Data Mining - Seismic Bumps

Giovanni Perri, Marco Recchioni, Alessia Montalbano

January 2022

Abstract

Mining activity was and is always connected with the occurrence of dangers which are commonly called mining hazards. A special case of such threat is a seismic hazard which frequently occurs in many underground mines. In this project we're going to identify hazardous seismic bumps in a coal mine by studying a data set given by UC Irvine Machine Learning Repository. It describes the problem of high energy (higher than $10^4 J$) seismic bumps forecasting in a coal mine. Data come from two of longwalls located in the Zabrze-Bielszowice coal mine, in Poland.

1 Introduction

This assignment is for a Data Mining class we take in University of Pisa. The assignment is a group project, divided into two parts. We were allowed to choose from two data set proposed, and here we are, dealing with seismic bumps.
Seismic hazard is the hardest detectable and predictable of natural hazards and in this respect it is comparable to an earthquake. More and more advanced seismic and seismoacoustic monitoring systems allow a better understanding of rock mass processes and definition of seismic hazard prediction methods. The UCI Machine Learning Repository (https://archive.ics.uci.edu) provides a 'seismic bumps' data set that contains many records of combined categorical and numeric variables that could be used to predict seismic hazards.
Our analysis attempts to use some techniques to predict whether a seismic 'bump' is predictive of a notable seismic hazard. Each observation summarizes seismic activity in the rock mass within one 8-hour shift. Note that the decision attribute, named "class", has values 1 and 0. This variable is the response variable we use in this project. A class value of "1" is categorized as "hazardous state", which essentially indicates a registered seismic bump with high energy ($> 10^4 J$) in the next shift. A class = 0 variable signifies that a hazardous bump did not, indeed, occur in the following shift to the measured data.

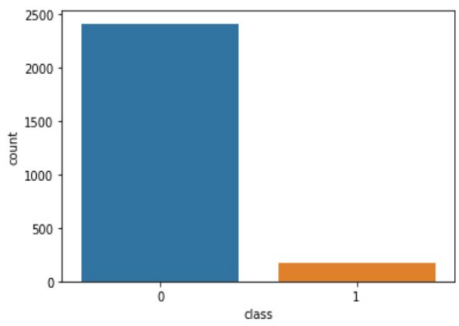
2 Data understanding and preparation

2.1 Data semantic

In the data set each row contains a summary statement about seismic activity in the rock mass within one shift (8 hours). There are 19 attributes for 2584 instances, with only 170 class = 1 variables, so the data are significantly skewed towards non-hazardous training data.
Mainly of the attributes are easy to understand and we can check their meaning in the official website, clicking here. However some of the attributes are not so obviously, let's add some information.
The geophones (or hydrophones) used for measure the seismic wave field are devices that convert ground movement (velocity) into voltage, giving electrical signals as output. If more than one geophone was assigned to the longwall, in order to assess hazard state in prediction horizon which was of our interest, the energy recorded by the *most* active geophone was considered. Energy is expressed in Joule, which in CGS units is defined as $J = 10^{-7} erg$, with an erg being equal to one gram centimetre-squared per second-squared $\frac{g \cdot cm^2}{s^2}$ or approximately *the amount of work done (or energy consumed) by one common house fly performing one "push up".*

2.2 Distribution of the variables and statistics

First of all we notice that out of 2584 records, there are only 6.5% hazardous instances. We can better see odds from the table below.



Variables 'nbumps(i)' contain integers from 0 to 9. They are all positively skewed distributions and they're shown in Figure 1.

Another interesting graph is the parallel coordinates (Figure 2). It shows some interesting features:

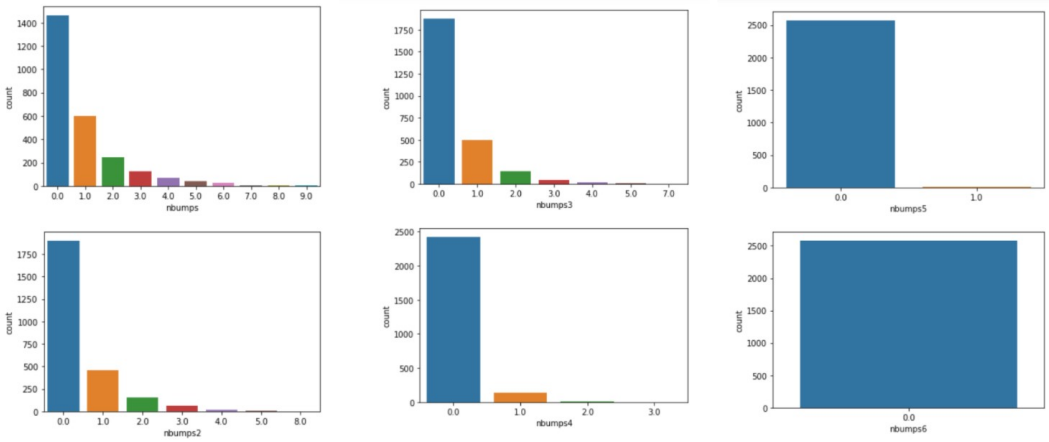
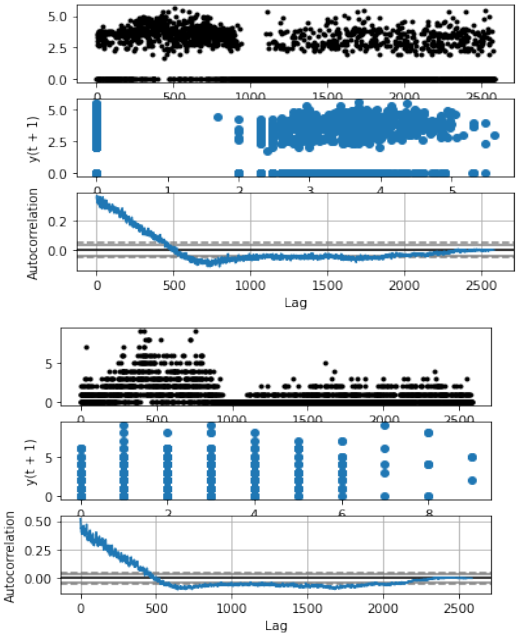


Figure 1: Occurrences of nbumps

events of class one are most of the time preceded by low energy seismic bumps; high energy events are matched with few nbumps (not bigger than 3); similarly events with class 1 (we recall that "class" label is referring to the next shift) have mainly low values of genergy and gpuls.

We have performed a time series analysis for the variables. This consist in 3 graphs: the first is the plot in respect to the time, showing the temporal sequence of the events. The second is a lag plot which display the value y(t) in respect to the next value y(t+1). The last one is the autocorrelation function showing the memory of the system for the next values. We report those graphs for the energy and the bumps which gives some important feature of the dataset.



We can identify two distinct periods, the first one much more intensive for energy and number of bumps, followed by a more stable one, where the bumps are less as well as the energy. From the lag plot we can assure that there is no pattern for the energy and the bumps, since for all the values, the next one to occur has no favourite value. The autocorralation display a positive value for the next 500 shifts which now turn out to be slightly negative and stabilize to zero at the end.

2.3 Assessing data quality

At first glance it seems there are no missing value in our data set. However, just plotting any of the main attributes vs time (in shift unit), we elicit that some hidden missing values, around shift n°950-1100, can exist. The zero values might come from a broken or blocked sensor and might be consider as missing values.

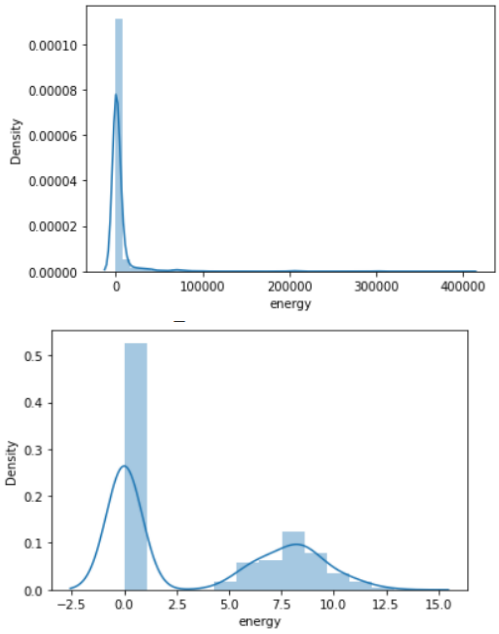
Columns referring to nbumps 6, nbumps7 and nbumps89 (hence number of seismic bumps in energy range $[10^6, 10^{10}]$ registered) are null, so no high seismic bumps are (fortunately) registered. Semantic is clear and coherent for all attributes, without inconsistencies.

So there are 15 input variables and one binary output variable ("class"). The data are mostly numeric with 4 categorical input variables. The categorical variables are seismic, seismoacoustic, shift, ghazard, while the numeric ones are the remaining. However we can treat nbumps(i) as a sort of categorical variables (we saw in the plot above that some of the numeric values only contain a handful of discrete values which can be viewed as coded categorical variables). The continuous variables are genergy, gdpuls, energy, maxenergy and the output variable is class.

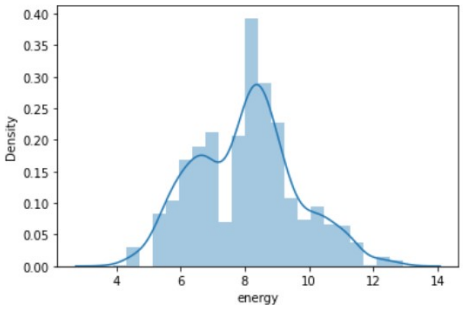
2.4 Variable transformations

Like all other numeric features, plotting energy's distribution (we'll see from the correlation matrix that maxenergy can be dropped due to the 1 to 1 correlation with energy) suggests us to transform variable, in order to avoid that high right-skew. We try log transformation after adding a constant 1 and obtain in this way the following graphs.

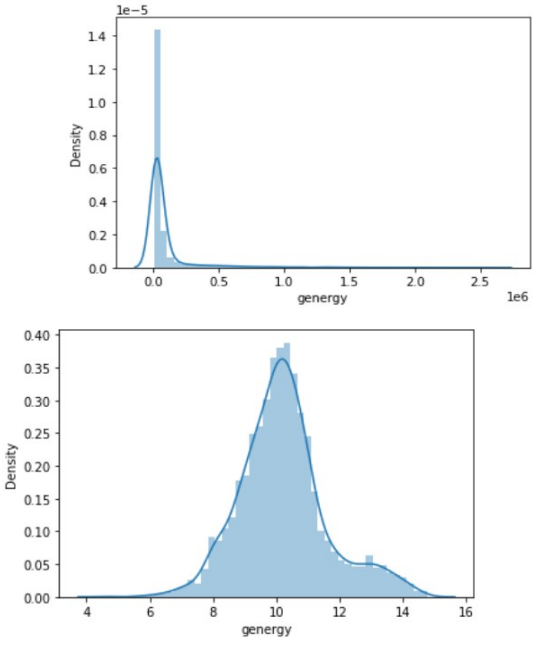
Figure 2: Parallel coordinates.



Most of the data have energy 0, as we can see. If we do not consider energy with zero values, pdf in logarithmic scale (except for some values) takes a more familiar shape. In this way we also avoid the problem of negative energy (that has no physical meaning), in the pdf.



Same considerations hold for genergy: however here, unlike for energy, there's not a concentration of zero values. In fact, taking the logarithmic values, the distribution is look like a Normal distribution (and we know from Central Limit Theorem that a sum of random variables of finite variance tends toward a normal distribution (informally a bell curve) even if the original variables themselves are not normally distributed).



2.5 Pairwise correlations and eventual elimination of variables

From the correlation matrix shown in Figure 3, we can see in the first instance that there are two high correlated and redundant features (energy and maxenergy): in fact they represent respectively the total energy and the maximum energy of the seismic bumps registered within previous shift, that coincide in most of the cases. In fact in the 79.8% of the total recorded shifts there's only one bump recorded.
There are also some high correlated features, like genergy-gpuls and gdenegy and gdpuls. These last two,

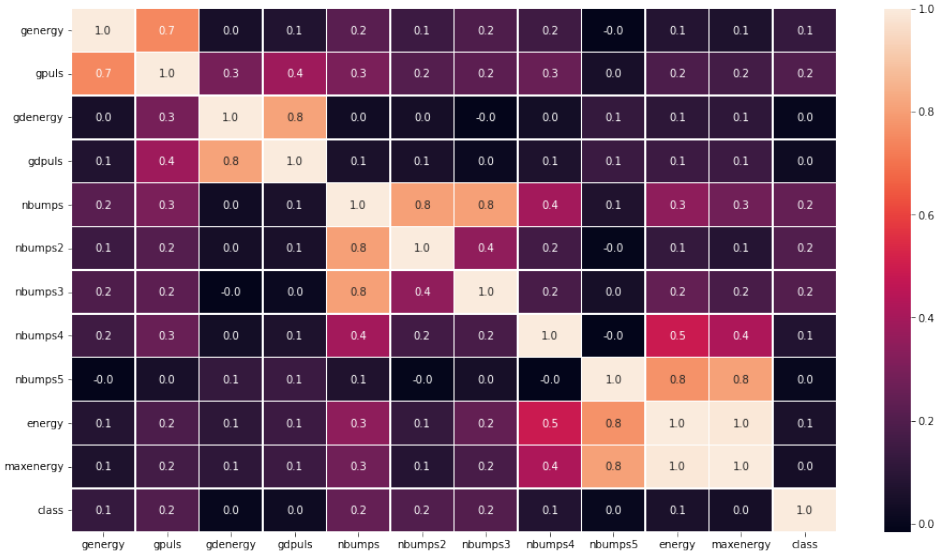
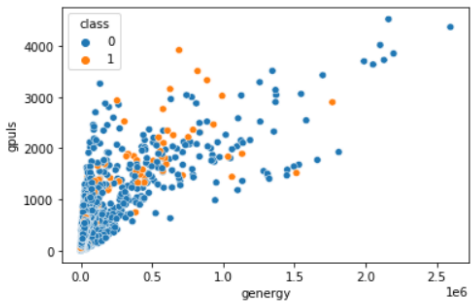


Figure 3: Correlation matrix

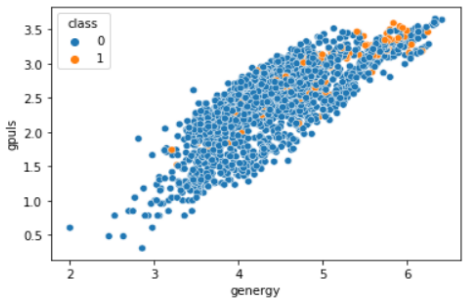
at the moment, will not be considered for further analysis, due to some contradiction with the official attribute values.

