Università di Pisa
Dipartamento di Informatica
Data Science and Business Informatics

DATA MINING II

Submitted by
Anna Gaydamaka 589768
Kseniia Parshina 589769
Aliyev Gurban 589206

A.A. 2018/2019

Contents

Introduction

The data is a time series dataset on air quality that is consisted of the following attributes: Date, Time, CO(GT), PT08.S1(CO), NMHC(GT), C6H6(GT), PT08.S2(NMHC), NOx(GT), PT08.S3(NOx), NO2(GT), PT08.S4(NO2), PT08.S5(O3), T, RH, AH. We are aimed at analyzation of the data: sequential patterns, time series, classification, clusterization and outlier detection.

Considering only attribute 'PT08.S1(CO)' and splitting the corresponding time series into daily series we obtained a new database. The data was distributed in the following way:

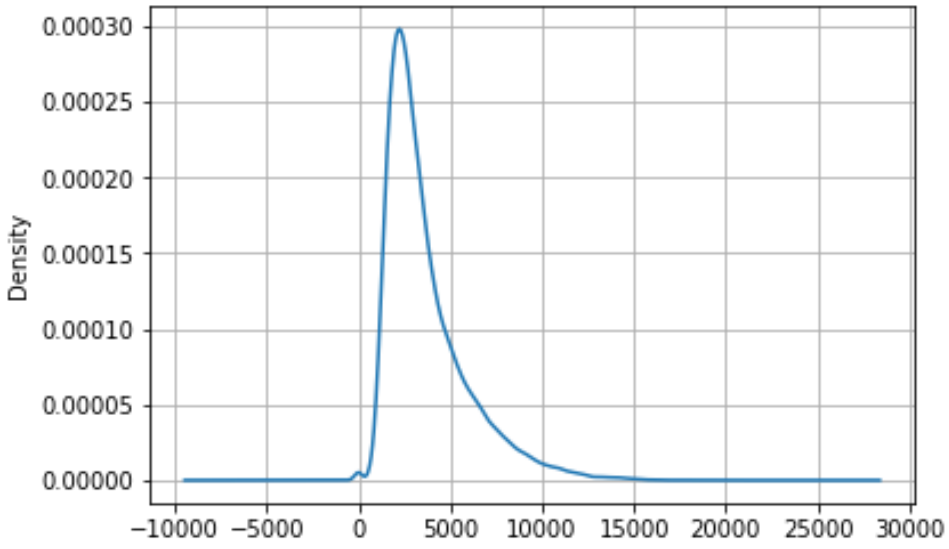

Figure 1.1. Distribution of data

The next step was to handle missing values. We deleted those with too many missing values (value = -200) and replaced the others considering average between previous and following value of the day and of the same hour of previous and following day (Fig.1.2). As the result among all the time series we observed 32 days with missing values and we proved that each day has 24 values.

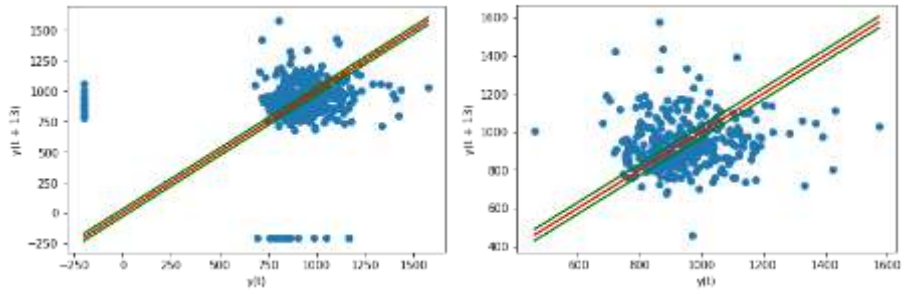

Figure 1.2. Handling missing values.

Some reference lines are presented to highlight strong correlation areas

Clustering

We were considering different approaches in order to group objects from initial dataset in clusters based only on information found in the data that describes the objects and their relationships using K-means, DBScan and Hierarchical algorithms. Application of DBScan with the best Silhuette value that is equal to 0.438 with 4 nodes and radius equals 1710 didn't give any useful results, because data has similar density (Fig. 2.1). That is why we moved to the next approach. Attributes of this dataset are numerical that allows us to use all of them and eliminated none of them. It's important because K-means is limited to numeric data only. Euclidean distance and DTW functions (Fig. 2.7) were used for data points.

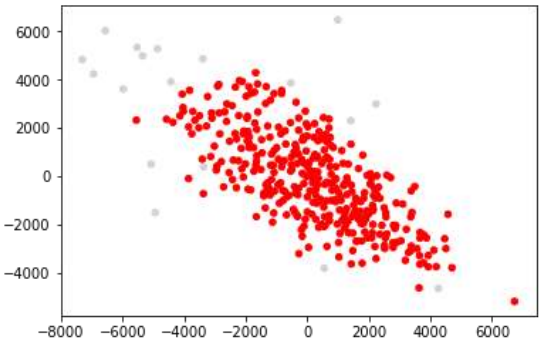

Figure 2.1. DBScan

To choose the best K we calculated two types of error: SSE and Silhouette, and as the result, the dependence between SSE and different amount of K was built as well as the dependence of Silhuette (Fig. 2.2). According to those plots, the best K is 7, which is a distinct knee in the SSE and a distinct peak in the silhouette coefficient. In this case SSE = 39 and Silhouette = 0.269.

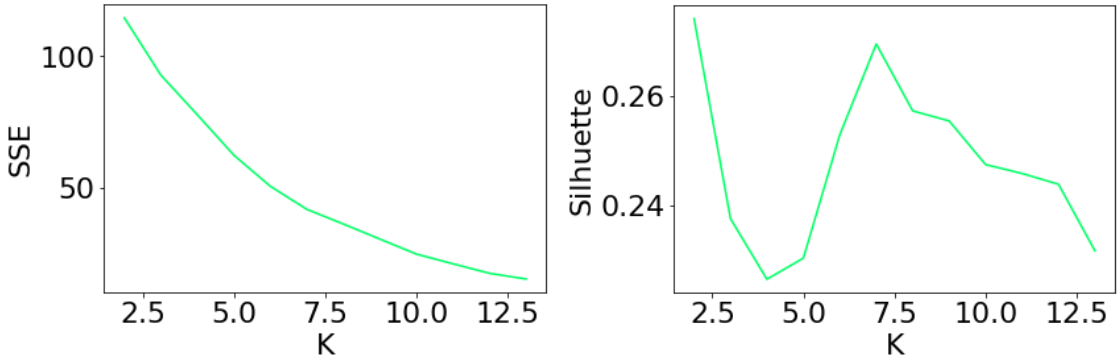

Figure 2.2. The dependence between errors and amount of labels

Using the result of the Figure 2.3, we can underline that mostly clusters have the same behavior, for instance, around 14 the value is lower than was at 13 as well as at 1 is lower than at 0.

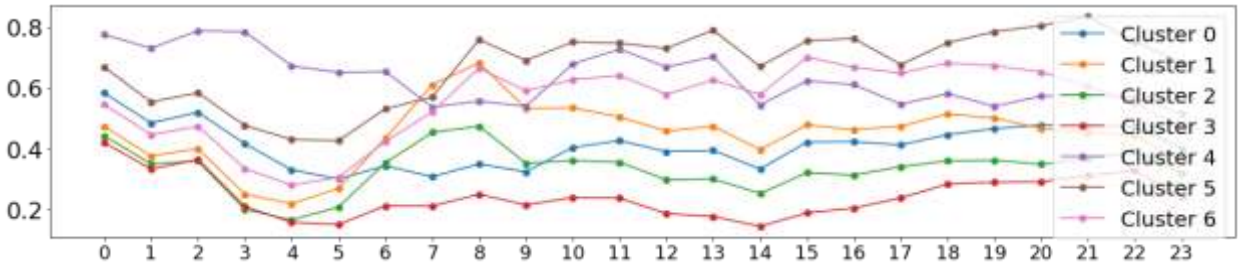

Figure 2.3. Illustration of seven obtained clusters by Euclidean distance.

Normalized values of centroids

To compare the results with DTW metric function, we assigned the same amount of clusters and illustrated obtained values of centroids by DTW (Fig. 2.4). Despite the fact that values in second approach are not normalized and colors of centroids are different, we could see some matches: the distinctive cluster 4 (Fig. 2.3) has his blue representor in Fig. 2.4 as well as cluster 5 has orange one and so on. We could conclude that the result is similar.

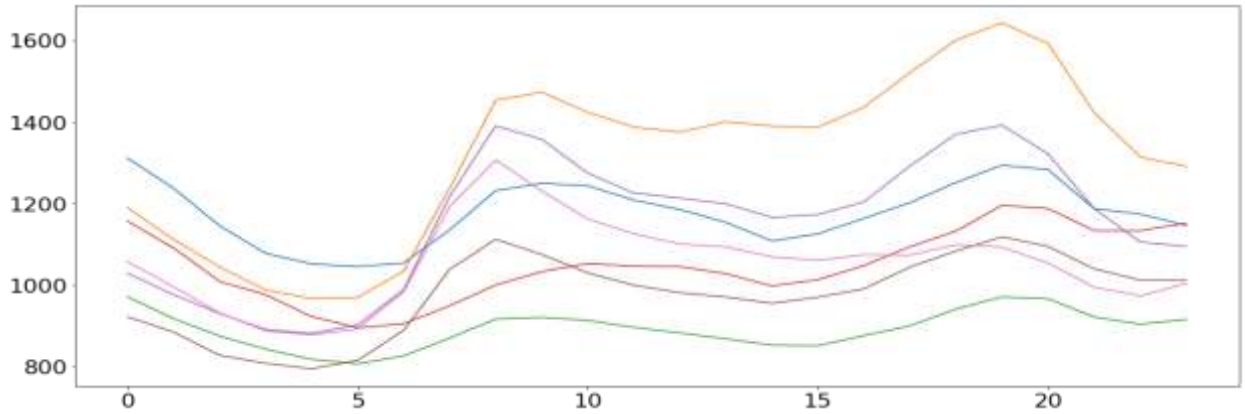

Figure 2.4. Illustration of seven obtained clusters by DTW metric. Values of centroids

At the same time computation of correlation matrix between Euclidean and DTW functions of K-means allows us to compare obtained results mechanically, according to which results of clustering are totally different: the correlation between them is 0.112926. Negative correlation means that one variable increases as the other decreases, and vice versa. As the perfect negative correlation we understand a value of -1, while 0 indicates no correlation, and +1 indicates a perfect positive correlation. To sum up we could underline that K-means algorithm arrange clusters in different way if it deals with the same number of clusters and with discrepant methods as DTW and Euclidean distance functions. Choosing not the best value for parameter K according to error presented allowed us to improved correlation:

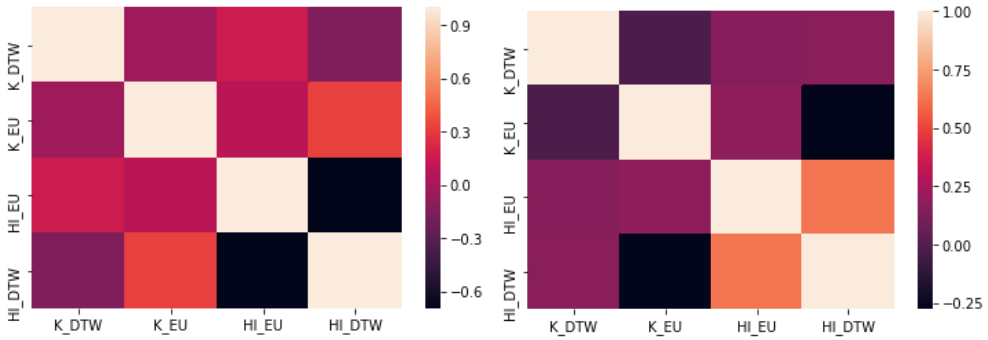

Figure 2.5. Correlation matrixes for K=4 (on the left) and K=5 (on the right) for DTW and Euclidean metrics.

Insufficient results of DBScan and K-means conduced us to construct Hierarchical algorithm with different number of clusters as well. The highest results were obtained with 4 and 5 clusters. As it is clear from the correlation matrixes, the best result belongs to two metrics (DTW and Euclidean distance) of Hierarchical algorithm, the value is 0.683. Below the illustration of Hierarchical algorithms presented with complete method for both metrics. Agglomerative basic approach is used for all of the following dendrograms (Fig. 2.6). These figures show a truncated dendrograms, which only demonstrates the last p=30 out of our 374 merges (initially each parameter creates a cluster itself). Most labels were already merged into clusters before the last 30 merges. Number of merges are on the x axis in parenthesis, while the index of single record is illustrated by number itself.

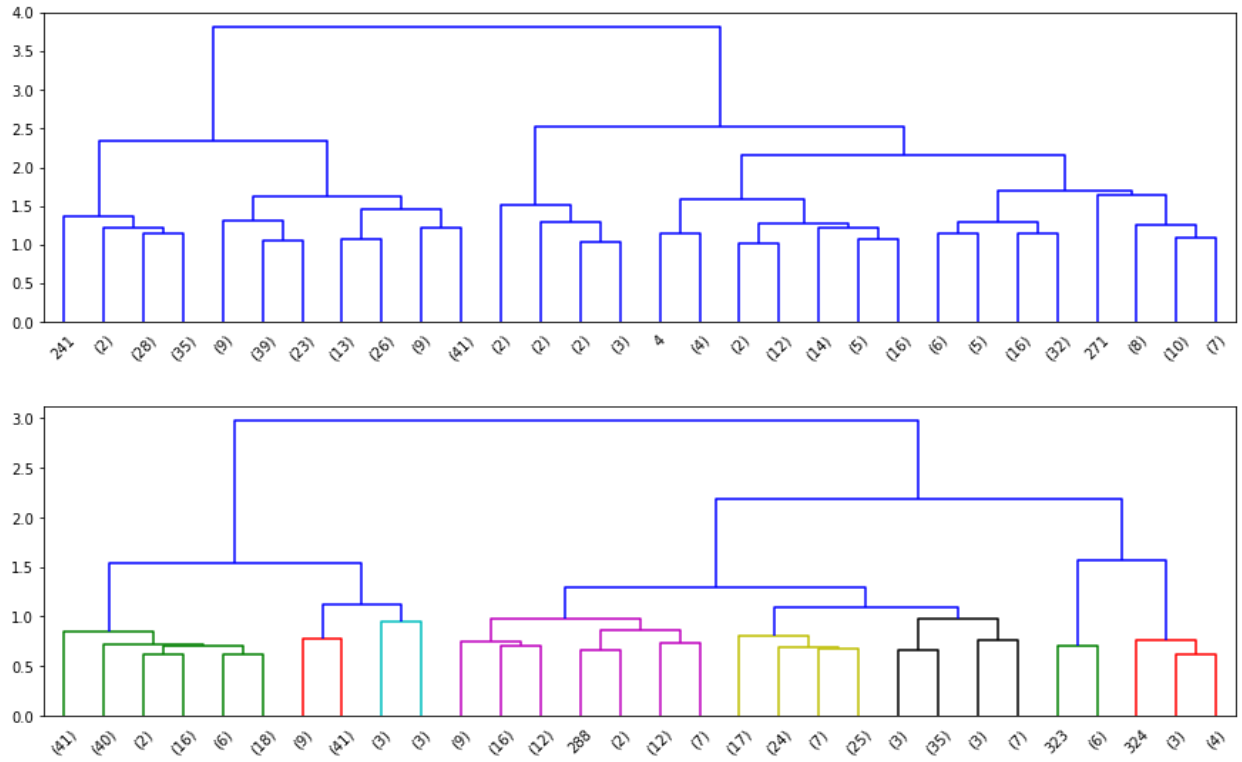

Figure 2.6. Hierarchical clustering by Complete method for Euclidean metric (on the top) and DTW (on the bottom)

All the algorithms are working on Python using precomputed DTW function of distances, where lower cumulative cost is represented by darker colors:

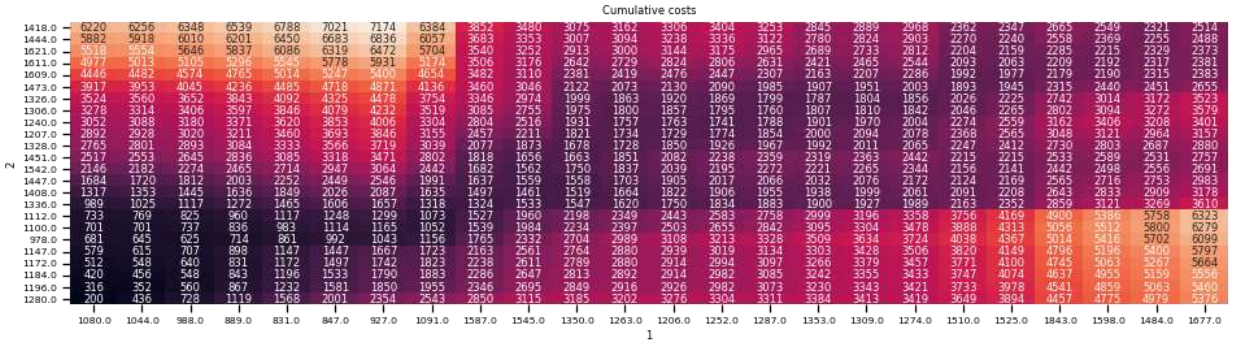

Figure 2.7. DTW matrix for days 2004-03-12 and 2004-03-13

Sequential Patterns

In the second part of the project, we discovered contiguous sequential patterns with minimum length of 4. The data consists of 374 input sequences, while each sequence consists of 24 values representing CO emission in each hour of the day. Before pattern discovery, time series were discretized.

Discretization

Before discovering sequential patterns, we discretized values, which could help to obtain more sequential patterns. If we kept original values, we would have less sequential patterns as each value was almost unique. As a simple and equivalent way to discretization, all values were divided to 100, and values obtained were rounded to integers. The values in the dataset vary from 818 to 1975, meaning the difference between maximum and minimum values is 1200. So, dividing data frame values by 100 should be an equivalent of assigning data values to 12 bins (each bin having width of 100). If we consider that our dataset has 24 columns, 12 is a reasonable number of bins to get sequential patterns. We also tried discretization by dividing values to 10, but there was no pattern with minimum length when support was higher than 90. It was possible to get sequential patterns having length more than 4 only by decreasing support significantly, which is not a beneficial condition. On the other extreme, dividing values to 1000 would leave us only values of ones and twos, which would not be able to give us any valuable information.

We also plotted distribution of values in Figure 3.1. We see that values around 1000 have higher ratio in the dataset. So, it is possible that many frequent patterns found will include value of 10.

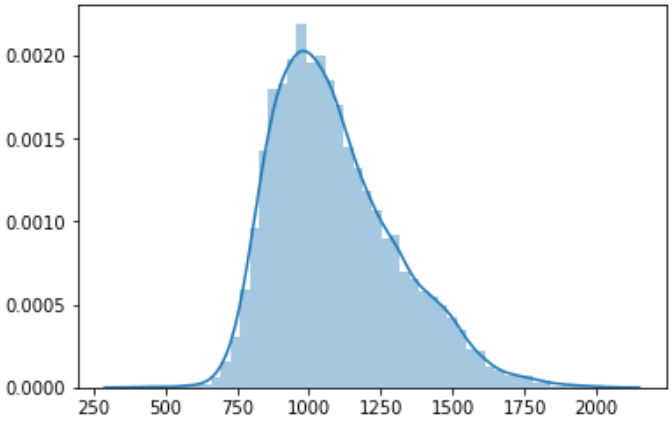

Figure 3.1. Distribution of CO values

Discovery of Contiguous Sequential Patterns

Two algorithms were used to discover contiguous sequential patterns with minimum length of 4: GSP (available both in Python and Java libraries) and SPAM (using SPMF library in Java). The main difference between two algorithms used is that SPAM takes time constraint into account while GSP does not.

GSP Algorithm

At an initial step, using GSP algorithm, the quantity of sequential patterns with minimal length of 4 were plotted for each support level between 80 and 200 (Fig. 3.2). The range of support between 80 and 200 was chosen to decrease calculation time. It is also obvious from the graph that the number of patterns is very high with support < 80, and there is no pattern with minimum length with support > 200. So, there was no need to run the algorithm for support levels lower than 80 and higher than 200.

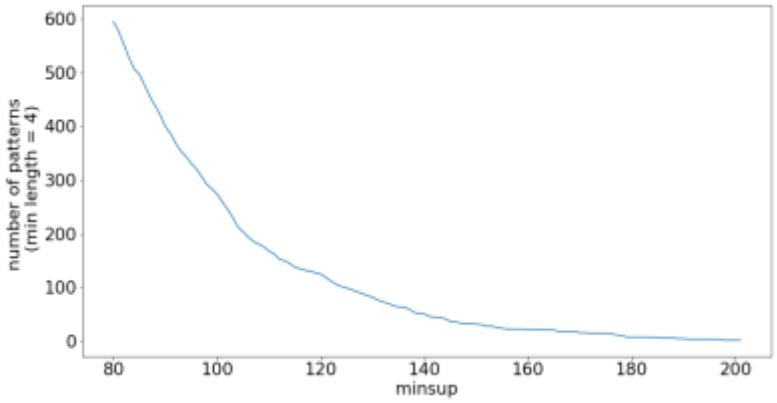

Figure 3.2. Distribution of patterns for different support values

The reason behind building the curve was to obtain a moderate number of patterns with min length setting minimum support level as high as possible. The curve shows the knee is at around 100 (exact value is 105). We also ran GSP algorithm for minimum support levels of 120, 140 and 188 (support > 50%). The number of patterns obtained for given support levels are provided in the table below:

Table 3.1. The number of patterns

minimum support	105	120	140	185
number of patterns with minsup	204	125	51	7

Also, some frequent sequences with highest support levels are provided below. The longest sequences have lengths of 6:

Table 3.2. The number of patterns

The most frequent subsequences of length 4 (support in brackets)	10-10-10-10 (220) 9-9-9-10 (204) 10-9-9-11(186) 11-11-11-11 (186) 11-10-10-10 (183) 9-9-9-11 (181) 12-11-11-11 (156) 8-9-11-11 (131)
The most frequent subsequences of length 5	10-10-10-10-10 (169) 9-9-9-9-10 (154) 11-10-10-10-10 (153) 10-9-10-11-11 (152) 9-9-9-10-11 (150)

	11-10-10-11-11 (130) 12-11-11-11-11 (111) 8-9-10-10-10 (108)
The most frequent subsequences of length 6	9-9-10-10-10-10 (125) 10-10-10-10-10-10 (124) 10-10-10-11-11-11 (124) 9-9-9-10-10-10 (118) 11-10-10-10-11-11 (108)

The main feature seen from examples above is that there are frequent patterns having values of 10 ($950 < x < 1050$) only. Another outcome is that frequent patterns with higher support mainly contain values of 9, 10 and 11 ($850 < x < 1150$). A frequent pattern with the highest variation is 8 – 9 – 11 – 11, which can imply that CO level can jump by 300-400 in 2 hours.

SPAM Algorithm

Most frequent patterns derived from GSP did not give us much information about deviation in CO emission. The most frequent deviation was between 9, 10 and 11, which can be maximum 300. However, CO values in the dataset are between 818 and 1975. We tried to solve the problem using time constraint, which is absent in GSP algorithm. SPAM algorithm was used to find contiguous sequential patterns with time constraints.

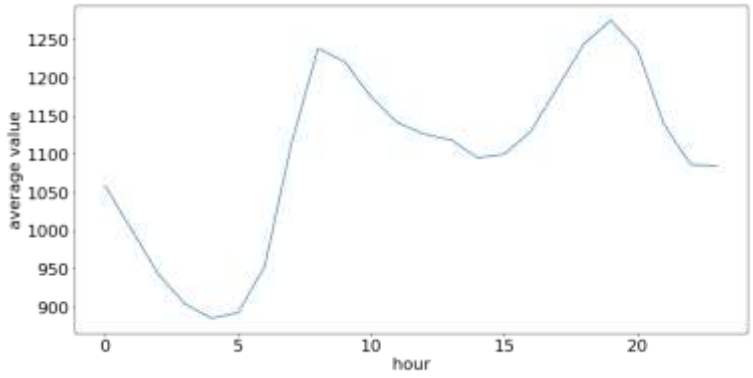

Figure 3.3. Distribution of an average value of CO for each hour

Before using SPAM algorithm, we tried to find out during which time gaps feature values are more related to each other. We plotted average value (amount of CO emission) of each column (hour) in Figure 3.3 to find those time intervals.

The graph shows that intervals of decrease are [0; 4], [8; 14], and [19; 23]; while intervals of increase are [4; 8] and [14, 19]. The longest interval is [8; 14] having 7 points. We assumed two optimistic scenarios to determine maximum gap constraint:

1) a frequent pattern of length 4 is 100% in the biggest interval of length 7;

2) a frequent pattern is in the biggest interval having maximum possible maximum gap constraint.

In the first case, maximum gap should be 0 or 1 to make sure that the pattern will fit the interval. If the maximum gap is 0, a pattern will lay on 4 points of the interval. If the maximum gap is 1, the pattern will lay on 7 points of the interval.

Maximum gap of the second case equals 3, it is possible when three events have a gap of 0 and the fourth event has a gap of 3 to the nearest event.

Quantities of patterns obtained for given support levels are provided in the table below (minimum support level was given 105):

Table 3.3. The number of patterns

Maximum gap	0	1	2	3
number of patterns with maxgap	0	2	8	20

Also, some frequent sequences with highest support levels are provided below. Consider that most of the sequential patterns found in given support levels have length of 4:

Table 3.4. The number of patterns

	maxgap = 1	maxgap = 3		
the most frequent subsequences (support in brackets)	9 - 9 - 9 - 10 (105) 10 - 9 - 9 - 9 (115)	9 - 8 - 8 - 9 (113) 9 - 9 - 8 - 9 (105) 9 - 9 - 9 - 9 (137) 9 - 9 - 9 - 10 (147) 10 - 9 - 8 - 9 (107) 10 - 9 - 9 - 9 (169) 10 - 9 - 9 - 9 - 9 (110)	10 - 9 - 9 - 9 - 10 (121) 10 - 9 - 9 - 10 (149) 10 - 9 - 10 - 11 (105) 10 - 10 - 9 - 9 (128) 10 - 10 - 9 - 10 (122) 10 - 10 - 10 - 10 (143) 10 - 10 - 10 - 11 (125)	10 - 10 - 11 - 11 (112) 11 - 9 - 9 - 9 (109) 11 - 10 - 9 - 9 (128) 11 - 10 - 9 - 10 (118) 11 - 10 - 10 - 10 (125) 11 - 11 - 11 - 11 (107)

The first pattern with length of 5 (10 – 9 – 9 – 9 – 9) probably represents the interval between 0 and 5. The second pattern (10 – 9 – 9 – 9 – 10) can be a version of the first pattern shifted to the right.

Patterns of length 4 mainly represent variation between 10 and 9, 9 and 10, or 10 and 11. These variations are seen in the intervals of [0; 4] and [4, 8]. However, there is no pattern showing increase from 9 to 11, which is seen in the interval of [4; 8]. The possible reason can be seasonality, in some months there can be an increase from 9 to 10 while in others can be from 10 to 11.

In other words, the use of maximum gap constraint did not change the range of values in patterns either. We came up with a further suggestion which can be helpful to solve the problem:

Daily sequences are cut in the wrong point (24). The reason that made us to come up with this idea is that CO level continuously decreases from 7 pm till 4 am (because much fewer people use cars at night). Therefore, the sequence should start at 4 am. Take also into account that CO level sharply increases between 4 and 8 (morning hours) when people drive to work. The same increase between 14 and 19 can be explained by the fact that people drive to restaurants to have a lunch at the afternoon and drive back home in the evening.

Classification

Classification of attributes allows to predict target value. In this part, we are aimed at defining all the objects of initial data set (attribute set) into predefined classes “true” and “false” (class label).

We would like to figure out which time series measures were made in weekend. In order to reach the highest result, the final classification model was constructed by two different classification methods with different parameters and approaches. Let us consider both of them.

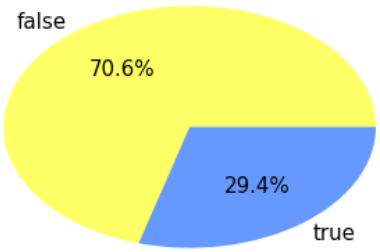

Figure 4.1. A pie chart for the fraction of classes

K Nearest Neighbors algorithm

To apply k-NN method with k = 3 and k=5, we constructed DTW (Dynamic Time Warping) matrix for each pair of time series to calculate the distances between them, for which we applied BallTree classes (Python 3) directly. The nearest neighbor of each point is the point itself, at a distance of zero. Analyzing the initial results gives an opportunity to improve the technique with considering the highest accuracy and F-score. In this approach, each record is assumed to be with unknown class when we predict it, that is why test set is the same as training set (374 records).

Normalization of data set is one of used approaches. It gives slightly worse results that are demonstrated in Table 1 in this chapter. Moreover, we could manage weights and amount of k, nearest neighbors. Considering the fact that the algorithm takes into an account the point itself, metrics ‘distance’ will predict everything perfectly, because closer neighbors of a query point will have a greater influence than neighbors which are further away. While weights ‘uniform’ means that all points in each neighborhood are weighted equally. Increasing number of k will influence on the prediction and will provide worse results, mainly because significance of the node itself is decreasing.

To analyze the results, we constructed confusion matrix (Fig. 4.2). We could say that having k=3 the biggest amount of records is predicted correctly: FN and FP have the same color, according to which amount of each type of error less than 50, and the biggest value is TN, because ‘false’ class in our data set significantly dominates.

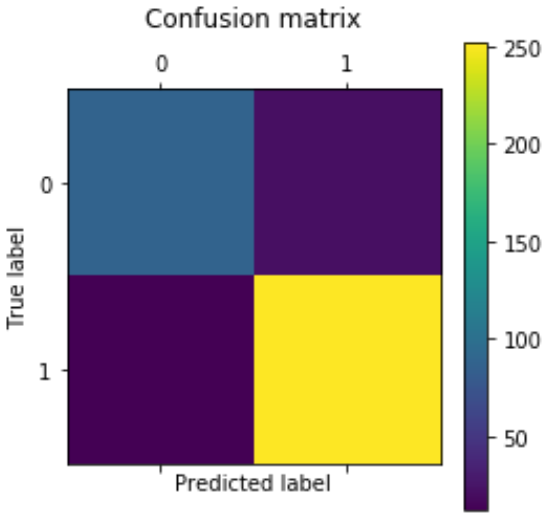

Figure 4.2. Confusion matrix for k=3

Decision Tree algorithm

We divided training set into 2 parts (80 % training and 20% test) in order to teach the model and confirm that the algorithm fits the input data and correctly predicts the class labels. We used Random Search to tune the parameters:

1. criterion='gini': the function to measure the quality of a split;
2. splitter='best': the strategy used to choose the split at each node;
3. max_depth=3: the maximum depth of the tree;
4. min_samples_split=2: the minimum number of samples required to split an internal node;
5. min_samples_leaf=10: the minimum number of samples required to be at a leaf node.

The most important step is an evaluation of the results. There is a confusion matrix (Fig. 4.3) that provides us an opportunity to compare output of different algorithms. As it is seen from the matrix, the majority of values are predicted correct: the true positive square is yellow (more than 50 ‘true’ values among 75 records).

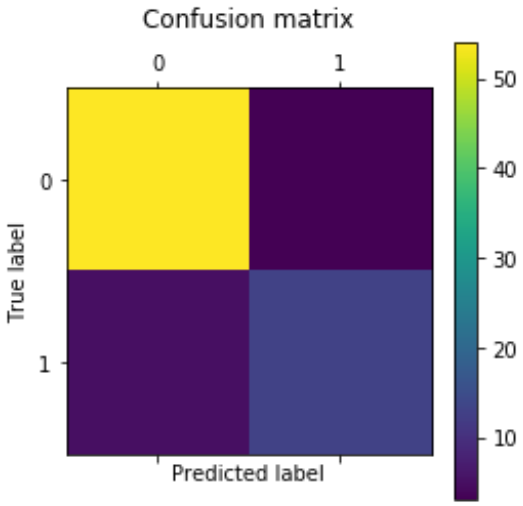

Figure 4.3. Confusion matrix for the test set of Decision Tree model

Let us compare obtained algorithms using Tables 4.1-4.2. As we know, it is not sufficient to consider only accuracy, because we have to take into account false positive and false negative errors too. That is why we also calculated precision, recall, and F1-score, according to which 3-NN algorithm has the best result.

Table 4.1. k-NN model

k-NN k=3	DTW. Weights=’uniform’		DTW. Weights=’distance’
	Data	Normalized Data	
Accuracy	0.909	0.901	1
F-score	0.838	0.823	1
Recall	0.8	0.782	1
Precision	0.88	0.869	1
Number of errors	34	37	0

k=5			
Accuracy	0.879	0.86	
F-score	0.783	0.729	
Recall	0.736	0.636	
Precision	0.835	0.853	
Number of errors	45	52	

Table 4.2. Decision Tree model

Decision tree	Train set	Test set
Accuracy	0.899	0.893
F-score	0.9	0.89
Recall	0.9	0.89
Precision	0.9	0.89

To visualize presented Tables 4.1 – 4.2 we constructed ROC curves. It proves that 3-NN model gives better results than Decision Tree, because square under the first curve is greater (26 is greater than 14).

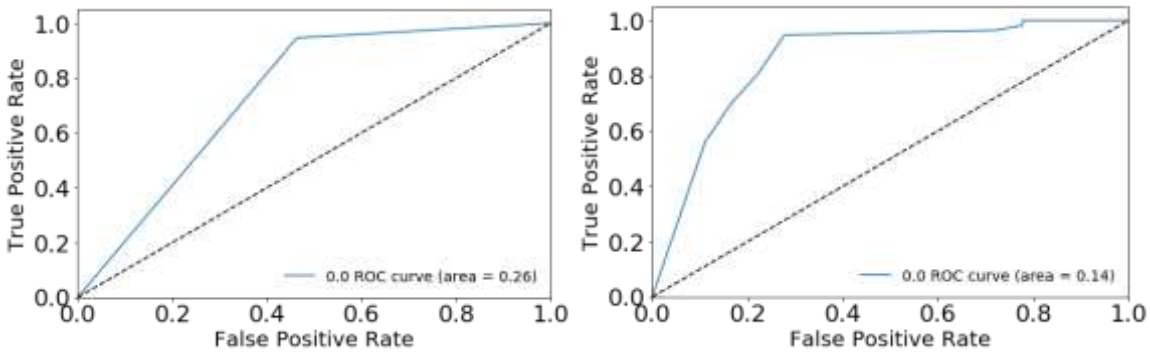

Figure 4.4. Computation of Receiver operating characteristic (ROC)
for 3-NN (on the left) and for Decision Tree (on the right)

Outlier detection

In our study, we are focused on two data quality dimensions, so we are interested in detection of outliers. According to the book “Introduction to Data Mining”, outliers are either (1) data objects that, in some sense, have characteristics that are different from most of the other data objects in the data set, or (2) values of an attribute that are unusual with respect to the typical values for that attribute. Significantly different values could help to figure out unusual behavior and help to understand their reasons. Construction of different families of methods allows us to compare results. We took our initial dataset and, eliminated Data and Time attributes, we constructed correlation matrix (Fig. 5.1).

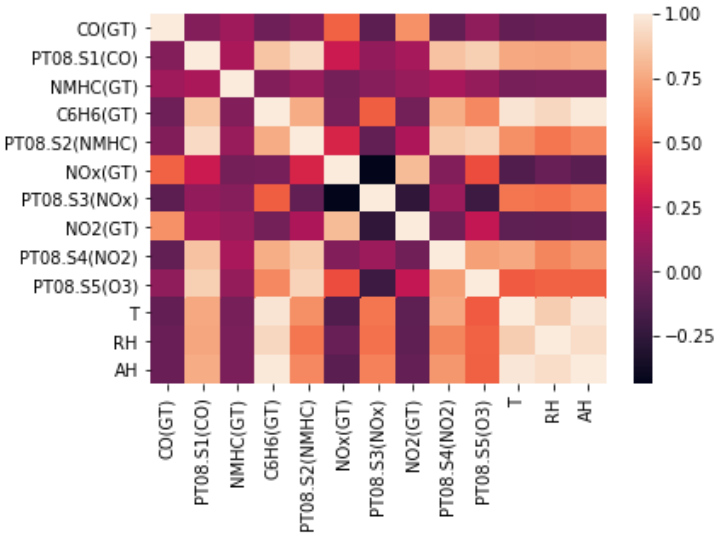

Figure 5.1. Correlation matrix

If the attributes are highly correlated, then they are dependent: an increase in one leads to an increase in the other, so both of them make no sense. That is why we decided to focus on outliers of low-correlated parameters. Let us describe deeply pair-wise comparison of attributes ‘NMHC(GT)’ and ‘NOx(GT)’ with correlation parameter equals -0.004427.

Depth-based outlier detection and statistical tests with the Mahalanobis distance gave us the following results, where outliers are represented by red colors:

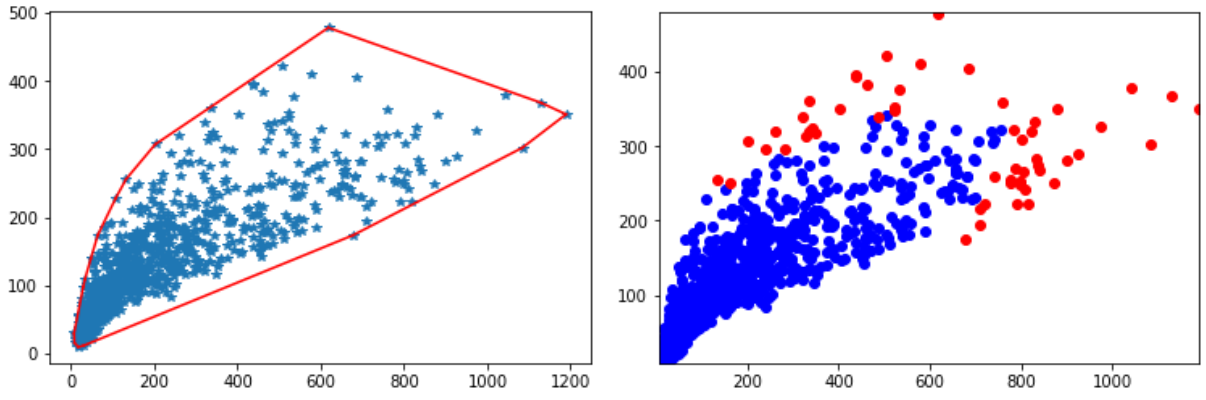

Figure 5.2. Depth-based outlier detection and statistical tests

Elimination of missing data from the dataset gave us as maximum 478 (vertical axis) and 1189 (horizontal axis) number of values for ‘NMHC(GT)’ and ‘NOx(GT)’ attributes respectively. Let us analyze obtained outliers using Table 5.1.

Table 5.1. List of outliers for attributes ‘NMHC(GT)' and 'NOx(GT)'

Approach	List of outliers	Number of outliers
Statistical tests Threshold= 0.26	24 25 36 47 58 59 60 61 93 104 105 106 107 115 129 130 140 150 151 161 162 173 174 247 328 397 408 421 430 431 432 453 454 466 467 512 522 523 650 663 664 665 666 676 688 699 700 708 709 710 824 825 826 837 848	55
Statistical tests Threshold= 0.143	105 106 129 174 397 466 665 837	8
Depth-based	665 837 105 36 58 103 13 57 35 12 33 10 764 485 454 466	16

We could underline that some of the outliers were selected by both algorithms (7 of them). The Depth-based approach is forming the simplical facets of the convex hull. As the result, we figured out 16 outliers that is the lowest possible number of outliers due to the nature of the algorithm. At the same time the approach called statistical tests with the Mahalanobis distance with the best threshold = 0.26, which is a probability of being an outlier, created 55 outliers. So, to identify the top 1% outliers, we set up threshold = 0.143 obtaining 8 outliers (Fig. 5.3) where all of them are presented in depth-based approach with any bigger threshold.

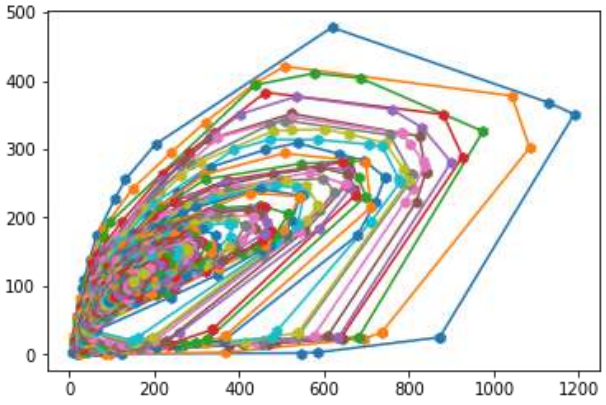

Figure 5.3. Visualization of outliers for depth-based method

The most likely outliers by depth-based approach are located in the external convex hull (blue color on Fig. 5.3), all of which, 49 hulls, are presented on Fig 5.3. We could conclude that values that are significantly different from others in attributes are represented as outliers. All the presented values in the Table 4 have probability of appearance less than the threshold. The mean for the attribute ‘NMHC(GT)’ is around 225 and for ‘NOx(GT)’ is around 141, that is why the density among this values (Fig. 5.4) is significantly higher and more colored. We also noticed that those outliers are presented only in one period of a year: from 15/03/2004 till 29/04/2004 and mostly from 8 am till 10 am and mostly values are noticeably higher than mean.

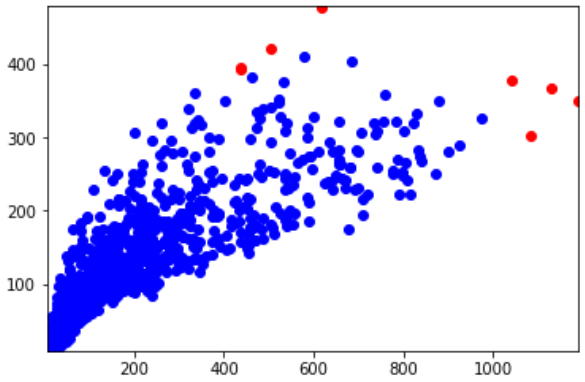

Figure 5.4. Visualization of layers for statistical tests

Conclusion

Visualized analysis of presented air-quality data is presented with sequential patterns, time series, classification, clusterization and outlier detection.

Clusterization of the daily series included all the records and gave us a group of days with the highest values from 8 am till 23 pm, which are significantly different from other days. Also, as the density of the graph is uniformly distributed, DBSCAN algorithm could not identify clusters.

When it comes to the second task, we did not observe any interesting frequent pattern because of two main reasons:

1) Distribution of CO values was centered around a small interval. Therefore, most patterns obtained had values of 9 and 10 only (e.g. 9 – 9 – 9 – 10 or 10 – 9 – 9 – 9).

2) Daily sequences were cut in a wrong point. So, we suggested to change the starting point of each sequence from 0 to 4.

Added attribute 'weekend' became a target function for classification of daily series. According to the fact that amount of work days is higher than weekend, we could assume that an algorithm will try to predict mostly work days. However, both classification algorithms presented with managed parameters gave us quite high results.

In the last part of the project we were aimed at finding interesting outliers applying algorithm of different nature. As the result, we observed 1% of them that appeared in the interval of two months (from 15/03/2004 till 29/04/2004). It means that such values were unusual for our dataset.