Anomaly Detection for Physical Threat Intelligence

Paolo Mignone[1,2][0000−0002−8641−7880], Donato Malerba[1,2][0000−0001−8432−4608], and Michelangelo Ceci[1,2][0000−0002−6690−7583]

[1] Department of Computer Science, Via Orabona, 4, 70125, University of Bari Aldo Moro, Bari, Italy
[2] Big Data Lab, National Interuniversity Consortium for Informatics (CINI), Via Ariosto, 25, 00185, Rome, Italy

Abstract. Anomaly detection is a machine learning task that has been investigated within diverse research areas and application domains. In this paper, we performed anomaly detection for Physical Threat Intelligence. Specifically, we performed anomaly detection for air pollution and public transport traffic analysis for the city of Oslo, Norway. To this aim, the state-of-the-art method SparkGHSOM was considered to learn predictive models for normal (i.e. regular) scenarios of air quality and traffic jams in a distributed fashion. Furthermore, we extended the main algorithm to make the detected anomalies explainable through an instance-based feature ranking approach. The results showed that SparkGHSOM is able to detect anomalies for both the real applications considered in this study, despite the fact it was designed for different tasks.

Keywords: Anomaly detection · Air pollution · Public transport traffic.

1 Introduction

Anomaly detection is a machine learning task that refers to the problem of identifying data that do not conform to patterns observed in historical data. These patterns represent the expected behaviour in normal conditions. Therefore, anomaly detection is usually performed through a data-driven algorithm to construct a model which will be able to detect a specific measurement/object/instance/observation as anomalous with respect to the historical data already seen. Anomaly detection is a very general task that finds applications in many real-domain scenarios such as fraud detection for credit cards, insurance, or health care, intrusion detection for cyber-security, fault detection in safety-critical systems, and military surveillance for enemy activities [6].

In this paper, we consider the Anomaly Detection task for the purposes of Physical Threat Intelligence. Specifically, we propose an algorithm for anomaly detection which works on data continuously collected by geo-located sensors located in urban areas. The data refer to physical information (e.g. temperature, number of vehicles crossing a gate, number of pedestrians in a given area, PM10

level at certain points in the town, etc.). The goal is to identify an anomalous, not expected, behaviour for one or many values simultaneously, considering the specific time, date and spatial coordinates of the considered observation. This would give the opportunity to Security Operators to understand potentially dangerous situations and take the appropriate actions in time.

The task we consider hereby is particularly challenging since data generated by sensors are big in size and have spatial and temporal coordinates that make the data not independent. Indeed, the spatial proximity of sensors introduces spatial autocorrelation in functional annotations and violates the usual assumption that observations are independently and identically distributed (i.i.d.). Although the explicit consideration of these spatial dependencies brings additional complexity to the learning process, it generally leads to increased accuracy of learned models [8]. In addition, data generated by sensors are also affected by temporal autocorrelation, since they *i)* tend to have similar values at the same time on close days; *ii)* have a cyclic and seasonal (over days and years) behavior; *iii)* tend to show the same trend over time.
While stream mining algorithms deal with both i) and ii), they may fail to consider iii), since they tend to better represent the most recently observed concepts, forgetting previously learned ones [1]. On the contrary, time series-based approaches are able to deal with iii), but may fail to consider i) and ii). In fact, they typically require the size of the temporal horizon as an input: Considering a short-term horizon (e.g., daily) excludes a long-term horizon (e.g., seasonal) and vice versa. On the contrary, in the approach presented in this paper, we propose a time-series approach that exploits both spatial and temporal features, in order to take into account all the aspects mentioned before. In particular, the method addresses the problem of identifying complex spatio-temporal patterns in sensor data by means of Self-Organizing Maps (SOMs).

A SOM [4] is a neural-network-based clustering algorithm that operates by mapping high-dimensional input data into a 2-dimensional space implemented by a grid of neurons called *feature map*. In this paper, we consider GHSOMs, (Growing Hierarchical SOMs) that are particularly suitable for time series data and better capture spatio-temporal information thanks to the hierarchical organization of the SOMs that better adapt to complex data distribution. Specifically, we consider the distributed extension Spark-GHSOM [6], that exploits the Spark architecture to process massive data, like those coming from sensors. Since GHSOMs are designed for clustering and not for anomaly detection tasks, we extend the learning algorithm Spark-GHSOM in order to learn GHSOMs for anomaly detection, in an unsupervised fashion.

