POLITECNICO DI TORINO

Dipartimento di Scienze Matematiche
Laurea Magistrale in Ingegneria Matematica

Master of Science Thesis

Predictive maintenance on a Permanent Magnet Synchronous Motor's battery

University advisor:
Tania Cerquitelli
Company advisor:
Davide Mazzucchi

Candidato:
Paolo Pastore

Accademic year 2020-2021

Ai miei nonni,
nonna Maria
e nonno Tonino

Contents

Introduction

The main goal of this paper is to explore Machine Learning (ML) algorithms related to the Predictive Maintenance (PdM) framework in the context of Electric Vehicle (EV).
In process industries, induction motors make up approximately 70% of all driven electrical loads. In this regard, there has been much interest on ways to better diagnose the wellness condition of these motors. Predictive Maintenance is an important maintenance tool in order to identify in advance the anomalies and potential faults.

In this thesis, three principal machine learning algorithms are explored, Support Vector Machine (SVM), XGBoost and Random Forest (RF). Another simple statistical model is performed, the Linear Regression, whose results will be taken as starting point.
The data on which this paper works, don't make available any maintenance information. So, some arguments are carried forward in order to extract from the data which records would have represented a fault in the in Permanent Magnet Synchronous Motor (PMSM).

The first chapter of this book describes the state of art of the PdM framework, referred to in the literature. It presents most of the research that are carried out in this context, highlighting the differences between them.
The second one illustrates the workflow, which algorithms are performed and how they are optimized. It explains that two kind of models are used, re-

gression and classification algorithms, in order to better explain the target variable.
In the third chapter, the results of the models analysed in the second chapter are shown.

Chapter 1

State of the art

To begin exploring the subject matter of this thesis, the state of the art should be approached in order to have a clear picture of what the recent research on this area of industry entails.
It refers to the intelligent monitoring of equipment in order to determine promptly the current damage state, scheduling an optimal maintenance plan to repair it and reduce the machine downtime. Machine learning approaches are viably used in the areas where the availability of data is increasing, such as the maintenance in industry sector (industry 4.0).
Here, it will follow an excursus of the different approches that are proposed for this theme.

1.1 Descriptive papers.

The first paper analysed is called "Predictive Maintenance for Motors Based on Vibration Analysis with Compact Rio" [1].
As the name suggests, it refers to the vibration data in order to take decision on the maintenance of the machine, with no reference to artificial intelligence algorithms.
Rotating machines generate deterministic signals, so the spectral analysis of these signals is necessary for the study of motors. The representation of

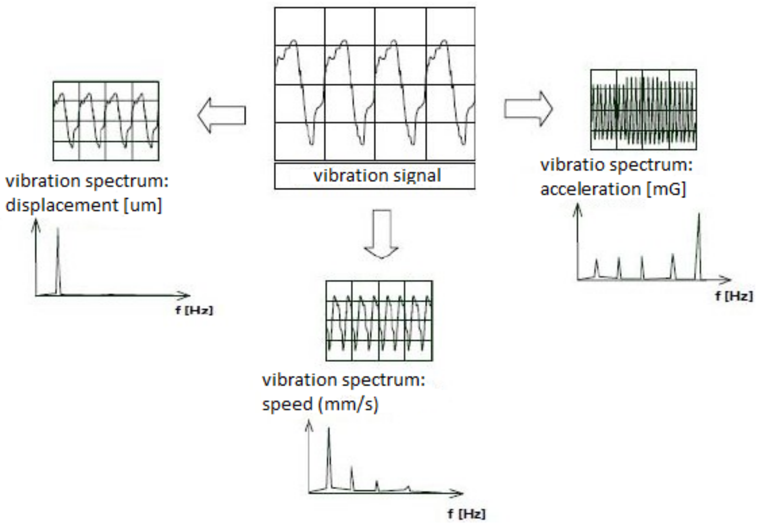
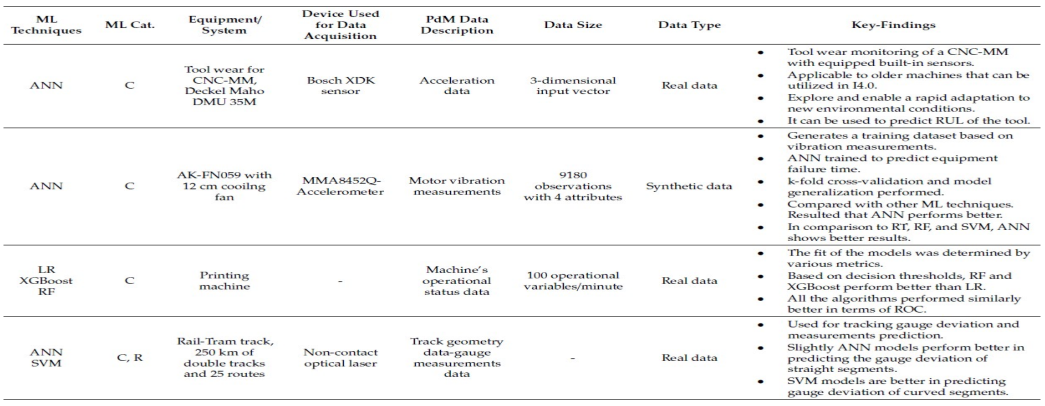


Figure 1.1: Vibration peaks according to measured variables.

the signal in the frequency domain, i.e. its spectrum, facilitates the study of vibrations. Frequency is defined as the number of events produced in a given time. A vibration is defined as a small-amplitude oscillatory motion (see Figure 1.1).

Figure 1.2: Compact Rio and Module 9215 Chassis

Each machine has its own vibration signal, in which information about each of its components is found. This means that an acquired vibration signal is the vector sum of the vibration of each of its components [1].
Vibration analysis refers to the spectral analysis of vibrations in which its har-



ML Techniques	ML Cat.	Equipment/ System	Device Used for Data Acquisition	PdM Data Description	Data Size	Data Type	Key-Findings
ANN	C	Tool wear for CNC-MM, Deckel Maho DMU 35M	Bosch XDK sensor	Acceleration data	3-dimensional input vector	Real data	• Tool wear monitoring of a CNC-MM with equipped built-in sensors. • Applicable to older machines that can be utilized in I4.0. • Explore and enable a rapid adaptation to new environmental conditions. • It can be used to predict RUL of the tool.
ANN	C	AK-FN059 with 12 cm cooilng fan	MMA8452Q-Accelerometer	Motor vibration measurements	9180 observations with 4 attributes	Synthetic data	• Generates a training dataset based on vibration measurements. • ANN trained to predict equipment failure time. • k-fold cross-validation and model generalization performed. • Compared with other ML techniques. Resulted that ANN performs better. • In comparison to RT, RF, and SVM, ANN shows better results.
LR XGBoost RF	C	Printing machine	-	Machine's operational status data	100 operational variables/minute	Real data	• The fit of the models was determined by various metrics. • Based on decision thresholds, RF and XGBoost perform better than LR. • All the algorithms performed similarly better in terms of ROC.
ANN SVM	C, R	Rail-Tram track, 250 km of double tracks and 25 routes	Non-contact optical laser	Track geometry data-gauge measurements data	-	Real data	• Used for tracking gauge deviation and measurements prediction. • Slightly ANN models perform better in predicting the gauge deviation of straight segments. • SVM models are better in predicting gauge deviation of curved segments.

Figure 1.3: The PdM topics are associated to the best machine learning techniques

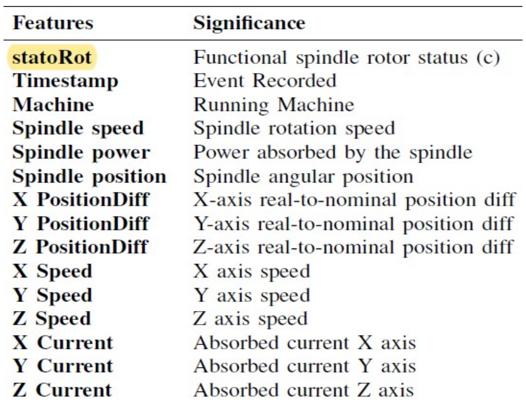
monic components are considered. The reconfigurable NI cRIO-9074 chassis is the instrument used (see Figure 1.2). It was used with intelligent real-time controller for CompactRio.

Another interesting document is the one cited in [3], which assign for each sector of manufacturing industry 4.0 a possible good Machine learning algorithm for predictive maintenance. The paper classifies the research according to the ML algorithms, ML category, machinery, and equipment used, device used in data acquisition, classification of data, size and type, and highlight the key contributions of the researchers, and thus others guidelines and foundation for further research, see Figure 1.3.

1.2 Papers that use Machine Learning Techniques

An important document cited in this thesis is called "Machine Learning-based for Predictive maintenance in industry 4.0" [2].

The first part explains how a PdM algorithm can be performed and what kind of data we need as an input. It explains that Machine Learning-based



Features	Significance
statoRot	Functional spindle rotor status (c)
Timestamp	Event Recorded
Machine	Running Machine
Spindle speed	Spindle rotation speed
Spindle power	Power absorbed by the spindle
Spindle position	Spindle angular position
X PositionDiff	X-axis real-to-nominal position diff
Y PositionDiff	Y-axis real-to-nominal position diff
Z PositionDiff	Z-axis real-to-nominal position diff
X Speed	X axis speed
Y Speed	Y axis speed
Z Speed	Z axis speed
X Current	Absorbed current X axis
Y Current	Absorbed current Y axis
Z Current	Absorbed current Z axis

Figure 1.4:

PdM can be divided into the following two main classes: supervised, where information on the occurrence of failures is present in the modelling dataset and unsupervised, where logistic and/or process information is available, but no maintenance data exists. In the second part, some details of the techniques used are given.

The historical data used to feed the machine learning algorithm is visible in figure 1.4, where the *statoRot* is the state of the main spindle rotor. This feature was treated as the target variable.
Maintenance management has been achieved by training a Random Forest approach on a cutting machine much used in the wood industry (see Figure 1.5).
The results showed a proper behavior of the approach on predicting different machine states (Fault/no Fault) with high accuracy (95%) on a data set of 530731 data readings.

Sometimes it's necessary to develop regression instead classification algorithms in order to do predictive maintenance [5]. This paper proposes a data science approach with embedded statistical data mining and a machine learning algorithm to predict the remaining useful life (RUL) of the rotary bearings in a motor. So the label column is a continuous variable and the methods used to make prediction were:

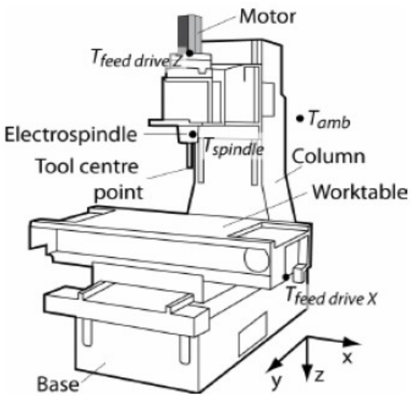
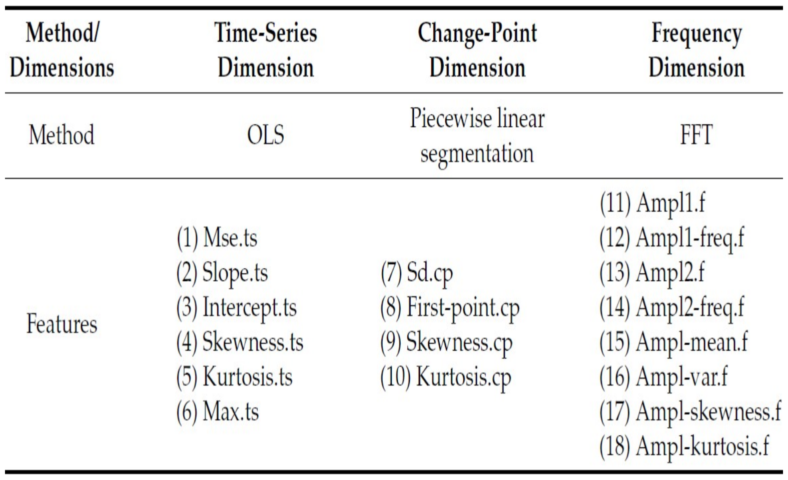


Figure 1.5: Example of the machine adopted in this paper.

- Ordinary Least Squares (OLS);
- Feasible Generalized Least Squares (FGLS), statistical method useful when the hypotesis of homoscedasticity is rejected;
- Support Vector Regression (SVR).



Method/ Dimensions	Time-Series Dimension	Change-Point Dimension	Frequency Dimension
Method	OLS	Piecewise linear segmentation	FFT
Features	(1) Mse.ts (2) Slope.ts (3) Intercept.ts (4) Skewness.ts (5) Kurtosis.ts (6) Max.ts	(7) Sd.cp (8) First-point.cp (9) Skewness.cp (10) Kurtosis.cp	(11) Ampl1.f (12) Ampl1-freq.f (13) Ampl2.f (14) Ampl2-freq.f (15) Ampl-mean.f (16) Ampl-var.f (17) Ampl-skewness.f (18) Ampl-kurtosis.f

Figure 1.6: Features extraction from the time domain and the frequency domain

The vibration data is collected from an operational real-world induction motor. The features are extracted from the time domain and the frequency domain, using three dimensions, as showed in the Figure 1.6. The features “Mse.ts”, “Intercept.ts”, “First-point.cp”, “Ampl1.f”, “Ampl-mean.f” and “Ampl-Kurtosis.f” were removed because of their high correlation with the other features, and it was used the Stepwise regression method in order to select the most important features that have been identified by “Kurtosis.ts”, “Max.ts”, “Ampl1-freq.f”, “Ampl2.f” and “Ampl2-freq.f. The outcomes are showed in figure see Figure 1.7.

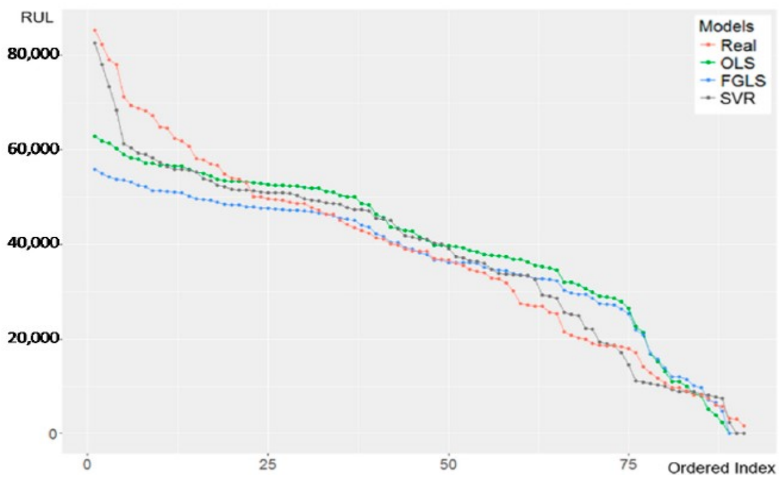


Figure 1.7: Results of the three models against the real values of the RUL.

It can be observed that SVR perfomed better than the other two methods but it has to be said that the interpretabiility is lower and the computation time is higher.

1.3 Papers that use Neural Network methods

The aim of the work [4] is to prevent fault progression and protect vital components of the power system by early detection of electrical faults of three phase induction motors using artificial neural network.

This time a multiclass classification Artificial Neural Network (ANN) has developed, since the target variable has represented by seven classes of elec-

Classes	V_1	V_2	V_3	I_1	I_2	I_3	Fault
1	2.661	2.624	2.701	0.491	0.479	0.493	No fault (1)
1	2.660	2.625	2.701	0.491	0.479	0.492	
2	2.648	2.599	2.672	0.006	0.643	0.640	Overload (2)
2	2.650	2.601	2.674	0.006	0.641	0.639	
3	0.920	2.621	2.627	0.172	0.772	0.663	Graund fault (3)
3	0.920	2.622	2.625	0.172	0.772	0.663	
4	1.875	1.855	1.875	0.287	0.287	0.281	Locked rotor (4)
4	1.453	1.450	1.442	0.245	0.250	0.234	
5	2.866	2.872	2.855	0.482	0.500	0.497	Unbalanced voltage (5)
5	2.869	2.876	2.860	0.484	0.500	0.497	
6	2.658	2.614	2.688	1.671	1.650	0.669	Single phasing, under voltage (6)
6	2.661	2.613	2.689	1.417	1.398	1.411	
7	2.638	2.600	2.674	0.803	0.783	0.798	Overvoltage (7)
7	2.650	2.608	2.683	0.857	0.838	0.848	

Table 1.1: Input data to feed the Neural Network

trical faults of induction motors; overload, ground fault, locked rotor, single phasing, over voltage, under voltage and unbalanced supply voltage.

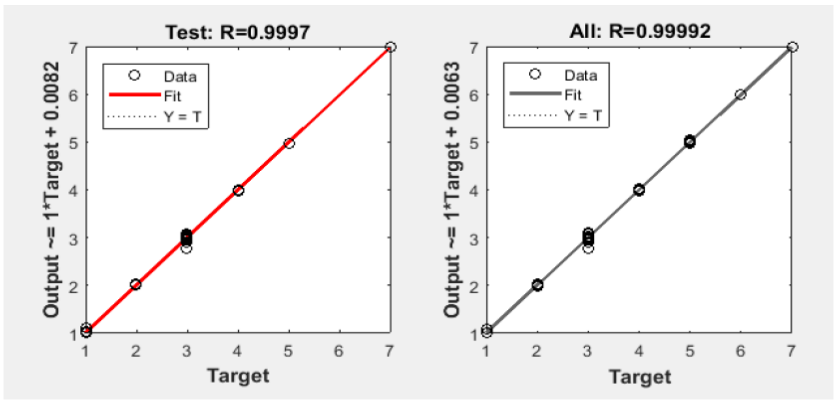


Figure 1.8: Results of the ANN model

A feed forward back loop neural network algorithm is performed by us-

	Parameters	**Units**
Var1	Vibration speed	m/s
Var2	Motor Torque	Nm
Var3	Acceleration	mm^2/s
Var4	Motor Speed	Hz/s
Var5	Air Pressure	bar
Var6	Product Weight	hg
Var7	Deceleration	mm^2/s
Var8	Current	A(Amps)
Var9	Belt tension	N/m
Var10	Motor tension	N/m
Var11	Temperature	*C

Table 1.2: Time-Series dataset variables

ing three phase voltages and currents as input data, as it can be seen in Table 1.1. After a little explanation of how the ANN works, the results are showed and all the test data are classified accurately, as the Figure 1.8 shows.

The last paper analysed [6] is a complex one, in the sense that it explores an innovative way to handle multivariate data. It is built a classification model using a combination of time-series imaging and Convolutional Neural Network (CNN) for Predictive Maintenance of conveyor motors.
In this research, time-series represent different observations recorded from the machine over time. The framework is designed to accommodate multivariate time-series as inputs of the model.
The experimental data is composed of 11 parameters (Vibration speed, Motor torque, Acceleration, Motor Speed, Air pressure, Product Weight, Deceleration, Current, Belt tension, Motor tension, Temperature, see Table 1.2) and one outcome which is the type of Fault detected in the system .Three states of the conveyor motor: (1) No-Fault (2) Minor Fault and (3) Critical Fault with urgent need of maintenance.

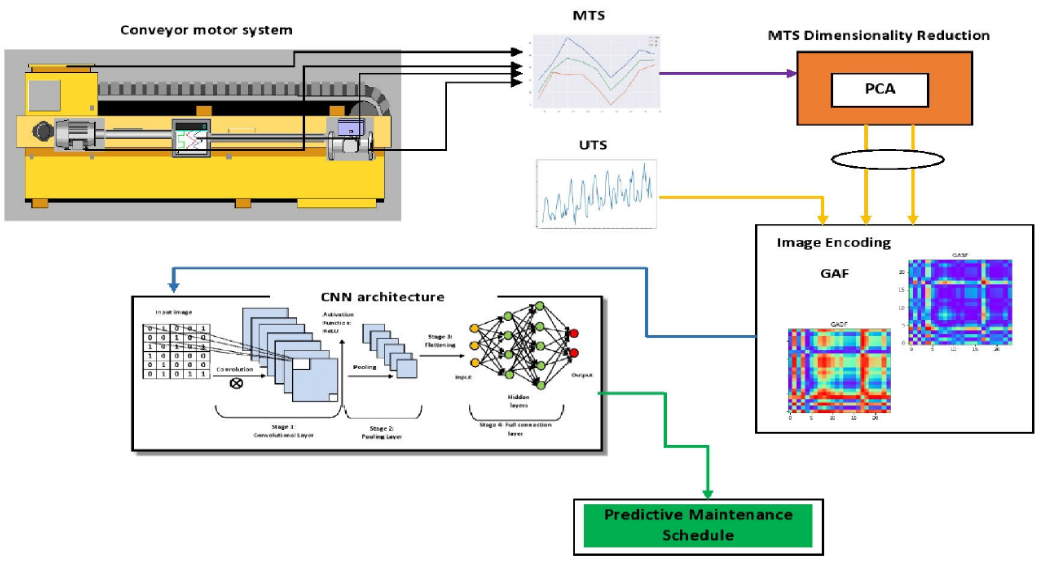


Figure 1.9: Overall system's architecture

As well explained in the Figure 1.9, the time-series are encoded into images via the Gramian Angular Field (GAF) method by using the poolar coordinates and used as inputs to feed the Convolution Neural Network model. It is achieved an overall accuracy of 100% by using the CNN + Rectified Linear Uni (ReLU) model, and only 55% by performing Support Vector Machine (SVM), see the Confusion Matrixs in Table 1.3.

SVM	**CF**	**MF**	**NF**
CF	612	169	224
MF	149	181	706
NF	0	96	864
CNN + ReLu	**CF**	**MF**	**NF**
CF	1000	0	0
MF	0	1000	0
NF	0	0	1000

Table 1.3: Confusion matrix result of the two models

Chapter 2

Proposed methodology

In this chapter, the steps of the work done will be explored and described. As usually the preprocessing process is analyzed first.

2.1 Explorative analysis

Advanced explorative analysis of the datasets through statistical techniques and their interpretation is carried out, i.e. each signal (feature[1]) is analyzed to understand how to synthesize them.
The raw data will pass through two principal steps to be readily used as input of the machine learning models:

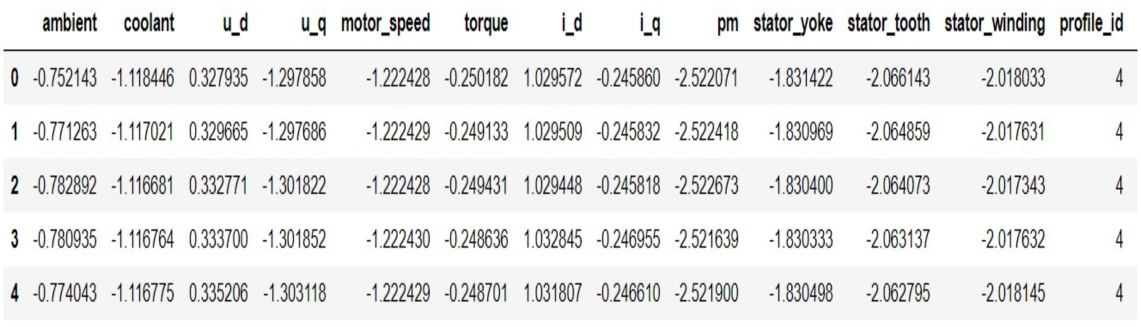
1. **Data Cleaning**. In this process, the data are prepared for analysis by removing or modifying those which are incorrect or incomplete. Moreover, the variables to be taken into consideration are selected, by considering only one of a group of features highly correlated. This improves the quality of the training data for analytics and enables accurate decision-making [20], [21].

2. **Feature extraction**. In this process an initial set of raw data, showed in Figure 2.1, is reduced to more manageable groups for processing [10].

[1]A feature is an attribute used to feed the Machine Learning algorithm. Feature, predictor, attribute, and variable have to be meant as the same.

First, the data are aggregated in different sessions with a similar period of time, second, it is extracted from each feature the most common statistics (the quality and quantity of features are key determinants which strongly influence the outcome of the prediction [11]). In this way, we will obtain a data set with more predictors but with a much fewer number of records.

2.1.1 Origin of the data



	ambient	coolant	u_d	u_q	motor_speed	torque	i_d	i_q	pm	stator_yoke	stator_tooth	stator_winding	profile_id
0	-0.752143	-1.118446	0.327935	-1.297858	-1.222428	-0.250182	1.029572	-0.245860	-2.522071	-1.831422	-2.066143	-2.018033	4
1	-0.771263	-1.117021	0.329665	-1.297686	-1.222429	-0.249133	1.029509	-0.245832	-2.522418	-1.830969	-2.064859	-2.017631	4
2	-0.782892	-1.116681	0.332771	-1.301822	-1.222428	-0.249431	1.029448	-0.245818	-2.522673	-1.830400	-2.064073	-2.017343	4
3	-0.780935	-1.116764	0.333700	-1.301852	-1.222430	-0.248636	1.032845	-0.246955	-2.521639	-1.830333	-2.063137	-2.017632	4
4	-0.774043	-1.116775	0.335206	-1.303118	-1.222429	-0.248701	1.031807	-0.246610	-2.521900	-1.830498	-2.062795	-2.018145	4

Figure 2.1: Summary of the raw data

The data on which this thesis is based, was taken from the Kaggle platform and consists of 140 hours recordings from a Permanent Magnet Synchronous Motor (PMSM) [7]. PMSMs are brushless and have very high reliability and efficiency. Due to their permanent magnet rotor, they also have higher torque with smaller frame size and no rotor current, all of which are advantages over AC Induction Motors (AICMs) [12]. Synchronous motors contain multiphase Alternating Current electromagnets on the stator of the motor that creates a magnetic field that rotates in time with the oscillations of the line current.
This kind of motors are widely used in robotics, machine tools, actuators, and they have been considered to be used in high-power applications such as industrial drives and vehicular propulsion [13].
The first five rows of the dataset are shown in Figure 2.1: it is composed of

998070 rows and 15 time series variables.
Each predictor can be detailed as follows:

- *ambient*: temperature as measured by a thermal sensor located closely to the stator
- *coolant*: coolant temperature. The motor is water cooled. Measurment is taken at outflow.
- *u_d*: voltage d-component
- *u_q*: voltage q-component
- *motor_speed*: the speed achieved by the motor;
- *torque*: torque induced by current
- *i_d*: current d-component
- *i_q*: current q-component
- *pm*: permanent Magnet surface temperature representing the rotor temperature. This was measured with an infrared thermography.
- *stator_yoke*: stator yoke temperature measured with a thermal sensor.
- *stator_tooth*: stator tooth temperature measured with a thermal sensor.
- *stator_winding*: stator winding temperature measured with a thermal sensor.
- *profile_id*: to identified Distinctive sessions. Each measurement session has a unique ID.
- *i_s*: the current vector defined as $\sqrt{{i_d}^2 + {i_q}^2}$.
- *u_s*: the voltage vector defined as $\sqrt{{u_d}^2 + {u_q}^2}$.

The last two features are added later [7].

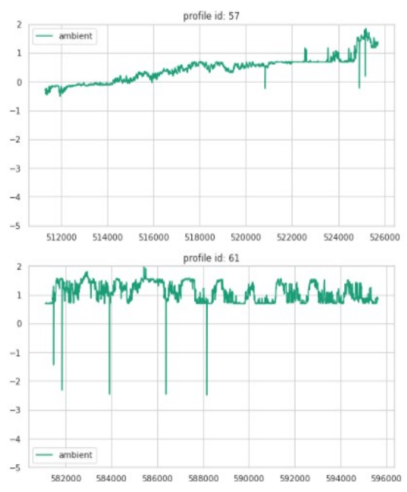


Figure 2.2: Different signals of the ambiental temperature. The downward peaks represent outliers

2.1.2 Outliers detection

When you work on a huge amount of data, an important process is detecting the outlier value, i.e. those data whose values are clearly a mistake in typing. Let's observe in Figure 2.2 the signal of the ambiental temperature for four different sessions. It seems necessary to remove all those values below a certain treshold. Infact it is not reasonable to think that the external temperature could have such sudden drops.
Even if it is known that the feature *ambient* is measured by a sensor close to the stator, it's evident from Figure 2.3 there are no common relationships between these signals. For some sessions and some points, the downward peaks (green line) are linked to the downward trend of the *stator_yoke* and *pm* features (orange and violet lines), see the central zone of the lower left image, for others, they are associated to increase values in both these temperatures, see the central zone of the upper right image.

To remove these values, it is adopted the following idea: collecting the signals (treated as independent for each sessions) in groups, let $x_{i,j}$ be the *ambient* value of the group i of the sessions j. For each of them, taking the

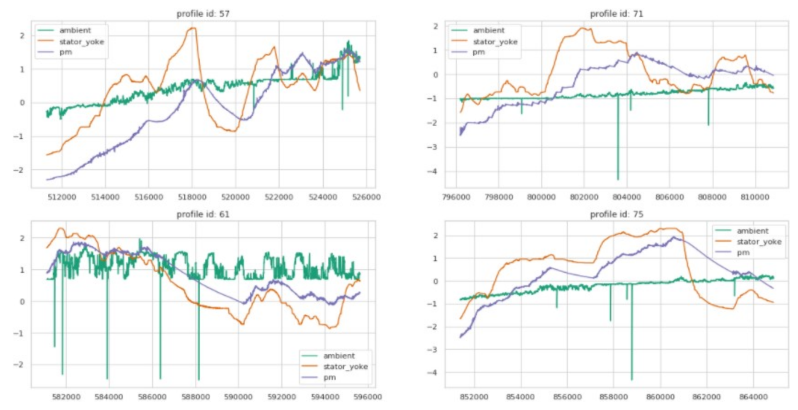


Figure 2.3: Relationship between the ambient (green line), stator (orange line) and rotor temperatures (violet line).

mean ($\bar{x}_{i,j}$) value and removing those values that were out of the interval:

$$(-0.5 + \bar{x}_{i,j}; \bar{x}_{i,j} + 0.5) \quad \forall\, i\ group \quad and \quad \forall\, j\ session.$$

After that, replace these values with the ones that will be obtained using the pre-built *interpolate* function of Python.

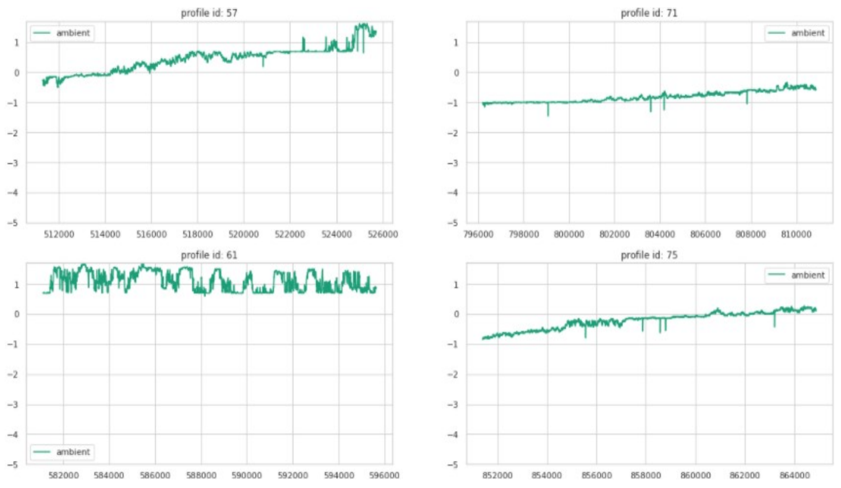


Figure 2.4: *Ambient* signals with outlier detected

It can be stated, by observing Figure 2.4 compared to Figure 2.2, the idea worked and the outliers have been removed.

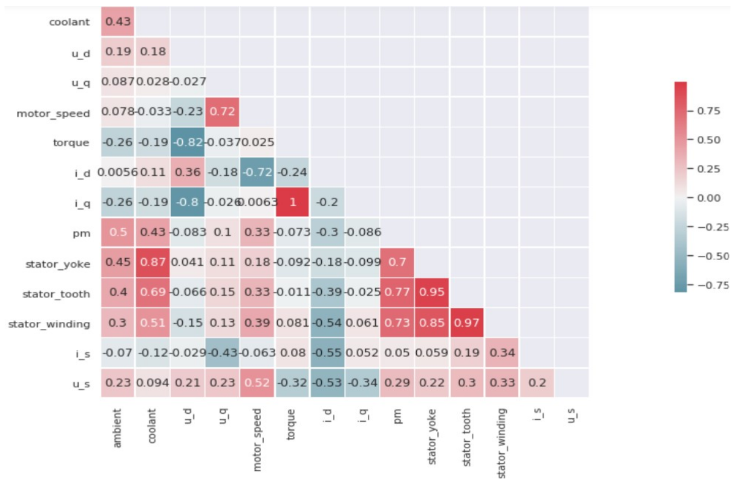


Figure 2.5: Correlation Matrix on the raw data

2.1.3 Target variable

At this point, it is important to have clear in mind which is the target to achieve in this work.
The dataset just explored, had as the initial goal the one to best predict the different stator and rotor temperatures. Now, you want to construct some models in order to be able to predict the status of the battery and if it needs some maintenance works. As can be seen from the description above, there is no label column about the health status of the battery. Moreover, there is no access to the original data, they have been given already standardized.
Having in mind that, some considerations are made. One could suppose that the extreme values of a certain variable are a symptom of a malfunction of it. So if it could be possible to understand from the dataset which are the anomalous values for the voltage in the battery, one could link these values to eventual damage.

Now, since each record of the dataset is equal to 0.5 seconds, it cannot be said that one instant of under or over voltage corresponds to a damage of the battery, but if this trend persists, then maintenance work is suggested.

Then the next step was to aggregate the data into a certain interval of time, and afterward to observe the distribution of the vector voltage u_s to understand which values have to be considered as extreme ones (the outlier values will be taken into account).

2.1.4 Features selection

In this first step, the focus is on reducing the dimensionality of the predictors' space by looking at the correlation matrix [18].
The Correlation matrix in Figure 2.5 shows that there is a significant correlation between all the three different stator temperatures. The lineplots in Figure 2.6 confirm that all three temperatures follow the same trend.
The stator winding temperatures shows the biggest variation followed by the stator tooth and stator yoke temperature.

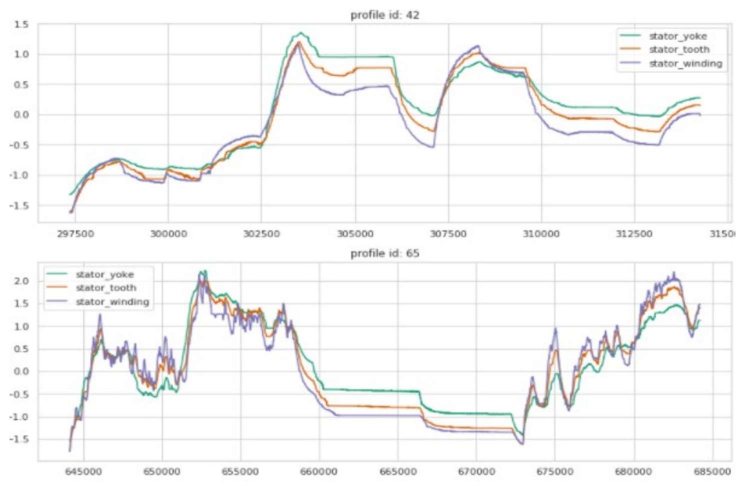


Figure 2.6: Correlation between the three temperatures

This is especially noticeable when there is a lot of variation in the stator winding temperature. If this is the case, the stator tooth and yoke temperatures follow a smoother path than the temperature recorded on the stator winding. In other words, the heat dissipated by the stator windings takes some time to heat the stator tooth and yoke due to the thermal inertia of both stator parts.

A very high positive linear correlation was observed between i_q and torque, and the q-component of the voltage u_d is highly negative linearly correlated with torque and i_q. The Figure 2.7 confirm this sentence: the i_q and *torque* curves are almost overlapped, compared to the u_q curve that follows almost an opposite trend.

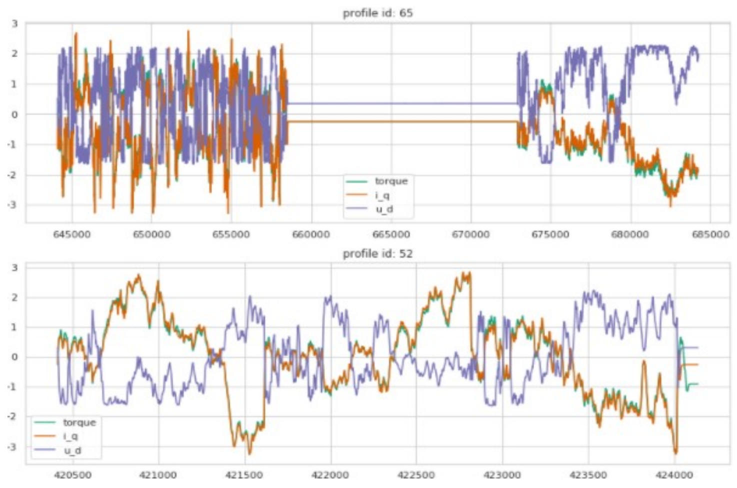


Figure 2.7: Relationship between i_q, u_d and torque.

Indeed, for the former insight, we can refer to electric drive theory, where either higher torque is exclusively dependent on i_q in case of similar-sized inductances in d- and q-axis, or increasing with higher i_q and slightly decreasing i_d else wise (more common in practice).

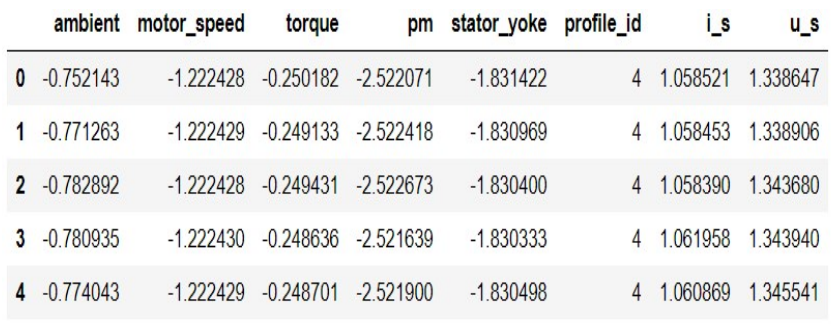
There is an almost high positive linear correlation between the temperatures and coolant. Moreover the features u_s and i_s are not highly linear correlated with respectively u_d, u_q and i_d, i_q, but we know that they have a strong quadratic correlation, since their definition.

Then, by fixing the threshold at 80% for considering a pair of predictors highly correlated, we proceed as follows:

- The feature *torque* is kept between i_q, *torque*, and u_q;
- u_d, u_q and i_d, i_q, are highly correlated with respectively u_s and i_s: The last two are kept;
- It is kept only the *stator_yoke* between the stator's temperatures;

- *Coolant* is removed because high correlated to *stator_ yoke*.

This choices brought to have the dataset in Figure 2.8, with 8 columns and the same number of rows of the initial dataset. It's important to observe that the



	ambient	motor_speed	torque	pm	stator_yoke	profile_id	i_s	u_s
0	-0.752143	-1.222428	-0.250182	-2.522071	-1.831422	4	1.058521	1.338647
1	-0.771263	-1.222429	-0.249133	-2.522418	-1.830969	4	1.058453	1.338906
2	-0.782892	-1.222428	-0.249431	-2.522673	-1.830400	4	1.058390	1.343680
3	-0.780935	-1.222430	-0.248636	-2.521639	-1.830333	4	1.061958	1.343940
4	-0.774043	-1.222429	-0.248701	-2.521900	-1.830498	4	1.060869	1.345541

Figure 2.8: Dataset after features selection.

predictors' space is composed of the statistics of three different temperatures (ambient, rotor, and stator), the speed, and the torque achieved by the motor and the current vector.

2.1.5 Aggregate the data

The Plot in Figure 2.9 shows that sessions' time is not the same and it is between almost 20 min to around 6 hours. The short sessions with ids "47", "46" might be not very representative as temperatures inside electric motors need time to vary. It can understand that longer sessions are more reliable as well as should properly consider in both train and testing.
So when the signals are analyzed for splitting the dataset into the same period, care must be taken in considering each session independent of each other.

After a lot of experimental trials, it was chosen to aggregate the data every 1850 rows, which are more or less equivalent to 15 minutes. To checking if this choice is reasonable for all the signals and all the profiles, some pictures are shown. Figure 2.10 shows the results of two signals: the permanent

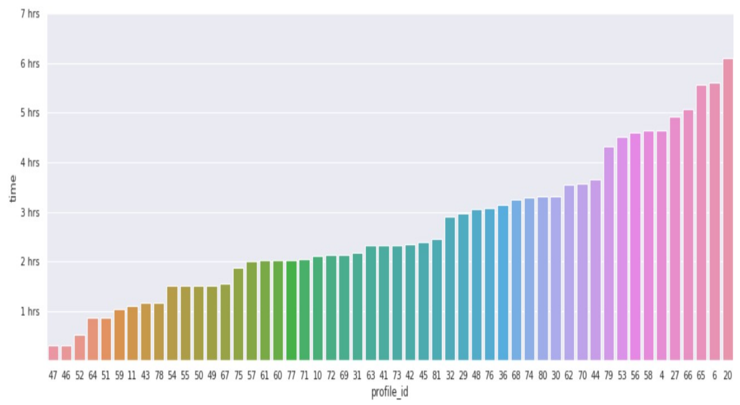


Figure 2.9: Differerent sessions' time

magnet temperature (*pm*) for three independent sessions, and the *torque*. The blue lines represent the *pm* signal and the vertical green dashed lines are 15 minutes away from each other.
The lines have clearly captured the smooth central zone. The left and right sides instead are broken into signals with a similar pattern.
Let's focus the attention on the third picture starting from the top (*pm* signal of the profile id 66). The broken lines on the left side present a small initial constant part, then a rapid growth, and finally a new small settling zone. The right side, on the other hand, has divided the signal into small parabolas with concavity upwards. Similar reasonings can be done for the other graph. Then, the next steps are summarized here.

- Aggregate the data in intervals of 15 minutes, by computing for each feature (except for the label u_s, where only the average is computed) the following statistics: *minimun, maximum, mean, root mean square, kurtosis, skewnees.* The first three are well explained itself, while the others are defined as follows:
  - *Root Mean Square* (RMS): the square root of the arithmetic mean square of the signal, and it represents the average power:

$$RMS = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}$$

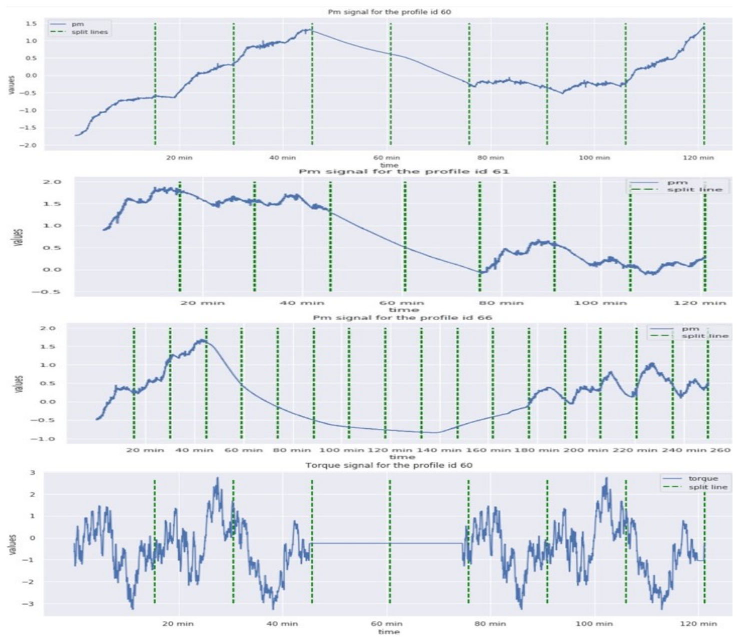


Figure 2.10: *Pm* signals for three independent sessions and the *torque* signal splitted into 15 minutes.

– *Skewness*: it characterizes the degree of asymmetry of a distribution around its mean. Positive skewness indicates a distribution with an asymmetric tail extending toward more positive values. Negative skewness commonly indicates that the tail is on the left side of the distribution, and positive skew indicates that the tail is on the right [9]. Let's define μ, σ respectively the mean and the std of the signal, then the *skewness* remains defined as follows:

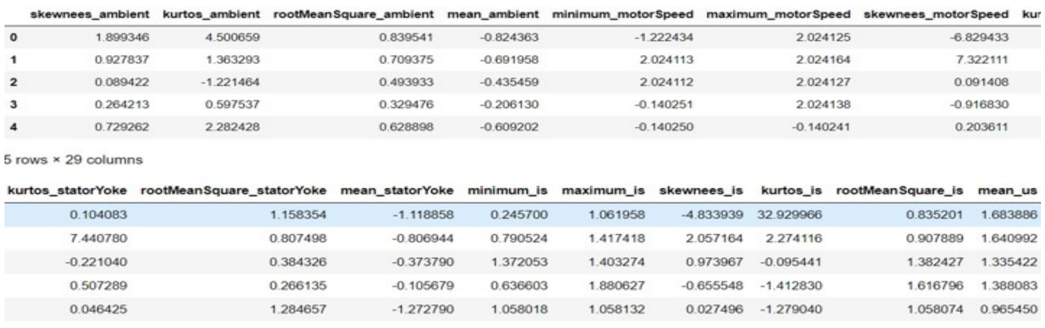
$$Skewness = \frac{1}{n}\frac{\sum_{i=1}^{n}(x_i-\mu)^3}{\sigma^3}$$

– *Kurtosis*: is a measure of the "tailedness" of the probability distribution of the signals.For this measure, higher kurtosis corresponds to greater extremity of deviations (or outliers), and not the configuration of data near the mean [8].

$$Kurtosis = \frac{1}{n}\frac{\sum_{i=1}^{n}(x_i-\mu)^4}{\sigma^4}$$

- Analyzed again the confusion matrix on those new features and take only one feature between the pairs highly correlated.

The final dataset containing the aggregated data on which the ML algorithms described in the next chapters will be trained, is showed in Figure 2.11.



	skewnees_ambient	kurtos_ambient	rootMeanSquare_ambient	mean_ambient	minimum_motorSpeed	maximum_motorSpeed	skewnees_motorSpeed	kur
0	1.899346	4.500659	0.839541	-0.824363	-1.222434	2.024125	-6.829433	
1	0.927837	1.363293	0.709375	-0.691958	2.024113	2.024164	7.322111	
2	0.089422	-1.221464	0.493933	-0.435459	2.024112	2.024127	0.091408	
3	0.264213	0.597537	0.329476	-0.206130	-0.140251	2.024138	-0.916830	
4	0.729262	2.282428	0.628898	-0.609202	-0.140250	-0.140241	0.203611	

5 rows × 29 columns

kurtos_statorYoke	rootMeanSquare_statorYoke	mean_statorYoke	minimum_is	maximum_is	skewnees_is	kurtos_is	rootMeanSquare_is	mean_us
0.104083	1.158354	-1.118858	0.245700	1.061958	-4.833939	32.929966	0.835201	1.683886
7.440780	0.807498	-0.806944	0.790524	1.417418	2.057164	2.274116	0.907889	1.640992
-0.221040	0.384326	-0.373790	1.372053	1.403274	0.973967	-0.095441	1.382427	1.335422
0.507289	0.266135	-0.105679	0.636603	1.880627	-0.655548	-1.412830	1.616796	1.388083
0.046425	1.284657	-1.272790	1.058018	1.058132	0.027496	-1.279040	1.058074	0.965450

Figure 2.11: First five rows of the final aggregated dataset.

The dataset consists of 565 records and 28 predictors and with one label which is called, making use of a lot of imagination, '*mean_us*'. It represents the vector voltage in the battery averaged every fifteen minutes on which to compute regression and classification algorithms:

- Multiple linear regression, XGBoost and Random Forest for the regression;
- Support Vector Machine, XGBoost and Random Forest for the classification.

To discretize the label '*mean_us*', we can have a look at the box plot in Figure 2.12. The vertical right (left) line instead represents the values equal to 1.5*range interquartile plus (minus) the third (the first) quartile. The values outside these lines could be some outliers or, in our case, potential values representing a fault in the work of the battery. The goal is to predict anomalies in the battery. If you are able to predict the outlier values, maybe you can figure out some faults, and make some Predictive Maintenance on the battery.

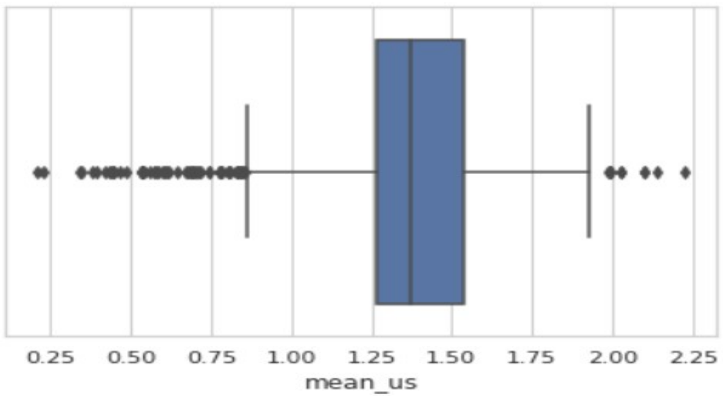


Figure 2.12: Box plot of the label column

Then, the mean voltage vector feature ('*mean_us*') can be discretized into five classes:

- class 0: the values less than 0.2;
- class 1: the values between 0.85 and 1.25;
- class 2: the values between 1.25 and 1.53;
- class 3: the values between 1.53 and 2.3;
- class 4: the values greater than 2.3.

In Figure 2.13 a histogram of the class distribution is shown. Since there are only 9 values of class 4, also less when we split them into training and test sets, class 4 was merged with the class 0. In this way, class 0 means "**Damaged**" battery, since it could represent both overload and underload voltage of the battery. Class 1 is the "**Medium Damage**" battery, class 2 is "**Operative**" battery and class 3 is "**New**" battery.
It is important to say that if one is able to predict with high performances the exact continuous values (reached by performing regression algorithm) assumed from the variable *mean_us*, a domain expert could say, from the high one of his experience, if the PMSM needs some maintenance works. By discretizing the label instead, the expert will be able to take a decision by

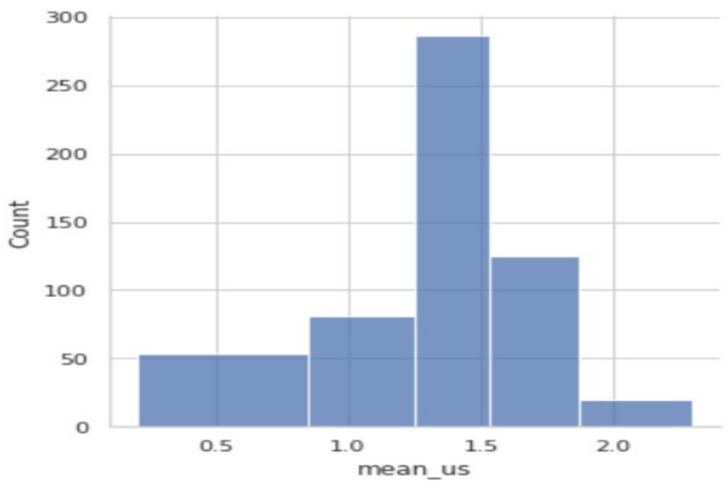


Figure 2.13: Histogram of the classes distribution

knowing that the battery has a mean voltage belonged to a set of values too high or too low.

2.2 Machine Learning algorithms

This section will be explored the algorithms that are used in this thesis and how they are developed.
For each algorithm, the following steps are taken into consideration:

- the data are split into train and the test set (respectively 80% and 20% of the entire set of data) for measuring the performances and, when Cross Validation is not performed, the train set is in turn divided into train and validation for tuning the parameters;
- the metrics have been computed by averaging on a different split of data, in order to have reliable estimations of such measurements;
- a random state is set at each step above for reproducible results.

Moreover, three models for each algorithm (both regression and classification ones) are trained: the first by using the given default values, the second with an optimization method for searching at the best Hyperparameters, and the third one, which is reduced the dimensionality of the features space by looking at the importance of the features and taking the ones whose cumulative percentage was above 95

2.2.1 Multiple Linear regression

One of the simple approach for predicting a quantitative response $\mathbf{Y}$ on the basis of a multiple predictor $\mathbf{X}$. Let's suppoose p the number of distinct predictors, is the Multiple Linear Regression. As the name suggests, it assumes that there is approximately a linear relationship between $\mathbf{Y}$ and $\mathbf{X}$ and can be explicitated by the following equation

$$\mathbf{Y} = \beta_0 + \beta_1\mathbf{X}_1 + \beta_2\mathbf{X}_2 + \ldots + \beta_p\mathbf{X}_p + \epsilon, \tag{2.1}$$

where β is the unknown coefficient vector and ϵ is the not observable error vector distribuited as multivariate gaussian with

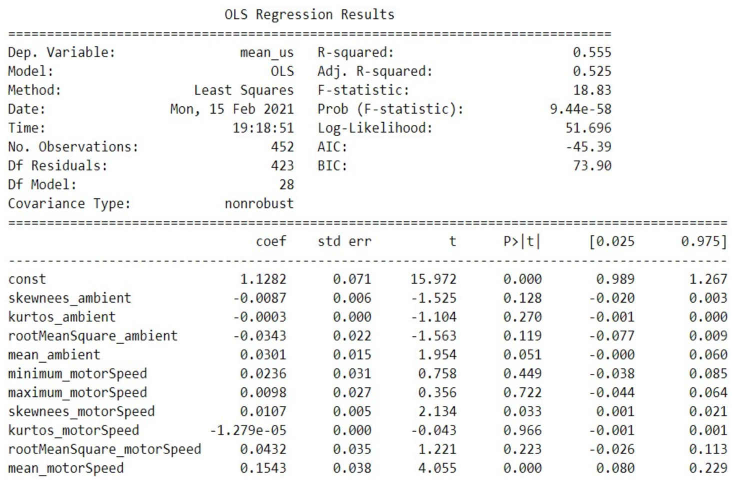
$$\mathbb{E}[\epsilon] = 0 \quad and \quad VarCov(\epsilon) = \sigma^2\mathbf{I}.$$

The fact that $\epsilon_1, \epsilon_2, \ldots, \epsilon_n$ are Indipendent Identically Distribuited (IID) is called homoskedasticity ([14]). In this work the coefficients β_i $\forall i = 0, \ldots, p$ are estimated by the Ordinary Least Squares (OLS), that is by minimizing the Residual Sum of Squares (RSS), defined as

$$RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \hat{\beta}_2 x_{i2} + \ldots + \hat{\beta}_p x_{ip})^2$$

where the $\hat{\beta}_i$ are the estimates of the true coefficients β_i.

In this work, the dependent variable $\mathbf{Y}$ is the column *mean_us*, while the predictors' matrix $\mathbf{X}$ is composed of all the other features. The trained linear model regression is shown in Figure 2.14.
It is common, trying to improve the linear model by removing one, more or even all the predictors.



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                mean_us   R-squared:                       0.555
Model:                            OLS   Adj. R-squared:                  0.525
Method:                 Least Squares   F-statistic:                     18.83
Date:                Mon, 15 Feb 2021   Prob (F-statistic):           9.44e-58
Time:                        19:18:51   Log-Likelihood:                 51.696
No. Observations:                 452   AIC:                            -45.39
Df Residuals:                     423   BIC:                             73.90
Df Model:                          28
Covariance Type:            nonrobust
============================================================================================
                              coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------
const                       1.1282      0.071     15.972      0.000       0.989       1.267
skewnees_ambient           -0.0087      0.006     -1.525      0.128      -0.020       0.003
kurtos_ambient             -0.0003      0.000     -1.104      0.270      -0.001       0.000
rootMeanSquare_ambient     -0.0343      0.022     -1.563      0.119      -0.077       0.009
mean_ambient                0.0301      0.015      1.954      0.051      -0.000       0.060
minimum_motorSpeed          0.0236      0.031      0.758      0.449      -0.038       0.085
maximum_motorSpeed          0.0098      0.027      0.356      0.722      -0.044       0.064
skewnees_motorSpeed         0.0107      0.005      2.134      0.033       0.001       0.021
kurtos_motorSpeed       -1.279e-05      0.000     -0.043      0.966      -0.001       0.001
rootMeanSquare_motorSpeed   0.0432      0.035      1.221      0.223      -0.026       0.113
mean_motorSpeed             0.1543      0.038      4.055      0.000       0.080       0.229
```

Figure 2.14: Linear regression model on the entire dataset

The first thing on which focus the attention is the value called 'Prob(F-statistic)'. It represents the probability associated with the hypothesis that this model is equal to the null one (the model with only the intercept). It can be seen, from Figure 2.14,that such value is so small ($= 4.4 * 10^{-58}$) that the hypothesis is rejected.
Going forward, it can be made inference on the coefficients $\hat{\beta}_i$. In fact, each coefficient has an associated column named '$P > |t|$', that is the p_{value} associated with the null hypothesis $H_0 : \hat{\beta}_i = 0$.
It derives from the fact that under the null hypothesis, the statistic

$$t = \frac{\hat{\beta}_i}{\sqrt{MSE(\mathbf{X}^T\mathbf{X})^{-1}}} \sim t(n-p)$$

where $t(n-p)$ is the t-student with $n-p$ degrees of freedom, the $p_{value} = \mathbb{P}(t(n-p) > t_{oss})$.
Then, it can be taken the choice of rejecting the null hypothesis with a confidence of 5% if those values are less than 0.05, and don't do that for the ones greater.

For example, one could think to remove *skewnees_ ambient* from the predictors' space, and keep *mean_ motorSpeed.* It is too important to say that

whenever the model changes (even if by removing only one feature) the p_{value} column will be different.
Then for example, if it is removed from the model in Figure 2.14 *kurtos_ambient* cause a higher p_{value} than 0.05, the feature *skewnees_ambient* could have (or not) a p_{value} less than 0.5.

An optimal search on the predictor space will be performed by looking at the best value of the AIC, defined as

$$AIC = 2 * numberOfParameters - 2log(L),$$

with L the maximum value of the likelihood function.
Another important statistic is the R^2, which represents the proportion of variability in $\mathbf{Y}$ explained using $\mathbf{X}$ (with respect to the model with only the intercept).

2.2.2 Support Vector Machine

A Support Vector Machine (SVM) is a discriminative classifier defined by a separator hyperplane.
It is a supervised learning algorithm that, starting from the data of the labeled training set (whose it is known the class to which it belongs), generates a hyperplane able to divide the data belonging to different classes and therefore also able to classify the new elements of space. For example, in two dimensions the hyperplane is given by a line that divides the space $\mathbb{R}^2$ into two parts in each of which lies a class ([15]). SVM is used to produce non-linear decision boundaries; it is in fact an extension of the support vector classifier that increases the size of the feature space (increases the number of predictors) through mathematical functions called kernels.
In general, a kernel is defined as follows:

$$K(x, y) =< \phi(x); \phi(y) >$$

where $< \cdot\, ; \,\cdot >$ represents the scalar product.
The most used kernels are the following:

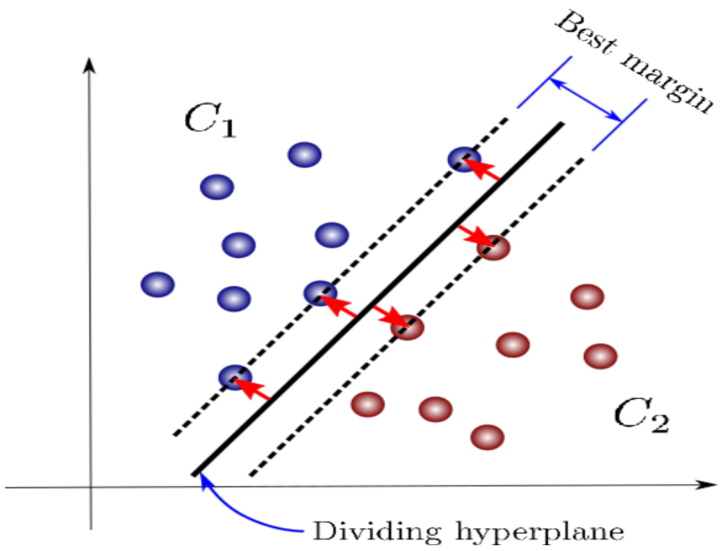


Figure 2.15: Simple idea of how the hyperplane generated by the SVM works for binary classification

- Linear Kernel: $K(x; y) =< x; y >$;
- Polinomial Kernel: $K(x; y) = (< x; y > +c)^d$;
- Radial Basis Function (RBF) Kernel: $K(x; y) = e^{\gamma(<x-y;x-y>)}$

This algorithm was created as a binary problem classifier (2 membership classes), but was subsequently adapted to multiclass problems as well.
The two most used approach are One-versus-One (OvO) and One-versus-All (OvA).
The OVO approach constructs K(K-1)/2 (with K $>$ 2, number of classes) SVMs, each of which only compares a pair of classes; counting the number of times a new datum is assigned to each of the K classes, we assign as label the one with the greatest number of occurrences.
The one-versus-all (OvA) approach fits K SVMs, comparing each time a single class with all the others; for each model we save the parameters $\beta_{0k}, \beta_{1k}, \ldots, \beta_{pk}$ (coefficients of the separating hyperplane) and assign the new data x to the class for which $\beta_{0k} + \beta_{1k}x_1 + \ldots + \beta_{pk}x_p$ is maximum ([16], [15]).

2.2.3 Random Forest

In this section, another kind of approach for both classification and regression purposes will be introduced, known as a "Random Forest".
Before to talk about it, it is reasonable to introduce the "decision tree" algorithm. The name derives from the fact that the set of rules used to divide the predictor space can be summarized and visible with a tree.
It starts from the root of the tree (at which point all observations belong to a single region) and gradually divides the space into two new branches ('top-down' approach). At each step, the tree will be divided into two further branches simply looking for the split which immediately leads to a greater reduction in variance and not looking at the one that would lead the entire branch to a smaller variance ('greedy' approach) [15]. A global view of this idea is shown in Figure 2.16

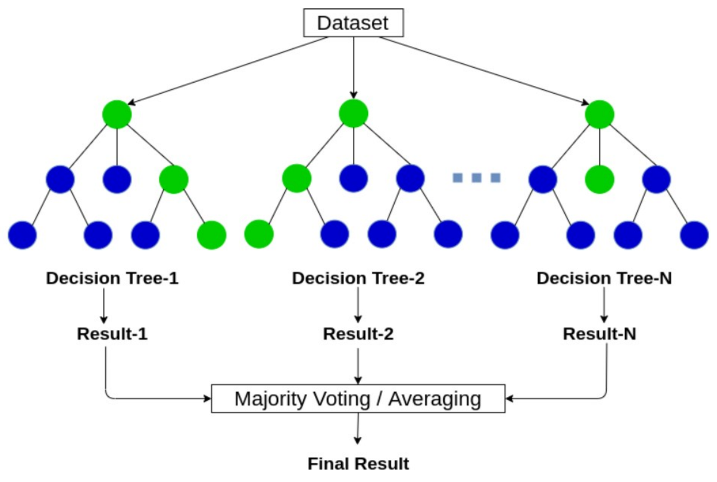


Figure 2.16: Simplify idea of the Random Forest approach

There are several criteria in choosing the best binary split, only those used in our analysis are described below. For the regression problem, the algorithm will look for the split such that the resulting tree has the lowest RSS.

In detail, let's define

$$R_1(j,s) = \{\mathbf{X}|\mathbf{X}_j < s\} \quad and \quad R_1(j,s) = \{\mathbf{X}|\mathbf{X}_j \geq s\}$$

for any $j = 1, \dots p$ and any cutpoint s. Then we seek for the value of j and s that minimize the equation

$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2 \tag{2.2}$$

where $\hat{y}_{R_1}$ and $\hat{y}_{R_2}$ are the mean response for the observations in $R_1(j,s)$ and $R_2(j,s)$.
Regarding the classification problem, the algorithm will split the tree into two parts in order to minimize the so-called 'Gini index', defined by the equation:

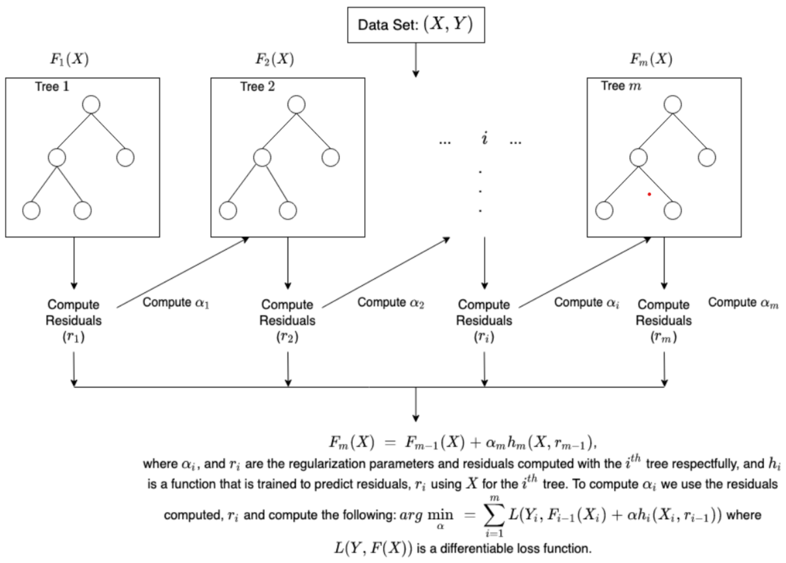
$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

where $\hat{p}_{mk}$ represents the percentage of the number of observations in region m belonging to class k.
It is observed that the Gini index assumes small values if $\hat{p}_{mk}$ is close to 1 (i.e. most of the observations in the m region come from class k) or close to 0 (very few observations of the class k). This is in line with the objectives of the method as we would like to have 'pure' regions, i.e. containing only one class.
The process of building the tree thus described could lead to a good prediction on the training data, but poor performance on the test data. One could therefore look for a tree with fewer splits, gaining both interpretability and lower variance at the cost of some bias.
The alternatives are manifold. You could cut the tree to a certain depth, or decide to make it grow only if that particular split leads to a reduction in variance above a predetermined threshold. An even better strategy is to grow a very large T_0 tree, and then prune it back to get a subtree. We will apply these strategies by tuning the hyperparameter.
By aggregating many decision trees, the performance obtained with respect

to the single tree could be improved. Random Forest is a particular method that allows you to do this. Several training data sets are generated (one possibility is through the bootstrap method), and a tree is fitted on each of them. The final result will be an average over all the predictions obtained (for regression trees), while the class with the highest number of occurrences in the various intermediate results for the classification problem will be predicted.
The B value concerning the number of training sets on which to build each tree, is a parameter to be fine-tuned. It doesn't lead to overfitting also for very large numbers. We fix it sufficiently large until the error has settled down.
The 'Random Forest' algorithm also provides an improvement by decorrelating the trees. In fact, every time a tree is built on the different training data sets generated with the bootstrap method, a random number m of predictors is chosen from the entire initial set p. The split is allowed to use only one of those m predictors. For greater detail of how "Random Forest" works, see [15].

2.2.4 XGBoost

XGBoost is a decision-tree-based ensemble Machine Learning algorithm (as Random Forest) that uses a gradient boosting framework.
Boosting, unlike *bagging* that involves creating multiple copies of the original training data set using the bootstrap and then combining all of the fitted trees on each copy in order to create the predictive model, works in a way that the trees are grown sequentially: each tree is grown using information from previously grown trees ([15]). The *boosting* approach learns slowly: given the current model we fit a decision tree to the current residuals, rather than the outcome y as the response. Then this new decision tree is added to the fitted function in order to update the residuals.
The parameter that controls the rate at which boosting learns is found by solving an optimization problem [17]. XGBoost in fact minimizes a regu-



$$F_m(X) = F_{m-1}(X) + \alpha_m h_m(X, r_{m-1}),$$

where α_i, and r_i are the regularization parameters and residuals computed with the i^{th} tree respectfully, and h_i is a function that is trained to predict residuals, r_i using X for the i^{th} tree. To compute α_i we use the residuals computed, r_i and compute the following: $arg \min_{\alpha} = \sum_{i=1}^{m} L(Y_i, F_{i-1}(X_i) + \alpha h_i(X_i, r_{i-1}))$ where $L(Y, F(X))$ is a differentiable loss function.

Figure 2.17: XGBoost illustration

larized (L1 and L2) objective function that combines a convex loss function (based on the difference between the predicted and target outputs) and a penalty term for model complexity. It's called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models [19].

Chapter 3

Experimental results

In this chapter will be examined the results of the different trained models. For better readability and interpretability, will be chosen always a certain type of metrics and images for the regression problem and another type for the classification one.

3.1 Evaluation of the regression models

About the regression, there will be taken into account the Mean Absolute Error (MAE), the Root Mean Square Error (RMSE), and the Accuracy. The first two are standard defined as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i) \tag{3.1}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.2}$$

where $\hat{y}_i$ is the predicted value of the true one y_i for the record i.
While the Accuracy is defined in this work as 100- the mean absolute percentage error, analytically:

$$Accuracy = 100\left(1 - \frac{1}{n}\sum_{i=1}^{n}\frac{(y_i - \hat{y}_i)}{y_i}\right) \tag{3.3}$$

The first image at which having a look is simple the right signal (blue line) against the predicted values (red points) (see the upper part of the Figure 3.1): the more red points coincide with the vertices of the interrupted blue signal, the more accurate the models will be.

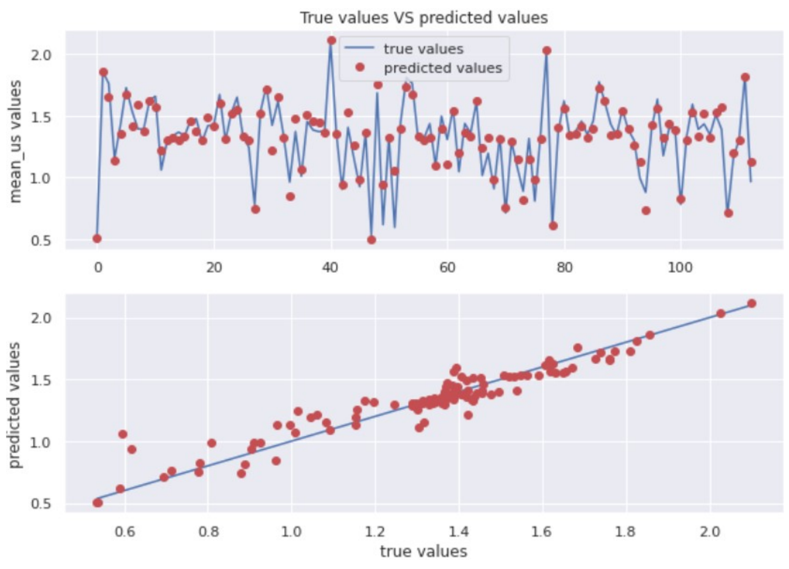


Figure 3.1: Images to analyze for the result of the regression algortihms

To better understand which are the areas with good performances and which no, the reader can view the images in Figure 3.1. The straight blue line represents the true value, while the red dots are the sorted predicted values. In this way the reader has a graphic idea of where the metrics come from, and which zones could be improved.

3.2 Evaluation of the classification models

Regarding the classification problem, the only metric analyzed is the Accuracy, giving much importance to the confusion matrix [22] and the ROC Curves.

3.2.1 Confusion Matrix

The confusion matrix can be read as follows: the classes are ordered as explained at the end of section 2.1; on the diagonal of the matrix you can read the amount of data of the test set correctly classified for each class; the sum of the values of a single column returns the number of data really belonging to that class (concerning Figure 3.2 for example, we can say that there are 9 data labeled as class 0); the sum of the values of a single row returns the number of data classified as belonging to the respective class (for example 10 data are classified as class 1).

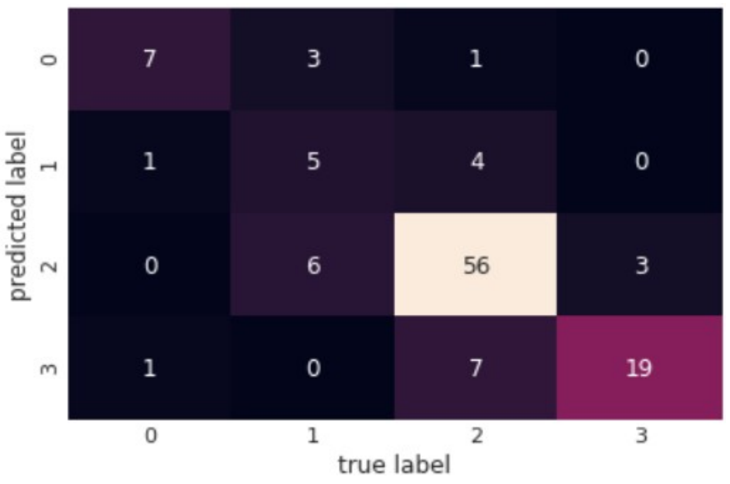


Figure 3.2: Example of Confusion Matrix.

Starting from here we can define useful statistics:

- *Accuracy*: defined as the ratio between the number of correctly classified observations (for a problem with two classes we speak of True positive-TP and True negative-TN) and the total number of remarks;
- *Precision*: also called value positive predictive, it is definite at the class level, i.e. for each class, and is the proportion of correctly classified observations (TP) to the number of observations totals that have been classified as belonging to that class (True positive-TP and False positive-FP);

- *Recall*: is always defined at the class level and is the proportion of correctly classified observations (TP) compared to the number of total observations that really belong to that class (True positive-TP and False negative-FN)

- *Specificity*: defined for every single class, measures the proportion between the number of True negative-TN, that is the observations correctly classified in the other classes, with respect to the total number of observations really belonging to the other classes (True negative-TN and False positive-FP)

- F_1*-score* is a measure of the test accuracy and it is defined as:

$$F_1 - score = 2\frac{precision * recall}{precision + recall}$$

3.2.2 ROC curves

Finally, it is analyzed a graphical method to evaluate the performance of the fitted models. The ROC curve (see Figure 3.3) shows the trade-off between

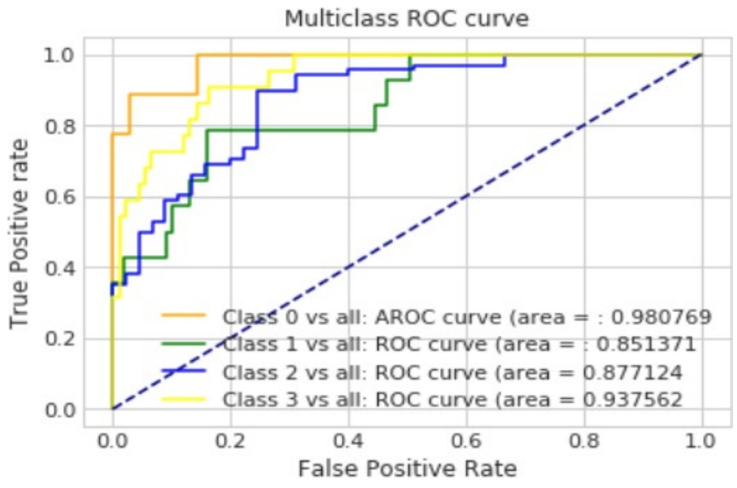


Figure 3.3: Example of a ROC curves.

recall and *specificity*.

Classifiers for which the curve is closer to the upper left corner have better performance; instead, the more the curve approaches the diagonal, the more accuracy of the test is low (for a random classifier we expect that the curve is close to the diagonal) [23].

3.3 Regression algorithms

3.3.1 Linear Regression

The first simple model trained on the aggregated data is the one shown in Figure 2.14, where some considerations have been already done.
Then, the results are showed in Figure 3.4.

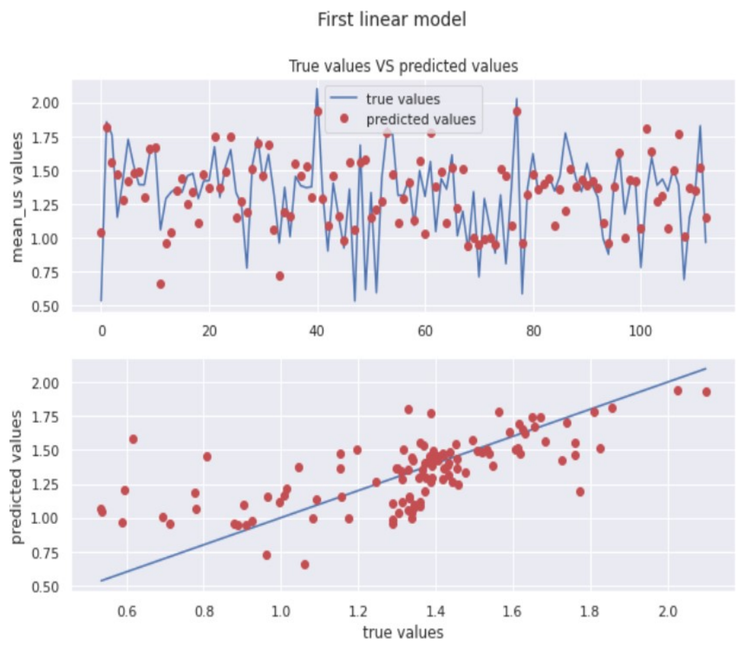


Figure 3.4: Graphical results of the Linear Model with all predictors.

It is obvious that the performances are very bad: almost no one points coincide with the vertices of the true signals and, especially for the low values of the response, the predictions are far away from the real values.
It is evident from Figure 2.14 (that image came from this model) that not all the features are important. That is, we can't exclude that most of the

coefficients are zero. In fact the p-value associated to the null hypothesis that some coefficients β_i are null, is very large for some of them.

Selecting optimal features.

We want to train another model by selecting only a subset of the initial features. We do that by taking into account the AIC's lowest values.
One would be tempted to consider all the possible combinations of the predictors. But that is infeasible! In fact, let's suppose that we want to compute all the combinations of the features by taking into account 14 of them, the machine should consider $\binom{28}{14} = \frac{28!}{14!14!} = 4.0 \cdot 10^6$ iterations. If we suppose that each iteration lasts 0.08 seconds, then the time needed to run the code is about 891 hours! Only by considering 14 variables!
Then we took the following decision: we will keep the features resulting the lowest AIC if we train a linear model with this number of predictors $[1, 2, 3, 4, 5, 6, 24, 25, 26, 27, 28]$.
Here follows the name of the features selected: *skewnees_ambient, kurtos_ambient, rootMeanSquare_ambient, mean_ambient, minimum_motorSpeed, skewnees_motorSpeed, rootMeanSquare_motorSpeed, mean_motorSpeed, minimum_torque, maximum_torque, skewnees_torque, kurtos_torque, mean_torque, skewnees_pm, kurtos_pm, rootMeanSquare_pm, mean_pm, kurtos_statorYoke, rootMeanSquare_statorYoke, minimum_is, maximum_is, skewnees_is, kurtos_is, rootMeanSquare_is.* The graphical results are visible in Figure 3.5.

Even if we haven't been able to select the best features ever for the linear regression, it is evident that this model doesn't fit good the data. It seems that the model performed better only the value of the *mean_us* close to 1.3 in the middle, the rest is very similar and for this reason, it has to be discarded.

The performance achieved by this last Linear regression model are:

- **Mean Absolute Error**: 0.17;

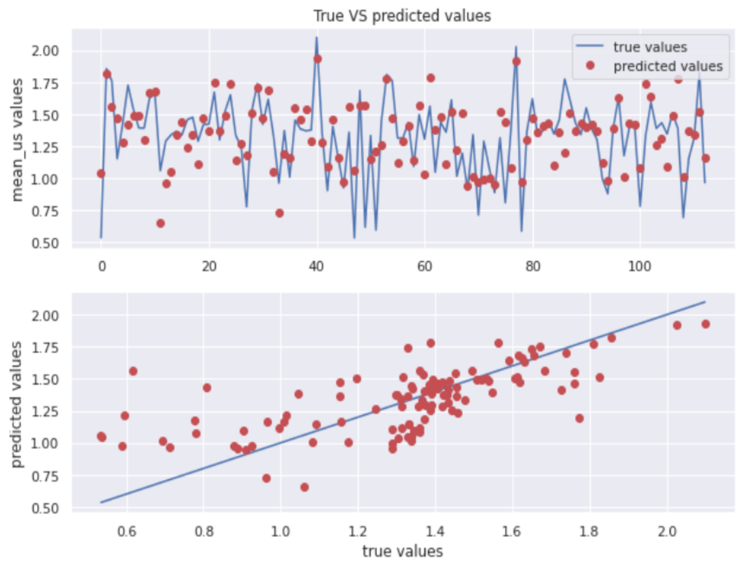


Figure 3.5: Graphical results of the linear model with the predictors chosen in order to obtain the lowest AIC

- **Root Mean Square Error**: 0.23;
- **Accuracy**: 82.54%

These results will be taken as starting point for the other models.

3.3.2 Random Forest

The first Random Forest model analysed is the one with the parameters chosen by default. The most important, are defined as follows [24]:

- *n_ estimators* (default=100): the number of trees in the forest;
- *max_ depth* (default=None). the maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than *min_ samples_ split* samples;
- *min_ samples_ split* (default=2) : the minimum number of samples required to split an internal node;

- *min_samples_leaf* (default=1): the minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least *min_samples_leaf* training samples in each of the left and right branches;

- *max_features* (default="auto"): the number of features to consider when looking for the best split. It can assume values "auto", "sqrt", "log2", or some int value less than the number of predictors.

- *bootstrap* (default=True): whether bootstrap samples are used when building trees. If False, the whole dataset is used to build each tree.

We can soon observe the graphical results, in Figure 3.6.

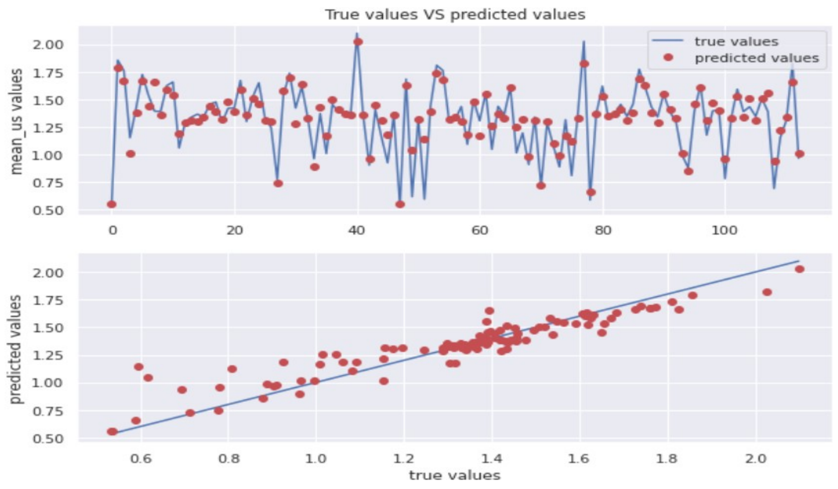


Figure 3.6: Graphical result of the first Random Forest with parameter chosen by default

It is clear that a step forward has been taken from the linear model in section 3.3.1. In the upper part much more values are close to the vertices of the real signal and they are even much closer to the blue line: the mass of the signal (the red points in the central zone) is predicted very good and the extreme values have slightly worse performance only for low values, where some of them are predicted as they belonged at the central zone.

These observations are corroborated from the metrics visible below, where the accuracy is almost 92.7%.

- **Mean Absolute Error**: 0.08;
- **Root Mean Square Error**: 0.11;
- **Accuracy**: 92.69%

Random Search for tuning the parameters.

Now, we instantiate the Random Search and fit it like any Scikit-Learn model for tuning the parameters. The first thing to do is create the grid of the parameters values that have to be taken into account by the Random Search [25]

- *bootstrap*: [True, False],
- *max_depth*: [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, None],
- *max_features*: ['auto', 'sqrt', 14.0],
- *max_samples*: [0.1, 0.3, 0.5, 0.7, 0.9, 1],
- *min_samples_leaf*: [1, 2, 4],
- *min_samples_split*: [2, 5, 10],
- *n_estimators*: [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000]

On each iteration, the algorithm will choose a different combination of the parameters. Altogether, there are $3 \cdot 11 \cdot 3 \cdot 6 \cdot 3 \cdot 3 \cdot 10 = 53460$ settings! However, the benefit of a random search is that we are not trying every combination, but selecting at random to sample a wide range of values.
The most important arguments in RandomizedSearchCV are *n_iter*, which controls the number of different combinations to try, and *cv* which is the number of folds to use for Cross Validation (it has chosen *n_iter*= 500 and *cv*=3 respectively). More iterations will cover a wider search space and more

cv folds reduce the chances of overfitting, but raising each will increase the run time.
Machine learning is a field of trade-off, and performance vs time is one of the most fundamental.

The best parameters after applied the Random Search are the following:

- $n\_estimators$: 500;
- $min\_samples\_split = 5$;
- $min\_samples\_leaf = 2$;
- $min\_samples$: 0.9;
- $max\_features =$ 'auto';
- $max\_depth = 60$;
- $bootstrap =$ True.

whose results are the following:

- **Mean Absolute Error**: 0.09;
- **Root Mean Square Error**: 0.12;
- **Accuracy**: 91.75%

We have no improved our model since the accuracy is decreased almost of 0.1%. That means the Random Search has no explored the default parameter solution, and all of the other ones had worse performance than the first one. The differences between Figures 3.7 and 3.6 are not visible at first glance. For example, we observe that the value greater than the lowest one is predicted 0.75 here with respect to 0.7 in the first trained Random Forest model.

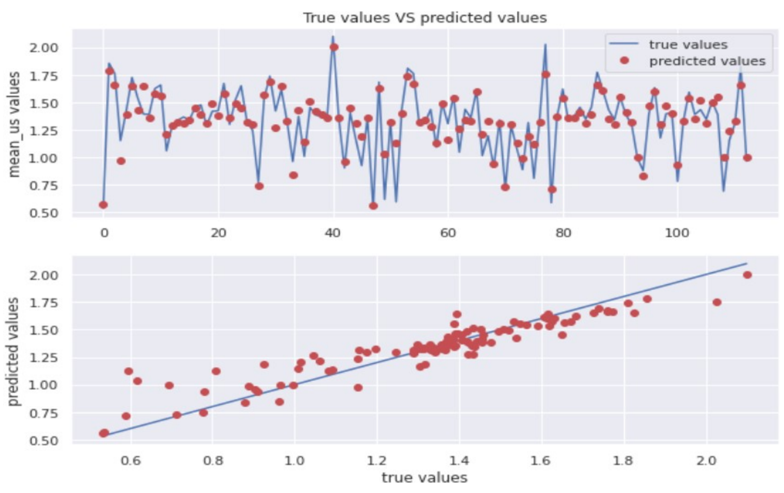


Figure 3.7: Graphical result of the Random Forest model with aRandom Search for tuning the parameters.

Random Forest applied on a reduced the predictors' space

In order to quantify the usefulness of all the variables in the entire predictors' space, we can look at the relative importance of the variables.
The importance returned in Skicit-learn represents how much including a particular variable improves the prediction [26].
It can be said, by looking at Figure 3.8, that for the Random Forest model the *maximum_motorSpeed* and *mean_motorSpeed* variables are the most two important ones.
Obviously, not only the first two will be considered but will take the features whose total gain will be at least 0.95. In particular will be chosen the 14 features, *maximum_motorSpeed, mean_motorSpeed, minimum_motorSpeed, mean_torque, rootMeanSquare_is, rootMeanSquare_motorSpeed, maximum_torque, minimum_torque, minimum_is, kurtos_motorSpeed, maximum_is, skewnees_motorSpeed, kurtos_torque, skewnees_torque*, whose total gain is equal 95.25%.

The model trained using the same parameters found with the Random Search and the important features just mentioned, achieves the following

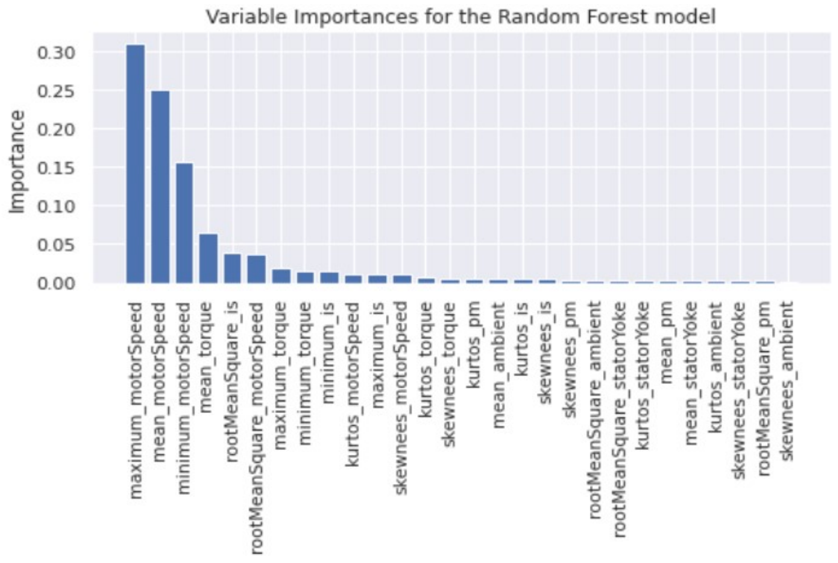


Figure 3.8: Usefulness of the Random Forest variables.

performances:

- **Mean Absolute Error**: 0.08;
- **Root Mean Square Error**: 0.12;
- **Accuracy**: 92.48%

Even if the number of features used is 21 instead of 28 considered in the first model, the MAE is the same and the accuracy is reduced only by 0.2%.

Grid Search and features selection

The last Random Forest model explored is the one with a Grid Search for tuning the parameters and a subset of the features selected from the same procedure introduced in the preceding paragraph.
The Grid Search is a method for tuning the parameters similar to the Random Search, but with the difference that this time all the combinations of the parameters have to be explored. Here for not boring reading, we will show only the final results. Figure 3.9 shows that this last model predicted very well for the values above 1.2, especially for the greatest two values which were

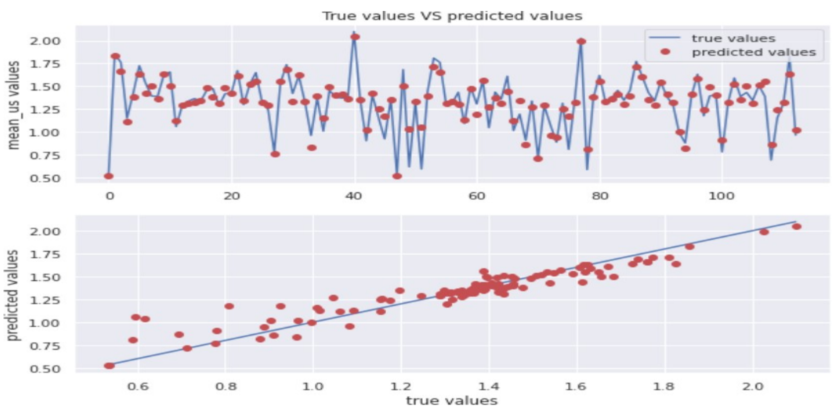


Figure 3.9: Graphical result of Random Forest with Grid Search and features selection

always understimated. Some improvements instead, wuold be indicated for the left side of the image below.
The metrics achieved are shown here:

- **Mean Absolute Error**: 0.07;
- **Root Mean Square Error**: 0.11;
- **Accuracy**: 92.72%

The accuracy is the greatest one achieved up to now: it is greater than the first model by 0.03% and almost 0.25% more than the model with parameters chosen with a Grid Search and features selection.
It is also necessary to say, that the run time of the Grid Search is definitely greater than the Random Search. So, which Random Forest model is better than the other, is not easy to decide. It depends on the goal and the equipment available from the company which will use this algorithm. If the machines are not so powerful and lose a percentage of 0.25 in the accuracy is not a big issue, it has to be preferred the model with the Random Search for selecting the best parameters. In addition, if the prediction has to be done in real-time (and in the PdM framework this would be necessary), the choice would be to use the Random Search method.

3.3.3 XGBoost

The next step is to instantiate an XGBoost regressor object by calling the XGBRegressor() class from the XGBoost library with the hyper-parameters passed as arguments.
The most common parameters that should be known, are [27]:

- *learning_rate* (default 0.3): step size shrinkage used to prevent overfitting. Range is [0,1]
- *max_depth* (default 6): determines how deeply each tree is allowed to grow during any boosting round.
- *colsample_bytree* (default 1): percentage of features used per tree. High value can lead to overfitting.
- *n_estimators* (default 100): number of trees you want to build.
- *min_child_weight* (default 1):is the minimum weight required in order to create a new node in the tree. A smaller *min_child_weigh*t allows for more complex trees, but again, more likely to overfit.
- *objective* : determines the loss function to be used like reg:squarederror for regression problems, reg:logistic for classification problems with only decision, binary:logistic for classification problems with probability.

XGBoost also supports regularization parameters to penalize models as they become more complex and reduce them to simple (parsimonious) models.

- *gamma* (default 0): controls whether a given node will split based on the expected reduction in loss after the split. A higher value leads to fewer splits. Supported only for tree-based learners.
- *alpha* (default 0): L1 regularization on leaf weights. A large value leads to more regularization.
- *lambda* (default 1): L2 regularization on leaf weights and is smoother than L1 regularization

Figure 3.10 shows immediately the graphical performance of the first XG-Boost model with the parameters chosen by default.

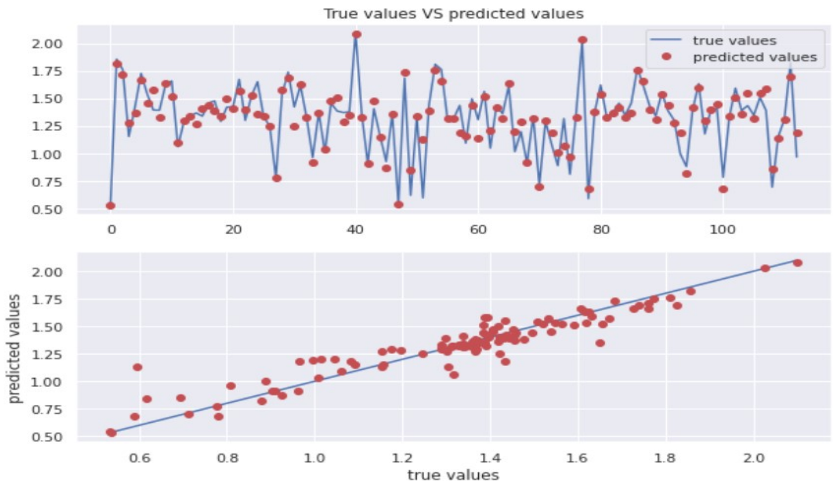


Figure 3.10: Graphical Results of the XGBoost model with parameter chosen by default

This time the predictions are really good. Not only for the right side of the image below, but also for the left whose red points seem to approach the blue line: only one point in the left area has been predicted about 1.13 while the corresponding true value was almost 0.65. It is necessary to remember that the data have been standardized and we don't know how that happened. The performances achieved by this first XGBoost model are:

- **Mean Absolute Error**: 0.07;
- **Root Mean Square Error**: 0.11;
- **Accuracy**: 93.1%

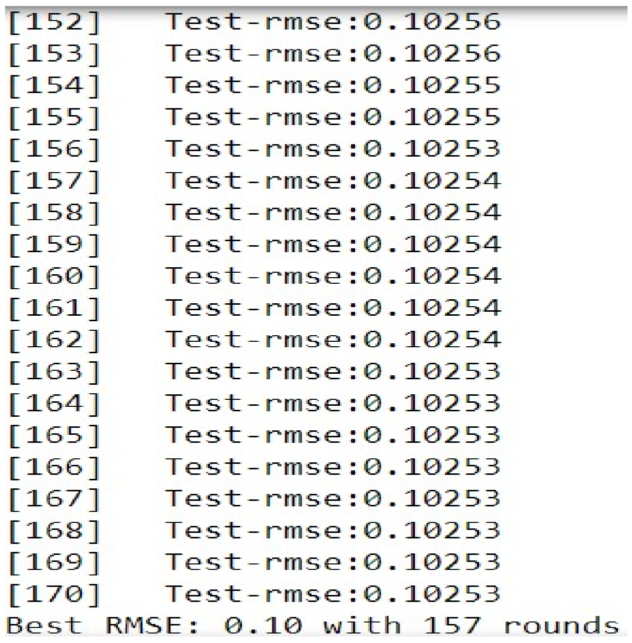
So a difference of 0.48($= 1.13 - 0.65$) could be significant. As we expected (see the image above), the metrics achieved from this model are very good, especially about the accuracy, which is greater than the 93%!

XGBoost model with tuning the parameters

In order to build more robust models, it is common to train a k-fold Cross Validation for tuning the parameters, where all the entries in the original training dataset are used for both train as well as validation. XGBoost supports k-fold cross validation via the cv() method. Cross Validation is a technique that involves reserving a particular sample of a dataset (the validation set) that is not used for training the model. Later, the model is tested on this sample [28].
All we need to do is specify the *nfolds* parameter, which is the number of cross validation sets I want to build.

The first parameter we will look at is not part of the parameters dictionary, but will be passed as a standalone argument to the training method. This parameter is called *num_boost_round* and corresponds to the number of boosting rounds or trees to build. XGBoost provides a nice way to find the best number of rounds whilst training. Since trees are built sequentially, instead of fixing the number of rounds at the beginning, we can test our model at each step and see if adding a new tree/round improves performance.
So we need to pass a *num_boost_round* which corresponds to the maximum number of boosting rounds that we allow. We set it to a large value (=1000) hoping to find the optimal number of rounds before reaching it, if we haven't improved performance on our test dataset in *early_stopping_round* rounds (it is fixed to 15).
As we can see from Figure 3.11, the cross validation stopped before reaching the maximum number of boosting rounds, that's because after the 157th tree adding more rounds did not lead to improvements of RMSE on the test dataset.

In order to tune the other hyperparameters, we will use the *cv* function from XGBoost. We don't need to pass a test dataset here as just said. It's because the cross-validation function is splitting the training dataset into



```
[152]   Test-rmse:0.10256
[153]   Test-rmse:0.10256
[154]   Test-rmse:0.10255
[155]   Test-rmse:0.10255
[156]   Test-rmse:0.10253
[157]   Test-rmse:0.10254
[158]   Test-rmse:0.10254
[159]   Test-rmse:0.10254
[160]   Test-rmse:0.10254
[161]   Test-rmse:0.10254
[162]   Test-rmse:0.10254
[163]   Test-rmse:0.10253
[164]   Test-rmse:0.10253
[165]   Test-rmse:0.10253
[166]   Test-rmse:0.10253
[167]   Test-rmse:0.10253
[168]   Test-rmse:0.10253
[169]   Test-rmse:0.10253
[170]   Test-rmse:0.10253
Best RMSE: 0.10 with 157 rounds
```

Figure 3.11: Number of trees with no improvements.

nfolds and iteratively keeps one of the folds for test purposes.
The parameters selected after applying the Cross Validation on the XGBoost model, are:

- $learning\_rate = 0.15$;
- $max\_depth = 3$;
- $colsample\_bytree = 0.9$;
- $gamma = 0.0$;
- $min\_child\_weight = 3$;
- $alpha = 0$.

It is also suggested to observe the elapsed time to run the Cross Validation. It was equal to 12328 $seconds = 12328/3600\ hours = 3.42\ hours$ (Processor Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz, 2.30 GHz, RAM 16,0 GB,

Operative System with 64 bit, processor based on x64).
The metrics achieved by this model are the following:

- **Mean Absolute Error**: 0.07;
- **Root Mean Square Error**: 0.10;
- **Accuracy**: 93.72%

The accuracy in this case is lightly greater than the one where the tuning parameters are not used.
To have a graphic idea of the XGBoost results, we can see the Figure 3.12

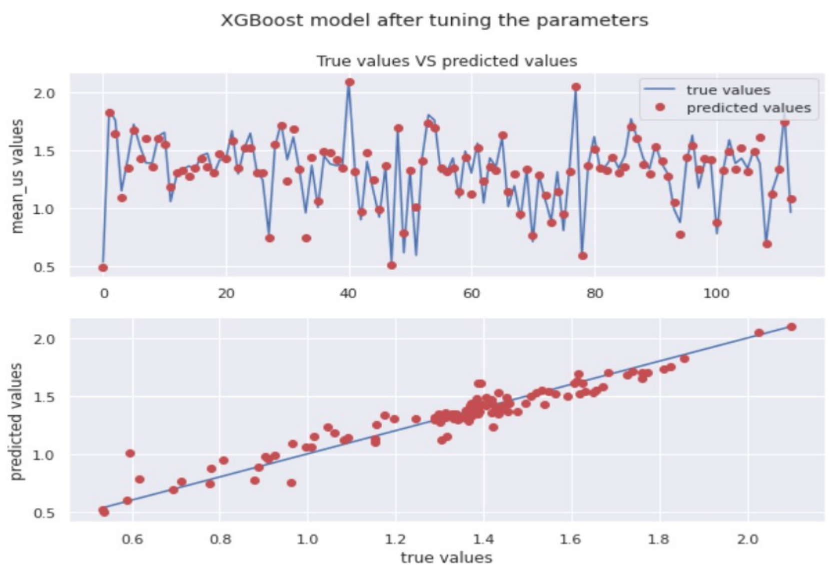
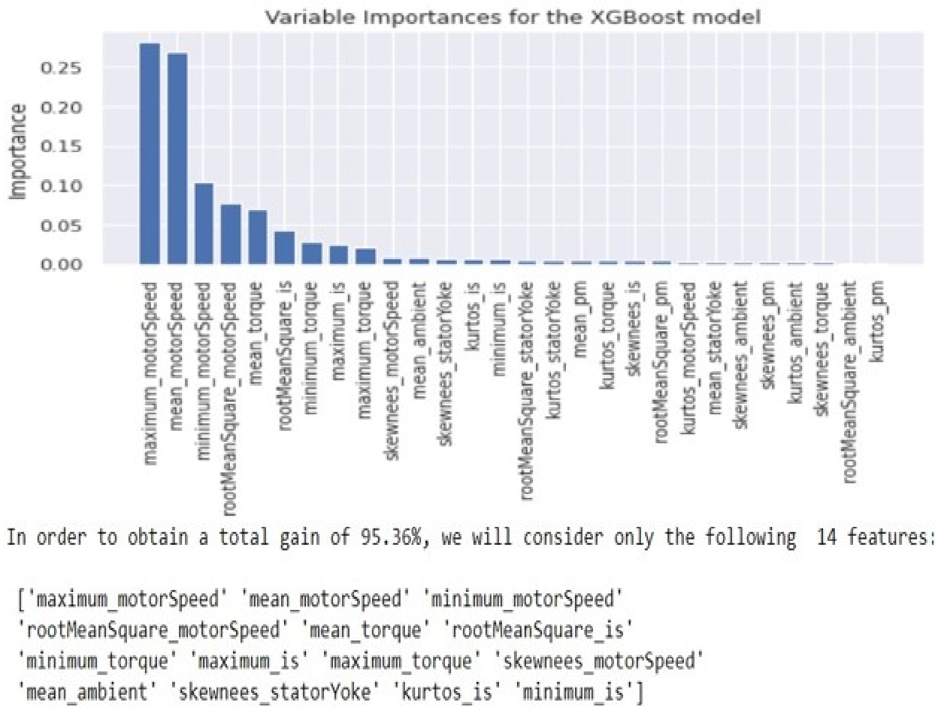


Figure 3.12: Graphic results of the XGBoost after tune the parameters.

We can see very good performances. The low predicted values are always more close to the blue line and the prediction of the right side are good as always.

XGBoost with features selection

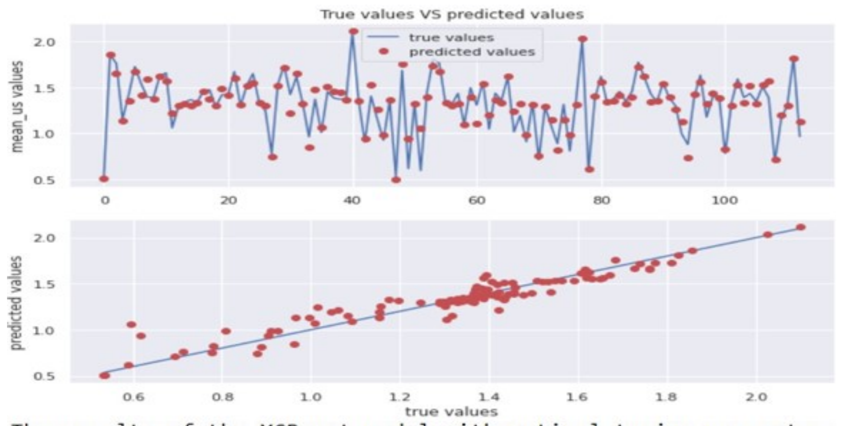
Another method to try to improve the performances is removing some useless features by visualizing the XGBoost model, is to examine the importance of



```
In order to obtain a total gain of 95.36%, we will consider only the following  14 features:

 ['maximum_motorSpeed' 'mean_motorSpeed' 'minimum_motorSpeed'
 'rootMeanSquare_motorSpeed' 'mean_torque' 'rootMeanSquare_is'
 'minimum_torque' 'maximum_is' 'maximum_torque' 'skewnees_motorSpeed'
 'mean_ambient' 'skewnees_statorYoke' 'kurtos_is' 'minimum_is']
```

Figure 3.13: Importance features for XGBoost model.

each feature column in the original dataset within the model.
There are several types of 'importance' in Xgboost, i.e. it can be computed in several different ways. The default type is the gain, which shows the average gain across all splits where the feature was used (just as used for the Random Forest).
In the upper part of the Figure 3.13, is shown the histogram of the features' name on the x-axis and their relative importance on the y-axis. Below instead are visible the only features that we will consider for the next model in order to have a total gain greater than 95%.
It is suggestive to note that the predictors' space is halved and basically the rotor and stator's temperatures are removed (will be considered only the variable *skewnees_statorYoke* and no one statistic relative to *pm*).
By looking at Figure 3.14 instead, the value of accuracy is highlighted, since it is almost 94%. The graphic results do not differ from the previous one. It



```
The results of the XGBoost model with optimal tuning parameter:

Mean Absolute Error: 0.07
Root Mean Square Error: 0.09
Accuracy: 93.94 %.
```

Figure 3.14: Results of the final XGBoost-Regressor model.

can be noted that the fourth value in ascending order is predicted slightly bad than the previous one, but all the others are more precise.
By outperforming the Linear and the RF regression models, the XGBoost regressor is the most appropriate model for the problem addressed along the section2.1.

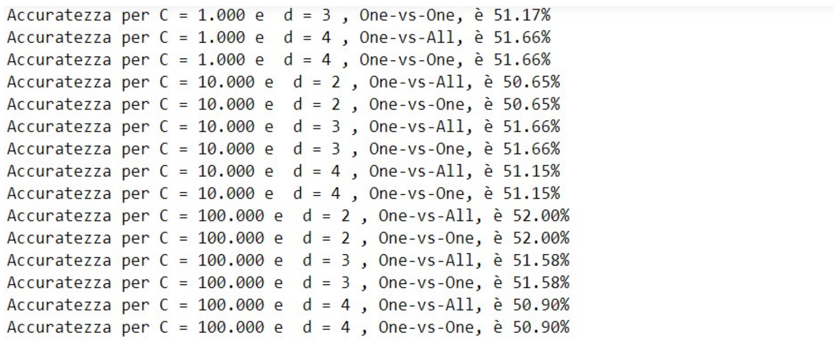
3.4 Classification algorithms

Here we address the classification problem, i.e. we do not want to predict the exact value of the vector voltage, but we want to predict the class (the range of values are explained at the end of section 2.1.5) at which each value belongs.

3.4.1 Support Vector Machine

SVM with polynomial kernel

Let's fit the first SVM model with the polynomial kernel.
The hyperparameters to be set are the coefficient C which corresponds to



```
Accuratezza per C = 1.000 e  d = 3 , One-vs-One, è 51.17%
Accuratezza per C = 1.000 e  d = 4 , One-vs-All, è 51.66%
Accuratezza per C = 1.000 e  d = 4 , One-vs-One, è 51.66%
Accuratezza per C = 10.000 e  d = 2 , One-vs-All, è 50.65%
Accuratezza per C = 10.000 e  d = 2 , One-vs-One, è 50.65%
Accuratezza per C = 10.000 e  d = 3 , One-vs-All, è 51.66%
Accuratezza per C = 10.000 e  d = 3 , One-vs-One, è 51.66%
Accuratezza per C = 10.000 e  d = 4 , One-vs-All, è 51.15%
Accuratezza per C = 10.000 e  d = 4 , One-vs-One, è 51.15%
Accuratezza per C = 100.000 e  d = 2 , One-vs-All, è 52.00%
Accuratezza per C = 100.000 e  d = 2 , One-vs-One, è 52.00%
Accuratezza per C = 100.000 e  d = 3 , One-vs-All, è 51.58%
Accuratezza per C = 100.000 e  d = 3 , One-vs-One, è 51.58%
Accuratezza per C = 100.000 e  d = 4 , One-vs-All, è 50.90%
Accuratezza per C = 100.000 e  d = 4 , One-vs-One, è 50.90%
```

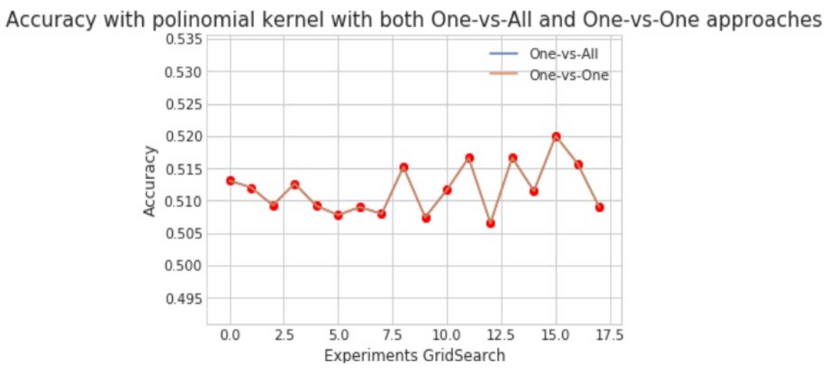
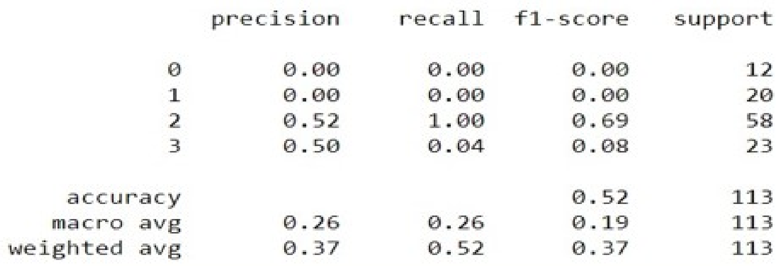


Figure 3.15: Grid Search for tuning the hyperparameters C and d

the penalty term (plus larger the more it penalizes classification errors) and the degree of the polynomial d. For the tuning of the hyperparameters we use a sort of GridSearch doing vary the coefficient C from a value of 10^3 to a value of 10^2 with one step multiplicative equal to 10 and degree d in range 2; 3; 4 and evaluating the accuracy of the results on the validation set. In the analysis we used both approaches OvO and OvA, see Figure 3.15.

The maximum accuracy (52,00%) is obtained for $C = 100$ and $d = 2$, while it is practically equal to the variation of the approach used, which is why we decide to fit the model with $C = 100$ and $d = 2$ with a One versus One approach.
We get the results shown in Figure 3.16 where, both the confusion matrix along with a number of statistics on the classification and the Multiclass ROC curve, are reported.

Such performances do not need any comment. All the records are classified as belonged to class 2 (the most popular one) apart from two considered as class 3, besides only one was predicted correctly.



```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        12
           1       0.00      0.00      0.00        20
           2       0.52      1.00      0.69        58
           3       0.50      0.04      0.08        23

    accuracy                           0.52       113
   macro avg       0.26      0.26      0.19       113
weighted avg       0.37      0.52      0.37       113
```

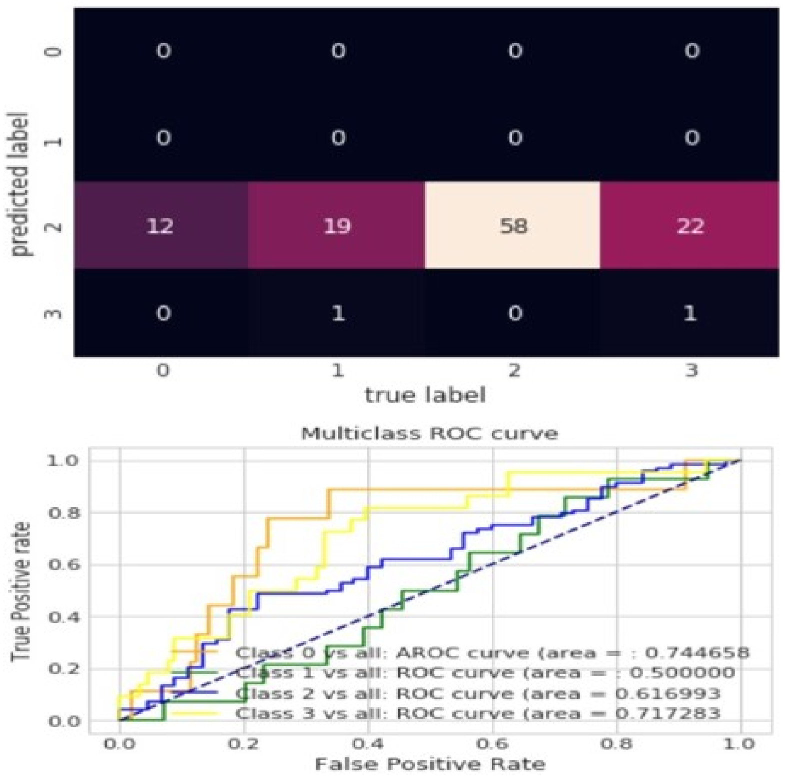


Figure 3.16: Confusion matrix and ROC curve for the SVM model with polynomial Kernel.

SVM with Radial kernel At this point, let's try to fit also an SVM model with a Radial kernel, following a logic similar to what was done before.
In this case, the hyperparameters to fit are the coefficient C and the coefficient

γ: the first is chosen by a GridSearch, while the second is set by the model in order to select the best possible value (gamma = 'auto').

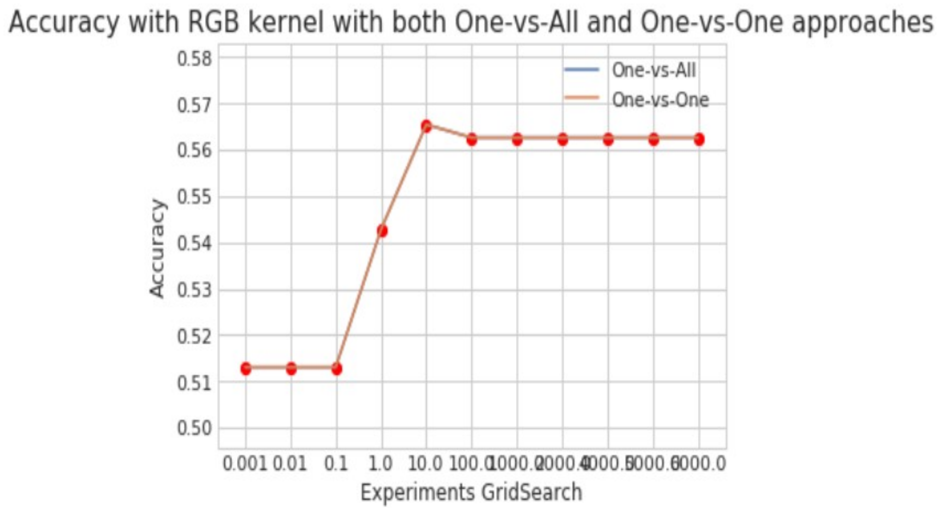
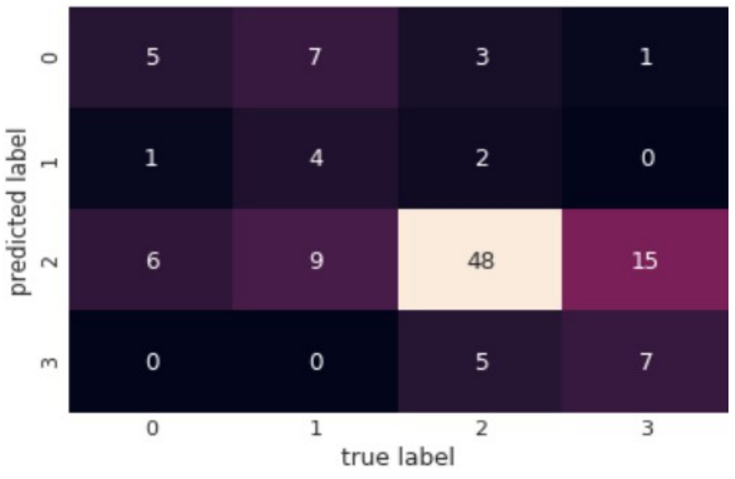


Figure 3.17: Grid Search on the parameter C of the SVM with Radial kernel

As can be seen from Figure 3.17, the best value is for $C = 10$ with which an accuracy of 56.55% on the validation set is reached.



predicted label \ true label	0	1	2	3
0	5	7	3	1
1	1	4	2	0
2	6	9	48	15
3	0	0	5	7

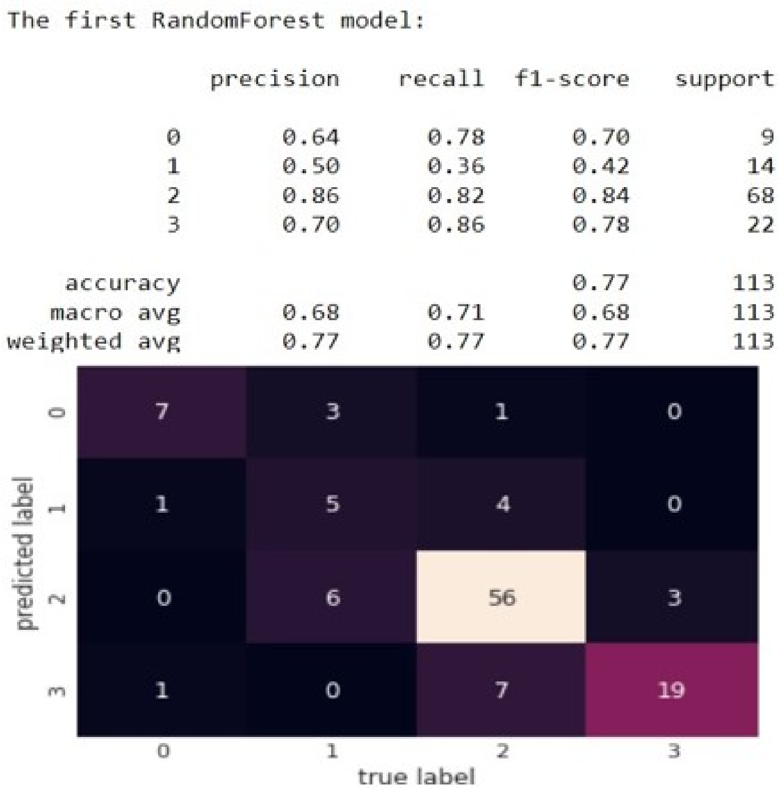
Figure 3.18: Confusion Matrix of the SVM with Radial Kernel

We can observe from Figure 3.18, as this model is definitely better than the one with Polynomial kernel. There are 5 and 4 records predict correctly for

the class 0 and 1, even if on a *support* of 12 and 20 respectively. Also, 12 records are predicted to belong to class 3 (only seven were really of class 3) at the expense of 'only' 48 records predicted of class 2.
However, we can state that the SVM models do not fit on this data, since the overall accuracy is slightly greater than 55%. We will do better with the other models.

3.4.2 Random Forest

In this section three Random Forest models will be explored in the classification framework:



```
The first RandomForest model:

              precision    recall  f1-score   support

           0       0.64      0.78      0.70         9
           1       0.50      0.36      0.42        14
           2       0.86      0.82      0.84        68
           3       0.70      0.86      0.78        22

    accuracy                           0.77       113
   macro avg       0.68      0.71      0.68       113
weighted avg       0.77      0.77      0.77       113
```

Figure 3.19: Confusion matrix of the Random Forest model with tthe parameters chosen by default.

1. with the parameters chosen by default;
2. a Grid Search will be instantiate for tuning the best Hyperparameters;
3. the predictors space is reduced by looking for the most important features.

The Figure shows immediately the confusion matrix of the first model whose accuracy is **76,73%**.

The algorithm has predicted correctly 7 records belonged to class 0. It made four errors for that class, and it has been not able to predict the other two numbers for the same class. The *recall* and the *precision* for class 2 and 3 are reasonable. Opposite direction for class 1, where the precision and the recall are just 50% and 36%. It has predicted rightly only ten records on a *support* of 14.

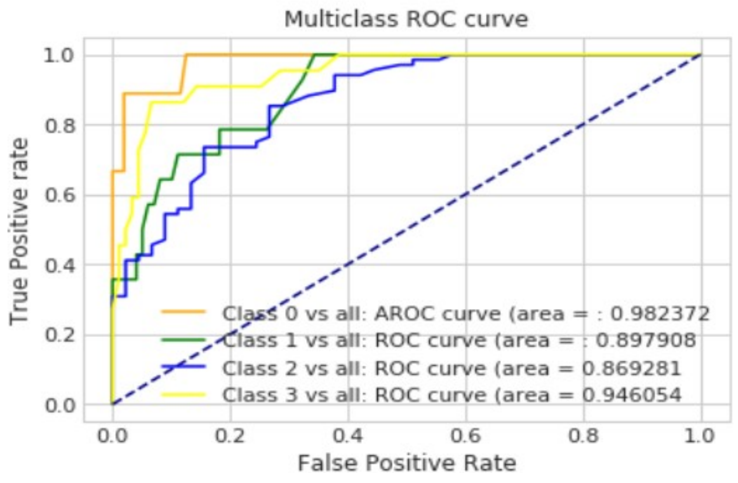


Figure 3.20: Multiclass ROC curve of the first Random Forest.

Moreover, in Figure 3.20, we can see that the Area Under the Curve (AUC) of the class 0, is really good, about 98%.

Random Forest with the custom parameters

Now the Grid Search is instantiated in order to fix the best parameters, whose values are:

- *bootstrap* = False;
- *ccp_alpha* = 0.0;
- *max_depth* = 40;
- *max_samples* = 0.2;
- *min_samples_split* = 5.

Once we choose the optimal parameters, it is very important to set also the parameters *max_features*. We do that by plotting the number of predictors against the number of trees, by evaluating the Out of Bag error (see Figure 3.21).

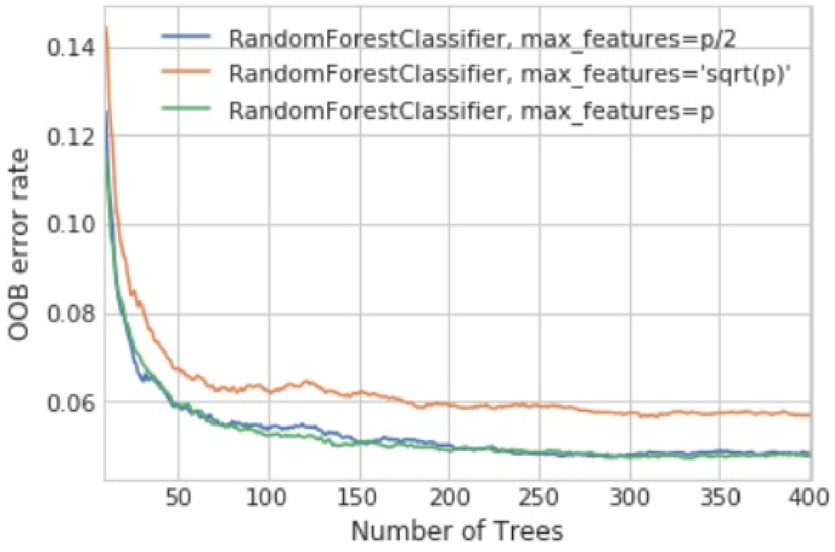
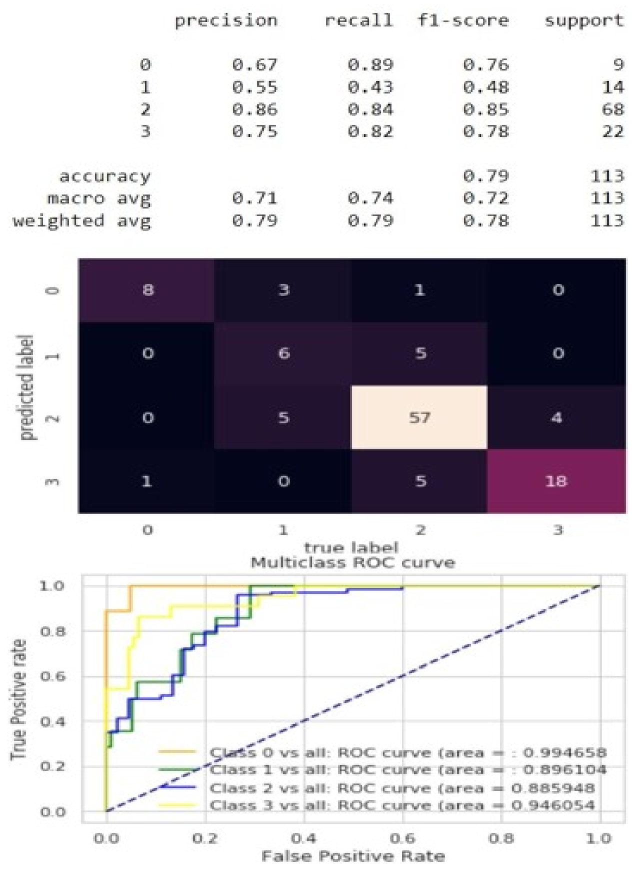


Figure 3.21: OOB error versus the number of trees for three different values of the parameter *max_features*.

The difference between Out-Of-Bag (OOB) error and k-fold cross-validation is that k-fold cross-validation and OOB assume different sizes of learning samples. For example, in 10-fold cross-validation, the learning set is 90%, while the testing set is 10%. However, in OOBE if each bag has n samples, such that n= total number of samples in the whole samples set, then this implies that the learning set is practically about 66% (two thirds) and the testing set is about 33% (one third).



```
              precision    recall  f1-score   support

           0       0.67      0.89      0.76         9
           1       0.55      0.43      0.48        14
           2       0.86      0.84      0.85        68
           3       0.75      0.82      0.78        22

    accuracy                           0.79       113
   macro avg       0.71      0.74      0.72       113
weighted avg       0.79      0.79      0.78       113
```

Figure 3.22: Global results of the Random Forest model with custom Hyper-parametes

It can be stated that *max_features* = 'p/2' and 'p' perform very similar, with respect to *max_features* = 'sqrt' whose curve is always above to the other ones. We can also say that the error made decreases as the number of trees increases, up to 300, where a flat area seems to begin. Some powerful machines could increase this number and observe the results.

This model has predicted correctly one more record for the class 0,1 and

2. Six records (on a total of 11) have been predicted correctly belonged to the class 1, by increasing the *precision* from 0.50 to 0.55 and the *recall* to 0.43 (see the upper part of the Figure 3.22).
Even more important is that the *precision* and the *recall* of the class 0 have reached 0.67 and 0.89 respectively.
These observations have increased the ROC curve of the class 2 to 0.886.

Random Forest with custom parameters and a subset of the entire predictors set

As usual, it would be nice to understand which features to remove based on their minor importance according to the last model trained. Figure 3.23, shows the results. According to this image, 23 features will be selected with

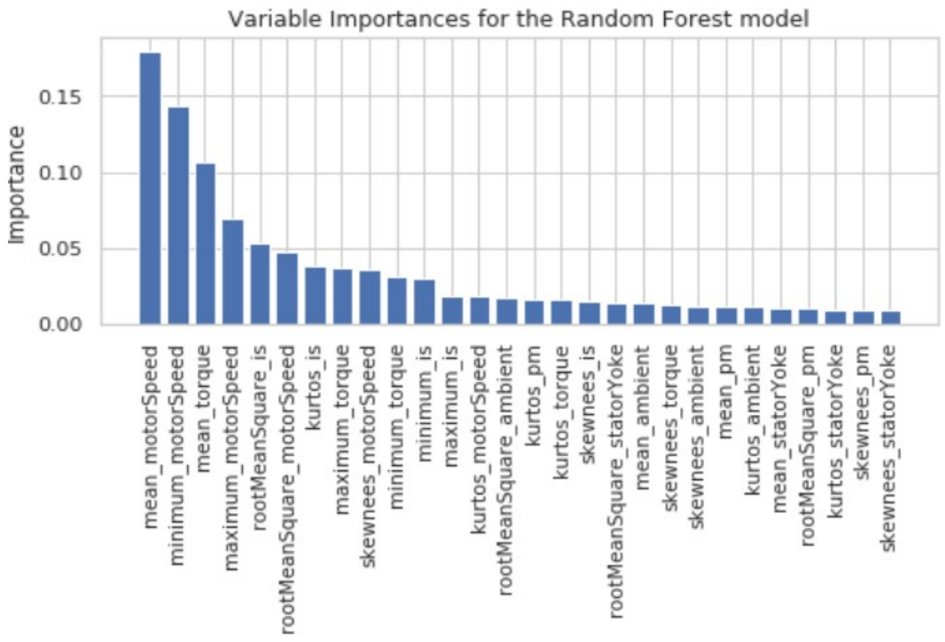
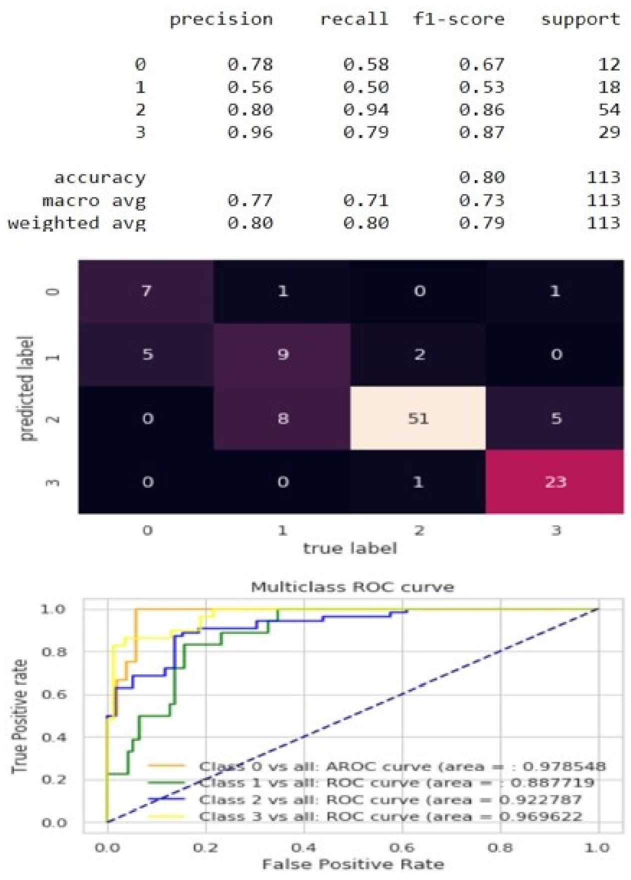


Figure 3.23: Features importance in the Random Forest

an overall gain of 95.17%. It should be noted that each time this model grows a tree, only an half of the entire predictor' space will be used for splitting it, so it is reasonable that more features (than for example the XGBoost

Regressor) are considerated as important.

The accuracy achieved by this last model is equal to **79,7%** and both the confusion matrix and the multiclass ROC curve are shown in Figure 3.24.



```
              precision    recall  f1-score   support

           0       0.78      0.58      0.67        12
           1       0.56      0.50      0.53        18
           2       0.80      0.94      0.86        54
           3       0.96      0.79      0.87        29

    accuracy                           0.80       113
   macro avg       0.77      0.71      0.73       113
weighted avg       0.80      0.80      0.79       113
```

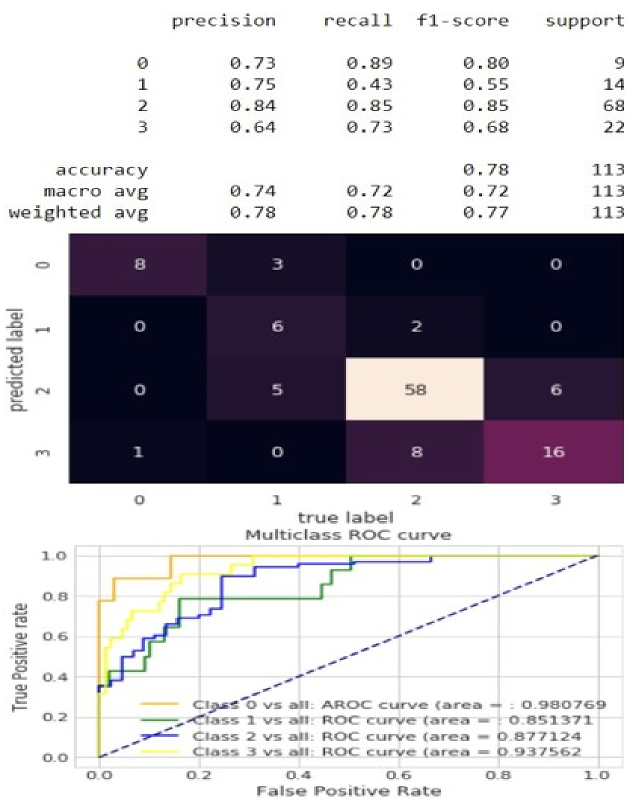
Figure 3.24: Results of the Random Forest with optimal parameters and feature selection.

Very similar results between the last two models. Even if the accuracy of the last model (the one with only 23 features) is greater than the model with all the features of 0.4% in mean, practically there may be no big changes. The *recall* of the class 0 is decreased because the *support* was 12 instead of 9; on the contrary, the precision is increased because only two wrong predictions have been made.

Some of these changes are due to the randomness of the Random Forest algorithm (for example when a tree is built) or to the splitting considered for plotting the confusion matrix or also to the number of simulations (100 is not so big for a more computational powerful machine).

3.4.3 XGBoost

The last model analyzed for classification purposes, is the XGBoost. As usual the steps will be to train three models: the first by using the given default values on it and then trying to improve it by optimizing settings and reducing the predictors' space.



```
              precision    recall  f1-score   support

           0       0.73      0.89      0.80         9
           1       0.75      0.43      0.55        14
           2       0.84      0.85      0.85        68
           3       0.64      0.73      0.68        22

    accuracy                           0.78       113
   macro avg       0.74      0.72      0.72       113
weighted avg       0.78      0.78      0.77       113
```

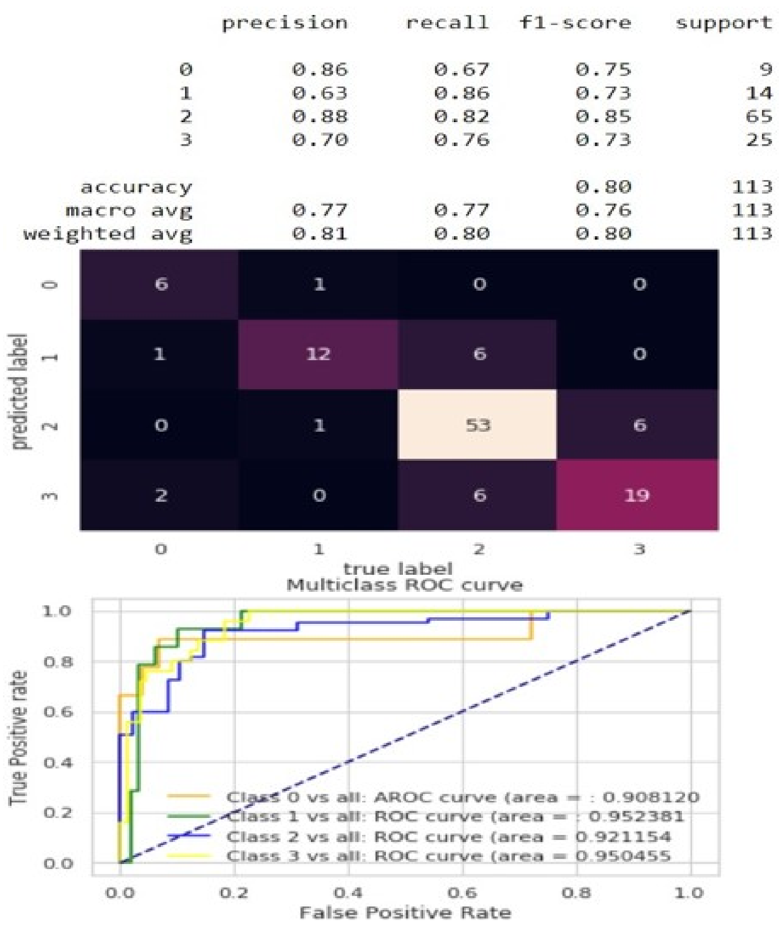
Figure 3.25: Global Final results of the XGBoost model

We can observe from Figure 3.25 that this model performs better than the first one performed by the Random Forest algorithm, whose accuracy is

78,88%. We can also observe well performances of the pair *precision* and *recall* for the class 0 (both above 70%), slightly worse than the ones of class 2 (see Figure 3.25).

XGBoost with optimal settings

Here it is instantiated a sort of Grid Search creating by myself in order to find the best Hyperparameters whose accuracy on the validation set is the greatest one.

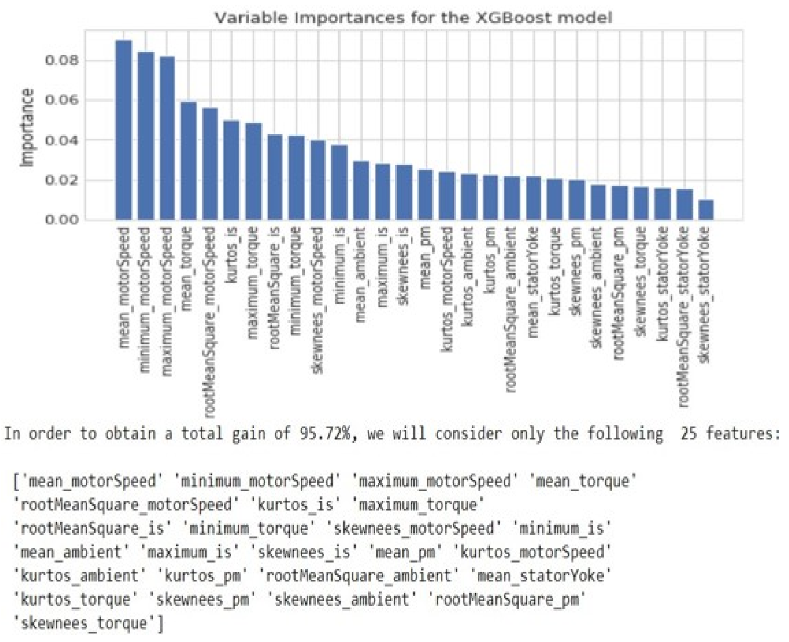


```
              precision    recall  f1-score   support

           0       0.86      0.67      0.75         9
           1       0.63      0.86      0.73        14
           2       0.88      0.82      0.85        65
           3       0.70      0.76      0.73        25

    accuracy                           0.80       113
   macro avg       0.77      0.77      0.76       113
weighted avg       0.81      0.80      0.80       113
```

Figure 3.26: Results of the XGBoost model whose parameters are chosen with a Grid Search

The outcomes are:

- *learning rate* = 0.1;
- *max depth* = 10;
- *min_ child_ weight* = 1;
- *gamma* =0;
- *colsample_ bytree* = 0.6;
- *n_ estimator* = 200.



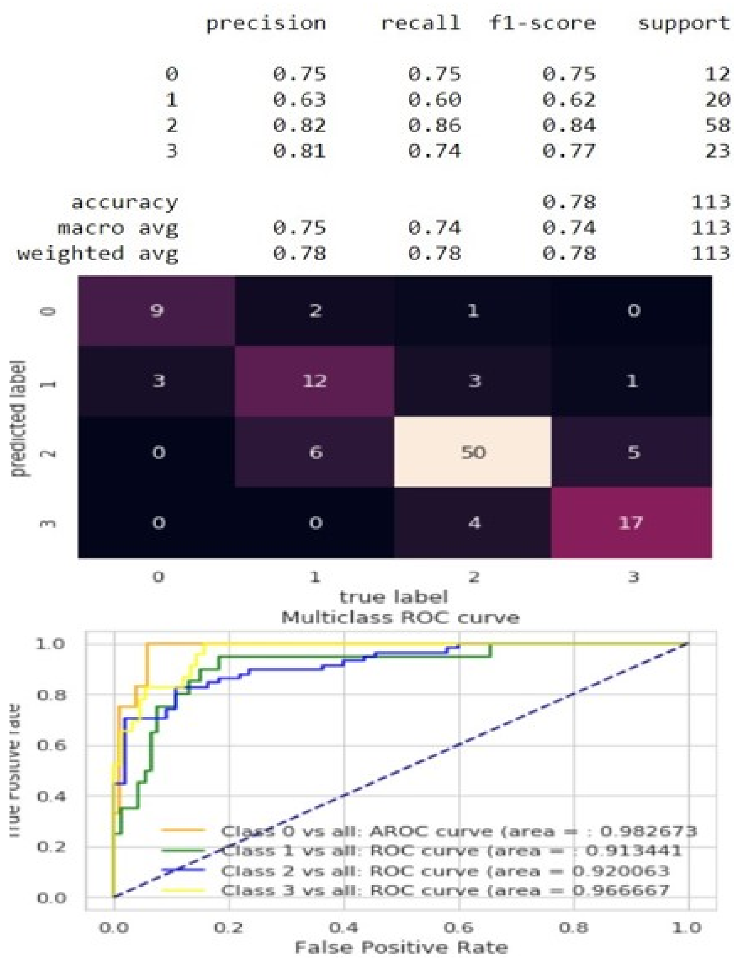
```
In order to obtain a total gain of 95.72%, we will consider only the following  25 features:

['mean_motorSpeed' 'minimum_motorSpeed' 'maximum_motorSpeed' 'mean_torque'
'rootMeanSquare_motorSpeed' 'kurtos_is' 'maximum_torque'
'rootMeanSquare_is' 'minimum_torque' 'skewnees_motorSpeed' 'minimum_is'
'mean_ambient' 'maximum_is' 'skewnees_is' 'mean_pm' 'kurtos_motorSpeed'
'kurtos_ambient' 'kurtos_pm' 'rootMeanSquare_ambient' 'mean_statorYoke'
'kurtos_torque' 'skewnees_pm' 'skewnees_ambient' 'rootMeanSquare_pm'
'skewnees_torque']
```

Figure 3.27: Important feature of the XGBoost for the classification problem

Let's say that this last model has found a sort of equilibrium between the classes (see Figure 3.26): all the statistics are greater or equal than 70%, except from the *recall* of the class 0, since the algorithm has predicted correctly only 6 values on 9, and the *precision* of the class 1 whose predicted records have been 19, but only 12 were right. The overall accuracy was equal to **79,33%**.

XGBoost with features selection

With the idea of reducing the computational effort, a feature importance analysis is performed on the XGBoost model.



```
              precision    recall  f1-score   support

           0       0.75      0.75      0.75        12
           1       0.63      0.60      0.62        20
           2       0.82      0.86      0.84        58
           3       0.81      0.74      0.77        23

    accuracy                           0.78       113
   macro avg       0.75      0.74      0.74       113
weighted avg       0.78      0.78      0.78       113
```

Figure 3.28: Global results of the XGBoost with optimal settings and features importance analysis

In Figure 3.27 is shown the relative importance of each feature in the XG-Boost model. We can see that also here the stator's temperature of the yoke is useless in order to find the best split in the model and, as consequence, to better classify the mean voltage vector.

The overall accuracy achieved by this model is **79,04%**, and the confusion matrix along with the multiclass ROC curves is presented in Figure 3.28. Again, no big changes happen compared to the previous one, due to the fact that the features removed do not have a big impact on the performances of the model itself.
The 3% less accuracy of this model with the previous one is due to the usual randomness with which the graphic results are generated.

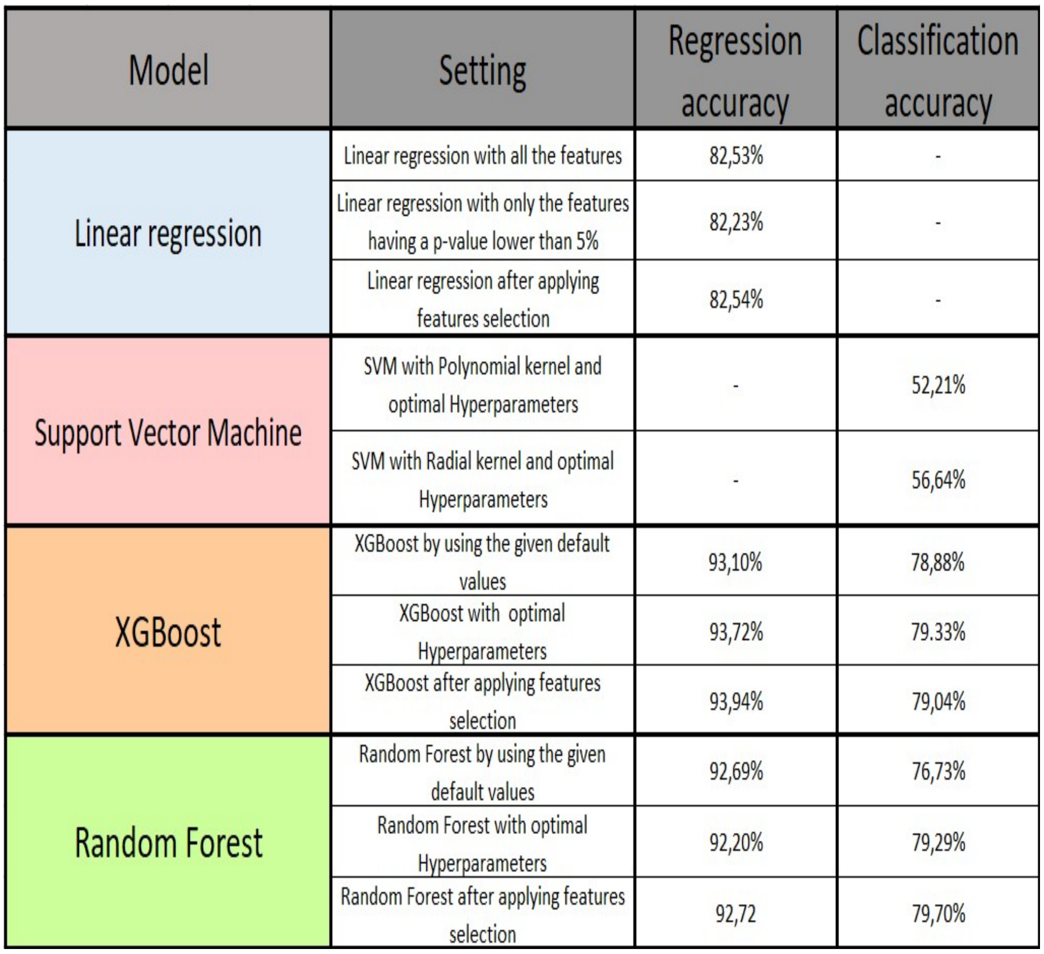
Conclusions

The objective of this thesis is to develop a framework suitable for Predictive Maintenance in the Electric Vehicle context, by joining Zirak's backend with specially researched and tested machine learning algorithms.

Figure 3.29 contains a summary of the mean accuracy achieved by applying different models for both classification and regression purposes .

By looking at the metric, it can be said that the goal of obtaining an excellent prediction model for the continuous standardized values of the battery's mean voltage vector, has been accomplished, since the XGBoost with optimal Hyperparameters and a selection of the features has achieved an accuracy of almost 94% and the extreme values have been perfectly predicted except for two of them (see Figure 3.14).

On the contrary, the performance for classification purposes could be in some way improved: if the last Random Forest trained seems to be the best, in reality, there aren't many differences in the performance with the XGBoost with all features and optimal Hyperparameters.
In any case, the fact that fewer features are used in training the RF, makes it the preferable model to apply.
Let's suppose that a company, for some reasons, has an issue to install sensors that take in real-time the temperature of the stator yoke, and suppose that, after applying a features selection, all the statistics related to the stator yoke are removed because their little importance (in the RF model of section 3.4.2



Model	Setting	Regression accuracy	Classification accuracy
Linear regression	Linear regression with all the features	82,53%	-
	Linear regression with only the features having a p-value lower than 5%	82,23%	-
	Linear regression after applying features selection	82,54%	-
Support Vector Machine	SVM with Polynomial kernel and optimal Hyperparameters	-	52,21%
	SVM with Radial kernel and optimal Hyperparameters	-	56,64%
XGBoost	XGBoost by using the given default values	93,10%	78,88%
	XGBoost with optimal Hyperparameters	93,72%	79.33%
	XGBoost after applying features selection	93,94%	79,04%
Random Forest	Random Forest by using the given default values	92,69%	76,73%
	Random Forest with optimal Hyperparameters	92,20%	79,29%
	Random Forest after applying features selection	92,72	79,70%

Figure 3.29: Global final summary of the results.

only *rootMeanSquare_statorYoke* and has been taken), this result is great. The company will get similar performances without the data that come from those sensors and it will avoid the problem of installing them, saving resources on the way.

Next steps However, other experiments could be carried out in order to achieve better classification performance.
It would be suitable to train these models on a dataset with the information on the occurrence of battery failures.
If this happened, the classes distribution would have been different and likely

also the performance. Still, some Neural Network (NN) could be reasonably performed in this context, as suggested from the papers in the last paragraph of the first work's chapter, where Feed Forward back loop NN and CNN had great results.

Bibliography

[1] Clarisa García Novoa - Gerardo Antonio Guzmán Berríos - Rodrigo Abrego Söderberg. "*Predictive Maintenance for Motors Based on Vibration Analysis with Compact Rio*". IEEE Central America and Panama Student Conference (CONESCAPAN). IEEE Exploree/2017

[2] Paolanti M., Romeo L., Felicetti A., Mancini A. Frontoni E., "*Machine Learning approach for Predictive Maintenance in Industry 4.0*". 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA). 2018.

[3] Murat Çınar Z., Abdussalam Nuhu A., Zeeshan Q., Korhan O., Asmael M., Safaei B., "*Machine Learning in Predictive Maintenance towards Sustainable Smart Manufacturing in Industry 4.0*". MDPI. 2020

[4] Kavana Venkatesh - Neethi M. "Fault Analysis and Predictive Maintenance of Induction Motor Using Machine Learning". Research Gate. 2018

[5] Chia-Yen Lee - Ting-Syun Huang - Meng-Kun Liu - Chen-Yang Lan. "Data Science for Vibration Heteroscedasticity and Predictive Maintenance of Rotary Bearings". MDPI. 2019

[6] Kahiomba Sonia Kiangala - Zenghui Wang. "An Effective Predictive Maintenance Framework for Conveyor Motors Using Dual Time-Series

Imaging and Convolutional Neural Network in an Industry 4.0 Environment". IEEE. 2019

[7] Kirchgässner W., Wallscheid O., Böcker J., "Deep Residual Convolutional and Recurrent Neural Networks for Temperature Estimation in Permanent Magnet Synchronous Motors". ResearchGate (2019)

[8] "Kurtosis | Wikipedia".
https://en.wikipedia.org/wiki/Kurtosis

[9] "Skewness | Wikipedia".
https://en.wikipedia.org/wiki/Skewness

[10] “Feature Extraction Definition | DeepAI.”
https://deepai.org/machine-learning-glossary-and-terms/feature-extraction

[11] “Feature engineering - Wikipedia.”
https://en.wikipedia.org/wiki/Feature_engineering

[12] "Permanent Magnet Synchronous Motors (PMSMs) | MICROCHIP"
https://www.microchip.com/design-centers/motor-control-and-drive/motor-types/permanent-magnet-synchronous-motor

[13] "Synchronous motor | Wikipedia"
https://en.wikipedia.org/wiki/Synchronous_motor

[14] Gasparini M., *Modelli probabilistici e statistici.* Torino, Clut (2014), Seconda edizione, pp. 99-114.

[15] Hastie T., James G., Tibshirani R., Witten D., *An Introduction to Statistical Learning.* Springer. pp. [60-92, 205-259, 303-330, 337-366].

[16] “How to Use One-vs-Rest and One-vs-One for Multi-Class Classification. | Machine Learning Mastery”
https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/

[17] Friedmann J., Hastie T., Tibshirani R., *The Elements of Statistical Learning, Data Mining, Inference and Prediction*, Springer, Second Edition, pp. 337-360.

[18] "Feature selection — Correlation and P-value | towards data science". https://towardsdatascience.com/feature-selection-correlation-and-p-value-da8921bfb3cf

[19] "How XGBoost works | AWS" https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost-HowItWorks.html

[20] "What is Data Cleaning? | Sisense." https://www.sisense.com/glossary/data-cleaning/

[21] "Data Cleaning in Machine Learning: Best Practices and Methods." https://www.einfochips.com/blog/data-cleaning-in-machine-learning-best-practices-and-methods/

[22] "Understanding Confusion Matrix | Towards Data Science." https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

[23] "Receiver operating characteristic | Wikipedia". https://en.wikipedia.org/wiki/Receiver_operating_characteristic

[24] "sklearn.ensemble.RandomForestClassifier | scikit learn". https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[25] "Hyperparameter Tuning the Random Forest in Python | Towards data science". https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74#: :text=In%20the%20case%20of %20a,each%20node%20learned%20during%20training).

[26] "Feature importances with forests of trees | scikit-learn.”.
https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html#

[27] "XGBoost Parameters | XGBoost"
https://xgboost.readthedocs.io/en/latest/parameter.html

[28] "Improve Your Model Performance using Cross Validation (in Python and R) | Analytics Vidhya"
https://www.analyticsvidhya.com/blog/2018/05/improve-model-performance-cross-validation-in-python-r

Acknowledgements (Ringraziamenti)

Ah, finalmente un pò di italiano...

Ci tengo a precisare che questo spazio non è dedicato al percorso in sé a cui questo elaborato fa riferimento (laurea magistrale), ma a tutto ciò che ne è stato di questi miei anni in questa buia, inquinata, pericolosa... ma anche ricca di luoghi, stimolante (dal punto di vista professionale) e affascinante città di Torino.
Non vi sarà un ordine con cui le persone, cose o animali vengono citati. Sarà dato dal caso.

Ringrazio innanzitutto l'azienda Zirak per avermi dato la possibilità di svolgere il mio lavoro di tesi in un luogo interessante e dinamico, che mi ha permesso di mettermi in gioco e fare un'importante esperienza lavorativa.
Ringrazio inoltre la Prof.ssa Tania Cerquitelli, relatore di questa tesi che mi ha trasmesso quel know-how necessario affinché la tesi prendesse forma.

Eh poi beh, veniamo a noi...

La prima persona a cui devo i miei ringraziamenti è me stesso. Laurearsi al Politecnico di Torino, in Ingegneria Matematica, ed ora ci starebbe bene 'con 110 e magari qualcos'altro'..., non è da tutti. Senza la mia tenacia e caparbietà, forse non sarei andato da nessuna parte. Ho dovuto rinunciare a tanto però. La pallacanestro italiana piange ancora la scoperta del più grande playmaker mai esistito, la mia cute tende con forza a nascondersi dai loschi pensieri della gente e il ph del mio stomaco conosce solo i valori delle temperature invernali. Ma non solo questo. Ho abbandonato quasi ogni forma di rapporto sociale: si contano le serate con un tasso alcolico sopra la media, i gesti romantici alla mia fidanzata e le giornate passate in famiglia senza pensieri. Ho evitato ogni forma di lavoro manuale per la manutenzione della mia casa in campagna ed ho lasciato a quelle grandissime persone di mia mamma e mio zio prendersi tutto il carico, l'amore e la gioia derivante dallo stare vicino ai miei nonni.
Non so quanto ne è valsa la pena, ma in ogni caso fiero di essere quel che sono.

E' doveroso a questo punto ringraziare la mia Mamma. Senza dubbio il motore portante di tutto ciò.
Non so quanta gente si sarebbe fratturato due vertebre, tre spondilolistesi L3-L4, L4-L5, L5-S1 al punto quasi da comprimere il canale vertebrale, due interventi di emorroidectomia, decompressione del nervo mediano per la sindrome del tunnel carpale, un intervento di colecistectomia (se qualcuno chiama al rialzo, mi interrompa pure), in procinto di qualche assaggio di chirurgia estetica alla parte inferiore

del corpo... il tutto per regalare un futuro più che dignitoso ai propri figli. Io non l'avrei fatto.
Ogni mio pensiero era anche il suo, ogni mia ansia era anche la sua ed anche ogni mia gioia è stata la sua. Cosa non banale, perché la felicità fine a se stessa, non la si può chiamare tale.
Da ammettere che non sempre è stata provvista del consiglio giusto, ma ero ben conscio che lei avrebbe appoggiato e sostenuto ogni mia scelta, e qualunque fossero stati i risultati, li avrebbe condivisi, nel bene o nel male. Devo tanto a questa donna, questo traguardo è anche il suo!
Grazie Mamma!

Seguo la scia e ringrazio mio papà. Banale dire che una gran parte delle finanze che un percorso del genere lascia inevitabilmente per strada, sono state le sue.
Ma non c'è solo questo. Il suo modo di rispondere e dare consigli nei giorni in cui gli ho comunicato un mio problema, il più delle volte irruento, mi dava una sorta di serenità. La persona in questione non mi fraintenda, questa serenità non è dipesa dall'utilità del consiglio, ma dalla leggerezza con la quale è stato dato. Non è stato facile, ma col tempo, ho imparato ad integrarlo e renderlo un fattore positivo in quel mio modo di affrontare la vita che è perfettamente riassunto dalle sue parole: 'Figlj mij tu piens tropp'.

Va beh a questo punto già che ci siamo ringrazio anche mia sorella. Senza dubbio il componente più colto della mia famiglia (prima di oggi) da cui ho imparato molto soprattutto nella prima parte di questo percorso.
L'ansia di un esame, il fatto di non essere valutati per ciò che si merita, le sensazioni di non superare un modulo o addirittura pensare: 'no va beh, questo è troppo difficile', io le conoscevo già.
Dove i consigli tecnici di mia mamma non erano adatti (vedi un paio di paragrafi sopra), lo diventavano i suoi.
Nella seconda parte di questo percorso i rapporti si sono un pò inclinati, ma ho spesso agito usufruendo delle sue parole. Ricordo ancora di aver accettato l'unico 26 perché in triennale lei mi disse: 'ma se rifiuti un voto del genere, a che costo lo riesci ad aumentare?'. Inoltre è stata d'esempio per le mie (poche rispetto alle sue) opere di carità, in quanto molto incline ad aiutare le persone più bisognose. Quando dopo 12-13 ore di studio, tutto sporco e puzzolente, mi mettevo una tuta bucata per andare al supermercato e decidere cosa mangiare (la tuta bucata non è citata a caso), beh quella cena donata al ragazzo che non ne aveva, dava un senso alla mia giornata.
Ricordo ancora con molto piacere le tappe fisse nel centro Italia prima di andare a Benevento, le partite a beach volley in estate e quella nuotata in cui provato, tentato e fatto di tutto per raggiungerla ma non c'è stato niente da fare nonostante partissi con un piccolo vantaggio... il mio avversario non vedeva! Quelle giornate sono state un ottimo momento di ricarica di energia.
Spero di fare ancora tesoro della sua esperienza, anche perché il lavoro incombe e i fallimenti sono dietro l'angolo (notare l'ironia...).

Ringrazio a questo punto ogni singola persona incontrata in questo cammino, perché quando un percorso così lungo termina in modo importante, ognuno ha fatto la sua parte. Nominarli tutti è impossibile però ricordo con piacere tutti i miei coinquilini Francesco, Arianna, Federica, Luca, Alessandro, Amin ed Ahmad da cui ho imparato ad adattarmi, condividere (con una qualche difficoltà) gli spazi comuni ed aprire la mia mente a nuove culture. Ringrazio i miei cognatini Mara e Jonathan perché di grande esempio per me nel vedere i propri studi ripagati con la giusta moneta (e non solo in senso figurativo). Ringrazio i ragazzini Marcolino, Nando, Michele e Vincenzino perché hanno aumentato di gran lunga la mia

autostima, sempre riconoscenti dei miei consigli. Infine ricordo con piacere le persone incontrate a Londra Sofia, Keno, Priyanka e soprattutto i miei cugini, Fabio e Sach, che hanno trasformato quell'esperienza da un potenziale 'terribile', ad uno splendido 'piacevole'.

Ringrazio ancora, Francesco, Giuseppe e Timothee perché hanno reso più dinamico questo percorso con qualche giornata al campetto e non solo... Ringrazio i miei carissimi amici Manuel e Gianmichele, che anche se non siamo riusciti a vederci tanto, sono sempre stati pronti a regalarmi chiacchierate piacevoli e momenti rilassanti. Molto simile alla situazione con mio cugino Michele dove le telefonate erano incentrate su piatti da cucinare, partite a play-station o formazione da schierare al fantacalcio.

Ringrazio anche la mia nonna Grazia, che molto simile a mio padre (sono l'una la mamma dell'altro), mi ha sempre voluto trasmettere quella sorta di spensieratezza e modo di agire 'd'istinto' che avrebbe sicuramente portato ad una miglioria nella mia vita e nel mio studio. Il tutto con l'esperienza di una nonna però...

Ringrazio poi i miei colleghi, ma soprattutto amici, Michele, Patrick, Sara, Serena, Giulia, Luca e Vittorio che hanno accompagnato in toto questi miei anni. Splendide persone con le quali ho trascorso la maggior parte del mio tempo. Ricordo giornate intere davanti ai libri o al PC, a cercare disperatamente un 'se e solo se' o a risolvere un difficilissimo sistema lineare di 12 incognite in 12 equazioni. I progetti sviluppati insieme sono troppi se paragonati alle bottiglie di vino stappate, spero che questa tendenza possa cambiare in futuro.

Non mi sono affatto dimenticato. Se c'è una persona a cui devo un grande Grazie, è il mio Amore. Ho iniziato questo percorso insieme a lei o lo concludo con lei al mio fianco.
Devo essere sincero, nessuno mi ha sopportato quanto lei. E non deve esser stato facile...
Qualunque esame andato male era pronta a subire la mia rabbia e tranquillizzarmi, ed io sapevo di poter andare da lei perché mi avrebbe dato sollievo o magari non creduto alle mie impressioni, dandomi ancora una speranza in un voto migliore.
L'ho spesso accusata di distarmi o di interrompere il mio studio, o di non lasciarmi libero di svagare come io volessi. Ma erano tutti alibi che cercavo perché la stanchezza aveva preso il sopravvento e la necessità superava la volontà di stare ancora sui libri.
Ho quasi sempre apprezzato quel suo modo di pretendere attenzioni perché mi dava la spinta necessaria per terminare il prima possibile quel mio studio, altrimenti sarebbe stato un incubo sostenere a lungo quelle pretese. Ho sempre amato quel suo modo di addolcire le mie giornate con qualche bacino o con qualche sguardo un pò triste ma affascinante. Quel suo modo di regalarmi un sorriso con qualche saltello a cofanetto o dei balletti improvvisati partiti dalle note di una stupida canzoncina appena sentita in pubblicità o ancora per delle balorde posizioni che una gatta assumeva durante la giornate. O quel suo modo di passare ore ed ore a fantasticare su possibili situazioni future, su nomi bizzarri da dare ai figli, le 10 combinazioni di piatti da fare a tavola o su matrimoni da organizzare su un monte a 750 mt di altezza dal livello del mare e con 7 scalini farsi il bagno. Sempre a mare, intendo...
Ho semplicemente amato il modo in cui sei stata te stessa. Grazie Amore!

Eh sì, tra tutti c'è anche una gatta che è stata parte della mia vita senza la quale molti dibattiti o chiarimenti non sarebbero avvenuti: come educarla, cosa darle da mangiare o come farle la pulizia anale, come dividere 90 o 120 cm di materasso tra me, la mia principessa ed una gatta. A parte ciò, l'energia che trasmettevano le sue fuse e l'incanto del suo sguardo sono sufficienti per meritarsi un grazie.

Spendo ancora due parole per ringraziare colui che rappresenta molto più di un cugino. Antonio è stato a tratti un mio coinquilino, spesso un amico, ma anche e soprattutto un fratello.
Ricordate i soldi lasciati per strada da mio padre? Bene, io con lui ne ho persi una cifra simile tra bollette e fantacalci. Diciamo però che sono stati ripagati dalle tante partite a ping-pong (una competizione che nemmeno Nadal-Federer si possono immaginare), le giocate a play-station, il beach-volley, le pizze, le partite a poker. Le chiaccherate fino a tarda sera a ragionare su quanto avrei guadagnato una volta uscito da questo ateneo (35-40 i primi anni, almeno 70 il secondo).
Le chiamate a mezza mattinata del week-end che iniziavano: 'oh ma a quant stamm?', oltre ad uno sperpero di denaro rappresentavano per me un momento di gioia e condivisione.
Ho avvertito sempre grande stima da parte sua nei miei confronti, e questo mi ha dato una forte carica. Io ne ho altrettanta per lui!

Desidero ringraziare infine anche mio zio Rolando. Sono sempre riuscito a comunicare con lui in modo speciale, sapevo di poter contare su di lui per qualunque consiglio, necessità o bisogno, e sapevo che lui avrebbe fatto lo stesso qualora io avrei fatto o potuto fare qualcosa che non andava.
Ho imparato molto da lui ed oggi mi ci rivedo molto. Quel suo modo di ragionare, di prendere decisioni, di organizzare una tavola, di intrattenere un'ampia platea o di apprezzare un buon bicchiere di vino, che qualcuno addita erroneamente con la frase 'tu si tropp lient', io lo definirei 'saggezza'.

Ringrazio infine un oggetto, che apparentemente dalla una forma perfetta ma con qualche imperfezione per accogliere dei polpastrelli, i miei: la palla da basket. In ogni dove, in ogni quando e perché bastava che c'era lei per liberare la mia mente e cambiare il mio umore. Quell'oggetto che magicamente esce dalla mia mano per ritornare lì, e poi con tocco sopraffino entra in un cestino posto a 3.05mt da terra (quando sono fortunato) è assolutamente da meritarsi una citazione.

Arrivo finalmente alla conclusione di questo discorso. Mi sembra riduttivo chiamarlo un ringraziamento... questa è una dedica, una lode alle persone più importanti della mia vita, i miei nonni, Nonna Maria e Nonno Tonino.
Se dopo 19 anni di studio ho avuto il coraggio di intraprendere l'università e continuare gli studi, è grazie ai loro insegnamenti, alla loro fiducia nelle mie capacità e al fatto di credere nella cultura come fonte principale di realizzazione personale.
Mio nonno mi ripeteva spesso: "studia a nonno che poi puoi entrare in politica" e la nonna mi chiedeva spesso : "ma allora a nonna esci col massimo dei voti?".
Ovviamente io in politica non ci sono entrato e non so con quale voto mi sarò laureato, ma sono convinto che saranno fieri di me.
Oggi il mio cuore è triste per non poter condividere con loro questo momento, voglio però che sappiano che ho vissuto tutti questi anni seguendo i valori ed i principi che mi hanno insegnato.
Se mi stanno vedendo da lassù, vorrei che mi immaginassero abbracciati a mia nonna con il braccio sinistro attorno al suo fianco e la mia mano destra incrociata con quella di mio nonno. Tac.. la festa è finita.
Li accompagno a casa e dormo da loro perché non possono restare soli, porto nonna a letto, le do un bacio e le auguro la buona notte. Vado sul divano e mi addormento scoperto perché sono stanco, mio nonno viene e mi 'accommuglia' le coperte. Mi bacia e mi dice: 'buona notte a nonno'... Ciao nonni.
Grazie di essere quel che siete stati.