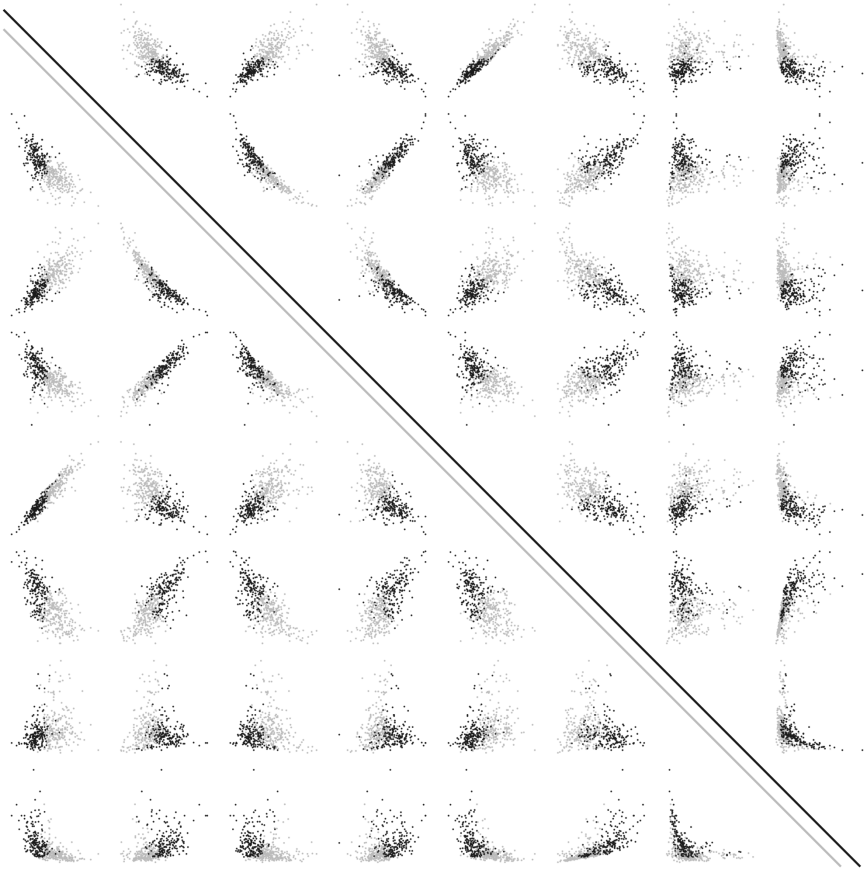


Defining, Analyzing, and Clustering Drive Cycles for Engine Applications Through Feature Engineering

Master's Thesis in Complex Adaptive Systems

Rasmus Hellrand
Jakob Malmer Göransson

DEPARTMENT OF PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Master's thesis 2025

Defining, Analyzing, and Clustering Drive Cycles for Engine Applications Through Feature Engineering

RASMUS HELLRAND
JAKOB MALMER GÖRANSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
Chalmers University of Technology
Gothenburg, Sweden 2025

Defining, Analyzing, and Clustering Drive Cycles for Engine Applications Through Feature Engineering
RASMUS HELLRAND
JAKOB MALMER GÖRANSSON

Supervisors: Mattias Roos & Erik Jansson, Volvo Penta
Examiner: Mohsen Mirkhalaf, Department of Physics

Master's Thesis 2025
Department of Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Pair plots of clustered drive cycles features

Typeset in LaTeX
Printed by TeknologTryck
Gothenburg, Sweden 2025

Defining, Analyzing, and Clustering Drive Cycles for Engine Applications Through Feature Engineering
RASMUS HELLRAND
JAKOB MALMER GÖRANSSON
Department of Physics
Chalmers University of Technology

Abstract

An ordered sequence of measurements, or time series, is the raw representation from which many analytics and data-driven decisions are made in the automotive industry. Volvo Penta is a company determined to become more data-driven to stay ahead of the competition. In this endeavor, signals from various manufactured engines in the field are logged as time series data. Through analysis of this data lies the information from which to obtain the repeated behavioral pattern of an engine in use, otherwise known as drive cycles. This thesis aims to segment time series data from an engine placed in a log stacker into drive cycles and engineer, extract, and select relevant features descriptive of engine usage from these. From the cycles with accompanying features, clustering techniques will be applied to group the drive cycles into behaviorally distinct categories. Drive cycles were defined by segmenting consecutive engine signals whenever the gap between recorded signal values exceeded ten minutes. Filtering was then applied to remove uninformative and outlier cycles. From the remaining cycles, features were derived, which, after selection, yielded eight temporal and statistical features that formed the basis for clustering. Using these features, the clustering algorithms k-means, agglomerative, and mean-shift were employed, resulting in clusters that depicted two main engine behaviors. These were high- and low-load behaviors. The exception was the resulting clusters produced by mean-shift, which exhibited a single dominant behavior and a few deviating cycles. A consensus partition integrated all three methods and agreed with the solution provided by k-means. This thesis demonstrates that feature-based, unsupervised clustering can group drive cycles segmented from engine field test data into behaviorally distinct categories. These findings further the goals of Volvo Penta's mission to become more data-driven and demonstrate the potential of clustering as a first step in areas such as engine simulation and predictive maintenance.

Keywords: agglomerative, consensus clustering, drive cycle, feature engineering, k-means, mean-shift, segmentation, time series, unsupervised clustering, Volvo Penta.

Acknowledgements

For the duration of this thesis, extensive support has been given by the Data Science & Engineering department at Volvo Penta. We want to express our sincere gratitude to Andreas Nyman, Mattias Roos, and Erik Jansson for providing more than sufficient support throughout the project, making this thesis a reality. An extra special thanks goes out to our supervisors, Mattias and Erik, for providing great experience and insights throughout the whole project.

Lastly, our thanks go out to Mohsen Mirkhalaf for agreeing to serve as our examiner for this thesis. He has shown great interest in our work and provided us with the necessary feedback to drive this project from start to finish.

Rasmus Hellrand & Jakob Malmer Göransson, Gothenburg, May 2025

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

CAN	Controller Area Network
CH	Calinski-Harabasz Index
DB	Davies-Bouldin Index
HAC	Hierarchical Agglomerative Clustering
MTS	Multivariate Time Series
PCA	Principal Component Analysis
PFA	Principal Feature Analysis
PC	Principal Component
RPM	Revolutions Per Minute
RPS	Revolutions Per Second
SAE	Society of Automotive Engineers
Sil	Silhouette Score
UTS	Univariate Time Series

Contents

List of Figures

List of Tables

1
Introduction

Volvo Penta (Penta) is a company known for manufacturing engines and complete power systems across various applications, mainly for boats and industrial machines. Penta stands at a crossroads where making data-driven decisions is paramount to staying ahead of the competition. As a result, the company must collect large amounts of data from the various types of products it manufactures. One form of data Penta collects is time series for various engine signals, and the company is exploring different ways to turn potential insights gained from this data into representative engine behaviors. Consequently, this project aims to explore statistically based methods for defining, analyzing, and clustering drive cycles.

1.1 Background

An ordered sequence of measurements, commonly referred to as a time series, provides fundamental information used for analysis and data-driven decision-making [1]. Uncovering the underlying structure of time series data is the central goal of time series analysis, and its applications span multiple industries, including healthcare, finance, and the automotive industry. In data analysis, various methods exist; the most commonly used method in the automotive sector is predictive maintenance, where historical sensor readings are used to forecast, for example, when engine components need replacement or the vehicle requires servicing. This thesis aims to explore another less frequently used method within the automotive industry: time series clustering, an unsupervised method used in categorizing data points produced by time series.

There are several approaches to time series clustering, each with its advantages and drawbacks. Common standard methods are raw-distance, model-based, and feature-based [2]. The latter, and what this project will focus on, is popular when processing extensive data. Here, the idea is to summarize the time series data into smaller characteristics (or features), preserving relevant information and reducing dimensionality. This technique has been applied to both univariate time series (UTS) and multivariate time series (MTS) across multiple disciplines, but is not yet utilized as extensively within the automotive industry [3].

One form of data collection performed by Penta comes from engines performing operational tasks, denoted as field test data. These signals are collected using CAN loggers, and the types of signals collected range from engine torque and speed to

fuel consumption, all of which describe engine usage. The repeated behavioral pattern of an engine is referred to as a *drive cycle*. Currently, there is no statistically or behaviorally grounded definition that quantifiably defines an engine drive cycle. In current practice, the definition used at Penta is typically determined by a user-inputted value representing the time between an engine start and stop, marking the separation between different cycles.

The project is relevant for Penta, where the ability to engineer meaningful features for the drive cycles can be used to advance engine usage simulations or enhance future product requirements. Segmenting drive cycles based on different engine applications and types of work performed enables the identification of specific operational patterns relevant to each application. For instance, improving the accuracy of engine simulations under varying conditions paired with their representative drive cycle, or aiding in predictive maintenance by identifying potential failures before they occur. Furthermore, understanding the distinctive patterns of different cycles can help tailor engine designs to specific use cases and ensure that engines are used for the appropriate applications based on their operational characteristics.

1.2 Aim and Objectives

The thesis aims to develop a comprehensive understanding of drive cycle characterization through statistical and data-driven methods. The primary objectives include identifying an appropriate drive cycle definition, determining relevant signals, and applying feature engineering techniques to derive characteristics that will be used to group cycles into behaviorally distinct clusters. These clusters could, for example, capture differences in how different engines experience high or low torque demand from the driver. Clustering will be achieved by applying machine-learning methods, chosen based on their ability to accurately differentiate between drive cycle patterns derived from the selected features. The ultimate goal is to explore drive cycle analysis, which will be the first step in improving areas of vehicle modeling, optimization, and engine simulation. The objectives in full are listed below:

- **Objective 1: Define a drive cycle**
  Develop a framework based on statistics and/or behavior to define a drive cycle and segment the time series into cycles based on the framework.
- **Objective 2: Identify representative signals for cycle explanation**
  Identify and evaluate key engine signals that best represent drive cycles.
- **Objective 3: Engineer, extract, and select meaningful features that characterize drive cycles**
  Find robust features that capture the essential characteristics of drive cycles.
- **Objective 4: Cluster cycles into interpretable categories for further analysis**
  Apply clustering to group cycles into meaningful behaviorally distinct categories.

1.3 Scope and limitations

The project's scope is limited to analyzing field test data provided by Penta. Furthermore, segmentation and clustering will be performed on the collected data for a single engine used in a specific type of application, as the general usage of an engine can vary significantly depending on the application. Signals will be limited to those most commonly used to describe engine usage: engine torque, speed, and power. The engine chosen for analysis will also be from the industrial segment, rather than the marine segment, as there are more parameters to consider when examining marine engines. The engine chosen is configured to communicate via the SAE J1939-71 protocol, the industry standard for heavy-duty vehicle networks. Consequently, all data signals conform to the Backbone 1 (BB1) specification, ensuring that results are directly reproducible for any BB1-compliant engines. Lastly, the features produced and the clustering should be explainable. This means choosing non-arbitrary features that are commonly used by engine researchers, as well as being able to explain the contribution made by each feature to each respective cluster.

1.4 Specification of the Issue Being Investigated

The thesis aims to answer the following questions:

- What signals best describe engine behavior?
- What features are most relevant for characterizing drive cycles?
- Which clustering algorithms yield the most coherent, interpretable groups?
- How do different clustering techniques compare in terms of performance and interpretability?
- Are the results representative of real-world driving patterns?

1.5 Related Work

This thesis work builds upon insights gained from another thesis, conducted by our supervisor, Mattias Johansson, and his thesis partner, Karl-Fredrik Zingaropoli, in 2021 [4]. Their goal was to detect drive cycles for various engine applications. They utilized supervised learning to detect types of cycles after applying feature extraction on time series data. Their work consisted of three approaches: detection, semantic segmentation, and generating engine drive cycles using a genetic algorithm. In contrast, our approach utilizes unsupervised learning and statistical techniques to define and segment drive cycles. Unlike their project, which used a genetic algorithm to create representative drive cycles, this thesis aims to use clustering methods to differentiate between different types of engine cycles across specific engine applications. This approach is unsupervised, which may uncover new engine behaviors that have not been previously examined.

Another approach to MTS clustering is Time2Feat [5]. This system was designed

with interpretability in mind, enhancing explainability by extracting both inter-signal and intra-signal features, and then applying dimensionality reduction to retain essential information. This improves clustering transparency while preserving key patterns in the data. Our approach utilizes the Principal Feature Analysis (PFA) method to achieve interpretable clustering results that remain within the original feature space.

2
Theory

This chapter provides the theoretical framework and methods used throughout the thesis. The sections covered are: *Time Series*, *Feature Engineering*, *Clustering*, *Evaluation Metrics for Clustering*, *Engine Signals*, and *Feature Selection*.

2.1 Time Series

A time series can be regarded as a sequence of data points recorded at successive points in time, and it is used as a representation of one or a set of variables evolving over time [6]. There are two main types of time series, univariate and multivariate.

2.1.1 Univariate Time Series

For the case of a single variable, the time series is referred to as a univariate time series (UTS). A univariate time series is represented as a sequence of scalar observations:

$$\{x_t\}_{t=1}^{T}, \quad \text{with } x_t \in \mathbb{R}. \tag{2.1}$$

Here, T denotes the number of time points.

Alternatively, it can be written as a column vector:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_T \end{pmatrix} \in \mathbb{R}^T.$$

2.1.2 Multivariate Time Series

Multivariate time series (MTS) data comprises multiple time-dependent variables. For example, two or more engine signals sampled simultaneously for the same system [7]. The variables can have different measurement counts, periodicity, and sampling frequencies. This can be expressed as a sequence of vectors:

$$\{\mathbf{x}_t\}_{t=1}^{T}, \quad \text{where } \mathbf{x}_t = \begin{pmatrix} x_{1,t} \\ x_{2,t} \\ \vdots \\ x_{p,t} \end{pmatrix} \in \mathbb{R}^p,$$

with T denoting the number of time points and p the number of variables.

Alternatively, the entire series can be represented as a matrix:

$$X \in \mathbb{R}^{T \times p},$$

where each row of X corresponds to a time point and each column corresponds to a variable.

2.2 Feature Engineering

Feature engineering transforms raw measurements into quantitative descriptive measures that capture underlying system behavior and facilitate downstream analysis, such as clustering or classification [8]. There are many different methods for performing feature engineering, and for time series data, input from domain experts on what features are important is essential. The following subsection covers features that are descriptive of engine usage.

2.2.1 Features Descriptive of Engine Usage

When examining drive cycles, the most common practice is to analyze data from the entire vehicle. These vehicles are usually road vehicles and not used for industrial purposes [9]. The Society of Automotive Engineers (SAE) is an organisation that produces standards for the automotive industry and defines several descriptive features for a drive cycle. Most of these features relate to velocity and acceleration, as well as the number of starts and stops, and duration in idle.

Penta is a company that strictly works with engines. Therefore, signals such as velocity and acceleration are not present in the logged data. However, several features can still be used to describe drive cycles based solely on engine signals. The relevant ones for this project were chosen based on the input of Penta experts and are listed below in Table 2.1. "Signal" implies calculations for all respective signals: Speed, Torque, and Power.

Table 2.1: Features describing engine usage with mathematical definitions.

Feature	Mathematical Definition
Mean of engine signal	$\bar{S} = \frac{1}{N}\sum_{i=1}^{N} S_i$
Standard deviation of engine signal	$\sigma_S = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(S_i - \bar{S}\right)^2}$
Skewness of engine signal	$\mathrm{Skew}(S) = \frac{\frac{1}{N}\sum_{i=1}^{N}\left(S_i - \bar{S}\right)^3}{\sigma_S^3}$
Kurtosis of engine signal	$\mathrm{Kurt}(S) = \frac{\frac{1}{N}\sum_{i=1}^{N}\left(S_i - \bar{S}\right)^4}{\sigma_S^4}$
Positive derivative of engine signal	$\left(\frac{dS}{dt}\right)_+ \approx \max_i \frac{\Delta S_i}{\Delta t_i} \quad (\Delta S_i > 0)$
Length of cycle	$L = t_{\mathrm{end}} - t_{\mathrm{start}}$
Fraction of idle time	$\frac{T_{\mathrm{idle}}}{L}$
Number of idle periods in cycle	n_{idle}
Idle-period duration	$T_{\mathrm{idle}} = \sum_{j=1}^{n_{\mathrm{idle}}}\left(t_{\mathrm{stop},j} - t_{\mathrm{start},j}\right)$

2.3 Clustering

Clustering is a technique used in unsupervised learning that groups data points into clusters based on inherent similarities and does not rely on labeled data [10]. It can be used to identify natural data structures or patterns that can be used for data segmentation or anomaly detection.

2.3.1 Euclidean Distance

The Euclidean distance is the distance between two points in Euclidean space and can be defined as follows:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n}\left(x_i - y_i\right)^2}.$$

Equivalently, using the euclidean norm $\|\cdot\|$, this can be written concisely as

$$d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|.$$

2.3.2 K-Means

When faced with a dataset consisting only of unlabeled data, it can be beneficial to identify similar groups of data within the dataset, also known as clusters. When deciding between clustering algorithms, k-means is a common option. In k-means, a

single data point belongs to a particular cluster if it is closest to the cluster centroid [10]. The steps for k-means are explained in Algorithm 1 below.

Algorithm 1 K-means clustering.

procedure K-MEANS(X, k) ▷ X is the set of data points, k is the number of clusters
  Input: Set of data points $X = \{x_1, x_2, \ldots, x_n\}$, number of clusters k
  Arbitrarily choose k initial centers $C = \{c_1, c_2, \ldots, c_k\}$.
  repeat
    for each $i \in \{1, 2, \ldots, k\}$ **do**
      Assign each point x to the cluster C_i if:

$$\|x - c_i\| \leq \|x - c_j\| \quad \text{for all } j \neq i$$

    for each $i \in \{1, 2, \ldots, k\}$ **do**
      Update the cluster center c_i as the centroid of cluster C_i:

$$c_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$$

  until centers C no longer change
  return The set of clusters $C = \{C_1, C_2, \ldots, C_k\}$

K-Means is a widely used clustering method that works well for clusters with a roughly hyper-spherical shape, but it struggles with clusters that have irregular or complex shapes [10]. It requires the number of clusters, k, to be specified beforehand. Additionally, while k-means is a popular centroid-based method that minimizes intra-cluster variance, the random initialization of cluster centers can often lead to poor clustering results. This randomness may result in clusters with no points, or multiple centroids might be placed within the same cluster. One way to overcome this is by using k-means++ initialization, which accelerates convergence by enhancing the initial cluster centroid selection [10].

K-means++ distributes the starting points based on each point's contribution to the overall inertia, beginning with the first cluster center randomly selected from the data. For the remaining centers, the next center is chosen according to its probability distribution, where the probability is proportional to the square of the distance between the data point and the closest selected center. Greedy k-means++ is an enhanced version of the standard k-means++ algorithm that conducts multiple trials at each step of centroid selection [11]. The centroid that maximally reduces inertia is chosen from these trials, which often results in better clustering results.

The algorithms for k-means++ and Greedy k-means++ are detailed in Algorithms 2 and 3 [12]:

Algorithm 2 K-means++ clustering.

procedure K-MEANS++(X, k) ▷ X is the set of data points, k is the number of clusters
  Input: Set of data points $X = \{x_1, x_2, \ldots, x_n\}$, number of clusters k
  Output: Initial cluster centers $C = \{c_1, c_2, \ldots, c_k\}$
  Choose the first center c_1 uniformly at random from X.
  for $i = 2$ to k **do**
    for each data point $x \in X$ **do**
      Compute the shortest distance $D(x)$ from x to the closest center already chosen.
    Select the next center $c_i = x' \in X$ with probability:

$$P(x') = \frac{D(x')^2}{\sum_{x \in X} D(x)^2}$$

  Use the k centers selected as the initial centers for the standard k-means algorithm.
  Proceed with standard k-means steps:
    1. Assign points to the nearest center.
    2. Update centers as the centroid of the assigned points.
    3. Repeat until convergence.
  return The set of clusters $C = \{C_1, C_2, \ldots, C_k\}$

Algorithm 3 Greedy k-means++ clustering.

procedure GREEDY K-MEANS++(X, k, t) ▷ X is the set of data points, k is the number of clusters, t is the number of trials
  Input: Data points $X = \{x_1, x_2, \ldots, x_n\}$, number of clusters k, number of trials t
  Output: Initial cluster centers $C = \{c_1, c_2, \ldots, c_k\}$
  Choose the first center c_1 uniformly at random from X.
  for $i = 2$ to k **do**
    for each data point $x \in X$ **do**
      Compute the shortest distance $D(x)$ from x to the closest chosen center.
    Initialize an empty set of candidates S.
    for $j = 1$ to t **do** ▷ Perform t trials
      Sample a candidate center $c_i^{(j)}$ from X with probability:

$$P(x') = \frac{D(x')^2}{\sum_{x \in X} D(x)^2}$$

      Add $c_i^{(j)}$ to S.
    Choose the candidate c_i from S that minimizes the sum of squared distances to the closest center.
  Use the k selected centers to initialize the standard k-means algorithm:
    1. Assign points to the nearest center.
    2. Update centers as the centroid of the assigned points.
    3. Repeat until convergence.
  return The set of clusters $C = \{C_1, C_2, \ldots, C_k\}$

2.3.3 Hierarchical Clustering

Hierarchical clustering is a form of clustering that groups data points in a tree-like structure of nested clusters [13]. This approach is beneficial when the number of clusters is unknown. Hierarchical clustering can be categorized into two main types: agglomerative and divisive. Agglomerative is a bottom-up procedure, whereas divisive is a top-down procedure. The difference lies in whether the Algorithm starts with one datapoint as its own cluster or if all datapoints start as connected in one cluster. This thesis will utilize agglomerative clustering, and the following paragraphs will provide a more detailed view of how this method works.

Hierarchical agglomerative clustering (HAC) starts with each data point being its own cluster, and iteratively merges clusters based on similarity [14]. The similarity between clusters is determined using linkage criteria such as single-, complete-, average-, or ward linkage. The iterative merge process occurs until every data point is grouped in a single cluster or a stop criterion is met. The detailed steps for HAC can be found in Algorithm 4 below.

Algorithm 4 Agglomerative clustering algorithm.

procedure AGGLOMERATIVE_CLUSTERING(S, d) ▷ S: node labels, d: pairwise dissimilarities
  $N \leftarrow |S|$ ▷ Number of input nodes
  $L \leftarrow [\,]$ ▷ Output list
  size$[x] \leftarrow 1$ for all $x \in S$
  for $i \leftarrow 0$ to $N - 2$ **do**
    $(a, b) \leftarrow \text{argmin}_{(S \times S) \setminus \Delta}\, d$
    Append $(a, b, d[a, b])$ to L
    $S \leftarrow S \setminus \{a, b\}$
    Create a new node label $n \notin S$
    for each $x \in S$ **do**
      $d[n, x] = d[x, n] \leftarrow \text{Formula}(d[a, x], d[b, x], d[a, b], \text{size}[a], \text{size}[b], \text{size}[x])$
    size$[n] \leftarrow$ size$[a]$ + size$[b]$
    $S \leftarrow S \cup \{n\}$
  return L ▷ The stepwise dendrogram, an $((N - 1) \times 3)$-matrix

Note: Δ denotes the diagonal of $S \times S$, i.e., pairs (x, x) which are excluded to avoid self-merging. `Formula` represents the distance update rule defined by the chosen linkage method.

The advantage of hierarchical clustering is that it does not require specifying the number of clusters in advance, unlike k-means clustering [13]. For drive cycle analysis, where the specific behaviors within the cycles are unknown beforehand, hierarchical clustering allows for an exploratory approach to grouping similar cycles based on their characteristics. The results can be visualized as a dendrogram, a tree-like diagram representing the sequence of merges or splits. By cutting the dendrogram at different stages, different groupings can be obtained.

2.3.3.1 Linkage

Different linkage methods are used to calculate the similarity between clusters in hierarchical clustering. Ward linkage minimizes the variance within the clusters being merged. Average linkage uses the average distance between points in two clusters. Complete linkage maximizes the distance between points, while single linkage minimizes the distance between points [15].

The advantage of single-linkage clustering is that it can effectively trace irregular, non-elliptical shapes, but it is less applicable for data with noise and outliers. By contrast, complete-linkage is more robust to such anomalies, but it may bias results when the clusters are large or globular [16]. This often drives the groupings toward more compact, nearly spherical forms than single-linkage. Ward's linkage, which seeks to minimize the increase in within-cluster variance, is especially applicable for quantitative data and tends to be the least influenced by noise and outliers. The distance update formula and cluster dissimilarities for additional standard linkage methods are presented in Table 2.2.

Table 2.2: Agglomerative clustering schemes.

Name	Distance Update Formula	Cluster Dissimilarity
Single	$\min(d(I,K), d(J,K))$	$\min_{a \in A, b \in B} d[a,b]$
Complete	$\max(d(I,K), d(J,K))$	$\max_{a \in A, b \in B} d[a,b]$
Average	$\frac{n_I \cdot d(I,K) + n_J \cdot d(J,K)}{n_I + n_J}$	$\frac{1}{\lvert A \rvert \lvert B \rvert} \sum_{a \in A} \sum_{b \in B} d[a,b]$
Weighted	$\frac{d(I,K) + d(J,K)}{2}$	–
Ward	$\sqrt{\frac{(n_I + n_K) \cdot d(I,K) + (n_J + n_K) \cdot d(J,K) - n_K \cdot d(I,J)}{n_I + n_J + n_K}}$	$\sqrt{\frac{2\lvert A \rvert \lvert B \rvert}{\lvert A \rvert + \lvert B \rvert} \cdot \lVert \mathbf{c_A} - \mathbf{c_B} \rVert^2}$
Centroid	$\sqrt{\frac{n_I \cdot d(I,K) + n_J \cdot d(J,K)}{n_I + n_J} - \frac{n_I n_J \cdot d(I,J)}{(n_I + n_J)^2}}$	$\lVert \mathbf{c_A} - \mathbf{c_B} \rVert^2$
Median	$\sqrt{\frac{d(I,K)}{2} + \frac{d(J,K)}{2} - \frac{d(I,J)}{4}}$	$\lVert \mathbf{w_A} - \mathbf{w_B} \rVert^2$

I and J are two clusters that merge to form a new cluster. K is any other (distinct) cluster. Clusters I, J, K has n_I, n_J, and n_K number of elements.

2.3.4 Mean-Shift Clustering

Mean-shift clustering is a non-parametric, mode-seeking procedure that iteratively shifts each data point to the average of its neighbors for a chosen kernel [17]. A Gaussian or Epanechnikov kernel is often used, but other options exist, such as the flat kernel used in scikit-learn's implementation [11].

The simplest form of kernel is the flat kernel, which is defined as [18]:

$$F(x) = \begin{cases} 1, & \textit{if } \lVert x] \rVert \leq 1, \\ 0, & \textit{if } \lVert x \rVert > 1. \end{cases}$$

The only tuning parameter for mean-shift clustering is the bandwidth, which corresponds to the radius of the kernel and thus determines the level of detail (cluster granularity) [17]. With larger bandwidth, nearby modes merge, and a smaller bandwidth yields tighter clusters.

One advantage of mean-shift clustering is that it makes no strong model assumptions besides choosing a kernel [19]. It can also determine clusters of non-convex, arbitrary shapes, and does not require a predefined number of clusters. The algorithm is also robust to outliers that, in the worst case, form clusters of single data points. One downside to the algorithm is that kernel density tends to break down for higher dimensions, meaning that mean-shift might fail beyond low-dimensional data.

2.3.5 Consensus Clustering

Different clustering methods, or the same with different initial conditions, capture various aspects of the provided input data. In real-world data, clusters often vary

in size and shape. They are affected by noise, making it challenging to identify a single clustering algorithm that performs optimally across all cases. This is especially true when little is known about the actual structure of the data. Moreover, evaluating the chosen clustering algorithm remains challenging since clustering is an unsupervised problem. To combine the strengths of multiple clustering algorithms and determine one solution that satisfies the results of many, one method that can be utilized is consensus clustering [20].

The input is created starting with different independent clustering results, which are generated using various algorithms or by adjusting the initial parameters. Representing each data point as a vector with cluster labels, from each respective clustering run, the consensus process can begin, and here different methods can be used depending on each user case:

2.3.5.1 Co-Association Matrix

When the partitions are generated for the dataset, the first step to creating the co-association matrix is to count the number of times a pair of data points appear in the same cluster across the cluster partitions (ensemble) [20]. For N clusterings in the ensemble, n_{ij} is the number of times two points occur in the same cluster in N.

After this step, to get the similarity between the partitions, each pair's co-occurrence count is normalized by the total number of clusterings [20]. This results in the co-association matrix C being given by:

$$C(i,j) = \frac{\mathrm{n}_{ij}}{N}. \tag{2.2}$$

After the normalization, the values will range from 0 to 1, indicating that a pair of points never co-occur to always co-occurring across the partitions. When formed, the co-association matrix can be combined with methods based on pairwise distance matrices, such as HAC, obtaining the final partition [20].

2.3.5.2 Voting/Relabeling

The voting and relabeling methods aim to aggregate the different clusterings and align and merge them into a single consensus cluster [21]. This is done by comparing the class prediction from different clustering results to determine a labeling scheme that maximizes agreement between the clusterings. One common approach is to obtain the most frequent classification of a data point and relabel it to agree with that particular class, which is called majority vote.

2.3.5.3 Graph-Based Method

Graph-based consensus clustering methods transform an ensemble of clusterings into a graph structure, allowing the aggregation of multiple clustering results into a single consensus solution [21]. In these methods, data points are represented as vertices, and edges or hyper-edges capture relationships derived from the clustering ensemble, such as similarity between data points or membership in the same cluster. The

goal is to obtain an optimal partition of the graph that best reflects the collective information from the ensemble.

These methods utilize graph partitioning techniques to achieve consensus, often focusing on optimizing criteria such as minimizing the cut size (the number of edges between different clusters) or maximizing intra-cluster similarity [21]. The transformation into a graph-based representation provides flexibility in capturing complex relationships and enables well-established graph partitioning algorithms to compute the consensus clustering efficiently. As a result, graph-based methods are particularly effective when the clustering ensemble contains diverse or conflicting results, as they robustly integrate multiple perspectives into a coherent consensus.

2.4 Evaluation Metrics for Clustering

Clustering, in contrast to supervised clustering, is an inherently unsupervised problem. Therefore, evaluating the resulting clusters is difficult, as the results cannot be compared to any true label. However, the problem is not impossible, and the following subsections explain commonly used evaluation metrics for clustering.

2.4.1 Silhouette Score

Silhouette score (Sil) is a metric used to evaluate clustering performance [22]. Using the mean intra-cluster distance a and mean nearest-cluster distance b, the silhouette coefficient can be calculated according to 2.3:

$$Sil = \frac{b - a}{max(a, b)} \tag{2.3}$$

Sil yields a sample between 1 and -1. A value of 1 shows datapoints are generally assigned to the correct cluster, and -1 means datapoints are generally assigned to the wrong cluster, given that another more suitable cluster exists.

2.4.2 Davies-Bouldin Index

Davies-Bouldin Index (DB) is commonly used to assess cluster compactness and separation, where lower values indicate better clustering [10]. DB is calculated by first defining the similarity, $\frac{inter}{intra}$-cluster distances [23]. The similarity value between each cluster and its closest neighbor is calculated and averaged over all clusters. The equation for a dataset X with k clusters is given by [24]:

$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{j \neq i} \left(D_{i,j} \right). \tag{2.4}$$

Here $D_{i,j}$ denotes the ratio $\frac{inter}{intra}$-cluster distances.

Compared to other methods, DB is flexible as it does not depend on the shape of clusters, like Sil, and it can be utilized independently of the number of clusters [10].

2.4.3 Calinski-Harabasz Index

The Calinski-Harabasz Index (CH), measures the quality of a clustering by comparing the dispersion between clusters to the dispersion within a cluster [25]. Similarity to its own cluster is quantified by the within-cluster dispersion W_k and is defined as:

$$W_k = \sum_{q=1}^{k} \sum_{x \in C_q} (x - c_q)(x - c_q)^T \tag{2.5}$$

where C_q is the set of points in cluster q and c_q is the centroid of cluster q.

The similarity to the other clusters is quantified by the between-cluster dispersion B_k and is given by:

$$B_k = \sum_{q=1}^{k} n_q (c_q - c_E)(c_q - c_E)^T \tag{2.6}$$

where n_q is the number of points in cluster q, c_q is the centroid of cluster q, and c_E is the global centroid of the dataset E.

Subsequently, the CH score is defined as:

$$CH = \frac{\mathrm{tr}(B_k)}{\mathrm{tr}(W_k)} \times \frac{n_E - k}{k - 1} \tag{2.7}$$

where:

- n_E is the total number of data points,
- k is the number of clusters,
- $\mathrm{tr}(W_k)$ is the trace of the within-cluster dispersion matrix,
- $\mathrm{tr}(B_k)$ is the trace of the between-cluster dispersion matrix.

A higher CH index indicates more well-defined and distinct clusters.

2.5 Engine Signals

This section introduces the primary measurements recorded from the engine and explains their physical meaning, units, and relevance for cycle analysis.

2.5.1 Key Parameters

Torque is the engine's rotational force, and engine speed indicates how fast the crankshaft rotates; their units are Nm and RPM, respectively. Together, they determine power output. In drive cycle analysis, torque and RPM are the primary signals to evaluate engine performance, fuel consumption, and emissions. For instance, the National Renewable Energy Laboratory emphasizes that monitoring these signals is essential for accurately assessing fuel economy and emission characteristics [26].

2.5.2 Power Calculation

Mechanical power represents the rate at which the engine's rotating components perform work [26]. In a rotational system, instantaneous power is the product of the torque applied and the angular velocity. When torque is measured in newton-metres (Nm) and speed is given in revolutions per second (RPS) or revolutions per minute (RPM), appropriate conversions yield power in kilowatts (kW). Below, the nominal friction percent torque is the signal quantifying the torque level lost due to friction when the engine is in operation.

Variables and Metrics

- T_{ref}: Reference torque (in Nm).
- T_{actual}: Actual engine percent torque.
- T_{nominal}: Nominal friction percent torque.
- n_{rps}: Engine speed in revolutions per second.
- n_{rpm}: Engine speed in revolutions per minute.
- P: Engine power in kilowatts (kW).

$$T_{\text{eff}} = T_{\text{ref}} \cdot \left(\frac{T_{\text{actual}}}{100} - \frac{T_{\text{nomial}}}{100} \right), \tag{2.8}$$

where

- T_{actual} is the actual engine percent torque,
- T_{nominal} is the nominal friction percent torque.

The engine power is derived as:

$$\text{Power [kW]} = \frac{\text{Torque [Nm]} \times 2\pi \times n \text{ [rps]}}{1000}, \tag{2.9}$$

This can be expressed in terms of revolutions per minute:

$$\text{Power [kW]} = \frac{\text{Torque [Nm]} \times \pi \times n \text{ [rpm]}}{30000}. \tag{2.10}$$

Thus, the engine power P is calculated as:

$$P = \frac{T_{\text{eff}} \cdot \pi \, n_{\text{rpm}}}{30000}. \tag{2.11}$$

2.6 Feature Selection

Feature extraction and feature selection aim to reduce dimensionality by removing redundant or noisy variables, improving model performance, and computational efficiency [27][28]. Extraction methods such as Principal Component Analysis (PCA) derive new features that are linear combinations of the originals that capture most of the data's variance in fewer dimensions [28]. Selection methods, instead, choose a subset of the original variables to enhance interpretability.

2.6.1 Principal Component Analysis

The problem with high-dimensional data is common in most areas of machine learning. Having too many features can, for example, lead to unnecessary computational power being required to solve the problem and might skew the results, as some features are better or worse at providing information for the issue at hand [28]. The general idea of dimension reduction is to retain relevant information and reduce dimensionality by removing unnecessary information [29].

One common dimension reduction approach is using PCA [28]. The primary goal is to transform a set of features into Principal Components (PCs), which are linear combinations of the original features. The PCs are ordered according to overall variance captured in the data, with the first PC capturing the most variance.

PCA involves decomposing the covariance matrix into eigenvalues and eigenvectors, where the eigenvalues correspond to the amount of variance explained by each PC and the eigenvector gives the components' direction [28]. When PCA has been performed, the dimensionality has been reduced, and the underlying structure and relationships between variables are revealed. However, one drawback to PCA is that, although these relationships are shown, it can be challenging to interpret their meaning. It may not be straightforward to understand what linear combinations of original values mean and to intuitively analyze the influence of individual features on each component. This lack of straightforward interpretability may impact the overall insights gained from the data. The detailed PCA procedure is outlined in Algorithm 5.

Algorithm 5 Principal Component a Analysis.

1: **procedure** PCA(X, d)
2: **Input:** $X \in \mathbb{R}^{D \times n}$ (centered), $d \leq \min(D, n)$
3: **Output:** $Y \in \mathbb{R}^{d \times n}$, data projected onto top d PCs
4: Form the Gram matrix:

$$S \leftarrow X^T X$$

5: Compute eigen-decomposition:

$$S \leftarrow V \Lambda V^T \quad (\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n))$$

6: Compute all principal axes:

$$U \leftarrow X V \Lambda^{-\frac{1}{2}}$$

7: Extract top d axes:

$$U_d \leftarrow [u_1, \ldots, u_d]$$

8: Project the data:

$$Y \leftarrow U_d^\top X$$

Note: $X \in \mathbb{R}^{D \times n}$ denotes the data matrix whose columns x_i have already been mean centered. V is the eigenvector matrix of S.

2.6.2 Principal Feature Analysis

An alternative method for dimension reduction and identifying features that retain most of the information from the given feature set is Principal Feature Analysis (PFA) [29]. This method utilizes PCA, but ultimately, a subset of the original features is selected, rather than PCs, thereby maintaining explainability with features that retain their physical meaning.

PFA works by leveraging the results produced from PCA to determine the structure of the features [29]. The next step is mapping the PCs to the features, selecting only the features that best describe each component. Feature selection is performed by choosing the feature with the most significant contribution to each PC. These principal features (PFs) share the strongest relationship with the respective PCs. This procedure reduces dimensionality while retaining most of the variance in the new reduced feature set.

Below in Algorithm 6, the PFA procedure, based on the original paper [29], can be found in its entirety:

Algorithm 6 Principal Feature Analysis.

Input: Data matrix $X \in \mathbb{R}^{m \times n}$ with m samples and n features; subspace dimension q; number of clusters/features p.
Output: Set S of selected principal feature indices.

Step 1: Compute Covariance Matrix
Center the data: $\tilde{X} \leftarrow X - \bar{X}$, where $\bar{X}$ is the mean of X.
Compute covariance matrix: $\Sigma \leftarrow \frac{1}{m-1}\tilde{X}^T\tilde{X}$.

Step 2: Perform PCA

Step 3: Represent Features in Lower Dimensional Space
Let A_q be the matrix produced by PCA
for $i = 1, \ldots, n$ **do**
    Let v_i be the i^{th} row of A_q, i.e., $v_i \in \mathbb{R}^q$. ▷ v_i is the representation of feature i in the q-dimensional subspace.
This is chosen based on the desired amount of variability to keep, computed as:

$$\text{Variability Retained} = \frac{\sum_{i=1}^{q} \lambda_i}{\sum_{i=1}^{n} \lambda_i} \times 100\%,$$

where λ_i are the eigenvalues.

Step 4: Cluster Feature Representations
Perform K-means clustering on $\{v_i\}_{i=1}^n$ using Euclidean distance to partition them into p clusters: $\{C_1, C_2, \ldots, C_p\}$.

Step 5: Select Principal Features from Clusters
Initialize the selected feature set: $S \leftarrow \emptyset$.
for $k = 1, \ldots, p$ **do**
    Compute the centroid: $c_k \leftarrow \frac{1}{|C_k|}\sum_{v \in C_k} v$.
    Select feature index: $i^* \leftarrow \arg\min_{i \in C_k} \|v_i - c_k\|_2$.
    Update the set: $S \leftarrow S \cup \{i^*\}$.

return S ▷ Return the set of principal features.

The advantages of using PFA lie in the fact that the dimensionality is reduced while remaining in the original feature space, thereby maintaining explainability. Additionally, it can be used as a means of determining which features in the dataset are most important in terms of best representing the variability [29].

3
Methodology

This section explains the methodology for defining, analyzing, and clustering drive cycles. The methodology is divided into stages, each focused on a specific aspect of the project. An overview of this can be seen in Figure 3.1. The sections covered in the methodology are: *Data Preparation*, *Selection of Clustering Algorithms*, *Clustering and Detection*, and *Evaluation* with corresponding subsections.

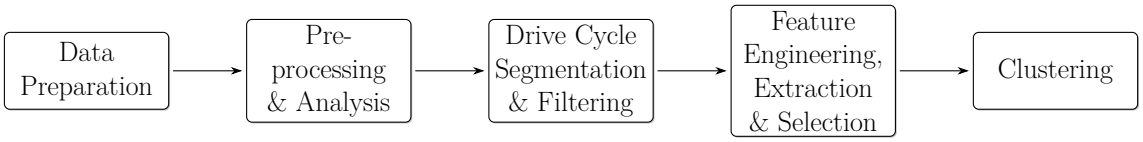


Figure 3.1: Overview of the methodology.

3.1 Data Preparation

The data was prepared to ensure it was in a suitable format for further analysis. The following sections will detail the data preparation, covering preprocessing and analysis, segmentation and filtering, feature engineering, and feature selection.

3.1.1 Preprocessing and Analysis

The initial state of the data was first evaluated. Given the vast amount of engine field test data available, across many different engines, the engine suitable for this thesis was chosen based on total time. The total number of timestamps available for each engine was counted and printed, leaving the engine with the most significant amount available to be chosen, given the signals needed: *Engine Speed*, *Engine Torque*, and *Nominal Friction Percent Torque* were present. Ultimately, the selected engine had 1382 logged hours at 1 Hz resolution and was placed in a log stacker vehicle. After the engine was chosen, the initial dataframe columns were *Engine Name*, *Timestamp*, *Signal Name*, and *Signal Value*.

The signals deemed relevant and chosen with input from Penta's domain experts were engine speed, torque, and nominal friction torque; the latter was only used to calculate engine power. Torque was given a percentage value of 0 to 100, where 100 is the maximum torque available. On the other hand, engine speed was given as the actual value in rpm. For the signals to have the same scale, the engine speed values were divided by the baseline value (more information regarding this in section 3.1.2)

3.1.2 Drive Cycle Segmentation and Filtering

Drive cycles were segmented based on the temporal gaps between consecutive signal readings. Histograms of these time gaps were analyzed to determine an optimal threshold that would allow us to extract as many meaningful cycles as possible while ensuring that the gap between cycles accurately represented known behaviors. Setting the threshold too low would result in overly fragmented cycles, potentially leading to misleading or unusable results. To make an informed decision, the distribution of time gaps (see Figure 4.1) was analyzed, and, in conjunction with domain experts at Penta, the maximal gap between stop and start was determined.

After deciding the time gap threshold, the drive cycle segmentation was done according to Algorithm 7 below:

Algorithm 7 Drive cycle segmentation.

Require: *df*: dataset with columns `Engine_Name`, `Signal_Name`, `timestamp`
Require: $THRESHOLD\_SECONDS$: time gap threshold (e.g., 10 minutes)
Ensure: *df_cycle*: dataset augmented with `cycle_id`

for all unique pair $(engine, signal)$ in *df* **do**
    Sort records of this group by ascending `timestamp`.
    $cycle\_counter \leftarrow 0$.
    $prev\_ts \leftarrow$ timestamp of first record in group.
    for each record r in this sorted group **do**
        $\Delta t \leftarrow r.\texttt{timestamp} - prev\_ts$.
        if $\Delta t > THRESHOLD\_SECONDS$ **then**
            $cycle\_counter \leftarrow cycle\_counter + 1$.
        $r.\texttt{cycle\_id} \leftarrow \text{concat}(r.\texttt{Engine\_Name}, \texttt{"\_"}, cycle\_counter)$.
        $prev\_ts \leftarrow r.\texttt{timestamp}$.
        Append r to output list.
Assemble output list into dataframe *df_cycle*. **return** *df_cycle*

When the cycles had been segmented, shorter cycles storing insufficient data were filtered out. To capture the most relevant cycles, the distribution of cycle lengths was analyzed as a histogram to decide which cycles to exclude.

After removing shorter cycles, a coverage filter was applied. Any cycle with an excessive proportion of missing or invalid readings was considered uninformative and was excluded from further analysis. In accordance with Penta domain experts, only cycles with at least 75% coverage were retained. This approach was implemented to ensure that the remaining cycles were both representative and of high enough quality for clustering analysis.

After cycles with insufficient coverage were removed, there still remained cycles where the engine was idling for long periods. When this is the case, no meaningful behavior can be analyzed, except for the engine running but not performing any active work. The baseline signal value had to be determined in order to obtain these idling periods, the value for when the engine is idling. This was achieved by computing the distribution of the engine signals and identifying the most frequent value.

Segmented cycles where the count of engine speed values and torque, with a deviation of 10% from baseline, were above 90% of the entire cycle were determined to be in the idling state and thus removed. The idling was constrained to be counted only if the signal remained within the baseline range for at least 10 seconds. This was done to add more realism, as realistically an engine is not idling when a driver lets the foot off the pedal for only a few seconds.

After the idling cycles were removed, a last visual inspection was made for the remaining cycles to eliminate any other noticeable outlier behavior that could have skewed the final clustering result.

3.1.3 Feature Engineering and Extraction

For all given signals, statistical features were extracted using built-in PySpark functions specific to each statistical case. Temporal features include the fraction of idling in a cycle and the number of starts and stops in a cycle. The counts of idling and average duration in idling were manually engineered and used for the initial feature set for each cycle before dimension reduction. The steps taken for these calculations are outlined in the following subsections; the complete list of features used can be found in 2.1.

3.1.3.1 Idle Periods and Idle Period Lengths

To calculate the idle periods and idle period lengths, the first step was to isolate the data points where idling occurred for each drive cycle. By measuring the gaps between successive idle points, using a gap threshold of 10 seconds, it was determined when one idle event ended and another began. When the periods were tagged as either a continuation of a current idling period or the beginning of a new one, a unique ID was assigned to each idle period. From this, the start time, end time, and duration were gathered to compute the total number of idling periods and the average length of these periods per cycle. Algorithm 8 below covers the steps taken.

Algorithm 8 Compute idle period count and D duration per cycle.

Require: data_rows with fields `engine`, `cycle`, `timestamp`, `idle_flag`

▷ 1. Filter idle records
idle_rows ← $\{r \in \text{data\_rows} \mid r.idle\_flag = 1\}$
▷ 2. Partition by engine and cycle, then sort
for all groups G in idle_rows partitioned by (engine, cycle) **do**
    sort G by $r.timestamp$ ascending
    prev_ts ← null
    new_idle_id ← 0
    ▷ 3–5. Detect new idle events and assign IDs
    for all row r in G **do**
        if prev_ts is null **then**
            gap ← 0
        else
            gap ← $r.timestamp$ − prev_ts
        if gap > 10 s **then**
            new_idle_id ← new_idle_id + 1
        r.idle_id ← new_idle_id
        prev_ts ← r.timestamp
    ▷ 6. Aggregate each idle period
idle_periods ← ∅
for all groups P in idle_rows partitioned by (engine, cycle, idle_id) **do**
    start_t ← $\min_{r \in P} r.timestamp$
    end_t ← $\max_{r \in P} r.timestamp$
    duration ← end_t − start_t
    add (engine, cycle, idle_id, start_t, end_t, duration) to idle_periods
    ▷ 7. Summarize per cycle
summaries ← ∅
for all groups C in idle_periods partitioned by (engine, cycle) **do**
    count_idle ← $|C|$
    avg_dur ← $\frac{1}{|C|} \sum_{p \in C} p.duration$
    add (engine, cycle, count_idle, avg_dur) to summaries
    ▷ 8. Join back to the main dataset
result ← main_cycles ⋈ summaries on (engine, cycle)

Idle Fraction was calculated by taking the Idle Duration per Cycle and dividing it by the total length of the cycle.

3.1.3.2 Power Calculation

Engine power was a signal that was not present in the dataset. However, it was a feature deemed necessary by domain experts at Penta and therefore needed to be added to the dataset. Using the present signals, engine speed, engine torque (in % increase from baseline value, when in idle), NominalFrictionPercentTorque, and a reference value for torque denoted by T_{ref}, taken from a Penta database, the engine power, P could be calculated using the equations presented in section 2.6.2.

3.1.4 Feature Selection

When all relevant features had been extracted, the most important were selected using PFA. The first step in the process was to utilize the results obtained from PCA. The cumulative explained variance that had to be retained in the process of choosing PCs was determined to be 95%, as is standard for this procedure. How the overall number of PCs impacted explained variance can be seen in Figure 3.2

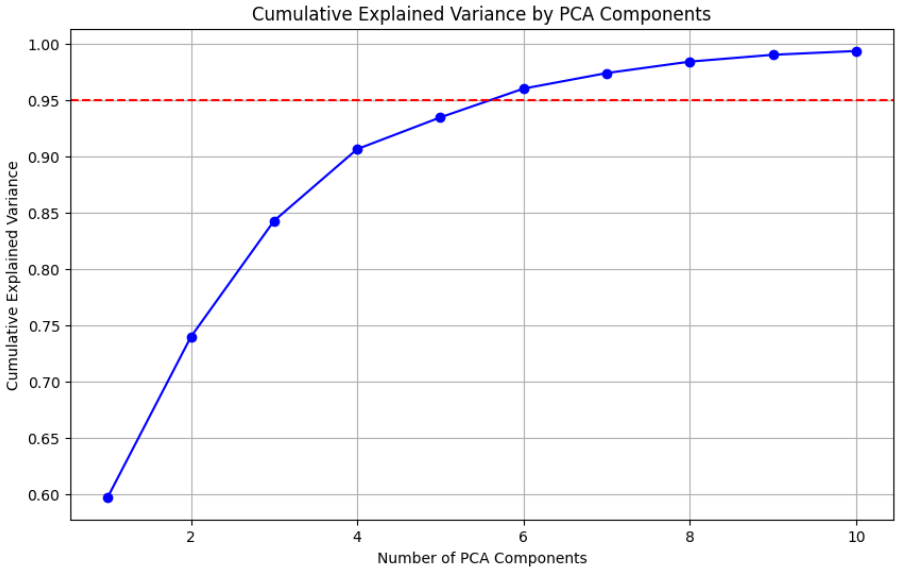


Figure 3.2: Explained variance over number of PCs.

The features most important to each PC were removed, continuing the PFA procedure and leaving a reduced feature set of the most information-retaining features.

3.2 Selection of Clustering Algorithms

There are several aspects to consider when selecting the clustering algorithms to use. For instance, the size of the data, the number of dimensions, a known or unknown number of clusters, the convex or arbitrary shape of clusters, etc. For this thesis, three clustering algorithms were considered. K-means and HAC were chosen due to their widespread use, ease of implementation, and because they represent two fundamentally different approaches to clustering. A more systematic approach was employed when selecting the third algorithm, following a previous methodology developed by Wegmann et al. [30].

Firstly, the dimensionality of the dataset was determined using Algorithm 9 [30].

Algorithm 9 Get dimensions of dataset.

```
1: function DIMENSION(Dataset)
2:     (n, m) ← Size(Dataset)
3:     if m ≤ 10 then
4:         Category of Dimension ← LOW
5:     else
6:         Category of Dimension ← HIGH
7:     return Category of Dimension
```

Secondly, the dataset needed to be placed in a size category, which follows Algorithm 10 [30].

Algorithm 10 Get size of dataset.

```
1: function GET SIZE(Dataset)
2:     (n, m) ← Size(Dataset)
3:     if n ≤ 50 then
4:         Category of Size ← SMALL
5:     else if n ≤ 10000 then
6:         Category of Size ← MEDIUM
7:     else
8:         Category of Size ← LARGE
9:     return Category of Size
```

Lastly, the shape of the clusters was estimated by observing the top 3 most important PCs in 3D space, which can be seen in Figure 3.3.

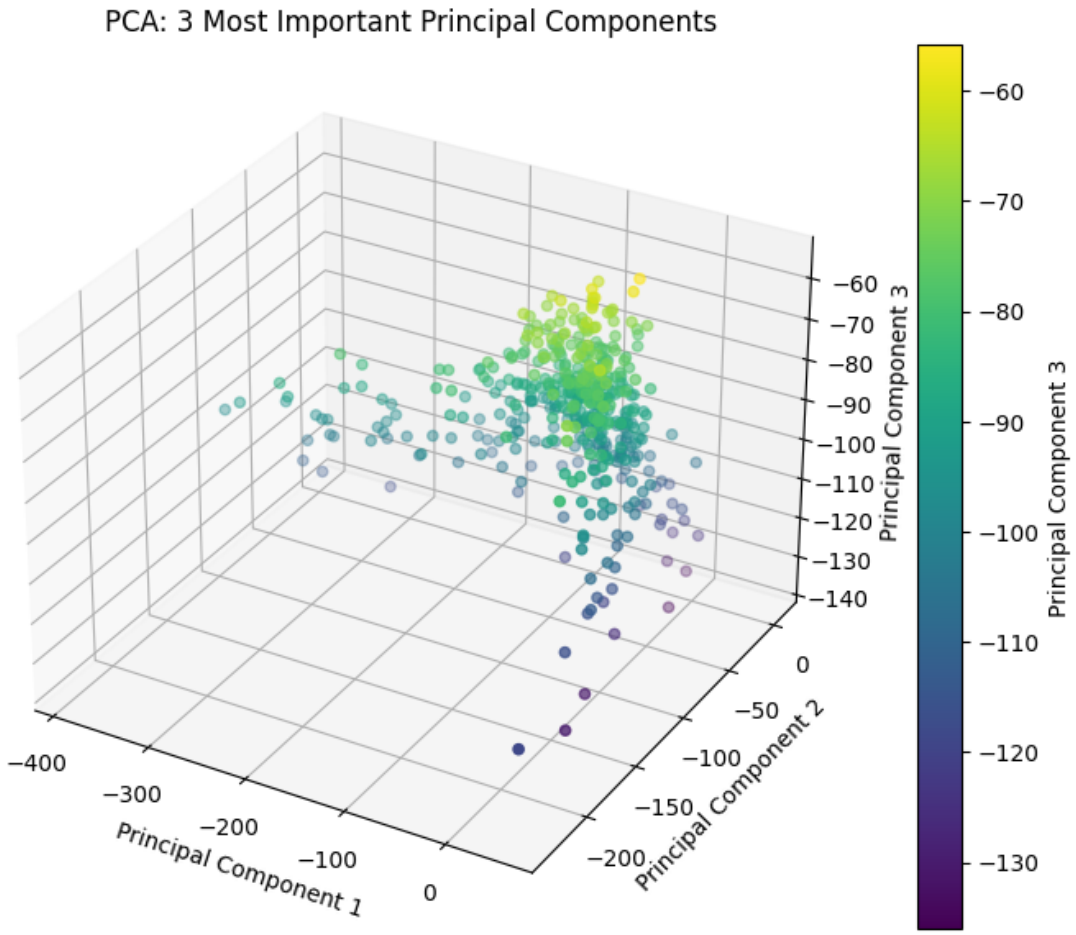


Figure 3.3: Top 3 most important PCs in 3D space. Here, the color gradient depicts the depth of PC 3.

Determining the dimensionality and size of the dataset and the shape of the clusters according to Algorithms 9 and 10 and Figure 3.3 ultimately led to mean-shift being chosen.

3.3 Clustering and Detection

Once the features were chosen, the next step was to cluster the cycles. Unsupervised clustering was employed to partition drive cycles into groups, each representing a particular engine usage or behavior type. The clustering techniques utilized were k-means, HAC, and mean-shift. Consensus clustering was then used to combine the individual partitions into a unified set of clusters. The full clustering pipeline can be seen in Figure 3.4

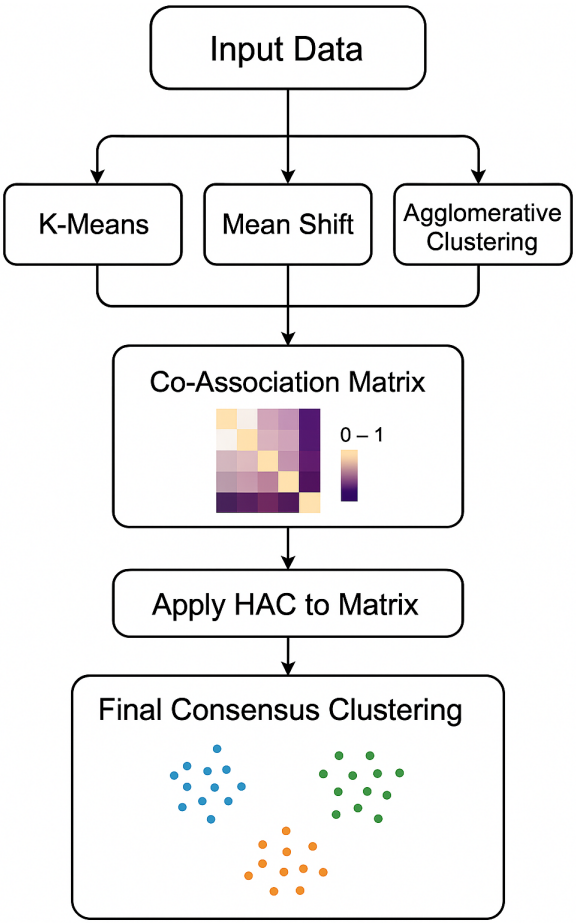


Figure 3.4: Schematic overview of the clustering flowchart.

3.3.1 K-Means

The number of clusters, k, was specified by inspecting the results for Sil score, CH index, and DB index for different values of $k$in the range of 2-10. The $k$that maximizes the Sil score and CH index, while minimizing the DB index, was chosen for the final clustering. K-means++ initialization was used to ensure more reliable seeding and faster convergence.

3.3.2 HAC

Ward's linkage was chosen to minimize the increase in total within-cluster variance at each merge in the HAC. As mentioned in Section 2.3.3.1, this enhanced the model's robustness to noise and outliers in the drive cycle feature space. A dendrogram was then constructed for all drive cycles. A distance threshold for cutting the dendrogram was selected by evaluating the number of clusters present depicted by the dendrogram in combination with observing the results for Sil score, CH index, and DB index for different values of k in the range of 2-10.

3.3.3 Mean-Shift

Mean-shift clustering does not require a predefined k. However, another parameter, the kernel radius, had to be chosen, which was determined by the *estimate_bandwidth* utility from scikit-learn, which approximates an optimal kernel radius based on the data distribution. Scikit-learn's implementation of mean-shift defaults to a flat (uniform) kernel, which implements a windowed mean, and that kernel was used for the clustering.

3.3.4 Consensus Clustering

To enhance the robustness and stability of the clustering results, consensus clustering was performed using a co-association matrix. The objective was to integrate multiple clustering solutions into a single, representative consensus partition. Using the three distinct partitions generated by k-means, mean-shift, and HAC, a co-association matrix was constructed where each entry $(i, j)(i, j)$ represents the frequency with which data points ii and jj were assigned to the same cluster across the different algorithms. The values in this matrix range from 0 to 1, with 1 indicating that two points were always clustered together, and 0 indicating that they were never clustered together. HAC was then applied to the co-association matrix to derive a consensus clustering. This final clustering aimed to capture the structure consistently supported across the different individual clustering algorithms, thereby improving reliability and interpretability.

3.3.5 Representative Cycle Selection and Detection

The centroid in feature space was computed for each resulting cluster. The cycle whose feature vector was the closest to the centroid by Euclidean distance was designated as the representative cycle for the cluster.

3.4 Evaluation

Ultimately, cluster quality was evaluated by computing Sil, CH index, and DB index for each clustering solution. For visual inspection, pair plots were generated for each respective clustering result and the time series for the representative cycles, as well as box plots for all clusters across all clustering methods. Feature importance was reviewed by analyzing the remaining features after completing feature selection using PFA. From the original feature set, they were deemed the most important according to PFA, and their importance was ranked in order of decreasing importance from top to bottom, as they stemmed from the first PC to the last. Finally, to validate the validity of the clusters and whether they represented meaningful engine behavior, Penta domain experts reviewed both the formal cycle specifications and the representative cycles.

4
Results and Discussion

This chapter presents the results and discussion, divided into the following sections: *Cycle Segmentation and Idle Removal*, *Feature Selection*, and *Clustering*, each with its respective subsections. Following each result section is a discussion of the results shown.

4.1 Segmentation and Idle Removal

Analyzing Figure 4.1 according to Section 3.1.2, the time between the stop and start of a new cycle was set to 10 minutes.

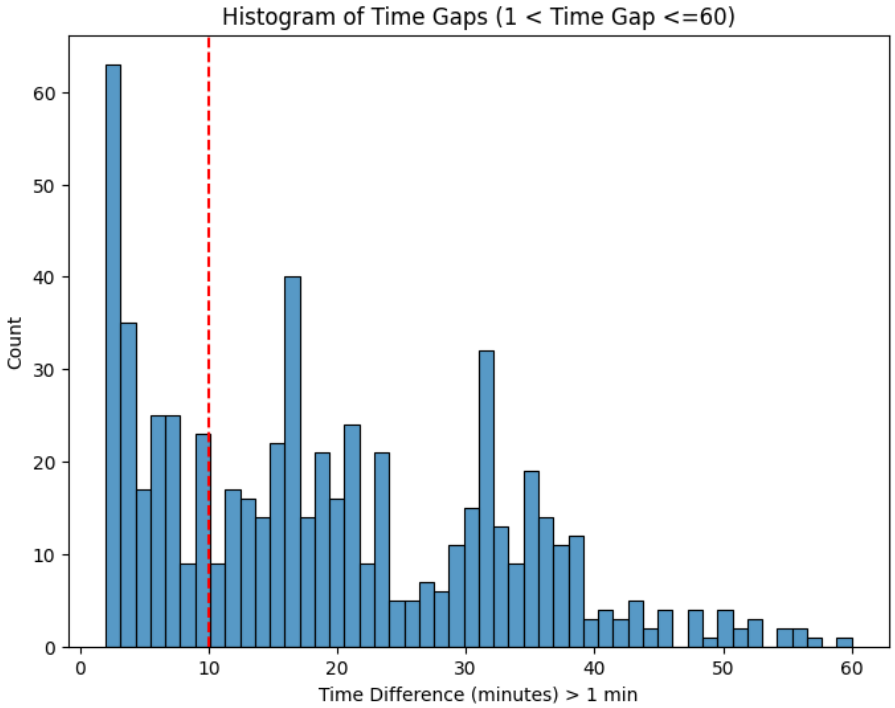


Figure 4.1: Histogram of time gaps between cycles. Gaps of 10 minutes or more determine the stop and start of a new cycle.

The segmentation yielded many cycles with varying lengths. To find cycles that describe meaningful behavior, shorter cycles were removed, as they were deemed insufficient to provide a useful time frame. Eventually, all cycles less than 60 minutes in length were removed as this captured most of the distribution of cycles

and, according to Penta domain experts, removed irrelevant engine behaviors. The distribution can be seen in Figure 4.2.

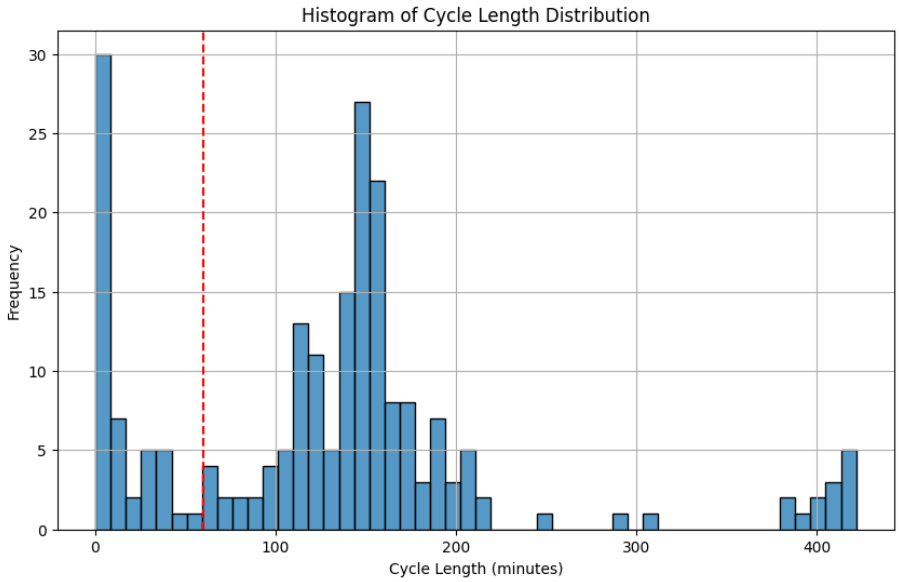


Figure 4.2: Histogram of cycle length distribution with cycle length below 60 minutes deemed irrelevant.

The baseline value used for idling was determined by evaluating the distributions for each signal and identifying the most frequent value. The resulting baseline values can be seen in Figures 4.3, 4.4, and 4.5, and Table 4.1 depicts the actual values.

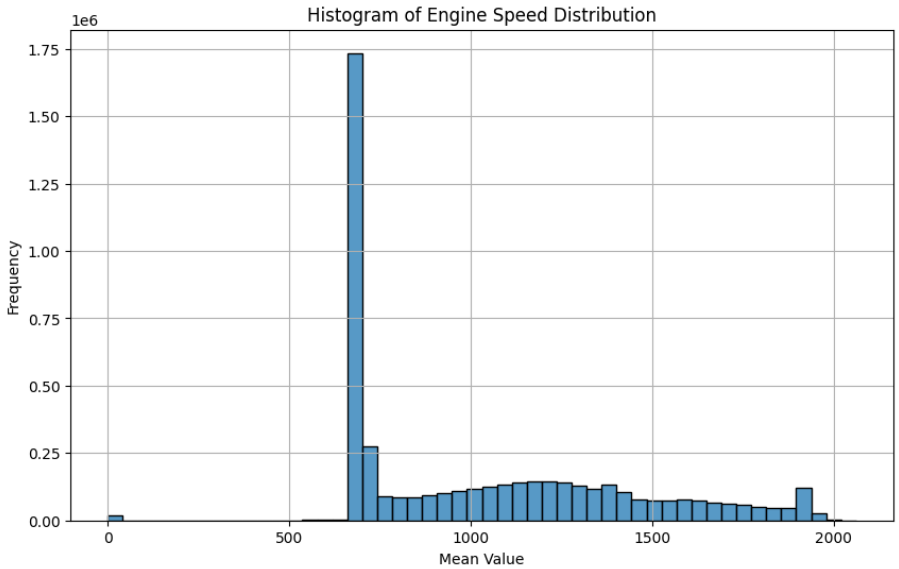


Figure 4.3: Distribution of signal values for Engine Speed.

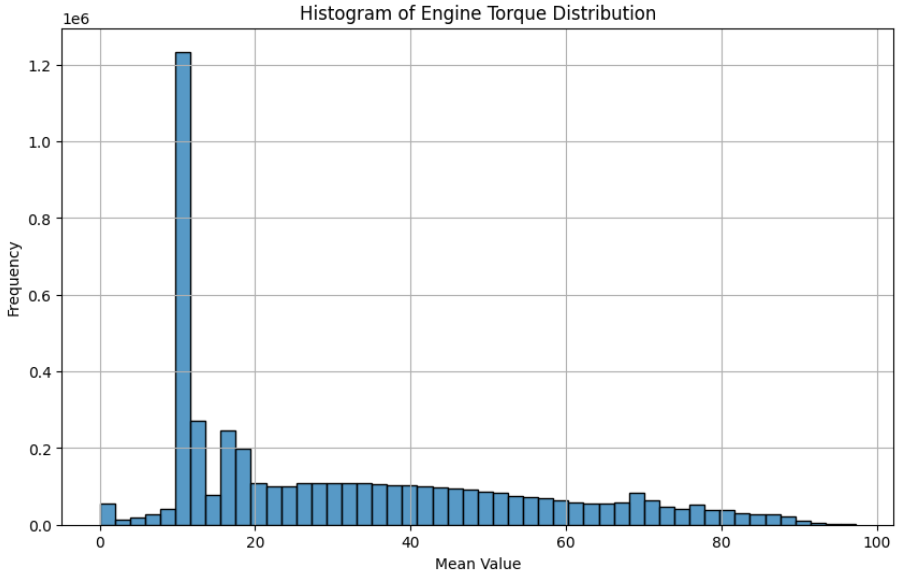


Figure 4.4: Distribution of signal values for Engine Torque.

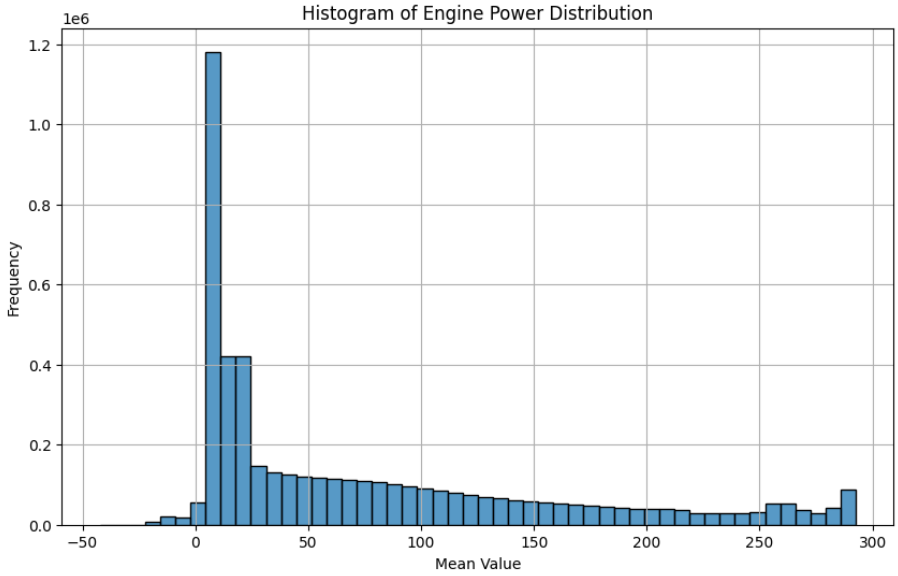


Figure 4.5: Distribution of signal values for Engine Power.

Table 4.1: Baseline Signal Values.

Signal Name	**Baseline Value**
Engine Torque	11
Engine Speed	700
Engine Power	11

In total, segmentation yielded 574 cycles. Out of these, the cycles shorter than 60 minutes were removed, leaving 463 cycles. After idle removal, three additional

cycles were removed, revealing 460 relevant cycles that were to be clustered. The cycles removed are listed in Table 4.2. The basis of removal was due to extreme values found in the cycles. For cycle 453, the duration idle is 423.8 minutes, around 200 minutes longer than the second-to-largest value in this feature category. Cycle 227 also showed vastly larger kurtosis than any other cycle. Finally, Cycle 535 was removed because engine signal values were below the baseline, which deviated drastically from expectations and was thus removed.

Table 4.2: Outlier cycles feature values. *Note:* Values in **bold** indicate those discussed in the text above.

Feature	**Cycle 453**	**Cycle 227**	**Cycle 535**
Torque +Δ	1.793	0.412	1.315
Torque Kurt	–0.569	**15.459**	1.895
Torque	28.143	16.754	16.325
Torque Stddev	20.282	11.073	21.006
Torque Skew	0.801	3.889	1.561
Power +Δ	5.829	1.429	4.171
Power Kurt	–0.068	**19.173**	5.774
Power	65.910	25.160	34.826
Power Stddev	68.426	39.073	63.585
Power Skew	1.010	4.321	2.523
Speed +Δ	15.397	5.098	14.937
Speed Kurt	–1.030	**12.766**	–0.392
Speed	1025.422	758.136	**511.239**
Speed Stddev	354.712	201.445	524.554
Speed Skew	0.572	3.616	0.642
Fraction Idle	0.471	0.252	0.479
Count Idle	5.000	18.000	48.000
Dur Idle	**423.800**	109.944	120.917

4.1.1 Discussion

The gap length distribution determined the cycle boundaries, which were defined at gaps of at least ten minutes. Yet slight variations could merge or split meaningful information. Removing too short cycles filtered out irrelevant data but may also have excluded brief yet normally significant behaviors. Domain experts agreed that cycles shorter than 60 minutes should be removed since they would not be helpful in further research.

Furthermore, idle baselines were set at each signal's modal value to ensure consistency, but with the risk of low-load misclassification. This could be avoided by incorporating other indicators, such as transmission status.

Removing outlier cycles before clustering is an essential step to ensure that the clustering algorithms more effectively capture the dominant patterns in the data.

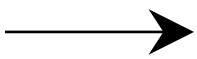
This is especially important when using methods sensitive to outliers, such as k-means and HAC, where extreme values can disproportionally affect cluster centers or skew the dendrogram. However, the removed outlier values in this case could represent occasional but valid drive cycle behavior. The removal of outlier values must align with the established goals. Since general behaviors, rather than behaviors for extreme operational conditions, were to be evaluated, the cycles detected in Table 4.2 had to be removed.

4.2 Results Feature Selection

After feature selection using PFA, the list of original features decreased from 18 to 8. The original and selected features are found below in Table 4.3. Based on the PFA procedure, the features are listed in decreasing order of importance, noting that Avg Torque $+\Delta$ is deemed most important. This means that this feature is the most significant in producing PC1, the PC explaining the most variance of the dataset.

Table 4.3: Feature list before PFA (left) and selected features (right).

Feature
Torque $+\Delta$
Torque Kurt
Torque
Torque Stddev
Torque Skew
Power $+\Delta$
Power Kurt
Power
Power Stddev
Power Skew
Speed $+\Delta$
Speed Kurt
Speed
Speed Stddev
Speed Skew
Fraction Idle
Count Idle
Dur Idle



Feature
Torque $+\Delta$
Torque Skew
Power
Speed Skew
Speed $+\Delta$
Fraction Idle
Count Idle
Dur Idle

4.2.1 Discussion

Examining Table 4.3, it is possible to observe that after PFA, the most crucial feature present is the average torque positive derivative. The average speed positive derivative is also noted as being of considerable importance, denoting that the magnitude of positive change within a cycle drives a large portion of the variance.

Another essential feature denoted by PFA is the torque and speed skewness. These values were all positive for the three most representative cycles of the clusters for the respective model, which means that the distribution is right-skewed. This indicates that most measurements lie below the mean, while a longer tail extends to the right of the distribution, reflecting occasional values of higher magnitude. In practice, this skewness indicates that the engine typically operates at moderate torques and speeds but is punctuated by infrequent bursts, whether accelerating under load or responding to sudden demands that push the signals into higher magnitude.

While not at the top of the list in the reduced feature set, all features that capture idling behavior in the first feature set are also present. This underscores idling as a core behavioral dimension for grouping cycle behavior.

PFA is a powerful tool for finding important features and decreasing dimensionality in the dataset. However, since PFA relies on PCA, a linear method, essential features according to non-linear metrics may be discarded. Using a non-linear method might have produced an entirely different reduced feature set while remaining valid. However, the most critical aspect of PFA lies in its interpretability, which is why it remains the most optimal method, staying within the original feature domain.

4.3 Clustering

This section presents the results obtained from all the clustering algorithms employed. Metric scores for each algorithm are provided, along with a scatter plot of the most essential features in the three most representative cycles, corresponding feature values, and additional cluster metrics. Appendix 1 provides the full pair plots for each clustering algorithm using all features clustered across all cycles. To view the complete raw signal time series for the top three representative cycles per cluster and clustering method, please refer to Appendix 2.

4.3.1 K-Means

For the used dataset, the optimal k for the final clustering was decided by examining the graphs in Figure 4.6. Since all metrics indicated the best score for 2 clusters, this number was used for the k parameter. Table 4.4 shows the final metric scores after clustering.

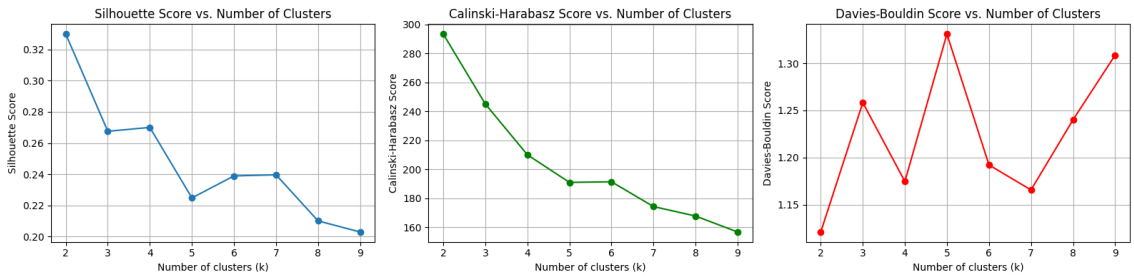


Figure 4.6: Metric scores for a range of numbers of clusters for k-means.

Table 4.4: Clustering evaluation metric scores for k-means (2 clusters).

Metric	Score
Sil	0.330
CH	293.314
DB	1.121

Figure 4.7 illustrates the scatter plot of the top three representative cycles for each cluster described by the two topmost important features. The figure highlights the presence of two distinct cycle behaviors.

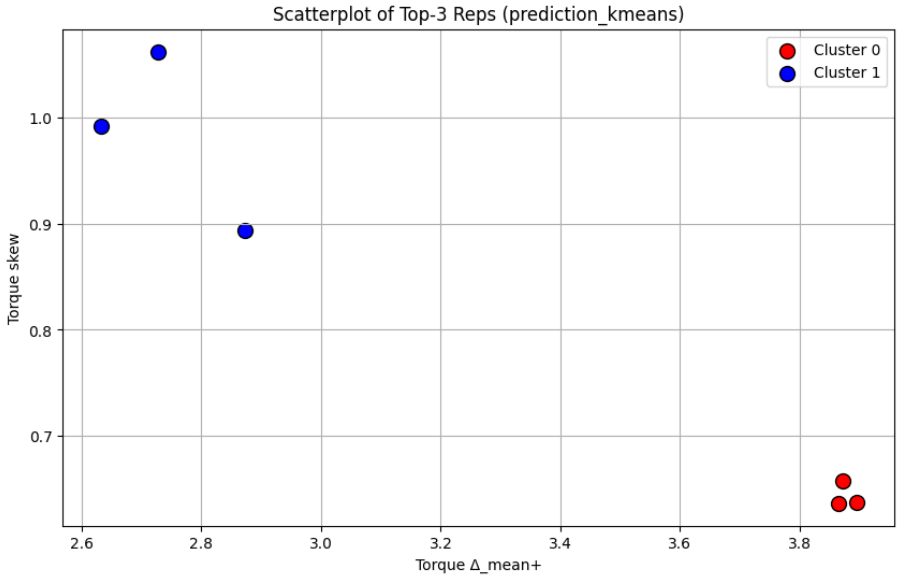


Figure 4.7: K-means pair plots of the three most representative cycles per cluster using the two most important principal features.

The overall distribution of cycles per cluster, the average length of each cycle, and the most representative cycle for the cluster are presented in Table 4.5. The results show cluster 0 as the larger cluster with a similar average cycle length, indicating that this has not been a significant factor for the clustering result.

Table 4.5: Summary of k-means clustering with representative cycle ID.

Cluster Prediction	No. Cycles	% of Cycles	Avg. Cycle Length (s)	Rep. Cycle ID
0	266	58%	10835	572
1	194	42%	10420	539

Table 4.6 depicts the top three representative cycles for each cluster, and Tables 4.7 and 4.8 contain their respective feature values. Nearly half of the feature values

from cluster 1 appear higher. The values that are higher for cluster 1 are: *Torque Skewness*, *Speed Skewness*, *Fraction Idle*, and *Duration Idle*. What stands out is that cluster 0 generally has shorter periods of idleness, whilst cluster 1 has longer stops but not as frequently. The power output is also larger for cluster 0, and the positive derivative for speed and torque. Comparing the feature values for the cycles within the same cluster reveals that they are similar for both clusters.

Table 4.6: Representative cycles per cluster.

Cluster	Rep Cycle ID
0	572
0	45
0	429
1	539
1	309
1	534

Table 4.7: Feature values for representative cycles from k-means cluster 0.

Feature	Cycle 572	Cycle 45	Cycle 429
Torque +Δ	3.873	3.866	3.900
Torque Skew	0.658	0.637	0.638
Power	84.816	84.971	90.791
Speed Skew	0.511	0.617	0.608
Speed +Δ	45.584	51.160	46.839
Fraction Idle	0.214	0.203	0.163
Count Idle	124	127	102
Duration Idle	18.024	15.520	15.111

Table 4.8: Feature values for representative cycles from k-means cluster 1.

Feature	Cycle 539	Cycle 309	Cycle 534
Torque +Δ	2.632	2.727	2.873
Torque Skew	0.991	1.062	0.893
Power	61.580	65.832	66.116
Speed Skew	1.140	0.990	1.023
Speed +Δ	30.249	31.004	29.804
Fraction Idle	0.435	0.472	0.421
Count Idle	73	96	80
Duration Idle	54.123	69.458	55.238

4.3.2 HAC

The dendrogram, in combination with the displayed metric values for a range of k values, was inspected to determine the number of clusters to use for the HAC algorithm. From Figure 4.8, it can be seen that there generally seems to be two significant clusters. Therefore, the cutoff was made at a distance of 35. This was further backed up by the findings from Figure 4.9. After clustering for this cutoff, the final metric scores can be seen in Table 4.9.

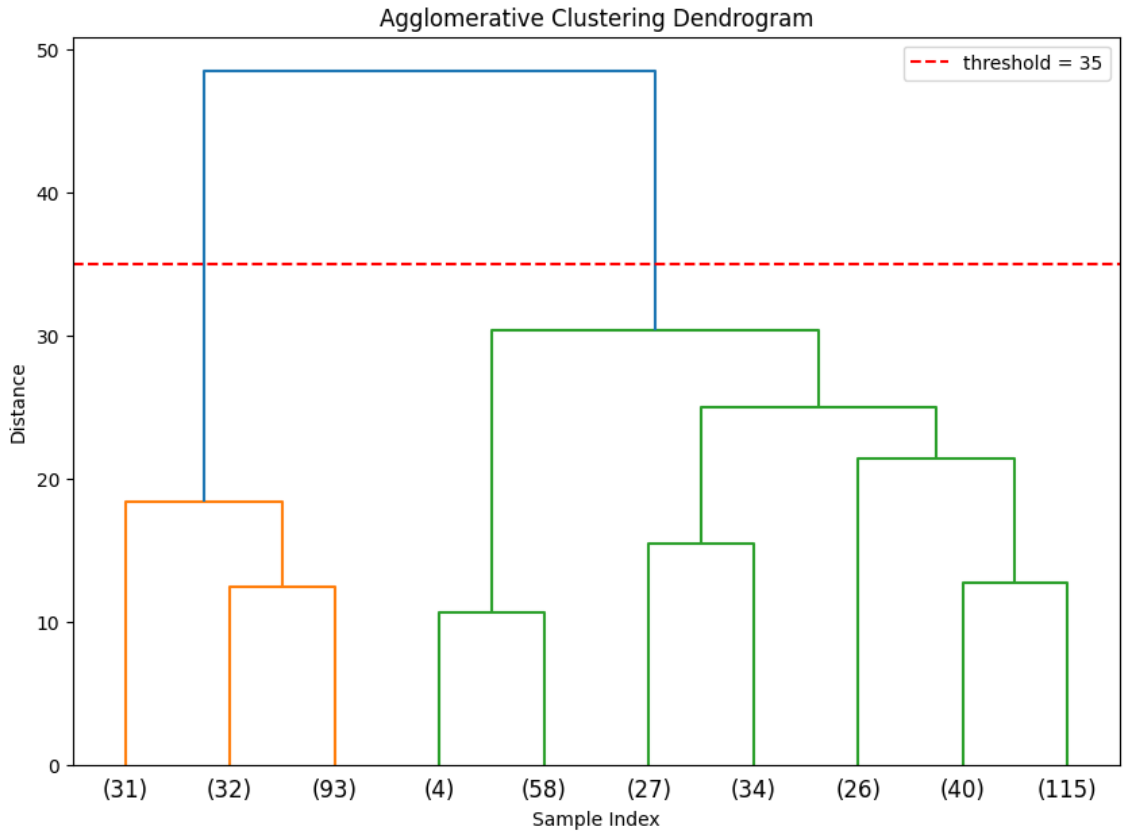


Figure 4.8: Dendrogram showing hierarchy of datapoints for HAC.

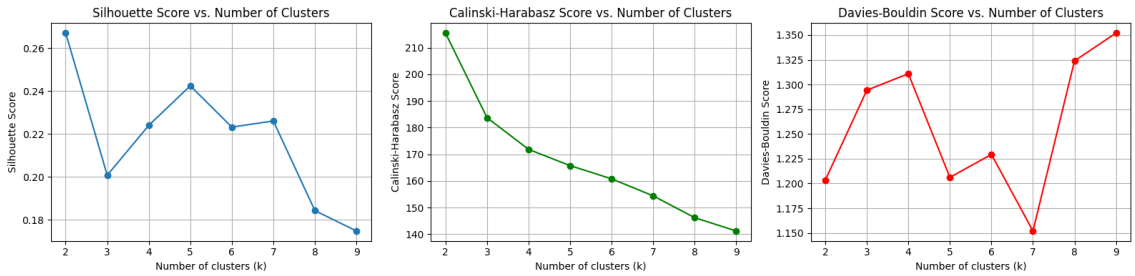


Figure 4.9: Metric scores for a range of numbers of clusters for HAC.

Table 4.9: Clustering evaluation metric scores for HAC.

Metric	Score
Sil	0.267
CH	215.486
DB	1.203

Figure 4.10 is the scatter plot of the top three representative cycles for each cluster described by the two topmost important features. Like k-means, the figure highlights the presence of two distinct cycle behaviors.

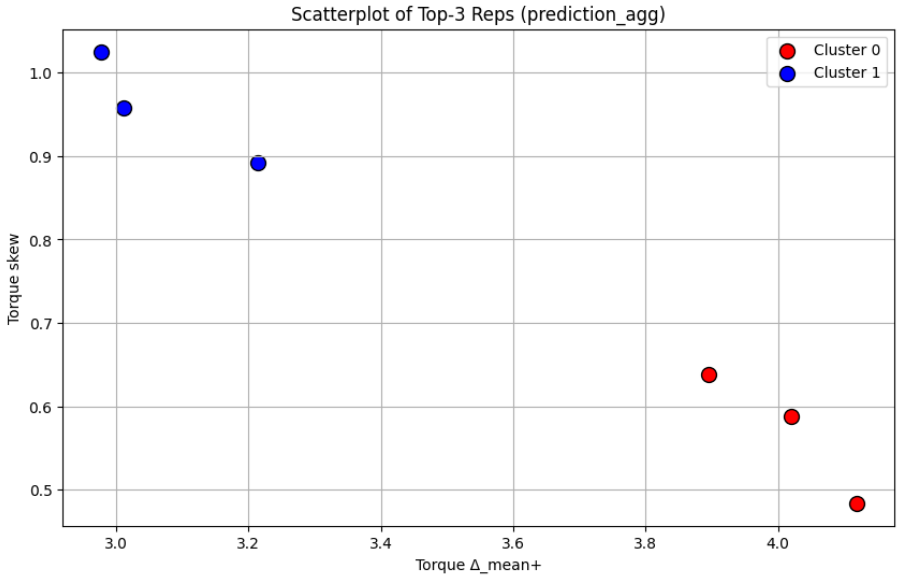


Figure 4.10: HAC pair plots of the three most representative cycles per cluster using the two most important principal features.

The overall distribution of cycles per cluster, the average length of each cycle, and the most representative cycle for the cluster are presented in Table 4.10. The results show that cluster 1 includes nearly $\frac{2}{3}$ of all cycles, with a higher average cycle length.

Table 4.10: Summary of HAC clusters with representative cycle ID.

Cluster Prediction	No. Cycles	% of Cycles	Avg. Cycle Length (s)	Rep. Cycle ID
0	156	34%	9750	430
1	304	66%	11127	80

Table 4.11 depicts the top three representative cycles for each cluster, and Tables 4.12 and 4.13 contain their respective feature values. Nearly half of the feature values from cluster 1 show up higher. The values that are higher for cluster 1 are: *Torque Skewness*, *Speed Skewness*, *Fraction Idle*, and *Duration Idle*. What stands out is that the Count of Idle periods is approximately the same for both clusters, while the $+\Delta$ for torque and speed is generally higher for the cycles in cluster 0. Like the clusters from k-means, one cluster spends more time in idling, typically with a lower power output and lower $+\Delta$ values. Comparing the feature values for the cycles within the same cluster reveals that they are similar for both clusters.

Table 4.11: Representative cycles per cluster.

Cluster	Rep Cycle ID
0	430
0	272
0	429
1	80
1	82
1	361

Table 4.12: Feature values for representative cycles from HAC cluster 0.

Feature	Cycle 430	Cycle 272	Cycle 429
Torque +Δ	4.119	4.020	3.900
Torque Skew	0.484	0.587	0.638
Power	94.925	90.917	90.791
Speed Skew	0.0501	0.591	0.608
Speed +Δ	45.532	48.358	46.839
Fraction Idle	0.179	0.157	0.163
Count Idle	88	72	102
Duration Idle	21.511	21.056	15.108

Table 4.13: Feature values for representative cycles from HAC cluster 1.

Feature	Cycle 80	Cycle 82	Cycle 361
Torque +Δ	2.977	3.011	3.214
Torque Skew	1.025	0.957	0.892
Power	68.433	71.520	74.250
Speed Skew	0.960	0.877	0.923
Speed +Δ	34.465	35.951	31.847
Fraction Idle	0.383	0.324	0.374
Count Idle	85	89	103
Duration Idle	45.906	33.067	43.573

4.3.3 Mean-Shift

As mean-Shift clustering is non-parametric, there was no need to pre-specify a number of k clusters or make a distance-based decision using a dendrogram. Table 4.14 shows the final metric scores after mean-Shift clustering.

Table 4.14: Clustering evaluation metric scores for mean-shift.

Metric	Score
Sil	0.373
CH	45.113
DB	1.064

Figure 4.11 presents a two-dimensional scatter plot of each cluster's three most representative cycle profiles, projected onto the two features with the highest importance scores. Consistent with both k-means and HAC, this projection reveals two primary behaviors. However, this reduced-dimension view does not fully capture the underlying structure: as seen in Table 4.15 and illustrated in more detail in Appendix Figure A.3, the vast majority of cycles coalesce into one dominant cluster, whereas a smaller secondary cluster seemingly depicts an outlier behavior pattern.

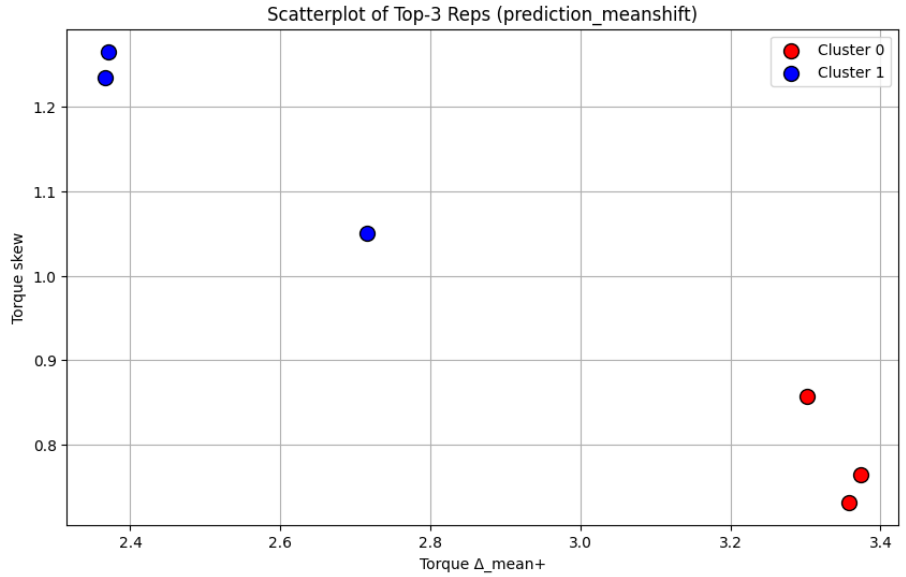


Figure 4.11: Mean-shift pair plots of the three most representative cycles per cluster using the two most important principal features.

Table 4.15: Summary of mean-shift clustering with representative cycle ID

Cluster Prediction	No. Cycles	% of Cycles	Avg. Cycle Length (s)	Rep. Cycle ID
0	445	97%	10710	240
1	15	3%	9192	420

Table 4.16 depicts the top three representative cycles for each cluster, and Tables 4.17 and 4.18 contain their respective feature values. Overall, nearly half of the feature values from cluster 1 are higher. The values that are higher for cluster 1 are: *Torque Skewness*, *Speed Skewness*, *Fraction Idle*, and *Duration Idle*. What stands out is that the count of idle periods is much larger for cluster 0 compared to cluster 1, but the duration of idle periods is much shorter. Compared to cluster 0 for k-means and HAC, the power output and $+\Delta$ for speed and torque in cluster 0 for mean-shift is generally slightly lower. Comparing the feature values for the cycles within the same cluster reveals that they are similar for both clusters.

Table 4.16: Representative cycles per cluster.

Cluster	Rep Cycle ID
0	240
0	189
0	286
1	420
1	161
1	308

Table 4.17: Feature values for representative cycles from mean-shift cluster 0.

Feature	Cycle 240	Cycle 189	Cycle 286
Torque +Δ	3.374	3.302	3.358
Torque Skew	0.764	0.858	0.732
Avg Power	76.644	75.816	77.876
Speed Skew	0.771	0.905	0.703
Speed +Δ	39.970	40.779	42.078
Fraction Idle	0.214	0.309	0.344
Count Idle	100	109	91
Duration Idle	41.590	36.569	39.484

Table 4.18: Feature values for representative cycles from mean-shift cluster 1.

Feature	Cycle 420	Cycle 161	Cycle 308
Torque +Δ	2.370	2.366	2.715
Torque Skew	1.266	1.235	1.051
Avg Power	65.991	64.380	71.773
Speed Skew	1.200	1.097	1.016
Speed +Δ	28.808	26.493	30.734
Fraction Idle	0.500	0.526	0.495
Count Idle	30	40	37
Duration Idle	142.467	126.150	130.892

4.3.4 Consensus

Given the clustering solutions produced previously, the consensus clusters could be created. Since HAC was utilized on the co-association matrix, a dendrogram was produced to visually make an informed decision on the number of consensus clusters to be produced. Based on the dendrogram, shown in Figure 4.12, in combination with previous knowledge of the number of clusters produced for the other clustering algorithms, the distance threshold was set at 0.4, giving two clusters. The final metric scores after Consensus clustering are presented in Table 4.19. Noticeably, the scores are the same as for k-means.

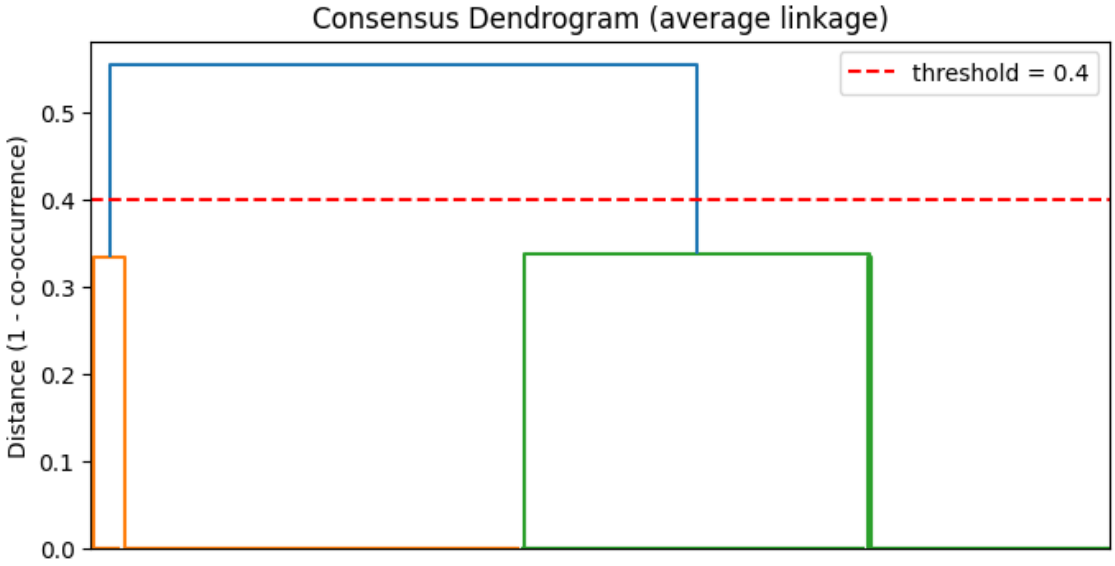


Figure 4.12: Dendrogram showing hierarchy of clusters for consensus clustering.

Table 4.19: Clustering evaluation metric scores for consensus.

Metric	Score
Sil	0.330
CH	293.314
DB	1.121

Figure 4.13 illustrates the scatter plot of the top three representative cycles for each cluster described by the two most important features. The Figure highlights the presence of two distinct cycle behaviors, and the clusters are exactly the same as for k-means.

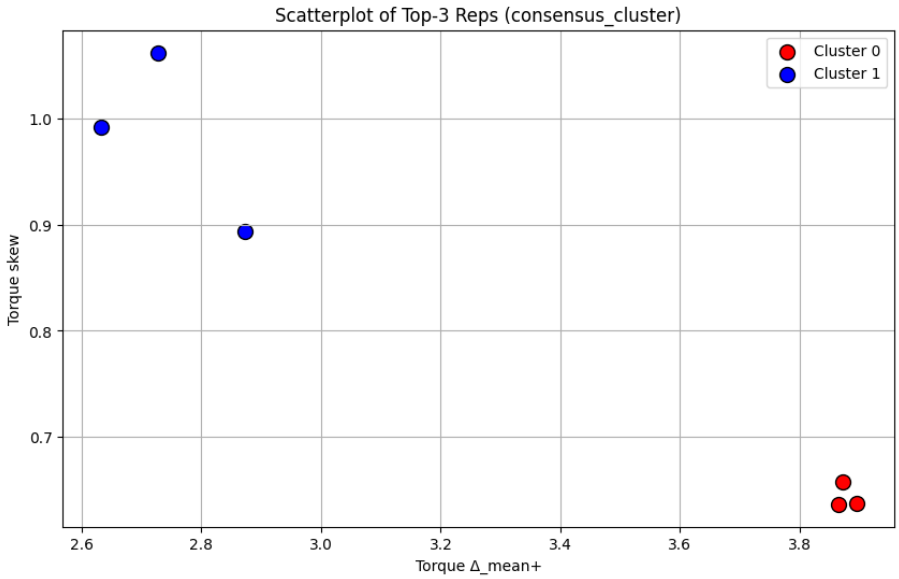


Figure 4.13: Consensus pair plots of the three most representative cycles per cluster using the two most important principal features.

The overall distribution of cycles per cluster, the average length of each cycle, and the most representative cycle for the cluster are presented in Table 4.20. The results show cluster 0 as the larger cluster and cluster 1 as the smaller cluster, consistent with the k-means distribution.

Table 4.20: Summary of consensus clustering with representative cycle ID.

Cluster Prediction	No. Cycles	% of Cycles	Avg. Cycle Length (s)	Rep. Cycle ID
0	266	58%	10835	572
1	194	42%	10420	539

Table 4.21 depicts the top three representative cycles for each cluster, and Tables 4.22 and 4.23 contain their respective feature values. Nearly half of the feature values from cluster 1 appear higher. The values that are higher for cluster 1 are: *Torque Skewness*, *Speed Skewness*, *Fraction Idle*, and *Duration Idle*. This again shows the same results as k-means.

Table 4.21: Representative cycles per cluster.

Cluster	Rep Cycle ID
0	572
0	45
0	429
1	539
1	309
1	534

Table 4.22: Feature values for representative cycles from Consensus cluster 0.

Feature	Cycle 572	Cycle 45	Cycle 429
Torque +Δ	3.873	3.866	3.900
Torque Skew	0.658	0.637	0.638
Avg Power	84.816	84.971	90.791
Speed Skew	0.511	0.617	0.608
Speed +Δ	45.584	51.160	46.839
Fraction Idle	0.214	0.203	0.163
Count Idle	124	127	102
Duration Idle	18.024	15.520	15.111

Table 4.23: Feature values for representative cycles from Consensus cluster 1.

Feature	Cycle 539	Cycle 309	Cycle 534
Torque +Δ	2.632	2.727	2.873
Torque Skew	0.991	1.062	0.893
Avg Power	61.580	65.832	66.116
Speed Skew	1.140	0.990	1.023
Speed +Δ	30.249	31.004	29.804
Fraction Idle	0.435	0.472	0.421
Count Idle	73	96	80
Duration Idle	54.123	69.458	55.238

Figures 4.14 and 4.15 depict the box plots of clustered features for both clusters across all methods. K-means, HAC, and consensus clustering largely share the same feature values, while mean-shift stands out in terms of idle values. This is most noticeable for the idle fraction.

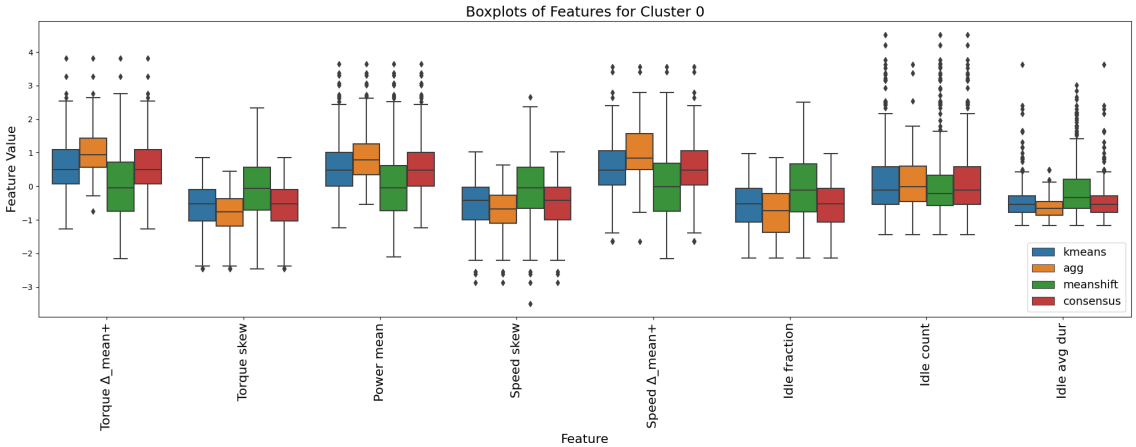


Figure 4.14: Boxplots of principal feature values across all clustering methods for cluster 0.

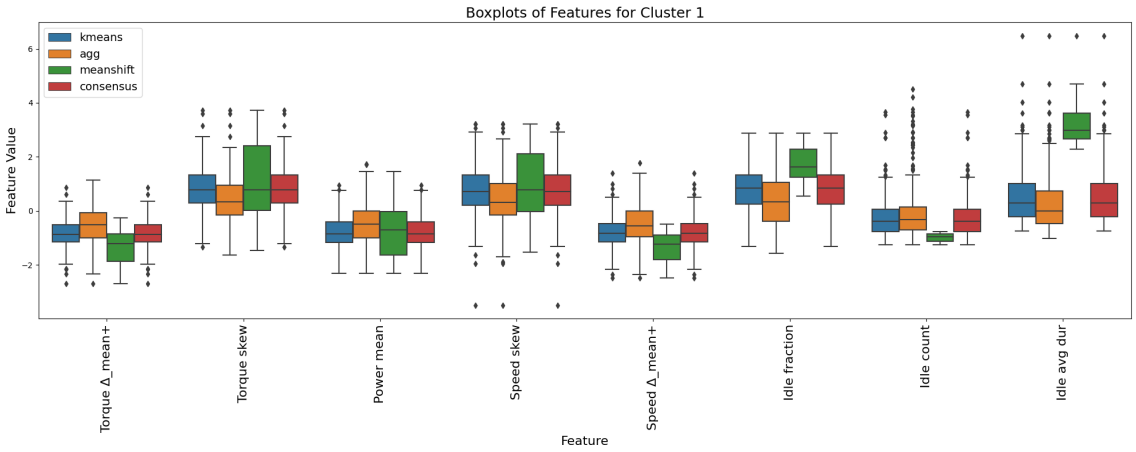


Figure 4.15: Boxplots of principal feature values across all clustering methods for cluster 1.

4.3.5 Discussion

Mean-shift obtained the highest silhouette score, as seen in Table 4.14 (0.373), compared to the other clustering algorithms. This suggests that, overall, its clusters are well-separated and internally coherent. However, it also produced a small secondary cluster containing only 3% of the cycles.

HAC yielded a lower silhouette score (0.267) and CH score (215.5), Table 4.9, compared to k-means, which indicates less compact or well-separated clusters at the chosen two-group cut. However, both HAC and k-means clustering provide similar results and are reasonably balanced compared to mean-shift in cluster sizes. This could be due to mean-shift capturing slightly different behavior. From Figure 4.15, it is evident that the main deviation mean-shift has from the other clustering methods lies in the idle fraction, count, and duration feature values in cluster 1. The most significant deviation is found in idle duration. Due to mean-shift cluster 1 only containing six cycles, this can be seen as the algorithm identifying outliers by their higher idle durations.

Comparing skewness to the positive derivative torque and speed, there is an inverse relationship between the two metrics. For each respective clustering method, cluster 0 has lower skewness values than cluster 1, but higher positive derivatives. In Cluster 0 cycles, the engine tends to ramp torque and speed more often and with more uniform increases. Rises are substantial but remain relatively consistent, which produces a compact distribution with few extreme outliers. In contrast, cluster 1 cycles exhibit slower increases most of the time, accompanied by rare but larger spikes that create the long right-hand tail in their distributions. The pattern found in cluster 0 can be identified as high-load, and for cluster 1, it is in line with low-load. Reviewing the average power for the two clusters, it is noted that cluster 0 has an overall higher average compared to cluster 1, which is in line with high, respectively low-load engine behavior.

One could think that a high idle count would correlate with a high idle fraction. However, this is not the case, indicating that engines that idle often do not idle for long periods of time. Comparing the high-load and low-load clusters reveals a relatively high idle fraction for the high-load cluster. This could be due to the cycles with low idle fractions often operating with low power output and being less aggressive. Moreover, the persistence of idling features suggests that even subtle variations in stop-start behavior can help distinguish between operational modes. For instance, two cycles with similar torque and speed profiles may serve different functions if one contains a series of short, frequent stops while the other features long, infrequent idles.

Consensus clustering, which was built on co-association from multiple algorithms, reproduces the k-means solution exactly. This could be seen as the k-means clustering solution being dominant. Still, it could also result from the mean shift having almost only one cluster, and the consensus clustering arbitrarily choosing one of the two methods' results. Other algorithms, despite their differing approaches, agreed on those core groupings. This means that the consensus process wrote over more subtle splits that mean shift or HAC might have shown. This reinforced the primary division between high- and low-load. At the same time, this consistent increase in confidence in § two main operating modes suggests that any deeper nuance will require either more sensitive clustering methods or extending features that capture the secondary patterns that those methods briefly surfaced.

5

Conclusion and Future Work

5.1 Conclusion

This thesis aimed to develop a framework for identifying representative engine behaviors starting from raw time series field data from one engine. A segmentation method, combined with filtering, yielded 460 cycles for clustering. Relevant statistical and temporal features for engine usage were derived from these cycles, and a final subset was identified through PFA feature selection, which balances dimension reduction with interpretable features.

Three clustering algorithms — k-means, HAC, and mean-shift — were used to group the cycles based on accompanying features. These fundamentally different clustering solutions provided the basis for a subsequently performed consensus clustering solution. Sil, DB, and CH scores were used for internal clustering validation of the methods. Based on these scores, mean-shift clustering performed the best.

Clustering solutions consistently displayed two clusters characterizing engine usage, except for mean-shift, which primarily showed one major behavior and cycles deviating from this behavior. The feature most deviant for these cycles is idle duration, suggesting that this feature had a high impact in singling out these outliers from the main behavior. The resulting clusters from the other clustering algorithms instead displayed two major behaviors: low-load and high-load. Low-load was characterized by low mean derivatives, low power values, and high skewness, signaling occasional large signal spikes. On the other hand, high-load cycles exhibited high mean derivatives and power valued but lower skewness. Thus, these load modes are not solely defined by average power or idle duration but also by the engine's acceleration. The high-load cluster's elevated idle count further reinforces this conclusion.

Consensus clustering ultimately mirrored k-means, likely due to mean-shift offering almost no second group. Thus, the ensemble reinforced the k-means split as the most agreed-upon consensus. Consensus clustering is designed to reveal the most robust and stable partition across clustering methods. However, its effectiveness may depend on improving ensemble strategies and determining partition sizes effectively to ensure meaningful results.

In conclusion, this thesis demonstrated that feature-based clustering of drive cycles can be used to uncover major engine behaviors, depending on the clustering

technique used. Additionally, it provided a method for identifying the most common engine behavior and distinguishing outlier cycles.

5.2 Future Work

This thesis work lays the foundation for analyzing and characterizing engine behavior via unsupervised clustering, but several avenues remain to extend and strengthen the methodology.

The current definition of a drive cycle could be improved. Future techniques to investigate include temporal segmentation and techniques such as adaptive windowing or dynamic time-warping thresholds to capture more detailed changes in engine operations. Refinements could enhance the consistency of feature extraction and enable more precise comparisons across datasets.

Currently, the dataset only contains one engine utilized for one application, which means that within-application variability can not be evaluated. Adding multiple engines used under the same application may not introduce new behavior classes for the application, but it will increase the diversity of the data utilized. This larger sample set could provide cluster separation between different applications, enabling the method to distinguish between application-driven behaviors and engine-to-engine variation accurately. Furthermore, validation on additional engines of varying application types would help further assess the generality of the feature set and clustering framework. The expansion could bestow insight into which features are universally discriminative and which features are specific to particular applications.

Another area of interest is analyzing the outcome's sensitivity to different cycle lengths or segmentations. This thesis discards cycles shorter than 60 minutes and removes short idling periods according to a fixed criterion. Whether these choices strongly influence the resulting feature distributions and clusterings remains unclear. A systematic investigation with varying minimum cycle durations and adjusted idle removal parameters would reveal whether the two main clustering structures stay the same or produce different results. Such analysis could help determine parameters that best capture meaningful behaviors of the engines.

Continuing, only three clustering methods were used. K-means, as it is one of the most common clustering techniques used, HAC for its capability to function without a predefined cluster count and a data-driven algorithmic selection approach detailed in Section 3.2, from which mean-shift was chosen. Two research directions are proposed to build on these findings. First, systematically evaluate the other two clustering methods from the algorithmic selection approach, Shared Nearest Neighbor and Support Vector Clustering. Another suggestion is to allow a larger ensemble of cluster methods to be used in the consensus clustering stage. Since only three base algorithms were used, the ensemble may lack sufficient partitions to achieve a robust consensus. Moreover, testing multiple consensus strategies, such as combining the co-association matrix approach with graph-based consensus clustering,

could use different modes of agreement among algorithms.

In conclusion, it is vital to remember that clustering results are ultimately shaped by the choices made at the outset. The definition of cycles, the selection of features, and the application of different algorithms, along with varying initial settings, would highlight different patterns. Therefore, systematically testing these parameters is key to understanding and trusting the findings.

A

Appendix 1

This appendix contains pair plots generated from each clustering algorithm. The feature importance goes from top to bottom and left to right according to the PFA selection.

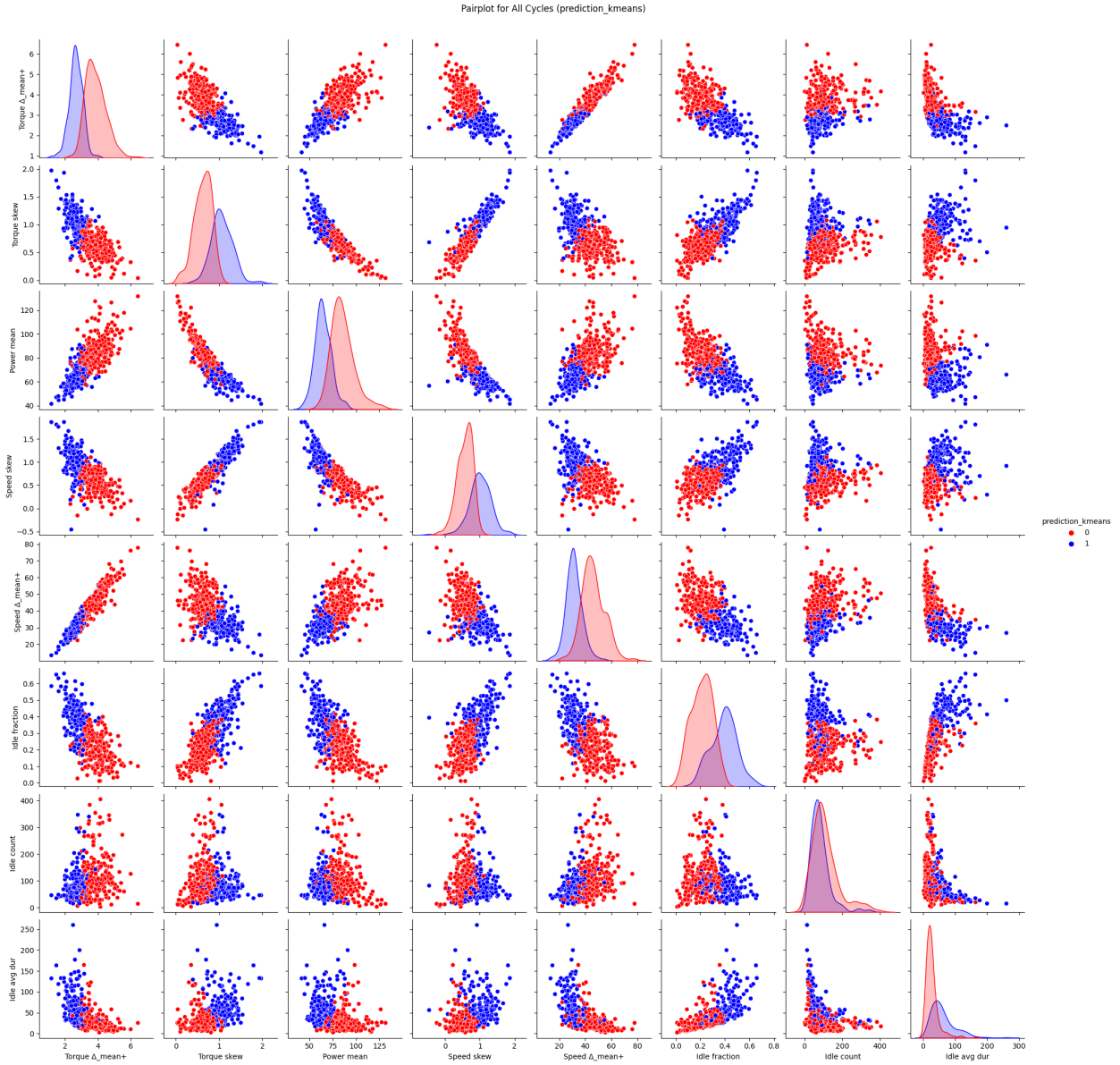


Figure A.1: K-means pair plots for all cycles using all principal features.

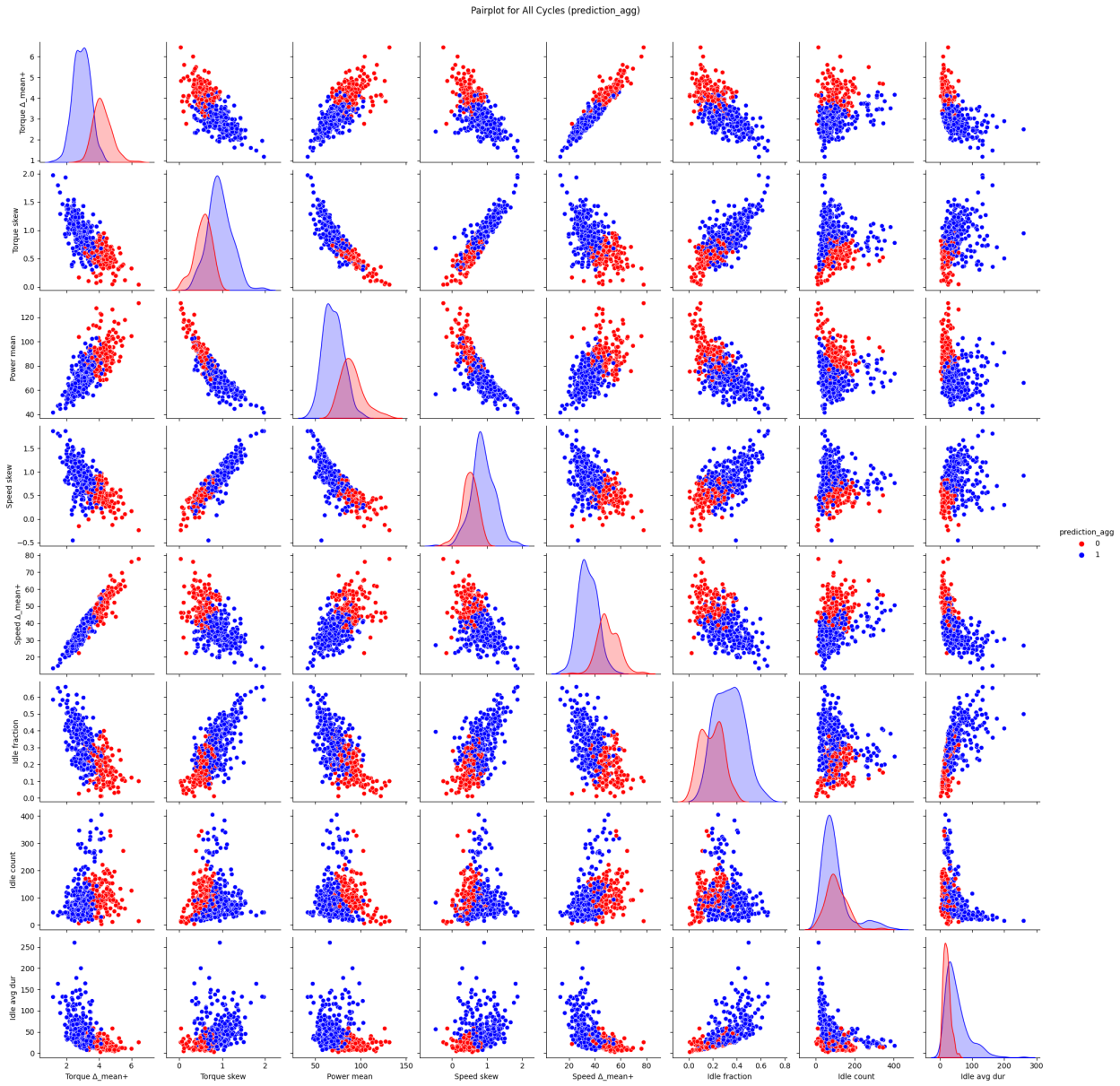


Figure A.2: HAC pair plots for all cycles using all principal features.

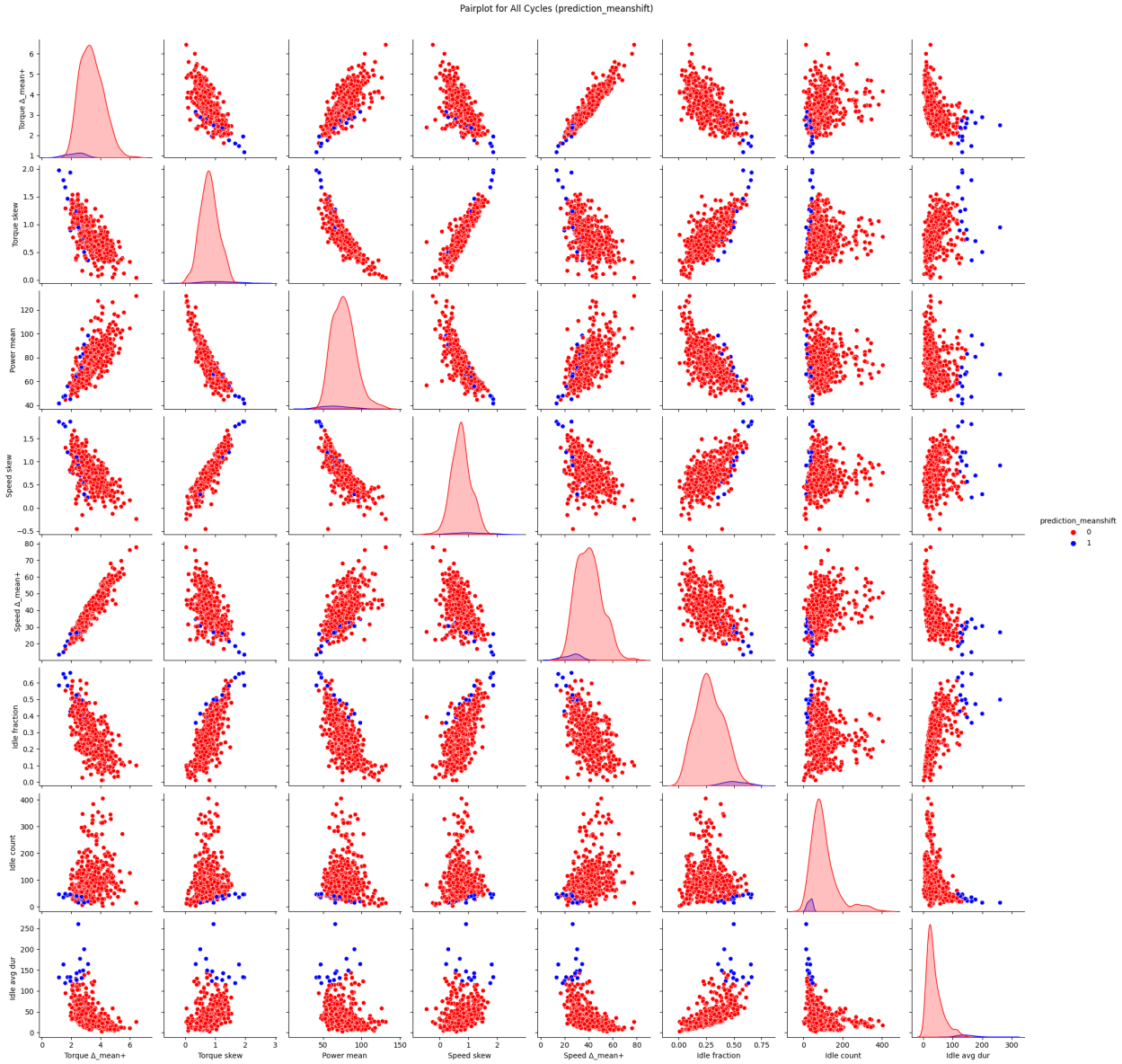


Figure A.3: Mean-shift pair plots for all cycles using all principal features.

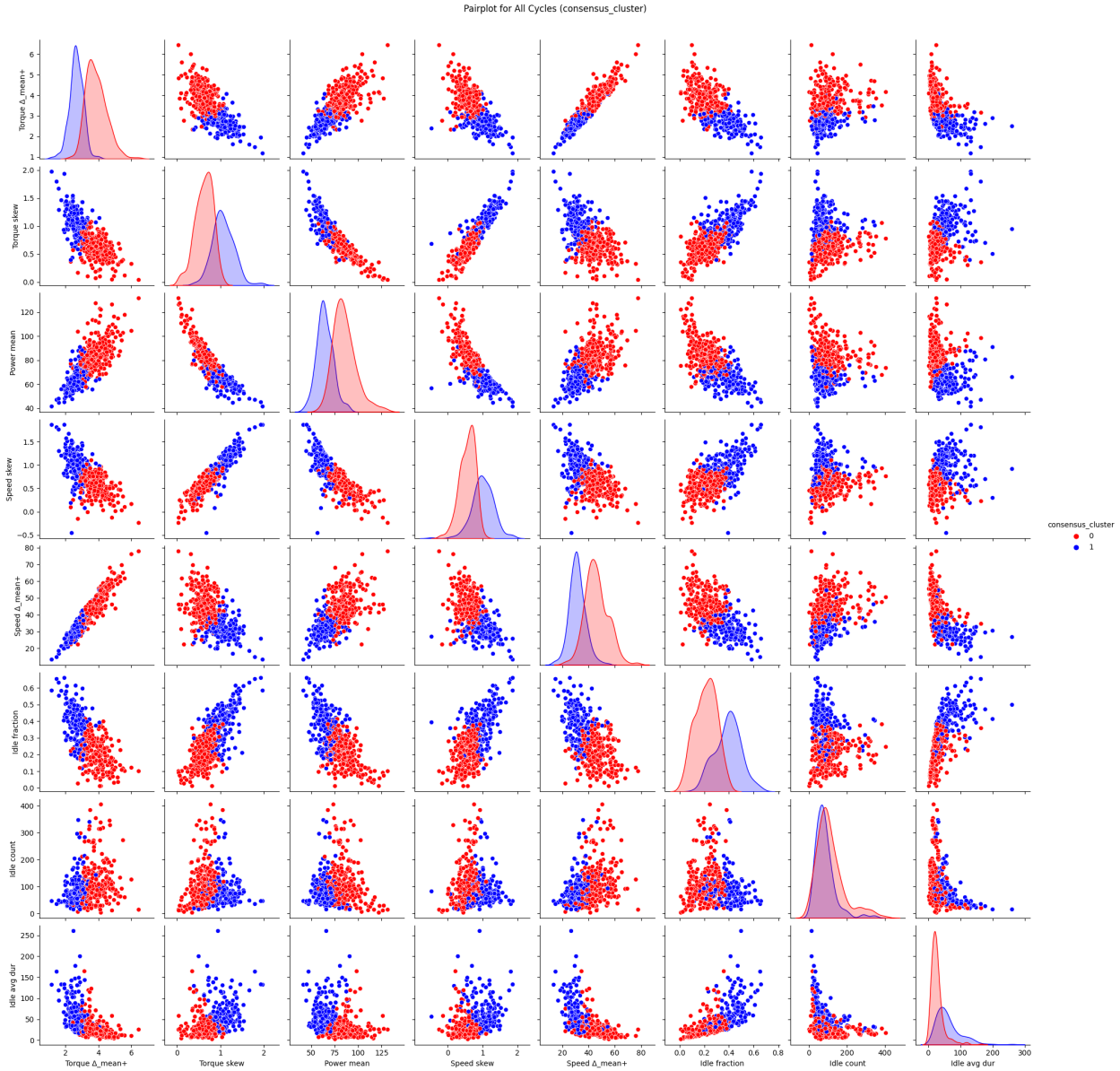


Figure A.4: Consensus pair plots for all cycles using all principal features.

B

Appendix 2

This appendix contains figures of the raw data time series signal values of the top three cycles for each cluster produced by the different clustering algorithms used in this thesis.

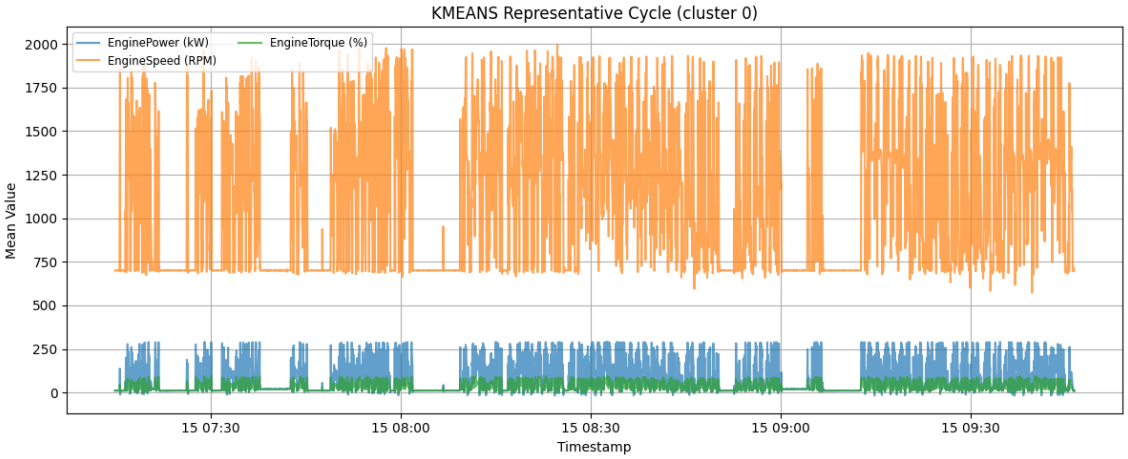


Figure B.1: Time series of signal values from cycle 572 — cluster 0 k-means.

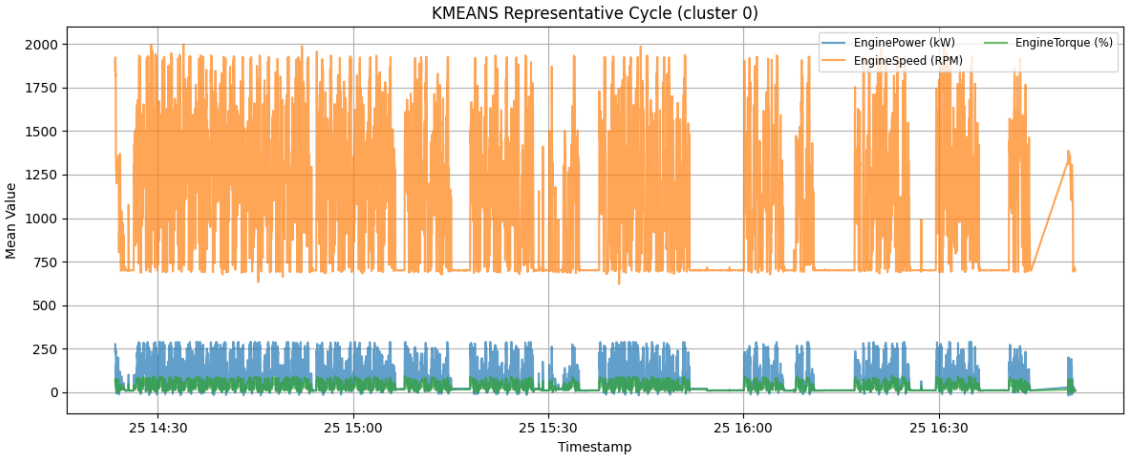


Figure B.2: Time series of signal values from cycle 45 — cluster 0 k-means.

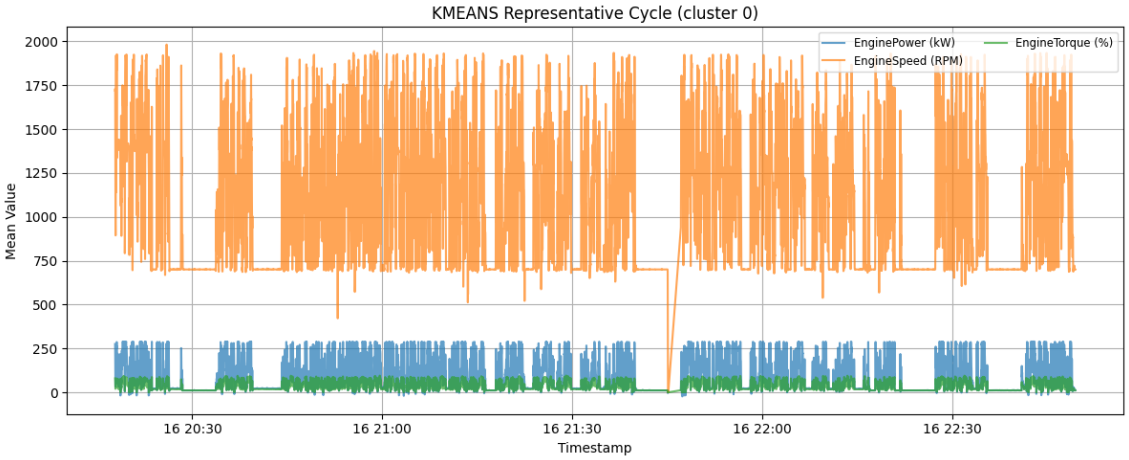


Figure B.3: Time series of signal values from cycle 429 — cluster 0 k-means.

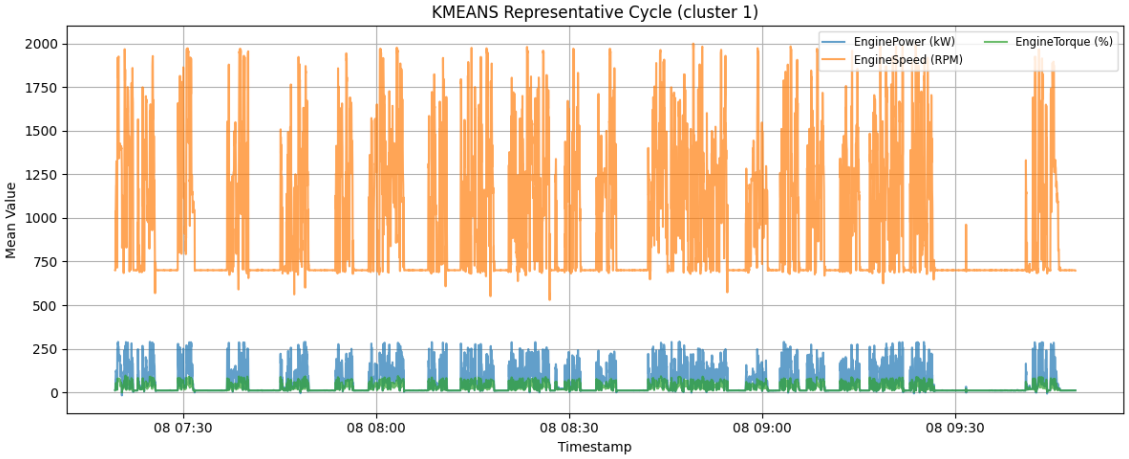


Figure B.4: Time series of signal values from cycle 539 — cluster 1 k-means.

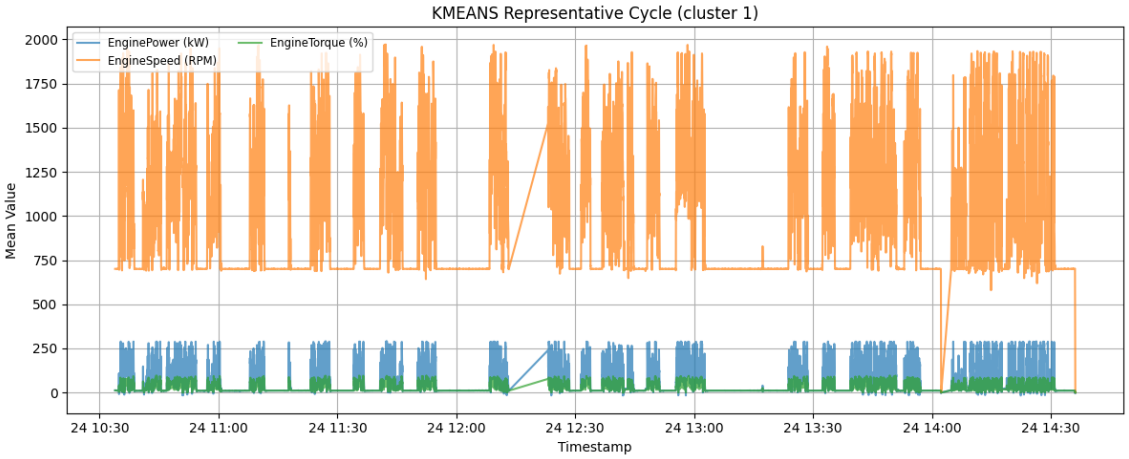


Figure B.5: Time series of signal values from cycle 309 — cluster 1 k-means.

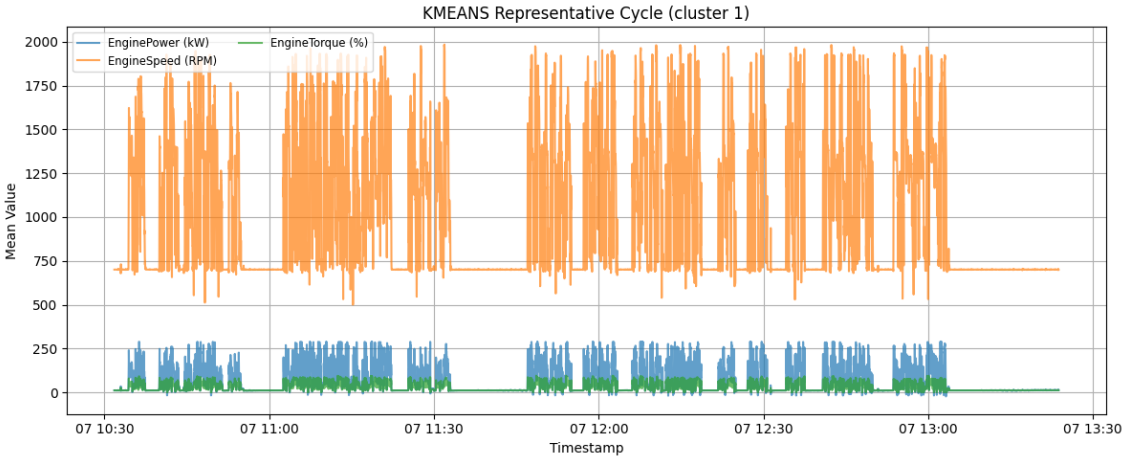


Figure B.6: Time series of signal values from cycle 534 — cluster 1 k-means.

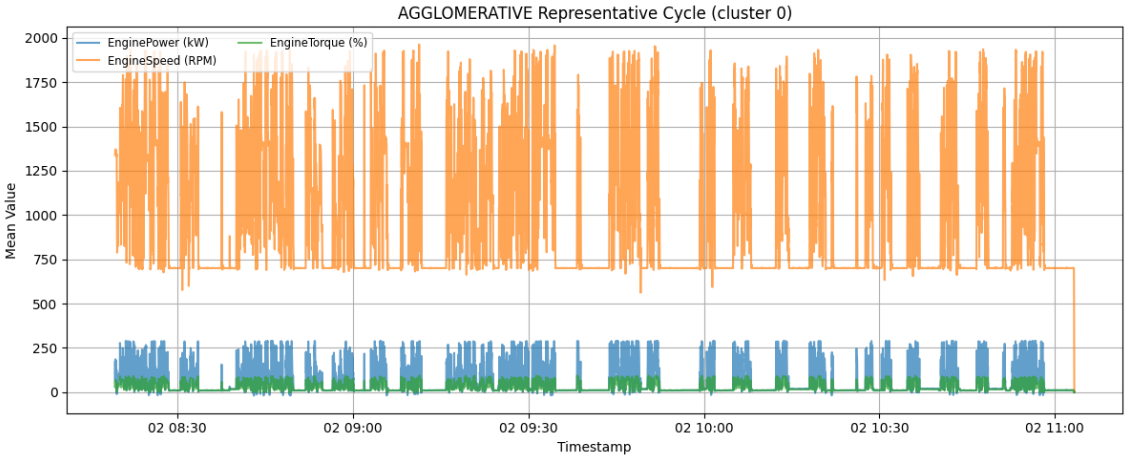


Figure B.7: Time series of signal values from cycle 80 — cluster 0 HAC.

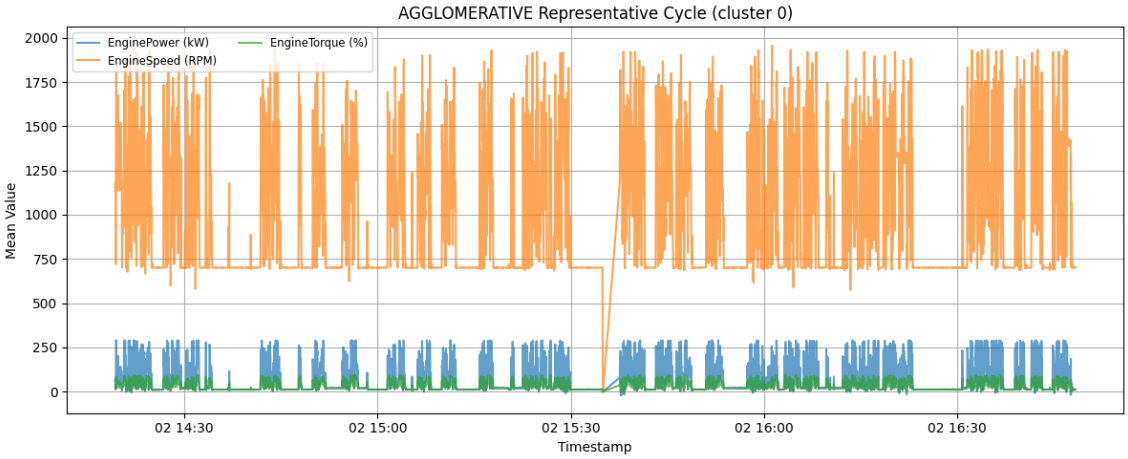


Figure B.8: Time series of signal values from cycle 82 — cluster 0 HAC.

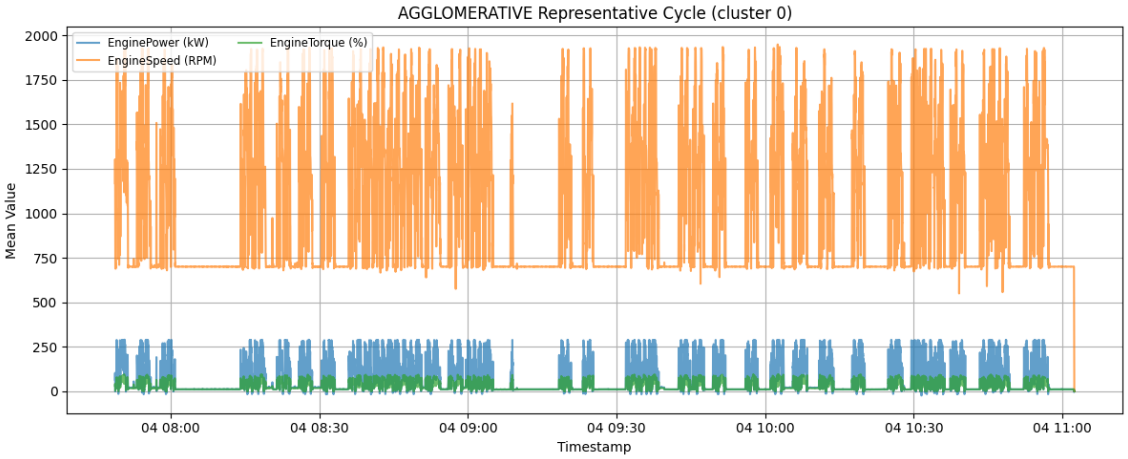


Figure B.9: Time series of signal values from cycle 361 — cluster 0 HAC.

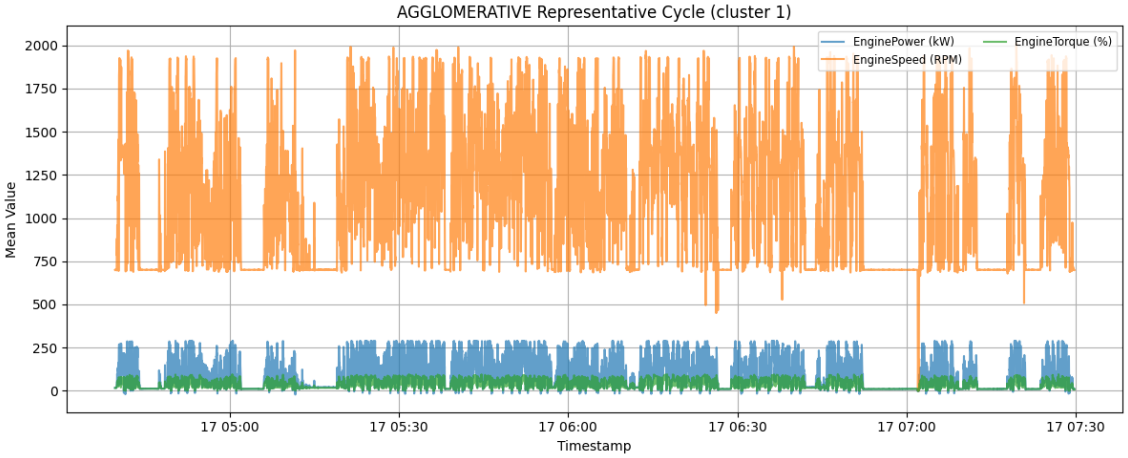


Figure B.10: Time series of signal values from cycle 430 — cluster 1 HAC.

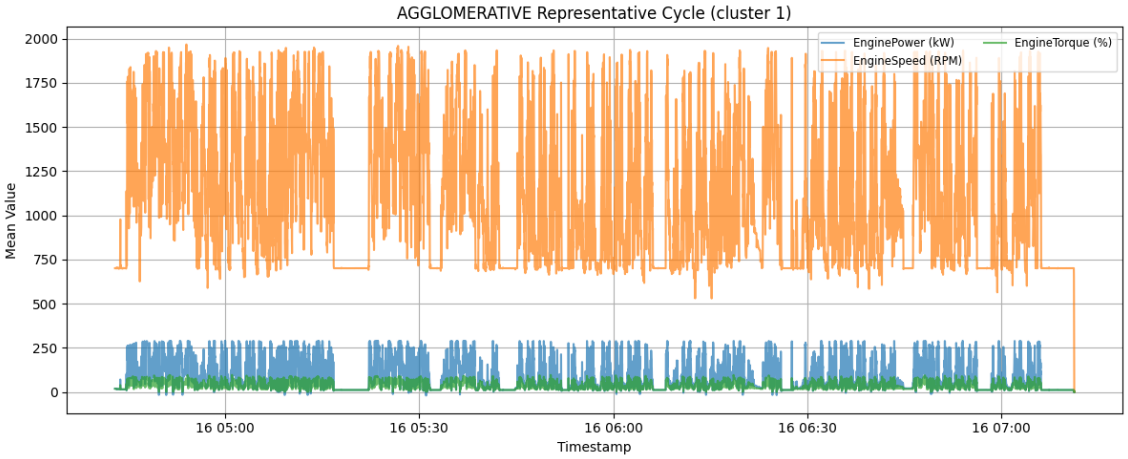


Figure B.11: Time series of signal values from cycle 272 — cluster 1 HAC.

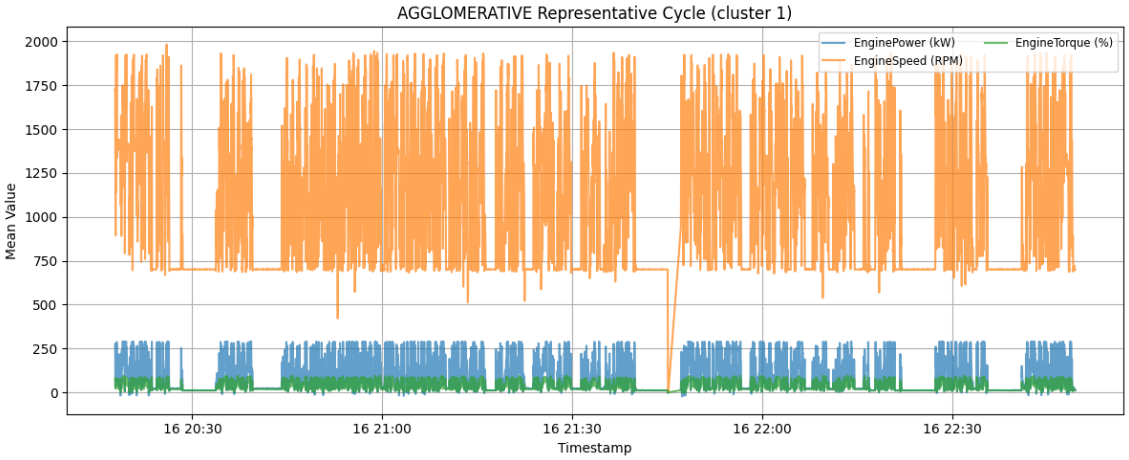


Figure B.12: Time series of signal values from cycle 429 — cluster 1 HAC.

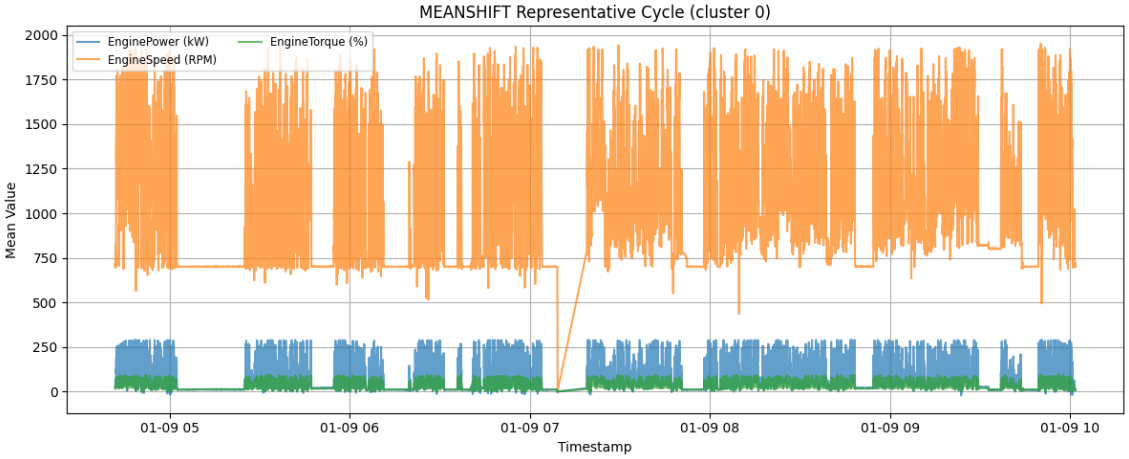


Figure B.13: Time series of signal values from cycle 240 — cluster 0 mean-shift.

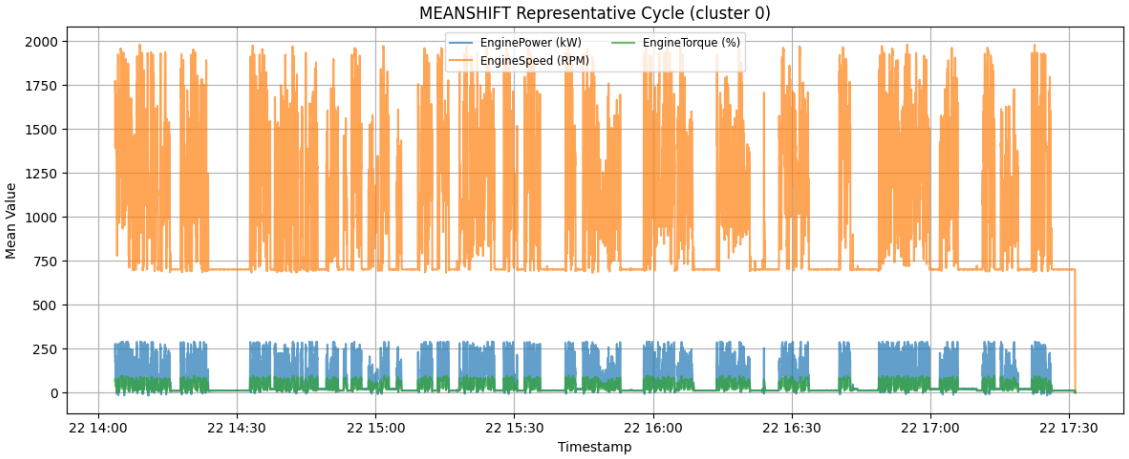


Figure B.14: Time series of signal values from cycle 189 — cluster 0 mean-shift.

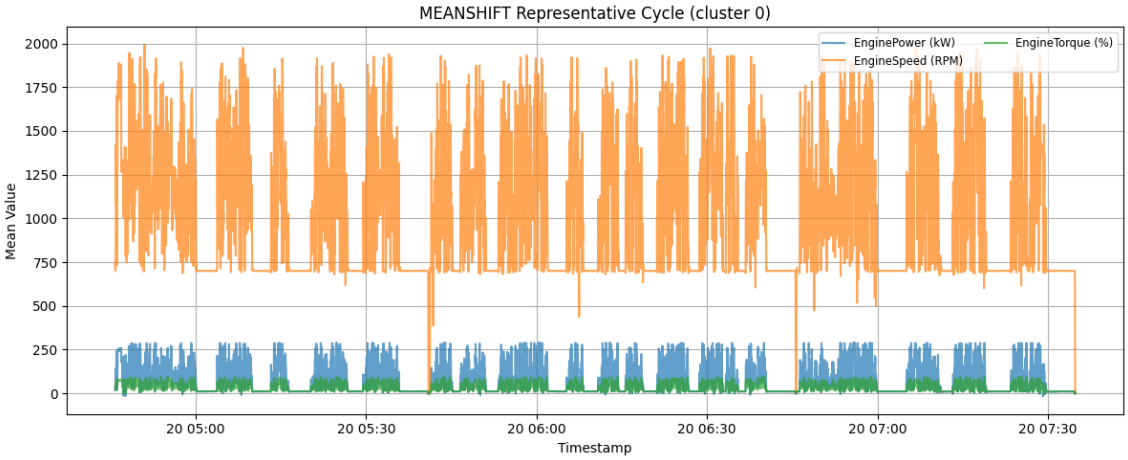


Figure B.15: Time series of signal values from cycle 286 — cluster 0 mean-shift.

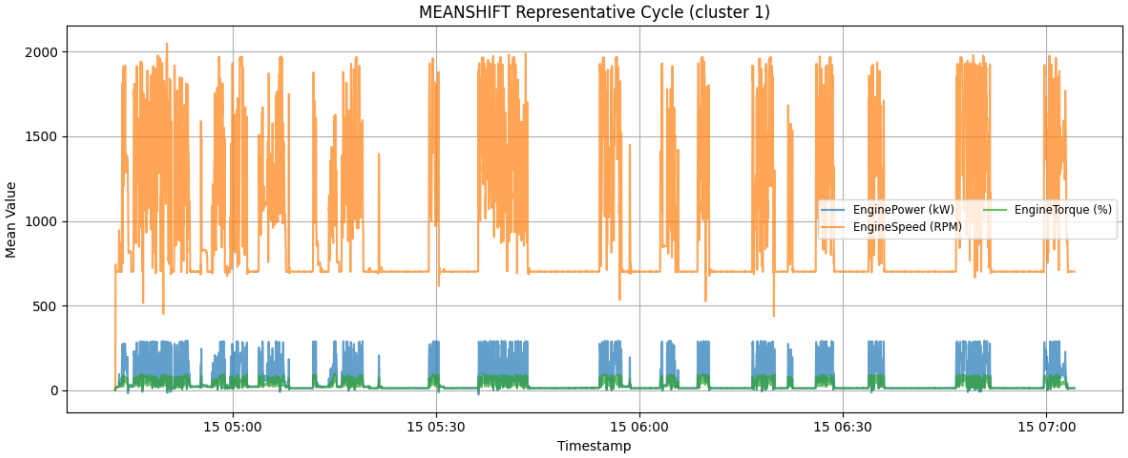


Figure B.16: Time series of signal values from cycle 420 — cluster 1 mean-shift.

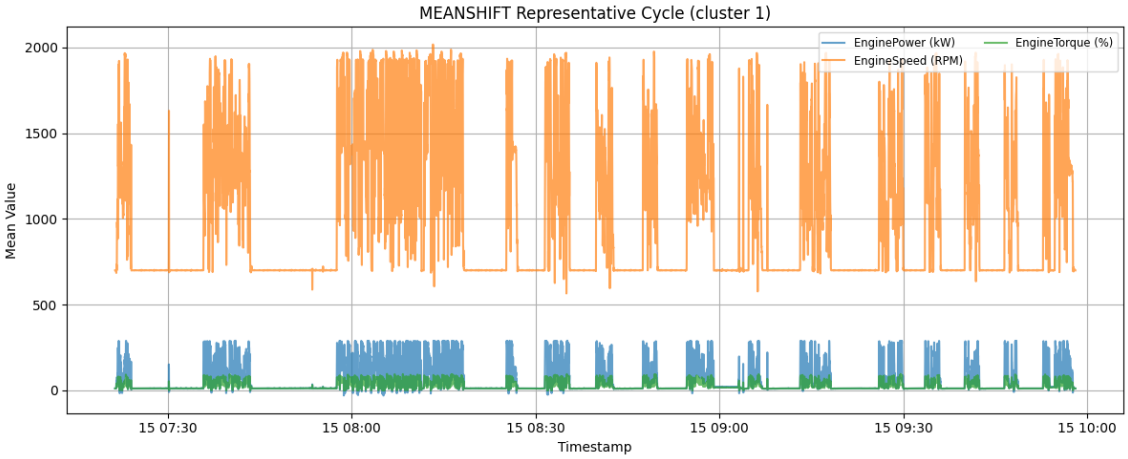


Figure B.17: Time series of signal values from cycle 161 — cluster 1 mean-shift.

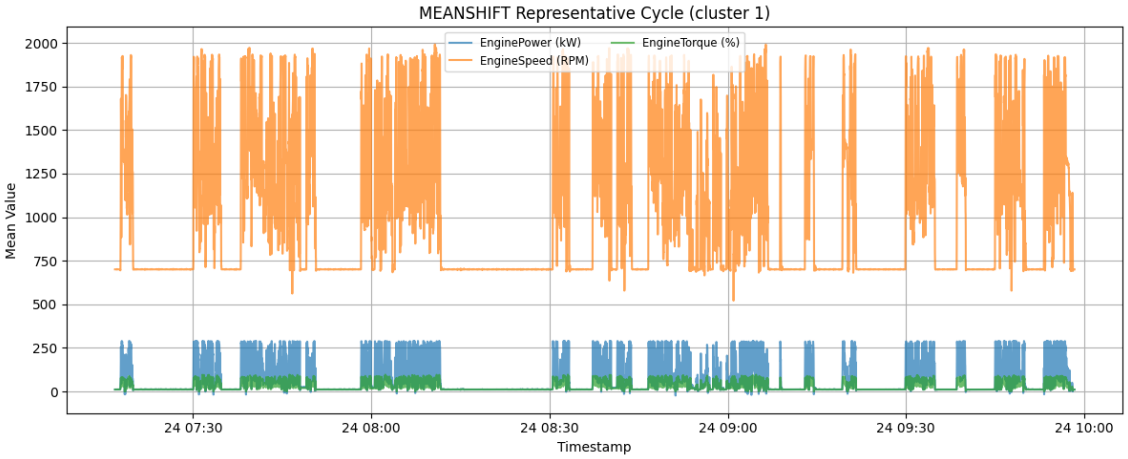


Figure B.18: Time series of signal values from cycle 308 — cluster 1 mean-shift.

Bibliography

[1] M. A. B. Syed, M. R. Hasan, N. I. Chowdhury, M. H. Rahman, and I. Ahmed, "A systematic review of time series algorithms and analytics in predictive maintenance," *Decision Analytics Journal*, vol. 15, p. 100 573, 2025, ISSN: 2772-6622. DOI: https://doi.org/10.1016/j.dajour.2025.100573. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2772662225000293.

[2] J. Enes, R. R. Expósito, J. Fuentes, J. L. Cacheiro, and J. Touriño, "A pipeline architecture for feature-based unsupervised clustering using multivariate time series from hpc jobs," *Information Fusion*, vol. 93, pp. 1–20, 2023, ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2022.12.017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253522002652.

[3] B. D. Fulcher and N. S. Jones, "Highly comparative feature-based time-series classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 12, pp. 3026–3037, 2014. DOI: 10.1109/TKDE.2014.2316504.

[4] M. Johansson and K.-F. Zingaropoli, "Detection, semantic segmentation & generation of engine drive cycles using machine learning," Department of Electrical Engineering, Master's thesis, Chalmers University of Technology, Gothenburg, Sweden, Jun. 2021. [Online]. Available: https://odr.chalmers.se/items/c64d5319-40df-4248-832e-89077273e3e2.

[5] A. Bonifati, F. D. Buono, F. Guerra, and D. Tiano, "Time2feat: Learning interpretable representations for multivariate time series clustering," *Proc. VLDB Endow.*, vol. 16, no. 2, pp. 193–201, Oct. 2022, ISSN: 2150-8097. DOI: 10.14778/3565816.3565822. [Online]. Available: https://doi.org/10.14778/3565816.3565822.

[6] W. Zhuang, J. Fan, J. Fang, W. Fang, and M. Xia, "Rethinking general time series analysis from a frequency domain perspective," *Knowledge-Based Systems*, vol. 301, p. 112 281, 2024, ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2024.112281. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705124009158.

[7] M. B. Bjerregård, J. K. Møller, and H. Madsen, "An introduction to multivariate probabilistic forecast evaluation," *Energy and AI*, vol. 4, p. 100 058, 2021, ISSN: 2666-5468. DOI: https://doi.org/10.1016/j.egyai.2021.100058. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666546821000124.

[8] T. Verdonck, B. Baesens, M. Óskarsdóttir, and S. vanden Broucke, "Special issue on feature engineering editorial," *Machine Learning*, vol. 113, 7 2024,

ISSN: 15730565. DOI: 10.1007/s10994-021-06042-2. [Online]. Available: https://link.springer.com/article/10.1007/s10994-021-06042-2#citeas.

[9] M. K. Kondaru, K. P. Telikepalli, S. V. Thimmalapura, and N. K. Pandey, "Generating a real world drive cycle–a statistical approach," in *WCX World Congress Experience*, SAE International, Apr. 2018. DOI: https://doi.org/10.4271/2018-01-0325. [Online]. Available: https://doi.org/10.4271/2018-01-0325.

[10] scikit-learn developers, *2.3. Clustering*. [Online]. Available: https://scikit-learn.org/stable/modules/clustering.html%5C#k-means.

[11] scikit-learn developers, *KMeans*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html.

[12] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, vol. 07-09-January-2007, 2007. [Online]. Available: https://theory.stanford.edu/~sergei/papers/kMeansPP-soda.pdf.

[13] R. P. Adams, *Hierarchical clustering*, 2018. [Online]. Available: https://www.cs.princeton.edu/courses/archive/fall18/cos324/files/hierarchical-clustering.pdf.

[14] D. Müllner, *Modern hierarchical, agglomerative clustering algorithms*, 2011. arXiv: 1109.2378 [stat.ML]. [Online]. Available: https://arxiv.org/abs/1109.2378.

[15] scikit-learn developers, *AgglomerativeClustering*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html.

[16] J. Noble, *What is hierarchical clustering?* 2024. [Online]. Available: https://www.ibm.com/think/topics/hierarchical-clustering.

[17] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790–799, 1995. DOI: 10.1109/34.400568.

[18] K.-L. Wu and M.-S. Yang, "Mean shift-based clustering," *Pattern Recognition*, vol. 40, no. 11, pp. 3035–3052, 2007, ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2007.02.006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320307000921.

[19] M. Á. Carreira-Perpiñán, *A review of mean-shift algorithms for clustering*, 2015. arXiv: 1503.00687 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1503.00687.

[20] A. L. Fred and A. K. Jain, "Combining multiple clusterings using evidence accumulation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 6, pp. 835–850, 2005. DOI: 10.1109/TPAMI.2005.113.

[21] N. Nguyen and R. Caruana, "Consensus clusterings," in *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, 2007, pp. 607–612. DOI: 10.1109/ICDM.2007.73.

[22] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987, ISSN: 0377-0427. DOI: https://doi.org/10.1016/

0377-0427(87)90125-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0377042787901257.

[23] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, 1979. DOI: 10.1109/TPAMI.1979.4766909.

[24] MathWorks, *Daviesbouldinevaluation*. [Online]. Available: https://se.mathworks.com/help/stats/clustering.evaluation.daviesbouldinevaluation.html.

[25] F. Pedregosa *et al.*, *Scikit-learn: Machine learning in python*, 2011. [Online]. Available: https://scikit-learn.org/stable/modules/clustering.html#calinski-harabasz-index.

[26] I. Veza, M. F. M. Said, and Z. A. Latiff, "Progress of acetone-butanol-ethanol (abe) as biofuel in gasoline and diesel engine: A review," *Fuel Processing Technology*, vol. 196, p. 106 179, 2019, ISSN: 0378-3820. DOI: https://doi.org/10.1016/j.fuproc.2019.106179. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S037838201930966X.

[27] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, no. null, pp. 1157–1182, Mar. 2003, ISSN: 1532-4435.

[28] I. T. Jolliffe and J. Cadima, "Principal component analysis: A review and recent developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, p. 20 150 202, 2016. DOI: 10.1098/rsta.2015.0202. eprint: https://royalsocietypublishing.org/doi/pdf/10.1098/rsta.2015.0202. [Online]. Available: https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2015.0202.

[29] Y. Lu, I. Cohen, X. S. Zhou, and Q. Tian, "Feature selection using principal feature analysis," in *Proceedings of the 15th ACM International Conference on Multimedia*, ser. MM '07, Augsburg, Germany: Association for Computing Machinery, 2007, pp. 301–304, ISBN: 9781595937025. DOI: 10.1145/1291233.1291297. [Online]. Available: https://doi.org/10.1145/1291233.1291297.

[30] M. Wegmann, D. Zipperling, J. Hillenbrand, and J. Fleischer, *A review of systematic selection of clustering algorithms and their evaluation*, 2021. arXiv: 2106.12792 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2106.12792.

DEPARTMENT OF PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY