UNIVERSITÀ DI PISA

DEPARTMENT OF COMPUTER SCIENCE

MSc in Data Science and
Business Informatics

Data Mining 1

Seismic bumps

Authors

Giovanni Battista D'Orsi 639186
Davide Di Virgilio 641917
Sara Lelli 578085
Lia Trapanese 628153

2021/2022

Abstract

The chosen dataset is a collection of seismic movements in the Zabrze-Bielszowice coal mine, in Poland. Whether the extraction of minerals is not predicted, it can cause threats not only to humans' lives (causing health problems), but also to the Earth's life: in fact, it is proved that the extraction of minerals has an impact of 37% of the earthquakes caused by human activities (induced seismicity). The aim of this project is to analyze and predict the risks of seismic bumps during mining activities. We will use clustering and pattern mining to see if there is any hidden structure or patterns in our data. Classification will be used instead to predict the target variable *Class*.

1 Data Understanding

1.1 Data Semantics

We were provided a dataset called "seismic-bumps.csv" in which we have 2584 instances and 19 attributes: the last one is the target variable *Class*, a binary variable. According to the dataset information, the attributes are the following:

- **ORDINAL**:
  - **Seismic**: result of shift seismic hazard assessment in the mine working obtained by the seismic method. This variable could assume four values: a - lack of hazard, b - low hazard, 3 - high hazard, 4 - danger state;
  - **Seismoacoustic**: result of shift seismic hazard assessment in the mine working obtained by the seismoacoustic method. This variable could assume four values: 1 - lack of hazard, 2 - low hazard, 3 - high hazard, 4 - danger state;
  - **Hazard**: result of shift seismic hazard assessment in the mine working obtained by the seismoacoustic method based on registration coming from GMax only. This variable could assume four values: 1 - lack of hazard, 2 - low hazard, 3 - high hazard, 4 - danger state;
- **CATEGORICAL**:
  - **Shift**(binary): information about the type of a shift. This variable could assume two values:
    - W - coal-getting;
    - N - preparation shift;
      Every single shift has a duration of 8 hours;
  - **Class**: this is the decision attribute, which can assume two values:
    - '1' means that a high energy seismic bump occurred in the next shift ('hazardous state');
    - '0' means that no high energy seismic bumps occurred in the next shift ('non-hazardous state').
- **NUMERICAL**:
  - **Nbumps**: the number of seismic bumps recorded within previous shift;
  - **Nbumps2**: the number of seismic bumps (in energy range $[10^2, 10^3)$ measured in Joule) registered within previous shift;
  - **Nbumps3**: the number of seismic bumps (in energy range $[10^3, 10^4)$ measured in Joule) registered within previous shift;
  - **Nbumps4**: the number of seismic bumps (in energy range $[10^4, 10^5)$ measured in Joule) registered within previous shift;
  - **Nbumps5**: the number of seismic bumps (in energy range $[10^5, 10^6)$ measured in Joule) registered within previous shift;
  - **Nbumps6**: the number of seismic bumps (in energy range $[10^6, 10^7)$ measured in Joule) registered within previous shift;

  - **Nbumps7**: the number of seismic bumps (in energy range $[10^7, 10^8)$ measured in Joule) registered within previous shift;
  - **Nbumps89**: the number of seismic bumps (in energy range $[10^8, 10^{10})$ measured in Joule) registered within previous shift;

- **CONTINUOUS**:
  - **Genergy**: seismic energy recorded within previous shift by the most active geophone (GMax) out of geophones monitoring the longwall;
  - **Gpuls**: is the number of pulses recorded within previous shift by GMax;
  - **Gdenergy**: a deviation of energy recorded within previous shift by GMax from average energy recorded;
  - **Gdpuls**: a deviation of a number of pulses recorded within previous shift by GMax from average number;
  - **Energy**: total energy of seismic bumps registered within previous shift;
  - **Maxenergy**: the maximum energy of the seismic bumps registered within previous shift;

Variables are all integers, except for '*Seismic*', '*Seismoacoustic*', '*Shift*' and '*Hazard*', which are objects (chars). All the observations *Seismic* assume only cases 'a' and 'b', *Seismoacoustic* only 'a', 'b', 'c', and *Hazard* has values 'a', 'b', 'c'. So, we can conclude that no seismic movements with maximum hazard levels have been detected. Through a statistical analysis, we notice that the mean and the median have values that are not very similar to each other: this means that the values of the dataset are not uniformly distributed.

We also observe that the decisional attribute *Class* takes the value 1 ('hazardous state') when the value of the attribute *Energy* is greater than or equal to 10^4 J. This correlation is directly connected to the attributes *Nbumps4* and *Nbumps5*: they indicate how many bumps have been measured in a given shift with an energy value ranging from 10^4 to 10^5 for *Nbumps4* and from 10^5 to 10^6 for *Nbumps5*. In particular, if the decision attribute has the value 1, then in the next shift any seismic bump with an energy higher than 10^4 J will be registered, so *Nbumps4* or *Nbumps5* have values greater than 0.

During the study, we have noticed that the dataset does not present missing values, but there are 6 duplicate rows. Duplicate rows in the seismic bumps data set can be present/detected for different reasons: firstly, the values duplicated can be mistakes made by the machines used during the data collection; secondly they can actually not be duplicated rows, but, because they are in different shifts, they can be related to different moments. Anyway, we decide to keep them, because they are sequentially related and deleting them could provoke an alteration in the data set.

The variables "*Nbumps*(N)" have values ranging from 0 to 9. We noticed that, according to the statistical measures and the histograms shown in ***figure 1***, the value of the attributes "*Nbumps6*", "*Nbumps7*" and "*Nbumps89*" is equal to zero: it means that no seismic movements with energy between 10^6 and 10^{10} were recorded in the previous shift. Thanks to this expedient, we have decided to drop these features inside the dataframe.

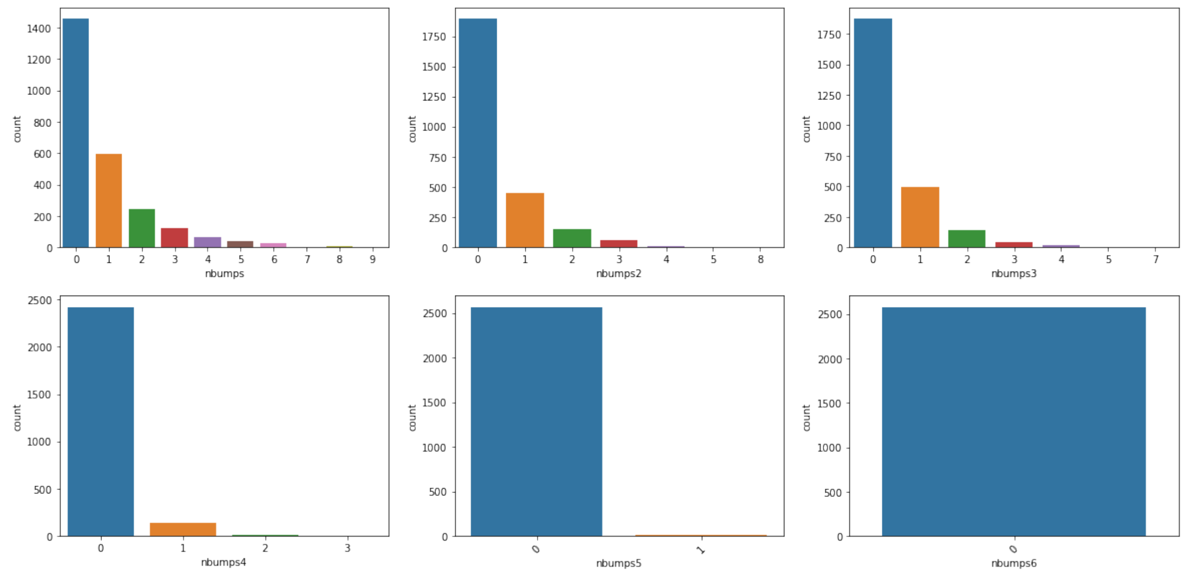


Figure 1: In the graph it is reported only nbump6 to demonstrate that it has zero values, as nbump7 and nbumps89.

2 Data Visualization

2.1 Analysis of the Categorical values

In this section we analyse the distribution of some particular attributes and try to understand what variables can explain the overall *Class*.

As we notice in the barcharts during our analysis, the target variable (*Class*) is highly imbalanced. Occurrences with *Class* equal to zero are 2414 and, on the other hand, cases with *Class* equal to 1 are 170, that is 7.04% of the total instances of *Class*. It means that in the next shift we don't have a lot of cases with high levels of energy. In fact, we see that the *Seismic*, *Seismoacoustic* and *Hazard* attributes, even though they are based on different registration methods, show values that confirm our theory (***figure 2***).

Moreover, in line with the following bar plot (***figure 2***) we can claim that the coal-getting *Shift* presents more high energy bumps instances than the preparation *Shift*. In fact, during the preparation *Shift* (N) we have 904 instances of *Class* 0 (non-hazardous state) and 17 of *Class* 1 (hazardous state). On the other hand, during the coal getting *Shift* (W), there are 1510 occurrences of *Class* 0 and 153 of *Class* 1.