2 Spark-GHSOM

Spark-GHSOM [6] was introduced to overcome two limitations of the classical GHSOMs. Indeed, a GHSOM *i)* requires multiple iterations over the input dataset making it intractable on large datasets; *ii)* it is designed to handle datasets with numeric attributes only, representing an important limitation as

most modern real-world datasets are characterized by mixed attributes (numerical and categorical). Therefore, Spark-GHSOM exploits the Spark platform to process massive datasets in a distributed fashion. Furthermore, it exploits the distance hierarchy [2] to modify the optimization function of GHSOM so that it can (also) coherently handle mixed-attribute datasets. Spark-GHSOM showed high accuracy, scalability, and descriptive power on different datasets.

The first step in the GHSOM algorithm is to compute the inherent dissimilarity in the input data with different types of attributes. Classical GHSOMs exploit the *mean quantization error*. However, this error is suitable for numerical attributes only. While there is no standard definition of mean for categorical attributes, SparkGHSOM replaces the mean quantization error by considering instead the variance in order to assess the quality of map and neurons. For categorical attributes, *unlikability* is a good measure to estimate how often the values differ from one another [3]. Formally, let $\mathbb{D}$ the dataset under analysis, the unlikability for a categorical attribute A of $\mathbb{D}$ is defined as:

$$\mathbb{U}(A) = \sum_{i \in domain(A)} p_i(1 - p_i) \tag{1}$$

where $p_i = \frac{frequency(A_i, \mathbb{D})}{|\mathbb{D}|}$, A_i is the i-th value of the attribute A and $frequency(A_i, \mathbb{D})$ is the absolute frequency of the value A_i for the attribute A in $\mathbb{D}$. Therefore, SparkGHSOM computes the overall variance of the dataset as follows:

$$\sigma = \sum_{A \in featureset} \mathbb{1}^{num(A)} \sigma(A) + \mathbb{1}^{cat(A)} \frac{\mathbb{U}(A)}{2} \tag{2}$$

where $\mathbb{1}^{num(A)}$ (resp. $\mathbb{1}^{cat(A)}$) is 1 when the attribute A is numerical (resp. categorical), 0 otherwise. $\sigma(A)$ represents the classical variance for the attribute A when it is numerical.

The distance hierarchy [2] is considered to compute the similarities among the categorical values. To compute the distance among categorical values, a distance hierarchy for each categorical attribute must be provided in advance. Similar values according to the concept hierarchy are placed under a common parent which represents an abstract concept. The GHSOM training process takes into account mixed attributes and consists in finding the *winner* (closest) neuron of the SOM w.r.t. the single input instance according to the distance hierarchy.

In the first step, the winner neuron is identified for the input instance according to the distance hierarchy. Therefore, the neuron's weight vector is modified by a certain amount to match the instance vector. In the hierarchy tree of the concepts, where the leaves represent the actual values of the instances and the non-leaf nodes represent the neurons, this process pulls the neuron point towards its leaf in order to "specialize" what the neuron describes.

In the second step, the closest winner neuron and its surrounding neighbor neurons of the SOM are adapted moving them towards the input instance. This training process requires a defined number training epochs over the input dataset. The training is governed by the Mean Quantization Error (MQE) of a

neuron, that is the total deviation of the neuron from its mapped input instances. The MQE for a SOM layer is computed as the average MQE of all the neurons representing instances. A higher value of the MQE means that the layer does not represent the input data well and requires more neurons to better represent the input domain. Moreover, when a single neuron is still not representing the surrounding instances, then the neuron is expanded as a SOM hierarchically (see figure 1).

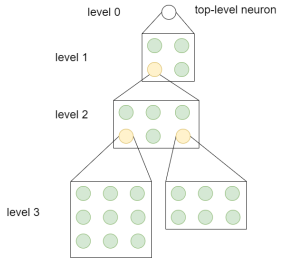

Fig. 1. A Growing Hierarchical Self Organizing Map.

3 Spark-GHSOM for Anomaly Detection

The training process of the Spark-GHSOM follows the classical process of the GHSOM training, except for the use of a different function for the calculation of the distance between the input vector and the neurons of the feature map, since the Euclidean distance is not computable on categorical attributes. For this reason, the hierarchical distance was chosen [6][2].

