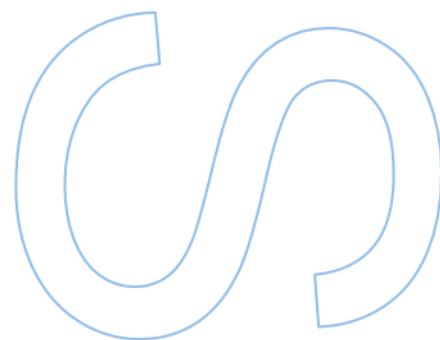
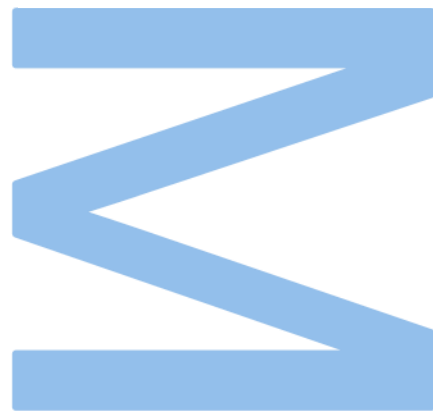
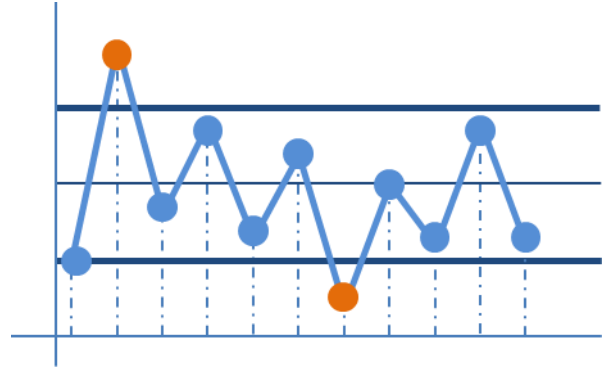


Outlier Explanations in Data Streams – Applications for Environmental Data



Catarina Ferreira Teixeira
Mestrado em Data Science (Ciência de Dados)
Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto
2023/2024



Outlier Explanations in Data Streams – Applications for Environmental Data

Catarina Ferreira Teixeira

Dissertação realizado no âmbito do Mestrado em Data
Science (Ciência de Dados) Departamento de Ciência de
Computadores
2023/2024

Orientadores

Rita P. Ribeiro, Professora Auxiliar, Faculdade de Ciências da
Universidade do Porto

João Gama, Professor Catedrático, Faculdade de Economia da
Faculdade do Porto

Abstract

In many real-world domains, outlier detection constitutes a very important task as it may convey critical information. In those cases, the goal is to trigger an alert so that appropriate preventive or corrective actions are taken in the domain. That is the case of environmental data. The emission or discharge of substances into the air, water and soil are among some of the data that are related to the climate and the environment. These data have a low risk of detection, since is very rare the chance of happening a value out of the normal. Currently, many installed sensors to access air or freshwater quality give us access to streaming data on which outlier detection techniques can improve the identification of abnormal concentrations of substances and potential environmental outliers. Additionally, applying outlier explanation methods can help the constitution of evidence.

Over the last few years, outlier detection in data streams has been a growing research area. Still, few proposals exist related to the explanation of the detected outliers. In a recent survey, different outlier explanations were identified. Still, the challenges and effectiveness of each existing approach are highly dependent on the application domain. For data stream scenarios, some challenges still exist regarding techniques able to cope with concept drift or spatial correlations.

The changes in the environment over time affect the distribution of collected data. This means that what is considered an outlier at a certain period can be regarded as a normal point in another period. Moreover, if the data has a spatial correlation, the data measurement on a particular sensor is related to the data measurement of its neighbouring sensors. Hence, outlier detection and explanation techniques should adapt according to the concept drift and capture all those cross-correlations to arrive at proper explanations.

In this dissertation, we will use outlier detection methods in the context of data streams. With the outlier score returned and a threshold calculated, we will infer an explanation (importance level, outlying attributes, and causal interactions) of why the model classifies the observation as an outlier. Moreover, we will use concept drift detection to explain why those outliers exist, and we will use xAI methods such as LIME to generate explanations for individual instances. Our findings enabled us to successfully explain most of the outliers we encountered and provide a good explanation for the generated figures.

Resumo

Em muitos domínios do mundo real, a detecção de outliers constitui uma tarefa muito importante, pois pode transmitir informações críticas. Nesses casos, o objetivo é gerar um alerta para que sejam tomadas ações preventivas ou corretivas apropriadas no domínio. Esse é o caso dos dados ambientais. A emissão ou descarga de substâncias no ar, água e solo estão entre alguns dos dados relacionados ao clima e ao ambiente. Esses dados têm um baixo risco de detecção, já que é muito rara a chance de ocorrer um valor fora do normal. Atualmente, muitos sensores instalados para acessar a qualidade do ar ou da água doce nos dão acesso a dados em streaming, nos quais técnicas de detecção de outliers podem melhorar a identificação de concentrações anormais de substâncias e potenciais outliers ambientais. Além disso, a aplicação de métodos de explicação de outliers pode ajudar na constituição de evidências.

Nos últimos anos, a detecção de outliers em fluxos de dados tem sido uma área de pesquisa em crescimento. No entanto, poucas propostas existem relacionadas à explicação dos outliers detectados. Em uma pesquisa recente, diferentes explicações de outliers foram identificadas. Ainda assim, os desafios e a eficácia de cada abordagem existente são altamente dependentes do domínio de aplicação. Para cenários de fluxos de dados, alguns desafios ainda existem em relação a técnicas capazes de lidar com deriva de conceito ou correlações espaciais.

As mudanças no ambiente ao longo do tempo afetam a distribuição dos dados coletados. Isso significa que o que é considerado um outlier em determinado período pode ser considerado um ponto normal em outro período. Além disso, se os dados têm uma correlação espacial, a medição dos dados em um sensor específico está relacionada à medição dos dados de seus sensores vizinhos. Portanto, as técnicas de detecção e explicação de outliers devem se adaptar de acordo com a deriva de conceito e capturar todas essas correlações cruzadas para chegar a explicações adequadas.

Nesta dissertação, utilizaremos métodos de detecção de outliers no contexto de fluxos de dados e, com a pontuação de outlier retornada e um limiar calculado, inferimos uma explicação (nível de importância, atributos de outlier e interações causais), justificando o porquê de o modelo sinalizar o outlier. Além disso, usaremos a detecção de desvio de conceito para explicar por que esses valores discrepantes existem e usaremos métodos xAI, como LIME, para gerar explicações para instâncias individuais. Os métodos usados permitem explicar com sucesso a maioria dos outliers que encontramos e fornecer uma boa explicação para as figuras geradas.

Acknowledgments

Queria agradecer antes de mais a toda a minha família e aos meus amigos que me apoiaram imenso ao longo da Licenciatura e Mestrado e que me deram a motivação que precisava para chegar tão longe. Todos os meus momentos de fraqueza que passei ao longo deste mestrado foram um bom desafio a superar e acho que não conseguia ultrapassar sem vocês a chamarem-me à realidade e mostrarem-me que tenho a capacidade para tudo e para não duvidar de mim mesma. Obrigado do fundo do meu coração.

Queria também agradecer a todos os professores com o qual tive o prazer de assistir e participar nas cadeiras que me inscrevi. Tive a oportunidade de aprofundar imenso os meus conhecimentos de *Data Science*, matemáticos e até de Gestão e Empreendedorismo. Obrigada por criarem um mestrado muito prestável para o futuro da informática e do IA.

Por último queria agradecer aos meus orientadores de mestrado, a professora Rita Ribeiro e o professor João Gama, que me deram a oportunidade de explorar um tema não muito estudado ainda na área de *Data Streams* e com o foco interessante num tema que agora é muito falado para termos de segurança e futuro do IA. Obrigada por todos os conselhos e conceitos que me deram que me permitiram terminar esta dissertação com sucesso e por toda a motivação e liberdade de exploração que me deram para deixar a minha veia de *Data Scientist* aqui nesta dissertação.

Para o resto das pessoas que me cruzei ao longo do meu caminho de estudante universitária e que infelizmente o contato não se manteve, obrigado por tudo, vocês permitiram-me evoluir como pessoa e como amiga.

Mais uma vez obrigada a todos por tudo e depois do mestrado segue a continuação da vida de trabalhadora, um passo grande na minha vida e com a capacidade de ultrapassar todos os desafios que irei ter na minha vida profissional graças a este mestrado. Obrigado.

Dedico esta tese a todas as pessoas que considero importantes na minha vida.

Contents

Abstract	i
Resumo	iii
Acknowledgments	v
Contents	ix
List of Tables	xi
List of Figures	xv
Listings	xvii
Acronyms	xix
1 Introduction	1
1.1 Objectives and Motivation	1
1.2 Organization	2
2 Related Work	3
2.1 Notion of Outlier	3
2.2 Outlier Detection Problem	4
2.2.1 Important Aspects	4
2.2.2 Outlier Detection Methods	8
2.2.3 Outlier Detection in Data Streams	12

2.3	Outlier Explanations	15
2.3.1	Types of Outlier Explanations	15
2.3.2	Importance Levels	16
2.3.3	Causal Interactions among Outliers	17
2.3.4	Outlying Attributes	18
2.3.5	Evaluation methods	19
2.4	Explainable Artificial Intelligence (xAI)	21
2.4.1	Types of explainable AI	21
2.5	Discussion	23
3	Methodology	25
3.1	Data Sources	25
3.2	Outlier Detection Approaches	27
3.2.1	Half-Space Trees	28
3.2.2	Incremental Local Outlier Factor	29
3.2.3	One-Class Support-Vector Machine	31
3.2.4	Local Interpretable Model-Agnostic Explanations (LIME)	34
4	Experimental Study	39
4.1	Experimental Setup	39
4.1.1	Data	39
4.1.2	Algorithms	41
4.2	Results	43
4.2.1	Half-Space Trees	43
4.2.2	Incremental Local Outlier Factor	48
4.2.3	One-Class Support-Vector Machine	54
5	Conclusion	61
A	Features Pairplot	63

List of Tables

4.1 Features that describes the dataset	40
---	----

List of Figures

1.1	Example of outlier detection in meteorological data [75].	2
2.1	Example of outliers with a 2-dimensional data set [20].	4
2.2	Key components of Outlier detection techniques (Adapted from [20]).	4
2.3	Examples of point and collective outliers [58].	5
2.4	Example of a contextual outlier in a temperature time series [20].	6
2.5	Classification based techniques [20].	8
2.6	Local density based technique [20].	10
2.7	Box plot for a univariate data set (Adapted from [20]).	11
2.8	An example of data (in a window) partitioned by a simple HST [78].	12
2.9	Sliding window approach (Adapted from [54]).	13
2.10	Landmark window approach (Adapted from [54]).	13
2.11	Damped window approach [87].	14
2.12	Adaptive window approach [16].	14
2.13	Non-overlapping (left) and overlapping time-based window (right) [26].	15
2.14	Type of Outlier Explanations Taxonomy.	16
2.15	Example of outliers in a dataset (Adapted from [59]).	17
2.16	Example of an outlier detected in the 3D space (x,y,z) and projected on the 2D subspace [59].	19
3.1	Original Dataset.	25
3.2	Dataset after the conversion to Dictionary.	26

3.3	Isolation Forest pseudo code.	28
3.4	Examples of isolating different type of points in a 2D Gaussian distribution [5]. .	29
3.5	LOF pseudo code.	30
3.6	Basic idea of ILOF [6].	31
3.7	OCSVM pseudo code.	32
3.8	One-class SVM classification to a 2D training dataset [85].	33
3.9	Influences of outliers to the decision boundary [85].	33
3.10	Black-Box model prediction logic process.	34
3.11	Example of a LIME algorithm for tabular data [64].	36
3.12	Example of a LIME explanation for a single instance [70].	36
4.1	Average of air pollutants in the world between 1750 and 2022. [3]	40
4.2	Annual average of seven air pollutants between 1750 and 2022 in the United Kingdom. [3]	41
4.3	Outlier scores using the HST algorithm.	43
4.4	Outlying attributes of an individual outlier using HST algorithm.	44
4.5	Concept drift detection using ADWIN with HST algorithm.	45
4.6	Iteration of HST algorithm excluding year 1921.	46
4.7	LIME explanation for instance year 2004.	46
4.8	Local explanation for instance year 2004.	47
4.9	Outlier scores using the ILOF algorithm.	48
4.10	Outlying attributes of an individual outlier using ILOF algorithm.	49
4.11	Concept drift detection using ADWIN with ILOF algorithm.	50
4.12	Iteration of ILOF algorithm excluding year 1830.	51
4.13	Iteration of ILOF algorithm excluding year 1921.	51
4.14	LIME explanation for instance year 1951.	52
4.15	Local explanation for instance year 1951.	53
4.16	Outlier scores using the OCSVM algorithm.	54

4.17	Outlying attributes of an individual outlier using OCSVM algorithm.	55
4.18	Concept drift detection using ADWIN with OCSVM algorithm.	55
4.19	Iteration of OCSVM algorithm excluding year 1760.	57
4.20	LIME explanation for instance year 1801.	57
4.21	Local explanation for instance year 1801.	58
A.1	Pairplot with outliers identified using HST algorithm.	63
A.2	Pairplot with outliers identified using ILOF algorithm.	64
A.3	Pairplot with outliers identified using OCSVM algorithm.	65

Acronyms

ALE	Accumulated Local Effects	KNN	K-th Nearest Neighbor
ADWIN	ADaptive WINdowing	LIME	Local Interpretable Model-Agnostic Explanations
AR	Auto Regressive	LOCI	Local Correlation Integral
ARMA	Auto Regressive Moving Average	LOF	Local Outlier Factor
ARIMA	Auto Regressive Integrated Moving Average	LRD	Local Reachability Density
EM	Expectation Maximization	ML	Machine Learning
ESD	Generalized Extreme Studentized Deviate	MLE	Maximum Likelihood Estimates
HST	Half-Space Trees	NN	Neural Network
ICE	Individual Conditional Expectation	OCSVM	One-Class Support-Vector Machine
ILOF	Incremental Local Outlier Factor	PDP	Partial Dependence Plot
iForest	isolation Forest	SHAP	SHapley Additive exPlanations
KDE	Kernel Density Estimator	SVM	Suport-vector Machine
		xAI	Explainable Artificial Intelligence

Chapter 1

Introduction

Outlier detection is a process that aims to identify instances that are significantly different from most cases and can represent a meaningful anomaly in the application domain. It is a noteworthy research topic that poses several challenges. Over recent years [25, 26, 34], a trend has been to extend outlier detection methods to data stream context to detect anomalies in environments where data is processed continuously and in real-time. This scenario makes the task even more complex, as the persistence of anomalies must be verified over time. The aim is to investigate whether an anomaly detected at a particular time is still considered anomalous after a certain period. This move towards implementation in data streaming contexts reflects the need to adapt anomaly detection techniques to dynamic and constantly evolving environments to enable a more robust and real-time understanding of anomalous patterns in the data.

More recently, a growing interest has emerged regarding the application of eXplainable Artificial Intelligence (XAI) techniques to explain outliers or anomalies. The idea is to clarify why a particular outlier was identified, how it was classified, and other potentially relevant aspects [10].

1.1 Objectives and Motivation

This thesis aims to apply outlier detection algorithms in a continuous data stream context. It also seeks to incorporate techniques to explain outliers to interpret the algorithms' results and justify classifying specific points as outliers over others.

The central motivation of this study is to investigate the premise of explaining outliers, focusing on the classification and interpretation of these outliers and considering the variations of algorithms and methods used in continuous data streaming. This topic has not yet received much attention in the field of data streams. Therefore, this study aims to contribute to the scientific community by filling this gap.

The application of the premise of explaining outliers in the context of data streaming will

be centered on environmental data, a problem that has received a lot of attention nowadays due to global warming or the illicit release of toxic components into the environment, which can have an influence on our daily lives. Figure 1.1 represents an example of outlier detection of meteorological data obtained from the Citizen Weather Observer Program (CWOP) [2] for the relative humidity values [75].

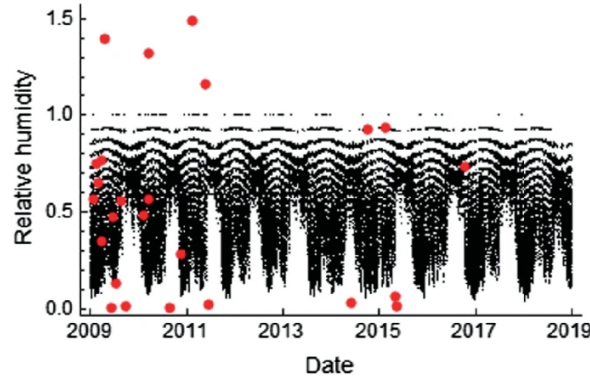


Figure 1.1: Example of outlier detection in meteorological data [75].

1.2 Organization

The rest of the document is organized as follows:

- **Chapter 2** provides comprehensive information about the underlying theory of outlier detection, the explanation of outliers and the processing of data streams. This chapter also details concepts that are particularly relevant to the purposes of this research.
- **Chapter 3** outlines the methodology adopted for the detection and explanation of outliers in the context of environmental data. It discusses relevant data sources, data processing and the application of data stream methods. Additionally, it describes different xAI methods and justifies the choice of LIME for this study.
- **Chapter 4** describes the experimental study conducted on a selected case study related to the concentration of air pollutants over a period of time and critically analyzes the results from the perspective of outlier detection and explanation.
- **Chapter 5** concludes this study by discussing the suitability of the selected methods for the task at hand. Additionally, it presents potential directions for further improvement of the work.

Chapter 2

Related Work

This chapter introduces basic concepts such as outlier detection, data streams, outlier explanations and reference to previous research in this field. The goal is to give a theoretical foundation for the comprehension of the main topics discussed throughout the thesis.

2.1 Notion of Outlier

Few outlier definitions exist in the literature. According to Grubbs [33], *"an outlier is an observation that appears to deviate markedly from other members of the sample in which it occurs"*. Hawkins [36] defined them as *"observations that deviate so strongly from other observations that they are suspected to have been produced by some other mechanism"*. Outliers are patterns in the data that do not correspond to a precisely defined idea of normal behaviour [20].

Similarly, anomalies are defined by the deviation from expected patterns or standards in data. While the term *"outlier"* is commonly used in statistical contexts, *"anomaly"* has a broader use in domains such as data science and machine learning, where it refers to single points and unexpected patterns or behaviours within the data. Outliers and anomalies are critical for spotting potential errors, significant occurrences, or underlying dataset flaws. Outliers may be anomalies, but the reverse is not always true.

To identify outliers in a dataset, keep in mind that outliers differ from the norm based on their features, resulting in a unique outlier score. Additionally, outliers are typically rare in comparison to normal instances, resulting in a small number of outliers (ranging from 2% to 5%) [32].

Figure 2.1 shows an example of outliers in a two-dimensional dataset as depicted in [20]. From this figure, we can see that there are two normal regions N_1 and N_2 because most of the points are located in dense regions and the points that are further away from these regions are outliers. These include the points o_1 , o_2 and the points included in the region O_3 . All of them with neighbour density different from the majority of the other points.

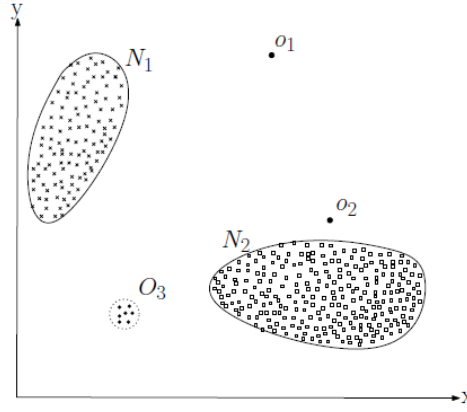


Figure 2.1: Example of outliers with a 2-dimensional data set [20].

2.2 Outlier Detection Problem

2.2.1 Important Aspects

There are many different approaches to detect outliers because the formulation of the problem is determined by different characteristics, as enunciated by Chandola in [17, 20] and depicted below in Figure 2.2.



Figure 2.2: Key components of Outlier detection techniques (Adapted from [20]).

Nature of Input Data

The nature of the input data is the most important aspect of using an outlier detection technique. This is generally a collection of data instances (e.g. object, observation, sample), and each data instance is described by a set of attributes (variables, features). These attributes can be of different types, e.g. binary, categorical or continuous. Each data instance can consist of a single attribute (univariate) or several attributes (multivariate). Multiple attributes can be a mixture of different data types or the same type for all attributes.

The nature of the attributes can determine the applicability of outlier detection techniques. Input data can also be categorized based on the relationship between data instances, but most techniques dealing with outlier detection on datasets do not assume a relationship between data instances.

Data instances can be related to each other, e.g. sequence, spatial and graphic data. In

sequence data, for example, the data instances are ordered linearly (e.g. time series); in spatial data, each data instance is related to its neighbouring instances (e.g. traffic data); in graph data, the data instances are represented as vertices in a graph and connected to other vertices by edges; and spatial data that has a sequential content is referred to as spatiotemporal data (e.g. climate data).

Type of Outliers

When choosing a strategy to explain the outliers noticed, we must first determine what type of outliers we are dealing with, as each will require a different solution approach (Figure 2.3 illustrates the types of outliers that exist.). We have the following categories of outliers [17, 20, 21, 34]:

- **Point Outliers** - If a single instance of data can be considered anomalous with respect to the rest of the data, that instance is called a point outlier. This is the simplest type of outlier. They can be further classified as:
 - *Global Outliers* - measured sample points with an extremely high or very low value compared to all the other values in a dataset. For example, if 99 out of 100 points have values between 300 and 400, but the 100-th point has a value of 750, the 100-th point could be considered a global outlier. In summary, it refers to the deviation from the entire dataset.
 - *Local Outliers* - determined based on characteristic differences (e.g. neighbourhood density) between the outlier and its nearest neighbours. They are measured sample points whose values are within the typical range for the entire dataset but are exceptionally high or low compared to the surrounding points.
- **Collective Outliers** - If related data instances are anomalous concerning the entire data set, this is a collective outlier. They do not have to be anomalous individually but are anomalous as a whole (very common with sequence data).

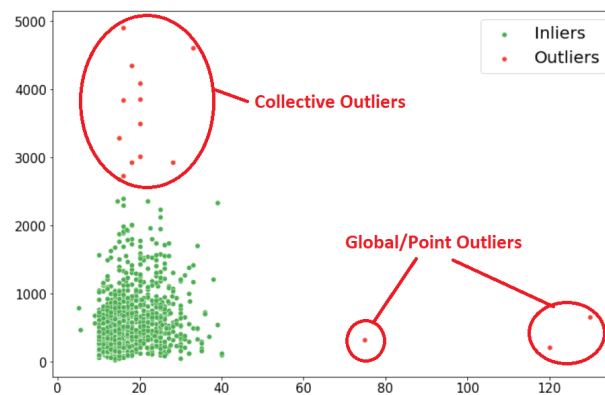


Figure 2.3: Examples of point and collective outliers [58].

