Machine learning model to quantify the accordance of work output and objectives in a complex business environment

Bachelor's Thesis

Bachelor of Science in Physics

Academic Year 2020 – 2021

Author:

Raúl Almuzara

Supervisors:

David Íñiguez
Francisco Marías

Contents

1 Introduction

Notice: *This a complete translation into English of the original report, which was written in Spanish. Each sentence has been translated manually. Only the variables described later keep their original names because they also appear in the code in that way.*

In recent years, the use of advanced computational techniques for data analysis has become an indispensable activity for decision-making and process optimization. An adequate processing can reveal patterns and extract hidden information in datasets of any size. In both research and business, terms such as *Big Data* and *Machine Learning* are often heard due to the revolution they have produced in performance, cost reduction, cybersecurity, recommendation systems, etc.

In this project, the aim is to show the effectiveness of some artificial intelligence algorithms to solve a specific problem proposed by a real company. This company is called *Distromel* and, among other activities, they develop software for urban waste management. In order to carry out this project, the data provided by this company are available. The objective is to design several models that allow us to obtain information from some input data about the degree of goodness associated with activities of urban waste collection.

Firstly, we must carry out an adequate selection, filtering and preparation of the data that will be used to train the machine learning models. Then, we must perform a careful analysis of the most relevant variables and their mathematical properties in order to obtain information with a real utility. Finally, with enough training data, we proceed with the design of the artificial intelligence models that will learn from the data in order to make intelligent predictions. These predictions will be made with classification algorithms (determination of a qualitative category) and regression algorithms (obtaining a numerical value). We will also present a sufficiently self-contained description of relevant statistical concepts which are useful to build and evaluate machine learning models. In particular, after the dataset is cleaned and preprocessed, we will create decision trees, neural networks and models based on these to observe the way they learn and the results that can be achieved depending on the model.

2 Problem statement

Many variables are involved in urban waste management and they produce huge amounts of data (GPS trajectories, location of dumpsters, fulfillment of the planning, etc.). They must be analyzed with an appropriate statistical processing and relevant analysis techniques. An intelligent management of waste collection can be carried out with artificial intelligence algorithms that optimize these processes and provide valuable information on the activity of the waste collection vehicles.

The problem that we will tackle is **the quantification of the degree of accomplishment associated with a specific work order**. The processing of the variables involved in the execution of work orders is not trivial. Neither is the definition of an objective goodness metric that can be obtained automatically. For this reason, we propose the evaluation of different artificial intelligence algorithms that process the available data and could be used in the future to estimate reliable results that can be generalized to more similar datasets.

We have information about theoretical plannings (each associated with a work order) that are proposed by the company's customers and include well-defined GPS tracks and planned dumpsters in specific locations. These work orders are designed in advance to be executed in real life as efficiently as possible. As the waste collection vehicles go through the actual tracks, they collect information about the same variables for which theoretical information is available.

Geographical trajectories are defined by points, which are also defined by their geographical coordinates (longitude and latitude). Each trajectory is defined by numerous points that are obtained differently for theoretical routes and actual tracks. For theoretical routes, these can be obtained with optimizers that generate the best route according to the specified conditions. For actual tracks, the points are those recorded periodically by the GPS. The size of each time interval depends on the speed of the vehicle, the angle it turns in certain sections and the distance traveled since the last recorded position. This is to ensure that the distribution of recorded points allows to reconstruct a reliable approximation of the tracks.

3 Data preprocessing and description of variables

In projects that involve the analysis of spatial information, visualization and processing of geographically referenced data, we use *Geographic Information Systems* to obtain graphical information such as the figure shown below:

Figure 1: Example of the geographical information of a certain work order. In red, the planned theoretical route and, in green, the actual track. The dots in yellow are the dumpsters that were planned and were collected, the dots in blue are the dumpsters that were planned, but were not collected and the dots in gray are the dumpsters that were not planned, but were collected. The importance of this differentiation will be explained on the next subsection.

3.1 Data exploration

During this project, we have carried out a thorough analysis of the company's databases to assess the amount of data needed, their quality and their availability according to all the variables of interest. In general, it is better to have a dataset with diverse samples so the models can learn from all possible cases that may occur in reality.

The information is divided into *work orders*. Our work orders are groups of data containing information about the geometry of the theoretical route planned to be traveled by the waste collection vehicle, the geometry of the track followed by the vehicle in reality when executing the order and information about the collection of all the dumpsters involved in the order.

In relation to these dumpsters, it has been observed that the vehicles record up to three types in the databases:

- Dumpsters that **WERE** planned and **WERE** collected: This is the optimal and desirable situation that occurs when the waste collection vehicle correctly executes the theoretical planning on a certain dumpster marked by an identifier. Let $N(1,1)$ be the number of such dumpsters in a given work order.
- Dumpsters that **WERE** planned, but **WERE NOT** collected: The opposite situation to the previous one and indicative of a bad execution of a work order on a certain dumpster, which will negatively affect the goodness of the order. Let $N(1,0)$ be the number of such dumpsters in a given work order.
- Dumpsters that **WERE NOT** planned, but **WERE** collected: Alternatively, the vehicle may collect a dumpster whose identifier does not appear in the original planning, but is still recorded in the work order execution data. Let $N(0,1)$ be the number of such dumpsters in a given work order.

From the available data, we have made a sufficiently representative selection of 623 work orders.

3.2 Variables of interest

We have analyzed large amounts of real data provided by the company to select those groups of data that best help to achieve the objective of this project. From these, we have built several numerical variables with the objective of modeling the most relevant features of a work order while maintaining a certain balance with the number of usable data so that the estimations are as reliable as possible. Studying the influence that the variables have on the final assessment that we can make of each work order, we will train models based on the variables defined and described below in order of importance.

3.2.1 Proportion of well-collected dumpsters

In each work order, we must prioritize the collection of the largest number of dumpsters among those planned. If for each work order we extract the number of dumpsters of each type, we can

define a variable that is the proportion of dumpsters that are planned and collected to the total number of planned dumpsters. This is how the variable `BienHechos` is defined:

$$A = \frac{N(1,1)}{N(1,1) + N(1,0)} \tag{1}$$

By construction, it is a variable that is already normalized between 0 and 1.

3.2.2 Degree of similarity between trajectories

Given two geographical trajectories defined by a set of points on the map, we wish to somehow calculate their degree of similarity. We need an objective and quantifiable measure of how similar a theoretical route and an actual track are, i.e., the *distance* between them. We want to obtain a numerical value that gives the level of similarity between the theoretical route that was planned for the waste collection vehicles and the actual track that the vehicle has followed in each of the work orders.

In general, the calculation of the similarity of pairs of trajectories is a complex mathematical problem that has been studied in recent years. As a result, a large number of algorithms have been proposed that may be suitable to a greater or lesser extent for each particular problem. The effectiveness of these algorithms depends on the complexity of the trajectories and their geometrical characteristics. In our case, it can be seen in figure 1 that theoretical routes and actual tracks can intersect and have a remarkably complicated shape, so it is not trivial to find an objective measure of how similar any two GPS tracks are. From among many of the most popular algorithms, the distance that has been finally chosen is detailed below as it is the most accurate in relation to the requirements of this project.

Let T^1 and T^2 be two trajectories, both defined as an array of n_1 and n_2 points of dimension 2 each, respectively. Let $P^1_{i_1}$ be the i_1-th point of the array of points of the trajectory T^1 and let $P^2_{i_2}$ be the i_2-th point of the array of points of the trajectory T^2. As a preliminary step, we define the *Segment-Path Distance* (SPD) from the trajectory T^1 to the trajectory T^2 as

$$D_{SPD}(T^1, T^2) = \frac{1}{n_1} \sum_{i_1=1}^{n_1} D_{PT}(P^1_{i_1}, T^2) \tag{2}$$

where

$$D_{PT}(P^1_{i_1}, T^2) = \min_{i_2 \in [0,\ldots,n_2-1]} D_{PS}(P^1_{i_1}, S^2_{i_2}) \tag{3}$$

and $D_{PS}(P^1_{i_1}, S^2_{i_2})$ is the distance from the point $P^1_{i_1}$ to the segment $S^2_{i_2}$.

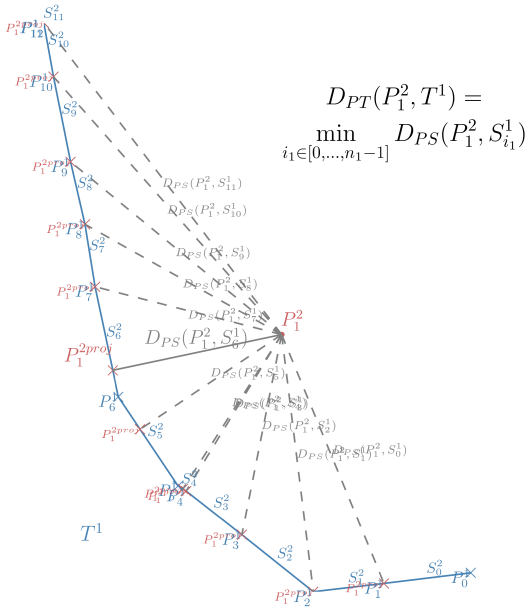

Figure 2: Calculation of the distance from the first point of the trajectory T^2 to the trajectory T^1.

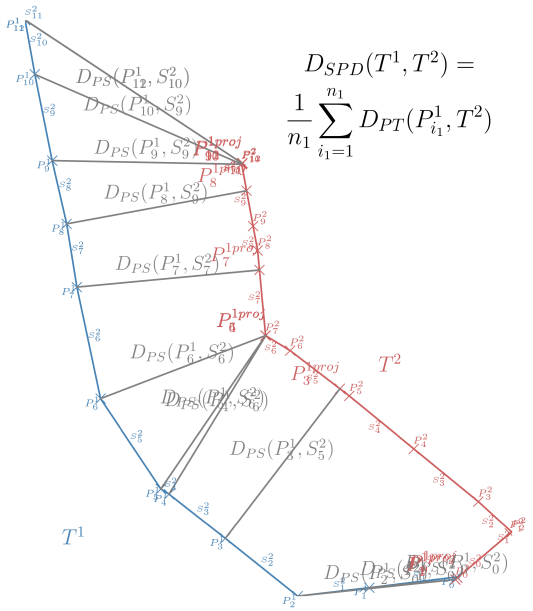

Figure 3: Relevant distances in the calculation of the SPD from the trajectory T^1 to the trajectory T^2.

As the points are geographical coordinates, an accurate distance between two points will be the one given by the haversine formula:

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \tag{4}$$

where r es the radius of the Earth, φ_1 and φ_2 are the latitudes of the two points and λ_1 and λ_2 are the longitudes of the two points. Nevertheless, in order to significantly reduce the computation time if we need to perform many similarity studies, it can be approximated by the Euclidean distance and give essentially similar results in our case.

Intuitively, the SPD can be interpreted as the average of all distances from the points that form trajectory T^1 to trajectory T^2. However, the degree of similarity between two trajectories should be independent of the order in which one or the other is considered, so we symmetrize the distance by defining

$$D_{SSPD}(T^1, T^2) = \frac{D_{SPD}(T^1, T^2) + D_{SPD}(T^2, T^1)}{2} \tag{5}$$

which is called Symmetrized Segment-Path Distance (SSPD) and will be used from now on to quantify the degree of similarity between pairs of theoretical routes and actual tracks. With this distance, we can compare paths of different lengths and it is not badly affected by the fact that the time indexing of the points of the paths is very different. Its computational complexity is $\mathcal{O}(n_1 n_2)$.

This distance is a number between zero and infinity. A very small distance indicates a high degree of similarity between two paths, whereas a high distance indicates that two paths are very dissimilar. However, the training will only be effective in the subsequent machine learning models if we normalize the variables to a common scale. If we want the degree of similarity as a function of the SSPD to take values between 0 and 1, we may use a transformation of the form

$$B = \frac{\xi}{\xi + D_{SSPD}(T^1, T^2)} \tag{6}$$

which is strictly decreasing and maps numbers in the interval $(0, \infty)$ to numbers in the interval $(0, 1)$. Furthermore, it always satisfies that when the distance tends to 0, the degree of similarity tends to 1 and when the distance tends to infinity, the degree of similarity tends to 0. In order to choose the parameter ξ, we have to make sure that the final distribution of the variable B is consistent with the percentage of the distances that can be considered good or bad according to our criteria. We can also manually set the value of the similarity of several pairs of theoretical routes and actual tracks and fit the function. In the appendix, a more exhaustive explanation of the empirical estimation of this parameter is detailed. For now, we can set it as $\xi = 0.001$. This variable B will be the variable `Similitud`.

3.2.3 Proportion of additional collected dumpsters

In reality, beyond the theoretical planning, some dumpsters which do not belong to the theoretical route may be collected. We notice this when the vehicle collects a dumpster with an identifier that does not belong to those included in the planning of a particular work order. This situation can happen due to inaccuracies in the GPS system that guides the drivers, unexpected road closures, human errors or a change of the planning after the routes have been designed because the contractor may have new needs. For the models that will be shown, we will consider that the collection of extra dumpsters is not necessarily a bad thing if it does not cause an excessive deviation from the theoretical GPS route and if it does not affect badly the collection of those dumpsters that were originally planned.

We define the variable `Adicionales` as the proportion of dumpsters that were not planned but were collected to the total number of dumpsters associated with a work order:

$$C = \frac{N(0,1)}{N(1,1) + N(1,0) + N(0,1)} \tag{7}$$

By construction, it is a variable that is already normalized between 0 and 1.

3.2.4 Relative deviation between actual and theoretical length

Returning to the analysis of the geometry of theoretical and actual GPS tracks, the length of these tracks may also be interesting. This variable may not necessarily appear to be as reliable as the degree of similarity between paths because lengths can be affected in some cases by unforeseen situations that do not necessarily reduce the quality of the execution of a work order. This includes the need to visit the dump to unload or detours of no particular importance around the urban centers of interest before starting or after finishing the collection of all the planned dumpsters. Nevertheless, we will see that it is a variable of interest because of its correlation with other indicators of goodness, although we will assign it the lowest weight.

We calculate the absolute value of the difference between the theoretical route length and the actual track length, divided by the theoretical route length:

$$D^* = \frac{|Theoretical\ Length - Actual\ Length|}{Theoretical\ Length} \tag{8}$$

Mathematically, this variable is not already normalized between 0 and 1 for all pairs of lengths. However, it has been observed that the distribution of this variable does not go too far from this range. Thus, it is easy to move the values to the desired range without altering the shape of the distribution with the linear transformation

$$D = \frac{D^* - D^*_{min}}{D^*_{max} - D^*_{min}} \tag{9}$$

which gives the value 0 to D^*_{min} (minimum value of D^*) and the value 1 to D^*_{max} (maximum value of D^*), as well as intermediate values to all other values of D^*. This normalized variable D will be called `RatioLongitudes`.

3.2.5 Goodness of a work order

As a dependent variable, we are looking for an indicator that gives us an idea of how well a given work order has been executed. As expected, there is not a single universal way to quantify the goodness associated with a work order. This depends on the level of importance that each contractor wants to assign to each of the variables.

Our dataset does not have specific values of goodness associated with each work order. Therefore, we are going to simulate our own goodness values with specific weights for each variable. We assume that the order of importance of the variables is the order in which they have been described in the previous sections (from highest to lowest). Suppose that a certain contractor wants to assign, according to their interests, the following relative weights: (10 , 4 , 2 , -1). Let us construct a linear combination of the four variables with weights 10, 4, 2 and -1 that show the level of importance that could be given to each variable:

$$10A + 4B + 2C - D \tag{10}$$

where variable D (relative deviation between actual and theoretical length) has a negative sign because a good work order must minimize it and not maximize it like the others. This linear combination tells us that the worst possible work order (with $A = 0$, $B = 0$, $C = 0$ and $D = 1$) would have the following value for goodness assigned to it:

$$10 \cdot 0 + 4 \cdot 0 + 2 \cdot 0 - 1 = -1 \tag{11}$$

whereas the best possible work order (with $A = 1$, $B = 1$, $C = 1$ and $D = 0$) would have the following value for goodness:

$$10 \cdot 1 + 4 \cdot 1 + 2 \cdot 1 - 0 = 16 \tag{12}$$

However, the variables A, B, C and D are, in general, non-integer real numbers and a human person would probably assign integer goodness values to each work order so we could have data with which to train the models. Also, if a person carried out such task, it would introduce some human error, which we will model by including some Gaussian noise.

Let us consider a normal random variable of mean $\mu = 0$ and standard deviation $\sigma = 0.7$. By adding this variable to the combination described in (10), the range $[-1, 16]$ that could be achieved with the combination (10) can be enlarged. For a better understanding of the goodness values, we apply again a linear transformation analogous to the one described in the expression (9), but multiplied by a factor 10 to adapt these goodness values to the range $[0, 10]$.

The next step is to divide these goodness values into 4 categories (or classes) so we can classify the work orders according to whether they are ***bad*** (class 0), ***mediocre*** (class 1), ***acceptable*** (class 2) or ***very good*** (class 3). With this, we will be able to train classification models that will automatically predict the group to which a new work order belongs. Given the range of goodness values we have between 0 and 10, we are going to perform a *clustering* using the k-`means` algorithm. This is the first machine learning algorithm that will be implemented in this project and it belongs to the group of unsupervised learning algorithms because it will be able to find $k = 4$ groups or *clusters* from unlabeled data. Initially, $k = 4$ *centroids* are chosen at random. The points in the set are associated with a particular cluster according to its nearest centroid. In each iteration, the centroids are updated and become the average of the points that make up each cluster. We proceed in this way until we minimize a cost function given by

$$J = \sum_{j=1}^{k} \sum_{b_i \in S_j} ||b_i - c_j||^2 \tag{13}$$

where $k = 4$ is the number of clusters, S_j is each of the clusters S_0, S_1, S_2 and S_3, b_i is each of our points representing values of goodness and c_j are the centroids. The optimal partition is the one with centroids c_j such that J is minimal. In our case, this partition will be one-dimensional in the range of goodness values compressed in the interval [0,10]. We will use this method to get an idea of how we could perform an objective partition into the 4 classes into which we will want to classify the work orders.

Consequently, we obtain the following partition into 4 real intervals.

Intervals of real goodness values	**Class**
$[0.00, 4.45)$	0 (bad)
$[4.45, 7.27)$	1 (mediocre)
$[7.27, 8.35)$	2 (acceptable)
$[8.35, 10.00]$	3 (very good)

Table 1: Partition into 4 classes according to goodness values.

Finally, we round the goodness values to the nearest integer to give more realism to the goodness scores that a human person would assign and we include them in one of the four intervals. In terms of integers, the division is as follows

Integer goodness values	**Class**
$0, 1, 2, 3, 4$	0 (bad)
$5, 6, 7$	1 (mediocre)
8	2 (acceptable)
$9, 10$	3 (very good)

Table 2: Assignment of integer goodness values to each class.

3.2.6 Joint review

Next, we show some plots about the variables that we have designed from the available raw data:

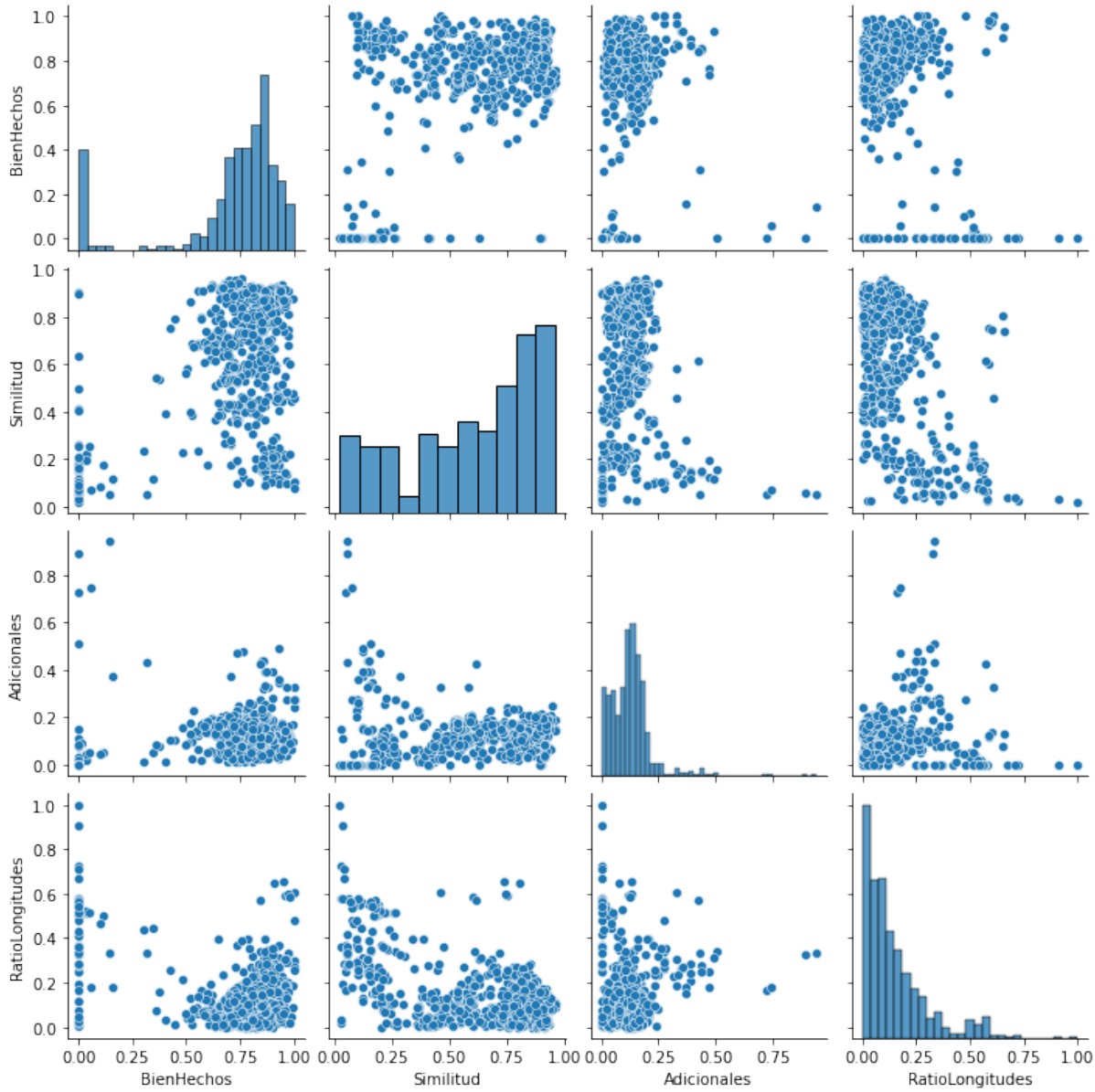

Figure 4: On the diagonal, distribution of the data of each variable. Outside the diagonal, plots of the variables taken in pairs.

We see that the proportion of well-collected dumpsters tends to be around high values, although it also has an isolated bar at zero due to the presence of work orders with extremely low goodness. This may be attributed to the fact that, in many cases, the waste collection vehicles do not have identification systems that allow the registration of dumpsters even if there is a high similarity between the planned route and the actual track. Nevertheless, the models will allow us to point out this type of anomalies.

Here, we can already see the correlation that exists in some cases as we see particularly dense

areas. There seems to be a relation between a high proportion of well-collected dumpsters, low proportion of additional collected dumpsters and low relative deviation between the length of the actual track and the length of the theoretical route. On the other hand, as expected, there is a correlation between high proportion of well-collected dumpsters and high degree of similarity. However, it is less clear because there is also a certain amount of orders with a high degree of accomplishment but low similarity due to last-minute changes in the contractor's plans or cases where the originally agreed theoretical planning is discarded.

Additionally, it has been obtained that the distribution of integer goodness values is as follows:

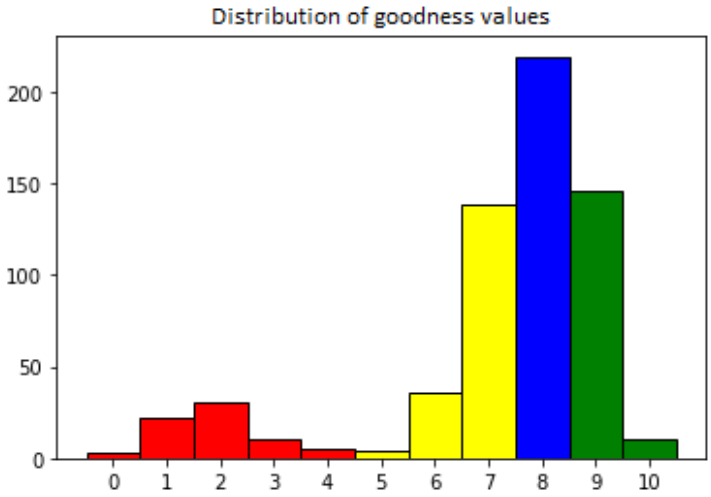

Figure 5: Number of work orders with each value of goodness from 0 to 10. In red, ***bad*** orders. In yellow, ***mediocre*** orders. In blue, ***acceptable*** orders. In green, ***very good*** orders.

4 Relevant statistical techniques in machine learning

Beyond a quick glance at the results obtained, we can use certain statistical techniques such as the ones described below to evaluate the capacity of the models.

4.1 ROC curves and confusion matrices

In classification problems, ROC (*Receiver Operating Characteristic*) curves are used to evaluate the effectiveness of models. ROC curves are graphical representations of probabilities and the area under the curve (AUC) is used to measure the degree of separability between the different classes that make up the model output.

Suppose the model is a binary classification model, i.e., it must predict to which of the two classes a certain sample belongs. Let us denote one of the two classes as the *positive class* and the other as the *negative class*. When trying to make a prediction about the class to which a certain sample belongs, four situations can occur:

- *True positive* (TP): The model correctly predicts the positive class.
- *False positive* (FP): The model incorrectly predicts the positive class.
- *False negative* (FN): The model incorrectly predicts the negative class.
- *True negative* (TN): The model correctly predicts the negative class.

We let the model make predictions on a dataset and we can construct a confusion matrix that reflects the number of predictions of each type. For the case of binary classification, this would be a 2×2 matrix of the form

$$\begin{pmatrix} TP & FP \\ FN & TN \end{pmatrix} \tag{14}$$

Also, the *true positive rate* (TPR), *false positive rate* (FPR) and *true negative rate* (TNR) are defined as follows

$$TPR = \frac{TP}{TP + FN} \qquad FPR = \frac{FP}{FP + TN} \qquad TNR = \frac{TN}{FP + TN} \tag{15}$$

Note that $FPR = 1 - TNR$. TPR is also known as *sensitivity* and TNR is known as *specificity*. ROC curves are two-dimensional plots of TPR (sensitivity) versus FPR (1−specificity) that we obtain after evaluating the model. Nevertheless, the models of this project are not binary classification models, but classification models into 4 classes. Therefore, for each of the 4 classes, we must reduce the problem to a problem of *one versus the rest* and we obtain a curve for each class.

4.2 Bootstrapping

Methods based on *bootstrapping* allow us to obtain a suitable approximation of the confidence intervals associated with the results of machine learning models in order to know their capacity. A confidence interval provides information about the range of the expected capacity of a model to produce reliable results.

In each iteration of the bootstrap method, we have a subset of training data and a different subset of test data. This training dataset has the same size as the original dataset and has been created by randomly extracting elements one at a time with replacement, so it may contain repeated samples. Samples from the dataset that have not been chosen become part of the test set. A certain model is trained with this selection of training data and its performance on this selection of test data is evaluated with a certain statistical metric such as, for example, the proportion of right predictions in a classification model. This process is repeated M times by forming each time different training subsets with samples that may be repeated, training the corresponding model and evaluating its capacity M times in total. Thus, we finally arrive at a distribution formed by M samples whose shape we can study to determine a confidence interval at the desired percentage.

For classification models, the metric that we measure can be the accuracy and for regression models, we could take into account an error metric such as the mean squared error. In this project, we will obtain 95% confidence intervals.

5 Decision trees

Decision trees are prediction models based on the creation of diagrams which represent logical decisions. It is a supervised learning technique that takes the four variables designed above (independent variables) and a certain target variable (dependent variable). Since decision trees can be used as classifiers or as regressors, the target variable will be the goodness in its qualitative form in the first case (class 0, 1, 2, or 3), or in its quantitative form in the second case (integer goodness from 0 to 10). The effectiveness of both types of algorithms will be evaluated. In the first case, we benefit from the graphical utility of the classifiers in the case of needing only a qualitative evaluation of a work order. In the second case, we benefit from the utility of providing a numerical evaluation.

A tree has several *nodes* linked by *branches*. Each node represents a certain logical proposition and each branch represents the decision made in a certain direction. The tree starts at a *root node* and, from there, new branches appear according to the most optimal partition of the dataset. The growth of the tree ends in the so-called *leaf nodes* where no logical proposition is evaluated.

Mathematically, we have a set of training vectors $x_i \in \mathbb{R}^n,\ i = 1, \ldots, l$ where n is the number of independent variables and l is the number of training vectors. We also have a vector $y \in \mathbb{R}^l$ which defines the values of the dependent variable (label for each class in classification or numerical values of goodness for regression). The idea is to make successive partitions that group data with the same label in classification or similar numerical values of goodness in regression. Let Q_m be the dataset in node m and assume it has N_m samples. Let $\theta = (j, t_m)$ be the parameters of a split performed according to the variable j with a threshold t_m. Then, from the set Q_m of the node m, we define two new sets after the split:

$$Q_m^{left}(\theta) = \{(x_i, y_i) | x_{ij} \leq t_m\} \tag{16}$$

$$Q_m^{right}(\theta) = Q_m \backslash Q_m^{left}(\theta) \tag{17}$$

Using an impurity function (classification) or error function (regression) which we will call H, we define the function G in each node such that

$$G(Q_m, \theta) = \frac{N_m^{left}}{N_m} H(Q_m^{left}(\theta)) + \frac{N_m^{right}}{N_m} H(Q_m^{right}(\theta)) \tag{18}$$

The parameters which define a certain split are those that minimize that function:

$$\theta^* = \text{argmin}_\theta\ G(Q_m, \theta) \tag{19}$$

And the process continues generating new sets $Q_m^{left}(\theta^*)$ and $Q_m^{right}(\theta^*)$ until the leaf nodes are reached. The functions H will be explicitly shown in later sections depending on whether we want to use decision trees as classifiers or as regressors.

In either case, the first step is to make a random partition of the dataset into two subsets:

- *Training data*: The model will learn the most descriptive features from these data. From the 623 available work orders, we will take 70% for training, i.e., 436 work orders.

- *Test data*: These will be used to test the effectiveness of the model after training. From the 623 available work orders, we will take 30% to test, i.e., 187 work orders.

5.1 Decision tree classifiers

As a measure of the homogeneity of a dataset, we are going to use the *Gini impurity* which has the form:

$$H(Q_m) = 1 - \sum_k p_{mk}^2 \tag{20}$$

where p_{mk} is the proportion of work orders of class k in node m. This is a number between 0 and 1 such that the minimum value is reached when the set of work orders in node m is perfectly homogeneous (all orders belong to the same class) and the maximum value is reached when the heterogeneity is maximal.

5.1.1 Representation of a tree

The plot of one of the decision trees generated by training a classifier is shown below.

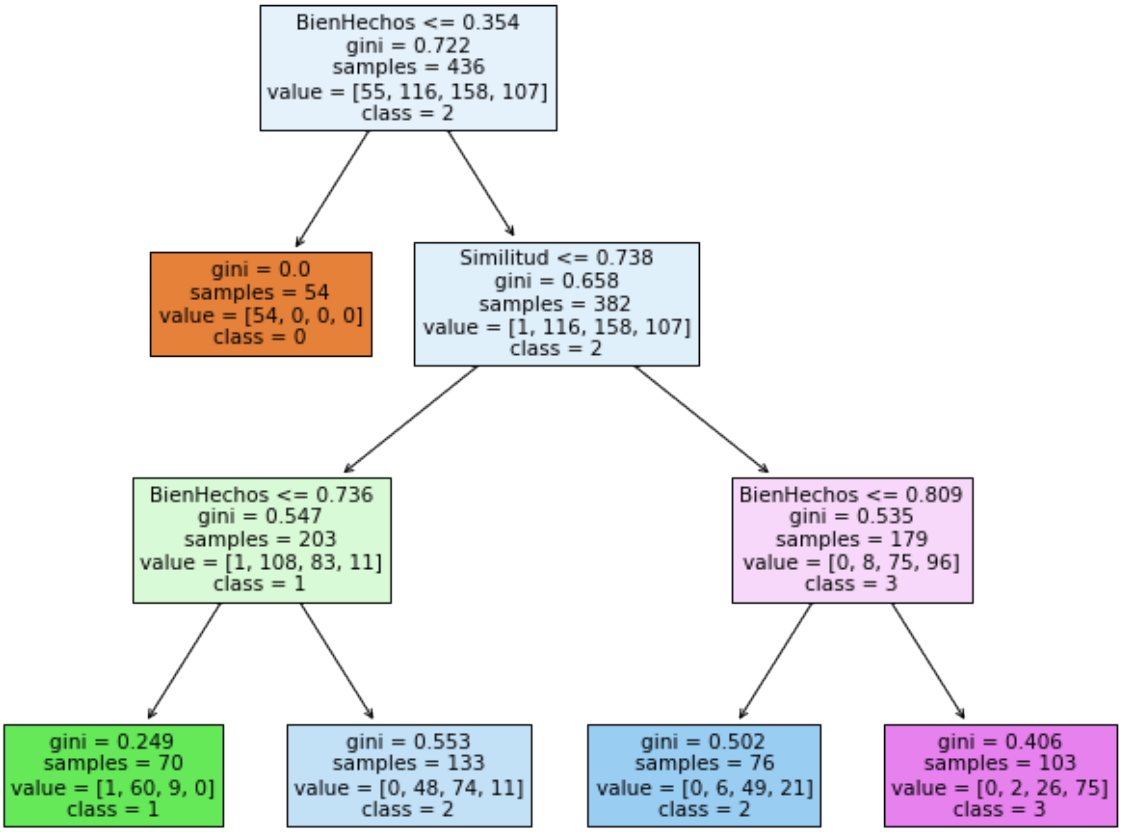

Figure 6: A decision tree with maximum depth 3.

where class 0 refers to ***bad*** orders, class 1 to ***mediocre*** orders, class 2 to ***acceptable*** orders and class 3 to ***very good*** orders.

Actually, a complete tree has numerous additional branches until the Gini impurity is zero at all leaf nodes. Moreover, if we allowed further growth, we would observe conditions depending on more variables since only two of them have appeared here. However, it is counter-productive to allow the growth of the tree up to the leaf nodes without *pruning*. This is because readability would be highly reduced and there is a risk of overfitting the model so it would not work well for other datasets apart from the training one.

In particular, from the tree in figure 6, we obtain the following information:

Conditions	**Class**
$BienHechos \leq 0'354$	0
$BienHechos \in (0'354, 0'736]$ and $Similitud \leq 0'738$	1
$BienHechos > 0'736$ and $Similitud \leq 0'738$	2
$BienHechos \in (0'354, 0'809]$ and $Similitud > 0'738$	2
$BienHechos > 0'809$ and $Similitud > 0'738$	3

Table 3: Interpretation of the decision tree.

When we make the model predict on the class to which each sample of the test set belongs, we get the following confusion matrix:

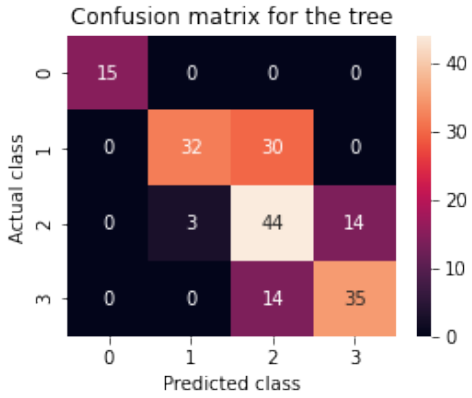

Figure 7: Confusion matrix of the decision tree comparing the classes predicted by the model and the actual classes in the subset of test data.

The number of right predictions of the model is the sum of the values in the diagonal. Therefore, the accuracy or proportion of right predictions is the trace of the matrix divided by the sum of all the entries of the matrix. In this case, the total accuracy is 67.4%. A glance at the matrix allows us to observe that the detection of work orders of class 0 (***bad***) is perfect, which is especially relevant for effectively detecting serious anomalies in certain work orders. On the contrary, we see that the model predicts too many times class 2 (***acceptable*** orders) when the work order does not actually belong to that class, although the deviations are to the neighboring classes.

We can generate as many decision trees as we want depending on the random partition of the dataset. In general, we should expect trees that provide similar information even if this information is presented in different order. However, there may be particular seeds that result

in unexpected trees with the other variables that have not appeared in the tree in figure 6. We do not have to rely on information from a single tree, so we can use other techniques such as *random forests*.

5.1.2 Random forest

This is a technique based on the construction of as many trees as desired, from randomly extracted datasets. Then, the predictions of each of the trees are grouped together to make a final prediction. That is why it belongs to the group of *ensemble* methods and can help reduce the high variance that is often present in individual decision trees. If we mantain a maximum depth of up to the third generation, the problem lies in finding the optimal number of trees that should be randomly extracted to maximize the accuracy of the model without severely increasing the computational cost.

In this case, it has been determined that, for the case of 70 estimators, an accuracy of up to 76.5% can be obtained, as can be deduced from the following confusion matrix:

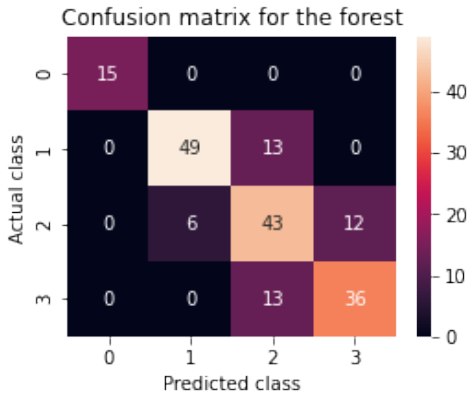

Figure 8: Confusion matrix of the random forest comparing the classes predicted by the model and the actual classes in the subset of test data.

where we observe a substantial improvement in the correct classification of class 1 orders that were previously classified in class 2 and an increase in the proportion of right predictions in orders where the model previously predicted class 2.

Let us now apply the bootstrapping method to find a 95% confidence interval. We train the model 1000 times using datasets constructed by randomly extracting the original data one by one to form datasets of the same size as the original dataset. This yields the following histogram of accuracies (proportion of right predictions of the classifier):

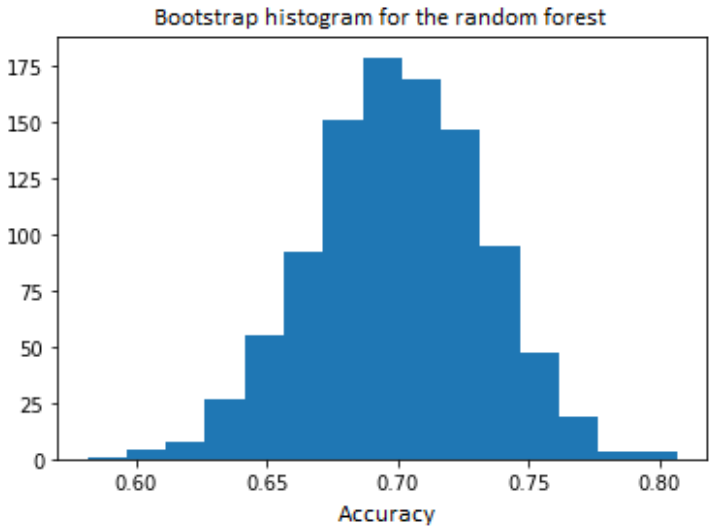

Figure 9: Distribution of accuracies for the random forest.

95% of the central values of the histogram are found in the interval $[0.635, 0.762]$, so we conclude that the actual capacity of the model lies between 63.5% and 76.2% with a confidence of 95%. As expected, this interval is below the accuracy of 76.5% obtained before because that was a maximal value, but not the most frequent for this model.

Likewise, we can calculate the feature importance of the model according to *Gini importance* taking into account how each variable contributes to the decrease of Gini impurity.

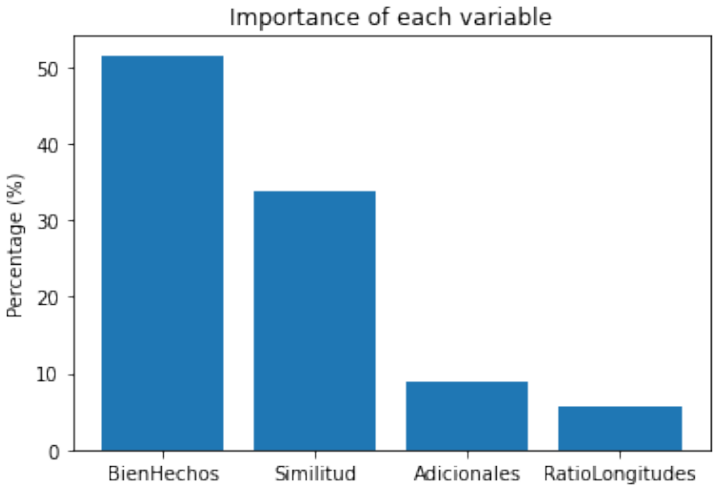

Figure 10: Feature importance in percentage.

where it can be seen that, due to the complexity of the dataset and the correlations between the variables, the percentage of importance of each variable is not necessarily proportional to the weight that we had originally assigned to it in the construction of the goodness, as expected.

We also plot the ROC curves for this classifier:

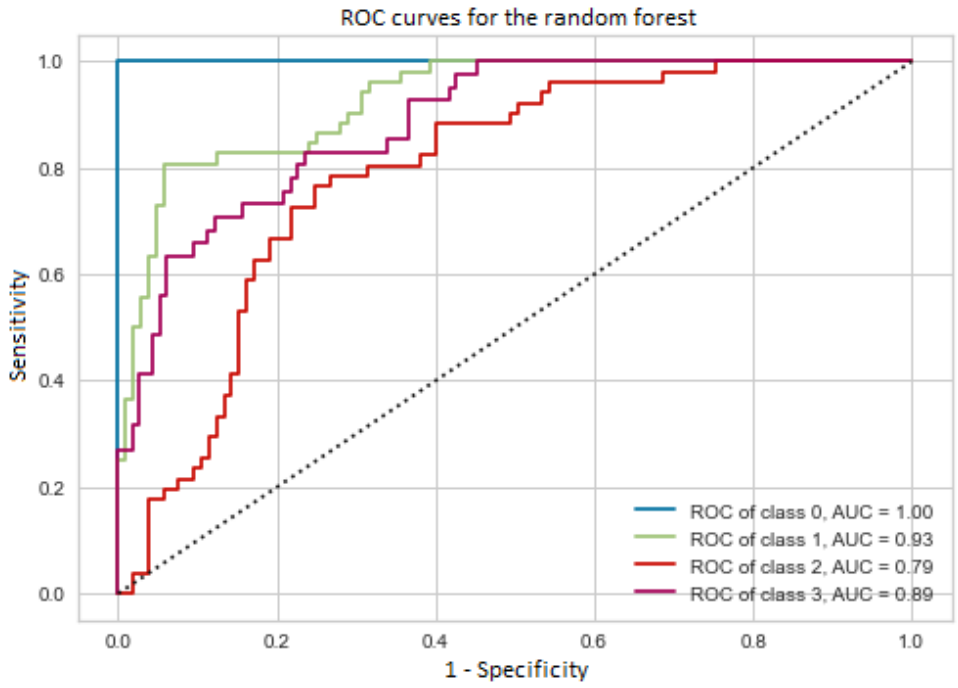

Figure 11: ROC curves for each class and area under the curve (AUC).

showing that the classification of class 0 is carried out perfectly, whereas the most problematic is class 2 because it is the closest to the diagonal and encloses the smallest area.

5.2 Decision tree regressors

In this case, the function we want to minimize in each node is the *mean squared error*:

$$H(Q_m) = \frac{1}{N_m} \sum_{y_j \in Q_m} (y_j - \overline{y}_m)^2 \tag{21}$$

where

$$\overline{y}_m = \frac{1}{N_m} \sum_{y_j \in Q_m} y_j \tag{22}$$

We will now use the goodness values in its quantitative form (integers between 0 and 10) instead of its qualitative form in 4 classes. We will directly use regression algorithms based on random forests with a selection of parameters equivalent to that of the previous section. A measure of how good the regression has been is given by the coefficient R^2 defined as

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2} \tag{23}$$

where y_i are the numerical values of the goodness of the test set, $\overline{y}$ their mean and $\hat{y}_i$ the predictions.

We obtained $R^2 = 0.8889$, close to 1, which means a reasonably good fit. Finally, we observe the similarity between the distribution of labeled integer values of goodness of the test set and

the predictions of the regressor model (non-integer goodness values, although with *bins* adapted to integers).

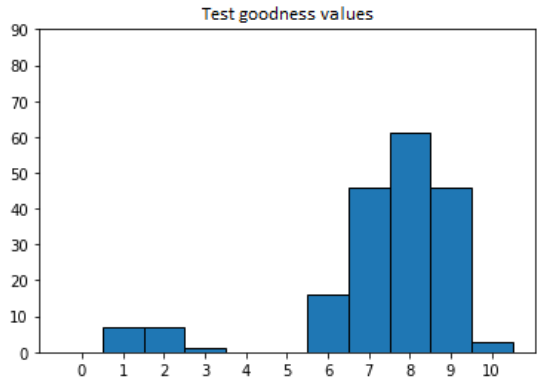

Figure 12: Labeled goodness values in the test set.

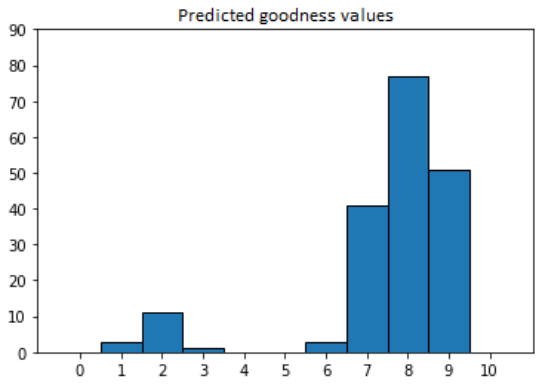

Figure 13: Predictions of the random forest regressor.

6 Neural networks

Apart from decision tree models, we will analyze the usefulness of *neural networks* for learning predictor functions. In particular, we will apply supervised neural network algorithms. We will test *multilayer perceptron* models, a neural network made up of multiple layers in which the connections between neurons do not form cycles.

First, there is an *input layer* with as many inputs as there are variables in the model. The intermediate layers are called *hidden layers* and they take the information from the previous layer and process it before transmitting it to the next layer. Finally, the information reaches the *output layer* and transforms it into the output value of our model.

Consider the first hidden layer and let i be the neuron whose dynamics we want to study. Each connection in the network will have a certain weight associated with it, so let w_{ij} be the weight associated with the connection $j \longrightarrow i$. These weights are parameters that can be adapted within the network and indicate the level of relevance assigned to the information coming from a neuron of the previous layer. The assignment of weights can be done with algorithms of *gradient descent* and *backpropagation*. Let us start with the first hidden layer.

Firstly, we calculate the variable

$$z_i^{(1)} = \sum_j w_{ij}^{(1)} x_j + b_i^{(1)} \tag{24}$$

That is, a linear combination of the input variables which is weighted with the weights and corrected with the parameter b_i which is known as *bias*.

Then, this variable passes through a certain activation function that determines the information that will be sent to the next layer. For example, for classification problems, some typical activation functions are

$$\phi_1(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \qquad \phi_2(z) = \frac{1}{1 + e^{-z}} \qquad \phi_3(z) = \max(0, z) \tag{25}$$

which are the hyperbolic tangent, the sigmoid and the ReLU (*Rectified Linear Unit*), respectively. Thus, the output of neuron i of the first hidden layer would be the activation

$$a_i^{(1)} = \phi^{(1)}\left(z_i^{(1)}\right) = \phi^{(1)}\left(\sum_j w_{ij}^{(1)} x_j + b_i^{(1)}\right) \tag{26}$$

If we assume that there are n hidden layers, the information is transmitted through the layers in a way such that neuron i from the n-th layer would transmit the following activation:

$$a_i^{(n)} = \phi^{(n)}\left(\sum_j w_{ij}^{(n)} a_j^{(n-1)} + b_i^{(n)}\right) \tag{27}$$

In general, a different function ϕ can be applied for each hidden layer. In our case, knowing the size of the dataset and the dimension of the problem, it will be enough to consider one hidden layer whose activation function will be one of those described in (25).

6.1 Neural network classifier

In our case, the aim is to create a classifier that predicts to which of the four designed classes a work order belongs. For this, we will need four neurons in the output layer. For classification problems into more than two classes, we need in the output layer a more advanced activation function that determines with enough accuracy the probability that a work order actually belongs to a certain class. This information is provided through the `Softmax` function. Given a vector $z = (z_1, \ldots, z_k)$, the `Softmax` function maps the values to the interval [0,1] obtaining the following transformed values:

$$\text{Softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \tag{28}$$

This is a generalization of the sigmoid function. The value k would be the number of classes (in our case, four) and we obtain the probability that a sample belongs to each of the four classes. In principle, the final result of the network would be the class which has the highest probability of all.

After obtaining this vector of probabilities, we define a loss function. A popular loss function is the *cross-entropy*. This function is defined as

$$H(p, q) = -\sum_k p(k) \cdot \log(q(k)) \tag{29}$$

where $p(k)$ is the probability of obtaining class k in the target variable that we provided for the training (1 or 0) and $q(k)$ is the probability of obtaining class k according to the values obtained after applying the `Softmax` function.
This loss would be 0 for a perfect prediction and infinity for a prediction that is totally opposite to the expected one. Additionally, we can add a regularization term of the form $\alpha||W||_2^2$ where $||W||_2^2$ is a squared sum of weights of the network and α is a parameter that should be chosen carefully. In this way, the large weights will have a larger penalty in the cost function if α is large and, analogously, the effect of the regularization will be smaller if α is smaller.

We will first train a multilayer perceptron through the implementation of the `scikit-learn` library. Neural networks present the difficulty of finding an optimal configuration, as there is a large number of hyperparameters that need to be tuned. We will try a different number of neurons in the hidden layer (4, 5 or 6), different activation functions (hyperbolic tangent, sigmoid or ReLU) and several regularization parameters in different orders of magnitude (10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}). Also, we can first use the `L-BFGS` solver due to the fact that the dataset is not particularly large and we expect faster convergence for sets with small size.

As we will see, neural networks applied to our dataset are more unstable than random forests in the sense that there is more variance in the results. Different runs of the same algorithm may lead to somewhat different results and will point out that the optimal network architecture may be different depending on the run. However, after an analysis of the most frequent results produced by the network, a remarkably optimal combination is 4 neurons in the hidden layer, activation function ReLU and regularization parameter 10^{-4}. With this configuration, we train the network and obtain an accuracy of 77% as we can also see in the following confusion matrix:

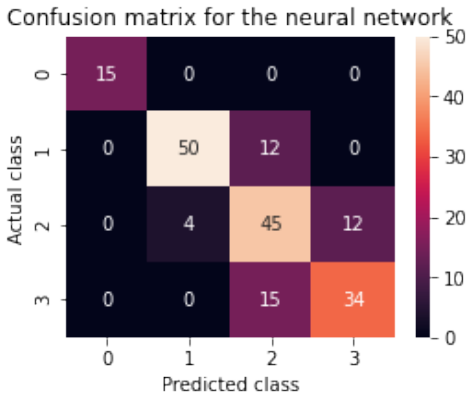

Figure 14: Confusion matrix of the neural network comparing the classes predicted by the model and the actual classes in the subset of test data.

Likewise, the histogram obtained by applying bootstrapping with 2000 trainings is

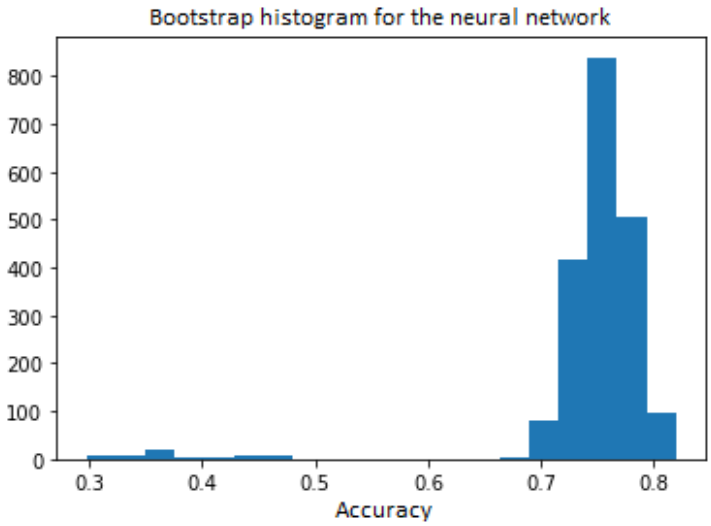

Figure 15: Distribution of accuracies for the neural network.

95% of the central accuracies are in the interval $[0.46, 0.80]$. Therefore, the actual capacity of the model is defined by a proportion of right predictions between 46% and 80% with 95% confidence. Clearly, this uncertainty is much higher than in the previously studied models since, occasionally, we could find runs that result in low accuracy due to the randomness of certain elements in the network. Even so, the histogram above tells us that the accuracy of the model will usually be between 70% and 80%.

We will also test the design of a network using `TensorFlow`. Although the dataset is not particularly large, we will test the implementation of the `Adam` solver. This requires the tuning of at least four hyperparameters described in the appendix. The quality of the results depends significantly on these hyperparameters. By testing different combinations, it can be found that a suitable combination is $\alpha = 0.07$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-7}$. The model is trained after choosing a certain *batch size* and a certain number of *epochs*. The *batch size* is the total number of training samples in each *batch*, with a *batch* being each of the groups into which the dataset is divided so we do not send all the data through the network at once. On the other hand, an epoch passes when a dataset traverses the network back and forth once. The greater the number of epochs, the more times the network parameters are updated. We have to choose values that result in a decrease in loss and an increase in accuracy at reasonable rates while maintaining approximate monotony except for small oscillations. Therefore, we have chosen 80 epochs and a batch size of 50. After training the model, we can observe the expected trends in the loss and accuracy curves by testing with the test dataset.

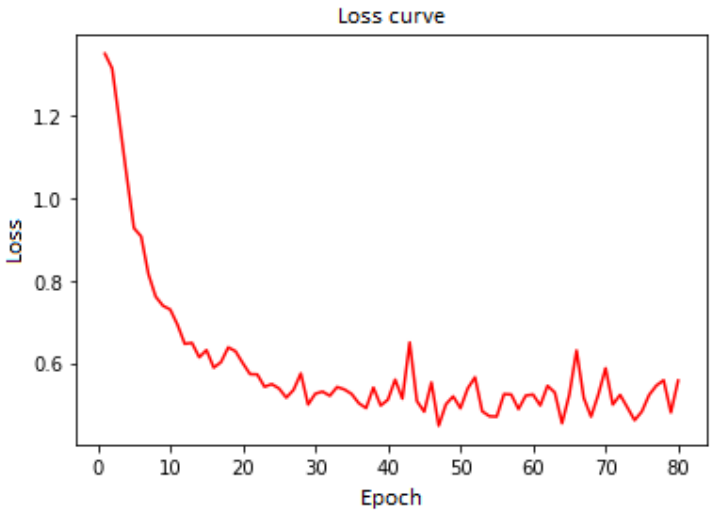

Figure 16: Loss versus epoch testing with the test dataset.

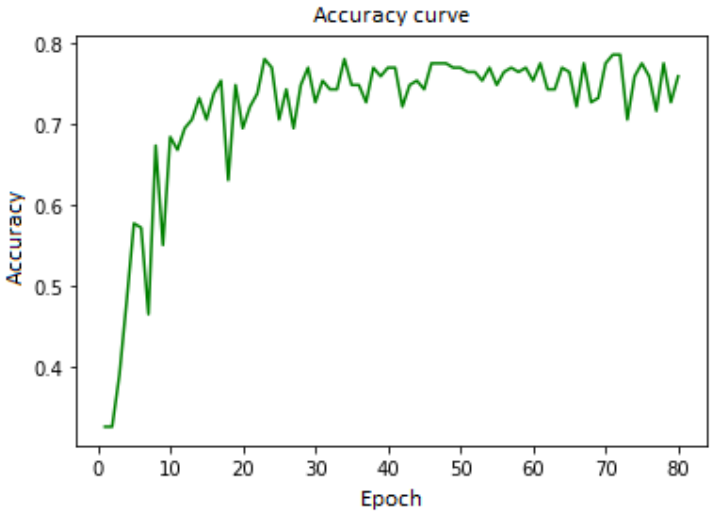

Figure 17: Accuracy versus epoch testing with the test dataset.

Indeed, the results improve over time on a dataset that has not been previously used to train the network. We see that the loss tends to decrease in an approximately monotonic way except for small oscillations and converges to a certain small value of loss. This value is not identically zero because the model has a limited capacity that could perhaps be improved by using a larger dataset. On the other hand, the accuracy tends to grow in an approximately monotonic way except for small oscillations and the accuracy to which it converges is of the same order as that of previous models.

Finally, we plot the ROC curves:

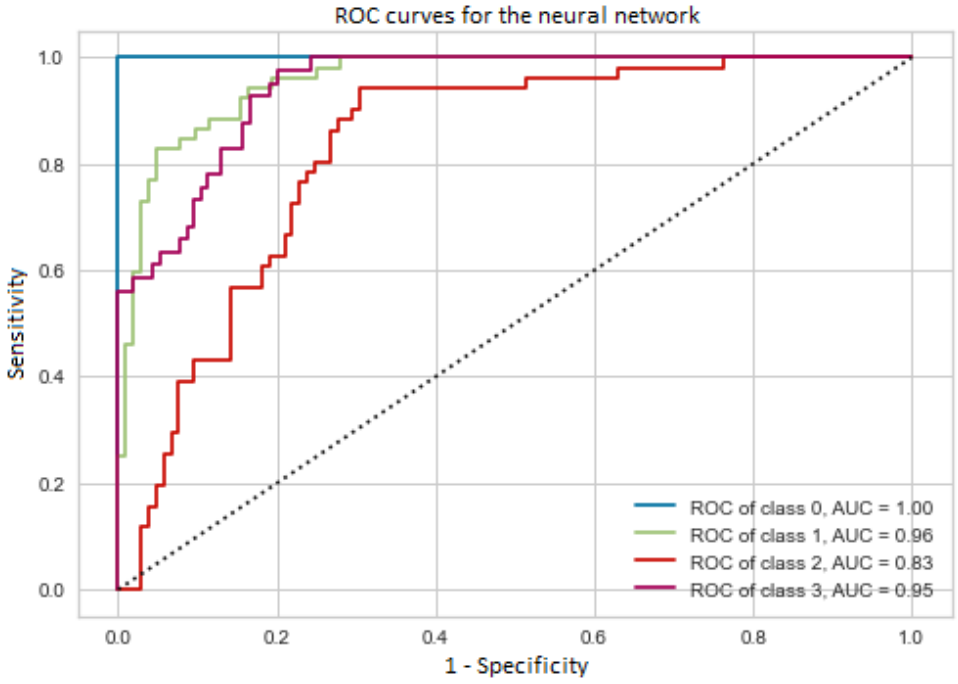

Figure 18: ROC curves for each class and area under each curve (AUC).

showing a behavior equivalent to that of the random forest, although slightly better as evidenced

by the calculation of the areas shown in the legend.

6.2 Neural network regressor

In the case of regression whose outputs are real numbers, the activation function of the output layer is directly the identity function and there will be only one neuron in that layer. Therefore, after applying the identity function, we go directly to the measurement of the loss through another function. Typically, we can use the mean squared error, or directly the half squared error (the only part that matters mathematically in the optimization process is the quadratic difference) together with a regularization term similar to the one described above in the case of classification. Thus, this cost function would be

$$J = \frac{1}{2}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 + \frac{\alpha}{2}||W||_2^2 \tag{30}$$

where $\hat{y}_i$ are the predicted values and y_i are the values that we provide.

Returning to the definition in (23), we now obtain $R^2 = 0.9359$, close to 1 and slightly more than in the case of the random forest. Comparing the integer goodness values of the test set with the predicted ones embedded in *bins* of integers, we observe a clear similarity:

Figure 19: Labeled goodness values in the test set.

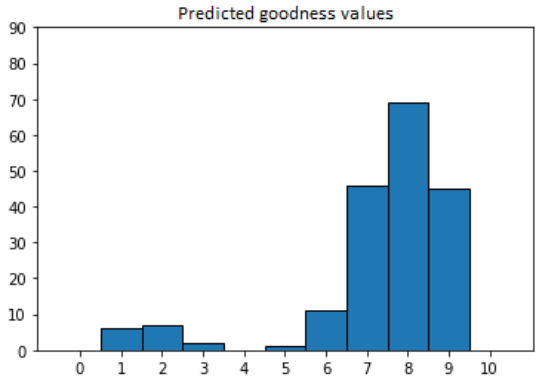

Figure 20: Predictions of the neural network regressor.

7 Conclusion

In this project, we have carried out a complete analysis of a dataset provided by the company Distromel about urban waste collection activities. These yield large amounts of data that must be processed intelligently. This includes a careful analysis of the data to understand its structure and the best ways to process them and extract relevant information. The analysis process has started with a revision of the company's databases to select a sufficiently representative dataset. After that, we calculated the most relevant variables for the determination of how well a work order is executed. Then, we built models based on artificial intelligence algorithms and analyzed the results.

The analysis of the decision trees let us observe their effectiveness when we want to obtain a partition of the dataset that is easily interpretable visually. This analysis can be further refined through random forests to obtain better classifiers and regressors, although at the cost of visual interpretability. The same is true for neural networks, which are able to approximate complex functions, albeit we lose some intuition. For this dataset, taking into account its shape and size, it has been observed that random forests and neural networks give very similar results. For the particular way of constructing the goodness of a work order that has been shown in this project, it seems that networks have offered a slightly better result. However, when the input data are given in tabular form, decision trees and random forests require a simpler training process with fewer parameters to tune than neural networks. The freedom to set more parameters can be a good thing, but it can also be a disadvantage because of the preprocessing work involved, and a neural network will not necessarily give substantially better results on all problems.

Regarding future directions, the data analysis presented here may be useful for the company to understand the structure of a typical dataset and to propose models that could be revisited in the future from a more customer-oriented point of view. Thus, this work could be a starting point toward more advanced models with larger datasets and more variables. That would make them more accurate, more objective and more robust in case we come across atypical data. On the other hand, the models presented here could also be used to create other smaller models. For example, this would be the case for the modeling of street cleaning activities that are also managed by the company. In that case, we would not have the variables of collected dumpsters, but we would have the variables related to the geometry of the paths. In this way, new models could be created by making a particularization of those presented in this project.

From the context of the Bachelor's Degree in Physics, this project has been very useful as an application of programming knowledge, advanced statistical analysis and comparison of complex models. Moreover, given the relevance of artificial intelligence and data processing in today's world, this project has been a very advantageous hands-on implementation of knowledge that will undoubtedly be useful to me in the years to come.

References

[1] Amat, J. (2020). *Machine learning con Python y Scikit-learn.* `https://www.cienciadedatos.net/documentos/py06_machine_learning_python_scikitlearn.html`

[2] Besse, P., Guillouet, B., Loubes, J.M. and Royer, F. (2015). *Review & Perspective for Distance Based Trajectory Clustering.* `https://arxiv.org/pdf/1508.04904.pdf`

[3] Breiman, L., Friedman, J., Olshen, R. and Stone, C. (1984). *Classification and Regression Trees.*

[4] Brownlee, J. (2018). *A Gentle Introduction to the Bootstrap Method.* `https://machinelearningmastery.com/a-gentle-introduction-to-the-bootstrap-method/`

[5] Géron, A. (2017). *Hands-On Machine Learning with Scikit-Learn & TensorFlow.*

[6] Grosse, R. (2019). *Lecture 5: MultiLayer Perceptrons.*
https://www.cs.toronto.edu/~mren/teach/csc411_19s/lec/lec10_notes1.pdf

[7] Hastie, T., Tibshirani, R. and Friedman, J. (2017). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction (segunda edición).*
https://web.stanford.edu/~hastie/Papers/ESLII.pdf

[8] Kuhn, M. and Johnson, K. (2013). *Applied Predictive Modeling.*

[9] Narkhede, S. (2018). *Understanding AUC - ROC Curve.*
https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

[10] *Python TensorFlow Tutorial – Build a Neural Network* (2020).
https://adventuresinmachinelearning.com/python-tensorflow-tutorial

[11] *Website of the scikit-learn library.*
https://scikit-learn.org