The response variable for the dataset is class, so in order to reduce data set's size, we look at the relationship of the four categorical variables (seismic, seismoacoustic, shift, ghazard) with the response variable class. We report the tables (in Figure 4) to parse out any relationship between the response and the variables. If there is not a relationship, we will assume the variables are independent and remove them from the analysis. Tables are provided in the next page and show that the variables seismic and shift have a relationship to the response variable, while the other two seem do not have any relationship therefore will be removed for further analysis.

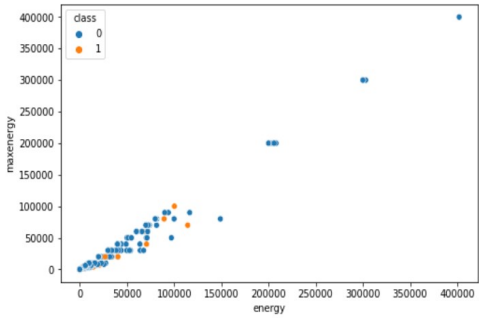
From the correlation matrix we see genergy and gpuls have high correlation, so we used a scatterplot to visualize these correlated features. The scatterplot to see how these numerical values are scattered is shown below.



At first impact it seems they are not that much correlated, but if we transform the axis into logarithmic, we can see points are placed on the same 45° line, clearly demonstrating the high correlation.



We report for completeness also the scatterplot between energy and maxenergy, that confirm our choose to drop the second redundant feature.



3 Clustering

Due to this rather generic view, clusters found by DBSCAN can be any shape, as opposed to k-means which assumes that clusters are convex shaped.

We can compare the outliers obtained with boxplot method with noise of dbscan.

3.1 K-means

By creating the wanted data space the algorithm group up the data in respect to the designed metric (some

class	0	1
seismic		
a	1599	83
b	815	87

class	0	1
seismoacoustic		
a	1479	101
b	890	66
c	45	3

class	0	1
hazard		
a	2186	156
b	198	14
c	30	0

class	0	1
shift		
N	904	17
W	1510	153

Figure 4: Tables of the comparison between class and categorical variables. Doing the ratio of the occurrences of ones and zeros, we obtain $\frac{83}{1599} \approx \frac{1}{20}$ vs $\frac{87}{815} \approx \frac{1}{10}$ for seismic, $\approx \frac{1}{15}$ for all the three in seismoacoustic, $\approx \frac{7}{100}$ for both a and b in ghazard (we do not consider c, the "most dangerous" hazard but without occurrences) and finally $\frac{1}{10}$ vs $\frac{1}{50}$ for shift. We infer that only seismic and shift are actually related with the output class.

feature have to be selected otherwise distance in high-dimension space can become meaningless).
To discover which number of cluster would suit the most the data set we thought to use the python method silhouette. This function gives an output value between [-1,1] describing how far the cluster are from each other. Good clustering would isolate the most a group of similar data so we've looked for the number of cluster which maximize this value and for which it turns out to be in good relation with the trade-off of the slope of the SSE for different number of cluster.
Afterwards we compute SSE distribution of the K-means for a random data set in the same range of our data. For instance close SSE value (within 5 sigma away from the peak of the SSE for the random distribution) would suggest a meaningless cluster for our data.

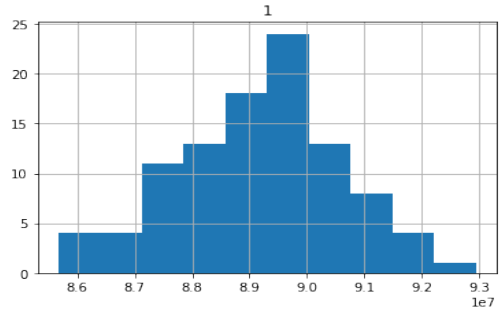


Figure 5: Histogram for a random sample in same range of the dataset 1.
Dataset1: mean 8.95e+07 std 1.54e+06
The result for the others sets:
Dataset2: mean 8.93e+07 std 1.57e+06
Dataset3: mean 12365 std 110

We've performed the K-means for 3 different data space, normalized to have 0 mean and unit variance:

1. gpuls,nbumps,genergy
2. Time, gpuls, genergy
3. nbumps,energy,gpuls,genergy

3.1.1 Dataset n. 1

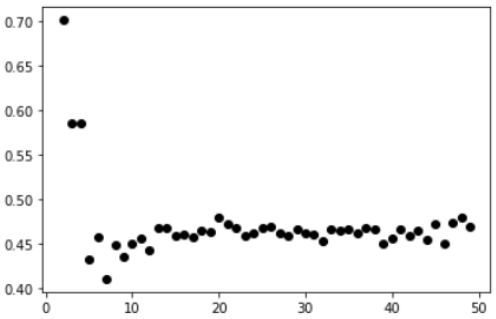


Figure 6: Number of cluster vs Silhouette
Best k=2

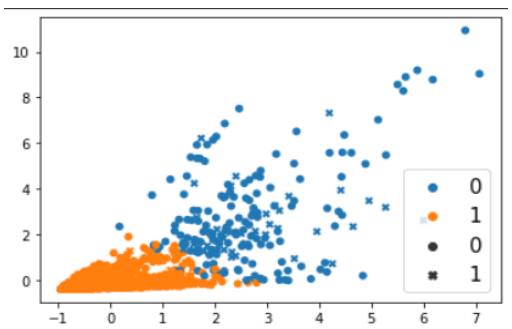


Figure 7: Genergy vs Gpuls.
SSE: 4438
Silhouette: 0.7
Number of elements: $0 \rightarrow 192, 1 \rightarrow 2392$

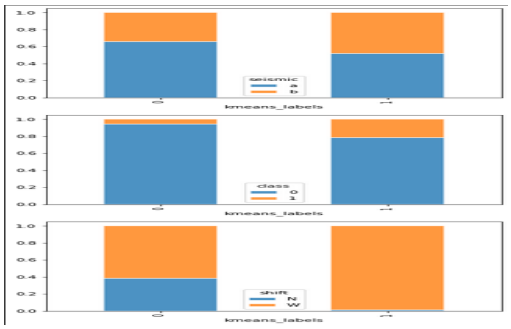


Figure 8: How categorical data distributes over the cluster. Cluster 1 seems to engroup the majority of class1 data as they all are Working shift

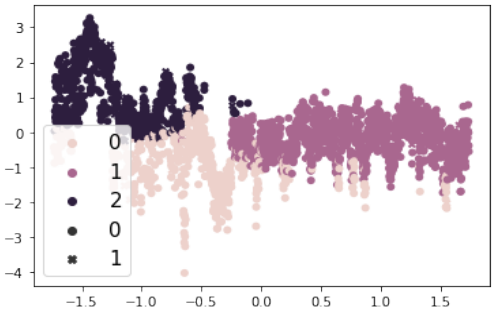


Figure 10: Time vs Genergy.
Low genergy are clusted all togheter while the high levels are clustered in 2 group in respect to the time.
SSE: 2813
Silhouette: 0.44
Number of elements: $0 \rightarrow 687, 1 \rightarrow 545, 2 \rightarrow 1352$

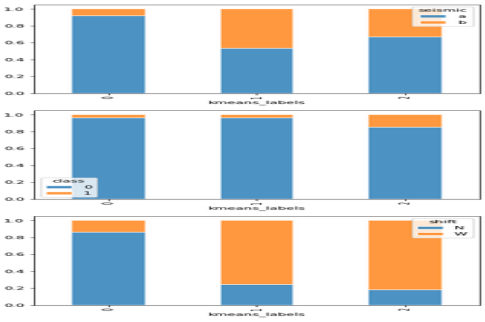


Figure 11: How categorical data distributes over the cluster.

3.1.2 Dataset n. 2

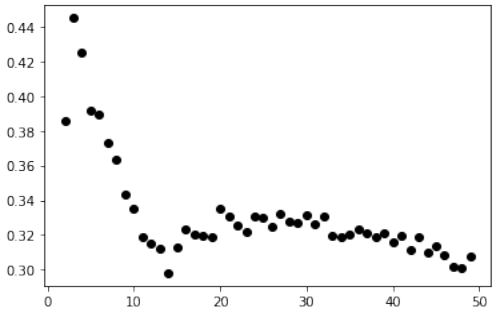
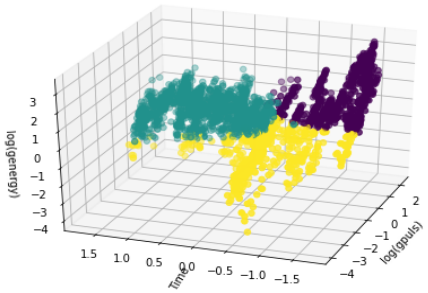


Figure 9: Number of cluster vs Silhouette
Best k=3



3.1.3 Dataset n. 3

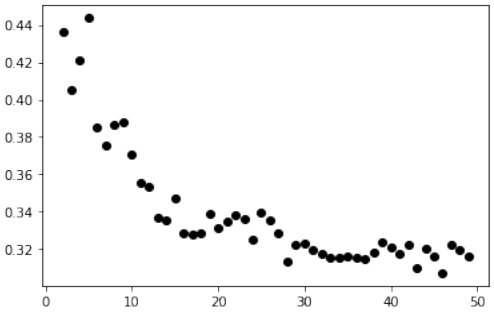


Figure 12: Number of cluster vs Silhouette
Best k=5

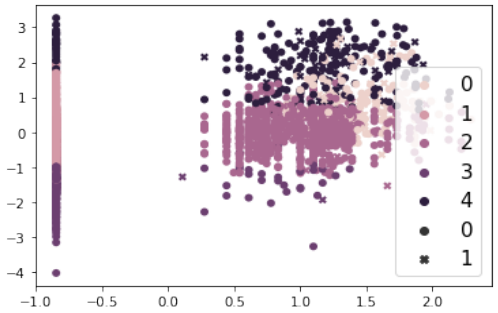
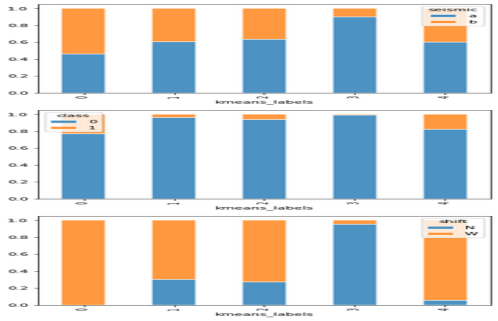


Figure 13: Energy vs Genergy.
SSE: 2493
Silhouette: 0.44
Number of elements: $0 \rightarrow 678, 1 \rightarrow 431, 2 \rightarrow 1038, 3 \rightarrow 191, 42 \rightarrow 246$

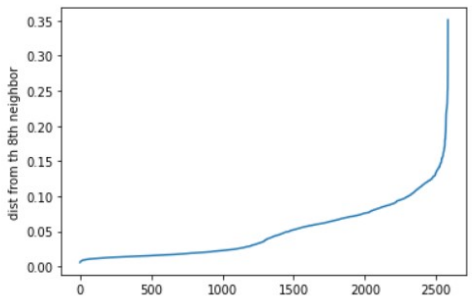


3.2 DBSCAN

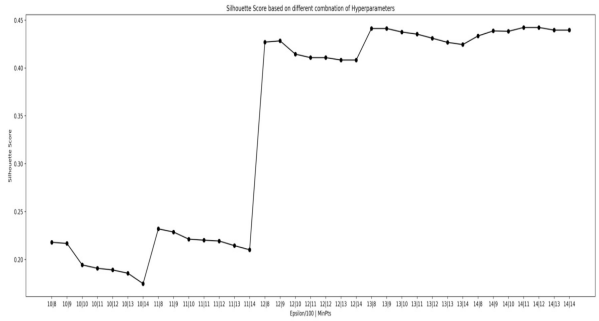
For this cluster there are two hyperparameters to set:

- Minimum samples ("MinPts"): the fewest number of points required to form a cluster:
- ϵ (epsilon): the maximum two points distance, which can be from one another while still belonging to the same cluster.

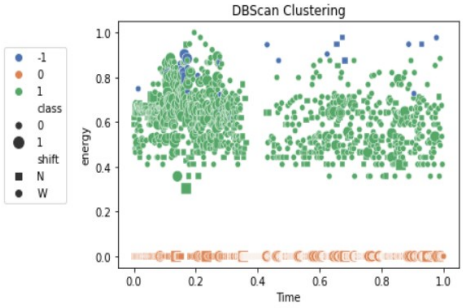
There is no automatic way to determine the MinPts value for DBSCAN. The larger (and noiser) the data set, the larger the value of MinPts should be. It was shown by Sander et al., 1998, that if our data has more than 2 dimensions, a possible choose is MinPts = 2*dim, where dim are the dimensions of our data set. In this section we decided to analyze only genergy, energy, nbumps and time.
With regard to the parameter ϵ, we calculated the average distance between each point and its k nearest neighbors, where k = MinPts value we selected. The average k-distances are then plotted in ascending order on a k-distance graph. The optimal value for ϵ is at the point of maximum curvature.