- **Contextual Outliers** [49] - If a data instance is anomalous in a particular context, it is a contextual or conditional outlier. The structure of the data set conditions the notion of context and must be specified as part of the problem formulation. Each data instance can be defined with the attributes below:
 - *Contextual Attributes* - define the context or neighbourhood for this instance (used in time series data);
 - *Behavioral Attributes* [25] - define the non-contextual characteristics of an instance (e.g. the spatial data set of the average amount of precipitation of the entire world)

Taking the example of Figure 2.4, a data instance may be a contextual outlier at a certain point in a certain context (t_2). In contrast, another data instance may be considered normal in a different context (t_1).

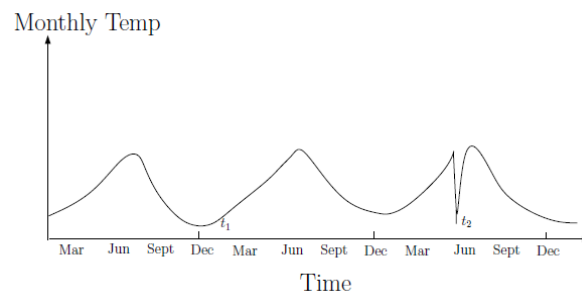


Figure 2.4: Example of a contextual outlier in a temperature time series [20].

Data Labels

If available, the labels associated with an instance can classify an instance as "normal" or "outlier". Still, it is difficult to obtain such labelled set of anomalous data instances that covers all possible types of anomalous behaviour than to get labels for normal behaviour, and abnormal behaviour is often dynamic (i.e., more outliers may occur). Depending on which labels are available, outlier detection techniques can be classified into one of the following learning tasks [17, 20, 37]:

- **Supervised** - Assumes the availability of a training data set containing labelled instances for normal and abnormal classes to build prediction models. It can be considered as a binary classification problem.
- **Semi-supervised** - Assumes that the training data only contains labelled instances for a partial set of instances, usually the ones from the normal class (e.g. One-class Random Forest [27]). Since they do not require labels for the outlier class, they are more common than supervised techniques.

- **Unsupervised** - Uses unlabelled data to either build model for outlier score calculation (e.g. Isolation Forest [48]) or directly calculate outlier scores for the input data without building models (e.g. LOF [18]).

Output of Outlier Detection

It is essential in any outlier detection technique to pay attention to how the outliers are reported, such as the following:

- **Labels** - Techniques in this category assign a label (normal or outlier) to each test instance.
- **Scores** - Assigns an outlier score to each instance in the test data, depending on how much that example is considered an outlier. Therefore, the output of this technique is a ranked list of outliers, with the top being the most anomalous and the bottom being the least anomalous. Scoring techniques allow the threshold value to be used to select the most important outliers.

Applications

There are various applications for the detection of outliers [28, 86], some examples are: *Intrusion Detection* that is a security mechanism designed to detect unauthorized access or breaches in a computer system or network; *Fraud Detection* that is a set of processes and techniques used to identify and prevent fraudulent activities; *Fault/Damage Detection* that refers to the processes and methods used to identify and diagnose faults or damage in systems, structures, or machinery; *Medical Informatics* that deals with the collection, storage, retrieval, and use of healthcare information to foster better collaboration among a patient's various healthcare provider and so many more.

For example, a number of models and methods have been developed in the literature to define and detect insider threats in the context of *Intrusion Detection* and *Fraud Detection*, respectively. These detection algorithms often create users' or nodes' profiles based on their actions such as login/logon, file access, email, instant messaging, and web surfing, with machine learning [41] and deep learning models [81] being used to train classification models for outlier detection. Existing insider threat detection and fraud detection models are mostly concerned with finding single outlier users or nodes, with few schemes capable of predicting or detecting complex insider threat or fraud scenarios involving collective users or nodes [42].

Outlier detection in *Medical Informatics*, which corresponds to atypical patient management behaviors, will aid in the identification of medical errors. The outlier-based monitoring and alerting strategy can be used in conjunction with knowledge-based alerting systems to improve overall clinical coverage. In clinical sub-areas where knowledge-based warning is still unavailable, this novel approach can act as a standalone system, with all statistical outliers corresponding to medical errors [35].

2.2.2 Outlier Detection Methods

Because outlier detection applications are used on-demand, they require high levels of efficiency, such as intrusion detection in computer networks. These specifications for outlier detection are also applicable to data streams.

Following the explanation of outlier detection and its characteristics, we have a variety of outlier detection methods [20, 21] that provides a solution to the problem in question. Based on their specifications, all these approaches can be used for unsupervised learning, which is the task our study will center on. The types of specifications [20, 34] are as follows:

Classification

Operate in a similar two-phase process, the training and test phases. Given the feature space to learn, they assume the classifier can distinguish between normal and outlier classes. Based on the labels available for the training phase, the classification techniques can be grouped into two categories: one-class outlier 2.5(b) and multi-class outlier 2.5(a). One-class outliers assume that all training instances have only one class label, and multi-class outliers assume that the training data contains labelled instances belonging to multiple normal classes.

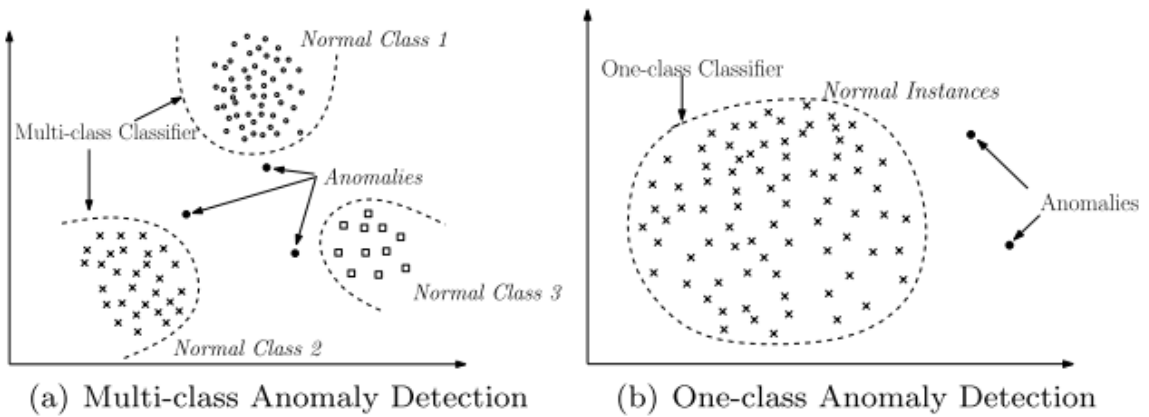


Figure 2.5: Classification based techniques [20].

And for that classification technique, we have different algorithms for every approach we want to follow.

- *Neural Networks* - able to function in both single and multi-class environments. The two steps of a multi-class outlier detection technique are as follows. Each test instance serves as an input for the Neural Network (NN) after the neural network has first been trained using the training set. It is seen as normal if it accepts the input; otherwise, it is deemed abnormal.

- *Bayesian Networks* - used when there are multiple classes. A naïve Bayesian network is used in a basic technique for univariate categorical data to estimate the posterior probability of observing a class label given a test data instance. The class label with the biggest posterior is then selected as the predicted class for the given instance.
- *Support Vector Machines* - Support-vector Machine (**SVM**) are applied to outlier detection as a one-class setting. Such a model learns a boundary or a region where all normal data is contained. If a test instance does not fit the region/boundary, it is classified as an outlier [51].

Nearest Neighbour

Is a distance-based technique that requires distance or a similarity measure between two data instances to be computed. The distance or similarity between them can be computed in many different ways. For example, we can use the Euclidean distance for continuous attributes, and for categorical attributes, the simple matching coefficient is one of the most used distance measures.

- *Distance the k^{th} Nearest Neighbour* [45] - Also known as the K-th Nearest Neighbor (**KNN**) technique [11]. For each test instance, determine the kth nearest neighbour in the training set being used after as outlier scores. It can also be applied to discrete sequences using the length of the longest common sub-sequence as a measuring distance [19, 21]. It can also compute using the pair-wise Euclidean distance between all data points, and if one data point has fewer than k neighbours within distance R , it is termed an outlier. Numerous strategies were proposed to extend outlier identification to streaming data ([31, 79]). One of the most frequent approaches for finding outliers is to utilize a sliding window to help detect outliers.
- *Relative Density* - Can estimate the density of the neighbourhood of each data instance. An instance that lies in a neighbourhood with a low density is considered an outlier. One of the most famous examples is Local Outlier Factor (**LOF**) [18, 34] that, for any data instance, the LOF score will be equal to the ratio of the average local density of the k-th nearest neighbours of the instance and the local density of the data instance itself. In Figure 2.6, we can verify that the LOF technique will capture $p1$ and $p2$ as outliers because the technique considers the density around the data instances. Another prominent density-based approach is Local Correlation Integral (**LOCI**) (Local Correlation Integral). This work [61] demonstrated an incremental version of **LOCI** on streaming data. The authors provided theoretical arguments to demonstrate that inserting new data points and identifying an old data point affects only a subset of neighbours. Outlier detection is independent of the total number of records in the current dataset and is limited by the time complexity after inserting n data points.

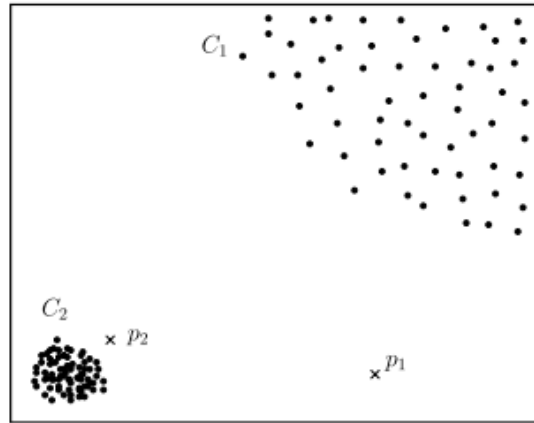


Figure 2.6: Local density based technique [20].

Clustering

Is used to group similar data instances into clusters. This technique is primarily unsupervised and usually takes a two-set procedure: grouping the data with clustering algorithms and analyzing the degree of deviation based on the clustering results. Several techniques follow the two-set procedure [34, 40, 74]; for example, K-Means clustering is a vector quantization method that divides n observations into k clusters, assigning each observation to the cluster with the nearest mean. Another one is Expectation Maximization (EM), which can find a parameter estimator that determines the distribution of the data [24]. It is employed to find outliers. The data in each sliding window is grouped, but unlike the distance-based approach, found outliers are not immediately reported but rather treated as potential outliers. A metric that measures the mean value of each cluster is kept and carried over to the next window in the stream, where it can be compared against data from subsequent windows.

Statistical

One of the assumptions is that normal data instances occur in high-probability regions of a stochastic model, while outliers occur in the low-probability regions of the stochastic model. They can be considered as *Parametric*, which is the detection model that is created based on data distribution (e.g., Gaussian distribution) [38]. One of the most common distributions is the Gaussian mixture, in which each data point is assigned a defined score. A data point with a score higher than a certain threshold is classified as an outlier. Another prominent model is autoregression, which creates a prediction model and specifies a threshold limit, with an outlier occurring if the value exceeds the threshold limit. It assumes that a parametric distribution and a probability density function generate the normal data. The outlier score of a test instance will be the inverse of the probability density function.

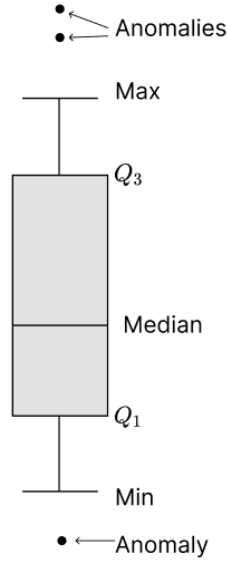


Figure 2.7: Box plot for a univariate data set (Adapted from [20]).

Inside the parametric type of a statistic based model, since we use a distribution function, we can also classify the type of model based on that. We will have the *Gaussian Model*, which is produced using a Gaussian distribution. The parameters are estimated by Maximum Likelihood Estimates (MLE). The gap between a data instance and the estimated mean determines its outlier score. After that, a threshold is applied to the outlier scores to determine which outliers are not; additionally will also have the *Regression Model* which consists of two steps: the regression model is fitted to the data, and the residual for each test case is utilized to calculate the outlier score. For example, in the Auto Regressive (AR) model [9, 34, 69], data points are linearly dependent on prior values and a stochastic term, resulting in a random process. The Auto Regressive Moving Average (ARMA) model [29, 34] considers outlier scores from earlier data. We shall use the Auto Regressive Integrated Moving Average (ARIMA) model [15, 34, 80] to address the issue of older data becoming uninteresting with time.

Kernel-Density

The Kernel Density Estimator (KDE) is a non-parametric approach to estimate the probability density function of random variables [68]. It has become an effective method for detecting outliers in streaming data. The probability density function $f(x)$ is defined as follows:

$$f(x) = \frac{1}{n} \sum_{i=1}^n k_{h_i}(x_i - x) \quad (2.1)$$

$k_{h_i}(x)$ represents the kernel functions with bandwidth h_i . It is calculated using Scott's Rule [68]. This method employs a non-overlapping sliding window and a statistical binned summary to detect high-volume and high-dimensional streaming data abnormalities.

Ensemble

Ensemble-based models are machine learning models that integrate the predictions or outputs of numerous independent models (known as base learners or weak learners) to produce a single prediction or score. These models are intended to increase prediction accuracy, robustness, and generalization by utilizing numerous learners' diversity and collective intelligence. Examples of that type of model are isolation Forest (**iForest**) and Half-Space Trees (**HST**) (Figure 2.8).

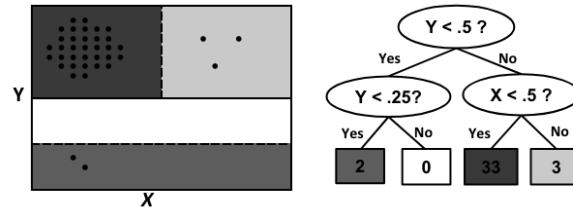


Figure 2.8: An example of data (in a window) partitioned by a simple HST [78].

2.2.3 Outlier Detection in Data Streams

Data stream is the continuous, real-time data transfer from a source system to a target. It refers to the efficient and continuous flow of data, often in small, manageable pieces or packets, from one location to another. Several studies have been conducted on data stream analysis [12–14, 26]. Many of them have dealt with detecting outliers in data streams. Recent work on this topic has mostly relied on statistical methods such as identifying sliding thresholds, Generalized Extreme Studentized Deviate (**ESD**) test, k-sigma, statistical hypothesis testing, and Holt-Winters exponential smoothing.

Recent work in the data stream field has focused mostly on statistical methods [15, 76], such as finding the sliding threshold and the test methods described above. This procedure enables the instantaneous processing and analysis of data as it is generated or received, eliminating the need to store the complete dataset before analysis. Streams might be time-series or multidimensional. The temporal dependencies in multidimensional streams differ from those in time series.

In the work presented in [76], spatial outliers can be detected using simple thresholds and Holt-Winters exponential smoothing. Other approaches involve modelling time series in two independent floating windows and identifying significant differences in their metrics. Statistical methods have rapid computation times and do not overburden memory. However, in a streaming environment, these systems perform worse at detecting outliers since they are sensitive to changes in window size and threshold. This is because constantly entering changing signals generates continuous threshold adjustments, resulting in the detection of false alarms.

2.2.3.1 Windowing approach

Outlier detection in data streams presents two key challenges. The initial challenge is dealing with storage memory. Keeping the entire stream of data points in memory from start to finish can be difficult due to their constant arrival and infinite sequence. Due to their on-the-fly nature, outlier detection applications require high efficiency. To address these issues, windowing is a popular method that entails creating incremental models based on a portion of the data stream, usually the most recent. These models are then updated to reflect changes in the data points. There are four kinds of windowing techniques [54, 66]:

- **Sliding Window** - A fixed-width window slides over the data stream. Outlier detection is limited to the most recent data points (Figure 2.9).

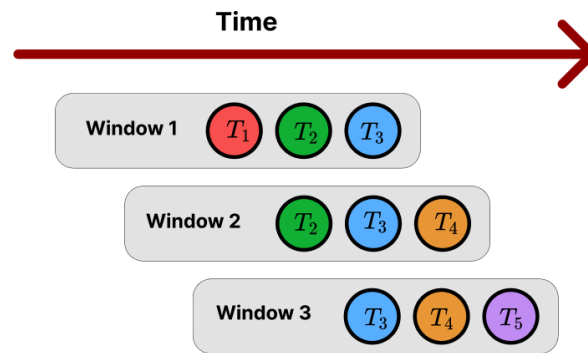


Figure 2.9: Sliding window approach (Adapted from [54]).

- **Landmark Window** - A certain point in the data stream is designated as the landmark. The outlier detection method considers the data sequence between the landmark and the current data point. As processing data grows, memory storage becomes a significant challenge (Figure 2.10).

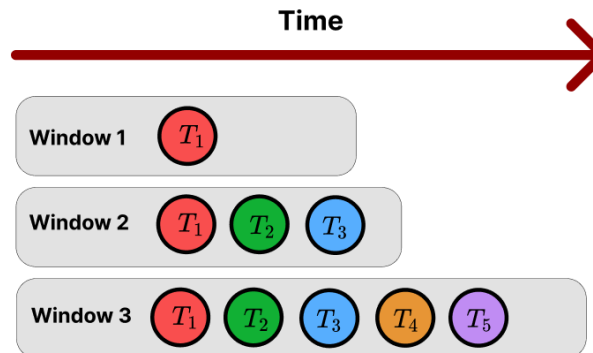


Figure 2.10: Landmark window approach (Adapted from [54]).

- **Damped Window** - Each data point is allocated a weight based on its timing and order of arrival. Newer data points are typically given higher weights to accurately reflect current trends in detection results (Figure 2.11) [87].

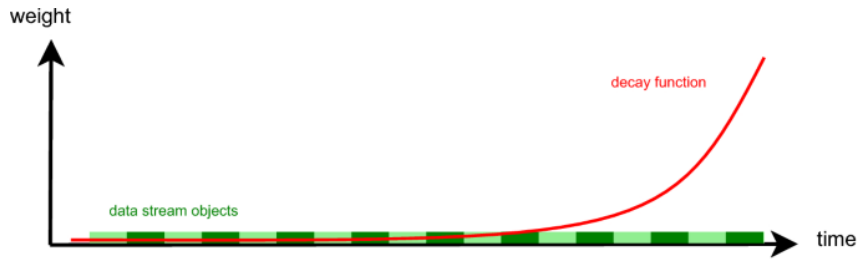


Figure 2.11: Damped window approach [87].

- **Adaptive Window** - The adaptive window is similar to the sliding window, but its width changes based on the rate of change in the data within the window. The window extends when the data is stationary and constricts when it changes (Figure 2.12) [16].

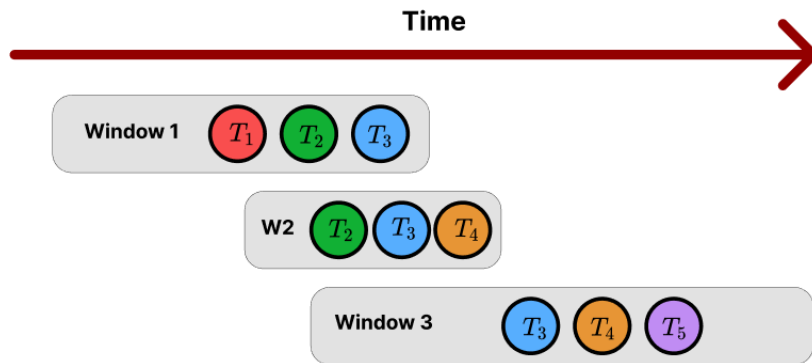


Figure 2.12: Adaptive window approach [16].

One of the most prominent outlier identification techniques that address the temporal notation of streaming data utilizes a sliding-window approach [31, 46, 79]. Each sliding window stores a chunk of the data with the same temporal context. Outliers are decided exclusively on the data in memory, which is destroyed when it expires.

The window size determines when the data point expires, whereas the slide size dictates how frequently an algorithm containing this data point is executed. A sliding window can be classified as:

- **Time-based** - the windows and slide size are both defined as time intervals [46];
- **Count-based** - Windows size is measured as a fixed number of data points, whereas slide size is a predetermined number of data points, after which the count for a new window begins.

For both categories, depending on the relationship between the window and slide size, a sliding window can be further classified (Figure 2.13) as:

- **Non-overlapping window** - If the window size equals the slide size, each data point will only be used for calculation once.
- **Overlapping window** - Each data point will be calculated many times because the window size differs from the slide size.

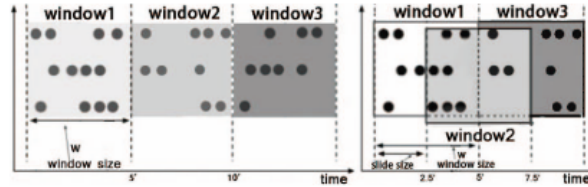


Figure 2.13: Non-overlapping (left) and overlapping time-based window (right) [26].

2.3 Outlier Explanations

When working on a project to forecast outliers, one of our primary objectives is explaining the outcomes, which involves understanding why and how the outlier was regarded as an outlier.

According to [59], an explanation is "the act or process of explaining", which provides a reason or cause for why a specific aspect occurs or is formed. This explanation may differ from how others define, select, and assess explanations.

In the case of outliers, the explanations will help the user comprehend them and can be used to better the outlier detection task. The explanations can lead to additional implementation, such as deciding what to do with the discovered outliers, how they are used, etc.

2.3.1 Types of Outlier Explanations

There are various factors to consider when determining whether an explanation is good from the user's standpoint. In general, there are seven different parts to human-friendly explanation [55], which are as follows:

1. Explanations are contrastive, which means they emphasize the most significant difference between the object of interest and other objects;
2. Explanations are chosen, which means that even if a real-world occurrence has multiple causes, individuals seek only one or two of them as explanations;
3. Explanations emphasize the abnormal, i.e., people tend to concentrate on unusual causes that were unlikely to occur but yet occurred;
4. Explanations are social, which means that they should fit the demands of the users examining the outliers;

5. Explanations are truthful, which implies that they have real-world validity;
6. Good explanations are consistent with the xAI methods past ideas, which implies that the explanations should be congruent with the user's prior judgment about the identified outliers;
7. Good explanations are general and probable, encompassing multiple events, which refers to the proportion of present and future outliers the job can explain.

Considering the task of outlier explanation, we can group the above points into groups:

- **Non-evaluative aspects** - Points (1) through (3) are connected with explanations generated by an outlier explanation algorithm, regardless of how the explanations are evaluated;
- **Evaluative aspects** - The remaining points (4–7) refer to the criteria used to evaluate the generated explanations.

Since our data will be unsupervised, we will primarily concentrate on the first group for this thesis. We will not address the second group because the methods presented can only be applied if we have the ground truth, which is not valid in our situation. The taxonomy of the existing types of outlier explanations is represented in the Figure 2.14.

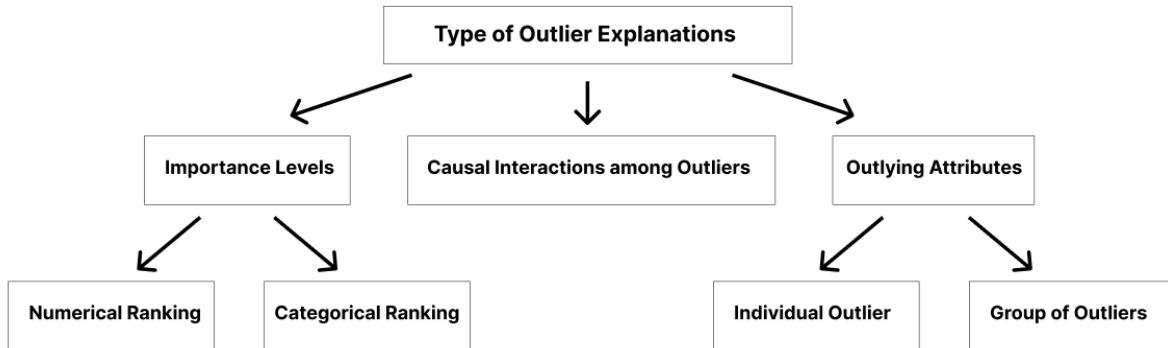


Figure 2.14: Type of Outlier Explanations Taxonomy.

2.3.2 Importance Levels

Two different strategies that can be used to convey the level of importance of outliers [59] are the numerical and categorical ranking. Below, we will discuss their definition and purpose.

2.3.2.1 Numerical Ranking of Outliers

This strategy ensures that the data objects are arranged according to their outlier ranking. An outlying scoring function produces a real number value known as the outlier score.

An object's outlier score increases with object deviance. More deviant objects should be prioritized over less deviant ones in adequate numerical ranking systems. In conclusion, the normal data is ordered last, and the strongest outliers are ranked first [67].

For instance, in Figure 2.15, x_1 has a higher deviation than x_2 ; therefore, x_1 's outlier score will be higher than x_2 's. x_1 will therefore rank higher than x_2 as a result.

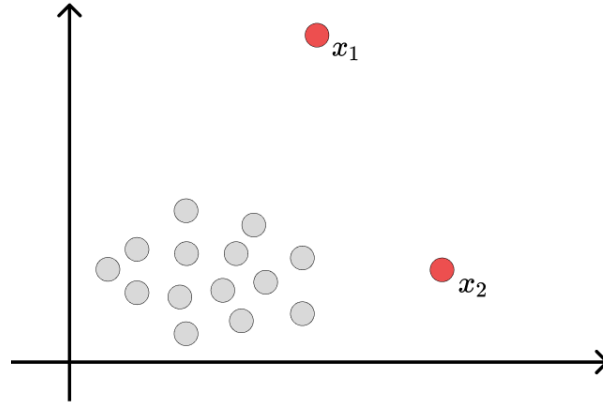


Figure 2.15: Example of outliers in a dataset (Adapted from [59]).

Outlier ranking is a method that helps speed up the process of solving problems by concentrating on the user's viewpoint to identify which points should be evaluated first. It can be used, for instance, to assist system administrators in prioritizing which course of action to take in the event of a network breach.

2.3.2.2 Categorical Ranking of Outliers

Outliers can be classified into several groups depending on the importance-level criterion. Thanks to the categorization, users can determine which outliers should be given priority. The categorization also shows a ranking of outliers.

According to the definition given in [67], outliers are classified according to how much they deviate from the rest of the dataset; the categories correspond to the relative significance or severity of the departure. In this type of outlier analysis, the noise in the data can be disregarded and eliminated because it will impact the data mining outcome. Significant outliers should then be the subject of additional investigation.

2.3.3 Causal Interactions among Outliers

An outlier's potential to cause the appearance of other outliers can be used to describe a causal interaction between outliers. Based on the database causality definition in [52] and the outlier casual relationship definition in [50], according to the definition on [67], it can be stated that:

- i) There is a relationship among outliers if an outlier can lead to the occurrence of other outliers;
- ii) Given a set of outliers $O = \{o_1, \dots, o_n\}$, o_i caused o_j :
 - If o_i 's timestamp is older than o_j 's timestamp;
 - Removing o_i from O also removes o_j from O .

One example of causal interaction among outliers can be found in traffic applications, where by examining the traffic outliers, one can uncover that some anomalies lead to others [50, 83].

2.3.4 Outlying Attributes

Refers to the feature subspace in which outliers deviate significantly from normal data, i.e. the features that contribute the most to the outliers' abnormality. This explanation can include the outlying characteristics of a single outlier or a group of outliers.

2.3.4.1 Individual Outlier

This outlier explanation, refers to the feature subspace or attribute subset accountable for the outliers and can be interpreted in two ways:

1. the smallest subspace where the outlier score is greater than a threshold [43, 71];
2. the subset of attributes where each member has a contribution score higher than a threshold [49, 53, 77].

An item can be discovered as an outlier in the entire dimension space D , but it can be easily distinguished from the inliners in the subset S of the original attributes [53].

In Figure 2.16, the red point is an outlier in the 3-dimensional space $D = \{x, y, z\}$. However, the 2-dimensional projection shows that $S = \{x, z\}$ or $S = \{y, z\}$ could be the outlying attributes of the red point. Assuming the red point has the following contribution score threshold for its characteristics $\{x = 0.1, y = 0.1, z = 0.8\}$ and the contribution score threshold is 0.2, the red point outlying attributes are $\{z\}$.

Knowing the subspace where an outlier stands out might help analysts save time and effort when determining its status. Furthermore, because it takes less time to analyze each outlier, analysts can check more highlighted points in a given period.

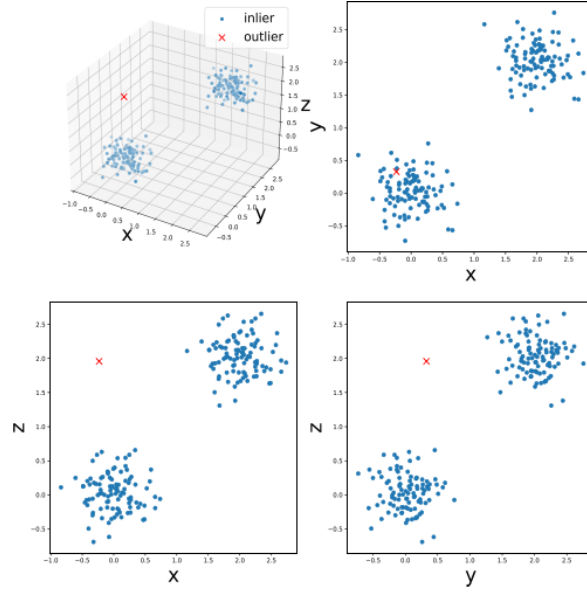


Figure 2.16: Example of an outlier detected in the 3D space (x,y,z) and projected on the 2D subspace [59].

2.3.4.2 Group of Outliers

This outlier explanation refers to the attributes responsible for the abnormality of each outlier in the group. In other words, it is the feature subspace in which the outliers diverge from the usual data in the dataset.

This explanation, like the outlying qualities of an individual outlier, is practical, particularly for high-dimensional datasets, because evaluating every subset of their attributes is wasteful, if not impossible. Analysts can benefit from understanding which attributes are responsible for a collection of outliers. Instead of validating each individual outlier, analysts can simultaneously verify the outlying attributes of each group of outliers. Furthermore, this explanation assists analysts in identifying potentially crucial outliers that occur repeatedly.

2.3.5 Evaluation methods

2.3.5.1 Evaluative aspects of outlier explanations

As these are abstract metrics, quantifying them is necessary when evaluating the outlier explanation task findings. We will discuss how the surveyed techniques solve these issues.

Most examples of evaluating numerical ranking of outliers primarily focus on these studies [47, 49, 72, 82]. The criteria used are ROCAUC, accuracy rate and the number of real outliers in the top ranking.

For categorical ranking of outliers, the sole example we have [44] does not analyse any of

these points. It prioritizes I/O performance, such as the minimum number of I/Os saved and the maximum number of times a database must be read while calculating outlier categories.

To assess the causal relationships among outliers the examples [50, 83] employs real-world traffic data, and they found that the outlier causal relationship discovered by their algorithms is consistent with the causal occurrences in real life. However, they do not provide accurate statistics for the matching causal events.

When analyzing the outlying attributes of an individual outlier, this study [23] concludes that the interpretation form for outliers is intuitive and understandable based on the feature visualization of the top 5 outliers discovered. Another study [53] uses the Jaccard Index and Precision to evaluate the accuracy rate of explanation task output. These and many more studies use evaluation points to explain outliers.

When assessing the outlying attributes of a group of outliers, these examples [22, 39] focus only on the execution times. Another example [73] uses manually labelled outliers and outlier injection to evaluate the accuracy of their classification algorithm, which includes discovering the outlying attributes.

2.3.5.2 Methods to generate outlier explanations ground truth

Modifying real-world datasets - According to [53], real-world datasets lack ground truth for outlying attributes. To utilize a real-world dataset, add M noise characteristics to all data instances, and assume that all original attributes are outliers while noise attributes are not. To avoid overlap between the outlier and inliner classes, the authors fixed the radius of the synthetic sampling for creating the outlier class to half the instance distance from the outlier point to the inliner classes. They will provide precision, recall, and F1-scores for accuracy.

Generating synthetic datasets from scratch - In [22], an empirical examination of outlier patterns is conducted using synthetic data rather than a real-world dataset. They use a Poisson distribution to establish the size of the knowledge set and then assign attributes at random. They use a uniform distribution to produce data points for features not included in the knowledge sets.

Simulating analysts - Most surveyed techniques do not engage expert users in evaluating their proposed strategies. However, [72] evaluates outlier ranking using six benchmark datasets based on audit logs created during red team attack campaigns. During their experiments, the outlier rating was reported to the red team, who then summarised each attack scenario, including the essential entities and events. The descriptions specify whether or not each object and event is a part of the attack. They use the descriptions to create a ground truth to evaluate their algorithm and imitate the input supplied by a systems analyst.

2.4 Explainable Artificial Intelligence (xAI)

Explainable artificial intelligence (xAI) is a set of processes and strategies that enable human users to understand and trust the results and output produced by machine learning algorithms.

Explainable AI describes an AI model's projected impact and probable biases. It contributes to model correctness, fairness, transparency, and outcomes in AI-powered decision making. Explainable AI is critical for a company to establish trust and confidence when bringing AI models into production. AI explainability also enables an organization to take a responsible approach to AI development.

There are numerous advantages to understanding how an AI-enabled system produced a given result. Explainability can help developers ensure that the system works as planned, it may be required to meet regulatory standards, or it may be critical in allowing persons affected by a decision to contest or amend the outcome.

Explainable AI is a critical criteria for adopting responsible AI, an approach for large-scale AI adoption in real-world businesses that prioritizes fairness, model explainability, and accountability. It can improve the user experience of a product or service by increasing the end user's faith in the AI's decision-making abilities [10].

2.4.1 Types of explainable AI

Existing types of explainable AI, such as model-agnostic interpretation approaches, isolate explanations from the ML model ([64]). The book [55] discusses model-agnostic interpretation approaches, categorizing them as Global Model-Agnostic methods and Local Model-Agnostic Methods, both of which contain different types of xAI methods. Because our work uses unsupervised learning, we only provide xAI methods that are relevant to that form of learning.

Global Model-Agnostic approaches describe the average behaviour of a ML model. Expected values are frequently represented as a function of data distribution. Because global interpretation methods represent typical behavior, they are especially valuable for understanding the general mechanisms in data or debugging a model. We will have the following techniques:

- *Partial Dependence Plot (PDP)* - Is a machine learning visualization tool that displays the link between one or more features and a model's predicted outcome while averaging out the influence of other factors. PDPs assist in the interpretation of complex models by providing insights into how specific properties influence predictions;
- *Accumulated Local Effects (ALE) Plot* - Are a model-independent interpretability tool for visualizing and comprehending the impact of features on the predictions of complex machine learning models. ALE plots overcome some of the shortcomings of Partial Dependence Plots (PDPs), particularly in terms of feature interactions and correlations;

- *Feature Interaction* - Occurs when the effect of one feature on the model's prediction is determined by the value of another feature. Understanding feature interactions is critical for comprehending complicated models because they can dramatically affect model performance and prediction accuracy;
- *Functional Decomposition* - Is the process of breaking down complex predictive models or functions into simpler, interpretable components. This technique aids in understanding how different sections of the model contribute to the overall forecast, hence improving interpretation and transparency;
- *Permutation Feature Importance* - Assess the importance of individual features in a prediction model. It aids in determining which features contribute the most to the model's predictions by evaluating the influence on model performance when feature values are randomly mixed;
- *Global Surrogate* - Is a simple, interpretable model that approximates the predictions of a more complicated, black-box model. Global surrogates contribute to understanding and describing the behavior of the original complex model by offering a straightforward approximation;
- *Prototypes and Criticisms* - Prototypes and critiques are interpretability techniques used in machine learning to provide information about model behavior and data representation. These strategies aid in data comprehension by distinguishing between typical (prototypes) and atypical or problematic cases (criticisms).

Local Model-Agnostic Methods are strategies for interpreting and explaining individual predictions from complex machine learning models. Unlike global approaches, which provide insights into the overall model behavior, local methods focus on understanding the model's behavior in specific cases. The techniques are as follows:

- *Individual Conditional Expectation (ICE)* - Are a machine learning visualization approach that helps users comprehend the link between a particular characteristic and the anticipated outcome for individual data points. ICE charts show how the prediction changes when one feature changes while the other features remain constant;
- *Local Interpretable Model-Agnostic Explanations (LIME)* - Is a machine learning interpretability technique that explains black-box model predictions at the local level. It approximates the decision boundary around a specific instance by fitting an interpretable model, providing information about how the model made its prediction for that instance [65];
- *Counterfactual Explanations* - Counterfactual explanations are a strategy used in machine learning interpretability to provide insights into why a model made a specific prediction by simulating alternative scenarios that might have resulted in a different prediction. These explanations help in understanding a model's decision-making process and can be useful for debugging, fairness, and trustworthiness evaluations;

- *Scoped Rules (Anchors)* - Are a machine learning interpretability technique that provides intelligible reasons for individual predictions. Anchors are interpretable if-then rules that represent a complicated model's decision logic and provide insights into why it made a given prediction;
- *SHapley Additive exPlanations (SHAP)* - Is a machine learning interpretability technique that explains the output of complex models by assigning the contribution of each attribute to the model's prediction. It is based on cooperative game theory and gives information about the significance and impact of specific variables on model predictions. SHAP values provide interpretable and visible explanations for both local and global model behavior, making them useful in any machine learning model. By breaking down predictions into feature contributions, SHAP values quantify feature relevance, capture interaction effects, and offer consistent explanations across instances and models. Despite their computational complexity and interpretability issues, SHAP values are an effective tool for increasing trust, comprehension, and transparency in machine learning models.

2.5 Discussion

This thesis focuses mostly on unsupervised outlier detection in an environmental context. We will use our knowledge of outliers and their methods to choose three algorithms to apply to the dataset we will use in the context of data streaming, with adaptive windowing as the windowing methodology chosen for generating our results, which will be the outlier scores of each point/instance.

Based on the results, we will attempt to explain why some outlier scores have been classified as outliers using the reasoning discussed in the previous subsections. We will use importance levels, causal relationships among outliers, outlying attributes, concept drift detection and xAI approaches to understand and trust the results and output produced by outlier algorithms.

For the xAI method, we chose LIME as the best method for outlier identification in data streams, because he uses unsupervised learning to explain expected outliers and does not need a ground truth of values to make explanations. Also, the LIME explanation method will allow us to seek the explanation for individual instance. In chapter 3 we will describe how LIME works.

Chapter 3

Methodology

In this chapter, we will mostly discuss the generic application of the algorithms in data to find outliers in the context of data streams. We will explain how we can find datasets about environment, how the process works when applied to that type of data, how we use our algorithms (including the benefits and critic analysis of each algorithm), and which xAI method we used and why.

3.1 Data Sources

To locate data on environmental, we must consider that such datasets must be in the form of time series, which may be applied to data streams to detect outliers. Furthermore, since we use the *river* package [56] to apply the algorithms, the datasets require some processing, particularly the conversion to dictionary, with each row of the dataset being a dictionary. Figure 3.1 shows a few rows of an environmental dataset, while Figure 3.2 shows the result of the dictionary conversion procedure.

	Nitrogen oxide (NOx)	Sulphur dioxide (SO ₂)	Carbon monoxide (CO)	Organic carbon (OC)	NMVOCs	Black carbon (BC)	Ammonia (NH ₃)	Nitrogen oxide (NOx).1	Sulphur dioxide (SO ₂).1	Carbon monoxide (CO).1	Organic carbon (OC).1	NMVOCs.1	Black carbon (BC).1	Ammonia (NH ₃).1	Entity	Year
0	13829	112844	1278854	25527	108305	7244	26999	1	12	137	2	11	0	2	United Kingdom	1750
1	14744	135968	1322798	26417	111682	7670	28508	1	13	134	2	11	0	2	United Kingdom	1760
2	17498	196516	1415385	28519	117253	9015	30050	1	18	136	2	11	0	2	United Kingdom	1770
3	20859	149032	1522108	30987	123403	10659	31627	1	13	139	2	11	0	2	United Kingdom	1780
4	25767	193054	1673432	36655	131366	14388	33247	2	16	145	3	11	1	2	United Kingdom	1790

Figure 3.1: Original Dataset.

The final dataset's goal is to include information for each row in the form of a dictionary with the structure key and value, with each key representing the name of the column and each value representing the column's value.

```

                                Dictionary
0    {'Year': 1750, 'Ammonia (NH3)': 26999, 'Black ...
1    {'Year': 1760, 'Ammonia (NH3)': 28508, 'Black ...
2    {'Year': 1770, 'Ammonia (NH3)': 30050, 'Black ...
3    {'Year': 1780, 'Ammonia (NH3)': 31627, 'Black ...
4    {'Year': 1790, 'Ammonia (NH3)': 33247, 'Black ...

```

Figure 3.2: Dataset after the conversion to Dictionary.

With this format, and given that the datasets will be noise-free and will not require any pre-processing, it is best to use the algorithms.

We can locate this kind of dataset, which permits this kind of analysis, on a number of websites which provide datasets. *TAIAO* (Time-Evolving Data Science / Artificial Intelligence for Advanced Open Environmental Science) is a data science program led by the University of Waikato and is one of the most well-known environmental dataset websites [57]. The Universities of Waikato, Auckland, and Canterbury, Beca, and MetService have joined forces to create this program, which features top data scientists, data engineers, and environmental scientists in the globe. There are a number of pertinent datasets there that are divided down into four categories: Weather, Marine, Biodiversity and Aerial.

Another source for this type of dataset is *Waikato Region* [8], a database that contains numerous datasets relating to the environment in New Zealand. They are also tied to *TAIAO* because they use their datasets and vice versa. They have an environmental data hub, which includes the following categories: air, biodiversity, climate, coastal and marine, geothermal, land and soil, and water.

Another well-known data portal for environmental datasets is *Our World in Data* [7]. It comprises little more than 12800 datasets, each with interactive charts charting the value of the characteristics. They are divided into several topics, including population and demographic change, health, energy and the environment, food and agriculture, poverty and economic development, education and knowledge, innovation and technological change, living conditions, community and well-being, human rights and democracy, and violence and war.

Since we are taking the category Energy and Environment for this type of study, they considered the datasets with the following topics: Access to Energy, Fossil Fuels, Renewable Energy, Energy, Nuclear Energy, Plastic Pollution, Oil Spills, Lead Pollution, CO_2 and Greenhouse Gas Emissions, Climate Change, Ozone Layer, Air Pollution, Outdoor Air Pollution, Indoor Air Pollution, Clean Water and Sanitation, Water Use and Stress, Natural Disasters, Biodiversity, Environmental Impacts of Food Production, Animal Welfare, Forests and Deforestation and Land Use.

We may discover additional dataset catalogs in *Data.World* [4], a data hub with a knowledge graph design that combines metadata, processes, policies, people, and systems into a language that AI understands. The search for environmental datasets here is based on the key phrases you enter into the search field (for example, "environmental", "pollution", "oil spills"). Based on

your search terms, the website will offer datasets containing those phrases, which we can then use as desired.

Kaggle [1] is a well-known database for huge datasets. It offers over 339K high-quality public datasets, 1.1M public notebooks, and access to a powerful notebook environment with free GPUs and TPUs, as well as 5,100 pre-trained, ready-to-deploy ML models. Kaggle is ideal for applying Machine Learning (ML) to datasets. It hosts project contests where contestants from all over the world compete to find the best solution to a problem. This website contains predefined filters to help us identify environmental datasets, but it also has a search window where we can enter in the keywords we desire and get results.

There are numerous fonts where we can find this type of dataset, one being a public data repository and the other being commercial companies that retrieve data and require special authorization for us to use that information. We merely listed a few examples to demonstrate that there is a vast amount of data that can be used to apply the algorithms presented in this thesis.

3.2 Outlier Detection Approaches

Since we primarily focus on applying the algorithms to environmental data with the goal of spotting outliers and explaining why they are regarded such, we must first describe what the purpose is and the benefits of using these algorithms.

The primary goal of detecting outliers is to determine whether they are data that deviates from the usual pattern of data, indicating that the value is higher or lower than normal. This detection will determine which points are regarded outliers or not. We shall overlook the points that are not regarded outliers (normal points), as we do not need to explain why they are considered typical. Our explanation will focus on the points that are deemed outliers.

Those points will go through the explanation process, where they will check if they are "strong" outliers based on their outlier score, will visualize in 3D where the points are, to see if any of the features is an outlying attribute that will cause that point to be an outlier, and will see if those points have any causal relationship, which means they will see if removing one of the anterior outliers will affect the outliers from that point forward. If so, that point has a causal relationship.

Another type of explanation that will be verified is that when an outlier occurs is the concept drift detection. If the outlier point is discovered right on the point where a drift happens or very close to the point where a drift occurs, the risk of an outlier occurring may increase. Finally, we will discuss how LIME works for this study.

Three algorithms for unsupervised learning through data streams will be selected for this study from the pool of existing ones. The same data will be used to test those algorithms, and the results will be used to run explanation algorithms to determine why those points were chosen

as outliers.

The algorithms selected were Half-Space Trees (**HST**), Incremental Local Outlier Factor (**ILOF**) and One-Class Support-Vector Machine (**OCSVM**). We will now go over how these three algorithms function and the rationale behind our choice of them for this research.

3.2.1 Half-Space Trees

Half-Space Trees is an online version of isolation Forest (isolation Forest (**iForest**)) (represented in Figure 3.3) and is a data structure used for nearest neighbour search (Nearest-Neighbor based algorithm). It is a machine-learning approach designed to find outliers in high-dimensional data. It is especially beneficial when dealing with data that predicts outliers to occupy specific regions or subspaces of a high-dimensional space. They divide the feature space into areas and isolate outliers inside smaller partitions.

Algorithm 1 : $iForest(X, t, \psi)$

Inputs: X - input data, t - number of trees, ψ - sub-sampling size

Output: a set of t *iTrees*

- 1: **Initialize** *Forest*
- 2: set height limit $l = \text{ceiling}(\log_2 \psi)$
- 3: **for** $i = 1$ to t **do**
- 4: $X' \leftarrow \text{sample}(X, \psi)$
- 5: $\text{Forest} \leftarrow \text{Forest} \cup \text{iTree}(X', 0, l)$
- 6: **end for**
- 7: **return** *Forest*

Figure 3.3: Isolation Forest pseudo code.

