POLITECNICO DI TORINO

Master degree course in Computer Engineering

Master Degree Thesis

Data Analytics For Predictive Maintenance

Supervisor:
Prof. Tania Cerquitelli

Candidate:
HEMA BHANDARI

April 2020

Acknowledgments

First and foremost I am highly grateful and indebted to my supervisor, Prof. Tania Cerquitelli, who was kind enough to let me pursue the thesis at Politecnico Di Torino for my master's studies.

I'm really thankful to her for her patience and her helpful comments and observations.

I would also like to extend my gratitude to Francesco Ventura for supporting and guiding me all along. The motivation and encouragement were like the beacon of light, enlightening my path with wisdom and knowledge. It would not be an exaggeration to say, that without the support I would not have been able to complete my thesis.

Moreover, I would like to thank all the other people of Lab5 for the welcoming working environment.

And lastly, a huge thanks goes to my family that supported me in all those years that brought me at this point.

Abstract

Predictive maintenance is a technique that tries to predict imminent problems, forecast future failures and discovers criticalities when a piece of equipment might stop working so that proactive strategies can be applied just before that happens. These predictions can be done on the basis of equipment's condition which is estimated based on data collected with the help of condition monitoring sensors and strategies.

To this aim, the predictive analytics has been measured to predict the belt tensioning level and to further support robot cycle labelling. A machine learning algorithm has been applied to smart data so as to forecast a tensioning level as per a new cycle of data. This helped in identifying clusters of production cycles through similar time independent features.

The best configuration has been selected after comparing various supervised and unsupervised metrics on different clustering algorithms.

And lastly, we talked about data distribution of patterns i.e. cluster characterization. The most relevant features of our clusters has been extracted. Cluster characterization helps domain expert in labelling of the data, in our case to robot cycles.

Keywords: Predictive maintenance, machine learning, clustering algorithms, Cluster characterization

Contents

List of Figures **5**

List of Tables **6**

1 Introduction **7**

2 Related Work **9**
2.1 Industry 4.0 . . . 9
2.1.1 Smart Data . . . 10
2.1.2 Experimental setting and Data exploration . . . 10
2.2 Types of predictive models . . . 13
2.2.1 Supervised learning . . . 14
2.2.2 Unsupervised learning . . . 16
2.2.3 Semi- supervised learning . . . 17

3 Proposed Approach **18**
3.1 Overview . . . 18
3.2 Clustering . . . 21
3.3 Clustering Algorithms . . . 23
3.3.1 k-means . . . 23
3.3.2 DBSCAN . . . 25
3.3.3 Agglomerative clustering . . . 28
3.4 Clustering Validation . . . 32
3.4.1 Supervised indices . . . 32
3.4.2 Unsupervised indices . . . 35
3.5 Cluster Characterization . . . 38
3.5.1 Decision Tree classifier . . . 40

4 Implementation Tools **42**
4.1 Python . . . 42
4.2 Pycharm . . . 43
4.3 Libraries . . . 44
4.3.1 Numpy . . . 44
4.3.2 Pandas . . . 45
4.3.3 JSON . . . 46
4.3.4 MatplotLib . . . 46

4.3.5 Scikit-Learn 47
4.3.6 Joblib 47
4.4 Google Colaboratory 48

5 Experimental results **50**
5.1 Model creation 50
5.2 Cluster analysis 53
5.2.1 Unsupervised metrics comparison 53
5.2.2 Supervised metrics comparison 61
5.2.3 Best Configuration Selection 66
5.3 Cluster characterization 69
5.3.1 Feature selection 69
5.3.2 Interpreting box plots 70

6 Conclusion and Future improvements **79**

List of Figures

1 Washers per class . . . 11
2 Electric signal of a single robot cycle . . . 12
3 Time domain feature computation . . . 13
4 Different machine learning approaches . . . 14
5 Application architecture . . . 18
6 The predictive analytics service . . . 19
7 Data analytics architecture . . . 20
8 Data analytics processes . . . 20
9 Clustering architechture . . . 22
10 DBSCAN clustering with min-points 4 . . . 27
11 An example of Hierarchical clustering . . . 29
12 Cluster charcterization architecture . . . 39
13 An example of Decision Tree Classifier . . . 41
14 An example with Minimum points 10 and 12 . . . 52
15 Silhouette scores vs n_clusters (k) . . . 54
19 CalinskiHarabasz scores vs n_clusters (k) . . . 58
20 Sum of Squared errors (SSE) vs n_clusters (k) . . . 59
21 Sum of Squared errors (SSE) for k-means . . . 60
22 Adjusted rand scores vs n_clusters (k) . . . 62
23 Homogeneity scores vs n_clusters (k) . . . 63
24 Completeness scores vs n_clusters (k) . . . 64
25 V-measure scores vs n_clusters (k) . . . 65
28 Boxplot for k-means (k=5) . . . 71
29 Outlier cluster for k-means (k=5) . . . 72
30 Boxplot for k-means (k=6) . . . 73
31 Outlier cluster for k-means (k=6) . . . 73
32 Boxplot for Agglomerative-Ward (k=5) . . . 75
33 Outlier cluster for Agglomerative-Ward (k=5) . . . 75
34 Boxplot for Agglomerative-Ward (k=6) . . . 77
35 Outlier cluster for Agglomerative-Ward (k=6) . . . 77
36 Electric signal with relevant segments highlighted . . . 78

List of Tables

1 Number of cycles per class (NumWashers) 12
2 Supervised vs Unsupervised indices 32
3 Model creation 50
4 Dbscan for eps 13 and 14 51
5 Dbscan for various metrics 53
6 Silhouette scores of all algorithms 54
7 Calinski-Harabasz scores of all algorithms 57
8 Sum of Squared errors (SSE) of all algorithms 59
9 Adjusted rand scores of all algorithms 61
10 Homogeneity scores of all algorithms 63
11 Completeness scores of all algorithms 64
12 V-measure scores of all algorithms 65
13 Best scores of supervised and unsupervised metrics 67
14 Total points in each cluster 68
15 Top 10 most relevant features (IDs) for both algorithms . . . 70
16 Top 10 most relevant features for k-means (k=5) 71
17 Top 10 most relevant features for k-means (k=6) 72
18 Top 10 most relevant features for Agglomerative-Ward (k=5) 74
19 Top 10 most relevant features for Agglomerative-Ward (k=6) 76

1 Introduction

The advent of Industry 4.0 trend in data exchange and automation, leads to evolution towards smart environments, including a thorough utilization of Cyber-Physical System (CPS). Robust cyber-physical architectures are becoming essential to analyze such huge amount of data, creating insight into the process of production, and thus enabling improvement and competitive business advantages.

Predictive maintenance helps in identifying imminent problems, forecast future failures and discovers criticalities when a piece of equipment might stop working so that proactive strategies can be applied just before that happens. These predictions can be done on the basis of equipment's condition which is estimated based on data collected with the help of condition monitoring sensors and strategies.

The main objective of this study is to implement the predictive maintenance on the Robot Box. The predictive analytics has been measured to predict the belt tensioning level and to further support robot cycle labelling. A machine learning algorithm has been applied to smart data so as to forecast a tensioning level as per a new cycle of data. This helped in identifying clusters of production cycles through similar time independent features. The goal is to forecast correct configuration of parameters and clustering algorithms through the estimation of cluster quality metrics. The best configuration has been selected after comparing various supervised and unsupervised metrics on different clustering algorithms. And lastly, the most relevant features of our clusters has been extracted to characterize the clusters. Cluster characterization helps domain expert in labelling of the data, in our case to robot cycles.

The experiments have been done by using semi- supervised clustering algorithms in Python language on a dataset imported as JSON file. Also, Google Colaboratory notebooks and Pycharm framework both have been used equally.

The work is organized as follows.

Chapter 2 discusses the related work focusing on Industry 4.0. The smart data is extracted from time domain feature computation. Also, we will see various well-differentiated techniques in Machine Learning approaches.

In chapter 3, we will talk about the proposed approach i.e. architecture for predictive analysis. Also, several implementation steps are discussed. The data pre-processing phase consists of extracting smart data. This phase is important to understand correct initial parameters for further analysis and therefore it lies on top of analysis phase. The data-aggregation phase discusses about various machine learning algorithms like K-Means, DbScan and Agglomerative. The outputs of clustering algorithms have been evaluated by the clustering validation phase and finally Cluster characterization phase talks about data distribution of patters.

Chapter 4 provides information on the implementation tools used to develop this work while chapter 5 discusses the preliminary results obtained by previously mentioned methodologies.

Lastly, chapter 6 draws conclusions and lists possible future developments for predictive analytics in Industry 4.0.

2 Related Work

2.1 Industry 4.0

Due to the huge amount of data generated by modern and connected industries, the essentiality of reliable and powerful architectures is becoming more prominent.

The advent of Industry 4.0 trend in data exchange and automation, leads to evolution towards smart environments, including a thorough utilization of Cyber-Physical System (CPS). Robust cyber-physical architectures are becoming essential to analyze such huge amount of data, creating insight into the process of production, and thus enabling improvement and competitive business advantages.

The authors in [1] presented a cloud-architecture designed for the Industry 4.0 vision. It bridges the gap between the cyber space and the real world, which provides raw data. The cyber space processes the raw data and creates insight, which further aims to enable predictive analytics at the edge. This data analytics architecture targets to anticipate failures and estimate the remaining useful life (RUL) of physical equipment. It can identify the symptoms of approaching machine failure at any given time, through the features of the current dynamics of the machine using on-line data collected. The work in [3] presented a deep-belief network ensemble method with different objectives to estimate remaining useful life.

The authors in [2] presented an integrated self-tuning engine for predictive maintenance in Industry 4.0. Specifically, a distributed architecture which is based on Spark Streaming, Apache Kafka, Cassandra and MLlib was proposed and discussed. This approach integrated the monitoring and prediction task along with a self-tuning approach which dynamically selects the best predictive algorithm and also provides interpretable knowledge to end users.

Advanced ICT technologies and Internet of Things (IoT) allow linking machines and physical manufacturing facilities in integrated applications. The work in [4], presented a predictive maintenance approach involving wide IoT capabilities and cyber-physical systems with complex event processing features.

Articles [4] and [5], utilizes Big Data frameworks to handle with these modern industrial scenarios. In [5], discussed techniques to improve health monitoring services using Big Data frameworks, which is also useful for var-

ious application on an aerospace and aviation industrial. The authors in [2], takes advantage of Big Data technologies and systems (like Apache spark, kafka) and runs on top containerized Docker environment. Article [6] describes the potential of big data analytics in healthcare. The author discussed the benefits and challenges, described examples reported in the literature and outlined an architectural framework and methodology.

2.1.1 Smart Data

The massive data generated and captured by smart devices and appliances contains important and valuable information that is useful in better decision making and facilitating timely actions. In this study we focus on the belt tensioning problem of a motor. The tensioning of the belt is necessary to assure the correct functionality of a robot. Real data is collected by monitoring a motor from a medium sized robot. During data collection phase, a cycle has been collected every 120 seconds. A cycle is the sequence of moves that the motor has been coded to run in loop (in our case the cycle lasts for 24 seconds)

The smart data includes relevant static features from the raw data (in this case raw data are time series), thus supporting the predictive maintenance vision. Smart data represents the main characteristics of the raw data. It also tells about the context information for example, how the data was collected, and the operating conditions of the equipment it was collected from. The model computes a huge variety of statistical indices including mean, standard deviation, maximum, minimum, inter-quartile ranges, root mean square, variance, peak to peak distance, kurtosis and skewness.

From the current raw data, 12 statistical features have been computed and used to classify each cycle independently.

These features have proved to be effective and relevant to model a current cycle.

2.1.2 Experimental setting and Data exploration

The target is to implement the predictive maintenance on the Robot Box. This box has one electric motor, a belt, a gearbox, a mechanical structure and an encoder. The acquired signals are angular position of the axes, electric current of the motor and vibration data from one external sensor.

In a predictive maintenance perspective two major motor failures have

been defined. One is incorrect belt tensioning and the other is backlash. In this study we will focus on the belt tensioning issue.

To study the belt tensioning phenomenon with a machine learning approach, six levels of tensioning have been defined. Each data point contains the information provided from the Robot Box controller and from the user setting connected to the choice of the belt tensioning degree. The later includes time series data (position and current data), header information (program number, machine id, cycle start time, cycle time) and label (level of belt tensioning)

The tensioning of the belt is calculated by the number of washers used to tension it. The effect of different belt tensions can be extracted from the measured current. The washers have been used to discretize the belt tensioning levels. A lower belt tensioning consumption corresponds to a higher number of washers. Fig 1 shows different number of washer corresponding to a different label i.e. divided by class, which means it has the ground truth knowledge. The goal is to predict the correct number of washer corresponding to the suitable belt tensioning for each incoming cycle.

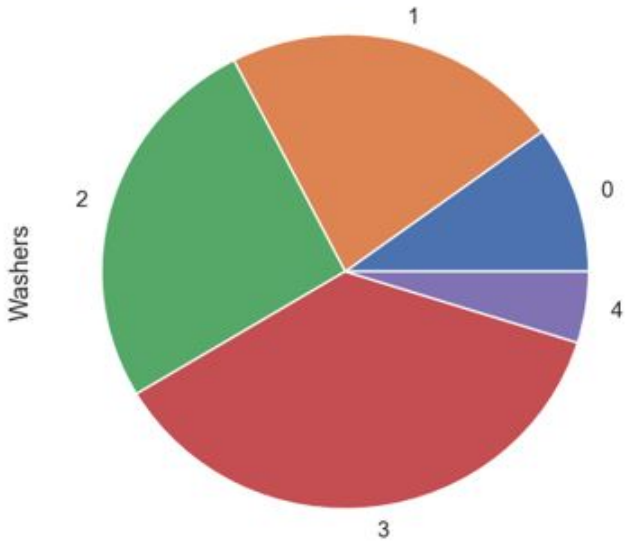


Figure 1: Washers per class

NumWashers	Number of cycles	% over the complete dataset
0	2,392	10.03
1	5,367	22.52
2	6,212	26.06
3	8,707	36.53
4	1,155	4.85
	23,833	

Table 1: Number of cycles per class (NumWashers)

Table 1 shows number of cycles (robot cycles) distribution with respect to washer number i.e. each washer represents label. The tensioning of the belt is important to assure the correct functioning of the robot. An incorrect configuration can cause malfunctioning of the system. Low tension causes overheating, slippage and premature wear of the pulley and belt. Too much tension causes excessive strain on shafts, belts and bearings. There is a vast range of tensions that guaranties a good functioning. The loss of tension occurs on all of the belts. Loss of tension goes from 50% to 70% with respect to the original tension.

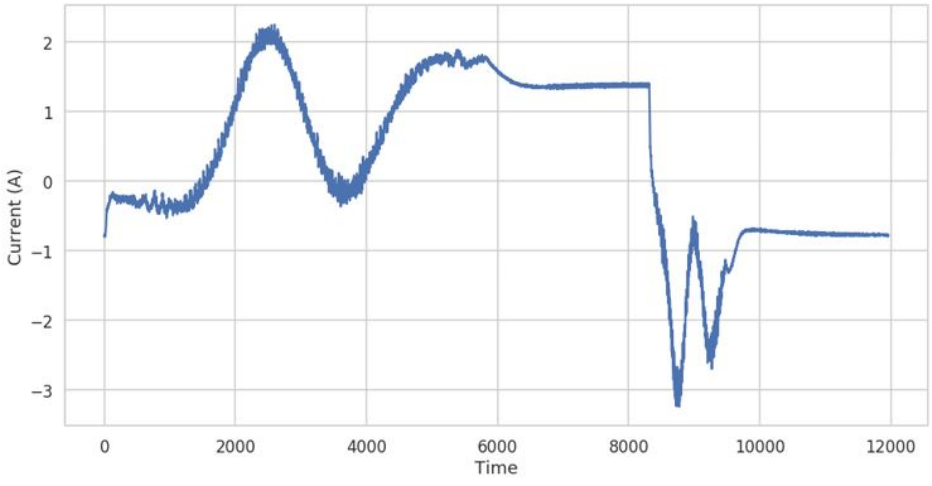


Figure 2: Electric signal of a single robot cycle

Fig 2 shows electric signal of a single robot cycle. It tells the electricity consumption trend for various washer numbers required for belt tensioning.

