

**SLOVAK TECHNICAL UNIVERSITY IN BRATISLAVA**

**FACULTY OF CHEMICAL AND FOOD TECHNOLOGY**

**MICROSERVICE ARCHITECTURE FOR STREAMING DATA PROCESSING**

SEMESTRAL PROJECT

| **2023** | **Bazarkhanova Dinara** |
| --- | --- |

**SLOVAK TECHNICAL UNIVERSITY IN BRATISLAVA**

**FACULTY OF CHEMICAL AND FOOD TECHNOLOGY**

**MICROSERVICE ARCHITECTURE FOR STREAMING DATA PROCESSING**

SEMESTRAL PROJECT

| Study programme: | Bachelor |
| --- | --- |
| Field of study: | (2621R03) B-RPxA Process Control |
| Training workplace: | Department of Information Engineering and Process Control - IIEAM FCFT |
| Supervisor: | prof. Ing. Michal Kvasnica, PhD. |
| Consultant: | Ing. Marek Wadinger |

| **Bratislava 2023** | **Bazarkhanova Dinara** |
| --- | --- |

# **ABSTRACT**

The increasing data availability and its higher importance in data-driven decision-making leads to the increasing interest in their processing. The Internet of Things (IoT) connects devices to a computer network and allows them to collect, analyse, process and transmit data streams to other objects through software, applications or technical devices. The data streams can be routinely stored in files as batches. This approach gives the possibility of asynchronous and offline data processing. Nevertheless, many situations require a low-latency evaluation and the ability to react to new information rapidly.

Moreover, significant acceleration of data collection made their storing non-feasible. Real-time stream processing gained research attention and popularity in businesses in recent years. The problem with this approach arises from the need to preserve states across different events and connecting services working with different frequencies of data processing. The main goal of this thesis is to develop a microservice architecture for streamed data. The architecture employs an MQTT client to connect remote IoT devices with the server. Further, the pub-sub streaming platform queues the data from the server into a machine learning training and prediction service. Lastly, the time series database stores (out-of-sample predictions/results of anomaly detection) based on streamed data is to be created.

Key words: Anomaly Detector, Python, RiverML, Streamz, machine learning.

# **CONTENT**

[ABSTRACT](#_699t12mc44b) 3

[CONTENT](#_agimhp47ccnz) 4

[INTRODUCTION](#_wot8c9j1a6tr) 5

[THEORETICAL PART](#_p6bdtesak8j5) 6

[Machine Learning](#_b34z9pjrqji3) 6

[Supervised vs. Unsupervised](#_ezhrtasbxi09) 7

[Batch vs. Streaming ML](#_mjm10rq5801) 8

[Anomaly Detection](#_wif3x34a09k6) 10

[Anomaly Detection using Streaming Unsupervised Machine Learning](#_7mq07xohh1sj) 11

[PRACTICAL PART](#_8n6q6k6hushr) 14

[Streamz](#_mvp93flji1or) 14

[RiverML](#_vggyi7ya70x9) 14

[Paho-mqtt](#_71pyk5l1p8a7) 15

[By combining the Streamz and RiverML and sometimes using paho-mqtt libraries I was able to make case studies which show the anomaly detection in use and are represented by metric, graph and schema.](#_f10uhcp2yy2w) 16

[Case Study #1](#_sjpefld1nkzy) 16

[Case Study #2](#_olq87wdxsah7) 17

[Case Study #3](#_55yddaf7jakt) 18

[CONCLUSION](#_fsmtmpo8kcto) 20

[LITERATURE](#_gckze8gf8ycb) 21

# **INTRODUCTION**

This work is describing a Machine Learning in general but diving deeper in unsupervised algorithm of Machine Learning focused on constrained environments where you have limitations in terms of storage space or other kinds of constraints that make it hard to perform a supervised learning with access to all the data that may be necessary to train the model. Instead I am using a streaming technique in order to be able to have learning on demand as input data is coming (is fed to the pipeline). I will be able to process parts of the data over time without having to store all the data. As an example of an application of Streamed Unsupervised Machine Learning I am focusing on anomaly detection problems for detecting anomalies in the stream of data and I will also explore some of the difficulties related to doing anomaly detection in this scenario. I will identify the difficulties of training and accuracy.

# **THEORETICAL PART**

## **Machine Learning**

Machine learning (ML) is a subset of data science, the field of knowledge that studies how to extract value from data. In turn, ML offers methods and practices for training algorithms on this data to solve problems such as classifying objects in images, without specifying the rules and programming patterns. In essence, we train the program to make decisions with little or no human intervention. From a business perspective, once implemented in production, the model can automate manual processes. (El Naqa, 2015)

Nowadays supervised and unsupervised learning models help companies better interact with their clients. Smart technology is everywhere, security systems can convert fingerprints and face scans into biometrics to unlock doors and devices, banking systems can detect unusual transactions and automatically send a message to the card owner, any processes can be controlled and adapted. All these wonderful technologies are constantly being improved through the use of machine learning (ML) algorithms.

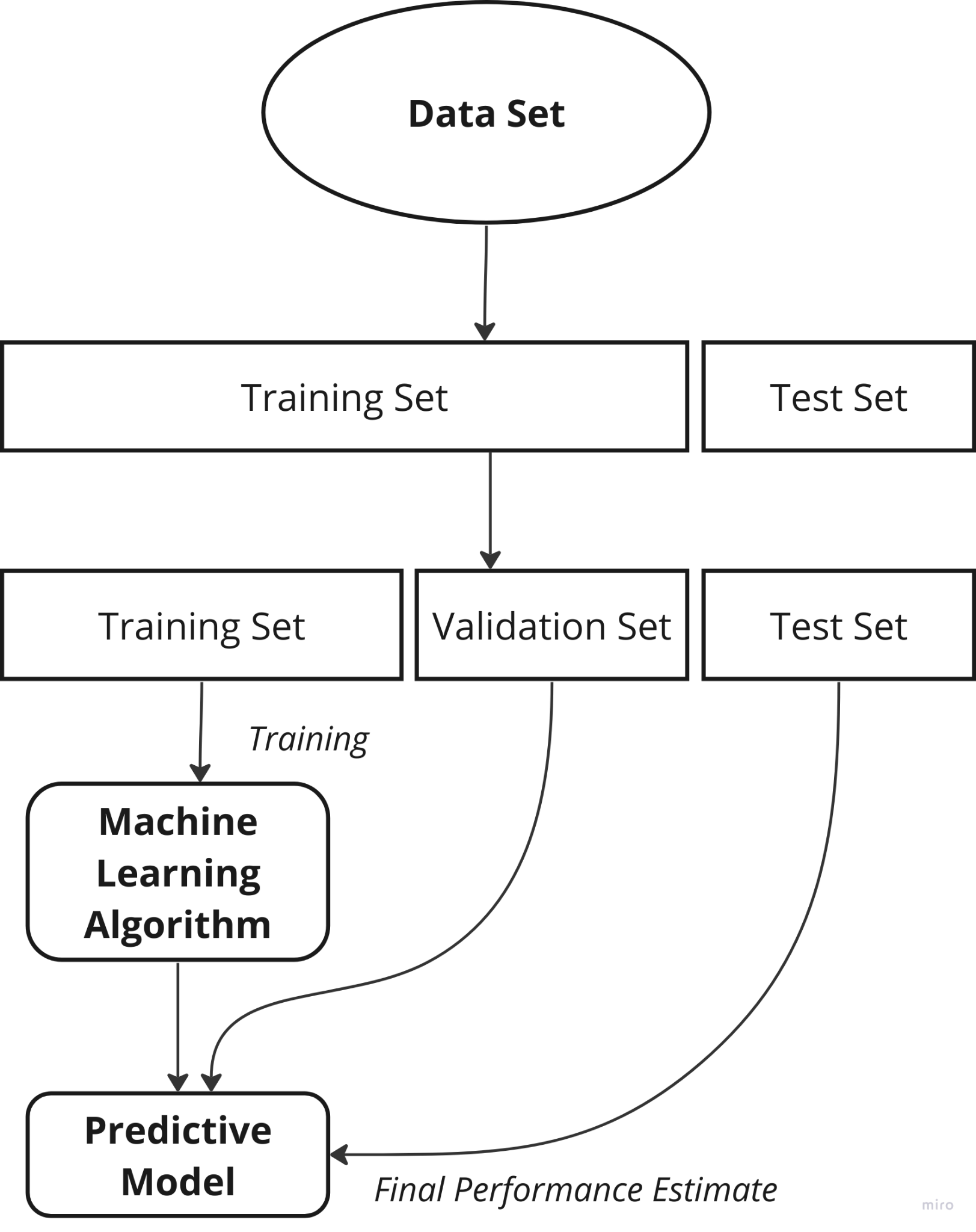
The process of using (not developing) a machine learning model (anomaly detector) has three steps: learning, prediction, validation. Similar to the human’s ability to learn from everyday experiences, ML can learn and improve predictions over multiple iterations. Internet of Things (IoT) devices provide training data for ML models, collected from processes, transactions and industrial machines.