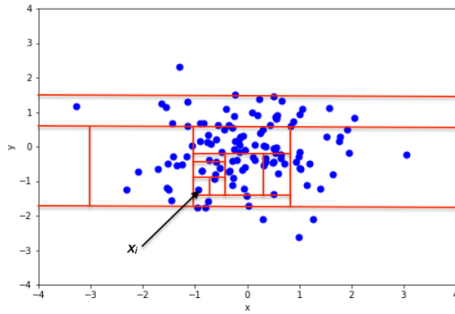
The algorithm is different from other decision tree algorithms in that only the path-length measure or approximation is used to generate the outlier score; no leaf node statistics on class distribution or target value are needed.

The Half-Space Tree technique assumes that anomalous data points are easier to distinguish from the rest of the sample.

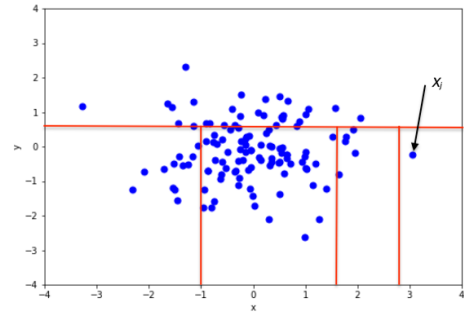
To find outliers, the algorithm first divides the data into a feature space using randomly generated hyperplanes. The procedure then recursively adds the planes, splits the window in question, and partitions it until the number of points in each partition meets an established minimum threshold. Each additional plane becomes a node in the tree. The depth of a point in the tree determines whether it is an outlier or exists in smaller partitions.

Figures 3.4a and 3.4b show examples of random partitioning in a 2D dataset of normally distributed points, respectively, for a non-anomalous point and a point more likely to be an outlier. The pictures show that outliers require fewer random partitions to be isolated than

typical spots. The path length of point x_i in Figure 3.4a is greater than that of x_j in Figure 3.4b.



(a) Isolating a non-anomalous point.



(b) Isolating an anomalous point.

Figure 3.4: Examples of isolating different type of points in a 2D Gaussian distribution [5].

The outlier score is calculated for each instance. If the score is close to one, the instance is very likely to be an outlier; if it is less than 0.5, the instance is likely to be a normal value. If, for a given window, all instances are assigned an outlier score of around 0.5, it is safe to assume that the sample does not contain any outliers.

There are certain advantages of using this type of approach for stream mining environmental data, which we shall experiment with. One of the advantages is increased efficiency. **HST** can be faster than brute-force search, especially in high-dimensional spaces. Another one is scalability. It can handle high-dimensional data better than other approaches, such as k-d trees, which lose performance as the dimensions increase.

3.2.2 Incremental Local Outlier Factor

The Incremental Local Outlier Factor (ILOF) algorithm extends the original LOF technique (represented in Figure 3.5), which is used to discover anomalies in data streams. It is meant to efficiently update outlier scores as new data points are added or current points are removed, eliminating the need to recompute everything from the beginning [60].

The ILOF is based on the notion of local density, in which locality is defined as the distance between k-nearest neighbours, which is used to estimate density. Comparing an object's local density to the local densities of its neighbours allows one to find regions of similar density and spots with much lower densities than their neighbours. These are considered outliers (Figure 3.6a).

Unlike the traditional Local Outlier Factor (LOF) technique, which needs a complete recalculation of outlier scores every time the dataset changes, ILOF progressively updates only the relevant components. This makes ILOF ideal for real-time applications and scenarios that require streaming data [60].

When a new data point is introduced, ILOF incorporates it into the existing data structure, such as a k-d tree or ball tree, to efficiently manage and retrieve k-nearest neighbours. The method

Algorithm LOF (Local Outlier Factor) Computation

Require: $k, \mathcal{D}$.
Ensure: L_k - LOF score for each object in $\mathcal{D}$

```

1:  $L_k = \emptyset$ .
2: for  $p \in \mathcal{D}$  do
3:    $N_k(p) = NULL$ 
4:   for  $q \in \mathcal{D}$  do
5:     if  $|N_k(p)| < k$  then
6:       Add  $q$  in  $N_k(p)$ 
7:     else
8:       Let  $s^* \in N_k(p)$  be such that  $dist(p, s^*) \geq dist(p, s)$  for all  $s \in N_k(p)$ ;
9:       if  $dist(p, s^*) > dist(p, q)$  then
10:        Replace  $s^* \in N_k(p)$  by  $q$ 
11:       end if
12:     end if
13:   end for
14:    $d_k(p) = \max\{dist(p, s) | s \in N_k(p)\}$ 
15: end for
16: for  $p \in \mathcal{D}$  do
17:   for  $q \in \mathcal{D}$  do
18:      $d_{reach}(p, q) = \max\{d_k(p), d(p, q)\}$ 
19:   end for
20: end for
21: for  $p \in \mathcal{D}$  do
22:    $l_k(p) = \frac{|N_k(p)|}{\sum_{q \in N_k(p)} d_{reach}(p, q)}$ 
23: end for
24: for  $p \in \mathcal{D}$  do
25:    $L_k(p) = \lceil \frac{\sum_{o \in N_k(p)} l_k(o)}{|N_k(p)|} \rceil$ 
26: end for
27: return  $L_k$ 

```

Figure 3.5: LOF pseudo code.

then updates the new point's k-nearest neighbours and any existing points whose neighbour connections may be changed by the addition. This ensures that the neighbourhood relationships remain accurate without recalculating them across the entire dataset.

Next, ILOF recalculates the k-distances for the new location and any affected neighbours. The k-distance represents the distance to the kth nearest neighbour, and this step ensures that the distance between each point and its neighbours follows the most recent dataset configuration. Following this, the algorithm updates the reachability distances, which determine how accessible a place is to its neighbours. These distances account for the neighbour's maximum k-distance and distance between the points (Figure 3.6b).

ILOF recalculates the Local Reachability Density (**LRD**) for the new location and any affected neighbours using the updated reachability distances. The LRD is effectively the inverse of the average reachability distance between the k-nearest neighbours and the site in question. It measures how densely the point is surrounded by its neighbours. Finally, ILOF updates these points' Local Outlier Factor (LOF) values. The LOF score is calculated as the average ratio of the point's LRD to that of its k-nearest neighbours. A higher LOF score implies a greater possibility of the point being an outlier. If the LOF score is nearly one, it means that it has a similar density to its neighbours; if the LOF score is less than one, it means that it has a higher density than its neighbours (inliner); and if the LOF score is greater than one, it means that it has a lower density than its neighbours (outlier) [60].

When a data point is eliminated, the procedure is roughly the same. The approach initially removes the point from the data structure before updating the k-nearest neighbours for the

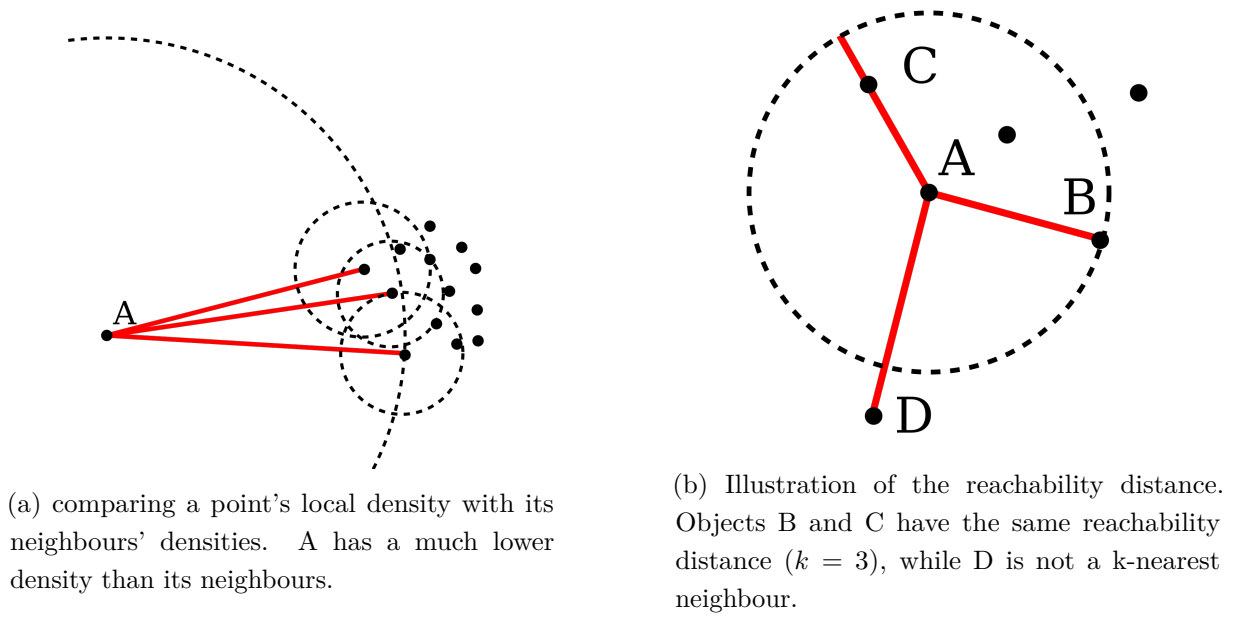


Figure 3.6: Basic idea of ILOF [6].

points that were neighbours to the deleted point. It then calculates the affected k -distances, reachability distances, LRDs, and LOF scores. ILOF focuses only on the points that are affected by adding or removing one point. ILOF maintains computational efficiency by focusing only on the points that are affected by the addition or removal of one point.

The effectiveness of ILOF stems from its ability to update only the relevant areas of the dataset, eliminating the need for massive re-computations. This makes it ideal for applications that demand constant monitoring and quick outlier identification, such as network security, fraud detection, and sensor networks. Furthermore, ILOF's capacity to adapt to dynamic datasets ensures that it remains accurate and dependable even if underlying data patterns change over time. This provides a very good algorithm to test in environmental datasets applied to data streams, because in environmental case studies, there is usually a lot of fluctuation between values, making the distance and reachability calculations ideal for detecting outliers.

3.2.3 One-Class Support-Vector Machine

We can use **OCSVM** to achieve outlier detection [85]. In one-class classification, only one target class of data is available because others are too difficult to characterize or expensive to acquire. One-class SVM learns an optimal decision boundary enclosing the normal instances, and the rest of the data outside the boundary are naturally identified as outliers. This idea makes OCSVM suitable for outlier detection (Figure 3.7).

It is an algorithm designed specifically for outlier identification. This approach is very useful when you have a dataset comprised primarily of one class and want to discover data points that vary greatly from that class. This approach is especially useful in fields such as fraud detection,

Algorithm 1: Federated OCSVM training process

Result: Trained $OCSVM_i$ model by each participant F_i
Define initial parameters: $(w_i^{(0)}, \rho_i^{(0)})$, $epoch=0$; num_synch, N ;
while The loss function does not converge **do**
 $synch_list = \text{random_ind}(N, num_synch)$;
 foreach F_i in parallel **do**
 NyströmTransform(T_i^{TR});
 Train($OCSVM_i^{epoch}, T_i^{TR}$);
 if F_i in $synch_list$ **then**
 F_i sends $(\Delta w_i^{epoch}, \Delta \rho_i^{epoch})$ to the server;
 Server aggregates the parameter updates and broadcasts them;
 $(\Delta w_{FD}^{epoch}, \Delta \rho_{FD}^{epoch}) = \text{FedAggregation}(\Delta w_i^{epoch}, \Delta \rho_i^{epoch}, F_i \in synch_list)$;
 foreach F_i in parallel **do**
 F_i receives parameter updates from the server and updates its model;
 $OCSVM_i^{epoch+1} = \text{Update}(OCSVM_i^{epoch}, \Delta w_{FD}^{epoch}, \Delta \rho_{FD}^{epoch})$;
 $epoch = epoch + 1$;

Figure 3.7: OCSVM pseudo code.

network security, manufacturing, and healthcare.

The basic principle behind a one-class SVM is to model the distribution of the normal class and then determine whether incoming data points fall within it or not. During the training phase, the algorithm is given data containing only examples from the normal class. It aims to define a boundary around these data points, usually as a hypersphere or hyperplane in the feature space that contains the vast bulk of the data.

The approach works by identifying a decision function that distinguishes the typical class from possible outliers. It tackles an optimization problem similar to that of classic SVMs, focusing on distinguishing the normal class from all others. The decision function learned during training increases the margin around the typical data points while allowing a few points outside the border, referred to as outliers [85].

Mathematically, the goal of the one-class optimization issue is to minimize the weight vector's norm while restricting the proportion of data points outside the decision border. Slack variables and a parameter v balance the trade-off between margin maximization and outlier tolerance.

When classifying new data points, the algorithm determines if they are within the established border. Points that fall inside are considered part of the regular class, while those outside are classified as oddities. The OCSVM's flexibility and performance are determined by the kernel function, which shapes the border, and hyperparameters v and γ (for specific kernels), which alter the algorithm's sensitivity to outliers and the impact of individual training points. Figure 3.8 will illustrate the classification results on a 2D training dataset by the OCSVM respectively in the original space (left side) and in the kernel feature space (right side) [85].

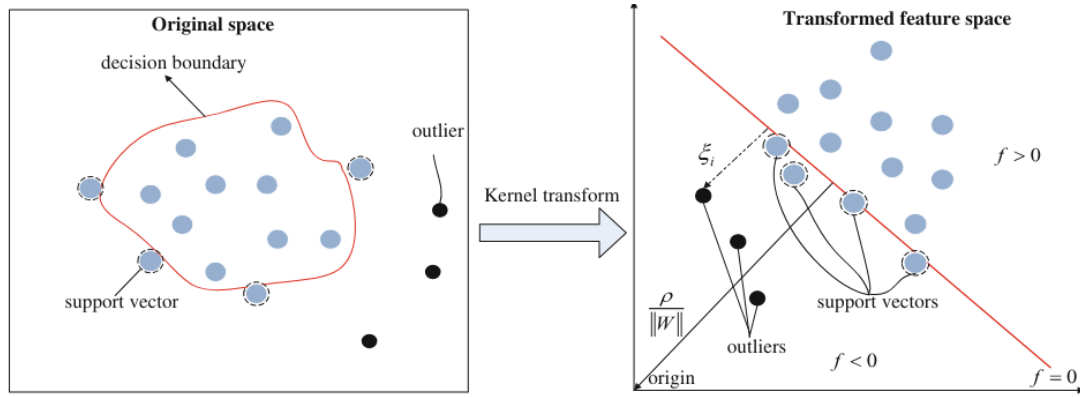


Figure 3.8: One-class SVM classification to a 2D training dataset [85].

The OCSVM classifier learns a boundary that encloses the blue normal data class. The black points lying outside the boundary are considered outliers. The blue points with black edges are support vectors which fall on the boundary [85].

Non-zero slack variables ζ_i allow outlying points to fall outside the decision boundary. The penalty factor $1/N_v$ in the optimization model (Figure 3.8) indicates the degree of trade-off between maximum margin and least classification error. In one-class SVM, the penalty factor for normal and outlying data is the same. In most datasets, there are certainly fewer outliers than normal points. Normal data generates significantly more error than outliers do. This means the normal data points are subjected to a higher error penalty, shifting the decision boundary to the outliers. The diagram illustrating the influences of outliers on the OCSVM is shown in Figure 3.9. Two outlier points are introduced to the same training dataset in Figure 3.8. The decision boundary changes to the outliers, and the principal outliers become support vectors. Outliers significantly affect the decision boundary location and classification accuracy.

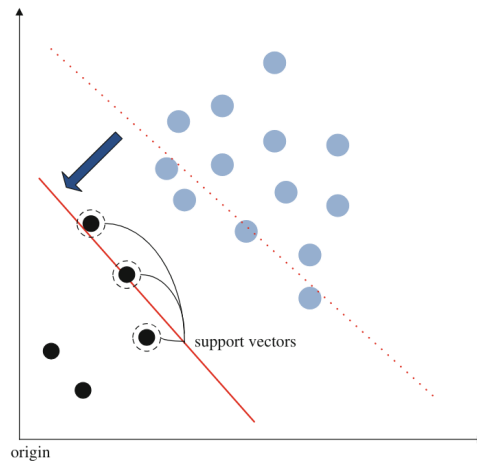


Figure 3.9: Influences of outliers to the decision boundary [85].

One-class SVMs are useful in cases with high-dimensional data or where only the normal class is well-represented. However, they are sensitive to kernel and hyperparameter selection, demand significant processing resources for large datasets, and may perform poorly if the normal class is

not sufficiently represented or if there is a significant overlap with the outlier class. They can be extremely useful in detecting environmental, which frequently require recognizing abnormal activity within massive databases of environmental data.

3.2.4 Local Interpretable Model-Agnostic Explanations (LIME)

According to Chapter 2, there are explanation methods that help us understand why an outlier was deemed as an outlier, enumerating various types of explanations, strategies that convey the importance levels of those outliers, the causal interaction that may exist between the outliers detected, the outlying attributes of outliers that refers mainly to the feature subspace where an outlier divides from the normal data, and the evaluation methods.

As previously stated in chapter 2, we have Global Model-Agnostic Methods and Local Model-Agnostic Methods which will have many xAI approaches to explain outliers. The benefits of using a model-agnostic explanation system (xAI methods) include model flexibility (the interpretation method can work with any ML model), explanation flexibility (not limited to a specific type of explanation), and representation flexibility (the explanation system can use a different feature representation than the model being explained) [64].

As described above, LIME explains black-box model predictions (Figure 3.10) at the local level and approximate the predictions. Instead of training a global surrogate model, LIME will focus on training local surrogate models to explain individual predictions.

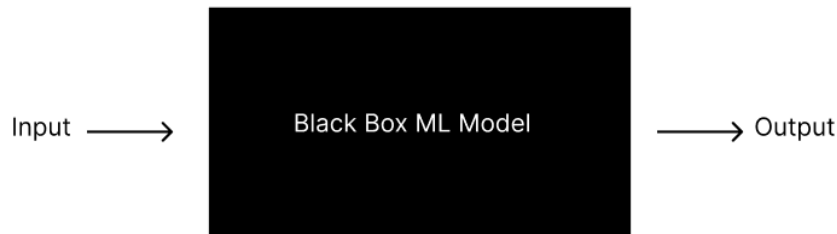


Figure 3.10: Black-Box model prediction logic process.

The goal is to understand why the ML model made a specific prediction. LIME investigates what happens to predictions when you feed variations of your data into a machine learning model. LIME creates a new dataset that includes altered samples as well as the black box model's predictions. This new dataset is then used to train an interpretable model, which is weighted by the sampled instances' closeness to the instance of interest. The interpretable model could be anything from the interpretable models chapter, such as Lasso or a decision tree. The learnt model should be a good local approximation of the machine learning model predictions, but not necessarily a good global approximation. This type of accuracy is also known as local fidelity [64].

Mathematically, local surrogate models with interpretability constraints are expressed as follows:

$$\text{explanation}(x) = \arg \min_{g \in G} L(f, g, \pi_x) + \Omega(g) \quad (3.1)$$

The explanation model for instance x is the model g (e.g. linear regression) that minimizes loss L (e.g. mean squared error), which measures how close the explanation is to the prediction of the original model f (e.g. an xgboost model), while keeping the model complexity $\Omega(g)$ low (e.g. prefer fewer features). G represents the family of possible explanations, such as all linear regression models. The proximity measure π_x determines the size of the neighborhood around instance x that we consider in the explanation. In practice, LIME just optimizes the loss function. The user must set the complexity, for as by selecting the maximum amount of features that the linear regression model can employ [64].

The steps for training local surrogate models:

- Choose the case of interest for which you want an explanation of its black box prediction;
- Modify your dataset to acquire the black box predictions for these new points;
- Weigh new samples based on their proximity to the occurrence of interest;
- Train a weighted, interpretable model on the dataset with variations;
- Explain the prediction by analyzing the local model.

To use LIME in tabular data, suppose that each row is an instance and each column is a feature. LIME samples are obtained from the training data's mass center, which is problematic. However, it increases the possibility that the predictions for some of the sample points will deviate from the data point of interest, allowing LIME to discover at least one explanation. Figure 3.11 shows an example of utilizing the LIME method on tabular data. The figure A) shows the Random Forest predictions for the features x_1 and x_2 (predict classes: 1 is dark, 0 is light). Image B) depicts an incident of interest (large dot) and data drawn from a normal distribution (little dots). Image C) gives more weight to points near the instance of interest, and image D) shows the signs of the grid that illustrates the classification of the locally learnt model from the weighted samples, with the white lines denoting the density boundary ($P(\text{class} = 1) = 0.5$) [64].

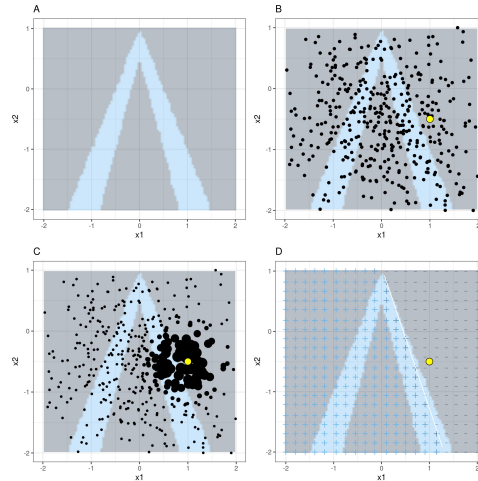


Figure 3.11: Example of a LIME algorithm for tabular data [64].

Python's *lime* [62] package has an implementation of this algorithm. This is precisely the package we will use, together with our algorithms applied to environmental data, to explain individual predictions and assess the findings in this thesis. We will now describe how this package works.

The steps of implementation are as follows: First, we need to load the data and train the model of our choice (the example we will talk about uses the *Titanic* classification dataset and the LightGBM algorithm to train the model). After training the model, we generate the explainable AI method. We must consider which type represents the feature. If the feature is numerical, we calculate the mean and standard deviation before partitioning it into quartiles. If the characteristic is categorical, we calculate the frequency for each value. In our research, we will use numerical features. We use these computed statistics to scale the data, allowing us to compute distances when the attributes are not on the same scale, as well as to select perturbed instances, which we accomplish by sampling from a $\text{Normal}(0,1)$, multiplying by the standard deviation, and adding back the mean [63, 70]. Lastly, we ask to generate an explanation for a single instance. An example of an explanation for one instance is in Figure 3.12.

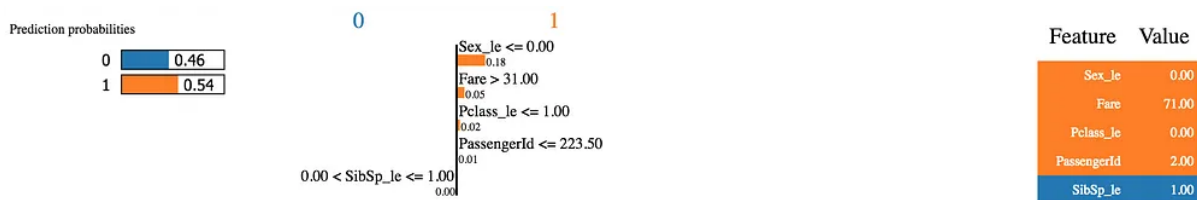


Figure 3.12: Example of a LIME explanation for a single instance [70].

Figure 3.12 shows a three-part explanation. The left-most section gives prediction probabilities, while the middle area returns the five most relevant attributes. The binary classification challenge would be in two colors (orange and blue). Attributes in orange support class 1, while those in blue support class 0. $\text{Sex_le} \leq 0$ supports class 1. Float point numbers on the horizontal bars

indicate the relative importance of various traits, and the color-coding is consistent between sections. It contains the actual values for the top five variables.

