1. Dataset and metadata

https://www.kaggle.com/dalpozz/creditcardfraud
The dataset contains transactions made by credit cards in September 2013 by European cardholders. The transactions all happened within 2 days, where there are 492 frauds out of 284, 807 transactions. (1)
It's in the format of .csv tabular data.
V1-V28 are the numerical principal components obtained through PCA transformation. Due to confidentiality issues, the original features and background information are not provided.
There are additional columns in the data that have the following information:

Time	the seconds elapsed between each transaction and the first transaction in the dataset.	Numeric
Amount	Transaction amount	Numeric
Class	The actual classification classes. 1: fraud; 0: otherwise	Numeric

2. Exploratory Data Analysis

After loading the csv original file, data is examined by using several methods and/or calling attributes. The process is summarized as below:

```
print(df.info())
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
Time      284807 non-null float64
V1        284807 non-null float64
V2        284807 non-null float64
V3        284807 non-null float64
V4        284807 non-null float64
V5        284807 non-null float64
V6        284807 non-null float64
V7        284807 non-null float64
V8        284807 non-null float64
V9        284807 non-null float64
```

```
V10       284807 non-null float64
V11       284807 non-null float64
V12       284807 non-null float64
V13       284807 non-null float64
V14       284807 non-null float64
V15       284807 non-null float64
V16       284807 non-null float64
V17       284807 non-null float64
V18       284807 non-null float64
V19       284807 non-null float64
V20       284807 non-null float64
V21       284807 non-null float64
V22       284807 non-null float64
V23       284807 non-null float64
V24       284807 non-null float64
V25       284807 non-null float64
V26       284807 non-null float64
V27       284807 non-null float64
V28       284807 non-null float64
Amount    284807 non-null float64
Class     284807 non-null int64
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
None
```

We can see that it has 284807 records in total, which is consistent with the description in the metadata. All of the features are numerical, with only 'Class' in the integer format, others being float. None of the columns have missing data.

Further .shape attribute is used to confirm the dimensions: (284807, 31) and .columns attribute to confirm the column names:

```
Index([u'Time', u'V1', u'V2', u'V3', u'V4', u'V5', u'V6', u'V7', u'V8', u'V9',
       u'V10', u'V11', u'V12', u'V13', u'V14', u'V15', u'V16', u'V17', u'V18',
       u'V19', u'V20', u'V21', u'V22', u'V23', u'V24', u'V25', u'V26', u'V27',
       u'V28', u'Amount', u'Class'],
      dtype='object')
```

Finally, .describe(), .head() and .tail() methods are used to inspect summary stats, and general data structures of the beginning, and the end of the data frame respectively. From the below results, we can easily see that time is not unique for each entry, which means there are multiple transactions happening within the same time in the dataset. This also implies that it might not be optimal to use 'Time' as the index column for analysis.

```
print(df.describe())
              Time            V1            V2            V3            V4  \
count  284807.000000  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean    94813.859575  3.919560e-15  5.688174e-16 -8.769071e-15  2.782312e-15
std     47488.145955  1.958696e+00  1.651309e+00  1.516255e+00  1.415869e+00
min         0.000000 -5.640751e+01 -7.271573e+01 -4.832559e+01 -5.683171e+00
25%     54201.500000 -9.203734e-01 -5.985499e-01 -8.903648e-01 -8.486401e-01
50%     84692.000000  1.810880e-02  6.548556e-02  1.798463e-01 -1.984653e-02
75%    139320.500000  1.315642e+00  8.037239e-01  1.027196e+00  7.433413e-01
max    172792.000000  2.454930e+00  2.205773e+01  9.382558e+00  1.687534e+01

                V5            V6            V7            V8            V9  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean  -1.552563e-15  2.010663e-15 -1.694249e-15 -1.927028e-16 -3.137024e-15
std    1.380247e+00  1.332271e+00  1.237094e+00  1.194353e+00  1.098632e+00
min   -1.137433e+02 -2.616051e+01 -4.355724e+01 -7.321672e+01 -1.343407e+01
25%   -6.915971e-01 -7.682956e-01 -5.540759e-01 -2.086297e-01 -6.430976e-01
50%   -5.433583e-02 -2.741871e-01  4.010308e-02  2.235804e-02 -5.142873e-02
75%    6.119264e-01  3.985649e-01  5.704361e-01  3.273459e-01  5.971390e-01
max    3.480167e+01  7.330163e+01  1.205895e+02  2.000721e+01  1.559499e+01

               V21           V22           V23           V24  \
count      ...       2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean       ...       1.537294e-16  7.959909e-16  5.367590e-16  4.458112e-15
std        ...       7.345240e-01  7.257016e-01  6.244603e-01  6.056471e-01
min        ...      -3.483038e+01 -1.093314e+01 -4.480774e+01 -2.836627e+00
25%        ...      -2.283949e-01 -5.423504e-01 -1.618463e-01 -3.545861e-01
50%        ...      -2.945017e-02  6.781943e-03 -1.119293e-02  4.097606e-02
75%        ...       1.863772e-01  5.285536e-01  1.476421e-01  4.395266e-01
max        ...       2.720284e+01  1.050309e+01  2.252841e+01  4.584549e+00

               V25           V26           V27           V28         Amount  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  284807.000000
mean   1.453003e-15  1.699104e-15 -3.660161e-16 -1.206049e-16      88.349619
std    5.212781e-01  4.822270e-01  4.036325e-01  3.300833e-01     250.120109
min   -1.029540e+01 -2.604551e+00 -2.256568e+01 -1.543008e+01       0.000000
25%   -3.171451e-01 -3.269839e-01 -7.083953e-02 -5.295979e-02       5.600000
50%    1.659350e-02 -5.213911e-02  1.342146e-03  1.124383e-02      22.000000
75%    3.507156e-01  2.409522e-01  9.104512e-02  7.827995e-02      77.165000
max    7.519589e+00  3.517346e+00  3.161220e+01  3.384781e+01   25691.160000

              Class
count  284807.000000
mean        0.001727
std         0.041527
min         0.000000
```

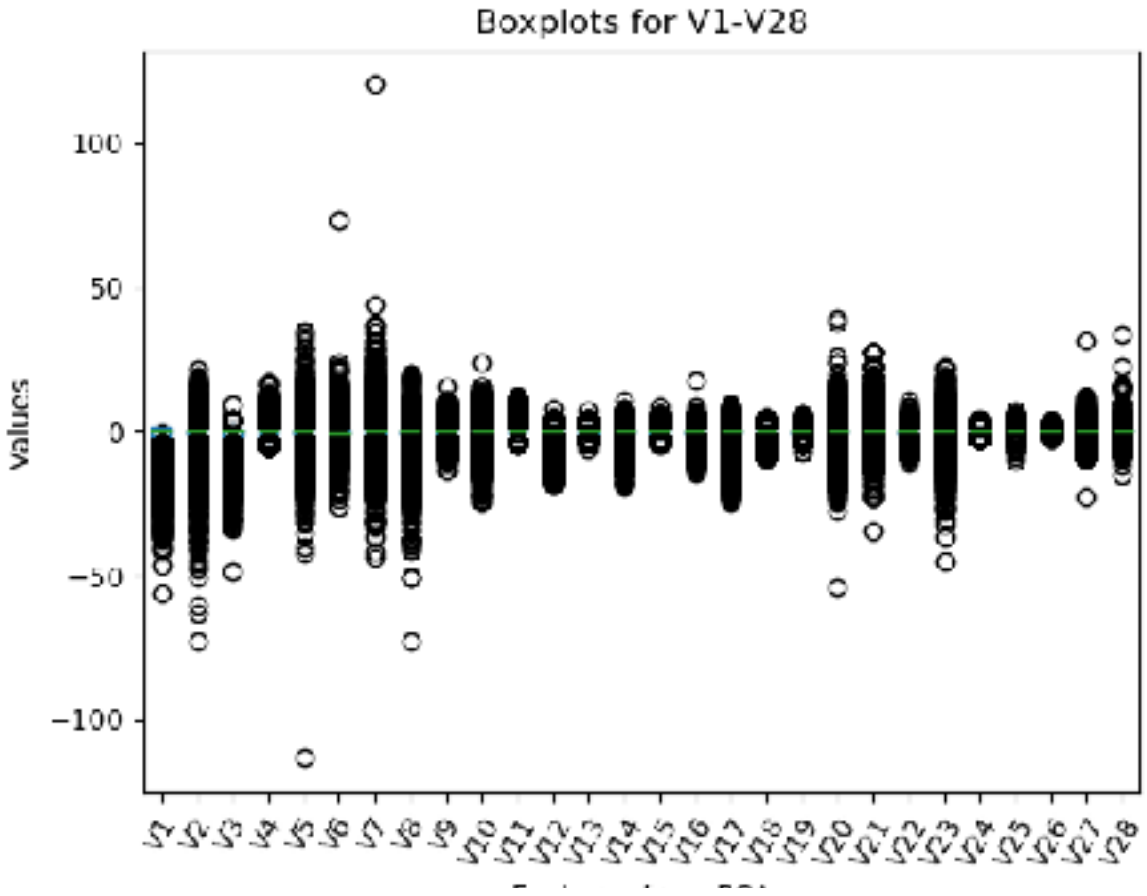
```
25%     0.000000
50%     0.000000
75%     0.000000
max     1.000000

[8 rows x 31 columns]
```

```
print(df.head(5))
   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
1   0.0  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -0.078803
2   1.0 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499  0.791461
3   1.0 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203  0.237609
4   2.0 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921  0.592941

         V8        V9  ...         V21       V22       V23       V24  \
0  0.098698  0.363787  ...   -0.018307  0.277838 -0.110474  0.066928
1  0.085102 -0.255425  ...   -0.225775 -0.638672  0.101288 -0.339846
2  0.247676 -1.514654  ...    0.247998  0.771679  0.909412 -0.689281
3  0.377436 -1.387024  ...   -0.108300  0.005274 -0.190321 -1.175575
4 -0.270533  0.817739  ...   -0.009431  0.798278 -0.137458  0.141267

        V25       V26       V27       V28  Amount  Class
0  0.128539 -0.189115  0.133558 -0.021053  149.62      0
1  0.167170  0.125895 -0.008983  0.014724    2.69      0
2 -0.327642 -0.139097 -0.055353 -0.059752  378.66      0
3  0.647376 -0.221929  0.062723  0.061458  123.50      0
4 -0.206010  0.502292  0.219422  0.215153   69.99      0

[5 rows x 31 columns]
```

```
print(df.tail(5))
        Time        V1        V2        V3        V4        V5  \
284802  172786.0 -11.881118  10.071785 -9.834783 -2.066656 -5.364473
284803  172787.0  -0.732789  -0.055080  2.035030 -0.738589  0.868229
284804  172788.0   1.919565  -0.301254 -3.249640 -0.557828  2.630515
284805  172788.0  -0.240440   0.530483  0.702510  0.689799 -0.377961
284806  172792.0  -0.533413  -0.189733  0.703337 -0.506271 -0.012546

              V6        V7        V8        V9  ...         V21       V22  \
284802 -2.606837 -4.918215  7.305334  1.914428  ...    0.213454  0.111864
284803  1.058415  0.024330  0.294869  0.584800  ...    0.214205  0.924384
284804  3.031260 -0.296827  0.708417  0.432454  ...    0.232045  0.578229
284805  0.623708 -0.686180  0.679145  0.392087  ...    0.265245  0.800049
```

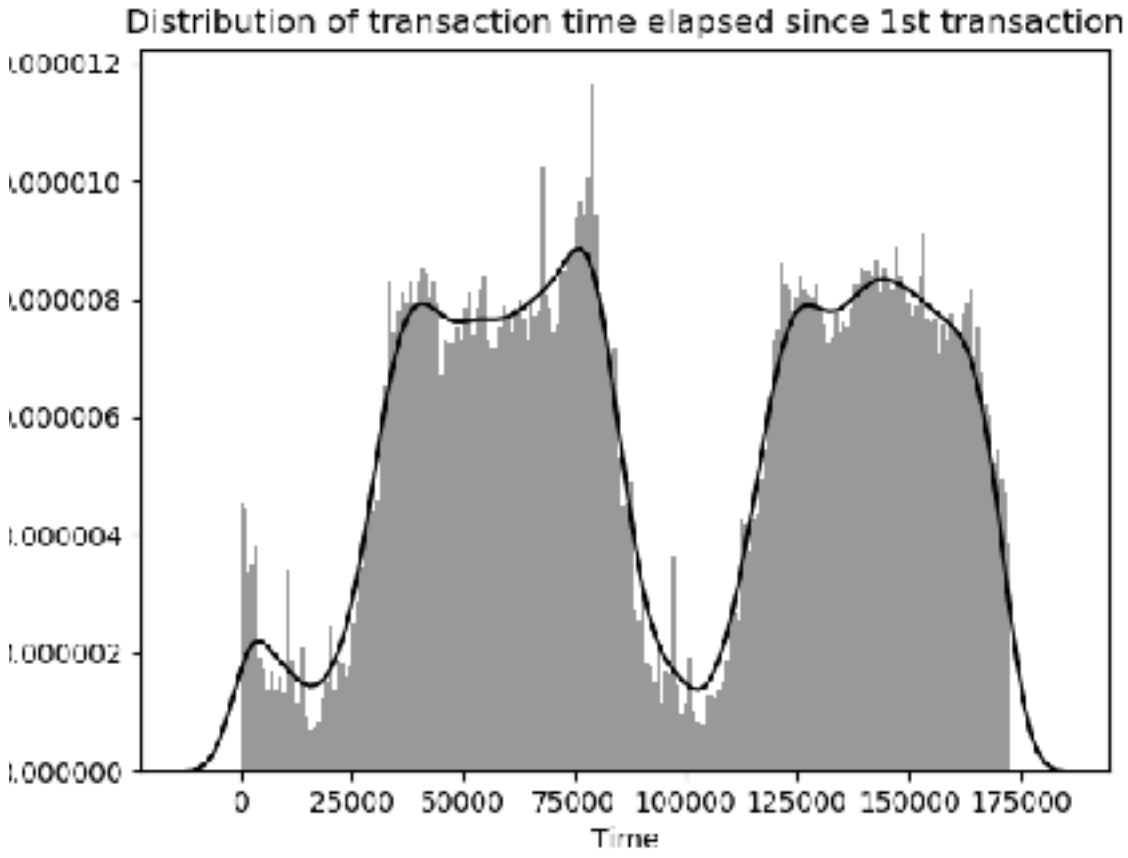
```
284806 -0.649617  1.577006 -0.414650  0.486180  ...    0.261057  0.643078

           V23       V24       V25       V26       V27       V28  Amount  \
284802  1.014480 -0.509348  1.436807  0.250034  0.943651  0.823731    0.77
284803  0.012463 -1.016226 -0.606624 -0.395255  0.068472 -0.053527   24.79
284804 -0.037501  0.640134  0.265745 -0.087371  0.004455 -0.026561   67.88
284805 -0.163298  0.123205 -0.569159  0.546668  0.108821  0.104533   10.00
284806  0.376777  0.008797 -0.473649 -0.818267 -0.002415  0.013649  217.00

        Class
284802      0
284803      0
284804      0
284805      0
284806      0

[5 rows x 31 columns]
```

Next, PCA features, ‘Time’, ‘Amount’ and ‘Class’ are examined separately.
For V1-V28 PCA features, box plots are drawn to show the distribution of each features.
The result is shown as below:



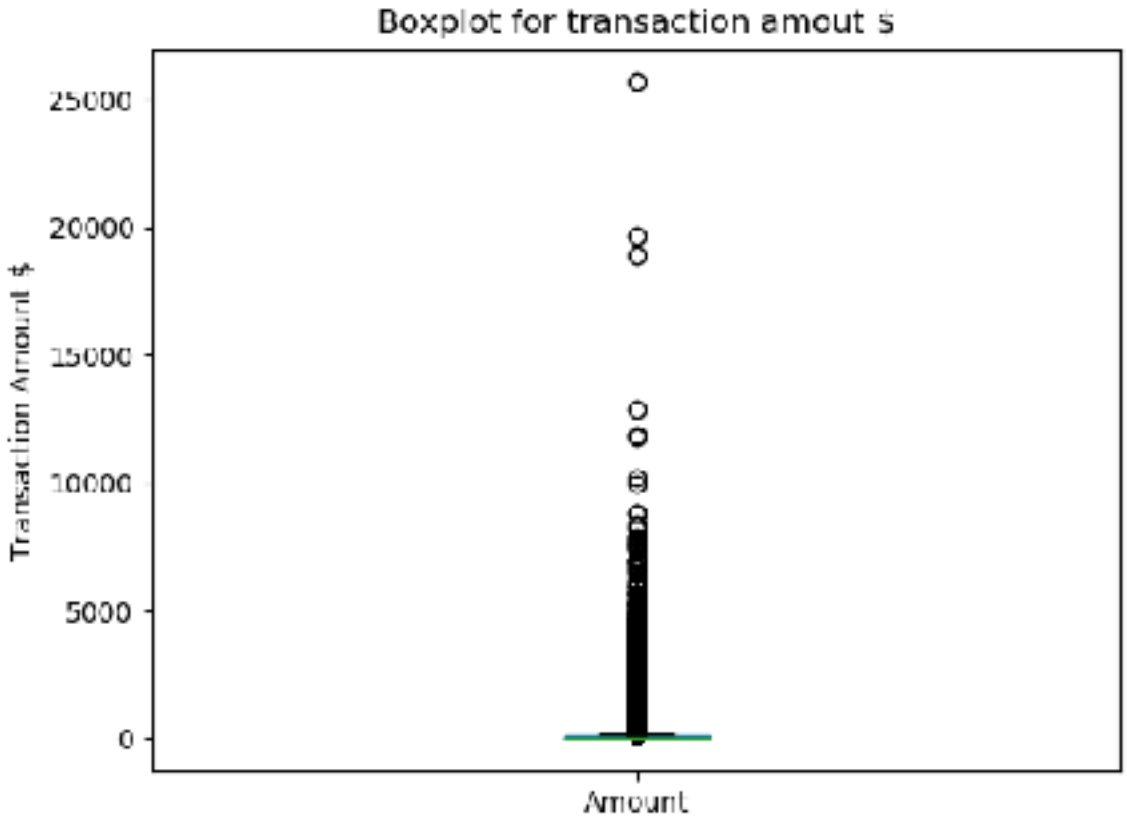
We can see from the plot that in general, these 28 features are on the same scale of within -50 to 50. Several particularly noticeable outliers exist in V5, V6, V7. Considering this is the box plot for all data including legal and fraud transactions, we will examine further later in the analysis to see where these outliers are from.

Next, we use .value_counts() on 'Time' column to confirm that Time is not unique, as the length reveals that there are only 124592 unique values. Further a plot is drawn to show the distribution of transaction times in the dataset.



(Note: Time refers to the time elapsed in seconds since the 1st transaction in the dataset)

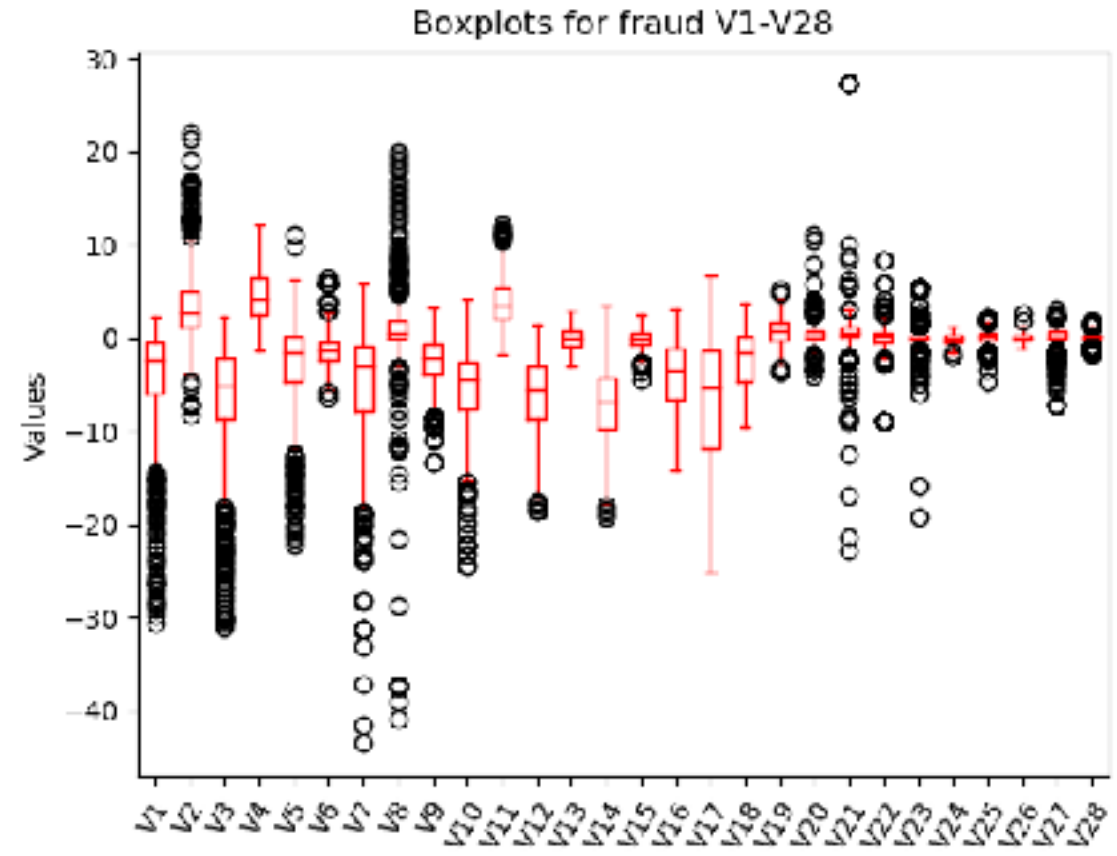
Then the transaction amount is also drawn as a box plot to show the simple stats.

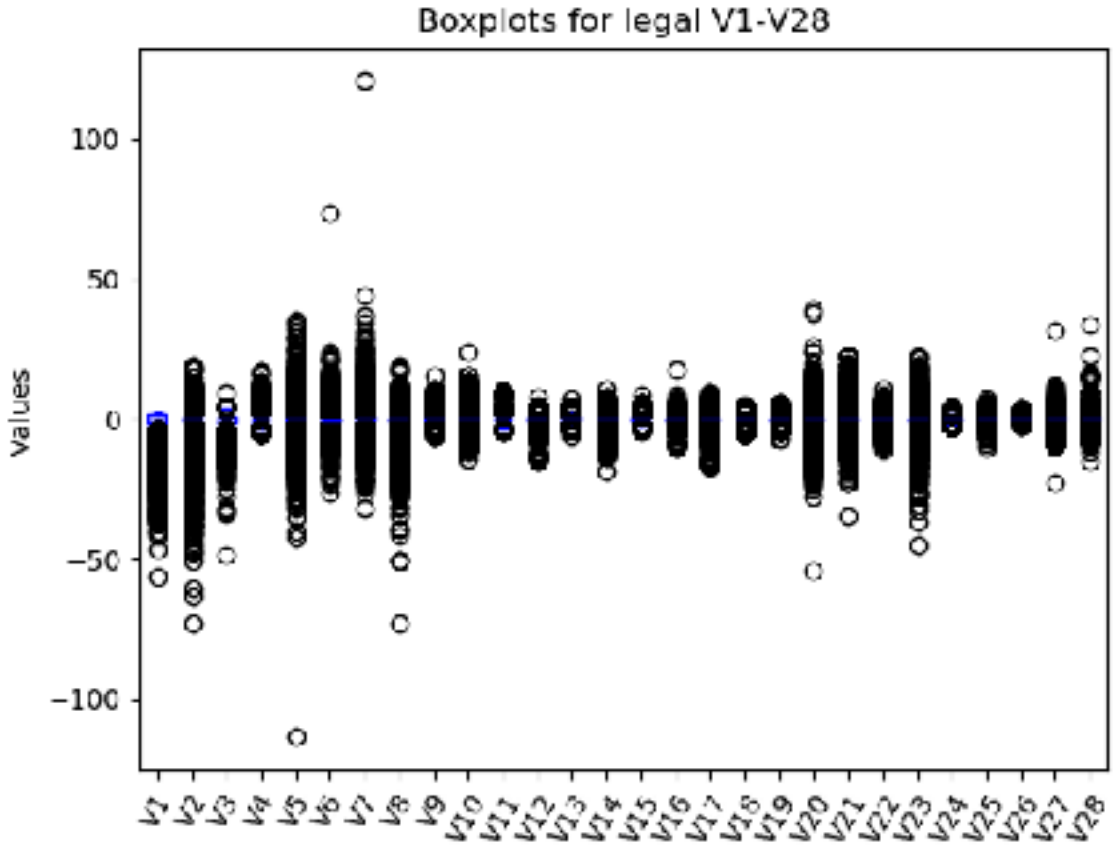


As we can see there are some outliers with really high values in this feature, and we'll examine it further in the analysis.

For 'Class', since Class == 1 denotes the fraud transaction, we can use .sum() to confirm the total number of frauds within the dataset. As it turns out, there are 492 frauds in the whole dataset, which constitutes only 0.172% of all transactions, making it a very unbalanced dataset.

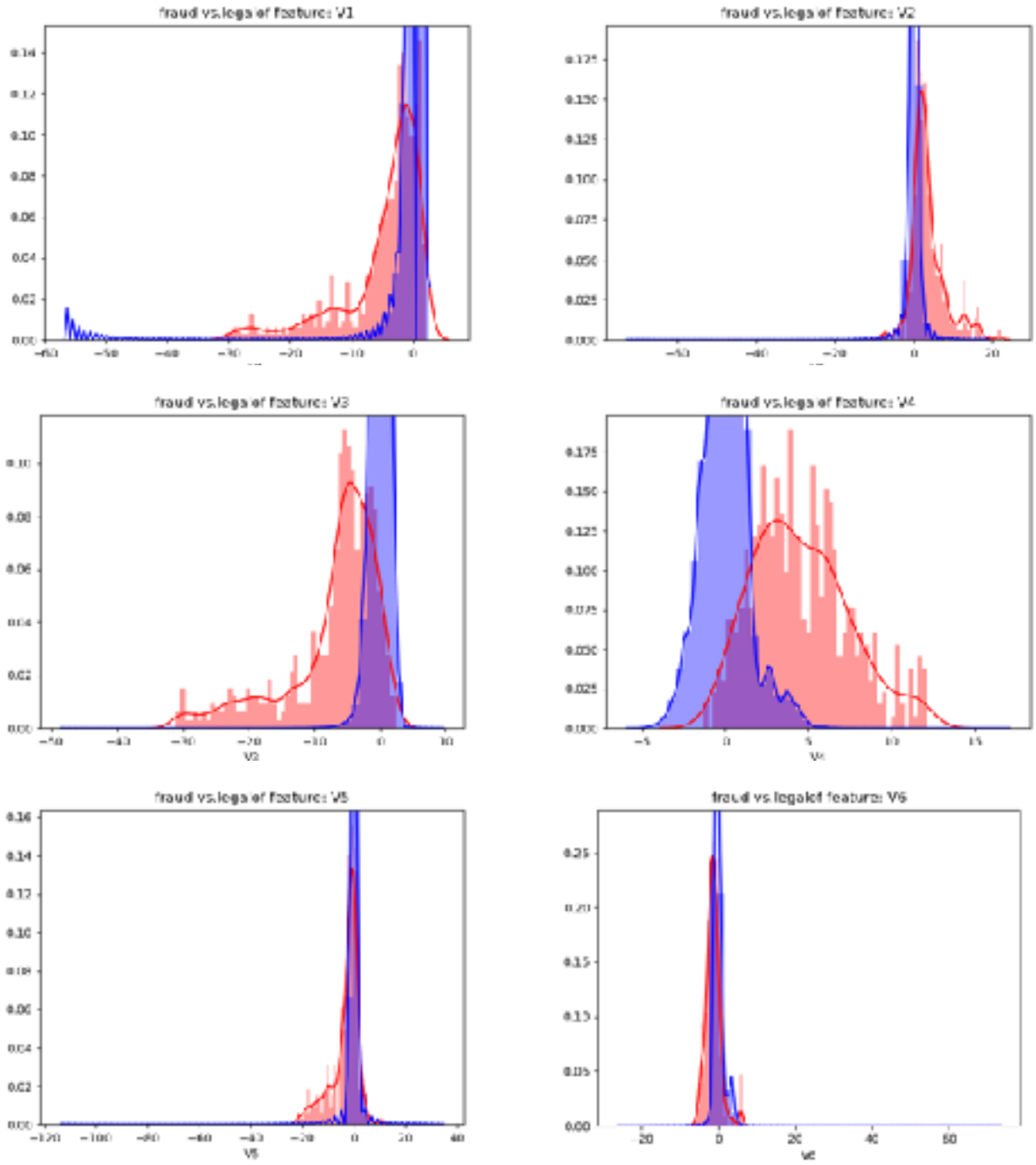
Based on 'Class' column, we can then further compare fraud and legal data sections. The box plot for V1-V12 in fraud vs. legal is shown as below:

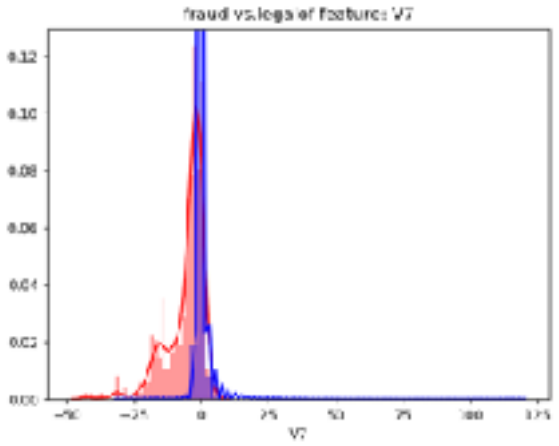

Boxplots for fraud V1-V28
Values
30
20
10
0
−10
−20
−30
−40
V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28

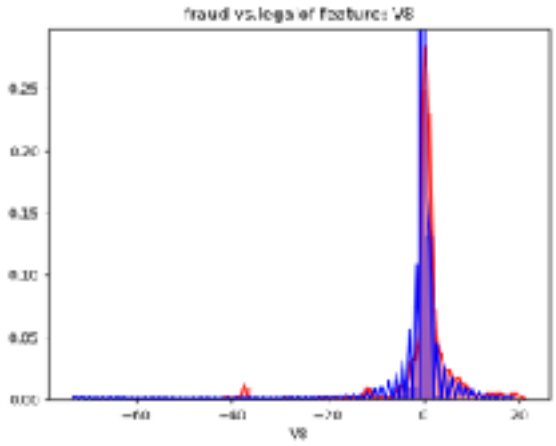

Boxplots for legal V1-V28
Values
100
50
0
−50
−100
V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28

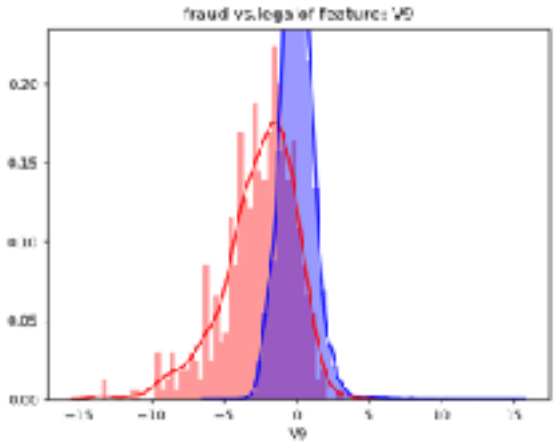
As V1-V28 of fraud transactions show relatively more variations and therefore more dispersed than those of legal transactions. However, there are still outliers in legal transactions in especially V5, V6, and V7.

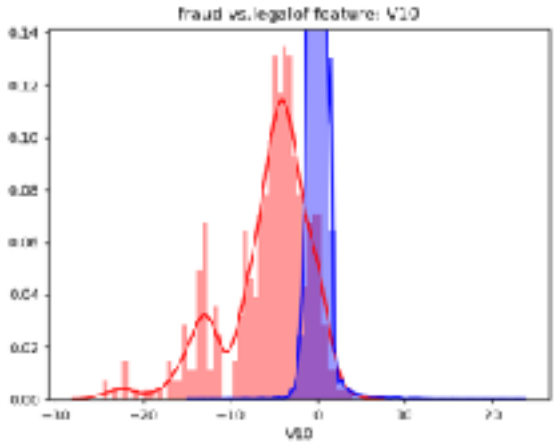
Further, the distribution plot of each fraud and legal pair for each PCA feature can be obtained as below, with red indicating ‘fraud’ and blue ‘legal’.

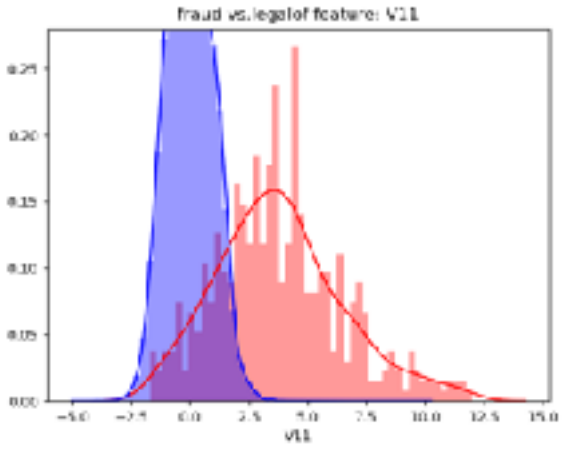


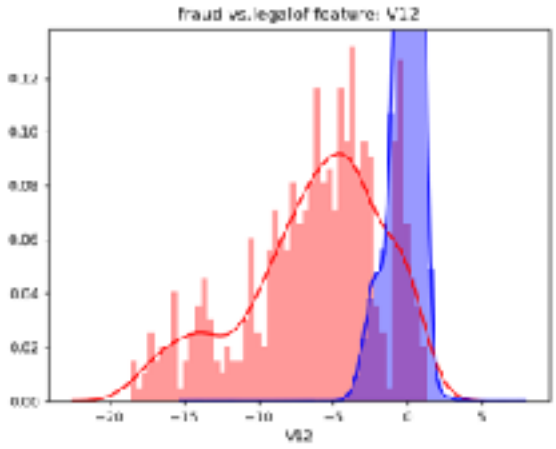

fraud vs.legal of feature: V7
V7

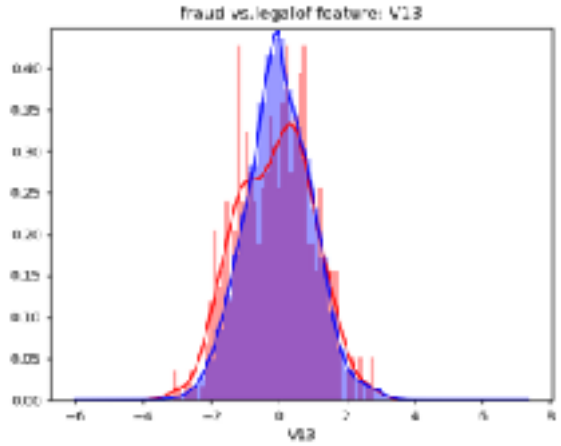

fraud vs.legal of feature: V8
V8

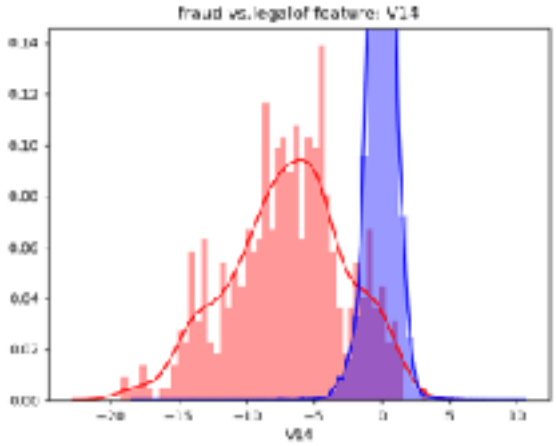

fraud vs.legal of feature: V9
V9

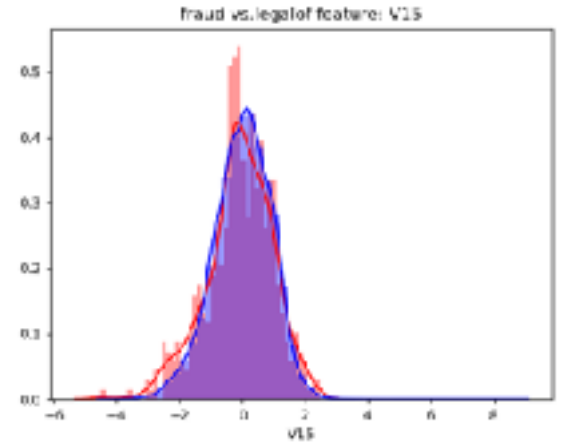

fraud vs.legal of feature: V10
V10

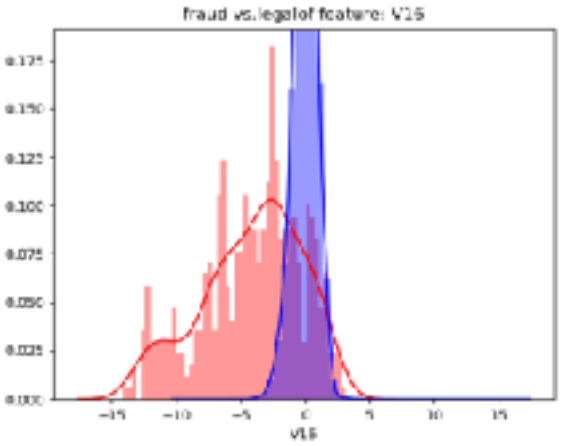

fraud vs.legal of feature: V11
V11

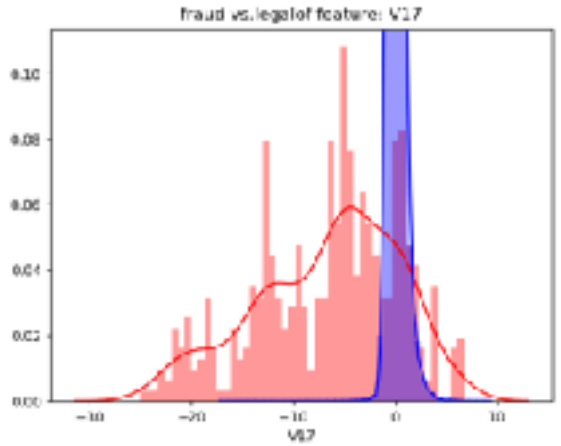

fraud vs.legal of feature: V12
V12

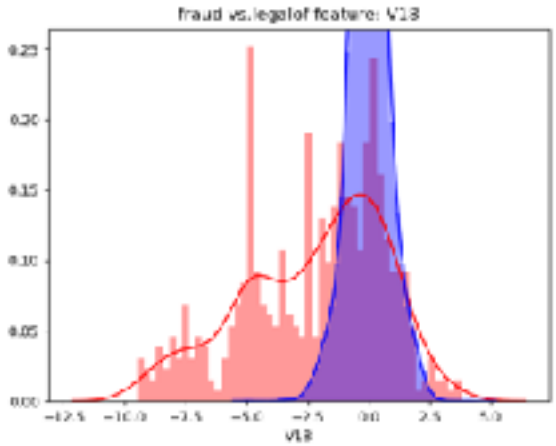

fraud vs.legalof feature: V13
V13

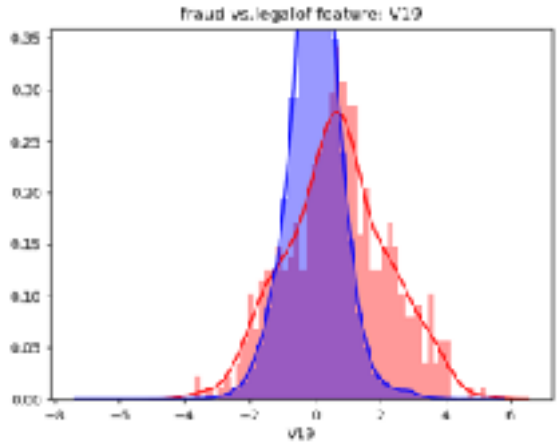

fraud vs.legalof feature: V14
V14

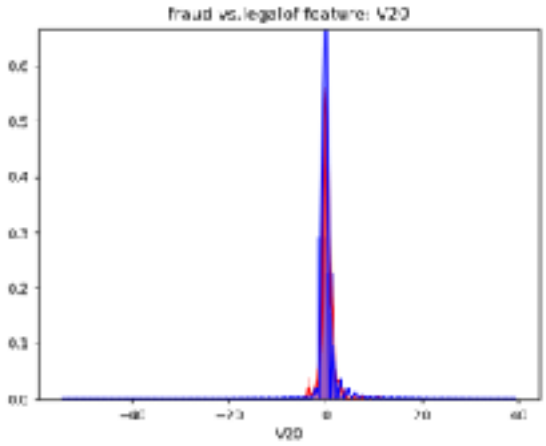

fraud vs.legalof feature: V15
V15

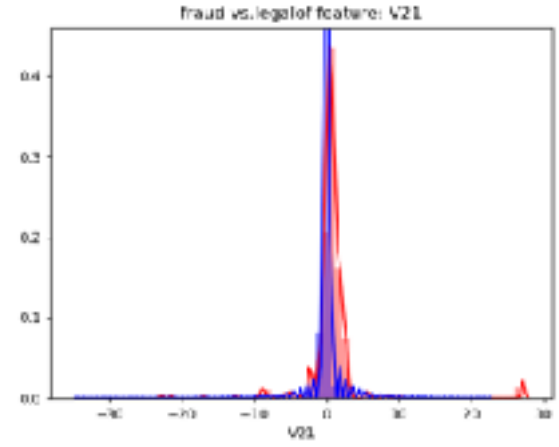

fraud vs.legalof feature: V16
V16

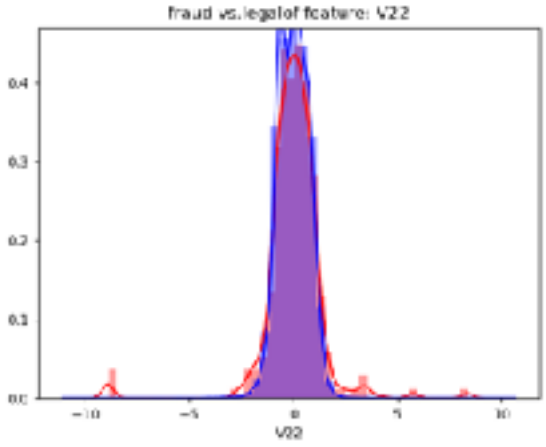

fraud vs.legalof feature: V17
V17

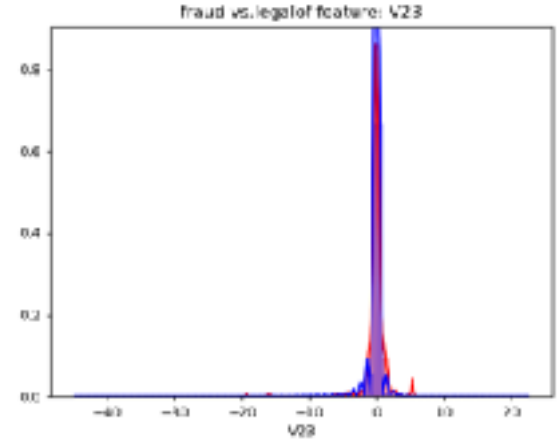

fraud vs.legalof feature: V18
V18

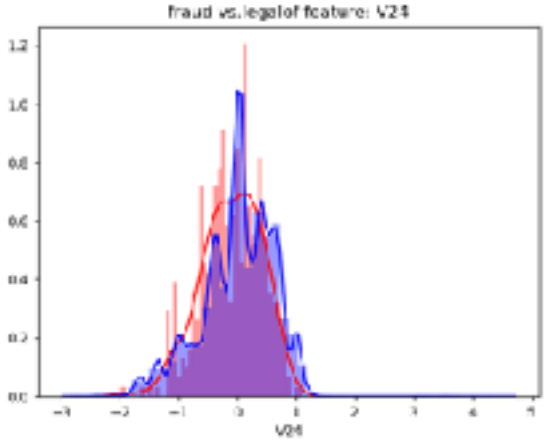

fraud vs.legalof feature: V19
V19

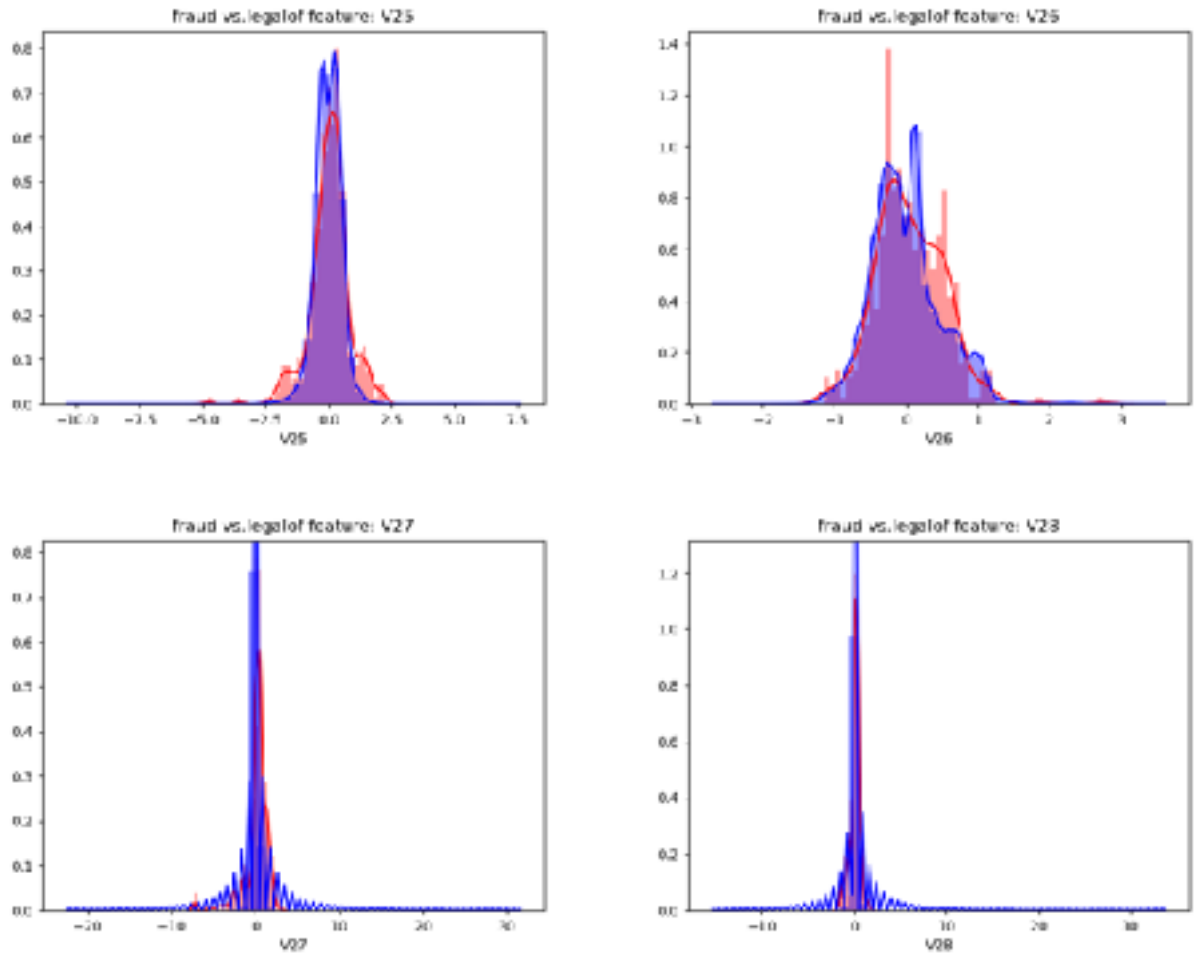

fraud vs.legalof feature: V20
V20


fraud vs.legalof feature: V21
V21

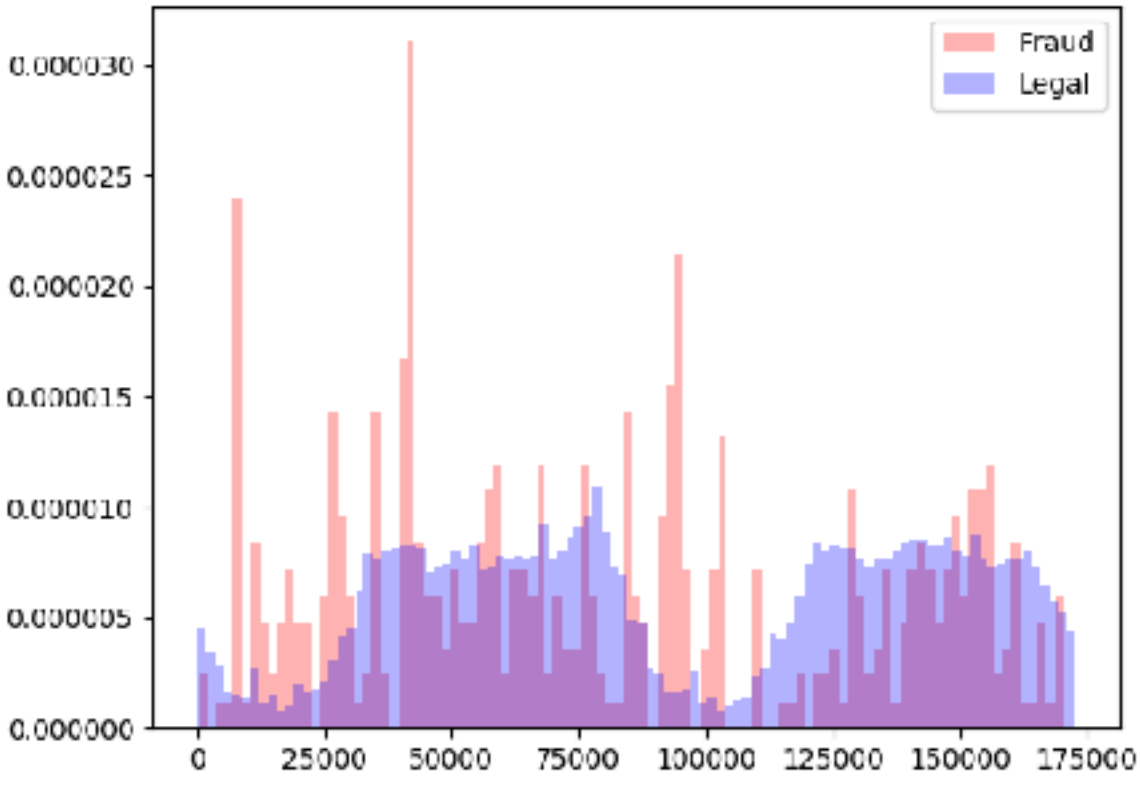

fraud vs.legalof feature: V22
V22


fraud vs.legalof feature: V23
V23


fraud vs.legalof feature: V24
V24

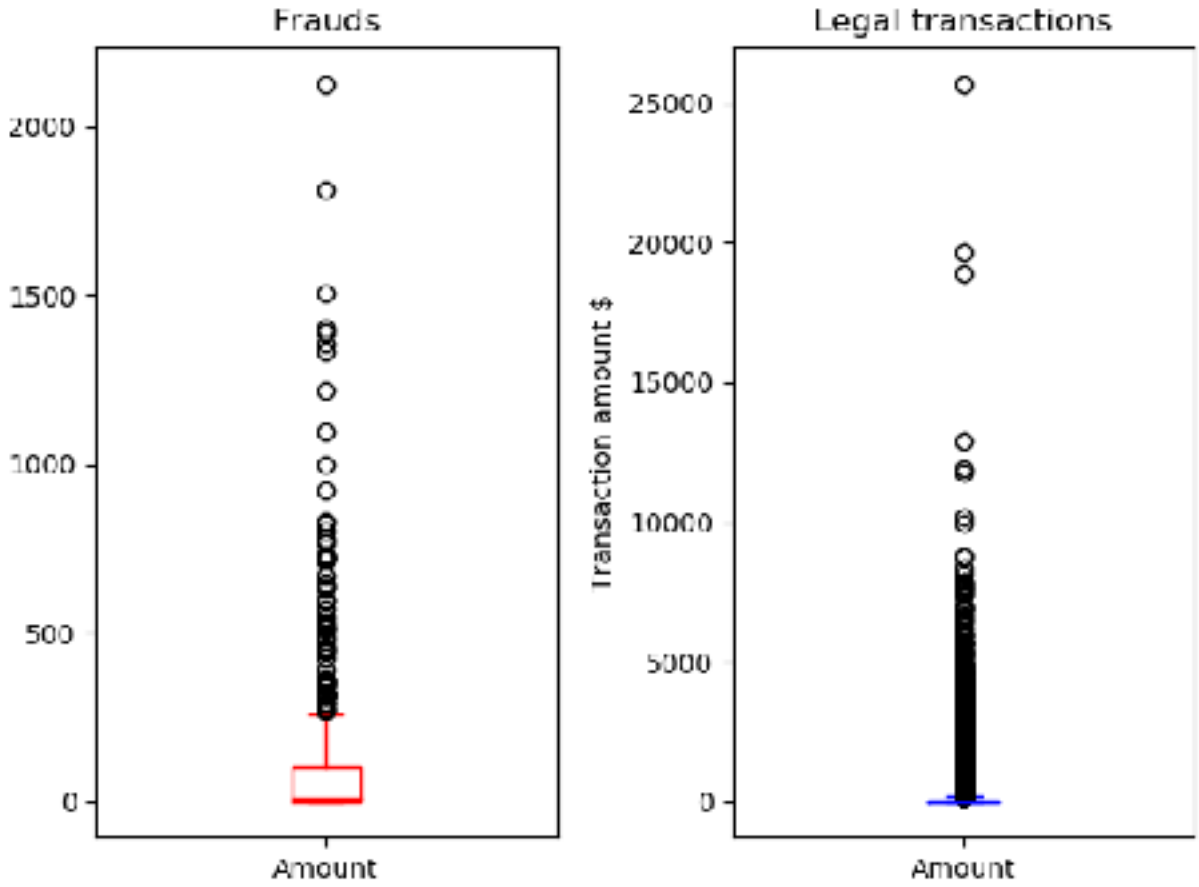


Transaction time histograms are also compared between fraud (red) and legal (blue).



Compared with time for legal transactions, fraud ones peak when the legal ones are low. We don't have detailed time of when the first transaction happened to calculate these transaction hours, but it would be logical to assume that these fraud transactions probably happen more frequently at late night and early morning for the 2 days.

Finally, we compare the transaction amounts for fraud and legal transactions.



Fraud transactions are all over 0 (which is logical), and dispersed throughout around 250 to 2000. And even though legal transactions have some extremely large values, the fraud transactions has a higher average (122 vs. 88).

After initial exploratory analysis, I think it might be reasonable to include V1-V28 PCA features, amount, (and time?) in the prediction model.

3. Questions and future actions

There are several questions need to be addressed in the project based on the analysis so far:

1. Since the dataset is very unbalanced (majority are legal, and only 0.172% is fraud), are there any techniques I need to employ for training the dataset, or does it depend on specific model I choose for the problem?
   From what I've learnt, there are following techniques for unbalanced classes, including (2):

- Up-sample minority class
- Down-sample majority class
- Change the performance metrics
- Penalize algorithms
- Use tree-based algorithms

   For the metrics part, it's easy to understand that 'accuracy' in the traditional confusion matrix isn't too meaningful for unbalanced dataset, and therefore I will consider using Area Under the Precision-Recall Curve (AUPRC) to evaluate the model performances. Also other metrics like: fraud detection rate, false alarm rate, balanced classification rate and Matthews correlation coefficient, could be used.(3)

2. Based on relevant literature, people have been experimenting and obtaining good results with SVM, NB, random forests-based classifiers, and neural network. I think I will probably choose two or three among these to see which will give the best performance.

3. I've created a new column that transforms 'Time' column to a datetime object of format ' xx (days): xx(hour): xx(min):xx(sec)'. Since time is not unique, it probably won't be optimal to use time as the column index. I'm not quite sure whether it's necessary to include the time/datetime object in later analysis at all? Should it be treated as a variable? Or should I use it in some other way?

References

(1) Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson and Gianluca Bontempi. Calibrating Probability with Undersampling for Unbalanced Classification. In Symposium on Computational Intelligence and Data Mining (CIDM), IEEE, 2015
(2) https://elitedatascience.com/imbalanced-classes
(3) http://damienfrancois.be/blog/files/modelperfcheatsheet.pdf

Appendix

```
#1.Import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

#2.Import data
fn='creditcard.csv'
df = pd.read_csv(fn)

#3.1 Summary stats for the entire dataset
print(df.info())
print(df.shape)
print(df.columns)
print(df.head(5))
print(df.tail(5))
print(df.describe())

#By different categories: PCA features, Time, Amount, and Class(fraud or not)
#Visual EDA, PCA features plot
df_pca = df.iloc[:,1:29]
df_pca.plot(kind='Box')
plt.xticks(rotation=60)
plt.title('Boxplots for V1-V28')
plt.xlabel('Features from PCA')
plt.ylabel('Values')
plt.show()
plt.savefig('Boxplots for V1-V28.png')

#EDA and visual EDA for time
print(df['Time'].describe())
print(len(df['Time'].value_counts()))
plt.figure()
sns.distplot(df['Time'], bins = 200, norm_hist = True, color = 'black')
plt.title('Distribution of transaction time elapsed since 1st transaction')
plt.show()
plt.savefig('time.png')

#Convert time into datetime obejct, is it necessary?
#Also it seems that python crashed easily here
from datetime import datetime, timedelta
def GetTime(time):
    sec = timedelta(seconds=time)
    d = datetime(1,1,1) + sec
    #("DAYS:HOURS:MIN:SEC")
    return d
    #return "%d:%d:%d:%d" % (d.day-1, d.hour, d.minute, d.second)
df_newtime = [GetTime(time) for ind, time in df['Time'].iteritems()]
df['Newtime'] = df_newtime

#EDA and visual EDA for amount
```

```
print(df['Amount'].describe())
plt.figure()
df.plot(y='Amount', kind='box')
plt.title('Boxplot for transaction amout $')
plt.ylabel('Transaction Amount $')
plt.show()
plt.savefig('Amount_box.png')

#Class ==1:fraud. Confirm the number of frauds in the dataset.(Expected: Inbalanced)
fnum = df['Class'].sum()
fperc = float(fnum)/len(df['Class'])
print(fperc)


#3.2 EDA comparing fraud vs. legal transactions
frauds=df[df['Class']==1]
legals=df[df['Class']==0]

#V1-V28, boxplots for fraud and legal transactions
frauds.iloc[:,1:29].plot(kind='Box', color = 'red')
plt.xticks(rotation=60)
plt.title('Boxplots for fraud V1-V28')
plt.xlabel('Features from PCA')
plt.ylabel('Values')
plt.show()
plt.savefig('Boxplots for fraud V1-V28.png')
legals.iloc[:,1:29].plot(kind='Box', color = 'blue')
plt.xticks(rotation=60)
plt.title('Boxplots for legal V1-V28')
plt.xlabel('Features from PCA')
plt.ylabel('Values')
plt.show()
plt.savefig('Boxplots for legal V1-V28.png')
#Iterate through the 28 features and compare distibution for each fraud&legal pair
pca_names = df_pca.columns
print(pca_names)
for i, V in enumerate(df[pca_names]):
    plt.figure(i+1)
    sns.distplot(frauds[V], bins = 50, norm_hist = True, color = 'red')
    sns.distplot(legals[V], bins = 50, norm_hist = True, color = 'blue')
    plt.title('Distribution of fraud vs.legal:feature: ' + str(V))
    plt.show()
    plt.savefig(str(V)+'.png')
    plt.close()

#Time, distribution plots for fraud vs. legal
plt.figure()
frauds.Time.plot(kind='hist', bins=100, color = 'red', normed= True,label= 'Fraud', alpha = 0.3)
legals.Time.plot(kind='hist', bins=100, color = 'blue', normed= True, label= 'Legal',alpha = 0.3)
plt.legend(loc='upper right')
plt.title('Distribution of transaction time elapsed since 1st transaction: Fraud vs. Legal')
plt.show()
plt.savefig('Fraud vs. Legal Transaction Time.png')
```

```
#Amount,summary stats and plots for fraud vs. legal
print(frauds.Amount.describe())
print(legals.Amount.describe())
plt.figure()
plt.subplot(1,2,1)
frauds.Amount.plot(kind='Box',color = 'red')
plt.title('Frauds')
plt.subplot(1,2,2)
legals.Amount.plot(kind='Box',color = 'blue')
plt.title('Legal transactions')
plt.ylabel('Transaction amount $')
plt.tight_layout()
plt.show()
plt.savefig('Frauds vs. Legal.png')

#4.Prepare Data for Training
```