Data processing algorithms help to sift, classify and group information based on various parameters of these machines. By processing and combining data, Machine Learning can then create models that accurately predict some of the human behaviours and result in appropriate responses.

Machine learning is a way to train computers without programming or explicit instructions, using only patterns and inference. In this situation, the machine is not told to “do 1 + 1”, but is shown an example and asked to do “similar”. To some extent, this is similar to the logic of human behaviour. Thanks to the development of ML approaches, machines are starting to “do similar” with a deeper level of elaboration.

For a long time, computers were used to solve problems that a person could handle on his own. But it gradually became clear that the advantage of machines was in the speed of operations, quick analytics of large amounts of data that a person would manually process for a very long time. If you just set the right input conditions the computer can quickly give a result. This is how machine learning technologies and artificial intelligence in its broadest sense appeared.

Often, artificial intelligence learns from examples of how this task was solved by a person (or by the machine itself). Sets of such examples are called datasets. Using datasets, developers train algorithms to classify information, make predictions, highlight important points, and find new ways to solve known problems. Over time, datasets grow, learning algorithms also change.



Thanks to the development of machine learning technologies, YouTube Music, Spotify has learned to independently find and add songs to the playlist that you might like. The service analyses your preferences, as well as the preferences of millions of other users, whose tastes may be similar to yours or different. This is the development of a technology that was once invented by Netflix and which completely changed the logic of content consumption in the modern world.

A person would never have coped with such a task - it is impossible to assign to each user an expert musicologist who is familiar with the entire musical heritage of mankind and knows how to select works for different tastes. And for the Youtube Music recommender system, this is not difficult thanks to machine learning methods and the enormous speed of operations.

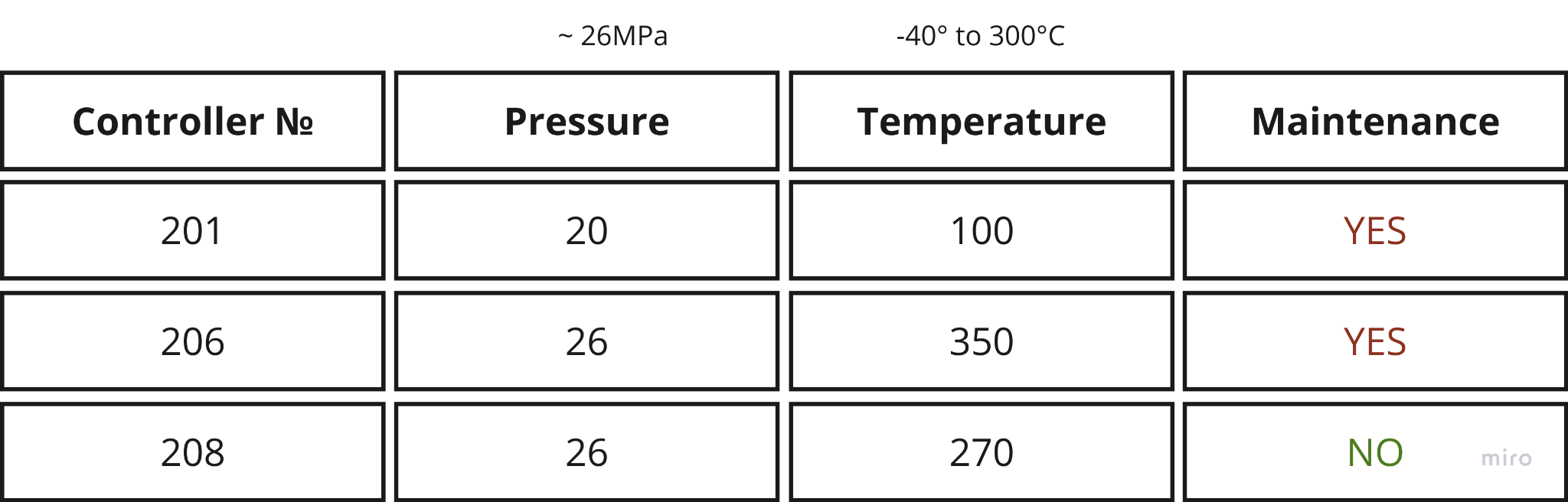
Furthermore, with the help of AI-powered social media platforms like Instagram and YouTube, preferences of users can be identified by scanning the content they read and watch. This allows for similar content to be recommended and personalised ads to be suggested based on the user's interests, increasing the probability of obtaining the products or services advertised.

## **Supervised vs. Unsupervised**

In general terms, based on the data processing there are two approaches to machine learning - supervised and unsupervised. The main difference is that the former uses labelled data to help predict, while the latter does not. But both approaches have more subtle differences and key areas in which they excel each other.

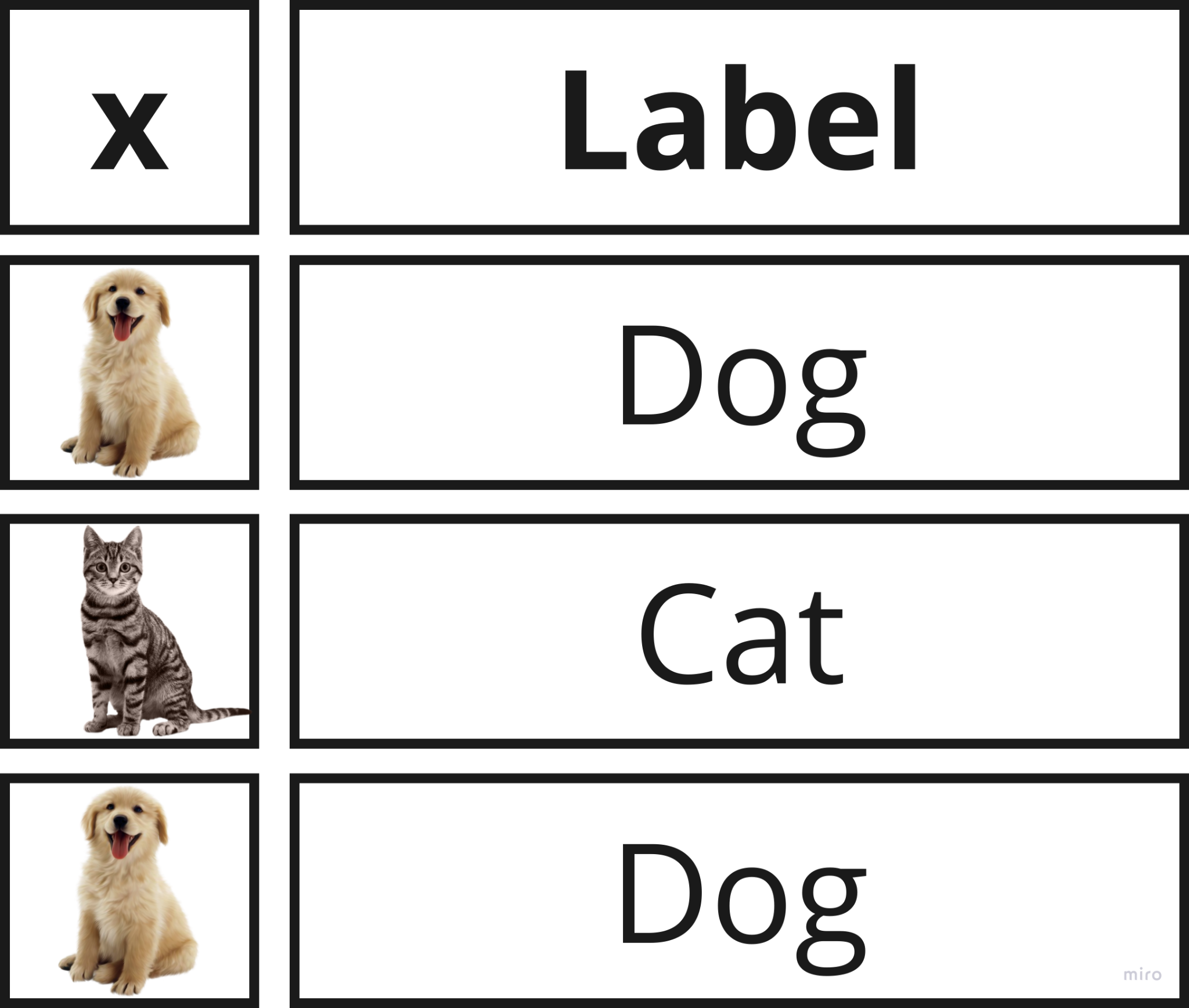
Supervised learning is a machine learning approach based on the use of labelled datasets. Such datasets are used to train algorithms aimed at classifying data or accurately predicting results. In this case “label” is a known value which is specified for each incoming data. It could easily be some simple binary (Yes/No), a score (0/1) or a category (cat/dog).