Chapter 4

Experimental Study

In this chapter, we will go over the work we did in detail, explaining the source of the data we used to test the algorithms, the algorithms results, and the LIME explanation method results.

One of the key goals of this research is to show that we can use outlier detection algorithms to a data stream and use outlier explanations to explain why the outlier points detected were discovered in this way. This type of study has few proposals, which is why we wish to go deeper into the behavior that algorithms may display.

4.1 Experimental Setup

4.1.1 Data

The dataset we used for this thesis was extracted from the *Our World in Data* website [3] and it talks about the average annual emissions of air pollutants across the world between 1750 and 2022. We chose this dataset because, first and foremost, the thesis is about environment, and air pollution fits perfectly with that issue. Second, this dataset is the one with the biggest amount of data, which means it covers the most range of years and is quite detailed regarding the values of the gases. In Figure 4.1 is a chart that represents the distribution of gases in the world on average.

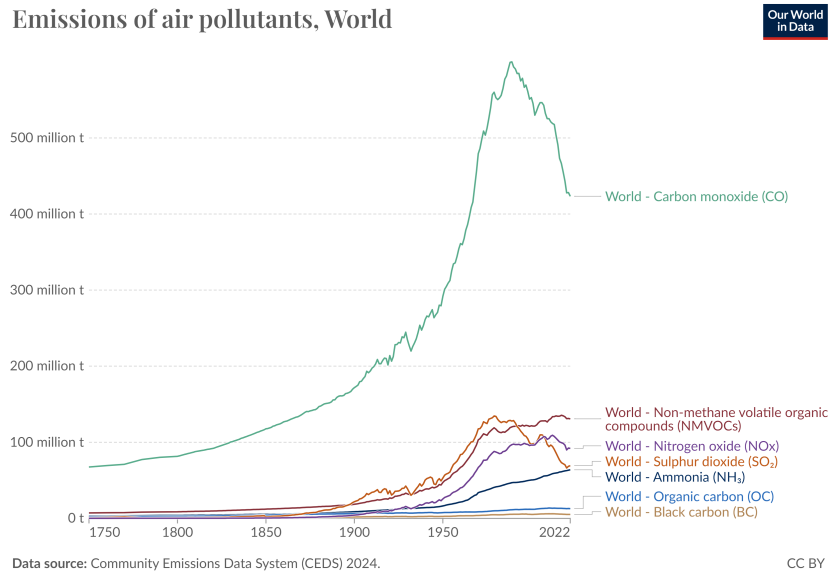


Figure 4.1: Average of air pollutants in the world between 1750 and 2022. [3]

The features that describes this dataset are the following:

Features	Description
Year	Year
SO_2	Sulphur Dioxide
CO	Carbon Monoxide
OC	Organic Carbon
$NMVOCs$	Non-methane Volatile Organic Compounds
BC	Black Carbon
NH_3	Ammonia
NOx	Nitrogen oxide

Table 4.1: Features that describes the dataset

Except for the *Year*, every feature have decimal values, which might be positive or negative.

This dataset contains records from all of the world's countries, however we chose only one. The country to be chosen must have a range of data values that allow our algorithms to quickly discover outliers. So the United Kingdom had the best considered ideals. The following Figure 4.2 represents a chart of air pollutants only from the United Kingdom.

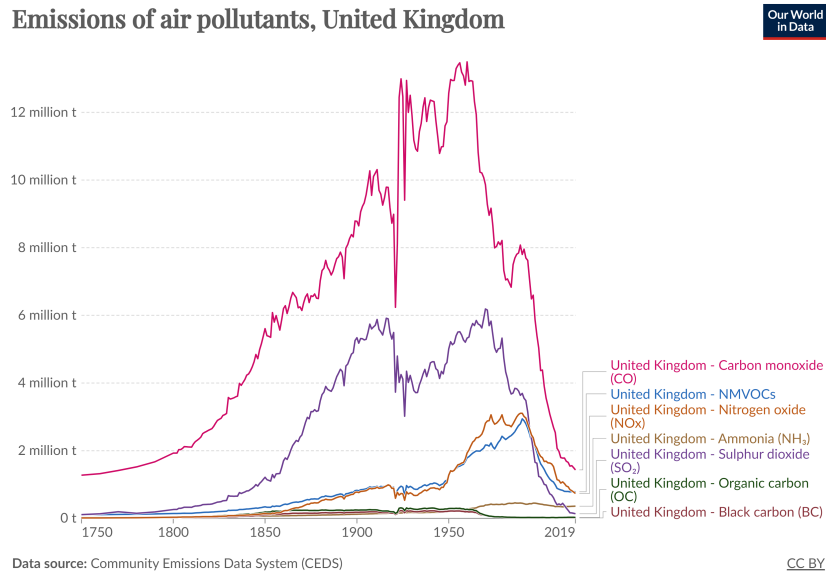


Figure 4.2: Annual average of seven air pollutants between 1750 and 2022 in the United Kingdom. [3]

The graphic above shows that carbon monoxide and sulphur dioxide readings vary more than the other properties over time. We'll observe in our implementation if these features have an impact on outlier identification and explanations.

4.1.2 Algorithms

We chose three algorithms to build and test to determine if they find the same outliers or if there is a discrepancy in detection. The algorithms are based from the package *river* and are the Half-Space Trees (**HST**), the Incremental Local Outlier Factor (**ILOF**) and One-Class Support-Vector Machine (**OCSVM**). We will go over how those algorithms operate and how the application gets done below.

We use some criteria that applies to the algorithms to help the detection and the explanation of outliers. Those criteria include processing the data to further use to run the algorithm to improve the precision of detection, applying a formula to the outlier score to detect the stronger outliers among a group of them (Threshold) and using a concept drift detector to show us where a concept drift was detected and if that has an impact on the explanation of the outliers.

Threshold Application - Our algorithms produce the outlier score for each point he predicts, which is insufficient to determine whether the points are outliers or normal. The study [84] provides numerous threshold-defining procedures to identify outlier scores. In this study, the threshold strategy we used assumes that the data is normally distributed and suggests outlier levels. In general, this technique calculates the threshold as follows:

$$thr_{min} = \mu - a * \sigma \quad (4.1)$$

$$thr_{max} = \mu + a * \sigma \quad (4.2)$$

where μ and σ are the mean and the standard deviation of the outlier scores, and a is a user-defined control parameter. The lower the value, the more objects will be tagged as outliers. Most cases uses $a = 3$ since outliers are assumed to be minor. In our situation, we adapt the control parameter to each of our algorithms, selecting the parameter that best suits each algorithm.

Concept Drift Detection - Refers to recognizing changes in a data stream's statistical characteristics over time, which will impact how well predictive models function. It involves recognizing when the relationships or patterns in the data that a model depends on have changed. In order to keep models current and functional when the underlying data changes, it is essential to identify concept drift. When a time window is used, the learner is only induced from the instances that are contained in the window at each time step. The main challenge here is figuring out what size window to use. Since we want the window in our models to change over time, we used the ADaptive WINdowing (**ADWIN**) algorithm, which will change our window and identify drifts as they occur [30].

The hyperparameters used to initialize the ADWIN drift detector are:

- ADWIN does not perform any change detection until a number of data points have arrived (*grace_period* = 1);
- The maximum number of buckets of each size that ADWIN should keep before merging buckets (*max_buckets* = 30);
- The minimum length allowed for a sub-window when checking for a concept drift that if is a lower value it may decrease delay in change detection but may also lead to more false positives (*min_window_length* = 1);
- How often ADWIN should check for changes (*clock* = 1);
- Significance value (*delta* = 0.5).

The setup for each algorithm that we will use will first pass by a preprocessing stage, which it will be applied a *MinMax Scaler* or a *Standard Scaler* method, and then the hyperparameters used will be the following:

- Half-Space Trees:
 - Number of trees (*n_trees* = 7);
 - Height of the tree (*height* = 5);

- Size of the window where it will compute a prediction ($window_size = 5$);
- The number used to initialize the algorithm ($seed = 75$);
- Constant to calculate the threshold of the outlier score ($constant = 3.2$).
- Incremental Local Outlier Factor:
 - The number of nearest neighbors to use for density estimation ($n_neighbors = 10$);
 - Constant to calculate the threshold of the outlier score ($constant = 3.2$).
- OCSVM:
 - An upper bound on the fraction of training errors and a lower bound of the fraction of support vectors ($nu = 0.1$);
 - The sequential optimizer used for updating the weights ($optimizer = optim.SGD(.2)$);
 - Constant to calculate the threshold of the outlier score ($constant = 2.3$).

4.2 Results

4.2.1 Half-Space Trees

The first outlier identification result shows that it was projected that three years (1921, 2003 and 2004) would be classified as outliers, with outlier scores of 0.738, 0.749 and 0.842 respectively, and a maximum threshold computed to be 0.71. Figure 4.3 shows the outlier scores over time.

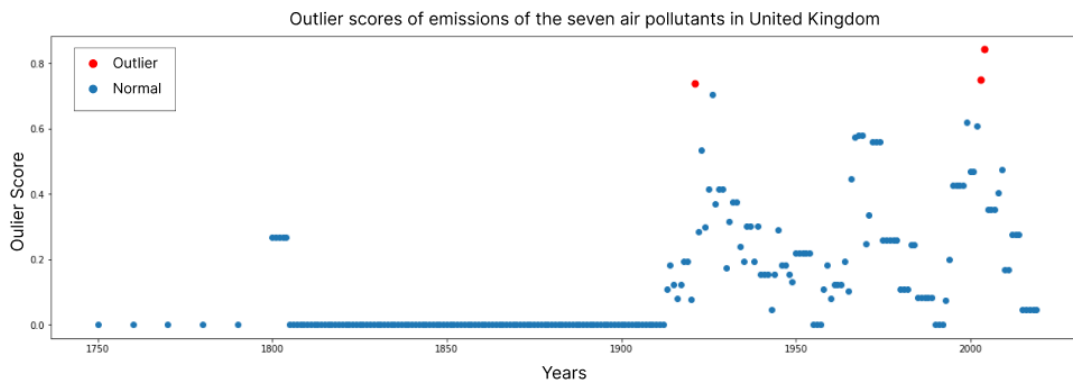


Figure 4.3: Outlier scores using the HST algorithm.

After identifying the outliers, we can go on to the explanation phase. The first step of the explanation process is the *numerical ranking of outliers*, which will choose which outliers to prioritize first. This stage has already been completed because the outliers identified were those with the highest outlier scores. After determining which points to evaluate, we proceeded to try to explain the outliers by their feature subspace, where we will look to see if the outliers are significantly deviated from the deviated data, attempting to determine whether an attribute

contributes to the abnormality of the outliers. In Chapter 2, we covered ways to create 3D plots with three specific attributes and then view them as 2D plots. To choose the attributes, we first performed a pairplot to determine the distribution of the data. The resulting plot is depicted in Figure A.1 in Appendix A. From there, we chose the attributes of *Sulphur Dioxide* (SO_2), *Organic Carbon* (OC), and *Black Carbon* (BC) based on their distribution. After selecting those attributes, we proceeded to create the 3D and 2D plots. Figure 4.4 illustrates a plot applying those variables to identify outlying attributes for the outliers.

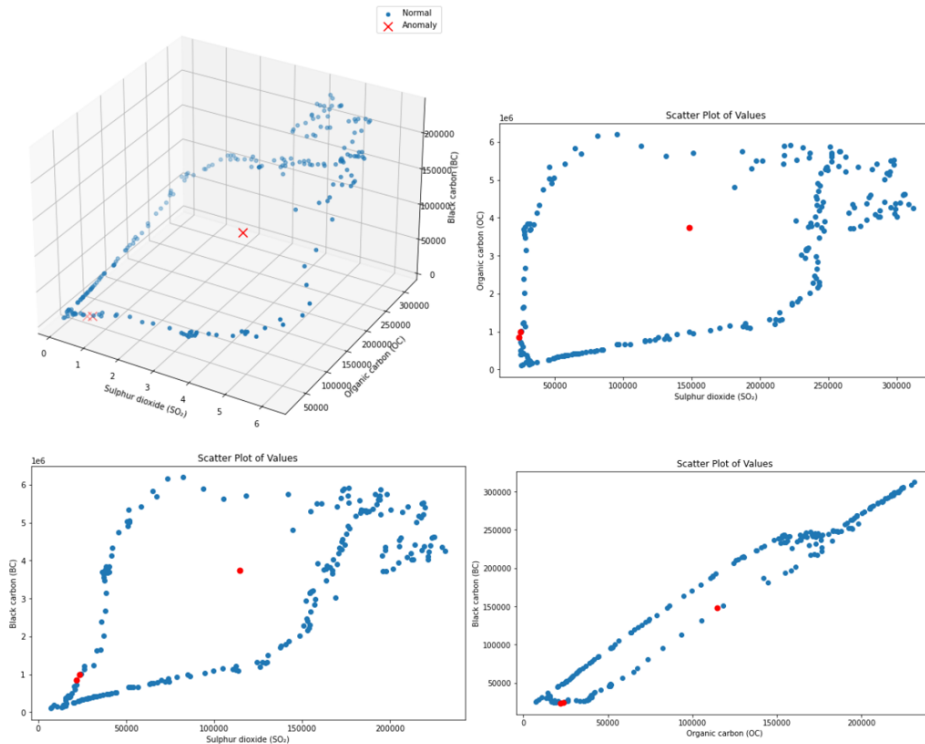


Figure 4.4: Outlying attributes of an individual outlier using HST algorithm.

The 3D plot shows that we have an isolated outlier and others that is not so isolated, indicating that there is an attribute that can influence this outlier, therefore we evaluate the 2D plots to see if this case confirms it.

By looking at the three 2D plots show that *Sulphur Dioxide* is the only feature that contributes to this outlier being classified as such. It is validated because when we use this attribute, the isolated outlier always emerge, and when we make the 2D plot with the other two attributes, the outlier is not isolated, therefore we can confirm that *Sulphur Dioxide* is an outlying attribute.

Knowing that an attribute can effectively contribute to a point being considered a outlier we then proceeded to find more explanations using the concept drift detection. To apply this method, as explained earlier we used **ADWIN** with the hyperparameter setup. The drift detection is showed by the yellow vertical bars in Figure 4.5.

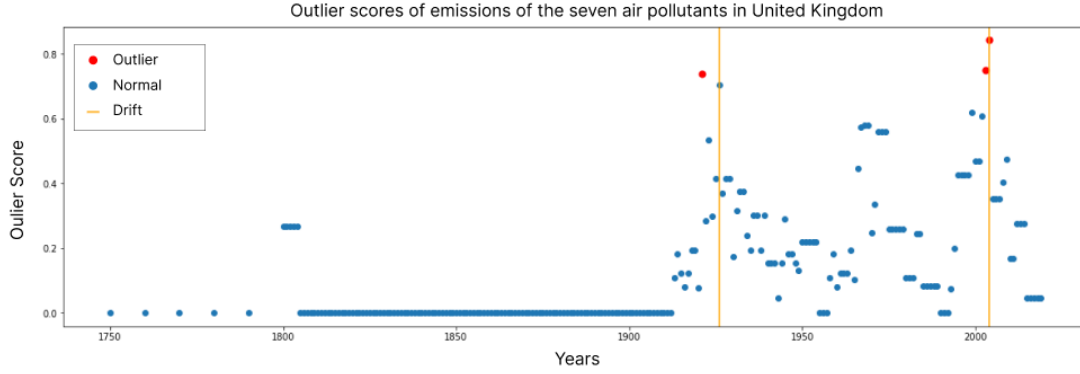


Figure 4.5: Concept drift detection using ADWIN with HST algorithm.

Analyzing the identified drift results, we can determine that only two drifts occurred in 1926 and 2004. This drift occurred when an outlier was found or was very close to a previously detected outlier. Using Euclidean distance to verify the distance from the outlier to the closer drift, we can see that the outlier detected in 1921 is closest to the drift of 1926, with a euclidean distance of 5.0. The other two years (2003 and 2004) have the drift of 2004 as the closest, with euclidean distances of 1.0 and 0.0, respectively.

This leads us to conclude that when a drift happens over time, there is a strong possibility of an outlier occurring at the same time or very close to the drift.

Next, we looked for any causal interactions that might exist in our model. Figure 4.4 shows one single outlier (year 1921) and two non-isolated outliers (years 2003 and 2004). Since the isolated outlier was explained, we proceeded to try to explain the other two outliers by attempting to find a causal relationship, which means that the outlier that appeared before these two could be influencing why these two appear; in other words, we will test whether or not these points can still be considered outliers if the outlier prior to the two we are explaining is removed.

Some approaches to defining the interaction include using tree processes, but because HST is a tree-based algorithm, this interaction is already taken into consideration. We removed the outlier point with the year 1921, ran the algorithm again with the same hyperparameters, and checked to see if the outliers remained, disappeared, or generated another outlier. The Figure 4.6 shows the result with the respective concept drift detection.

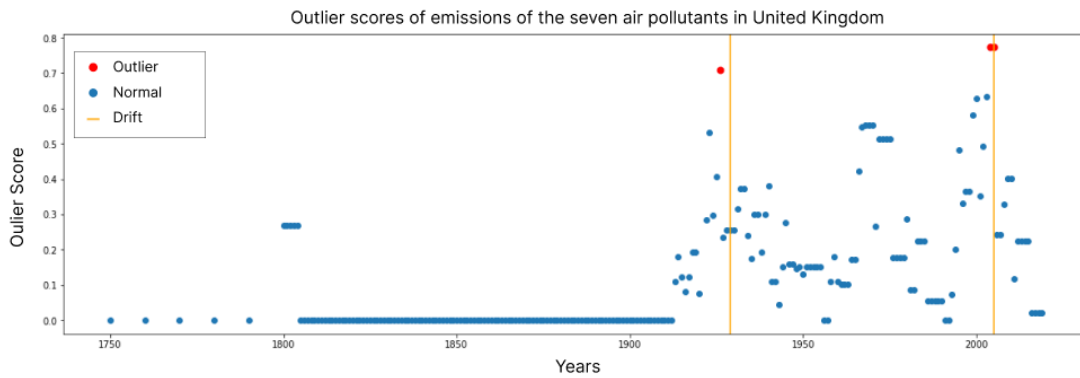


Figure 4.6: Iteration of HST algorithm excluding year 1921.

As shown in the above illustration, removing the point linked with the year 1921 removed the outlier point for the year 2003, concluding that the year 1921 will have a causal relationship with year 2003. In contrast, it produced two new outliers, one with the year 1926 and another with the year 2005. The concept drift detected will reach the same conclusion as verified above, with one drift detected in year 1929 with the closest outlier being year 1926 with a euclidean distance of 3.0, and another drift detected in year 2005 with the outliers being year 2004 and 2005 with euclidean distances of 1.0 and 0.0, respectively.

Finally we will proceed with the LIME implementation of the explanation for an individual instance.

LIME Explanation

Using LIME to explain a single instance, we will use just one outlier instance (year 2004) to generate explanations. In Figure 4.7 it represents the generated explanation for year 2004.

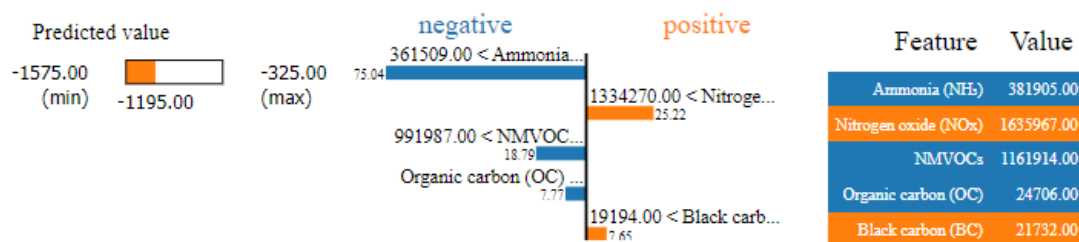


Figure 4.7: LIME explanation for instance year 2004.

Looking at the generated explanation we can see that the left-most section will give the probability of the outlier score for that instance we are trying to explain, given us a predicted value of -1195.0 from the range of -1575.0 to -325.0, being the predicted value being classified as an outlier (most negative is considered more a stronger outlier). The middle area returns the top five most relevant attributes that we defined, being the horizontal bars the importance of each attribute, being the attribute importance that influence positively is the *Nitrogen oxide (NO_x)* with 25.22 and *Black Carbon (BC)* with 7.65.

Negatively influencing attributes include *Ammonia* (NH_3) with 75.04, *Non-methane Volatile Organic Compounds* (NMVOCs) with 18.79, and *Organic Carbon* (OC) with 7.77. The rightmost portion described the actual values of the top five qualities. As we can see, NH_3 and NMVOCs will have the highest and second lowest values, respectively. Concluding by merely checking the feature importance, we can see that NH_3 will be the most influential attribute in this instance considered an outlier, with the other two positive attributes attempting to balance this negative characteristic.

Figure 4.8 will represent both the middle and right-most section of the local explanation into one plot for better understanding.

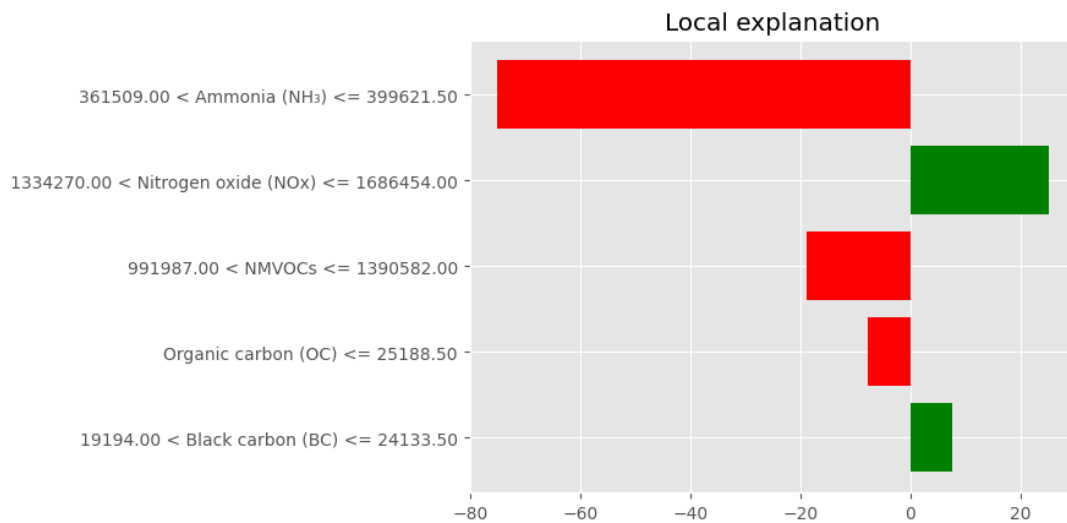


Figure 4.8: Local explanation for instance year 2004.

This process also give some more output values that will explained with detail what is interpreted with this explanation.

The *Intercept* value is -988.009. This is the baseline value for the locally linear approximation generated by LIME. Simply said, it is the predicted outcome when all feature contributions are at their average or baseline amounts.

The *Prediction_local* value is -1056.728, which is the LIME-trained local surrogate model's forecast. For the HST algorithm, this value is an approximation of the outlier score for the given instance. More negative scores usually suggest that the incident is more of an outlier. The particular interpretation of the scores varies depending on the implementation, but a large negative result indicates that the instance is a strong outlier.

The *Right* value is -1195.0, which is the outlier score assigned by the original, global HST model to the identical instance. This score reflects how much of an outlier the global model thinks this instance to be, with larger negative values indicating a greater level of outlier.

The *Explanation Local Prediction* repeats the local surrogate model's prediction of -1056.728, indicating that this number was utilized for local interpretation. Given the extremely low score,

this implies that the case is seen as a significant outlier locally.

The *Explanation Global Prediction* restates the global model's forecast of -1195.0, which corresponds to the previous *Right* value. This extremely negative score suggests that the global model regards the case as a serious outlier.

In conclusion, the global HST model predicted an outlier score of -1195.0, suggesting that the event is deemed a substantial outlier. The local surrogate model, which approximates the behavior of the HST model, predicted a score of -1056.728, indicating that the instance remains an outlier but is less extreme in the local context. The intercept of -988.009 is used as the baseline for the local model's approximation. The near but differing values between the local and global projections suggest that the occurrence is regarded as outlier both locally and globally, with the global model rating it as a stronger outlier than the local model.

4.2.2 Incremental Local Outlier Factor

The first outlier identification result shows that it was projected that four years (1830, 1921, 1951 and 1952) would be classified as outliers, with outlier scores of 3.258, 3.213, 3.382 and 3.082 respectively, and a maximum threshold computed to be 2.91. Figure 4.9 will visualize the outlier scores over time.

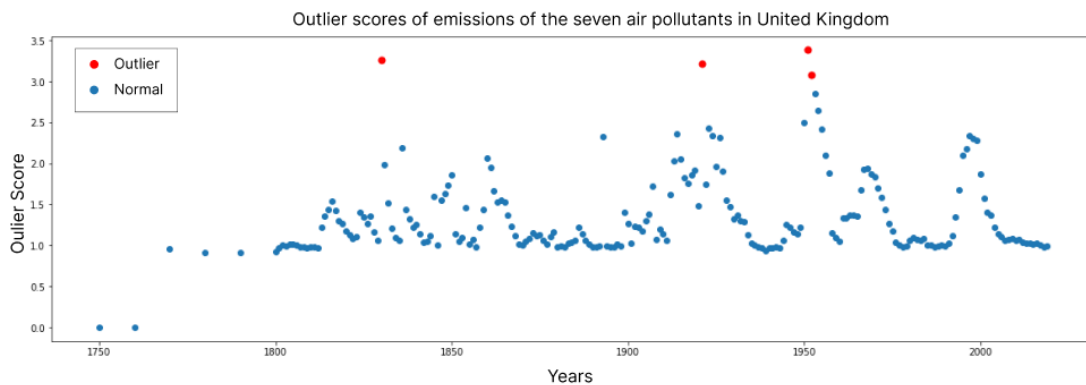


Figure 4.9: Outlier scores using the ILOF algorithm.

After identifying the outliers, we will to the same steps as we did in the HST algorithm. The *numerical ranking of outliers* is already checked, being this four outliers the points we should look and explain because of their high outlier score. After then we proceed to the *Outlying attribute* step, where we determine whether an attribute contributes to the abnormality of the outliers. To choose the attributes, we did a pairplot to determine the distribution of the data. The appendix A.2 shows the pairplot results.

From the pairplot in A.2, we also chose the same attributes as we did in the HST algorithm (*Sulphur Dioxide SO₂*, *Organic Carbon OC*, and *Black Carbon BC*) based on their distribution. Then we proceeded to create the 3D and 2D plots. Figure 4.10 illustrates a plot applying those variables to identify outlying attributes for the outliers.

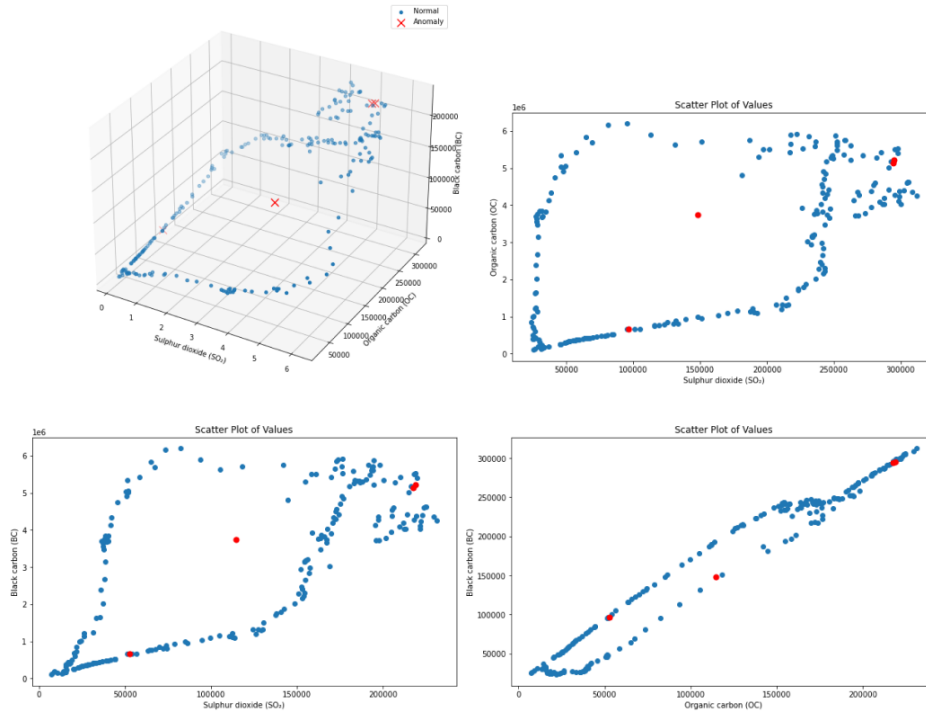


Figure 4.10: Outlying attributes of an individual outlier using ILOF algorithm.

The 3D plot shows that we have an isolated outlier and three others that are not so isolated, indicating that there is an attribute that can influence this outlier, therefore we evaluate the 2D plots to see if this case confirms it.

By looking at the three 2D plots, we can confirm that this case is very similar to the outlier we explained like this in the HST algorithm. The *Sulphur Dioxide* SO_2 is the only feature that contributes to this outlier being classified as such, because from the plots, when we use this attribute, the isolated outlier always emerge, therefore the *Sulphur Dioxide* SO_2 is also an outlying attribute in the ILOF algorithm.

Next step, following the explanation phases as the HST algorithm, we proceed to evaluate using the concept drift detection. We also used **ADWIN** with the hyperparameter setup specified in the beginning of the chapter. The drift detection is showed by the yellow vertical bars in Figure 4.11.

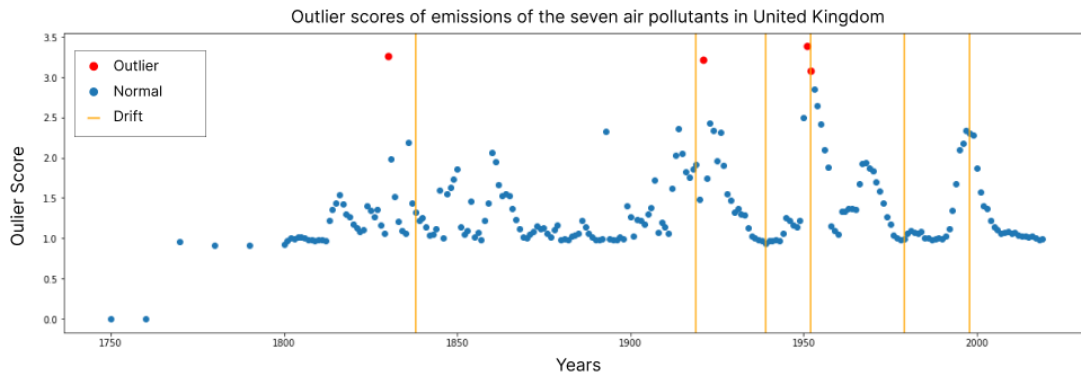


Figure 4.11: Concept drift detection using ADWIN with ILOF algorithm.

Analyzing the identified drift results, we can determine that six drifts occurred in 1838, 1919, 1939, 1952, 1979 and 1998. In this case the drift detection is a little different from what we got in the HST algorithm. Three out of the six drift detected show us that a outlier was found or was very close to a predicted outlier. This drift without a detected outlier can be influenced by the outlier score itself, as we can see from the outlier score in general that there is seasonality over time and that the variation of points increases with time. It can also be influenced by the number of neighbors that the algorithm receives each time it runs. Because we only run with 10 points at a time, utilizing the ADWIN method with that number of points allows us to detect drift without outliers. So we can see that where there is an outlier, there might be a drift, but a drift does not always result in an outlier being detected.

Using Euclidean distance to verify the distance from the outliers to the closer drift, we can see that the outlier found in 1830 is closest to the drift of 1838, with a euclidean distance of 8.0 (this is the greatest distance we know so far between an outlier and a drift). The outlier discovered in 1921 is the closest to the 1919 drift, with a euclidean distance of 2.0. The other two years, 1951 and 1952, had the closest drifts, with euclidean distances of 1.0 and 0.0, respectively.

This leads us to the conclusion that when a drift occurs over time, there is a high probability that an outlier will occur at the same time or very close to the drift, not considering seasonality.

Next, we look for causal interactions between outliers. Figure 4.10 shows a single outlier (year 1921) and three non-isolated outliers (years 1830, 1951 and 1952). The method we will use to find the relationship is identical to what we used with the HST algorithm. In this situation, we performed two tests to determine the causal relationship between outliers (removing year 1830 or removing year 1921), to see if the points could still be deemed outliers if the outlier preceding the one we are explaining was deleted.

Because ILOF is a density-based algorithm, this interaction is already taken into account. The first test we performed was to remove the outlier point with the year 1830, then run the algorithm again with the same hyperparameters and check to see if the outliers remained, disappeared, or generated another outlier. Figure 4.12 displays the results of idea drift detection.

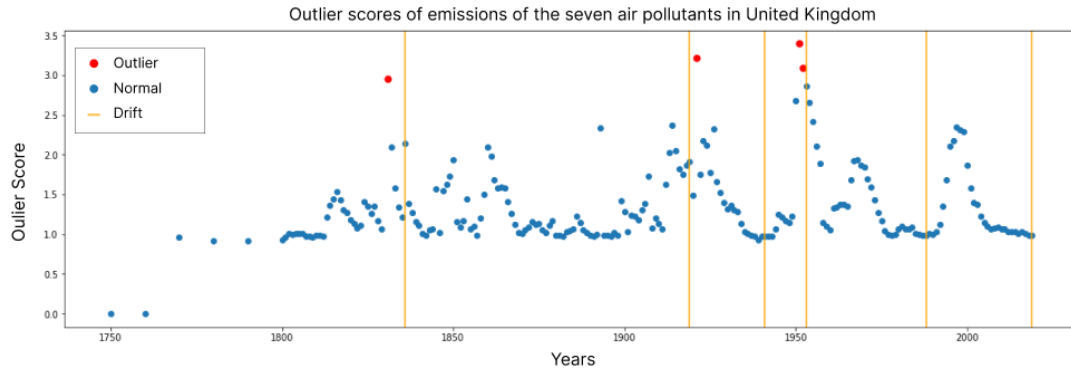


Figure 4.12: Iteration of ILOF algorithm excluding year 1830.

As illustrated in the image above, eliminating the point associated with the year 1830 would not remove any of the initial outliers, but instead add another outlier point for the year 1831, the year after the one removed. The first detected drift will also shift, now occurring in 1836 and changing the euclidean distance from 8.0 to 5.0. This suggests that when we remove the outlier point (year 1831 ahead), the outlier gets closer and closer to the drift, validating the probability theory we discussed before.

Because we didn't get any significant findings with these points, we decided to run another test with another outlier point (year 1921) to see if these points have a causal relationship with the other points. Using the same approach as before, Figure 4.13 illustrates the result of idea drift detection.

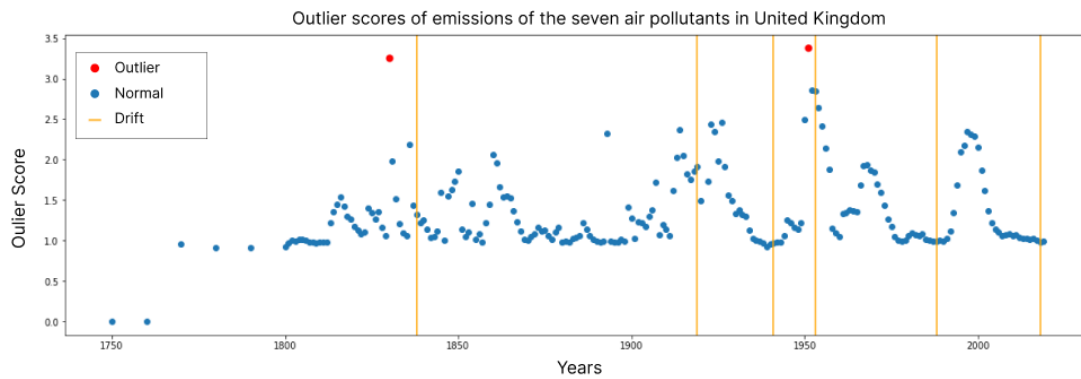


Figure 4.13: Iteration of ILOF algorithm excluding year 1921.

The Figure shows that there will be a relationship between the years 1921 and 1952, because when we remove the year 1921 and run the algorithm again, the year 1952 is no longer considered an outlier and no new outliers appear, leaving only the outliers 1830 and 1951, implying that the year 1921 will have a causal relationship with the year 1952. Since we removed the year 1921, the drift detected from that year onward will also be different, having now drift detected in the years 1838, 1919, 1941, 1953, 1988, and 2018, finding the same conclusion that we had earlier, with the outlier found in 1830 having the closest drift at 1838 with a euclidean distance of 8.0 and year 1951 having the closest drift at 1953 with a euclidean distance of 2.0.

Finally we will proceed with the LIME implementation of the explanation for an individual instance.

LIME Explanation

Proceeding to the LIME explanation for a single instance, we will use just one instance to generate explanations for one of the outliers that our algorithm considered as such. In Figure 4.14 it is the generated explanation for the instance we wanted to generated explanations (year 1951).

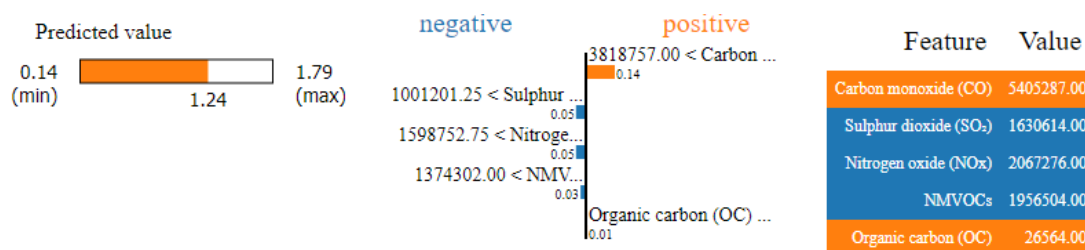


Figure 4.14: LIME explanation for instance year 1951.

Looking at the generated explanation we can see that the left-most section will give the probability of the outlier score for that instance we are trying to explain, given us a predicted value of 1.24 from the range of 0.14 to 1.79, being the predicted value being classified as a outlier (more positive is considered a stronger outlier). The middle area returns the top five most relevant attributes that we defined, being the horizontal bars the importance of each attribute, being the attributes importance that influence positively are the *Carbon Monoxide (CO)* with 0.14 and *Organic Carbon (OC)* with 0.01.

The attributes importance that influence negatively are the *Sulphur Dioxide (SO₂)* with 0.05, *Nitrogen Oxide (NO_x)* with 0.05 and *Non-methane Volatile Organic Compounds (NMVOCs)* with 0.03. The right-most section described the actual values of those top five attributes. As we can see *CO* and *OC* will have the highest and lowest values respectively. Concluding only verifying the feature importance, we can notice that *CO* will be a attribute that will influence this instance to be considered a outlier, having the other three attributes to try to counter the positive attribute.

Figure 4.15 will represent both the middle and right-most section of the local explanation into one plot for better understanding.

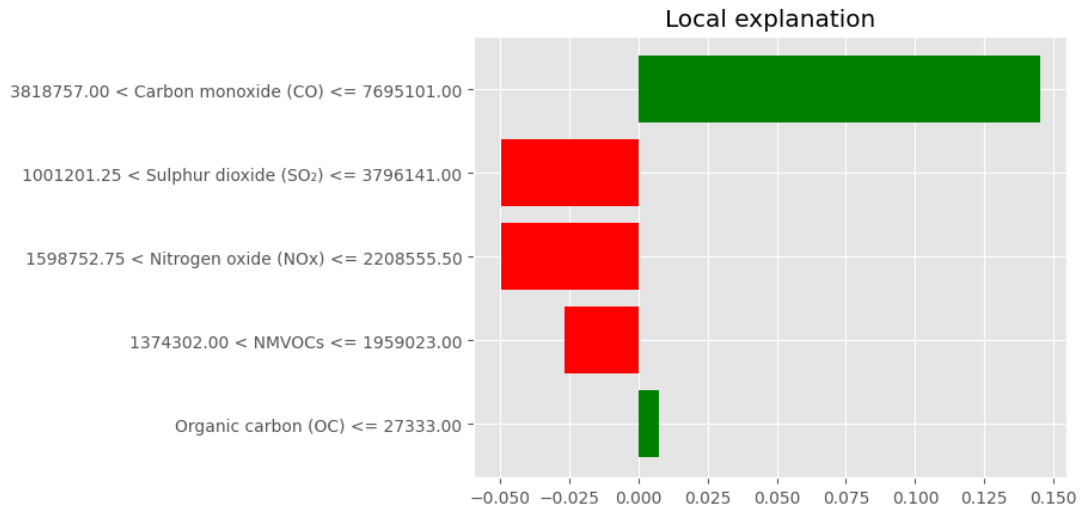


Figure 4.15: Local explanation for instance year 1951.

This process also give some more values that we will explain and interpret the results.

The *Intercept* value is 0.929. This is the baseline value for the locally linear approximation generated by LIME. Simply said, it is the predicted outcome when all feature contributions are at their average or baseline amounts.

The *Prediction_{local}* value of 0.956 represents the prediction produced by the LIME-trained local surrogate algorithm. For an ILOF algorithm, this value approximates the outlier score for the given occurrence. Typically, a score significantly less than 1 indicates that the occurrence is normal, whereas values greater than or close to 1 indicate that the case may be a outlier.

The *Right* value of 1.239 is the outlier score awarded by the original, global ILOF algorithm for the same instance. This score indicates how much of an outlier the global algorithm believes this instance is, with higher values indicating a greater degree of outlier.

The *Explanation Local Prediction* repeats the local surrogate algorithm's prediction of 0.956, indicating that this value was used for local interpretation. Given that the score is near to one, it indicates that the instance is considered a fairly outlier locally.

The *Explanation Global forecast* restates the global algorithm's forecast of 1.239, which matches the previous *Right* value. This higher number suggests that the global algorithm regards the case as somewhat of an outlier.

In conclusion, the global ILOF algorithm projected an outlier score of 1.239, indicating that the case is something of an outlier. The local surrogate algorithm, which approximates the behavior of the ILOF algorithm, projected a score of 0.956, indicating that the instance is pretty normal in its local context. The intercept of 0.929 is used as the baseline for the local algorithm's approximation. The close approximation between the local and global projections suggests that the local surrogate algorithm provides a reasonable explanation of the global ILOF algorithm's choice to classify the event as an outlier.

4.2.3 One-Class Support-Vector Machine

The first outlier identification result shows that it was projected that thirteen years (1760, 1801, 1802, 1803, 1804, 1805, 1806, 1807, 1808, 1809, 1810, 1811, and 1812) would be classified as outliers, with outlier scores of 6.798, 6.323, 7.234, 6.85, 7.086, 7.014, 6.839, 6.939, 6.956, 6.963, 6.966, 6.667, and 6.362 respectively, and a maximum threshold computed to be 6.255. Figure 4.16 illustrates the outlier scores over time.

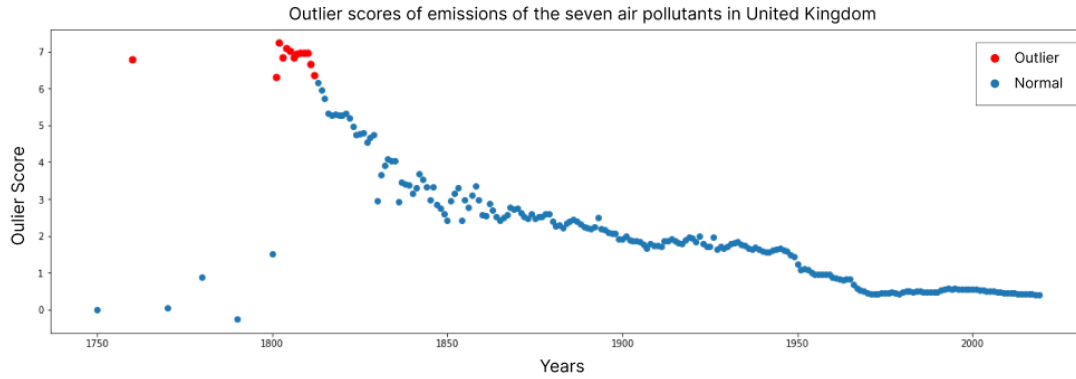


Figure 4.16: Outlier scores using the OCSVM algorithm.

After identifying the outliers, we will follow the identical processes as in the HST and ILOF algorithms. The *numerical ranking of outliers* is already validated above. We should look and explain the points of these thirteen outliers because their outlier score exceeds the maximum threshold defined. We proceed to the *Outlying attribute* stage to examine whether an attribute contributes to the abnormality of the outliers. To choose the qualities, we used a pairplot to determine the distribution of the data. Appendix A.3 displays the pairplot results.

The pairplot in A.3 shows that there is no isolated outlier, unlike the other algorithms where there was at least one isolated outlier. Additionally, there is a group of twelve outliers (years 1801 to 1812) that are not present in the other algorithms, making this case unique to try to explain. To ensure consistency, we will use the same features for both the 3D and 2D plots (*Sulphur Dioxide SO₂*, *Organic Carbon OC*, and *Black Carbon BC*). Figure 4.17 shows a visualization using these criteria to determine outlying attributes for the outliers.

