POLITECNICO DI TORINO

Department of Control and Computer Engineering

Master's degree program in
Computer Engineering

Master Thesis

Vehicle utilization analysis with data mining

Supervisor

Prof. Luca Cagliero

Candidate

Giulio Bonetto
Student ID: 234455

April 2019

SUMMARY

1. INTRODUCTION 5

2. CONTEXT OF THIS WORK 7

3. INTRODUCTION TO MACHINE LEARNING 9

3.1 CLUSTERING 10

3.2 ASSOCIATION RULE LEARNING 10

3.3 REGRESSION AND CLASSIFICATION 11

4. STATE OF THE ART 17

5. METHODOLOGY OF ANALYSIS 20

5.1 DATA PREPARATION 21

5.2 MODEL GENERATION 28

5.3 MODEL APPLICATION 30

6. EXPERIMENTS 32

6.1 EVALUATION OF PERFORMANCES 32

6.2 SELECTION OF THE TRAINING SET 33

6.3 NEXT-DAY PREDICTIONS 35

6.4 SELECTION OF THE PARAMETERS OF THE ALGORITHMS 38

6.5 ANALYSIS OF SINGLE VEHICLES 42

6.6 NEXT-WORKING-DAY PREDICTIONS 45

6.7 ADDITIONAL FEATURES OF THE MODEL 48

7. CONCLUSIONS 53

8. REFERENCES 54

INDEX OF FIGURES

Figure 1: General schema of methodology of analysis 20
Figure 2: Sample of data aggregated per day. 21
Figure 3: Quantity of information per type of vehicle 22
Figure 4: Quantity of information per model of vehicle 23
Figure 5: Daily utilization hours of vehicles 4193, 3717 and 4272. 24
Figure 6: Daily utilization hours of vehicle 4193 (including zeroes) 25
Figure 7: Distribution of utilization hours for model 1254 26
Figure 8: Distribution of utilization hours for model 1254 (including zeroes) 27
Figure 9: Discretized series, features matrix and target vector 30
Figure 10: Sliding vs. expanding window 31
Figure 11: Confusion matrix 33
Figure 12: Accuracy with different size of the windows 34
Figure 13: Accuracy with sliding vs. expanding window 35
Figure 14: Results in the binary next-day scenario 36
Figure 15: Results in the 3-class next-day scenario 37
Figure 16: Accuracy with different parameters of the linear kernel SVM 38
Figure 17: Accuracy with different parameters of the polynomial SVM 39
Figure 18: Accuracy with different parameters of the RBF SVM 40
Figure 19: Accuracy with different values of the n_estimators of the Random Forest 41
Figure 20: Accuracy with different values of the min_samples_split of the Random Forest .. 41
Figure 21: Binary discrete series for vehicle 4272 42
Figure 22: True and predicted values for vehicle 4272 43
Figure 23: True and predicted values for vehicle with id 3717 44
Figure 24: Vehicles which show frequent 0-1 transitions 44
Figure 25: True and predicted values for vehicle 3853 45
Figure 26: Accuracy with different size of the windows in next-working-day predictions 46
Figure 27: Results in the binary next-working-day scenario 47
Figure 28: Results in the 3-class next-working-day scenario 48
Figure 29: Positions of the vehicles of model 1254 working on December 20th, 2017 49
Figure 30: Correlation ranking of features with utilization hours 50

Figure 31: Training set with one additional feature 51
Figure 32: Results in the binary next-day scenario with additional features 51
Figure 33: Results in the 3-class next-day scenario with additional features 52
Figure 34: Computational time with or without additional features 53

1. INTRODUCTION

Nowadays technology has increased its importance in humans' life, and it has given us the possibility to make everything smarter and simpler. In fact, we are developing devices based on a continuous exchange of data, and this data can be explored and used in some way.

In this thesis we use data supplied by Tierra S.p.A. (a company for IoT solutions), collected from devices installed on construction vehicles.

Our objective is examining the past utilization patterns of these vehicles in order to forecast their future utilization hours. This can be useful because, knowing the future usage, refuel operations, maintenance, budgets, etc., can be scheduled in a better way.

We use machine learning techniques, and since our variables are categorical, our problem can be solved by classification methods.

Our data contain the usage patterns of 2239 different vehicles, collected during a period of more than three years, and compose a time series. Data are aggregated per day and include information such as the number of hours the vehicle is active, the fuel consumed, the distance travelled during the day, etc.

Since during our analysis we notice that the utilization patterns of different vehicles, even if they belong to the same model and type, are very dissimilar to each other, we decide to use a vehicle-per-vehicle approach.

We define two scenarios of work:

- **Next-day** predictions: we forecast the utilization hours on the next calendar day. The original time series contains records only for the days in which a vehicle is working, thus in order to use this scenario, we must add zeroes for all the other days.
- **Next-working-day** predictions: we forecast the utilization hours on the next day in which a vehicle will work (we suppose to know it). In this scenario, we can use the original series.

Then, we should transform our data to have a correct input for a machine learning algorithm.

We need a feature matrix (X), which is mapped to the target vector (y) by the function applied by the classification algorithm. Features are the values of past observations of utilization hours and some additional fields associated to the past days (e.g. the season of the year, the city where the vehicle is working, etc.)

Our task is a classification problem, so the values of utilization hours must be discretized into different classes before being inserted into matrix X and into vector y.

We can identify two different scenarios of discretization:

- **Two classes** (binary): we are interested in predicting whether a vehicle will be used on a specific date, independently on the precise number of hours it will work.
- **Three classes**: in addition to the objective of the previous scenario, we are also interested in knowing if a vehicle will be working for a high number of hours on a given date. This can be useful because high levels of utilization are the most critical in terms of fuel consumption optimization, refuel operations and maintenance scheduling.

After the discretization phase, we define our way of proceeding: at the beginning of the time series, a model is trained by using a certain number of examples (training phase). Then, the model makes a prediction for the next day and this prediction is compared to the real value (testing phase).

Finally, the window is moved on to the next day, and all this procedure is repeated: so, a new model is built and trained for each day. This approach is chosen in order to add new data (collected every day) to our model.

As classification algorithms, we try Naïve Bayes classifiers as a simple solution, Support Vector Machines (SVM) and Multilayer Perceptron (MLP) as more advanced techniques, Random Forests as an ensemble method. Before applying these algorithms, some parameters are configured.

We execute algorithms on all 66 vehicles of one model of the type Refuse Compactor, in the next-day scenario. For each vehicle, results are evaluated by means of the confusion matrix, and then they are averaged on all the vehicles of that model.

By examining the obtained results, we notice that machine learning algorithms do not show significant differences in performance. Anyway, Random Forest seems to be the classifier which performs better among them, with an average accuracy (the number of correct classifications divided by the total number of samples) equal to 89.4% by using a specific setup.

Since from the usage patterns we see that some vehicles present frequent zero-one transitions and long intervals of zero utilization, and predicting can be difficult for classifiers, we decide to introduce the next-working-day scenario. However, even if in this case the problem is easier, results are slightly worse, and we do not observe notable differences in performance.

On the contrary, we see some differences between the two discretization scenarios: binary predictions in general perform better than 3-classes predictions, since the problem is much simpler.

In the end, we try adding some additional features to our problem: we perform a correlation analysis in order to understand which of the information available for a vehicle can be correlated with its utilization hours. So, we obtain that the most correlated attributes are the fuel consumed, the distance travelled, the average latitude and the average longitude during the day.

We add these features to the training set, and we apply again the machine learning algorithms, in both the defined prediction scenarios. However, in this case, the problem becomes more computationally expensive, since the model should handle a much higher number of features, and the improvement in accuracy is not high.

In conclusion, in this thesis we explore the problem of forecasting the future usage of construction vehicles by using classification techniques, and we find some methods which can reach an accuracy above 80% in the binary scenario (in this case a random classifier would reach an accuracy of 50%).

A possible future development of our work can be grouping vehicles based on their utilization patterns and make predictions per group of vehicles, instead of per each single vehicle.

2. CONTEXT OF THIS WORK

As introduced in chapter 1, this thesis is developed as part of a partnership between the Politecnico di Torino and Tierra S.p.A., a company for IoT solutions. Tierra develops tailored solutions for geo-localization, management, maintenance and it offers services of monitoring and control of assets in the fields of Agriculture, Construction and Industry.

In this thesis we focus on the Construction field: on-board devices (provided by Tierra) are placed on construction vehicles and track their activity. The collected data are related to the engine RPM, fuel consumption, state of workload (Idle, Moving, High Workload), location, movement, etc.

Data are sent to a server, processed and then displayed to the owner of the vehicle in real-time through a web interface.

Our analysis is performed on data coming from construction vehicles of companies which are costumers of Tierra.

On-board devices send messages to the server at fixed time-intervals by means of the CAN (Controller Area Network) bus, which is a vehicle bus standard created to allow microcontrollers and devices (nodes) to interact with each other without a host computer.

Nodes are linked to each other by means of a two-wire bus and can send and receive messages, even if not simultaneously.

In high speed CAN (512 kbps), the linear bus is terminated with 120 Ω resistors at each end, in low speed CAN (128 kbps) each node is terminated by a fraction of the overall termination resistance, which is about 100 Ω.

Each node needs a central processing unit (microprocessor), a CAN controller (receiving and sending serial bits from/to the bus) and a transceiver (converting the data stream from the CAN bus to the CAN controller level and vice versa).

The CAN bus is in general adopted in the context of:

- Passenger and construction vehicles (as in our problem)
- Medical equipment
- Building automation
- Aviation and navigation equipment
- Elevators and escalators
- Industrial automation

The transmission system exploits a bitwise arbitration method: the CAN specifications define “dominant” (logical 0) and “recessive” bits (logical 1). The idle state is 1.

If a node sends a dominant bit while another node is sending a recessive bit, a collision happens and the dominant bit wins, so the higher priority message is not delayed.

Bit rates can be up to 1 Mbit/s, if the network length is below 40 meters.

In this thesis, for each vehicle equipped with the CAN bus, available information is:

- Information from **CAN bus**: engine on/off, CAN parameters messages, diagnostic messages, status reports.
- Information from **customers**: asset info, maintenance service.
- Information from **devices**: unit/asset code, timestamp, geographic position, digital inputs report.

We focus on CAN parameters messages, which are generated at a high frequency (up to 100 Hz) and are collected and processed by a controller. Then, a report is sent to the server at fixed time-intervals (10 minutes).

This report is stored in a cloud environment and it includes information about the situation of the vehicle in that moment: Fuel Level, Engine Oil Pressure, Engine Coolant Temperature, Engine Fuel Rate Usage, Engine Speed, Engine Hours, Engine Percent Load, Digging Press, Pump Drive Temp, Oil Tank Temp, etc.

The objective of this thesis is analyzing time series composed by utilization hours of a construction vehicle and predicting future values, by exploiting machine learning algorithms (classification methods, specifically).

Focusing on forecasting the utilization hours of a vehicle can be useful for the owner because, knowing utilization hours, fuel consumption can be quickly computed in order to optimize refuel operations and learning future patterns can help organizing the budgets for the vehicle.

Moreover, maintenance can be scheduled in a better way if we know how much a vehicle will be used and even productivity can increase, since we see if a vehicle works more or less than needed.

3. INTRODUCTION TO MACHINE LEARNING

A possible definition (Samuel, 1959) of machine learning is:

"Field of study that gives computers the ability to learn without being explicitly programmed"

Machine learning algorithms create mathematical models of sample data (training data), which are used to make decisions and predictions.

In the field of machine learning, two major categories can be identified:

- In **unsupervised** learning tasks, a model is created from a set of data which includes only inputs, in order to find patterns in the data and to arrange the inputs into categories.
- In **supervised** learning tasks, the algorithm creates a model from a set of data including both inputs and outputs.

The main types of problems belonging to the category of unsupervised learning are:

- **Clustering**: grouping objects so that the ones in the same group are more related to each other than to the ones in other groups.
- **Association rule learning**: finding relations among variables (such as among products bought together).

On the contrary, the category of supervised learning can be divided into two types of problem:

- **Regression**: evaluating the relationships among data or datasets.
- **Classification**: recognizing to which category an observation belongs.

3.1 CLUSTERING

"Cluster analysis or simply clustering is the process of partitioning a set of data objects (or observations) into subsets." (Han, Kamber, Pei, 2012)

It consists of grouping objects so that the ones in the same group (cluster) are more related to each other than to the ones in other groups.

Clustering is not a single algorithm, but it may be obtained by various algorithms, each one with different definitions of cluster: a cluster can be a group with short distance between members, a dense area, a statistical distribution, etc.

The convenient clustering algorithm and parameter setting depends on the dataset and on the planned use of results.

3.2 ASSOCIATION RULE LEARNING

"Association rules are if-then statements that help to show the probability of relationships between data items within large data sets in various types of databases." (Rouse, 2018)

Association rule learning is a rule-based machine learning technique for finding relations among variables in big databases. Its objective is supporting a machine in extracting features and associations from data.

Association rules have been introduced to find regularities between products purchased in supermarkets, by analyzing transactions recorded by points-of-sale. Identifying these relations can be useful for choices in marketing activities, such as product placements, promotions, etc.

Today, association rules are used also in other application areas (intrusion detection, web usage mining, etc.).

3.3 REGRESSION AND CLASSIFICATION

Regression analysis is a set of methods for evaluating the relationships among variables.

It helps knowing how the value of the dependent variable evolves when one of the independent variables is changed, while the other independent variables are fixed.

Classification is, instead, the task of recognizing to which class an observation belongs, based on a training set of data whose class is known. It is an example of pattern recognition.