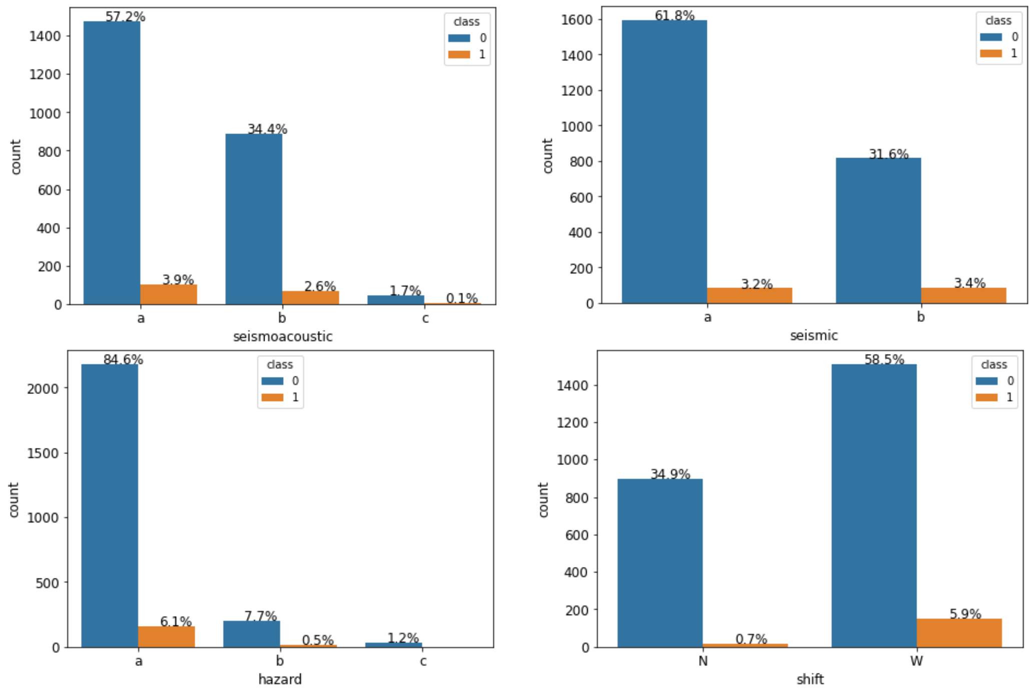


Figure 2: Graphical analysis of categorical attributes.

2.2 Analysis of the continuous values

First of all, looking at the histograms that demonstrate the Gaussian distribution of these attributes, we notice that we have outliers. In fact, we can clearly see that most of the values are concentrated around the average, as soon as the values increase, we have their distributions verged to 0 because the graphs are right skewed.

This phenomena is also evident by looking at the value of the standard deviation which is very high. In the ***figure 3*** there are the logarithmic distributions of the attributes *Gpuls* and *Genergy*, that demonstrate the previous statement. These two attributes are the ones that mainly characterize artificial seismic bumps, so we decide to study them to extract useful information about this phenomenon and what provokes it. Because of the wide range of the observation values, we decide to use the logarithmic scale: thanks to this normalization, the following graphs are more readable.

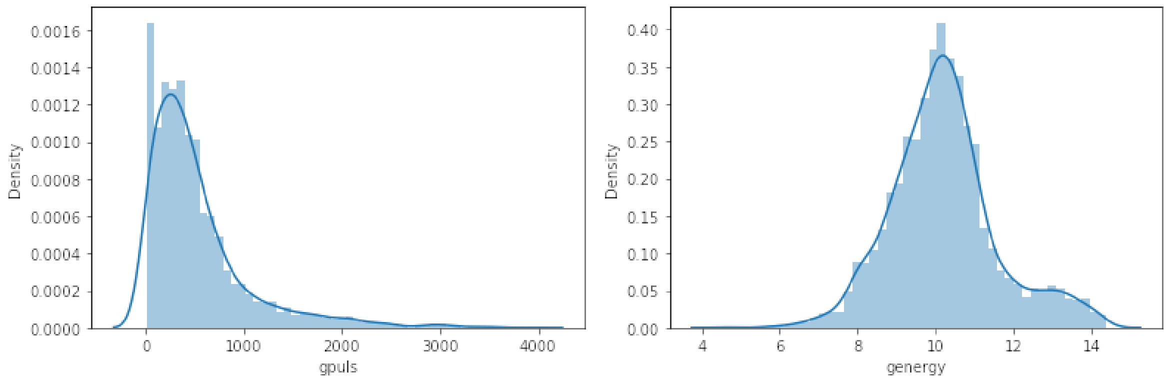


Figure 3: Graphical analysis of continuous attributes.

2.3 Data quality analysis

For a better understanding of the *Gpuls* and *Genergy* values, we look at the corresponding *Class* of those instances that are in the right skewed section of the histogram in ***figure 3***: if these instances have *Class* equal to 1, we can state that they're not outliers, but they are interesting values because they detect a dangerous situation in the next shift. Whether they have *Class* 0, we assume that they can be errors from the energy-detection machines, either noise inside the dataset or not interesting cases.

Having these considerations in mind, we use coding and boxplots (***figure 4***) to see the target variable of those instances. First we take in consideration the *Energy* attribute: it has twelve rows with *Class* equal to 0 and energy higher than 10^5 J, while it has just two rows with the same energy but *Class* 1. The same considerations are made for the *MaxEnergy* and *Genergy* attributes with *Class* equal to 0, where we have, in sequence, 11 rows with *MaxEnergy* higher than 10^5 J and 8 rows with *Genergy* higher than 1.8×10^6 J. Then, we drop the rows with *Class* equal to 0, considering these points in our dataset as outliers.

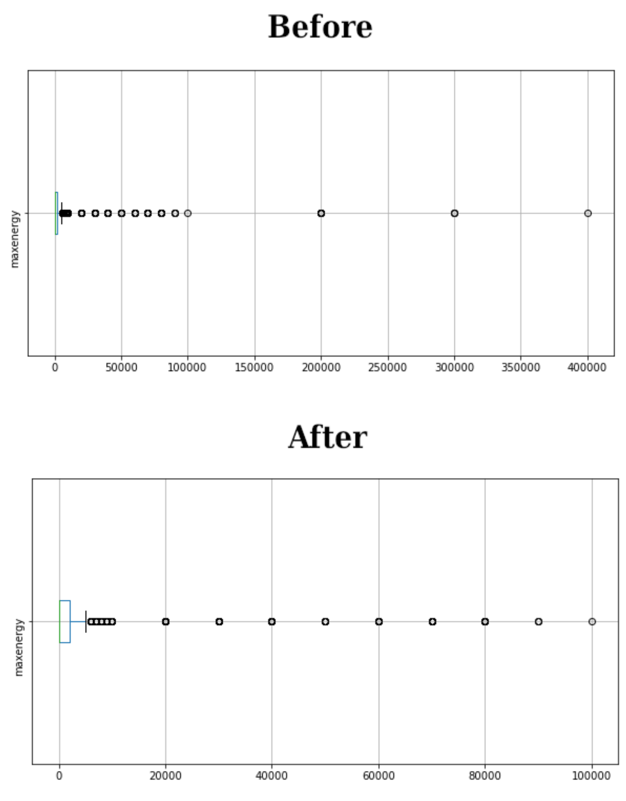


Figure 4: Example of a boxplot before and after the deletion of outliers; we can assume that, after this operation, the values are more compact.

2.4 Correlation analysis

After the outliers' discovery, we create a heatmap that shows a correlation between all the numerical attributes in the dataset. In the figure below (***figure 5***) we see that there is a strong and relevant relation between the following attributes:

- **Maxenergy and Energy (99% correlated)**
- **Maxenergy and Nbump5 (81% correlated)**

- **Gpuls and Genergy (75% correlated)**
  - The standard deviations of them are represented by *Gdpuls* and *Gdenergy*, so they are correlated as well.

Due to the strong correlation between *Maxenergy* and *Energy*, we decide to drop the attribute *Maxenergy* because *Energy* is the attribute at which the target variable refers. Moreover, this correlation is easily visible by looking at a scatter plot in which we analyze the dependence among these two variables.

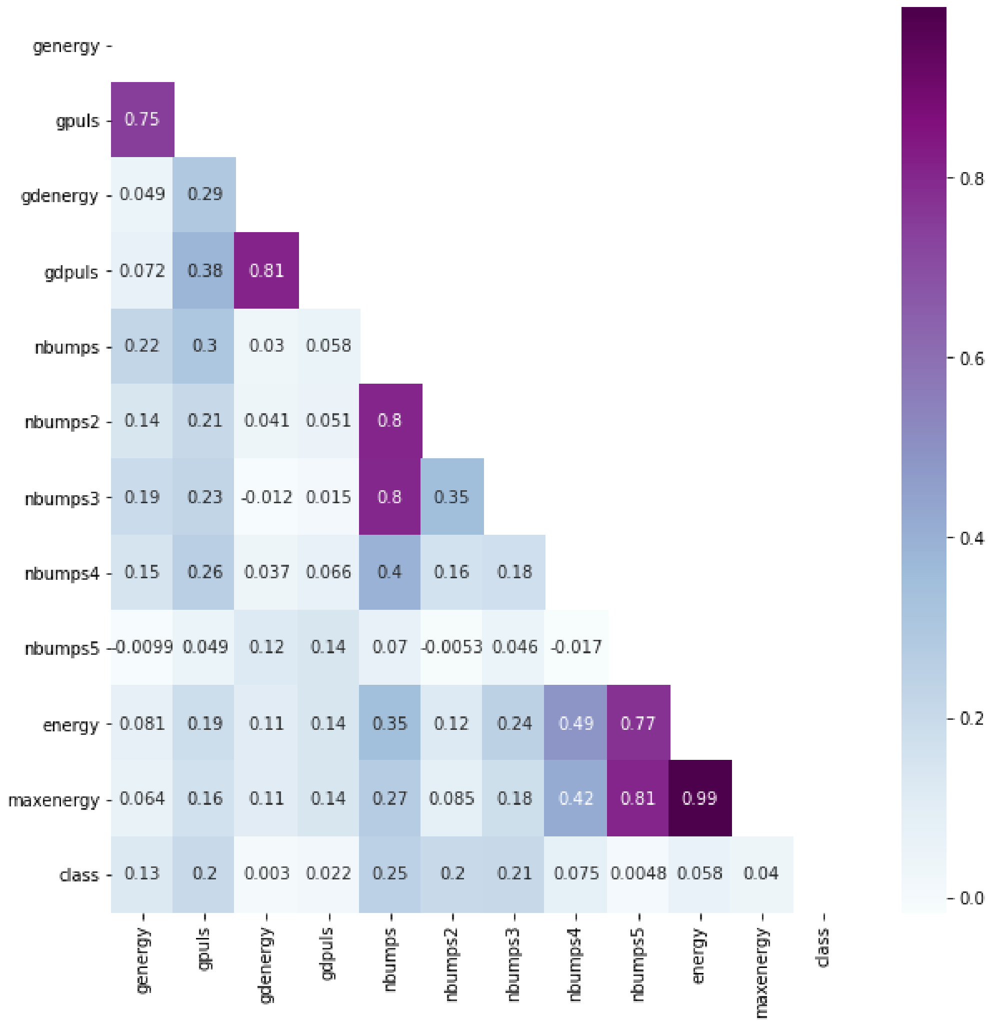


Figure 5: Correlation matrix.