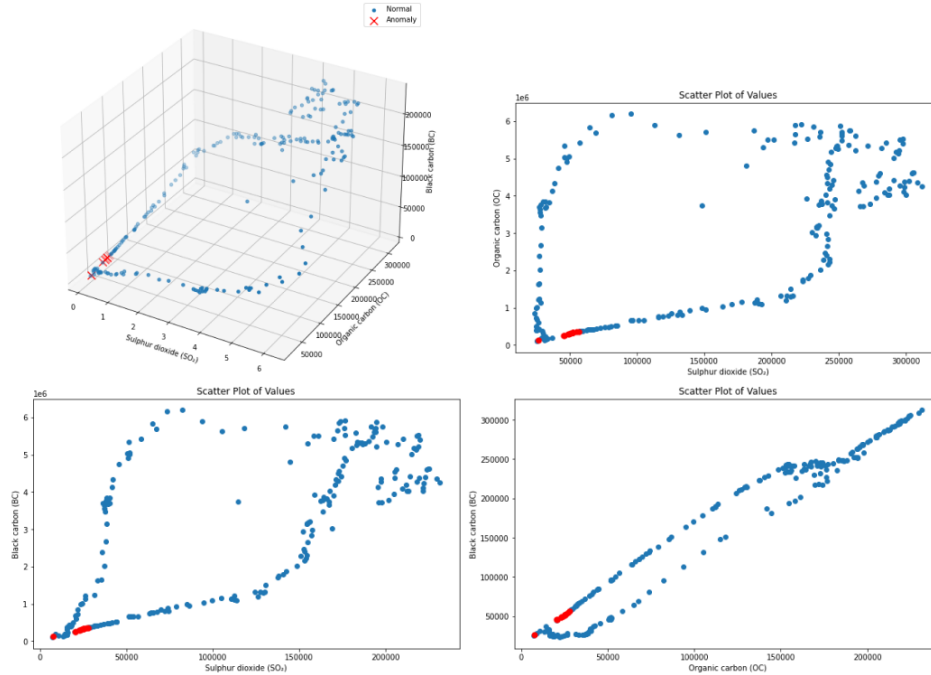


Figure 4.17: Outlying attributes of an individual outlier using OCSVM algorithm.

The 3D plot shows that there are no individual outliers when compared to the graphs from the HST and ILOF algorithms. It was also indicated before that there is a group of outliers (twelve in total). According to this study [59], the outlying attributes of a collection of outliers are the subset of attributes that cause the abnormality of each outlier in the group. This is the feature subspace where the outliers diverge from normal data. Because those outliers are sequential and similar in terms of outlier score and attribute values, they will be grouped together as an outlier cluster. Because neither of the features in A.3 is clearly deviated from the typical data, neither of the features will be regarded an outlying attribute, unlike in the other two methods where *Sulphur Dioxide* SO_2 was considered an outlier.

Following this explanation, we will analyze using concept drift detection. We used ADWIN with the hyperparameters supplied at the beginning of the chapter. Figure 4.18 depicts the detection of drift using yellow vertical bars.

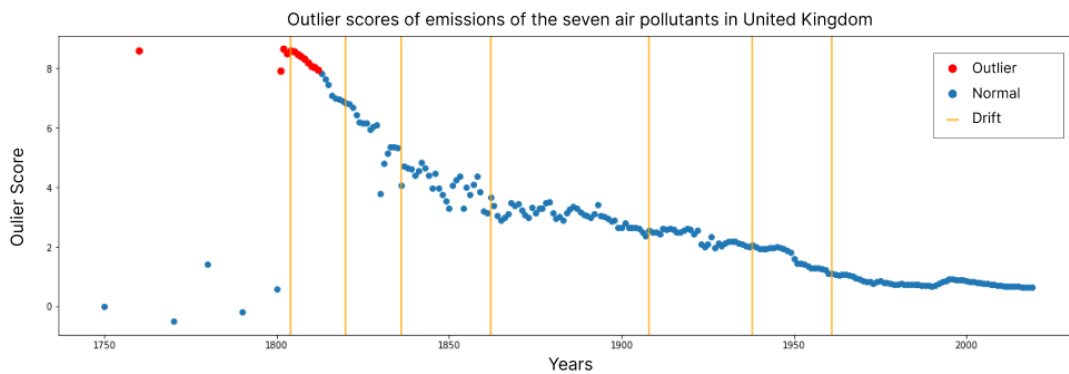


Figure 4.18: Concept drift detection using ADWIN with OCSVM algorithm.

Figure 4.18 shows that the implementation of concept drift detection slightly changed the outlier scores of the previous result in Figure 4.16 (will increase the outlier score for every point in the illustration) and the maximum threshold will change from 6.255 to 7.898. The outliers will remain the same, but the outlier scores from year 1760 to year 1812 will be 8.634, 7.92, 8.602, 8.482, 8.587, 8.553, 8.473, 8.402, 8.299, 8.186, 8.062, 8.046, and 7.938. Just looking at the outlier scores, we can see that the ranking of evaluating outliers will vary when the concept drift is implemented; previously, the highest outlier score was in the year 1802, but now it is in year 1760.

Analyzing the detected drift results, we may conclude that seven drifts happened in 1804, 1820, 1836, 1862, 1908, 1938, and 1961. In this situation, the drift detection differs slightly from the HST method but is similar to the ILOF algorithm's results. Two of the seven discovered drifts indicate the presence of an outlier or a near match to a predicted outlier. This drift without a recognized outlier can be influenced by the outlier score itself, as we can see from the outlier score in general that the score has decreased over time, showing that a slight change in the forecast could cause a drift. It can also be influenced by the upper and lower bounds on the fraction of training errors and support vectors that we specified in the hyperparameters; using the ADWIN technique with those bounds selected allows us to detect drift without outliers. As a result, the premise provided in the idea drift explanation for the ILOF algorithm applies here as well.

Using Euclidean distance to verify the distance between the outliers and the closer drift, we can observe that the outlier detected in 1760 is closest to the drift of 1804, with a euclidean distance of 44.0. In this example, because the outlier occurs before the drift, we cannot explain it using this type of argument. The outliers detected between years 1801 and 1812 are the most closely related to the 1804 drift, with euclidean distances of 3.0, 2.0, 1.0, 0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, and 8.0. The twelve outliers that are relatively close to a drift confirm what we previously concluded: when a drift occurs over time, there is a high probability that a group of outliers or a point outlier will occur at the same time or very close to the drift, regardless of seasonality or other variations.

Next, we investigate the causal interaction between outliers. Figure 4.17 does not show any isolated outlier, however Figure 4.18 clearly shows year 1760 as an isolated outlier from the previously found outliers. The way we will use to find the relationship is the same as what we used with the HST and ILOF algorithms. Because OCSVM is a boundary-based algorithm, this interaction has already been taken into consideration. We will simply execute one test, deleting the year 1760 and running the algorithm again to see if there is any outlier from the set of outliers that will be eliminated, as well as whether this outcome will generate new outliers. In the Figure 4.19 displays the results of idea drift detection.

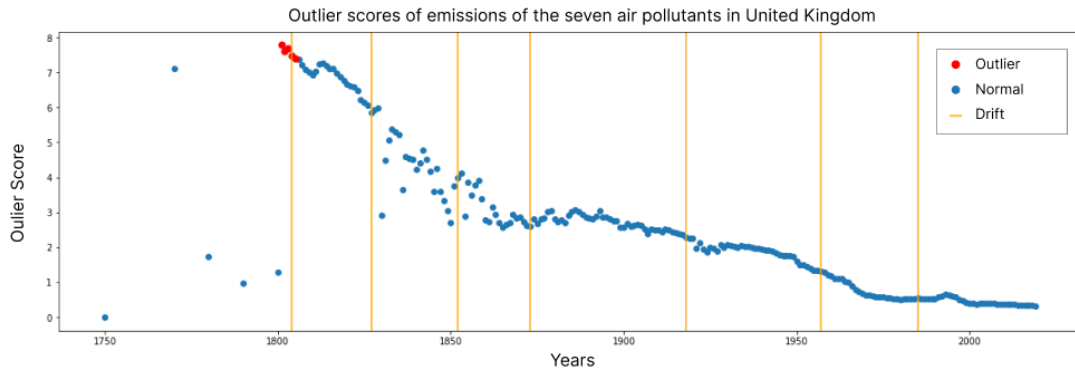


Figure 4.19: Iteration of OCSVM algorithm excluding year 1760.

Figure 4.19 shows that there will be a relationship between the years 1760, 1806, 1807, 1808, 1809, 1810, 1811 and 1812. When we remove the year 1760 and run the algorithm again, the years 1806 to 1812 will no longer be considered outliers and no new outliers will appear as an outcome of this removal, leaving only the outliers between years 1801 and 1805, implying that the year 1760 will have a causal relationship with the years 1806 to 1812. Since we removed the year 1760, the drift detected from that year onward will be different, with drift detected in the years 1804, 1827, 1852, 1873, 1918, 1957, and 1985, yielding the same conclusion as before, with the outliers remaining in the group of outliers found in years 1801 to 1805 having the closest drift at 1804 with euclidean distances of 3.0, 2.0, 1.0, 0.0, and 1.0, respectively.

Next, we will construct the explanation in LIME for a single instance.

LIME Explanation

Continuing the method employed in the previous algorithms, we are going to generate an explanation for just one instance, in this case, the outlier in the year 1801. Figure 4.20 shows the generated explanation for the instance for the specified year.

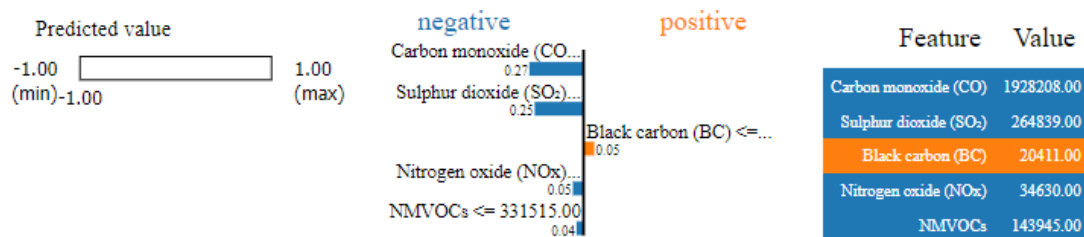


Figure 4.20: LIME explanation for instance year 1801.

Looking at the generated explanation we can see that the left-most section will give the probability of the outlier score for that instance we are trying to explain, given us a predicted value of -1.0 from the range from -1.0 to 1.0, being the predicted value being classified as a outlier (-1.0 is considered a outlier). The middle area returns the top five most relevant attributes that we defined, being the horizontal bars the importance of each attribute, being the attribute

importance that influence positively is the *Black Carbon (BC)* with 0.05.

The attributes importance that influence negatively are the *Carbon Monoxide (CO)* with 0.27, *Sulphur Dioxide (SO₂)* with 0.25, *Nitrogen Oxide (NO_x)* with 0.05 and *Non-methane Volatile Organic Compounds (NMVOCs)* with 0.04. The right-most section described the actual values of those top five attributes. As we can see *CO* and *SO₂* will have the highest values. Concluding only verifying the feature importance, we can notice that *CO* and *SO₂* will be the attributes that will influence the most this instance to be considered a outlier, having the other attribute (*BC*) to try to counter the negative attribute.

Figure 4.21 will represent both the middle and right-most section of the local explanation into one plot for better understanding.

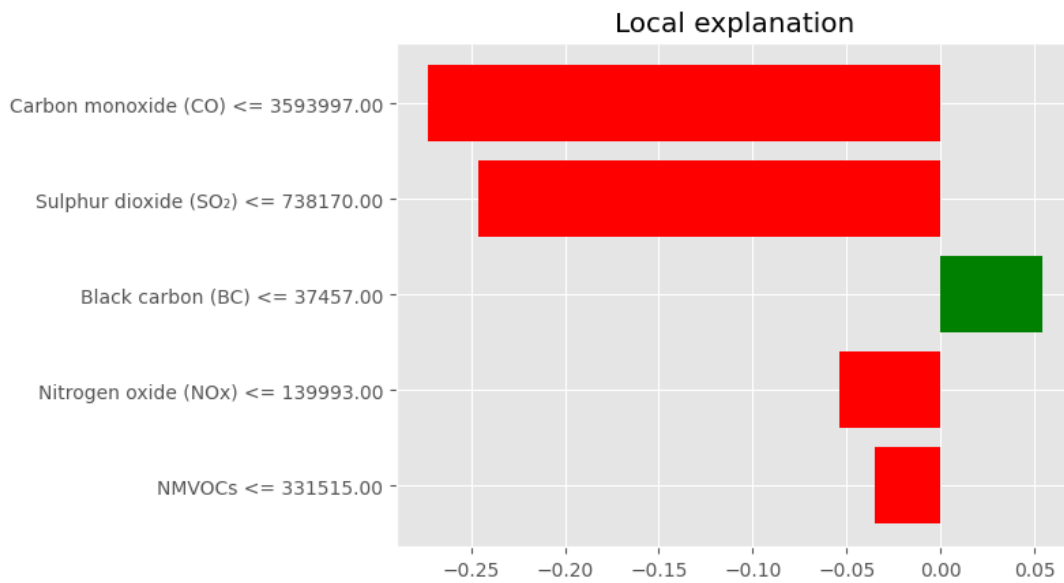


Figure 4.21: Local explanation for instance year 1801.

This process also give us complementary values that we will explain and interpret the results.

The *Intercept* value is 0.937. This is the baseline value for the locally linear approximation generated by LIME. Simply said, it is the predicted outcome when all feature contributions are at their average or baseline amounts.

The *Prediction_local* value of 0.382 is the prediction generated by LIME's trained local surrogate model. For the One-Class SVM algorithm, this value is an estimate of the decision function for the given instance. A higher result usually implies that the event is normal, whereas a lower value shows that it could be an outlier.

The *Right* value is -1, which corresponds to the decision function value assigned by the initial, global One-Class SVM model for this instance. This negative score implies that the global model regards the case as an outlier.

The *Explanation Local Prediction* reiterates the local surrogate model's prediction of 0.382,

indicating that it was used for local interpretation. This lower score indicates that the situation is regarded as outlier locally.

The *Explanation Global Prediction* restates the global model's forecast of -1, which corresponds to the earlier *Right* value. This negative value shows that the global model regards the instance as an outlier.

In summary, the global One-Class SVM model predicted a decision function value of -1, suggesting that the instance is classified as an outlier. The local surrogate model, which approximates the behavior of the One-Class SVM model, projected a lower positive value of 0.382, indicating that the instance is a outlier in the local context. The intercept of 0.937 is used as the baseline for the local model's approximation. The close approximation between the local and global projections suggests that the local surrogate algorithm provides a reasonable explanation of the global One-Class SVM algorithm's choice to classify the event as an outlier.

Chapter 5

Conclusion

The primary goal of this dissertation was to demonstrate that we can use outlier detection algorithms to data streams and use explanation methods to try to explain why the identified outliers were classified as such based on their outlier score being more than the stated threshold. To explain different types of outliers, we used three distinct algorithms previously implemented in the *river* [56] package: Half-Space Trees (**HST**), Incremental Local Outlier Factor (**ILOF**), and One-Class Support-Vector Machine (**OCSVM**). The use of these techniques, combined with the dataset we chose that focused on the environment, and the threshold formula with the constant used, proved to be useful in identifying "strong" outliers. We used a preprocessing phase to prepare the dataset for use with the algorithms discussed above. These methods are based on datasets with a dictionary in each row, which is the only way we can implement using the *river* framework.

The step of explaining the cause of the outliers being detected as such was proved using the theoretic methods of explanation, which will investigate the cause using the individual outlier and the group of outliers, together with the xAI method (Local Interpretable Model-Agnostic Explanations (**LIME**)), which will generate explanations for an individual instance.

All results were effective in detecting outliers with the computation of the threshold adapted to each algorithm, as well as successfully explaining the source of each outlier and the accompanying methodologies that proved it. The LIME explanation approach was also effective for determining which attributes are more likely to lead to an instance being classified as an outlier, as well as whether that outlier is classified as such on a local and global level.

The contributions developed for this dissertation include implementing and applying theoretic assumptions of explanations to a real-world example, namely the importance levels, outlying attributes, and causal interactions. We also performed a new interpretation based on the concept of drift detection, in which we verified that when a drift occurs, an outlier is typically identified at the exact time or slightly before or after the drift occurs, making it a critical factor in attempting to identify outliers in the context of data streaming. This conclusion is a significant step for this type of issue and study, as it attempts to bridge a gap in the field of data streaming and

environmental causes that are affecting our globe even more than they previously did.

Despite the great results, there were several constraints that impacted the flow of our implementation. One of them is that we utilize the *river* framework to apply the algorithms, which employs a dataframe of dictionaries. It took us a bit of time to process the data and make it ready to use in the explanations. Also, the explanation component was challenging since we had so little information about how to apply the theory and discover alternate ways to explain circumstances when what we had was not proven. The most difficult forms of outliers that we sought to explain were the group of outliers and the first individual outlier that we detected using the OCSVM algorithm. The individual outlier had no detectable drift since it occurred much earlier, and the only way we could explain it was to use the causal interaction method. The group of outliers were difficult to explain because in the study that talked about the explanation methods [59], there was little to no information about how to explain this case, but with the use of causal interaction and concept drift detection, we found a way to explain it. Finally, we encountered some difficulty using the LIME technique in the *river* framework since LIME only uses algorithms that are a *sklearn* object, whereas we have algorithms in the *river* object. We attempted to integrate a new implementation into the existing one in the *river* framework to convert the *river* object to a *sklearn* object for usage with the LIME technique. Unfortunately, this implementation only successfully generated explanations for the **ILOF** algorithm but not for the other two, so we decided to scrap it and use the algorithms in *sklearn* that were very similar to the ones we used to generate explanations for individual instances, in this case, our outliers.

Finally, using a mix of investigations, this dissertation was able to demonstrate an important point in the data stream community, prompting improvement/future work on finding new explanation methods to try to explain a group of outliers, as it lacks theory methods compared to those used to explain individual outliers. Another area of future research that should be pursued is finding more ways to explain the case of outliers that are not separated in a group of outliers for certain features and do not show any concept drift near that point.

Appendix A

Features Pairplot

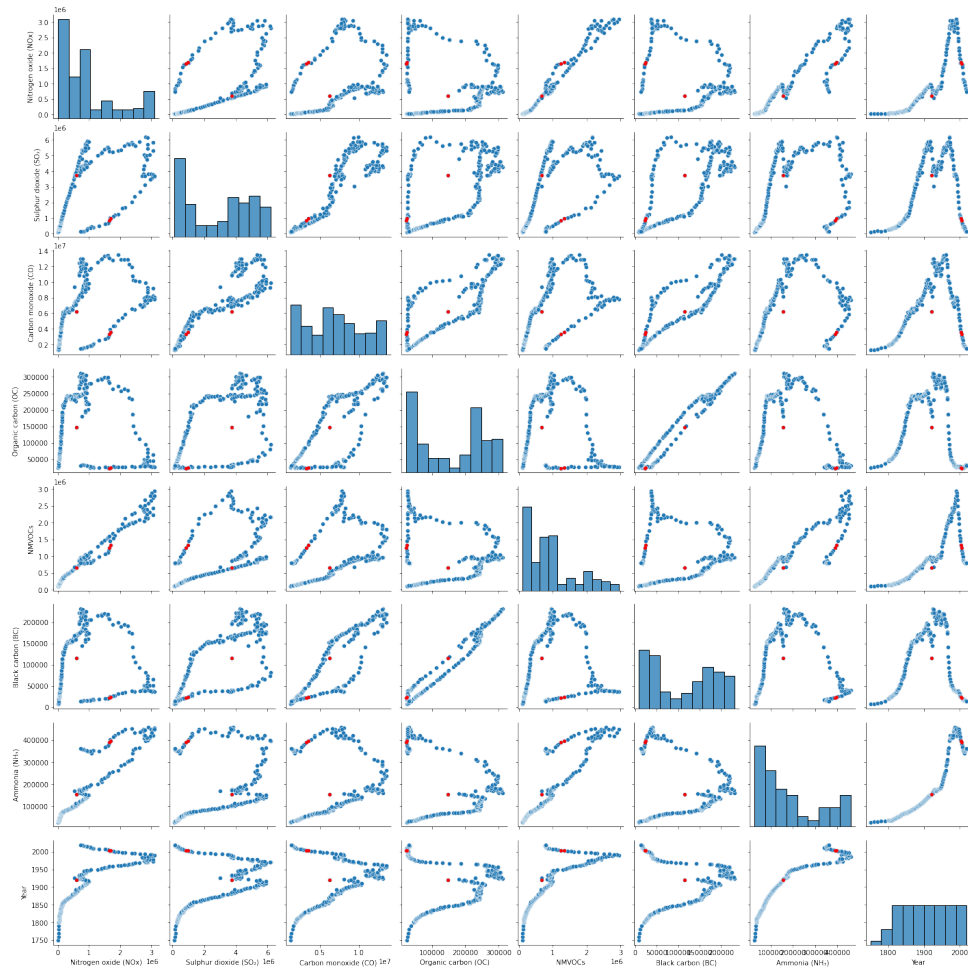


Figure A.1: Pairplot with outliers identified using HST algorithm.

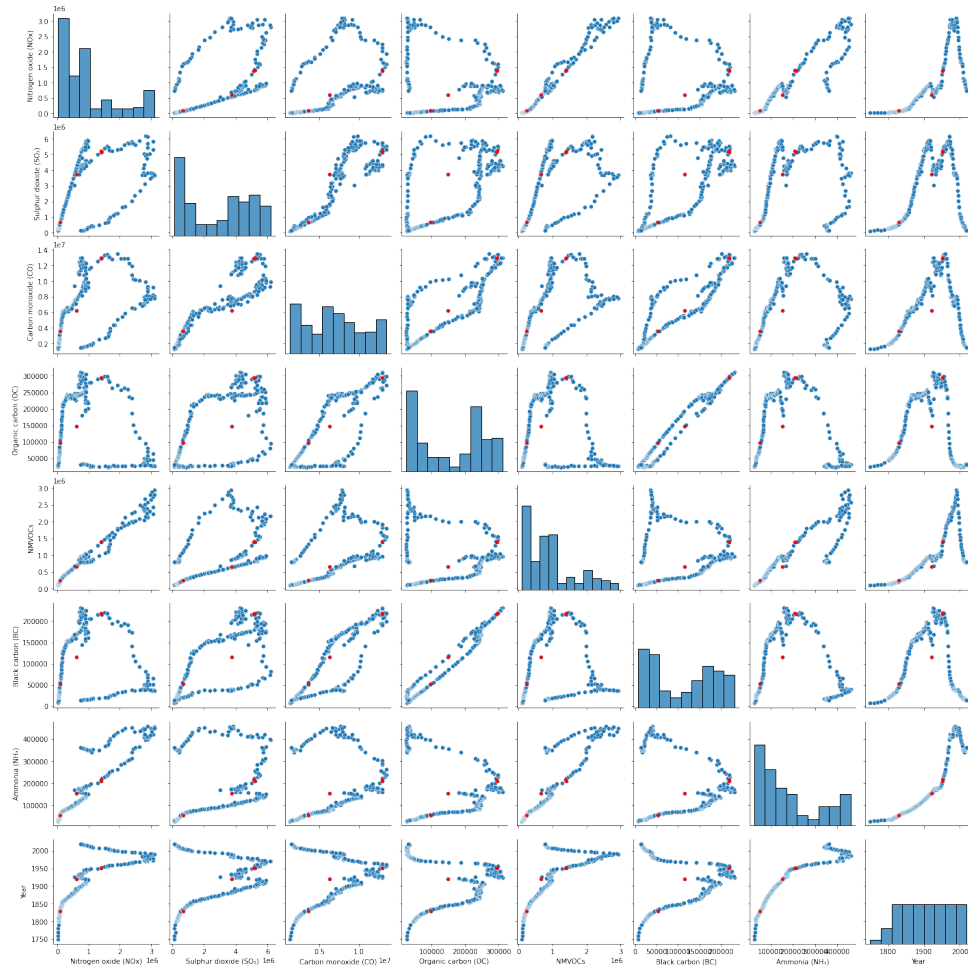


Figure A.2: Pairplot with outliers identified using ILOF algorithm.

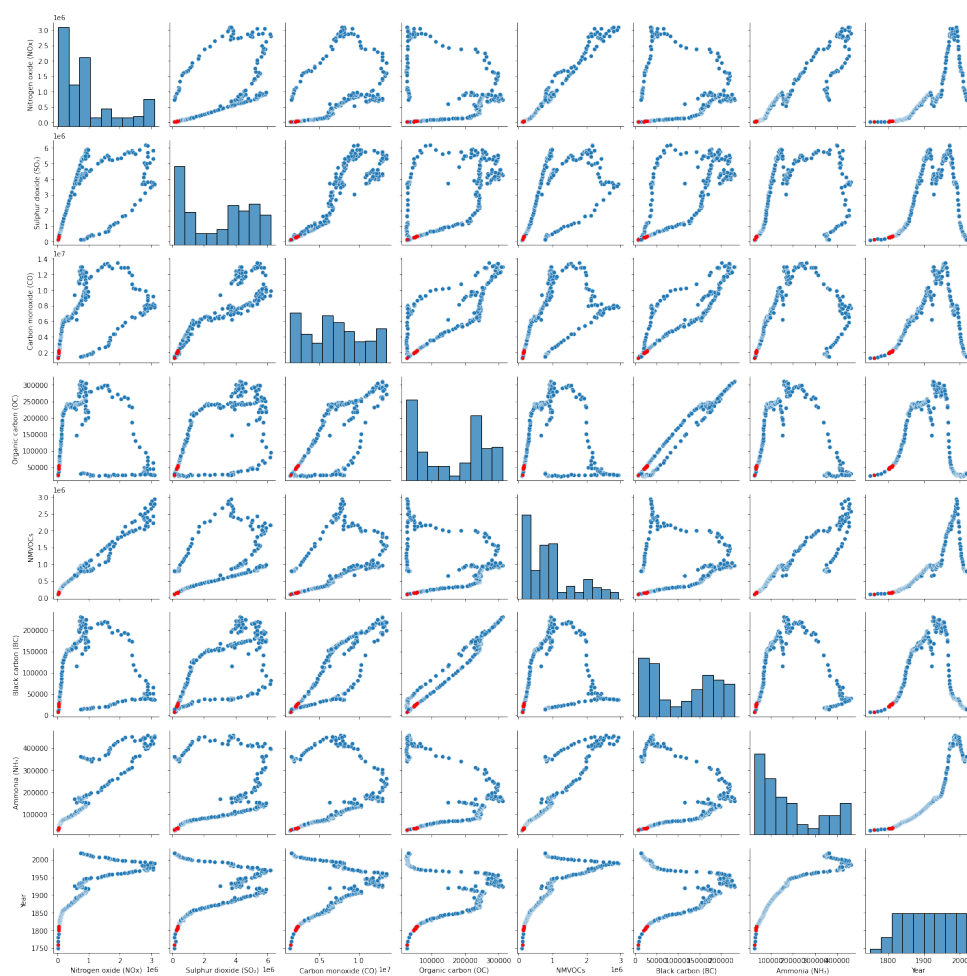


Figure A.3: Pairplot with outliers identified using OCSVM algorithm.

Bibliography

- [1] [Kaggle](#). Last accessed 6 June 2024.
- [2] [Citizen weather observer program](#). Last accessed 16 June 2024.
- [3] [Emissions of air pollutants](#), . Last accessed 27 May 2024.
- [4] [Data.world](#), . Last accessed 6 June 2024.
- [5] [Isolation forest](#). Last accessed 4 June 2024.
- [6] [Local outlier factor](#). Last accessed 4 June 2024.
- [7] [Our world in data](#). Last accessed 6 June 2024.
- [8] [Waikato region](#). Last accessed 6 June 2024.
- [9] C.C. Aggarwal. *Outlier Analysis*. Springer International Publishing, 2016. ISBN: 9783319475783.
- [10] Sajid Ali, Tamer Abuhmed, Shaker El-Sappagh, Khan Muhammad, Jose M. Alonso-Moral, Roberto Confalonieri, Riccardo Guidotti, Javier Del Ser, Natalia Díaz-Rodríguez, and Francisco Herrera. [Explainable artificial intelligence \(xai\): What we know and what is left to attain trustworthy artificial intelligence](#). *Information Fusion*, 99:101805, 2023. ISSN: 1566-2535. doi:<https://doi.org/10.1016/j.inffus.2023.101805>.
- [11] Fabrizio Angiulli and Clara Pizzuti. Fast outlier detection in high dimensional spaces. In Tapio Elomaa, Heikki Mannila, and Hannu Toivonen, editors, *Principles of Data Mining and Knowledge Discovery*, pages 15–27, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.
- [12] Piotr Augustyniak. [Time–frequency modelling and discrimination of noise in the electrocardiogram](#). *Physiological Measurement*, 24(3):753, jul 2003. doi:10.1088/0967-3334/24/3/311.
- [13] Piotr Augustyniak. [Wearable wireless heart rate monitor for continuous long-term variability studies](#). *Journal of Electrocardiology*, 44(2):195–200, 2011. ISSN: 0022-0736. doi:<https://doi.org/10.1016/j.jelectrocard.2010.11.014>.

- [14] Piotr Augustyniak, Magdalena Smoleń, Zbigniew Mikrut, and Eliaż Kańtoch. [Seamless tracing of human behavior using complementary wearable and house-embedded sensors](#). *Sensors*, 14(5):7831–7856, 2014. ISSN: 1424-8220. doi:10.3390/s140507831.
- [15] A. M. Bianco, M. García Ben, E. J. Martínez, and V. J. Yohai. [Outlier detection in regression models with arima errors using robust estimates](#). *Journal of Forecasting*, 20(8):565–579, 2001. doi:https://doi.org/10.1002/for.768.
- [16] Albert Bifet and Ricard Gavaldà. [Learning from time-changing data with adaptive windowing](#). volume 7, 04 2007. doi:10.1137/1.9781611972771.42.
- [17] Azzedine Boukerche, Lining Zheng, and Omar Alfandi. [Outlier detection: Methods, models, and classification](#). *ACM Comput. Surv.*, 53(3), jun 2020. ISSN: 0360-0300. doi:10.1145/3381028.
- [18] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. [Lof: Identifying density-based local outliers](#). In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, page 93–104, New York, NY, USA, 2000. Association for Computing Machinery. ISBN: 1581132174. doi:10.1145/342009.335388.
- [19] Suratna Budalakoti, Ashok Srivastava, R. Akella, and Eugene Turkov. Anomaly detection in large sets of high-dimensional symbol sequences. 01 2006.
- [20] Arindam Banerjee Chandola, Varun and Vipin Kumar. [Anomaly detection: A survey](#). *ACM Computing Surveys (CSUR)*, 41(3):15, 2009. doi:https://dl.acm.org/doi/abs/10.1145/1541880.1541882.
- [21] Varun Chandola, Arindam Banerjee, and Vipin Kumar. [Anomaly detection for discrete sequences: A survey](#). *IEEE Transactions on Knowledge and Data Engineering*, 24(5): 823–839, 2012. doi:10.1109/TKDE.2010.235.
- [22] Zhong-Zhou Chen, Yan-Mei Li, and Yu-Fen Zhao. [Modeling and efficient mining of intentional knowledge of outliers](#). *Journal of Chemical Research*, 2003:44–53, 01 2003. doi:10.3184/030823403103172779.
- [23] Xuan Hong Dang, Barbora Micenková, Ira Assent, and Raymond T. Ng. Local outlier detection with interpretation. In Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip Železný, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 304–320, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN: 978-3-642-40994-3.
- [24] A. P. Dempster, N. M. Laird, and D. B. Rubin. [Maximum likelihood from incomplete data via the em algorithm](#). *Journal of the Royal Statistical Society. Series B (Methodological)*, 39 (1):1–38, 1977. ISSN: 00359246.
- [25] Maria Bala Duggimpudi, Shaaban Abbady, Jian Chen, and Vijay V. Raghavan. [Spatio-temporal outlier detection algorithms based on computing behavioral outlier-ness factor](#). *Data Knowledge Engineering*, 122:1–24, 2019. ISSN: 0169-023X. doi:https://doi.org/10.1016/j.datak.2017.12.001.

- [26] Agnieszka Duraj and Piotr S. Szczepaniak. [Outlier detection in data streams — a comparative study of selected methods](#). *Procedia Computer Science*, 192:2769–2778, 2021. ISSN: 1877-0509. Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 25th International Conference KES2021. doi:<https://doi.org/10.1016/j.procs.2021.09.047>.
- [27] Chesner Désir, Simon Bernard, Caroline Petitjean, and Laurent Heutte. [One class random forests](#). *Pattern Recognition*, 46(12):3490–3506, 2013. ISSN: 0031-3203. doi:<https://doi.org/10.1016/j.patcog.2013.05.022>.
- [28] Asmaa Fawzy, Hoda M.O. Mokhtar, and Osman Hegazy. [Outliers detection and classification in wireless sensor networks](#). *Egyptian Informatics Journal*, 14(2):157–164, 2013. ISSN: 1110-8665. doi:<https://doi.org/10.1016/j.eij.2013.06.001>.
- [29] Pedro Galeano, Daniel Peña, and Ruey S. Tsay. [Outlier detection in multivariate time series by projection pursuit](#). *Journal of the American Statistical Association*, 101(474):654–669, 2006. ISSN: 01621459.
- [30] João Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In Ana L. C. Bazzan and Sofiane Labidi, editors, *Advances in Artificial Intelligence – SBIA 2004*, pages 286–295, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN: 978-3-540-28645-5.
- [31] Dimitrios Georgiadis, Maria Kontaki, Anastasios Gounaris, Apostolos Papadopoulos, Konstantinos Tsichlas, and Yannis Manolopoulos. [Continuous outlier detection in data streams: An extensible framework and state-of-the-art algorithms](#). pages 1061–1064, 06 2013. doi:[10.1145/2463676.2463691](https://doi.org/10.1145/2463676.2463691).
- [32] Markus Goldstein and Seiichi Uchida. [A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data](#). *PLOS ONE*, 11(4):1–31, 04 2016. doi:[10.1371/journal.pone.0152173](https://doi.org/10.1371/journal.pone.0152173).
- [33] Frank E. Grubbs. [Procedures for detecting outlying observations in samples](#). *Technometrics*, 11(1):1–21, 1969.
- [34] Manish Gupta, Jing Gao, Charu C. Aggarwal, and Jiawei Han. [Outlier detection for temporal data: A survey](#). *IEEE Transactions on Knowledge and Data Engineering*, 26(9): 2250–2267, 2014. doi:[10.1109/TKDE.2013.184](https://doi.org/10.1109/TKDE.2013.184).
- [35] Milos Hauskrecht, Iyad Batal, Michal Valko, Shyam Visweswaran, Gregory Cooper, and Gilles Clermont. [Outlier detection for patient monitoring and alerting](#). *Journal of biomedical informatics*, 46, 08 2012. doi:[10.1016/j.jbi.2012.08.004](https://doi.org/10.1016/j.jbi.2012.08.004).
- [36] D.M. Hawkins. [Identification of Outliers](#). Monographs on applied probability and statistics. Chapman and Hall, 1980. ISBN: 9780412219009.
- [37] Victoria Hodge and Jim Austin. [A survey of outlier detection methodologies](#). *Artificial Intelligence Review*, 22:85–126, 10 2004. doi:[10.1023/B:AIRE.0000045502.10941.a9](https://doi.org/10.1023/B:AIRE.0000045502.10941.a9).

- [38] Victoria Hodge and Jim Austin. [A survey of outlier detection methodologies](#). *Artificial Intelligence Review*, 22:85–126, 10 2004. doi:10.1023/B:AIRE.0000045502.10941.a9.
- [39] Biao Huang and Peng Yang. [Finding key knowledge attribute subspace of outliers in high-dimensional dataset](#). *Expert Systems with Applications*, 38(8):10147–10152, 2011. ISSN: 0957-4174. doi:https://doi.org/10.1016/j.eswa.2011.02.077.
- [40] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., USA, 1988. ISBN: 013022278X.
- [41] Jianguo Jiang, Jiuming Chen, Tianbo Gu, Kim-Kwang Raymond Choo, Chao Liu, Min Yu, Weiqing Huang, and Prasant Mohapatra. [Warder: Online insider threat detection system using multi-feature modeling and graph-based correlation](#). In *MILCOM 2019 - 2019 IEEE Military Communications Conference (MILCOM)*, page 1–6. IEEE Press, 2019. doi:10.1109/MILCOM47813.2019.9020931.
- [42] Jianguo Jiang, Jiuming Chen, Tianbo Gu, Kim-Kwang Raymond Choo, Chao Liu, Min Yu, Weiqing Huang, and Prasant Mohapatra. [Anomaly detection with graph convolutional networks for insider threat and fraud detection](#). In *MILCOM 2019 - 2019 IEEE Military Communications Conference (MILCOM)*, pages 109–114, 2019. doi:10.1109/MILCOM47813.2019.9020760.
- [43] Fabian Keller, Emmanuel Müller, Andreas Wixler, and Klemens Böhm. [Flexible and adaptive subspace search for outlier analysis](#). In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, CIKM '13*, page 1381–1390, New York, NY, USA, 2013. Association for Computing Machinery. ISBN: 9781450322638. doi:10.1145/2505515.2505560.
- [44] Edwin Knorr and Raymond Ng. Finding intensional knowledge of distance-based outliers. 99, 06 1999.
- [45] Edwin M. Knorr and Raymond T. Ng. [Algorithms for mining distance-based outliers in large datasets](#). In *Very Large Data Bases Conference*, 1998.
- [46] Maria Kontaki, Anastasios Gounaris, Apostolos N. Papadopoulos, Kostas Tsichlas, and Yannis Manolopoulos. [Continuous monitoring of distance-based outliers over data streams](#). *2011 IEEE 27th International Conference on Data Engineering*, pages 135–146, 2011.
- [47] Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. [Interpreting and unifying outlier scores](#). pages 13–24, 04 2011. doi:10.1137/1.9781611972818.2.
- [48] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. [Isolation forest](#). In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, 2008. doi:10.1109/ICDM.2008.17.
- [49] Ninghao Liu, Donghwa Shin, and Xia Hu. [Contextual outlier interpretation](#). In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 2461–2467. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi:10.24963/ijcai.2018/341.

- [50] Wei Liu, Yu Zheng, Sanjay Chawla, Jing Yuan, and Xie Xing. [Discovering spatio-temporal causal interactions in traffic data streams](#). In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, page 1010–1018, New York, NY, USA, 2011. Association for Computing Machinery. ISBN: 9781450308137. doi:10.1145/2020408.2020571.
- [51] J. Ma and S. Perkins. [Time-series novelty detection using one-class support vector machines](#). In *Proceedings of the International Joint Conference on Neural Networks, 2003.*, volume 3, pages 1741–1745 vol.3, 2003. doi:10.1109/IJCNN.2003.1223670.
- [52] Alexandra Meliou, Wolfgang Gatterbauer, Joseph Halpern, Christoph Koch, Katherine Moore, and Dan Suciu. Causality in databases. *IEEE Data Eng. Bull.*, 33:59–67, 01 2010.
- [53] Barbora Micenkova, Raymond Ng, Xuan-Hong Dang, and Ira Assent. [Explaining outliers by subspace separability](#). pages 518–527, 12 2013. ISBN: 978-0-7695-5108-1. doi:10.1109/ICDM.2013.132.
- [54] HebaTallah Mohamed, Ahmed Sharaf, and Mohamed Abd. [Mining frequent itemsets from online data streams: Comparative study](#). *International Journal of Advanced Computer Science and Applications*, 4, 08 2013. doi:10.14569/IJACSA.2013.040717.
- [55] Christoph Molnar. [Interpretable Machine Learning](#). 2 edition, 2022.
- [56] Jacob Montiel, Max Halford, Saulo Martiello Mastelini, Geoffrey Bolmier, Raphael Sourty, Robin Vaysse, Adil Zouitine, Heitor Murilo Gomes, Jesse Read, Talel Abdessalem, et al. River: machine learning for streaming data in python. 2021.
- [57] Daniel Bull Eibe Frank Yunzhe Jia Jacob Montiel Nick Lim, Albert Bifet and Bernhard Pfahringer. [Showcasing the taiao project: providing resources for machine learning from images of new zealand’s natural environment](#). *Journal of the Royal Society of New Zealand*, 53(1):69–81, 2023. doi:10.1080/03036758.2022.2118321.
- [58] Krithiq p. [Outlier handling](#), 2023. Last accessed 27 May 2024.
- [59] Egawati Panjei, Le Gruenwald, Eleazar Leal, Christopher Nguyen, and Shejuti Silvia. [A survey on outlier explanations](#). *The VLDB Journal*, 31(5):977–1008, jan 2022. ISSN: 1066-8888. doi:10.1007/s00778-021-00721-1.
- [60] Dragoljub Pokrajac, Aleksandar Lazarevic, and Longin Jan Latecki. [Incremental local outlier detection for data streams](#). In *2007 IEEE Symposium on Computational Intelligence and Data Mining*, pages 504–515, 2007. doi:10.1109/CIDM.2007.368917.
- [61] Dragoljub Pokrajac, Aleksandar Lazarevic, and Longin Jan Latecki. [Incremental local outlier detection for data streams](#). *2007 IEEE Symposium on Computational Intelligence and Data Mining*, pages 504–515, 2007.
- [62] Marco Tulio Ribeiro. [Local interpretable model-agnostic explanations \(lime\)](#), . Last accessed 10 June 2024.

- [63] Marco Tulio Ribeiro. [Lime implementation tutorial](#), . Last accessed 10 June 2024.
- [64] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. [Model-agnostic interpretability of machine learning](#). *ArXiv*, abs/1606.05386, 2016.
- [65] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.
- [66] Mahsa Salehi and Lida Rashidi. [A survey on anomaly detection in evolving data: \[with application to forest fire risk prediction\]](#). *SIGKDD Explor. Newsl.*, 20(1):13–23, may 2018. ISSN: 1931-0145. doi:10.1145/3229329.3229332.
- [67] Erich Schubert, Remigius Wojdanowski, Arthur Zimek, and Peer Kröger. [On Evaluation of Outlier Rankings and Outlier Scores](#), pages 1047–1058. 04 2012. ISBN: 978-1-61197-232-0. doi:10.1137/1.9781611972825.90.
- [68] David W. Scott. [Multivariate density estimation and visualization](#). 2012.
- [69] Allan Seheult, P. Green, Peter Rousseeuw, and Annick Leroy. [Robust regression and outlier detection](#). *Journal of the Royal Statistical Society. Series A (Statistics in Society)*, 152:133, 01 1989. doi:10.2307/2982847.
- [70] Abhishek Sharma. [Decrypting your machine learning model using lime](#). Last accessed 10 June 2024.
- [71] Amran Siddiqui, Alan Fern, Thomas Dietterich, and Weng-Keen Wong. [Sequential feature explanations for anomaly detection](#). *ACM Transactions on Knowledge Discovery from Data*, 13, 02 2015. doi:10.1145/3230666.
- [72] Amran Siddiqui, Alan Fern, Thomas Dietterich, Ryan Wright, Alec Theriault, and David Archer. [Feedback-guided anomaly discovery via online optimization](#). pages 2200–2209, 07 2018. doi:10.1145/3219819.3220083.
- [73] Fernando Silveira and Christophe Diot. [Urca: Pulling out anomalies by their root causes](#). pages 1 – 9, 04 2010. doi:10.1109/INFCOM.2010.5462151.
- [74] R. Smith, A. Bivens, M. Embrechts, C. Palagiri, and Boleslaw Szymanski. Clustering approaches for anomaly based intrusion detection. *Proceedings of Intelligent Engineering Systems Through Artificial Neural Networks*, pages 579–584, 01 2002.
- [75] Jamiyan Sukhbaatar, Bold Zagd, and Nyamjav Jambaljav. Detection of point outliers in meteorological data (case study: Ulaanbaatar, mongolia). In Jeng-Shyang Pan, Jianpo Li, Keun Ho Ryu, Zhenyu Meng, and Aleksandra Klasnja-Milicevic, editors, *Advances in Intelligent Information Hiding and Multimedia Signal Processing*, pages 68–75, Singapore, 2021. Springer Singapore. ISBN: 978-981-33-6757-9.

- [76] Maciej Szmit and Anna Szmit. [Usage of modified holt-winters method in the anomaly detection of network traffic: Case studies](#). *J. Comput. Networks Commun.*, 2012:192913:1–192913:5, 2012.
- [77] Naoya Takeishi. [Shapley values of reconstruction errors of pca for explaining anomaly detection](#). pages 793–798, 11 2019. doi:10.1109/ICDMW.2019.00117.
- [78] Swee Tan, Kai Ting, and Fei Tony Liu. [Fast anomaly detection for streaming data](#). pages 1511–1516, 01 2011. doi:10.5591/978-1-57735-516-8/IJCAI11-254.
- [79] Luan Tran, Liyue Fan, and Cyrus Shahabi. [Distance-based outlier detection in data streams](#). *Proc. VLDB Endow.*, 9(12):1089–1100, aug 2016. ISSN: 2150-8097. doi:10.14778/2994509.2994526.
- [80] Ruey S. Tsay, Daniel Pena, and Alan E. Pankratz. [Outliers in multivariate time series](#). *Biometrika*, 87(4):789–804, 2000. ISSN: 00063444.
- [81] Aaron Tuor, Samuel Kaplan, Brian Hutchinson, Nicole Nichols, and Sean Robinson. Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. 10 2017.
- [82] Krishnamurthy Viswanathan, Choudur Lakshminarayan, Vanish Talwar, Chengwei Wang, G. Macdonald, and Wade Satterfield. [Ranking anomalies in data centers](#). pages 79–87, 04 2012. ISBN: 978-1-4673-0267-8. doi:10.1109/NOMS.2012.6211885.
- [83] Lei Xing, Wenjun Wang, Guixiang Xue, Hao Yu, Xiaotong Chi, and Weidi Dai. Discovering traffic outlier causal relationship based on anomalous dag. In Ying Tan, Yuhui Shi, Fernando Buarque, Alexander Gelbukh, Swagatam Das, and Andries Engelbrecht, editors, *Advances in Swarm and Computational Intelligence*, pages 71–80, Cham, 2015. Springer International Publishing.
- [84] Jiawei Yang, Susanto Rahardja, and Pasi Fränti. [Outlier detection: how to threshold outlier scores?](#) In *Proceedings of the International Conference on Artificial Intelligence, Information Processing and Cloud Computing*, AIIPCC '19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN: 9781450376334. doi:10.1145/3371425.3371427.
- [85] Jinhong Yang, Tingquan Deng, and Ran Sui. An adaptive weighted one-class svm for robust outlier detection. In Yingmin Jia, Junping Du, Hongbo Li, and Weicun Zhang, editors, *Proceedings of the 2015 Chinese Intelligent Systems Conference*, pages 475–484, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg. ISBN: 978-3-662-48386-2.
- [86] Rose Yu, Huida Qiu, Zhen Wen, ChingYung Lin, and Yan Liu. [A survey on social media anomaly detection](#). 18(1):1–14, aug 2016. ISSN: 1931-0145. doi:10.1145/2980765.2980767.
- [87] Alaettin Zubaroglu and Mehmet Volkan Atalay. [Data stream clustering: a review](#). *Artificial Intelligence Review*, 54, 02 2021. doi:10.1007/s10462-020-09874-x.