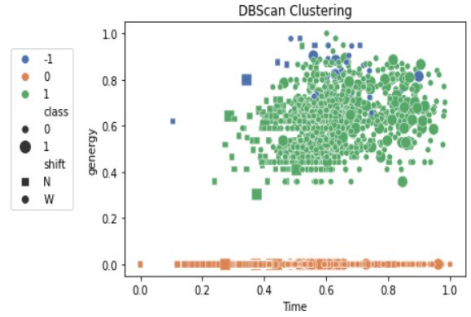


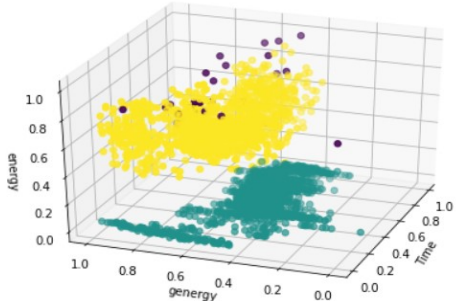
Another way to check our hyperparameters guess is based on Silhouette score. For this, we will use the Silhouette score available in the sklearn library. Here is a direct quote from its documentation: *The Silhouette Coefficient for a set of samples is given as the mean of the Silhouette Coefficient for each sample.* So we created multiple DBSCAN models using different combinations of epsilon and MinPts and plot Silhouette scores, obtaining the following graph, on which we can deduce that our guess (k=0.13,minsamp=8) is quite good.



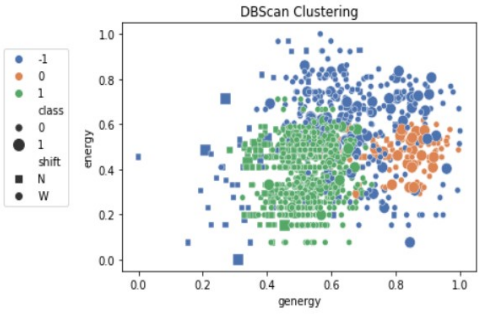
For normalizing variables, we use min-max scaling because it did not affect the underlying distributions of the features. Instead, it scaled the range down, so all of them now fit in between 0 and 1.
We obtained, using $\epsilon = 0.13$ e minsampl=8 (2*4) the following clustering, with n° of cluster equal to 3. First two are referring to the time series of energy and genergy respectively. Third one is the relation between genergy and energy, and the last is a 3d plot attained with `Axes3D`.







Clusters seem to be really affected by energy zero values' presence. Thus we made another scatterplot of genergy and energy, but this time dropping the whole rows (in the data frame) whenever energy zero values appeared. The plot shows 3 clusters (2+1 as before) in which we can notice how points classified as outliers, are in proportional "bigger" (legend says that bigger point are referring to event of class 1) than other.

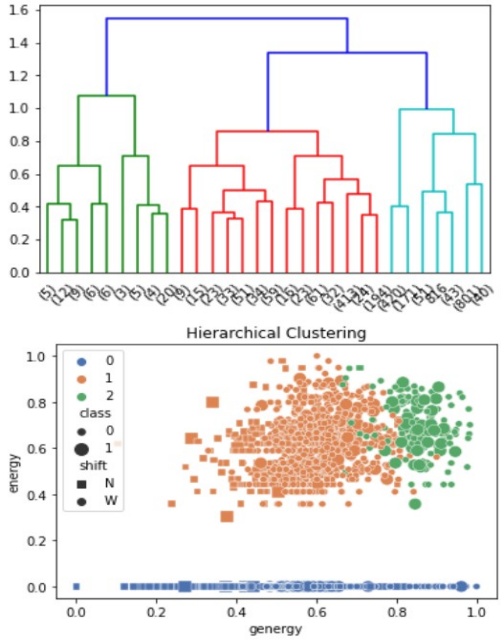


3.3 Hierarchical clustering

Hierarchical clustering is another type of unsupervised machine learning algorithm used to cluster unlabeled data points: like K-means clustering, hierarchical clustering also groups together the data points with similar characteristics. We use the minmax scaler for normalization and the truncatedmode set to 'lastp' (is like to put an horizontal line and merge all the lower clusters), for a better visualization of the dendograms shown above.

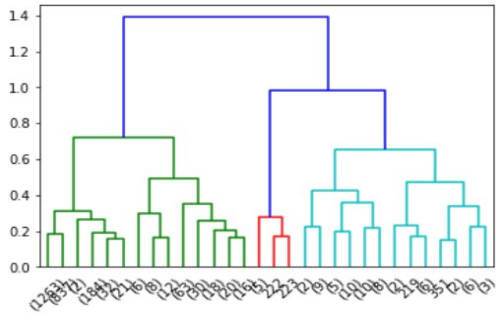
The number of n.clusters parameter is set to 2, while the affinity is set to "euclidean" (distance between the datapoints). Finally we have linkage parameter, which determines the metric used for the cluster merging strategy: it is set to "ward", which minimizes the sum of squared differences (the variant) between the clusters.

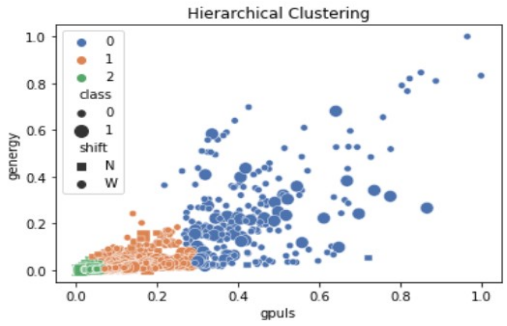
It can be also set to "complete", that minimizes the maximum distance between observations of pairs of clusters or average, that averages similarity between members, minimizing the average of the distances between all observations of pairs of clusters. Changing these three parameters and doing the graphs, we can state that ward, for our aim, is the best. With a Silhouette score of 0.64, we obtain the following graph and its relative dendogram:



We can seen that 3 clusters are found, and none representing an "outlier cluster". It's also interesting, for our next aim, to notice that most of the big point (i.e. with class 1) are in the green cluster, that is, with high energy and genergy.

Also for genergy and gpuls, we repeat the steps and obtain always 3 clusters, with a less Silhouette coeff., equal to 0.44:





3.4 Final consideration

After the analysis with the three clustering algorithms, although no algorithm outperforms the others, since no one of them can group the dataset into really specific informative cluster, it seems, by looking at the silhouette coefficient, that KMeans can better describe our data. In fact, as we can see from Figure 8, the first cluster engroup all working shift, in which the majority of class 1 data are in.

4 Classification

4.1 Classification by Decision Trees (18 pts)

4.1.1 The dataset

The dataset is divided in two parts: the training and the test set. This has been done with a proportion of 30% of data to the test and the rest for training our model. Moreover since the decision tree doesn't take into account the autocorrelation, the data is shuffled and partitioned with same frequency for the class attribute over the training and the test set.
Afterwards we build a very complex decision tree for the training set just to infer the importance of the attributes.
For the classification tree we use the instance "$class_weight = balanced$" which automatically adjust weights inversely proportional to class frequencies . It is basically equivalent to oversampling the minority frequency class and undersampling the majority one.
Feature importance are calculated like the difference between the impurity measure of the parent node minus the impurity of the children (weighted with the number data reaching the child node divided by the sample size of their father).
We use as impurity measure the GINI index which we want to minimize to gain more information. As a consequence the most important attribute correspond to the greatest important feature.
For all the attributes we get the following features:

f1: 0.21527777777777776

feature_names	importancies
nbumps	0.355211
gpuls	0.192952
gdpuls	0.129070
gdenergy	0.115832
energy	0.065272
genergy	0.053947
shift_W	0.051953
seismic_b	0.015876
seismoacoustic_a	0.013636
nbumps2	0.006250
hazard_b	0.000000
hazard_a	0.000000
seismoacoustic_c	0.000000
seismoacoustic_b	0.000000
maxenergy	0.000000
shift_N	0.000000
seismic_a	0.000000
nbumps5	0.000000
nbumps4	0.000000
nbumps3	0.000000
hazard_c	0.000000

Figure 14: F1 score and feature importance of all attributes
We see that many of them have value zero, which means that there is no gain in information in splitting over that attribute and we can omit them.
Moreover the attributes 'gdpuls' and 'gdenergy' seems to differ from the real value of their definitions so we've chosen to omit them and recalculate them as they are defined like the deviation of energy and puls from the average of the 8 previous shifts.

f1: 0.22068965517241376

feature_names	importancies
gpuls	0.354876
nbumps	0.321943
genergy	0.151354
energy	0.103792
seismoacoustic_a	0.045943
seismic_b	0.016029
nbumps2	0.006063
shift_W	0.000000

Figure 15: F1 score and feature importance but gdpuls and gdenergy

f1: 0.2229965156794425

feature_names	importancies
nbumps	0.357144
gpuls	0.202271
realgdpuls	0.129772
realgdenergy	0.116462
energy	0.065627
genergy	0.054241
shift_W	0.052236
seismic_b	0.015963
nbumps2	0.006284

Figure 16: F1 score and feature importance including gdpuls and gdenergy recalculated
We can see that by omitting those variable the f1 score increases and it increases more by adding the one recalculated from their definitions.

4.1.2 Finding the hyperparameters

We seek the best hyperparameters for our classification tree.
The first approach is to compare the error we commit on the train set with the one over the test set. In doing so we increase the number of nodes and calculate each time the error calculated with the following formula:

$$err = \frac{FP + FN}{FP + FN + TP} \quad (1)$$

Since the class 0 data outnumber the class 1 we omit the TN.
We get the following graph:

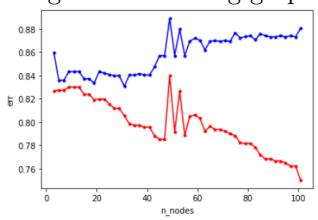


Figure 17: Error variation for increasing number of nodes. The red line is for the training set, while the blue for the test set.
From this graph we can see the model perform better on the test set in a range between 5-40 nodes after which the overfitting problem starts to raise.
Using this range for the nodes we use a GridSearchCV to find the all the best hyperparameters, which tries out all combination for the given parameters choosing those who maximize the F1 score averaged on the cross validation of the training set.
Those are the best 2 model found with greatest mean validation score (there were more than 1 model with same mean validation score over with we've chosen the simplest one for each value of the mvs):

```
Model with rank: 0
Mean validation score: 0.296 (std: 0.044)
Parameters: {'max_depth': None, 'max_leaf_nodes': 5, 'min_samples_leaf': 10, 'min_samples_split': 50}

Model with rank: 1
Mean validation score: 0.285 (std: 0.069)
Parameters: {'max_depth': None, 'max_leaf_nodes': 6, 'min_samples_leaf': 50, 'min_samples_split': 50}
```

4.1.3 Best model

Those are the classification tree of the two models.

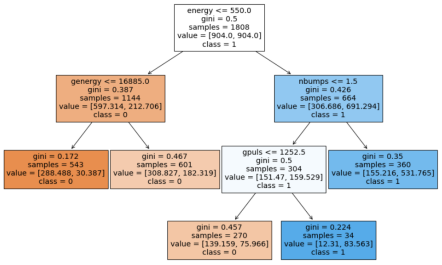


Figure 18: Rank 0 model tree.

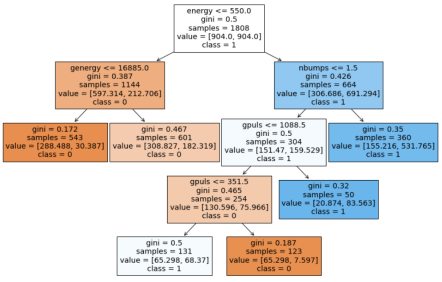


Figure 19: Rank 1 model tree.

We can see how those two trees differ from each other for the split on the attribute gpuls and the rank 1 model, undergoes to another split.

Anyway this further split doesn't improve the efficiency of the model it rather lowers the information gain.
In fact we've calculated such a coefficient starting from the 'gpuls' splitting attribute node. What we get is:
$IG_{model0} = 0.178$
$IG_{model1} = -0.647$
So we can see while the split on model0 actually get an information gain (hence a lower entropy), model1 gets a loss of information along the double split.
As well the classification error we commit on those model starting from the usual node are respectively:
$CE_{model0} = 0.283$
$CE_{model1} = 0.420$
Afterwards we can look at the confusion matrix computed on the train set for both models.

```
Cf Train model0
[[1376  313]
 [  38   81]]
```

```
Cf Train model1
[[1238  451]
 [  29   90]]
```

Figure 20: Confusion matrix computed for the training set for both models [[TN][FP]],[[FN][TP]]

By looking at this matrix seems like model 1 could be better because it actually gets more TP. But it actually fails a lot more on the FP.
We can check this problem by computing a cost matrix in an heuristic way, giving 0 weight to the TP and TN and since the class 1 objects are 7% of the total, we give 93 cost to the FN and 7 cost to the FP. The results we get are:
$cost_{model0} = 5877$
$cost_{model1} = 5970$
Here once again the model 0 seems to be a better representation of the problem.
We can plot the True Positive Ratio to the False Positive Ratio to get the ROC curve on both model.
We get the following curves and relative area:

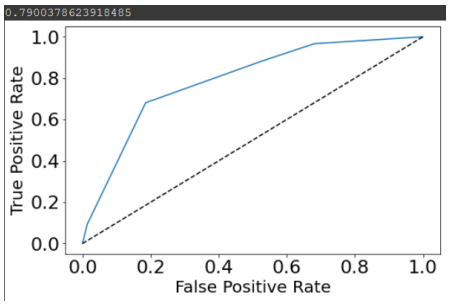


Figure 21: Roc curve for model 0. Area = 0.790

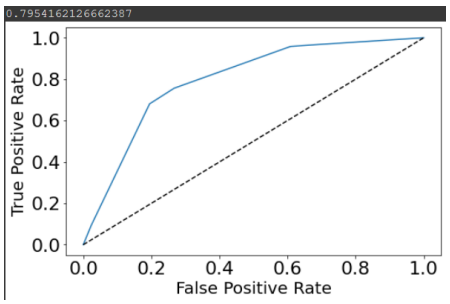


Figure 22: Roc curve for model 0. Area = 0.795

For which the model1 seems to be a little better. But since our data are unbalanced the precision-recall curve give a better understanding of the situation:

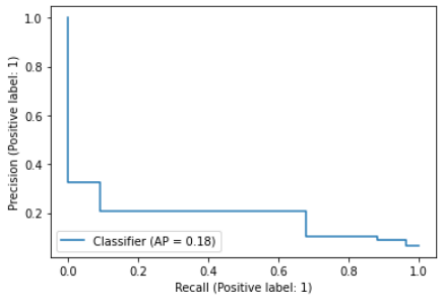


Figure 23: Precision-recall curve for model 0.

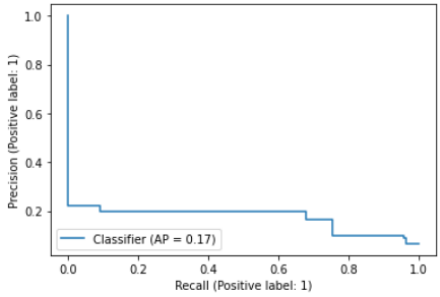


Figure 24: Precision-recall curve for model 1.

We can see how, once again, model0 is preferred to model1.

4.1.4 Test set

Setting model0 as the best model we've tried it out on the test set with the following results:

```
F1test model0 0.2790
Cf Test model0
[[591 134]
 [ 21  30]]
```

Figure 25: Confusion matrix for the test set using model0.

The ROC and precision-recall curves are:

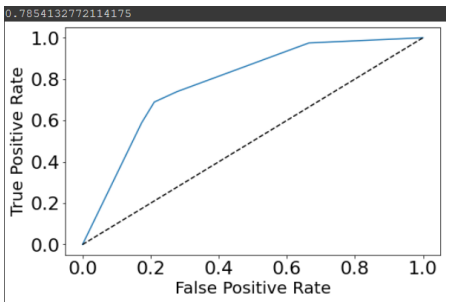


Figure 26: ROC curve for the test set (model0). Area = 0.785

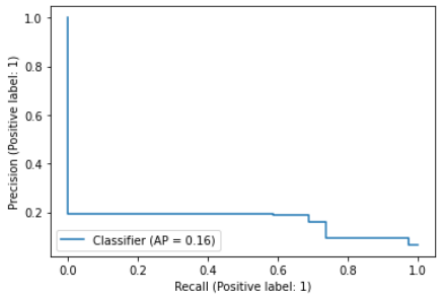


Figure 27: Precision-recall curve for test set (model0).

4.2 Classification by Other Algorithms or Baselines (6 pts)

4.2.1 Decision boundary

We split the data in two when the series of 0s of the energy showed up around at the 1000-shift.
The behaviour before and after seems to be quite different and we've performed the same method used previously to study each part.
For first period of time hence Period1:
Hyperparameters: max.leaf.nodes=10, min.samples.leaf=60, min.samples.split=40
mean validation score: 0.34 ± 0.01

```
F1 test: 0.26086
Test periodo1
[[142 105]
 [ 14  21]]
```

Figure 28: F1 score and confusion matrix for period 1.
Periodo 2:
Hyperparameters: max.leaf.nodes=6,min.samples.leaf=60, min.samples.split=70
mean validation score: 0.13 ± 0.01

```
F1 test: 0.10256
Test Periodo 2
[[260 169]
 [  6  10]]
```

Figure 29: F1 score and confusion matrix for period 2.
Comparing the fig.26 with the following:

```
Sum of confusion matrix p1+p2
[[402 274]
 [ 20  31]]
```

Figure 30: Confusion matrix for period1 and period 2. The number of TP is greater but from the cost matrix we can see how this model is more cost expensive.

4.2.2 K-NN

We've performed K-NN algorithm. It uses a metric distance to decide the neighbour so we've restricted the space to the first 2 feature who got the best importancy feature score for the classification tree, hence 'gpuls', 'genergy'.
To find the best number of neighbour we've plotted the f1 score for test and train set. We get the following graph:

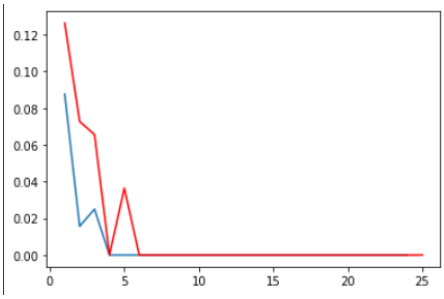


Figure 31: F1 score comparison between different K neighbours for test(red) and train(blue) set.

From this graph the best choice seems to be K=1 but has large error (underfitting) or K=2 for which we get the following f1 score and confusion matrix:

```
F1 score K=2: 0.25999
Confusion matrix K=2:
 [[689  36]
 [ 38  13]]
```

Figure 32: Results k=2

And by plotting the scatterplot between those 2 variable with the contour plot for the cluster we get:

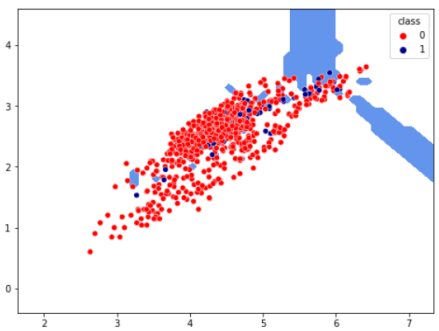


Figure 33: Graph k=2

4.2.3 Random forest

Here once again we use the grid search cross validation to evaluate the best tree hyperparameters with such an algorithm.
The best hyperparameters are:
'max.leaf.nodes': 10
'min.samples.leaf': 20
'min.samples.split': 70
Mean validation score: $0.303(std : 0.035)$
Applying this model on the test set return:

```
F1 0.278481
cf
 [[572 153]
 [ 18  33]]
```

Figure 34: F1 score and cf for random forest

ROC curve and precision-recall:

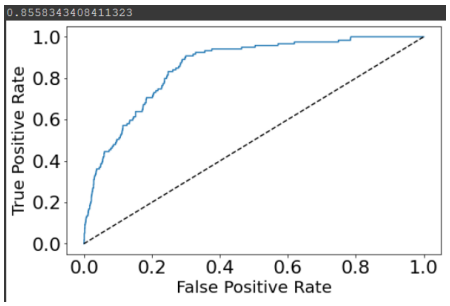


Figure 35: ROC curve. Area = 0.855

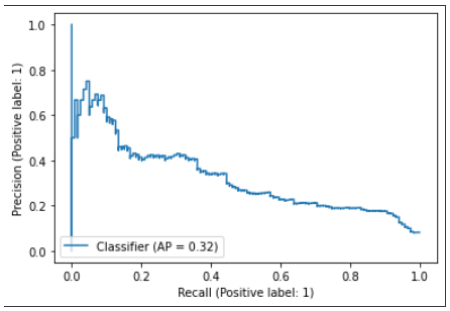


Figure 36: precision-recall

4.3 Final discussion

In the end from the confusion matrix and f1score the best model seems to be the one given from the random forest algorithm, which provides the less cost from the cost matrix.
While the K-NN provides the less informative model since it has a big proportion of FN.

5 Pattern Mining

In this last part, we will use association rules to extract some more information to our dataset. The association rule mining problem formally says that given a set of data, you have to find all the rules having support and confidence higher than two thresholds, respectively

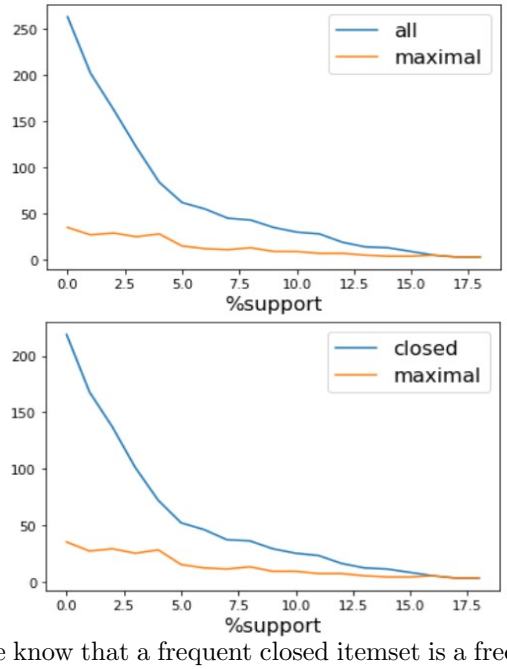
minsup and *minconf*. Of course we won't compute the support and confidence for every possible rule, using instead the Python library *apriori*.
A common strategy adopted in these cases is to decompose the problem into two major sub-tasks. The first one is to find all itemsets that satisfy the *minsup* threshold (frequent itemsets) and then extract all the high-confidence rules from those itemsets (strong rules).

5.1 Frequent Pattern extraction and discussion

First of all we have to discretize our data. The Python documentation says that *Continuous values can be discretized using the cut (bins based on values) and qcut (bins based on sample quantiles) functions*. For instances, we select only to some features (e.g. seismic, shift, nbumps, energy, genergy, gpuls), binning the continuous values (last three) as mentioned before into 4 bins, while nbumps into three (no bumps, 1 to 3 bumps or more than 3), obtaining the follow (head) dataset.

	class	shift	nbumps	energy_bin	genergy_bin	gpuls_bin
0	Not Dangerous	Preparation Shift	No bumps	(-0.001, 300.0]_energy	(11660.0, 25485.0]_genergy	(1.999, 190.0]_gpuls
1	Not Dangerous	Preparation Shift	1-3 bumps	(300.0, 4000.0]_energy	(11660.0, 25485.0]_genergy	(1.999, 190.0]_gpuls
2	Not Dangerous	Preparation Shift	No bumps	(-0.001, 300.0]_energy	(99.999, 11660.0]_genergy	(1.999, 190.0]_gpuls
3	Not Dangerous	Preparation Shift	1-3 bumps	(300.0, 4000.0]_energy	(25485.0, 52832.5]_genergy	(1.999, 190.0]_gpuls
4	Not Dangerous	Preparation Shift	No bumps	(-0.001, 300.0]_energy	(11660.0, 25485.0]_genergy	(1.999, 190.0]_gpuls

We use the Apriori function, firstly doing some guesses to find and set best parameters. In particular we are interested in the minimum support and the minimum number of items in the future itemset (zmin). As documentation reports, the parameter *target* lets the algorithm look for all, maximum and close itemsets respectively for "a", "m" and "c". We know from the theory that maximum itemsets are a subpart of closed one, that are subpart of all itemsets.
We can observe from the figures above that as the support threshold decreases, the number of frequent itemsets increases. That's what we expect: there are only a few itemsets with very large support, and vice versa.



So we know that a frequent closed itemset is a frequent itemset that is not included in a proper superset having exactly the same support, thus must be a subset of the set of frequent itemsets. One prefers to use them because the set of frequent closed itemsets is usually much smaller than the set of frequent ones and it can be shown that no information is lost (all the frequent itemsets can be regenerated from the set of frequent closed itemsets). But not in this case, in which all frequent and closed frequent itemsets are of the same order. It means that almost all of itemsets has no superset with the same frequency.
Using target='a', supp=10, zmin=4, we found the following itemsets (we report only some of those in the table 1.) As we expected, we found that the most frequent itemsets are those shift (no matter if the workers are in the coal or not) in which no bumps (or at least some "light" bumps) occur, so itemsets classified with class 0. As we didn't find any class 1, we repeat the calculus but lowing the threshold to 3, same for zmin=3. Out of more than 200 frequent itemset, we found only three frequent itemsets in which 'Dangerous' (so class 1) appears. Those are, with their respectively support, ('Dangerous', '(669.0, 4518.0]_gpuls', 'At Work') with support 3.32, ('Dangerous', '(52832.5, 2595650.0]_genergy', 'At Work') with support 3.44 and ('Dangerous', '1-3 bumps', 'At Work') with support 3.40. As we could see from Figure 4, we have found more Work shift with class 1.

5.2 Association Rules extraction and discussion

The next step in our pattern mining task is to find the association rules. As before, we use the apriori module and try different combination of confidence, support and zmin values. We notice that no itemsets with class 1 appear with high minsup, as evidenced by the fact that our dataset is very unbalanced, that is, as seen before, out of 2584 records there are only 6.5% hazardous instances. So adjusting the parameters (written in table 2), we found some interesting association rules. First of all we notice that lowering the support, the lift is increasing. Secondly, we found some energy, gpuls and genergy ranges that suggest us the shift (most of the time, "at work" shift) may be dangerous.

Features	Support (%
('No bumps', 'At Work', 'Not Dangerous', '(-0.001, 300.0]_energy')	28.59
('Preparation Shift', 'No bumps', 'Not Dangerous', '(-0.001, 300.0]_energy')	26.50
('(1.999, 190.0]_gpuls', 'No bumps', 'Not Dangerous','(-0.001, 300.0]_energy')	17.99
('(99.999, 11660.0]_genergy', '(1.999, 190.0]_gpuls', 'Preparation Shift', '(-0.001, 300.0]_energy', 'Not Dangerous')	13.47
('(669.0, 4518.0]_gpuls', '1-3 bumps', 'At Work', 'Not Dangerous')	10.56 height

Table 1: Table of some frequent itemsets found with the Apriori algorithm, using min_sup = 10.

Association Rules	Support (%)	Confidence	Lift
('(4000.0, 402000.0]_energy', '(669.0, 4518.0]_gpuls', 'At Work') → Dangerous	1.78	0.21	3.29
('(4000.0, 402000.0]_energy', '(52832.5, 2595650.0]_genergy', 'At Work') → Dangerous	1.97	0.21	3.23
('(669.0, 4518.0]_gpuls', '(52832.5, 2595650.0]_genergy', '1-3 bumps', 'At Work') → Dangerous	2.12	0.21	3.1
('(669.0, 4518.0]_gpuls', '(52832.5, 2595650.0]_genergy', '1-3 bumps') → Dangerous	2.16	0.20	3.07

Table 2: Table of some Association rules found with the Apriori algorithm, using min_sup = 8, zmin=4, conf=20.