For example, in the oil production factory we have Crude Oil Storage Tank equipment. One of the important parts of this tank is the UTK Displacer Level Controller which is used to control the liquid level. Time by time the controllers need maintenance or replacement based on its functionality. The controller’s working condition is 26MPa for pressure and -40℃~+300℃ for temperature. If it is higher or lower than these values, they need maintenance. On the table below, it is given the actual values for pressure and temperature. Based on it, our supervised machine learning model can identify the controller which is out of work.



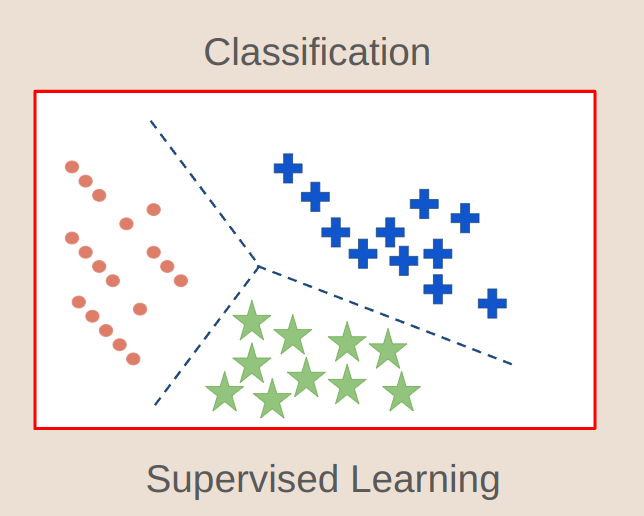
UTK Displacer Level Controller for the Crude Oil Storage Tank

<https://www.ddtop.com/en/wp-content/uploads/2020/11/Crude-Oil-Storage-Tank-Area.pdf>  
Basically, this is our supervised model, where we provided labelled data and it created a model which can accurately predict the output. When the prediction output is something like maintenance needed or not (Yes/ No) it is called binary classification.



Supervised learning can be divided into two types (according to tasks related to data mining): classification and regression.

* In solving classification problems, for example, to separate apples from oranges, an algorithm is used to accurately categorise the test data. In the real world, machine learning algorithms can be used to sort spam into a separate email folder. Linear classifiers, support vector machines, decision trees, and random forest are all common classification algorithms.

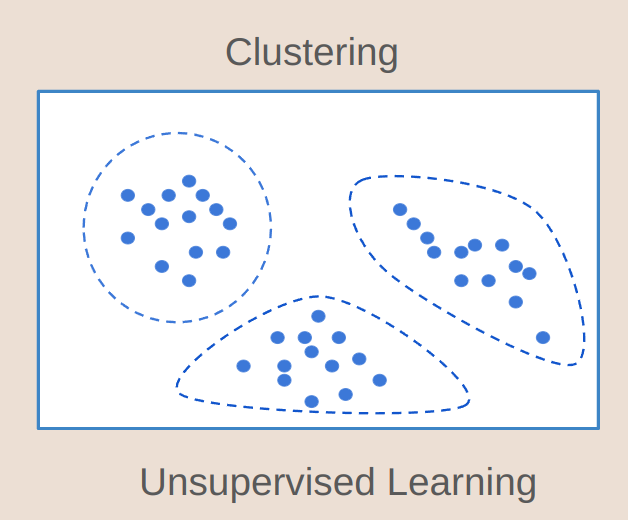


* Regression is another type of supervised learning method that uses an algorithm to discover the relationship between dependent and independent variables. Regression data models help you predict numbers based on point data, such as future sales revenue for a particular company. Common regression analysis algorithms include linear regression, logistic regression, and polynomial regression.

In unsupervised machine learning, algorithms are used to analyse and group raw datasets. These algorithms identify patterns in the data without human intervention (therefore they are "out of control").

Unsupervised learning models are used to perform three main tasks - clustering, association and dimensionality reduction:

* **Clustering** is a data mining technique used to group unlabeled data based on their similarities and differences. For example, K-means clustering algorithms cluster similar data points into groups, where the K value represents the size of the group and the degree of structure. This method is suitable for market segmentation, image compression, etc.



* **Association** is an unsupervised learning method that uses certain rules to identify relationships between variables and a given set of data. These methods are often used to analyse shopping behaviour and create recommendation services and select products in the "Buy with this product" categories.
* **Dimensionality reduction** is a learning technique that is used when there are too many features (or dimensions) in a particular data set. It reduces the number of inputs to manageable while maintaining their integrity. This technique is often used in the data processing phase, such as when autoencoders remove noise from visual data to improve image quality.

## Other Key Differences Between Supervised and Unsupervised Learning

Goals. The purpose of supervised learning is to predict outcomes from new data. You know in advance what kind of result to expect. The goal of unsupervised learning is to get useful information from a huge amount of new data. In the course of learning, the machine itself determines which information from the set is unusual or of interest.

Areas of use. Supervised learning models are ideal for spam detection, sentiment analysis, weather forecasting, price changes, and more. Unsupervised learning models are designed to detect outliers, improve recommendation services, predict customer behaviour, and medical imaging.

Complexity. Supervised learning is a simple machine learning technique that is usually calculated using programs such as R or Python. Unsupervised learning requires powerful tools to deal with large amounts of unclassified data. Unsupervised learning models are highly computationally complex, since a large training set is needed to obtain the desired results.

Flaws. Unsupervised learning models can be time consuming, and labelling inputs and outputs requires experience and expertise. Unsupervised learning methods can produce very inaccurate results if the output variables are not human validated.

Supervised and unsupervised learning: which is better?

Classifying big data in supervised learning is not an easy task. However, the output results are accurate and reliable. Conversely, unsupervised learning allows you to process large amounts of data in real time. However, in this case there is a lack of transparency regarding data clustering and there is a higher risk of inaccurate results. The way out is partially supervised learning.

Partially supervised learning is the sweet spot. This method allows you to use a set that includes both labelled and unlabeled data. It is especially useful when it is difficult to extract relevant data features and you are working with a large amount of data.

Partially supervised training is ideal for medical imaging: a small amount of training data can greatly improve accuracy. For example, a radiologist might label a small set of CT scans with tumours or pathologies so that the machine can more accurately identify patients who need more attention.

To sum up, ML works based on a few algorithm types, which are supervised or unsupervised. Supervised and unsupervised learning algorithms have a crucial difference - supervised learning uses “labelled as a right answer” data, while unsupervised learns and predicts in a real time. By “labelled” data it is meant that we have the training dataset, which consists of samples of inputs and their desired outputs. (Singh, 2016)

The reality for unsupervised machine learning is that this data is not available. Unsupervised learning can identify unknown patterns in data without human intervention. Instead of that the algorithm should find relationships within the data by itself. Clustering, anomaly detection, and dimensionality reduction are examples of unsupervised learning tasks. (Hahne, 2008)

The final goals of supervised and unsupervised learning are different. While the aim of supervised learning is about predicting outcomes for a new data that is being introduced, the aim of unsupervised learning is about getting new insights from huge amounts of new data. In the former one, the user will know what results they can expect, while in the last one, they hope to discover something new and unknown.