The most common algorithms of supervised learning are:

- **Decision trees**: a tree is used as a predictive model to know the target value of an item (represented in the leaves) from the observations about the same item (in the branches).
- **Ensemble methods**: multiple learning algorithms are exploited to get better predictive performance than the one obtained by a single algorithm (e.g. bootstrap aggregating, boosting, random forest).
- **Linear regression**: the relationship between the dependent variable and one or more independent variables is modelled in a linear way.
- **Naïve Bayes classifier**: it is a group of probabilistic classifiers based on Bayes' theorem. These classifiers assume that the features are independent on each other.
- **Artificial neural networks**: they are systems inspired by the biological neural networks which compose animal brains. Such systems learn how to execute tasks by considering samples of data, without being specifically programmed.
- **Support vector machine**: an SVM model represents training examples as points in space, so that the examples in different classes are separated by a gap as large as

possible. Then, new examples are mapped into the same space and their class is predicted based on which side of the gap they are.

In supervised machine learning problems, the dataset is divided into training set and testing set. Given the training set, which contains both the inputs (X) and the outputs (y), the chosen algorithm identifies a function to define the relationship $y = g(X)$.

Later, the same function is applied on the inputs of the testing set and results are compared to the real values of the outputs.

We will discuss more in detail our way of proceeding in next chapters.

The main difference between regression and classification problems is given by the nature of data: in regression, variables to predict are continuous, while in classification variables are categorical, and the dataset is labelled with different classes.

Our problem is specifically a classification one: the dataset is divided into classes and our target variable is itself a class.

Now we explain the classification algorithms applied to predict our variables.

These algorithms, in general, have some parameters to be configured, we will discuss their tuning in next chapters.

Naïve Bayes

As first algorithms, we try Naïve Bayes classifiers.

"A Naïve Bayes classifier is a probabilistic machine learning model that's used for classification task. The crux of the classifier is based on the Bayes theorem." (Gandhi, 2018)

Naïve Bayes classifiers assume features are strongly (naïve) independent of each other.

They were initially introduced into the text retrieval community in 1960s, and nowadays they are a common technique for text categorization, with word frequencies as the features, and for medical diagnosis. In these fields they can compete with Support Vector Machines classifiers, even if they are less advanced methods.

Naïve Bayes classifiers are scalable and need several parameters, whose number is linear in the quantity of features in the learning problem.

There is not a unique algorithm for training Naïve Bayes classifiers, but all of them assume the value of a specific feature is not dependent of the value of any other, knowing the class variable (class-conditional independence).

Bayes' theorem is stated mathematically as:

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}$$

where:

- $P(H|X)$ is the posterior probability of H conditioned on X.
- $P(X|H)$ is the posterior probability of X conditioned on H.
- $P(H)$ is the prior probability of H.
- $P(X)$ is the prior probability of X.

Assuming to have a set of tuples, where each tuple $X = (x_1, x_2, \ldots, x_n)$ represents n observations made from n attributes, and m classes $C_1, C_2, \ldots, C_m$, a Naïve Bayes classifier will predict that X belongs to the class with the highest posterior probability conditioned on X.

X belongs to class C_i if and only if $P(C_i|X) > P(C_j|X)$, for $1 \leq j \leq m, j \neq i$.

So, we must maximize $P(C_i|X)$, which, by Bayes' Theorem is equal to:

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}$$

$P(X)$ is constant and we can assume the classes are equally likely, i.e. $P(C_1) = P(C_2) = \cdots = P(C_m)$, thus we maximize $P(X|C_i)$. By using the class-conditional independence assumption we obtain:

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \cdot P(x_2|C_i) \ldots P(x_n|C_i)$$

The probabilities $P(x_1|C_i), P(x_2|C_i), \ldots, P(x_n|C_i)$ can be easily estimated from the training tuples, since if attribute A_k is categorical, then $P(x_k|C_i)$ is the number of tuples of class C_i having the value x_k for A_k divided by the total number of tuples of class C_i.

In conclusions, the predicted class label is the class C_i for which $P(X|C_i)P(C_i)$ is the maximum.

Training of Naïve Bayes classifiers can be performed quite efficiently in a supervised learning scenario.

For discrete features, multinomial and Bernoulli distributions of features are common.

Support Vector Machines

As more advanced algorithms, we try Support Vector Machines (SVM), by using the implementation in scikit-learn (Python).

"A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two-dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side." (Patel, 2017)

An SVM model represents training examples as points in space, so that the examples in different classes are separated by a gap as large as possible. Then, new examples are mapped into the same space and their class is predicted based on which side of the gap they are.

A good separation is reached by the hyperplane with the largest distance to the nearest training point of any class (margin).

A separating hyperplane can be written as:

$$W \cdot X + b = 0$$

where:

- W is a weight vector, $W = \{w_1, w_2, \dots, w_n\}$, and n is the number of attributes.
- b is a scalar (bias).

If we have two input attributes A_1 and A_2, training tuples are bidimensional, i.e. $X = (x_1, x_2)$), and by considering b as an additional weight w_0, our hyperplane becomes:

$$w_0 + w_1 x_1 + w_2 x_2 = 0$$

Any point lying above the hyperplane satisfies $w_0 + w_1 x_1 + w_2 x_2 > 0$, while any point lying below it satisfies $w_0 + w_1 x_1 + w_2 x_2 < 0$.

Thus, we obtain these two hyperplanes, defining the sides of the margin:

$$H_1: w_0 + w_1 x_1 + w_2 x_2 \geq 1 \text{ for } y_i = +1$$

$$H_2: w_0 + w_1 x_1 + w_2 x_2 \leq -1 \text{ for } y_i = -1$$

Any tuple falling above H_1 belongs to class +1, any tuple falling below H_2 belongs to class -1.

The tuples that fall on hyperplanes H_1 and H_2 are called support vectors.

Combining the previous two inequalities, we get $y_i(w_0 + w_1 x_1 + w_2 x_2) \geq 1, \forall i$.

This equation can be rewritten as a constrained quadratic optimization problem and solved by using KKT conditions. Once we have found the maximum-margin hyperplane, we have a trained support vector machine.

By using the Lagrangian formulation, the maximum-margin hyperplane can be rewritten as decision boundary:

$$d(X^T) = \sum_{i=1}^{l} y_i \alpha_i X_i X^T + b_0$$

where y_i is the class label of support vector X_i, X^T is a test tuple, α_i and b_0 are numeric parameters and l is the number of support vectors.

The sign of the result tells us the size of the hyperplane on which the test tuple falls: if it positive, then X^T falls above the maximum-margin hyperplane and the class prediction is +1, if it is negative, then X^T falls below and belongs to class -1.

By applying the kernel trick to maximum-margin hyperplanes, a non-linear classifier can be created: each dot product is substituted by a non-linear kernel function. Examples of non-linear functions that we use are the polynomial and the RBF kernels.

The efficiency of SVM depends on some parameters, such as the kernel's parameters, and soft margin parameter C, which can be chosen by a grid search.

Multilayer Perceptron

A multilayer perceptron (MLP) is a category of artificial neural network.

Artificial neural networks are systems inspired by the biological neural networks which compose animal brains. These systems are based on a set of connected units or nodes (artificial

neurons): each connection can send a signal (commonly, a real number) from one artificial neuron to the other.

An MLP comprises, at least, three layers of units: an input layer, a hidden layer and an output layer. Connections (edges) and artificial neurons have a weight that increases or decreases the strength of the signal.

The inputs to the network are the attributes measured for each training tuple and are fed at the same time into the units composing the input layer. Then, these inputs are weighted and fed simultaneously to a second layer, called hidden layer, whose units are called neurodes.

The hidden layer can be more than one and the outputs of the first hidden layer can be used as inputs for another one, and so on.

The weighted outputs of the last hidden layer are used as inputs for the units of the output layer, which emits the predictions for tuples.

Weights do not cycle back to a previous layer, so this is a category of feed-forward network. Each unit provides input to each unit of the next layer.

In the perceptron, learning takes place by modifying connection weights after processing each piece of data, based on the quantity of error in the output compared to the expected result.

The method used for training is named backpropagation and consists of the computation of a gradient needed to compute the weights in the network.

Each unit (except for the input units) applies a non-linear activation function, which is a function that maps the weighted input to the output produced by each neuron.

Random forest

Random forests are an ensemble learning method. They work by building a certain number of decision trees and producing the mode of the classes predicted by the individual trees.

"Random forest is a supervised learning algorithm. Like you can already see from its name, it creates a forest and makes it somehow random. The forest it builds, is an ensemble of decision trees, most of the time trained with the bagging method. The general idea of the bagging method is that a combination of learning models increases the overall result." (Donges, 2018)

In this thesis we use the implementation in Python scikit-learn.

Decision trees are a common method to solve different machine learning problems, but if they are very deep, they can overfit training sets and learn irregular patterns.

Random forests are a method to average different decision trees, trained on separated parts of the same training set, in order to decrease the variance. The disadvantages of this model are, however, a slight increase in the bias and loss of interpretability.

Random forests can be created by using random attribute selection: supposing to have a training set D, for each iteration i, where $i = 1, 2, \ldots, k$, a training set D_i, of d tuples, is sampled with replacement from D. So, each tuple can occur more than once in D_i, while other tuples can be excluded.

In order to build a decision tree classifier, F attributes, where F is the number of attributes to be used to determine the split, are selected at each node as candidate for the split at the node.

Then, trees are grown to the maximum size and not pruned.

Some parameters to be configured in the Random Forest classifier are:

- **criterion**: the function to determine the quality of a split of a tree (e.g. Gini impurity, information gain).
- **n_estimators**: the number of trees in the forest.
- **max_depth**: the maximum depth of a tree (it can be unlimited).

A useful attribute of the Random Forest classifier is **feature_importances_**, which returns the importance assigned to each feature in the final prediction.

This will be exploited to perform a correlation analysis in order to select the additional features to add.

4. STATE OF THE ART

In literature, a lot of research has been done for vehicles, even if most scientific papers are related to passenger vehicles, rather than to construction ones.

Some recent papers, which use data mining techniques, are cited here.

- **"Application of Machine Learning for Fuel Consumption Modelling of Trucks" (University of Nottingham, 2017)**

  This paper presents the application of three Machine Learning techniques to fuel consumption modelling of articulated trucks for a large dataset.

  Analyzed data: Information from sensors installed on trucks which deals with the vehicle weight, the vehicle speed, the average acceleration, the geographical position, the revolutions of the engine, the activation of cruise control, the travelled distance and the fuel used.

  Used techniques: Support Vector Machine (SVM), Random Forest (RF), and Artificial Neural Network (ANN) models.

  Main results: 14 of 56 variables initially available show significant correlation with fuel consumption. These are: the vehicle weight, the road gradient, the vehicle speed, the average acceleration, the engine rpms at the start of the record, the used gear, the cruise control, the absolute value of the radius of curvature of the road.

  Differences with our work: Our sensors are installed on construction vehicles and collect information on the engine, on the timestamp and on the geographical position.

- **"Exploring Trip Fuel Consumption by Machine Learning from GPS and CAN bus Data" (Nagoya University, 2015)**

  This study aims to explore the trip fuel consumption from a large-scale dataset.

  Analyzed data: Large-scale GPS and CAN (Controller Area Network) bus data provided by 153 probe vehicles during the period of one month.

  Used techniques: Support Vector Machine (SVM), Linear Regression (LR), Artificial Neural Network (ANN) models.

  Main results: Experimental results show that the proposed SVM model, considering the feature of the whole trip, is more accurate than that of the SVM model based on the summation of link fuel consumption.

  Differences with our work: Instead of on passenger vehicles, our analysis focuses on construction vehicles, equipped with sensors which collect information during a period of three years.

- **"Prediction of fuel consumption of mining dump trucks: A neural networks approach" (Missouri University of Science and Technology, 2015)**

  In this paper, fuel consumption of mining dump trucks per cycle of operation is predicted using an Artificial Neural Networks (ANN) technique.

  Analyzed data: Context data related to pay load, loading time, idled while loaded, loaded travel time, empty travel time, idled while empty

  Used techniques: Artificial Neural Network (ANN) model.

  Main results: The results reveal that the trucks empty idle time (ES) is a major contributor to unnecessary fuel consumption. Considering both the production (profit) loss and the unnecessary energy consumption due to ES, the importance of implementing proper remedies is obvious.

  Differences with our work: Our data, collected by sensors through a CAN bus, is mainly related to the engine usage and to the geographical position.

- **"Fuel Consumption Models Applied to Automobiles Using Real-Time Data" (Istanbul Technical University, 2016)**

  In this paper, three statistical models are used in term of prediction of total and instant fuel consumption.

  Analyzed data: Data collected in real-time from three different passenger vehicles on three routes by casual drive, using a mobile phone application.

  Used techniques: Artificial Neural Network (ANN), Support Vector Machine (SVM), Multiple Linear Regression (MLR) models.

  Main results: Speed, Acceleration, Engine RPM, Volumetric Efficiency, EGR Commanded and Slope of the road are used as input variables. When all the variables are used in analysis, the correlation between real data and predicted data is quite high.

  Differences with our work: We collect data by using a CAN bus system on construction vehicles, during a period of three years.

- **"Fuel Consumption Prediction of Fleet Vehicles Using Machine Learning" (University of Moratuwa, 2016)**

  In this paper, three ML techniques are compared to predict the fuel consumption of a bus, given all available parameters as a time series.

Analyzed data: The dataset corresponds to a long distance, public bus in Sri Lanka. The bus starts from Depot around 4:00pm and then goes to Colombo (i.e., commercial capital).

Used techniques: Random Forest (RF), Gradient Boosting (GB), Artificial Neural Network (ANN) models.

Main results: the RF model could predict the fuel consumption more accurately while capturing the trends in data. Such a model is useful detection of fuel fraud where the actual consumption of the vehicle can be compared against the predicted value based on other parameters like distance, location, elevation, speed, and day of the week.

Differences with our work: Our dataset is not composed by passenger vehicles data, but by information collected in three years by sensors which are installed on construction vehicles.

