Linear Algebra (Credit card fraud detection)

Please download the data from https://www.kaggle.com/dalpozz/creditcardfraud/data

Info about data: it is a CSV file, contains 31 features, the last feature is used to classify the transaction whether it is a fraud or not

Task 1: please do proper analysis of the whole data, plot all relevant plots, note down all observations.

Task 2: Let's define a matric

similarity(i,j) = dot product (vi, vj) / length(vi) * length(vj) Take out any sample from the data set which contains no less than 100 transactions, for every transaction in the sample find out top 10 transactions in the dataset which have the highest similarity(i,j).

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

url = "creditcard.csv"

creditcard = pd.read_csv(url)

creditcard
```

...	...	...	...	...	...	...	...	
284777	172764.0	2.079137	-0.028723	-1.343392	0.358000	-0.045791	-1.345452	0.22
284778	172764.0	-0.764523	0.588379	-0.907599	-0.418847	0.901528	-0.760802	0.75
284779	172766.0	1.975178	-0.616244	-2.628295	-0.406246	2.327804	3.664740	-0.53
284780	172766.0	-1.727503	1.108356	2.219561	1.148583	-0.884199	0.793083	-0.52
284781	172766.0	-1.139015	-0.155510	1.894478	-1.138957	1.451777	0.093598	0.19
284782	172767.0	-0.268061	2.540315	-1.400915	4.846661	0.639105	0.186479	-0.04
284783	172768.0	-1.796092	1.929178	-2.828417	-1.689844	2.199572	3.123732	-0.27
284784	172768.0	-0.669662	0.923769	-1.543167	-1.560729	2.833960	3.240843	0.18
284785	172768.0	0.032887	0.545338	-1.185844	-1.729828	2.932315	3.401529	0.33
284786	172768.0	-2.076175	2.142238	-2.522704	-1.888063	1.982785	3.732950	-1.21
284787	172769.0	-1.029719	-1.110670	-0.636179	-0.840816	2.424360	-2.956733	0.28
284788	172770.0	2.007418	-0.280235	-0.208113	0.335261	-0.715798	-0.751373	-0.45
284789	172770.0	-0.446951	1.302212	-0.168583	0.981577	0.578957	-0.605641	1.25
284790	172771.0	-0.515513	0.971950	-1.014580	-0.677037	0.912430	-0.316187	0.39
284791	172774.0	-0.863506	0.874701	0.420358	-0.530365	0.356561	-1.046238	0.75
284792	172774.0	-0.724123	1.485216	-1.132218	-0.607190	0.709499	-0.482638	0.54
284793	172775.0	1.971002	-0.699067	-1.697541	-0.617643	1.718797	3.911336	-1.25
284794	172777.0	-1.266580	-0.400461	0.956221	-0.723919	1.531993	-1.788600	0.31
284795	172778.0	-12.516732	10.187818	-8.476671	-2.510473	-4.586669	-1.394465	-3.63
284796	172780.0	1.884849	-0.143540	-0.999943	1.506772	-0.035300	-0.613638	0.19
284797	172782.0	-0.241923	0.712247	0.399806	-0.463406	0.244531	-1.343668	0.92
284798	172782.0	0.219529	0.881246	-0.635891	0.960928	-0.152971	-1.014307	0.42
284799	172783.0	-1.775135	-0.004235	1.189786	0.331096	1.196063	5.519980	-1.51
284800	172784.0	2.039560	-0.175233	-1.196825	0.234580	-0.008713	-0.726571	0.01
284801	172785.0	0.120316	0.931005	-0.546012	-0.745097	1.130314	-0.235973	0.81
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.91
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.02
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.29
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.68
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.57

284807 rows × 31 columns

Information about data set

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

It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, ... V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-senstive learning. **Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.**

Given the class imbalance ratio, we recommend measuring the accuracy using the Area Under the Precision-Recall Curve (AUPRC). Confusion matrix accuracy is not meaningful for unbalanced classification.

The dataset has been collected and analysed during a research collaboration of Worldline and the Machine Learning Group (http://mlg.ulb.ac.be) of ULB (Université Libre de Bruxelles) on big data mining and fraud detection.

```
print(creditcard.shape)
```

```
(284807, 31)
```

```
print(creditcard.columns)
```

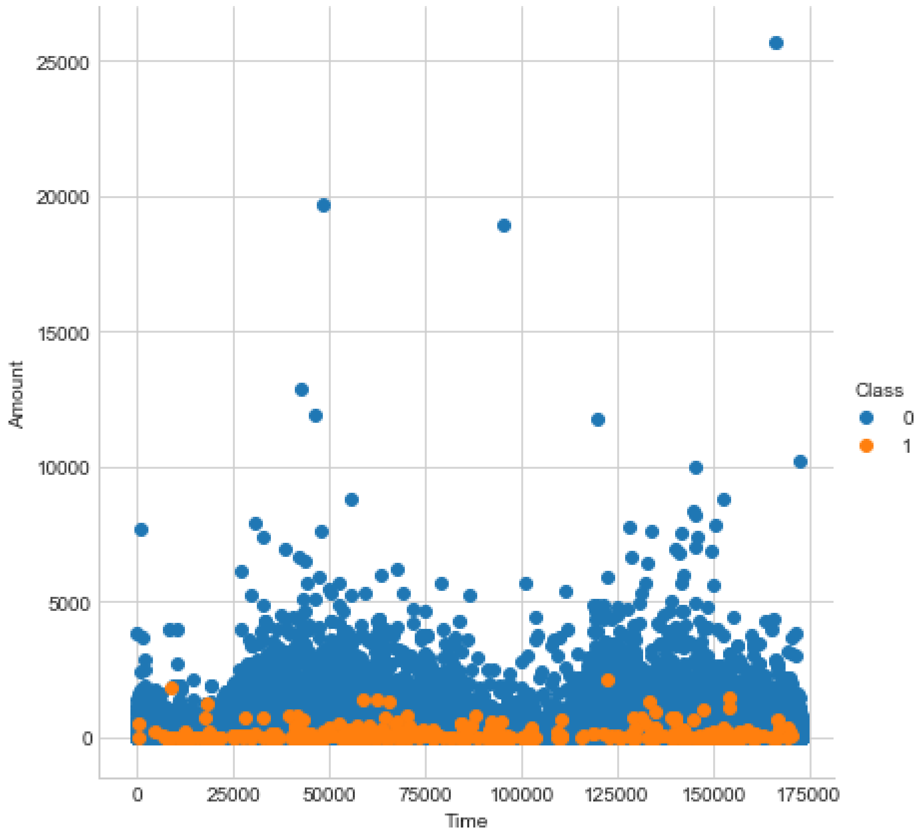
```
Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
       'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
       'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',
       'Class'],
      dtype='object')
```

```
creditcard["Class"].value_counts()
```

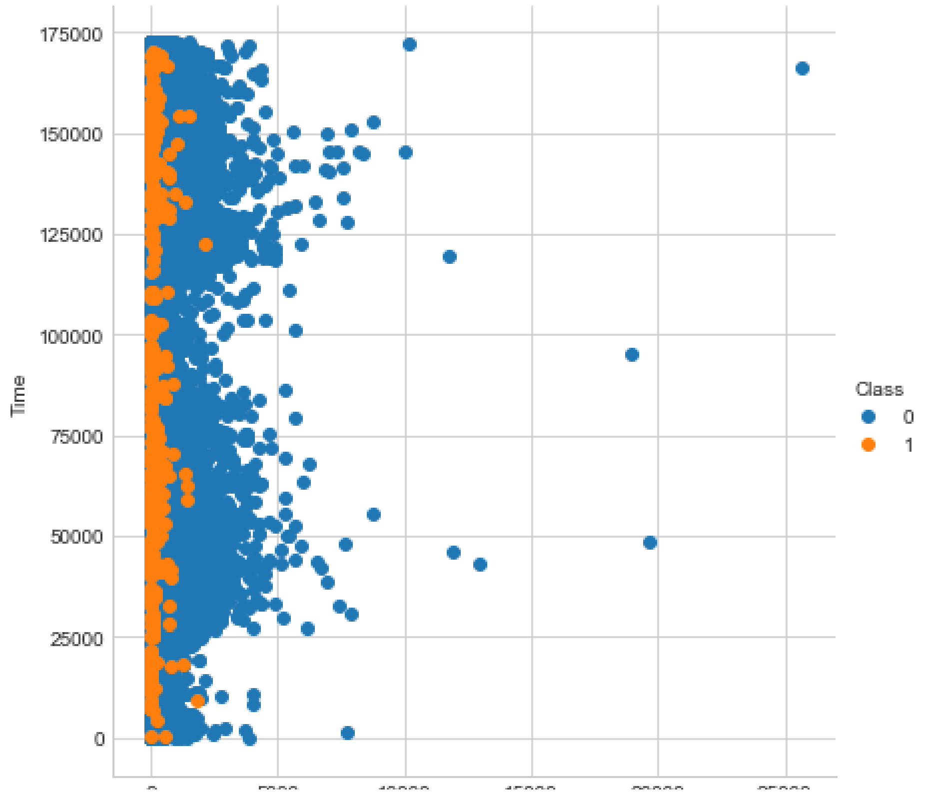
```
0    284315
1       492
Name: Class, dtype: int64
```

2-D Scatter Plot

```
sns.set_style("whitegrid")
sns.FacetGrid(creditcard, hue="Class", size = 6).map(plt.scatter, "Time", "Amount").add_le
plt.show()
```




```
sns.set_style("whitegrid")
sns.FacetGrid(creditcard, hue="Class", size = 6).map(plt.scatter, "Amount", "Time").add_le
plt.show()
```



Observations:

1. From the above two plots it is clearly visible that there are frauds only on the transactions which have transaction amount approximately less than 2500. Transactions which have transaction amount approximately above 2500 have no fraud.
2. As per with the time, the frauds in the transactions are evenly distributed throughout time.

3D Scatter plot

```
FilteredData = creditcard[['Time','Amount', 'Class']]
```

```
FilteredData
```

	Time	Amount	Class
0	0.0	149.62	0
1	0.0	2.69	0
2	1.0	378.66	0
3	1.0	123.50	0
4	2.0	69.99	0
5	2.0	3.67	0
6	4.0	4.99	0
7	7.0	40.80	0
8	7.0	93.20	0
9	9.0	3.68	0
10	10.0	7.80	0
11	10.0	9.99	0
12	10.0	121.50	0
13	11.0	27.50	0
14	12.0	58.80	0
15	12.0	15.99	0
16	12.0	12.99	0
17	13.0	0.89	0
18	14.0	46.80	0
19	15.0	5.00	0
20	16.0	231.71	0
21	17.0	34.09	0
22	18.0	2.28	0
23	18.0	22.75	0
24	22.0	0.89	0
25	22.0	26.43	0
26	23.0	41.88	0
27	23.0	16.00	0
28	23.0	33.00	0
29	23.0	12.99	0
...	...	...	...
284777	172764.0	1.00	0
284778	172764.0	80.00	0

284778	172764.0	80.00	0
284779	172766.0	25.00	0
284780	172766.0	30.00	0
284781	172766.0	13.00	0
284782	172767.0	12.82	0
284783	172768.0	11.46	0
284784	172768.0	40.00	0
284785	172768.0	1.79	0
284786	172768.0	8.95	0
284787	172769.0	9.99	0
284788	172770.0	3.99	0
284789	172770.0	60.50	0
284790	172771.0	9.81	0
284791	172774.0	20.32	0
284792	172774.0	3.99	0
284793	172775.0	4.99	0
284794	172777.0	0.89	0
284795	172778.0	9.87	0
284796	172780.0	60.00	0
284797	172782.0	5.49	0
284798	172782.0	24.05	0

```
print(FilteredData.shape)
```

```
(284807, 3)
```

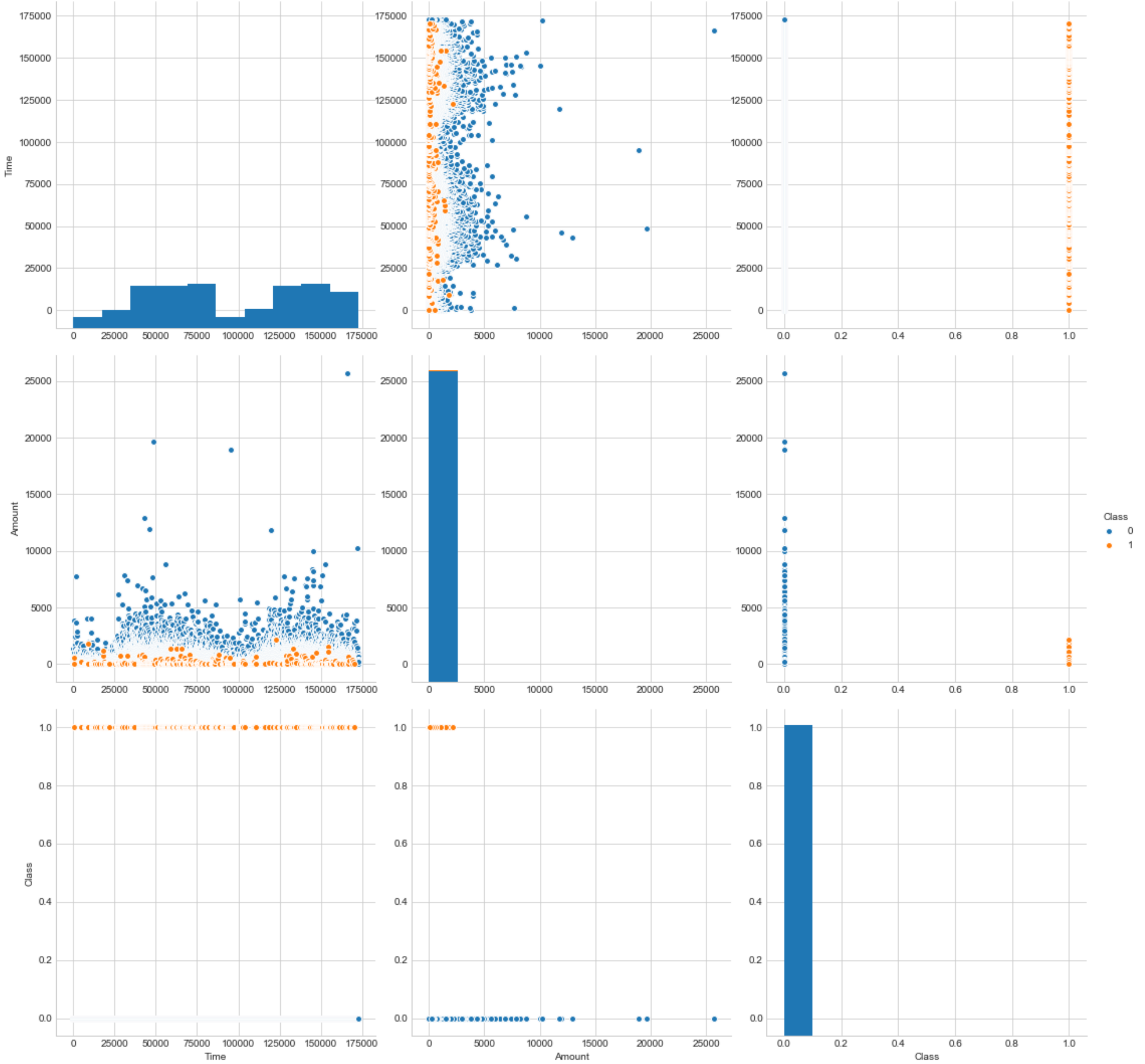
284801	172785.0	2.69	0

```
FilteredData["Class"].value_counts()
```

```
0    284315
1       492
Name: Class, dtype: int64
```

284805	172788.0	10.00	0

```
plt.close();
sns.set_style("whitegrid");
sns.pairplot(FilteredData, hue="Class", size=5);
plt.show()
```




```
countLess = 0
countMore= 0
for i in range(284806):
```

```
for i in range(284806):
    if(FilteredData.iloc[i]["Amount"] < 2500):
        countLess = countLess + 1
    else:
        countMore = countMore + 1
print(countLess)
print(countMore)
```

```
284357
449
```

```
percentage = (countLess/284807)*100
percentage
```

```
99.84199826549207
```

Observations:

Now it has been calculated that there are 284357 transactions which has a transaction amount less than 2500. Means 99.84% of transactions have transaction amount less than 2500

```
class0 = 0
class1 = 0
for i in range(284806):
    if(FilteredData.iloc[i]["Amount"] < 2500):
        if(FilteredData.iloc[i]["Class"] == 0):
            class0 = class0 + 1
        else:
            class1 = class1 + 1

print(class0)
print(class1)
```

```
283865
492
```

```
FilteredData["Class"].value_counts()
```

```
0    284315
1       492
Name: Class, dtype: int64
```

Observations:

Now the total number of fraud transactions in whole data are 492. It has been calculated that total number of fraud transactions in data where transaction amount is less than 2500 is also 492. Therefore, all 100% fraud transactions have transaction amount less than 2500 and there is no fraud transaction where transaction amount is more than 2500.

Histogram, PDF and CDF

↳ 10 cells hidden

Mean, Variance and Std-dev

```
print("Means:")
print("Mean of transaction amount of genuine transactions: ",np.mean(creditCard_genuine["A
print("Mean of transaction amount of fraud transactions: ",np.mean(creditCard_fraud["Amoun
```

```
Means:
Mean of transaction amount of genuine transactions:  88.29102242225574
Mean of transaction amount of fraud transactions:  122.21132113821133
```

```
print("Standard Deviation:")
print("Std-Deviation of transaction amount of genuine transactions: ", np.std(creditCard_g
print("Std-Deviation of transaction amount of fraud transactions: ", np.std(creditCard_fra
```

```
Standard Deviation:
Std-Deviation of transaction amount of genuine transactions:  250.1046523874637
Std-Deviation of transaction amount of fraud transactions:  256.42229861324483
```

```
print("Median:")
print("Median of transaction amount of genuine transactions: ", np.median(creditCard_genui
print("Median of transaction amount of fraud transactions: ", np.median(creditCard_fraud["
```

```
Median:
Median of transaction amount of genuine transactions:  22.0
Median of transaction amount of fraud transactions:  9.25
```

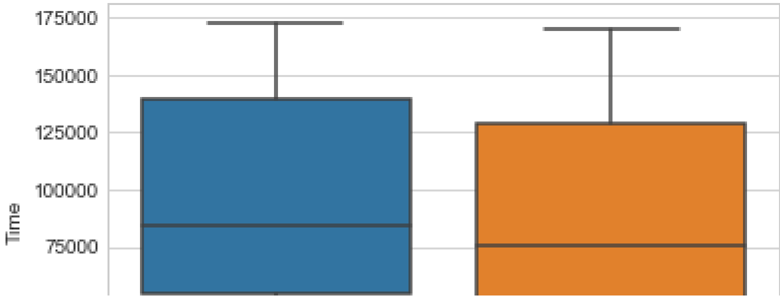
```
print("\nQuantiles:")
print(np.percentile(creditCard_genuine["Amount"],np.arange(0, 100, 25)))
print(np.percentile(creditCard_fraud["Amount"],np.arange(0, 100, 25)))
```

```

Quantiles:
[ 0.    5.65 22.   77.05]
[  0.     1.     9.25 105.89]
```

Box plot and Whiskers

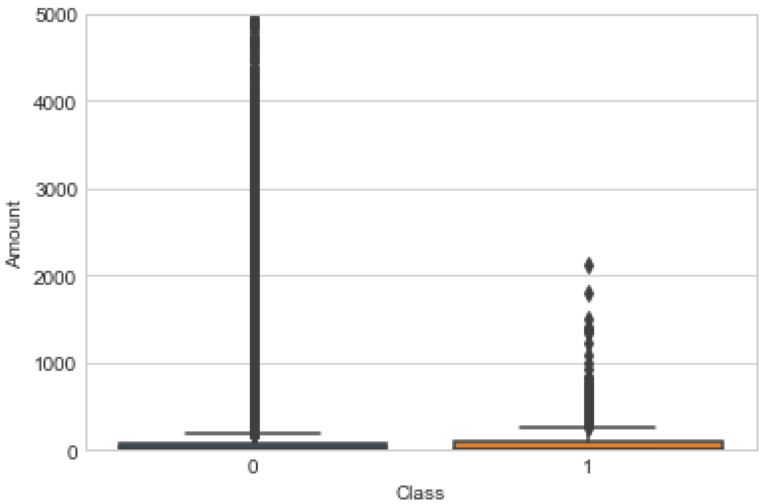
```
sns.boxplot(x = "Class", y = "Time", data = creditcard)
plt.show()
```



Observations:

By looking at the above box plot we can say that both fraud & genuine transactions occur throughout time and there is no distinction between them.

```
sns.boxplot(x = "Class", y = "Amount", data = creditcard)
plt.ylim(0, 5000)
plt.show()
```



Observations:

From above box plot we can easily infer that there are no fraud transactions occur above the transaction amount of 3000. All of the fraud transactions have transaction amount less than 3000. However, there are many transactions which have a transaction amount greater than 3000 and all of them are genuine.

Similarity

```
from scipy import spatial
```

```
sampleData = creditcard.head(20000)   #Sample the data from original data so as to save th
```

```
samples = creditcard.loc[30401:30500]    #Taking sample of size 100 from index 30401 to 30
```