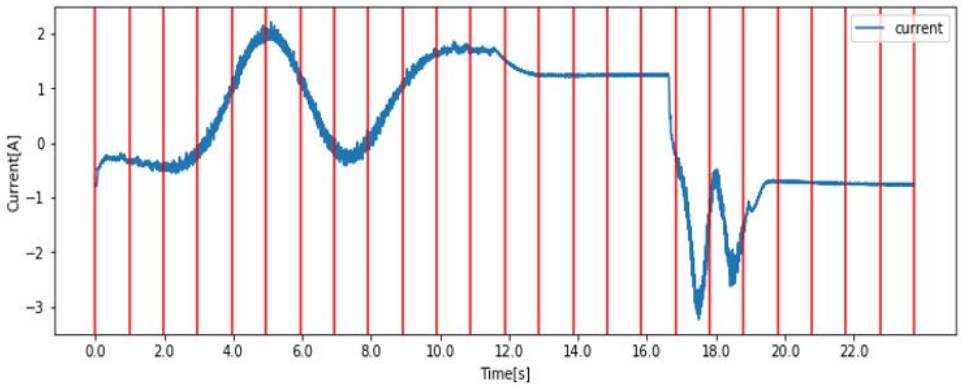


Figure 3: Time domain feature computation

Fig 3 shows that the smart data is extracted from time domain feature computation. The effect of different belt tensions can be extracted from the measured current.

2.2 Types of predictive models

Predictive modeling is the concept of creating a model which is capable of making predictions. This model uses machine learning algorithm to learn certain characteristics from a training data set in order to make those predictions.

In Machine Learning approaches, we can see various well-differentiated techniques (as shown in Figure 4).

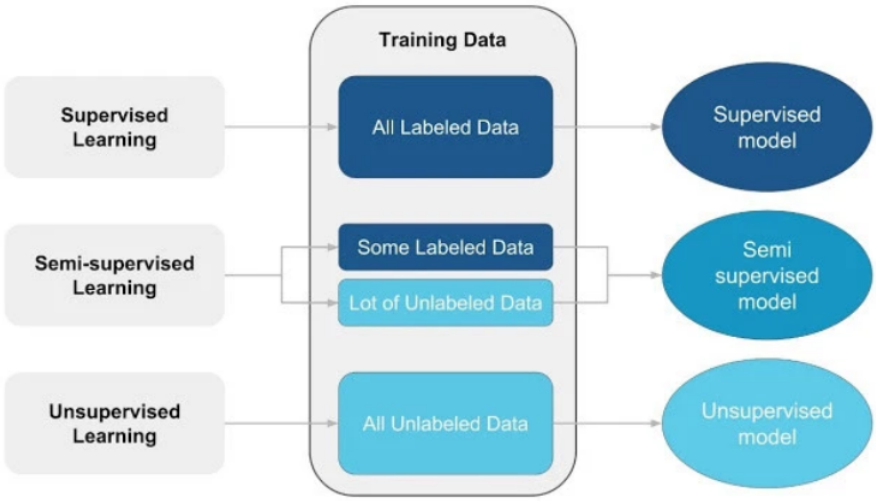


Figure 4: Different machine learning approaches

2.2.1 Supervised learning

In supervised learning, the class labels in the dataset are known, which is used to build the classification model. For instance, a dataset for spam filtering would have spam messages as well as not-spam (="ham") messages. In a supervised learning, we would already know which message in the training set is ham or spam, and we would use this knowledge to train our model and eventually classify new unseen messages.

Supervised learning is used by majority of practical machine learning. For example, we have input variables (x) and one output variable (Y) and we apply an algorithm to learn the mapping function from the input to the output.

$$Y = F(x)$$

When we have new input variables (x), we need to predict the output variable (Y) for that data so the objective is to approximate the mapping function very well.

The process of a technique learning from the training data set is like a teacher supervising the learning process and therefore this process is called

supervised learning. We know the right answers and the algorithm repetitively makes predictions on the training data set and is then corrected by the teacher. When the algorithm reaches an acceptable level of performance, learning stops.

Supervised learning problems can be grouped into broad categories: classification and regression problems.

- Classification problem: when the output variable (Y) is a category, for example "green" or "brown" or "Yes" and "No".
- Regression problem: when the output variable (Y) is a real value, for example "height" or "euros".

Some types of problems are on top of regression and classification include time series and recommendation prediction respectively. Some examples of supervised machine learning algorithms:

- Linear regression for regression problems.
- Support vector machines for classification problems.
- Random forest for regression and classification problems.
- Artificial neural networks (ANN).
- Decision trees
- Random forests.

There are a few things that should be considered before choosing a supervised learning algorithm. The first is the variance and bias that exists within the algorithm since there is a fine line between being too flexible and flexible enough. The second is the complexity of the function or model that the system is trying to learn. Additionally, the accuracy, heterogeneity, linearity and redundancy of the data should be analyzed before selecting an algorithm.

2.2.2 Unsupervised learning

In unsupervised learning, the job deals with unlabeled instances. The classes have to make decisions from the unstructured dataset. Typically, such a learning employs a clustering technique to group the unlabeled instances based on certain similarity measures (or distance). It can be seen as a problem where we only have input data variables (x) and no corresponding output variable.

The objective for unsupervised learning is to model the underlying distribution or structure in the data in order to learn even more about the data.

These problems are called unsupervised learning because there is no teacher and no correct answers unlike supervised learning above. Algorithms have to come up with their own to discover and present the relevant structure in the data.

Unsupervised learning problems can be grouped into broad categories: clustering and association problems.

- Clustering problem: when we want to find out the inherent groupings in the data, for example grouping customers by their purchasing behavior.
- Association problem: It is an association rule learning problem where we want to discover rules that can describe huge portions of our data, for example people that buy product X also tend to buy product Y.

Some examples of unsupervised learning algorithms:

- Apriori algorithm for association rule learning problems.
- K-means for clustering problems.

There are some advantages of supervised learning models over the unsupervised algorithms, but they also have limitations.

The models are more likely to make predictions that humans can relate to, for instance, because humans have given the basis for decisions. However, in retrieval-based method, supervised learning models have issues dealing with new information. For example, if a system with categories for trucks and cars is presented with a bicycle, it would have to be incorrectly merged in one category or the other.

However, if the AI system was powerful, it may not know how the bicycle is but would be able to identify it as belonging to a separate category.

2.2.3 Semi- supervised learning

In semi-supervised learning, some data is labeled and some is unlabeled. Typically, most of it is not labeled. It is a mixture of supervised and unsupervised techniques.

It is a problem when we have a huge amount of input data variables (x) and only few labeled output variable (Y). This problem lies between supervised and unsupervised learning. For example, a photo archive where only few images are labeled, (ex. cat, dog, person) and most of them are unlabeled.

Many real world machine learning problems are semi-supervised learning problems. It is because it can be time-consuming or expensive to label data as it might require access to domain experts. Whereas unlabeled data is cheap. It can be easily collected and stored.

We can use unsupervised learning algorithms to discover and learn the structure in the new input variables. We can also use supervised learning algorithms to make best predictions for the unlabeled data set, feed that data back into the supervised learning algorithm as training data set and use the model to make decisions on new unseen data.

In this study, we will be dealing with semi-supervised learning, using both labeled and unlabeled data.

3 Proposed Approach

3.1 Overview

Industrial robots are deployed around the world and therefore there has been an increasing demand to collect the data that monitors the health status of its machines and avoid sudden failure. To handle this complexity, learning about predictive maintenance approaches is important.

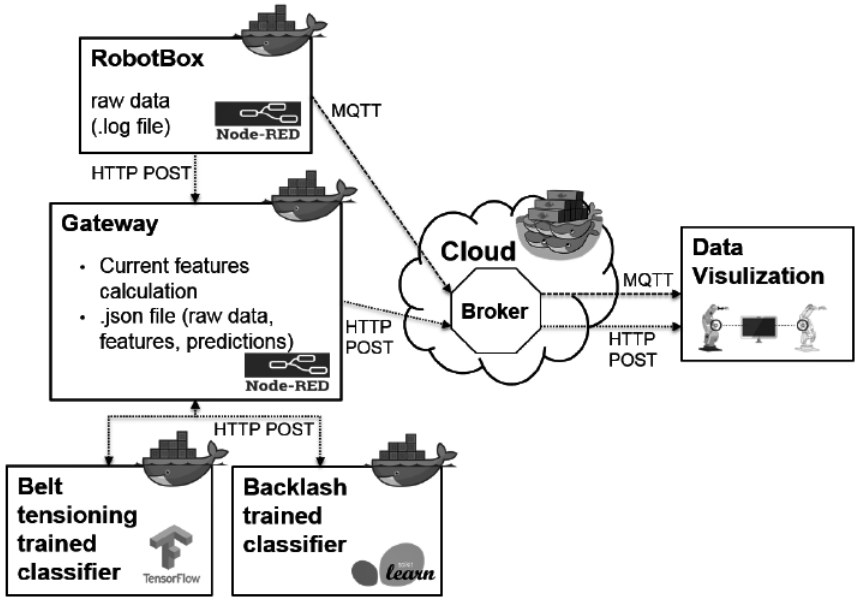


Figure 5: Application architecture

Fig 5 shows a prototype of the application architecture. RobotBox controller acquires position and current data and transmits to the Gateway in a JSON format i.e. log files. Then, the Gateway calculates some statistical characteristics i.e. smart data of the current time series. After that the Gateway communicates with other two services to obtain classification information about the RobotBox cycle. First service is neural network classifier which is able to identify the belt tensioning level. Another is a classifier capable of providing a qualitative backlash status and rough information of the remaining useful life expressed in number of days.

At the end, all the data i.e. the output of the classifiers and the current features are sent to the Borker service which is running on the cloud. The Node-RED service inside the RobotBox controller, makes it possible to run each block which implements the necessary functionalities remotely as the operator has just to launch or connect the flow. The architecture uses NoSQL database as a cloud storage layer.

HTTP and real time feed (MQTT) service has been deployed in a Docker container for real time data streaming. This is located in the RobotBox controller and it sends data to a MQTT broker in the cloud and this is how data is made available to the visualization application.

The predictive analytics has been measured to predict the belt tensioning level. A machine learning algorithm has been applied to smart data so as to forecast a tensioning level as per a new cycle of data.

A predictive analytics service is used to forecast future failures of equipment based on machine learning Techniques.

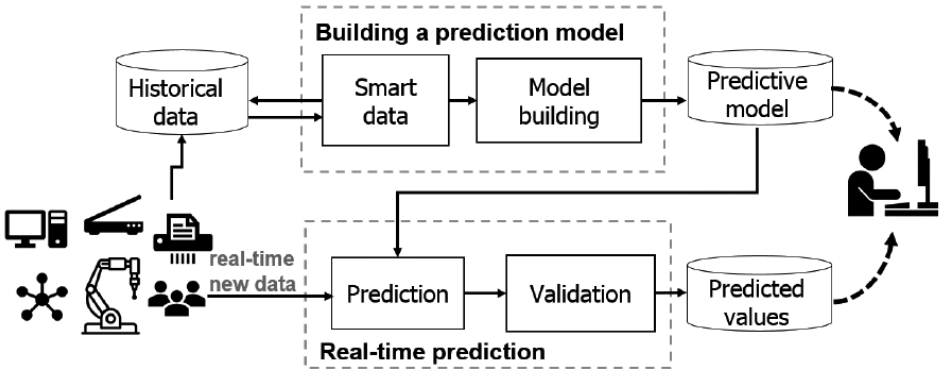


Figure 6: The predictive analytics service

The functional building blocks of the service is shown in fig 6. Based on historical data, a prediction model is built by means of machine learning algorithms and applying this model in real time to new incoming data-streams, to recognize possible failures.

The smart data block includes relevant static features from the raw data (in this case raw data are time series), thus supporting the predictive maintenance vision. Smart data represents the main characteristics of the raw

data. It also tells about the context information for example, how the data was collected, the operating conditions of the equipment it was collected from.

The model building block is executed on historical data. These data include the smart data features computed over the real time series and their respective class labels (e.g., failure absence or presence, category of failures). The validation block evaluates the performance of the prediction block. This block gives a 3D view of the relevant equipment by using the data collected at the field. It also provides the results of the predictive maintenance techniques.

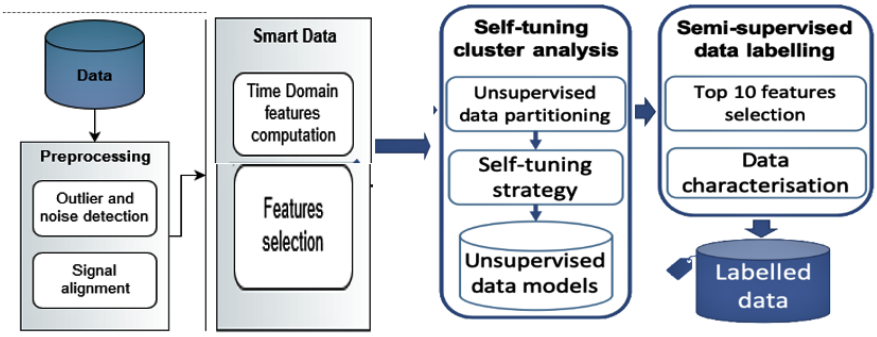


Figure 7: Data analytics architecture

As already discussed, in this study we focus on the belt tensioning problem of a motor. The tensioning of the belt is necessary to assure the correct functionality of a robot. Fig 7 shows the data analytics architecture of the overall process of this study.

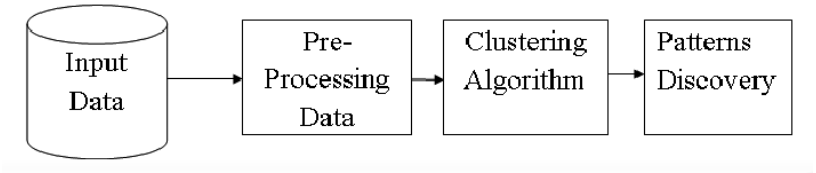


Figure 8: Data analytics processes

The goal of this study is therefore focused on the subsequent processes (as shown in figure 8):

- Data pre-processing
- Smart Data computation
- Clustering
- Cluster validation
- Top 10 feature selection
- Data characterization

3.2 Clustering

Clustering is a type of unsupervised learning method . An unsupervised learning is a method in which we pull out references from data sets comprised of unlabeled input data. Typically, it is a process to recognize meaningful structure, generative characteristics, explanatory underlying processes, and assembling inherent in a set of examples.

It is the process of dividing the data points into several groups such that points in the same groups are more identical to other points in the same group and different to the points in other groups. It is actually a collection of objects based on similarity and dissimilarity between them.

Why Clustering?
Clustering [17] is important as it finds out the inherent grouping among the without labeled data present. There is no specific standard for a good clustering. It totally depends on the end user: what are the norms they can use which satisfy their demands. For example, we want to find the representatives for the homogeneous groups i.e. data reduction, in identifying real clusters and mention their unknown features i.e. natural data types, in identifying important and relevant groupings i.e. useful data classes, or in identifying abnormal data objects (i.e. outlier detection). This method should make some assumptions which account for the similarity of data points. Each and every assumption should make different and well-founded clusters.

In this study, I have applied many different types of clustering algorithms. In Fig 9, I have mentioned a small architecture of algorithms used in my work.

