POLITECNICO DI TORINO

Master's Degree in DATA SCIENCE AND ENGINEERING

Master's Degree Thesis in collaboration with Tierra S.p.A

An autoencoder-based clustering strategy for usage pattern detection on heavy duty's vehicles' CAN bus data

Supervisors

Prof. Francesco VACCARINO

Prof. Luca CAGLIERO

Dr. Silvia BUCCAFUSCO

Candidate

Ruggiero FRANCAVILLA

Company Tutors

Dr. Lucia SALVATORI

Dr. Riccardo LOTI

October 2021

Summary

This thesis work addresses a real case problem that concern heavy duty's vehicle and patterns. Due to the diffusion of IoT devices and the establishing of cars connected mobility in firms, the connectivity of heavy-duty in-vehicle is becoming a more and more important task in pattern identification tasks.

This work is developed in Tierra S.p.A., a company that creates innovative solutions in advanced telematics and IoT fields and that is part of the collaboration between the applied research and data analytics department and Politecnico di Torino.

The purpose of this work is the identification of pattern thresholds in heavy duty's vehicles, due to clients' failures in manual detection. For this task, multivariate time series data analysis with an innovative autoencoder-based technique is presented. After a first exploration of signals, and the identification of the most relevant ones for the task, a proper combination of series with different sampling rates is performed. Then, for data preparation purposes, a segmentation strategy for the multivariate time series based on VALMOD algorithm, is proposed. Moreover, combined action of autoencoder models and clustering techniques is used for the pattern identification task. In particular, this approach looks at finding patterns in data, by exploiting the reconstruction ability of signals of autoencoder models. Therefore, different usage patterns are identified with the application of a clustering technique on a customized dissimilarity matrix based on autoencoders reconstruction error. Finally, a score-based strategy helps at identifying thresholds between usage patterns.

As last step, after a manual validation with the help of experts, with visualization tools, a silhouette-based approach analyzing clustering results coming from different distance measures is described analyzing both separability and cohesion of groups, and the usage patterns detected.

Acknowledgements

Ringrazio prima di tutto il mio relatore, il Professor Francesco Vaccarino, per avermi offerto la possibilità di entrare a far parte di questa realtà e per le preziose indicazioni, la sua costante presenza ed il sostegno non solo dal lato universitario, ma anche morale in questo percorso.

Vorrei inoltre ringraziare i miei correlatori, il Professor Luca Cagliero, per la supervisione costante del mio lavoro, e per essere stato sempre fonte di possibili soluzioni, ed alternative durante il lavoro di tesi, e la Dottoressa Silvia Buccafusco, la mia guida in questo percorso, senza le cui indicazioni, consigli e disponibilità in qualsiasi momento, avrei avuto molte più difficoltà durante questi mesi.

I miei più sentiti ringraziamenti vanno all'azienda Tierra S.p.A., ed ai suoi rappresentanti con cui ho avuto il piacere di collaborare, tra cui Lucia Salvatori e Riccardo Loti. I meeting settimanali non solo mi hanno dato una dimensione aziendale, ma sono stati utili per i vostri continui suggerimenti e attenzioni che mi avete rivolto in questo lavoro di tesi.

Table of Contents

List of Tables

List of Figures

Acronyms

AE

AutoEncoders

AI

Artificial Intelligence

ANN

Artificial Neural Network

AWS

Amazon Web Service

BIBO

Bounded-Input Bounded-Output

CAN

Controller Area Network

CAN FD

Controller Area Network Flexible Data-Rate

CNN

Convolutional Neural Network

DL

Deep Learning

DTFT

Discrete Time Fourier Transform

FIR

Finite Impulse Response

GPS

Global Positioning System

IIR

Infinite Impulse Response

IoT

Internet of Things

LSI

Linear Shift-Invariant

LSTM

Long Short-Term Memory

LTI

Linear Time-Invariant

ML

Machine Learning

MSE

Mean Squared Error

OBD

On-Board Diagnostics

PGN

Parameter Group Number

RNN

Recurrent Neural Network

SAE

Society of Automotive Engineers

SGD	Stochastic Gradient Descent
SPN	Suspect Parameter Number
VAE	Variational AutoEncoder
VALMOD	Variable Length Motifs Discovery

Chapter 1

Introduction and Relative Work

In this section, the main topics covered by the work are presented. Moreover, an overview of the chapters is provided.

Machine Learning (ML) is a set of methods useful to automatic detection of patterns in data, with the objective of using it to support decisions under uncertainty, or to use them for future data prediction. It is a branch of *Artificial Intelligence (AI)*, composed of different algorithms, automatically improved through experience and use of data. *Deep Learning (DL)* is instead a class of ML that uses multiple layers to progressively extract higher-level features from the raw input.

The spread in these years of these topics, often related to big data, a large amount of information, make them the most important computer science areas.

Learning is, of course, a very wide domain, and consequently, several subfields have been generated from the principal one, each dealing with different learning tasks types. Since learning involves an interaction between the learner and the environment, two different categories can be identified by looking at the nature of these interactions.

Supervised learning, or predictive learning, has the aim of mapping an input to output by looking for a model that explains the relationships between input and output variables. The main goal of these kinds of algorithms is to forecast an unknown response variable for new observations. In supervised learning context, each example in the dataset is associated with a *label* or *target*, that "is provided" by an instructor or a teacher that supervises the ML system. Examples of supervised algorithms are bayesian networks, decision trees, support vector machines...

On the other hand, *unsupervised learning* tries to learn patterns from untagged data. The learner processes input data to come up with a summary or a compressed version of that data. In these algorithms, there is no supervision, so the learning must have sense without an instructor's guide. One of the most important unsupervised algorithms is *clustering*.

The method proposed in this work involves the combination of unsupervised DL and ML techniques to solve a real case problem in the *Internet of Things* (IoT) field.

IoT provides a description of the network of "things", physical objects, that with the help of sensors, software and other technologies aim to connect devices and systems each other and enable exchange of data over the Internet.

Thanks to the spread of IoT devices and the establishing of cars connected mobility in firms, heavy-duty in-vehicle connectivity is becoming a more and more important task in patterns identification field.

This work exploits heavy-duty vehicles' data for identifying usage patterns and workload states from *Controller Area Network (CAN)* bus data. The information coming from these kinds of vehicles is not frequently subject of analysis although several car usage patterns identification tools can be found in the literature. These data are used by clients through the computation of some statistics.

An important criterion for the management of the clients' requests, is the identification of the states of the vehicles, and the detection of thresholds between different patterns, to avoid clients' failures in manual detection. IoT devices allow firms to monitor their equipment from the large number of sensors installed on each type of vehicle. ML algorithms can be very useful for the identification of hidden relationships in data and detect patterns.

Specifically, this work concerns heavy-duty vehicle patterns and workload states identification from CAN bus data, generated in the vehicle at high frequency, gathered by a controller, and finally sent to centralized servers. In this work, a $Z55$ device, an experimental on-board data logger provided by the company Tierra S.p.A. collects CAN bus data. Data are then transmitted by a SIM card to Tierra cloud infrastructure and can be visualized and handled by their clients through a customized web-based remote management system.

The thesis is organized into 6 chapters, which can be summarized as follows.

In *Chapter 2*, after a theoretical introduction to CAN bus data and signal processing fundamentals, a preliminary univariate analysis of CAN bus signal is described. To this purpose, details about data cleaning and data preprocessing steps required by the kind of data under analysis are provided.

In *Chapter 3* after Multi-rate systems description, a proper combination of time series, preceded by signals synchronization, is performed to obtain a multivariate object. Then, it follows a deeper analysis exploiting relations between variables in the multivariate time series.

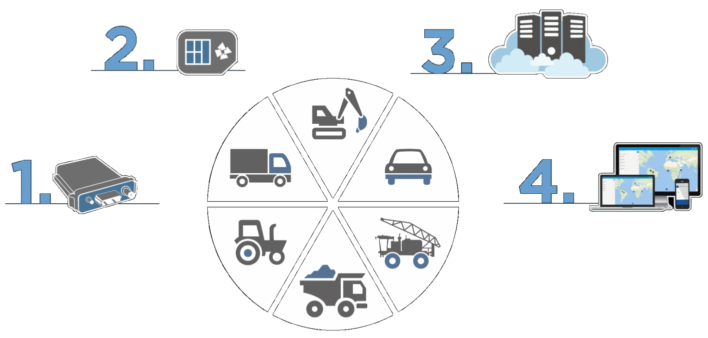


Figure 1.1: Tierra Solution [1]

In *Chapter 4* an innovative autoencoder-based method to cluster multivariate time series is employed to solve the problem under analysis. Since it requires data segmentation, the most appropriate segment length is inferred as an application of VALMOD, an algorithm to discover variable-length repeated patterns in data. Furthermore, it is described how different usage patterns are detected with an application of a clustering algorithm and Silhouette measure for the correct number of clusters.

Given the clusters obtained in the previous chapter, *Chapter 5* consists of a qualitative and quantitative evaluation of results. The former exploits domain experts knowledge to compare obtained results with expected ones. A quantitative evaluation can be instead obtained by comparing the silhouette score of the proposed method with the ones achieved by applying standard clustering with classical distances.

Therefore in the last chapter, conclusions, and further improvements are highlighted.

[1] `www.tierratelematics.com`

Chapter 2

Controller Area Network Bus Data Analysis

In this chapter, a brief theoretical introduction to CAN Bus Data, the relative protocol, and signals processing fundamentals is provided. Then, details about the preliminary phases of data collection, data cleaning and data preparation are described.

For each CAN signal, a univariate analysis is firstly performed, while interactions between physical quantities are deeply analyzed in the last part of the chapter.

2.1 Controller Area Network Bus Data

2.1.1 CAN Bus Data Introduction

CAN bus is a robust vehicle bus standard designed to allow devices data communication with many applications in automotive fields. Specifically, CAN is a message-based protocol developed in 1983 at Robert Bosch GmbH and officially released in 1986. CAN 2.0, published in 1991, is the last released version of the CAN specification. The CAN bus standard is used in almost all vehicles and many machines due to its simplicity and low cost, because it is fully centralized, extremely robust, and efficient.

Some of the most common standards include:

- *SAE J1939* for heavy-duty vehicles (e.g. trucks and buses)
- *OBD2* (On-board diagnostics), which is a self-diagnostic and reporting capability used to identify car issues.
- *CANopen*, that is used in embedded control application.

- *CAN FD*, an extension of classical CAN bus with flexible data-rate.

Since this thesis concerns data extracted from heavy-duty vehicles, J1939 protocol is considered in the following. This protocol is used for all kinds of mobile industrial vehicles, including emergency vehicles, trucks, dozers, buses, cranes, etc. It is also used in mining, agriculture, forestry, generators, oil and gas, ships, military, and any diesel engine. More specifically, J1939 runs over CAN Bus, its physical "medium", as a standardized communications “language”.

J1939 speeds are typically 250 kbps or 500 kbps and it is built on CAN2.0B, which is an extended 29-bit message identifier [1]. An SAE J1939 message typically consists of a minimum of 93 bits, and it is composed of an identifier and its associated parameters, all coded into a hexadecimal format. The first part of the signal (29 bits) is the "message", is used to identify the source, whereas the remaining part (64 bits) represents the signal associated with the message, 8-byte data parameters. The identifier contains:

- A *Parameter Group Number (PGN)*, a unique ID to identify the message function and the related data parameters. It is composed of three different parts: the first 3 bits indicate the message priority, the following 18 bits represent the PGN, while the remaining 8 bits are related to the source address.

- A *Suspect Parameter Number (SPN)* to completely identify the measured CAN parameter, and also to define message priority, which is inversely proportional to the SPN value. A single PGN can be associated with different SPNs.

For priority purposes, *standard* messages are CAN messages characterized by SPNs smaller than 30000, while the other, associated with higher SPNs, can be customized.

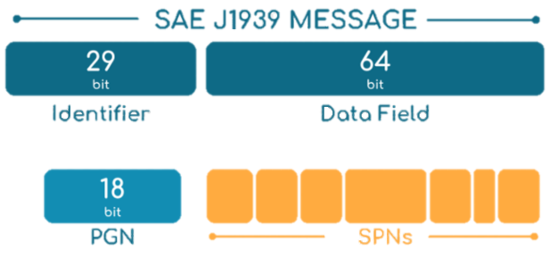


Figure 2.1: SAE J1939 Message[1]

2.1.2 CAN Bus Data Exploration

In this work, CAN bus data are collected by a *Z*55 device, an experimental data logger installed on the vehicle that is capable of uploading data on *Amazon Web Service (AWS)* storage platform. Raw data are then manipulated by Tierra and showed, with some statistics, on the company web platforms and mobile applications.

For this research, data are collected from an excavator employed in real construction sites. It is a heavy-duty vehicle expected to perform a wide range of tasks for which it is designed during the period of data collection, ranging from ignition to high workloads, as well as idle phases.

At first, *Z*55 data logger locally saves raw data, then sends them to the Amazon storage service in two different cases: when there is a shutdown of the vehicle or if the file reaches the maximum memory size of 3.6 MB. Information from raw data is decoded by a parser script developed by Tierra. During this phase, only CAN IDs with structure and number of parameters known can be decoded. The parser output consists then of two different files:

- *Parsed.txt*: it includes decoded messages for CAN IDs found in Tierra databases, enriched with additional information provided by the company.
- *notFound.txt*: it includes raw messages that cannot be decoded because of the lack of information.

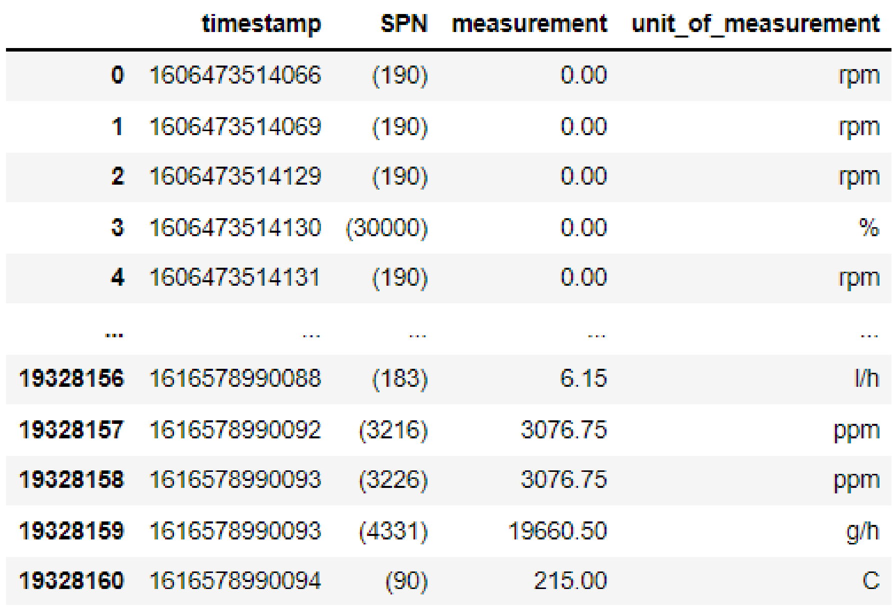
For *Parsed* files, through the combination of the decoded information with some metadata, each row of the resulting files is composed of 7 fields:

- Source address
- PGN

[1]`https://www.electriciansjournal.com/home/can-bus-j1939`

- Timestamp (UNIX format)
- Message description
- SPN
- Measured value for the considered SPN
- Unit of measurement

Data are collected from 27 November 2020 to 24 March 2021, resulting in a dataset composed of 19328161 rows, with no null and no missing values. In the first steps of processing, to limit computational costs, the subset of columns containing relevant information for the following analysis is extracted from Parsed.txt files namely, timestamp, SPN and the measured value for that SPN at that timestamp, with its unit of measurement (Figure 2.2).



	timestamp	SPN	measurement	unit_of_measurement
0	1606473514066	(190)	0.00	rpm
1	1606473514069	(190)	0.00	rpm
2	1606473514129	(190)	0.00	rpm
3	1606473514130	(30000)	0.00	%
4	1606473514131	(190)	0.00	rpm
...	...	...	...	...
19328156	1616578990088	(183)	6.15	l/h
19328157	1616578990092	(3216)	3076.75	ppm
19328158	1616578990093	(3226)	3076.75	ppm
19328159	1616578990093	(4331)	19660.50	g/h
19328160	1616578990094	(90)	215.00	C

Figure 2.2: Dataset under analysis

The resulting dataset is composed of 34 SPNs, which are the subset of CAN messages IDs sent by the vehicle which are registered in Tierra databases. However, the number of parameters is significantly higher, since additional data about the remaining SPNs is contained in *notFound.txt* files. However, since Tierra does not have sufficient information to properly decode them, they cannot be used for further analysis.

The list of the analyzed SPNs and the corresponding description and feasible range, can be found in Table 2.1

SPN	**SPN description**	**Feasible Range**
80	Washer Fluid Level	0 to 100%
81	Engine Diesel Particulate Filter Inlet Pressure	0 to 125 kPa
90	Power Takeoff Oil Temperature	−40 to 210 deg C
94	Engine Fuel Delivery Pressure	0 to 1000 kPa
108	Barometric Pressure	0 to 1000 kPa
110	Engine Coolant Temperature	−40 to 210 deg C
114	Net Battery Current	−125 to 125 A
164	Engine Injection Control Pressure	0 to 251 MPa
183	Engine Fuel Rate	0 to 3212.75 L/h
190	Engine Speed	0 to 8031.875 rpm
247	Engine Total Hours of Operation	0 to $+\infty$ h
441	Auxiliary Temperature 1	−40 to 210 deg C
514	Nominal Friction - Percent Torque	−125 to 125%
1127	Engine Turbocharger 1 Boost Pressure	0 to 8031.875 kPa
1380	Engine Oil Level Remote Reservoir	0 to 100%
1761	Aftertreatment 1 SCR Catalyst Tank Level	0 to 100%
2433	Engin Exhaust Gas Temperature Right Manifold	−273 to 1735 deg C
2809	Engine Air Filter 2 Differential Pressure	0 to 12.5 kPa
3216	Aftertreatment 1 Intake NOx	−200 to 3012.75 ppm
3485	Aftertreatment 1 Supply Air Pressure	0 to 6425.5 kPa
3509	Sensor supply voltage 1	0 to 3212.75 V
3515	Aftertreatment 1 SCR Catalyst Reagent Temperature 2	−40 to 210 deg C
3609	Diesel Particulate Filter Intake Pressure 1	0 to 6425.5 kPa
3830	Aftertreatment 1 Secondary Air Differential Pressure	−250 to 251.99 kPa
4077	Aftertreatment 1 Fuel Pressure 2	0 to 6425.5 kPa
4331	Aftertreatment 1 SCR Actual Dosing Reagent Quantity (instantaneous)	0 to 19276.5 g/h
4335	Aftertreatment 1 SCR Dosing Air Assist Absolute Pressure	0 to 2000 kPa
4360	Aftertreatment 1 SCR Catalyst Intake Gas Temperature	−273 to 1735 deg C
4374	Aftertreatment 1 SCR Catalyst Reagent Pump Motor Speed	0 to 32127.5 rpm
30000	Engine Load	0 to 250%

Table 2.1: SPN description

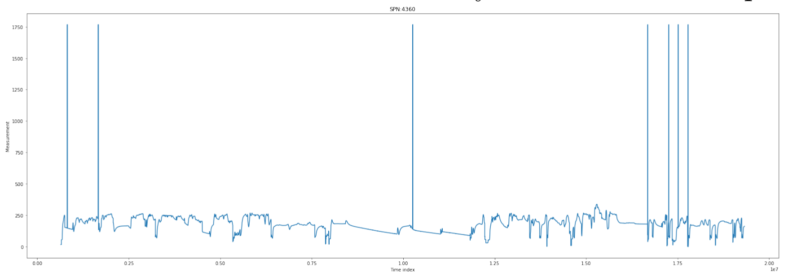
As preliminary step of data cleaning, an analysis of duplicated values is performed. Because of CAN network nature or device's and transmitting failures, it might happen both that the same identical message is repeated several times, both that for a given SPN and fixed timestamp, different values are measured. Table 2.2 shows the list of SPNs with repetitions and the corresponding percentage of duplicate rows. The variable not in table, does not show duplicated values. As can be noticed, repeated values are only a small portion of available data, hence for each pair (SPN, timestamp) the percentage of duplicates is computed. Once duplicated values are detected, the first value sent is kept, since the second could be related to transmission errors of malfunctioning of the data collecting device.

SPN	Percentage of duplicates
90	0.0009%
183	0.0006%
190	0.11%
514	0.0002%
3216	0.0003%
3226	0.0002%
4331	0.0005%
30000	0.0007%

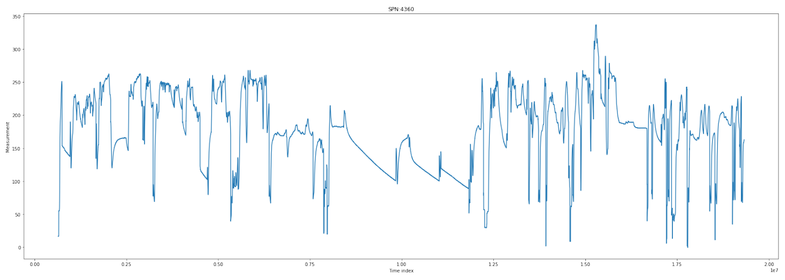
Table 2.2: Percentage of duplicates per SPN where repetitions are present

Feasible ranges are an important information for data cleaning purpose, since SPNs signals are particluarly noisy, both because of decoding and transmitting errors. If wrong or unexpected values are identified, they can be removed if completely misleading or replaced by estimated values. Comparing feasible ranges with the data under analysis, it is possible to notice that the Aftertreatment 1 SCR Catalyst Intake Gas Temperature (SPN 4360), shown in Figure 2.3-(a), takes some unfeasible values, in correspondence of the signal peaks. Since they are more likely to be due to errors, they are removed. The resulting signal is shown in Figure 2.3-(b).

SPN 4360: "Aftertreatment 1 SCR Catalyst Intake Gas Temperature"



(a)



(b)

Figure 2.3: Time Series plot: SPN 4360 with unfeasible values (a), and after data cleaning (b)

Data, as shown in Figure 2.4, is not heterogeneously distributed over all working days from November to March. This behaviour is mainly due to the highly variable time of work per day typically associated with this type of vehicle.

The CAN bus standard specifies a maximum signaling rate of 1 Mbps and, depending on the specific version and its characteristics, the CAN messages can be transmitted at different rates [2]. Because of non constant sampling rates, SPN observations will be not equally spaced in time. Figure 2.5 shows the histogram plots of *interarrivals*, i.e. the elapsed time between two consecutive messages for a given SPN. Not only CAN messages regarding the same SPN will be sent at different rates, but it also changes depending on different SPNs. This will result in some signals with more observations, whose average sampling rate will be higher than others, as demonstrated in Figure 2.6. Exploiting the representation of interarrivals distribution, all the signals are regularly sampled, as indicated from the single peak in the graph, saying that very often, the time passed from one observation to another was the same (Figure 2.5).

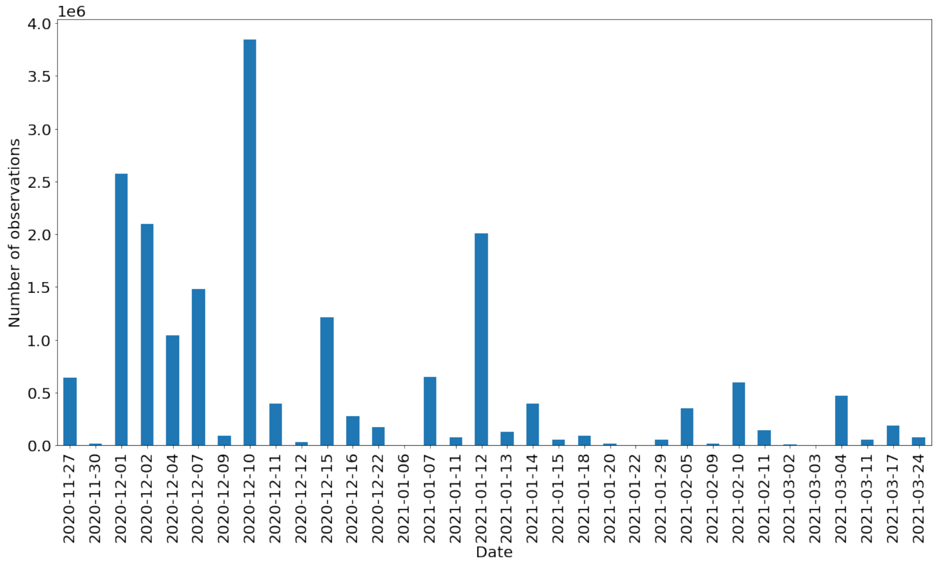


Figure 2.4: Number of observations per date

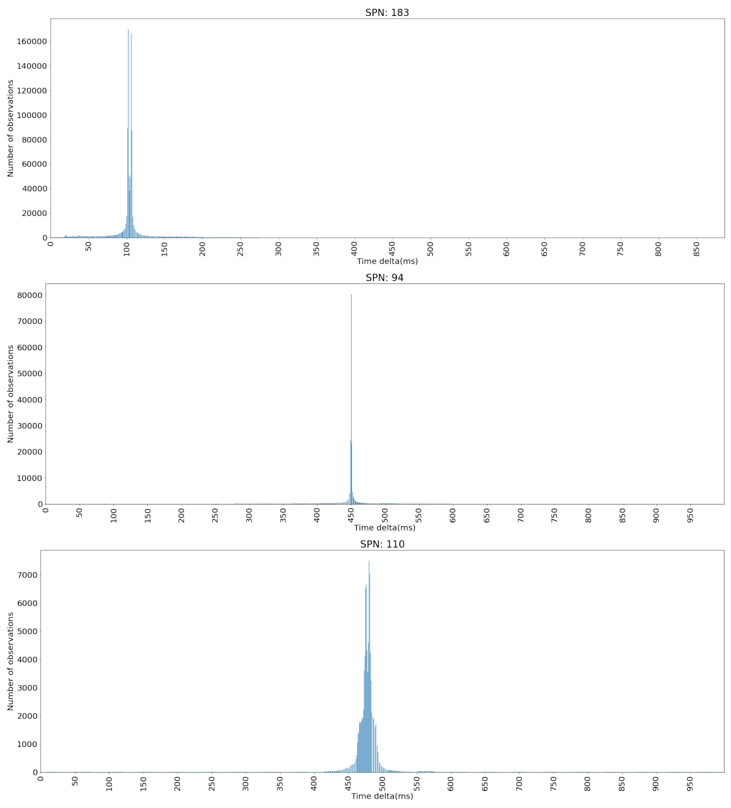


Figure 2.5: Interarrivals distributions for SPNs 183, 94 and 110

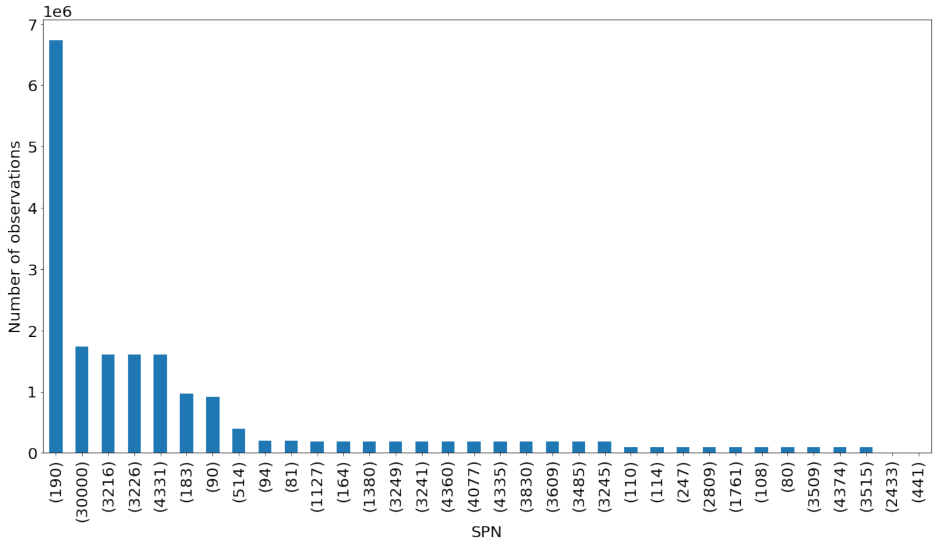


Figure 2.6: Number of observations per SPN

Following the study on interarrivals, with all the distributions with an approximable centered shape, and stated that the mode of the distribution corresponds to its mean value, it is possible to find sampling rates for all the considered signals. For centered distributions, the sampling rate is approximable to the inverse of the mode value. The found modes and therefore the sampling rates of the signals are resumed in Table 2.3.

SPN	**Mode**	**Approximate Sampling Rate**
94	0.51 sec	1.96 Hz
108	1 sec	1 Hz
110	1 sec	1 Hz
183	0.1 sec	10 Hz
190	0.02 sec	50 Hz
247	0.51 sec	1.96 Hz
514	0.237 sec	4.2 Hz
1127	0.51 sec	1.96 Hz
1761	1 sec	1 Hz
3509	1 sec	1 Hz
3609	0.51 sec	1.96 Hz
4360	0.51 sec	1.96 Hz
30000	0.06 sec	17 Hz

Table 2.3: Modes and sampling rates approximations per SPN

Since most of the algorithms commonly applied in multivariate time series context require synchronized and equally spaced in time observations, a strategy to align SPNs series will presented in Chapter 3.

From a preliminary visualization of SPNs series variations over time, it is possible to identify 21 constant signals. These variables are reported to domain experts for future investigations and not considered in further analysis since it is not ensured that they are characterized by constant behaviour.

As last step of data exploration, a procedure for detecting working cycles is presented. It is relevant for fleet management and maintenance issues. Because of the nature of CAN network and data logger specification, CAN messages are collected only when the vehicle is on, but information regarding ignition and shutdowns are not available. It is then studied the interarrivals distribution between subsequent observations of the same variable, to detect the sampling rate for each signal and the working cycles.

However, sufficiently long interarrivals can suggest the shutdown of the vehicle. For the sake of simplicity, working cycles are inferred basing on the SPNs sampled "more regularly", SPN 94 (Figure 2.5): *"Engine Fuel Delivery Pressure"*, as it can be seen from its interarrivals distribution. Then, to establish when the vehicle is turned on after a shutdown, a threshold value equal to the 99.97*th* percentile is set on interarrivals. The final result is the detection of 54 working cycles, of variable duration as can be deducted in Figure 2.7.

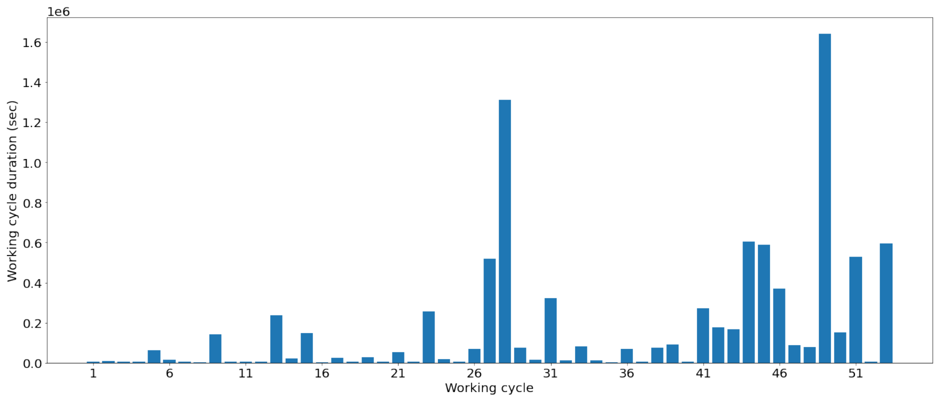


Figure 2.7: Working cycles duration

The proposed procedure, can be validated by other information, such as SPN 110, *Engine Coolant Temperature*: indeed, its values typically reach lower temperatures when the vehicle is restarted after a shutdown since the cooling fluid had time, in the meanwhile, to cool down. This kind of behaviour is highlighted in Figure 2.8, where it is clearly visible that the working cycles start in correspondence with the decrease of the temperature.

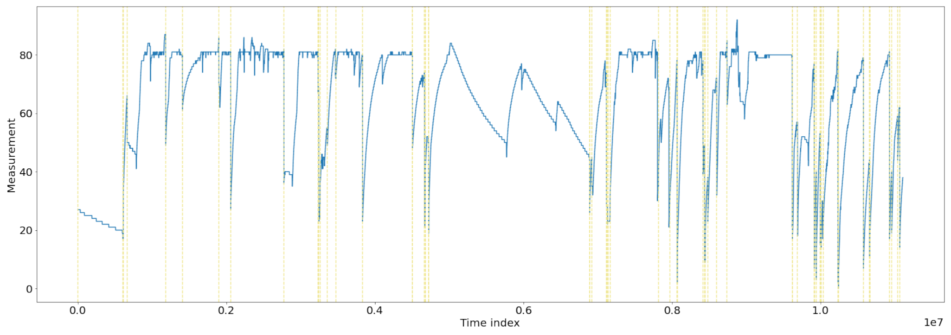


Figure 2.8: Relation between SPN 110 and working cycles

In the case of the vehicle under analysis, the identified working cycles can be verified thanks to additional information collected by a second device installed on the vehicle. Device $AM53$ is a marketed product developer by Topcon for tracking and data acquisition, often used in automotive applications, including heavy-duty vehicles. This device is able to collect also information regarding actual vehicle ignitions and shutdowns, that can be compared with the ones obtained with the proposed procedure. Crossing the working cycle timestamps found with the over mentioned approach with the information coming from this device, the results of the procedure are validated.

2.2 Signal Processing

Since each SPN describes a physical quantity over time and can be interpreted as a univariate signal, in this chapter some basic notions about signal processing are introduced to the purpose of applying them to the real case data under analysis.

2.2.1 Signal Processing Fundamentals

(*Theoretical concepts of this chapter are borrowed from [3]*).

A signal is a variable physical quantity to which some form of information is associated. This information could be of different nature: the light intensity and color on a screen in the case of a television signal, the electrical voltage or the current for a signal measured on an electric circuit. The evolution of many monodimensional signals, which depend on only one variable, relies on time.

Another distinction of signals can be done according to the values of the independent variable.

- *Continuous-time signals*, where the independent variable takes values in the set of real numbers.

- *Discrete-time signals*, for which the domain of the function has the cardinality of natural numbers.

A similar classification can be done based on values assumed by signals.

- *Signals of continuous amplitude*, that can continuously assume real values of an interval (eventually unlimited).
- *Signals of discrete amplitude*, having a countable set as codomain.

Continuous-Time signals and of continuous amplitude are said to be *analog signals*, while the ones with discrete-time and amplitude are *digital signals.* Furthermore, a signal is said to be a *samples sequence* if it is discrete-time but real-valued, while it is *quantized* if discrete-valued and defined at each point in time.

For each analog signal $x(t)$, it is possible to define the associated *energy* to it as:

$$E_x \triangleq \int_{-\infty}^{+\infty} |x(t)|^2 dt$$

The integral result is convergent for all the physical signals, since each signal describing a physical system is a carrier of finite energy. Alternatively, finite power signals can be considered: they are ideal signals associated with unlimited energy, not physical and not existing in nature, for which the power is defined as:

$$P_x \triangleq \lim_{T \to +\infty} \frac{1}{T} \int_{-\frac{T}{2}}^{+\frac{T}{2}} |x(t)|^2 dt$$

For discrete-time signals, energy and power are defined in the following way:

$$E_x \triangleq \sum_{n=-\infty}^{+\infty} |x(n)|^2$$

$$P_x \triangleq \lim_{N \to +\infty} \frac{1}{2N+1} \sum_{n=-N}^{+N} |x(n)|^2$$

Moreover, the main operations that could be done on signals are the following:

- *Translation of* t_0: it consists of shifting the axis of the indipendent variable of t_0, that is:

  $$T_{t_0}(x(t)) = x(t - t_0)$$

  If the independent variable is the time, and if $t_0 > 0$, then the signal is said to be *delayed* of t_0, anticipated otherwise.
- *Overturning*: It consists of reflecting the signal with respect to the ordinate axis, that is:

  $$O(x(t)) = x(-t)$$
- *Axis scaling*: Given a real number $\alpha > 0$, the operation of axis scaling by α is:

  $$S_\alpha(x(t)) = x(\alpha t)$$

  a change of scale reflects the following transformation: $x(t) \to x(at)$.

  If $\alpha > 1$, the signal has been shrunk, while is $0 < \alpha < 1$, the signal has been expanded.

- *Convolution between signals*: Given two analog signals $x(t)$ and $h(t)$, the convolution product between two signals is defined as:

  $$y(t) = x(t) \circledast h(t) = \int_{-\infty}^{+\infty} x(\tau)h(t-\tau)d\tau$$

  The convolution $\circledast$ is a commutative operation, enjoys associative property, and is distributive with respect to the sum operation.

From now on, only digital signals, will be considered, because applications of this thesis work.

2.2.2 Stochastic Processes

Stochastic processes are considered one of the most widely used object for mathematically modelling systems and physical phenomena. In this chapter, basic notions are provided. (*The concepts of this section are borrowed from [3]*)

Definition 2.2.1 *Taking values in a measurable space S, a stochastic process is a collection of S-valued random variables, that can be written as:*

$$\{X(t) : t \in T\}$$

A discrete-time stochastic process X is a countably infinite collection of jointly distributed random variables $\{..., x_0, x_1, x_2, ...\}$. When the index represents time, a stochastic process is often called *time series.*

Due to its inherent randomness, differently from a *deterministic process*, it is not possible to characterize a stochastic process in terms of evolution over time.

The definition of the principal characteristics of a statistical process, without the full knowledge of it, is allowed through the use of some statistical functions [3].

Definition 2.2.2 *The mean function of a univariate time series $X(t)$ such that $E[X(t)^2] < \infty$, is defined as:*

$$\mu_X(t) = E[X(t)]$$

Definition 2.2.3 *The variance function of a univariate time series $X(t)$ is defined as:*

$$\sigma_X^2(t) = E[(X(t) - \mu_X(t))^2]$$

The standard deviation function is defined as:

$$\sigma_X(t) = \sqrt{\sigma_X^2(t)}$$

Definition 2.2.4 *The covariance function of a univariate time series $X(t)$ is defined as:*

$$\gamma_X(t,s) = Cov(X(t), X(s)) = E[(X(t) - \mu_X(t))(X(s) - \mu_X(s))] \quad \forall t, s \in \mathbb{R}$$

Definition 2.2.5 *The correlation of a univariate time series $X(t)$ is defined as:*

$$\rho(t,s) = \frac{Cov(X(t),X(s))}{\sigma_X(t)\sigma_X(s)}$$

Definition 2.2.6 *A stochastic process $X(t) \in \mathbb{Z}$ is said to be stationary if*

- $E[X(t)^2] < \infty \quad \forall t \in \mathbb{Z}$
- $E[X(t)] = \mu \quad \forall t \in \mathbb{Z}$
- $\gamma_X(s,t) = \gamma_X(s+h, t+h) \quad \forall s,t,h \in \mathbb{Z}$

Specifically, a process $X(t)$ is said to be *strictly stationary* if the joint distribution of $(X_{t_1}, X_{t_2}, ..., X_{t_k})$ is the same as that of $(X_{t_1+h}, X_{t_2+h}, ..., X_{t_k+h})$, for all h, while a process is *weakly stationary* if $\mu_X(t)$ is independent on t and $\gamma_X(t+h,t)$ is independent on t for each h.

Given a process X and a lag h, *autocovariance function* γ_X and autocorrelation function ρ_X, related to the process periodicity over time are respectively defined as:

$$\gamma_X(h) = \gamma_X(h{,}0) = \gamma_X(t+h,t) \quad \forall t \in T$$

and

$$\rho_X(h) = \frac{\gamma_X(h)}{\gamma_X(0)} \quad \forall t \in T$$

Finally, there could be a statistical dependence between consecutive lags. Given a temporal lag $h > 1$, the correlation between two different observations could be influenced by correlation of them with intermediate values. Indeed, the time series tends to carry information from previous observations. *Partial autocorrelation function* is a measure of correlation between the series $X(t)$ and its lagged version $X(t+h)$ without taking into account the correlation with the middle values $(X_{t+1}, ..., X_{t+h+-1})$. At lag h, it is defined as:

$$PACF(h) = \frac{E[(X_t - \mu)(X_{t+h} - \mu)|x_{t+1}, ..., x_{t+h-1}]}{E[(X_t - \mu)^2|x_{t+1}, ..., x_{t+h-1}]} \quad \forall t \in \mathbb{R}$$

2.2.3 Univariate Time Series Analysis

Univariate signals, time-varying quantities representing physical quantities, can be easily represented by *time series.* Given the preliminary feature selection performed in previous sections, in the following the selected 13 signals are considered.

As first step of univariate time series analysis, SPNs series plots over time are fundamental to proceed, since they highlight trends and cyclical variations, as well as the presence of outliers or other inconsistencies. A time series plot of the most representative SPNs of each kind is showed, followed by a study of correlation and partial correlation, useful tools to discover systematic patterns in time series, and inspect stationarity.

The three main internal characteristics that can be used to describe systematic patterns in time series are:

- *Seasonality*, that represents periodic and repetitive variations;
- *Trend*, that can be defined with a downward or an upward movement;
- *Cyclical changes*, that are repetitive movements not characterized by a fixed period. Typically, this component is considered as part of trend.

In trend analysis, the objective is the estimation of a monotonous component, in the long run. Autocorrelation function will be an important tool to discover seasonality behaviours in signals. Indeed, the correlation dependency at lag k defines seasonality of order k. For this reason, correlogram and partial autocorrelation plot, can be useful for further analysis about it [4].

Engine Total Hours of Operation (SPN 247), showed in Figure 2.9, Aftertreatment 1 SCR Catalyst Tank Level (SPN 1761), and Barometric Pressure (SPN 108) represent signals clearly showing trends, due to their specific nature. For example, SPN 247 shows a clear upward trend, indeed the total hours of operations of the engine cannot diminish if not manually set to zero, as it probably happened almost at the beginning of data collection. Because of its linearity trend, it is expected that the autocorrelation will be significant for multiple consecutive lags (Figure 2.10).

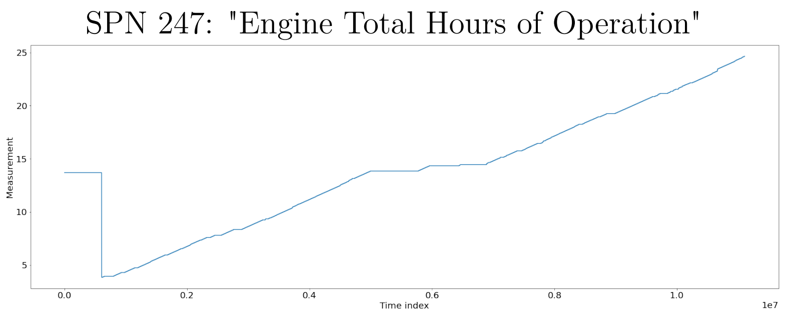


Figure 2.9: Time Series plot: SPN 247

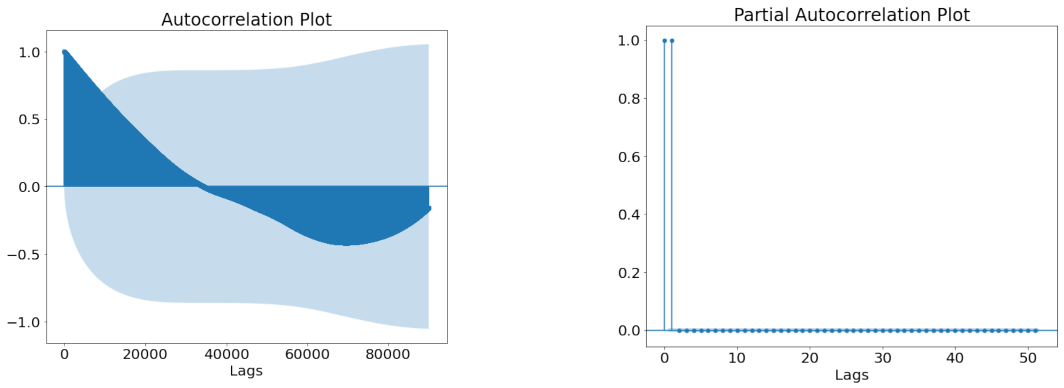


Figure 2.10: SPN 247: Correlogram and Partial AutoCorrelation

Categorical variables, such as the Sensor supply voltage 1 (SPN 3509), show different behaviours. As it can be seen from Figure 2.12, there is no significant correlation even for smaller lags.

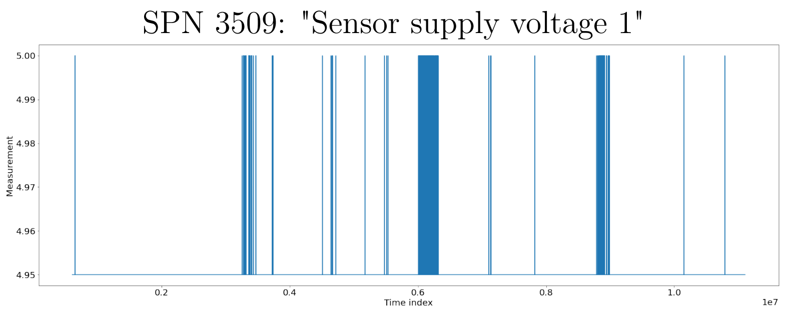


Figure 2.11: Time Series plot: SPN 3509

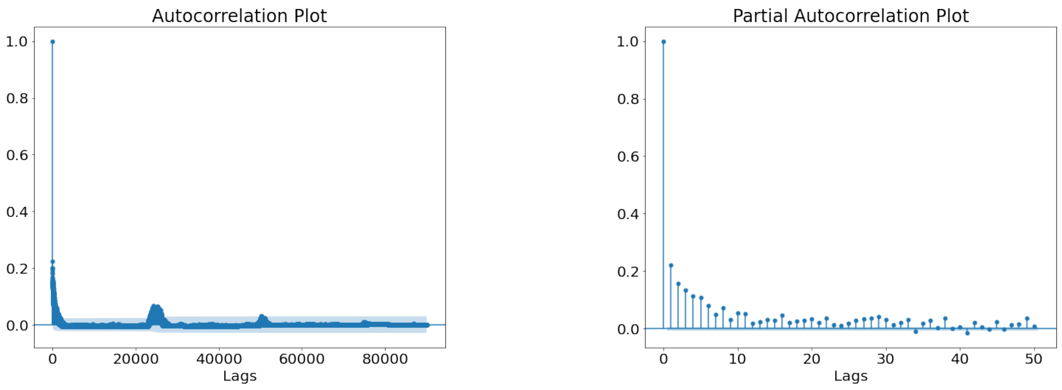


Figure 2.12: SPN 3509: Correlogram and Partial AutoCorrelation

The remaining SPNs, represented by Engine Speed shown in Figure 2.13, are characterized by a cyclical behaviour. These measures are obviously related to working cycles, since most of the time in one working cycle, the vehicle is expected to repeat the same basic operations, reflected in these signals. In this case, the correlogram shows a positive correlation for a significant number of lags (Figure 2.14).

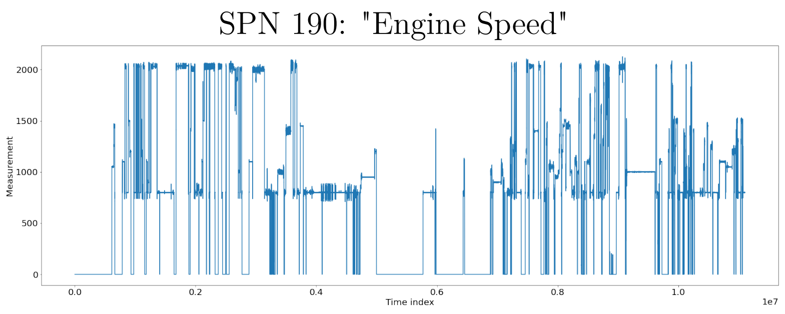


Figure 2.13: Time Series plot: SPN 190

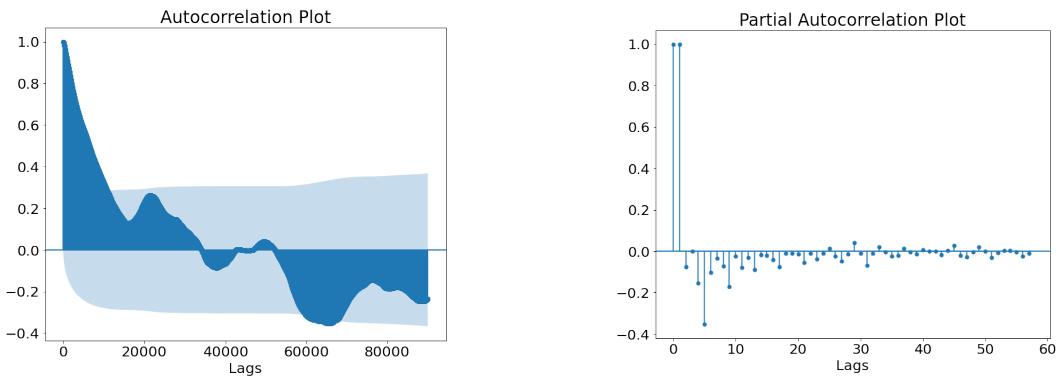
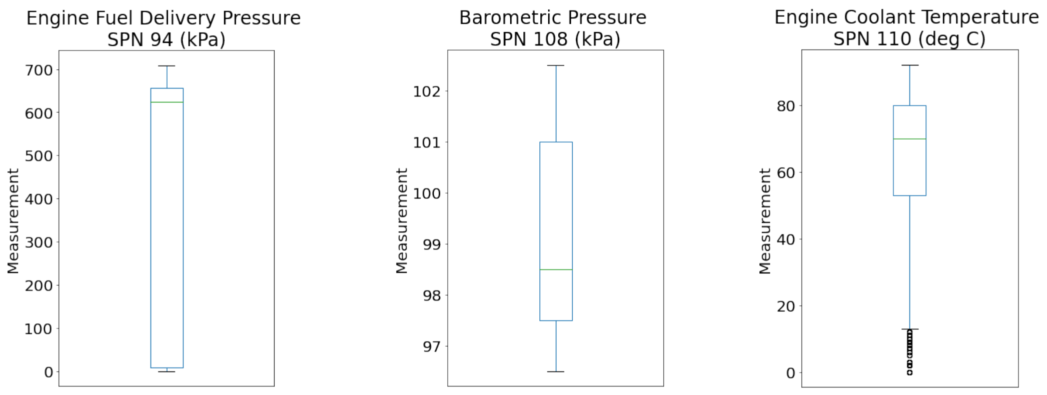
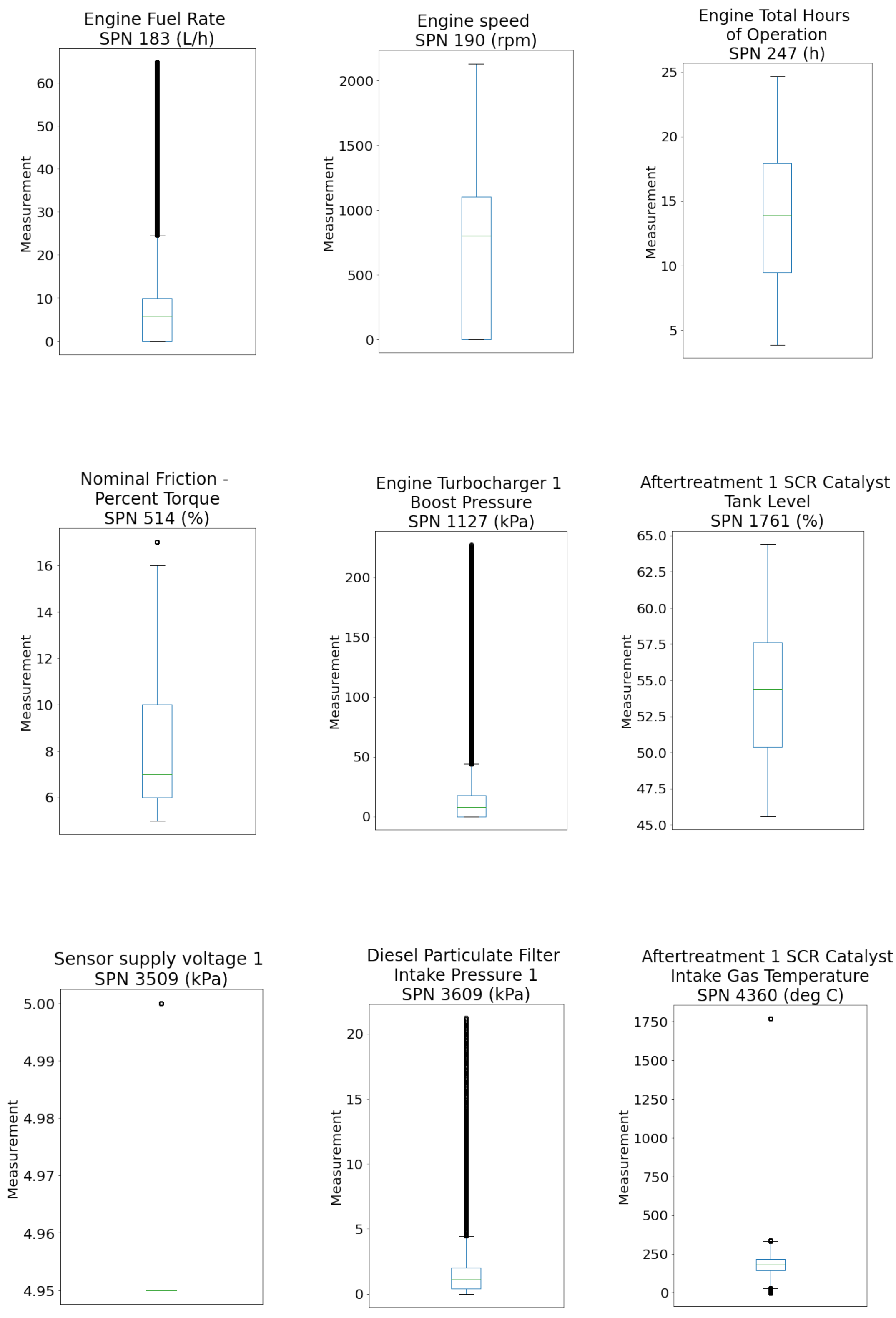


Figure 2.14: SPN 190: Correlogram and Partial AutoCorrelation

As conclusion of univariate analysis, a box plot representation for each SPN is presented. Box plot is a standardized, non-parametric way for displaying a five-number summary: the *minimum* and the *maximum*, lowest and higher data point for the considered measure, excluding *outliers*, the measure *median* and *first* and *third quartile*, respectively intended as the median of the lower and of the higher half of the dataset. In Figure 2.15, box plot representations of informative variables on which the final analysis is performed are shown.




Engine Fuel Rate
SPN 183 (L/h)
Engine speed
SPN 190 (rpm)
Engine Total Hours
of Operation
SPN 247 (h)
Nominal Friction -
Percent Torque
SPN 514 (%)
Engine Turbocharger 1
Boost Pressure
SPN 1127 (kPa)
Aftertreatment 1 SCR Catalyst
Tank Level
SPN 1761 (%)
Sensor supply voltage 1
SPN 3509 (kPa)
Diesel Particulate Filter
Intake Pressure 1
SPN 3609 (kPa)
Aftertreatment 1 SCR Catalyst
Intake Gas Temperature
SPN 4360 (deg C)
Measurement

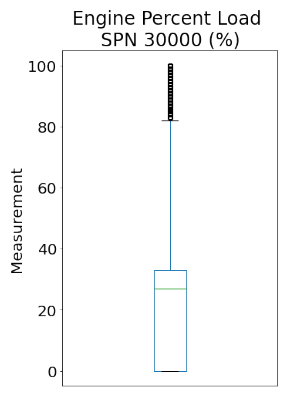


Figure 2.15: Box plot of SPNs: 94, 108, 110, 183, 190, 247, 514, 1127, 1761, 3509, 3609, 4360, 30000

As it can be seen in Figure 2.15, some of the variables are characterized by outliers, samples showing significant deviation from the rest of data. Their presence can lead to inflated error rates or substantial distortion of parameter and statistic estimates [5]. *Outliers Detection* is a process useful to detect these data. They can be due to different reasons, such as human errors in collecting and recording, or they can be the result of from intentional or motivated misreporting, sampling error, incorrect assumptions... [6] These values can be detected with many different approaches.

One of these, make use of box plots.Box plot method can be useful to identify them, as the values which fall outside the interval:

$$[Q_1 - 1.5 * IQR, Q_3 + 1.5 * IQR]$$

where IQR is the Inter-Quartile range, calculated as the difference between the third and first quartile, respectively denoted as $Q3$ and $Q1$.

Chapter 3

Multivariate Sequences Processing and Analysis

In this chapter, a deeper analysis of signals is performed by studying their interaction considering them as components of a multivariate time series.

To this purpose signals are aligned and synchronized in time to allow Pearson correlation and cross-correlation analysis.

Finally, drawing conclusions from the described analysis, feature selection is performed.

3.1 Theoretical Introduction

In Sections 3.1.2, and 3.1.3, some theoretical concepts due to constitute the basis for aligning and synchronizing signals are introduced.

3.1.1 Discrete-Time Systems

(*The concepts of this section are borrowed from [7]*)

A physical system is an operator that takes one or more inputs and reacts by producing one or more outputs. Graphically it is represented as a rectangle (sometimes called *black box*), with two oriented branches, one entering, and one exiting the system (Figure 3.1). The relation between these branches is typically indicated inside the box.

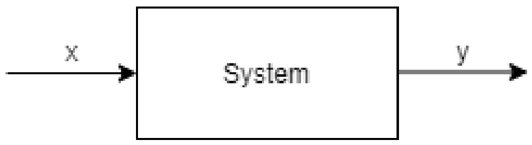


Figure 3.1: Graphical representation of a system [7]

Discrete-time systems are operators having discrete-time signals as their inputs and outputs. Specifically, it is an operator S, that maps an input sequence $x \in V$ into an output sequence $y \in V$.

$$y = S(x)$$

Definition 3.1.1 *A discrete-time system S is linear if, for each inputs x and y and each $\alpha, \beta \in \mathbb{C}$,*

$$S(\alpha x + \beta y) = \alpha S(x) + \beta S(y)$$

Sometimes it will be useful to use a matrix representation for a linear system, especially when matrix structure reveals characteristics of the system. A linear operator has a unique matrix representation once bases have been chosen for the domain and the codomain of the operator.

Definition 3.1.2 *A discrete-time system S is memoryless if, for any k, integer value and inputs x and x',*

$$1_{\{k\}}x = 1_{\{k\}}x' \implies 1_{\{k\}}S(x) = 1_{\{k\}}S(x')$$

where it is used the domain restriction operator defined as:

$$1_{\{k\}}x = \begin{cases} x_n, & \textit{for } n \in \mathbb{N}; \\ 0, & \textit{otherwise} \end{cases}$$

The matrix of a memoryless system will be diagonal.

Definition 3.1.3 *A discrete-time system S is called causal when, for inputs x and x', and integer k*

$$1_{\{-\infty,\dots,k\}}x = 1_{\{-\infty,\dots,k\}}x' \implies 1_{\{-\infty,\dots,k\}}S(x) = 1_{\{-\infty,\dots,k\}}S(x')$$

Lower-triangular matrix representation of a system will indicate a linear and causal system.

Definition 3.1.4 *A discrete-time system S is said to be Bounded-Input Bounded-Output stable, or BIBO stable, when given a bounded input x, the system produces a bounded output $y = S(x)$ such that:*

$$sup_i|x_i| < \infty \implies sup_i|y_i| < \infty$$

Definition 3.1.5 *A discrete-time system S is said to be shift-invariant if, for any integer value k and input signal x,*

$$y = S(x) \implies y' = S(x'), \quad \textit{where } x'_n = x_{n-k} \textit{ and } y'_n = y_{n-k}$$

Definition 3.1.6 *A discrete-time system S is called periodically shift-varying of order (L,M) when, for any integer k and input x,*

$$y = S(x) \implies y' = S(x'), \quad \textit{where } x'_n = x_{n-Lk} \textit{ and } y'_n = y_{n-Mk}.$$

3.1.2 Linear Shift-Invariant Systems

(*Theorems and definitions of this chapter are borrowed from [7]*)

Linear Time-Invariant (LTI) or Linear Shift-Invariant (LSI) systems are desirable for their mathematical properties.

Definition 3.1.7 *A sequence h is the impulse response of LSI discrete-time system S when, given the Kronecker delta sequence δ as input, it produces an output h.*

$$h = S\delta$$

where

$$\delta_k = \begin{cases} 1, & \textit{if } k = 0 \\ 0, & \textit{if } k \neq 0 \end{cases}$$

The causal linear system impulse response h always satisfies $h_n = 0$, for all $n > 0$.

Considering a LSI system S, given an input x, it can be written as:

$$x_n = \sum_{k \in \mathbb{Z}} x_k \delta_{n-k} \quad \forall n \in \mathbb{Z}$$

Then, the output obtained applying S to x can be expressed as:

$$y = Sx = \sum_{k \in \mathbb{Z}} x_k S \delta_{n-k} = \sum_{k \in \mathbb{Z}} x_k h_{n-k} = h \circledast x.$$

The impulse response of a system is often called *filter*, and the convolution operation with the impulse response is called *filtering*. Some basic classes of filters are described in the following:

- *Causal filters*, such that for all $n < 0$, $h_n = 0$.
- *Anti-causal filters*, such that for all $n > 0$, $h_n = 0$.
- *Two-sided filters*, neither causal nor anti-causal.
- *Finite Impulse Response (FIR) filters*, filters having only a finite number of coefficients $h_n \neq 0$.
- *Infinite Impulse Response (IIR) filters*, having infinitely many nonzero terms.

Theorem 3.1.1 *An LSI system is Bounded Input Bounded Output (BIBO)-stable if and only if it is characterized by an impulse response that is absolutely summable.*

3.1.3 Discrete-Time Fourier Transform

(*Concepts of this chapter are borrowed from [7]*)

Fourier's methods play a prominent role in sequences analysis and discrete-time systems [7]. Indeed, as previously introduced, the computation of the response signal of a *LTI* system requires a convolution operation. However, thanks to Fourier algorithms optimization, it is a commonly adopted strategy to map the signal from time to frequency domain by applying the so-called Fast Fourier Transform algorithm to simplify the convolution computation.

Definition 3.1.8 *The Discrete-Time Fourier Transform (DTFT) of a sequence x is*

$$X(e^{j\omega}) = \sum_{n\in\mathbb{Z}} x_n e^{-j\omega n}, \quad \omega \in \mathbb{R}$$

If this expression converges for all $\omega \in \mathbb{R}$, the DTFT is well defined.

Definition 3.1.9 *The inverse DTFT of a $2\pi-$periodic function $X(e^{j\omega})$ is*

$$x_n = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\omega})e^{j\omega n}d\omega, \quad n \in \mathbb{Z}$$

If the Discrete-Time Fourier Transform exists, we denote the DTFT pair as

$$x_n \xleftrightarrow{\text{DTFT}} X(e^{j\omega})$$

The DTFT is always a 2π-periodic function, since $e^{-j\omega n}$ is a $2\pi-$periodic function of ω for every $n \in \mathbb{Z}$.

The existence of the DTFT is strongly dependent on the sequence x and on its convergence. This immediately implies the existence of the DTFT for all sequences in $l^1(\mathbb{Z})$.

For series not in $l^1(\mathbb{Z})$, the convergence of x is not directly ensured. For extension beyond $l^1(\mathbb{Z})$, a limiting process and a sense of convergence must be specified.

Considering the partial sums,

$$X_N(e^{j\omega}) = \sum_{n=-N}^{N} x_n e^{-j\omega n}, \quad N = 0,1,...$$

if the following limit exists, the discrete-time Fourier transform is defined as:

$$X(e^{j\omega}) = \lim_{N\to+\infty} X_N(e^{j\omega})$$

Hence, if it is assumed the DTFT to exist whenever the sequence of partial sums converges under the $L^2([-\pi,\pi))$ norm, then it exists for all sequences in $l^2(\mathbb{Z})$.

In the following, some basic properties of the DTFT, are shown.

- **Linearity**: The DTFT operator is a linear operator:

$$\alpha x_n + \beta y_n \xleftrightarrow{\text{DTFT}} \alpha X(e^{j\omega}) + \beta Y(e^{j\omega})).$$

- **Shift in time**: The DTFT pair which corresponds to a shift in time by n_0 is

$$x_{n-n_0} \xleftrightarrow{\text{DTFT}} e^{-j\omega n_0}X(e^{j\omega}).$$

- **Scaling in time**

  (i) *Downsampling*: The DTFT pair corresponding to scaling in time by N is

$$x_{N_n} \xleftrightarrow{\text{DTFT}} \frac{1}{N}\sum_{k=0}^{N-1} X(e^{\frac{j(\omega-2\pi k)}{N}})$$

.

(ii) *Upsampling*: The DTFT pair which correspond to scaling in time by $\frac{1}{N}$ is

$$\begin{cases} x_{\frac{n}{N}}, & \text{for } \frac{n}{N} \in \mathbb{Z}; \\ 0, & \text{otherwise} \end{cases}$$

The Discrete-Time Fourier Transform of a filter h is called the *frequency response*:

$$H(e^{j\omega}) = \sum_{n\in\mathbb{Z}} h_n e^{-j\omega n}, \quad \omega \in \mathbb{R}.$$

On the other hand, the inverse DTFT of the frequency response recovers the impulse response,

$$h_n = \frac{1}{2\pi}\int_{-\pi}^{\pi} H(e^{j\omega})e^{j\omega n}d\omega, \quad n \in \mathbb{Z}$$

The magnitude and the phase of the frequency response can be also written separately:

$$H(e^{j\omega}) = |H(e^{j\omega})|e^{j\arg(H(e^{j\omega}))}$$

where the *magnitude response* is a $2\pi-$periodic real-valued, nonnegative function, and the *phase response* is a $2\pi-$periodic real-valued function taking values between $-\pi$ and π.

The frequency response of a filter is typically used to design a filter that satisfies desired properties, letting some frequencies pass (the *passband*), while blocking others (the *stopband*). The magnitude response of an *ideal filter* is constant in its passband and zero outside its passband.

A system with frequency response, as in Figure 3.2, is called *ideal low-pass filter* [3]. This kind of filter leaves unchanged some frequencies f, corresponding to the passband, for which $H(f) = 1$. On the other hand, an *ideal high-pass filter*, (Figure 3.2) is a filter that blocks the passband while leaving unchanged the remaining frequencies ($H(f) = 0$).

Given the frequency B, representing the *band limit*, the frequency response of an *ideal low-pass filter* (Figure 3.2), named H_{LP}, can be defined as:

$$H_{LP}(f) = rect(\frac{f}{2B})$$

The frequency response of an *ideal high-pass filter* (Figure 3.3), named H_{HP} , is instead:

$$H_{HP}(f) = 1 - rect(\frac{f}{2B})$$

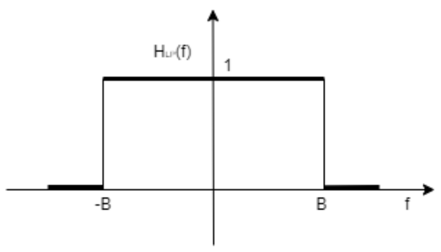


Figure 3.2: Frequency response of an ideal low-pass filter [3]

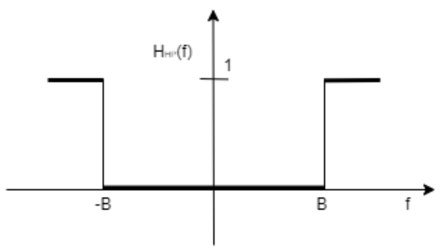


Figure 3.3: Frequency response of an ideal high-pass filter [3]

These filters are non-causal systems, and then physically not feasible. Therefore, it is also important to explore some examples of filters with realizable frequency responses, such as Finite Impulse Response (FIR) or Infinite Impulse Response (IIR) filters.

An FIR filter is characterized by finite duration impulse response, meaning that it settles to 0 in finite time. If this is not the case, it is a IIR filter.

FIR filters, unlike IIR ones,

- *require no feedback*, making implementation simpler.
- *are inherently stable*, being the sum of finite multiples of input values.
- *can easily be designed to be linear phase*, for phase-sensitive applications.

3.1.4 Multi-Rate Sequences

(*The theoretical concepts are borrowed from [7]*)

In multi-rate sequence processing, the index of different sequences may refer to different physical times [7]. The introduction of *Multirate Operations*, such as Downsampling or Upsampling, allows aligning different components along the same time axis.

The ***Upsampling*** operation corresponds to the increase of the sampling rate by an integer factor N. This is graphically resumed in Figure 3.4.

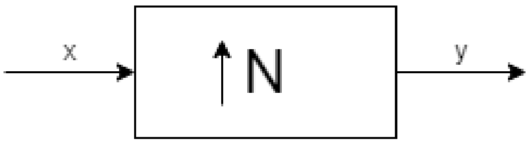


Figure 3.4: Block diagram representation of Upsampling by N [7]

Given a sequence x, the corresponding sequence upsampled by N, positive integer, is

$$y_n = \begin{cases} x_{\frac{n}{N}}, & \text{for } \frac{n}{N} \in \mathbb{Z}; \\ 0, & \textit{otherwise} \end{cases}$$

The corresponding DTFT is

$$Y(e^{j\omega}) = X(e^{jN\omega}).$$

The ***Downsampling*** operation, also called *Decimation*, corresponds to the decrease of the sampling rate by an integer factor N. This is graphically resumed in Figure 3.5.

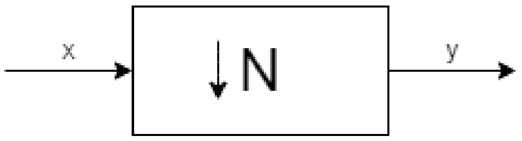


Figure 3.5: Block diagram representation of Downsampling by N [7]

Given the sequence x, a positive integer N, the downsampled-by-N sequence y is:

$$y_n = x_{Nn}$$

Since both operations are performed by scaling the rate of an integer value, upsampling and downsampling processes can be combined to obtain the desired rate.

However, to avoid undesired aliasing phenomena due to the scaling of frequencies, these operatores are combined with suitable filters. Specifically, downsampling operator is preceded by a filter, while upsampling operator is followed by an ideal low pass filter.

3.2 Multivariate Time Series Analysis

The definition of a multivariate time series needs the synchronization of the signals.

As previously seen from the time delta representation of variables (Figure 2.5), data are not transmitted at constant rates but irregularly sampled. Since upsampling and downsampling operators require equally spaced in time observations, nearest interpolation is applied to align each SPN series to its own average sampling rate. This operation is not introducing bias in the SPNs series since observations are just shifted in time by milliseconds and, at the same time, the measured quantities are not expected to have rapid and unexpected variations since they refer to physical quantities associated with heavy-duty vehicles, typically characterized by slow changes over time.

Once the signals are characterized by constant sampling rates, it is possible to proceed with signals synchronization. It can be achieved with different strategies:

- Resampling each signal to the highest average sampling rate of SPN;
- Resampling signals to the lowest average sampling rate of SPN.

The first option has the main drawback of generating extra points, potentially introducing noise and distortion [8]. On the other hand, downsampling is a cheaper operation in terms of computational costs, and moreover it is coherent with heavy-duty vehicles' data, which are expected to change gradually, in a slow way.

For the reason explained, signals are downsampled to the lowest average sampling rate of SPNs, which in this application corresponds to 1 Hz. To this purpose, depending on the average sampling rate of the considered signal, different operations are performed.

- Signals with an average sampling rate of 1 Hz are left unchanged. This is the case of SPNs 108, 110 247, 1761 and 3509.
- Signals with an average sampling rate that is an integer multiple of 1 Hz, namely SPNs 190 and 183, are downsampled with a downsampling factor equal to desired sampling rate / SPN sampling rate. As seen in theory (Section 2.2.1), the downsampling operator consists of a downsampling operation preceded by a filtering operation, with an anti-aliasing filter, that in this specific case is an FIR filter.
- Finally, there is a set of SPNs - namely SPNs 94, 514, 1127, 3609, 4360 and 30000, for which does not exist any integer downsampling factor to obtain the desired rate. In this case, it can be achieved by the sequential combination of an upsampling followed by a downsampling operator.

The set of described operations allow signals alignment necessary to treat the dataset as a *multivariate time series* with 13 features corresponding to the 13 SPNs, each obviously depending on time. From now on, having a multivariate object some other contents to perform a deeper analysis of signals interactions are introduced, specifically observing in deep relations between different SPNs.

From [9]:

> "In the realm of statistics, cross-correlation functions provide a measure of association between signals"

It is a procedure that quantifies the degree of similarity between two different sets of numbers. The common practice is to shift one time series with respect to the other, to inspect dependencies even if a parameter change affects the curve with a certain delay. The cross-correlation, at lag h, that is the number of samples of the shift performed, is defined as:

$$\tau_{x,y}(h) = \frac{\sigma_{x,y}(h)}{\sqrt{\sigma_{x,x}(0)}\sqrt{\sigma_{y,y}(0)}}$$

where x_t and y_t are the two time series, bot composed of N observations, μ_x and μ_y respectively indicate their means and $\sigma_{x,y}(h)$ is instead, the *cross-covariance function* defined as:

$$\sigma_{x,y}(h) = \frac{1}{N-1}\sum_{i=1}^{N}(x_{t-h} - \mu_x)(y_t - \mu_y)$$

Now, since $\sigma_{x,x}(0) = \sigma_x^2$ and $\sigma_{y,y}(0) = \sigma_y^2$ represent the variances of the time series x and y, then it is possible to define the *Pearson correlation* between the two variables as:

$$\tau_{x,y}(0) = \frac{\sigma_{x,y}(0)}{\sigma_x \sigma_y}$$

The Pearson Correlation Matrix between each pair of SPNs is displayed in Figure 3.6

From the analysis of Pearson correlation, it can be noticed that there is a set of highly correlated SPNs, describing the vehicle engine. This value is justifiable and does not represent a problem when variables are dependent on each other, because does not bring to multicollinearity problem, which concerns independent variables [10]. Removing these features would mean losing relevant data, and possible hidden information between these variables.

On the other hand, the variables showing trends, namely SPNs 108, 247, 1761, in general show low correlations with all the other variables, except between them. Considering the trivial trend previously noticed, and the high correlation between them, these variable can be discardable.

At last, variable 3509 is the only variable, showing none correlation with all the others.

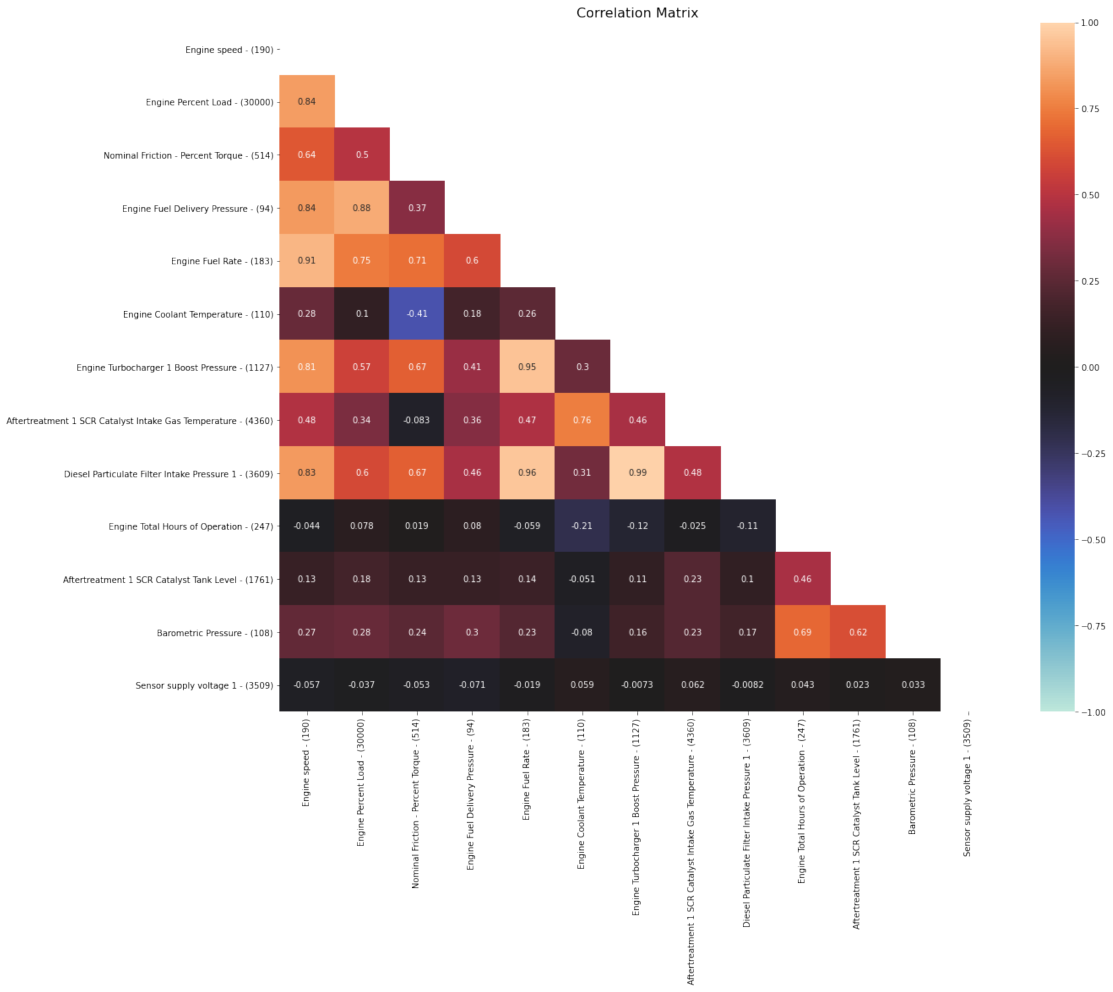


Figure 3.6: Pearson Correlation matrix

Further analysis can be done exploiting *Cross-correlation plots*. This plots display the behavior of cross correlation between two variables for strictly positive lags, since it is a symmetric function. Due to computational costs, this plot reports only 7200 lags, corresponding to 2 hours. Since it is a higher value than the average length of working cycles, this value is appropriate for cross-correlation description between parameters.

Figure 3.7 shows different correlograms. In these plots, the thin blue lines represent upper and lower limits of coefficient correlation, while zero value is denoted by the red line, indicating no correlation. The first plot shows cross-correlation between SPN 190 and SPN 30000, and it is representative of all the other highly correlated variables of the first analyzed group. After a strong correlation coefficient at the first lag, it slightly decreases until zero is reached, and then it almost has a constant behaviour around zero, sometimes showing smooth and not significant oscillations. The opposite behavior is displayed in the second plot, where the most negative highly correlated features are exploited, SPN

110 and SPN 514. Instead, looking at the cross-correlation plot between two poorly correlated features, it is noticeable the constant zero behaviour of the coefficient. An example is provided from the correlogram between SPN 514 and SPN 247 in the third plot.

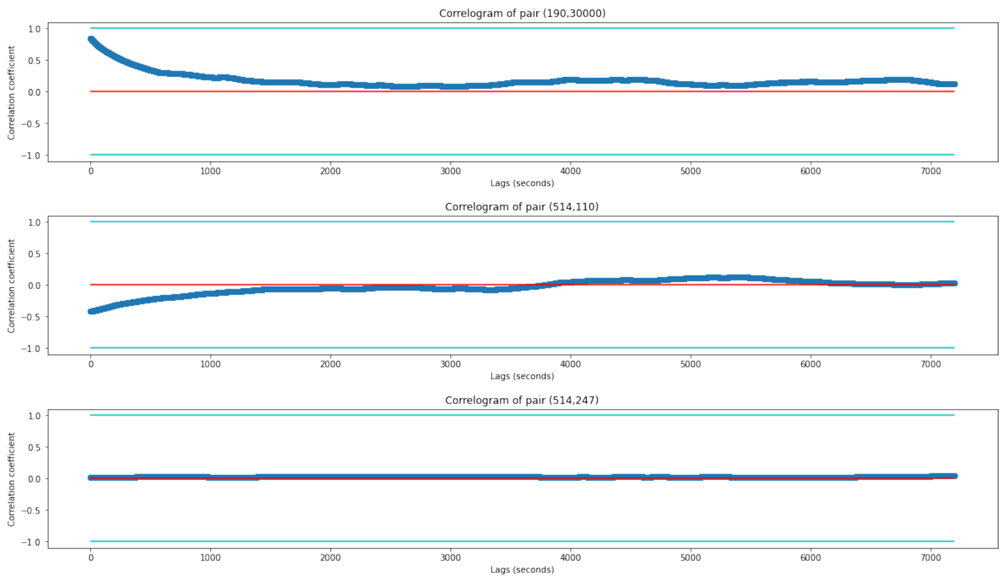


Figure 3.7: Correlogram of different pairs of SPNs

Moreover, keeping in mind the goal of this work, and the previous analysis, some variables are considered not useful, keeping few informative data for the task. For this reason, the following variables are no further considered:

- SPN 108 - Barometric Pressure refers to ambient conditions, that have no relation with the task. Specifically, the barometric pressure sensor responds to pressure changes in the atmospheric pressure.
- SPN 247 - Engine Total Hours of Operation: this variable, as previously introduced shows a trivial upward trend independent on vehicle states.
- SPN 1761 - Aftertreatment 1 SCR Catalyst Tank Level: as previously seen, this measure, indicating the tank level of the Catalyst, shows a downward trend, interrupted with a tank refill. Following the same reasoning as the previous case, it can be discarded.
- SPN 3509 - Sensor Supply Voltage 1: even if it is a useful indicator to power up sensors with a certain voltage, it is not informative for clustering purposes.

From now on, data are stored in a multivariate time series composed of the most informative 9 SPNs extracted from raw data:

- *SPN* 94: "Engine Fuel Delivery Pressure"
- *SPN* 110: "Engine Coolant Temperature"

- *SPN* 183: "Engine Fuel Rate"
- *SPN* 190: "Engine Speed"
- *SPN* 514: "Nominal Friction - Percent Torque"
- *SPN* 1127: "Engine Turbocharger 1 Boost Pressure"
- *SPN* 3609: "Diesel Particulate Filter Intake Pressure 1"
- *SPN* 4360: "Aftertreatement 1 SCR Catalyst Intake Gas Temperature"
- *SPN* 30000: "Engine Load"

Chapter 4

Auto Encoder-based Model for Time Series Analysis

In this Chapter, an innovative autoencoder-based method for time series clustering is applied.

To prepare data, several steps of preprocessing are performed as data segmentation using VALMOD algorithm and data normalization.

Then, autoencoders are introduced. In this context, the reasons which led to the exploitation of this special kind of artificial neural networks for time series data analysis are explained. Furthermore, details about their architecture and the training process parameters are provided.

Finally, as final phase of this work, after an introduction of the Clustering process, describing its goals, and the different ways by which this process can be performed, a clustering method is applied on the distance matrix, described in the following.

4.1 Auto Encoder-based approach

This procedure is based on the *Regression Cluster* concept [11], according to which similar objects are characterized by similar regression function. The regression function that is exploited in this task is the autoencoder, while the objects under analysis are homogeneous multivariate segments extracted from the original time series object. It is expected that autoencoders will be characterized by the same performances when tested on series describing the same duty of the fragments on which they are trained.

4.1.1 Data Preparation Procedures

Data preparation is a process of manipulating data, to the purpose of transforming them into a more structured and useful form for the following analysis. This is a critical part of the pipeline since the way in which data are treated can greatly affect the models that are learned.

Data can be prepared in different ways, strongly depending on the analytic objective and the specific adopted learning techniques. It is relevant for different reasons: algorithms require input data with a specific format and normalized.

An important step of data preparation is normalization. The effectiveness of any learning algorithm is heavily dependent on the normalization method [12]. In neural networks, unscaled input variables can result in a slow or unstable learning process, whereas normalize them not only speeds up learning but leads to faster convergence and better results [13].

Even if a high number of normalization techniques are available in the literature, a brief description of some of them is provided in the following.

Min-Max Normalization is an approach in which the data is scaled to a range of [0,1] or [−1,1]. For the conversion of an input value x of the attribute X to the range $[low, high]$, it is used the formula

$$x_{norm} = \frac{(high - low) * (x - min(X))}{max(X) - min(X)}$$

where $min(X)$ and $max(X)$ are the minimum and maximum values of the attribute X of the input set.

Z-score Normalization, transforms data to obtain zero mean and unit variance observations. Here, the conversion of the value x of the attribute X is obtained applying the formula

$$x_{norm} = \frac{(x - \mu(X))}{\delta(X)}$$

where $\mu(X)$and $\sigma(X)$ are the mean value and the standard deviation of the feature X.

VALMOD Algorithm

(*Definitions in this Section are borrowed from [14]*)

Time series segmentation is discussed in the literature in different contexts. It is a preprocessing step that aims at finding a partitioning of a time-series X into c homogeneous segments [4]. In modern production systems, a huge amount of historical process data are recorded with distributed control systems, and the segmentation of multivariate time series is especially important in the analysis of this kind of data. In the literature, many algorithms have been proposed for the representation of time series in their segmented form, and for the determination of an adequate number of homogeneous segments. In the multivariate context, however, there are not many applications, and the individual segmentation of each variable, followed by a combination of results sometimes cannot be sufficient for a correct segmentation.

To infer optimal length of segments, since no multivariate procedure exists, what is typically done is to monitor some principal components[15].

Specifically, in this work it is used VALMOD algorithm [14], a scalable one-dimensional approach to discover the repeated patterns of variable length in data.

This algorithm, in order to find the optimal length of segments, given a user-provided range $[L, U]$ and a univariate time series $X(t)$ finds the subsequence pairs, characterized with the smallest Euclidean distance of each length.

Definition 4.1.1 *A subsequence $X_{i,l} \in \mathbb{R}^l$ of a data series X is a continuous subset of values from X of length l starting from position i.*

$$X_{i,l} = [x_i, x_{i+1}, ..., x_{i+l-1}]$$

The motif pairs are instead identified as the subsequence pairs characterized by lowest Euclidean distance.

Definition 4.1.2 *$X_{a,l}$ and $X_{b,l}$ is a motif pair if:*

$$dist(X_{a,l}, X_{b,l}) \leq dist(X_{i,l}, X_{j,l}) \quad \forall i, j \in [1{,}2, ..., n - l + 1]$$

where $a \neq b$ and $i \neq j$, and dist is a function which computes the z-normalized Euclidean distance between the input subsequences.

All the distances between a subsequence and all the other subsequences of the same data series is an ordered array called *distance profile*.

Definition 4.1.3 *A distance profile $D \in \mathbb{R}^{(n-l+1)}$ of a data series X regarding subsequence $X_{i,l}$, is a vector that stores $dist(X_{i,l}, X_{j,l}) \quad \forall j \in [1{,}2, ..., n - l + 1]$, where $i \neq j$.*

Finally, for the location of the data series motif, it is necessary the computation of the *matrix profile*.

Definition 4.1.4 *A matrix profile $MP \in \mathbb{R}^{(n-l+1)}$ of a data series is a metadata series that stores the distances between each subsequence and its nearest neighbour, where n is the length of the data series and l is the given subsequence length.*

In order to find the data series motif, the two lowest values in the matrix profile must be exploited. It could be possible to have trivial matches, with a pattern matched or overlapped with itself, and to avoid this, the matrix profile incorporates an *exclusion zone*, a region before and after the location of a given query, that should be ignored, heuristically set to $\frac{l}{2}$.

As output, VALMOD algorithm returns the list of most similar pairs with the associated distance. However, since it searches for different length segments, ranking motifs requires computing a normalized distance, taking into account the segments length. The adopted distance is the Euclidean distance, where a normalization factor equal to $\sqrt{\frac{1}{l}}$ with l equal to the segments length, is introduced.

To limit computational costs, VALMOD exploits the idea that, if the nearest neighbor of $X_{i,l_{min}}$ is $X_{j,l_{min}}$, then probably the nearest neighbor of $X_{i,l_{min}+1}$ will be $X_{j,l_{min}+1}$. However, this is not always true and another rank-preventing measure is exploited to accelerate the computation. It is created and sorted a new vector called *lower bound distance profile*, containing the lower bound distance between $X_{i,l+k}$ and $X_{j,l+k}$, $\forall k \in [1{,}2, ...]$ which can be helpful to prune the number of needed computations.

The first step to evaluate the lower bound distance profile is the definition of the lower bounding Euclidean distance. Supposed the knowledge of the z-normalized Euclidean distance $d^l_{i,j}$ between two subsequences of length l, $X_{i,l}$ and $X_{j,l}$, the distance of two longer subsequences of length $l+k$, $d^{l+k}_{i,j}$, can be estimated by finding a lower bound function $LB(d)$ such that

$$LB(d^{l+k}_{i,j}) \leq d^{l+k}_{i,j}$$

In this work, the task is to find the most frequent lengths among the top k repeated patterns in order to find the most suitable length for segmentation.

4.1.2 Introduction to Auto Encoders

The method applied for time series clustering is based on autoencoders, a specific type of neural network architectures.

Artificial Neural Networks (ANN) are artificial adaptive systems inspired by human brain processing functioning [16], able to modify their internal structure in relation to an objective function. The basic elements of the ANN are the nodes, also called *neurons* and the connections, named *synapses*. The idea behind neural networks is to carry out complex computations through the communication of many neurons linked together [17].

Each single neuron is modelled as a simple scalar function, called *activation function*, and compute its output as the weighted sum of its inputs. Each edge in the graph links the output of some neuron to the input of another neuron and corresponds to a certain weight.

Neurons are organized in *layers*, where the first one is the *input layer*, and contains $n+1$ neurons, with n dimensionality of the input space. Then, it is followed by *hidden layers*, whose nodes are connected with the nodes of the previous layer, and the first one linked to the input. The output nodes, collectively referred as *output layer*, are linked with the last hidden layer and are responsible for transferring information from the network to the outside world. The values of weights are initially unknown and learned and updated through an iterative process, also called *learning process*. The process of learning of a neural network aims to find a configuration for the weights that minimize the training error and consists of two main processes:

- *Forward propagation* process consists of the propagation of the input, up to the intermediate units, until it arrives to the output layer, where the output is computed [18].
- *Backward propagation* process aims at computing the gradient of the loss one layer at the time, to the purpose of finding the values of the weights that minimize the training loss.

From [19]:

> *"An Autoencoder is a bottleneck architecture that turns a high-dimensional input into a latent low-dimensional code (encoder), and then performs a reconstruction of the input with this latent code (decoder)"*

Using two multilayer networks, where one provides dimensionality reduction and another reconstructs the input [19], it represents a nonlinear generalization of Principal Component Analysis (PCA). However, the non-linearity of neural networks allows property handle more complex and nonlinear data [20]. The main characteristic of autoencoders is the representation of input in a more compressed form, obtained as output of the encoder, in the *latent space*. Latent space is relevant since its value is at the deep learning's core: learn the features of data, and simplify data representations for the goal of finding patterns. Data compression wants to encode information with fewer bits with respect to the original representation. The encoded representation focuses on the most important features, typically resulting in removing noise or external alterations. It is very useful in many clustering and classification applications.

Bias-variance trade-off is an important autoencoder trade-off. The architecture must be able to provide a good reconstruction of the input, and at the same time to produce a latent space able to generalize to a meaningful one [21].

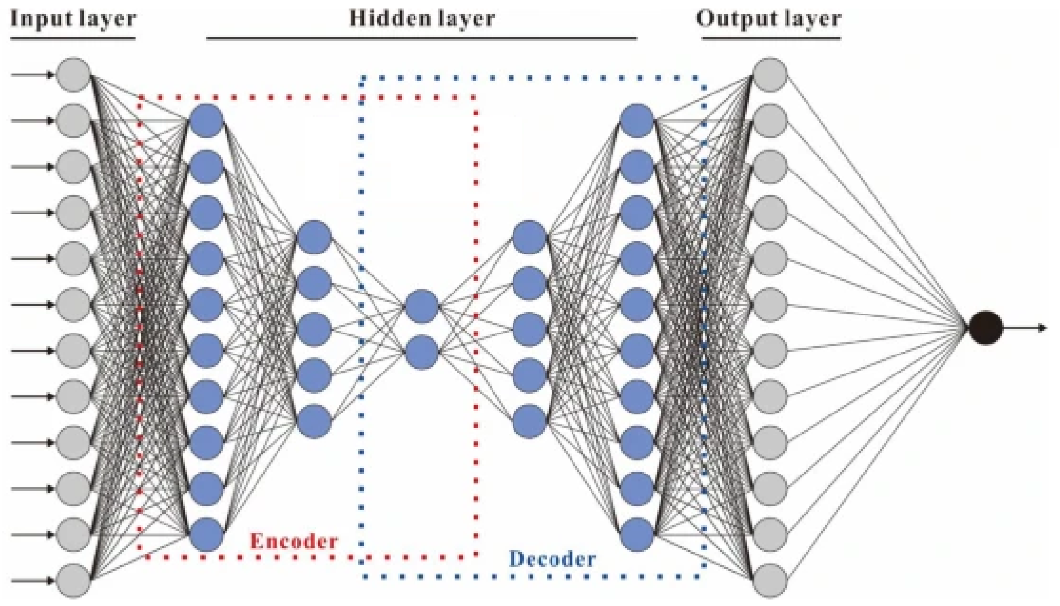


Figure 4.1: Architecture of an autoencoder based on deep neural network [22]

Starting with random initialization of weights of the network, it is trained by minimizing the error between the original data, input of the encoder, and the reconstruction of the decoder. Considering f the encoding function defined from the input space to the latent space, and g the decoding function, mapping data from latent space to output space, the learning process can be easily described as minimizing a loss function

$$L(x, g(f(x)))$$

where L is a loss function penalizing $g(f(x))$ for being dissimilar from x.

The required gradients are easily obtained by using the chain rule to backpropagate error derivatives first through the decoder network, and then through the encoder network.

Autoencoders are very popular in many applications for their variety. Some of the main kinds of autoencoder are briefly described in the following [21]:

- *Undercomplete Autoencoders* are the simplest kind of autoencoders. Hidden layer has a smaller dimension in comparison to the input layer, since the goal is obtaining representative features from the data.

- *Deep Autoencoders* exploits depth of the network. This can exponentially lead to a reduction of computational costs for the representation of some functions, but also to an important decrease of the needed training data for learning purposes. For these models, there are higher chances for overfitting to occur.

- *Convolutional Autoencoders* are simple autoencoders characterized by convolution layers in the encoding network and deconvolution or upsampling layers in the decoding network.

- *Sparse Autoencoders* handle the autoencoders trade-off enforcing sparsity on the hidden activation. They consist of the introduction in the autoencoder optimization objective, of a regularization term, as *L1 regularization*, inducing sparsity, or *KL-divergence*, that is a measure of the distance between two probability distributions.

- *Denoising Autoencoders* can be viewed in two different ways. It is used as a robust autoencoder for error correction or even as a regularization option. Indeed, the approach followed in these models is to disrupt the input with some noise, forcing the autoencoder to reconstruct the cleaned version of it.

- *Variational Autoencoders (VAE)* use a variational approach for latent representation learning, by making strong assumptions on latent variables distribution. VAEs are models exploiting a probabilistic distribution, in order to describe data generation.

4.1.3 Autoencoder-based Deep Learning Approach for Analysis of Time Series Data

As previously introduced, some data preparation steps are performed.

As seen in Section 4.1.1, different data normalization methods are available, and for each of the previously described ones, a box plot representation is graphically displayed in Figure 4.2 to the purpose of evaluating results. MinMax Scaler is a technique to rescale data in the range between 0 and 1, while Standard scaler transforms input observations to obtain zero mean and unit variance data. In multivariate context, each component is treated separately. In this work, to preserve the general shape of input data that are not likely to be drawn from a normal distribution, minmax scaler is applied.

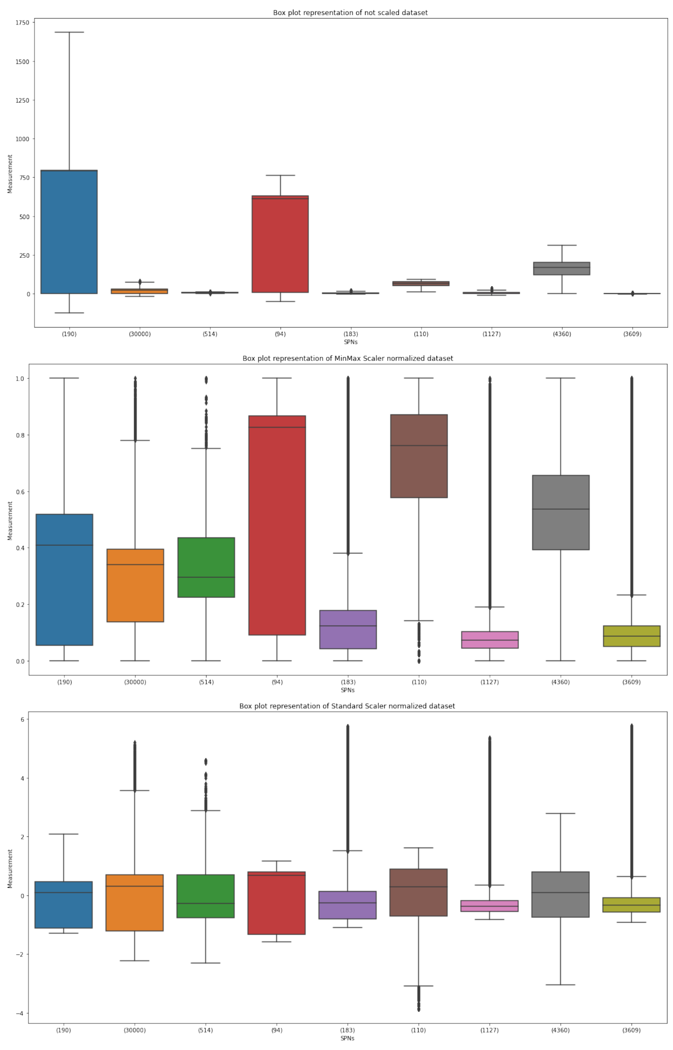


Figure 4.2: Boxplot of SPN, showing effects of normalization

Since the aim of the thesis work is to find repeated patterns describing different workload states, the multivariate time series must be divided into segments, possibly homogeneous. For this purpose it is important to identify a proper window size.

The larger the window, the more information about time series can be taken into account. On the other hand, this can incur heterogeneous segments describing more than one vehicle state. For this reason, with the help of domain experts, an iterative approach is used to properly identify the window length. According to domain experts, the searched fragment length ranges from 5 to 10 minutes, by looking at a trade-off between too tiny windows, that would not be enough informative to support the analysis, and too wide fragments, leading to heterogeneity. Starting from the lower limit of the indicated interval, the segment length is iteratively increased until a sufficiently high number of observations is available to train, for each segment, an autoencoder. This procedure leads to 6 minutes segments, resulting into 328 windows explained by the 9 considered features (Figure 4.3).

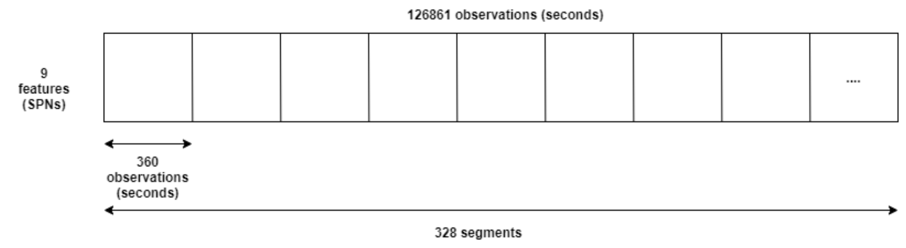


Figure 4.3: Time series window definition

Each segment is then used to train an autoencoder, resulting in 328 different models. The objective is to define an object able to learn how to distinguish different vehicle states homogeneously described by fragments. Once the training phase of each model is ended, there will be 328 autoencoders able to recognize and then reconstruct different patterns.

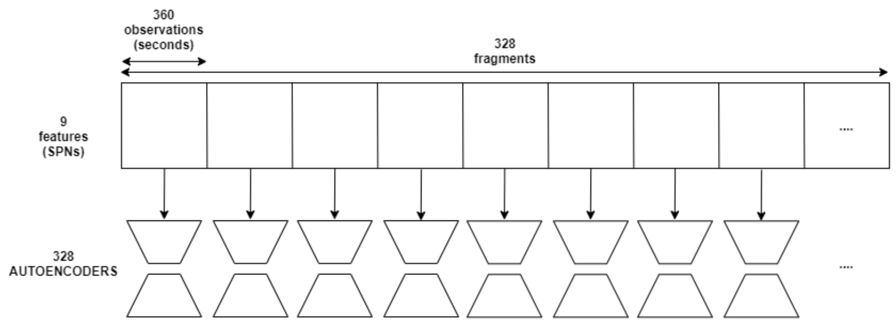


Figure 4.4: Proposed approach: an autoencoder-based model for time series analysis

Each model is trained on subfragments extracted from each previously identified windows. In this case, the choice of the windows' dimension is critical since a smaller window size will result in more subsegments to train the network, while too tiny subsegments risk to be not informative. For this reason, VALMOD algorithm is applied to infer the proper subsegment length. To extend the algorithm to multivariate context, it takes as input the principal component extracted from the original dataset [23], since VALMOD is a scalable one variable approach. The algorithm returns the rank of most similar subfragments pairs, with lengths from 30 to 50 seconds. Then, counting the number of motifs per length in the top 20 motifs and taking the length that maximizes that count, the most suitable size for subsegments results equal to 30 seconds.

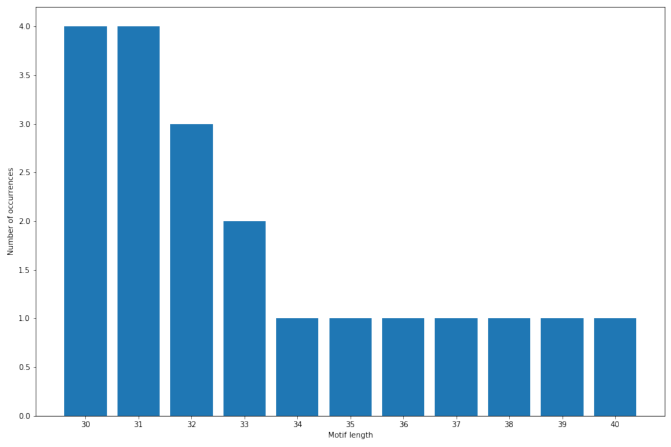


Figure 4.5: Number of motifs occurrences per length

Offset1	Offset2	Normalized distance	motif length (s)
216	1722	0.019662	35
217	1723	0.019817	34
215	1721	0.019846	36
218	1724	0.020739	33
219	1725	0.021369	32
42569	42813	0.021539	30
42568	42812	0.021764	31
221	1726	0.022058	31
881	58118	0.022072	32
880	58117	0.022133	33
220	1725	0.022139	32
882	58119	0.022229	31
883	58120	0.022443	30
225	1727	0.023137	30
222	1724	0.023360	30
224	1726	0.023661	31
874	58111	0.023710	39
873	58110	0.023730	40
875	58112	0.023882	38
876	58113	0.023980	37
872	58109	0.023981	41

Table 4.1: Top 20 VALMOD found motifs

In order to increase the available training data for each model, overlapping fragments with dimensions of 30×9 and stride $= 1$, are considered. The described approach, applied to each segment, will lead to 330 subfragments of shape 30×9 per segment (Figure 4.6).

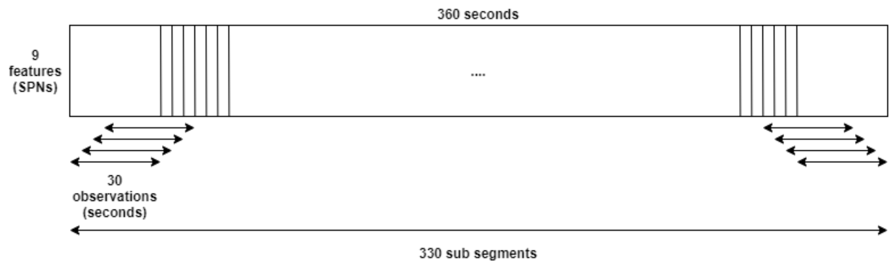


Figure 4.6: Data input generation process

The architecture of each model, described in Table 4.2, is mainly inspired to [24], and it is based on convolutional neural networks, widely used for time series analysis, thanks to their ability of extracting spatial features given their great success in images processing [25], [26].

The size of the input of the encoder, which is the same as the output of the decoder,is then a tensor of shape (30,9,1). The autoencoder architecture is built considering the hints in [27]. Then, in order to improve generalization in terms of reconstruction error, and speeding up training, the height and width of the bottleneck are kept large, as well as the number of channels. Each autoencoder is trained with a loss composed of the Mean Squared Error (MSE) term plus a term of regularization, equal to the $L2$ norm of the coefficients weighted with $\lambda = 0.05$ to the purpose of improving the network generalization power.

Mean Squared Error is a way to measure the model performances [28]. In general, if $\hat{x}$ represents the output of decoder, x represents the input, and n the input size, then the MSE is given by:

$$MSE = \frac{\sum_{k=1}^{n}(x_k - \hat{x}_k)^2}{n}$$

$L2$ regularization introduces additional information to avoid overfitting phenomena. It introduces, as a penalty to the loss function, the squared values of the magnitude of the coefficients. This penalty is weighted with a parameter λ.

$$L2 = \lambda \sum_{i=1}^{n} \omega_i^2$$

Another option is $L1$ regularization, which instead introduces as penalty term, absolute magnitude of coefficients. $L2$ is useful with collinearity or with codependent features since it tends to uniformly shrink the coefficients, while $L1$ is typically useful for feature selection since it shrinks coefficients of the less important features to zero.

As optimizer, Adam method has been selected. Adam [29], exploits estimates of first and second moments of the gradients for the computation of different parameters of individual adaptive learning rates. It differs from Stochastic Gradient Descent (SGD)

Encoder						
Layer	**Num. filters**	**Kernel size**	**Stride**	**Input Size**	**Output Size**	**Act. Func**
Conv2D	16	3	1	(?,30,9,1)	(?,30,9,16)	ReLU
BatchNorm				(?,30,9,16)	(?,30,9,16)	
MaxPool2D		(2,1)		(?,30,9,16)	(?,15,9,16)	
Conv2D	32	3	1	(?,15,9,16)	(?,15,9,32)	ReLU
BatchNorm				(?,15,9,32)	(?,15,9,32)	
Conv2D	32	3	1	(?,15,9,32)	(?,15,9,32)	ReLU
BatchNorm				(?,15,9,32)	(?,15,9,32)	
Conv2D	64	3	1	(?,15,9,32)	(?,15,9,64)	ReLU
BatchNorm				(?,15,9,64)	(?,15,9,64)	
Conv2D	32	3	1	(?,15,9,64)	(?,15,9,32)	ReLU
BatchNorm				(?,15,9,32)	(?,15,9,32)	
Conv2D	16	3	1	(?,15,9,32)	(?,15,9,16)	ReLU
BatchNorm				(?,15,9,16)	(?,15,9,16)	
Conv2D	8	3	1	(?,15,9,16)	(?,15,9,8)	ReLU
BatchNorm				(?,15,9,8)	(?,15,9,8)	
Decoder						
Layer	**Num. filters**	**Kernel size**	**Stride**	**Input Size**	**Output Size**	**Act. Func**
Conv2D	8	3	1	(?,15,9,8)	(?,15,9,8)	ReLU
BatchNorm				(?,15,9,8)	(?,15,9,8)	
Conv2D	16	3	1	(?,15,9,8)	(?,15,9,16)	ReLU
BatchNorm				(?,15,9,16)	(?,15,9,16)	
Conv2D	32	3	1	(?,15,9,16)	(?,15,9,32)	ReLU
BatchNorm				(?,15,9,32)	(?,15,9,32)	
Conv2D	64	3	1	(?,15,9,32)	(?,15,9,64)	ReLU
BatchNorm				(?,15,9,64)	(?,15,9,64)	
Conv2D	32	3	1	(?,15,9,64)	(?,15,9,32)	ReLU
BatchNorm				(?,15,9,32)	(?,15,9,32)	
Conv2D	32	3	1	(?,15,9,32)	(?,15,9,32)	ReLU
BatchNorm				(?,15,9,32)	(?,15,9,32)	
UpSample2D		(2,1)		(?,15,9,32)	(?,30,9,32)	
Conv2D	16	1	1	(?,30,9,32)	(?,30,9,16)	ReLU
BatchNorm				(?,30,9,16)	(?,30,9,16)	
Conv2D	1	1	1	(?,30,9,16)	(?,30,9,1)	ReLU
Loss					MSE + L2	

Table 4.2: Architecture of each of the autoencoders

that considers a fixed learning rate for the entire training process. Adam is faster with respect to SGD [30].

Finally, batch normalization after each convolutional layer is adopted to reduce the vanishing gradient problem, while the usage of small filters is helpful to have fewer parameters and at the same time increase the nonlinearity.

Before starting the training process, the last 20% of the dataset is extracted in order to be used for testing autoencoders performances. Moreover, for parameters tuning purposes, a grid search on learning rate, batch size and number of epochs is performed. It is combined with a 3-fold cross-validation process, which in time series fields consists of splitting training and validation sets into contiguous observations, as shown in Figure 4.7). In the process, the validation set is equal to the 10% of remaining observations. Finally, early stopping technique is used to prevent overfitting phenomena.

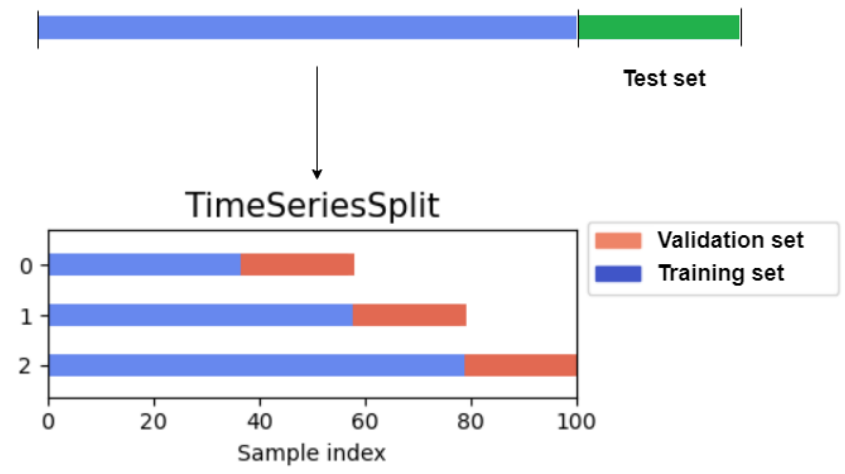


Figure 4.7: Time series split[1]

The trained autoencoders will be used in the clustering process that will be explained in the next sections.

4.2 Time Series Clustering

4.2.1 Introduction to Clustering

Data clustering is the process useful to identify clusters or natural groupings in a multidimensional context by exploiting some similarity measure [31]. Intuitively, data within the same cluster show greater similarity to each other in comparison to patterns

[1]`https://scikit-learn.org/stable/modules/cross_validation.html`

that belong to different clusters [32]. The assortment of techniques for representing data, grouping them by measuring proximity between data elements, has produced a rich variety of clustering methods. Clustering is a very useful tool in several machine learning realities, grouping, decision-making tasks, and exploratory pattern-analysis situations.

Given an objective function which it is denoted by G, the goal of a clustering algorithm is finding, through the use of an appropriate search algorithm, a clustering C, for a given input (χ, d), so that $G((\chi, d), C)$ is minimized [17].

Because of the specific purpose of this thesis work, the following considerations concern pattern clustering. Considering a *pattern* x as a single data item used by the clustering method, it typically consists of a vector of d vector components called *features* $\mathbf{x} = (x_1, .., x_d)$, where d is the dimensionality of the pattern space. In this case, a *class* indicates a state of the nature that control the generation process of the patterns [32].

The goal of the clustering techniques is to group patterns in such a way that the obtained classes offer a representation of the different patterns obtained from the process, in the pattern set. The goal is not only the maximization of the similarity of observations clustered in the same group, but also the maximization of dissimilarity of patterns that are grouped in different clusters.

In case of pattern clustering, the analysis becomes more challenging because of the introduction of dependencies over time [33]. In addition, the kind of problem under analysis typically deals with:

- *Unlabeled data*: time series data are less likely to be associated with data labels, making impossible to apply supervised learning techniques. On the other hand, unsupervised learning procedures can be useful tools for understanding data and finding unknown patterns, not easily findable with standard methods;
- *High dimensionality*: time series data are commonly associated with several components. However, it is very important to identify the contribution of each feature with the aim of summarizing the informative content into few parameters;
- *Hidden features*: the most critical issue is represented by the possible existence of some hidden features that can be missed when conducting direct data analysis.

Some of the most standard methods for time series clustering are briefly described in the following:

- Hierarchical time series clustering: A hierarchy is generated using:
  - *Agglomerative (bottom-up) approach*: starting from the situation in which each item is considered as a cluster, with a certain appropriate measure, clusters are merged together.
  - *Divisive (top-down) approach*: starting situation is a single cluster that includes all items, that will be continuously split.
- *Partitioning time series clustering*: it is an approach consisting of the generation of k clusters, exploiting their similarity (e.g. K-means).
- *Density-based time series clustering*: this class of methods defines a cluster as a subspace of dense objects (e.g. DBScan [34]).

- *Grid-based Time-series clustering*: the clustering operations are performed on a grid structure built quantizing the object space into a finite number of cells.
- *Model-based Time-series clustering*: a model is built for each cluster, with the aim of finding the best fit of input data.
- *Multi-step time series clustering*: it improves the cluster representation quality by combining different methods.

The focus in this work is on partitioning clustering. The most relevant partitioning clustering methods are *K-means* [35] and *K-medoids* [36].
(*The following definitions are borrowed from [17]*)

K-means algorithm partitions the dataset into disjoint sets $C_1, ..., C_k$, with centroids $\mu_1, ..., \mu_k$ providing a representation of them. A centroid is computed for each cluster and it is the element used by the algorithm ti represent all the items assigned to that cluster. K-Means algorithm iteratively assigns data items to the cluster with the closest centroid, that corresponds to solve the following optimization problem:

$$\min_{C_1, C_2, ... C_k} \sum_{i=1}^{K} \sum_{x \in C_i} d(x, \mu_i)$$

where $\mu_i = \sum_{x \in C_i} x/|C_i|$ is the cluster centroid and k and d, respectively the number of desired clusters and the distance to be used in computations, are user-specified hyperparameters. As it can be noticed, the centroid of each cluster is not necessarily an element of the dataset.

The main difference between K-means and K-medoids algorithms is that the latter forced centroids, referred as *medoids*, to be elements of the dataset. For this reason, it is not sensible to noise and outliers. Since the data under analysis are noisy for nature, K-Medoids will be the method that will be exploited in this thesis work.

The K-medoids algorithm is resumed in Algorithm 1

Algorithm 1 ***K-medoids algorithm***
Inputs: Input set $(x_1, ..., x_n)$, Number of desired clusters k and the distance to use
Output: Clusters assignments $(C_1, ..., C_n)$

```
1: Random selection of $k$ samples from $(x_1, ..., x_n)$ as the initial medoids $(\mu_1, ..., \mu_k)$.
2: repeat:
3:     Form $k$ clusters by assigning all points to the closest medoid
4:     Recompute the centroid of each cluster and take the closest example as the medoid of the cluster.
5: until the medoids don't change.
6: return Cluster assignments $(C_1, ..., C_n)$
```

The evaluation of clustering results, in unsupervised context, take into account several aspects [37]:

- *Internal or unsupervised validation*
  - Clustering tendency determination in data;
  - Determination of the correct number of data clusters;
  - Quality assessment of the clustering results without needing extra information.
- *External or supervised validation*
  - Comparison of the obtained clustering assignments with true labels;
  - Comparison between different sets of clusters.

In general, in unsupervised context, two types of validation metrics are considered:

- Cohesion within each cluster;
- Separation between different clusters.

Silhouette score combines in a single value these two metrics. It ranges in the interval [−1,1], where high separation is indicated from positive values, while poor clustering performances are characterized by negative values.

For each example i in the data set, Silhouette score is defined as:

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}$$

Averaging silhouette coefficients for each example, the *global Silhouette coefficient*, is obtained:

$$S = \frac{1}{n}\sum_{i=1}^{n} s(i)$$

4.2.2 K-medoids Clustering Approach

Since an hyperparameter of the K-Medoids algorithm is the distance to be used in computation, in this specific application a distance based on the reconstruction errors of autoencoders is exploited. Once the training phase of the 328 autoencoders is ended and autoencoders are tested on respective test sets previously extracted, the objective is to define a distance measure useful to distinguish different fragments, and mainly the representing states.

Given two 6 minutes fragments $\hat{x}$ and $\hat{y}$, considering the set subsegments $x = (x_1, ..., x_N)$ and $y = (y_1, ..., y_N)$ extracted respectively from them, the distance exploited in clustering is defined as:

$$dist(\hat{x}, \hat{y}) =$$
$$\frac{1}{N}(|\sum_{i=0}^{N}(x_i - AE_y(x_i))^2 - \sum_{i=0}^{N}(x_i - AE_x(x_i))^2| + |\sum_{i=0}^{N}(y_i - AE_x(y_i))^2 - \sum_{i=0}^{N}(y_i - AE_y(y_i))^2|)$$

where AE_x indicates that this autoencoder is trained on the subfragments set $x = (x_1, ..., x_N)$, and AE_y indicates that this autoencoder is trained on the subfragments set $y = (y_1, ..., y_N)$.

This distance measures how much the subfragments extracted from a segment are close to the reconstruction of the autoencoder trained on the other segment. The idea is that the more the input segments are similar, the better they will be the reconstruction performed by the autoencoders. It can be noticed that the formula actually defines a distance since it is non-negative, it satisfies the symmetric properties and it is null for $\hat{y} = \hat{x}$.

To run the K-medoids algorithm, the number of desired clusters must be specified as input parameter. However, real life applications as the one under analysis, the number of desired cluster is often unknown. The common strategy adopted to solve this problem is to iteratively evaluating the performance achieved by multiple runs of the algorithm by varying the number k of desired clusters. At the end, the optimal value of k is determined as the k that maximizes the silhouette score [38]. In this application, the number of desired clusters is ranging from 2 to 9, and the better performances are achieved by 3 clusters, with a silhouette score equal to 0.58, as shown in Table 4.3.

Number of clusters	**Silhouette**
2	0.13
3	**0.58**
4	0.41
5	0.37
6	0.46
7	0.37
8	0.38
9	0.37

Table 4.3: Silhouette values for different number of clusters

Chapter 5

Experimental results and evaluation

In this chapter, results obtained by applying the proposed strategy are deeply analyzed and evaluated, both qualitatively and quantitatively.

5.1 Qualitative Validation of Clusters

Once the clustering algorithm has generated different clusters, these are evaluated by a deep exploration with the experts' supervision.

Exploiting boxplot visualizations of the measures, separately for each cluster as shown in Figure 5.9, as well as time series plots over time, with the help of domain is possible to assign duties to the identified groups (Figure 5.1).

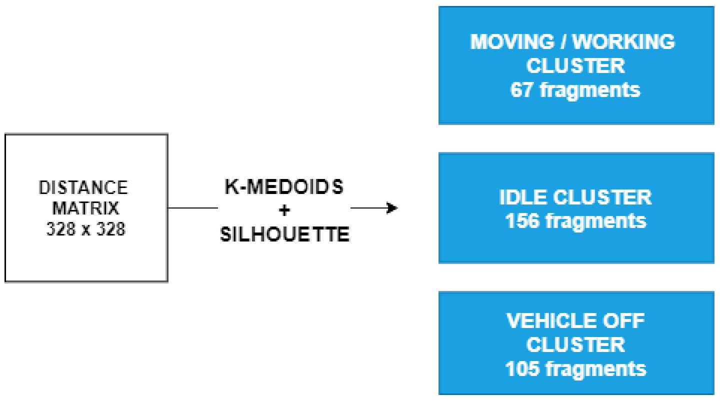

DISTANCE MATRIX 328 x 328
K-MEDOIDS + SILHOUETTE
MOVING / WORKING CLUSTER 67 fragments
IDLE CLUSTER 156 fragments
VEHICLE OFF CLUSTER 105 fragments

Figure 5.1: K-Medoids + Silhouette application

The first cluster, composed of 67 fragments, is characterized by fragments with SPNs that take the highest values (Figure 5.6). Engine Speed, as displayed in Figure 5.9, takes values around 2000 rpm, indicating probably a moving or working duty, that can be confirmed also by the higher load associated with this cluster with respect to the others.

The second group, which consists of 156 fragments, is the most populated one. Engine Speed is ranging between 800 – 1000 rpm as can be noticed in Figure 5.9, is probably denoting a vehicle turned on, but without moving or working. Even if engine load values are close to values assumed by moving/working segments, in this case they show a more stable and stationary behaviour with respect to the irregular one characterizing the cluster associated with higher workloads (Figure 5.2).

The same behaviour can be noticed also for the remaining measures, shown in Figure 5.7, typically denoted by stationary conditions at medium levels.

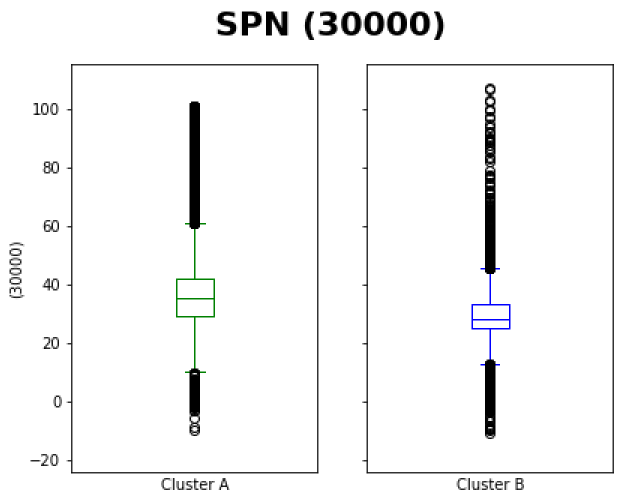


Figure 5.2: SPN 30000 Boxplots for Moving/Working Cluster and Idle Cluster

Finally, the last group is composed of 105 segments during which SPNs take the lowest values. The segments of this cluster are associated with periods in which the engine is off but the instrument panel of the vehicle is on, meaning that the CAN network is transmitting messages. As can be noticed in Figures 5.3, 5.4 and 5.5 the engine can be off for all observations of the segment, or an ignition or a shutdown can happen.

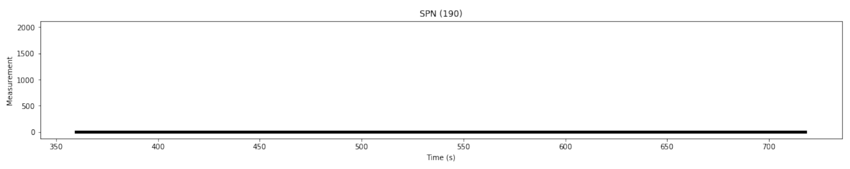


Figure 5.3: Vehicle Off fragment

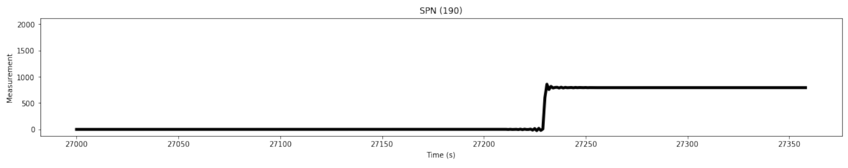


Figure 5.4: Vehicle Ignition fragment

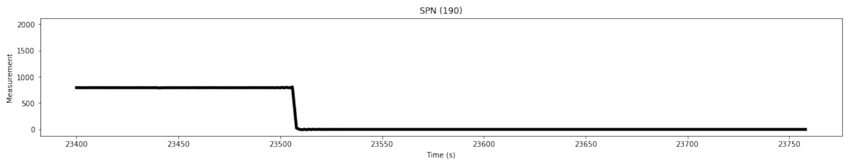


Figure 5.5: Vehicle Shutdown fragment

A global overview is provided by the time series plot in Figure 5.8.

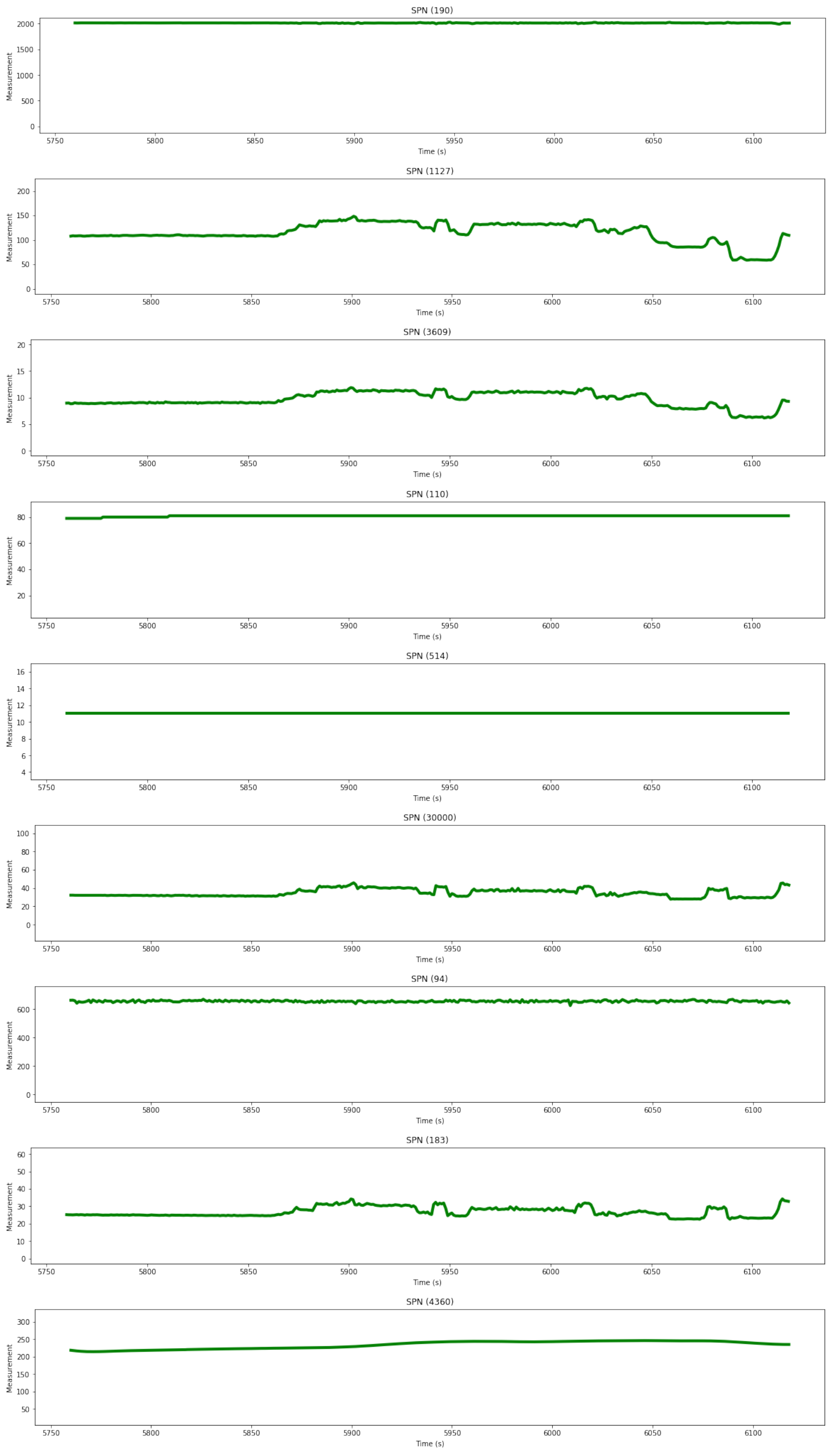


Figure 5.6: Time series plots Moving/Working Cluster

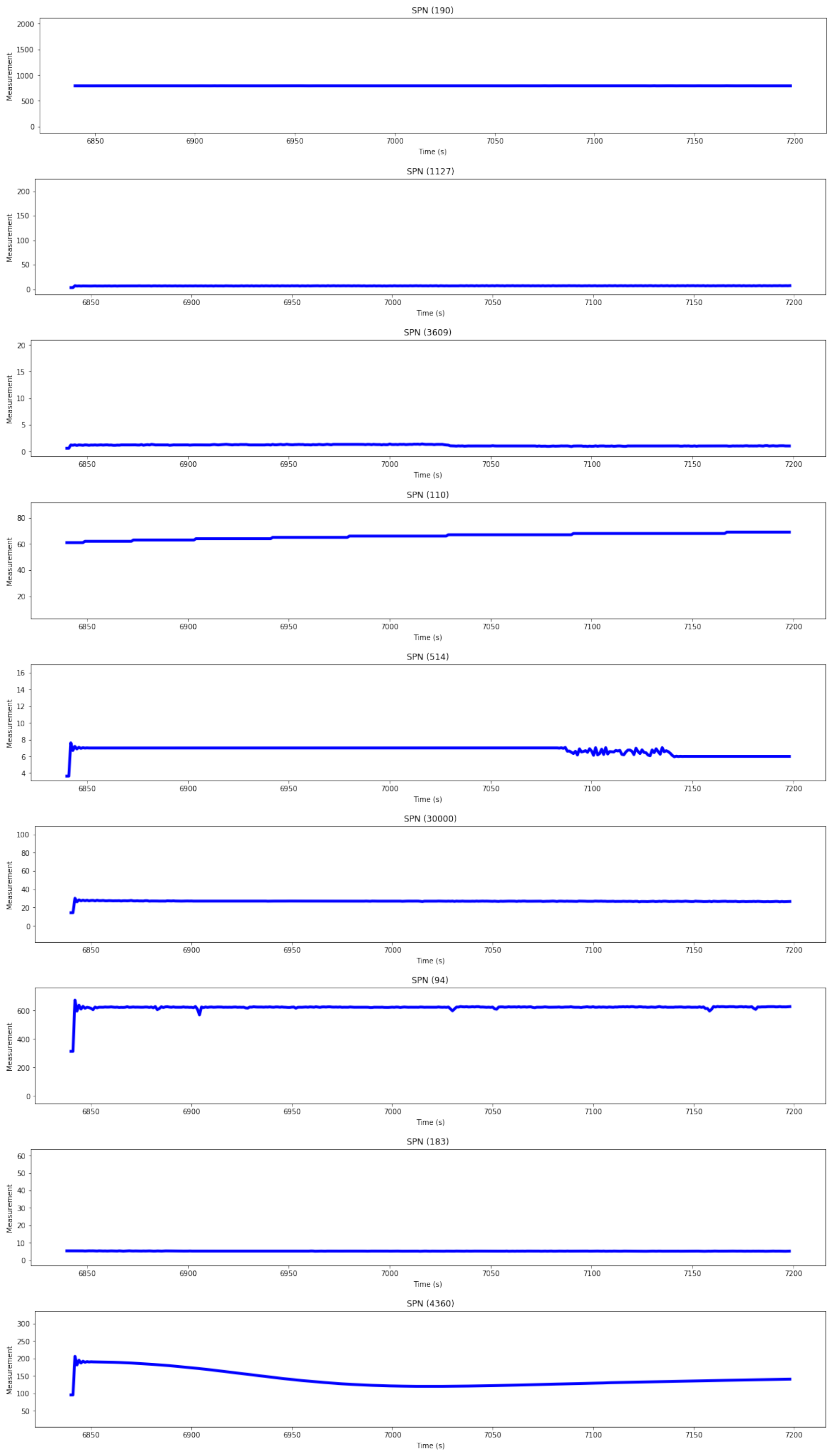


Figure 5.7: Time series plots Idle Cluster

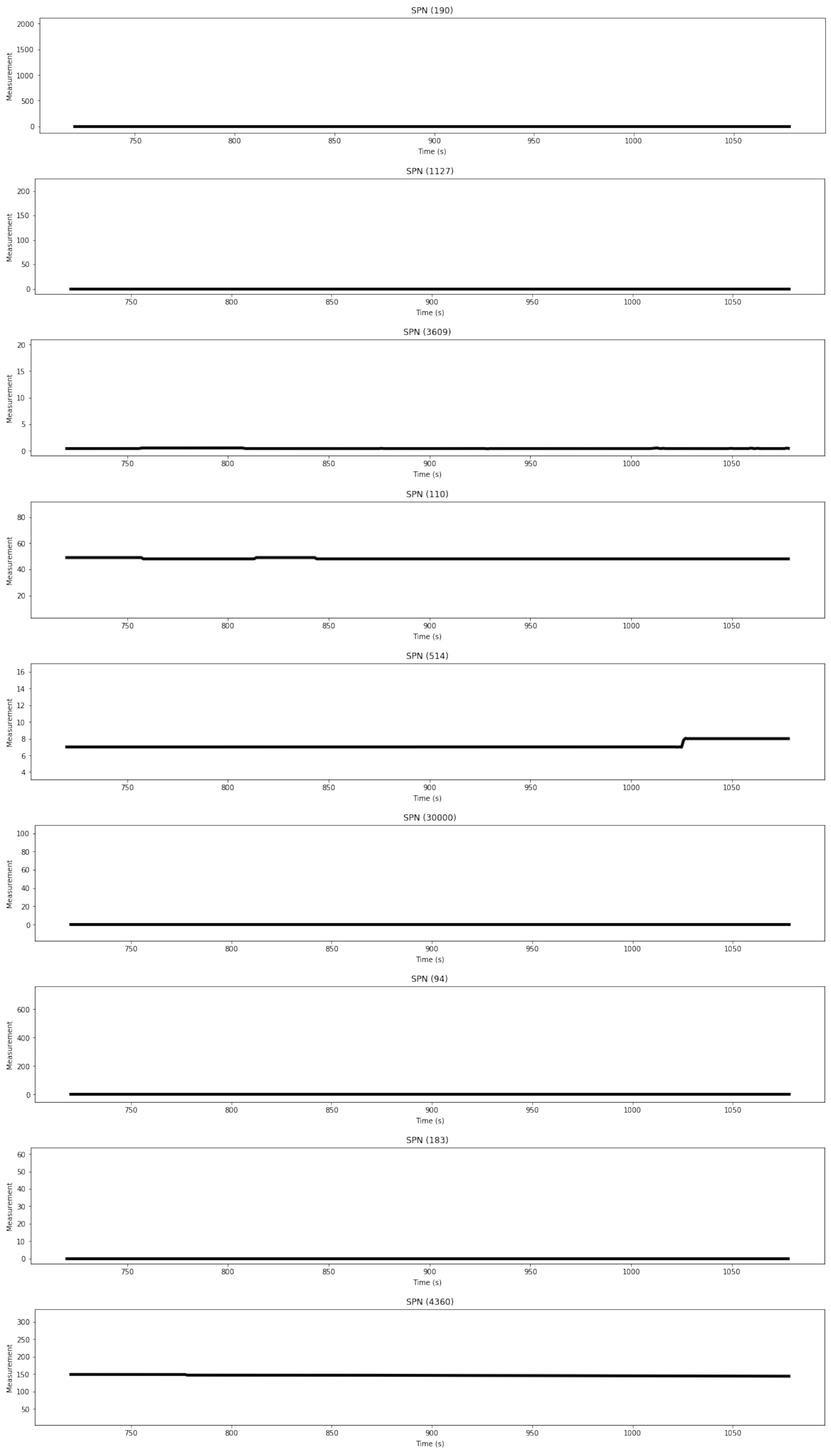


Figure 5.8: Time series plots Vehicle Off Cluster

Finally, a global view of the clusters found during the application of *K-Medoids* method is given by SPNs box plots, separately for each cluster.

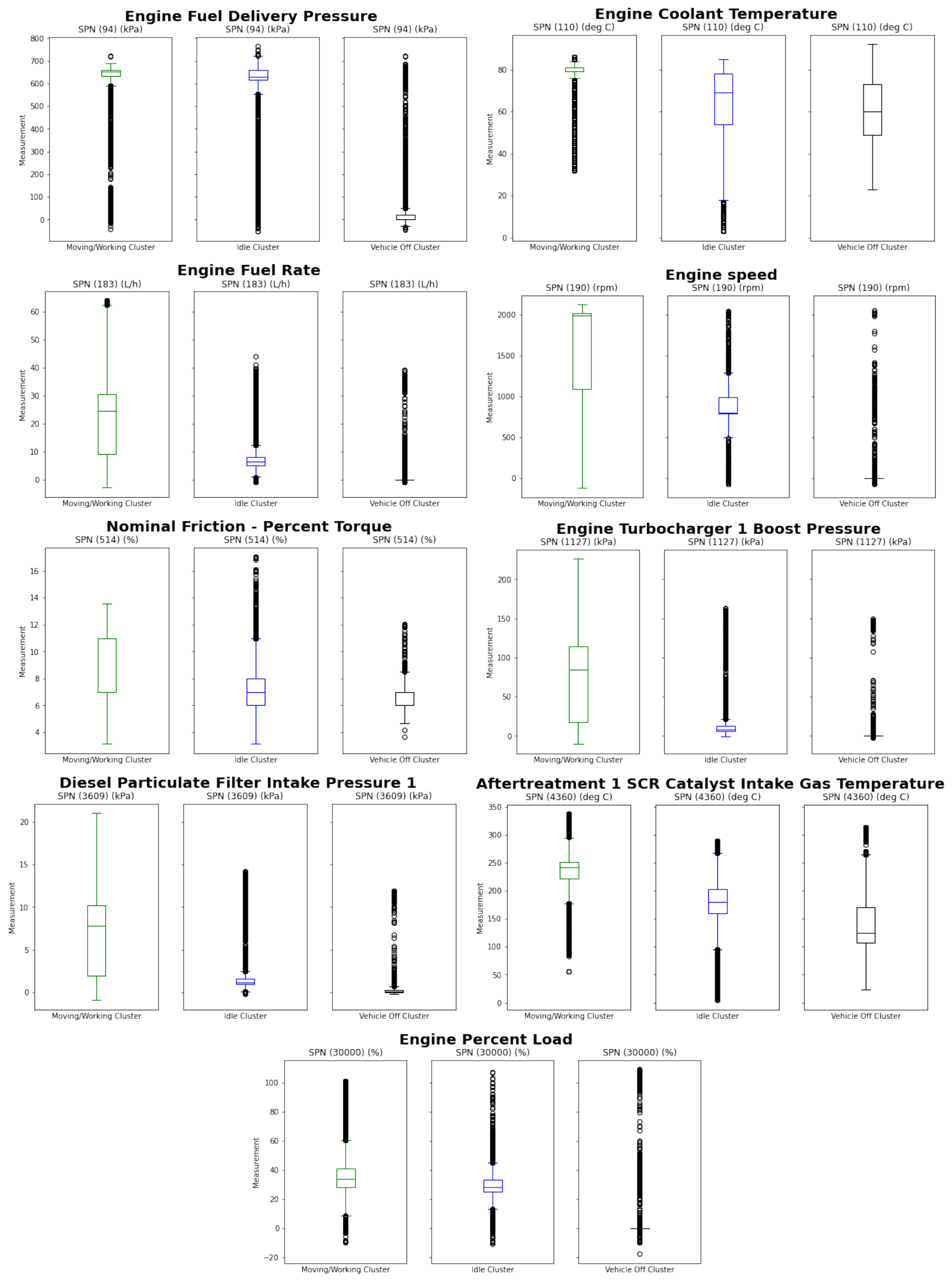


Figure 5.9: Box plot of the found clusters

However, the context of this analysis is completely unsupervised. Hence, it is not possible to confirm results with respect to ground truth labels.

5.2 Quantitative Validation of Clusters

To further validate results from a quantitative point of view, obtained clusters are compared with the performance achieved by standard distances commonly employed in time series clustering. The metric used in this unsupervised comparison is once again the silhouette score.

The purpose of this final analysis is to demonstrate that the proposed distance, with the drawback of higher computation costs, is preferable over standard measures since it allows to achieve better performances in time series clustering.

To this purpose, two different kinds of distances are analyzed, namely a lock-step measure that, given two segments, compares pairs of observations at the same time index, and an elastic measure, that creates a non-linear mapping to shift in time one signal with respect to the other. As representative distance for the first group of measures, the Euclidean distance is considered, while for the last group Dynamic Time Warping (DTW) is used [39].

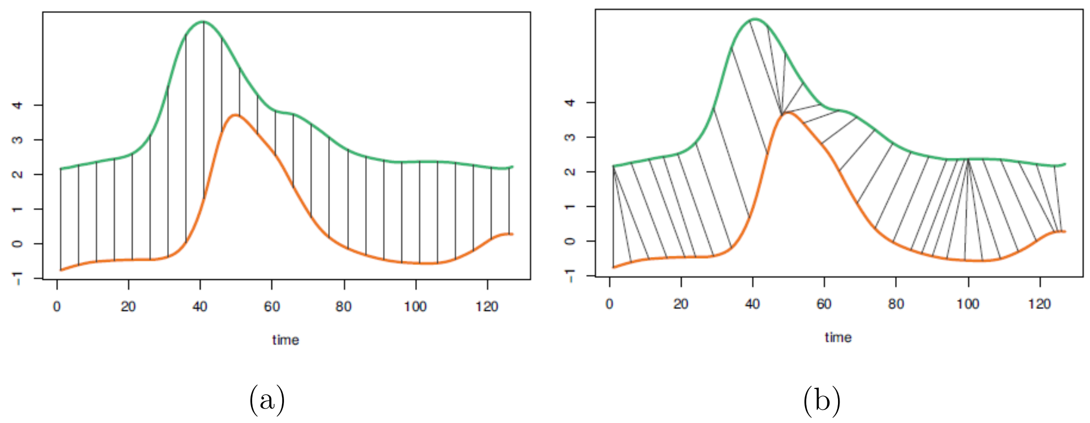


Figure 5.10: Mapping of Euclidean distance (a) vs. mapping of DTW distance (b) [39]

The dynamic time warping technique exploits a dynamic programming approach for time series alignment in order to minimize some distance measure [40]. Given two time series $X = (x_1, x_2, ..., x_N)$, $N \in \mathbb{N}$ and $Y = (y_1, y_2, ..., y_M)$, $M \in \mathbb{N}$, these can be arranged in a $N - by - M$ grid, where each point corresponds to the alignment between the elements of the two sequences x_i and y_i. Once the distance measure to use in the algorithm is defined, it is introduced a *warping path* $W = (w_1, w_2, ..., w_k)$, that aligns the elements of the different sequences, in such a way that the distance between X and Y is minimized. So the dynamic time warping problem can be formally defined as:

$$DTW(X,Y) = \min_{W}[\sum_{i=1}^{k} dist(w_k)]$$

where *dist* is the distance measure exploited.

Similarly to previous introduced strategy, a distance matrix is built for each of the considered measures that is used as input for the same clustering process. The number of desired clusters is once again empirically determined by silhouette approach, leading to results shown in Table 5.1.

Distance measure	**Number of clusters**	**Silhouette measure**
Customized	**3**	**0.58**
Euclidean	2	0.46
DTW	2	0.46

Table 5.1: Number of clusters found and Silhouette measure value associate to cluster assignments, for each measure

As it can be noticed, the silhouette score for clusters obtained setting Euclidean and DTW distances are quite similar, but the score achieved with customized distance is significantly higher.

Clusters have been qualitatively validated with the help of domain experts and are coherent with the expected ones, given the set of activities performed by the vehicle during data collection, and quantitatively validated with comparison with other measures. Their plots and their descriptions can support domain experts during duties levels identification.

Chapter 6

Conclusions and Future Works

This work has pointed out the importance of automated process for the identification, in heavy-duty vehicles field, of usage patterns and workload states. Instead of manually detecting the states of the vehicle, CAN bus data analysis with appropriate developed techniques can overcome the main drawbacks related to domain experts manual settings.

As previously seen, CAN bus data are generated at high frequencies by sensors installed on the vehicle with irregular sampling rates. Then, once stored in Tierra infrastructure, data are parsed by exploiting the database firm information, and handled to be more human-readable and for performing analysis on them. Being generated at high frequency, then gathered, and at last sent to a centralized server, where managed, this data can be easily subject to transmission errors or other errors of different nature.

From a preliminary data selection analysis, only a limited number of informative measures are identified. Then, since messages over the CAN network are sampled at different and non constant sampling rates, in order to obtain a multivariate time series, signals are aligned and synchronized by applying a suitable combination of upsampling and downsampling operators in the frequency domain.

Further analysis is performed before the process definition, taking into account how the autoencoder model must be fed in the following step. Considering the nature of data, and exploiting VALMOD algorithm for detection of variable length repeated patterns in time series, signals are divided into fixed length segments, with the aim of generating a high number of homogeneous multivariate time series describing single usage patterns of the vehicle.

Finally, an autoencoder-based clustering is applied. In this work, a convolutional deep autoencoder is exploited, considering the nature of data.

An autoencoder is trained for each segment to the purpose of defining a distance between segments based on the reconstruction ability of autoencoder. The distance is used in the selected clustering algorithm, k-Medoids, applied to identified duties. Since the number of desired clusters is unknown, it is empirically identified by assessing the silhouette score achieved by multiple runs, varying the number of clusters k. The value of k that maximizes the silhouette results equal to 3, leading to the identification of 3 duties, that can be interpreted as vehicle off, idle and moving or working. Results have been validated by domain experts and are considered coherent with the expected once, given the activity performed by the vehicle during the data collection phase. In addition, a quantitative evaluation based on silhouette scores achieved by the same clustering method with standard distances such as the euclidean distance and the dynamic time warping, demonstrated the better performance achieved by the proposed method.

This thesis work represents a real life application of an innovative clustering method capable of achieving significantly higher results, but of course it represents just a starting point. As future work, several improvents can be made: first of all, the number of signals sent on the network can be reduced to save storage and computational costs. Furthermore, the framework can be extended to handle variable length fragments, probably characterized by more homogeneous behaviour. Finally, the proposed autoencoder architecture can be improved, for example considering *Recurrent Neural Networks (RNN)* or *Long Short-Term Memory (LSTM)* networks, commonly employed with time series data thanks to their memory properties.

Bibliography

[1] Uwe Koppe. «Combining CANopen and SAE J 1939 networks». In: 2013 (cit. on p. 5).

[2] Fugiglando Umberto, Massaro Emanuele, Santi Paolo, Milardo Sebastiano, Abida Kacem, Stahlmann Rainer, Netter Florian, and Ratti Carlo. «Driving Behavior Analysis through CAN Bus Data in an Uncontrolled Environment». In: *IEEE Transactions on Intelligent Transportation Systems* PP (Oct. 2017). DOI: 10.1109/TITS.2018.2836308 (cit. on p. 10).

[3] Marco Luise and Giorgio M. Vitetta. *Teoria dei segnali.* McGraw-Hill, 1981. ISBN: 88-386-0809-1 (cit. on pp. 14, 16, 27).

[4] Janos Abonyi and Balazs Feil. *Cluster Analysis for Data Mining and System Identification.* 2007. ISBN: 978-3-7643-7987-2. DOI: 10.1007/978-3-7643-7988-9 (cit. on pp. 18, 35).

[5] Zimmerman Donald. «A Note on the Influence of Outliers on Parametric and Nonparametric Tests». In: *Journal of General Psychology - J GEN PSYCHOL* 121 (Oct. 1994), pp. 391–401. DOI: 10.1080/00221309.1994.9921213 (cit. on p. 22).

[6] Osborne Jason and Overbay Amy. «The Power of Outliers (and Why Researchers Should Always Check for Them)». In: *Pract. Assess. Res. Eval.* 9 (Jan. 2004) (cit. on p. 22).

[7] Martin Vetterli, Jelena Kovacevic, and Vivek K. Goyal. *Foundations of Signal Processing.* Cambridge Univ. Press, 2014 (cit. on pp. 23, 25, 28).

[8] Huybrechts Thomas, Vanommeslaeghe Yon, Blontrock Dries, Van Barel Gregory, and Hellinckx Peter. «Automatic Reverse Engineering of CAN Bus Data Using Machine Learning Techniques». In: Jan. 2018, pp. 751–761. ISBN: 978-3-319-69834-2. DOI: 10.1007/978-3-319-69835-9_71 (cit. on p. 29).

[9] Derrick T.R. and Thomas J.M. «Time series analysis: The cross-correlation function». In: *Innovative analyses of human movement: Analytical tools for human movement research* (Jan. 2004), pp. 187–206 (cit. on p. 30).

[10] H. M. Blalock. «Correlated Independent Variables: The Problem of Multicollinearity». In: *Social Forces* 42.2 (1963), pp. 233–237. ISSN: 00377732, 15347605. URL: `http://www.jstor.org/stable/2575696` (cit. on p. 30).

[11] Zhang B. «Regression clustering». In: *Third IEEE International Conference on Data Mining*. 2003, pp. 451–458. DOI: `10.1109/ICDM.2003.1250952` (cit. on p. 34).

[12] Nayak Sarat, Misra Bijan, and Behera Dr. H. «Impact of Data Normalization on Stock Index Forecasting». In: *International Journal of Computer Information Systems and Industrial Management Applications* 6 (Dec. 2014), pp. 357–369 (cit. on p. 35).

[13] Sola J. and Sevilla J. «Importance of input data normalization for the application of neural networks to complex industrial problems». In: *IEEE Transactions on Nuclear Science* 44.3 (1997), pp. 1464–1468. DOI: `10.1109/23.589532` (cit. on p. 35).

[14] Linardi Michele, Zhu Yan, Palpanas Themis, and Keogh Eamonn. «Matrix Profile X: VALMOD - Scalable Discovery of Variable-Length Motifs in Data Series». In: May 2018, pp. 1053–1066. ISBN: 978-1-4503-4703-7. DOI: `10.1145/3183713.3183744` (cit. on p. 35).

[15] J. Abonyi, Balazs Feil, S. Németh, and P. Arva. «Principal Component Analysis based Time Series Segmentation:A New Sensor Fusion Algorithm». In: 2004 (cit. on p. 35).

[16] McCulloch Warren S. and Pitts Walter. «A Logical Calculus of the Ideas Immanent in Nervous Activity». In: *Journal of Symbolic Logic* 9.2 (1944), pp. 49–50. DOI: `10.2307/2268029` (cit. on p. 37).

[17] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: from theory to algorithms*. `http://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning`. Cambridge University Press, 2014. ISBN: 978-1-107-05713-5 (cit. on pp. 37, 46, 47).

[18] Kevin P. Murphy. *Machine Learning: a probabilistic perspective*. MIT Press, 2013 (cit. on p. 37).

[19] G. E. Hinton and R. R. Salakhutdinov. «Reducing the Dimensionality of Data with Neural Networks». In: *Science* 313.5786 (2006), pp. 504–507. DOI: `10.1126/science.1127647`. URL: `https://www.science.org/doi/abs/10.1126/science.1127647` (cit. on p. 37).

[20] Tyler Manning-Dahan. «èCA and Autoencoders». In: (2017) (cit. on p. 37).

[21] Dor Bank, Noam Koenigstein, and Raja Giryes. *Autoencoders*. 2021. arXiv: `2003.05991 [cs.LG]` (cit. on p. 38).

[22] Nam Kounghoon and Wang Fawu. «The performance of using an autoencoder for prediction and susceptibility assessment of landslides: A case study on landslides triggered by the 2018 Hokkaido Eastern Iburi earthquake in Japan». In: *Geoenvironmental Disasters* 6 (Dec. 2019), p. 19. DOI: `10.1186/s40677-019-0137-5` (cit. on p. 38).

[23] Tanaka Yoshiki, Iwamoto Kazuhisa, and Uehara Kuniaki. «Discovery of Time-Series Motif from MultiDimensional Data Based on MDL Principle». In: *Machine Learning - ML* 58 (Feb. 2005), pp. 269–300. DOI: `10.1007/s10994-005-5829-2` (cit. on p. 41).

[24] yıldırım Özal, Tan Ru San, and Acharya U Rajendra. «An Efficient Compression of ECG Signals Using Deep Convolutional Autoencoders». In: *Cognitive Systems Research* 52 (July 2018), pp. 198–211. DOI: `10.1016/j.cogsys.2018.07.004` (cit. on p. 43).

[25] Liu Chien-Liang, Hsaio Wen-Hoar, and Tu Yao-Chung. «Time Series Classification With Multivariate Convolutional Neural Network». In: *IEEE Transactions on Industrial Electronics* PP (Aug. 2018), pp. 1–1. DOI: `10.1109/TIE.2018.2864702` (cit. on p. 43).

[26] Jastrzebska Agnieszka. «Lagged encoding for image-based time series classification using convolutional neural networks». In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 13 (Mar. 2020). DOI: `10.1002/sam.11455` (cit. on p. 43).

[27] Manakov Ilja, Rohm Markus, and Tresp Volker. «Walking the Tightrope: An Investigation of the Convolutional Autoencoder Bottleneck». In: (Nov. 2019) (cit. on p. 43).

[28] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016 (cit. on p. 43).

[29] Kingma Diederik and Ba Jimmy. «Adam: A Method for Stochastic Optimization». In: *International Conference on Learning Representations* (Dec. 2014) (cit. on p. 43).

[30] Ruder Sebastian. «An overview of gradient descent optimization algorithms». In: (Sept. 2016) (cit. on p. 45).

[31] Sibei Yang, Tao Liangde, and Gong Bingchen. «Introduction to Clustering Algorithms and Applications». In: (Aug. 2014) (cit. on p. 45).

[32] Jain A. K., Murty M. N., and Flynn P. J. «Data Clustering: A Review». In: 31.3 (1999). ISSN: 0360-0300. DOI: `10.1145/331499.331504`. URL: `https://doi.org/10.1145/331499.331504` (cit. on p. 46).

[33] Tavakoli Neda, Siami Namini Sima, Khanghah Mahdi, Soltani Fahimeh, and Siami Namin Akbar. «Clustering Time Series Data through Autoencoder-based Deep Learning Models». In: (Apr. 2020) (cit. on p. 46).

[34] Ester Martin, Kriegel Hans-Peter, Sander Joerg, and Xu Xiaowei. «A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise». In: vol. 96. Jan. 1996, pp. 226–231 (cit. on p. 46).

[35] Jin Xin and Han Jiawei. «K-Means Clustering». In: Jan. 2011, pp. 563–564. DOI: `10.1007/978-0-387-30164-8_425` (cit. on p. 47).

[36] Jin Xin and Han Jiawei. «K-Medoids Clustering». In: Jan. 2016, pp. 1–3. DOI: `10.1007/978-1-4899-7502-7_432-1` (cit. on p. 47).

[37] Palacio Niño Julio. «Evaluation Metrics for Unsupervised Learning Algorithms». In: (May 2019) (cit. on p. 47).

[38] Wang Fei, Franco-Penya Hector-Hugo, Kelleher John, Pugh John, and Ross Robert. «An Analysis of the Application of Simplified Silhouette to the Evaluation of k-means Clustering Validity». In: July 2017. ISBN: 978-3-319-62415-0. DOI: `10.1007/978-3-319-62416-7_21` (cit. on p. 49).

[39] Amaia Abanda, Usue Mori, and Jose A. Lozano. *A review on distance based time series classification*. 2018. arXiv: `1806.04509 [stat.ML]` (cit. on p. 57).

[40] Donald J. Berndt and James Clifford. «Using Dynamic Time Warping to Find Patterns in Time Series». In: *KDD Workshop*. 1994 (cit. on p. 57).