As we can see from these papers, researches on predicting utilization hours of construction vehicles have not been conducted.

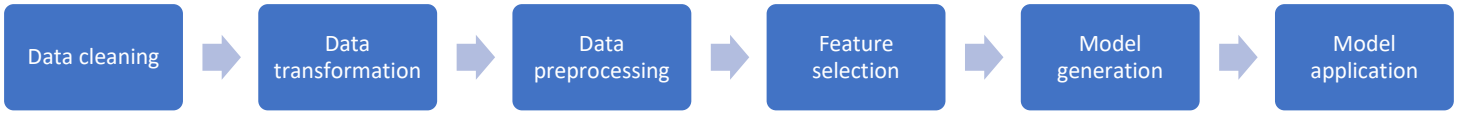
The used techniques are based on machine learning algorithms, even if they differ from the approach analyzed in this thesis because they are mainly regression methods, rather than classification methods.

5. METHODOLOGY OF ANALYSIS

In this chapter we will explain how to adapt the problem to be solved by machine learning algorithms, by transforming our data to a suitable form of input and by generating and applying models.

On figure 1 we can see a schema of this process.

Figure 1: General schema of methodology of analysis



5.1 DATA PREPARATION

As introduced in chapter 2, CAN-messages data consist of a series of reports sent at a fixed time interval to the server.

Each report contains a descriptor (SPN), which tells us the type of information (Fuel Level, Engine Oil Pressure, etc.). We are interested in SPN 183 (Engine Fuel Rate Usage), corresponding to the fuel consumed during the time interval considered in the report.

Since we know this time interval, we can easily compute the number of hours that the vehicle is active and, by summing up and applying SQL queries, we can aggregate data per day, obtaining a table which includes 601462 rows, each one for a single day that a vehicle has worked.

A short sample of this table is presented on figure 2.

Figure 2: Sample of data aggregated per day.

Unit type name	Unit model id	Unit id	Day	On time [h]	Fuel consumption [l]	Latitude	Longitude	Travelled distance [m]
Refuse Compactor	1254	3717	2015-01-02	1.79	34.51	48.11	17.2	0
Refuse Compactor	1254	3717	2015-01-07	3.69	79.75	48.11	17.2	0
…								
Refuse Compactor	1254	4272	2015-02-10	0.32	5.19	50.21	7.56	0
…								
Refuse Compactor	1288	5199	2016-07-15	0.37	7.64	50.21	7.56	0
…								
Tandem Roller	1265	4343	2015-03-31	2.64	6.44	50.21	7.56	723

The first column (Unit type name) is the type of the construction vehicle: 10 different types are present in the table (Refuse Compactor, Single Drum Roller, Paver, Recycler, Tandem Roller, Cold planner, Grader, Rubber Wheel Roller, Material Feeder, Coring Machine).

The second column (Unit model id) represents the model of the vehicle. For each type there are several models, e.g. 44 distinct models of Refuse Compactor, 43 models of Tandem Roller, etc.

For each model, different vehicles are present in the table, each one identified by an id (third column): there are 66 vehicles for model 1254 of type Refuse Compactor, for example.

The fourth column is the day in which the vehicle has worked a certain number of hours (fifth column). In our analysis, we predict this number of hours, which is also the field we use as historical data.

The sixth column represents the number of liters of fuel consumed, while the seventh and the eighth columns are the average position of the vehicle, expressed in latitude and longitude, during the day.

Finally, the ninth field is the distance travelled (expressed in meters) during the day.

On figure 3 we show the quantity of information we have for each type of vehicle.

Figure 3: Quantity of information per type of vehicle

Unit type name	No. vehicles	Total no. records	No. records per vehicle
Single Drum Roller	646	182620	282
Refuse Compactor	311	153240	492
Tandem Roller	412	99524	241
Cold planner	522	95216	182
Paver	206	43710	212
Recycler	114	20841	182
Grader	8	2454	306
Rubber Wheel Roller	10	1970	197
Material Feeder	8	1788	223
Coring Machine	2	99	49

The unit type with most records overall is the Single Drum Roller. However, we are more interested in the number of records (days) per each vehicle, since we make predictions vehicle-per-vehicle, and the unit type with most records per vehicle is the Refuse Compactor, with 492 days of utilization per each vehicle.

But we can also analyze the quantity of data for each specific model of vehicle, shown on figure 4 for the type Refuse Compactor.

In this figure rows are sorted by the total number of records and we see the top-ten models.

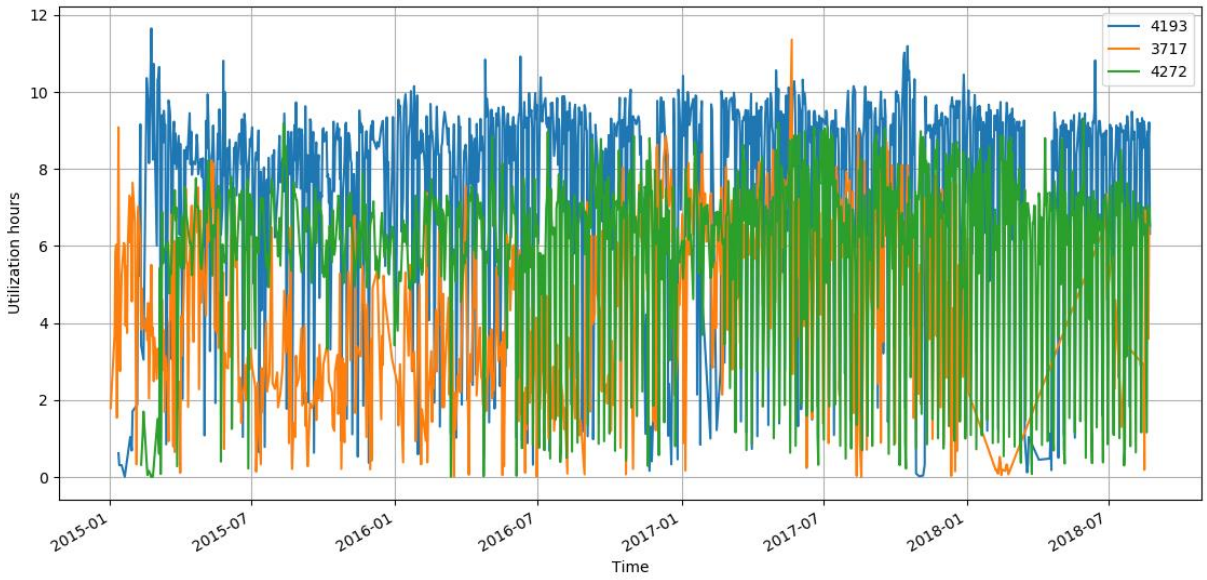
The model with most records is 1254, which is also the model with the highest number of vehicles (66). It is not the model with most records per vehicle, as we can notice from the fourth column of the table, but it has a good trade-off among the three columns.

Figure 4: Quantity of information per model of vehicle

Unit model id	No. vehicles	Total no. records	No. records per vehicle
1254	66	38001	575
1288	29	15319	528
1289	23	12946	562
1316	16	9960	622
1269	9	8355	928
1331	15	7409	493
1250	12	7215	601
1195	13	6885	529
1116	7	5625	803
1280	17	3998	235

Going more in detail with vehicles of model 1254, on figure 5 we can see the daily utilization pattern of three vehicles of this model.

Figure 5: Daily utilization hours of vehicles 4193, 3717 and 4272.



We notice that vehicles have different behaviors: vehicles 3717 and 4193 are added to the dataset in January 2015, while vehicle 4272 in February of the same year.

When a vehicle starts collecting data, there is a brief period in which it is working only for few hours, because devices are tested in a safe environment before being installed on the real vehicle. Moreover, some vehicles are not working for a certain period, such as vehicle 3717, which has not collected data from February 2018 to June 2018.

The utilization pattern of these vehicles is quite constant during the year, as it is common for the type Refuse Compactor. Only vehicle 3717 shows some significant variations, since in 2017 it is used more than in previous years.

No clear seasonality nor utilization trend can be identified from these plots.

These time series are obtained from the values of utilization hours measured on the days a vehicle is working, but we do not have information on the days in which a vehicle is not working.

Thus, these series are not equally spaced, and we should apply some extrapolation in order to add the days not present in the dataset. This extrapolation consists in adding zeroes for the utilization hours of the days in which the vehicle is not working.

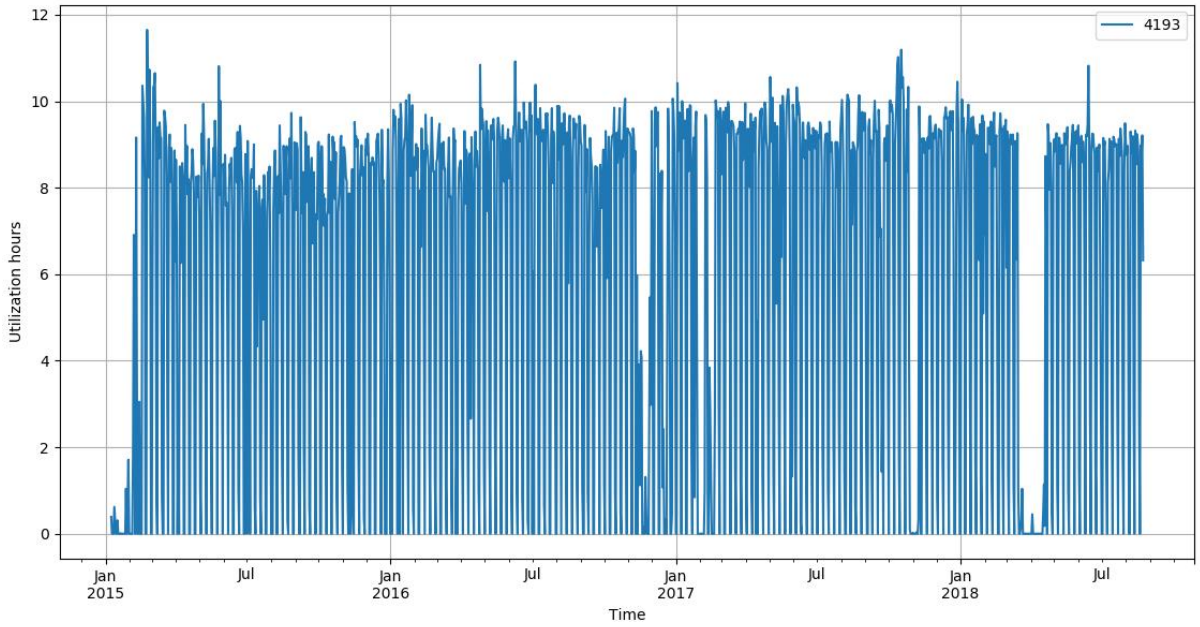
Since not all vehicles have started collecting data at the beginning of the period (December 2014), the number of points of the series is still not equal for all of them.

The equally spaced time series of vehicle 4193 is shown on figure 6.

We can observe that the number of zeroes is high and that most values are equal to zero or greater than eight. So, when this vehicle is used, the utilization pattern is regular, but the days in which it is not active are a lot.

By analyzing all vehicles of model 1254 we can notice that, in these series, zeroes are about the 31% of the total number of values.

Figure 6: Daily utilization hours of vehicle 4193 (including zeroes)



For what concerns utilization patterns, in our analysis we observe that each vehicle is in some way unique. Even the division of vehicles in types and models is not very relevant for the way a vehicle will be used.

For this reason, we decide to handle each vehicle independently to solve our problem. The usage of a specific vehicle is forecast by using only the previous data of the same vehicle.

Since the type of vehicle which has the highest number of records for each vehicle is the Refuse Compactor, all the experiments in this work are performed on it.

We also distinguish two major scenarios, one with equally spaced time series (next day) and one with unequally spaced series (next working day).

In our dataset, the utilization hours of vehicles are continuous values, but as inputs for our classification problem we need to have categorical values, thus we must divide them into different classes.

A possible method of discretization uses percentiles: we can define a percentile as a measure that indicates the value below which a certain percentage of observations falls.

For example, if a value is the 75th percentile, it means that 75% of our observations are smaller than it.

So, by using percentiles we can define the classes into which our values of utilization hours will be sub-divided, and we can identify two different scenarios:

- **Two classes** (binary): we are interested in predicting whether a vehicle will be used on a specific date, independently on the precise number of hours it will work.
- **Three classes**: in addition to the objective of the previous scenario, we are also interested in knowing if a vehicle will be working for a high number of hours (more than the 75th percentile) on a given date. This can be useful because the high levels of utilization are the most critical in terms of fuel consumption optimization, refuel operations and maintenance scheduling.

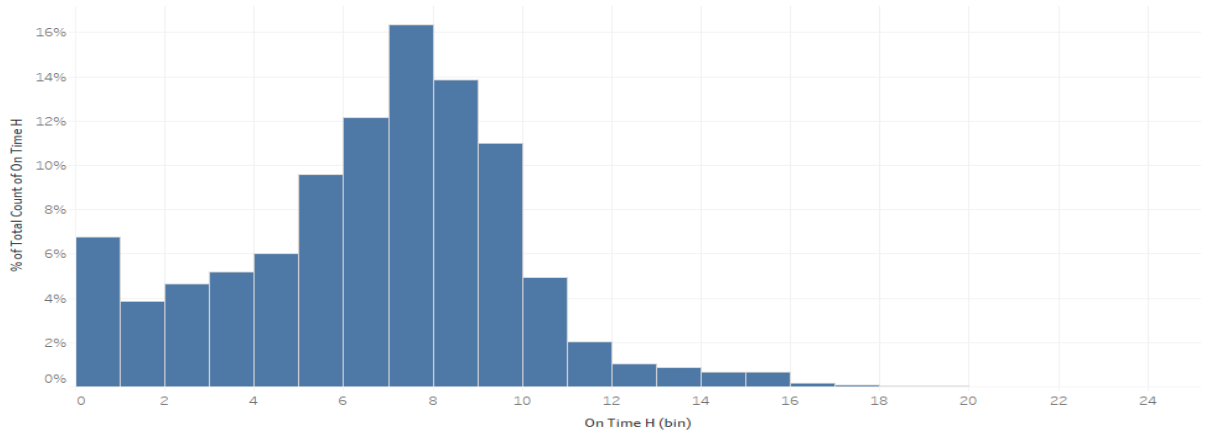
In this work the computation of percentiles and the division into classes is made per model of vehicle, i.e. classes are valid for all the vehicles of that model and percentiles are based on the utilization hours of the same model.

Figure 7 shows the distribution of utilization hours in the original time series (without zeroes) for model 1254 of the Refuse Compactor.

We can notice that most values are centered around eight hours of usage (about 16% of them are between seven and eight hours).

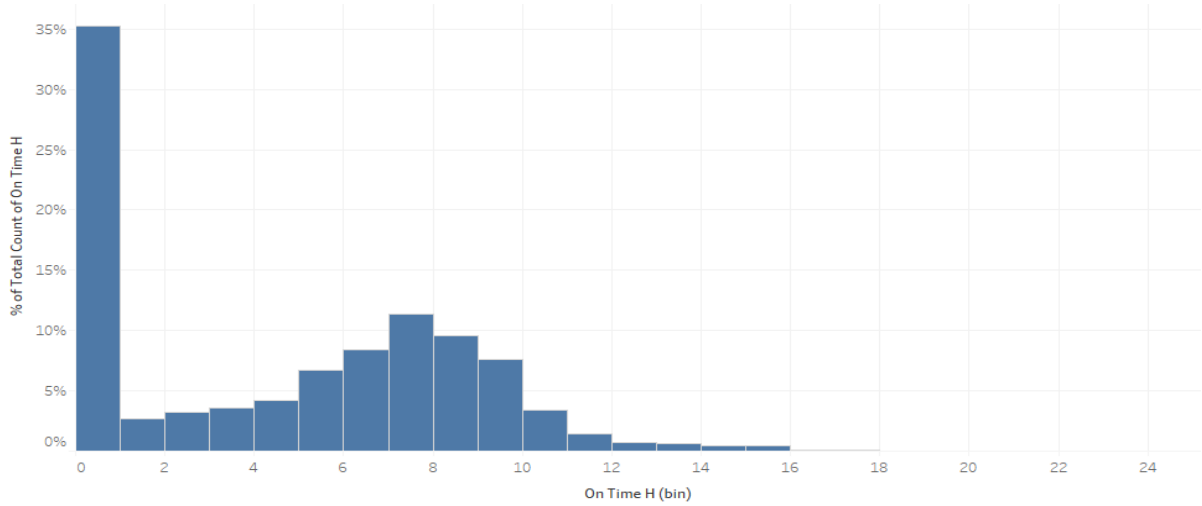
Indeed, we obtain that the 25th percentile is equal to 4.79 hours and that the 75th percentile is equal to 8.78 hours.

Figure 7: Distribution of utilization hours for model 1254



On the contrary, if we compute the 75th percentile on the time series containing zeroes, we obtain a value of 7.95, which is lower due to the very high number of zeroes (about 35% of values are smaller than one) in the series, as we can see on figure 8.

Figure 8: Distribution of utilization hours for model 1254 (including zeroes)



Until now, we have considered as features only the past values of utilization hours, but in order to have a different perspective on the problem, we can insert other features into our model.

These features are pieces of information associated to the past days and, in some cases, they are directly available in the dataset:

- **fuel consumption** and **travelled distance**: directly available for the days in which the vehicle is working, they are set equal to zero for the other days.
- **longitude** and **latitude**: directly available for the days in which the vehicle is working. For the other days we use the last values registered.
- **location**: the nearest city and the country where the vehicle is working. It is computed from the values of latitude and longitude.
- **day of the week**: computed from the date.
- **season of the year**: computed from the date.
- **working day/holiday**: computed from the date, knowing the country where the vehicle is working.

So, we can perform a correlation analysis, to understand which features, among these ones, are the most correlated with the utilization hours of the vehicle. We will explore the results of this analysis in the next chapter.

After recognizing the most correlated features, we can add them to the model.

5.2 MODEL GENERATION

Our problem consists of predicting the usage pattern of a vehicle for the next time instant (next day/next working day), knowing the utilization hours of the same vehicle in the past.

This can be a mathematical representation of the problem:

$$y = g(x_{t-1}, x_{t-2} \dots x_{t-n}, f_1, f_2 \dots f_m)$$

Where:

- y is our target variable (the prediction).
- $x_{t-1}, x_{t-2} \dots x_{t-n}$ are past observations, where n is the size of the window of observation and t is the current time instant.
- $f_1, f_2 \dots f_m$ are the supplementary features which can be used to reach better results, and m is the number of these features.

As introduced in the previous section, we choose to solve our problem in two different main scenarios:

- **Next day:** x is an equally spaced time series (zeroes are added for the days in which the vehicle is not working). $y = x_t$ is the next calendar day.
- **Next working day:** x is not an equally spaced time series and it contains only the days in which the vehicle is working, i.e. the ones present in the dataset. $y = x_t$ is the next day the vehicle will work, supposing to know it.

Features f are pieces of information associated to x: the season of the year, if it is a working day or a holiday, the city and the country where the vehicle is working, etc.

Finally, function g depends on the algorithm we will use for predictions and it tries guessing y by using the values of x and f.

An important parameter which must be set is the horizon: it is the number of time instants between the past data and the data to be forecast. Since we want to predict the utilization hours of the day which is immediately following (next day/next working day) in the time series, we set this value to one.

On the contrary, the size of the observation window will be discussed later.

Combining these two scenarios with the ones already defined in the previous section for discretization, we obtain the four cases we will analyze in this thesis:

- **Binary class next-day predictions**: our time series contains zeroes, so we can define one class including zeroes and one including values greater than zero.
- **3-class next-day predictions**: in addition to the binary case, we can add one class containing high values of usage, greater than the 75th percentile.
- **Binary class next-working-day predictions**: since in this type of predictions the time series does not include the dates on which vehicles are not working, we can define one class containing low values of usage (smaller than the 25th percentile) and one containing all the other values.
- **3-class next-working-day predictions**: again, in this scenario we can add one class including high values of usage to the two classes in the binary case.

In machine learning, the input is a matrix (X) and the output is a vector (y).

Matrix X has columns called features, which are the attributes that influence our prediction: these attributes are the past values of utilization hours (x_i) and the additional features (f_i).

On figure 9, we can see a short sample of the discretized series, of this matrix and of this vector: each row of matrix X contains the values in the observation window (its size is three in this example).

The first three values of the time series (highlighted in green) form the first row of the matrix, while the fourth value (highlighted in blue) is the target and the first element of vector y. So, for each day t, we predict its utilization level by using the values in t-1 (the day before), t-2 (the day before t-1) and t-3 (the day before t-2).

Figure 9: Discretized series, features matrix and target vector

Index	Utilization level
0	0
1	0
2	1
3	1
4	0
5	2
6	0
7	0
8	0
9	1

Index	x(t-3)	x(t-2)	x(t-1)
3	0	0	1
4	0	1	1
5	1	1	0
6	1	0	2
7	0	2	0
8	2	0	0
9	0	0	0

x(t)
1
0
2
0
0
0
1

5.3 MODEL APPLICATION

In common machine learning tasks, the dataset may be directly divided in a random way, for example by using 70% of values for the training set and the remaining 30% for the testing set.

But, in our problem, we cannot proceed in this way, because data is time-dependent and selecting values randomly would cause us to lose this dependency.

So, we decide to create a new model for each day: at the beginning of the time series, a certain number of examples is used for the training set. Then, the model makes a prediction for the next day, which is the testing set, and we compare it to the real value.

With this technique, we do not build one single model per vehicle, but many models, each one valid for a day, and we find the way to choose a correct model for the next day.

The disadvantage of this method is, obviously, the increase in computational time due to the creation of all these models.

We define two different window sizes:

- **Training window size (T)**: equal to the number of the past days used for training (ex. size=90).
- **Features window size (F)**: equal to the number of features, including additional features (ex. size=7). It is always smaller than T.

In the training set, the input matrix (*X_train*) has size TxF, while the output vector (*y_train*) has size Tx1.

In the testing set, the input vector (*x_test*) has size 1xF, while *y_test* is a single value.

The number of created models is, instead, N – T – F, where N is the total number of days available for that vehicle, since we need to have one complete training window and one complete features window before being able to build the first model of the series.

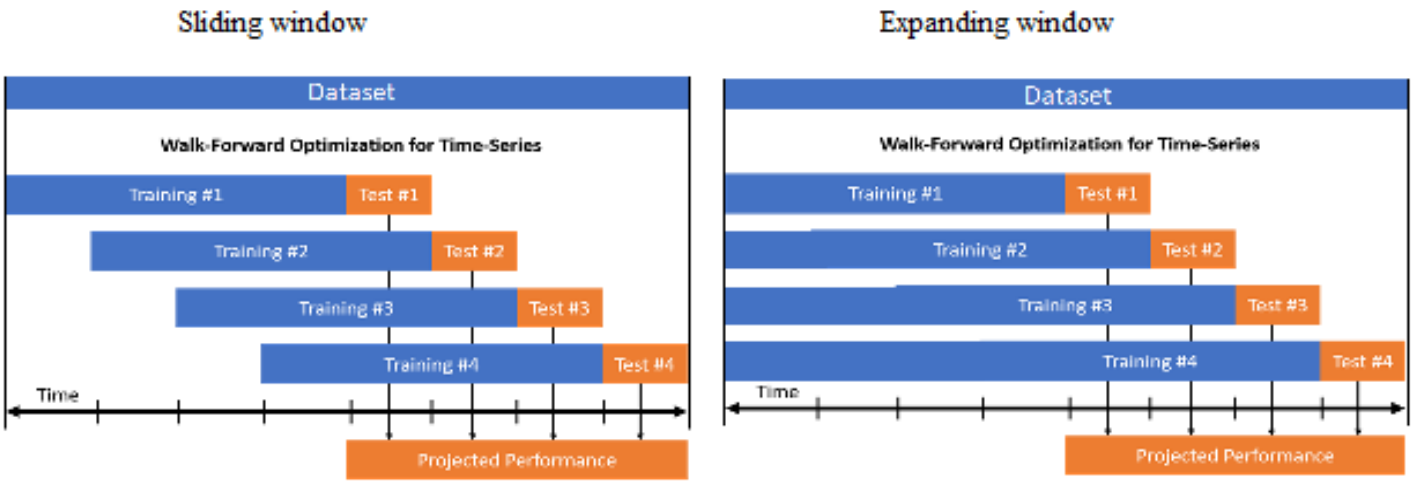
The methods used to combine the results of all obtained models will be explained in detail in the next chapters.

Moving ahead in time, the quantity of our data increases each day, so we must decide how to create the training window. Two possible methods are:

- **Sliding window**: the size of the training window is fixed, and the window contains only the data in the recent past, not considering what happens before. This method limits the size of the training set, but it is less computationally expensive.
- **Expanding window**: the size of the training window increases while we are moving ahead in time. T grows each day and the training examples are many more, the computational time can be an issue in this case.
  Furthermore, we do not know if the values in a far past can be much correlated with the present.

On figure 10 we can see the two different methods, and we can notice the differences in building the training and the testing sets.

Figure 10: Sliding vs. expanding window



Finally, we must choose the features window size, the training window size and the type of algorithm we will use, among the ones introduced in section 3.3.

6. EXPERIMENTS

In this chapter, we will describe the experiments conducted by using the data and the methodologies explained in the previous chapters.

In the first scenario (next-day predictions), we add zeroes for the days not present in the dataset, as introduced in section 5.1, while in the second scenario (next-working-day predictions) we use the original time series. For both scenarios, we run experiments in the both the binary and the 3-class cases.

Before presenting the results, we will introduce the methodology used to evaluate the obtained results, and we will explain the configuration of some general parameters (size of the windows, parameters of the algorithms, etc.) and the choice of the methodology of validation (sliding vs. expanding window).

We run experiments on an AMD A8-7410 CPU and 8 GB of RAM, running Windows 10 64-bits.

6.1 EVALUATION OF PERFORMANCES

In order to know how precise our analysis is, we should exploit some metrics to determine the performance of our models.

To assess the quality of the prediction process, we compute the confusion matrix, which is a table that allows visualizing the results of algorithms.

"The confusion matrix is a useful tool for analyzing how well your classifier can recognize tuples of different classes. TP and TN tell us when the classifier is getting things right, while FP and FN tell us when the classifier is getting things wrong." (Han, Kamber, Pei, 2012)

Each row shows the instances in a predicted class, and each column shows the instances in an actual class. An example of confusion matrix (with only two classes) is shown on figure 11.

Figure 11: Confusion matrix

		ACTUAL CLASS	
		NEGATIVE	POSITIVE
PREDICTED CLASS	NEGATIVE	True negative (TN)	False negative (FN)
	POSITIVE	False positive (FP)	True positive (TP)

From this table we can deduce the accuracy of the model, defined as the number of correct classifications: $\frac{TP+TN}{TP+TN+FP+FN}$.

But accuracy is not a completely trustworthy metric, since if classes are not balanced (i.e. if the number of elements in different classes is very variable), it can give misleading results.

So, for each class, we consider precision ($\frac{TP}{TP+FP}$) and recall ($\frac{TP}{TP+FN}$), in order to understand better the behavior of the classifier.

If, for example, we had two classes, one containing the 90% of values and the other one containing the remaining 10%, an algorithm could classify all the values as belonging to the first class. In this case, the total accuracy would be 90%, but the recall for the second class would be 0%.

We consider the confusion matrix for each vehicle, and then the values of accuracy, precision and recall are average on all the analyzed vehicles.

6.2 SELECTION OF THE TRAINING SET

Before applying the classification algorithms, we must configure some parameters, such as:

- The size of the features window.
- The size of the training window.
- The type of the training window (sliding vs. expanding).

In this section we perform a grid search using the Naïve Bayes classifier on 10 example vehicles of model 1254 (3959, 3988, 4256, 4551, 4700, 4891, 5003, 5229, 5504, 6034), in order to find the best parameter setting.

The size of the training window tells us the number of the examples used for training and corresponds to the size of the sliding window, while the size of the features window is the number of the past days used as features.

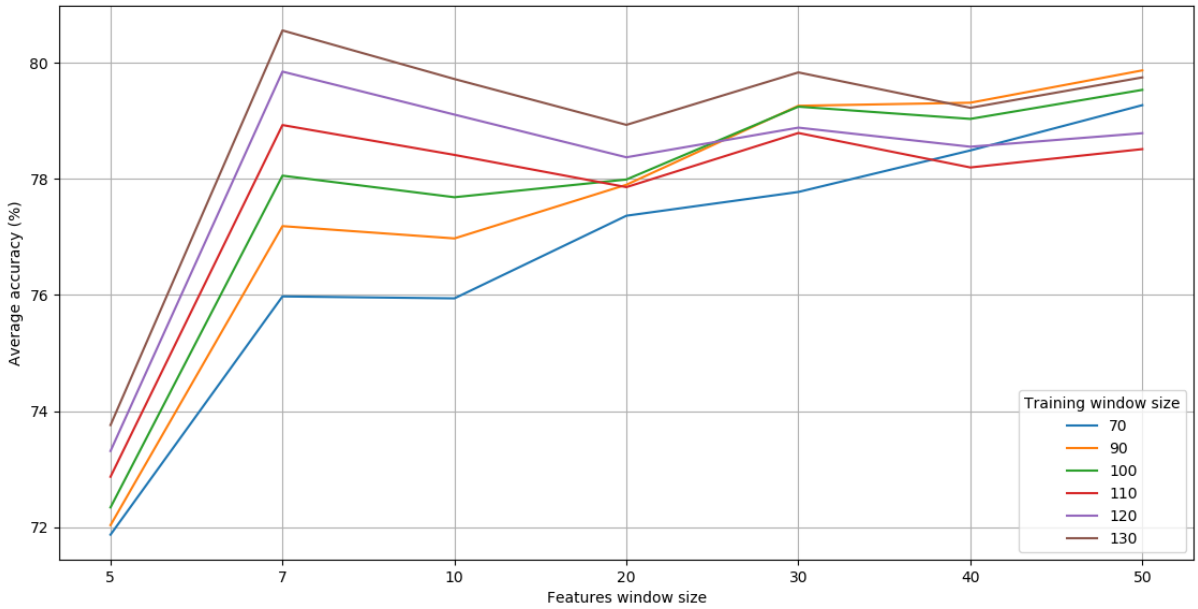
On figure 12 we can observe the average accuracy obtained with different sizes of the features window and of the training window, in the binary next-day scenario.

The values of the features window are on the x-axis, while the values of the training window are shown by the various curves.

We notice different behaviors: in general, larger sizes of the features window give better results, but however the best accuracy overall is obtained when the features window size is 7 and when the training window size is 130.

The best performances are reached with larger sizes of the windows, because large windows consider the correlation of a greater number of past values of utilization hours with the value to predict. On the contrary, if windows are small, a lot of values are ignored.

Figure 12: Accuracy with different size of the windows

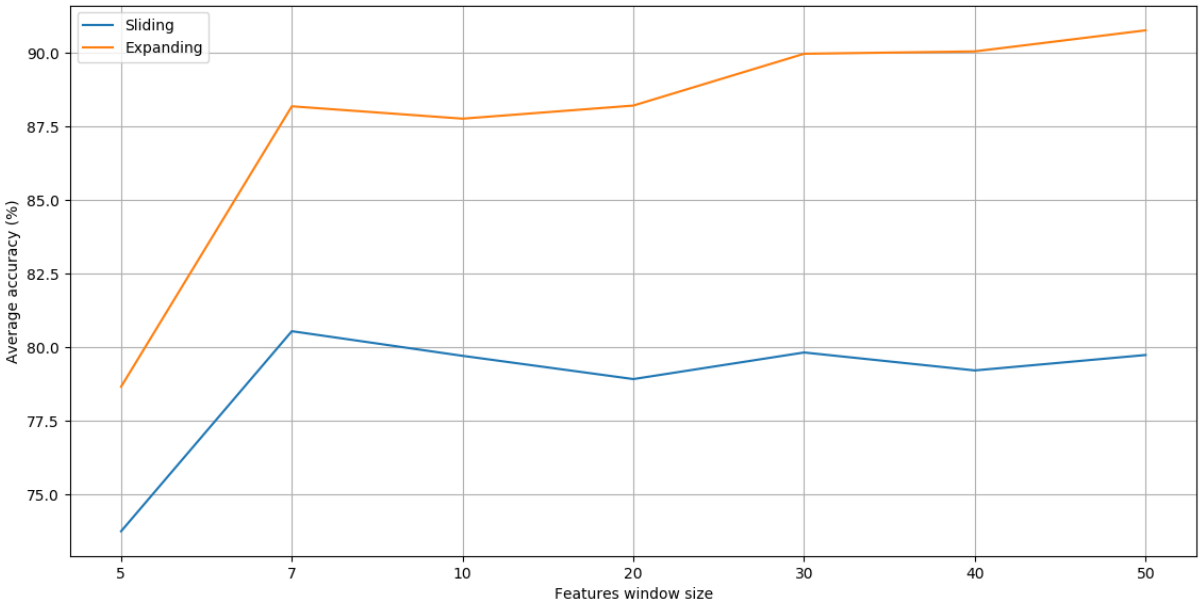


In these results we use a sliding window approach: the size of the training window is fixed.

We have tried also to run some experiments using the expanding window validation process, where the size increases while time is going on.

Figure 13 shows the average accuracy obtained by the two approaches, with different values of the features window size and with a training window size equal to 130 (for the expanding window, it is the initial size).

Figure 13: Accuracy with sliding vs. expanding window



The accuracy of the classifier is higher using the expanding window approach, independently on the size of the features window, since the expanding window is larger, and we can identify more periodicities and occurrences among past data. In the case of the sliding window, instead, this is ignored, as explained before for small sizes of the windows.

However, the computational time required to run our model with an expanding window approach is very high (227 seconds, on average). On the contrary, for the sliding window this computational time is lower (57.5 seconds, on average).

But the difference in performance when the features window is small is not that high, thus the trade-off between the improvement in accuracy (around 5%) and the increase in computational complexity (around 400%, on average, up to 1000% in the worst case) cause us to prefer using a sliding window approach.

For these reasons, hereafter in this work we will use a sliding window by setting a training window large 130 and a features window large 7.

6.3 NEXT-DAY PREDICTIONS

In this section we start exploring the results obtained in our experiments, by comparing the performances of different classification algorithms.

All tests are executed on the 66 vehicles of model 1254 of the Refuse Compactor, with a training window size equal to 130 and a features window size equal to 7.

For these preliminary experiments, we have set the standard configuration setting recommended by the scikit-learn implementation:

- **Support Vector Machines (SVM)**:
  - C = 1
  - kernel = "rbf"
  - gamma = "auto"
- **Multilayer Perceptron (MLP)**:
  - hidden_layer_sizes = 100
  - learning_rate = "constant"
- **Random Forest**:
  - n_estimators = 10
  - max_depth = None
  - min_samples_split = 2
  - min_samples_leaf = 1
  - max_features = "auto"

Beyond the Machine Learning algorithm, we have collected also the results achieved by a baseline approach, namely the Last Value algorithm. The predicted value of the baseline approach is the last observed one, i.e., the utilization level of tomorrow is assumed to be the same as the one recorded today.

Figure 14 shows some results in the binary next-day predictions scenario.

Class 0 contains the values in the series which are equal to zero, while class 1 contains all the other values, corresponding to the days in which a vehicle is working.

Figure 14: Results in the binary next-day scenario

Algorithm	Accuracy	Standard deviation	95% confidence interval	Precision class 0	Precision class 1	Recall class 0	Recall class 1
Last Value	71.9%	6.5	(70.4, 73.5)	41.2%	80.2%	41.3%	80.1%
Naïve Bayes	75.8%	10.4	(73.3, 78.3)	55.6%	89.2%	74.8%	77.2%
SVM	89%	5.6	(87.6, 90.4)	79.3%	91.9%	77.1%	92.5%
Random Forest	89.4%	5.6	(88.1, 90.8)	79.9%	91.5%	74.3%	93.1%

From these results, we can notice that results are similar for all the tested algorithms. Anyway, the Random Forest classifier achieves the best accuracy, around 90%, and the lowest standard deviation.

However, the computational time required to run ensemble methods, such as the Random Forest, is very high (up to 2110 seconds to be executed on all the 66 vehicles), because a certain number of trees is built for each model, and a new model is created for each day.

On the contrary, simpler algorithms such as the Naïve Bayes, are much faster and the execution time is around 160 seconds.

In figure 15 we show the results achieved in the 3-class next-day predictions scenario.

Class 0 contains the values in the series which are equal to zero, class 1 contains low and medium values of utilization (smaller than the 75th percentile), while class 2 contains high values of utilization.

We can immediately see that the average values of accuracy are lower than the ones obtained in the previous scenario. Indeed, this problem is much more difficult, since algorithms must predict three different classes.

Figure 15: Results in the 3-class next-day scenario

Algorithm	Accuracy	Standard deviation	95% confidence interval	Precision class 0	Precision class 1	Precision class 2	Recall class 0	Recall class 1	Recall class 2
Last Value	61%	10.6	(58.4, 63.5)	41.2%	54.8%	39.8%	41.3%	54.7%	39.8%
Naïve Bayes	67.7%	11.8	(64.8, 70.5)	67.8%	65.4%	39.5%	69%	57%	48.9%
SVM	79.6%	9.4	(77.3, 81.9)	79.1%	72.2%	41.1%	76.6%	67.5%	40.5%
Random Forest	78.6%	10.6	(76, 81.1)	77%	69.5%	44%	75.6%	68.3%	40.7%

We observe that precision and recall for class 2 (high level of utilization) are lower, probably this class is harder to predict. Again, we do not notice significative differences among the results obtained with different classifiers.

Some tuning of the parameters of the algorithms can be done in order to try improving these results.

6.4 SELECTION OF THE PARAMETERS OF THE ALGORITHMS

In the preliminary experiments we have used the standard configurations of all the algorithms. To tune the performance of the classification algorithms, we explored a variety of different configuration settings and investigated the motivations behind the achieved results.

Parameter tuning: Support Vector Machines

For the Support Vector Machine, the considered parameters are the kernel, C (penalty parameter of the error term) and γ (kernel coefficient if we use the RBF kernel). We perform some tests on the 10 vehicles mentioned in section 6.2, belonging to model 1254.

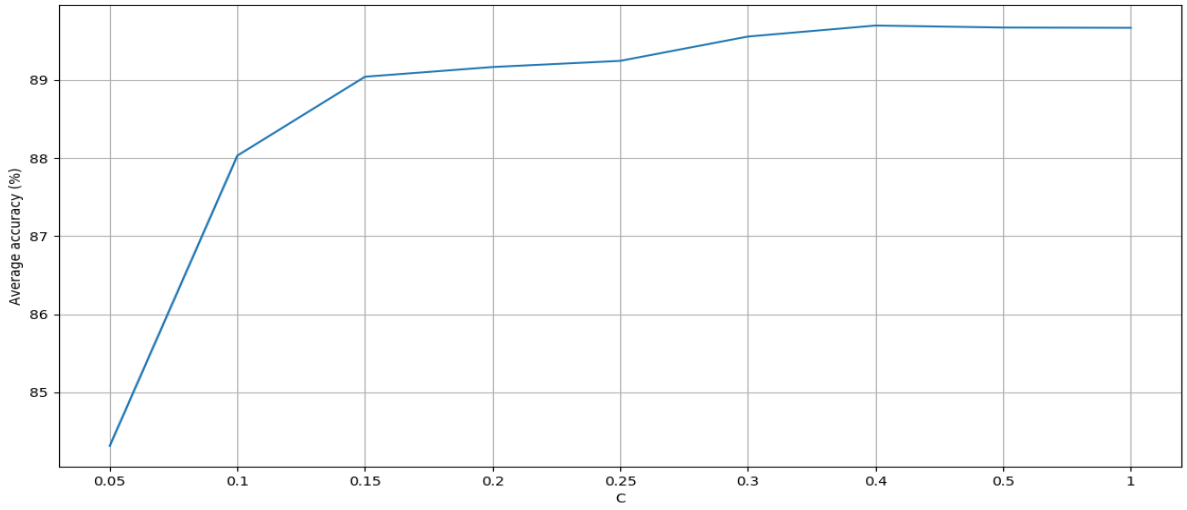
We try a linear kernel, a polynomial kernel and an RBF kernel, and we configure the parameters for each of them.

For the linear kernel, we can see the average accuracy in the binary scenario on figure 16, with different values of C.

The best accuracy overall (89.7%) is reached when C is equal to 0.4, while higher values of C (such as the default value, 1) give a slightly worse result. However, the differences are not significative, and even a small value of C is enough to guarantee an accuracy of 88-89%.

On the contrary, if C is very small, accuracy decreases significantly.

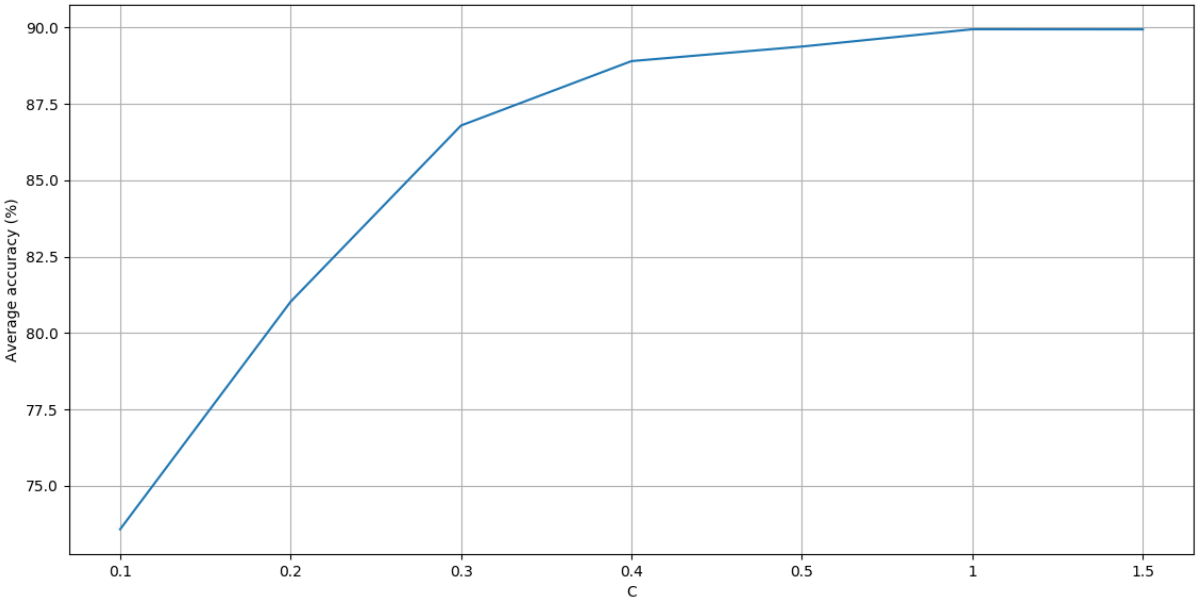
Figure 16: Accuracy with different parameters of the linear kernel SVM



Another type of kernel we can try is the polynomial one, again with different values of C.

On figure 17 we can see the average accuracy when the degree of the polynomial function is 3 (default value).

Figure 17: Accuracy with different parameters of the polynomial SVM



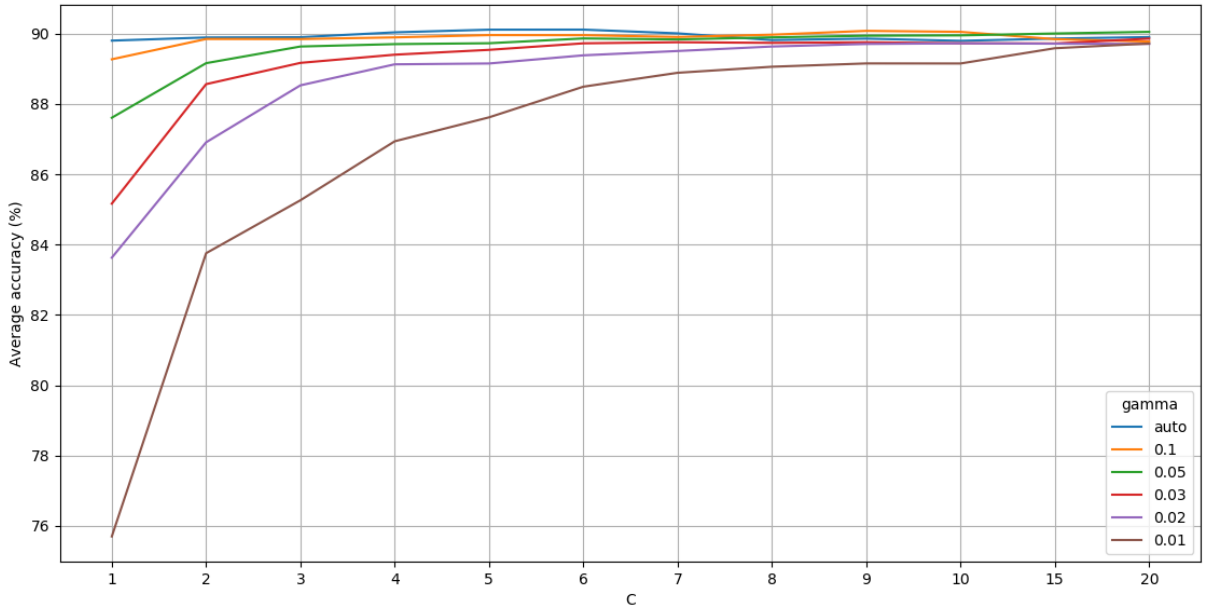
With C equal to 1 we obtain the best results, with an accuracy of 89.9%. So, the best performances are very similar to the ones reached with the linear kernel, if C is sufficiently high, while with smaller values of C we get worse results.

The last kernel we try is RBF (Radial Basis Function), which is the default configuration in scikit-learn. In this case we can tune also parameter γ, whose default value ("auto") is computed as 1/n_features.

Figure 18 shows the different values of accuracy obtained by varying C and γ.

All the curves have a similar behavior, and in general accuracy increases when the value of C is incremented. However, the best result overall is reached if γ is set to "auto" and C is equal to 6, since when γ is "auto" the obtained accuracy is around 90% independently on the value of C. This is the best accuracy we reach by using Support Vector Machines in the binary scenario, so in the next experiments we will use an RBF kernel with these parameters.

Figure 18: Accuracy with different parameters of the RBF SVM



With this setup we can achieve an accuracy of 90.1%, which is a slight improvement over the 89.8% obtained on the same 10 vehicles with the standard configuration.

Parameter tuning: Random Forest

For the Random Forest, we perform again some tests on the same 10 vehicles with different values of parameters. This algorithm is not easy to tune, because the computational time to be run is high and parameters to be configured are a lot.

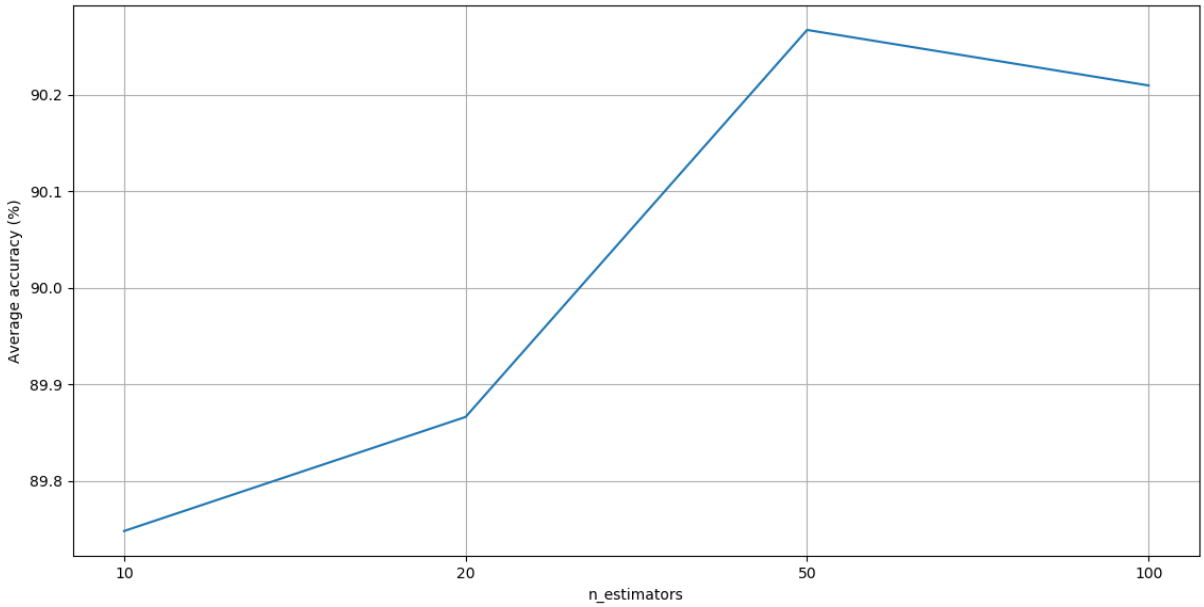
In these experiments, we always use the same value of the random state of the algorithm, in order to be able to compare the results.

The first parameter we try is max_features, i.e. the number of features to consider when looking for the best split. The best result is obtained when max_features is set to the default value "auto" (square root of the total number of features), and it gives an accuracy of 89.7%.

So, we can try optimizing other parameters, such n_estimators (number of trees in the forest), max_depth (maximum depth of the tree) and min_samples_split (minimum number of samples needed to split a node).

The average accuracy reached with different values of n_estimators is shown on figure 19.

Figure 19: Accuracy with different values of the n_estimators of the Random Forest

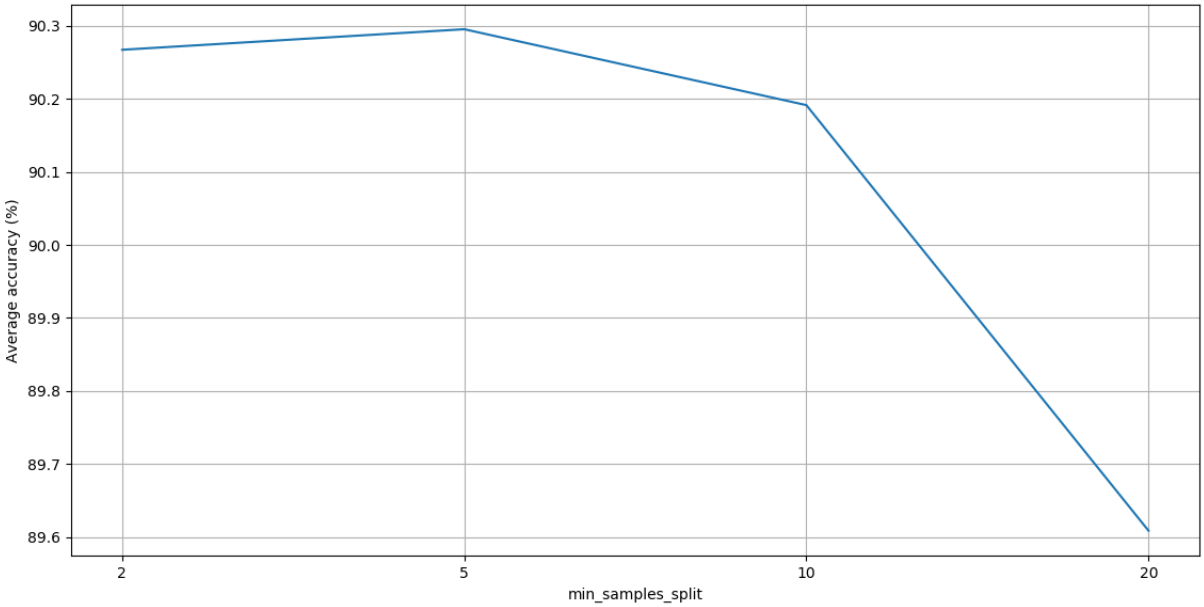


The best accuracy (90.3%) is given when the number of trees is equal to 50 and the maximum depth of the tree is unlimited (default value). By using smaller values of the number of trees, performances decrease, even if we notice an improvement in computational time.

Another parameter we can tune is min_samples_split, whose accuracy is on figure 20.

The best result is reached when min_samples_split is equal to 5, but anyway the differences in performances among the values are minimal, less than 1%.

Figure 20: Accuracy with different values of the min_samples_split of the Random Forest



In conclusions, the best configuration for the Random Forest is:

- max_features = “auto”
- n_estimators = 50
- max_depth = None
- min_samples_split = 5

With this setup we can achieve an accuracy of 90.3% in the binary scenario, which is a slight improvement over the 89.7% obtained on the same 10 vehicles with the default configuration.

6.5 ANALYSIS OF SINGLE VEHICLES

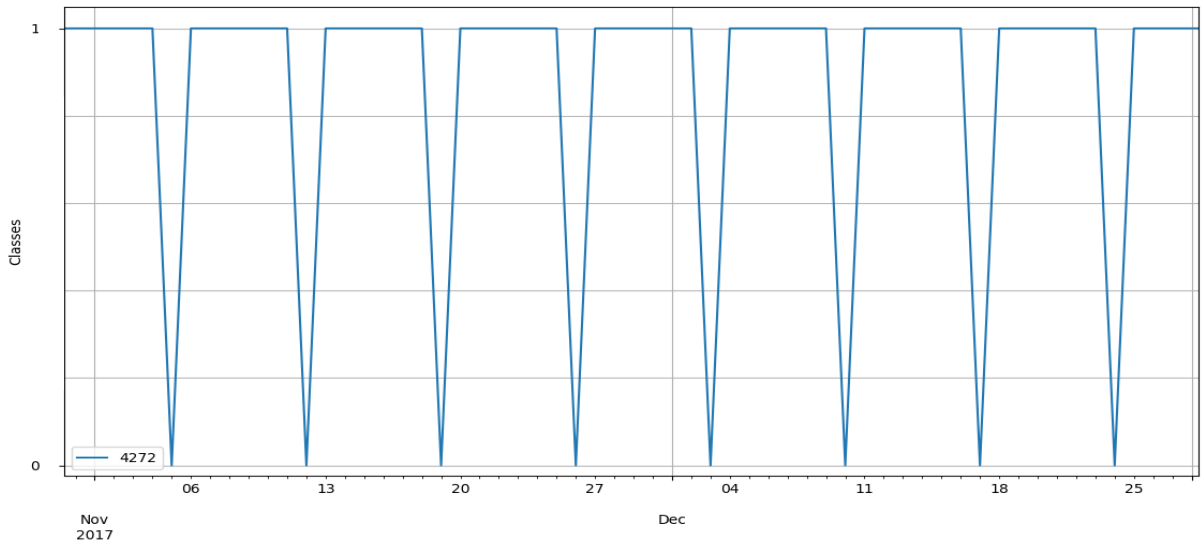
The results we have analyzed in the previous sections are averaged on all the vehicles belonging to model 1254. Since predictions are made vehicle-per-vehicle, in this section we examine in detail the predictions made on some specific vehicles.

An example of vehicle with stationary utilization level trends is the one with Identifier 4272. It has worked from February 2015 to August 2018. Figure 21 shows an extract of its discrete time series.

In this plot we can see that the usage trend in this period (November and December 2017) is very regular, i.e., the vehicle has worked all the days of the week except for Sundays.

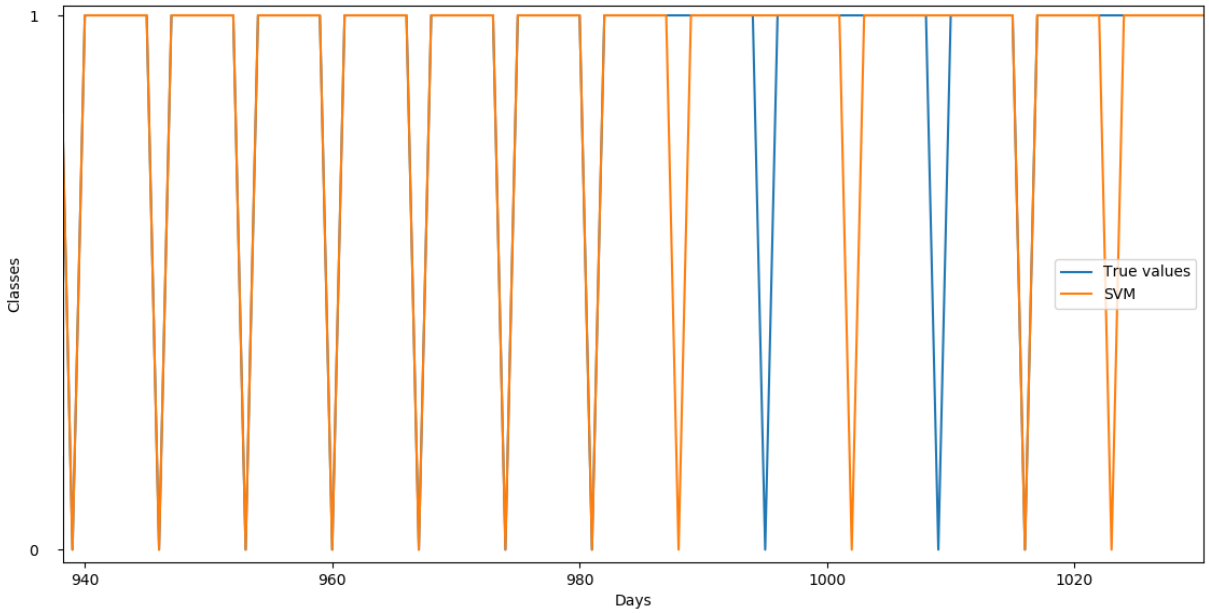
Indeed, if we consider the whole period, the number of zeroes present for this vehicle is equal to 253, which is not that different from the number of Sundays in that period.

Figure 21: Binary discrete series for vehicle 4272



In figure 22 we compare this time series with the time series of values predicted by a Support Vector Machine classifier. The sizes of the windows used in the reported experiment are the ones defined in the previous sections, i.e. 130 for the training window and 7 for the features window.

Figure 22: True and predicted values for vehicle 4272

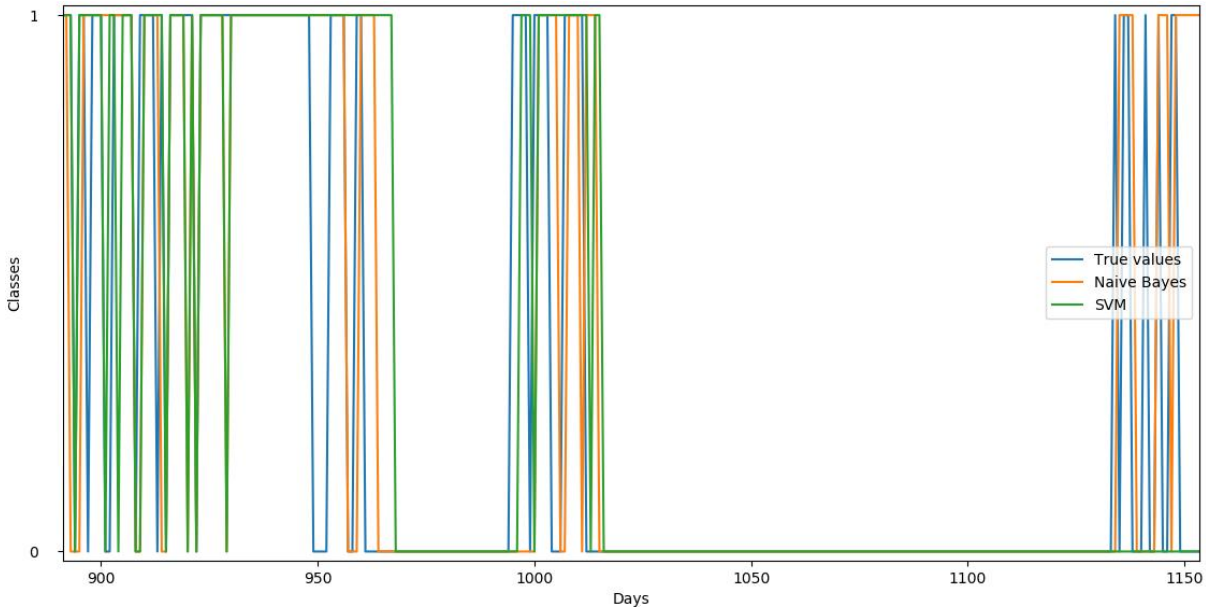


We notice that the algorithm in general can capture the utilization pattern, but in this sample, we observe some variations, i.e. some weeks in which the vehicle is working on Sundays too, and we see a sort of delay introduced by the classifier. Anyway, the accuracy reached by this algorithm for vehicle 4272 is 95.1%, which is higher than the average value for all the vehicles (89%).

Notably, a significant number of vehicles show non-stationary trends: as discussed in section 5.1, in the dataset we have identified vehicles that show long periods of zero utilization.

Frequent transitions between zero-one utilization levels are hard to predict. An example is vehicle 3717, which in 2018 is not working for a period of four months, shown on figure 23 in the binary scenario.

Figure 23: True and predicted values for vehicle with id 3717



We can see the true discrete series (in blue) and the values predicted by a Naïve Bayes classifier (in orange) and by a Support Vector Machine (in green).

When the vehicle stops working, both algorithms show a transitory period in which the predicted values are adjusted to the true class, and we can observe a delay.

Between the two shown algorithms, Naïve Bayes seems to be the one which behaves better in this situation, with a shorter delay also when the vehicle begins working again.

Other examples of vehicles which show frequent transitions between class 0 and class 1 can be seen on figure 24.

Figure 24: Vehicles which show frequent 0-1 transitions

Vehicle ID
3853
5512
5505
5231
5548

Indeed, these vehicles give a value of accuracy lower than 60% by using a Naïve Bayes classifier: the worst case is vehicle 3853, for which we get a value of only 42.5%.

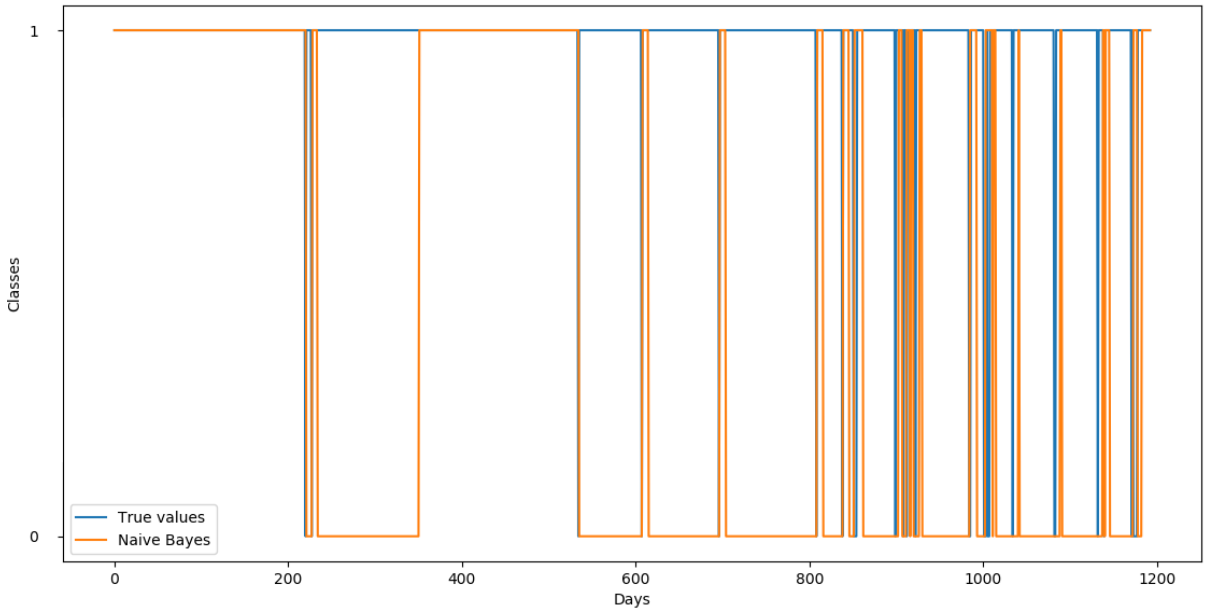
In figure 25 we can see its true discrete series compared to the values predicted by the algorithm.

For this vehicle the number of zeroes is low, but we can notice some unexpected changes in its usage, when utilization suddenly goes from class 1 to class 0, and then returns to class 1.

This creates some problems to the classifier, which is not able to identify the correct class for a certain period and, again, a sort of delay can be observed.

For these reasons we try introducing a different scenario, by removing zeroes from the series, in order to understand if these problems can be solved.

Figure 25: True and predicted values for vehicle 3853



6.6 NEXT-WORKING-DAY PREDICTIONS

Since it is hard to predict transitions between zero and one in the utilization level after a quite long period of no utilization, we consider also the next-working-day scenario. This simpler scenario focuses on predicting the date on which the vehicle will be working again.

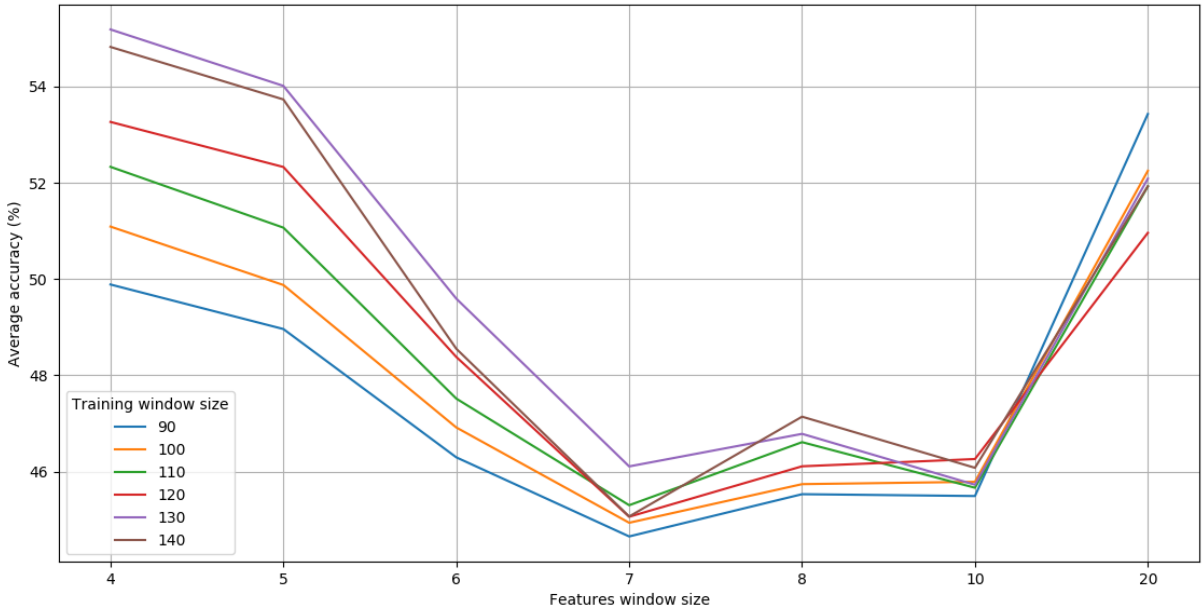
Even in this scenario, we will use a sliding window approach, due to significant efficiency improvement achieved in terms of computational time respect to an expanding window, as explained in section 6.2.

We can try again some tests on the same 10 vehicles by running a Naïve Bayes classifier, with different values of the training window size and of the features window size. The average accuracy in the binary case is shown on figure 26.

In general, with larger sizes of the training window we can get better results, even if some variations are present, due to the size of the features window. Anyway, the best accuracy overall is obtained when the training window is large 130 and the features window is large 4, so in the next experiments we will use these values.

We notice that in this scenario, by using this algorithm, we reach an average accuracy around 50%, which is lower than the one achieved in next-day predictions. This is probably due to some difficulties encountered by Naïve Bayes classifier in handling this time series, since the performances obtained by other ML algorithms are much better, as we will see later in this section.

Figure 26: Accuracy with different size of the windows in next-working-day predictions



We execute again the machine learning algorithms on all the vehicles belonging to model 1254, and we compute the average values of accuracy, precision and recall, in the binary scenario.

We can see some results on figure 27. Class 0 contains the values in the series which are smaller than the 25th percentile (low utilization), while class 1 contains all the other values.

Figure 27: Results in the binary next-working-day scenario

Algorithm	Accuracy	Standard deviation	95% confidence interval	Precision class 0	Precision class 1	Recall class 0	Recall class 1
Last Value	77.2%	11.8	(74.3, 80.1)	37.2%	79%	37%	79%
Naïve Bayes	56.7%	19.5	(52, 61.4)	32.1%	79%	54%	60.2%
SVM	81.1%	11.8	(78.2, 84.1)	41%	78.4%	22.1%	87.8%
Random Forest	81.9%	11.6	(78.9, 84.5)	44.6%	77.7%	25.9%	85.3%

These results show that, indeed, the values of recall for class 0 are low, because of the small number of observations belonging to that class, even if the overall accuracy is quite high.

In this context, the accuracy achieved by the baseline approach (predict the last value) is comparable to those obtained by the machine learning algorithms. Hence, due to the inherent simplicity of the addressed task, applying ML techniques to address the binary classification problem (utilization level zero or one) is not effective.

In this scenario, in general, the average accuracy is lower than in the next-day scenario. The best classifier is, again, the Random Forest, but its performances are very similar to the one reached by Support Vector Machines.

On figure 28 we show some results in the 3-class next-working-day scenario. Class 0 contains the values in the series which are smaller than the 25^{th} percentile (low utilization), class 1 contains medium values of utilization, while class 2 contains high values of utilization (larger than the 75^{th} percentile).

We observe that precision and recall for class 0 and 2 are lower, since even in this case, classes are not balanced: class 1 contains the values between the 25^{th} and the 75^{th} percentile and is approximately as large as the other two classes together. Again, we do not notice significative differences between the results obtained with Support Vector Machines and with the Random Forest, while we observe that the accuracy obtained by the Last Value algorithm is quite high even in this scenario.

Figure 28: Results in the 3-class next-working-day scenario

Algorithm	Accuracy	Standard deviation	95% confidence interval	Precision class 0	Precision class 1	Precision class 2	Recall class 0	Recall class 1	Recall class 2
Last Value	65.5%	12.5	(62.4, 68.5)	37.2%	57.9%	30.9%	37%	58%	31%
Naïve Bayes	54.6%	13.6	(51.3, 57.9)	34.9%	57.7%	26.6%	36.7%	51%	31.6%
SVM	70.2%	12.8	(67, 73.4)	37.4%	58.6%	29.4%	24.3%	62.8%	31.6%
Random Forest	70.3%	13.2	(67.1, 73.6)	39.6%	57.7%	28.1%	28%	60.8%	30.5%

6.7 ADDITIONAL FEATURES OF THE MODEL

Until now, we have considered as features of our model only the past values of utilization hours, but we can try inserting additional information in order to understand if we can improve the results.

As introduced in section 5.1, we perform a correlation analysis to know which of the features available for the vehicle can be the most correlated with its utilization hours.

For each date on which a vehicle is working, in our dataset we have the fuel consumption, the travelled distance, and the average latitude and longitude during the day. From the values of latitude and longitude we can also deduce the location (name of the nearest city and the country) where the vehicle is working.

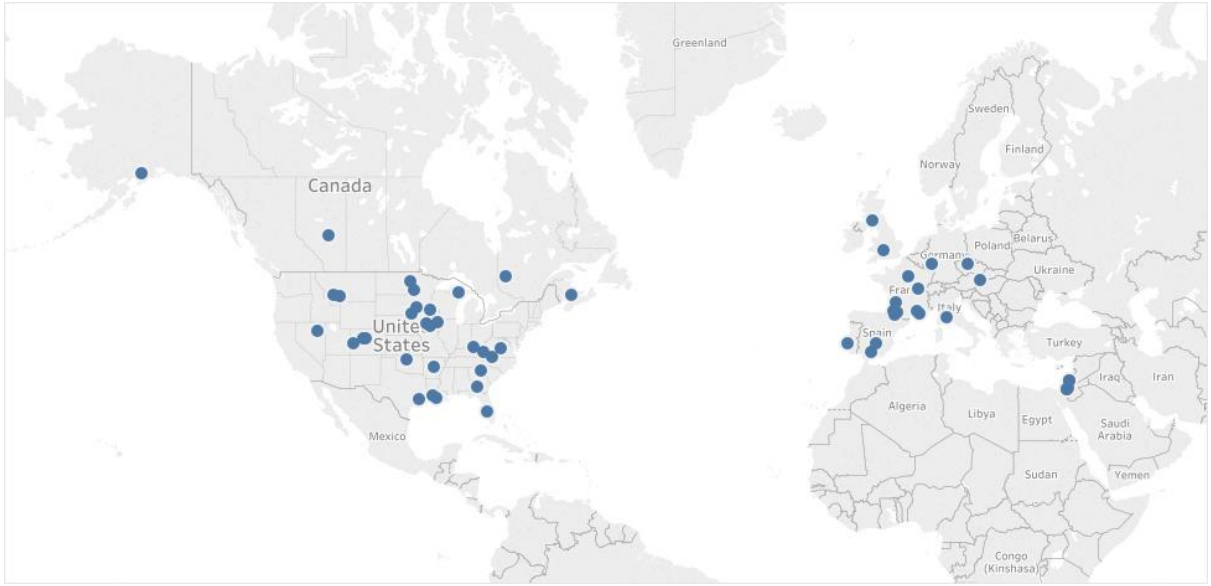
On the contrary, for the dates on which a vehicle has not worked, the same information is missing, thus we set the fuel consumption and the travelled distance equal to zero.

For what concerns the position of the vehicle (latitude, longitude, location), we assume that it is the same position collected on the last date on which the vehicle is working.

From the date we can know the season of the year and the day of the week, and we can understand whether that specific date was a working day or a holiday.

On figure 29 we can observe the geographical distribution of the vehicles of model 1254 on December 20th, 2017.

Figure 29: Positions of the vehicles of model 1254 working on December 20th, 2017



On this date most vehicles are working in Europe, especially in France, and in the United States. We notice some vehicles in Canada and in Israel too.

Since the countries where vehicles are working are several, to understand if a specific day is a working day or a holiday, we must know the country where the vehicle is working, since a day can be a holiday in some countries and a working day in others.

We add all these features to the dataset of model 1254, and we apply the Random Forest classifier, to understand which features are the most correlated with utilization hours, by exploiting the attribute features_importances_, available in scikit-learn.

Our approach is iterative: we first apply the algorithm to identify the most correlated feature, we exclude it from the dataset, then we apply again the algorithm to identify the second most correlated feature, etc. This analysis is performed on the dataset which contains zeroes, with the values of utilization hours discretized in two classes (binary next-day scenario).

The obtained ranking of features is shown on figure 30.

The most correlated feature with utilization hours is fuel consumption. This is not surprising, since the number of liters consumed depends on the number of hours the vehicle is active.

The second feature in our ranking is the distance travelled during the day and we notice that spatial features (longitude, latitude, location) seem to be more correlated than temporal features (day of the week, etc.).

Figure 30: Correlation ranking of features with utilization hours

Feature ranking
fuel_consumption_l
traveled_distance
lon_mean
lat_mean
location
weekday
holiday
season

So, we can try inserting the most correlated features (fuel consumption, travelled distance, longitude, latitude) into our model, in order to understand if we can achieve better results than the ones obtained in the previous sections.

The approach used for predictions is always the same, a new model is built every day and we use a sliding window as large as the training window size. What we change is the features window size, which in this case is the number of features (including past utilization hours) multiplied by the number of days.

For example, if in the previous cases we had a features window size of 3 (corresponding to 3 days), in this case we would have a size equal to 3 times the number of the different features, which is 5 (utilization hours, fuel consumption, travelled distance, longitude, latitude), i.e. the features window is large 15.

In figure 31 we can see an example: with two features only, the features window is large 6.

Figure 31: Training set with one additional feature

Index	Utilization level (x)	Fuel consumption (c)
0	1	34.5
1	0	0
2	0	0
3	0	0
4	0	0
5	1	79.8
6	1	135
7	1	122
8	1	26.6
9	0	0

Index	x(t-3)	x(t-2)	x(t-1)	c(t-3)	c(t-2)	c(t-1)
3	1	0	0	34.5	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	1	0	0	79.8
7	0	1	1	0	79.8	135
8	1	1	1	79.8	135	122
9	1	1	1	135	122	26.6

After adding these features, we execute again the machine learning algorithms on all the vehicles belonging to model 1254, and we compute the average values of accuracy, precision and recall, in the binary scenario.

We can see some results on figure 32.

Figure 32: Results in the binary next-day scenario with additional features

Algorithm	Accuracy	Standard deviation	95% confidence interval	Precision class 0	Precision class 1	Recall class 0	Recall class 1
Naïve Bayes	84%	8.2	(82, 86)	68.5%	89%	70%	88.2%
SVM	76%	7.4	(74.1, 77.8)	44.2%	75.9%	11%	98.1%
Random Forest	89.8%	4.9	(88.7, 91)	80.6%	91.9%	75%	93.5%

Figure 33 shows instead some results in the 3-class scenario.

Figure 33: Results in the 3-class next-day scenario with additional features

Algorithm	Accuracy	Standard deviation	95% confidence interval	Precision class 0	Precision class 1	Precision class 2	Recall class 0	Recall class 1	Recall class 2
Naïve Bayes	70.3%	12.5	(67.3, 73.3)	71.3%	64.6%	37.6%	66%	61%	45.6%
SVM	62.7%	11.9	(59.7, 65.7)	42.3%	47.9%	25.9%	13.5%	61.6%	36.6%
Random Forest	80%	9.4	(77.8, 82.3)	79.9%	71.4%	43.5%	77%	68.6%	41.2%

The best performances overall are reached again with the Random Forest. As already noticed in section 6.3, the results in the 3-class case are worse than the ones obtained in the binary case, since it is a more challenging problem.

If we analyze the computational time required to run the machine learning algorithms on all the vehicles of model 1254 with or without these additional features, we obtain the results shown on figure 34.

These tests are executed in the binary next-day scenario. We notice that for all the algorithms the execution time is higher (up to five times) if we use additional features, since the features window is larger, and the model must handle a bigger training set.

The most visible increment in execution time is observed for the Support Vector Machine classifier, and, for this algorithm, adding features does not improve results.

On the contrary, for the Naïve Bayes classifier, by using additional features, accuracy can reach an improvement of the 8%, but the execution time is approximately four times larger.

Figure 34: Computational time with or without additional features

Algorithm	Additional features	Execution time
Naïve Bayes	No	160 s
Naïve Bayes	Yes	697 s
SVM	No	145 s
SVM	Yes	815 s
Random Forest	No	2110 s
Random Forest	Yes	6020 s

7. CONCLUSIONS

In this thesis, we have presented the problem of forecasting the usage of construction vehicles by proposing some methodologies that use machine learning algorithms.

We have encountered some difficulties in handling equally spaced time series, due to the high number of zeroes and of zero-one transitions, so we have tried simplifying the problem by removing the zeroes from the series.

The different machine learning algorithms have not shown significative differences among each other. Then, in addition to the values of the past observation of utilization hours, we have introduced new spatial and temporal features, such as the city where the vehicle is working, the season of the year and the day of the week.

For a possible future development, some other features could be added, like the category of work executed by the vehicle, the weather on that day, etc.

Since vehicles have shown very different patterns of utilization among each other, we have chosen to make our predictions vehicle-per-vehicle and then to average the obtained results on all the vehicles belonging to a specific model.

Another possible evolution of the problem could be grouping (clustering) vehicles based on their usage, in order to make group predictions by using the past data of more than one vehicle.

In the construction of our model, we have not built a single model per vehicle, but a new model is created for each day, and then results are evaluated by means of the confusion matrix, by comparing the predicted series to the original one.

By examining all the results, we have found some techniques that can reach an accuracy above 80% (averaged on 66 vehicles) in a binary scenario, in which the algorithm must classify values into two classes. In a 3-class scenario, instead, the obtained accuracy is lower, due to the higher difficulty of the problem.

8. REFERENCES

Perrotta F., Parry T., Neves L. (2017). "Application of Machine Learning for Fuel Consumption Modelling of Trucks".

Zeng W., Miwa T., Wakita Y., Morikawa T. (2015). "Exploring Trip Fuel Consumption by Machine Learning from GPS and CAN bus Data".

Siami-Irdemoosa E., Dindarloo S. (2015). "Prediction of fuel consumption of mining dump trucks: A neural networks approach".

Çapraz A., Özel P., Şevkli M., Beyca Ö. (2016). "Fuel Consumption Models Applied to Automobiles Using Real-Time Data".

Wickramanayake S., Bandara H. (2016). "Fuel Consumption Prediction of Fleet Vehicles Using Machine Learning".

Han J., Kamber M., Pei J. (2012). "Data Mining Concepts and Techniques".

Tierra introduction presentation (2018).

Gandhi R., Donges N. (2018). Towards Data Science. Available at: https://towardsdatascience.com/

Patel S. (2017). Machine Learning. Available at: https://medium.com/machine-learning-101/

Beaulieu K., Dalisay D. (2019). Machine Learning Mastery. Available at: https://machinelearningmastery.com/

Python.org. (2019). Available at: https://www.python.org/

Scikit-learn.org. (2019). scikit-learn: machine learning in Python. Available at: https://scikit-learn.org/stable/