## Batch vs. Streaming ML

Real-time (streaming) data processing is becoming almost the de-facto standard today. It is important for businesses to perform calculations on the fly, gaining valuable insights while the data is still relevant. This is made possible by streaming event processing technologies which are supported by tools such as Apache Kafka, Flink, NiFi, Storm, Spark Structured Streaming and platforms based on them. With these technologies, fintech companies can detect fraud in real time, retailers get a complete view of customer activity, and cloud service providers prevent software and infrastructure failures before users experience downtime.

Initially, analytical and transactional systems used batch technologies, when the synchronisation of the data accumulated during the day was performed at night. Therefore, the actual data differed from the calculated ones by 7-13 hours, which can be critical for a number of domains. Although modern batch technologies have greatly increased the frequency of processing batches, it can still take hours to compute with a large data set, and the results will still not be 100% accurate. In addition, different systems based on different packet processing schedules can produce inconsistent calculations and in order to store thousands of data systems’ storage needs to be big enough which can cause an additional expense for the companies.

Overall, there are two main types of incoming data: batch data (allows only the prediction) and streaming data (allows to improve a model over time).

*Batch processing* is the processing of a large amount of dataset all at once. The data consists of thousands of records for a day and can be stored in a variety of ways (txt file, json, record, etc). The work is usually completed simultaneously in non-stop and sequential order.c

A machine learning batch pipeline typically includes:

* splitting data into training and test sets;
* fitting the model for training;
* calculating model performance on a test set;
* launching the model into production.

However, after launching into production, you should make sure that the model is reliable when the input data changes and periodically retrain it on a new set. As the volume of data grows, training a machine learning model requires more time and resources. Therefore, batch training is not the best option when the application requires frequent model updates or there are limited computing resources for storing and processing big data. (Shalev-Shwartz, 2004)

Whereas *data streaming* is a technology for working with live data, such as information from sensors that transmit values once a minute, seconds or other time values.

Stream processing is based on the following key aspects:

* continuous processing of the incoming data stream as the event happens;
* low latency up to a fraction of a second;
* accounting for the time sequence and order of occurrence of events during processing, even if they reached the stream processor with an offset due to network delays.

With streaming learning, a ML model is gradually trained on a data stream without waiting for a complete package. Therefore, every learning step becomes fast and cheap, making it ideal for applications that change quickly and for companies with limited computing resources.

One of the benefits of using the Streaming approach for Machine Learning is that you do not need to hold all the data needed for training your data at once. You may be able to feed your model pieces of the data over time and in doing so you never need to hold it all in memory at the same time. It may be impossible depending on the amount of data you need to train your model. That is one of the benefits. You are able to train your model without saving all the data needed for training, but feeding the model all the data by demand. For example, phones or any embedded hardware will not have infinite storage, it is limited. That is the nature of any embedded device which needs to deal with constraints such as limited amount of memory, storage. There should be strategies that work in such a constrained environment. And in the context of Machine Learning Streamed ML is a way to overcome this barrier. It is also best to use if the event needs to be detected right away and responded quickly. (Gomes, 2019)

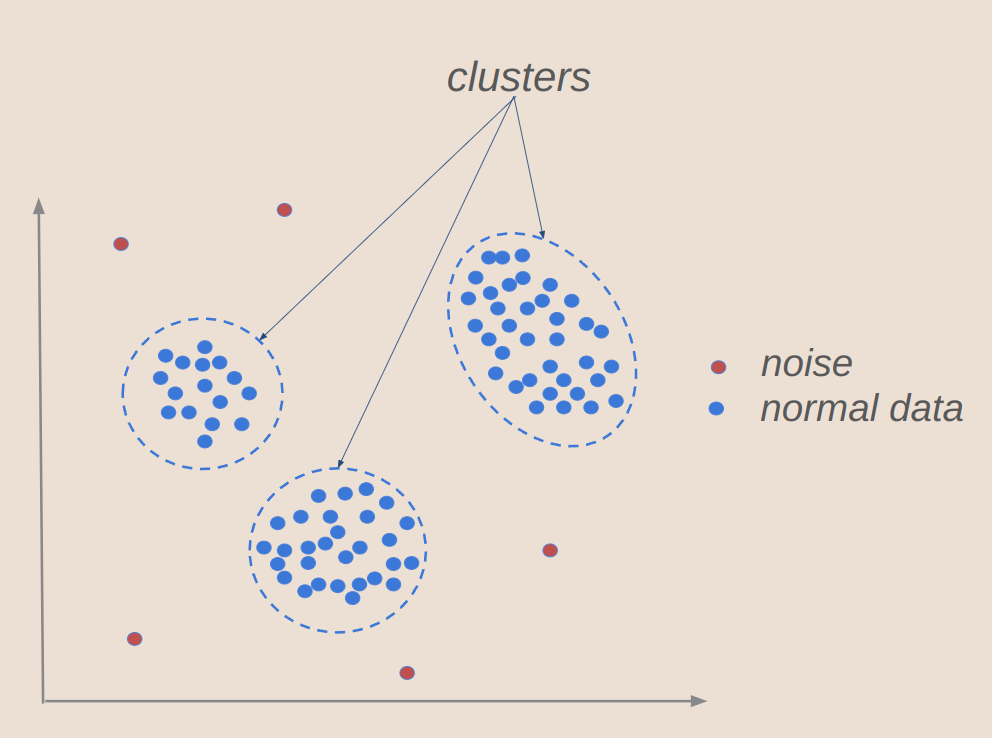
Streaming machine learning is a method of applying machine learning algorithms to data streams - data points sequences arriving in real-time. The main goal of streaming machine learning is to learn from the data as it arrives, instead of waiting for all the data to be collected and only then train a model.

By combining Streaming ML and Unsupervised ML we get Streaming unsupervised machine learning, which refers to the process of using unsupervised learning algorithms on data streams, which are continuous and potentially infinite sequences of data points. This can be useful for analysing real-time data, such as monitoring sensor readings, detecting patterns in financial transactions, or tracking customer behaviour on a website. The unsupervised learning algorithms used in streaming unsupervised machine learning may include clustering, anomaly detection, or density estimation, among others.

## **Anomaly Detection**

According to the Cambridge Dictionary anomaly is a thing that is different from what is usual, or not in agreement with something else and therefore not satisfactory. (Cambridge University Press, 2023)

An anomaly can be identified by clustering data, which gives normal data and noises.



Anomaly Detector is an AI service with an API that allows you to track and detect anomalies in time series data with minimal knowledge of machine learning (batch validation or real-time inference). The main goal of anomaly detection is to discover unexpected events or rare occurrences in data.

Anomaly detection is used in areas such as:

1) Equipment breakdown prediction

So, in 2010, Iranian centrifuges were attacked by the Stuxnet virus, which set a non-optimal mode of equipment operation and disabled part of the equipment due to accelerated wear.

If anomaly search algorithms were used on the equipment, the failure situation could have been avoided.

The search for anomalies in the operation of equipment is used not only in the nuclear industry, but also in metallurgy and the operation of aircraft turbines. And in other areas where the use of predictive diagnostics is cheaper than possible losses in the event of an unpredictable breakdown.

2) Prediction of fraudulent activities

If money is being withdrawn from the card you use in Podolsk in Albania, the transactions may need to be further verified.

3) Identification of abnormal consumer patterns

If some of the clients exhibit anomalous behaviour, there may be a problem that you are not aware of.

4) Identification of abnormal demand and load

If sales in an FMCG store have fallen below the forecast confidence interval, it is worth finding the reason for what is happening.

Searching for anomalies and detection of suspicious transactions is widely used in customer analytics, bank audit and other types of business intelligence. The importance of it is due to the fact that anomalies in data often means critical, actionable information in a wide variety of applications. For example, anomalies in card transaction data could mean credit card or confidential information theft (Aleskerov, 1997), or anomalous readings from a sensor could signify a fault in some component of the equipment. Unusual traffic in a computer network could mean that a hacked computer is sending sensitive data to an unauthorised addressee (Kumar, 2005). An anomalous MRI image may point to the presence of malignant tumours (Spence, 2001).

The essence of this technique is to analyse large amounts of data and identify suppliers, customers, transactions or other activities with extremely abnormal behaviour. Often, such anomalies are an indicator of fraud or a reason for a more detailed analysis of such business activities.

**Russian web-pages:** [**https://habr.com/ru/articles/530574/**](https://habr.com/ru/articles/530574/) **-** Deep Anomaly detection

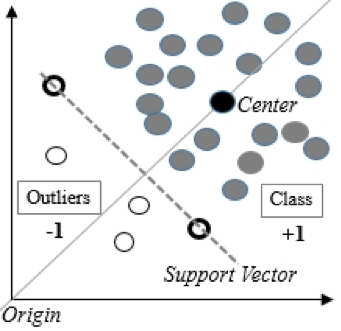
**Anomaly detection approaches -** [**https://habr.com/ru/articles/477450/**](https://habr.com/ru/articles/477450/)

1) Single-Class Support Vector Machine One-Class SVM

It is suitable when the data in the training set follows a normal distribution, and in the test set it contains anomalies.

A single-class support vector machine constructs a non-linear surface around the origin. It is possible to set a cutoff limit for what data to consider as anomalous.

Based on the experience of our DATA4 team, One-Class SVM is the most commonly used algorithm for solving the anomaly search problem.



<https://habr.com/ru/articles/477450/>

## Anomaly Detection using Streaming Unsupervised Machine Learning

Anomaly detection can be performed using both supervised and unsupervised machine learning methods. In supervised anomaly detection, a labelled dataset is used to train the model, where the labelled data includes both normal and anomalous examples. The model learns to distinguish between normal and anomalous data, and is then used to classify new data points as normal or anomalous. Supervised anomaly detection is useful when anomalous examples are rare and the number of normal examples is large, it also helps when there is a clear definition of what constitutes an anomaly. Unsupervised anomaly detection, on the other hand, does not require labelled data. Instead, the model is trained on a dataset of only normal examples, and is then used to identify data points that deviate from the "normal" behaviour. This is useful when anomalous examples are rare and the definition of an anomaly is not clear. In my case studies as the anomaly is not clear I have chosen to use an unsupervised algorithm.

Even so, unsupervised machine learning can solve many issues in a real time and has many advantages to deal with the lack of storage place, there are several problems that appears during anomaly detection using it:

1. Data variability: Unsupervised anomaly detection methods based on determining patterns or characteristics that deviate from the norm. However, the definition of "normal" behaviour can differ depending on the context, and it may be difficult to identify a consistent set of features that are indicative of anomalies.

2. Lack of labelled data: Unsupervised anomaly detection does not require labelled data, but the lack of labelled data can make it difficult to evaluate the performance of the model.

3. High dimensionality: High-dimensional data can make it difficult to identify patterns or features that are indicative of anomalies. This is because the number of possible combinations of features increases exponentially with the number of dimensions.

4. Concept drift: Concept drift refers to the slight change in behaviour of the model over time. This can make it difficult for unsupervised anomaly detection models to keep up with changes in the data, leading to a decrease in performance over time.

5. Scaling: Anomaly detection on large datasets can be computationally intensive and expensive, which can be a challenge for unsupervised anomaly detection in streaming data and big data scenarios when data is generated in large volumes, at high speed, and in high variety.

6. False positive and false negative results. Unsupervised anomaly detection methods can produce false positives, which are normal data points that are misclassified as anomalous, or false negatives, which are anomalous data points that are not detected. These errors can be costly and should be kept to a minimum.

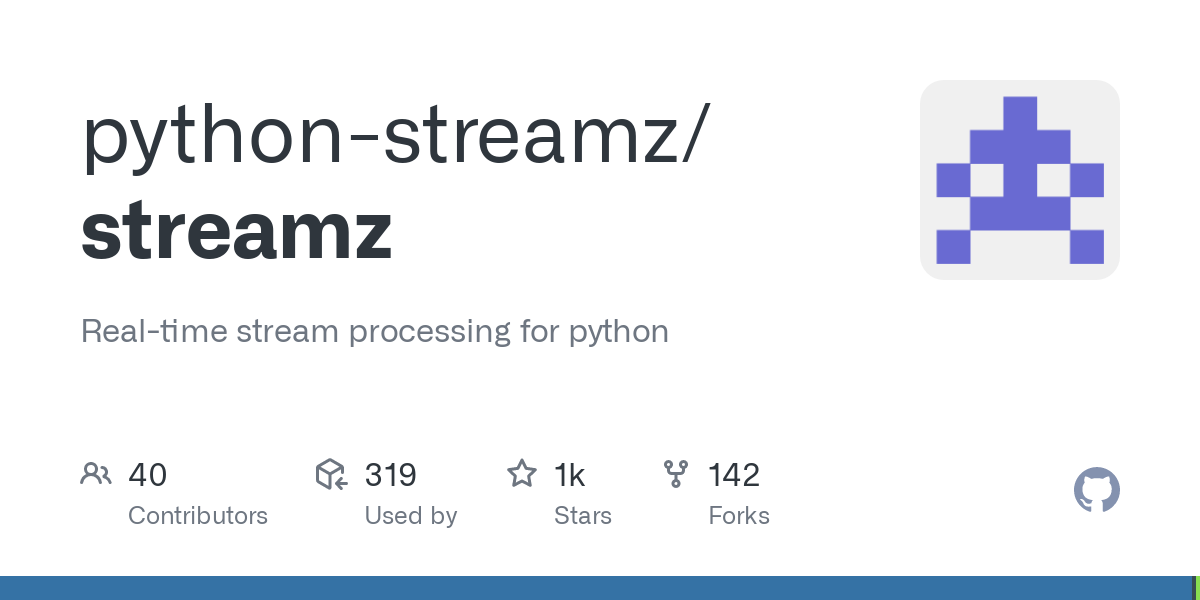
Despite these challenges, unsupervised anomaly detection remains a valuable tool for detecting unusual or anomalous behaviour in data, and can be used in a variety of applications such as network intrusion detection, credit card fraud detection, and manufacturing quality control.

# **PRACTICAL PART**

## **Streamz**

Streamz is a Python library that helps to construct pipelines through which continuous streams of data can flow and be transformed over time. It supports simple as well as complex pipelines that involve branching, joining and control of flow.

It can be used to work with ongoing data streams for machine learning pipelines for real-time and online learning.



## **RiverML**

One of the application tools that allows the user to organise on-the-wing training of an ML model is the River Python library. It's open source, fast, and plays nicely with the rest of the Python ecosystem.

The origin of a data stream can vary, and usually it is not that important. The user should be able to use River regardless of where the data comes from. But it is important to remember the difference between reactive and proactive data streams.

When the data comes to you it is reactive data streams. For example, you do not control when a user visits your website. You have no influence on this event, when it happens you just have to react to it.

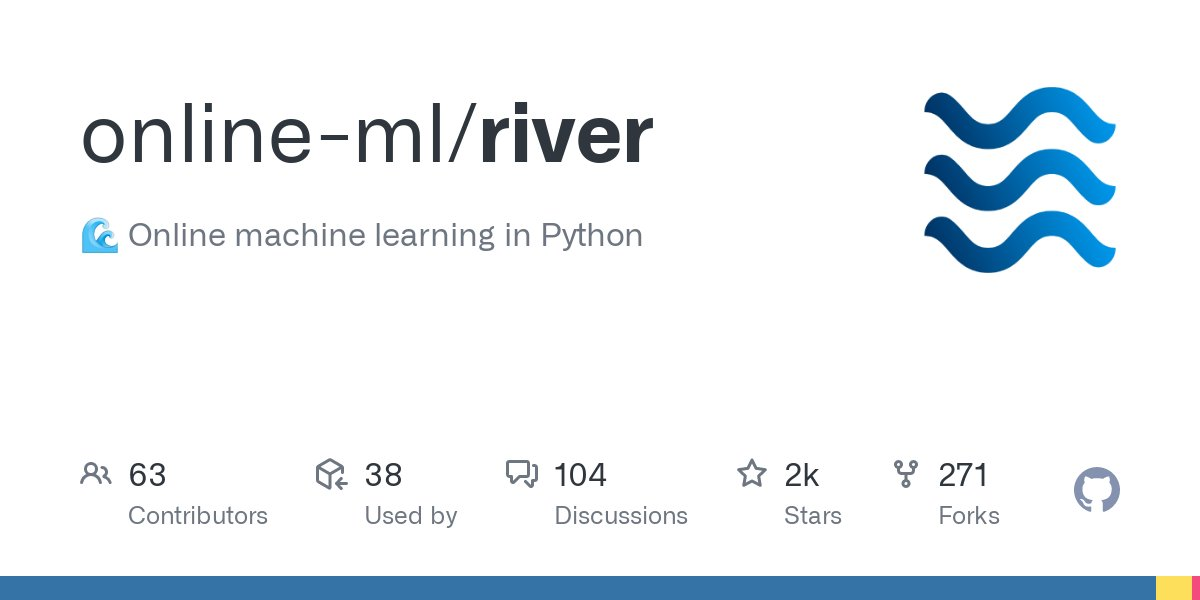
Whereas proactive data streams are ones where the user has control on the data stream. For instance, when you are reading the data from a file, you decide at which velocity you want to read the data, in what order, etc.

The challenge for machine learning is to ensure models which are trained offline on proactive datasets will perform correctly in production on reactive data streams.(Montiel, 2021)

In my thesis work I used online processing, which is the way of processing a data stream one element at a time. In the case of machine learning, it means training a model by teaching one sample at a time. This is the opposite to the traditional way of machine learning, which is to train a model on a whole *batch data* at a time, by feeding a model with a definite amount of data.

That is why an online model is a stateful, dynamic object. During the process it keeps learning and doesn't have to go through the past data. It's a different way of machine learning and therefore has its own set of advantages and disadvantages.

For the online anomaly detection model of the RiverML library I used OneClassSVM in combination with QuantileFilter. The later made precision check shows that this model is quite accurate.

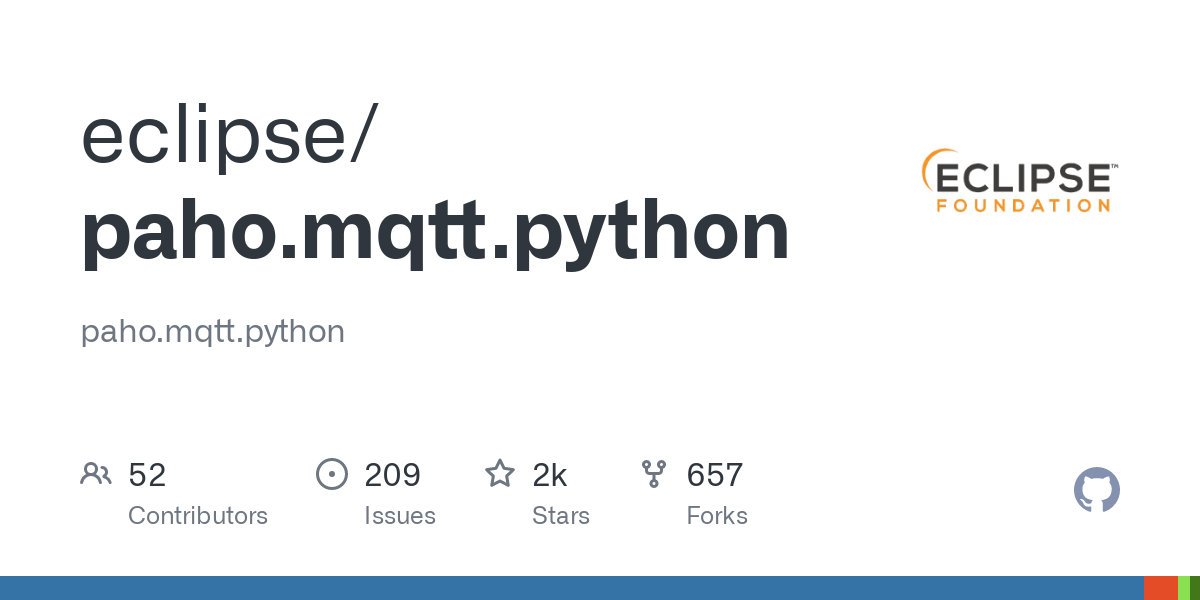


## **Paho-mqtt**

The MQTT protocol is a machine-to-machine (M2M)/”Internet of Things” connection protocol. It is designed as an extremely light publish/subscribe message transportation tool and is used to connect with remote locations.

MQTT (MQ Telemetry Transport) is used for communication with IoT devices over network protocol using Python workspace.(Bender, 2021)

I have used the paho-mqtt library for my case study to connect to the remote devices and get the real time data to analyse anomalies. For example to check if the coffee machine is off or on, and identify the power consumption.

****

## 

## By combining the Streamz and RiverML and sometimes using paho-mqtt libraries I was able to make case studies which show the anomaly detection in use and are represented by metric, graph and schema.

## **Case Study #1**

Credit card frauds.

The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

It contains only numeric input variables which are the result of a PCA transformation.

Unfortunately, due to confidentiality issues, the original features and more background information about the data cannot be provided. Features V1, V2, ... V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependent cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

*Name CreditCard*

*Task Binary classification*

*Samples 284,807*

*Features 30*

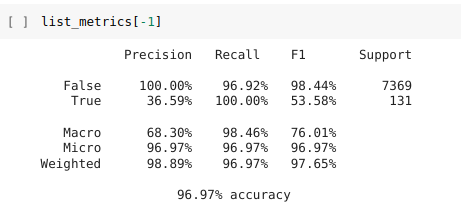
*Sparse False*

*Path /home/dinara/river\_data/CreditCard/creditcard.csv*

*URL https://maxhalford.github.io/files/datasets/creditcardfraud.zip*

*Size 143.84 MB*

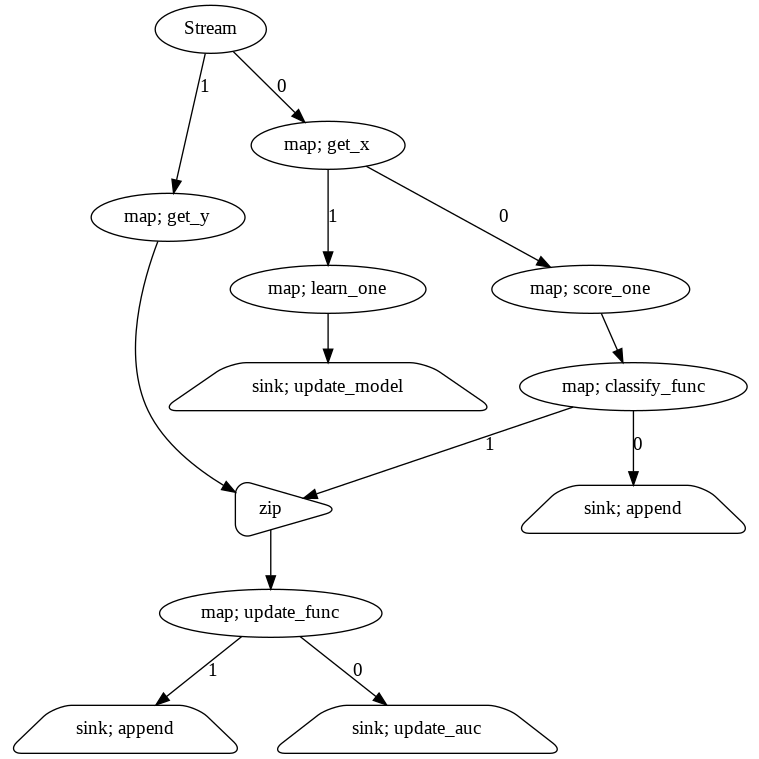
*Downloaded False*



## **Case Study #2**

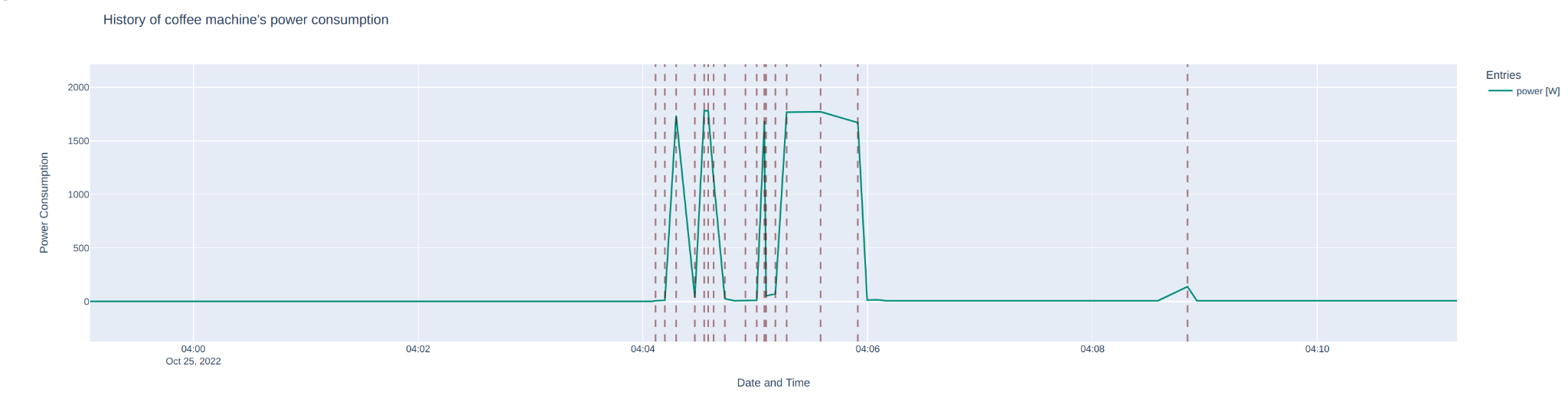
In the second case study I connected Streamz with a real IOT device (a coffee machine) using MQTT network protocol.

Based on their documentation, I combined the *Eclipse Paho* MQTT Python client library, which implements versions 5.0, 3.1.1, and 3.1 of the MQTT protocol and Streamz Python library for processing pipelines. Streamz does not document from\_mqtt function, however, it documents other sources which might come handy. Paho-mqtt is the all running in the background of streamz/from\_mqtt.



Streaming data visualisation of Credit Card and Coffee Machine’s power consumption.

As a metric for anomaly detection F1 is used which is a great universal metric which takes into consideration sensitivity and specificity of your model. In statistical analysis of binary classification, the F-score or F-measure is a measure of a test's accuracy. It is calculated from the precision and recall of the test, where the precision is the number of true positive results divided by the number of all positive results, including those not identified correctly, and the recall is the number of true positive results divided by the number of all samples that should have been identified as positive. Precision is also known as positive predictive value, and recall is also known as sensitivity in diagnostic binary classification. The highest value is 1.0 which indicates a perfect precision and recall, while the lowest is 0 where precision and recall is 0 respectively.

As an output I have extracted the information about the history of the coffee machine’s power consumption. Below is a graph of the history. The anomalies are represented by red dashed lines. It is visible that the anomaly detector reacts not only to big deviations, but even to small and rare events. This shows the importance and precision of our anomaly detector.

## **Case Study #3**

The source of the data is an industrial battery energy storage system installed at the MECOM company in Slovakia. The mecom\_P signal is the power output of the battery. and is expressed in kilowatts.

The third case is in the development process as I have faced some constraints and challenges which will be studied further.

# **CONCLUSION**

Streamed Machine Learning is one of the best approach which can be beneficial in case of storage optimisation and

One problem that a lot of people are worried about is the cloud cost. Model training costs money, so you might think that the more often you train the model, the more expensive it's going to be. That’s actually not always the case for Unsupervised Streamed ML.

By using Streamed Data in ML it is possible to quickly react to the changes in the process and get feedback for improvements which makes the production more agile.

As case studies demonstrate using River ML and Streamz might have

Despite the realisation of the third case with MECOM company’s power anomaly, my further developments are going to be:

* To decrease the possible problems that can appear during anomaly detection using unsupervised machine learning.
* To develop more models applied for different types of input data: to finish third case for MECOM company power.

By leveraging streaming architecture, companies can build a platform to make it easier to do real-time machine learning.

# **LITERATURE**

1. Alteryx, 2023, January 10, *Supervised vs Unsupervised Learning*, <https://www.alteryx.com/glossary/supervised-vs-unsupervised-learning>
2. Bender, M., Kirdan, E., Pahl, M. O., & Carle, G. (2021, January). Open-source mqtt evaluation. In 2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC) (pp. 1-4). IEEE.
3. Cambridge University Press, 2023
4. Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM computing surveys (CSUR), 41(3), 1-58.
5. Chip Huyen, Anthony Alford 2022, August 22, S*treaming-First Infrastructure for Real-Time Machine Learning* <https://www.infoq.com/articles/streaming-first-real-time-ml/>
6. El Naqa, I., & Murphy, M. J. (2015). What is machine learning?. In machine learning in radiation oncology (pp. 3-11). Springer, Cham.
7. Gomes, H. M., Read, J., Bifet, A., Barddal, J. P., & Gama, J. (2019). Machine learning for streaming data: state of the art, challenges, and opportunities. ACM SIGKDD Explorations Newsletter, 21(2), 6-22.
8. Hahne, F., Huber, W., Gentleman, R., Falcon, S., Gentleman, R., & Carey, V. J. (2008). Unsupervised machine learning. Bioconductor case studies, 137-157.
9. Kevin Petrie 2021, November 11, Machine Learning and Streaming Data Pipelines, Part I: Definitions and Architecture <https://www.eckerson.com/articles/machine-learning-and-streaming-data-pipelines-part-i-definitions-and-architecture>
10. Kumar, V. 2005. Parallel and distributed computing for cybersecurity. IEEE Distrib. Syst. Online 6, 10.
11. Montiel, J., Halford, M., Mastelini, S. M., Bolmier, G., Sourty, R., Vaysse, R., ... & Bifet, A. (2021). River: machine learning for streaming data in Python.
12. Paho 2022, December 18, *paho-mqtt 1.6.1* <https://pypi.org/project/paho-mqtt/>
13. River *Basic concepts* <https://riverml.xyz/0.14.0/introduction/basic-concepts/>
14. RiverML *anomaly* <https://riverml.xyz/0.14.0/api/overview/>
15. Shalev-Shwartz, S., Singer, Y., & Ng, A. Y. (2004, July). Online and batch learning of pseudo-metrics. In Proceedings of the twenty-first international conference on Machine learning (p. 94).
16. Singh, A., Thakur, N., & Sharma, A. (2016, March). A review of supervised machine learning algorithms. In 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom) (pp. 1310-1315). Ieee.
17. Streamz *Core Streams* <https://streamz.readthedocs.io/en/latest/core.html>

Addition - explanation of my code

This is a Python script for processing streaming data using anomaly detection. It uses the One-Class SVM algorithm to detect anomalies and the MQTT protocol to receive and publish data. The script defines a class called OneClassSVM that inherits from anomaly.base.AnomalyDetector, which implements the anomaly detection logic. The preprocess function processes the raw input data into a dictionary that includes the timestamp and relevant data columns. The fit\_transform function applies the anomaly detection model to each data point and returns a dictionary that includes the timestamp, anomaly score, and the dynamic threshold value. Finally, the process\_limits\_streaming function sets up the streaming pipeline using the Streamz library and starts the processing. It also handles the signal interrupt to gracefully stop the app and prints a summary of the anomaly detection results.

\*\*\*

This code appears to be a streaming anomaly detection system that reads data from an MQTT broker or a Pandas DataFrame, applies a one-class SVM algorithm to detect anomalies, and then writes the results to an output stream or a file.

The OneClassSVM class is used to initialize the model and defines the learn(), score(), and predict() methods that are used to train the model, score new data, and make predictions respectively. The preprocess() function is used to convert the input data to a dictionary that contains only the relevant columns, while fit\_transform() is used to apply the model to the data and add the anomaly detection results to the dictionary.

The Streamz library is used to create a streaming pipeline that applies the preprocessing and model application functions to the data, and then writes the output to a file or an MQTT broker. Finally, the signal\_handler() function is used to handle the termination of the application and print a summary of the anomaly detection results.

However, there are some undefined variables that could cause errors, such as ".model" in line 56 and the missing line of code in line 31.

\*\*\*

Line 31 defines a class OneClassSVM that inherits from the anomaly.base.AnomalyDetector class. The OneClassSVM class is an implementation of the one-class SVM algorithm for anomaly detection using the river library.

The class has three methods:

* \_\_init\_\_(self, grace\_period=GRACE\_PERIOD): the constructor that initializes the class instance with a grace period of GRACE\_PERIOD minutes (which is set to 1440 by default).
* learn(self, x): updates the one-class SVM model by adding the new observation x.
* score(self, x) -> float: returns the anomaly score of the new observation x using the trained one-class SVM model.

The OneClassSVM class is designed to work with a Stream object in order to process data in a streaming fashion. The process method is not used in this script, but it could be used in a streaming context to process incoming data and detect anomalies.

\*\*\*

he fit\_transform function takes in data x and a model (in this case, an instance of OneClassSVM anomaly detector), and returns a dictionary with two keys: "time" and "anomaly". The "time" key holds the timestamp of the data, and the "anomaly" key holds a boolean value indicating whether the data is considered an anomaly by the anomaly detector.

The function applies the anomaly detector to the incoming data using the process\_one method of the model. This method takes in the data and the timestamp and returns a boolean value indicating whether the data is an anomaly or not. The timestamp is converted to a string before being returned in the output dictionary.

Note that this function assumes that the input data has already been preprocessed using the preprocess function.

\*\*\*

what does preprocess function do?

The preprocess function takes in a data point x and a list of column names col. It returns a dictionary with two keys: "time" and "data".

The "time" key is assigned the timestamp value of the data point converted to UTC timezone and without microseconds. The "data" key is assigned the value of the data point for the column specified by col.

The function checks the type of the input x to determine the appropriate method of extracting the timestamp and data values. If x is a Pandas Series, it extracts the timestamp from the series index and the data value from the column specified by col. If x is a tuple containing a timestamp and a Pandas Series, it extracts the timestamp and data value from the same way. If x is a dictionary, it assumes that the dictionary contains the "data" key and the "time" key with appropriate values. Finally, if x is an MQTTMessage object, it extracts the timestamp from the message timestamp field and the data from the message payload field.

\*\*\*

Hello! It looks like you're trying to import the Stream class from the streamz library in Python.

The Stream class in streamz is a core data structure that allows you to create a stream of data that can be processed in real-time using a series of operations. You can create a Stream object by specifying the source of the data, such as a file, a network socket, or an in-memory buffer.

Here's an example of how you can use the Stream class to read data from a file:

from streamz import Stream

# Create a stream object that reads lines from a file

stream = Stream.from\_textfile('data.txt')

# Define a processing function that prints each line

def process(line):

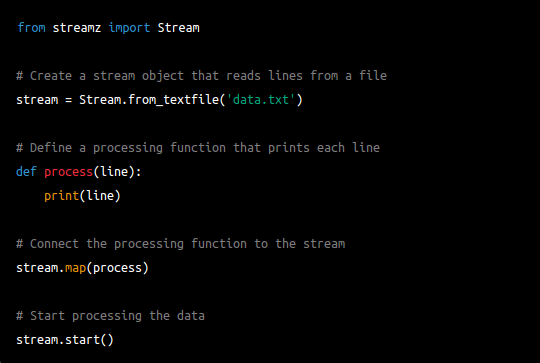
print(line)

# Connect the processing function to the stream

stream.map(process)

# Start processing the data

stream.start()

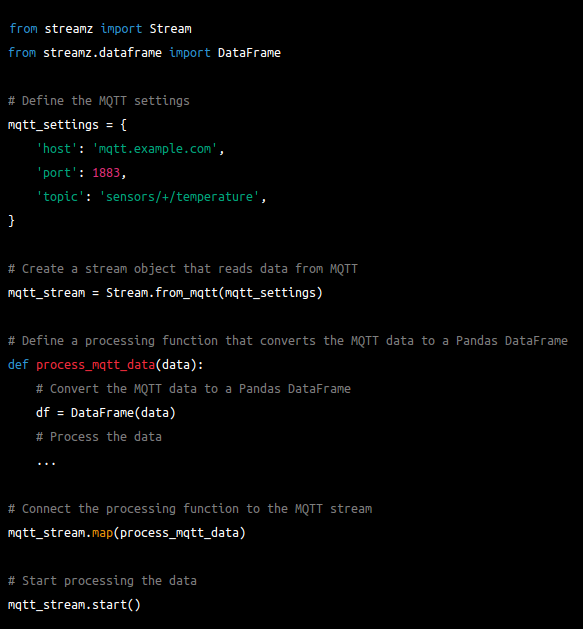


In this example, the Stream.from\_textfile() method creates a Stream object that reads lines from the data.txt file. The stream.map() method then connects a processing function, process(), to the stream. Finally, the stream.start() method starts processing the data in real-time.

I hope that helps! Let me know if you have any further questions.

\*\*\*

there is a from\_mqtt function in the streamz library. This function allows you to create a stream of data from an MQTT broker. Here's an example of how you can use the from\_mqtt function:



from streamz import Stream from streamz.dataframe import DataFrame # Define the MQTT settings mqtt\_settings = { 'host': 'mqtt.example.com', 'port': 1883, 'topic': 'sensors/+/temperature', } # Create a stream object that reads data from MQTT mqtt\_stream = Stream.from\_mqtt(mqtt\_settings) # Define a processing function that converts the MQTT data to a Pandas DataFrame def process\_mqtt\_data(data): # Convert the MQTT data to a Pandas DataFrame df = DataFrame(data) # Process the data ... # Connect the processing function to the MQTT stream mqtt\_stream.map(process\_mqtt\_data) # Start processing the data mqtt\_stream.start()

In this example, the from\_mqtt function creates a stream that reads data from the sensors/+/temperature topic on an MQTT broker hosted at mqtt.example.com on port 1883. The mqtt\_stream.map() method connects a processing function, process\_mqtt\_data(), to the MQTT stream. Finally, the mqtt\_stream.start() method starts processing the data in real-time.

I hope that helps! Let me know if you have any further questions.

\*\*\*\*

Main

the code defines a function process\_limits\_streaming that takes two arguments, topic and data. The purpose of this function is to detect anomalies in a streaming fashion based on the input data.

If data is a string, it assumes that it is a topic to which it subscribes for MQTT messages. If data is a pandas DataFrame, it assumes that it contains the data directly. If data is not a string or a DataFrame, it raises a RuntimeError.

The function starts by initializing an anomaly detection model called OneClassSVM using the anomaly module from the river package. The OneClassSVM model is a type of one-class classification algorithm that can be used for anomaly detection. The model is initialized with a grace period of 60 \* 24 (1440) samples, which means that the first 1440 samples will not be considered as anomalies.

Then, the input data is converted into a Stream using the Stream.from\_mqtt or Stream.from\_iterable method, depending on the type of the input data. The data is then preprocessed using the preprocess function, which extracts the relevant fields from the data and returns them in a dictionary.

The preprocessed data is then passed to the fit\_transform function, which applies the anomaly detection model to each data point and returns a dictionary containing the time stamp and the anomaly score.

Finally, the output of the fit\_transform function is either published to an MQTT topic using the to\_mqtt method of the Stream or written to a file using the dump\_to\_file function.

MAIN

\*\*\*\*

This is a Python script that uses Streamlit to build an anomaly detection dashboard. The script starts by importing necessary libraries, including pandas, json, os, and streamlit. The MyOneClassSVM class is defined with an \_\_init\_\_ function that takes a grace\_period argument, which sets the number of samples that the model will learn from before making predictions. The class also has a learn function that takes in a data point and trains the model on that data point, a score function that calculates the anomaly score for a given data point, a predict function that returns a binary classification (anomaly or not) for a given data point, and a process function that calls predict and learn functions to update the model with new data points.

The preprocess function is used to prepare the incoming data for the anomaly detection model by extracting the timestamp and the value from the input data. The fit\_transform function takes in a preprocessed data point and the anomaly detection model and applies the model's process method to the data point, returning a dictionary with the timestamp and the anomaly score. The dump\_to\_file function is used to write the output data to a file.

The process\_limits\_streaming function takes in a data source (either a dictionary with MQTT settings or a Pandas DataFrame) and returns a Stream object that applies the anomaly detection model to the data source. The signal\_handler function is used to stop the data source when the user terminates the application.

In the main body of the script, the user can select the data type (either MQTT topic or JSON/CSV file) and enter the necessary settings for the data source. If the user selects a JSON or CSV file, the script reads the file into a Pandas DataFrame and applies the process\_limits\_streaming function to the DataFrame. If the user selects an MQTT topic, the script sets up a connection to the MQTT broker and applies the process\_limits\_streaming function to the MQTT settings. Finally, the script creates a line chart to display the output data.

\*\*\*\*