The hierarchy obtained can thus be used to solve an anomaly detection task. In particular, when a new input vector is supplied to the hierarchy, the algorithm looks for the SOM that succeeds in better approximating the input data (that is, the SOM with the shortest distance with respect to the input vector). Once found, it is used to carry out the prediction for the new input data, based on the distance between the input vector and the closest neuron (the *winner neuron*) in the map.

More formally, let x_i be the new example to be considered, and let $e(x_i) = \arg\min_e dist(x_i, e)$ the closest neuron to x_i according to the distance measure described before, the example is considered an anomaly if the following inequality holds:

$$dist(x_i, e(x_i)) > (d_{avg} + tf * \sigma) \tag{3}$$

In the formula, d_{avg} is the average distance among the training instances and the neurons of the model after the training, σ the standard deviation of such distances, and tf the user-defined threshold.

As data distributions tend to change over time, it may be necessary to update the knowledge of the anomaly detector using more recent data. For this reason, Spark-GHSOM for anomaly detection provides the possibility to update the weights vectors of the neurons while keeping the generated hierarchy unchanged. This process can be particularly useful if end users do not have enough time or data availability to train a new anomaly detector from scratch. Consequently, having a pre-trained model already available, it is possible to provide the model with a micro-batch of data, in order to update the knowledge extracted by the model and adapt it to the user's needs. This aspect is particularly useful in our case, where data generated by the sensors can be relatively few.

The anomaly detector could produce different types of output depending on the level of detail. The simplest approach provides feedback for the current data in the form of a Boolean response. This kind of output could support raising an alert if the response is equal to "anomaly".

This approach presents the advantage that is simple to handle and transmits the prediction as a binary variable (e.g., anomaly/normal, 0/1, true/false). Its drawback is that it makes it difficult for the end-user to interpret the raised alert/anomaly. Therefore, a more informative approach could be considered by combining the previous one with a ranking of the variables (feature ranking) according to their importance, indicating the contribution to catching the variable's anomaly.

Feature ranking is a ranking of the entire set of features composing the data collection, ordered with respect to the feature importance. Feature importance is a numerical value between 0 and 1, which expresses how anomalous the value expressed by the feature is with respect to the data collection, such that the sum of all the features importance in the feature ranking is equal to 1. The importance score is determined starting from a distance function between the current data under analysis and the winner neuron. Specifically, the ranking is proportional to the contribution provided by the single feature in the Euclidean distance between x_i and $e(x_i)$. More formally, the ranking function for the instance x_i, $r_f(x_i)$, is computed as follows:

$$r_f(x_i) = \frac{(x_i[l] - e(x_i)[l])^2}{\sum_{l'}(x_i[l'] - e(x_i)[l'])^2} \tag{4}$$

where l represents the feature index.

This approach helps to identify the feature(s) that most contributed to the anomaly and, therefore, the "reason" for the anomaly.

Experiments

The experiments were conducted for the city of Oslo (Norway) by considering two real domains for the following analyses: air pollution and public transport traffic.

Air pollution analysis

The proposed method was tested using data coming from air quality monitoring sensors to identify pollutant concentrations deemed abnormal.

At each location, different pollutants are monitored by the sensors:

- Hjortnes: NO, NO2, NOx, PM10 and PM2.5
- Loallmenningen: NO, NO2, NOx, PM1, PM10 and PM2.5
- Spikersuppa: PM10 and PM2.5

The information on the concentration of pollutants comes with both a timestamp and the geo-coordinates (latitude and longitude), so that the time series can be reconstructed. Data, which is publicly available, can be downloaded through a REST API [3].

The period considered for training was from January 2021 to September 2021, with an hourly sampling rate, totalling 18.286 data points from the chosen locations. The period considered for testing is October 2021, totalling 720 acquisitions from the chosen locations. The best value for the parameter tf has been selected according to an internal cross-validation on the training instances in the interval [0, 15].

Figure 2 shows the concentrations per hour of NO, NOx, and NO2 pollutants during the identified test period, i.e., October 2021, from the station of Hjortnes. The choice fell on these pollutants because they are present within the top-3 of the feature ranking, for those time instants considered anomalous by the algorithm, indicated with black arrows in the graph.

