Lea Setruk 345226179
Aviva Shneor Simchon 317766731

UNSUPERVISED LEARNING

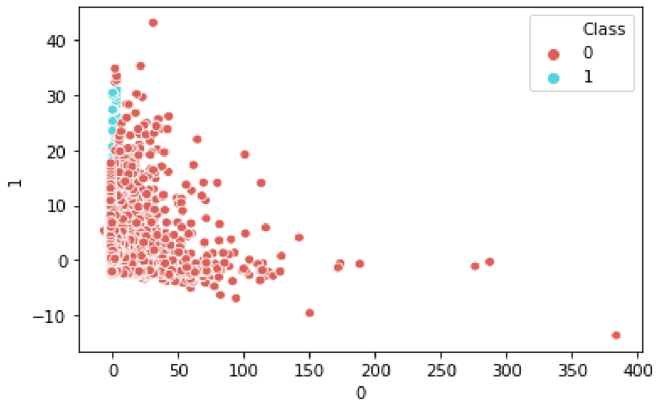
Dataset

We chose to work with the **credit card** dataset. Its purpose is to recognize fraudulent credit card transactions.
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.

Data Analysis

Visiualization of the data after PCA dimension reduction



We can see that the data is very unbalanced.

We want to see what are the differences between the normal transactions and the fraud ones. We compare them for every variables. We checked their distributions, their mean, standard deviation, max, min etc…

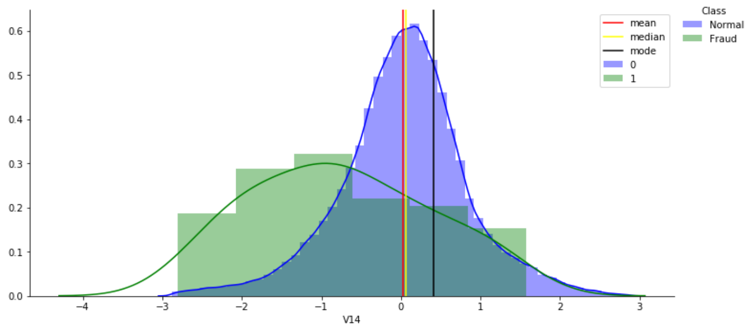
To visualise the data in the graphs, we eliminated the outliers that way : we want to see only the absolute value of the data between mean + 3 std to still get the information.

We did this observations for every variables. We found differences between the distributions of the fraud and the normal data for those variables : V1, V2, V3, V4, V5, V7, V9, V10, V11, V12, V14, V16, V17, V18. So, we chose to work with those features for our algorithms.

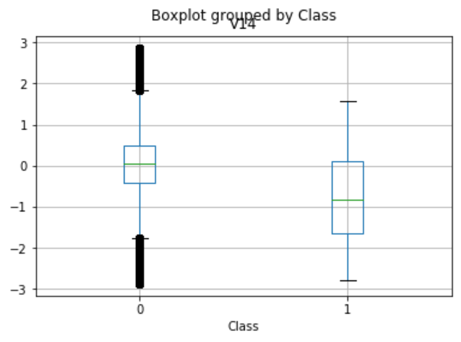
We would have liked to talk more about those different variables and their influence, but, unfortunately, we don't know what they are and what kind of influence they can have.

Here's some examples : **V14**

Lea Setruk 345226179
Aviva Shneor Simchon 317766731



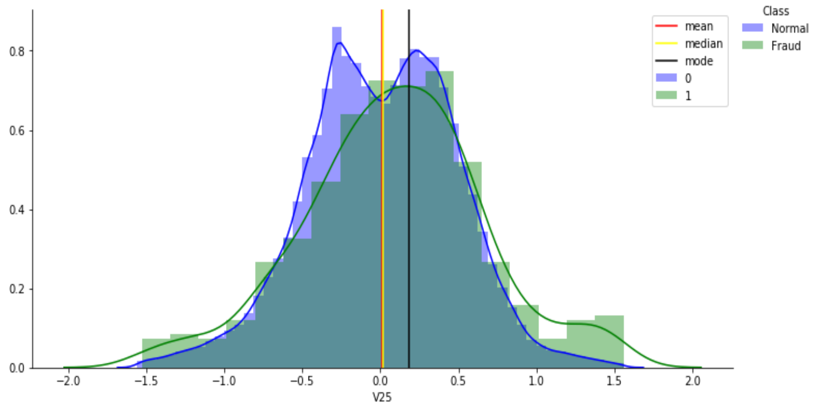
The distributions here for fraud and normal are clearly different.



