**UNSUPERVISED LEARNING**

**Dataset**

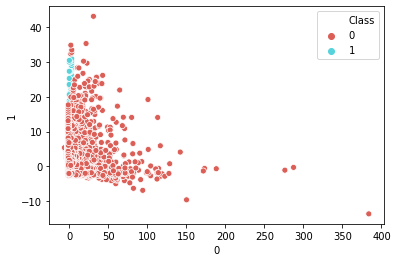
We chose to work with the **credit card** dataset. Its purpose is to recognize fraudulent credit card transactions.

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.

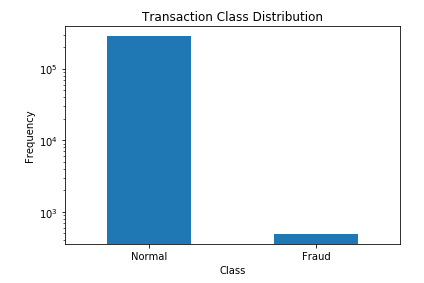
**Data Analysis**

*Visiualization of the data after PCA dimension reduction*



We checked the covariance matrix and we obtained that every varaible countains about the same amount of information.

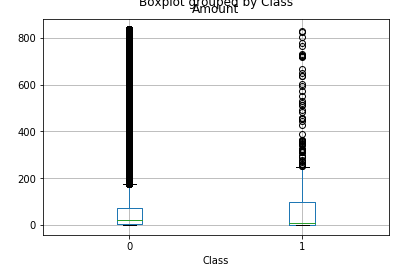
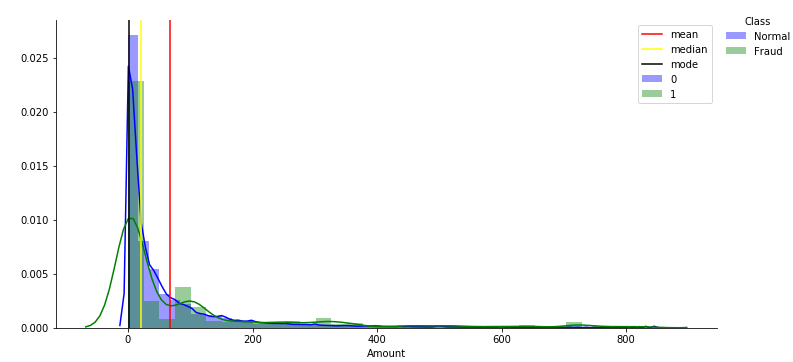
*Class distribution* :



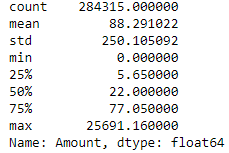
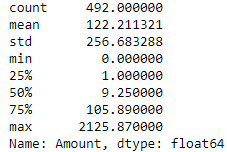
We want to see what are the differences between the normal transactions and the fraud ones. We compare them for every variables. We checked their distributions, their mean, standard deviation, max, min etc…

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.

For example, let’s observe the Amount variable. We tend to think that there will be a difference here :

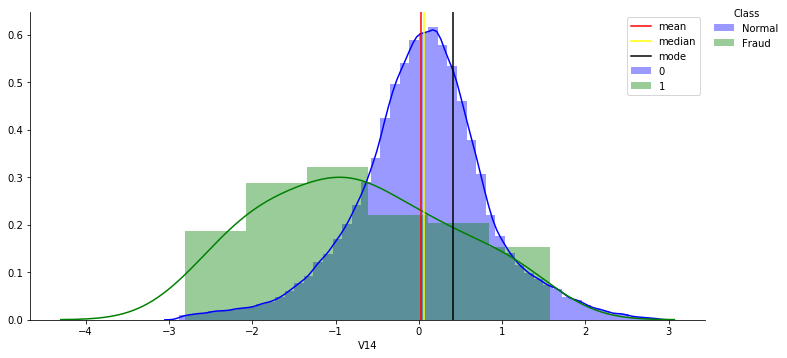
Normal amounts : Fraud amounts :

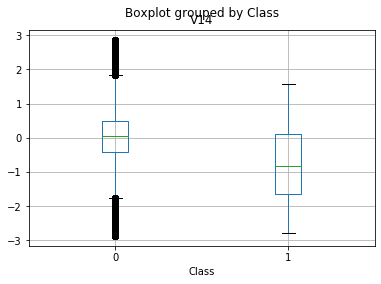
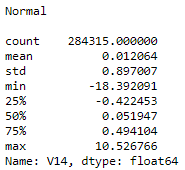
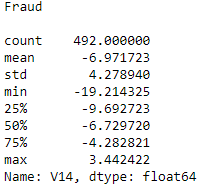
 

We did this observations for every variables. We found differences between the distributions of the fraud and the normal data for those variables : V1, V2, V3, V4, V5, V7, V9, V10, V11, V12, V14, V16, V17, V18. So, we chose to work with those features for our algorithms.

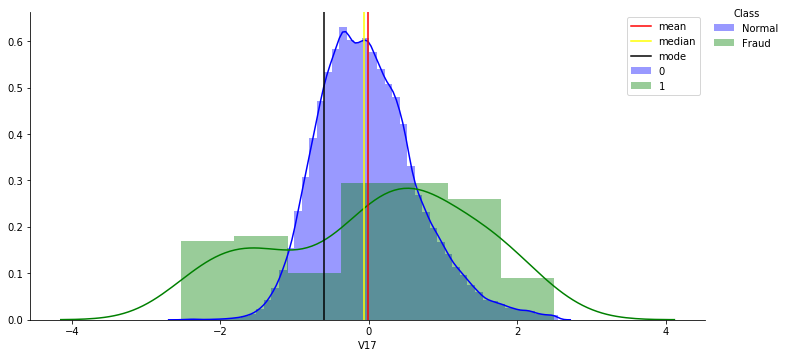
We would have liked to talk more about those different variables and their influence, but, unfortunately, we don’t know what they are and what kind of influence they can have.

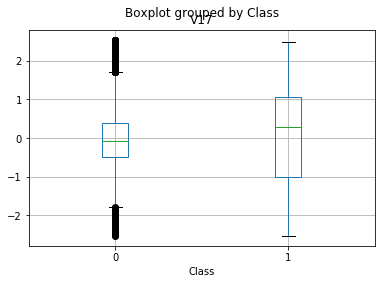
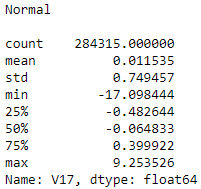
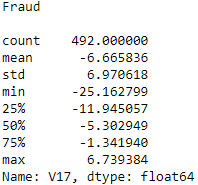
Here’s some examples : **V14**



**V17**



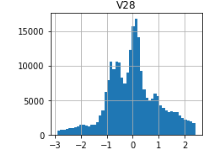
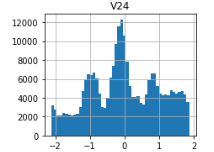
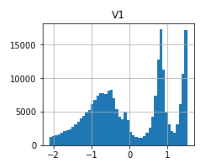
  

**Data transformation**

To apply our algorithms (GMM, Kmeans, Agglomerative), we need to transform our data into a normal distribution.

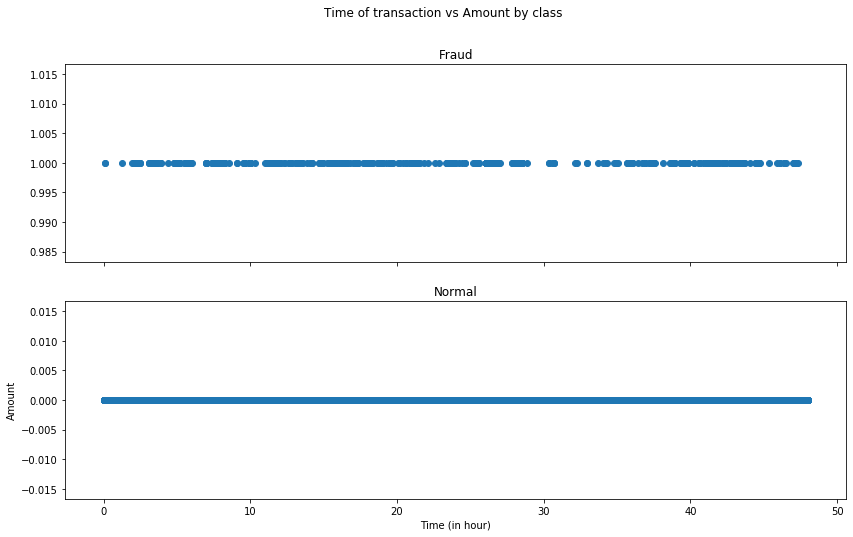
Some of our features give us more information than others. We can see that most of the features, after transformation, are normally distributed. However, some of them, are different.

For example :



**Time**

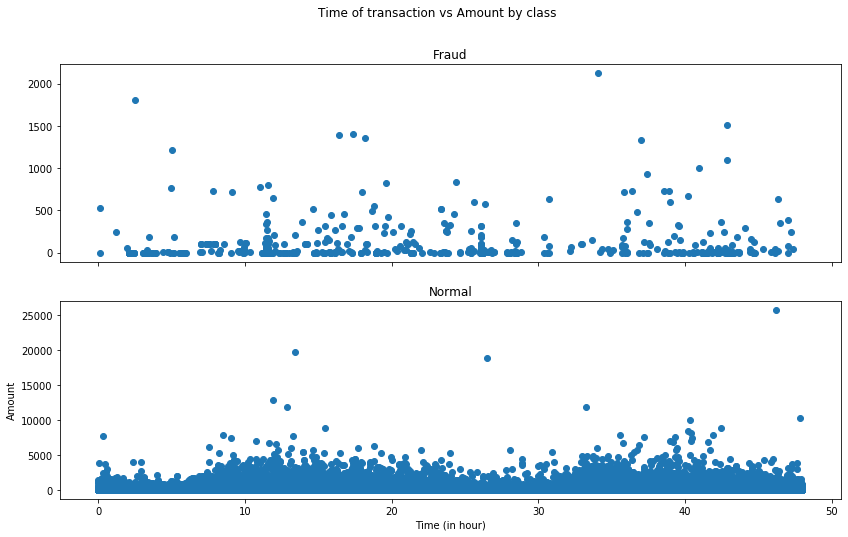
We tried to check if the variable « time » has an inluence on the fraud. Maybe the fraudulent transactions are happening more during a special time ?



We don’t see a significant difference. We also tried to separate in different time lines, but it wasn’t interesting.

We also, 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.

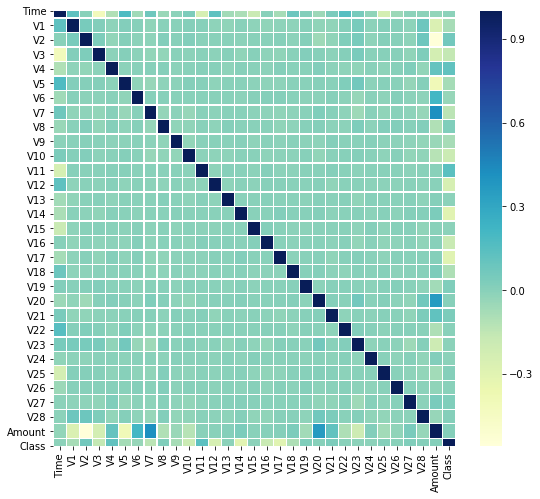
We tried also to see if time has an influence on the amount.



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.

**Correlation**



**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 reason, 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.

The second method is PCA that works on normal data. It’s a strong model and we succeed 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.

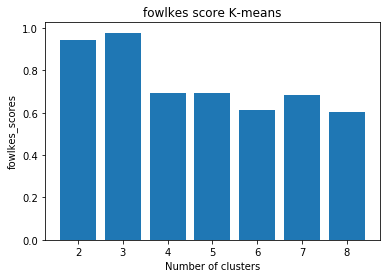
**Clustering Part**

We chose to work with the algorithms : KMEAN, GAUSIAN MIXTURE MODEL, ALGORR.

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**

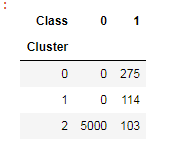
: 

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

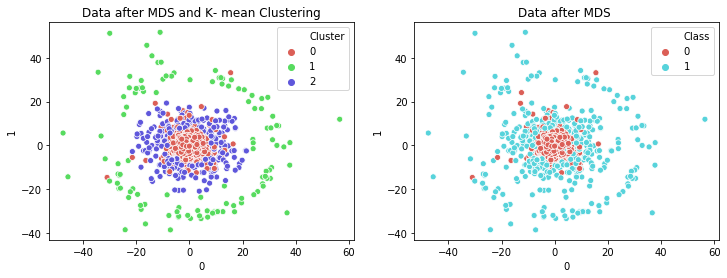
To do so, we use the silhouette score and the elbow score folkes and v\_mesure score.

Those scores tell us to use 3 clusters for Kmean.

As we say we run the model to



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

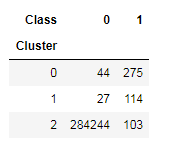
**LEA: to do p value**

(for 2 cluster silouhette score give 0.793 score but we see that 1 cluster take 390 fraud transaction)

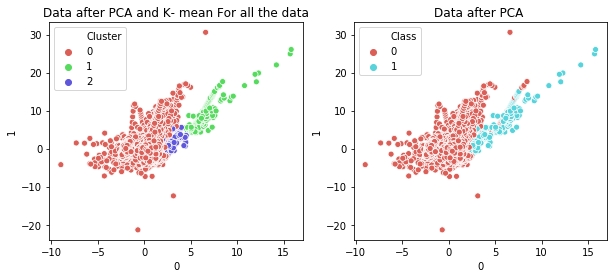
**to add also in the code: we see the distribution in the feature on each cluster.**

We ran the K-mean model on the data after PCA and we get same results.

*On all data the kmean model*:



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



**K-means model give us nice result**:

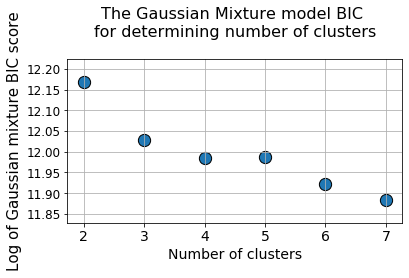
Homogeneity: 0.692, Completeness: 0.677, V-measure: 0.684, Normalized mutual info: 0.684

**Gaussian mixture model**

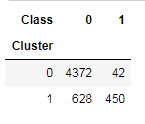
For GMM, we use BIC and AIC score to choose the optimal number of clusters.

We got that 2 clusters is the best number for our model.

Both gave us the same score.

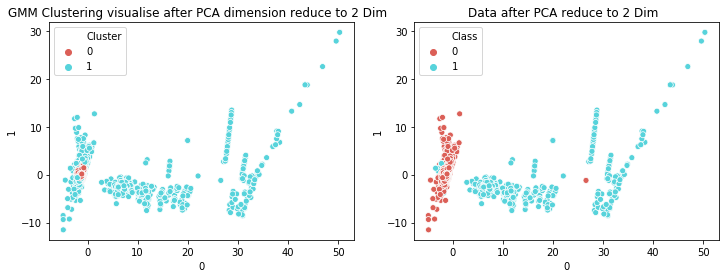


We run gmm model with 2 clusters :



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 that are clusterd here.

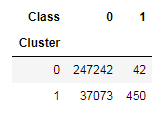


**Evaluation score:**

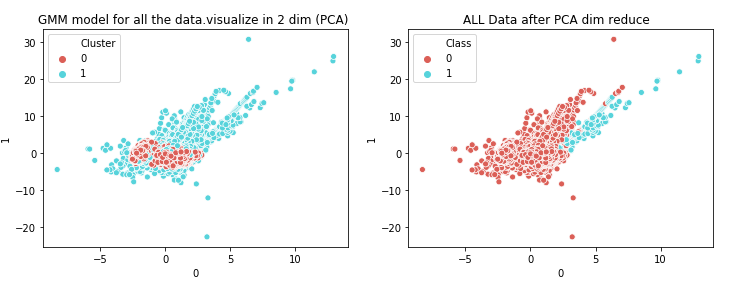
Homogeneity: 0.407, Completeness: 0.247, V-measure: 0.308, Normalized mutual info: 0.317, Silouhette score : 0.4315170411544054

P-value!!

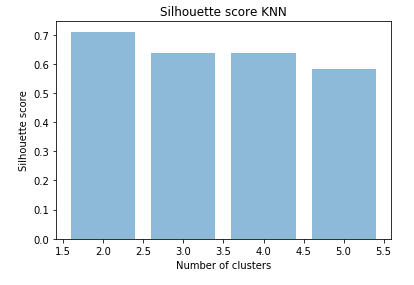
*On all data the GMM model*:



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



# Agglomerative model



The optimal number of clusters to use is 2.