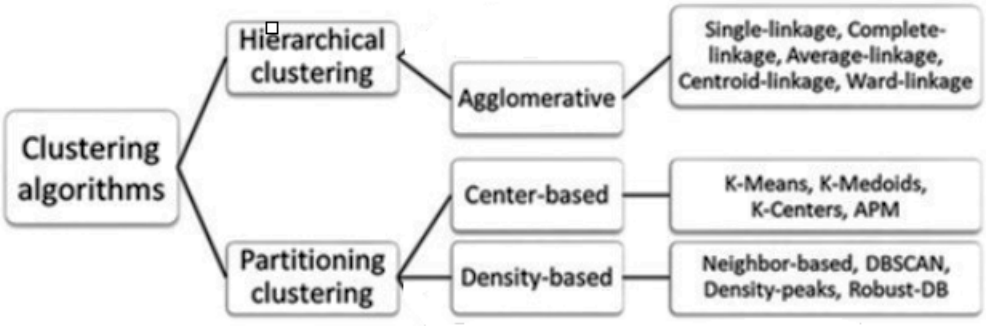


Figure 9: Clustering architechture

Types of clustering methods:

1. Partitioning Methods
2. Grid-based Methods
3. Density-Based Methods
4. Hierarchical Based Methods

Hierarchical Based Methods is further divided into two categories:

1. Agglomerative (bottom up approach)
2. Divisive (top down approach)

Applications of Clustering in various fields:

1. Marketing - It is used to discover & characterize user segments for marketing goals.
2. Biology - It is used for classification among different kinds of animals and plants.
3. Insurance - It is used to acknowledge the users, identifying their policies and recognizing the frauds.

4. Earthquake studies – It can be used to learn about the areas affected by earthquake and find out the dangerous zones.

5. City Planning - It can be used in making groups of houses and study their rates on the basis of their geographical locality and few other factors.

3.3 Clustering Algorithms

There are many types of clustering Algorithms. Here I would like the most well-known examples of clustering algorithms that I used in my work.

3.3.1 k-means

Kmeans [18] is considered as the most used clustering algorithms because of its simplicity.

This is an iterative method that partitions the data set into K distinct non overlapping groups (clusters) where every data point is a member of only one group. It makes the inter-cluster points as identical as possible and tries to keep the clusters as far (different) as possible. It allocates the data points to a cluster in such a way that the sum of the squared distance between the points and the centroid, arithmetic mean of all the points belonging to the same cluster, is minimum.

The data points within the same cluster are more homogeneous i.e. similar if the variation within clusters is less.

Kmeans algorithm steps:

1. Choose number of clusters K.

2. First, shuffle the data set to initialize centroids and then select K data points randomly for the centroids with no replacement.

3. Iterate until centroids are fixed and there is no need to change (allocation of data points to clusters is not changing anymore)

Next we can compute the sum of the squared distance (SSE) between data points and all the centroids.

Kmeans follows Expectation-Maximization approach to solve the problem. The E-step is allocating the data points to the nearest cluster. The

M-step is calculating the centroid of every cluster. Let's solve it mathematically: The objective function is:

$$J = \sum_{i=1}^{m} \sum_{k=1}^{K} w_{ik} \|x^i - \mu_k\|^2$$

where "wik=1" for point "xi" if it belongs to cluster k, else "wik=0". Also, the centroid of "xi" cluster is "uk".

It's a two part minimization problem. First, we minimize J w.r.t. wik and consider "uk" as fixed. Next, we minimize J w.r.t. "uk" and consider wik as fixed. Technically, we are differentiating J w.r.t. wik first and then updating cluster assignments which is E-step. Then we differentiate J with respect to "uk" and recalculate the cluster centroids after the cluster formations from previous step, this is M-step.

Hence, E-step is:

$$\frac{\partial J}{\partial w_{ik}} = \sum_{i=1}^{m} \sum_{k=1}^{K} \|x^i - \mu_k\|^2$$

$$\Rightarrow w_{ik} = \begin{cases} 1 & \text{if } k = argmin_j \|x^i - \mu_j\|^2 \\ 0 & \text{otherwise.} \end{cases}$$

In short, assign the point xi to the nearest cluster based on its sum of squared distance from centroid.

And M-step is:

$$\frac{\partial J}{\partial \mu_k} = 2 \sum_{i=1}^{m} w_{ik}(x^i - \mu_k) = 0$$

$$\Rightarrow \mu_k = \frac{\sum_{i=1}^{m} w_{ik} x^i}{\sum_{i=1}^{m} w_{ik}}$$

This is recalculating the centroid of each cluster to see the new cluster assignments.

Elbow method:

Elbow method is a way to find out good number of clusters K. This calculation is based on sum of squared distance (SSE) between points and their cluster centroids. We choose k at the point where SSE begins to flatten out and making an elbow.

Drawbacks:

If clusters have a spherical kind of shape, it's good to use Kmeans algorithm. This algorithm is good in capturing this kind of data structure because it always tries to form a nice spherical shape around the cluster centroid. This also means, if the clusters have a complex geometric shapes, this algorithm won't work as expected and may result in a poor clustering of data points.

Key points to remember:

- Bigger clusters are given more weight.
- Always standardize the data when using kmeans algorithm.
- Elbow method in choosing number of clusters does not usually work as the error function monotonically decreases for all values of k.
- Kmeans does not work well if clusters are in complex shapes for example elliptical clusters because it's based on centroid concept and assumes spherical-like shapes of clusters.
- Kmeans still cluster the data which can't be clustered for instance data coming from uniform distributions.

3.3.2 DBSCAN

DBSCAN stands for Density-Based Spatial Clustering [19] with Noise. It identifies core samples of high density and forms clusters from them. It is good for data that contains clusters of same density.

DBSCAN groups together data points that are near to each other on the basis of two parameters: first is distance measurement, which is usually Euclidean distance, and second is a minimum number of points. Also, it marks as outliers the data points which are in low density regions.

Two main parameters:

- Epsilon (eps): it tells how close data points should be to be considered a member of a cluster. It means that these points are considered as neighbors if the distance between two points is lesser or equal to this value (eps).
- Minimum Points (minPts): it is the minimum number of points to build a dense region. For instance, if we choose the minPoints as 5, then we want at least 5 points to make a dense region.

Data points can be categorized into 3 types based on above parameters:

1. Core point: It is a core point if it has, within its eps neighborhood, at least minimum MinPts including itself.
2. Border point: A point that is not a core point but is in the neighborhood of a core point.
3. Noise point: A point that is neither a border point nor a core point. This means they are outliers and doesn't belong to any dense cluster.

Parameter estimation:

The parameter estimation is an issue for almost every data mining task. To select good parameters we must understand how they work and should have at least a basic knowledge about the data that will be used.

- eps: if the chosen eps value is too small, it won't be able to cluster a large part of the data and It will be considered as outliers because it doesn't satisfy the number of data points to make a dense region. Also, if the value is too high, it will merge the clusters and majority of points will be in the same cluster. Therefore eps should be chosen keeping in mind the distance of the dataset. In general, small eps values are preferable. However, a k-distance graph can be used to find eps value.
- minPoints: In general, we can derive minPoints from a number of dimensions (D) in the data-set, as minPoints $\geqslant$ D + 1. Larger values will make more significant clusters and are mostly better for data-sets

with noise. The minimum value for minPts must be 3, but it depends on how large is the data-set. The larger the data-set, the larger the minPts value should be chosen.

DBSCAN algorithm steps:

- First it divides the data-set into n dimensions
- Dbscan makes an n dimensional shape, for each point in the dataset, around that point. Then it counts the number of data points that falls within that shape.
- It counts this shape and assumes it as a cluster.
- It goes through each data point within the cluster and counts the number of other points nearby. This is how it iteratively enlarges the cluster.

Fig 10 shows an example of Dbscan with min-points as 4.

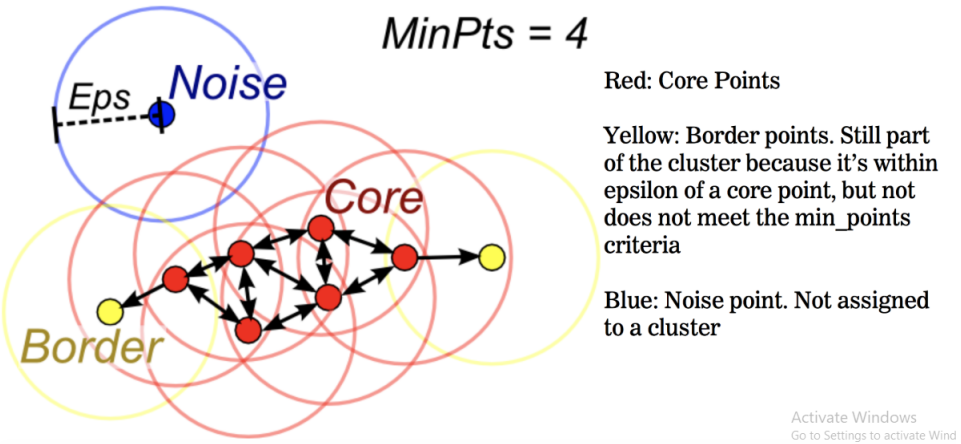


Figure 10: DBSCAN clustering with min-points 4

Why should we use DBSCAN??

The DBSCAN algorithm is used to identify association and structure in datasets that are difficult to find manually but can be appropriate and useful to find patterns and forecast trends. For example, recommendations in websites.

Advantages:

- Good at separating high density clusters from low density clusters within a given dataset.
- handles outliers within the data-set.

Disadvantages:

- Doesn't perform well when dealing with different density clusters. It struggles with similar density clusters.
- If data has many dimensions (high dimensionality data), it doesn't work well.

3.3.3 Agglomerative clustering

Hierarchical clustering algorithms [20] group identical objects into groups called clusters. There are 2 types of hierarchical clustering algorithms:

Agglomerative — Bottom up approach. It starts with small clusters and then merges them together to form bigger clusters.

Divisive — Top down approach. It starts with one single cluster and then breaks it into smaller clusters.

An example of hierarchical clustering is shown in Fig 11

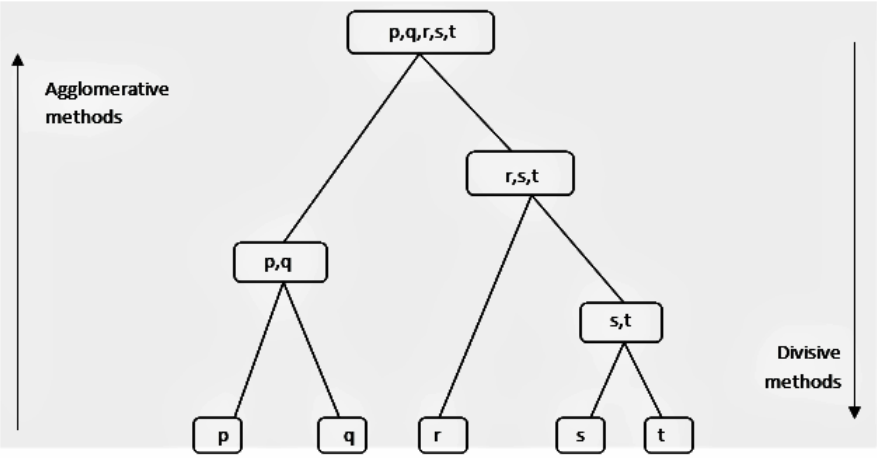
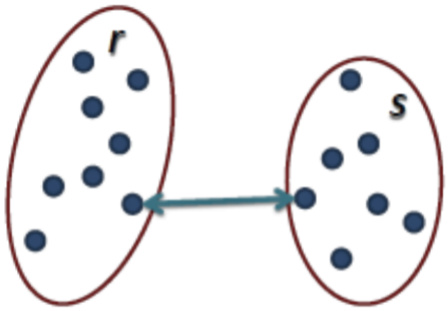


Figure 11: An example of Hierarchical clustering

The linkage criteria specifies how the distance between clusters is computed.There are basically four types of linkage criteria:

Single Linkage

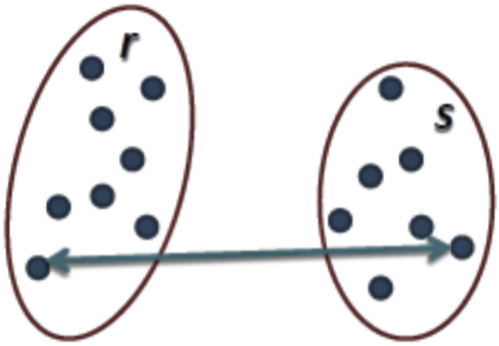
The distance between two clusters is computed as the shortest distance between two data points in each cluster.



$$L(r,s) = \min(D(x_{ri}, x_{sj}))$$

Complete Linkage

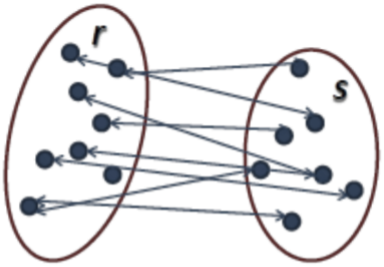
The distance between two clusters is computed as the longest distance between two data points in each cluster.



$$L(r,s) = \max(D(x_{ri}, x_{sj}))$$

Average Linkage

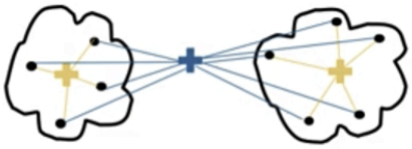
The distance between clusters is computed as the average distance between each data point in one cluster to every data point in other cluster.



$$L(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} D(x_{ri}, x_{sj})$$

Ward Linkage

The distance between clusters is computed as the sum of squared differences with in all clusters.



Distance Metric: It is a method used to measure the distance between data points. This parameter can affect or eventually change the end result.

Euclidean Distance: The shortest distance between two data points. For example: if x=(a, b) and y=(c, d) then the Euclidean distance between x and y is $sqrt((a-c)^2 + (b-d)^2)$

Advantages:

- Easy to implement
- Hierarchical clustering results in a hierarchy, i.e. a structure which is more informative while k-means returns the unstructured set of flat clusters. Hence, it is easy to decide the number of clusters by seeing at the dendrogram.

Disadvantages:

- It's not possible to undo the last step i.e. once the data points have been assigned to a cluster, they can't move around any longer.
- It is very conscious of outliers
- Time complexity. It is not good for big datasets.
- Initial parameters and order of data have a strong effect on the final results.

3.4 Clustering Validation

The term clustering validation is used to create a process of evaluating the outputs of a clustering algorithm. The goal of this section is to:

- Specify the various methods for clustering validation
- Compare the quality of results obtained from different clustering algorithms

There are two types of evaluation:

- Supervised- it uses a ground truth class values for samples.
- Unsupervised- it doesn't have any previous knowledge and measures the quality of the model by itself.

We will be using following clustering metrics to evaluate our clustering results (see Table 2).

Unsupervised	Supervised
1. Silhouette score	1. Adjusted rand score
2. Silhouette samples	2. Homogeneity score
3. Sum of squared errors	3. Completeness score
4. Calinski and Harabasz score	4. V-Measure score

Table 2: Supervised vs Unsupervised indices

3.4.1 Supervised indices

As discussed before, in supervised learning, the class labels in the dataset are known, which is used to build the classification model.

It's a machine learning approach which involves assigning labeled data such that a certain sample or pattern can be taken out from that data. The good thing about supervised learning is that the input data point is known and thus labeled appropriately.

Let's talk about few supervised metrics that I used to evaluate the outputs of my clustering algorithms.

Adjusted rand score

The Rand Index does a similarity check between tw0 clusterings by considering all pairs of groups and counting pairs that are allocated to the same or different clusters in the forecasted and true clusterings.

The adjusted Rand index is thus guarantees to have a value near to 0.0 for random labeling independently of the no. of clusters and groups and exactly 1.0 when the clusterings are similar.

Similarity score is between -1.0 and 1.0. Random labelings (e.g. Bad or independent labelings) have an ARI close to 0.0 or negative. Perfect labeling is exactly 1.0.

$$AdjustedIndex = \frac{Index - ExpectedIndex}{MaxIndex - ExpectedIndex}$$

ARI is symmetric which means swapping the records does not change the score.

ARI needs information of the ground truth classes, contrary to inertia, which is mostly never available or needs manual allocation (like in the supervised learning setting).

However, ARI can also be helpful in a completely unsupervised setting like a building block for a Consensus Index which can be used in clustering model selection

Homogeneity Completeness and V-measure score

Given the prior information of the ground truth class labels of the samples, it is quite possible to define some metric using conditional entropy analysis.

A clustering output satisfies homogeneity if all its clusters hold only points that are members of a single class.

A clustering output satisfies completeness if all the points that are members of a given class are associated to the same cluster.

Both scores have values between 0.0 and 1.0 and are positive. However, larger value is more desirable. Score is between 0.0 and 1.0 where 1.0 stands for best homogeneous and complete labeling.

V-measure is the harmonic mean of homogeneous and completeness. Score is between 0.0 and 1.0 where 1.0 stands for best complete labeling.

MATHEMATICAL FORMULATION

Homogeneity: each cluster holds only members of a single class.

$$h = 1 - \frac{H(C|K)}{H(C)}$$

Completeness: all class members are allocated to the same cluster.

$$c = 1 - \frac{H(K|C)}{H(K)}$$

where H(C—K) is the conditional class entropy where the cluster assignments, defined as:

$$H(C|K) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|}\frac{n_{c,k}}{n}\cdot log\left(\frac{n_{c,k}}{n_k}\right)$$

and H(C) is the class entropy:

$$H(C) = -\sum_{c=1}^{|C|}\frac{n_c}{n}\cdot log\left(\frac{n_c}{n}\right)$$

V-measure is defined as the harmonic mean of above two metrics:

$$v = 2 \cdot \frac{h \cdot c}{h + c}$$

Advantages:

- Bounded scores: 0.0 is a very bad score and 1.0 is a perfect score.
- Intuitive interpretation: Bad V-measure score can be statistically examined in terms of homogeneity and completeness to better understand what type of mistakes was done by the project.
- No speculations about cluster structure

Disadvantages:

- The metrics are not normalized with respect to random labeling. This means that depending on the number of clusters, samples and ground truth classes, a whole random labeling will not always give the same result for homogeneity, completeness and thus v-measure. In general, random labeling will not give zero score especially in case of large the number of clusters.

This issue can be ignored safely when the number of clusters is less than 10 and number of samples is more than thousand. For larger number of clusters or smaller sample sizes, it is good to use any adjusted index, for example the Adjusted Rand Index (ARI).

3.4.2 Unsupervised indices

We already discussed about unsupervised learning in previous section. Just to give a brief, unsupervised learning is a machine learning approach in which conclusions are made from unlabeled input data. The aim of unsupervised learning is to grouping in data from unlabeled data or find out the hidden patterns. The important thing to highlight is that in unsupervised learning both input and output are not known.

Let's talk about few unsupervised metrics that I used to evaluate the outputs of my clustering algorithms.

Silhouette score

Computes mean Silhouette Coefficient of all the samples. The Silhouette value of a single sample is defined as follows:

$$s_i = \frac{b_i - a_i}{max(a_i, b_i)}$$

where:

a is the mean distance between a point and all other data points within the same cluster

b is the mean distance between a point and all other data points in the closest cluster

The range of Silhouette score is in [-1, 1].

- +1 Score – it indicates that the sample is very far away from the other neighboring cluster. The best value is 1
- 0 Score – it indicates that the sample is either overlapping or very near to the decision boundary which is separating two neighboring clusters.
- -1 Score – it generally indicates that the samples have been allocated to the wrong clusters because different cluster was more similar. The worst value is -1.

Silhouette samples

The Silhouette Coefficient is a metric of how well data is clustered with samples that are identical to themselves. Clustering models are dense if they have a high Silhouette Coefficient which means samples in the same group are identical to each other and well-separated, whereas samples in other clusters are not very identical to each other.

Silhouette analysis is used to learn the separation distance between the predicted clusters. The silhouette plot shows a measure of how close each

data point in one cluster is to data points in the neighboring clusters and hence it provides a way to evaluate parameters such as number of clusters visually. This measure has a range [-1, 1].

For this to be a good value for a particular number of cluster, we must consider the following points: First, the mean value should be as near to 1 as possible. Second, the plot of every cluster should be above, as much as possible, the mean value. Plot region below the mean value is not preferable. Last, the width of plot should be uniform.

Sum of squared errors

Sum of squared error (SSE) is a scale to measure dissimilarity within a cluster. If all the cases with in a cluster are similar, the SSE would be equal to zero.

The mathematical formula for SSE is:

$$SSE = \sum_{i=1}^{n} (x_i - \bar{x})^2$$

Where n is the no. of observations, "xi" is value of the "ith" observation and zero is the mean of all observations.

If sum of squared errors (SSE) is zero, it means a perfect match and there isn't any error. This is performed by repeated calculations designed to bring the samples tighter or closer.

This is very unlikely to happen with the real world data. Therefore, we should look for an approach that has lower SSE. The lower the SSE, more similar are the samples in that dataset. A high SSE value suggests that the samples have a reasonable amount of differences between them and might not be a usable cluster.

Calinski-Harabasz Index

The Calinski-Harabasz index can be defined as ratio between- the within-cluster dispersion / the between-cluster dispersion.

The score is higher for dense and well-separated clusters, which is the standard concept of a cluster.

If ground truth class labels are not known, Calinski-Harabasz index can also called as the Variance Ratio Criterion which means it can be used to evaluate the model. A higher Variance Ratio score means model has better defined clusters.

For dataset E of size nE and k clusters, the Calinski-Harabasz score is defined as:

$$s = \frac{\mathrm{tr}(B_k)}{\mathrm{tr}(W_k)} \times \frac{n_E - k}{k - 1}$$

where "tr(Bk)" is the trace of between group dispersion matrix and "tr(Wk)" is the trace of within-cluster dispersion matrix.

Advantages

The score is higher for dense and well-separated clusters. Fast to compute.

Disadvantages

The Calinski-Harabasz index is usually higher for convex clusters than other types of clusters for example density based clusters (DBSCAN).

3.5 Cluster Characterization

In this section, we will study about Cluster characterization. Basically, characterization helps in identifying the type of cluster, its features and data distribution of patterns. It also identifies outlier clusters. Cluster characterization helps domain expert in labelling of the data, in our case to robot cycles.

First, we extract top 10 most relevant features of our clusters, using Decision Tree Classifier, and then individually characterize each group. This

helps domain experts to take better decisions about the data by going through the most relevant features that better describes each group.

After choosing these most relevant features, their data distribution has been shown using boxplots, which will help to identify the most characterizing properties of a cluster in terms of relevant features and content. This information may help experts to understand every group thoroughly because it is impossible to inspect all the samples manually. Therefore, cluster characterization is an important support to take better decisions.

As mentioned above the steps of cluster characterization, Fig 12 shows a short architecture about the same.

Figure 12: Cluster charcterization architecture

Why Boxplot?

A boxplot is a graph that tells us about how the values in the dataset are spread out. It is a helpful way to visualize dissimilarities among various groups or samples. Also, they provide statistical information such as ranges, medians and outliers.

3.5.1 Decision Tree classifier

Decision tree classifiers is a well-known classification approach in various pattern recognition issues, for instance, character recognition and image classification. Decision tree classifiers [21] work more amazingly, specifically for complex and complicated classification issues, because of their computationally effective and high adaptability features. Apart from this, decision tree classifiers are way better than numerous old supervised classification methods.

In particular, decision tree classifiers doesn't need any distribution assumption regarding the input data which gives it a higher adaptability to manage with various kinds of datasets (numeric, categorical, even data with missing values). Decision trees are also good for dealing with non-linear relations among classes and features. Besides, the classification procedure given by a tree-like structure is always interpretable and natural.

The goal of a classification tree is to classify the input data. Given an input data point, it should assign it to a specific class/label. A decision tree is designed through a procedure known as binary recursive partitioning. It is an iterative process of breaking the data into partitions, and then again breaking it up on each branch.

A Decision tree has 3 main segments:

1. Nodes – test for the value of a certain feature.
2. Edges/ Branch - compare the outcome of a test and attach to the next leaf or node.
3. Leaf nodes - Terminal nodes that forecast the outcome (mostly class distribution or class labels).

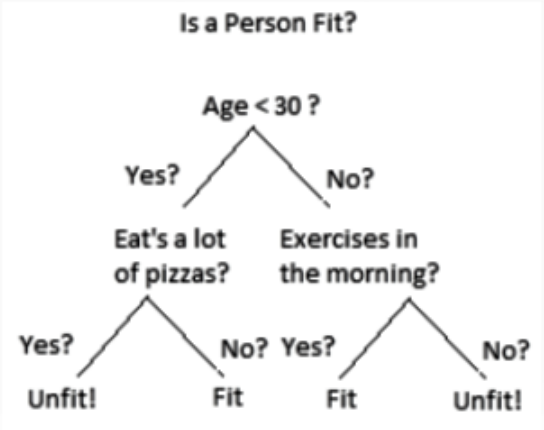


Figure 13: An example of Decision Tree Classifier

To better understand the idea of Decision Tree consider the example shown in Fig 13. Let's say we want to know, given prior information such as eating habits, age, physical activity etc., whether a person is fit or unfit.

The decision nodes can be the questions like 'Does he exercise?', 'What's his age?", 'Does he eat a lot of burgers? And the leaf nodes represent outputs like 'fit' or 'unfit'.

4 Implementation Tools

This chapter provides information on the implementation tools used to develop this work.

4.1 Python

To overcome the difference between C and shell programming, Python [11], an uncomplicated but still robust tool is available in the plethora of programming languages. It is a perfect fit to "Throw-away programming" and rapid prototyping. Python's syntax is made from constructs derived from different languages like Icon, C, Modula-3, ABC etc. The Python interpreter is extended with new function and data type implemented in C.

For highly customizable C applications such as editors or window managers, python can serve as an extension language. It is available for many operating systems like UNIX (including Linux), MS-DOS, Windows NT, MS-Windows 3.1, and OS/2 the Apple Macintosh operating system. In the Python Library Reference there is a description of the built in functions and modules and of the non-essential built in object types.

With python you can separate your program into modules that may be reused in other Python programs. It has a huge collection of standard modules that you can use as the basis of your programs — or as examples to start learning to program in Python. Things like file I/O, sockets, system calls, and even interfaces to graphical user interface toolkits like Tk are provided by some of these modules.

Since Python is an interpreted language, there is no compilation and linking is necessary and hence you can save considerable time during program development. The advantages of such an interactive interpreter are that we can easily experiment with features of the language, write throw-away programs and even test functions during bottom-up program development. Python allows programs to be written compactly and readably.

Python programs are usually much shorter than equivalent C, C++, or Java programs because of several reasons:

- complex operations in a single statement are expressed by the high level data types
- declarations of variables or arguments are unnecessary

- beginning and ending brackets are omitted and indentation is used for statement grouping

4.2 Pycharm

The Czech company JetBrains created an integrated development environment (IDE) named PyCharm specifically for the Python language. It provides a graphical debugger, an integrated unit tester, code analysis, integration with version control systems (VCSes), and supports web development with Django as well as Data Science with Anaconda.

PyCharm is cross-platform, with macOS, Windows, and Linux versions [14]. The Community Edition is released under the Apache License, and there is also Professional Edition with extra features – released under a proprietary license.

Features:

- Code and project navigation: file structure views ,specialized project views, and speedy jumping between files, methods ,classes and usages
- Coding analysis and assistance, with code completion, error and syntax highlighting, linter integration, and quick fixes.
- Integrated Python debugger
- Google App Engine Python development [only in professional edition]
- Python refactoring: consists of rename, extract method, introduce constant, introduce variable, push down ,pull up etc.
- web frameworks like Django, web2py and Flask are supported [only in professional edition]
- Integrated unit testing, with line-by-line code coverage
- Version control integration i.e. unified user interface for Git, Mercurial and CVS with change lists
- scientific tools like matplotlib, numpy and scipy are also supported [only in professional edition]

Plugins

With the provision of API in PyCharm - developers can write their own plugins .This helps to extend PyCharm features. Several plugins from other JetBrains IDE can also work with PyCharm. PyCharm has compatibility with more than 1000 plugins.

4.3 Libraries

Here I would like to mention about all the libraries that my work uses with python files. There are several libraries that this project imports from.

4.3.1 Numpy

Numpy is a fundamental-package for array-computing with python. Numerical Python, in short, NumPy, is a library including multidimensional array objects and a collection of routines which help in processing those arrays. Mathematical and logical operations on arrays can be performed using NumPy [7]. It also discusses the several array functions, types of indexing etc. An introduction to Matplotlib is also provided.

It contributes:

- an N-dimensional array object
- broadcasting (sophisticated) functions
- tools for integrating Fortran and C/C++ code
- useful Fourier transform, linear algebra, and random number capabilities and a lot more

Other than its scientific uses, NumPy is also useful as an efficient multidimensional container of generic data. Arbitrary data-types may also be defined. This grants NumPy the ability to quickly and seamlessly integrate with a wide variety of different kinds of databases.

4.3.2 Pandas

To make working with structured (multidimensional, tabular, potentially heterogeneous) and time series data both easy and intuitive we have a python package named – pandas [9], which provides fast , flexible, and expressive data structures. It aspires to be the fundamental high-level building block for doing practical, real world data analysis in Python. Pandas can become the most flexible and powerful open source data analysis or manipulation tool available in any language at present.

Pandas is appropriate for a variety of data:

- Tabular data with mixed typed columns, as in an Excel spreadsheet or in an SQL table
- Ordered or unordered time series data, not necessarily fixed- frequency
- Arbitrary matrix data (homogeneously typed or heterogeneous) with column and row labels
- Any form of statistical or observational data sets. There is no need for data to be labeled at all for the data to be placed into a pandas data structure

Few main features of Pandas:

- missing data (represented as NaN) in floating point and non-floating point data is easily handled
- Automatic and explicit data alignment in which objects can be explicitly aligned to a set of labels, or the user can just ignore the labels and let DataFrame ,Series , etc. automatically align the data for you in computations
- Size mutability - columns can be deleted from and inserted in DataFrame and higher dimensional objects
- Great fancy indexing, label-based slicing and subsetting of huge data sets
- Flexible group-by functionality to nicely perform split-apply-combine operations on datasets, for both transforming and aggregating data.

- ragged, differently-indexed data in other Python and NumPy data structures can be easily converted into DataFrame objects

4.3.3 JSON

JSON abbreviation for JavaScript Object Notation, [10] is a lightweight format for transporting as well as storing data. When we have to send data from a server to a web page, JSON come in handy. Json is very easy to understand and self-describing.

If we have data stored in a JavaScript object, first, we can convert the object into JSON, and then we can send it to a server. If we have received data in JSON format, we can simply convert it to a JavaScript object.

The advantage of JSON format being text only is that it can simply be sent to and from a server and can be easily used as a data format by any programming language.

If we have to convert a string, written in JSON format into native JavaScript objects we can simply use the built in function- JSON.parse() provided by JavaScript. Hence, data received from a server in JSON format can be used like any other JavaScript object.

JSON Schema highlights a JSON-based format which defines the structure of JSON data for documentation, validation, and interaction control. A contract is provided by it for the JSON data, required by a given application, and the ways that data can be modified.

JSON Schema is derived from the concepts from XML Schema (XSD), but it is JSON-based. Similar to XSD, the same serialization or deserialization tools may be useful both for the data and schema; and is self-describing.

There have been suggestions of .schema.json being a standard filename extension but as such, there are none.

4.3.4 MatplotLib

For 2D plots of arrays , we have Matplotlib, [12] which is a splendid visualization library in Python. It is a multi-platform data visualization library that has been built on NumPy arrays and is usually designed to work with the broader SciPy stack. John Hunter introduced it in the year 2002.

Visualization is extremely useful as it allows us visual access to enormous amounts of data in straightforward visuals. Line, bar, scatter, histogram etc. are all different types of plots that Matplotlib consists of.

Windows, macOS and Linux distributions have matplotlib and the majority of its dependencies are as wheel packages. Run the command matplotlib to install the corresponding package.

Matplotlib has a comprehensive selection of plots. Plots helps to understand patterns, trends and also to make correlations. They're usually used for reasoning about quantitative information.

4.3.5 Scikit-Learn

Python's premier general-purpose machine learning library - Scikit-Learn (abbreviation- sklearn) [16] is a Python module that integrates classical machine learning algorithms in the clannish world of scientific Python packages (numpy, matplotlib, scipy).

Scikit-Learn is very versatile and hence bags the title as the best starting place for most ML problems.

The aim is to provide simple and efficient solutions to learning problems that are accessible to everybody and reusable in various contexts: machine-learning as a versatile tool for science and engineering.

For beginners, It's a great library because it offers a high-level interface for many tasks (e.g. cross-validation, preprocessing data, etc.) allowing us to better use the whole machine learning workflow and also understand the big picture.

4.3.6 Joblib

Joblib is a tool which provides lightweight pipelining in Python.

In general:

- transparent disk caching of functions and lazy reevaluation
- very simple and easy parallel computing

Joblib [8] is optimized to be quick and robust on huge data usually. It has certain optimizations for numpy arrays.

The goal of this library is to provide tools to quickly achieve better reproducibility and performance when working with long running jobs.

- Avoid calculating the same thing twice: usually we rerun the code again and again, for example when prototyping complex and computational heavy jobs (such as in scientific development), but manual solutions to remove these problems are error-prone and can lead to unreproducible results.
- Persist to disk transparently: accurately persisting random objects containing huge data is tough. Joblib's caching mechanism removes manually written persistence and automatically links the file on disk to the runnable context of the original Python object. Hence, joblib's persistence is best for computational job or resuming an application status, for example after a crash.

Joblib addresses these issues while leaving our code and flow control unmodified (no new paradigms, no framework).

Features:

- Transparent and fast disk caching of resulting value: similar functionality for Python functions that performs well for random Python objects, including very huge numpy arrays.
- Embarrassingly parallel helper: Easy to write readable code in parallel and debug it easily.
- Fast compressed Persistence: helps us to work efficiently on Python objects, especially objects containing huge data (joblib.load and joblib.dump).

4.4 Google Colaboratory

Google Colaboratory, often referred to as "Google Colab" or simply "Colab" [13] is a research project with which we can prototype machine learning models on powerful hardware options such as TPUs and GPUs. A server less Jupyter notebook environment is provided by Colab for interactive development. Similar to other G Suite products, Google Colab is free to use [15].

It can be very useful as a tool for improving your coding skills. It can also allow anyone to develop deep learning applications with the help of popular libraries such as PyTorch, Keras, TensorFlow, and OpenCV.

Although it supports Python 2.7 and 3.6, but it loses the race as it doesn't support R or Scala yet. Sessions and sizes are limited, but we can definitely get around that if we simply re-upload our files.

In Colab we can create, upload, store, and share notebooks, we can mount our Google Drive and use whatever we've got stored in there, upload our personal Jupyter Notebooks, import majority of our favorite directories, upload Kaggle files, upload notebooks directly from GitHub, download our notebooks etc.

It provides high computational resources like:

- 13GB RAM
- 33GB Free Space
- 2vCPU @ 2.2GHz
- n1-highmem-2 instance
- 90 minutes of idle cut-off
- 12 hours maximum
- GPU instance upgraded to 350 GB

5 Experimental results

In this section we will talk about the model creation, analysis of the clusters (by comparing supervised and unsupervised metrics), best configuration selection on the basis of results obtained and lastly, data distribution of patterns i.e. cluster characterization.

5.1 Model creation

Let's first talk about the parameters used in the clustering algorithms, such as input parameters, expected output parameter, number of models etc.

Algorithm used	Input	Desired output	Number of models
K-means	Cluster range: (2,11)	Labels	9
Agglomerative	Cluster range: (2,11) 4 different Linkage Types: (Single, Complete, Average, Ward)	Labels	9*4 = 36 (9 for each linkage type)
DBSCAN	Epsilon range: (5,25) with Min samples: (10,30) Epsilon range: (8,15) with Min samples: (5,10)	Labels Number of clusters Noise	20*20 = 400 7*5 = 35 Total= 435

Table 3: Model creation

As shown in Table 3, we have used 3 different clustering algorithms in this study.

1. K-means: n_cluster (k) ranges from [2, 11] which is an input parameter for this algorithm and we get a model corresponding to every value of k.

2. Agglomerative: linkage criteria is an important input parameter for this algorithm along with n_cluster (k). We get models corresponding to each linkage criteria against every value of k.

3. DBSCAN: Firstly, I tried with eps range [5, 25] and min_samples [10, 30] to try with all possible permutations and combinations to get good models to further analysis the data. Later, I tried with another combination of eps and min_samples to check if I could get some different results with few more models.

Few facts about Dbscan:

1. Though we generated 435 models to observe all kinds of permutations and combinations to get the best outcome but used only models falling under Epsilon range(11, 14) because others were with silhouette score either negative or NaN, or were producing very low/high number of clusters(Ex 0,1, 70,80,100+)

2. We also observed that if Epsilon and number of clusters are constant, then Noise (amount of outliers) increases when min_samples is increased. Outliers are the samples that do not satisfy the density factor of the Dbscan algorithm. (For example, see records in Table 4)

eps(radius)	min_samples(points)	n_clusters_	n_noise_
13	11	2	898
13	12	2	970
13	13	2	1038
13	17	2	1364
13	19	2	1537
13	20	2	1620
14	13	3	649
14	17	3	822
14	18	3	852
14	28	3	1332
14	29	3	1373

Table 4: Dbscan for eps 13 and 14

Clustering data points are spread across some geography (for example GPS coordinate points). The eps parameter is linked with the geographic scale of the study region. The min_samples parameter defines the minimum number of points needed to create a new cluster. A smaller value may extract many clusters but the final clusters includes

noise as well. A bigger value of min_samples produces a more robust cluster. However, it may remove some potentially smaller regions as it tries to merge them in a bigger one.

For example, let's consider 2 clusters produced with min_samples 10, and later increase min_samples to 12 keeping eps and k constant. (As shown in figures 14)

Blue-cluster 0
Orange-cluster 1
Yellow- noise

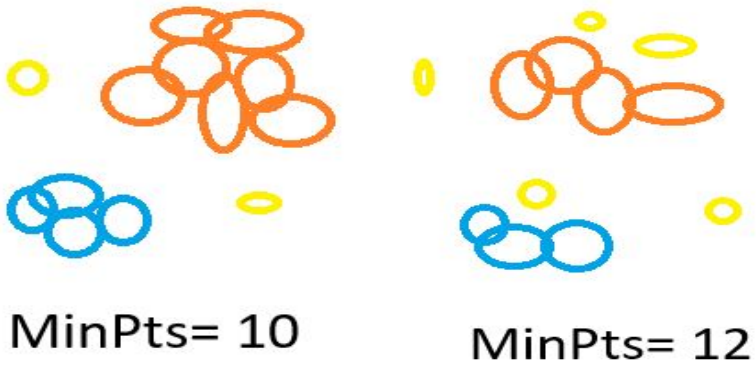


Figure 14: An example with Minimum points 10 and 12

If we increase minPts from 10 to 12, we observed that few circles do not satisfy the condition and can't be a cluster member anymore of that particular cluster and thus converts into noise.

3. Since we got many records against the same value of K (n_clusters), we selected the maximum score (highlighted in green) from the filtered dataset i.e. considering only relevant data (discussed before). (For example, see records in Table 5)

eps(radius)	min_samples (points)	n_clusters_	n_noise_	Silhouette Coeff	ARI	CH
13	19	2	1537	0.187795	0.430807	1815.431
13	20	2	1620	0.185509	0.427865	1806.753
14	14	2	694	0.207247	0.455285	1886.055
14	15	2	735	0.205274	0.454786	1886.93
14	16	2	773	0.203483	0.453249	1882.791
14	19	2	900	0.201238	0.449544	1860.339
13	28	3	2380	0.127189	0.499612	2431.642
13	29	3	2456	0.124792	0.495824	2416.875
14	13	3	649	0.157659	0.456553	1275.757
14	18	3	852	0.130302	0.450475	1253.32
14	28	3	1332	0.16092	0.554178	2642.573
14	29	3	1373	0.159338	0.55221	2631.14

Table 5: Dbscan for various metrics

5.2 Cluster analysis

Here, we will interpret clustering algorithm results by analyzing and comparing various supervised and unsupervised metrics.

5.2.1 Unsupervised metrics comparison

Silhouette score

As we already discussed in previous sections that this metric specifies how well samples are grouped with samples that are identical to themselves.

Table 6 shows Silhouette scores of all 3 clustering algorithms in numerical figures while Fig 15 shows the same results graphically to better visualize, understand and compare the scores.

K (n_clusters)	Kmeans	Agg_Average	Agg_Complete	Agg_Single	Agg_Ward	DBSCAN
2	0.449973	0.958987	0.958987	0.956988	0.386194	0.207247
3	0.449009	0.95717	0.95717	0.95717	0.387175	0.16092
4	0.302427	0.931095	0.931095	0.931095	0.310438	0.037363
5	0.297781	0.931068	0.931068	0.931068	0.219781	0.016589
6	0.297761	0.526724	0.353698	0.019198	0.233841	-0.06492
7	0.298824	0.367841	0.269941	-0.0466	0.247938	-0.11676
8	0.309273	0.350522	0.1967	-0.08538	0.270779	-0.19882
9	0.304312	0.334225	0.18439	-0.11692	0.265993	-0.29362
10	0.304264	0.276468	0.153424	-0.14043	0.276278	-0.23967

Table 6: Silhouette scores of all algorithms

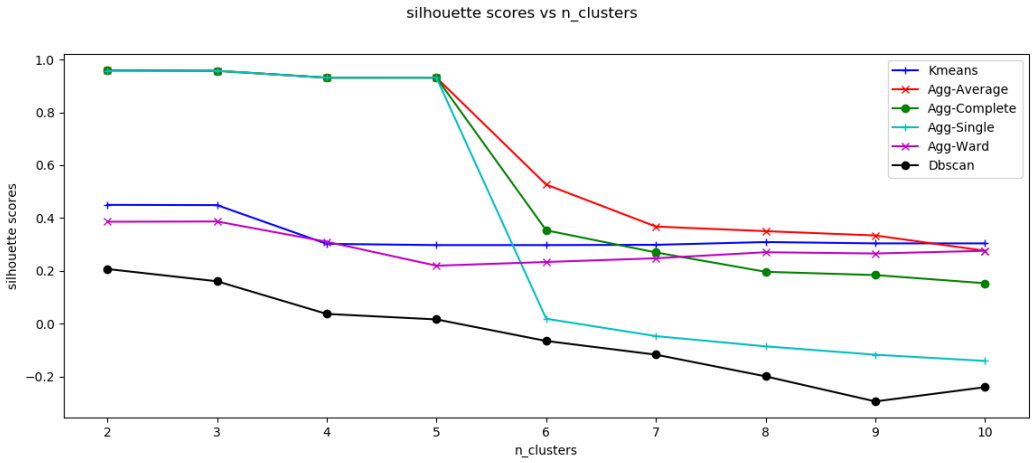


Figure 15: Silhouette scores vs n_clusters (k)

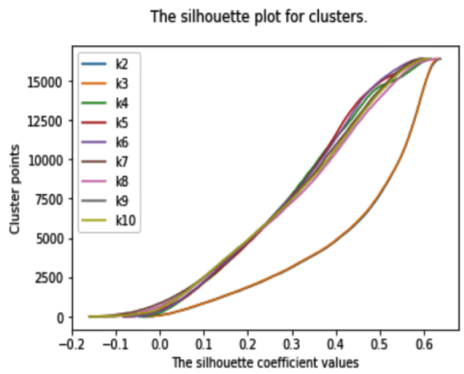
Observations:

- We observed that Agglomerative (except linkage ward) gives a very high Silhouette score, nearly equal to 1, for range (k=2 to 5) which also means that these clusters are well separated from each other. However, score gradually decreases from k=6 onward.

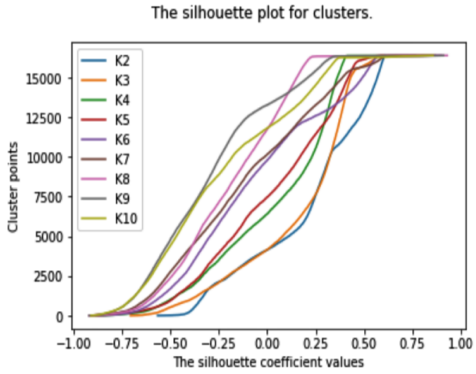
- Next best is K-means and Agglomerative Ward which is nearly constant throughout all the values of K. However, k-means performed slightly better than Agglomerative ward.
- Dbscan performed worst as compared to others. We can see negative or zero silhouette score also after k=6 which means clusters are overlapping or samples are assigned to a wrong cluster, as different cluster was more identical.

Silhouette samples

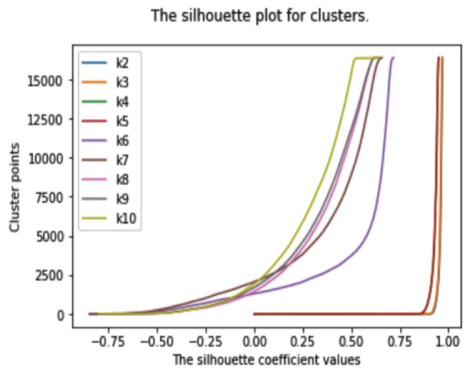
Below figures show Silhouette samples i.e. silhouette coefficient values for all the data points against the cluster points (for all clustering algorithms)



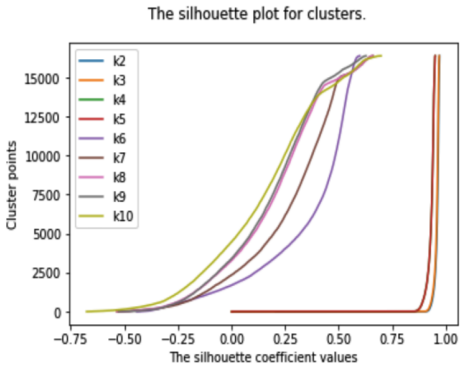
(a) k-means



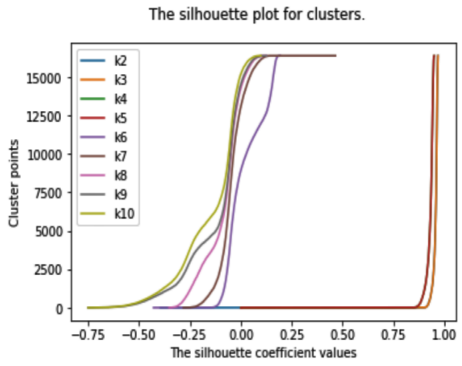
(b) DBSCAN



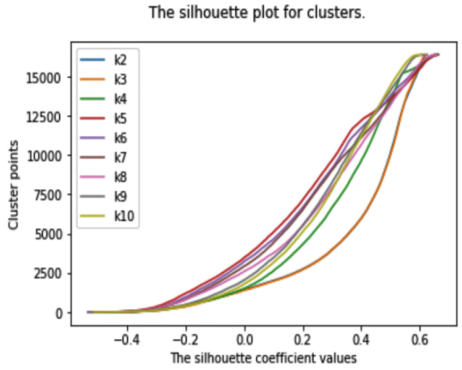
(a) Agglomerative- Average



(b) Agglomerative- Complete



(a) Agglomerative- Single



(b) Agglomerative- Ward

Observations:

- We observed that for Kmeans, for almost all the data points, silhouette coefficient is more than 0, and quite wide (a wide silhouette is better than a narrow one). This behavior is similar for all the values of k except for k=2 and 3 (even more better and wider).
- DBSCAN, for a lot of data points silhouette coefficient is less than 0, less wider.
- Agglomerative (except for linkage ward) performed best only for k=2 to 5 (all data points more than 0) but performance decreases as k is increased (after k=6).
- Agglomerative ward performed almost the same way as k-means

Calinski-Harabasz Index

A higher Calinski-Harabasz score shows that the clusters are dense and well separated. It is considered to be higher for convex type clusters than other concepts of clusters (For ex. DBSCAN)

Table 7 shows Calinski-Harabasz scores of all 3 clustering algorithms in numerical figures while Fig 19 shows the same results graphically to better visualize, understand and compare the scores.

K (n-clusters)	Kmeans	Agg-Average	Agg-Complete	Agg-Single	Agg-Ward	DBSCAN
2	13562.55	2359.135	2359.135	765.664	11411.47	1886.93
3	10874.03	1361.853	1361.853	1361.853	9233.707	2642.573
4	10845.02	1036.146	1036.146	1036.146	9321.743	1934.169
5	10491.87	812.0595	812.0595	812.0595	8913.215	1624.773
6	10060.51	651.6714	4011.05	649.8116	8346.051	632.5436
7	9865.422	544.3462	5129.036	541.9789	8176.573	1056.559
8	9452.842	3310.111	5107.774	464.5544	8113.66	471.7099
9	9237.341	2970.942	4678.405	406.924	7856.39	417.9173
10	9183.14	2642.901	4600.455	361.8543	7728.762	684.9945

Table 7: Calinski-Harabasz scores of all algorithms

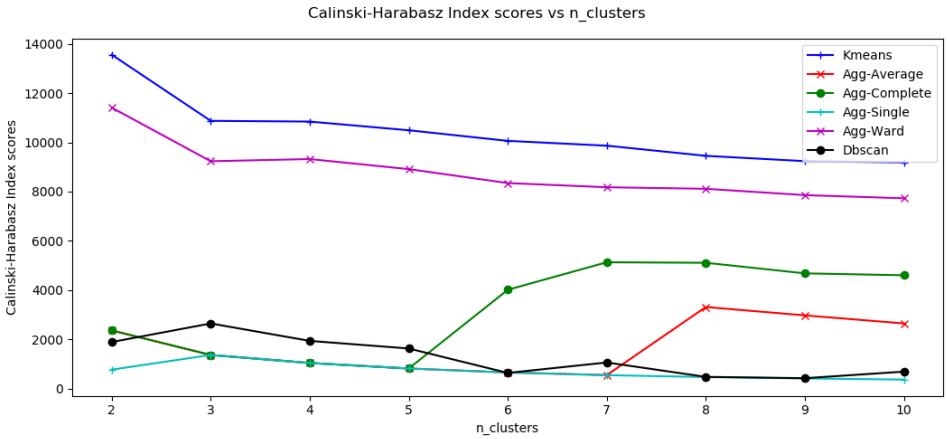


Figure 19: CalinskiHarabasz scores vs n_clusters (k)

Observations:

- K-means performs the best throughout all the values of K which means clusters are dense and well separated. Also, clusters are more or less spherical and compact in their middle (such as normally distributed).
- After k-means, next best is Agglomerative Ward behaving the same way as k-means.
- For all other algorithms, CH score is quite low i.e. clusters formed are not well compact.

Sum of Squared errors (SSE)

Just to give a brief, Sum of squared error (SSE) is a metric of variation with in a cluster and is used to inspect the number of clusters.

Table 8 shows Sum of Squared errors (SSE) scores of all 3 clustering algorithms in numerical figures while Fig 20 shows the same results graphically to better visualize, understand and compare the scores.

K (n_clusters)	Kmeans	Agg-Average	Agg-Complete	Agg-Single	Agg-Ward	DBSCAN
2	66987071.82	106035865.5	1.06E+08	1.1E+08	68672841	80935631
3	52783012.87	104061045.4	1.04E+08	1.04E+08	54371102	62953299
4	41254302.23	102055056	1.02E+08	1.02E+08	42672710	61591766
5	34637867.53	101344487.4	1.01E+08	1.01E+08	36261680	55388013
6	30337676.26	88290664.04	50257452	1.01E+08	32402681	63258855
7	26789998.7	84154441.86	38112422	99431228	28505333	49253573
8	24537425.55	46267127.97	33091226	91848083	25464770	76395994
9	22446641.39	44970431.4	31813360	79543639	23838208	50378509
10	20471813.33	44882863.17	28493061	78879689	21894021	49561191

Table 8: Sum of Squared errors (SSE) of all algorithms

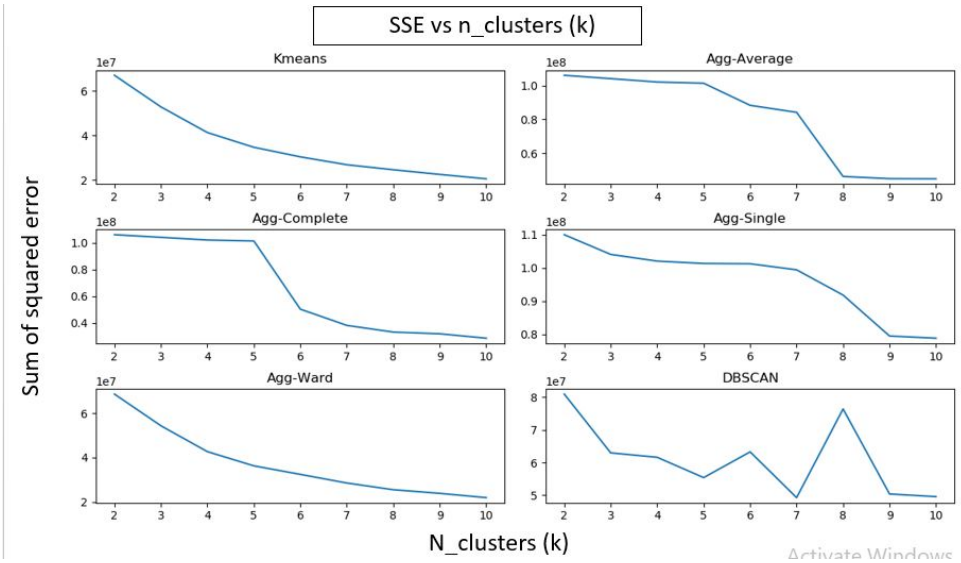


Figure 20: Sum of Squared errors (SSE) vs n_clusters (k)

Observations:

- K-means and Agglomerative ward, we can observe an exponentially

decreasing graph as the value of k decreased. And appropriate number at the knee can be at (k = 4) or (k = 5) for both the algorithms, which can be the appropriate number of cluster.

- About Dbscan, we observe a non-ideal behavior, SSE increases as k increases (at k=6 and k=8).
  Reason? Inertia calculates the internal cluster sum of squares (i.e. sum of squares is the sum of all the residuals). Inertia mostly depends on the assumption that the clusters will be convex i.e. of spherical shape. DBSCAN doesn't divide data into spherical clusters, hence inertia is not a good index to use for evaluating DBSCAN type of models. Inertia is mostly used in other type of clustering methods, for example K-means clustering.

K-means (The Elbow method):

Inertia is more often used in K-means clustering because it always tries to form a nice spherical shape around the cluster centroid, and also known as The Elbow method. Considering the fact, if we look at Fig 21, it is observed that appropriate number at the knee can be at (k = 5), which can be the appropriate number of cluster.

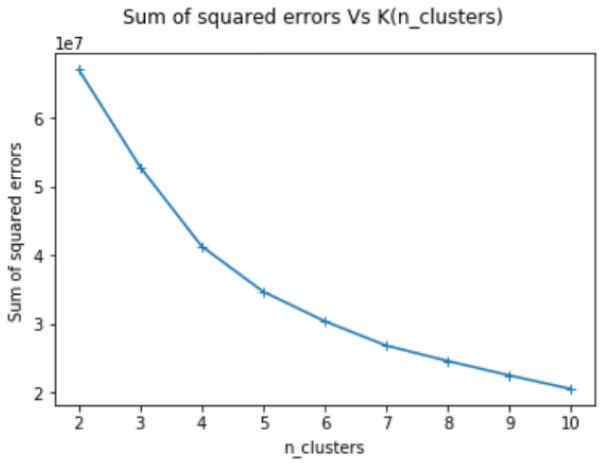


Figure 21: Sum of Squared errors (SSE) for k-means

5.2.2 Supervised metrics comparison

Adjusted rand index

ARI does a similarity check between samples present in the cluster. A higher ARI score means that the clusters are more similar.

Table 9 shows Adjusted rand scores of all 3 clustering algorithms in numerical figures while Fig 22 shows the same results graphically to better visualize, understand and compare the scores.

K (n_clusters)	Kmeans	Agg-Average	Agg-Complete	Agg-Single	Agg-Ward	Dbscan
2	0.119615	0.000141	0.000141	4.69E-05	0.236377	0.455285
3	0.121973	0.000141	0.000141	0.000141	0.236559	0.554178
4	0.124233	0.000188	0.000188	0.000188	0.169265	0.535742
5	0.258855	0.000188	0.000188	0.000188	0.213385	0.601825
6	0.26135	0.000228	0.096836	0.000205	0.20368	0.457574
7	0.252921	0.000275	0.072736	0.000137	0.275253	0.543317
8	0.261804	0.153736	0.080328	0.000137	0.263319	0.445833
9	0.279589	0.156431	0.083598	0.000176	0.267143	0.442241
10	0.279509	0.156477	0.070595	0.000216	0.315726	0.461786

Table 9: Adjusted rand scores of all algorithms

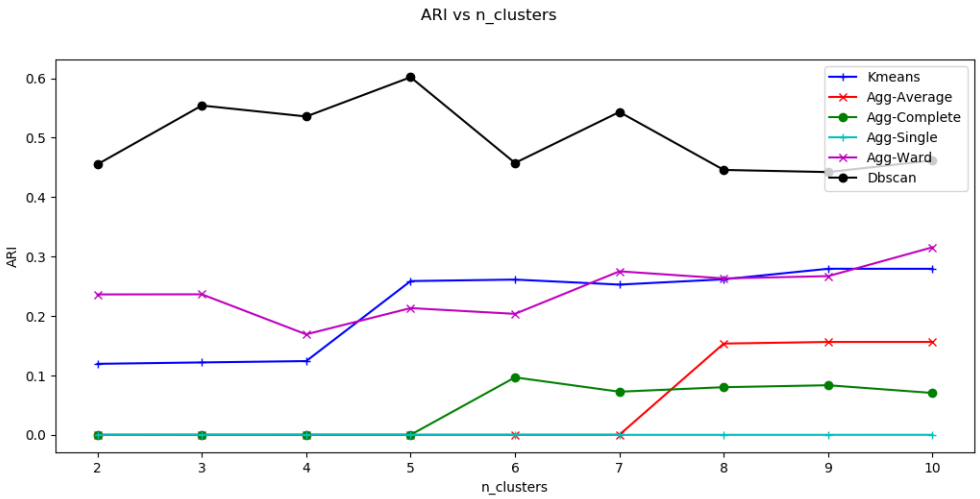


Figure 22: Adjusted rand scores vs n_clusters (k)

Observations:

- We observed that ARI score is best for Dbscan for all the values of k which means good similarity in the clusters.
- Next best is again k-means and Agglomerative ward where k-means performs better at k= 5 and 6 (just to highlight)
- All other algorithms gives bad results. (almost equal to 0 indicating a bad cluster similarity)

Homogeneity, Completeness & V-measure

For all 3 scores, a larger value is more desirable for best homogeneous and complete labeling.

Homogeneity

K (n_clusters)	Kmeans	Agg-Average	Agg-Complete	Agg-Single	Agg-Ward	Dbscan
2	0.150153	0.00028	0.00028	9.33E-05	0.211627	0.437817
3	0.151939	0.00028	0.00028	0.00028	0.21217	0.569952
4	0.177765	0.000373	0.000373	0.000373	0.223785	0.558792
5	0.34467	0.000373	0.000373	0.000373	0.31224	0.678575
6	0.39855	0.000458	0.106003	0.000439	0.372131	0.440803
7	0.416155	0.000551	0.139588	0.000521	0.493117	0.645042
8	0.466268	0.189677	0.152476	0.000521	0.522529	0.433892
9	0.517547	0.19863	0.158436	0.000606	0.529904	0.43094
10	0.517471	0.198729	0.183492	0.000691	0.630755	0.597367

Table 10: Homogeneity scores of all algorithms

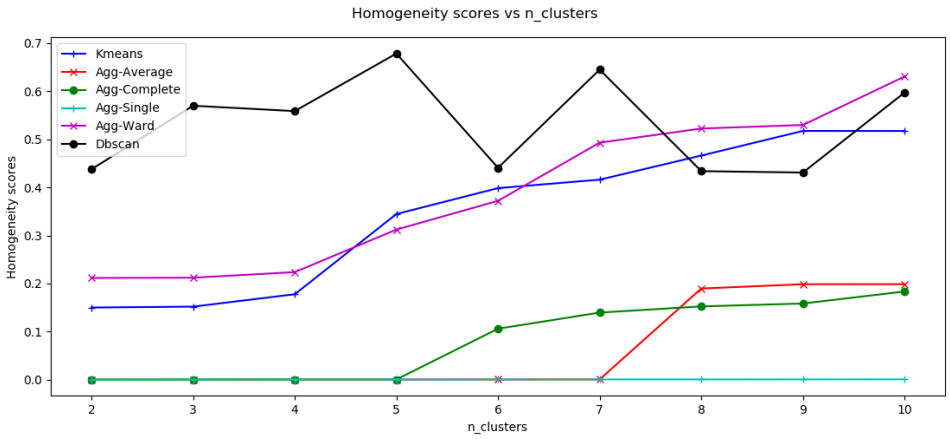


Figure 23: Homogeneity scores vs n_clusters (k)

Completeness

K (n_clusters)	Kmeans	Agg-Average	Agg-Complete	Agg-Single	Agg-Ward	Dbscan
2	0.356828	0.245124	0.245124	0.219979	0.463756	0.793149
3	0.359193	0.229933	0.229933	0.229933	0.463885	0.797855
4	0.250208	0.227403	0.227403	0.227403	0.354661	0.716235
5	0.380157	0.22006	0.22006	0.22006	0.355364	0.714927
6	0.386285	0.216029	0.23082	0.207168	0.365311	0.795583
7	0.353499	0.216723	0.219808	0.178997	0.432689	0.671808
8	0.365804	0.438659	0.200445	0.175694	0.422823	0.730239
9	0.37646	0.444677	0.198206	0.178719	0.400968	0.714336
10	0.376381	0.443813	0.205422	0.181076	0.443441	0.611867

Table 11: Completeness scores of all algorithms

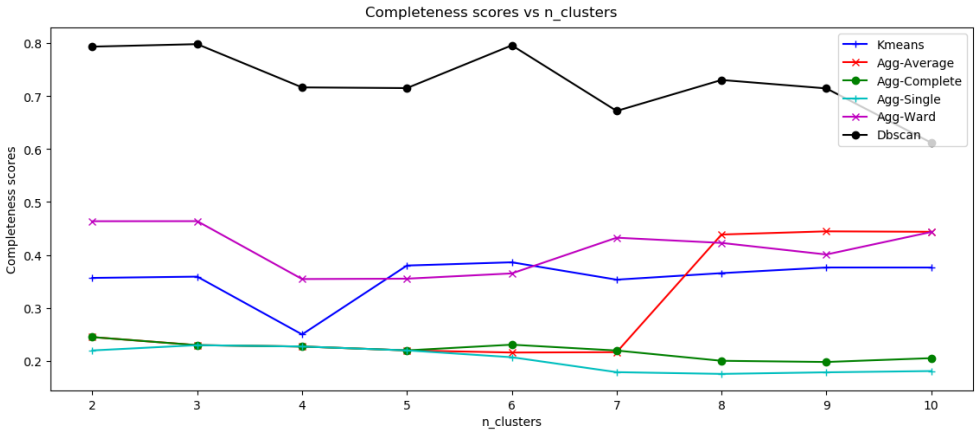


Figure 24: Completeness scores vs n_clusters (k)

V-measure

K (n_clusters)	Kmeans	Agg-Average	Agg-Complete	Agg-Single	Agg-Ward	Dbscan
2	0.211364	0.000559	0.000559	0.000186	0.29063	0.564198
3	0.213547	0.000559	0.000559	0.000559	0.291167	0.648367
4	0.207855	0.000745	0.000745	0.000745	0.274417	0.627793
5	0.361545	0.000745	0.000745	0.000745	0.332409	0.696277
6	0.392322	0.000914	0.145284	0.000876	0.368689	0.567291
7	0.382276	0.0011	0.170745	0.00104	0.460931	0.658153
8	0.409971	0.264837	0.173201	0.00104	0.467418	0.544346
9	0.43587	0.274601	0.176104	0.001208	0.456507	0.537575
10	0.435791	0.27453	0.193839	0.001377	0.520766	0.60453

Table 12: V-measure scores of all algorithms

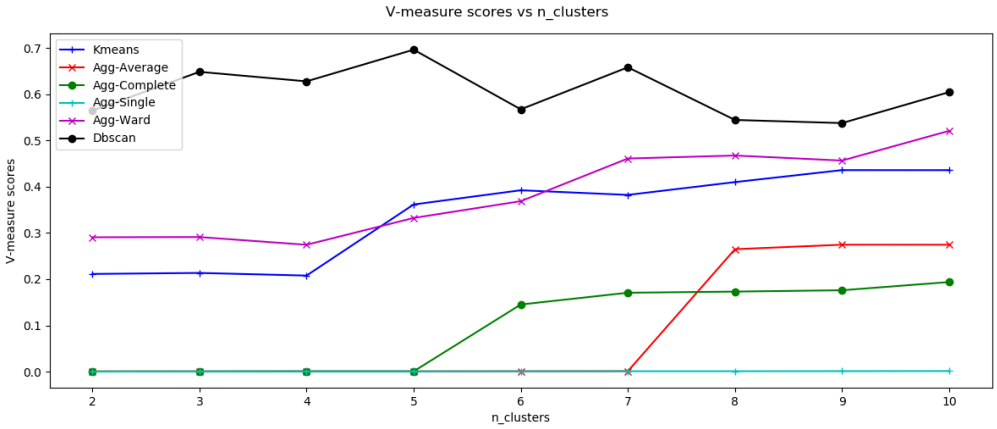


Figure 25: V-measure scores vs n_clusters (k)

Observations:

- We observe that Dbscan gives the best results over other algorithms for all the values of k, and for all three metrics.
- Next best is Agglomerative ward and K-means. However here Agglomerative ward performing better than K-means. But notice that only for k=5 and k=6, k-means performs better than Agglomerative ward. (following the same trend of unsupervised metrics)
- All other algorithms give poor results.

5.2.3 Best Configuration Selection

In previous section, we observed that though Dbscan gives good results for supervised metrics but it performs the worst with unsupervised metrics i.e. not even giving average results to consider.

Clustering is an unsupervised machine learning approach, we cannot conclude a solution on the basis of algorithms which are working well with supervised metrics because in reality we might not be having labeled ground truth to predict the outcome. In real world, labeled data is not very common and therefore unsupervised metrics should be given preference to select best configuration.

Also, high CH index states that there might be a possibility that the clusters are more or less spherical, convex shaped.

After comparing all the results of different metrics, we can say that K-means and Agglomerative ward can be a good solution for this particular dataset especially at k=5 and 6. We shall now discuss about the same for further analysis.

Table 13 shows the best scores (highlighted in green) of K-means and Agglomerative ward for all the metrics together at k=5 and 6. We see that somewhere k=5 score is higher than k= 6 or vice-versa.

Unsupervised Metrics			Supervised Metrics		
N_cluster (k)	Kmeans	Agg-Ward	N_cluster (k)	Kmeans	Agg-Ward
Silhouette score			Adjusted rand score		
5	0.297781	0.219781	5	0.258855	0.213385
6	0.297761	0.233841	6	0.26135	0.20368
Calinski-Harabasz Index			Homogeneity		
5	10491.87	8913.215	5	0.34467	0.31224
6	10060.51	8346.051	6	0.39855	0.372131
Sum of squared errors (SSE)			Completeness		
Appropriate number at the knee at (k = 4) or (k=5) for both the algorithms			5	0.380157	0.355364
			6	0.386285	0.365311
			V-measure		
			5	0.361545	0.332409
			6	0.392322	0.368689

Table 13: Best scores of supervised and unsupervised metrics

Though both Kmeans and Agglomerative ward gives consistently good results through all the values of K, for all the metrics but they are more consistent at (k=5) and (k=6). They are more stable and reliable.

However k-mean scores higher than Agglomerative ward and produces best average results for both supervised and unsupervised metrics.

Hence, best configuration could be:

- 1st best solution:
  K-means with (K=5) or (K=6)
- 2nd best solution:
  Agglomerative Ward with (K=5) or (K=6)

Table 14 shows total number of data points in each cluster for both algorithms.

	Kmeans		Agglomerative Ward	
Cluster Label	K=5	K=6	K=5	K=6
0	4853	2246	5265	5967
1	3040	2039	5967	3156
2	3529	3883	3	3
3	3	3352	2094	2094
4	5437	3	3533	3533
5	NA	5339	NA	2109

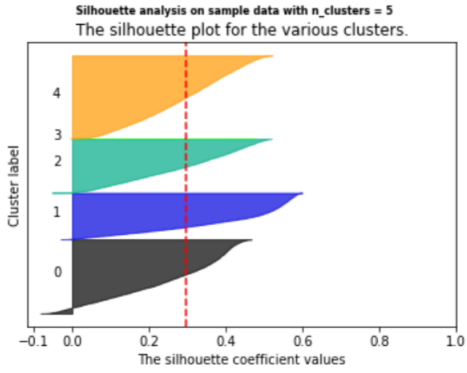
Table 14: Total points in each cluster

Below figures show the Silhouette plot of both algorithms for the best configuration selected.

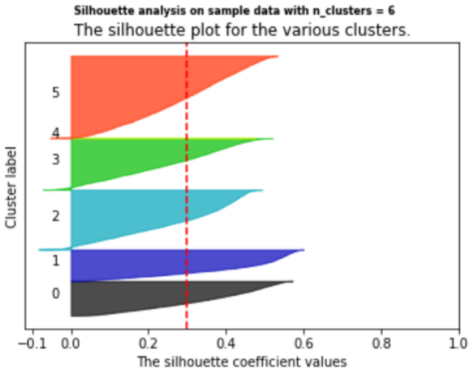
For this to be a good value for a particular number of cluster, we must consider the following points: First, the mean value should be as near to 1 as possible. Second, the plot of every cluster should be above, as much as possible, the mean value. Plot region below the mean value is not preferable. Last, the width of plot should be uniform.

The plot shows a comparison of the density and separation of each cluster. For K-means, both the plots (k=5 and 6) looks quite similar. We need to choose value of k yielding the highest average silhouette width, over all the values of k. Considering the fact, K =5 seems more uniform. Also, the density and separation is optimal. Similar observations with Agglomerative ward.

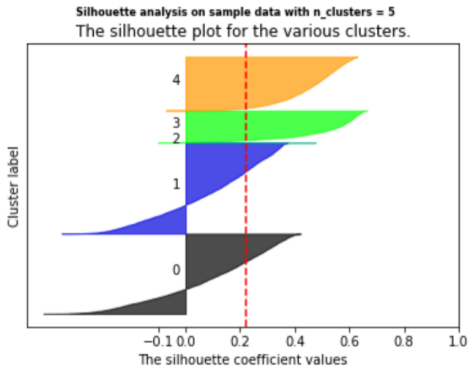
Also, it is easy to identify the outlier cluster through the plot. For example, for k-means, cluster 3 is the outlier cluster as it contains least number of data point (see Table 14) and not able to form a dense region like other clusters.



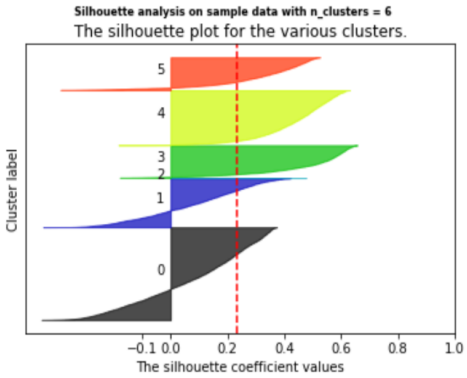
(a) Silhouette plot for k-means (k=5)



(b) Silhouette plot for k-means (k=6)



(a) Silhouette plot for Agg-Ward (k=5)



(b) Silhouette plot for Agg-Ward (k=6)

5.3 Cluster characterization

Basically, characterization helps in identifying the type of cluster, its features and data distribution of patterns. It also identifies outlier clusters. Cluster characterization helps domain expert in labelling of the data, in our case to robot cycles.

5.3.1 Feature selection

First, we extract top 10 most relevant features of our clusters, using Decision Tree Classifier, and then individually characterize each group. This helps domain experts to take better decisions about the data by going through the most relevant features that better describes each group.

After choosing these most relevant features, their data distribution has been shown using boxplots, which will help to identify the most characterizing properties of a cluster in terms of relevant features and content. This information may help experts to understand every group thoroughly because it is impossible to inspect all the samples manually. Therefore, cluster characterization is an important support to take better decisions.

Table 15 show the Top 10 most relevant features (IDs) and their feature values extracted using Decision Tree classifier, for both algorithms (best configuration only).

	Kmeans				Agglomerative Ward			
Index	K= 5		K =6		K=5		K=6	
	Feature_ID	Feature_Val	Feature_ID	Feature_Val	Feature_ID	Feature_Val	Feature_ID	Feature_Val
1	149	0.306832	171	0.244474	171	0.310735	137	0.245883
2	205	0.278115	140	0.243844	170	0.177038	214	0.185894
3	141	0.188972	209	0.178203	137	0.137506	170	0.181311
4	170	0.041625	170	0.172649	182	0.101088	109	0.118833
5	119	0.018966	205	0.027689	203	0.081723	128	0.025098
6	140	0.014246	143	0.00743	184	0.01562	178	0.020994
7	206	0.009987	121	0.007427	128	0.011306	205	0.015852
8	126	0.009129	64	0.005504	56	0.009744	171	0.015321
9	175	0.008742	161	0.005365	166	0.009533	185	0.009837
10	147	0.008382	154	0.004653	125	0.007719	56	0.009499

Table 15: Top 10 most relevant features (IDs) for both algorithms

5.3.2 Interpreting box plots

K-means (k=5)

Table 16 shows Top 10 most relevant features for k-means (k=5), Feature ID and the corresponding feature name. The data tells that these are the main features or information that can help domain experts in measurement to predict the belt tensioning level.

Feature_ID	Feature_Name
149	max_15
205	std_8
141	third_quartile_16
170	abs_energy_14
119	first_quartile_23
140	mean_abs_change_3
206	std_11
126	mean_abs_change_12
175	median_13
147	mean_14

Table 16: Top 10 most relevant features for k-means (k=5)

Fig 28 shows the boxplot for k-means (k=5). Cluster 0 is more close to limit (0, 2) , cluster 1 (-1,-2), cluster 2 (0,-1) and cluster 4 (1,-1). This tells us that features are well separated.

Feature_ID 206 (std_11) has higher distribution in cluster 1 than others.

Also, it can be easily identified that cluster 3 is an outlier cluster, more closely shown in Fig 29

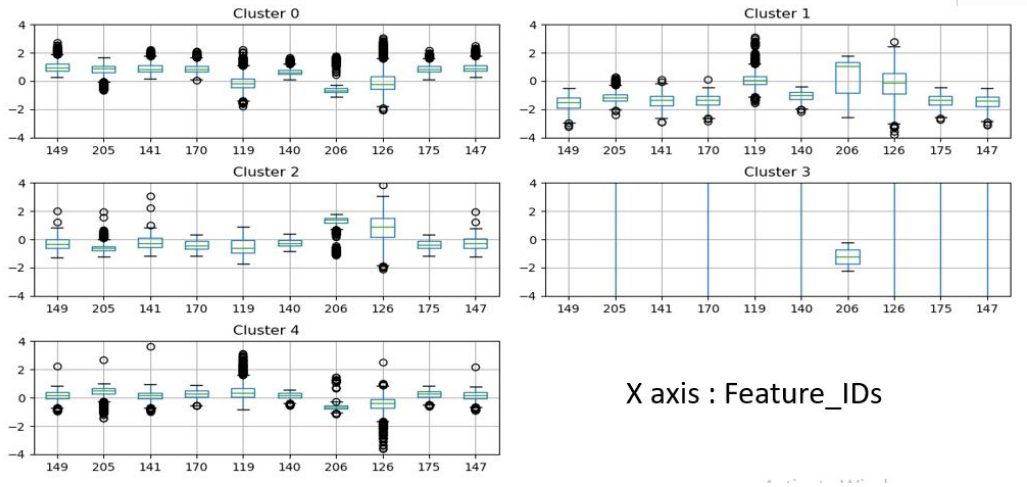


Figure 28: Boxplot for k-means (k=5)

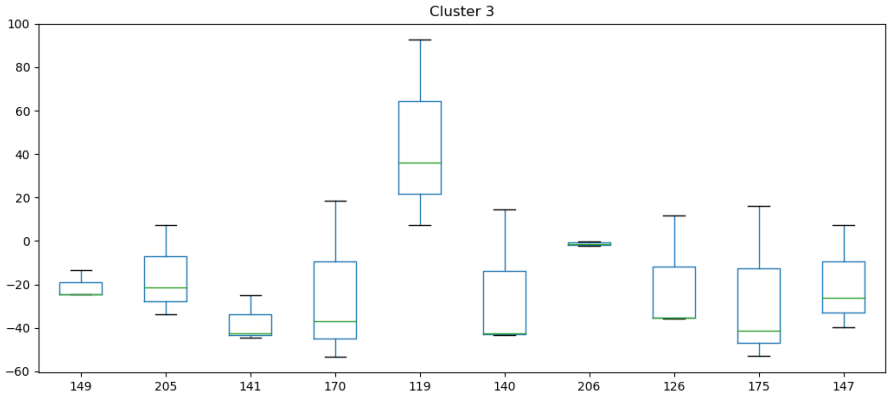


Figure 29: Outlier cluster for k-means (k=5)

K-means (k=6)

Similarly, Table 17 shows Top 10 most relevant features for k-means (k=6), Feature ID and the corresponding feature name.

Feature_ID	Feature_Name
171	first_quartile_19
140	mean_abs_change_3
209	mean_16
170	abs_energy_14
205	std_8
143	first_quartile_15
121	first_quartile_23
64	std_10
161	abs_values_sum_11
154	third_quartile_14

Table 17: Top 10 most relevant features for k-means (k=6)

Fig 30 shows the boxplot for k-means (k=6). Cluster 0 is more close to limit(1,-1) , cluster 1 (0,-2), cluster 2 (0,1), cluster 3 (0,-1) and cluster 5(1,-1). This tells us that features are well separated.

Feature_ID 209 (mean_16) has higher distribution in cluster 1 and 3 than others.

Also, it can be easily identified that cluster 4 is an outlier cluster, more closely shown in Fig 31

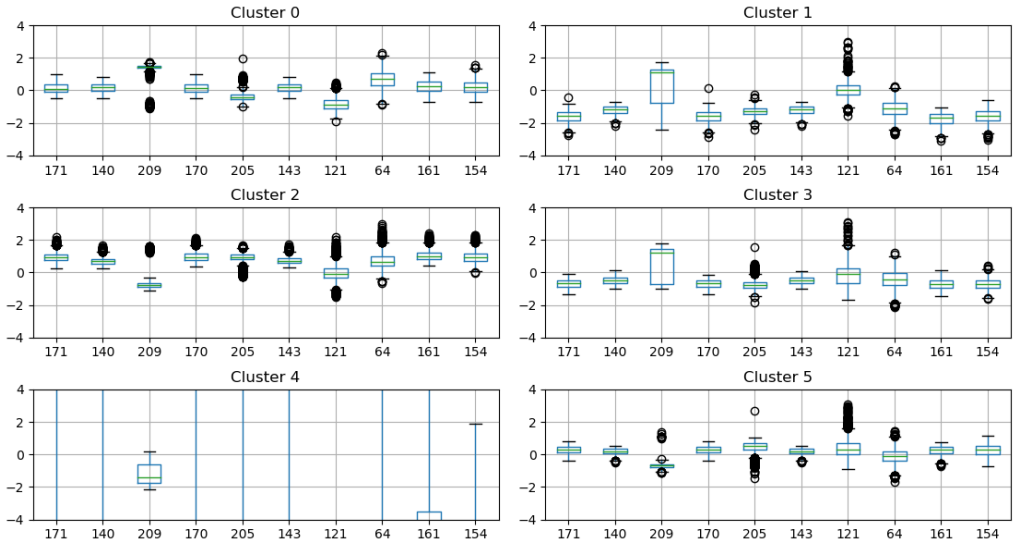


Figure 30: Boxplot for k-means (k=6)

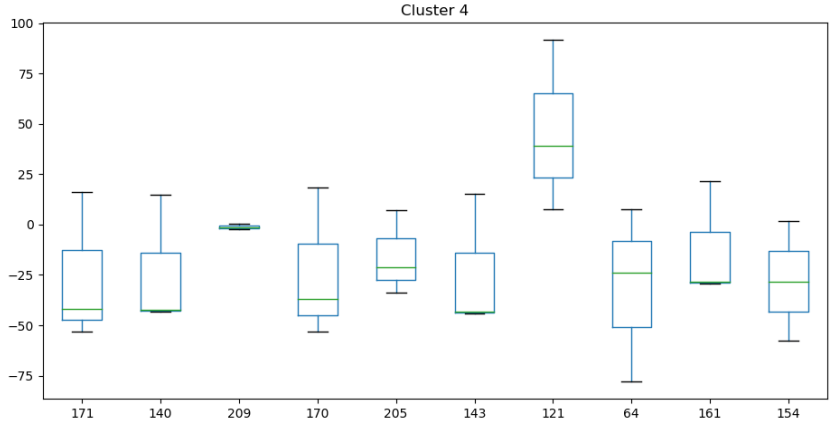


Figure 31: Outlier cluster for k-means (k=6)

Agglomerative-Ward (k=5)

Table 18 shows Top 10 most relevant features for Agglomerative-Ward (k=5), Feature ID and the corresponding feature name.

Feature_ID	Feature_Name
171	first_quartile_19
170	abs_energy_14
137	abs_values_sum_23
182	RMS_11
203	max_10
184	abs_values_sum_13
128	RMS_12
56	skewness_2
166	abs_energy_15
125	mean_23

Table 18: Top 10 most relevant features for Agglomerative-Ward (k=5)

Fig 32 shows the boxplot for Agglomerative-Ward (k=5). Cluster 0 is more close to limit(1,-1) , cluster 1 (1,-1) and cluster 3 (1,-2) and cluster 4 (1,-1).

Feature_ID 56 (skewness_2) has higher distribution in cluster 0 and cluster 1 than others.

Also, it can be easily identified that cluster 2 is an outlier cluster, more closely shown in Fig 33

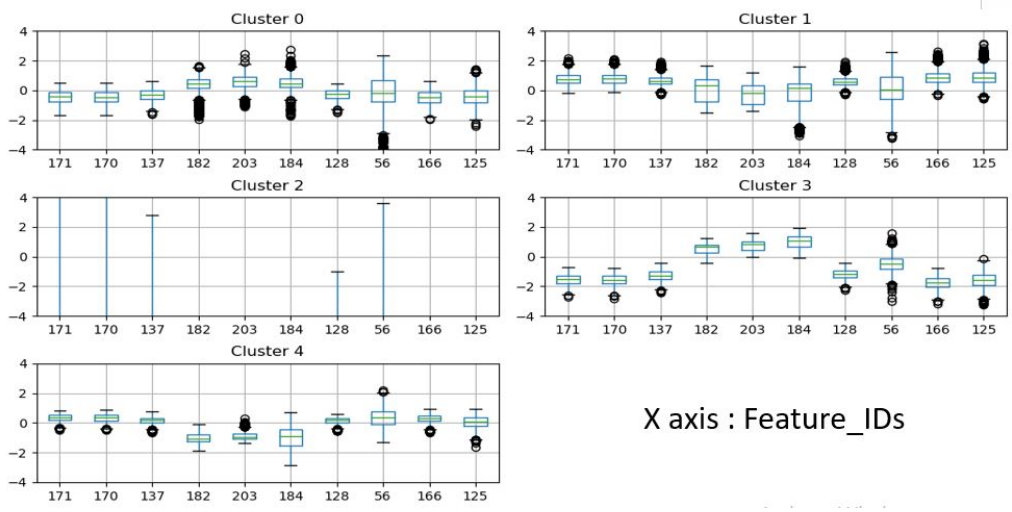


Figure 32: Boxplot for Agglomerative-Ward (k=5)

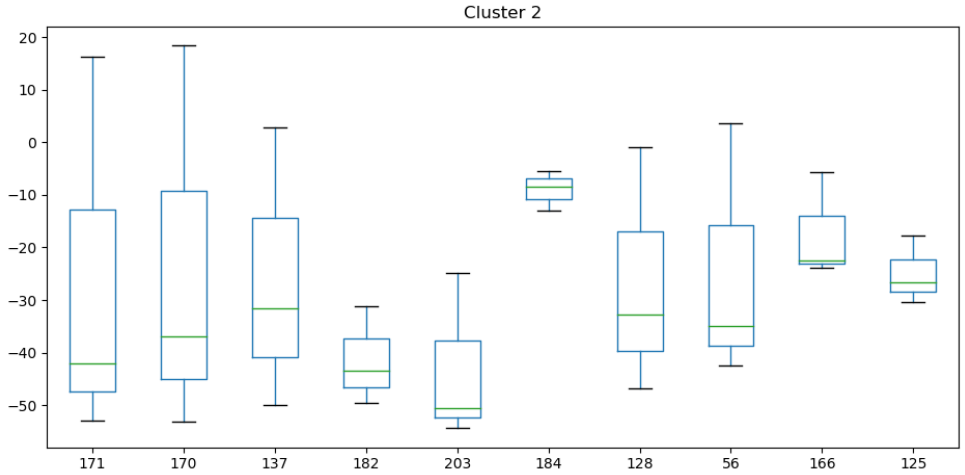


Figure 33: Outlier cluster for Agglomerative-Ward (k=5)

Agglomerative-Ward (k=6)

Table 19 shows Top 10 most relevant features for Agglomerative-Ward (k=6), Feature ID and the corresponding feature name.

Feature_ID	Feature_Name
137	abs_values_sum_23
214	first_quartile_16
170	abs_energy_14
109	skewness_16
128	RMS_12
178	kurtosis_4
205	std_8
171	first_quartile_19
185	min_19
56	skewness_2

Table 19: Top 10 most relevant features for Agglomerative-Ward (k=6)

Fig 34 shows the boxplot for Agglomerative-Ward (k=6). Cluster 0 is more close to limit(1,-1) , cluster 1 (1,-1) and cluster 3 (1,-2), cluster 4 (1,-1) and cluster 5(1,-1).

Feature_ID 178 (kurtosis_4) has higher distribution in cluster 0 than others.

Also, it can be easily identified that cluster 2 is an outlier cluster, more closely shown in Fig 35

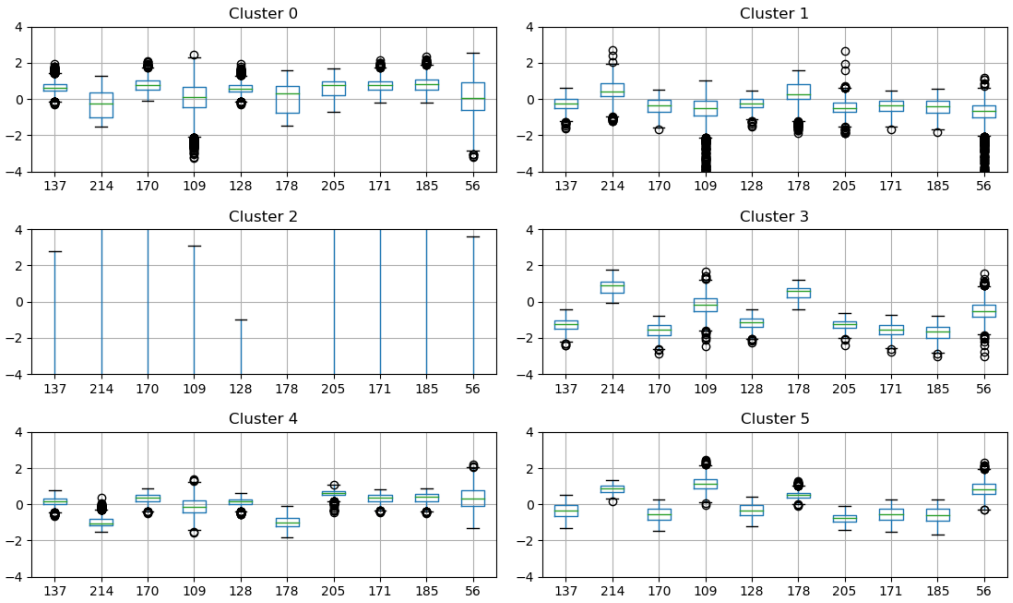


Figure 34: Boxplot for Agglomerative-Ward (k=6)

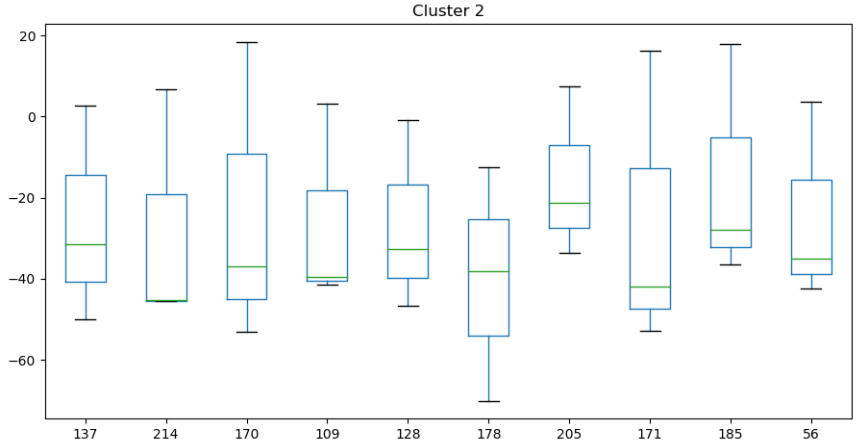


Figure 35: Outlier cluster for Agglomerative-Ward (k=6)

Looking at figure 28, 30, 32 and 34 the most relevant segments correspond to n between 11 to 17 and other portion between 20 to 24 where n is the number of seconds for smart data extracted from time domain feature Computation (see figure 36).

Figure 36 shows the electric signal of a single production cycle with relevant segments highlighted in green.

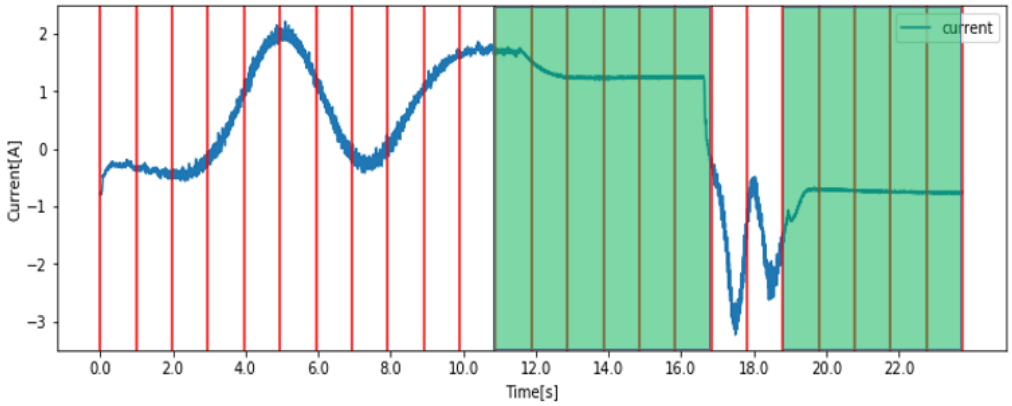


Figure 36: Electric signal with relevant segments highlighted

The production cycle works as follows. In the beginning, the angle is positioned at -500 degrees and later reaches +90 degrees after 20% of max speed and maintains the position for about 5 seconds. Then, it comes back to -500 degrees. Lastly, it keeps the position for another 5 seconds.

The highlighted segment blocks shows electricity consumption when motor reaches and maintains +90 degrees and then returns back to initial position at -500 degrees.

6 Conclusion and Future improvements

This work presents a supporting solution to predictive maintenance analysis and discussed deeply in previous chapters. The predictive analytics has been measured to predict the belt tensioning level and to further support robot cycle labelling. A machine learning algorithm has been applied to smart data so as to forecast a tensioning level as per a new cycle of data. This helped in identifying clusters of production cycles through similar time independent features.

The goal was to forecast correct configuration of parameters and clustering algorithms through the estimation of cluster quality metrics in the semi-supervised context.

Clustering algorithm results have been interpreted by analyzing and comparing various supervised and unsupervised metrics. The experiments highlighted the best configuration selection on the basis of results obtained. The solution is based on the fact that clustering is an unsupervised machine learning approach, we cannot conclude a solution on the basis of algorithms which are working well with supervised metrics because in reality we might not be having labeled ground truth to predict the outcome. In real world, labeled data is not very common and therefore unsupervised metrics should be given preference to select best configuration. Also, a more stable trend tested in all possible different scenarios must be chosen.

Future activities of this study will focus on further research and investigation which is important to recognize the most appropriate algorithms for data driven predictive analysis and evaluating the outcomes. The predictive model should be able to analyze the new incoming unlabeled data that does not fit anymore the trained model with original distribution i.e. a model re-training with different parameters.

Also, degradation of model performance can be implemented which means triggering model updates in possibly all different scenarios. Our model should be a generalize one for all Industry 4.0 cases.

References

[1] D.Bowden A.Marguglio L.Morabito C.Napione S.Panicucci N.Nikolakis S.Makris G.Coppo S.Andolina A.Macii E.Macii N.O'Mahony P.Becker S.Jung "A cloud-to-edge architecture for predictive analytics".

[2] T.Cerquitelli A.Macii E.Macii M.Poncino D.Apiletti C.Barberis and F.Ventura. "iSTEP: an integrated Self-Tuning Engine for Predictive maintenance in industry 4.0".

[3] C. Zhang, P. Lim, A. K. Qin, and K. C. Tan. 2016. Multiobjective Deep Belief Networks Ensemble for Remaining Useful Life Estimation in Prognostics. IEEE Transactions on Neural Networks and Learning Systems PP, 99 (2016), 1–13. https://doi.org/10.1109/TNNLS.2016.2582798

[4] Radu F. Babiceanu and Remzi Seker. 2016. Big Data and virtualization for manufacturing cyber-physical systems: A survey of the current status and future outlook. Computers in Industry 81 (2016), 128 – 137. https://doi.org/ 10.1016/j.compind.2016.02.004 Emerging ICT concepts for smart, safe and sustainable industrial systems.

[5] B.Xu and S.A.Kumar. "Big data analytics framework for system health monitoring". In: IEEE International Congress on Big Data (2015), pp. 401 –408.

[6] Raghupathi, W. & Raghupathi, V. Health Inf Sci Syst (2014) 2: 3. https://doi.org/10.1186/2047-2501-2-3

[7] NumPy.org. url: https://numpy.org/.

[8] JobLib Developers. Joblib: running Python functions as pipeline jobs. url: https://joblib.readthedocs.io/en/latest/.

[9] Python Data Analysis Library. url: https://pandas.pydata.org/.

[10] JSON encoder and decoder. url: https : / / docs . python . org / 3 / library/json.html.

[11] Python Software Foundation. Python Language Reference, version 3.5. url: https://docs.python.org/3.5/.

[12] J. D. Hunter. “Matplotlib: A 2D graphics environment”. In: Computing in Science & Engineering 9.3 (2007), pp. 90–95. doi: 10.1109/MCSE.2007.55.

[13] Bisong E. (2019) Google Colaboratory. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform. Apress, Berkeley, CA

[14] Quazi Nafiul Islam. ”Mastering PyCharm”. Use Pycharm with fluid efficiency.

[15] Google Colaboratory. url: https://colab.research.google.com/notebooks/intro.ipynb#

[16] SciKit-learn, Machine Learning in Python. url: https : / / scikit - learn.org/stable/.

[17] P.Arabie, L.J.Hubert, G.De Soete ”Clustering And Classification”

[18] Shi Na, Liu Xumin, Guan Yong ”Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm”. In ”2010 Third International Symposium on Intelligent Information Technology and Security Informatics”

[19] Thanh N.Tran, Klaudia Drab, Michal Daszykowski ”Revised DBSCAN algorithm to cluster data with dense adjacent clusters” url: https://doi.org/10.1016/j.chemolab.2012.11.006

[20] Day, W.H.E. & Edelsbrunner, H. Journal of Classification (1984) 1: 7. ”Efficient algorithms for agglomerative hierarchical clustering methods” https://doi.org/10.1007/BF01890115

[21] S.R. Safavian, D. Landgrebe ”A survey of decision tree classifier methodology”. In ”IEEE Transactions on Systems, Man, and Cybernetics (Volume: 21 , Issue: 3 , May/Jun 1991)”