We consider the correlation between *Genergy* and *Gpuls* interesting: so, for a better examination, we use a scatter plot. As the ***figure 6*** shows, we see that as soon as the values of both variables increase, they become more sparse and irregular. We can also observe that the majority of the observations having *Class* 1, have *Genergy* values less than ~0,20*1e^6 and *Gpuls* less than 1500. While the most records belonging to *Class* 0 have a *Gpuls* that reach a maximum value of ~2000 and *Genergy* ~0,40*1e^6. We also notice that, as *Gpuls* values increase we don't have high *Genergy* values, but still we can see that there are instances with *Class* equal to 1 (hazardous state) (right part of ***figure 6***).

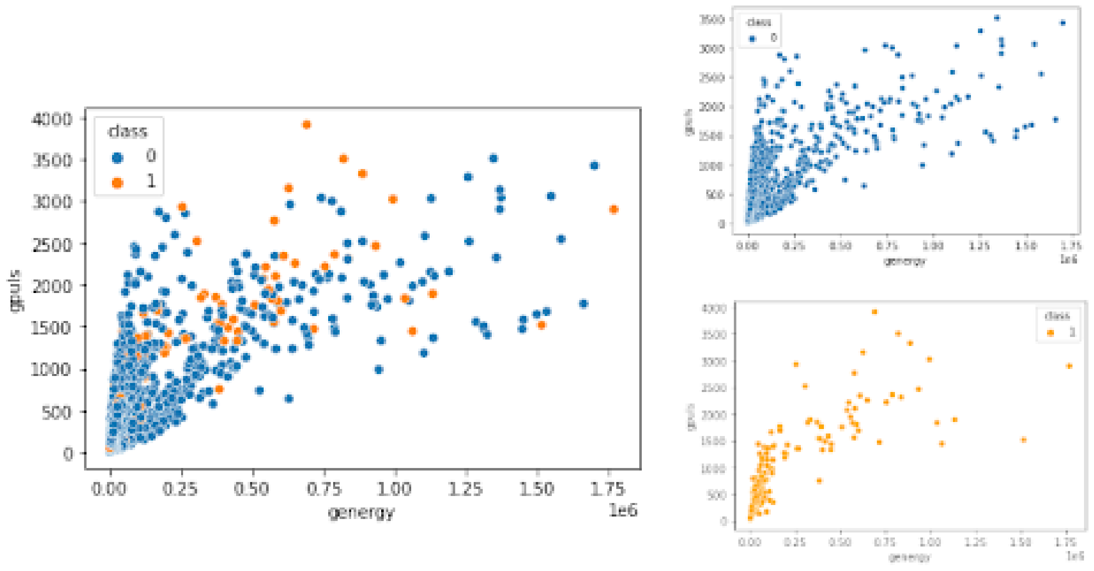


Figure 6: Scatterplot.

3 Clustering

Before starting the cluster analysis, the dataset has to be filtered, selecting only numerical attributes and, then, it has to be normalized because the attributes have different ranges of values. As a scaling metric, we tried both Robust scaler and MinMax scaler. However, we decide to use the first method, because in our case we are in the presence of many outliers: this algorithm can deal better with them, scaling features using statistics that are robust to outliers.

3.1 K-means clustering

The K-Means clustering algorithm is performed by using the Euclidean distance as proximity metric. In order to reach convergence, we initialized our centroids by running K-Means 100 times with different centroid seeds. Firstly, we calculate the SSE (Sum of Squared Error) for K and the Silhouette coefficient for understanding which K number of clusters is the best. We plot the Elbow method graph between the SSE and K to see at which point the curve changes direction: we clearly notice that the elbow point is in correspondence of K equal to 3.

Moreover, the KElbowVisualizer plot shows that, even in this case, the highest value of the Silhouette coefficient (0.78) is found at the value of K equal to 3, which confirms the statement made before (***figure 7***).

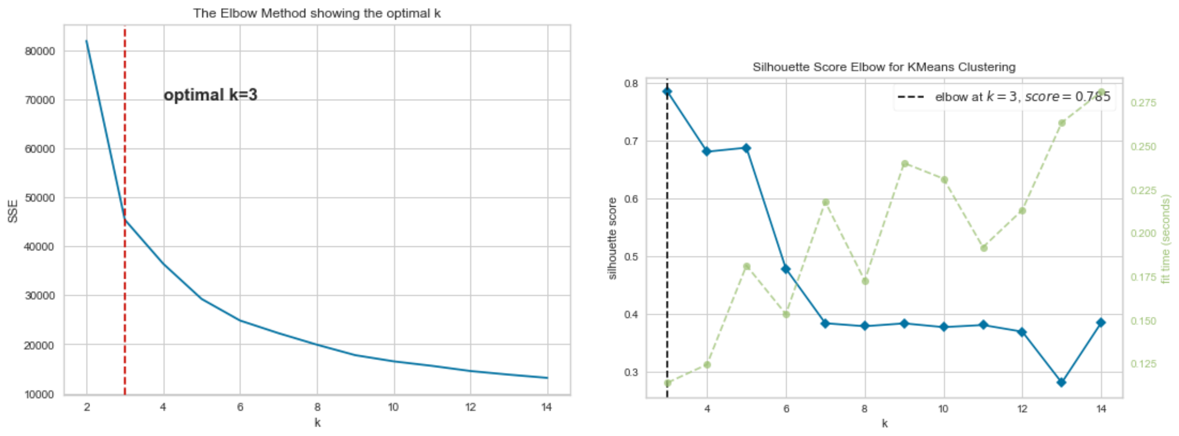


Figure 7: Elbow Method and Silhouette Score Method

The K-Means algorithm yielded interesting results. The data points are distributed among clusters as follows: { Cluster 0: 96; Cluster 1: 2347; Cluster 2: 120 }

Plotting all the possible combinations of the features, we notice that most of the plots are less meaningful because of the overlapped cluster structure, as shown in the right part of the ***figure 8***.

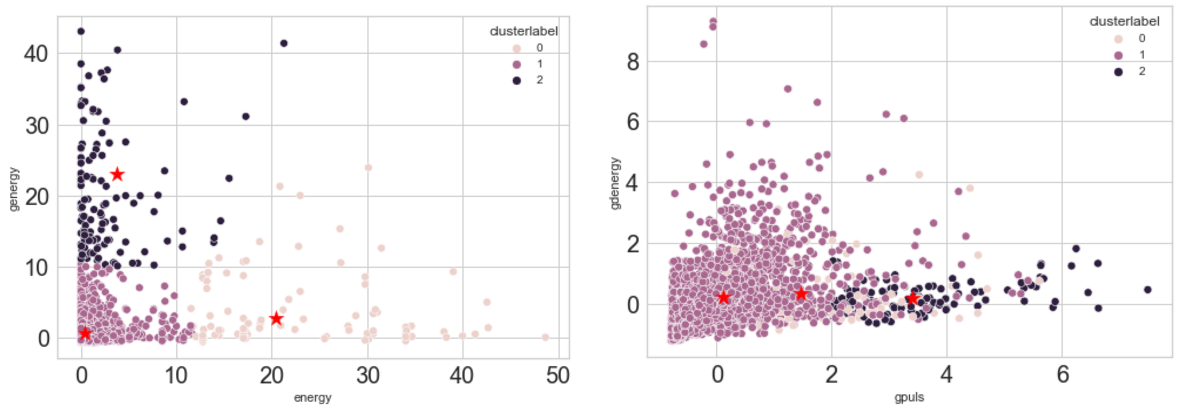


Figure 8: Energy/Genergy clustering VS. Gdenergy/Gpuls clustering.

The other plot chosen (*Energy* and *Genergy* clustering) shows that:

- **Cluster 0**: has low values of both *Genergy* and *Energy*.
- **Cluster 1**: is characterized by high values of *Genergy* (>10) and low values of *Energy* (<10).
- **Cluster 2**: groups records having high *Energy* (>10) and low *Genergy* (<10).

Even though the *Genergy* and *Energy* plot seems interesting, we need to keep in mind that K-Means is a partitional algorithm: it means that outliers and noise points are part of clusters as well. So, we can declare that K-Means doesn't fit well with our sparse dataset. In the graph below (***figure 9***) we group the records by cluster label and then we take the average of each attribute in such a way that, when we plot the graph, the differences between clusters are more readable and evident. In fact, in all three clusters the occurrences of *Gdenergy* and *Gdpuls* are the same. Cluster 2 is characterized by high values of *Nbumps* and *Energy*, whereas in cluster 1 we can find instances with high *Genergy*. Lastly, we notice that the less meaningful cluster is cluster 0 because it has low value of all features.

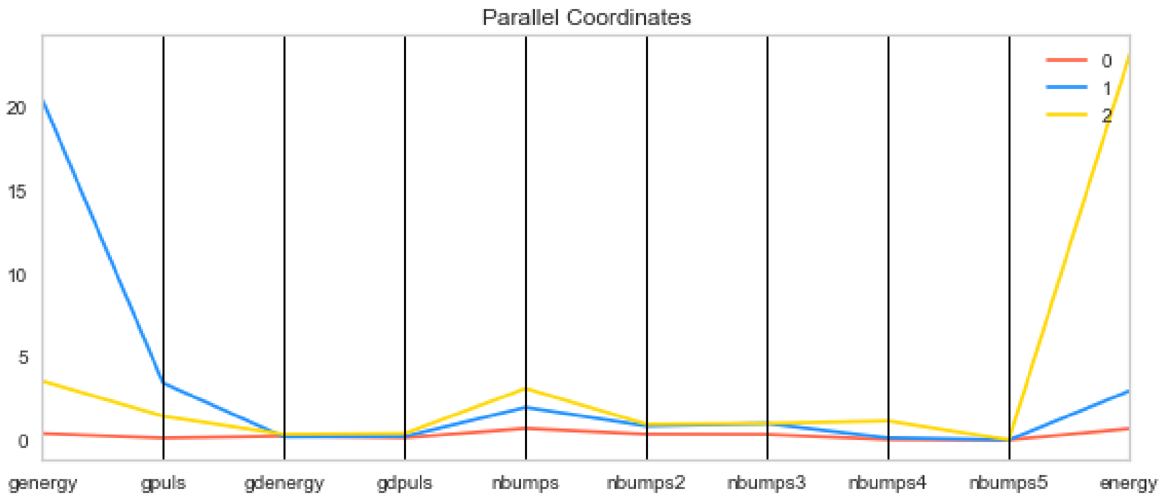


Figure 9: Parallel plot of the three clusters.

Then, we study the disposition of the target variable (*Class*) in those three clusters. As we stated before, the values of *Class* are mostly equal to 0: as we can see in ***figure 10***, the 86% of these belong to cluster 0, the remaining part is uniformly distributed, instead, in clusters 1 (3,6%) and cluster 2 (3.2%). On the other hand, the occurrences of *Class* 1 are generally arranged in cluster 0, the 5.0%, while the other clusters present very few of them.

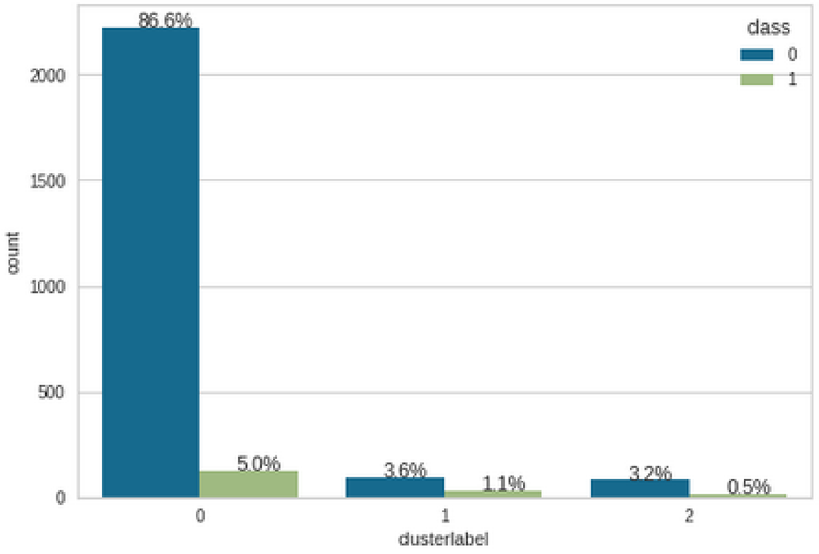


Figure 10: Distribution of the target variable in the three clusters.

3.2 DBSCAN

In this section we explore the results obtained by applying the DBSCAN clustering algorithm.

DBSCAN is surely a better clustering method for our dataset than the K-means: in fact, as the chosen dataset is very sparse, it can deal with outliers and noise points in a better way.

After several combinations, setting the minPoints as 10 and looking at the K-distances plot, we start by choosing an epsilon in range [1.8, 3.5), but we notice that, with this setting, DBSCAN is not able to identify any meaningful clusters, since it only groups the data into one big dense cloud, leaving out 115 points labeled as noise points. So, we decide to increase epsilon to 5, a value that gives as results 3 clusters, and identifies 32 noise points (the red ones).

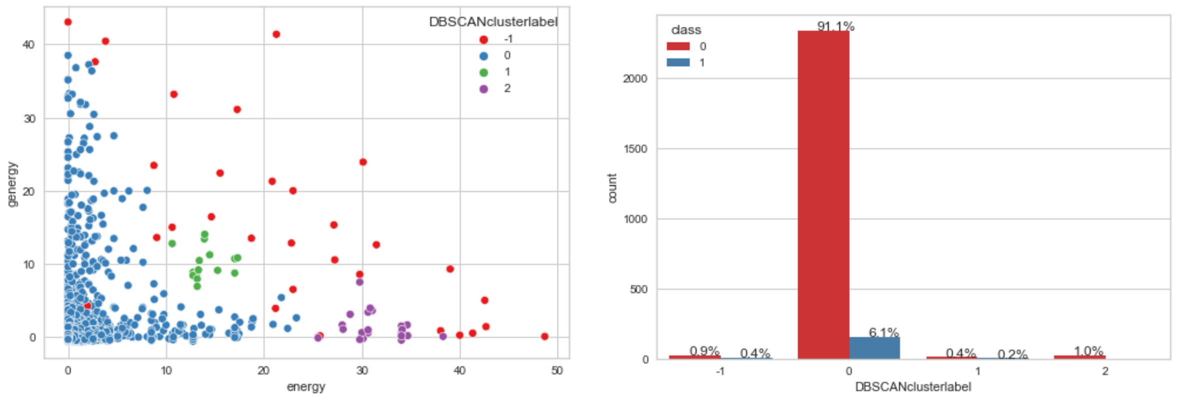


Figure 11: DBSCAN clustering and target variable distribution.

As an example, ***figure 11 (left part)*** shows the DBSCAN clustering between *Energy* and *Genergy*. The first thing we notice is that the clusters are unbalanced: 2490 points to cluster 0; 26 points to cluster 2; 15 points to cluster 1. Although it seems that the three clusters are well separated, this is not completely true: there are few points in cluster 1 and 2 that are hiding behind cluster 0. In our opinion, it happens due to varying density in that specific area.

Looking at the K-Means *Energy/Genergy* clustering (***figure 8***) and this new plot, we can clearly see the differences between those two methods: K-means chooses mean centroids, so that no points are left outside and the clusters are more balanced, while DBSCAN works with the radius (epsilon) and the density of the points.

As we see in the right part of ***figure 11***, records with attribute '*Class*' equal to 0 are highly concentrated in the denser region of points (cluster 0), while in cluster 1 and 2 are almost absent (***figure 11***). Points that are classified as noise points have both values of the attribute *Class* very low; we can also state that the majority of the records with *Class* 1 is distributed in cluster 0, while in cluster 2 there are no points with *Class* equal to 1.

3.3 Hierarchical clustering

We proceed with the hierarchical clustering analysis. First of all, as with the other clustering algorithms, we need to normalize the dataset and prepare it before plotting.

We try to analyze the behaviour, in terms of Silhouette score, of each clustering methods (the complete linkage, the single link, the group average and Ward's method) as we vary the parameter K (range from 2 to 12) using two proximity metrics: Euclidean and Manhattan.

As we can observe from ***figure 12***, the silhouette score of Ward's method, using the Euclidean distance, drops drastically when K goes from 3 to 4 (drop of 46%), whereas for the other three methods the score slightly drops when the number of clusters becomes more than 3. In addition to this, we notice that, apart from the average and complete linkage, the single link reports a large drop when clusters go from 8 to 12. We can deduce that the best K in this case is equal to 3.

Instead, using the Manhattan distance, the Silhouette score (SS) of single and complete links goes down from 3 to 4 and, then, the first drops drastically from K= 9 to 10 and the second one from 6 to 7; successively they have a steady trend. Whereas, the average link goes down until K equal to 3 and then decreases slowly.

In the Manhattan line plot the highest SS is obtained when K is equal to 2, so we perform the clustering and we build the related dendrograms and, then, we compare the results with the outcome of K=3, mined from the Euclidean distance. Because in this case Manhattan distance is not able to create meaningful clusters, we decide to use K=3.

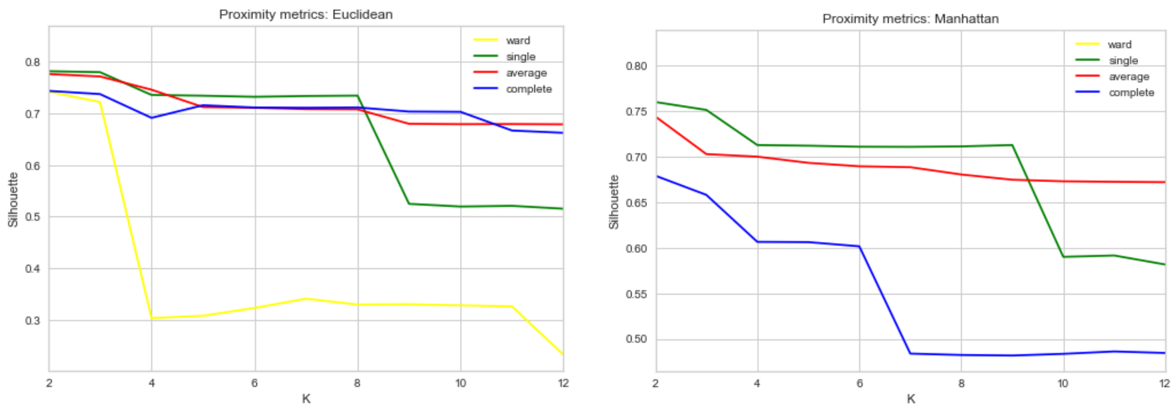


Figure 12: Proximity Metrics with euclidean and Manhattan distance.

Interesting clusters are produced by Ward's method with Euclidean distance and number of clusters = 3. The ***figure 13*** shows the best and the worst method for scatter plot distributed on two dimensions considering the attributes *Genergy* and *Energy*: the Single and the Ward's ones.

The first is bad for our dataset, because it's sensitive to noise and outliers. In fact, in the single-link clustering there is just one big red cluster while the other two contain respectively two and one point. On the contrary, the second method presented (Ward's method) it's very similar to the other two as they have equivalent characteristics: they're less susceptible to noise and they're biased towards globular clusters.

As a matter of fact, the clusters created are very similar to each other. They're different in the composition of the cluster: the complete and the average method tend to split big clusters using, respectively, the maximum and the average distances, while Ward's method depends on the squared error. So, we can deduce that the best hierarchical method is actually Ward's method.

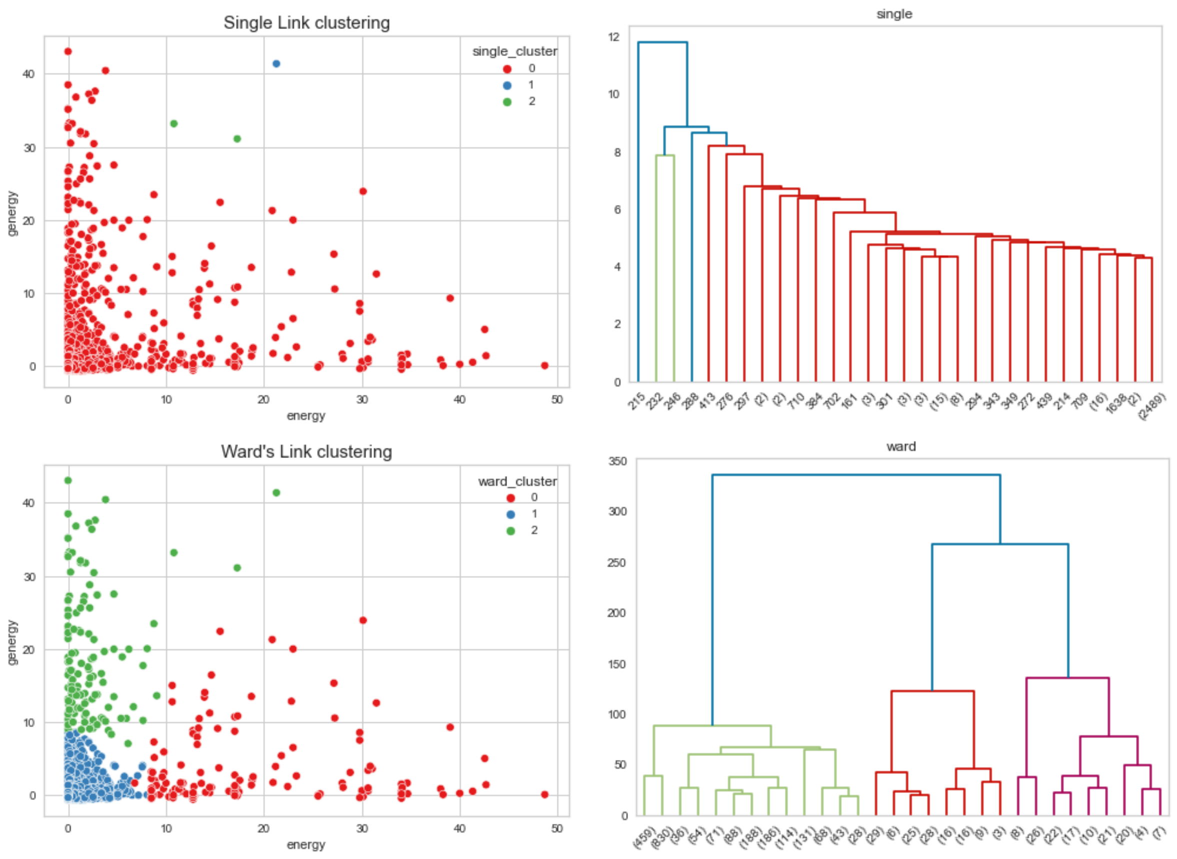


Figure 13: Proximity Metrics with euclidean and Manhattan distance.

3.4 Final Discussion on clustering analysis

Between all the clustering methods analyzed, we can see that there is a similarity between the K-means and the Hierarchical algorithm, especially for Ward's method. As we have a very sparse dataset, it's clear that creating clusters based on some kind of algorithms is quite difficult. We can use the last algorithm analyzed (Hierarchical - Ward's) as it can deal a little bit better than K-means with outliers, but it's not specific enough for them as the DBSCAN. On the other hand, DBSCAN doesn't show strong division between clusters, but we can conclude that maybe it's the best method for our dataset.

4 Classification

4.1 Preprocessing

For the classification process, the dataset has to be cleaned from missing values and useless features. Before dropping unimportant attributes, we have trained the decision tree using all the features in the original dataset, with the result that not all the features have to be used in the process. After building the model we have computed the feature importances (***figure 14***) noting that some attributes have values equal (or close) to 0, hence they are irrelevant for the classification.

Then we have decided to re-train a new decision tree removing the features that have an importance equal to 0 to see if the results are different or not. For the new analysis we have considered the features *Genergy*, *Gpuls*, *Gdenergy*, *Nbumps*, *Energy* and *Class*. As we will see later on, the usage of those features brings improvement in the performances of our model.

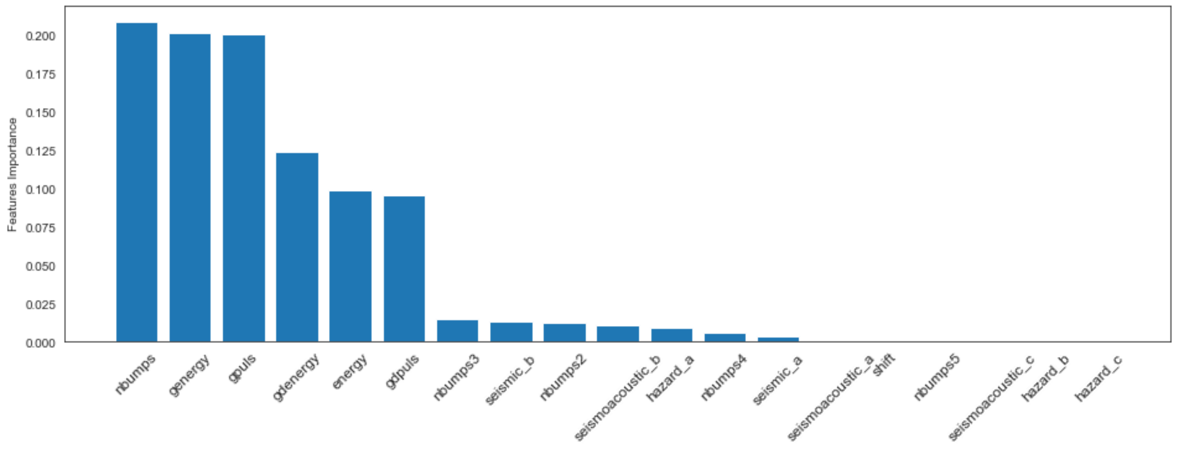


Figure 14: Features importance of the decision tree.

In the preprocessing part, we transform all the categorical features into numerical ones using the methods fit_transform and get_Dummies, and we divide the dataset in the training set (70% of the total) and test set (the remaining 30%). The split is layered on the "class" attribute: each set contains approximately the same percentage of samples of each target class as the complete set.

4.2 Decision tree

Finally, we can start to build the classification tree model. Grid search is used for understanding what parameters are the best for the dataset. The optimal configuration consists of: entropy as the criterion, class weight balanced, maximum depth equal to 12 and minimum sample split as 3. ***Figure 15*** shows a part of the total decision tree created using those parameters.

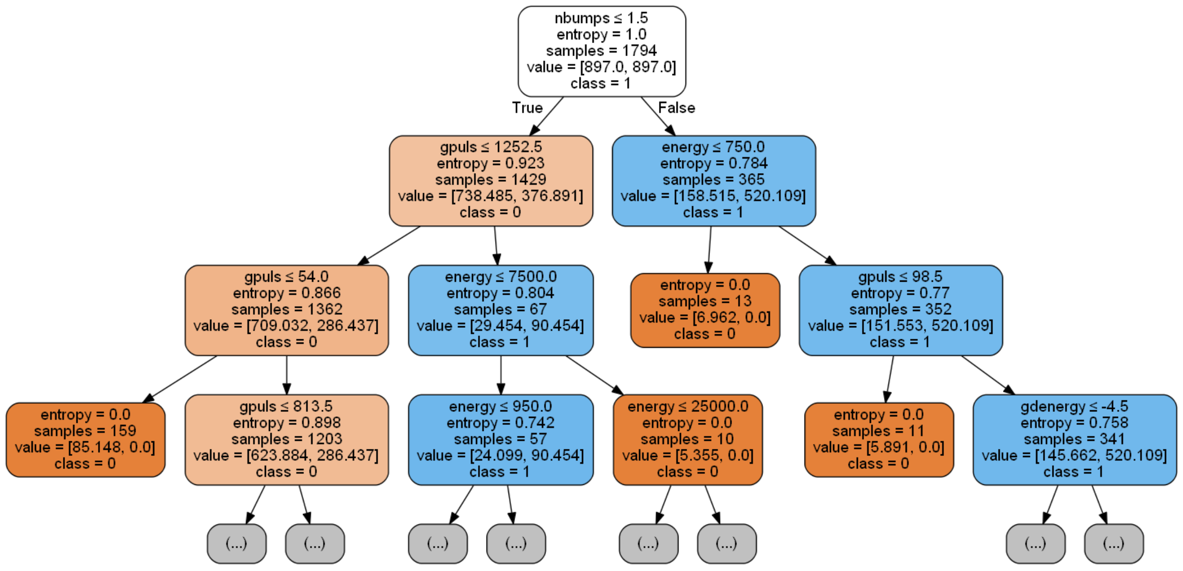
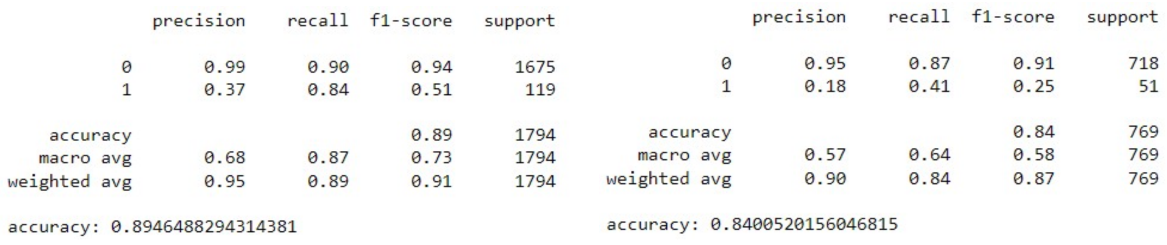


Figure 15: Decision tree.

In ***figure 16*** we show the performances obtained on both training and test set.


```
              precision    recall  f1-score   support

           0       0.99      0.90      0.94      1675
           1       0.37      0.84      0.51       119

    accuracy                           0.89      1794
   macro avg       0.68      0.87      0.73      1794
weighted avg       0.95      0.89      0.91      1794

accuracy: 0.8946488294314381
```

```
              precision    recall  f1-score   support

           0       0.95      0.87      0.91       718
           1       0.18      0.41      0.25        51

    accuracy                           0.84       769
   macro avg       0.57      0.64      0.58       769
weighted avg       0.90      0.84      0.87       769

accuracy: 0.8400520156046815
```

Figure 16: Performances of the model for training and test sets.

The accuracy on the training set is 90%, the precision on *Class 0* is almost equal to 1 so it means that all records belonging to *Class 0* are correctly predicted while, for the minority class there are most false positives because the precision is equal to 37%. The recall has optimal performance for both values of class.

As our dataset is very unbalanced, on the test set, as expected, the performance on the *Class 0* are almost optimal, while on the minority class for the precision we have that are predicted more false positives and for the recall more false negative. The accuracy in this case is high but this measure is misleading; so, the most precise measure we need to take into account is the recall of the minority class, as we want the model to be able to correctly predict the class "hazardous" (*Class 1*) rather than the majority one.

This metric calculates the percentage of the positive records' previsions (the ones with *Class* equal to 1 that the model predicted well) over the total positive records, that, looking at the matrix in ***figure 17***, are the true positive (24) plus the false negatives (27). Our goal is to increase the true positives and decrease the false negatives.

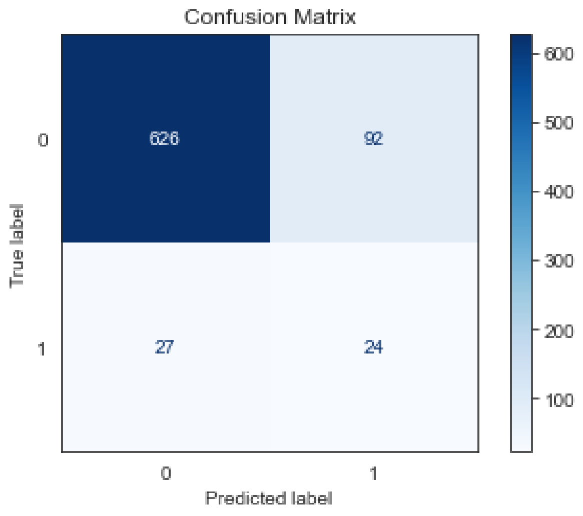


Figure 17: Confusion matrix for the test set.

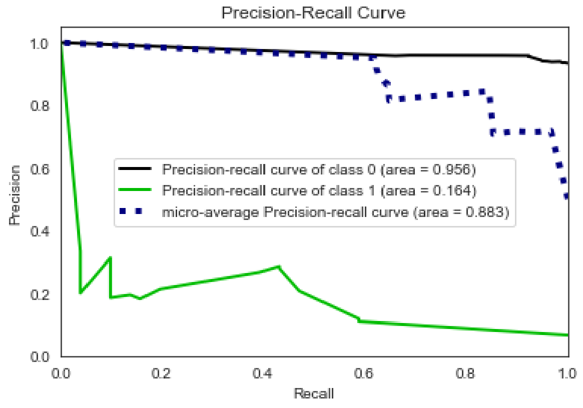


Figure 18: Precision-Recall graph.

We also plot the precision-recall graph (***figure 18***) that shows us the relationship between the two measures for the records with *Class 1* (green line) and the records with *Class 0* (black line). The precision drops drastically as the recall increases when our target variable has detected a dangerous state (*Class* equal to 1), and, for *Class* equal to 0, it starts to drop when the recall hits 60%.

The discrepancy between the precision-recall curve for *Class 0* and *Class 1* is clear if we look at the area under the curves. The performances on *Class 0* are optimal as the value for the area is close to 1 (0.956), whereas the performances on *Class 1* are worse because the value of the area is equal to 0.164.

Lastly, we plot the ROC curve, which illustrates the sensitivity of a model, putting in comparison the true positive records with the false positive ones. In the following graph (***figure 19***) we can see the changing of the rates: while we have the 40% of the total records detected as true positive, we have a smallest error in the false positive rate, but, as we go on with the classification, the rate of the false positive records increase until they both reach 1. We computed the area under this classifier and the result shows an auc score equal to 0.66. We noticed that classifier performs 0.16 better than a Random guessing classifier.

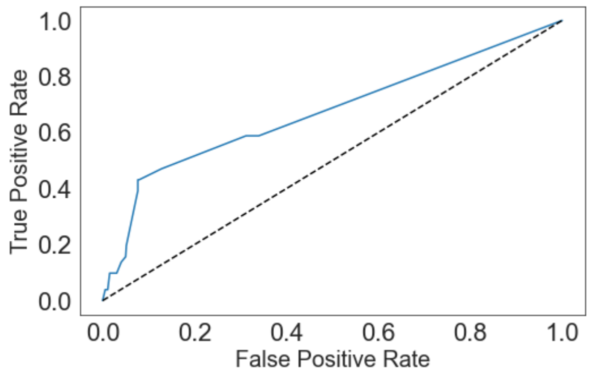
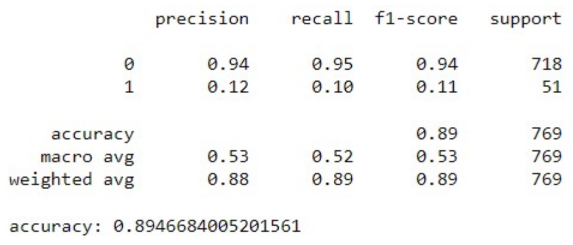


Figure 19: Roc Curve.



```
              precision    recall  f1-score   support

           0       0.94      0.95      0.94       718
           1       0.12      0.10      0.11        51

    accuracy                           0.89       769
   macro avg       0.53      0.52      0.53       769
weighted avg       0.88      0.89      0.89       769

accuracy: 0.8946684005201561
```

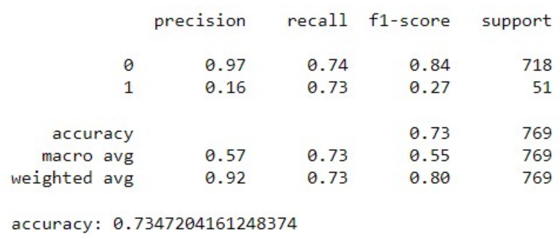
Figure 20: Classification report with class weight= None.

Then, we wanted to see if by removing the parameter ”class_weight” equal to balanced, the performances of the model changed. As expected, the ability in predicting *Class 1* gets worse. In fact compared to the results described in ***figure 20***, there is a decrease in the recall, the accuracy and the f1score on the test set. We can conclude that this model does not perform well in fact the recall measure decreases by 55%, whereas the precision decreases by 3%.

To sum up, if we focus on the minority class, the model has better performance with the parameter “class_weight” set to balanced.

To improve the performance of the model, we also try to use the balancing techniques called **”undersampling”** and **”oversampling”** on the training set. In this case, before starting the analysis, we use the RandomSample method to balance the target variable in the dataset: so, we have 119 cases of *Class 1* and 119 of *Class 1* in the undersampling, and 1675 each class in the oversampling.

Through the GridSearch, we find out that the best values for the model are similar to those used for the first model, except for the minimum sample leaf that increases by one for the undersampling, and decreases by one for the oversampling. In ***figure 21*** and ***figure 22***, a comparison between the performance’s measures of the two methods is shown.



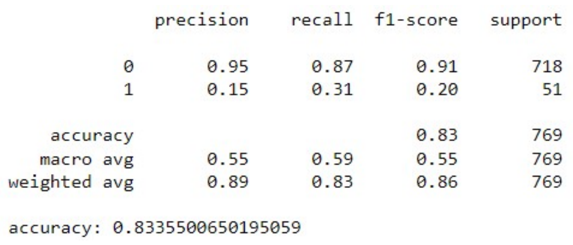
```
              precision    recall  f1-score   support

           0       0.97      0.74      0.84       718
           1       0.16      0.73      0.27        51

    accuracy                           0.73       769
   macro avg       0.57      0.73      0.55       769
weighted avg       0.92      0.73      0.80       769

accuracy: 0.7347204161248374
```

Figure 21: Undersampling technique



```
              precision    recall  f1-score   support

           0       0.95      0.87      0.91       718
           1       0.15      0.31      0.20        51

    accuracy                           0.83       769
   macro avg       0.55      0.59      0.55       769
weighted avg       0.89      0.83      0.86       769

accuracy: 0.8335500650195059
```

Figure 22: Oversampling technique

Certainly, between both models, the undersampling performs better, especially for the prediction of *Class 1* cases. As we compare with the previous model created from the simpler technique (***figure 16***), the error of both these methods is higher for the precision and the f1 score, but there is a vast improvement for the recall measure in the undersampling, showing a 29% boost. This means that this kind of model can predict a high number of dangerous states over all the total predicted dangerous observations, fulfilling our goal to forecast as much as possible the dangerous state inside coal’s mines.

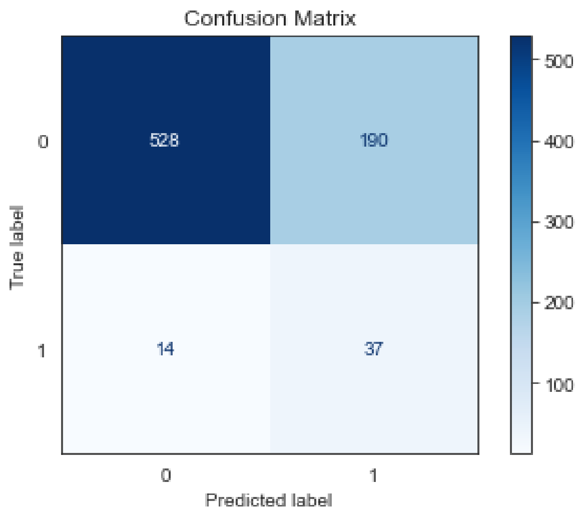


Figure 23: Confusion metrics Undersampling

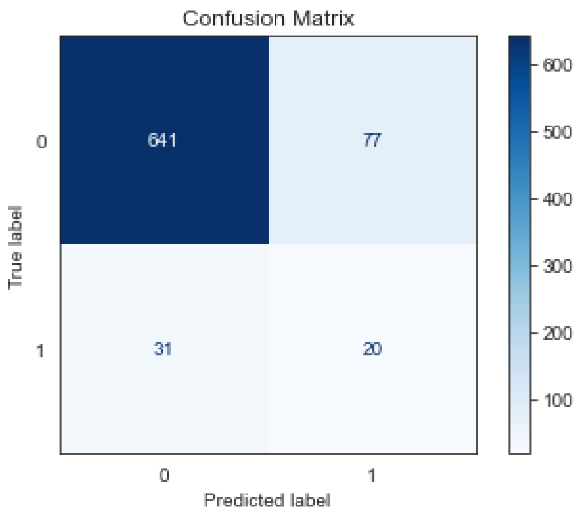


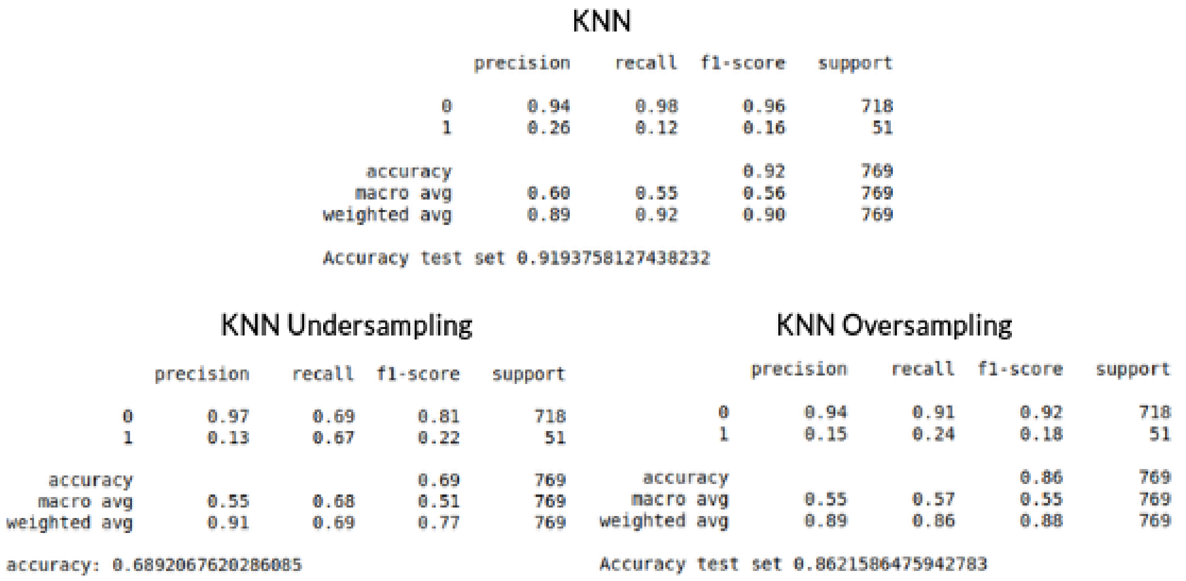
Figure 24: Confusion matrix Oversampling

However, the confusion matrix (***figure 23***) shows that undersampling has generated many False Positives (190), but it has increased True Positives (37 - our goal) as well. Those who use our model want to focus their analysis on predicting almost all the earthquakes, so they must try to monitor the recall considering a high value. Unfortunately, proceeding in this manner leads them to a lot of false earthquake alarms. Even on the training set, we performed the Oversampling technique, which shows in the confusion matrix (***figure 24***) an increase in False Negatives and a decrease in True Positives. We can therefore deduce that this technique does not bring improvements but can actually make the situation worse in terms of predicting dangerous states.

4.3 KNN

Ultimately, we try the KNN method, a classification algorithm that uses the k-nearest neighbour to find the best way to classify a dataset. This method compares the classes of the k closest data to determine the class of the new data (test set). First, we normalize the dataset using the robust scaler that deals better with outliers, and we divide the dataset in training and test, as before. Then, a model is created using the best parameters fitting for our dataset both for the basic KNN and for the undersampling and oversampling of this same method.

For the basic and the oversampling algorithms, we use the euclidean for the metric, 3 for the number of neighbours, and for the weight we use the distance. On the other hand, for the undersampling, Manhattan metric and 9 neighbours are used. ***Figure 25*** shows the three performance's results.



```
              precision    recall  f1-score   support

           0       0.94      0.98      0.96       718
           1       0.26      0.12      0.16        51

    accuracy                           0.92       769
   macro avg       0.60      0.55      0.56       769
weighted avg       0.89      0.92      0.90       769

Accuracy test set 0.9193758127438232
```

```
              precision    recall  f1-score   support

           0       0.97      0.69      0.81       718
           1       0.13      0.67      0.22        51

    accuracy                           0.69       769
   macro avg       0.55      0.68      0.51       769
weighted avg       0.91      0.69      0.77       769

accuracy: 0.6892067620286085
```

```
              precision    recall  f1-score   support

           0       0.94      0.91      0.92       718
           1       0.15      0.24      0.18        51

    accuracy                           0.86       769
   macro avg       0.55      0.57      0.55       769
weighted avg       0.89      0.86      0.88       769

Accuracy test set 0.8621586475942783
```

Figure 25: KNN performances.

As we can see, also in this case the undersampling seems the best way to classify the target variable. While the accuracy is, as usual, a misleading measure, we can clearly see in the undersampling an improvement in the recall measure for *Class 1*, that goes up by 55% and it shows very good results.

The figures below (***figure 26***) show the confusion matrix and the (***figure 27***) the ROC curve for the KNN basic method. The confusion matrix (on the left side) indicates that the model predicts well the true negative records, but just the 12% of the total positive records, a too low percentage for our scope. In fact, the ROC curve (right side), expresses an initial bouncing ratio between true positive and false positive records, while, as the analysis in the the dataset continues, both of the rates reach 1, declaring, again, that the sensibility of this model is not as we would like to be.

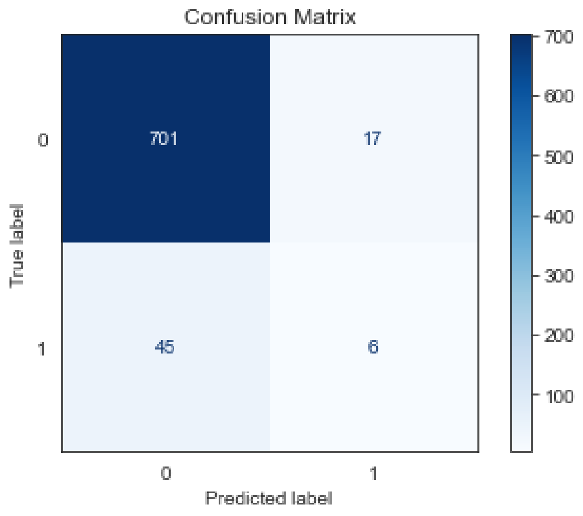


Figure 26: Confusion metrics KNN

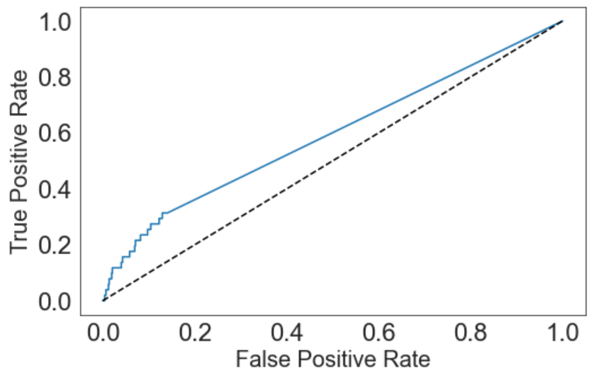


Figure 27: KNN Roc curve

Comparing the decision tree and KNN model, we can conclude that, because of the better performances, the best method we can use for the classification process is the decision tree with undersampling, because it better predicts our target variable.

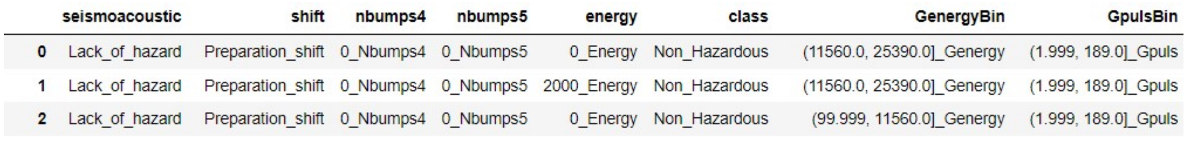
5 Pattern mining

Finally, we start the pattern mining process, in which we are going to find if there are some interesting association rules inside the dataset.

5.1 Preprocessing

In the preprocessing part, we select the features that are going to be used in this process: *Seismoacustic*, *Shift*, *Genergy*, *Gpuls*, *Nbumps4* and *Nbumps5*, *Energy* and *Class*. We have filtered these features removing those characterized by similar measurement (they were mostly redundant in this process).

Later, we need to change the numeric values into categorical ones, using dictionaries and map functions that transform the values into more readable strings. The attributes we change are: all the *Nbumps*, *Energy*, *Genergy* and *Gpuls*. A result's example of the first five records of the dataset is shown in ***figure 28***.



	seismoacoustic	shift	nbumps4	nbumps5	energy	class	GenergyBin	GpulsBin
0	Lack_of_hazard	Preparation_shift	0_Nbumps4	0_Nbumps5	0_Energy	Non_Hazardous	(11560.0, 25390.0]_Genergy	(1.999, 189.0]_Gpuls
1	Lack_of_hazard	Preparation_shift	0_Nbumps4	0_Nbumps5	2000_Energy	Non_Hazardous	(11560.0, 25390.0]_Genergy	(1.999, 189.0]_Gpuls
2	Lack_of_hazard	Preparation_shift	0_Nbumps4	0_Nbumps5	0_Energy	Non_Hazardous	(99.999, 11560.0]_Genergy	(1.999, 189.0]_Gpuls

Figure 28: Newly changed dataset for the chosen eight attributes.

5.2 Frequent itemsets

Then, after transforming the dataset in to a basket (list of lists), we use it in the Apriori algorithm. The first step to do is deciding which values are the best for our dataset in regards to the minimum support, the minimum number of items in the future itemsets (zmin) and the maximum number (zmax).

Starting from the research of the support, we find the frequent itemsets, the maximum frequent itemsets and the closed ones, using the target values "a", "m" and "c" in the apriori module, and we apply eleven random values of support, from values 100 to 10. This gives us a better overview of the changing of the frequent itemsets as the support changes. We can observe that as the support threshold decreases, the number of frequent itemsets increases.

For example we have just 7 frequent itemsets, in which 1 is the maximum frequent and 5 are closed when the threshold is 80; while, when support is equal to 10, we can find 607 frequent itemsets, in which 268 are closed and 30 are maximum. A bar plot that shows how the number of the frequent itemsets changes as the minsup changes is shown ***figure 29***.

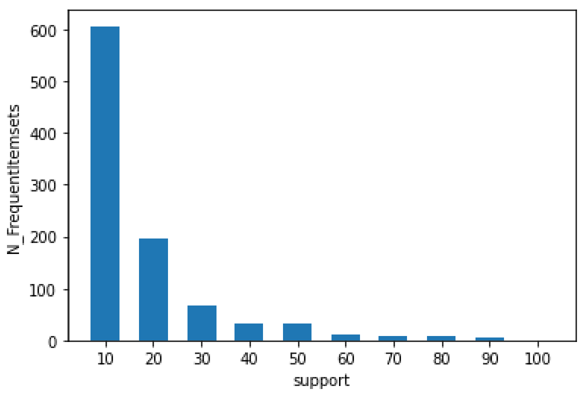


Figure 29: Number of frequent itemsets as the minsup changes.

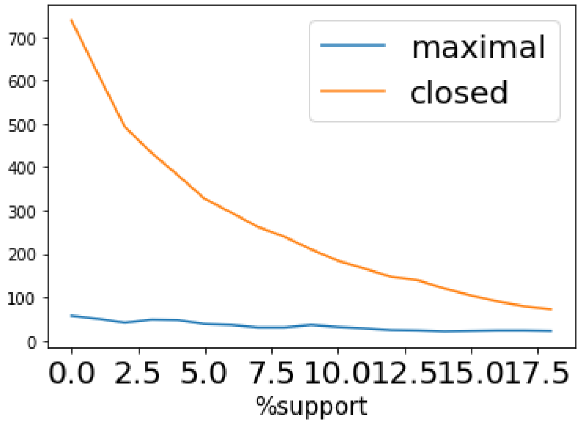


Figure 30: Changes of the number of maximal and closed itemset as the support changes.

As we want to find as many information as possible, we also try to use the whole dataset (considering also the features that are discarded in ***section 5.1***): the results show that the more features we have, the more possible combinations for pattern mining and association rules are found, but it can also cause heterogeneity in the frequency or, on the other hand, values' permutations, making this attempt too complex for our scope.

Then, we look at the frequent itemsets and their support, changing zmin and zmax. Also in this case, we do some attempts using both a very high and a very low zmin. As expected, with this variable set as 7 there aren't any frequent itemsets, because the support is very low (around 13.3 if we set the minsup equal to 13). On the other hand, if we set the zmin as 2, the support level will be much higher than what we would like to. So, we find the best values for zmin and zmax as 3 and 6. With these evaluations, we have made several attempts with different levels of support. In the graph in (***figure 30***), there is a comparison between maximal and closed itemsets when the support changes.

We choose to set a min support equal to 20 as it allow us to extract interesting itemsets. Then, looking at the frequent itemsets (***Table 1***), we notice that there are 195 frequent itemsets, in which 99 are closed and 22 are maximum. The most frequent itemset is {Non_Hazardous, 0_Nbumps4 and 0_Nbumps5}, with a max support equal to 88% and support count as 2261. As we can see in the following table, the other itemsets with large support are characterized by the values {Coal_getting_shift, Lack_of_Hazard and 0_Energy}.

Frequent Itemsets	**Support**	**Frequency**
'Non_Hazardous', '0_Nbumps4', '0_Nbumps5'	88,2%	2261
'Coal_getting_shift', '0_Nbumps4', '0_Nbumps5'	58,6%	1501
'Coal_getting_shift', 'Non_Hazardous', '0_Nbumps5'	58,3%	1493
'Lack_of_hazard', '0_Nbumps4', '0_Nbumps5'	57,6%	1476
'Lack_of_hazard', 'Non_Hazardous', '0_Nbumps5'	57,2%	1467
'0_Energy', '0_Nbumps4', '0_Nbumps5'	57,0%	1462
'0_Energy', 'Non_Hazardous', '0_Nbumps4', '0_Nbumps5'	55,5%	1422
'0_Energy', 'Non_Hazardous', '0_Nbumps4'	55,5%	1422
'0_Energy', 'Non_Hazardous', '0_Nbumps5'	55,5%	1422
'Lack_of_hazard', 'Non_Hazardous', '0_Nbumps4', '0_Nbumps5'	54,1%	1387

Table 1: Table with the frequent itemsets with higher support.

5.3 Association Rules

The next step in the pattern mining process is finding the association rules. We calculate the confidence of a certain association rule using the report in the Apriori module. We have tried various combinations of parameters for this analysis as well, including minimum support equal to 50 and confidence equal to 90. Although the confidence is high, we notice that the lift, which measures how strong the association between two itemsets is, is very low with values less than 1. So we tried decreasing the confidence to 18 and also the support parameter equal to 20, and actually got improvement on the lift.

In the scatter plot below (***figure 31***), the relationship between confidence, support and lift is shown. As we can see, it illustrates that the lift is high when the support is low.

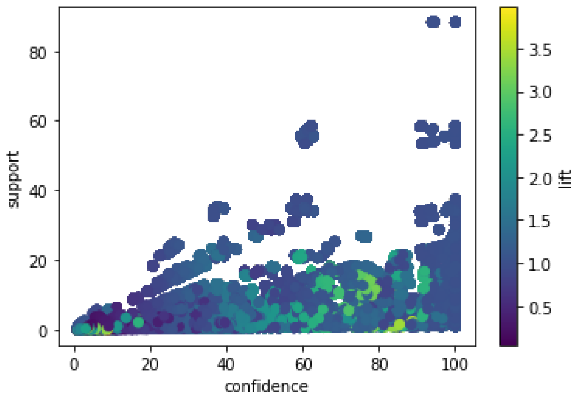


Figure 31: Scatter plot between Support, Confidence and Lift.

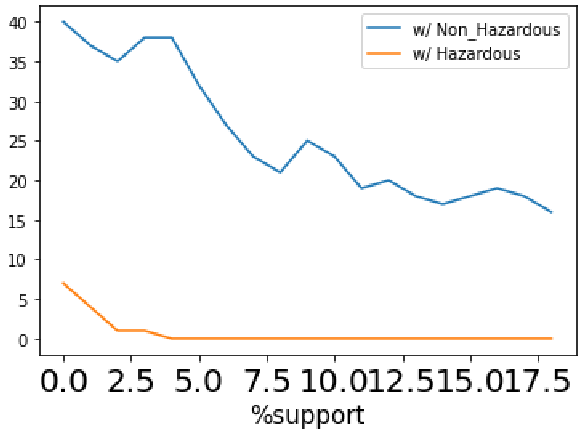


Figure 32: Hazardous and Non Hazardous graph as the support changes.

In the next paragraph, we analyse the itemsets with Hazardous state. As we can see in the previous graphs (and we will see later on), any itemsets with Hazardous state (target variable equal to 1) are found with the support threshold equal to 20, as the dataset is very unbalanced towards a non hazardous state. ***Figure 32*** illustrates this displacement between Hazardous and Non_Hazardous itemsets when the support increments.

The support of the Hazardous frequent itemsets is less than 2.9% and has a confidence lower than 19.1%. So, because we want to find Association Rules having hazardous as rule head, we lower the minsup to 1.5 . We filtered the top five association rules having the highest lift, this is shown in the table below.

Association Rules	**Support**	**Confidence**	**Lift**
'Hazardous' <– ('(52010.0, 1766860.0]_Genergy' , '(661.5, 3915.0]_Gpuls' , 'Lack_of_hazard' , '0_Nbumps5')	1.99	19.17	2.89
'Hazardous' <– ('(52010.0, 1766860.0]_Genergy' , '(661.5, 3915.0]_Gpuls' , 'Lack_of_hazard')	1.98	19.17	2.89
'Hazardous' <– ('(52010.0, 1766860.0]_Genergy' , '(661.5, 3915.0]_Gpuls' , 'Lack_of_hazard')	2.41	18.56	2.79
'Hazardous' <– ('(52010.0, 1766860.0]_Genergy' , '(661.5, 3915.0]_Gpuls' , ' Coal_getting_shift' , '0_Nbumps4' , '0_Nbumps5')	2.38	18.31	2.76
'Hazardous' <– ('(52010.0, 1766860.0]_Genergy' , '(661.5, 3915.0]_Gpuls' , ' Coal_getting_shift' , '0_Nbumps4' , '0_Nbumps5')	2.96	18.18	2.74

5.4 Predicting Class with Association Rules

After finding the most significant Association Rules, our analysis is moving on predicting the target variable *'Class'*. To accomplish our classification goal, we use the Association Rules listed in table above. We choose to classify the records using only rules having "Hazardous" as rule head. Those records not matching the selected rules are automatically labeled as "Non hazardous" (as this class represents the majority class). The results are showing as follows:

	Precision	**Recall**	**F1-score**
Hazardous	18%	45%	25%
Non_Hazardous	96%	85%	90%

In the next table, a confusion matrix is shown:

	Predicted Class: Non Hazardous	**Predicted Class: Hazardous**
Actual class: Non Hazardous	2035	358
Actual class: Hazardous	93	77

As we can see, the classifier does a great job dealing with the majority class ('Non Hazardous'), taking into account 93 False Positives; on the other side, we can see that the classifier built using association rules do not perform well in predicting the minority class ("Hazardous") even though the accuracy is pretty high (82%).

6 Conclusions

In the analysis of the Seismic bumps dataset, we started with data cleaning and visualization processes, so that we could better understand all the features and gets interesting in sights of our dataset. The scope of our analysis was to understand and, then, to predict when and how there could be dangerous situations in the Zabrze-Bielszowice coal mine, Poland.

To do so, we found the correlations between attributes and, then, we carried on with the clusterization process, which helped us to find some correlations and possible clusters between records. We used the K-means, the DBSCAN and the Hierarchical methods, discovering that the best method is the DBSCAN, as the dataset is very sparse.

Then, we proceeded with the Classification using the decision tree and the KNN algorithms, in combination with the Undersampling and Oversampling techniques.

At last, we used Pattern mining to find frequent itemsets and possible association rules, and we created a prediction's model based on the most frequent association rules that could predict the target variable.