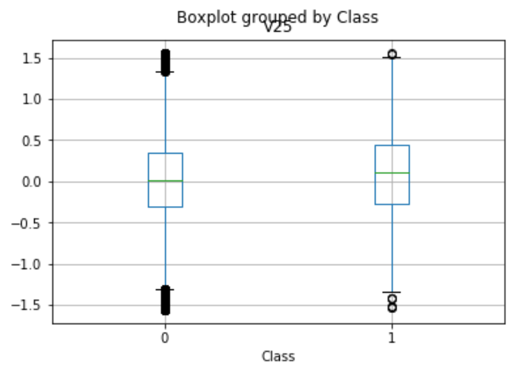
```
Normal

count      284315.000000
mean            0.012064
std             0.897007
min           -18.392091
25%            -0.422453
50%             0.051947
75%             0.494104
max            10.526766
Name: V14, dtype: float64
```

```
Fraud

count      492.000000
mean        -6.971723
std          4.278940
min        -19.214325
25%         -9.692723
50%         -6.729720
75%         -4.282821
max          3.442422
Name: V14, dtype: float64
```

V25

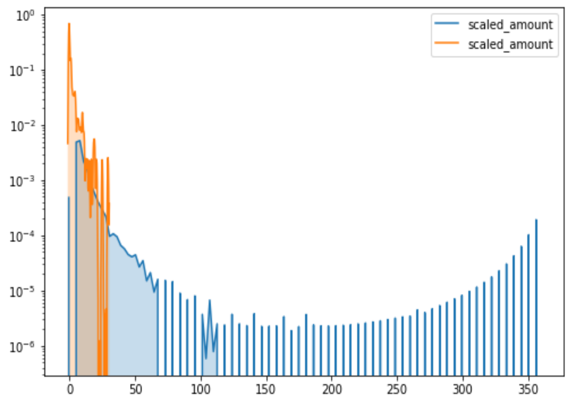


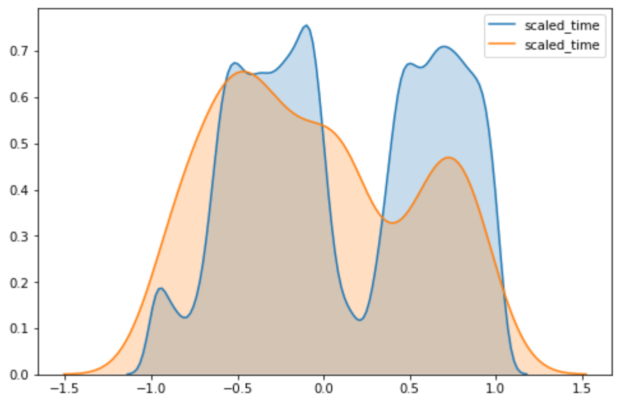
Here, the distributions are very similar.



```
Normal

count      284315.000000
mean           -0.000072
std             0.520673
min           -10.295397
25%            -0.317145
50%             0.016417
75%             0.350594
max             7.519589

Name: V25, dtype: float64
```

```
Fraud

count     492.000000
mean        0.041449
std         0.797205
min        -4.781606
25%        -0.314348
50%         0.088371
75%         0.456515
max         2.208209
Name: V25, dtype: float64
```





The distributions for the Amount variable are quite similar.

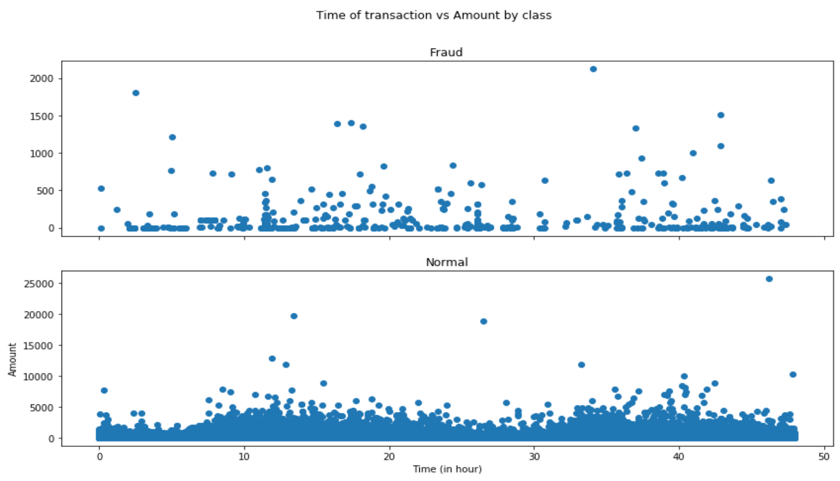
The distributions for the Time variable are different. We'll use it for our algorithms.

TIME

We tried to check if the variable Time has an inluence on the fraud. Maybe the fraudulent transactions are happening more during a special time ?
When we check their distributions, we can see a difference.

We can't really know the time. The data is in seconds, it doesn't tell us when it begins. Moreover, the transactions have been made by europeans. Maybe they were in different countries, continent, there might be a jet lag. We don't have enough information to conclude exactly what influence the time has on the transactions.

We tried also to see if time has an influence on the <u>amount</u>.



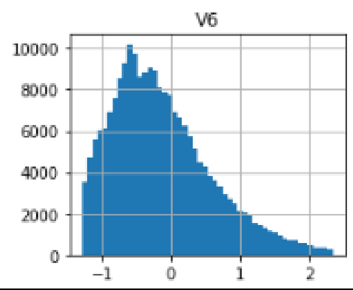
We also don't have enough information on the time variable to conclude. It seems like it doesn't influence.

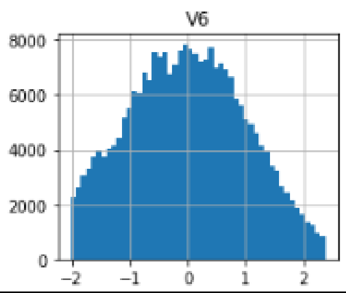
Time and Amount have numbers very different from the other features, way higher. So it could be more confortable, we scaled them.

Lea Setruk 345226179
Aviva Shneor Simchon 317766731

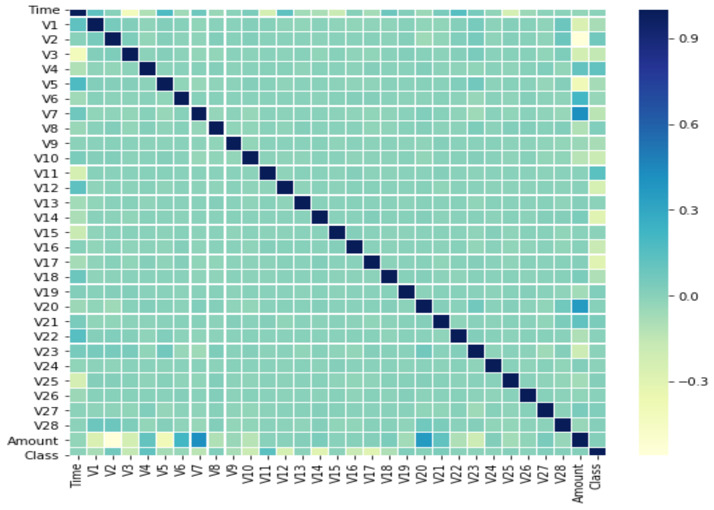
Data transformation

To apply our algorithms (GMM, Kmeans, Agglomerative), we need to transform our data into a normal distribution.
Some of our features give us more information than others. We can see that most of the features, after transformation, are normally distributed. However, some of them aren't normal afterwards but we chose to let it because we got good results anyway.
For example :
Before transformation : After transformation :





Correlation



There is no correlation between the variables. Or if there is, it is very weak.

PCA has already been applied on the data set, thus, it's an unsurprising result

Data - features for algorithms

Like said before, we chose to work with those features because of the distribution differences between fraud and not fraud : V1, V2, V3, V4, V5, V7, V9, V10, V11, V12, V14, V16, V17, V18 and also scaled time.
We are based on the Baysian method and we got good results without taking all of the features.

The purpose of this work is not to predict but to draw good conclusions. Therefore, we didn't split the data to train and test.

Our dataset is huge and very unbalanced (only 0.172% of the transactions are fraudulent). For this reasons, we decided to take a sample with 5000 observations of normal transactions and all of the fraudulent ones (492).
We run our models on this sample, analyze the results and then, we do predict over the data (with the chosen features).

We used two methods to reduce dimensions : MDS that works on unnormal data but the disadvantage of it is that we didn't succeed to run it on all the data but only on the sample.

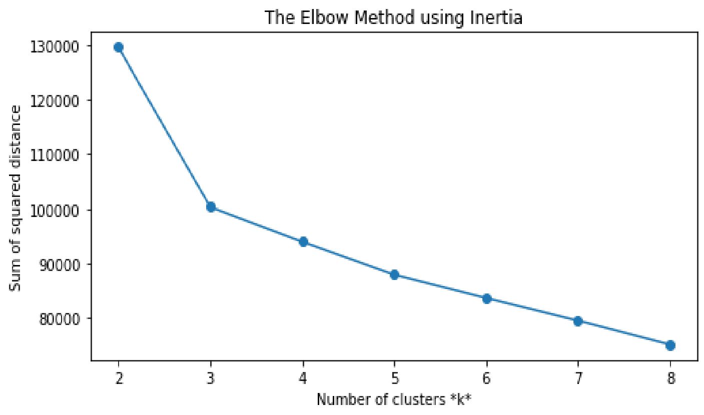
Lea Setruk 345226179
Aviva Shneor Simchon 317766731

The second method is PCA that works on normal data. It's a strong model and we succeeded to run it on all the data
We used them for visiualization. We ran our models on the data after PCA and we compare the results.

Clustering Part

We chose to work with the algorithms : KMEANS, GAUSSIAN MIXTURE MODEL, AGGLOMERATIVE MODEL.
For every model, we checked the optimal number of clusters using different tests based on the model.
We ran the model with their variations and we calculated the P_value of several score methods.

Kmeans

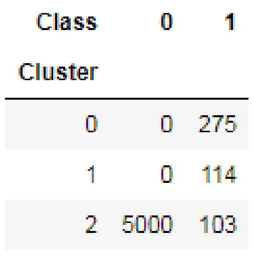


We want to know how many clusters to use for the Kmeans .

To do so, we use the silhouette score and the elbow score folkes and v_mesure score.

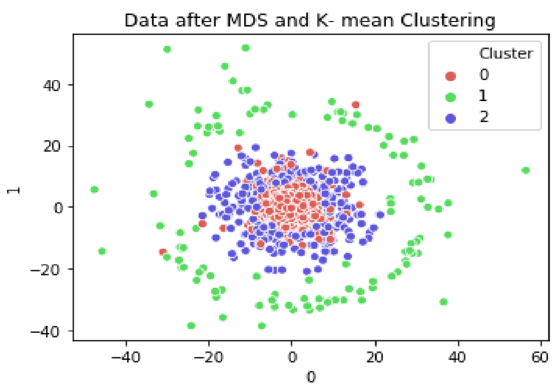
Those scores tell us to use 3 clusters for Kmean.

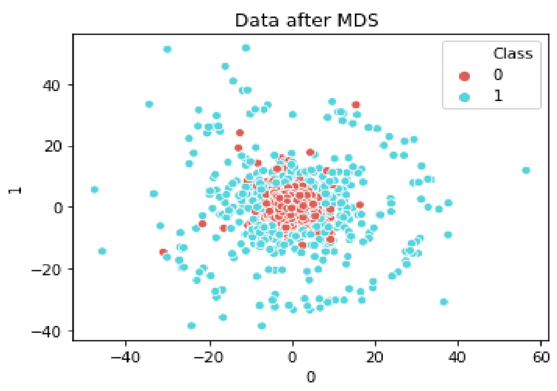
As we say we run the model to 3 clusters :
:



Class	0	1
Cluster		
0	0	275
1	0	114
2	5000	103

All of the normal transactions are in one cluster only !





Score of the model:
Homogeneity: 0.692 , Completeness: 0.698 , V-measure: 0.695, Normalized mutual info: 0.695, Silouhette_score: 0.630
(For 2 clusters silouhette score give 0.793 score but we see that 1 cluster take 390 fraud transaction)

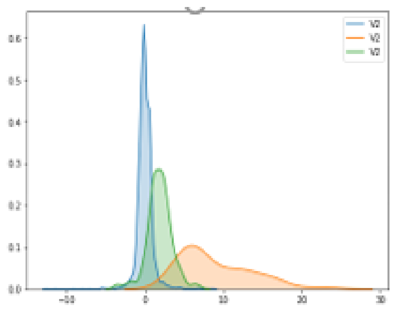
Lea Setruk 345226179
Aviva Shneor Simchon 317766731

We ran the K-means model on the data after PCA and we get same results - normalized mutual info: 0.697

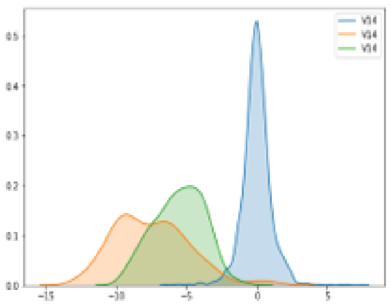
<u>We analyzed the features that are in each cluster</u>

The blue curve represents the variable in the cluster number 2 that contains all of the normal transactions. We can see that, mostly, the distribution of the variables in this cluster is different. For example :

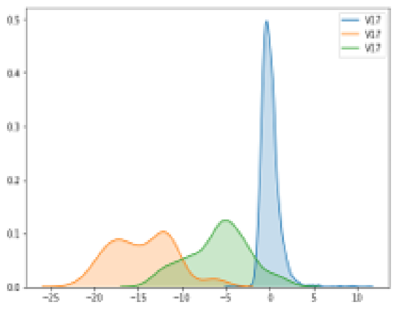
V2 :



V14 :



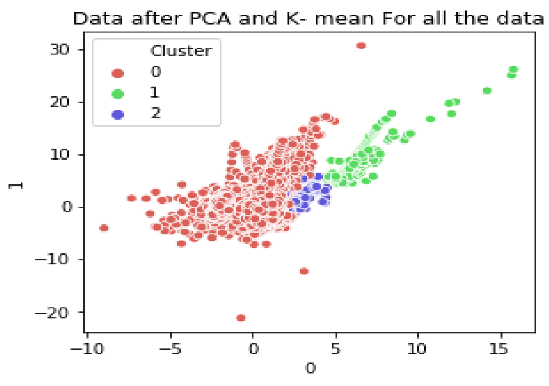
V17 :

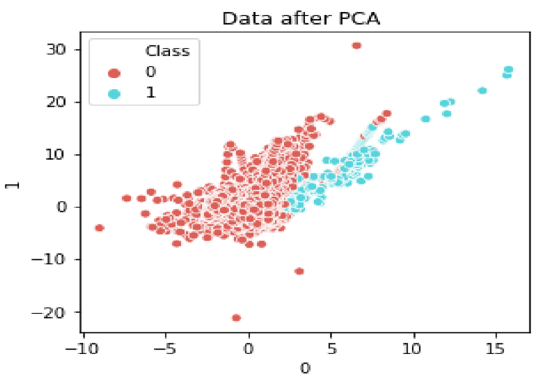


On all the data, the kmeans model gives us :

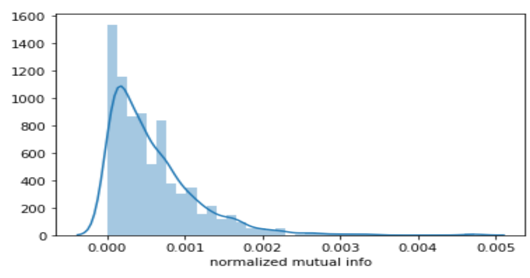
Class	0	1
Cluster		
0	44	275
1	27	114
2	284244	103

We clearly see that almost all of the normal transactions are in one cluster.





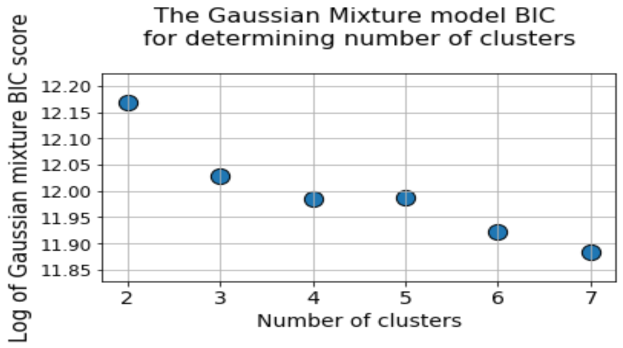
K-means model give us nice results :
<u>Homogeneity</u>: 0.692, <u>Completeness</u>: 0.677, <u>V-measure</u>: 0.684, <u>Normalized mutual info</u>: 0.699
<u>p value for normalized mutual info</u>- 0.0



Lea Setruk 345226179
Aviva Shneor Simchon 317766731

Gaussian mixture model

For GMM, we use BIC and AIC score to choose the optimal number of clusters.
We got that 2 clusters is the best number for our model.
Both gave us the same score.

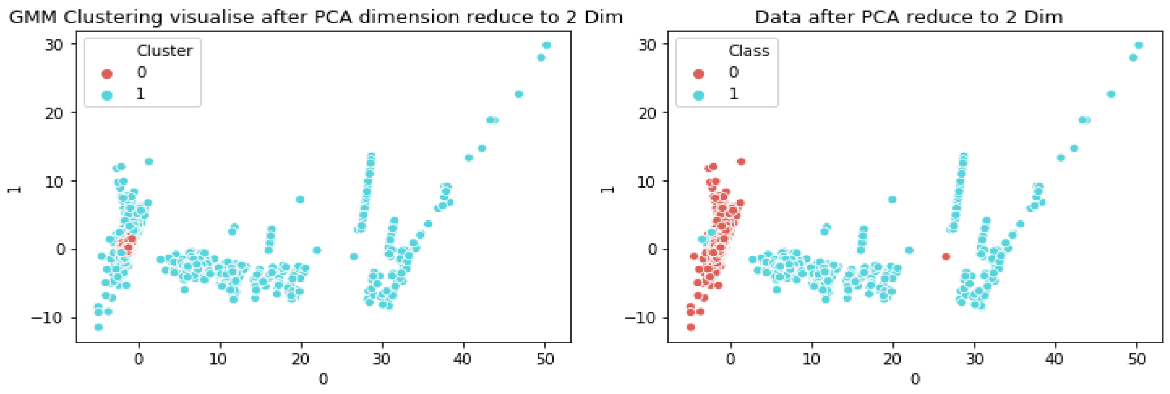


We ran GMM model with 2 clusters :

Class	0	1
Cluster		
0	4372	42
1	628	450

We have one cluster with most of the normal transactions and only 8% of the fraud ones. The second cluster is smaller and contains 58% of normal and most of the fraudulent transactions.

We suggest to apply another model for the transactions only on cluster 1. We tried some models (isolation forrest, SVM one code…) but we didn't have time to continue with it.



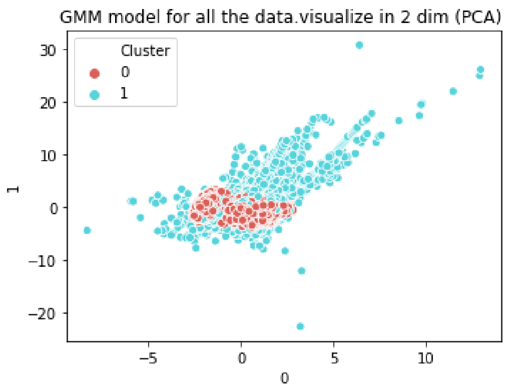
Evaluation score:
<u>Homogeneity</u>: 0.407, <u>Completeness</u>: 0.247, <u>V-measure</u>: 0.308, <u>Normalized mutual info</u>: 0.317, <u>Silouhette score</u> : 0.4315170411544054
<u>p value for normalized mutual info</u>- 0.0

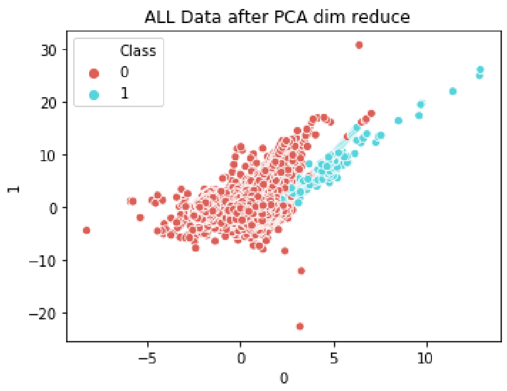
On all data the GMM model:

Class	0	1
Cluster		
0	247242	42
1	37073	450

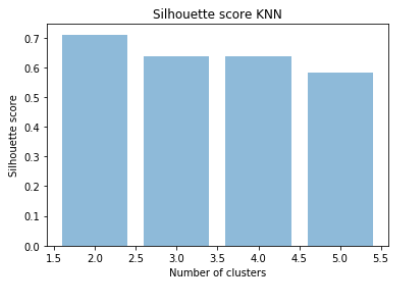
We clearly see that most of the normal transactions are in one cluster and most of the fraud transactions are in the second cluster !

Lea Setruk 345226179
Aviva Shneor Simchon 317766731

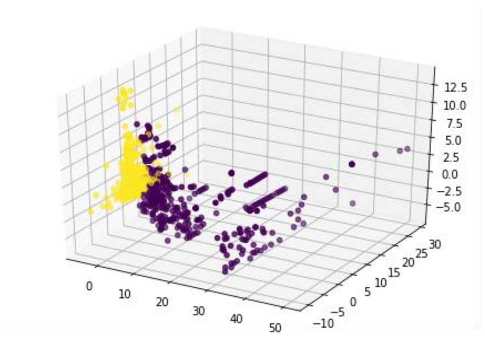




Agglomerative model



The optimal number of clusters to use is 2.



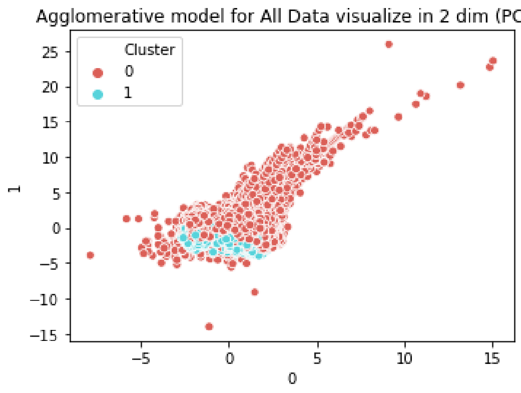
In the sample data

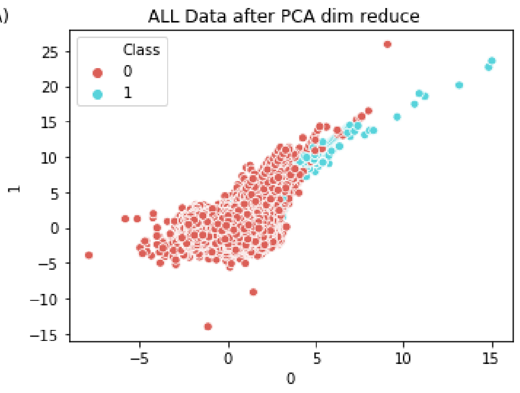
Cluster	normal	fraud
0	0	394
1	5000	98

All of the normal transactions are in one cluster and most of the fraud ones are in the second cluster.

Evaluation score: Homogeneity: 0.708, Completeness: 0.827, V-measure: 0.763, normalized mutual info: 0.765, p value for normalized mutual info- 0.0, silhouette score: 0.708

On the contrary, we didn't get good result for the all data : normalized mutual info : 0.003
Agg is not a good model for our data.





Lea Setruk 345226179
Aviva Shneor Simchon 317766731

DBSCAN MODEL

DBSCAN works with not normal data so we used the data without transformation.
We took a sample of 5000 normal transaction and all the fraud (492).

We tried to find optimal parameters for the model.
There are two main constants for this algorithm : Epsilun that represents the positive real radius and minPts that is the number of neighbors in a positive epsilun.

We played a lot with theses parameter, we have not a conclusion about them. It's about experiments and business comprehension.

We ran the model : DBSCAN(eps=5.8, min_samples=8.8) and we get nice result:
The model classifies as outliers (cluster -1) 42 normal transactions an 101 fraud ones.
We checked and found out that the optimal number of clusters is 5 and the estimated number of noise points is 143.

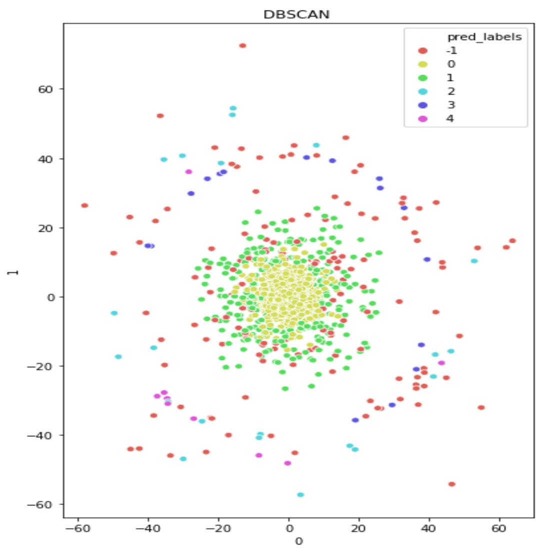
Evaluation score: <u>Homogeneity</u>: 0.653, <u>Completeness:</u> 0.542, <u>V-measure</u>: 0.593, <u>Adjusted Rand Index</u>: 0.805, <u>Adjusted Mutual Information</u>: 0.542, <u>Silhouette Coefficient:</u> 0.624

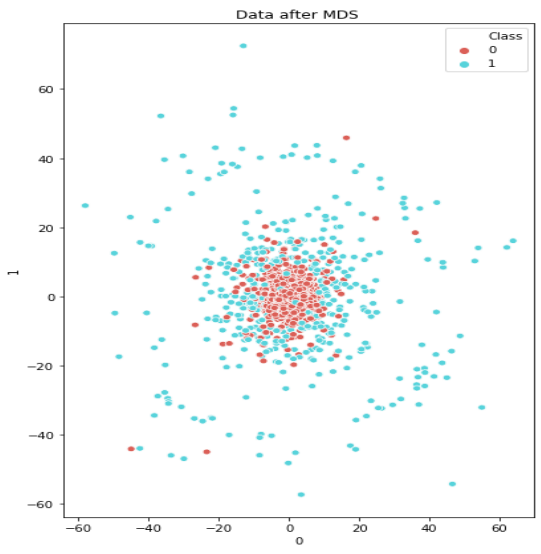
class	0	1
cluster		
-1	42	101
0	4955	95
1	3	249
2	0	21
3	0	17
4	0	9

The big cluster (0) contains most of the normal transactions and almost 20% of the fraud.

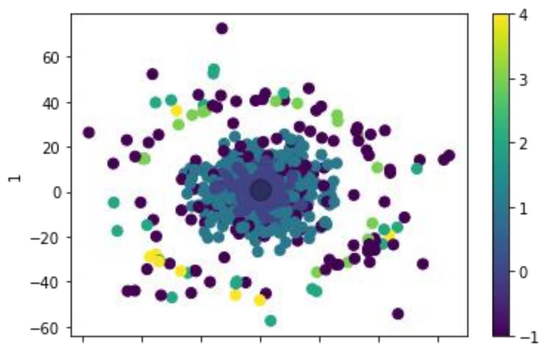
The other clusters are mostly fraud clusters.

Ce believe that we can approve this model with more time.





Lea Setruk 345226179
Aviva Shneor Simchon 317766731



Conclusion

We got a numeric data with a lot of observations and variables that we don't know their signification (V1-V28), Amount and Time. Not knowing what are the variables, was a difficulty for the analysis and also for the anomaly detection. The data was given to us after PCA, without empty cells, and only numeric variables, therefore less things were to do for the data cleaning. We worked on understanding what to do with the data, how to clean it, how to transform it, to be able to run algorithms. The data was very unbalanced.
We tried a lot of different ways to clean the data, tried different features, different normalizations, differents samples etc… and we chose the best one, that gave us the best results as we explained below.

The clustering algorithms that we chose to run are : Kmeans, GMM, Agglomerative and DBSCAN.
To run them, we first ran on a sample and then generalized to the all data.
For every algorithm, we checked different scores to evaluate : Homogeneity, Completeness, V-measure, Adjusted Rand Index, Adjusted Mutual Information, Silhouette Coefficient.
They are scores for clustering models.

For Kmeans, we got very good results (as we can see in the evaluation score). When we generalized on all the data, it gave us also very good results. We get one cluster with most of the normal transactions. The model succeeds to separate normal and fraud transactions to different clusters.

For GMM, we got pretty good results. Not as good as Kmeans, but the scores are good and it also succeeds to separate normal and fraud transactions to different clusters.

The agglomerative model gives us very good results on the sample only, but on all the data, it doesn't succeed to cluster. It's not an algorithm for our data. The scores for all of the data are bad !

DBSCAN is a strong algorithm and works very well on anomaly detection. It gave us very good results. It succeeds to cluster the transactions. Indeed, one cluster is a normal transactions cluster and the others ones are for the fraud transactions. The scores we got on this algorithm are the best ones !

Kmeans and DBSCAN gave us the best results for the clustering.
Despite that Kmeans is a simple model, it works well on our data.
DBSCAN is the best algorithm for our data with the best scores.