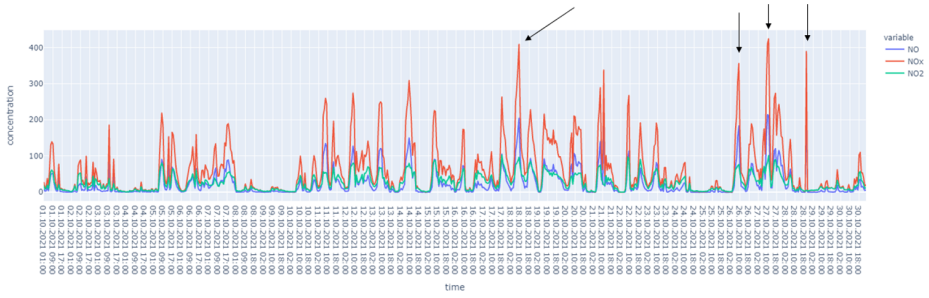

Fig. 2. Concentrations per hour of NO, NOx and NO2 pollutants during October 2021, from Hjortnes station.

It is worth to note that we did not find an abnormal situation during October 21 at 10 a.m., indicated with a green arrow in Figure 3, when very high concentrations of PM1 were recorded, even though at this time point the pollutant PM1 is correctly present in the first position of the feature ranking.

[3] https://api.nilu.no/

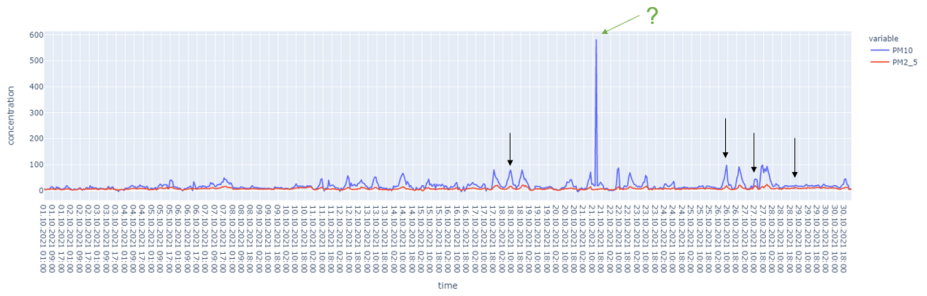

Fig. 3. Concentrations per hour of PM10 and PM2.5 pollutants during October 2021, from Hjortnes station.

The motivation is because several pollutants are being observed together and the sudden increase of concentrations of one of them is sometimes not sufficient to classify the time instant as a potential abnormal situation.

Figure 4 shows the concentration per hour of PM1 pollutant during the test period, from Loallmenningen. For this place, PM1 is the most decisive pollutant for the detection of abnormal situations that occurred during October 2021.

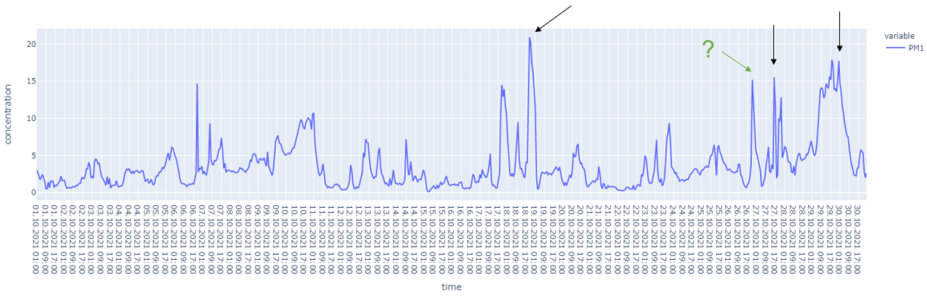

Fig. 4. Concentrations per hour of PM1 pollutant during October 2021, from Loallmenningen station.

As in the previous graphs, the black arrows indicate the time instants in which we detected abnormal concentrations of the pollutants considered. As expected, the algorithm was able to correctly detect high concentrations of the PM1 pollutant.

However, on October 26 at 9 p.m., indicated by the green arrow, the concentrations of PM1 were very similar to those of October 27 at 4 p.m., but only in the latter case, an anomalous situation was found by the algorithm. A more detailed graph is shown in Figure 5.

The reason is due to a sudden increase in concentrations of the remaining pollutants, which occurred on October 27 at 4 p.m. This situation, as shown

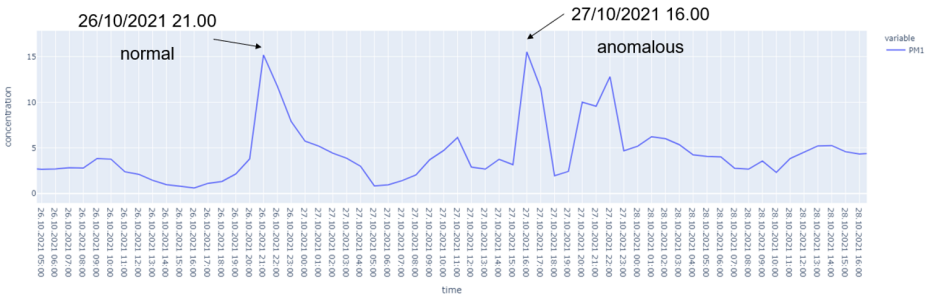

Fig. 5. A zoom in with respect to the time interval for PM1 pollutant during October 2021, from Loallmenningen station.

in Figure 6, allowed the algorithm to identify an anomalous situation at this timestamp.

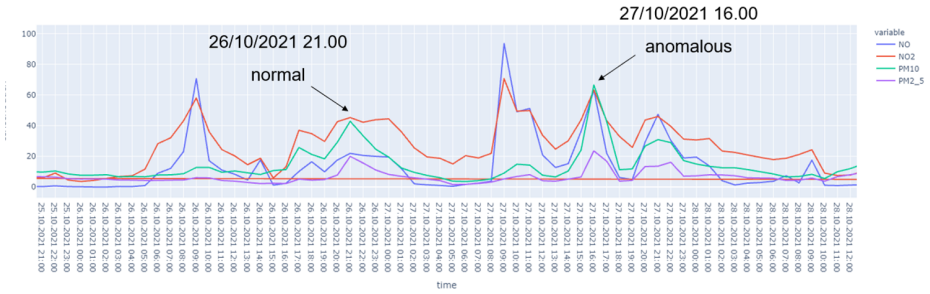

Fig. 6. Concentrations per hour of NO, NO2, PM10 and PM2.5 pollutants during October 2021, from Loallmenningen station.

Figure 7 shows the concentrations per hour of PM10 and PM2.5 pollutants during the test period, from the area of Spikersuppa. The pollutants shown in the graph are the only ones the station can monitor. As expected, the algorithm did not identify any situations deemed abnormal for this place, as the concentrations of October are quite regular.

Public transport traffic

This data consists of one week of data regarding Oslo's public transport. The instances represent GPS-tracked busses with latitude and longitude. Each instance is timestamped according to the standard ISO 8601 with a resolution in seconds. The Service Interface for Real time Information - Vehicle Monitoring

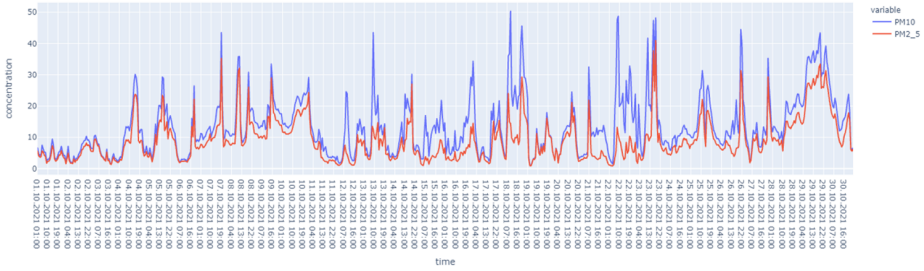

Fig. 7. Concentrations per hour of PM10 and PM2.5 pollutants during October 2021, from Spikersuppa station.

(SIRI-VM) is used to model vehicle-movements and their progress compared to a planned timetable [4].

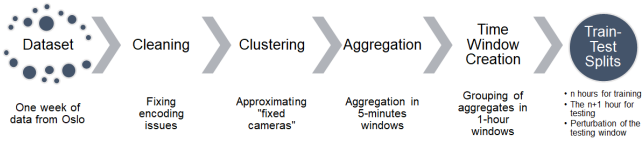

Fig. 8. The data processing pipeline

For this dataset, the processing pipeline illustrated in Figure 8 was executed. Therefore, starting from the week of data from Oslo traffic transport, we performed data cleaning in order to fix some encoding issues. We also aggregated data by 5-minutes interval periods and by spatial areas according to some preliminary clustering. This step was crucial since the data provided refer to movable points in the map making the aggregation operations unfeasible. Clustering on the spatial location was performed by exploiting K-Means algorithm [5]. The variables of the considered data were extended by considering the cluster identifier (cluster ID) and the cluster's centroid latitude and longitude to the data. Since K-Means algorithm needs the number of clusters to identify, we performed the well-known silhouette cluster analysis [7] with the aim to identify the number of areas for monitoring the traffic. According to silhouette analysis, we considered 100 different regions for traffic monitoring (see Figure 9).

The instances are therefore grouped by two levels: first the time, then the cluster id previously identified. Various new features are computed as part of the aggregation (e.g., the average "delay" of the buses in seconds) for each identified clustered monitoring area. Multiple train and test sets were created as illustrated in figure 10. The n-th evaluation step uses n hours for training, and the $(n +$

[4] https://api.entur.io/realtime/v1/rest/vm?datasetId=RUT

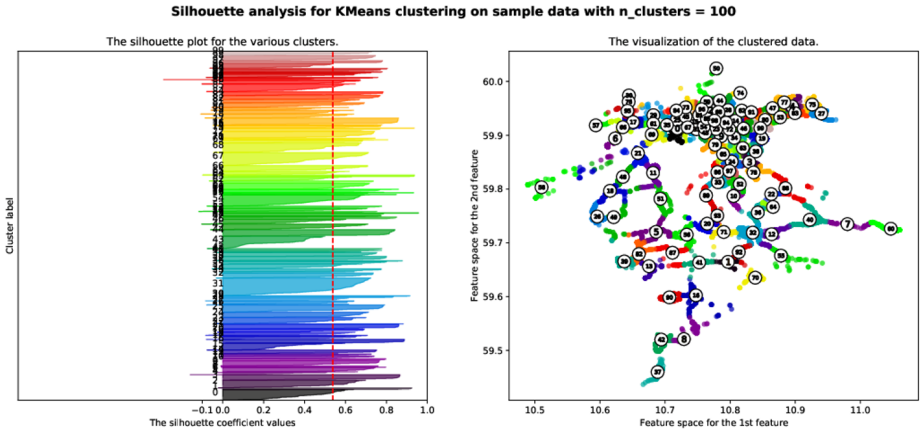

Fig. 9. The Oslo street map and the best locations for monitoring traffic according to the clustering step.

1)-th hour for testing. The 10% of the available test windows are perturbed randomly selecting 3 columns for each instance and randomly assigning a new value for each selected feature. These test windows are considered as anomalous. The remaining 90% of the available test windows are used without perturbation and considered non-anomalous for the evaluation. The aim of this setting is to perform an evaluation based on landmark windows. The best value for the parameter tf has been selected according to an internal cross-validation on the training instances in the interval [0, 15].

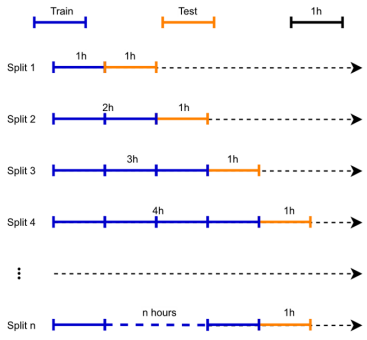

Fig. 10. Training and testing sets.

In Figure 11 hour-by-hour histograms are reported for the first day. Stacked green bars indicate the correct predictions, while the red ones the wrong predictions. The red text in the date indicates that the window is perturbed (anomaly). The top label contains the total number of instances in the test set. During nor-

mal windows, the anomaly detector results are effective since false positives are generally avoided. Most of the normal scenarios that occurred during different time slots (in the morning, afternoon, evening, and night) were recognized as normal situations: 99.7% accuracy (we have only 5 false positives at the beginning, when the model is still unstable). From the figure, we can also see that the system identifies many false negatives at the beginning [01:40-03:40]. This is expected since the model is still unstable to detect anomalies. Moreover, the lack of data, due to the lack of public transport late in the night (or early in the morning, only 48 instances), further complicated the problem. During the day, after 22 hours of training, the anomaly detector appears to be much more stable and capable to predict most of the anomalies occurred during the two-hours anomalous time slot [16:40-18:40] in the afternoon. After 26 hours of training, the anomaly detector becomes further stable and capable to predict most of the anomalies occurred during the anomalous time slot [20:40-21:40] in the evening. After 28 hours of training, the anomaly detector becomes further more stable and capable to predict most of the anomalies occurred during the anomalous time slot [05:40-06:40] in the evening/early morning.

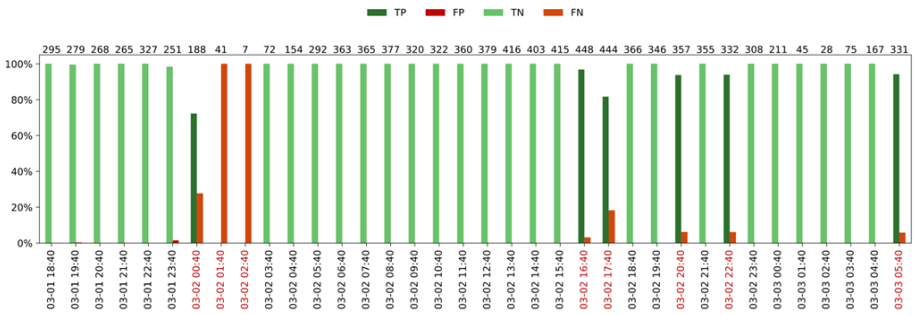

Fig. 11. Hour-by-hour histograms indicating True positives, True negatives, False positives and False negatives for the first day of data.

accuracy	**precision**	**recall**	**f1-score**
91.33%	99.77%	85.74%	92.22%

Table 1. Oslo public transport traffic: quantitative results in terms of accuracy, precision, recall, and f1-score.

In table 1, where we report the overall quantitative results which confirm the fact that the algorithm, after sufficient data for training, shows very high prediction scores, with very high precision.

Conclusions

In this paper, we tackle the task of anomaly detection. For this purpose we extended the algorithm SparkGHSOM, originally designed for the clustering task, in order to consider the task at hand. Furthermore, the main algorithm has been made more explainable by providing the reasons for each detected anomaly in the form of an instance-based feature ranking. The results show the effectiveness of the proposed approach both qualitatively and quantitatively in real application scenarios. For future work, we aim to perform further and more robust experiments with the aim to better evaluate the predictive quality, the explainability, and the scalability of this new extended version of SparkGHSOM.

Acknowledgment

We acknowledge the project IMPETUS (Intelligent Management of Processes, Ethics and Technology for Urban Safety) that receives funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 883286. https://cordis.europa.eu/project/id/883286. Dr. Paolo Mignone acknowledges the support of Apulia Region through the REFIN project "Metodi per l'ottimizzazione delle reti di distribuzione di energia e per la pianificazione di interventi manutentivi ed evolutivi" (CUP H94I20000410008, Grant n. 7EDD092A).

References

1. Gonçalves Jr, P.M., Barros, R.S.: Rcd: A recurring concept drift framework. Pattern Recognition Letters **34**(9), 1018 – 1025 (2013). https://doi.org/10.1016/j.patrec.2013.02.005
2. Hsu, C.C.: Generalizing self-organizing map for categorical data. IEEE Transactions on Neural Networks **17**(2), 294–304 (2006). https://doi.org/10.1109/TNN.2005.863415
3. Kader, G.D., Perry, M.: Variability for categorical variables. Journal of Statistics Education **15**(2) (2007). https://doi.org/10.1080/10691898.2007.11889465
4. Kohonen, T.: The self-organizing map. Proceedings of the IEEE **78**(9), 1464–1480 (1990). https://doi.org/10.1109/5.58325
5. Lloyd, S.: Least squares quantization in pcm. IEEE Transactions on Information Theory **28**(2), 129–137 (1982). https://doi.org/10.1109/TIT.1982.1056489
6. Malondkar, A., Corizzo, R., Kiringa, I., Ceci, M., Japkowicz, N.: Spark-ghsom: Growing hierarchical self-organizing map for large scale mixed attribute datasets. Information Sciences (2018). https://doi.org/10.1016/j.ins.2018.12.007
7. Rousseeuw, P.J.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics **20**, 53 – 65 (1987). https://doi.org/10.1016/0377-0427(87)90125-7
8. Stojanova, D., Ceci, M., Appice, A., Džeroski, S.: Network regression with predictive clustering trees. Data Mining and Knowledge Discovery **25**(2), 378 – 413 (2012). https://doi.org/10.1007/s10618-012-0278-6