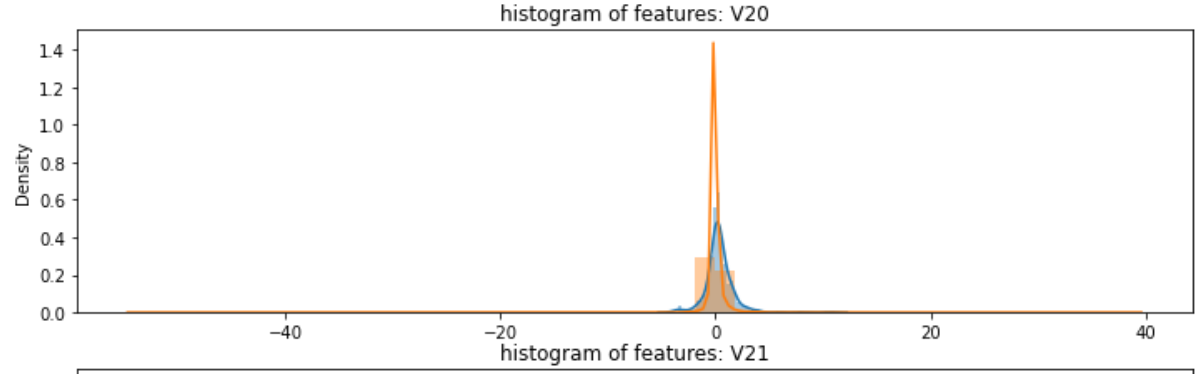


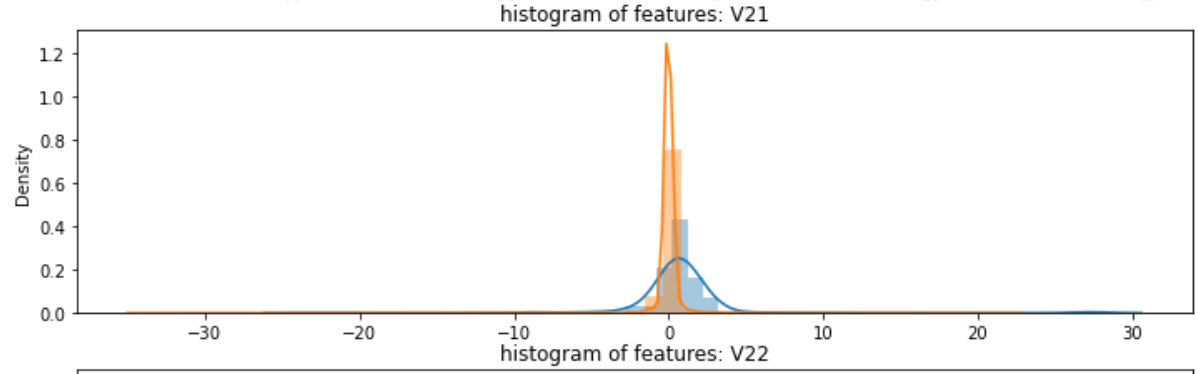
histogram of features: V17

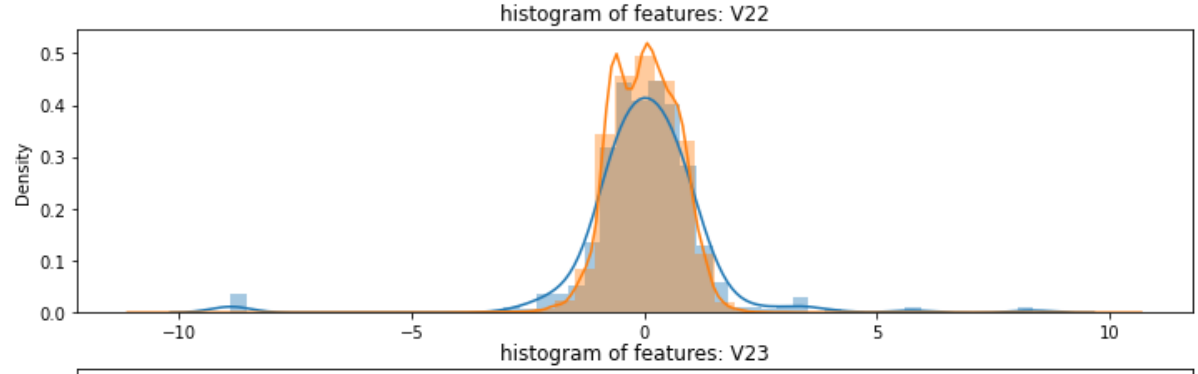


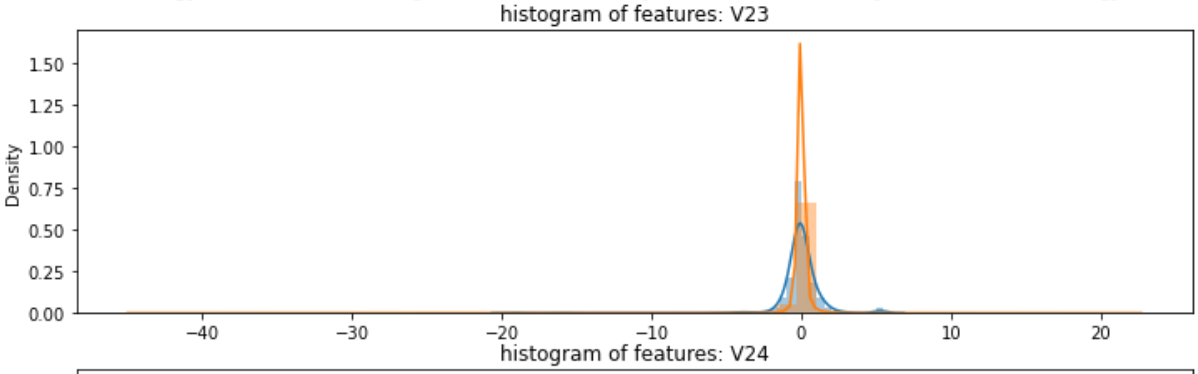

histogram of features: V18
Density

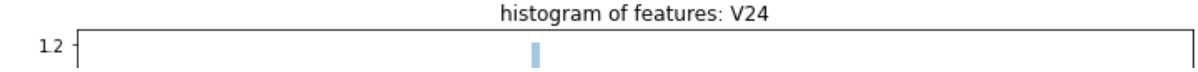

histogram of features: V19
Density

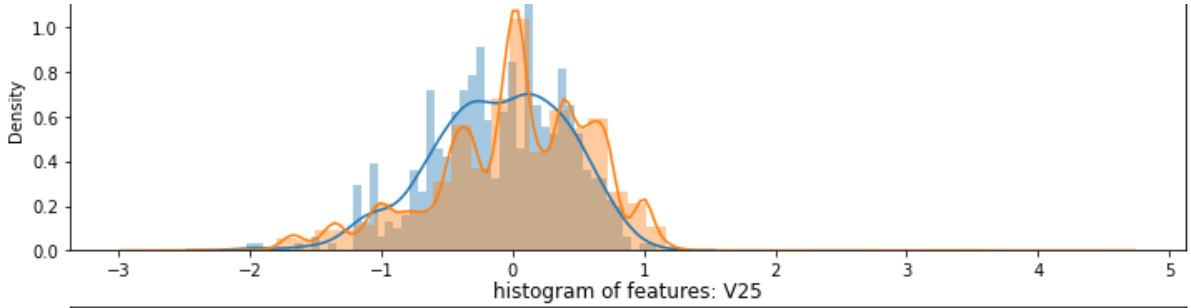

histogram of features: V20
Density

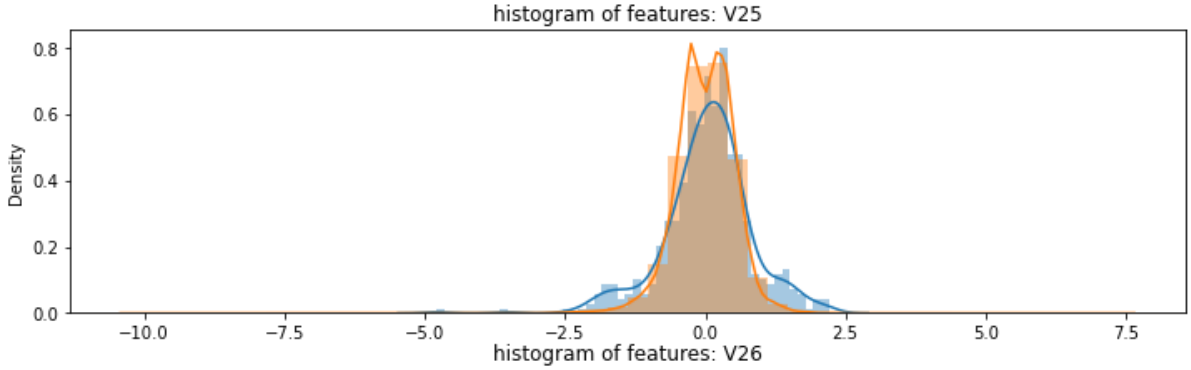

histogram of features: V21
Density

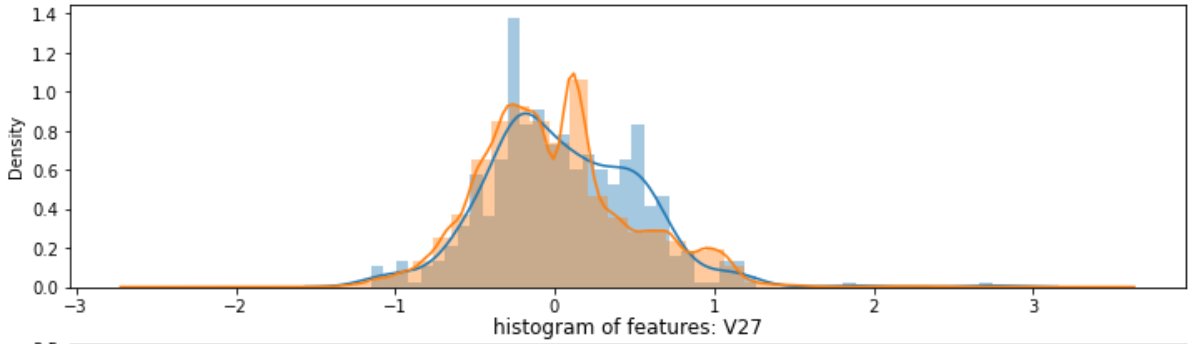

histogram of features: V22
Density

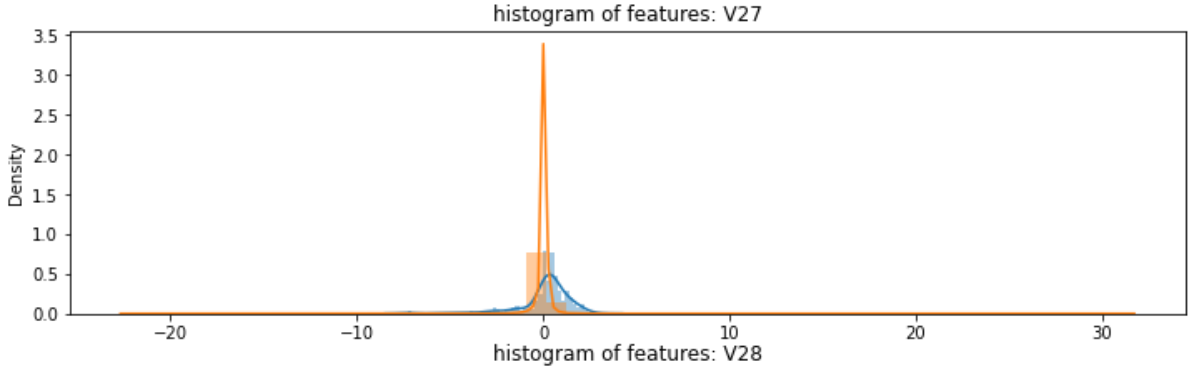

histogram of features: V23
Density

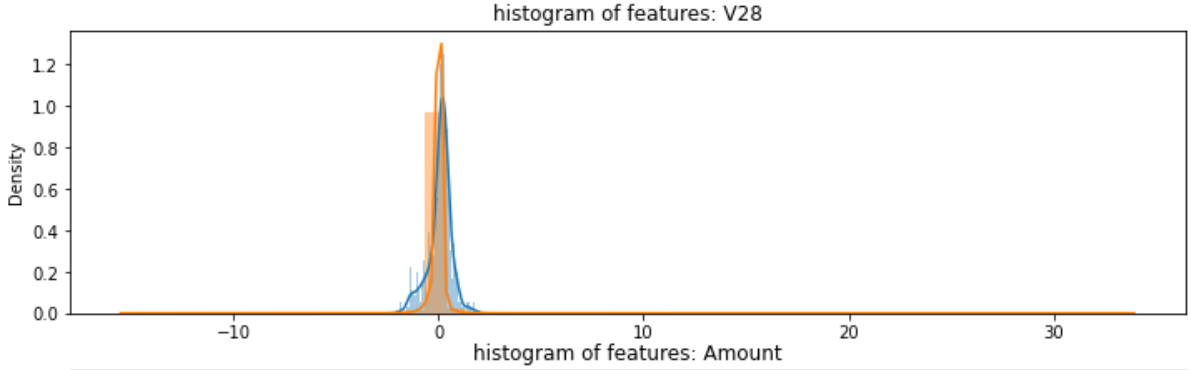

histogram of features: V24

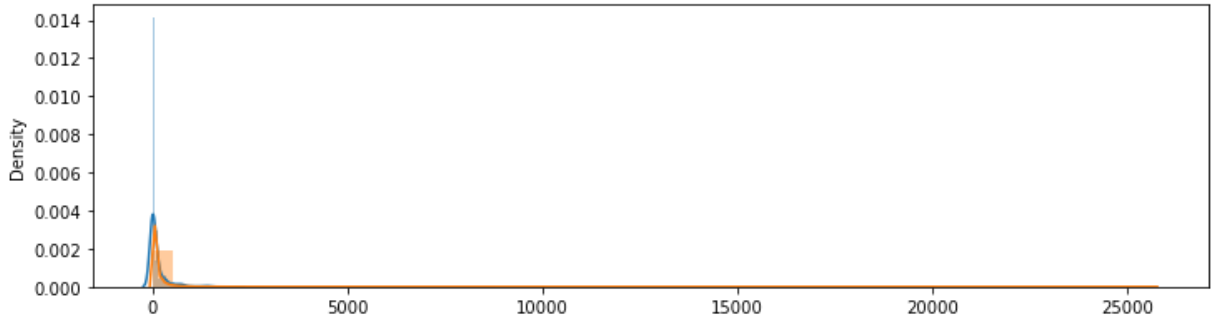












In [10]:
```
#From plots, idenfity variables where there is significant difference between the hi
#i.e the variable values are more seperable
```

```
dfs=df[['V2','V3','V4','V9','V10','V11','V12','V14','V17','V19','Class']]
#dfs=df.drop(['Amount'], axis=1
#use line above to include all variables accept amount, does not gain much accuracy
dfs.shape
```

Out[10]: (284807, 11)

Removes duplicate values (And Null values)

In [11]:
```
dfs1 = dfs.drop_duplicates(keep='first').dropna()

print(len(dfs)-len(dfs1), 'rows were removed')

dffs=dfs1.loc[dfs1['Class']==1]
dfns=dfs1.loc[dfs1['Class']==0]

print('There are:',len(dffs),'fraud transactions, and:', len(dfns),'legit transactio
```

```
9144 rows were removed
There are: 473 fraud transactions, and: 275190 legit transactions after clearning
```

Identify and remove outliers

In [12]:
```
dffs.describe()
```

Out[12]:

	V2	V3	V4	V9	V10	V11	V12	V
count	473.000000	473.000000	473.000000	473.000000	473.000000	473.000000	473.000000	473.0000
mean	3.405965	-6.729599	4.472591	-2.522124	-5.453274	3.716347	-6.103254	-6.8359
std	4.122500	6.909647	2.871523	2.465047	4.706451	2.672817	4.582331	4.2532
min	-8.402154	-31.103685	-1.313275	-13.434066	-24.588262	-1.702228	-18.683715	-19.2143
25%	1.145381	-7.926507	2.288644	-3.796760	-7.297803	1.928568	-8.601648	-9.5051
50%	2.617105	-4.875397	4.100098	-2.099049	-4.466284	3.525726	-5.437354	-6.5905
75%	4.571743	-2.171454	6.290918	-0.788388	-2.447469	5.224124	-2.824946	-4.2524
max	22.057729	2.250210	12.114672	3.353525	4.031435	12.018913	1.375941	3.4424

In [13]:
```
dfns.describe() #we see some values are significantly higher/lower that mean +-2x St
```

Out[13]:

	V2	V3	V4	V9	V10	V11
count	275190.000000	275190.000000	275190.000000	275190.000000	275190.000000	275190.000000
mean	-0.008288	0.037131	-0.012054	-0.008050	0.012492	-0.013574
std	1.653899	1.454690	1.408379	1.091353	1.046136	1.003609
min	-72.715728	-48.325589	-5.683171	-6.290730	-14.741096	-4.797473
25%	-0.615265	-0.836237	-0.864528	-0.657122	-0.536756	-0.773865
50%	0.068826	0.203635	-0.037813	-0.063328	-0.090628	-0.041255
75%	0.816240	1.050013	0.748570	0.594172	0.472001	0.731389
max	18.902453	9.382558	16.875344	15.594995	23.745136	10.002190

In [14]:
```
#Define statistical quartiles

q1ns=dfns.quantile(0.25)
q3ns=dfns.quantile(0.75)
iqrns=q3ns-q1ns

q1fs=dffs.quantile(0.25)
q3fs=dffs.quantile(0.75)
iqrfs=q3fs-q1fs

iqrfs.head()
print( q1ns,q3ns, iqrns, q1fs, q3fs, iqrfs)
```

```
V2      -0.615265
V3      -0.836237
V4      -0.864528
V9      -0.657122
V10     -0.536756
V11     -0.773865
V12     -0.410829
V14     -0.422985
V17     -0.481199
V19     -0.464476
Class    0.000000
Name: 0.25, dtype: float64 V2       0.816240
V3       1.050013
V4       0.748570
V9       0.594172
V10      0.472001
V11      0.731389
V12      0.615362
V14      0.492951
V17      0.401660
V19      0.464522
Class    0.000000
Name: 0.75, dtype: float64 V2       1.431505
V3       1.886250
V4       1.613098
V9       1.251294
V10      1.008757
V11      1.505254
V12      1.026191
V14      0.915936
V17      0.882859
V19      0.928998
Class    0.000000
dtype: float64 V2        1.145381
V3       -7.926507
V4        2.288644
V9       -3.796760
V10      -7.297803
V11       1.928568
V12      -8.601648
V14      -9.505141
V17     -11.588544
V19      -0.300931
Class     1.000000
Name: 0.25, dtype: float64 V2       4.571743
V3      -2.171454
V4       6.290918
V9      -0.788388
V10     -2.447469
V11      5.224124
V12     -2.824946
V14     -4.252466
V17     -1.129056
V19      1.661029
```

```
Class    1.000000
Name: 0.75, dtype: float64 V2        3.426362
V3         5.755052
V4         4.002274
V9         3.008372
V10        4.850334
V11        3.295556
V12        5.776703
V14        5.252675
V17       10.459488
V19        1.961960
Class      0.000000
dtype: float64
```

In [15]:
```
#Drop outlier data rows
cols=list(dfs.columns)
cols.pop()
cols
for i in cols:

    dfns2=dfns.loc[dfns[i] < (q3ns[i] + 1.5*iqrns[i])]
    dffs2=dffs.loc[dffs[i] < (q3ns[i] + 2*iqrns[i])]

    dfns2=dfns2.loc[dfns2[i] > (q1ns[i] - 1.5*iqrns[i])]
    dffs2=dffs2.loc[dffs2[i] > (q1fs[i] - 2*iqrfs[i])]

    #dfns[i][dfns[i]> q3ns[i] + 1.5*iqrns[i] ]= np.nan  #dfns[i].mean()
    #dffs[i][dffs[i]> q3fs[i] + 1.5*iqrfs[i] ]= np.nan
    #dfns[i][dfns[i]< q1ns[i] - 1.5*iqrns[i] ]= np.nan
    #dffs[i][dffs[i]< q1fs[i] - 1.5*iqrfs[i] ]= np.nan
```

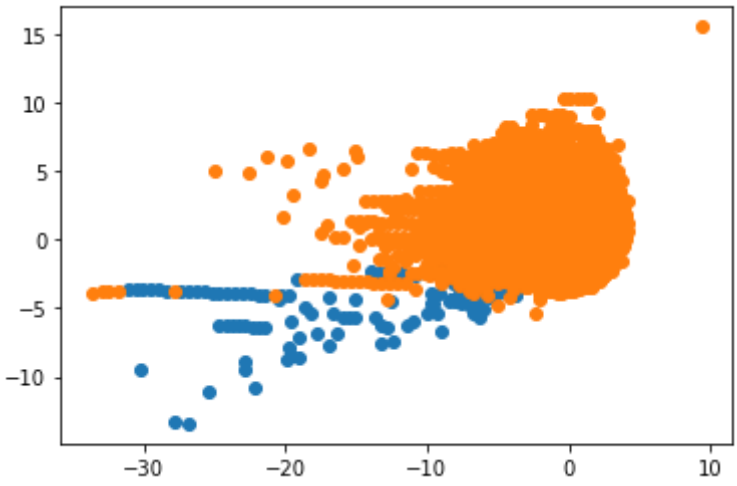
In [16]:
```
print ( 'Normal Transaction rows dropped =', len(dfns)-len(dfns2), 'Fraud transactio

iqrns['V2']
```

```
Normal Transaction rows dropped = 9373 Fraud transaction rows dropped = 74
```

Out[16]:
```
1.4315048018686738
```

In [17]:
```
# Scatterplot of variable comparison
plt.scatter(dffs2['V3'],dffs2['V9'] )
plt.scatter(dfns2['V3'],dfns2['V9'] )
dfns2.shape
```

Out[17]:
```
(265817, 11)
```



In [18]:
```
dfns2.describe()
```

Out[18]:

	V2	V3	V4	V9	V10	V11

	V2	V3	V4	V9	V10	V11
count	265817.000000	265817.000000	265817.000000	265817.000000	265817.000000	265817.000000
mean	0.001991	0.025542	-0.037175	0.004413	0.007875	-0.015771
std	1.560612	1.438668	1.378939	1.085483	1.036736	1.000961
min	-48.060856	-33.680984	-5.683171	-5.310922	-11.208723	-4.049895
25%	-0.616462	-0.855114	-0.869930	-0.640114	-0.538174	-0.777169
50%	0.062576	0.188300	-0.054528	-0.052607	-0.096175	-0.042926
75%	0.813524	1.033254	0.712439	0.603909	0.457936	0.730007
max	18.902453	9.382558	13.143668	15.594995	23.745136	10.002190

In [19]:
```
#combine as cleaned dataset
df_cleaned=dfns2.append(dffs2, ignore_index=True)
df_cleaned
```

Out[19]:

	V2	V3	V4	V9	V10	V11	V12	V14	
0	-0.072781	2.536347	1.378155	0.363787	0.090794	-0.551600	-0.617801	-0.311169	0.207
1	0.266151	0.166480	0.448154	-0.255425	-0.166974	1.612727	1.065235	-0.143772	-0.114
2	-0.185226	1.792993	-0.863291	-1.387024	-0.054952	-0.226487	0.178228	-0.287924	-0.684
3	0.877737	1.548718	0.403034	0.817739	0.753074	-0.822843	0.538196	-1.119670	-0.237
4	0.960523	1.141109	-0.168252	-0.568671	-0.371407	1.341262	0.359894	-0.137134	-0.058
...	...	...	...	...	...	...	...	...	
266211	1.125653	-4.518331	1.749293	-2.064945	-5.587794	2.115795	-5.417424	-6.665177	-4.570
266212	1.289381	-5.004247	1.411850	-1.127396	-3.232153	2.858466	-3.096915	-5.210141	-3.267
266213	1.126366	-2.213700	0.468308	-0.652250	-3.463891	1.794969	-2.775022	-4.057162	-5.035
266214	0.585864	-5.399730	1.817092	-1.632333	-5.245984	1.933520	-5.030465	-6.416628	-4.614
266215	0.158476	-2.583441	0.408670	0.577829	-0.888722	0.491140	0.728903	-1.948883	0.903

266216 rows × 11 columns

In [20]:
```
X=df_cleaned.drop(['Class'], axis=1)
y=df_cleaned['Class']
np.bincount(y)
```

Out[20]:
```
array([265817,    399], dtype=int64)
```

2 Initial evaluation with ML models

In [21]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=1
```

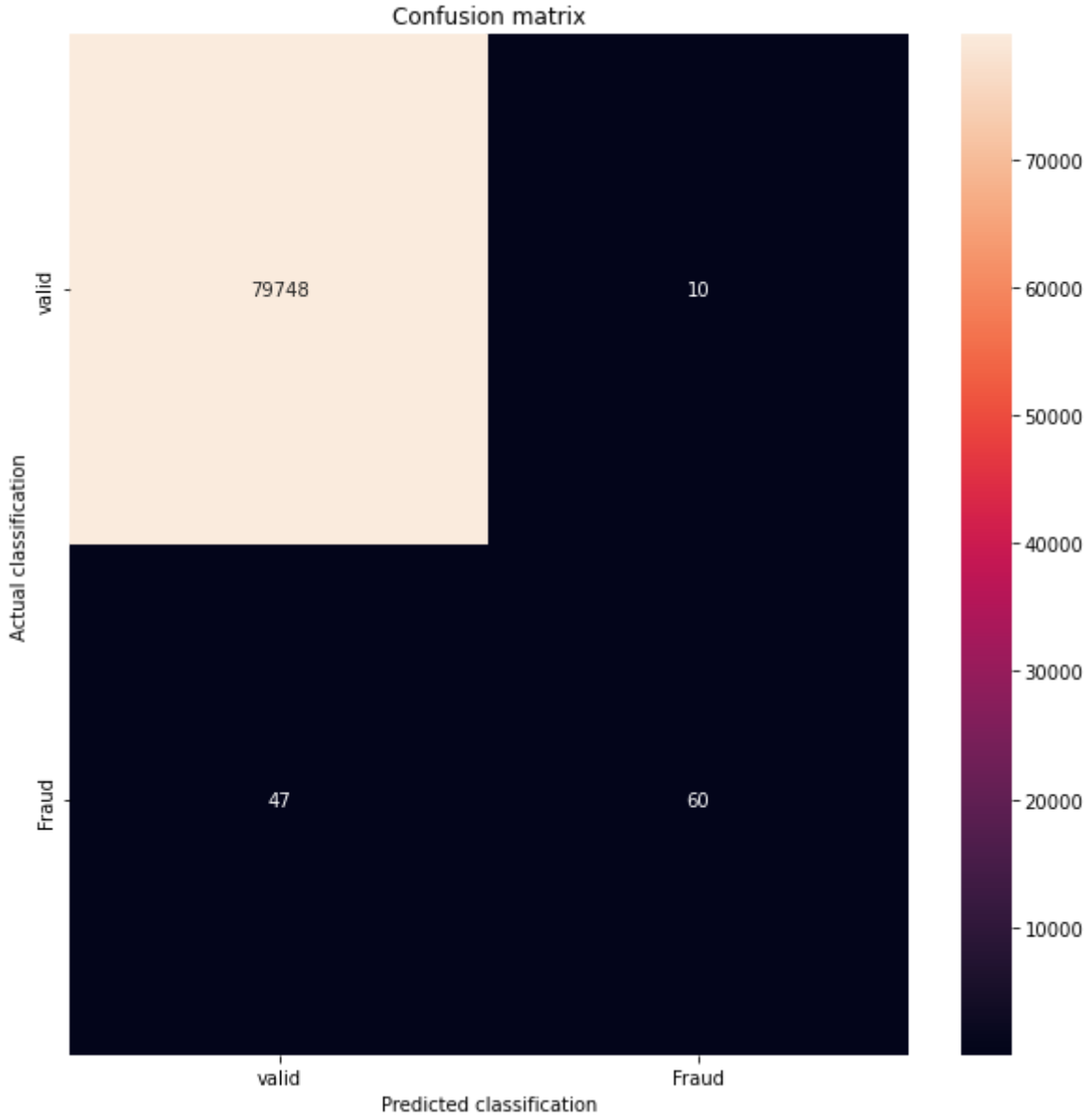
In [22]:
```
from sklearn.linear_model import LogisticRegression
```

```
m1=LogisticRegression(random_state=0)
#fit data
m1.fit(X, y)
y_pred=m1.predict(x_test)


#print(classification_report(Y_test, Y_pred1))
```

In [23]:
```
#from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
#print(accuracy_score(y_test,y_pred))
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test,y_pred))
#pd.crosstab(y_test,y_pred)
```

```
[[79748    10]
 [   47    60]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     79758
           1       0.86      0.56      0.68       107

    accuracy                           1.00     79865
   macro avg       0.93      0.78      0.84     79865
weighted avg       1.00      1.00      1.00     79865
```

In [24]:
```
## plotting confusion matrix with heatmap

LABELS = ['valid', 'Fraud']
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(10, 10))
sns.heatmap(conf_matrix, xticklabels=LABELS,
            yticklabels=LABELS, annot=True, fmt="d");
plt.title("Confusion matrix")
plt.ylabel('Actual classification')
plt.xlabel('Predicted classification')
plt.show()
```



3 Repeat with SMOTE oversampling technique

In [25]:
```
from sklearn.metrics import confusion_matrix
from imblearn.over_sampling import SMOTE
```

In [26]:
```
oversample = SMOTE()
# IF USING AN OLDER VERSION OF IMBLEARN USE THE FOLLOWING:
#X_res, y_res = oversample.fit_sample(x_train, y_train)
X_res, y_res = oversample.fit_resample(x_train, y_train)

np.bincount(y_res)
```

Out[26]:
```
array([186059, 186059], dtype=int64)
```

In [27]:
```
m1_smote=LogisticRegression(random_state=0)

m1_smote.fit(X_res, y_res)
y_pred_smote=m1_smote.predict(x_test)
```

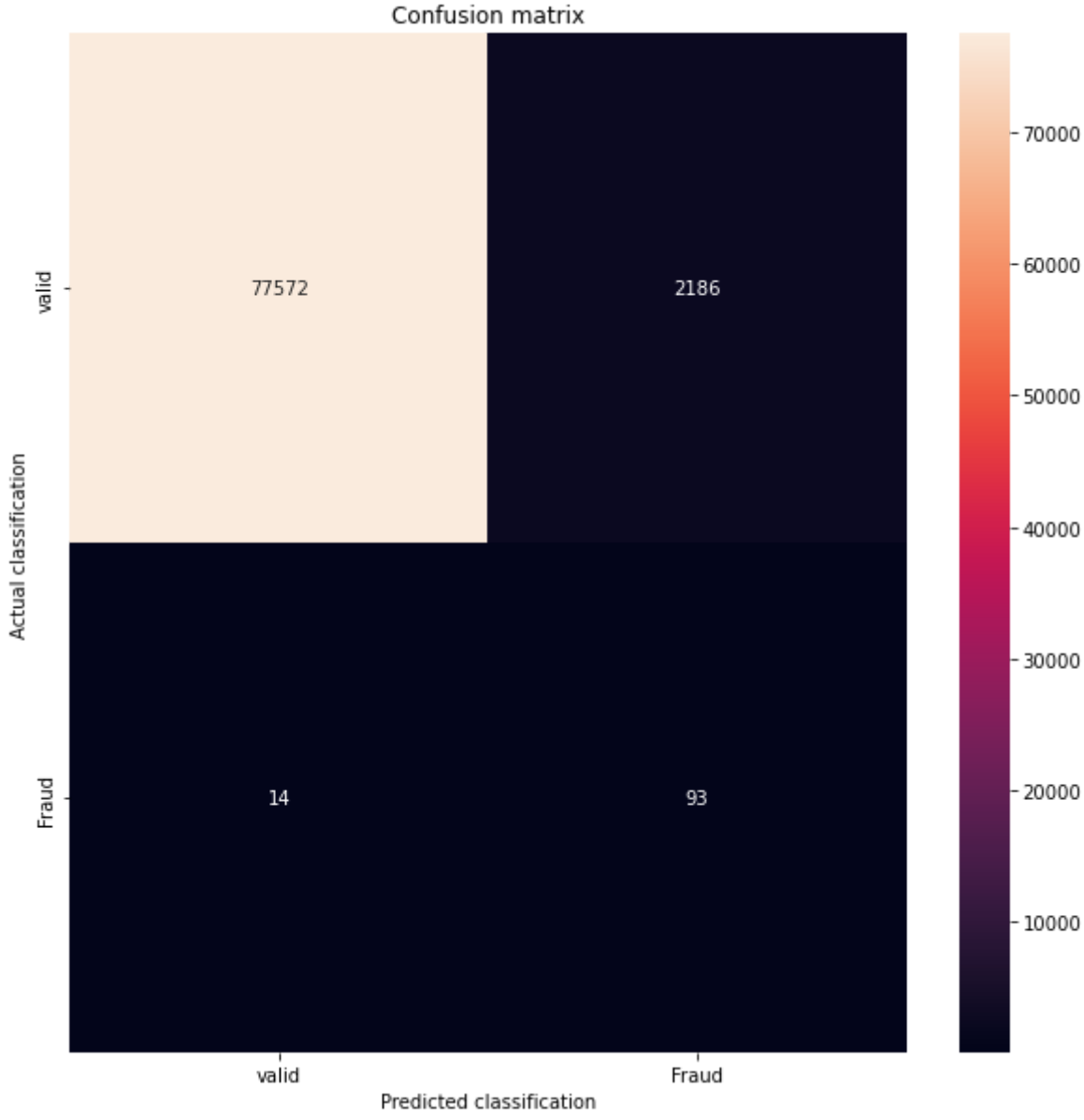
In [28]:
```
print(confusion_matrix(y_test, y_pred_smote))
print(classification_report(y_test,y_pred_smote))
```

```
[[77572  2186]
 [   14    93]]
```

```
                precision    recall  f1-score   support

           0       1.00      0.97      0.99     79758
           1       0.04      0.87      0.08       107

    accuracy                           0.97     79865
   macro avg       0.52      0.92      0.53     79865
weighted avg       1.00      0.97      0.98     79865
```

In [29]:

```
## plotting confusion matrix with heatmap

LABELS = ['valid', 'Fraud']
conf_matrix = confusion_matrix(y_test, y_pred_smote)
plt.figure(figsize=(10, 10))
sns.heatmap(conf_matrix, xticklabels=LABELS,
            yticklabels=LABELS, annot=True, fmt="d");
plt.title("Confusion matrix")
plt.ylabel('Actual classification')
plt.xlabel('Predicted classification')
plt.show()
```



Random Forest Classifier

In [30]:

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(max_depth=3, random_state=0)
```

```
rfc.fit(X_res, y_res)
#RandomForestClassifier(...)
```

Out[30]:
```
RandomForestClassifier(max_depth=3, random_state=0)
```

In [31]:
```
y_pred_rfc_smote=rfc.predict(x_test)
```

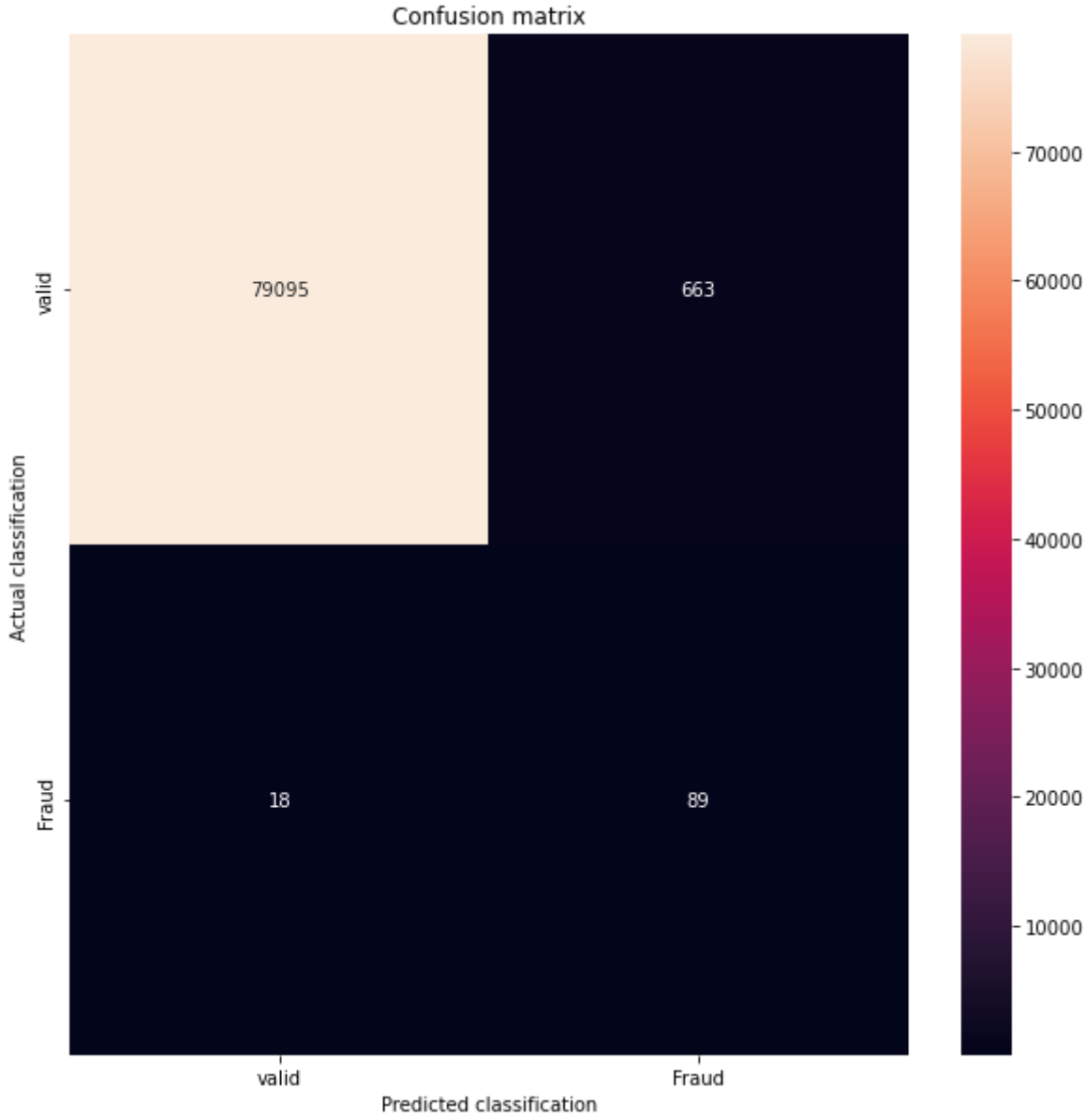
In [32]:
```
np.bincount(y_pred_rfc_smote)
```

Out[32]:
```
array([79113,   752], dtype=int64)
```

In [33]:
```
print(confusion_matrix(y_test, y_pred_rfc_smote))
print(classification_report(y_test,y_pred_rfc_smote))
#pd.crosstab(y_test,y_pred)
```

```
[[79095   663]
 [   18    89]]
              precision    recall  f1-score   support

           0       1.00      0.99      1.00     79758
           1       0.12      0.83      0.21       107

    accuracy                           0.99     79865
   macro avg       0.56      0.91      0.60     79865
weighted avg       1.00      0.99      0.99     79865
```

In [34]:
```
LABELS = ['valid', 'Fraud']
conf_matrix = confusion_matrix(y_test, y_pred_rfc_smote)
plt.figure(figsize=(10, 10))
sns.heatmap(conf_matrix, xticklabels=LABELS,
            yticklabels=LABELS, annot=True, fmt="d");
plt.title("Confusion matrix")
plt.ylabel('Actual classification')
plt.xlabel('Predicted classification')
plt.show()
```



5 ML Accuracy scores

In [35]:
```
from sklearn.model_selection import StratifiedKFold
from imblearn.over_sampling import SMOTE
fold=StratifiedKFold(n_splits=4)
```

In [36]:
```
def model_scores (model, x_train, x_test, y_train, y_test):
    model.fit(x_train, y_train)
    y_pred=model.predict(x_test)
    cm=confusion_matrix(y_test, y_pred)
    tp=cm[1,1]
    fp=cm[1,0]

    fn=cm[0,1]
    tn=cm[0,0]

    pres_pos=tp/(tp+fp)
    pres_neg = tn/(tn+fn)

    recall_pos=tp/(tp+fn)
    recall_neg=tn/(fp+tn)

    accuracy= (tp+tn)/(tp+fp+fn+tn)
```

```
    print(cm)
    return pres_pos, recall_pos , accuracy , str(model)
```

In [37]:
```
#RandomForest

stats_arr=[]
valid_arr=[]
fraud_arr=[]
fold_number=[]
n=0
for train_idx, test_idx in fold.split(X,y):

    X_train, X_val     = X.iloc[train_idx], X.iloc[test_idx]
    y_train, y_val     = y.iloc[train_idx], y.iloc[test_idx]

    m=model_scores(RandomForestClassifier(n_estimators=30), X_train, X_val, y_train,

    print(m)

    n=n+1

    stats_arr.append(m)
    valid_arr.append(m[1]) #Recall
    fraud_arr.append(m[0]) #Precision
    fold_number.append(n)
```

```
[[66421    34]
 [   10    89]]
(0.898989898989899, 0.7235772357723578, 0.999338882711783, 'RandomForestClassifier(n
_estimators=30)')
[[66452     2]
 [   35    65]]
(0.65, 0.9701492537313433, 0.999444060462181, 'RandomForestClassifier(n_estimators=3
0)')
[[66446     8]
 [   20    80]]
(0.8, 0.9090909090909091, 0.9995792889984073, 'RandomForestClassifier(n_estimators=3
0)')
[[66454     0]
 [   33    67]]
(0.67, 1.0, 0.9995041620338372, 'RandomForestClassifier(n_estimators=30)')
```

In [38]:
```
fraud_ave_prec=sum(fraud_arr)/len(fraud_arr)
valid_ave_prec=sum(valid_arr)/len(valid_arr)

data= {'Fraud Precsision': fraud_arr, 'Fraud Recall': valid_arr }
df_results=pd.DataFrame(data=data, index= fold_number )
RF_mean=df_results.mean()
print('Random Forest Classifier with imbalanced dataset')
print(df_results)
print(RF_mean, 'Average')

#print('Valid Average Precision:',valid_ave_prec, 'Fraud Average Precision:', fraud_
```

```
Random Forest Classifier with imbalanced dataset
   Fraud Precsision  Fraud Recall
1           0.89899      0.723577
2           0.65000      0.970149
3           0.80000      0.909091
4           0.67000      1.000000
Fraud Precsision    0.754747
```

```
Fraud Recall        0.900704
dtype: float64 Average
```

In [39]:
```
#Logistic Regression

stats_arr=[]
valid_arr=[]
fraud_arr=[]
fold_number=[]
n=0
for train_idx, test_idx in fold.split(X,y):

    X_train, X_val     = X.iloc[train_idx], X.iloc[test_idx]
    y_train, y_val     = y.iloc[train_idx], y.iloc[test_idx]

    m=model_scores(LogisticRegression(), X_train, X_val, y_train, y_val)

    print(m)

    n=n+1

    stats_arr.append(m)
    valid_arr.append(m[1])
    fraud_arr.append(m[0])
    fold_number.append(n)
```

```
[[66421    34]
 [   11    88]]
(0.8888888888888888, 0.7213114754098361, 0.9993238573188689, 'LogisticRegression()')
[[66454     0]
 [   61    39]]
(0.39, 1.0, 0.9990834510322445, 'LogisticRegression()')
[[66451     3]
 [   44    56]]
(0.56, 0.9491525423728814, 0.9992938065330408, 'LogisticRegression()')
[[66447     7]
 [   53    47]]
(0.47, 0.8703703703703703, 0.9990984764251585, 'LogisticRegression()')
```

In [40]:
```
fraud_ave_prec=sum(fraud_arr)/len(fraud_arr)
valid_ave_prec=sum(valid_arr)/len(valid_arr)

data= {'Fraud Precsision': fraud_arr, 'Fraud Recall': valid_arr }
df_results=pd.DataFrame(data=data, index= fold_number )
LC_mean=df_results.mean()
print('Logistic classifier with imbalanced dataset')
print(df_results)
print(LC_mean, 'Average')
#print('Valid Average Precision:',valid_ave_prec, 'Fraud Average Precision:', fraud_
```

```
Logistic classifier with imbalanced dataset
   Fraud Precsision  Fraud Recall
1          0.888889      0.721311
2          0.390000      1.000000
3          0.560000      0.949153
4          0.470000      0.870370
Fraud Precsision    0.577222
Fraud Recall        0.885209
dtype: float64 Average
```

SMOTE Oversampling scores

In [41]:
```
#RandomForest with SMOTE
stats_arr=[]
valid_arr=[]
fraud_arr=[]
```

```
fold_number=[]
n=0
for train_idx, test_idx in fold.split(X,y):

    X_train, X_val     = X.iloc[train_idx], X.iloc[test_idx]
    y_train, y_val     = y.iloc[train_idx], y.iloc[test_idx]

    oversample = SMOTE()
    X_res, y_res = oversample.fit_resample(X_train, y_train)

    m=model_scores(RandomForestClassifier(n_estimators=30), X_res, X_val, y_res, y_v

    print(m)

    n=n+1

    stats_arr.append(m)
    valid_arr.append(m[1])
    fraud_arr.append(m[0])
    fold_number.append(n)
```

```
[[66394    61]
 [   13    86]]
(0.8686868686868687, 0.5850340136054422, 0.9988881209243622, 'RandomForestClassifier
(n_estimators=30)')
[[66448     6]
 [   29    71]]
(0.71, 0.922077922077922, 0.9994741112480091, 'RandomForestClassifier(n_estimators=3
0)')
[[66421    33]
 [   25    75]]
(0.75, 0.6944444444444444, 0.9991285272109866, 'RandomForestClassifier(n_estimators=
30)')
[[66443    11]
 [   26    74]]
(0.74, 0.8705882352941177, 0.999444060462181, 'RandomForestClassifier(n_estimators=3
0)')
```

In [42]:

```
#ML performance results
fraud_ave_prec=sum(fraud_arr)/len(fraud_arr)
valid_ave_prec=sum(valid_arr)/len(valid_arr)

data= {'Fraud Precsision': fraud_arr, 'Fraud Recall': valid_arr }
df_results=pd.DataFrame(data=data, index= fold_number )
RF_SMOTE_mean=df_results.mean()
print('Random Forest Classifier with SMOTE')
print(df_results)
print(RF_SMOTE_mean, 'Average')
#print('Valid Average Precision:',valid_ave_prec, 'Fraud Average Precision:', fraud_
```

```
Random Forest Classifier with SMOTE
   Fraud Precsision  Fraud Recall
1          0.868687      0.585034
2          0.710000      0.922078
3          0.750000      0.694444
4          0.740000      0.870588
Fraud Precsision    0.767172
Fraud Recall        0.768036
dtype: float64 Average
```

In [43]:

```
#Logistic  Regression with SMOTE
stats_arr=[]
valid_arr=[]
fraud_arr=[]
fold_number=[]
n=0
```

```
for train_idx, test_idx in fold.split(X,y):

    X_train, X_val     = X.iloc[train_idx], X.iloc[test_idx]
    y_train, y_val     = y.iloc[train_idx], y.iloc[test_idx]

    oversample = SMOTE()
    X_res, y_res = oversample.fit_resample(X_train, y_train)

    m=model_scores(LogisticRegression(), X_res, X_val, y_res, y_val)

    print(m)

    n=n+1

    stats_arr.append(m)
    valid_arr.append(m[1])
    fraud_arr.append(m[0])
    fold_number.append(n)
```

```
[[63890  2565]
 [    4    95]]
(0.9595959595959596, 0.03571428571428571, 0.9613997656038705, 'LogisticRegression
()')
[[65505   949]
 [   21    79]]
(0.79, 0.07684824902723736, 0.985425368873396, 'LogisticRegression()')
[[63762  2692]
 [    9    91]]
(0.91, 0.03269852676967301, 0.9594164137392193, 'LogisticRegression()')
[[64532  1922]
 [   12    88]]
(0.88, 0.04378109452736319, 0.9709408901042762, 'LogisticRegression()')
```

In [44]:
```
#ML performance results
fraud_ave_prec=sum(fraud_arr)/len(fraud_arr)
valid_ave_prec=sum(valid_arr)/len(valid_arr)

data= {'Fraud Precsision': fraud_arr, 'Fraud Recall': valid_arr }
df_results=pd.DataFrame(data=data, index= fold_number )
LC_SMOTE_mean=df_results.mean()
print('Logistic Classifier with SMOTE')
print(df_results)
print(LC_SMOTE_mean, 'Average')
#print('Valid Average Precision:',valid_ave_prec, 'Fraud Average Precision:', fraud_
```

```
Logistic Classifier with SMOTE
   Fraud Precsision  Fraud Recall
1          0.959596      0.035714
2          0.790000      0.076848
3          0.910000      0.032699
4          0.880000      0.043781
Fraud Precsision    0.884899
Fraud Recall        0.047261
dtype: float64 Average
```

In [45]:
```
print(RF_mean, 'RandomForest Average Score')
print(RF_SMOTE_mean, 'RandomForest SMOTE Average Score')
print(LC_mean, 'Logistic Regression Average Score')
print(LC_SMOTE_mean, 'Logistic Regression Average Score with SMOTE')
```

```
Fraud Precsision    0.754747
Fraud Recall        0.900704
dtype: float64 RandomForest Average Score
Fraud Precsision    0.767172
Fraud Recall        0.768036
dtype: float64 RandomForest SMOTE Average Score
Fraud Precsision    0.577222
```

```
Fraud Recall        0.885209
dtype: float64 Logistic Regression Average Score
Fraud Precsision    0.884899
Fraud Recall        0.047261
dtype: float64 Logistic Regression Average Score with SMOTE
```

We see from the scores that sythetic oversampling increases precision of both RandomForest and Logistic classifiers in an imbalanced dataset. When SMOTE is applied the recall scores drop due to the ML algorithm is no longer biased towards the majority class. The logistic classifier has the highest precision but also a significant reduction in the recall score due to many false positives this is a significant drawback.

LSTM Method

In [46]:
```
import pandas as pd
import numpy as np
#import keras
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from keras.layers import Input, Dense
```

Data Importing Visualizing & Understanding for LSTM

In [47]:
```
data = pd.read_csv('creditcard.csv')
```

In [48]:
```
data
```

Out[48]:

	Time	V1	V2	V3	V4	V5	V6	V7	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.09
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.08
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.24
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.37
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.27
...	...	...	...	...	...	...	...	...	
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.30
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.29
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.70
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.67
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.41

284807 rows × 31 columns

In [49]:
```
data.describe()
```

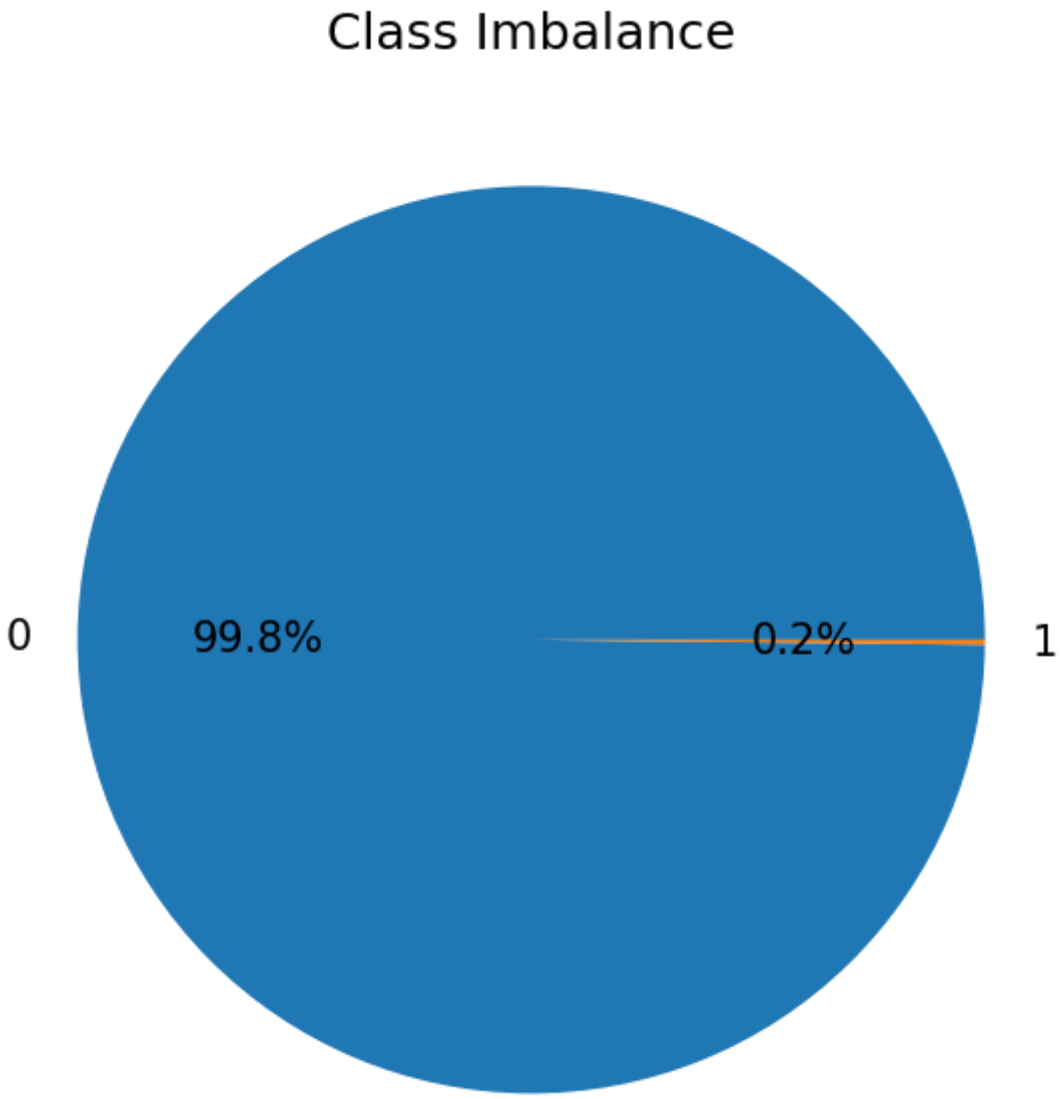
Out[49]:

	Time	V1	V2	V3	V4	V5
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05
mean	94813.859575	3.918649e-15	5.682686e-16	-8.761736e-15	2.811118e-15	-1.552103e-15
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01

8 rows × 31 columns

In [50]:

```
classes = data['Class'].value_counts(normalize=True)
fig = plt.figure(figsize=(5, 5), dpi=150)
plt.pie(classes.values, labels=classes.index, autopct='%1.1f%%')
plt.title('Class Imbalance')
plt.show()
```

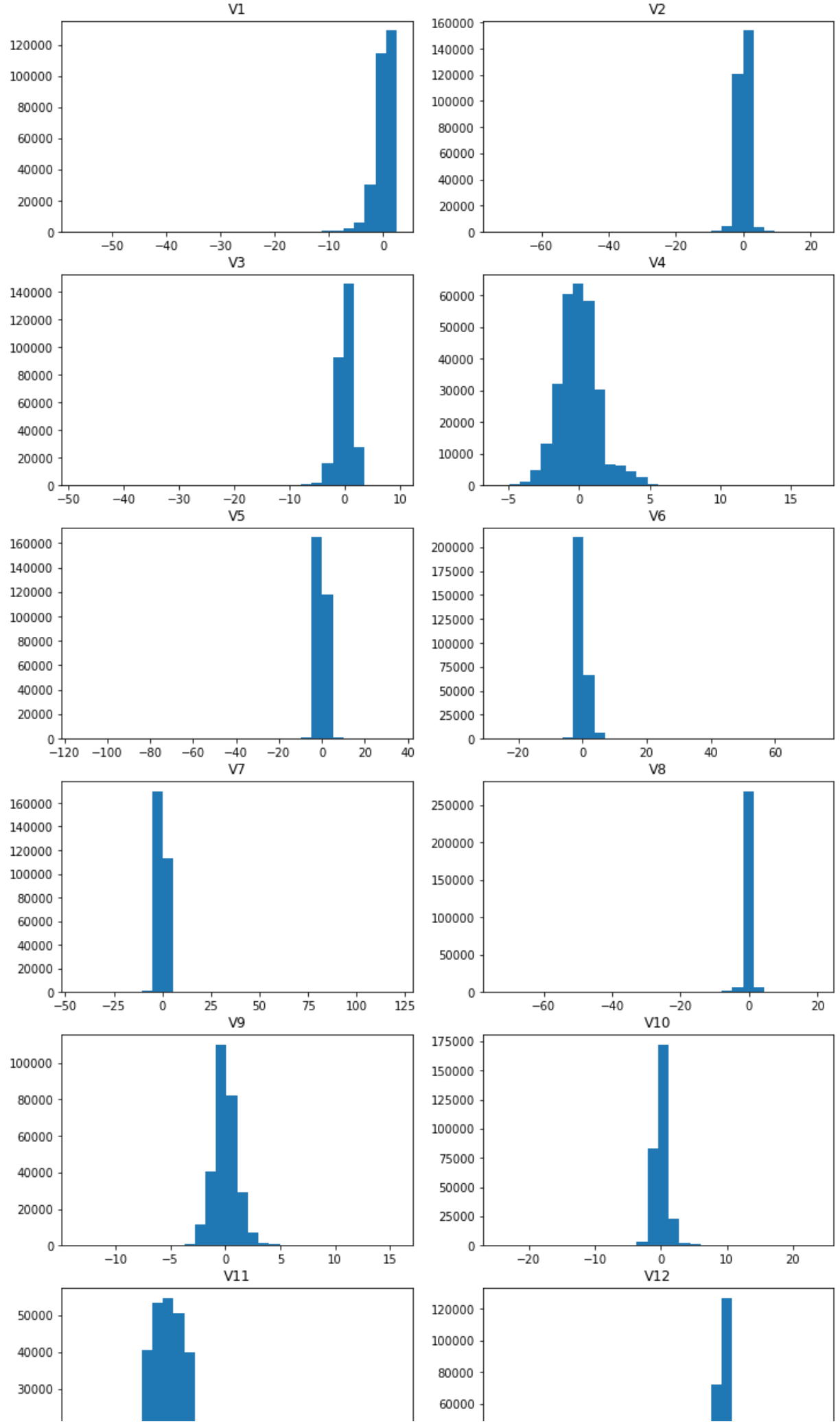


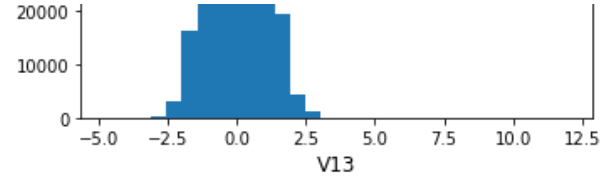
Dropping Duplicates and Incomplete data

In [79]:
```
data.drop_duplicates(inplace = True)

if  data.isnull().values.any() != False:
        df.fillna(0)
```

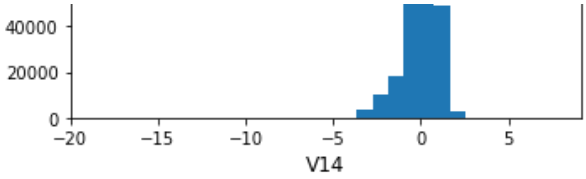
Visualization of Columns

In [51]:
```
num_features = [col for col in data.columns if col.find('V') > -1]

n_cols = 2
n_rows = 14

fig = plt.gcf()
fig.set_size_inches(n_cols * 6, n_rows * 4)

for pos, feature in enumerate(num_features):
    sp = plt.subplot(n_rows, n_cols, pos + 1)
    plt.hist(data[feature], bins=30)
    plt.title(feature)

plt.show()
```

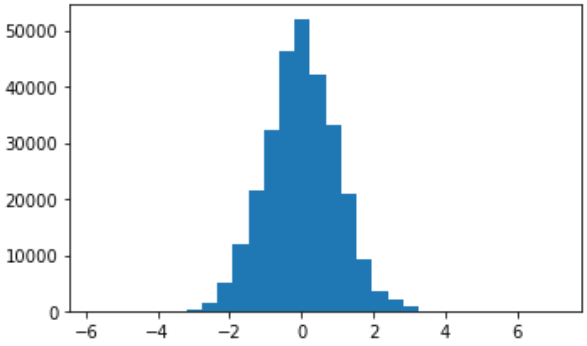

V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12



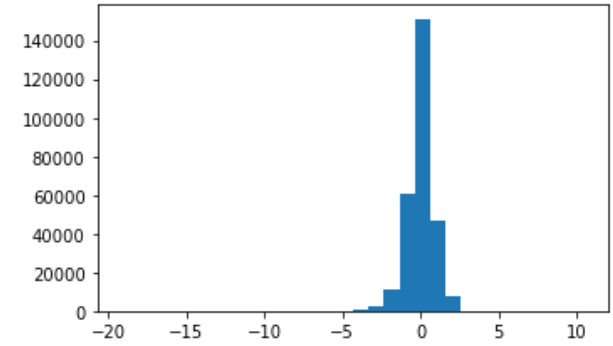
V13



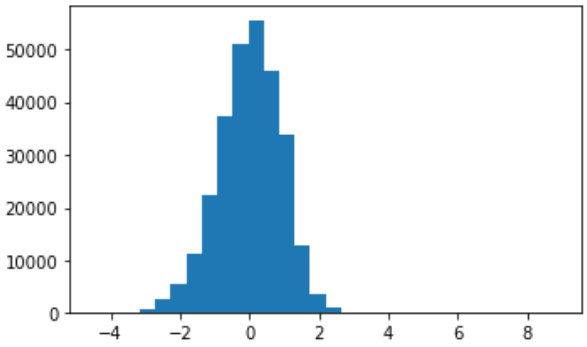
V14



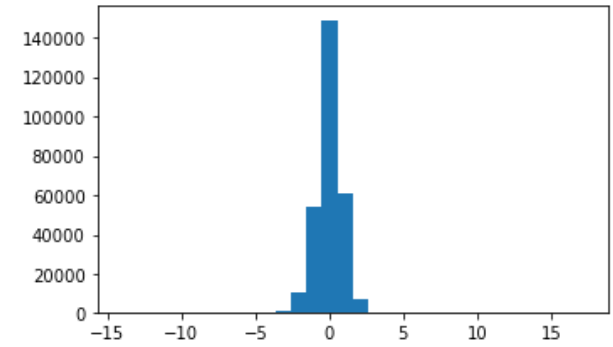
V15



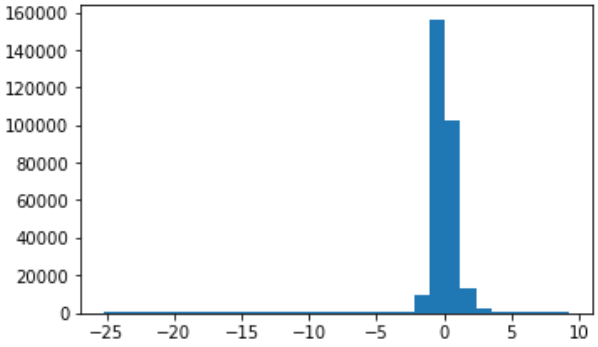
V16



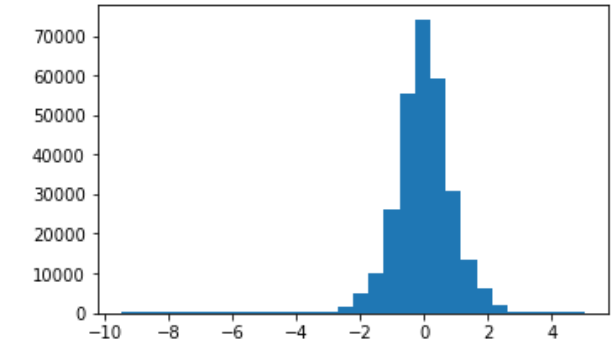
V17



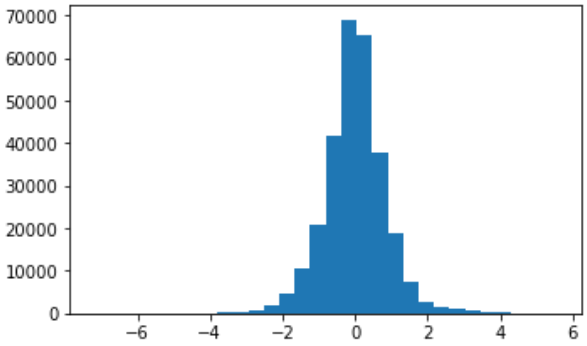
V18



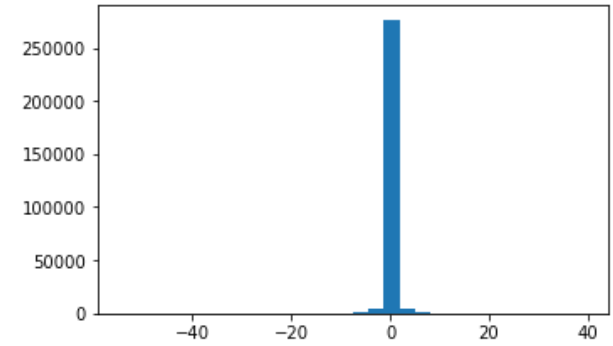
V19



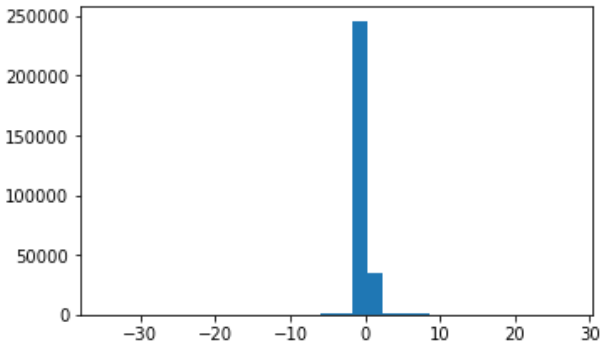
V20



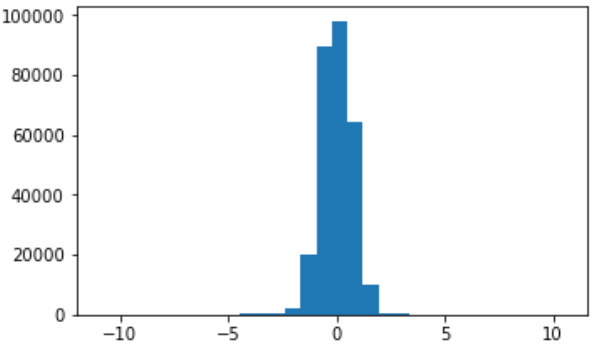
V21



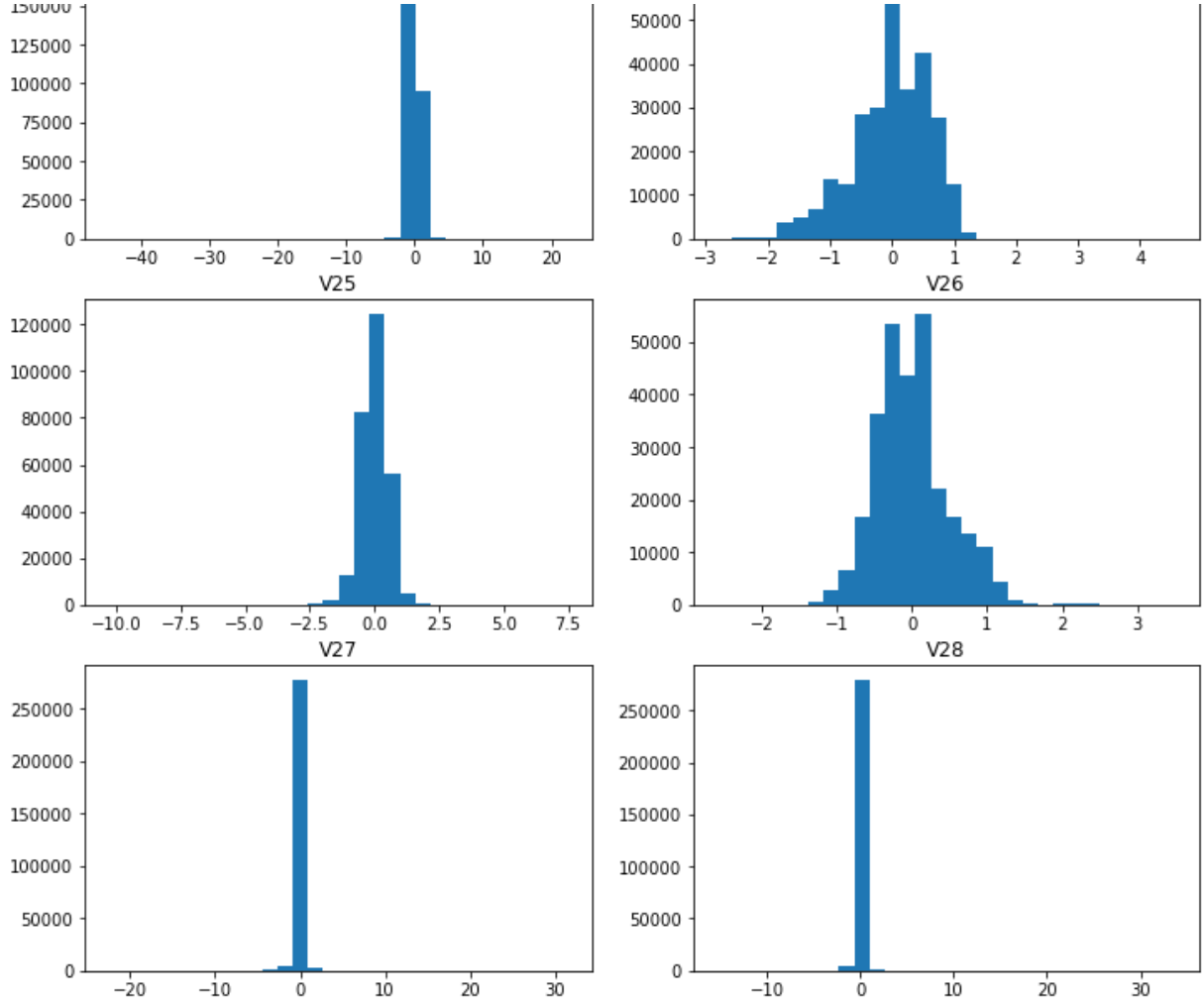
V22



V23

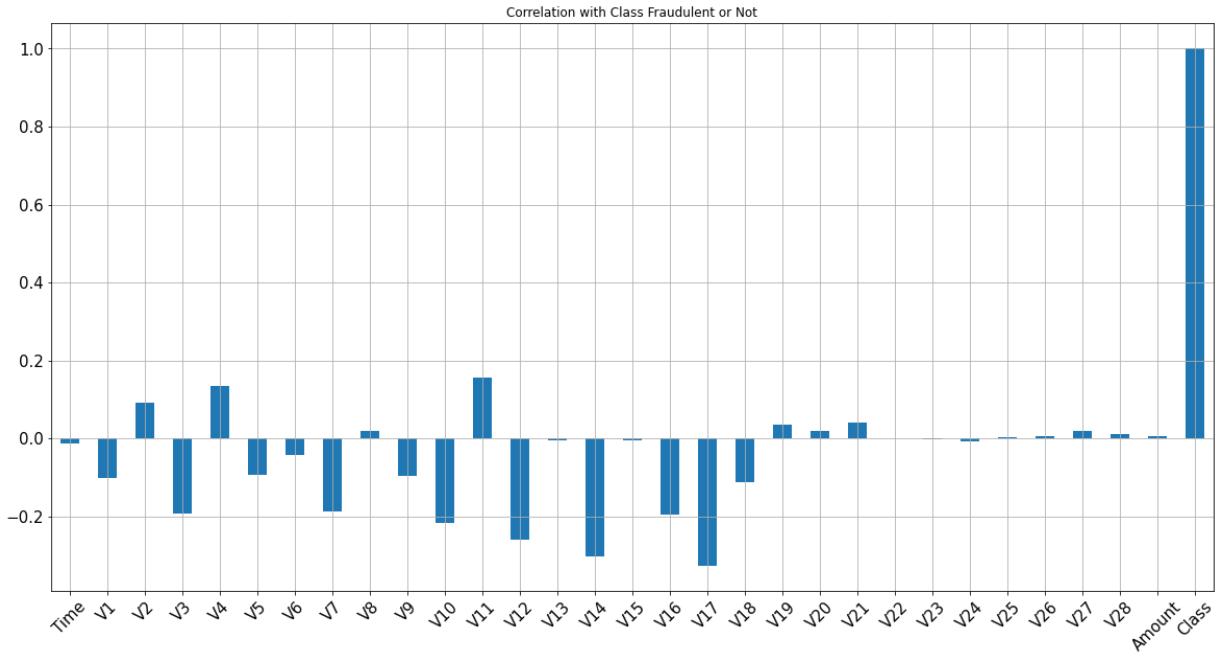


V24



Determining Corelations with the Class

In [52]:
```
  # drop non numerical columns
data.corrwith(data.Class).plot.bar(
        figsize = (20, 10), title = "Correlation with Class Fraudulent or Not", font
        rot = 45, grid = True)
plt.show()
```

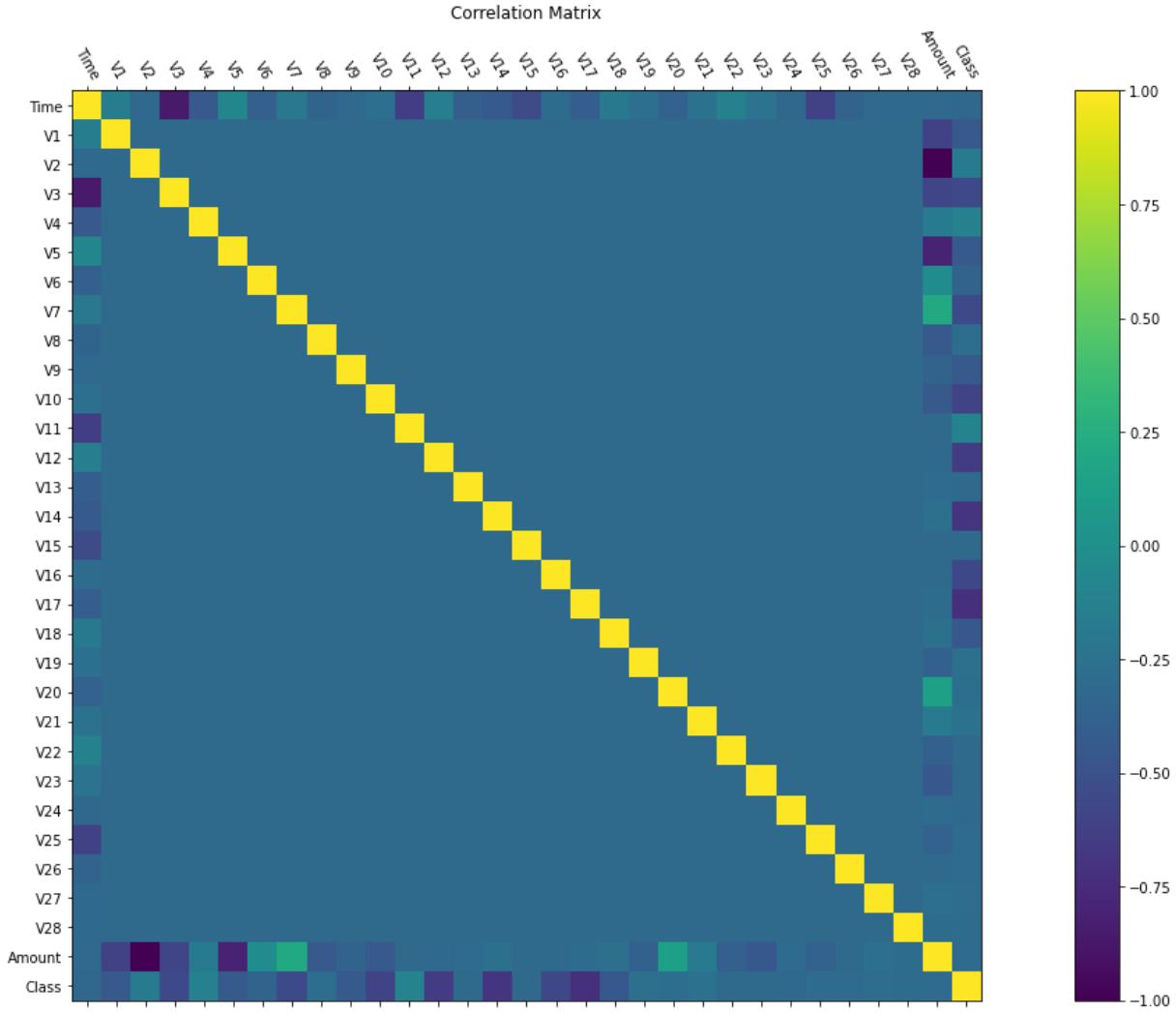


In [53]:
```
from matplotlib.pyplot import figure
```

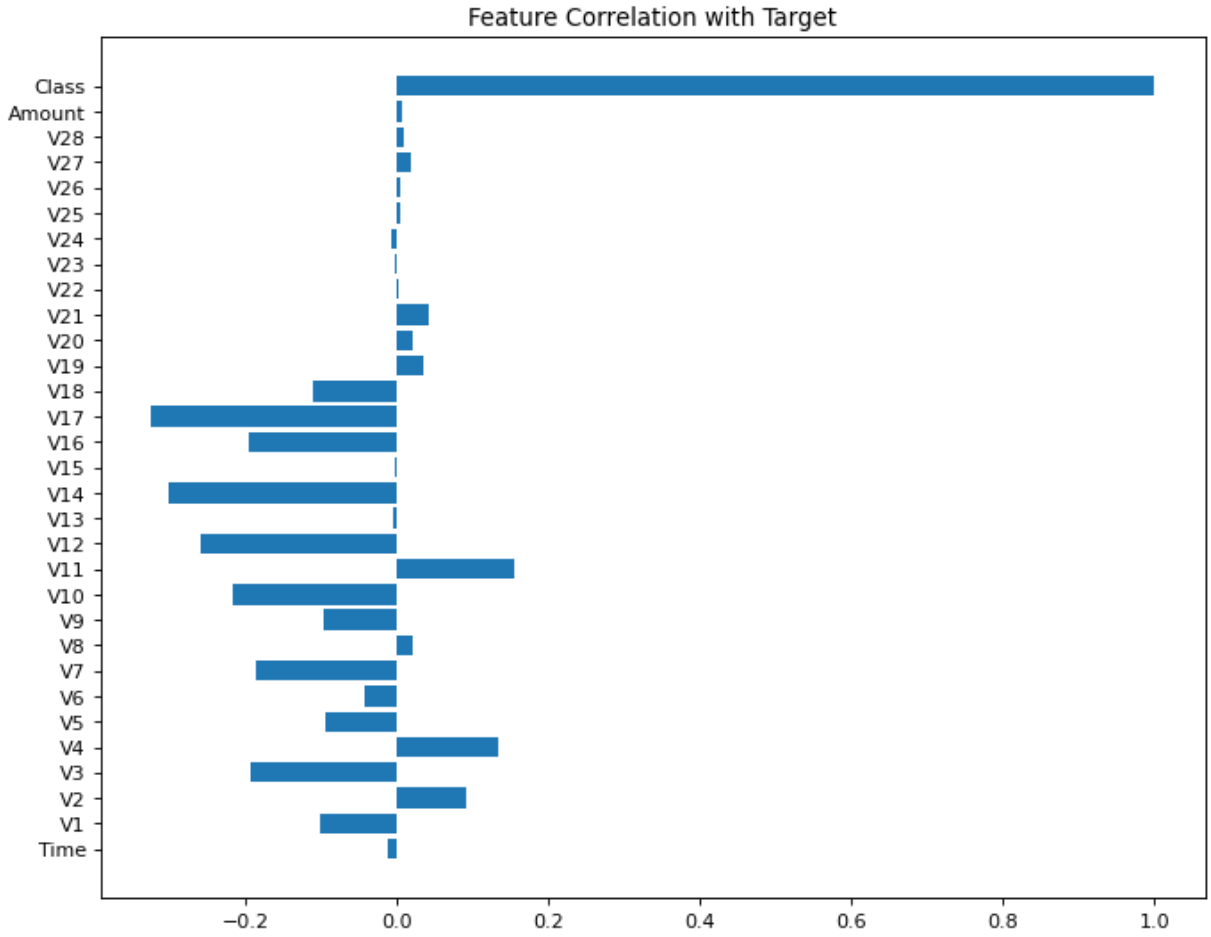
In [ ]:

In [54]:
```
corr = data.corr()
fig, ax = plt.subplots(figsize=(24, 12))
cax=ax.matshow(corr,vmin=-1,vmax=1)
ax.matshow(corr)
plt.xticks(range(len(corr.columns)), corr.columns)
plt.yticks(range(len(corr.columns)), corr.columns)
plt.xticks(rotation=300)
plt.title('Correlation Matrix')
plt.colorbar(cax)
```

Out[54]: <matplotlib.colorbar.Colorbar at 0x1f0a5e6a460>



In [55]:
```
idx = [col for col in data.columns]
figure(figsize=(10, 8), dpi=80)
plt.barh(corr.loc[idx, 'Class'].index, corr.loc[idx, 'Class'].values)
plt.title('Feature Correlation with Target')
plt.show()
```



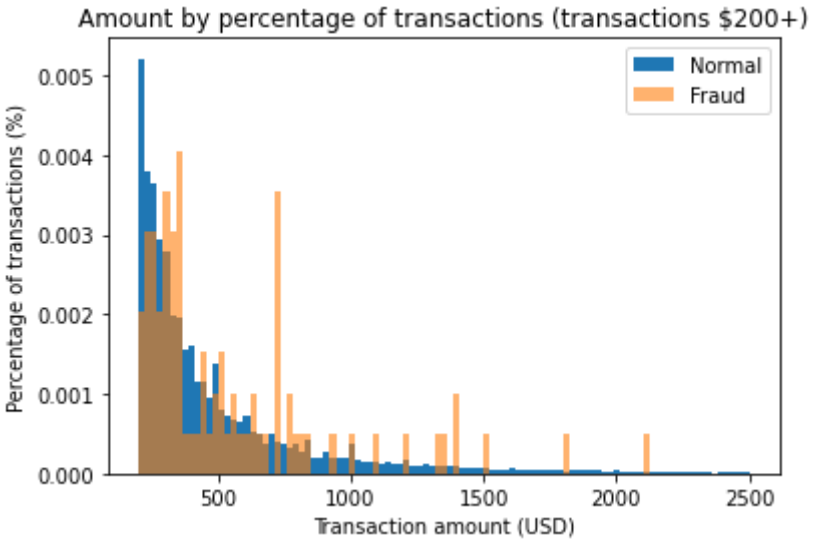
In [56]:
```
normal_df = data[data.Class == 0] #save normal_df observations into a separate df
normal_df.Amount.describe()
```

Out[56]:
```
count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
```

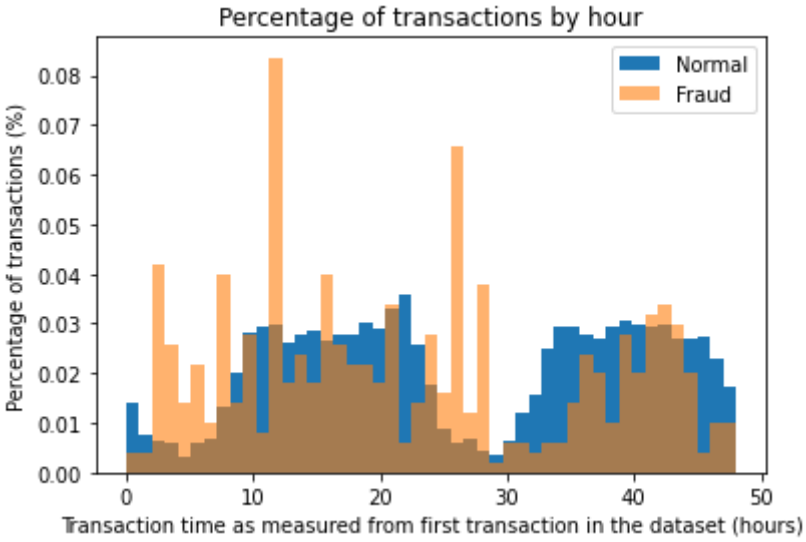
In [57]:
```
fraud_df = data[data.Class == 1] #do the same for frauds
fraud_df.Amount.describe()
```

Out[57]:
```
count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
```

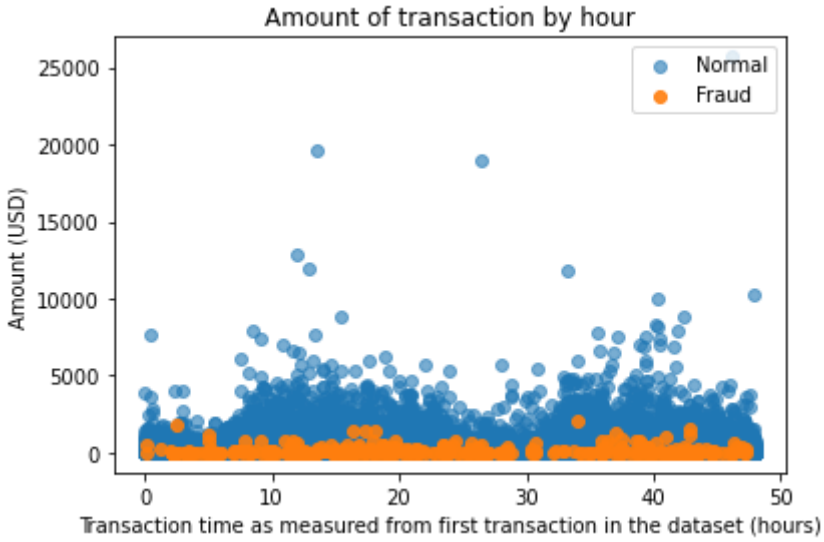
In [58]:
```
#plot of high value transactions
bins = np.linspace(200, 2500, 100)
plt.hist(normal_df.Amount, bins, alpha=1, density = True, label='Normal')
plt.hist(fraud_df.Amount, bins, alpha=0.6, density = True, label='Fraud')
plt.legend(loc='upper right')
plt.title("Amount by percentage of transactions (transactions \$200+)")
plt.xlabel("Transaction amount (USD)")
plt.ylabel("Percentage of transactions (%)");
plt.show()
```



In [59]:
```
bins = np.linspace(0, 48, 48) #48 hours
plt.hist((normal_df.Time/(60*60)), bins, alpha=1, density=True, label='Normal')
plt.hist((fraud_df.Time/(60*60)), bins, alpha=0.6, density=True, label='Fraud')
plt.legend(loc='upper right')
plt.title("Percentage of transactions by hour")
plt.xlabel("Transaction time as measured from first transaction in the dataset (hour
plt.ylabel("Percentage of transactions (%)");
#plt.hist((df.Time/(60*60)),bins)
plt.show()
```



In [60]:
```
plt.scatter((normal_df.Time/(60*60)), normal_df.Amount, alpha=0.6, label='Normal')
plt.scatter((fraud_df.Time/(60*60)), fraud_df.Amount, alpha=0.9, label='Fraud')
plt.title("Amount of transaction by hour")
plt.xlabel("Transaction time as measured from first transaction in the dataset (hour
plt.ylabel('Amount (USD)')
plt.legend(loc='upper right')
plt.show()
```



In [61]:
```
#data = df.drop(['Time'], axis=1) #if you think the var is unimportant
df_norm = data
df_norm['Time'] = StandardScaler().fit_transform(df_norm['Time'].values.reshape(-1,
df_norm['Amount'] = StandardScaler().fit_transform(df_norm['Amount'].values.reshape(
```

In [62]:
```
df_norm
```

Out[62]:

	Time	V1	V2	V3	V4	V5	V6	V7	
0	-1.996583	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.0
1	-1.996583	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.0
2	-1.996562	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.2
3	-1.996562	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.3
4	-1.996541	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.2
...	...	...	...	...	...	...	...	...	
284802	1.641931	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.3
284803	1.641952	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.2
284804	1.641974	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.7
284805	1.641974	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.6
284806	1.642058	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.4

284807 rows × 31 columns

In [63]:
```
np.random.seed(7)
```

In [64]:
```
from sklearn.model_selection import train_test_split
from keras.callbacks import ModelCheckpoint, TensorBoard
from keras import regularizers
from keras.models import Model, load_model
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
import math
```

```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import recall_score, classification_report, auc, roc_curve
```

Splitting data

In [65]:
```
train_x, test_x = train_test_split(df_norm, test_size=0.15, random_state=314) # chan

train_x = train_x[train_x.Class == 0] #where normal transactions
train_x = train_x.drop(['Class'], axis=1) #drop the class column


test_y = test_x['Class'] #save the class column for the test set
test_x = test_x.drop(['Class'], axis=1) #drop the class column

train_x = train_x.values #transform to ndarray
test_x = test_x.values
```

In [66]:
```
train_x.shape
```

Out[66]: (241674, 30)

Running LSTM

In [67]:
```
nb_epoch = 100
batch_size = 120
input_dim = train_x.shape[1] #num of columns, 30
encoding_dim = 14
hidden_dim = int(encoding_dim / 2) #i.e. 7
learning_rate = 1e-7

input_layer = Input(shape=(input_dim, ))
encoder = Dense(encoding_dim, activation="tanh", activity_regularizer=regularizers.l
encoder = Dense(hidden_dim, activation="relu")(encoder)
decoder = Dense(hidden_dim, activation='tanh')(encoder)
decoder = Dense(input_dim, activation='relu')(decoder)
autoencoder = Model(inputs=input_layer, outputs=decoder)
```

In [68]:
```
autoencoder.compile(metrics=['accuracy'],
                    loss='mean_squared_error',
                    optimizer='adam')

cp = ModelCheckpoint(filepath="autoencoder_fraud.h5",
                                save_best_only=True,
                                verbose=0)

tb = TensorBoard(log_dir='./logs',
                 histogram_freq=0,
                 write_graph=True,
                 write_images=True)

history = autoencoder.fit(train_x, train_x,
                    epochs=nb_epoch,
                    batch_size=batch_size,
                    shuffle=True,
                    validation_data=(test_x, test_x),
                    verbose=1,
                    callbacks=[cp, tb]).history


autoencoder = load_model('autoencoder_fraud.h5')
```

```
Epoch 1/100
```

```
2014/2014 [==============================] - 3s 1ms/step - loss: 0.8579 - accuracy:
0.5048 - val_loss: 0.8377 - val_accuracy: 0.5793
Epoch 2/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.7716 - accuracy:
0.6003 - val_loss: 0.7994 - val_accuracy: 0.6218
Epoch 3/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.7472 - accuracy:
0.6307 - val_loss: 0.7840 - val_accuracy: 0.6349
Epoch 4/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.7344 - accuracy:
0.6403 - val_loss: 0.7755 - val_accuracy: 0.6437
Epoch 5/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.7265 - accuracy:
0.6569 - val_loss: 0.7691 - val_accuracy: 0.6599
Epoch 6/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.7219 - accuracy:
0.6629 - val_loss: 0.7659 - val_accuracy: 0.6614
Epoch 7/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.7188 - accuracy:
0.6648 - val_loss: 0.7630 - val_accuracy: 0.6605
Epoch 8/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.7168 - accuracy:
0.6670 - val_loss: 0.7619 - val_accuracy: 0.6661
Epoch 9/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.7149 - accuracy:
0.6686 - val_loss: 0.7605 - val_accuracy: 0.6690
Epoch 10/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.7130 - accuracy:
0.6709 - val_loss: 0.7585 - val_accuracy: 0.6670
Epoch 11/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.7108 - accuracy:
0.6737 - val_loss: 0.7547 - val_accuracy: 0.6796
Epoch 12/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.7084 - accuracy:
0.6870 - val_loss: 0.7522 - val_accuracy: 0.6903
Epoch 13/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.7063 - accuracy:
0.6941 - val_loss: 0.7504 - val_accuracy: 0.6936
Epoch 14/100
2014/2014 [==============================] - 3s 2ms/step - loss: 0.7054 - accuracy:
0.6962 - val_loss: 0.7510 - val_accuracy: 0.6903
Epoch 15/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.7049 - accuracy:
0.6980 - val_loss: 0.7487 - val_accuracy: 0.6975
Epoch 16/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.7038 - accuracy:
0.6993 - val_loss: 0.7483 - val_accuracy: 0.7010
Epoch 17/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.7033 - accuracy:
0.7012 - val_loss: 0.7476 - val_accuracy: 0.7034
Epoch 18/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.7027 - accuracy:
0.7025 - val_loss: 0.7469 - val_accuracy: 0.6990
Epoch 19/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.7022 - accuracy:
0.7038 - val_loss: 0.7449 - val_accuracy: 0.7043
Epoch 20/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.7015 - accuracy:
0.7054 - val_loss: 0.7450 - val_accuracy: 0.7031
Epoch 21/100
2014/2014 [==============================] - 3s 2ms/step - loss: 0.7010 - accuracy:
0.7063 - val_loss: 0.7438 - val_accuracy: 0.7003
Epoch 22/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.7006 - accuracy:
0.7055 - val_loss: 0.7444 - val_accuracy: 0.7067
Epoch 23/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.7004 - accuracy:
0.7052 - val_loss: 0.7442 - val_accuracy: 0.7046
Epoch 24/100
```

```
2014/2014 [==============================] - 3s 2ms/step - loss: 0.6997 - accuracy:
0.7064 - val_loss: 0.7442 - val_accuracy: 0.7049
Epoch 25/100
2014/2014 [==============================] - 3s 2ms/step - loss: 0.6997 - accuracy:
0.7059 - val_loss: 0.7431 - val_accuracy: 0.7064
Epoch 26/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6995 - accuracy:
0.7066 - val_loss: 0.7428 - val_accuracy: 0.7041
Epoch 27/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6992 - accuracy:
0.7069 - val_loss: 0.7420 - val_accuracy: 0.7056
Epoch 28/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6988 - accuracy:
0.7070 - val_loss: 0.7454 - val_accuracy: 0.7020
Epoch 29/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6989 - accuracy:
0.7068 - val_loss: 0.7436 - val_accuracy: 0.7043
Epoch 30/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6983 - accuracy:
0.7074 - val_loss: 0.7422 - val_accuracy: 0.7104
Epoch 31/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6980 - accuracy:
0.7081 - val_loss: 0.7427 - val_accuracy: 0.7016
Epoch 32/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6980 - accuracy:
0.7078 - val_loss: 0.7427 - val_accuracy: 0.7020
Epoch 33/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6976 - accuracy:
0.7086 - val_loss: 0.7421 - val_accuracy: 0.7065
Epoch 34/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6977 - accuracy:
0.7085 - val_loss: 0.7412 - val_accuracy: 0.7089
Epoch 35/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6974 - accuracy:
0.7084 - val_loss: 0.7411 - val_accuracy: 0.7062
Epoch 36/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6970 - accuracy:
0.7087 - val_loss: 0.7411 - val_accuracy: 0.7080
Epoch 37/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6970 - accuracy:
0.7100 - val_loss: 0.7414 - val_accuracy: 0.7049
Epoch 38/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6966 - accuracy:
0.7109 - val_loss: 0.7406 - val_accuracy: 0.7119
Epoch 39/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6967 - accuracy:
0.7108 - val_loss: 0.7408 - val_accuracy: 0.7116
Epoch 40/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6964 - accuracy:
0.7118 - val_loss: 0.7412 - val_accuracy: 0.7051
Epoch 41/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6962 - accuracy:
0.7126 - val_loss: 0.7408 - val_accuracy: 0.7099
Epoch 42/100
2014/2014 [==============================] - 2s 1ms/step - loss: 0.6962 - accuracy:
0.7122 - val_loss: 0.7412 - val_accuracy: 0.7104
Epoch 43/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6958 - accuracy:
0.7133 - val_loss: 0.7409 - val_accuracy: 0.7094
Epoch 44/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6959 - accuracy:
0.7133 - val_loss: 0.7407 - val_accuracy: 0.7100
Epoch 45/100
2014/2014 [==============================] - 2s 1ms/step - loss: 0.6957 - accuracy:
0.7141 - val_loss: 0.7394 - val_accuracy: 0.7134
Epoch 46/100
2014/2014 [==============================] - 2s 1ms/step - loss: 0.6954 - accuracy:
0.7138 - val_loss: 0.7399 - val_accuracy: 0.7088
Epoch 47/100
```

```
2014/2014 [==============================] - 2s 1ms/step - loss: 0.6953 - accuracy:
0.7149 - val_loss: 0.7396 - val_accuracy: 0.7169
Epoch 48/100
2014/2014 [==============================] - 2s 1ms/step - loss: 0.6951 - accuracy:
0.7156 - val_loss: 0.7396 - val_accuracy: 0.7076
Epoch 49/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6950 - accuracy:
0.7146 - val_loss: 0.7390 - val_accuracy: 0.7174
Epoch 50/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6947 - accuracy:
0.7143 - val_loss: 0.7420 - val_accuracy: 0.7131
Epoch 51/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6948 - accuracy:
0.7132 - val_loss: 0.7380 - val_accuracy: 0.7145
Epoch 52/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6947 - accuracy:
0.7141 - val_loss: 0.7431 - val_accuracy: 0.7115
Epoch 53/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6943 - accuracy:
0.7137 - val_loss: 0.7383 - val_accuracy: 0.7168
Epoch 54/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6946 - accuracy:
0.7133 - val_loss: 0.7385 - val_accuracy: 0.7121
Epoch 55/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6944 - accuracy:
0.7133 - val_loss: 0.7382 - val_accuracy: 0.7176
Epoch 56/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6940 - accuracy:
0.7132 - val_loss: 0.7392 - val_accuracy: 0.7090
Epoch 57/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6937 - accuracy:
0.7133 - val_loss: 0.7380 - val_accuracy: 0.7128
Epoch 58/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6937 - accuracy:
0.7130 - val_loss: 0.7406 - val_accuracy: 0.7120
Epoch 59/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6937 - accuracy:
0.7134 - val_loss: 0.7416 - val_accuracy: 0.7042
Epoch 60/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6934 - accuracy:
0.7134 - val_loss: 0.7457 - val_accuracy: 0.7106
Epoch 61/100
2014/2014 [==============================] - 2s 1ms/step - loss: 0.6934 - accuracy:
0.7138 - val_loss: 0.7376 - val_accuracy: 0.7143
Epoch 62/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6933 - accuracy:
0.7131 - val_loss: 0.7386 - val_accuracy: 0.7101
Epoch 63/100
2014/2014 [==============================] - 2s 1ms/step - loss: 0.6931 - accuracy:
0.7135 - val_loss: 0.7385 - val_accuracy: 0.7164
Epoch 64/100
2014/2014 [==============================] - 2s 1ms/step - loss: 0.6935 - accuracy:
0.7134 - val_loss: 0.7375 - val_accuracy: 0.7096
Epoch 65/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6930 - accuracy:
0.7134 - val_loss: 0.7372 - val_accuracy: 0.7138
Epoch 66/100
2014/2014 [==============================] - 2s 1ms/step - loss: 0.6932 - accuracy:
0.7124 - val_loss: 0.7370 - val_accuracy: 0.7156
Epoch 67/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6926 - accuracy:
0.7129 - val_loss: 0.7377 - val_accuracy: 0.7183
Epoch 68/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6929 - accuracy:
0.7128 - val_loss: 0.7369 - val_accuracy: 0.7175
Epoch 69/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6930 - accuracy:
0.7127 - val_loss: 0.7366 - val_accuracy: 0.7164
Epoch 70/100
```

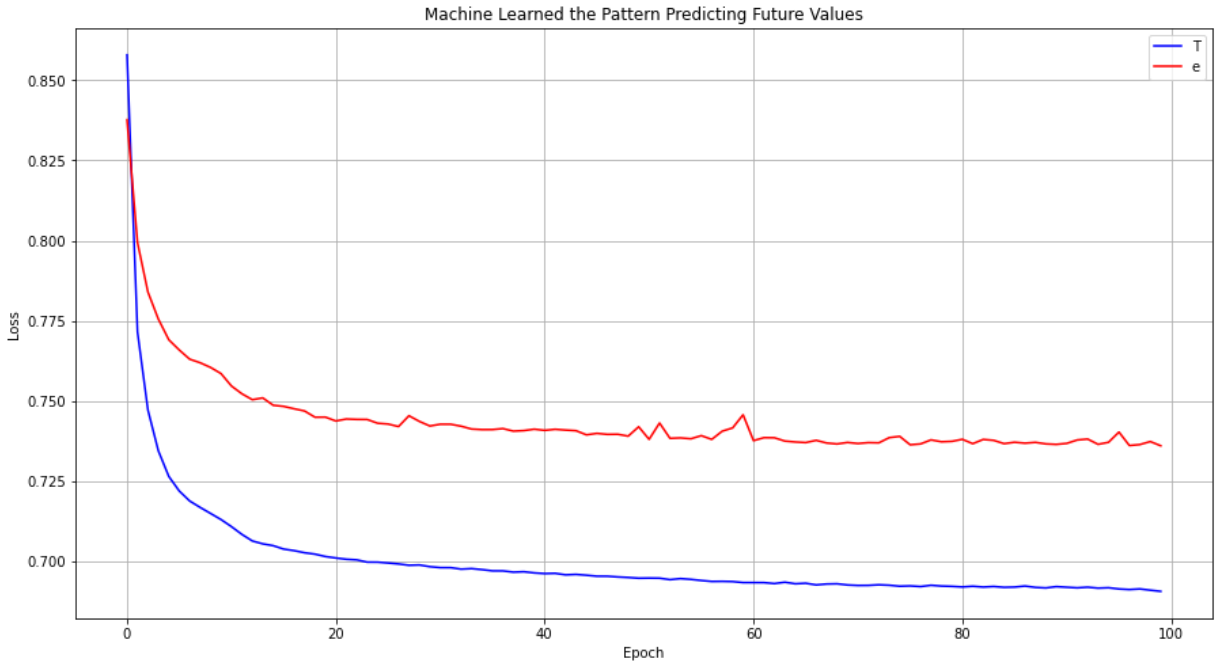
```
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6926 - accuracy:
0.7137 - val_loss: 0.7371 - val_accuracy: 0.7133
Epoch 71/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6925 - accuracy:
0.7143 - val_loss: 0.7367 - val_accuracy: 0.7155
Epoch 72/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6925 - accuracy:
0.7126 - val_loss: 0.7370 - val_accuracy: 0.7191
Epoch 73/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6927 - accuracy:
0.7145 - val_loss: 0.7369 - val_accuracy: 0.7173
Epoch 74/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6925 - accuracy:
0.7142 - val_loss: 0.7386 - val_accuracy: 0.7119
Epoch 75/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6922 - accuracy:
0.7140 - val_loss: 0.7390 - val_accuracy: 0.7074
Epoch 76/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6923 - accuracy:
0.7146 - val_loss: 0.7363 - val_accuracy: 0.7197
Epoch 77/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6921 - accuracy:
0.7139 - val_loss: 0.7366 - val_accuracy: 0.7138
Epoch 78/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6925 - accuracy:
0.7132 - val_loss: 0.7379 - val_accuracy: 0.7121
Epoch 79/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6923 - accuracy:
0.7138 - val_loss: 0.7373 - val_accuracy: 0.7124
Epoch 80/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6922 - accuracy:
0.7125 - val_loss: 0.7374 - val_accuracy: 0.7148
Epoch 81/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6920 - accuracy:
0.7133 - val_loss: 0.7380 - val_accuracy: 0.7108
Epoch 82/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6922 - accuracy:
0.7128 - val_loss: 0.7367 - val_accuracy: 0.7133
Epoch 83/100
2014/2014 [==============================] - 2s 1ms/step - loss: 0.6920 - accuracy:
0.7135 - val_loss: 0.7380 - val_accuracy: 0.7127
Epoch 84/100
2014/2014 [==============================] - 2s 1ms/step - loss: 0.6921 - accuracy:
0.7138 - val_loss: 0.7377 - val_accuracy: 0.7147
Epoch 85/100
2014/2014 [==============================] - 2s 1ms/step - loss: 0.6919 - accuracy:
0.7131 - val_loss: 0.7367 - val_accuracy: 0.7159
Epoch 86/100
2014/2014 [==============================] - 2s 1ms/step - loss: 0.6920 - accuracy:
0.7129 - val_loss: 0.7371 - val_accuracy: 0.7113
Epoch 87/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6923 - accuracy:
0.7128 - val_loss: 0.7368 - val_accuracy: 0.7182
Epoch 88/100
2014/2014 [==============================] - 2s 1ms/step - loss: 0.6919 - accuracy:
0.7135 - val_loss: 0.7371 - val_accuracy: 0.7135
Epoch 89/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6917 - accuracy:
0.7137 - val_loss: 0.7366 - val_accuracy: 0.7124
Epoch 90/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6921 - accuracy:
0.7128 - val_loss: 0.7365 - val_accuracy: 0.7153
Epoch 91/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6919 - accuracy:
0.7128 - val_loss: 0.7368 - val_accuracy: 0.7166
Epoch 92/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6917 - accuracy:
0.7131 - val_loss: 0.7378 - val_accuracy: 0.7140
Epoch 93/100
```

```
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6919 - accuracy:
0.7133 - val_loss: 0.7381 - val_accuracy: 0.7152
Epoch 94/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6916 - accuracy:
0.7140 - val_loss: 0.7365 - val_accuracy: 0.7153
Epoch 95/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6918 - accuracy:
0.7126 - val_loss: 0.7371 - val_accuracy: 0.7108
Epoch 96/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6914 - accuracy:
0.7111 - val_loss: 0.7403 - val_accuracy: 0.7020
Epoch 97/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6912 - accuracy:
0.7113 - val_loss: 0.7361 - val_accuracy: 0.7130
Epoch 98/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6914 - accuracy:
0.7103 - val_loss: 0.7364 - val_accuracy: 0.7089
Epoch 99/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6910 - accuracy:
0.7102 - val_loss: 0.7373 - val_accuracy: 0.7087
Epoch 100/100
2014/2014 [==============================] - 3s 1ms/step - loss: 0.6906 - accuracy:
0.7103 - val_loss: 0.7360 - val_accuracy: 0.7093
```

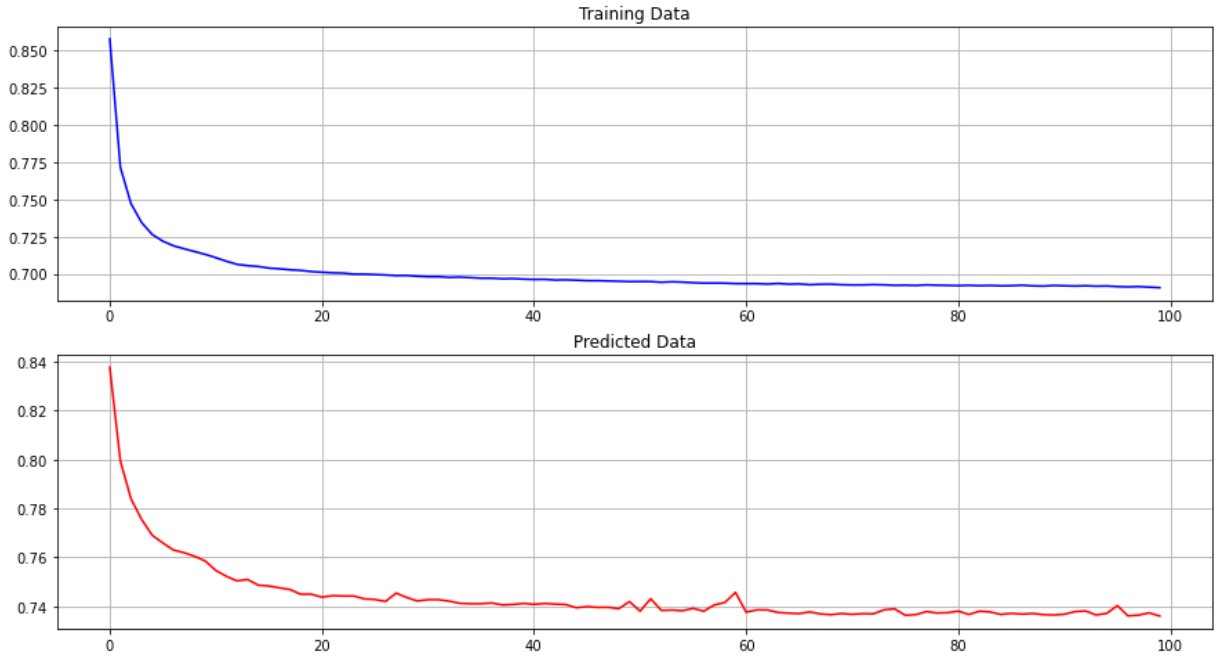
Results and Comparison

In [70]:

```
plt.rcParams['figure.figsize'] = [15, 8]

#dataset1.plot(x='date', y='total daily KW', style='-')
#clusters_dis[0].plot(x='date', y='average per day in cluster', style='-')
#plt.plot(dataset1)
plt.plot(history['loss'], color='blue')
plt.legend('Train')
plt.plot(history['val_loss'], color='red')
plt.legend('Test')

plt.title("Machine Learned the Pattern Predicting Future Values")
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.grid()
plt.show()

#plot_model(model, to_file='model.png')
fig, axs = plt.subplots(2)
fig.suptitle('Vertically stacked subplots')
axs[0].title.set_text('Training Data')
axs[0].plot(history['loss'], color='blue')
axs[0].grid()
axs[1].title.set_text('Predicted Data')
axs[1].plot(history['val_loss'], color='red')
axs[1].grid()
plt.show()
```



Vertically stacked subplots



In [ ]:

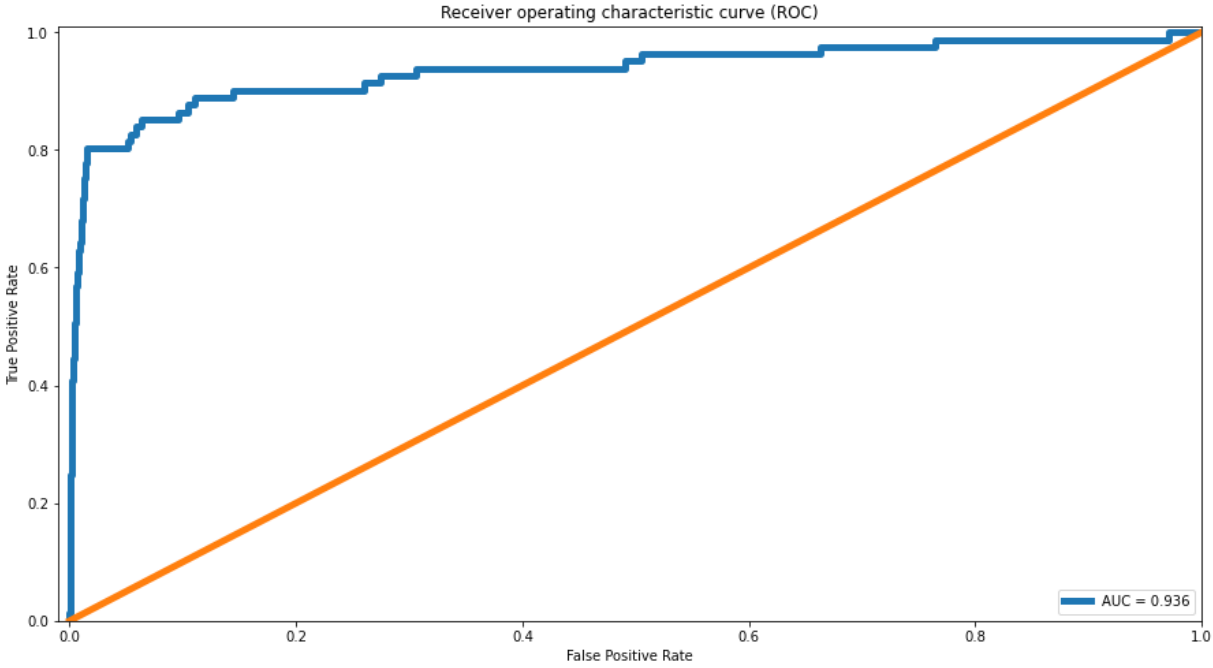
In [71]:
```
test_x_predictions = autoencoder.predict(test_x)
mse = np.mean(np.power(test_x - test_x_predictions, 2), axis=1)
error_df = pd.DataFrame({'Reconstruction_error': mse,
                         'True_class': test_y})
error_df.describe()
```

Out[71]:

	Reconstruction_error	True_class
count	42722.000000	42722.000000
mean	0.736026	0.001896
std	3.260654	0.043502
min	0.059947	0.000000
25%	0.249152	0.000000

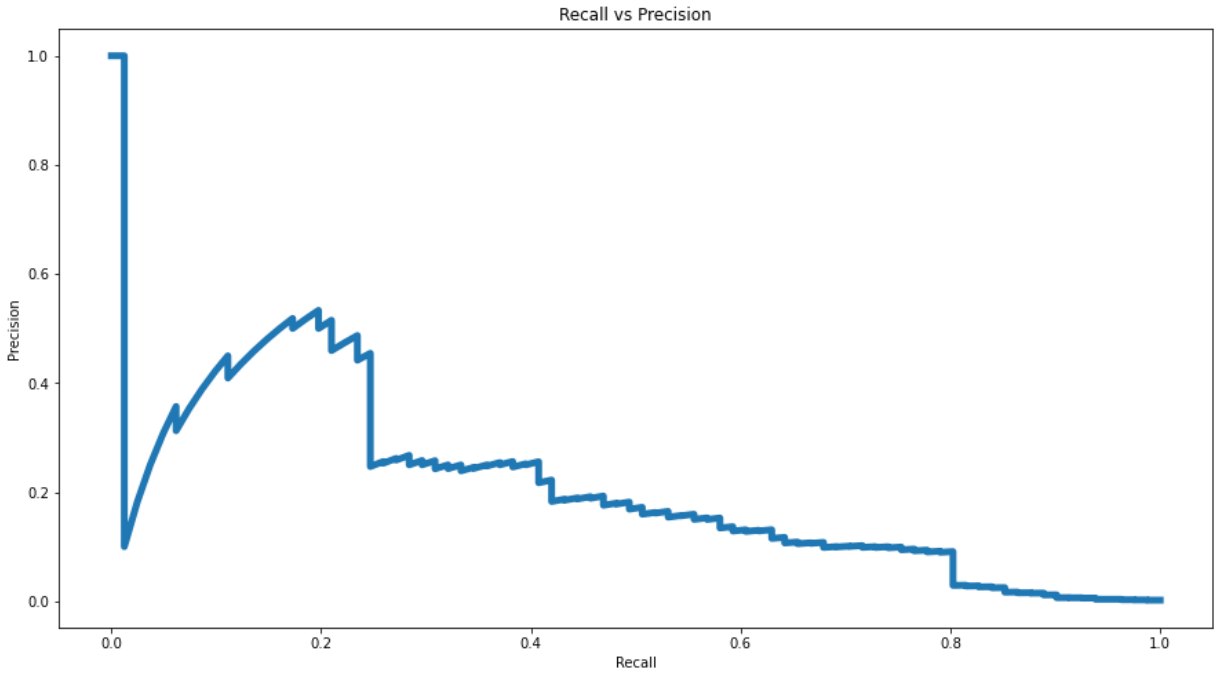
	Reconstruction_error	True_class
50%	0.396743	0.000000
75%	0.632434	0.000000
max	208.377927	1.000000

In [72]:
```
false_pos_rate, true_pos_rate, thresholds = roc_curve(error_df.True_class, error_df.
roc_auc = auc(false_pos_rate, true_pos_rate,)

plt.plot(false_pos_rate, true_pos_rate, linewidth=5, label='AUC = %0.3f'% roc_auc)
plt.plot([0,1],[0,1], linewidth=5)

plt.xlim([-0.01, 1])
plt.ylim([0, 1.01])
plt.legend(loc='lower right')
plt.title('Receiver operating characteristic curve (ROC)')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

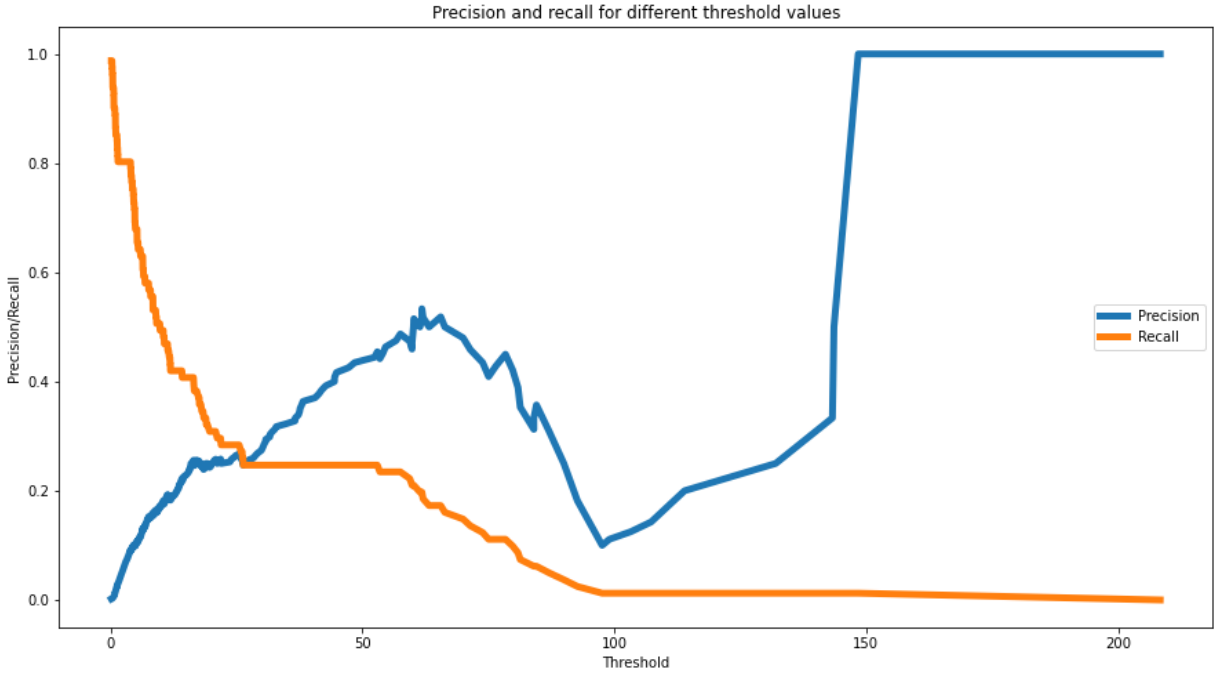


In [73]:
```
from sklearn.metrics import confusion_matrix, precision_recall_curve
```

In [74]:
```
precision_rt, recall_rt, threshold_rt = precision_recall_curve(error_df.True_class,
plt.plot(recall_rt, precision_rt, linewidth=5, label='Precision-Recall curve')
plt.title('Recall vs Precision')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.show()
```



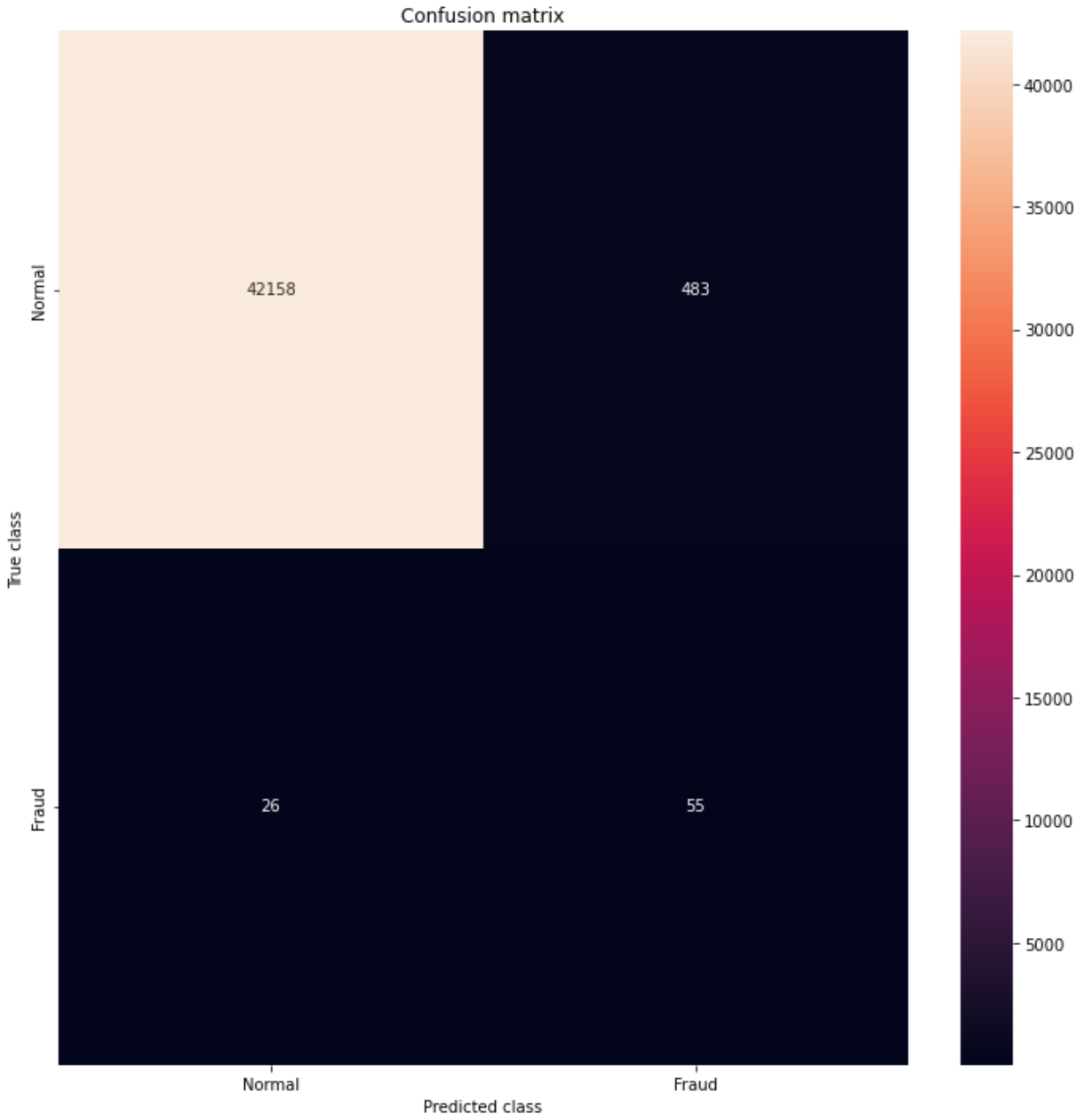
In [75]:
```
plt.plot(threshold_rt, precision_rt[1:], label="Precision",linewidth=5)
plt.plot(threshold_rt, recall_rt[1:], label="Recall",linewidth=5)
plt.title('Precision and recall for different threshold values')
plt.xlabel('Threshold')
plt.ylabel('Precision/Recall')
plt.legend()
plt.show()
```



In [76]:
```
threshold_fixed = 5
LABELS = ["Normal","Fraud"]
```

In [77]:
```
pred_y = [1 if e > threshold_fixed else 0 for e in error_df.Reconstruction_error.val
conf_matrix = confusion_matrix(error_df.True_class, pred_y)

plt.figure(figsize=(12, 12))
sns.heatmap(conf_matrix, xticklabels=LABELS, yticklabels=LABELS, annot=True, fmt="d"
plt.title("Confusion matrix")
plt.ylabel('True class')
```

```
plt.xlabel('Predicted class')
plt.show()
```



Reference:

[1] https://github.com/AdilKhurshid/Electron/

[2] https://github.com/LaurentVeyssier/Credit-Card-fraud-detection-using-Machine-Learning

[3] https://github.com/codenamewei/anomaly-detection-with-use-case

[4] https://gist.github.com/bigsnarfdude/

All group members contributed equally

End of Project

In [ ]:

In [ ]: