POLITECNICO DI TORINO

Department of Control and Computer Engineering

Master's degree program in Computer Engineering

Master Thesis

Predictive maintenance of industrial vehicles based on supervised machine learning techniques

Candidate: Saverio Cinieri

Supervisor: prof. Luca Cagliero

December 2019

Contents

List of Figures

Acknowledgements

Special thanks to Professor Luca Cagliero, who helped me in this thesis work, Luca Vassio and Dena Markudova for their assistance and their valuable suggestions.
Thanks also to the employees of Tierra S.p.A for their precious availability.
Last, but not least, thanks to my mother, my father and my sister for having always supported and helped me in the most difficult moments of my university career. Furthermore, my friends with whom I shared periods of study, confrontation and fun. Thank you all for always being there and for always making me feel at home.

Chapter 1

Introduction

The remarkable development of Information Technology has improved many aspects of our lifes, making easier tasks usually performed by humans and decreasing the probability of error associated. As regards the positive contribution of these new technologies to the field known as Data Science, automatic and innovative techniques have been used to collect and process the data.

Deepening the analysis related to the collection of data it is inevitable to introduce the term **Internet of Things** identified by the abbreviation IoT, that is one of the most popular IT word of the moment. The term has been introduced twenty years ago more or less and Kevin Ashton, co-founder of MIT's Auto-ID Center, is credited by most sources with coining it. IoT identifies a network of devices connected exploiting a wireless connection that collect a set of data in any kind of environment, and finally redirect this stream to a cloud infrastructure. The automotive field has enjoyed the benefit of IoT widely replacing the old techniques based on cumbersome and heavy cables with the CAN(Controller Area Network) bus technology. This kind of technology is based on a smart network that allows to connect easily a certain number of devices employed to collect data, moreover they solved most of the problems related to the previous technologies. For instance, the centralized error management ensures that the collisions between messages sent by different sensors in the same moment are transmitted again with a short delay, furthermore the good resistance to external disturbances represents an innovative feature. The automotive field, benefiting from this innovative tool, identifies a large variety of heterogeneous vehicles. For instance the normal cars or street sweepers used in a urban context or

further vehicles employed in a more specific environment(excavators, refuse trucks, etc.)

The data collected with the CAN bus is used mainly to represent statistics and predict the most important aspects associated to the vehicles, for example the fuel consumption, the usage in a future time period or when a predefined set of operations must be performed on a vehicle to prevent failures of critical issues. The latter application is generally called predictive maintenance and is the real-time analysis and monitoring operation to check all the values related to the performance of an asset. If the data collected exceeded a threshold value certain actions must be performed. Forecasting means applying a set of techniques, belonging to the subfield of artificial intelligence, known as Machine Learning.

The work done in this thesis has been carried out in collaboration with TIERRA SpA, which is a private company offering telematics services to industrial vehicle makers. The considered vehicles are equipped with innovative devices and are located on working sites, a large variety of machines are available(e.g. excavator, street sweeper machines, refuse compactor, etc.). TIERRA SpA has collected a large amount of historical vehicle usage data from a large set of heterogeneous vehicles located in different countries, data describes per-vehicle usage patterns in terms of utilization hours, fuel consumption, the info indicating the usage threshold at which the maintenance operation must be performed and further aspects associated to the vehicle usage. The data were aggregated on a daily basis and are collected during a period of four years in the best case, or two years in the worst case. The consideration on the available data suggest us that they are ordered observations equally distributed over the time, technically they are a **Time Series**. Like any data set subjected to a deep analysis, a set of operations must be performed to transform the raw data provided by TIERRA Spa in a more representative format, this phase is usually known as data preparation. In our case the data belonging to different tables are integrated in a relational dataset, preprocessed using established data mining techniques, and transformed to perform further analysis.

Main goal of this thesis is to predict, using Machine Learning algorithms(supervised techniques), how many days/weeks until to the maintenance and if it will be necessary to carry out maintenance on the next day or week, all the ML models were

trained exploiting the historical vehicle usage series. This task is linked to achieving a usage threshold expressed in hours for most vehicles. The correlation between the usage of a vehicle and the maintenance aspects required an analysis aimed at illustrating for each model the usage patterns, and the usage values distribution. These parameters are relevant because allow understanding if a single prediction model can be used for different models, or vehicles of a same model, and how well the regression algorithms could work in general having to forecast intervals of values more or less limited.
We need to predict either the number of days/weeks to maintenance or whether the next day/week a maintenance operation will be due, for this reason regression and classification algorithms have been applied choosing techniques based on different approaches: Linear Regression, Support Vector Machine, Random Forest(based on bagging) and Gradient Boosting(based on boosting). As regard the regression phase, the mentioned algorithms have been applied following two methodologies:

- **Methodology 1**: the target variable we forecast is *days_to_maintenance* or *weeks_to_maintenance*, we identify it with the name **PredictDays2Maintenance** and **PredictWeeks2Maintenance**

- **Methodology 2**: this approach predict the utilization hours on the next days until the usage threshold is reached. However, we can exploit these forecasts to achieve the values of *days_to_maintenance*, computing the difference in days between the date the threshold is reached and the date on which we make the prediction. We identify it with the name **CumulateDailyPredictions**

To quantify the results achieved with the two methodologies the residual error was computed for the weeks and days closer to the date of maintenance task(thirty days and ten weeks), for classification algorithms the classic metrics precision, recall and accuracy have been used.
The final results showed greater efficacy for regression rather than classification techniques, due to the nature of unbalanced data on a class label. Moreover, as regards the regression algorithms and the two methodologies used, large differences associated with execution times and the accuracy of results and have been observed, the first presented methodology predicting the remaining time to the maintenance

scored the best results. Remedial actions (i.e. choice of a fixed size training set) have been taken to improve the prediction model training and the error obtained in the test phase, especially for the algorithms based on bagging(RF) and boosting(GB). However, all the regression algorithms achieved results better than the chosen baseline especially on a weekly basis, therefore future studies on the subject should tend to follow this approach.

A short summary is provided to present the content of the following chapters:
Chapter 2 describes in short the context related to the IoT with an extensive reference to the CAN bus, and in particular the use of the data collected with this technology for the predictive maintenance and predictive data-mining in general. Later, the basic aim of this activity is explained in details.
Chapter 3 lists some scientific papers consulted to have an overview of the work done about the application of predictive data-mining with vehicular data. The difference with this work is pointed out.
Chapter 4 is a short reminder about the basic principles of Machine Learning, the regression algorithms used in this thesis and the theory of time series.
Chapter 5 presents in detail the type of data and the preprocessing phase(e.g. data aggregation, features engineering, etc.) to set up the structure of the final dataframe on which the data exploration and the predictive analysis were executed.
Chapter 6 contains the description of the steps followed to perform data exploration. Different kinds of plots will be exploited to visualize the distribution of target variable values, all the results collected will give a general idea concerning the application of regression techniques and the possibility of grouping vehicles of same and different models.
In Chapter 7 the methodologies/algorithms applied along with the regression algorithms on the dataset are explained in details pointing out the associated advantages and disadvantages.
Chapter 8 points out all the aspects relative to the experiments performed and the most important results we obtained. In particular for each methodology and algorithm applied the regression/classification errors, and tuning results are compared

to understand which of them is best to use in this context.
Chapter 9 points out the final considerations and describes briefly the future activities that can be worked on based on the results obtained in this thesis.

Chapter 2

Context overview

2.1 CAN bus and IoT technology

The need to collect huge amount of data has required an increased use of technology, especially in certain sectors such as automation, automotive field and industrial areas in general. At the beginning of the 80's new technologies have been developed to accomplish this task, for instance PROFIBUS and INTERBUS replaced the old serial communication tools used in the automation world [28]. In the automotive field a system based on Electronic Control Units(ECUs), sensors and actuators was introduced, however this technology wasn't a good solution considering the non-safe data transfer among the connected modules and the bulky amount of cables.
With the introduction in 1986 of a new tool called 'Controller Area Network'(CAN) most problems mentioned above were solved. Some years after the first public presentation the most import electronic companies presented their first implementation of this technology.
The CAN bus can be seen as the connection system enabling communication between all parts of a network, more or less complex. The link is implemented without complex dedicated wiring and exploiting a proper protocol.
The advantages provided by the CAN technology can be summarized in the next points, as explained in [28]:

- **bus access safe method and centralized errors management** - in the previous technologies based on Ethernet the nodes of a network started transmitting when there was no transmission, however if two or more nodes started

transmitting at the same time there were collisions. This conflict had to be solved destroying the messages already sent and delaying the transmission. With the innovative mechanism of CAN each message has a level of priority and when a collision occurs all the messages are queued and then sent according to their level of importance, the nodes of the network don't have to transmit again their messages because they are stored safely. The network is in charge of managing this kind of problems.

- **data consistent transmission** - if any kind of error occurred and is identified by a node a notification is sent to the other ones, in this way the erroneous messages are discarded and a new trasmission is required

- **resistance to external disturbances** - the bus electrical structure based on two twisted lines and the symmetric transmission compensate electromagnetic interference

- **flexible inclusion of additional nodes** - all the nodes connected to the CAN bus communicate with a single CAN interface, the main positive aspect associated to this feature is that is not necessary to repeat for a new single device connected to the network a multiple configuration in different points.

The CAN technology has been improved especially considering its two main relevant drawbacks. The first kind of CAN bus allowed the transmission of a single message not bigger than 8 bytes, this problem was simply solved splitting a single message(with a size greater than 8 bytes) in multiple units however they couldn't arrive at the same time. The second problem was the low bit rate. A new version of CAN bus, called CAN bus flexible data-rate, was introduced in 2012 with new features addressed to solve the limitations mentioned above, the new key solutions are the transmission up to 8Mbit/s and data packets of 64 bytes. Nevertheless, it still represents a popular technology only for the automotive field.

The use of CAN needs to be assisted with a proper environment equipped with further technologies that should able to store all the data collected, in this regard the role played by the cloud services is crucial. Once the data are collected then they are

transferred to the cloud that are able to offer different services(e.g. preprocessing, displaying, etc.) in an easy-to-read format.

2.2 Preventive, reactive and predictive maintenance

Focusing the attention on the main topic faced in this thesis is necessary to introduce the problem known as predictive maintenance(PdM), that is typically addressed exploiting the information contained in CAN messages or collected with devices used by IoT. In many industrial sectors the maintenance is splitted in three different subclasses:

- **preventive maintenance**(or preventative maintenance, PM): *'it is maintenance that is regularly performed on a piece of equipment to reduce the probability of failure'* [15]. This type of maintenance is carried out before the component of a system breaks and is programmed considering the achieving of usage thresholds(e.g the device has accomplished a certain operation a number of times) or time thresholds. To notify that the maximum limit has been exceeded a signal is properly set. From a practical point of view this kind of maintenance can be performed only using a predefined set of software that allow to set special operations according to the triggers that are appropriate for each piece of equipment. A proper action must be performed once the trigger occurs.

  Planning maintenance allows reducing costs associated to breakdowns that include lost production, higher amount of money spent to replace an entire component, and finally time lost to repair the failures that have occurred. Finally, the usage of condition monitoring equipment is not needed and so the related costs are eliminated. All the positive aspects cited don't have to overshadow the investment required in time and resources required by PM, moreover the maintenance tasks should be planned with meaningful frequencies, a high or low number of maintenance operations in a certain time period would be inefficient.

- **reactive maintenance**(RM): in contrast with PM it refers to a collection of tasks executed when a component is already broken down and for this reason requires an effort only when it is strictly necessary. This approach has many weaknesses: first it is time-consuming(e.g. many operations planned could be pushed or cancelled completely) and requires an expensive economic effort, secondly a restored component will be subjected to faster deterioration. In addition, we must consider the element of surprise to which technicians are subjected when a breakdown occurs.

- **predictive maintenance**(PdM): this kind of methodology can be considered a solution to prevent unplanned reactive maintenance, with all its associated weaknesses, and the costs associated with doing too much preventive maintenance. The primary purposes are to predict when a component will break down and secondly to prevent the occurrence of the failure by performing maintenance. A real-time analysis and monitoring operation allows to check all the values related to the performance of a system component, the most important tools to perform these kinds of monitoring tasks can be grouped in the IoT(of course CAN bus technology belongs to IoT).

  Comparing PdM with PM, the first one ensures that a piece of equipment requiring maintenance is only shut down right before an imminent failure, or at least when the damage caused by exceeding the fixed threshold is not so limiting for the entire device kept under observation. This allows to save time and money to repair the failure that has occurred. On the other hand we must consider all the costs for human and material resources involved in this process.

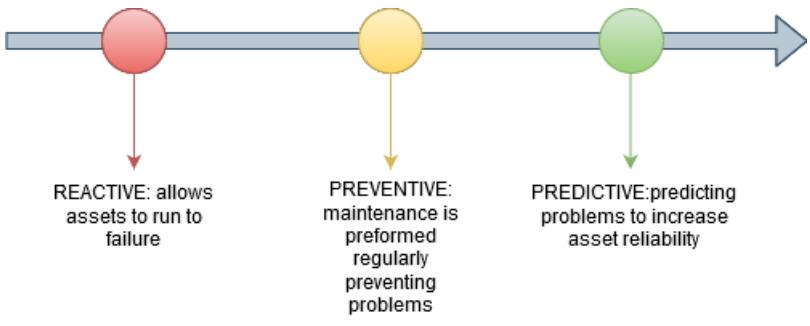


Figure 2.1: Key concepts for reactive, preventive and predictive maintenance

Considering the descriptions provided for all the methodologies presented, one can say with certainty that RM features many drawbacks if compared with PM and PdM. On the other hand it is not possible to say if PdM is better than PM or viceversa, a deeper analysis on a particular asset could justify an approach rather than the other.

2.3 General considerations and the main goal

This thesis has been developed thanks to the collaboration between Politecnico di Torino and TIERRA Telematics, moreover represents the continuation of a work previously started, which aims to analyze with data mining and machine learning techniques the large amounts of data collected(mainly data related to industrial vehicles). TIERRA is an international company working mainly in IoT providing telematics solutions to collect, preprocess and manage data from vehicles, machinery and objects. The typical solution to accomplish the tasks mentioned above is based on a simple technology: an on-board device, a sim-card valid in 194 countries, a cloud infrastructure for data transmission and a web-based and mobile remote management system. This solution allows improving maintenance and visualize in real-time the performance for the system on which monitoring devices are installed. The fields of application for the solutions offered by TIERRA are generally construction, agriculture and automotive in particular in this thesis we focused on information coming from vehicles working in construction sites, dumps and for street cleaning.

The work done, for a set of vehicles opportunely chosen, is based on the idea of predicting for the current day in how long the usage threshold(generally in hours) will be reached or if the vehicle servicing will be necessary on the next day or week . The threshold is a value given for each company and can be considered representative for all its models. Once the threshold has been achieved, it is necessary to carry out maintenance(e.g. oil change, filter cleaning, etc.). In short, what we want to do is go from a preventive maintenance(perform the maintenance when the threshold is reached) to a predictive one(forecast when the threshold is exceeded and carry out appropriate actions).

The solution to our problem is provided by machine learning algorithms properly executed on the available data, in particular regression and classification algorithms have been applied. It's important to point out that the dataset provided by TIERRA are a sequence of observations taken sequentially in time, keeping in mind the nature of the problem we are forced to use historical data to predict future ones. Moreover, the data temporal dependence requires different approaches from the classical ones, for instance the forecasted values must be validated without using the classic technique known as cross-validation.

Chapter 3

State of the art

The literature review showed the data have been collected in different industrial fields, and with different kinds of vehicles on which the CAN bus technology has been fitted. The principal field of interest in which the CAN messages are used is the predictive data-mining, this being the data mining that is done for the purpose of using business intelligence or other data to forecast or predict trends. The main tools exploited for this task are machine learning techniques such regression and classification algorithms.
The research on this topic(predictive data-mining with CAN messages) has been conducted trying to analyze different target variables, in particular we focused on a restricted set of scientific papers where the main objective is the prediction of fuel consumption:

- in [1] after a deep analysis to understand which variables are more correlated with the target one, three machine learning techniques, Support Vector Machine (SVM), Random Forest (RF), and Artificial Neural Network(ANN), have been applied to fuel consumption modelling of articulated trucks for a large dataset. SVM and ANN gave the best results.

- [2] aims to explore the trip fuel consumption from large scale dataset for different kinds of industrial vehicles. The algorithm Support Vector Machine obtained the best results, moreover it was employed to understand the correlation between the target variable and further variable(e.g. distance travelled, average speed). The three further regression methods the artificial neural network (ANN), multiple linear regression model and the link fuel summation

SVM model (LSSVM) have produced worse results than SVM. Besides the data collected with the CAN technology, also the GPS was used.

- in [3] an approach totally based on artificial neural networks has been used to predict the fuel consumption of mining dump trucks per cycle of operation. The independent variables employed to train the machine learning models are strictly related to the amount of time spent performing an operation. The outcomes demonstrated a high fuel consumption during the idle times. For this reason it is necessary to implement certain actions to reduce the negative effects generated, of course avoiding idle times.

- in [4] a predictive analysis has been performed on the fuel consumption of a bus, the problem in this case has been addressed as a time series and considering the aspects associated to this type of approach. In this case the main problem was the limited number of available factors to train the model for the prediction, for instance all the parameters associated to the driver would be very useful but should be collected employing additional tools. The regression techniques used are neural networks, gradient boosting and random forest. The last one got the best results

In all the papers mentioned above the machine learning algorithms applied on the data gave results with different levels of accuracy. The final score depends on the kind of relationship(linear or non-linear) between the variable predicted and the independent ones, the kind of data collected for the final experiments(e.g. data strictly related to the vehicles engine, geographical or temporal information, etc.), the type of vehicles(passengers vehicles, trucks or vehicles working in construction sites), and finally the time frame during which data were collected. Therefore, there is not a regression or classification technique working better than others and that can be chosen a priori, and you need to try different algorithms according to the available data.

However fuel consumption prediction has not been the only matter analyzed with predictive data-mining, forecasting the usage of different kinds of vehicles has been a further topic analyzed with different regression techniques [5], also in this type of problem the final results depends on the environmental conditions in which the

data are collected. Furthermore, having available a huge number of vehicles and their related data you need to select in the best possible way the vehicles more representative.

Chapter 4

The data mining pipeline

4.1 Machine learning introduction

As stated by Arthur Samuel, *'an American pioneer in the field of computer gaming and artificial intelligence'* [22], Machine Learning(ML) enables computers to learn and increase their performance from the data received in input. ML is used anywhere and automates tasks performed by humans bringing significant benefits. With the passing of time the huge amount of data and the increase computation power favoured the application of ML algorithms.
ML algorithms can be classified in three main categories [23]:

- **Supervised Learning**: the related techniques receive datasets with a certain number of fields(the datasets columns), some of them are called independent variables(usually indicated by X) whereas there is a single field that is the target variable also known as label(y). The application of supervised algorithms allow to create a mapping function between the independent fields and the target variable both received in input, this phase is usually known as training and a model is generated. The mathematical function/model learned in the previous step can be used to predict the output variable for the new data received. In short

  $$y = f(X)$$

  f should be able to map in the best possible way the input data X. Moreover, it is necessary to distinguish between two type of Supervised Learning:

  - **Regression**: the output variable is numerical and so it can assume any continues values
  - **Classification**: the variable we need to predict is a category and so it is a categorical attribute(e.g. healthy or sick)

- **Unsupervised Learning**: this method differs completely from the previous one for the lack of the target variable, this means that for the input data X the expected result y is not provided. Keeping in mind this relevant aspect, the final task is to identify the similarities and classify different patterns in the samples. Also in this case is possible to identify two different categories for this ML technique:
  - **Clustering**: samples with analogous characteristics are grouped in the same collection. The metrics employed to understand the level of similarity are usually based on mathematical measures.
  - **Association**: the purpose is to find the rules describing big portions of data samples.

- **Reinforcement Learning**: a reinforcement learning algorithm, or agent, receives a reward if an operation was performed correctly or a penalty otherwise for each action performed. The main goal is to maximize the rewards and minimize the punishments.

Before applying any kind of ML algorithm it is necessary to follow a predefined number of steps, which enables to make the most of the ML techniques [24]. Typically, this process is identified by a universal workflow made of the next points. In some cases this workflow undergoes small variations:

- **define appropriately the model**: the target variable we need to predict is identified and also the kind of data that should be selected to perform the analysis. Once that the previous questions have been answered we can choose the approach to get the final results. What kind of ML algorithms is better to

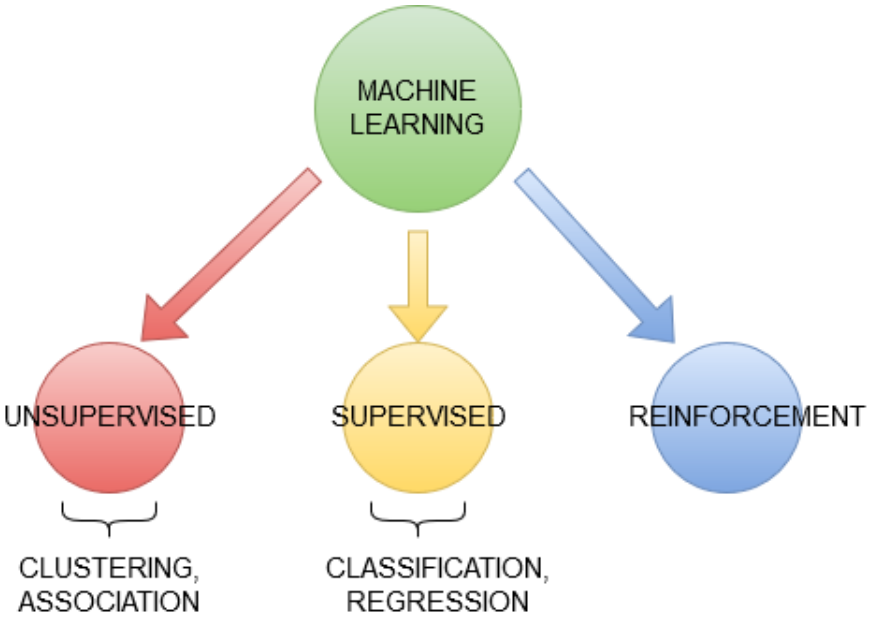


Figure 4.1: Machine Learning overview

use (e.g. supervised or unsupervised learning, etc.) ? How will they be used and what are the meaningful metrics to quantify the achieved results ?

- **collect the data**: this is the most critical aspect because the quality of collected data influence the results obtained with the ML algorithms. The growth of the IoT has made this step easy, allowing us to carry out this task in many different fields and also with an increasing quality

- **data preparation**: in most of the cases the available data are in a raw format that can not be handled by the ML techniques. In particular, we need to fix all the problems associated to missing data, typically represented with the *NaN* or *Null* indicators, categorical data, feature scaling etc. There are not fixed rules to solve these drawbacks, but a specific analysis on each dataset can suggest the most appropriate strategy to adopt. The process identified by the keyword feature engineering is part of this step and allows creating and introduce the independent variables, of course related to the data already available, required in the next stages.

- **data exploration**: this step allows us to understand better what are the main characteristics for the available data. In our analysis this step gives the opportunity to point out what are the most significant vehicles to be

considered(e.g. having a lot of data and a meaningful usage)

- **application and evaluation of ML algorithms**: for supervised and unsupervised ML algorithms a certain number of models is available(random forest, support vector machine, etc.), after choosing one we can begin to apply that on a training set to obtain a model and then we test it collecting and evaluating the achieved results with the metrics chosen. When this process is executed we can decide to use a subset of the starting features

- **parameters tuning**: each ML model mentioned in the previous point is executed setting their parameters, known as hyperparameters, with default values(a second category contains parameters that are learned through the training phase). Tuning the hyperparameters is the best way to obtain the best predictive power possible.

4.2 Regression algorithms

The usage of regression algorithms is necessary to accomplish the main objective of this work considering the type of variable(days/weeks to maintenance) we need to predict. Moreover, a long list of regression algorithms is available in literature but here we decided to work by applying the basic algorithm Linear Regression that doesn't require the tuning of hyperparameters, Support Vector Regression, Random Forest and Gradient Boosting Regression. The last two are based on two different paradigms that are **bagging** and **boosting**, both ensemble methods in ML. What do these terms mean ? As explained in [25], Bagging and Boosting are both ensemble techniques, meaning with this word a Machine Learning technique training multiple models using the same learning algorithm. The ensembles belong to a group of methods, called multiclassifiers, where more than one learner is employed together with others to accomplish a common task. The second group of multiclassifiers contain the hybrid methods that use a set of learners too, but they can be fitted using different learning algorithms. But what is the main difference between these two approaches ? The training phase is parallel for bagging (the quality of each learner doesn't depend on the other ones) and each single result is combined using some model averaging techniques(e.g. weighted average, majority vote or normal

average), boosting builds the new learner in a sequential way taking into account the previous classifiers' ability to predict a value or class label. The new learner depends on the previous one.The first step to use bagging or boosting is the choice of a base learner algorithm that must be applied several times with different subsets of starting data using a random sampling with replacement. In this regard in the case of bagging any element has the same probability to appear in a new data set, however for Boosting the observations are associated to a weight that are used to report the level of importance allowing appearing more often in the new sets. After each training step, the weights are redistributed and data classified with a mistake enhance its weights to point out the toughest cases. In this way, subsequent learners will refine the processing of this data.

All the mathematical aspects related to the ML algorithms exploited in this work are set out below. All the algorithms have been applied using the open source library **Scikit-learn** [20].

4.2.1 Linear Regression

Linear regression is probably one of the simplest algorithm and a good starting point before introducing more elaborated techniques, moreover further regression algorithms exploit the basic characteristics of linear regression and add extra features.
From a mathematical point of view and assuming that our dataset has only one independent variable, the linear regression can be expressed by the following equation:

$$y = \beta_0 + \beta_1 x$$

The beta parameters are the coefficients that we need in order perform forecasts with our model. To find them, we need to minimize the least squares or the sum of squared errors

$$e = \|y_{real} - y_{pred}\|^2$$

The error is squared because the prediction can be either above(expressed by a positive value) or below(expressed by a negative value) the true value. Summing signed

value we can obtain a skewed final result that could decrease because we perform mathematical operation with values having opposite sign and not because the accuracy of our model is improving. This explains the reason why we didn't simply consider the subtraction between the real value and the predicted one. Moreover, squaring the errors penalizes large differences and so minimizing the squared errors allows to generate a model with a better accuracy. In many applications, there is more than one factor that influences the response

$$y = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + ... + \beta_n x_{mn}$$

where n is the number of features and m is the number of samples. Of course, the linear model doesn't generate perfect results and it will not predict all the data accurately, meaning that there is in any case an error between the actual value and the prediction. However, it can be considered an efficient tool to understand if a linear relationship exists between our target variable and the independent ones.
The software implementation is realized with the method *sklearn.linear_model.LinearRegression* available at [9], as already mentioned with the regression algorithms introduction there are not hyperparameters to set up.

4.2.2 Support Vector Regression

Support Vector Regression(SVR) is based on the same basic ideas of Support Vector Machine(for classification tasks), however *'it is considered a non parametric technique because it is based on kernel functions'* [26]. The main objective is to find a function *f(x)* that deviates from the data point y_i by a value no greater than ϵ and as low as possible for each training sample.
Suppose we have a set of training samples, each one identified by x_i, with observed response values y_i. Therefore, we want to find the linear function

$$f(x) = wx + \beta$$

and the solution is given by

$$J(w) = \frac{1}{2}\|w\|^2 \qquad with \qquad \forall i : |y_i - (wx_i + \beta)| \leq \epsilon$$

However, it could happen that *f(x)* satisfying the constraints doesn't exist for all the training samples and for this reason the slack variables ξ_i and $\xi_i{}^*$ are introduced for

each point and the new solution is

$$J(w) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i + {\xi_i}^*)$$

subject to

$$\forall i : \xi_i, {\xi_i}^* \geq 0$$
$$y_i - (wx_i + \beta) \leq \epsilon + \xi_i$$
$$(wx_i + \beta) - y_i \leq \epsilon + {\xi_i}^*$$

The constant C identifies the box constraint, a positive numeric value checking the penalization awarded to data samples that are over the epsilon margin (ϵ). Moreover, it helps to avoid overfitting and therefore final results with low quality. This value is the threshold up to which deviations larger than ϵ are accepted. Any error inside the ϵ distance of the observed value are not considered by the ϵ-insensitive loss function. Measuring the distance between the ϵ margin and the real data samples y we can obtain the loss value. To forecast new values a new function is used that is strictly related to the support vectors:

$$\sum_{i=1}^{n}(\alpha_i + {\alpha_i}^*)\langle x_i, x\rangle + \beta$$

where α_i and ${\alpha^*}_i$ are non negative multipliers for each observation x_i, if either α_i or ${\alpha^*}_i$ is not zero, then the corresponding observation is called a support vector. The constraints for the alpha parameters are

$$\sum_{i=1}^{n}(\alpha_i + {\alpha_i}^*) = 0$$
$$\forall i : \alpha_i, {\alpha_i}^* \geq 0$$
$$\forall i : \alpha_i, {\alpha_i}^* \leq C$$

In some cases a regression problems cannot be characterized using a linear model, in these situations the dot product $\langle x_i, x\rangle$ is replaced by a nonlinear kernel function G(x1,x2) = $\langle \phi(x_i), \phi(x)\rangle$, where $\phi(x)$ is a transformation that maps x to a high-dimensional space.

Scikit provides the method *sklearn.svm.SVR* performing the regression algorithm described above, the relevant hyperparameters are

- the kernel type
- the C parameter
- the range ϵ
- the multiplier γ if we use a polynomial or RBF kernel

4.2.3 Random Forest Regression

As mentioned in 4.2 Random Forest(RF) is an ensemble machine learning technique, it is used for both classification and regression tasks exploiting multiple decision trees on various sub-samples of the starting dataset and an approach known as bagging. Reduction in overfitting, that is the failure to apply a trained model to new data and so its negative ability to generalize, is the strength of RF. This is possible exploiting the general approach provided by the bagging.
The main ideas behind this technique are strictly related to the term *random*, in particular *'random subsets of features for splitting nodes'* and *'random sampling of training observations when building trees'* [27]. Random Forest exploits multiple parallel classifiers and their single predictions are fused together to get accurate results.

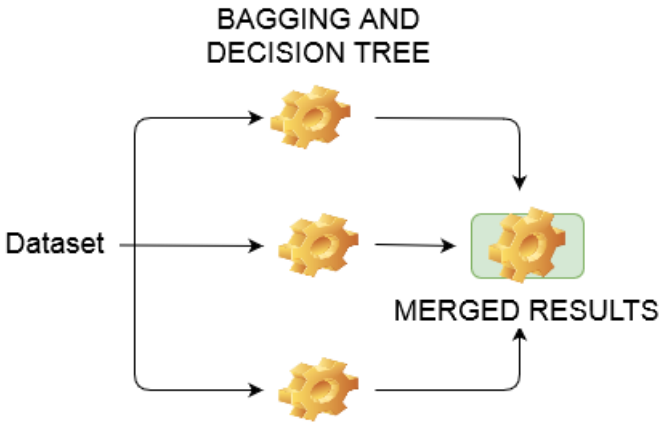


Figure 4.2: Random Forest: the fundamental idea behind a random forest is to use parallel weak learners to get results with high accuracy

As regards the random sampling of training observations, samples are extracted with replacement known as bootstrapping. This means that some data samples will

be employed multiple times in a single tree. The subset size doesn't change, however the data samples are extracted with replacement, this approach is applied with the software implementation of RF setting the parameter identified by the name *bootstrap* with the boolean value True. By default, the bootstrapping is set to True. The Scikit library provides the method *sklearn.ensembleRandomForestRegressor* to accomplish this kind of regression. On the other hand, if the parameter *bootstrap* contains the value False the randomness is not applied and so the same set of data will be used by each tree composing the forest. The second main concept in the RF is that each tree in the forest applies the regression on a restricted subset of all the available features when deciding to split a node. As for the random sampling of training observations, also this aspect is properly handled in Scikit-learn by specifying a value for the parameter *max_features*, for instance if there are 25 features and *max_features = sqrt(n_features)*, at each node in each tree, only 5 random features will be considered for splitting the node.
There are further parameters that must be passed to *RandomForestRegressor* in addition to *bootstrap* and *max_features*:

- n_estimators : the number of trees in the forest, the default value is 10

- max_depth: the maximum depth of each tree. As indicated by the official Scikit documentation if None(default value), then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples

- min_samples_split: the minimum number of samples required to split an internal node(default=2)

- 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. This may have the effect of smoothing the model, especially in regression

all the remaining aspects not set out above can be found at [10].

4.2.4 Gradient Boosting Regression

The main features of Gradient Boosting(GB) were already discussed with 4.2 where the basic principles of boosting have been mentioned. In short, a predefined number of weak learners(weak prediction models) are built and used sequentially. Most of the time these basical learners are decision tree, as for RF. The main purpose of this algorithm is to minimize a loss function $L(y_i,\ y_{i,pred})$, associated to the results $y_{i,pred}$ achieved with our algorithm, by using gradient descent and updating our predictions based on a learning rate. Following these guidelines each predicted value can be updated exploiting the following formula

$$y_{i,pred} = y_{i,pred} - \alpha * 2 * \sum(y_i - y_{i,pred})$$

where α is the learning rate and $\sum(y_i - y_{i,pred})$ the sum of residuals. The main disadvantage of GB is its tendency to overfitting, however this drawback can be tackled by restricting the number of boosting stages to perform.
Scikit provides the method *GradientBoostingRegressor*, belonging to the module *sklearn.ensemble*, that allows to perform this regression algorithm. As for RF a predefined number of hyperparameters must be setup, some of them match with that ones introduced with sklearn.ensemble.RandomForestRegressor(e.g. all the stuff related to a single decision tree that is max_depth, max_features, min_samples_split, min_samples_leaf), a further predefined set of parameteres are strictly related to the boosting operations:

- loss: it is the loss function to be optimized, many choices are available(e.g *least squares*, *least absolute deviation, etc.*)
- learning_rate: (default=0.1) correspond to the α coefficient in the previous equation, it shrinks the contribution of each tree
- n_estimators: (default=100) the number of boosting stages to perform. Gradient boosting is fairly robust to over-fitting so a large number usually results in better performance.
- subsample: (default=1.0) the fraction of samples received in input by each single base learner for their fitting

further information at [11].

4.3 Time Series forecasting

As already said in Chapter 2, a time series is a collection of observations taken sequentially in time. The data temporal order is therefore the characteristic element forcing us to face the problem in a specific way.

To work properly with time series a first preliminary analysis is required, in this way relevant trends can be visualized and it is possible to assume how much they will affect, in broad outline, the quality of predictions.
From a technical point of a view, a time series can be splitted in four components [14]

- level: the baseline value for the series if it were a straight line
- trend: the optional and often linear increasing or decreasing behavior of the series over time
- seasonality: optional repeating behaviors over time
- noise: the optional variability in the observations that cannot be explained by the model

Level is always present in any time series while most have noise, on the other hand trend and seasonality are optional components. In the light of what has been said, any time series can be represented as follows

$$y = level + trend + seasonality + noise$$

A practical application of the concepts mentioned above will be presented in the phase of data exploration on the data used for the forecasts. There is almost an endless supply of time series forecasting problems, for instance forecasting the closing price of a stock each day, unemployment for a state each quarter, the average price of gasoline in a city each day, utilization demand on a server each hour, etc.

4.4 Validation of the predictions

As pointed out with the universal workflow followed to build a ML model, in particular with reference to the second-last point, once a model was trained we need to

test its accuracy on data that it has not seen before. This process is called **validation** and helps us to understand if a model is underfitting(bad results for training and test data), overfitting(bad outcomes only for data never seen before) or well generalized. The most used approach is **Cross-validation**(CV) with its related techniques [19]:

- **train-test split** approach: all the available data are splitted into training and test set, then the model training is performed on the training set and use the test set for validation purpose, the data are usually splitted into 80:20 or 70:30. If the total amount of available data is not huge the trained model could miss some information about the data not used for training, otherwise this approach is acceptable

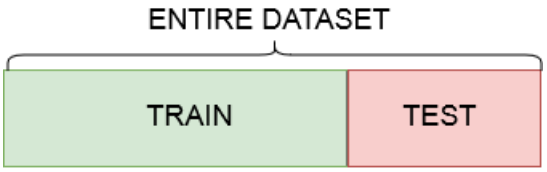


Figure 4.3: Train-test split validation

- **k-folds cross validation**: this technique is very useful in case we have limited input data because it ensures that every sample from the original dataset has the opportunity of appearing in training and test set. By getting more into the details, the entire data is splitted randomly into k folds (k is usually greater than 5 and less than 10, however it depends on the data size). A model is created using the *k-1* folds and validated with the remaining kth fold, the associated score is saved. This process is repeated until every K-fold serves as the test set. Finally, the average of scores recorded is computed and represents the final metric to quantify the error

Cross-validation for time series is different from machine-learning problems, that's because in the case of the absence of time we select a random subset of data as a validation set whereas shuffling the samples belonging to a time series we don't consider the temporal dependence. Two schemas are employed to perform validation for time series:

- **sliding window**: as shown in the second plots of 4.4, the model is trained on n samples and tested on the next m data points, then the training set is updated shifting the first and the last endpoint of the window. The same operation is executed for the test set.

- **expanding window** also known as Forward Chaining validation: the main difference with the previous approach is that here the training set is updated shifting only the last endpoint of the window, whereas both endpoints of the test set are moved

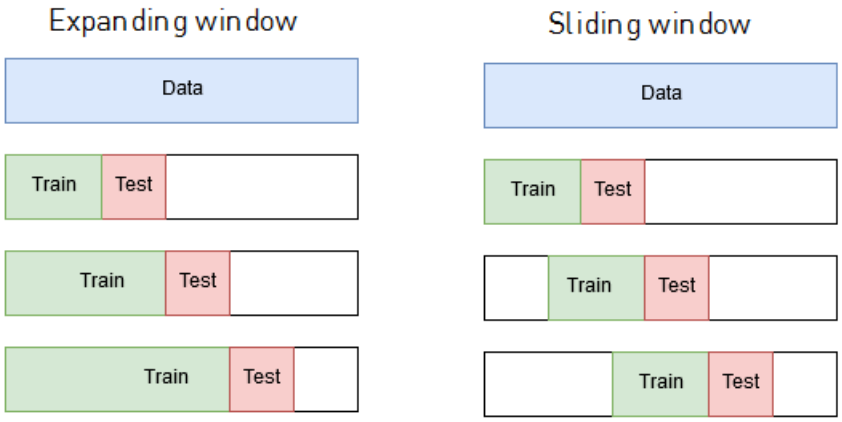


Figure 4.4: Expanding and sliding window for validation

Chapter 5

Data preprocessing

5.1 Vehicles selection

All the vehicles belong to four different companies that are identified by C1, C2, C3, C4. Each company produces different types of machines, these are then divided in different models. This means that vehicles produced by the same tenant and of the same type can belong to different models. Each type is identified by the letter T and a number(e.g. type 1) whereas each model has an identification number that we translate in *'model and identification letter'* (e.g. model A), the id code for a single machine is called *unit_uuid*. It's important to point out that a type is produced by a single tenant, for instance the vehicles for street cleaning are manufactured only by C2 whereas the remaining companies are specialized in the production of different kinds of vehicles(e.g. vehicles for work on construction site or in waste disposal plants, etc.). In the light of this, the model C1-T1-modelA represents a model different by C2-T1-modelA (despite the same type and model may suggest otherwise) because C1 and C2 produce different kinds of vehicles. The same reasoning applies to other models, types and companies. The hierarchy explained above for the vehicles analyzed is represented in 5.1 for a generic tenant.

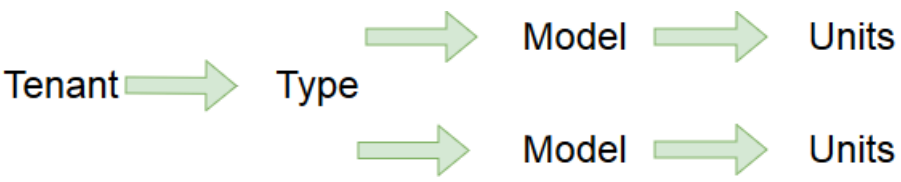


Figure 5.1: Vehicles hierarchy

Before starting the real data exploration we needed to select for each tenant a subset of types, with the associated models and vehicles, for the subsequent analysis. We have selected the companies C1, C2 and C3(C4 has been discarded for the lack of data). For each tenant two types have been chosen and for each one three models, those with the largest number of units and average number of rows. The selection phase was carried out using a table containing the CAN messages. The images below summarizes the choices made.

tenant	unit_type_name	unit_model_id	distinct_uuid	average rows
C2	T1	A	21	302741
C2	T1	B	4	312691
C2	T1	C	1	213678
C2	T2	A	23	155083
C2	T2	B	30	71648
C2	T2	C	70	54787

Figure 5.2: C2 models selected for the further analysis

It's important to have a high number of distinct units for each model because in this way we can retrieve info representing the behavior of an entire set of vehicles. Fortunately the cases of models with a number of units less ten are uncommon.

tenant	unit_type_name	unit_model_id	distinct_uuid	average_rows
C3	T1	A	19	101516
C3	T1	B	4	379454
C3	T1	C	7	32515
C3	T2	A	24	434071
C3	T2	B	32	490214
C3	T2	C	25	297239
C1	T1	A	13	138477
C1	T1	B	29	127390
C1	T1	C	16	196980
C1	T2	A	29	59906
C1	T2	B	18	74804
C1	T2	C	32	80591

Figure 5.3: C1, C3 models selected for the further analysis

5.2 Data preparation

A further step, that we called data preparation, is necessary before performing the data exploration. The main objective of this phase is to transform the starting raw data in a convenient format.
For each model four tables, identified with the prefix *usage*, are available:

- usage_fuelburnt: for each couple day-unit_uuid the amount of fuel used is showed
- usage_distance: each record reports the information related to the geographical aspects such as the country, distance traveled during the day, first and last coordinates recorded, first latitude and longitude values for the previous day and finally the difference between the first coordinates in the previous day and the current day(to understand if the vehicles has been moved in a different working site)
- usage_ontime: a single record contains the information about the number of seconds a vehicle has worked during the day
- usage_duty: the number of seconds the vehicle has worked is splitted into the status 'ON/Idle', 'Moving /Working', 'High workload' and 'Long Idle'. Each column of this table contains the number of seconds passed in each of these states

Each table contains data related to multiple units with a daily granularity, when a unit didn't work on a specific day the corresponding record misses. As already said in 2.3, a further table called 'maintenance_profile' was used to retrieve the info about the usage threshold for each tenant, considering this value applicable for all their models.

5.2.1 Data aggregation

The first step followed in data preparation is the aggregation of information contained in the four starting tables, mentioned above, in a final one. In fact, working

with different dataframes would be uncomfortable and there would be much more chances of making mistakes in the next operations(data exploration and prediction). The final dataframe can be easily accomplished with the join operation with respect to the common fields 'unit_uuid' and 'date'(the 'merge' method provided by the Pandas library performs this kind of task). From a technical point of view, all the records belonging to the tables we want to aggregate and having the same values for a specific set of fields are merged in a single line.

The image 5.4 represents schematically the rows aligning belonging to two different tables, in our case we have performed this operation on four tables but the principle of implementation is the same.

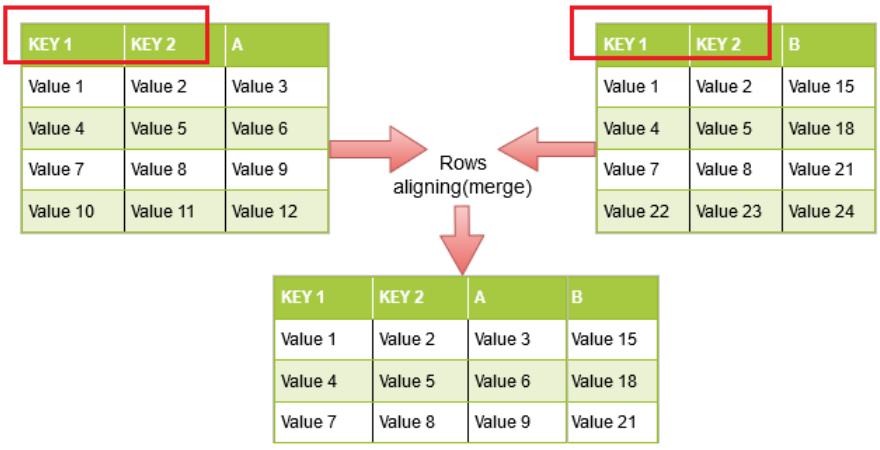


Figure 5.4: Merge operation: both the tables have two fields in common used for the rows aligning

5.2.2 Missing values

All the raw data contained in the starting tables don't suffer from a huge lack of valid values(e.g. equal to NaN). Only the features *lat_d_1*, *lon_d_1* and *dist_d_1*, belonging to the table *usage_distance*, are affected by this problem. These fields contain for the current row the first geographical info recorded in the previous day, and so to the preceding record. That's why the values of these fields miss in the first row recorded in temporal order for each single vehicle(e.g. all the info associated to the day before the first one in the tables, are not available). For instance, if the first record for unit_uuid abc1234... is related to 2015-01-01 the geographical

coordinates on date 2014-12-31 are unknown.
For any unit_uuid the percentage of records with missing values is equal to one over the total number of records, that justifies the choice of removing for each machine analyzed the related record containing values equal to NaN.

5.2.3 Features engineering

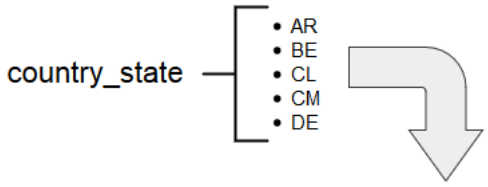
Once all the starting data are joined, the subsequent task is the application of the process called *feature engineering*, that is the creation of new features used in the model training. This operation is extremely sensitive and fundamental for the application of machine learning algorithms. Considering our problem(time series forecasting) we are forced to face the problem in a special way.
Feature engineering is basically performed distinguishing between feature conversion and feature creation. Feature conversion aims essentially to convert categorical features in numerical ones, this operation is strictly necessary if we want to give in input to any machine learning algorithm the values of fields that could improve the results. On the other hand, with feature creation new contextual features are created. Those new fields are created using those already available(e.g. 'season' based on the values of feature 'date') or retrieving information related to the dataset we want to enrich. The subsequent list presents all the new features:

- day_of_week(an integer from 1 to 7 associated to the name of the days)
- month(integer from 1 to 12)
- year(integer from 2015 to 2018)
- week of year(integer from 1 to 48)
- season(integer value from 1,for winter, to 4,for autumn, to identify the season)
- workingday(binary value, if the current day is not a holiday its value is equal to 1 or 0 otherwise)
- holiday(binary value, equal to 1 if the current day is a holiday or 0 otherwise)
- ontime_second(timestamp related to the time the vehicles has been switched on the first time in the current day)

- offtime_second(timestamp related to the time the vehicles has been switched off the last time in the current day)
- countryState_code, created using the mechanism known as 'one hot encoding'
- hours_to_maintenance(for the current record contains the number of hours to maintenance)
- days_to_maintenance(target variable for regression)
- weeks_to_maintenance(target variable for regression)
- maintenance_next_week(target variable for classification)
- maintenance_tomorrow(target variable for classification)

When a categorical feature is converted in a numerical one you need to pay attention to the implementing rules. In fact, if a natural ordinal relationship exists between the values of a feature the association categorical-numerical value can be accomplished using increasing numerical values. For instance looking at the previous list of new features, the values for the column day_of_week are no longer strings(e.g Monday) in the new format but in the new dataframe a list of numbers, from one to seven, is used to point out the ordinal relationship for the values of this column. There may be problems when there is no ordinal relationship and allowing the representation to lean on any such relationship might be damaging to learning to solve the problem. The most popular solution in this situation is the usage of *one hot encoding*. This approach considers the set of categorical values for each column of a table and introduces for each one a new field in the table that is a binary feature, zero and one are the typical values used. The advantage of using this technique is that it makes available the use of features that otherwise could not be used by the various machine learning algorithms, moreover the use of limited numerical values(zero and one) does not introduce any relation of order that could alter the analysis results. In our case this approach was used for the features *country_state*, *holiday* and *workingday*. The figure 5.5 represents the new columns introduced in a dataframe with one hot encoding on country_state field.

As well as mentioned for the join operation, even for one hot encoding the tools provided by Python and the Scikit-learn library was used to perform this task, otherwise would be much expensive performing this operation by hand.



offtime_second	countryState_AR	countryState_BE	countryState_CL	countryState_CM	countryState_DE	countryState_FRA
36748.0	0	0	0	0	0	0
75665.0	0	0	0	0	0	0
85952.0	0	0	0	0	0	0
65229.0	0	0	0	0	0	0
86086.0	0	0	0	0	0	0

Figure 5.5: One hot encoding representation

The temporal libraries *datetime*, *workalendar* [6] and *holidays* [7] provided by Python was used to fill up the features holiday and workingday. These last two libraries contain info about a restricted set of countries, the missing information was retrieved at the following link [8].

5.2.4 Setup of lag features

As mentioned in 2.3, the problem we want to solve is a **time series forecasting** and this aspect force us to face the problem unlike the classical regression tasks. In particular, we have to fit models on historical data and then use them to predict future observations(the problem will be formulated formally in the next chapter). Keeping in mind the data temporal dependence we update the schema of the dataframe introducing the *lag features*, also called *lag times*. The number of lags introduced is a parameter that should be analyzed performing a set of experiments. The drawback associated to the use of a certain number of past values for the fitting and prediction phase is the disposal of records with index from 0 to n-1 belonging to the dataframe(denoting with n the number of lags) and containing NaN values. Removing the records is the unique way to deal with the problem because applying any other kind of approach aimed to replace the NaN would be damaging, that's because the future predictions would be based on an inconsistent set of past values that don't exist. For instance the first record(index equal to 0) in a dataframe have

no lag features because there aren't further rows before, the record with index n-1 misses only the nth lag. In conclusion, the more lags we consider the more records are removed from the dataframe.

Figure 5.6 shows how the dataframe schema and its content change with the introduction of lag features.

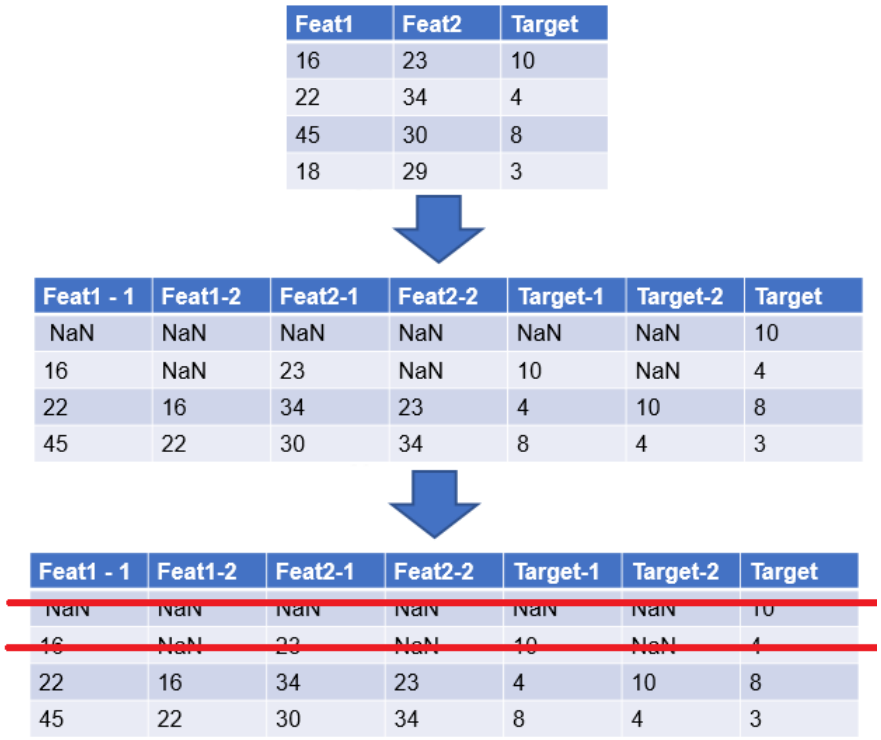


Figure 5.6: Lag features representation

Chapter 6

Data exploration

6.1 Motivations

The main objective in this thesis is the analysis of an approach based on predictive maintenance with predictive data-mining techniques applied on a set of work vehicles. The predictive maintenance aims to predict when a usage threshold will be reached, and so basically we are interested to analyze the usage of the vehicles and all its associated aspects. Understanding how the machines are used(e.g. number of days with usage equal to zero, typical daily range of usage) gives an overview to evaluate the maintenance aspects we are interested in(e.g. which vehicles require a more or less frequent maintenance, how well the machine learning techniques can work with the available data). The entire analysis presented later is intended to analyze the utilization hours for the single vehicles or for entire models, that's because the regression and classification algorithms were fitted using for the first methodology an independent variable called *hours_to_maintenance*, dependent on the utilization hours, and introduced with 5.2.3 whereas a second methodology seeks to predict the utilization hours directly. In particular the analysis aims to identify patterns of vehicles usage, evaluate utilization hours and finally make a comparison between different models, types and tenants.
The results will be showed using the following methodologies:

- time series
- boxplots

- cumulative distribution function(CDF)
- histograms

6.2 Analysis of utilization hours with time series

The first approach followed is the time series representation, used for all the models selected and belonging to the three tenants C1, C2 and C3. The dataframe on which the operations have been performed is that one obtained at the end of preprocessing phase. At first the data was aggregated with a weekly and a monthly temporal scale, in this way it was possible to analyze the behavior of different models. Then, using a daily representation some units have been analyzed in depth. Considering the table from which we retrieve the data, the records belonging to the same week of the year/month and model was aggregated then for each group the values of the onofftime_second field were summed up. Finally, the sum was divided by the number of distinct units in the considered period(week or month). All the following graphs represent on the x axis the time(month or week), on the y axis the utilization hours. Each graph(one for each company) contains three different lines(one for each model belonging the company analyzed) depicted using different colors. Most of the models are used in a period of time from 2015 to 2018, few of them have records until 2019.

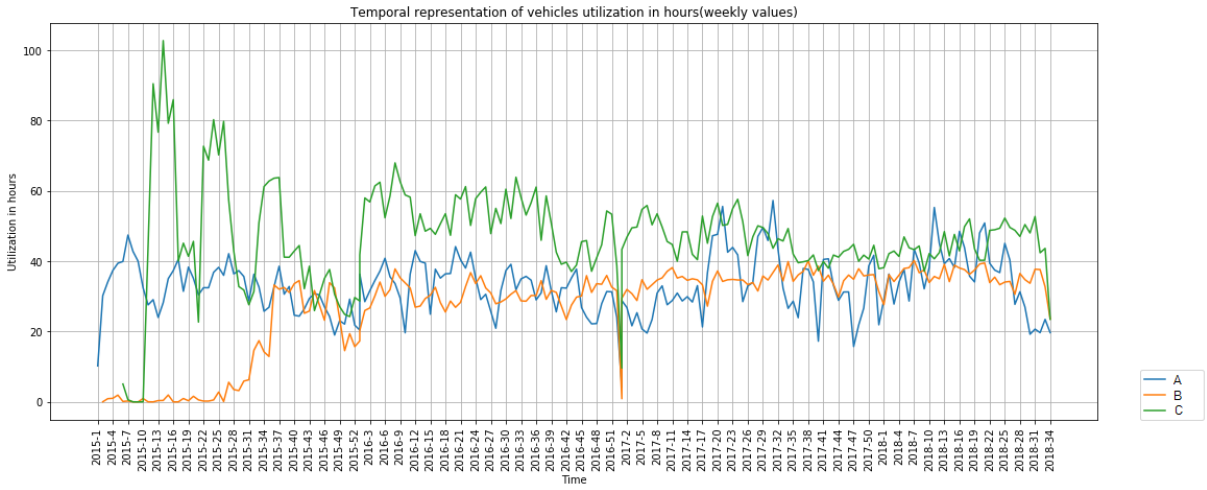


Figure 6.1: Weekly time series for C1-T1

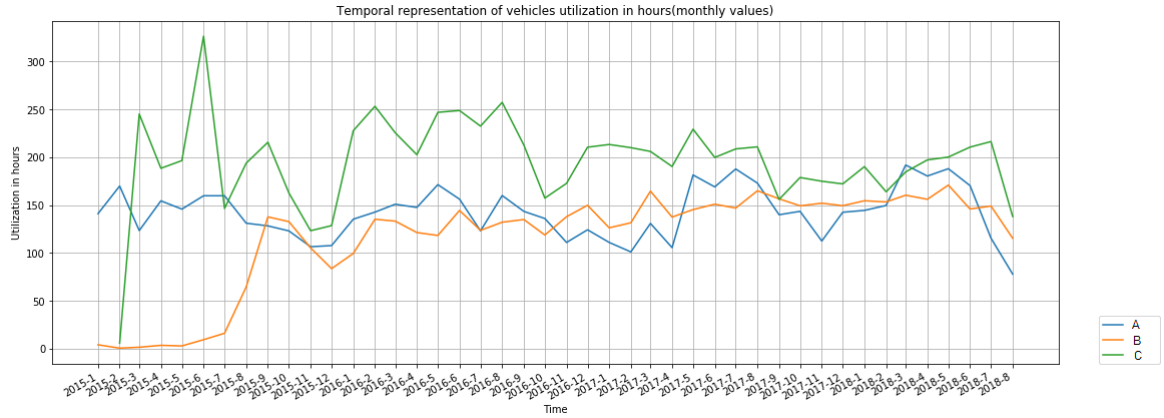


Figure 6.2: Monthly time series for C1-T1

As planned, the weekly representation is harder to analyze than the monthly one due to its less compact values and the associated jagged trend. This is clear for example looking at 6.1 and 6.2. That's why the monthly representations are more used to retrieve meaning details about the usage of vehicles.

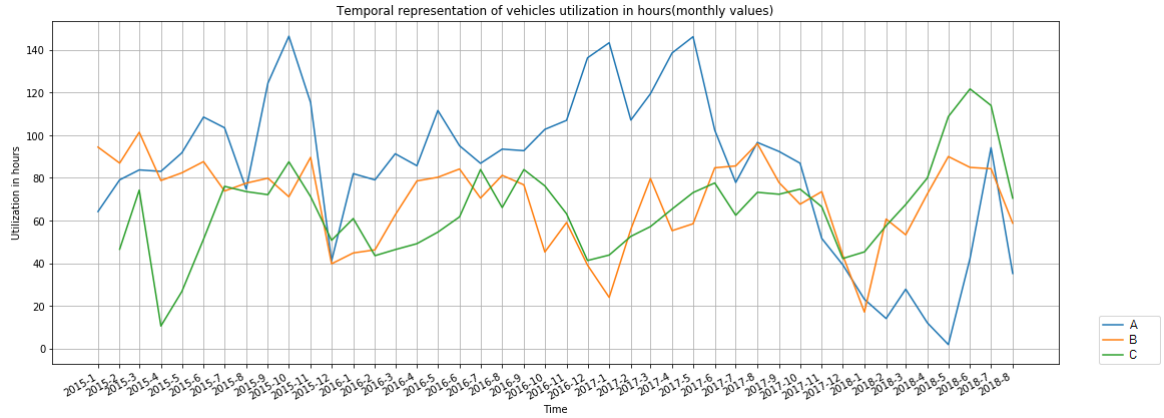


Figure 6.3: Monthly time series for C1-T2

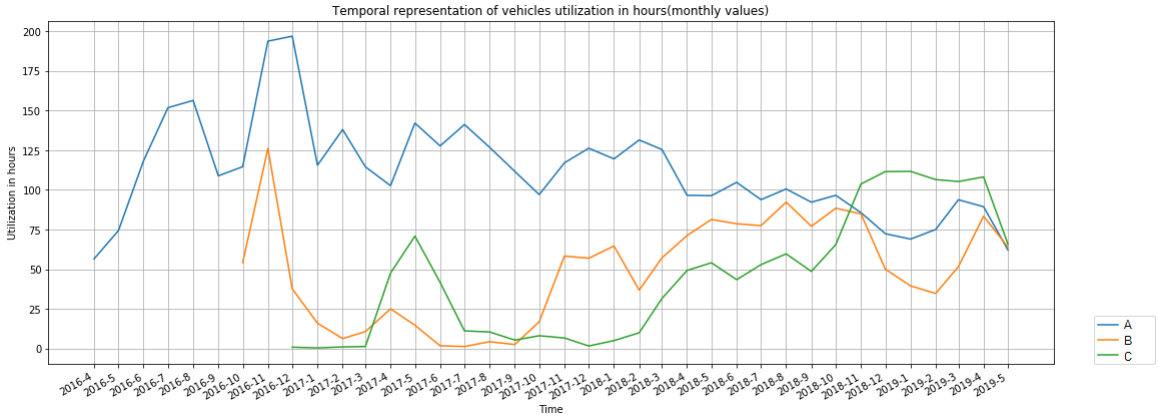


Figure 6.4: Monthly time series for C2-T2

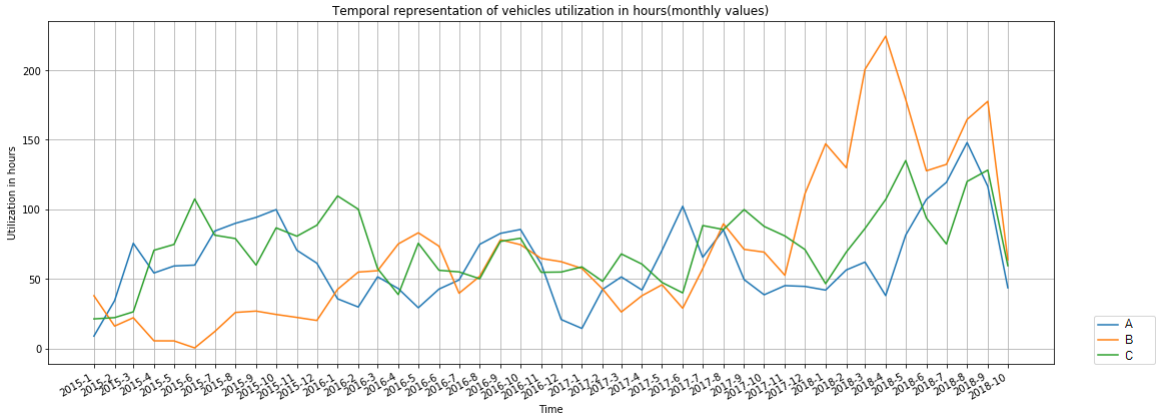


Figure 6.5: Monthly time series for C3-T2

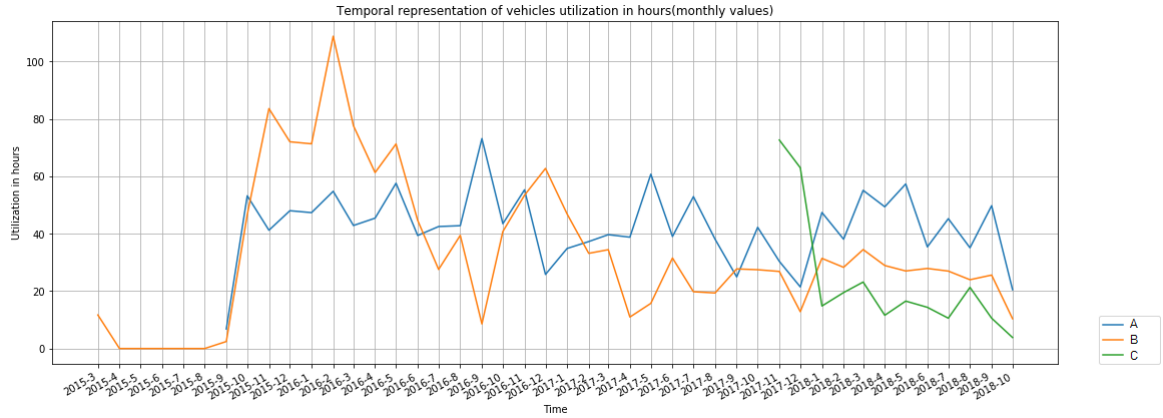


Figure 6.6: Monthly time series for C3-T1

Analyzing the plots it is possible to identify different trends

- stationary trend: for the first behavior identified the following table lists the models with this kind of trend

tenant	unit_type_name	model_id
C1	T1	A
C1	T1	B
C2	T1	A
C2	T2	A
C3	T1	C

Figure 6.7: Models having stationary trends

- non-stationary trends, with specific uptrends in the first months of use e.g. C2-T2-C, or uptrends in the last available ones e.g. C3-T2-B

- non-stationary trends, with peak of usage: e.g. C1-T2-C with decreasing peaks in December or C2-T2-B with decreasing peaks in February

Peaks and stationary trends for the models mentioned above can be justified considering for instance the necessary work done day by day(e.g. vehicles used in waste dumps cannot stop working), unfavorable weather conditions for a time period or finally maintenance periods scheduled. Of course these considerations are insights that should be detailed analyzing the environment(work site, weather conditions, e.g) in which the vehicles work. Summarizing the results achieved with the time

series representation the vehicles obviously have an extremely heterogeneous usage, moreover the predominant non-stationary trends suggest that it is hard to generalize models to types basically because the vehicles usage depends on the context. Finally, aggregating by tenant is not appropriate.

6.2.1 Focus on daily time series

By focusing attention on the daily time series it is possible to identify within a model different behaviors for all their units(each one identified by a unique code called unit_uuid), for instance:

- C3-T2-A:
  - usage with relevant peaks, null and repeated values in random time period
  - winter months with usage close to 0, spring and summer months with average usage higher than 6 hours
- C2-T1-B
  - occasional values close to 0 and remaining values in the range 3-10 hours repeated cyclically over time
  - patterns repeated over time. Usage in the range 0-6 hours in the winter months, 4-10 hours in the months from June to September
- C1-T1-B
  - cyclic usage with repeated values in a well defined range and a restricted sequence of values close to 0
  - starting months(4 or 7) with usage from 0 to 2 hours and higher in the next months
- C1-T2-C
  - usage equal to 0 hours in December
  - utilization hours equal to 0 in random months

The following images highlight the relevant behaviors for the units mentioned above and belonging to C3.

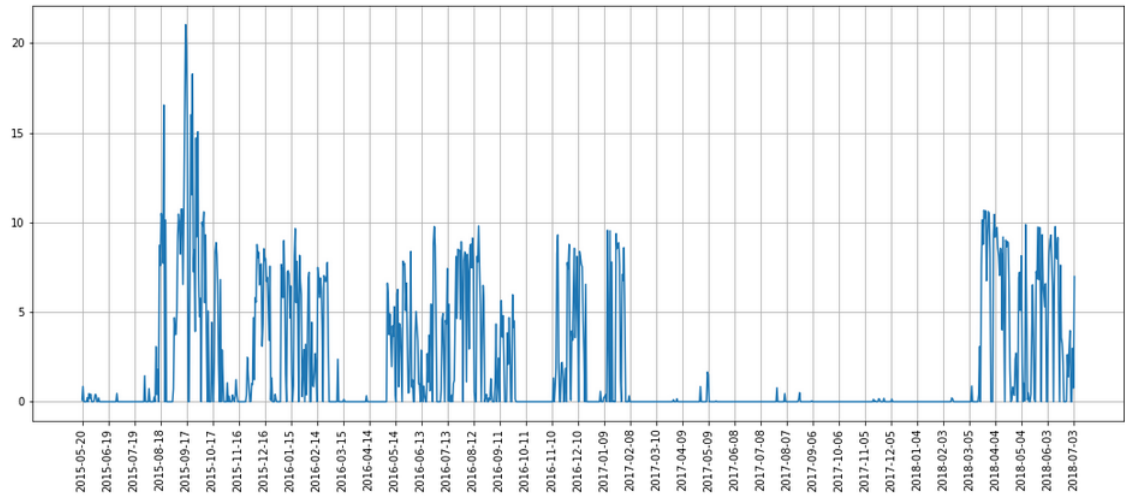


Figure 6.8: Daily time series without defined patterns

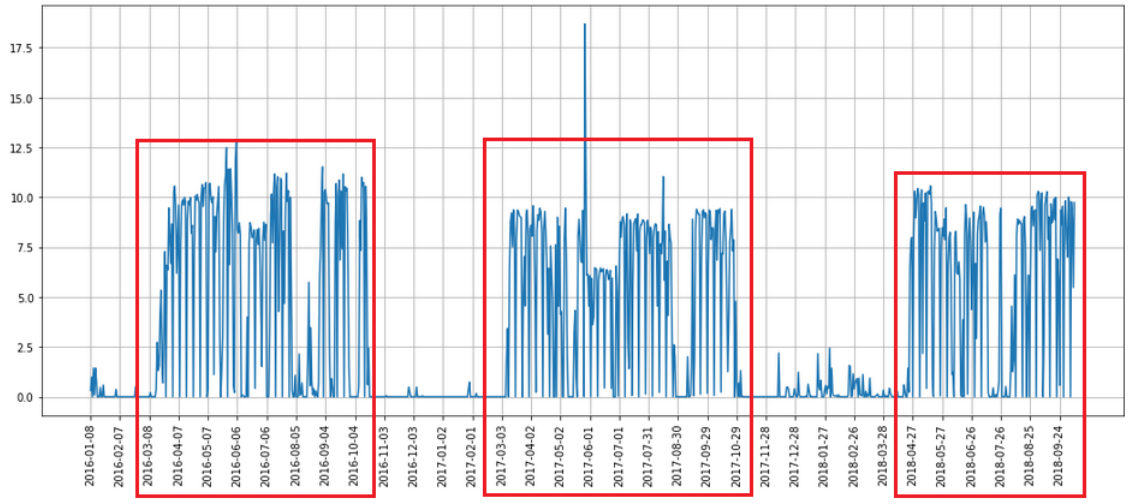


Figure 6.9: Daily time series with repeated patterns

The only exception is the set of vehicles belonging to the model C3-T1-B where there are not different kind of use.

All the possible reasons presented to explain the trends identified with time series analysis per model can be used also to understand the different behaviors for single different units. Furthermore, it is not possible to aggregate units by model. The most important consequence of this heterogeneous usage is the need for predictive models which are customized for each single vehicle.

6.3 Boxplots representation

Figuring out how the daily values are distributed is much easier exploiting the boxplots, especially as regards the daily values. As explained in [29] *'a boxplot is a standardized way of displaying the distribution of data based on a five number summary (minimum, first quartile (Q1), median, third quartile (Q3), and maximum)'*. Moreover, it allows representing data sample considered outliers as it is outside the range of data pointed out. In a few words this tool allows us to visualize in a series of data which values are more frequent (showing the interval that contains them, typically a colored rectangle) and which less. Even in this case a representation for each tenant, and for their models chosen, is available distinguishing between daily, weekly and monthly values.

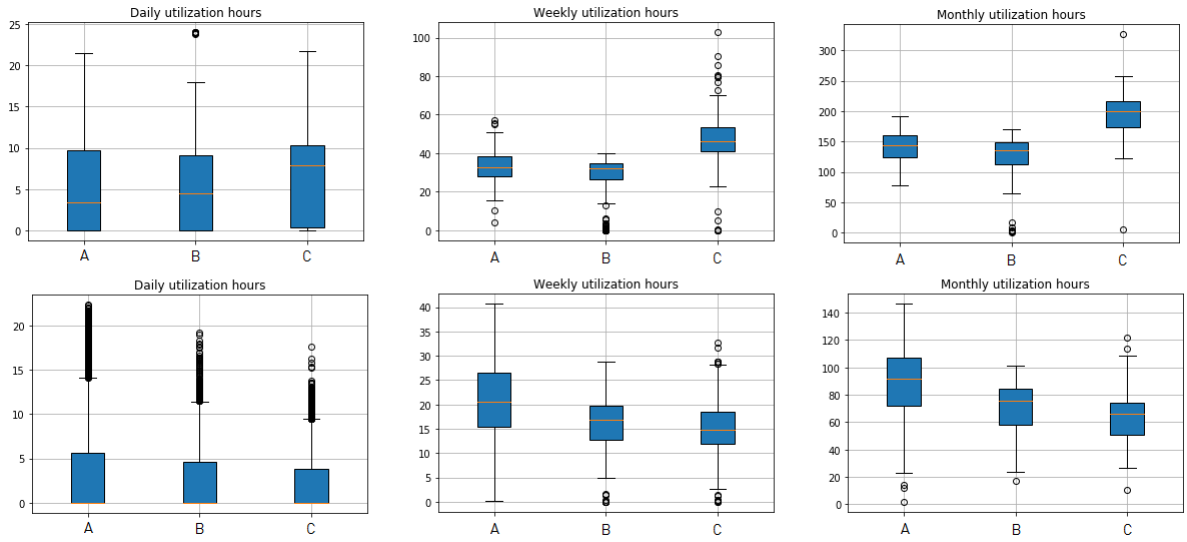


Figure 6.10: Boxplots for C1-T1 C1-T2

Looking at the daily representations it is noticeable a lower usage for the units belonging to C3-T1-C having an interquartile range(IQR) that can be considered negligible and located close to 0, this means that the vehicles belonging to this model have a very frequent low daily utilization hours. On the other hand vehicles of type C1-T1 and C2-T1 have the highest percentage of daily utilization hours over the threshold of five hours, the same predominance is present in the monthly and weekly values. Analyzing the models of a same company there are not relevant differences, in fact most of the time the lower whisker size is negligible whereas boxplot for the vehicles C2-T1-B is the unique case having a boxplot with a normal

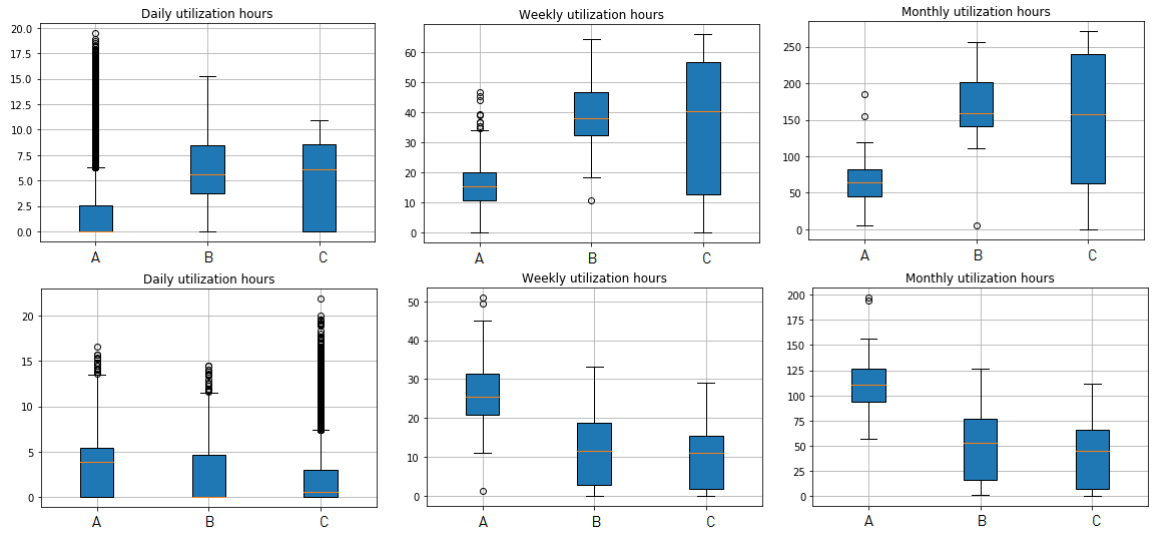


Figure 6.11: Boxplots for C2-T1 and C2-T2

distribution. In addition to this many other aspects are shared between models of different companies, for instance the next models show a similar interquartile range with 25° percentile close to 0 and 75° percentile close to 5 hours, this means that the 75% of possible values for utilization hours are shared:

- C1-T2 with models A, B and C
- C2-T2 with models A, B,and C

Moreover, the median is in most of the cases very close to 0. Only the models of type C1-T1 and C2-T1(in particular models B and C) have the higher values for the median, in the range 4-6 hours. Finally, the presence of a high outliers density for the types C2-T2 C3-T1 and C3-T2 show that despite the low utilization hours, the vehicles of this type are sometimes used with a higher number of hours. The results with weekly and monthly granularity can be summarized as follows:

- weekly boxplots: the median is predominantly located in the range 10-25 hours, the two extreme cases are given by the ranges 0-10 hours(related to C3-T1) and 30-45 hours(for some models of C1-T1)
- monthly boxplots: the median is predominantly located in the range 50-115 hours, the two extreme cases are given by the ranges 20-40 hours(related to C3-T1) and 130-200 hours(for some models of C1-T1)

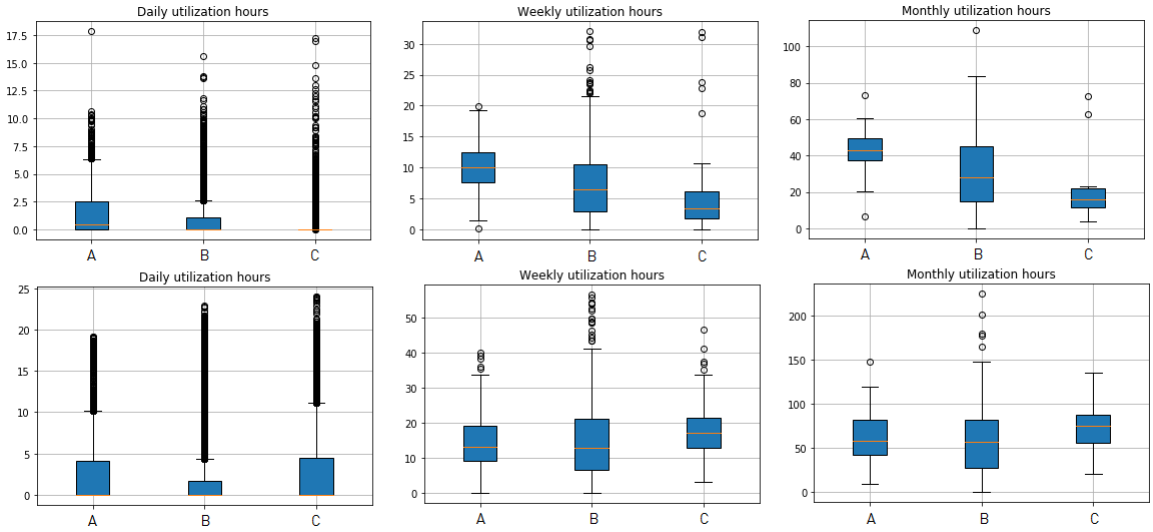


Figure 6.12: Boxplots for C3-T1 and C3-T2

The differences but also the common ground between models of same and different tenants, pointed out with the time series representation, are showed further with this approach. As for the predictive data-mining considerations associated to the results collected with boxplots, the daily values distribution can suggest if the regression machine learning techniques will be or not put a strain, being associated to a large or short range of utilization hours values which influence the timetable related to maintenance tasks. That said, for vehicles belonging to C3-T1 and C3-T2 the short daily boxplots could suggest more favorable conditions, whereas the same cannot be said for models belonging to the companies C1 and C2.

6.4 Analysis of the Cumulative Density Functions

The cumulative distribution function(CDF) enables us to analyze the data from a probabilistic point of view. In fact, the CDF of a real-valued random variable X is the function given by:

$$F(x) = P(X \leq x)$$

in this way we can compute the probability that the target variable X(the utilization hours in our case) is less or equal to x(all the possible values for the variable we are interested in). The main goal associated to the use of this mathematical tool is to understand which models have a high number of records with utilization equal

to zero, that's because it is the aspect that impacts more on the application of regression algorithms, and so the quality of results. On the other hand, a low use of a vehicle means a less frequent number of maintenance tasks and therefore a not interesting analysis.

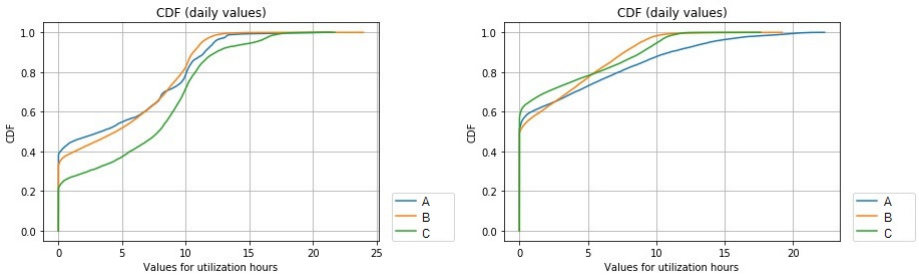


Figure 6.13: Daily CDFs for C1-T1 and C1-T2

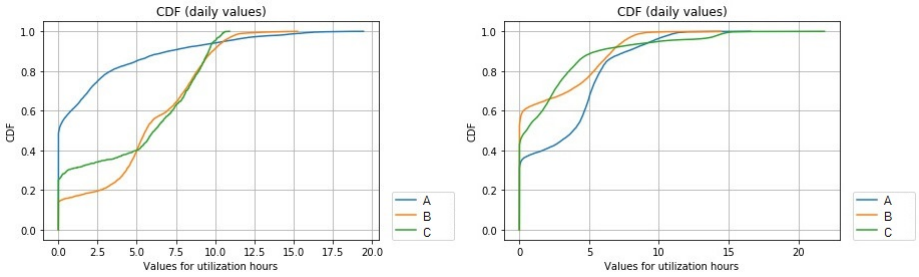


Figure 6.14: Daily CDFs for C2-T1 and C2-T2

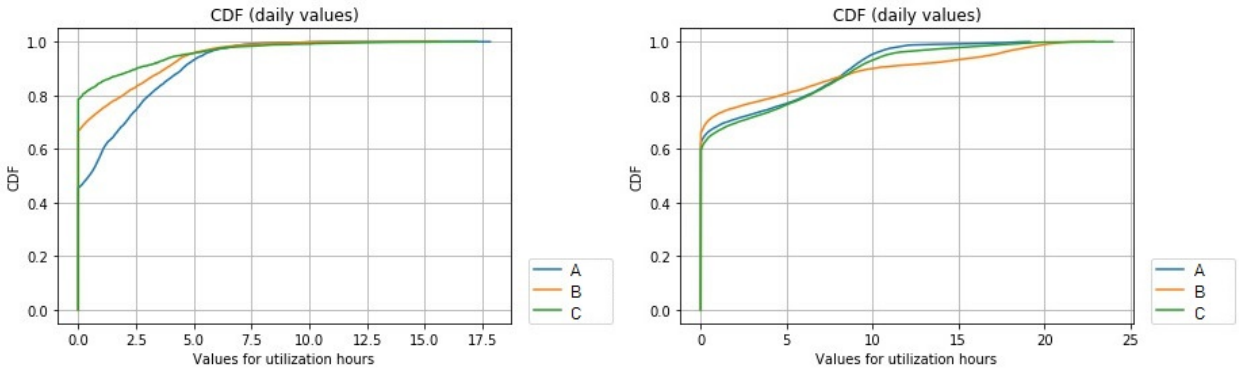


Figure 6.15: Daily CDFs for C3-T1 and C3-T2

The models of C3-T1 have the higher average percentage(≈70%) of records with daily utilization hours equal to zero. This result can be easily related to the daily boxplots representation where the IQR for all the models is located inside the range 0-2 hours or even totally overlapped and coincident with the median very close to 0 hours, whereas higher values are considered outliers and so occasional values. The most different behaviors between models of a same type occur for models of C2-T1, but in general only the models of C3-T2 and C1-T2 have similar CDFs.

6.5 Final considerations

In view of the approaches used to perform the data exploration it is possible to summarize schematically the results achieved

- units belonging to the same model have heterogenous usage, this means that a certain number of well-defined patterns related to the daily utilization hours are shared between vehicles of a same model. Because of this, the maintenance tasks are performed with a different time progression and a predictive data-mining model should be fitted for each different pattern identified.

- the analysis of data distribution pointed out how for some models the range of values associated to the utilization hours is very restricted and could guarantee favorable conditions for the phase of prediction, about this the types C3-T1 and C3-T2 have this feature. The daily representation plays a major role,

weekly and monthly plots shift the daily results on a bigger temporal scale without adding relevant info

- applying the regression algorithms on all the models selected would require a massive effort, for this reason it's necessary to perform a further selection choosing a model with certain characteristics. The idea is to choose for a single company a set of vehicles belonging to a model having common features with the two remaining(we can discard the models belonging to C2-T1 and C2-T2). Furthermore, exploiting the info retrieved with the data exploration we can select units belonging to models with a well defined range of utilization hours(so we can't consider C1-T1) and a usage neither high nor low(C3-T1 models are not widely used). Given the choice between C1-T2 and C3-T2 the choice fell on this last.

Chapter 7

Methodologies

In this chapter the problem is formulated from a mathematical point of view, moreover the regression and classification algorithms and the two methodologies(with the advantages and disadvantages associated) are presented.

7.1 Problem formulation

As already said in 2.3, the main goal of this thesis is the study of predictive datamining algorithms in a context related to predictive maintenance. In particular, we want to predict how many days/weeks are left until the next maintenance operation. Considering the nature of the problem, we must forecast continuous values and regression algorithms can accomplish this task. *days_to_maintenance* is not the only target variable. The strategy adopted in this work is to forecast the days remaining to the maintenance and also the number of weeks but with a larger time horizon. This means that if the daily predictions are performed focusing on the thirty days before maintenance, the weekly forecasts are run and their errors represents for the 10 weeks(70 days) before the vehicles servicing. This approach allows understanding in principle when the usage threshold will be exceeded many days, more than one month, before the servicing. On the other hand the daily predictions focus on a more restricted time interval and should achieve a better accuracy(error expressed in terms of days and not weeks) in the forecasts. Moreover, classification algorithms are useful to forecast(exploiting binary classification) if the next week the usage threshold is exceeded or not. From a mathematical perspective the problem can be

represented as follows

$$y = f(feat_{t-1}, feat_{t-2}, ..., feat_{t-n}, \qquad add\_feat_{m,t-1}, ..., add\_feat_{m,t-n})$$

The meaning of the terms associated to the formula:

- y represents the target variable: *days_to_maintenance*, *weeks_to_maintenance*, *maintenance_next_week*, *maintenance_next_day*
- f is the function receiving in input a set of values and giving in output the value for the variable we are interested in
- *feat* is the main feature used to train a model. Of course, we are working with a time series and for this reason we need to refer to historical values, properly indicated by the subscripts *t-i*(i = 1, ..., n), to forecast the future ones
- *add_feat* is the generic additional feature that can be exploited to improve the quality of the prediction. Considering that there are multiple additional features, the subscript m identifies one.

In the next sections the distinction between the main and additional features will be clarified.
The following results are related to the daily experiments and focus on the different number of lags used to train the models, subsequently the weekly results will be presented having chosen a predefined number of features for the training phase.

7.2 Methodologies for predictions

The regression algorithms are used with two different methodologies, meaning the steps followed to obtain the final forecasts. Two approaches were used that we identify with the name *"Predict the remaining days to maintenance"* and *"Cumulate daily utilization time predictions"*.

7.2.1 Methodology 1: Predict the remaining days to maintenance

With the first approach the target variable is *days_to_maintenance*, and we forecast it straight away after fitting a model on a training set. The approach de-

scribed in the next rows is used also for the target variable *weeks_to_maintenance* *hours_to_maintenance* is the main independent variable exploited to train the model(the variable we called *feat* in 7.1) and introduced with features engineering during the data preparation, of course we use values referred to previous time frames(as indicated by the subscripts in 7.1). Moreover, considering the context(time series) we are working in certain rules must be followed.

Before starting to explain in details how this methodology works, it is necessary to refer to the picture 7.1, where the horizontal axis represents the time whereas the vertical lines are drawn in correspondence to the dates in which maintenance tasks have been performed.

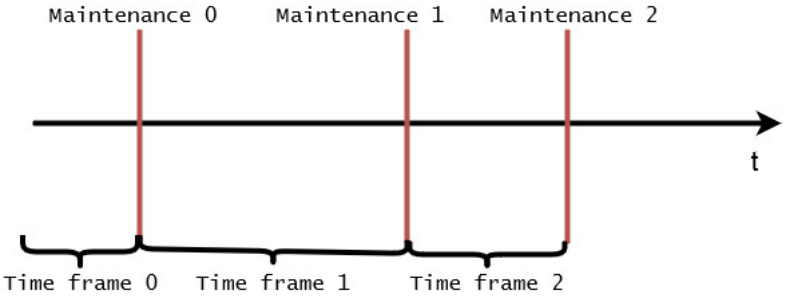


Figure 7.1: Base temporal schema

The vertical lines split the horizontal one in sectors or time periods, each one(except the first and the last) is bordered by two lines. The second vertical line represents, for the current time period, its real maintenance date. For each date belonging to each sector we predict the days to its maintenance date. Having said this, it is important to point out that if we predict the days to maintenance for the dates inside the i-th sector, the training set must be fitted using the data belonging to the previous one. 7.2 and 7.3 show this aspect. Following this rule is necessary, otherwise if we train a model using the data belonging to the same sector we are predicting, we skew the final results because actually we don't know the values of the target variable passed to fit the model. Using this approach it is not possible to forecast the days to the first available maintenance.

As regards the training set used to train a model we already said that it is composed by the records related only to previous maintenances, in particular we decided to select all the available data for each previous time period preceding that one we are analyzing. This methodology was applied using a training set built concatenating

all the training sets previously used(an expanding approach). A further approach could be based on the usage of data belonging to a single time frame that is that one preceding the temporal window currently analyzed(sliding approach). We didn't apply this approach because we wouldn't take advantage of the available samples. The window of records is updated, concatenating the records related to the data already analyzed, when the target variable is predicted for a new time period.

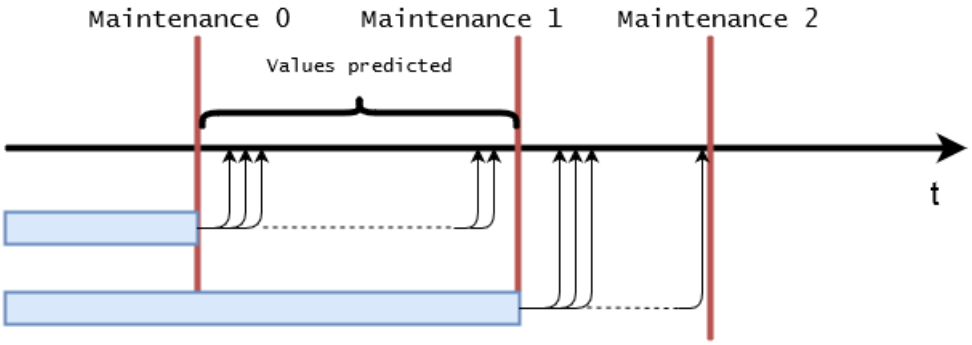


Figure 7.2: Application of methodology 1 with the expanding approach.

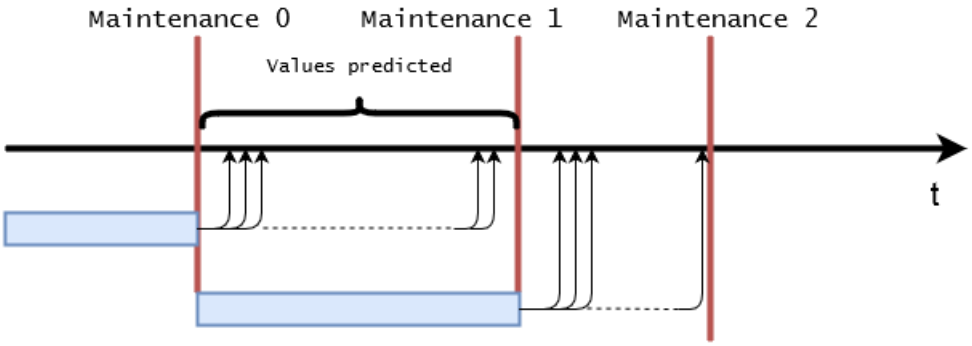


Figure 7.3: Application of methodology 1 with the sliding approach.

The blue rectangles in 7.2 and 7.3 represents the records used to fit the model for the prediction phase.

A variation that we applied is the use of training sets with a predefined number of data samples. The final decision was to select the 30 records preceding the maintenance, as we are interested in obtaining an increasing accuracy in the forecasts as we approach the exceeding of the usage threshold. In this way we expect the model fitted is able to get a better accuracy when we get close to the maintenance, because we apply a model trained on specific data samples. On the other hand we expect a worse performance for predictions not included inside the temporal range chosen.

The main advantage of this methodology is the usage of a single model whenever we forecast the target variable for the current temporal interval between two maintenances, we need to generate a new one when another temporal interval is analyzed. In this way much time is saved, and the most computational effort is spent for the forecasting phase.

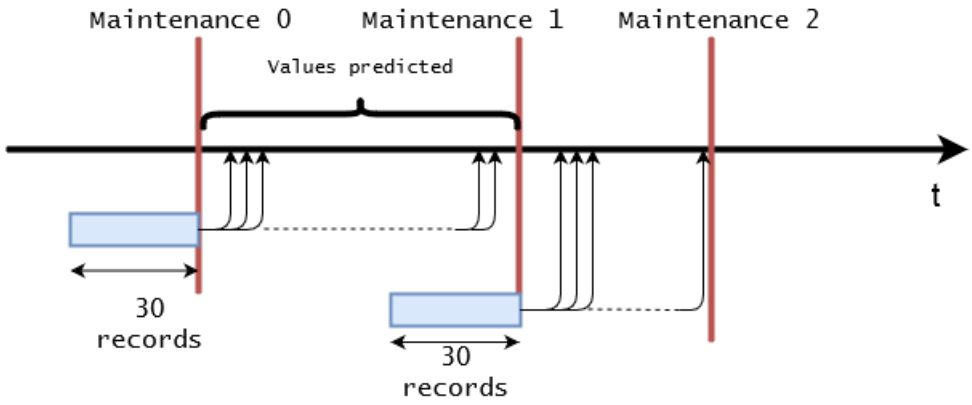


Figure 7.4: Application of methodology 1 with training set reduced .

7.2.2 Methodology 2: Cumulate daily utilization time predictions

The main idea of this methodology is: for the current training set the integral sum of utilization hours is available, predicting the variable *onofftime_second* for the subsequent days we can know how this integral sum changes and when the usage threshold is reached(in this case the date is saved and the window is updated). This way the remaining days to maintenance for the current date, the last in the training set, are computed by subtracting it from the predicted one. The independent variables with this approach are the dataframe's columns $onofftime\_second_{\text{t-i}}$. As well as the first approach presented, also in this case it is necessary to consider that we are working with a time series and the consequences associated to this aspect influence the application of this methodology. We start applying the algorithm with a predefined number of data samples, used to train the first model. We perform the first prediction for the day after the current one and compute the integral sum of utilization hours using the value just forecasted. If the result exceeds the usage threshold the days remaining to maintenance are computed subtracting the date

the threshold is reached by the current one(the window can be updated with the next real value of utilization hours), otherwise we continue the forecasting phase adding the value predicted to the starting training set and fitting a new model for the subsequent predictions. This approach is affected by two weakness

- the current training set must be updated every time a new forecasting is performed adding the predicted element to the training set, a new value of utilization hours is predicted with the model fitted on the new training set. The direct consequence of this is a **propagation of the forecast error** when the regression activity is performed for data temporally far away from the starting window of records

- **big computational effort** - the fitting phase is repeated for each value predicted in the future(e.g. y_{t+i}) because the dataframe on which the operations are performed has to be modified adding the samples predicted generating a new model

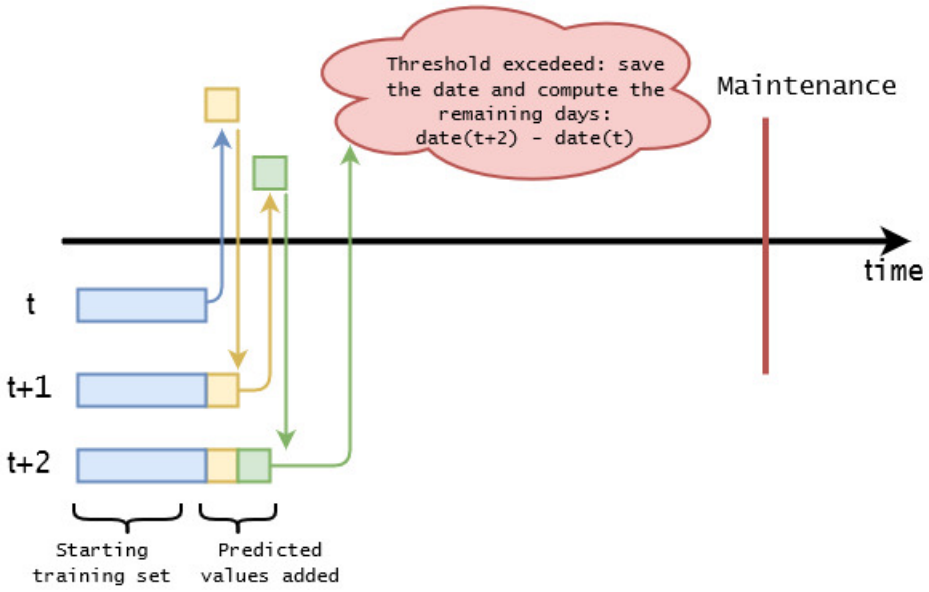


Figure 7.5: Application of methodology 2: each value predicted (indicated with a different color) is integrated to the starting training set, then a new model is fitted for the next prediction

Chapter 8

Experiments

8.1 Experimental design

The experiments were run on an Intel Core i7-4600U CPU 2.10GHz x 4, running Ubuntu 16.04 LTS 64-bits.

For the methodologies PredictDays2Maintenance/PredictWeeks2Maintenance and CumulateDailyPredictions seven units belonging to C3-T2-A were analyzed. For each one the results have been collected executing the approach based on expanding window because we want to exploit all the available data. The critical parameter associated to the execution of the scripts is the number of records to train the regression model, that we indicate as **N**, and the number of features **M** referred to the past(the lags), that we have indicated as $feat_{\text{t-i}}$ in 7.1. We expect that by varying the dataframe sizes **N**x**M** we obtain different outcomes. All the possible values for the number of lags are presented in details with the next presentations of results.

As already mentioned in 4.2 the algorithms used to obtain the forecasts of days to maintenance are linear regression, random forest, gradient boosting and support vector regression. As already mentioned, the Scikit-learn library [20] was used to perform the analysis. The hyperparameters, for the algorithms requiring them, have been set up with the following values:

- RF
  - n_estimators=50

  - max_features=auto
  - max_depth=10
  - min_samples_split=5,
  - min_samples_leaf=5

- GB
  - n_estimators=500
  - max_depth=4
  - min_samples_split=2
  - learning_rate=0.01
  - los=ls

- SVR
  - kernel=linear
  - C=10
  - epsilon=0.1

8.2 Evaluation and visualizations of performances

Considering that we are applying two different techniques of supervised machine learning, classification and regression, the performance evaluation for the performed experiments must be based on two different analysis metrics.

8.2.1 Evaluation metrics for regression techniques

Performing a regression task we get for each real value its predicted one, in statistics the difference between this two numbers is called *residual*, as residuals are the difference between data points they are sometimes called errors. The concept expressed above can be briefly represented by the mathematical formula

$$error_{y_{i,real}} = ||y_{i,pred} - y_{i,real}|| \qquad with \qquad 1 \leq y_{i,real} \leq 30$$

The same formula is used to quantify the error associated to the weekly experiments. Of course $y_{i,real}$ is refered to the number of weeks and is a number in a more restricted interval [1,10]. This error is averaged considering the entire set of vehicles on which the experiments are performed. We consider the absolute value, if not when we compute the negative and positive means they offset each other by generating a skewed value. The average compute is represented separately for a predefined number of days to maintenance(for instance from 1 to 31), that's because allows understanding better the quality of predictions when we get closer to the maintenance task. If we compute an evaluation metric without distinguishing between days closer to the maintenance and that ones more far the analysis of results doesn't make sense, because we treat a prediction made one month before in the same way as that one made one day before. In this way we can have a general idea about the results obtained and how the distribution change considering the decreasing number of days tilling the maintenance task.

All the techniques mentioned above are executed considering the results got with the two methodologies presented above, both applied with different lags used as features and regression/classification techniques.

Before starting to analyze the results associated to the regression algorithms we need to introduce some benchmark algorithms, allowing us to have a judgment. The first one computes the remaining days to maintenance for the i-th day using the formula

$$days\_to\_maint(t) = \frac{(hours\_to\_maint(t-1) – avg\_util\_hours(0,t-1))}{avg\_util\_hours(0,t-1)}$$

the average is computed using all the available utilization hours in the past, otherwise employing a restricted set of values it is more probable that the average is a number close to zero and the ratio would generate a high number, and so a bad prediction that would affect the quality of the final results. A second method using a formula similar to the previous one but exploiting the utilization hours at t-1, instead of the average, has been used but relevant and frequent peaks has been generated that don't allow comparing graphically the benchmark with the ML algorithms. For this reason only the error trend for the moving average technique will be displayed and compared with the regression techniques. All the following plots have on the x axis the number of days/weeks to maintenance while on the y axis the average residual error, the legends show the color associated to the error obtained with the baseline

or with a certain number of lags used to train the models .

8.2.2 Evaluation metrics for classification techniques

When a classification problem is faced, in our case binary classification, the precision associated with the model used is deduced by calculating four different parameters: true positive (TP), false positive (FP), true negative (TN) and false negative (FN). TP is a value used to understand how well the model predicts correctly the positive class, while TN is used for the same purpose but with the negative class. FP and FN quantify the incorrectly predicted labels. Starting from these parameters further metrics can be computed: *accuracy*, *precision*, and *recall*. An exhaustive explanation is available at [30]. Accuracy is the fraction of predicted labels correctly, for the binary classification the formula is calculated as

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

However, where there is a significant disparity between the number of positive and negative labels this parameter is not a significant value. For instance, if in a binary classification(yes/no) problem the 90% of values belong to the class *no* and the remaining 10% has *yes* label, the total accuracy will be a high value but the recall for the class with few data samples has a recall close to 0%.
Precision and recall give more info relative to a single class, allowing getting more details than accuracy.

$$precision = \frac{TP}{TP + FP} \qquad recall = \frac{TP}{TP + FN}$$

For each vehicle precision, recall and accuracy are computed, finally an average value for each metric is calcuted.

8.3 PredictDays2Maintenance results

After deciding the values for the hyperparameters you need to select a value for the number of lags used as features. Regarding this, the subsequent values were chosen **1, 7, 15 and 20** lags. The number of records belonging to the training set is a value that varies based on the temporal distance, in days, between two

maintenance operations. Therefore, it cannot be set a priori because for each vehicle two maintenances occur with variable time distance.

The next pictures show the trend of the error for each regression algorithm

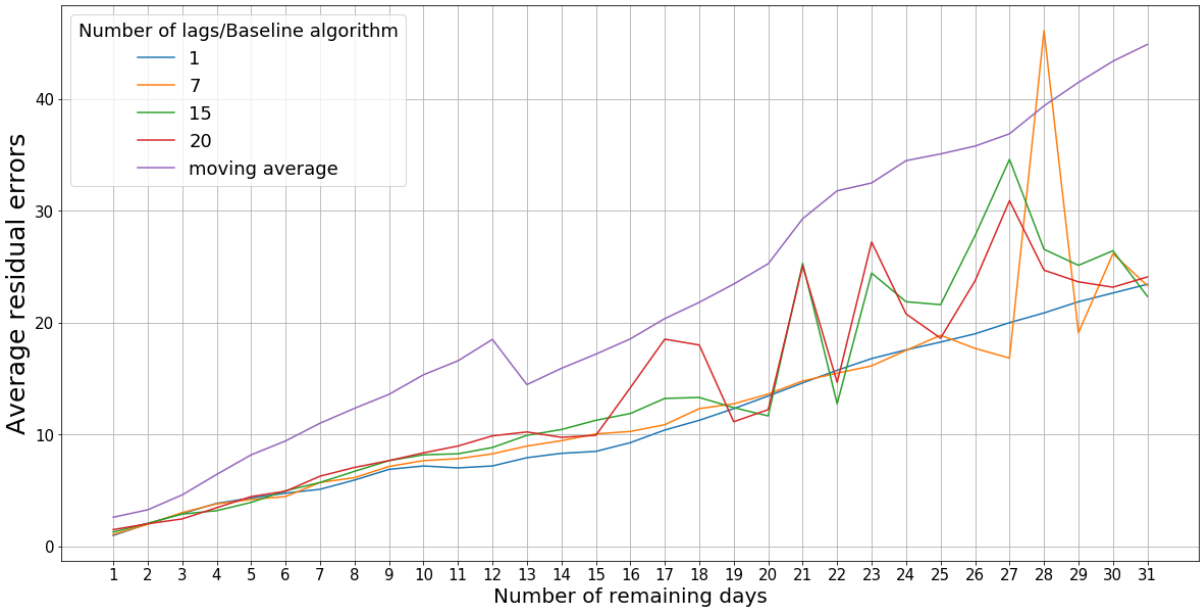


Figure 8.1: Residual error for LR with PredictDays2Maintenance

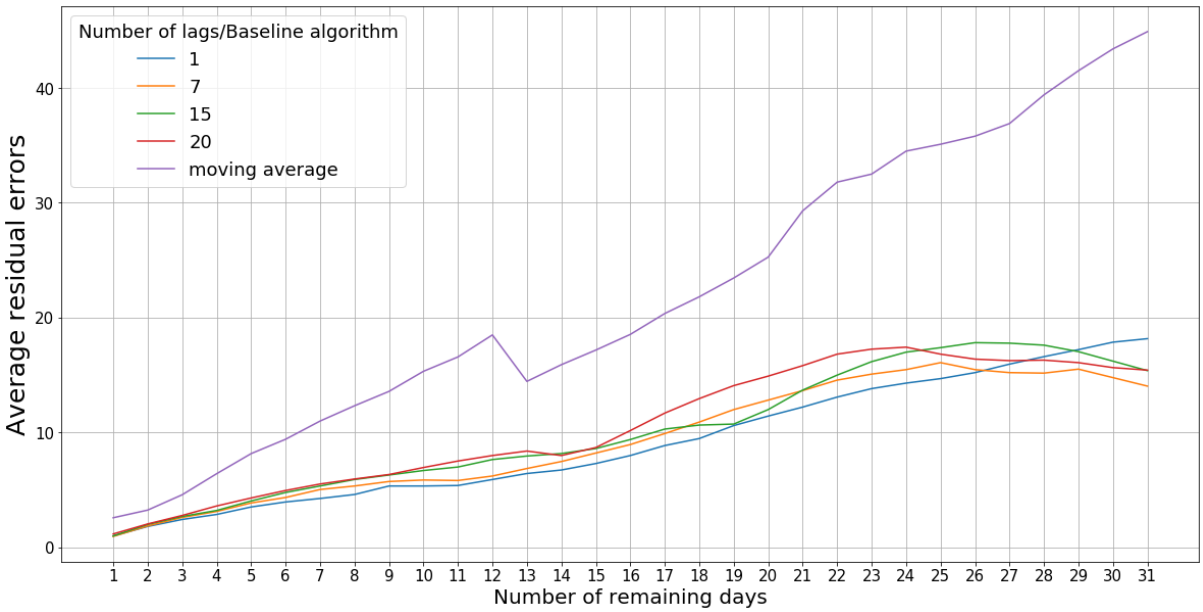


Figure 8.2: Residual error for SVR with PredictDays2Maintenance

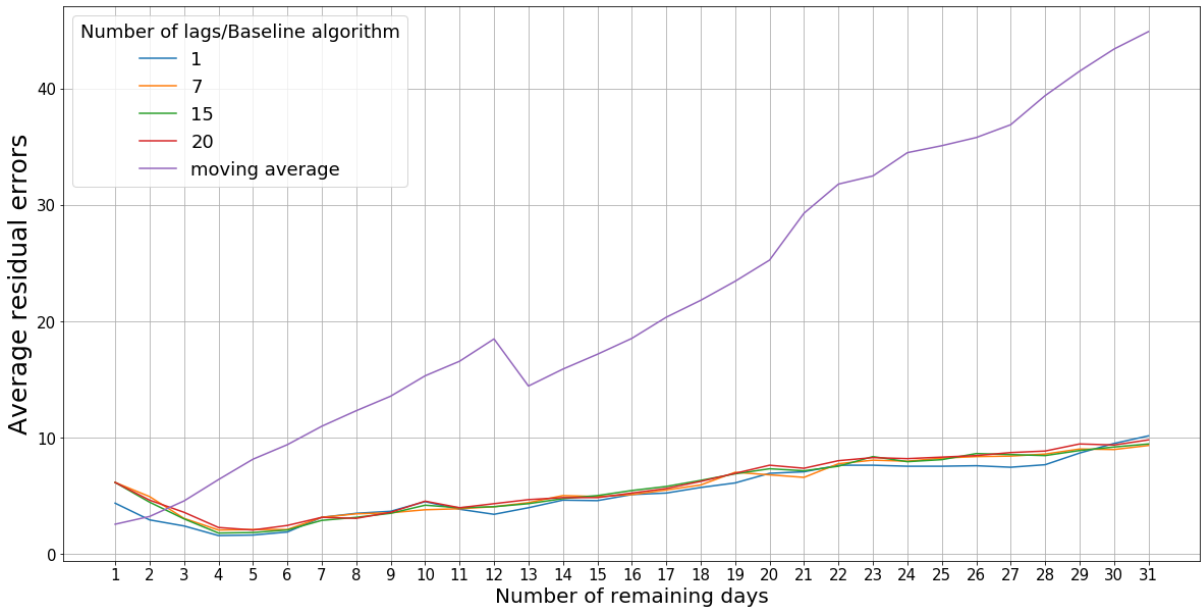


Figure 8.3: Residual error for RF with PredictDays2Maintenance

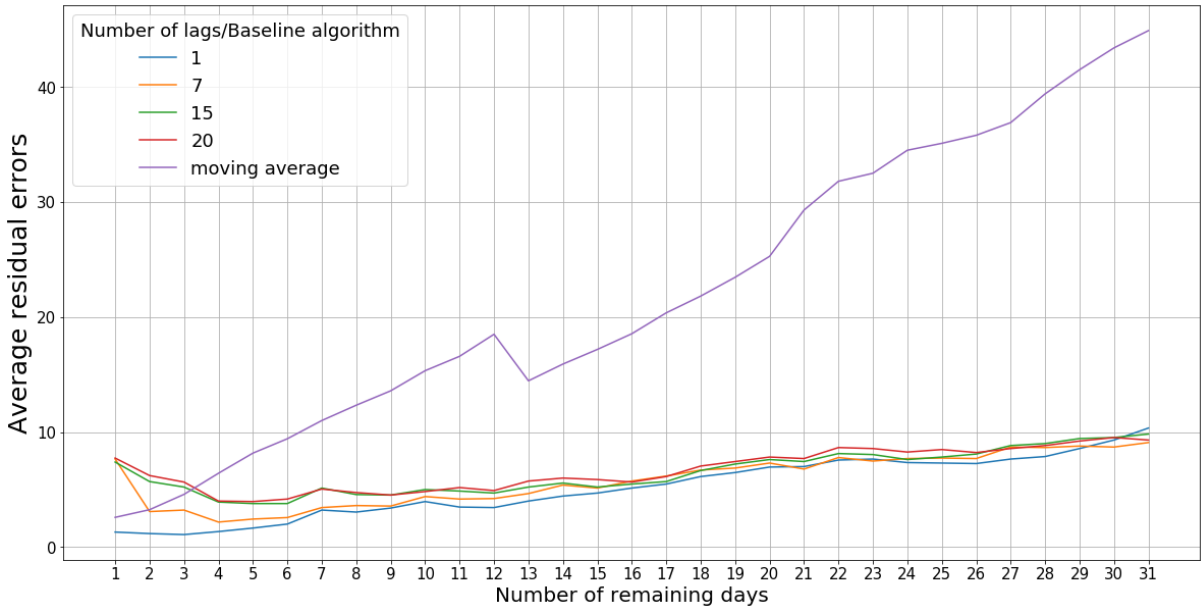


Figure 8.4: Residual error for GBR with PredictDays2Maintenance

It is clear by observing the graphs that the most performing algorithm is SVR with linear kernel. The error for this algorithm decreases as you approach the day of maintenance. The best results are obtained using **1 and 7 lags**, however even the remaining values do not deviate significantly. Good performance is also achieved with linear regression, provided that a single lag is used. Otherwise, the error has a jagged trend especially for the last two weeks preceding the maintenance task.

On the hand, RF and GB have a good decreasing error trend for the days less close to maintenance, but an unexpected increase in the week before. The inability to predict the days close to maintenance could be due to the nature of the algorithms, or the choice of inappropriate values for hyperparameters.
Considering the results achieved by the baseline is evident that any regression algorithm works better than the baseline, except for the gradient boosting and random forest having the worst performance for the three/four days near to the maintenance. Support vector regression with linear kernel pointed out the positive results discussed above, indeed the error is lower than that one obtained with the baseline using any number of lags as features. On the other hand, using linear regression we need to select carefully the number of lags in order to get better results, using 1 lag the error trend is not jagged.

8.3.1 PredictDays2Maintenance, results with alternative approach

The strange behavior of RF and GB for the predictions in the days close to maintenance has led us to use the variation in the choice of the training set for the PredictDays2Maintenance, as explained in detail in 7.2.1. The results improved especially for GB and RF allowing to eliminate the surge of error for predictions in days near to maintenance. Furthermore, linear regression has achieved good results in a longer time horizon with all the lags employed. We can therefore conclude that the non-uniformity in the time length of training sets, used to train regression models, leads to non-negligible errors in the vicinity of maintenance for the bagging and boosting algorithms. For the rest, the improvements in the results are evident even not in the vicinity of maintenance. It is important to emphasize this aspect, since the temporal distance between the maintenance of the same vehicle is a variable parameter, therefore it is appropriate to choose it adequately. A further variation, based on the choice of records with utilization hours greater than zero has been applied, however no significant improvements were noted. The most relevant results mentioned above have been represented in the following pictures.

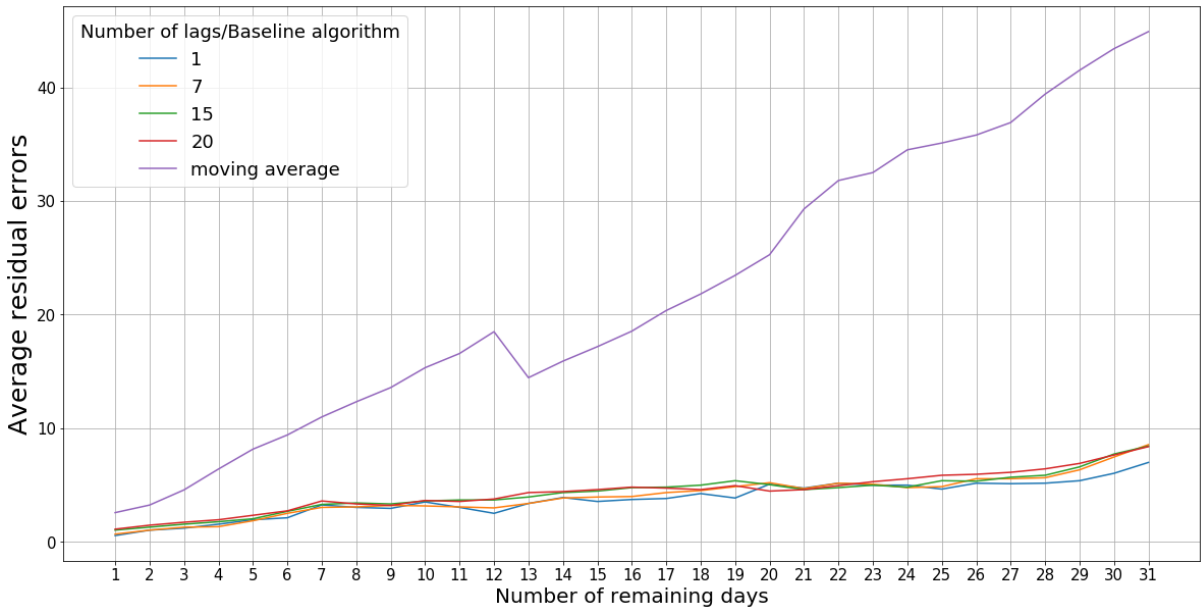


Figure 8.5: Residual error for GBR with PredictDays2Maintenance and reduced training set

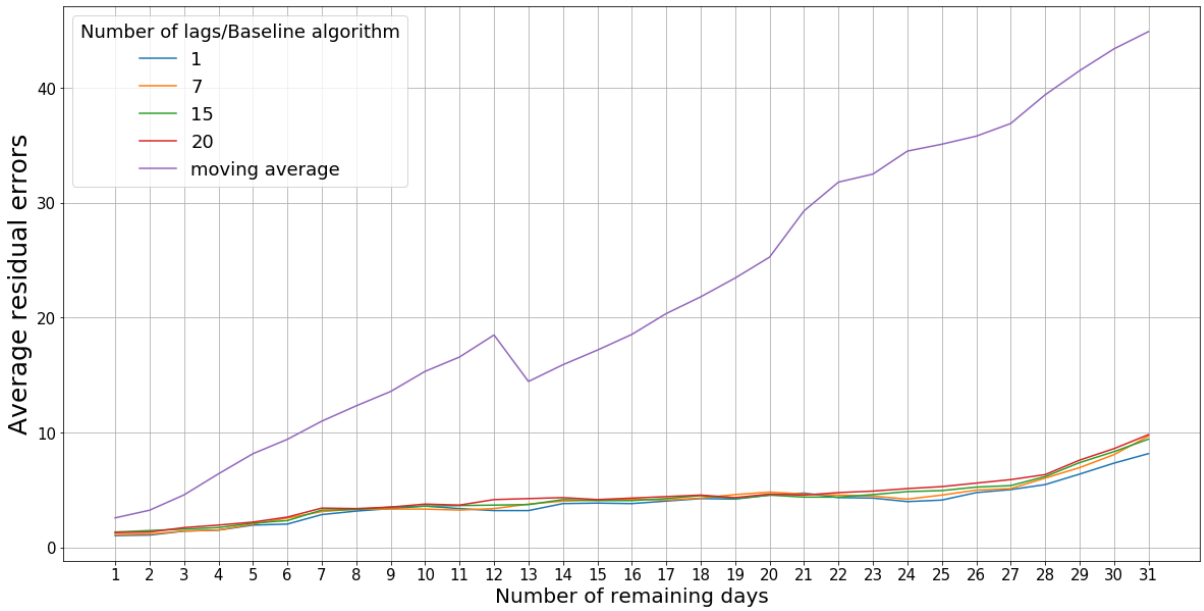


Figure 8.6: Residual error for RF with PredictDays2Maintenance and reduced training set

8.4 CumulateDailyPredictions results

With this methodology the number of lags used as features is the same of the first one, therefore 1, 7, 15 and 20. We focus the attention on the last 30 days

close to the maintenance, for this reason the window of records employed to train the regression model contains all the available data samples collected in the days preceding the thirtieth for the current maintenance task. Plotting the residual error we can point out the good performance scored by linear regression, better than the baseline chosen. The good results mentioned above have been achieved with all the lags used as features. However, more than one lag allows to have a better accuracy for the last three weeks preceding the maintenance task. The error detected on the thirtieth day is, in the best case, about 17(meaning that the error is predicted with 17 days in advance or delay) and decreases linearly getting close to the upkeep scoring an error close to zero on the last three days(i.e. it is very likely to predict when the vehicle usage threshold will be exceeded near the upkeep). On the other hand, GB and RF mark the worst performance especially for the days from the thirtieth(about 60) to the tenth preceding the maintenance, and using any number of lags. The accuracy improves in the last week of predictions getting close to zero. If the error trend decreases linearly for LR, GB and RF, a different behavior has been detected for SVR with a rbf kernel and a linear one. Relevant peaks have been detected plotting the residual error. The worst score is obtained with a radial basis function kernel, in fact the error plot is over the baseline plot for most of the values on the x axis. Using a linear kernel the performance enhance, as long as more than 1 lag is employed to train the regression model. The following plots show the results

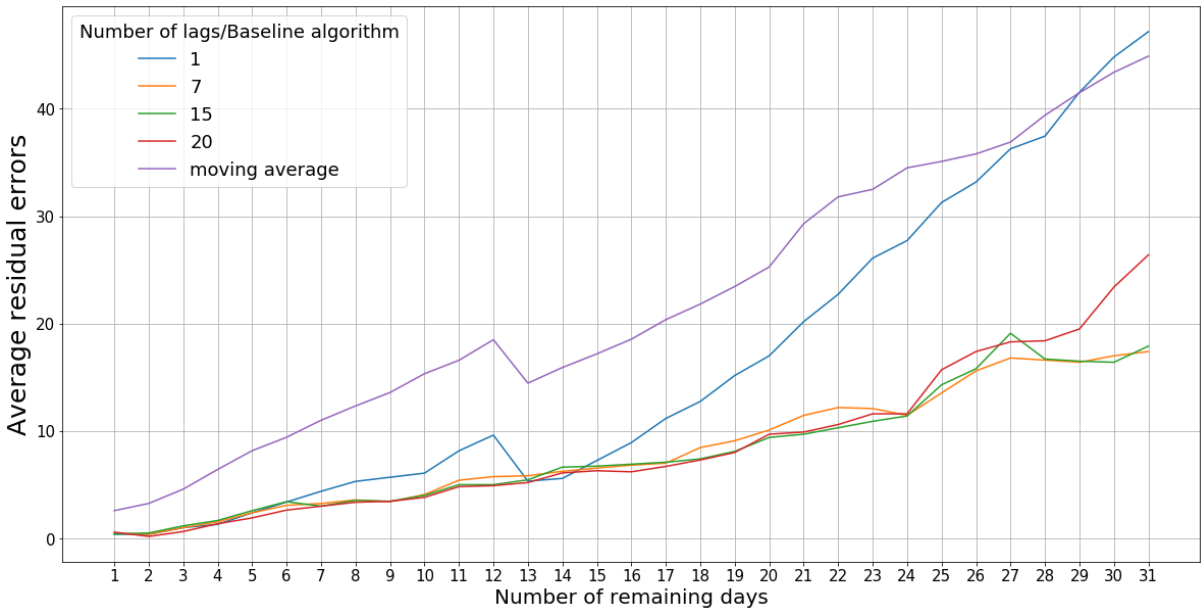


Figure 8.7: Residual error for LR with CumulateDailyPredictions

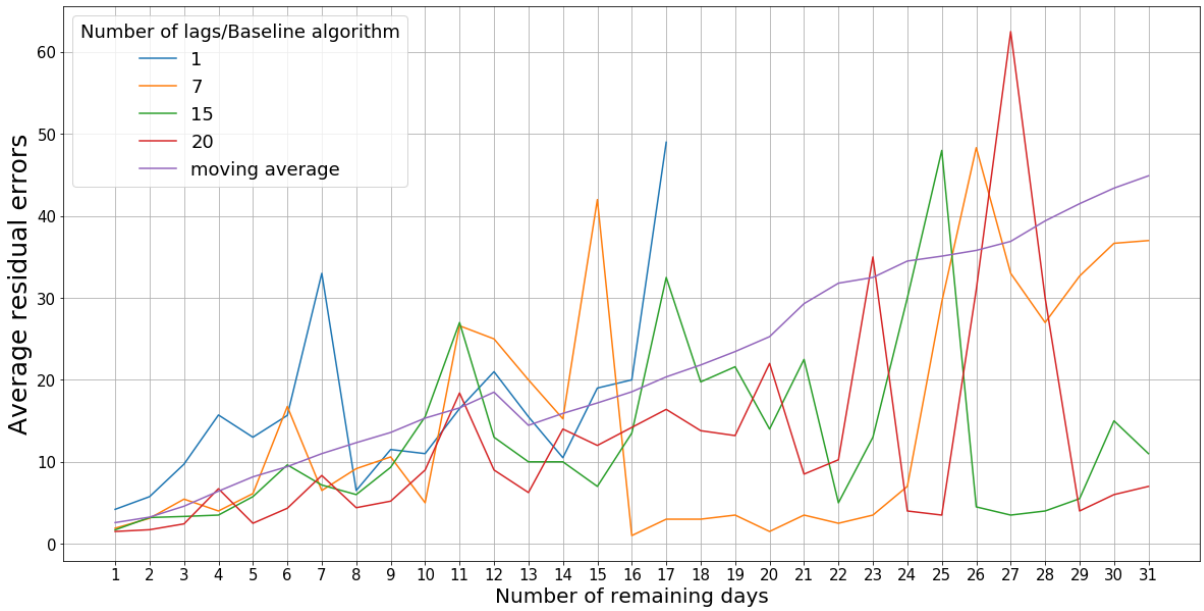


Figure 8.8: Residual error for SVR with CumulateDailyPredictions

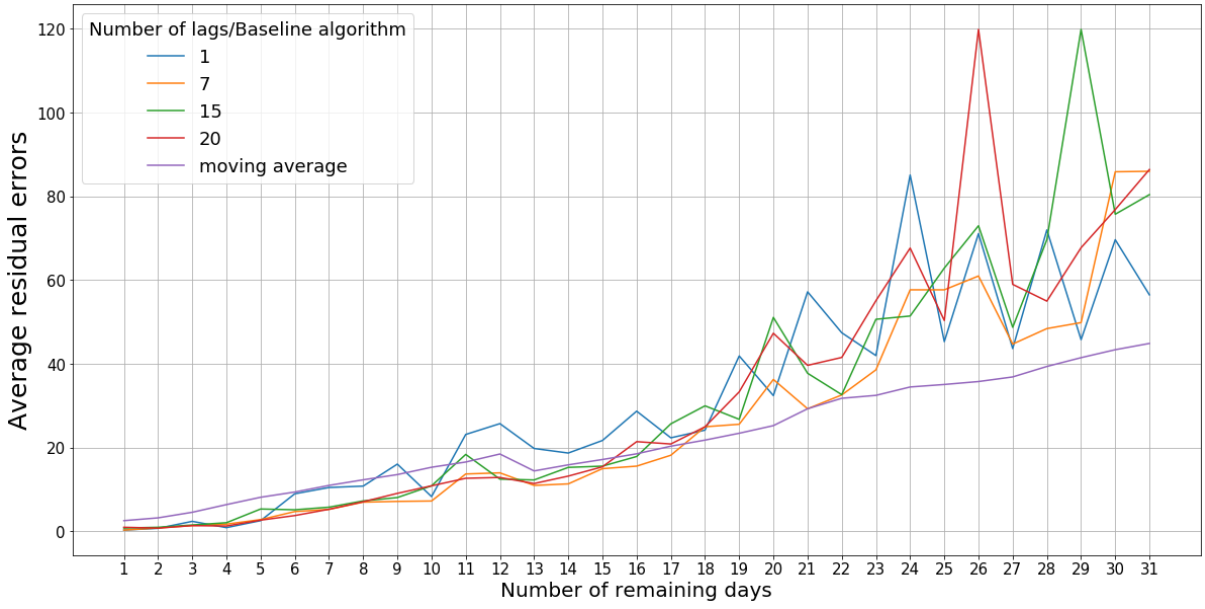


Figure 8.9: Residual error for RF with CumulateDailyPredictions

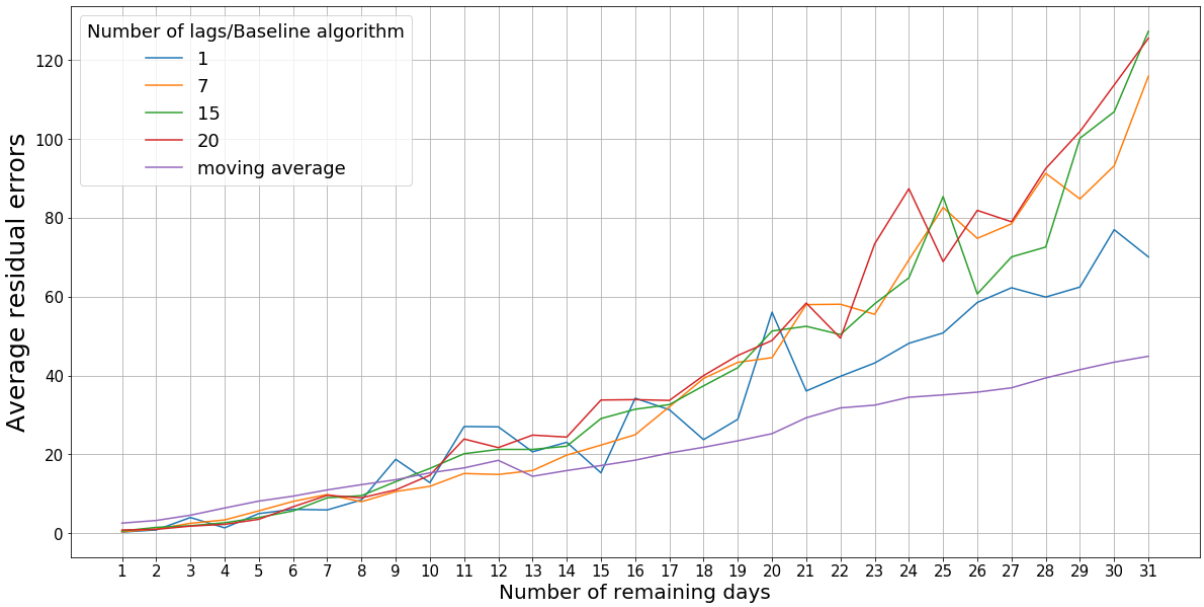


Figure 8.10: Residual error for GBR with CumulateDailyPredictions

The variation related to the CumulateDailyPredictions performs the forecasts discarding the record with daily utilization hours equal to zero and so predicts the working days to maintenance. In this case all the regression algorithms have an error greater than that of the baseline in the weeks next to that one preceding the maintenance. Therefore, relevant improvements are not detected. The best results are achieved by linear regression, especially with a number of lags greater than 1. The error plot overlaps with that one of the new baseline(scoring better performance compared with the version with zeros), however in the seven days preceding the maintenance the linear regression guarantees us to make forecasts with greater precision.

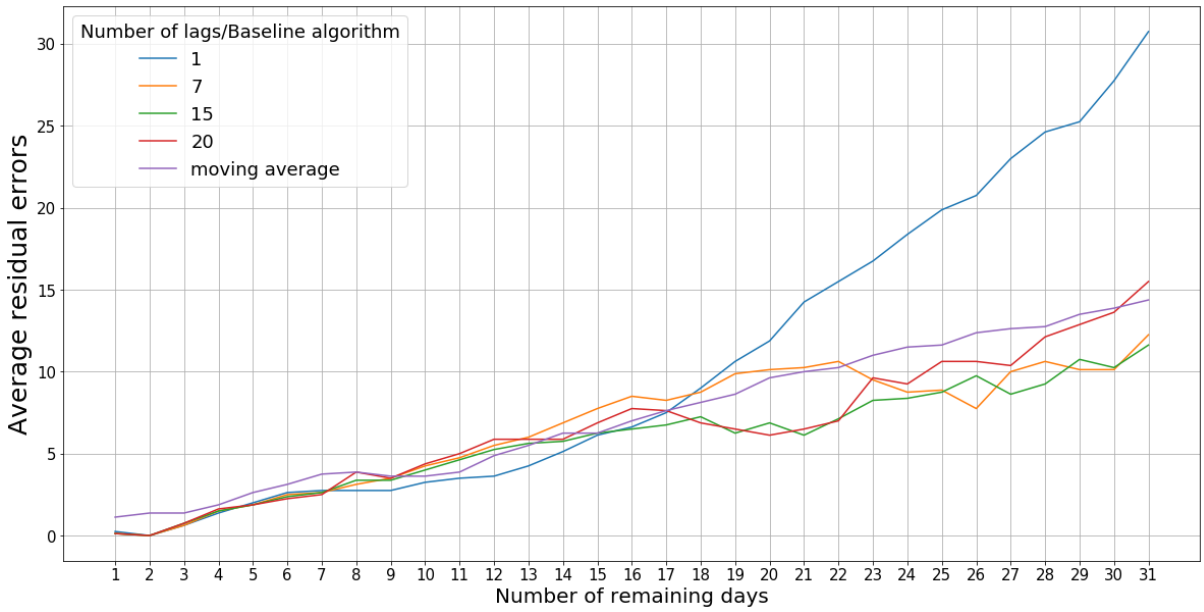


Figure 8.11: Residual error for LR with CumulateDailyPredictions without zeros

8.5 Tuning of hyperparameters for daily experiments

The values for the hyperparameters of the regression algorithms have been chosen using the default parameters suggested by the Scikit-learn library, however it is necessary to repeat the experiments(we decided to run again the PredictDays2Maintenance to save time and choosing 7 lags, that seems to be a reasonable value) on the selected vehicles using different values for the hyperparameters. For each hyperparameter different values were selected, the regression algorithms were performed by testing all the possible combinations for the values chosen. In this way it is possible to understand if certain values can generate results with greater accuracy, or if there are no relevant differences. For each combination of values an **average residual error**, related to the thirty days preceding the maintenance task, has been computed because it is not possible to show the large number of plots representing the error trend. For this reason we preferred to represent the error associated to a combination of values in tabular form. For each algoithm the values selected to perform the tuning are:

- gradient boosting:

  - number of estimators: 400, 500, 700
  - max depth: 3, 6, 9
  - mean samples leaf: 2, 4, 6
  - learning rate: 0.01, 0.5, 1

- random forest:
  - number of estimators: 50, 70, 100
  - max depth: 10, 20, 30
  - mean samples split: 2, 5, 10
  - mean samples leaf: 1, 3, 5

- support vector regression:
  - kernel: rbf, linear
  - C: 1, 5, 10
  - epsilon: 0.01, 0.05, 0.1
  - gamma: auto, 0.01, 0.05, 0.1

The most relevant result can be seen for the svr algorithm with rbf type kernel, in fact the average residual error in the best case is equal to 5.63 while in the worst case it is 27.95, in general the worst results are achieved with the lowest values of C and gamma among those chosen. In case this algorithm, with kernel rbf, is chosen the hyperparameters must be selected carefully. For the remaining algorithms there are not considerable differences between the best and worst results. The following tables show the best and worst average errors obtained.

n_estimat	max_dept	min_sam	learning_r	error
700	3	4	0.01	5.646
700	3	2	0.01	5.646
700	3	6	0.01	5.671
500	3	6	0.01	5.68
500	3	2	0.01	5.684
500	3	4	0.01	5.686
700	6	2	0.01	5.739
700	3	4	0.5	5.741
700	6	6	0.01	5.762
400	3	4	0.5	5.781
700	6	4	0.01	5.784
500	3	6	0.5	5.786
500	6	2	0.01	5.804
500	6	4	0.01	5.816
500	6	6	0.01	5.823
500	6	2	0.5	5.843
400	3	2	0.01	5.862
400	3	6	0.01	5.862
700	9	6	0.01	5.875
700	3	2	0.5	5.877
400	3	4	0.01	5.881

Figure 8.12: Tuning results with GB

C	eps	gamma	error
10	0.05	auto	5.632
10	0.1	auto	5.633
10	0.01	auto	5.633
10	0.01	0.1	5.982
10	0.1	0.1	6.086
10	0.05	0.1	6.293
5	0.01	auto	6.301
5	0.1	auto	6.304
5	0.05	auto	6.306
10	0.01	0.05	6.733
1	0.1	0.01	27.946
1	0.05	0.01	27.95
1	0.01	0.01	27.952

Figure 8.13: Tuning results with SVR and kernel rbf. The red rectangle points out how the average error gets worse using low values for C and gamma.

As regards the tuning results associated to random forest and svr with linear kernel, for which we didn't represent their complete scores, they are the ones that resulted in less significant differences in the use of different parameters. We show the best and worst configuration and the associated average error scored:

- svr with linear kernel:
  - best configuration: C=1 eps=0.01 gamma=auto, error=9.23
  - worst configuration: C=5 eps=0.01 gamma=auto, error=9.31
- random forest:
  - best configuration: number of estimators=50 max depth=20 min samples split=2 min samples leaf=3, error=5.49
  - worst configuration: number of estimators=70 max depth=30 min samples split=10 min samples leaf=5, error=5.91

8.6 PredictWeeks2Maintenance, weekly results

After performing these experiments we analyzed the errors achieved using 7 lags and figuring out in the same plot, and for each algorithm, the results obtained keeping records with usage hours equal to 0(calendar weeks) and greater than 0(working weeks). As for the daily plots, the baselines is represented.
Looking at the plots it is clear the bad performance scored by the baseline to predict the calendar weeks and indicated as moving average in the legend, with an error equal to 40 when it's 10 weeks away to the vehicles maintenance. On the other hand, the same baseline but predicting the working weeks presents an error between the values 1 and 5 with a decreasing trend from the ninth week. Considering the results related to the calendar predictions, all the regression algorithms generate more or less the same results: an error trend between 0 and 5(maintenance predicted at most with 5 weeks late or in advance) with a decreasing trend. The only exception is the linear regression generating a more jagged trend. Using regression algorithms the predictions are much more accurate than the baseline.
The relevant difference between the baseline and the regression algorithms detected above are not confirmed for the errors associated to the working weeks predictions. The error range is the same for baseline and regression algorithms, however the accuracy is better with machine learning techniques from the sixth week, in the following weeks the two approaches generate overlapping plots. It is important to note that rfr and gbr, which generated an increase in error in the next maintenance stages, do not exhibit the same behavior for weekly forecasts. This can be justified by the narrower range of values of the target variable that the algorithms use in the training and forecast phase. It is therefore useless to use the approach based on reduction of the training set applied in method 1 with daily forecasts.

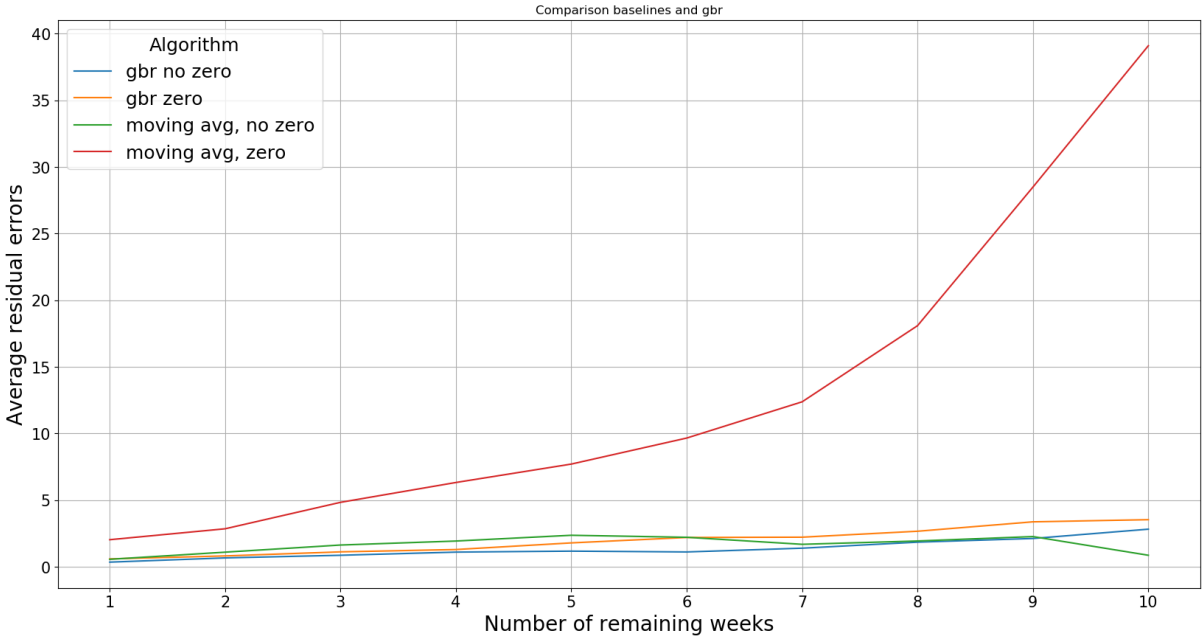


Figure 8.14: Weekly comparisons between baselines and GB

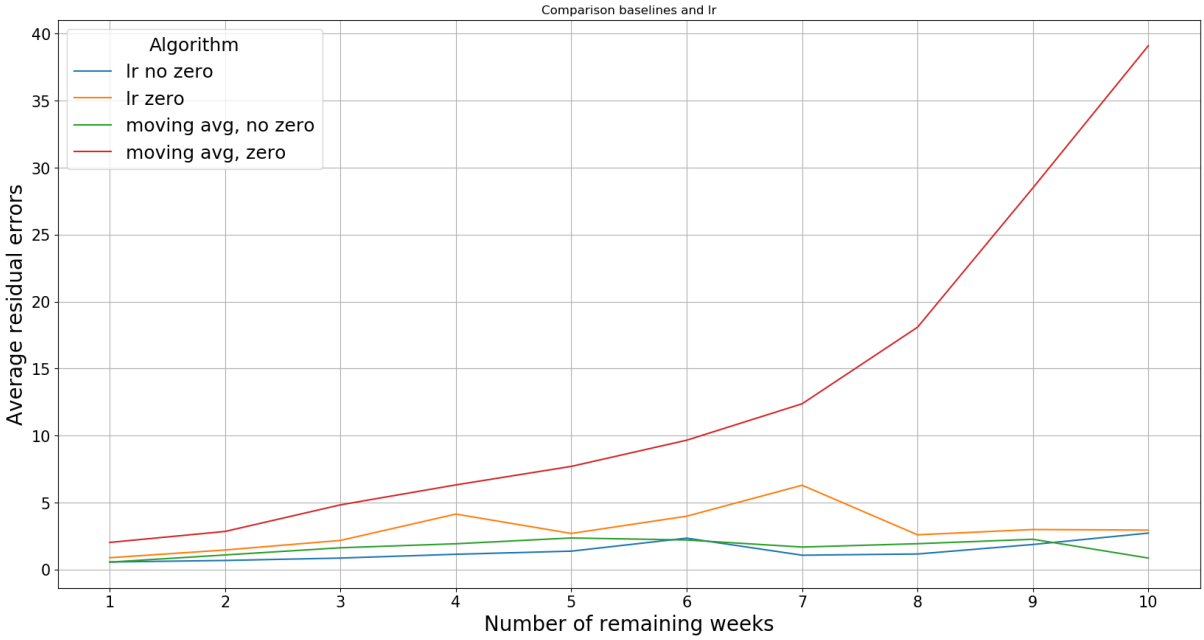


Figure 8.15: Weekly comparisons between baselines and LR

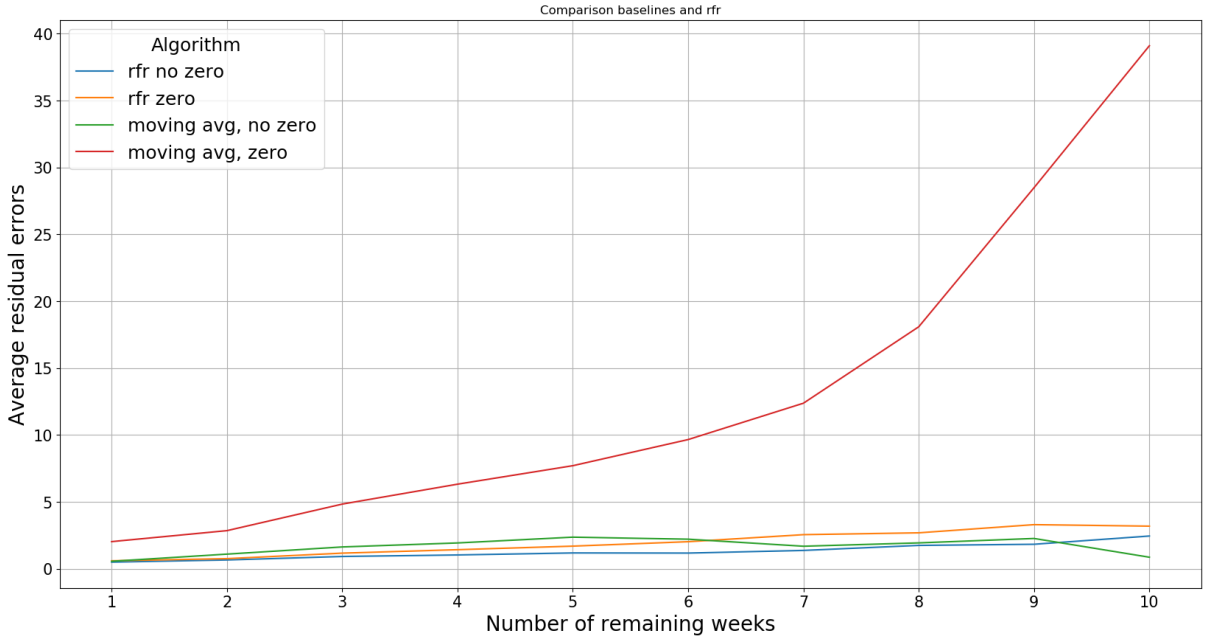


Figure 8.16: Weekly comparisons between baselines and RF

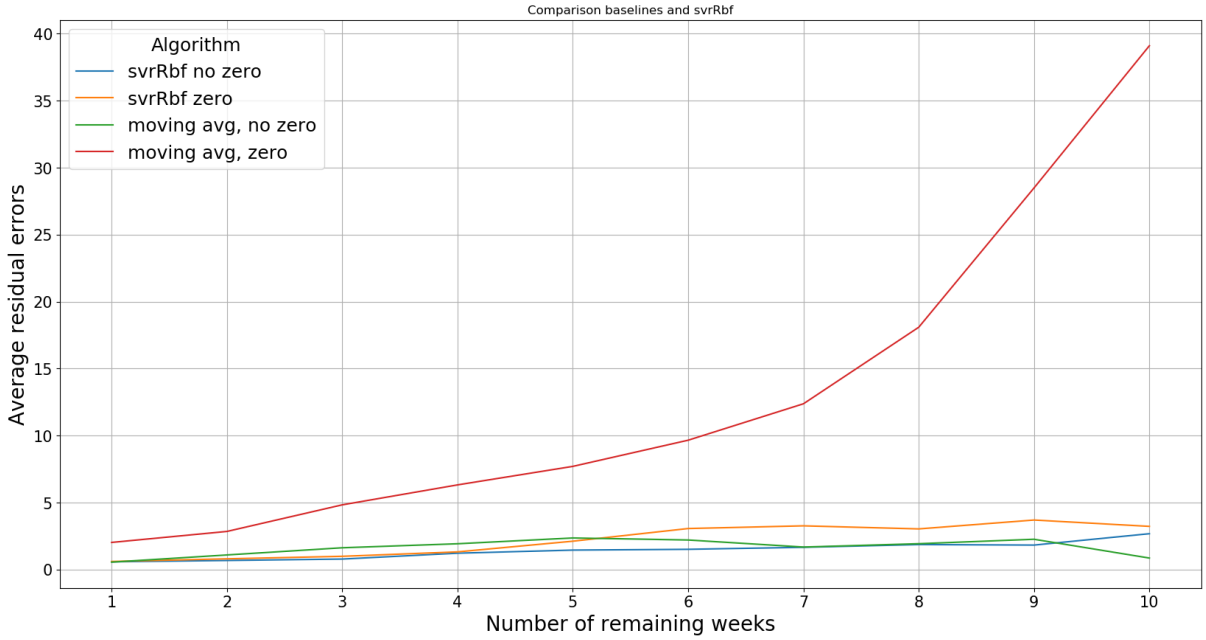


Figure 8.17: Weekly comparisons between baselines and SVR with kernel rbf

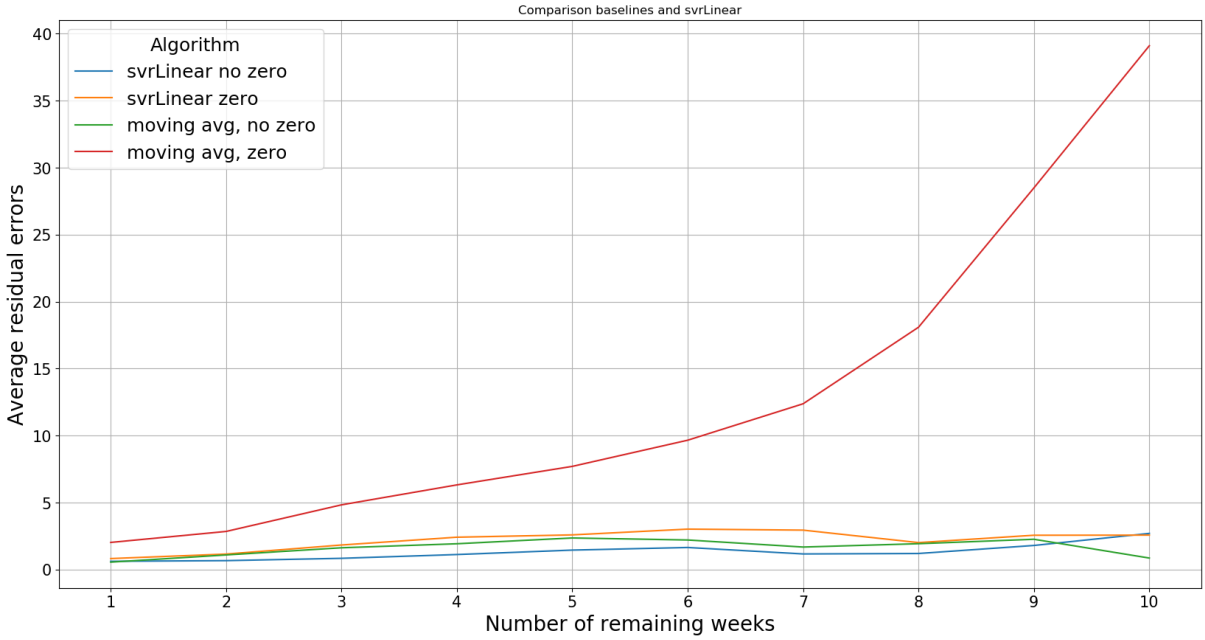


Figure 8.18: Weekly comparisons between baselines and SVR with linear kernel

In light of what is shown by the previous graphs, in particular by observing the comparison between the baseline obtained without excluding records with usage equal to zero and the error trend for the regression algorithms with the same calendar data, this methodology and relative weekly granularity for the target variable are to be considered the best strategy to use. Especially considering further developments proposed by Tierra related to this line of work.

8.7 Tuning for weekly experiments

The experiments related to the tuning of the hyperparameters has been repeated also for the weekly target variable, trying to understand if for some regression algorithms the choice of certain values can lead to results with a better or worst accuracy. The calculated average residual error refers to the five weeks prior to maintenance. By analyzing, for each algorithm, the error obtained with the best and the worst configuration we can briefly summarize the results obtained

- svr with linear kernel:
  - best configuration: C=1 eps=0.01 gamma=auto, error=1.504
  - worst configuration: C=10 eps=0.01 gamma=auto, error=1.559

- svr with rbf kernel:
  - best configuration: C=10 eps=0.1 gamma=auto, error=0.843
  - worst configuration: C=1 eps=0.1 gamma=0.01, error=1.445
- random forest:
  - best configuration: number of estimators=50 max depth=10 min samples split=10 min samples leaf=3, error=0.84
  - worst configuration: number of estimators=100 max depth=30 min samples split=5 min samples leaf=1, error=0.886
- gbr:
  - best configuration: number of estimators=400 max depth=3 min samples split=2 learning rate=0.01, error=0.843
  - worst configuration: number of estimators=400 max depth=6 min samples split=4 min samples leaf=1, error=1.217

Observing the errors obtained with the various combinations of parameters it is clear that there are no significant differences between the worst and best case errors. Therefore, for any algorithm, an approximate choice of values for hyperparameters would lead us to obtain results with errors that cannot be improved by a lot.

8.8 Classification results

Before analyzing the results it must be premised that the number of class labels with value *yes*(i.e.there will be maintenance tomorrow or next week) is negligible compared to the labels with *no* value, therefore any kind of classification algorithm shouldn't be able to get high scores. The best results related to the classification have been obtained for both the target variables *maint_next_week* and *maint_next_day* using data frames in which records with use equal to 0 have been deleted, therefore using info associated to the working days and not the calendar ones. That's because decreasing the number of records with usage equal to 0 the related number of records with class target variable *no* decreases too, therefore it's much easier correctly predict the class labels in particular the class label *yes*. Moreover, the best

results have been obtained by using the algorithm Gradient Boosting and Random Forest, significant differences associated with the number of lag used are visible only with experiments on a daily basis. Precision and recall presented in the following tables are referred to the label *yes*.

Daily results with zeros

alg	n_lags	accuracy	precision	recall
svrLinear	1	0.985931	0.000000	0.000000
svrLinear	7	0.985931	0.000000	0.000000
svrLinear	15	0.985931	0.000000	0.000000
svrLinear	20	0.985931	0.000000	0.000000
svrRbf	1	0.985931	0.000000	0.000000
svrRbf	7	0.985931	0.000000	0.000000
svrRbf	15	0.985931	0.000000	0.000000
svrRbf	20	0.985931	0.000000	0.000000
rfr	1	0.988196	0.500000	0.114583
rfr	7	0.986542	0.250000	0.027778
rfr	15	0.986542	0.250000	0.027778
rfr	20	0.986542	0.250000	0.027778
gbr	1	0.984727	0.357143	0.312500
gbr	7	0.986919	0.625000	0.340278
gbr	15	0.986255	0.305556	0.201389
gbr	20	0.985644	0.291667	0.173611

Daily resuts without zeros

alg	n_lags	accuracy	precision	recall
svrLinear	1	0.981971	0.000000	0.000000
svrLinear	7	0.981971	0.000000	0.000000
svrLinear	15	0.981971	0.000000	0.000000
svrLinear	20	0.981971	0.000000	0.000000
svrRbf	1	0.981971	0.000000	0.000000
svrRbf	7	0.981971	0.000000	0.000000
svrRbf	15	0.981971	0.000000	0.000000
svrRbf	20	0.981971	0.000000	0.000000
rfr	1	0.986248	0.500000	0.173611
rfr	7	0.983643	0.250000	0.055556
rfr	15	0.982807	0.250000	0.027778
rfr	20	0.981971	0.000000	0.000000
gbr	1	0.983482	0.525000	0.447917
gbr	7	0.984318	0.631250	0.368056
gbr	15	0.983202	0.489583	0.253472
gbr	20	0.982897	0.261905	0.170139

Figure 8.19: Daily classification results

Weekly results with zeros

alg	n_lags	accuracy	precision	recall
svrLinear	1	0.901516	0.000000	0.000000
svrLinear	7	0.891920	0.000000	0.000000
svrLinear	15	0.886474	0.000000	0.000000
svrLinear	20	0.875676	0.094697	0.051587
svrRbf	1	0.897238	0.114198	0.146825
svrRbf	7	0.895404	0.109375	0.138889
svrRbf	15	0.893570	0.105882	0.142857
svrRbf	20	0.901516	0.125000	0.134921
rfr	1	0.908051	0.533590	0.518849
rfr	7	0.903457	0.526475	0.457837
rfr	15	0.891115	0.411765	0.320933
rfr	20	0.901500	0.484270	0.368056
gbr	1	0.899083	0.503913	0.500496
gbr	7	0.899270	0.471334	0.365575
gbr	15	0.888132	0.323964	0.264385
gbr	20	0.894620	0.406250	0.298115

Weekly results without zeros

alg	n_lags	accuracy	precision	recall
svrLinear	1	0.873795	0.000000	0.000000
svrLinear	7	0.865150	0.000000	0.000000
svrLinear	15	0.858444	0.000000	0.000000
svrLinear	20	0.862897	0.121795	0.075397
svrRbf	1	0.893651	0.176667	0.210317
svrRbf	7	0.896370	0.171233	0.198413
svrRbf	15	0.896145	0.416667	0.214286
svrRbf	20	0.898879	0.185484	0.182540
rfr	1	0.920369	0.655776	0.706349
rfr	7	0.909014	0.623343	0.542163
rfr	15	0.909882	0.637798	0.524306
rfr	20	0.911297	0.634790	0.519345
gbr	1	0.910387	0.634745	0.671131
gbr	7	0.908117	0.605303	0.573909
gbr	15	0.907080	0.600965	0.518353
gbr	20	0.899809	0.538128	0.474702

Figure 8.20: Weekly classification results

The precision and recall scores never exceed 70%, based on these results the classification-based approach does not allow us to have good guarantees on vehicle maintenance forecasts.

Chapter 9

Conclusions and future works

A final comparison between the two methodologies show how predicting the number of days to maintenance, instead of forecasting it indirectly with the methodology CumulateDailyPredictions, can give better results especially for RF and GB choosing a training set with a predefined number of records. With this methodology the error decreases linearly getting close to the vehicles maintenance, achieving a value close to zero. Moreover, the residual error is always better than that one scored by the baseline. The worst results obtained with the second technique can be explained keeping in mind that every predicted value of utilization hours is added to the starting training set to train models for subsequent forecasts, this means that each model is trained using values that are affected by an error that is propagated in the following predictions. A further aspect we need to consider is the execution time needed to train models and perform the predictions. The methodologies PredictDays2Maintenance/PredictWeeks2Maintenance require a single model for the predictions in a time period, the second one trains a new model for each following value predicted in the future due to the update on the starting training set. The long execution times for the second methodology are the direct result of several training phases. PredictDays2Maintenance and PredictWeeks2Maintenance are therefore preferable for the shorter times required other than the best accuracy. Furthermore, the weekly granularity for the target variable allows to have a more reliable accuracy for the predictions and the difference with the baseline is more evident in particular for the most distant weeks at vehicle maintenance. As regards the classification, this approach shouldn't be used due to the unbalanced number of class labels('no

maintenance' label is more numerous than 'yes maintenance') and the associated negative results.

Obviously the work done in this thesis should not be considered completely concluded, relative to the topic of predictive maintenance. This means that the results related to machine learning algorithms must be associated with a final product to be delivered to the manufacturers of industrial vehicles supported by Tierra, that has chosen to continue working with **PredictWeeks2Maintenance**. Obviously the analysis presented in this thesis can be extended by taking into consideration new vehicles/models in different scenarios, but above all by adding a greater quantity of data relating to a longer time span.

Bibliography

[1] Federico Perrotta, Tony Parrya and Luis C. Nevesb (2017): 'Application of Machine Learning for Fuel Consumption modelling of Trucks'. University of Nottingham

[2] Weiliang ZENG , Tomio MIWA, Takayuki MORIKAWA (2015): 'Exploring Trip Fuel Consumption by Machine Learning from GPS and CAN Bus Data'. Nagoya University(Japan)

[3] Elnaz Siami-Irdemoosa, Saeid R.Dindarloo (2015): 'Prediction of fuel consumption of mining dump trucks: A neural networks approach'. Missouri University of Science and Technology (USA)

[4] Sandareka Wickramanayake, H.M.N. Dilum Bandara (2016): 'Fuel consumption prediction of fleet vehicles using machine learning'. University of Moratuwa (Sri Lanka)

[5] Dena Markudova, Elena Baralis, Luca Cagliero, Marco Mellia, Luca Vassio, Elvio Amparore, Riccardo Loti, Lucia Salvatori: 'Heterogeneous Industrial Vehicle Usage Predicitons: A Real Case'. Politecnico di Torino (2019)

[6] https://pypi.org/project/workalendar/

[7] https://pypi.org/project/holidays/

[8] https://www.officeholidays.com/countries/

[9] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

[10] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

[11] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html

[12] https://www.csselectronics.com/screen/page/simple-intro-to-can-bus/

[13] https://esd.eu/en/products/can-cloud-gateway

[14] Jason Brownlee (January 2017). How to Decompose Time Series Data into Trend and Seasonality. Retrieved from https://machinelearningmastery.com/decompose-time-series-data-trend-seasonality/

[15] What is preventive maintenance (PM)? Retrieved from https://www.fiixsoftware.com/maintenance-strategies/preventative-maintenance/

[16] Learning Every Function with Machine Learning. Retrieved from https://www.meconferences.com/blog/learning-every-function-with-machine-learning/

[17] Krishni. A Beginners Guide to Random Forest Regression. Retrieved from https://medium.com/datadriveninvestor/random-forest-regression-9871bc9a25eb

[18] Pourya. Time Series Machine Learning Regression Framework. Retrieved from https://towardsdatascience.com/time-series-machine-learning-regression-framework-9ea33929009a

[19] Sanjay.M (November 2018). Why and how to Cross Validate a Model? https://towardsdatascience.com/why-and-how-to-cross-validate-a-model-d6424b45261f

[20] https://scikit-learn.org/stable/

[21] https://www.tierratelematics.com/solutions/

[22] https://en.wikipedia.org/wiki/Arthur_Samuel

[23] Ayush Pant. Introduction to Machine Learning for Beginners. Retrieved from https://towardsdatascience.com/introduction-to-machine-learning-for-beginners-eed6024fdb08

[24] Ayush Pant. Workflow of a Machine Learning project. Retrieved from https://towardsdatascience.com/workflow-of-a-machine-learning-project-ec1dba419b94

[25] Ana Porras Garrido. What is the difference between Bagging and Boosting? Retrieved from https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/

[26] Understanding Support Vector Machine Regression. Retrieved from https://it.mathworks.com/help/stats/understanding-support-vector-machine-regression.html

[27] Georgios Drakos(June 2019). Random Forest Regression model explained in depth. Retrieved from https://gdcoder.com/random-forest-regressor-explained-in-depth/

[28] Christian Schlegel, HMS Industrial Networks (2017): 'The role of CAN in the age of Ethernet and IOT'

[29] Michael Galarnyk: https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51

[30] https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative