Universitat Politècnica de Catalunya

Facultat d'Informàtica de Barcelona

Master in Data Science

Study and analysis of discriminative models for horse survival prediction

Advanced Machine Learning

Course project

Arnau Arqué and Daniel Esquina

Professor: Lluís A. Belanche

December 22, 2022

Contents

List of Figures

List of Tables

1 Introduction

This article is the project report of the Advanced Machine Learning subject of the Master in Data Science taken at the Facultat d'Informàtica de Barcelona (UPC). In this work we have to apply different Machine Learning concepts and models studied during the course in order to solve a real-world problem. All of this will allow us to provide a final solution intended for the end user.

This first section represents the introduction to the project. Here, we will carry out a description of the work and its objectives. In addition, we will comment on the data we will work with and its main characteristics and peculiarities.

After the introduction, we will carry out a state of the art on the previous work associated with our data that we can find in the literature. Next, we will start the data exploration process, carrying out the preprocessing, feature selection and extraction, clustering and visualization of the data. Then, we will state and justify the resampling protocol we have chosen and which methods we have considered to solve the problem in question. After that, once the code has been implemented and executed, we will comment and analyze the results obtained with each methodology, as well as the best parameterization for each. We will also compare all the results and choose a final model. To finish, we will state our scientific and personal conclusions and analyze the possible extensions and known limitations of the project.

1.1 Problem description

For this course project we were required to choose a real world problem that motivated us. There was the possibility of carrying out a practical-oriented project or one with solid theoretical foundations. Given that one of the main objectives is to expand our knowledge in the field of Machine Learning, we have decided to undertake a project that combines the acquisition of theoretical and practical knowledge.

Apart from the constraints discussed above, we wanted to work with a data set that we had not worked with before. We also considered it interesting to use mixed multivariate datasets, since they would allow us to work with more methodologies than if we used data of a single type.

The dataset we chose is called *Horse Colic Data Set* and you can find it in the UCI Machine Learning Repository [1] [2] (among other repositories). It is a multivariate problem with categorical, integer and real attributes made up of a total of 368 instances and 27 attributes with missing values. The data encodes the medical conditions of horses treated in different operating rooms across Canada. Among the different variables, we can highlight the following[1]:

1. Surgery (categorical). Indicates whether the horse has been treated with surgery or not.
2. Age (categorical). Indicates whether the horse was adult (>=6 months) or young (<6 months).
5. Pulse (real). It codifies the horse's heart rate in beats per minute.
11. Pain (categorical). It is a subjective indicator of the horse's pain coded in Alert/NoPain, Depressed, Intermittent/MildPain, Intermittent/SeverePain and Continuous.
13. Abdominal distension (categorical). It indicates the horse's abdominal pain. An animal with abdominal distension usually has a lot of pain and low intestinal activity. Values can be None, Slight, Moderate or Severe.

[1] In the file `annex.pdf` you can find all the variables listed with a brief description of what they encode.

16. pH of nasogastric reflux (real). pH of the nasogastric reflux measured in the measurement specified above.

18. Abdomen (categorical). It indicates the state of the abdomen of the animal in question. It is measured in Normal, Other, FirmFecesLargeIntestine, DistendedSmallIntestine and DistendedLargeIntestine.

23. Outcome (categorical). It shows what eventually happened to the horse: Lived, Died or Euthanised.

24. Surgical lesion (categorical, binary). It tells us if the lesion or problem of the horse was surgical with possible values Yes or No.

The main objective of this problem is to try to predict the outcome of the animal's intervention based on its characteristics and the value of the medical indicators mentioned above. It will therefore be necessary to predict whether the horse has lived (outcome = Lived), died (outcome = Died) or had to be euthanised (outcome = Euthanised).

We consider that the analysis of this data set can be very interesting in multiple areas. For example, in the field of veterinary medicine, the ability to predict how animals —in this case, horses— will react to the interventions carried out on them can help to choose and personalize treatments in order to avoid the suffering of these mammals. This issue becomes paramount considering the obvious inability to communicate the ailments or sensations that animals suffer by themselves in an efficient and effective manner.

After analyzing the data set, we considered that the problem has enough potential to try to solve it with the tools we acquired during the course. The variables it includes are of various types and we believe they can be useful in determining the outcome. For example, according to the authors of the dataset, the attribute *Abdominal Distension* (item 1.1) can be of great importance, since an animal presenting this pathology is very likely to require surgery and, therefore, can be decisive for its survival.

While it is true that the problem seems interesting, it also presents some challenging characteristics that will require special attention and care. For example, we have detected that there is a significant percentage of missing values (≈30%). Therefore, it will be necessary to carefully study how we impute or treat these values. Another challenging fact is that in the description of the dataset it is indicated that some of the variables (for example, the Pain variable at item 1.1) have been compiled subjectively (*i.e.* according to the perception of the medical professional present in the intervention) given the absence of any objective method to determine their value. In conclusion, all of this will mean that we will have to carry out an exhaustive analysis of the variables in order to assess their validity or usefulness in the problem.

2 Related work

In this section we will present the different studies we have found concerning this data collection after doing some research. This will set some grounds to work on besides helping us to decide the scope of the solution to contemplate additional approaches.

In article [3], this dataset is used to run some experiments with various Nearest Neighbours approaches with LOOCV and accuracies around 71% are found for every method. Despite that, the preprocessing steps are not defined and no comment on the testing set or the target variable was found. Values of up to 73% were presented by this other article with a boosting derived

weighting also with Nearest Neighbours. Once again, the exact circumstances of the particular experiments are not described and the target variable is unknown. This is a problem because for this dataset is presented as having different possibilities of target variables.

While searching for other papers which had used this collection we faced up with the same controversy. All of them define a two-class classification task [4], [5], [6], [7] and [8]. Although not all of them specify the target variable, it can be assumed that it is the `surgical` feature from the ones that do. There are cases like [9] which yield higher results than others. However, this is a double-edge sword for our analysis because while it will allow us to contribute with something new, we won't be able to compare the results with other solutions.

We explored related work with hopes of finding some hints on how to deal with this collection in terms of best practices for classification purposes but that was not found in the aforementioned articles as their purpose was never to dive into the dataset. The next logical step was to check competition websites with this same purpose.

Luckily, we were able to find two solutions with similar approaches to ours. One of them [10] used Decision Trees and Random Forest and yielded accuracies of 67.2% and 74.6% respectively. The other solution used a SVM with a radial kernel yielding accuracies of 67.8% for the default solution and 71.1% with tuned parameters [11]. In both solutions and secondary ones with different goals very minimal work was committed to pre-processing [12] the data and attempting different approaches so we will focus on that aspect in this study.

3 Data exploration

3.1 Data visualization and Clustering

The initial part of every data mining task includes an inspection of the data. This examination should allow the analyst to gain some insight on the variables at hand and their associations in order to make better decisions.

We started this process by plotting the numerical variables with our target feature `outcome`. We can see this result in figure 1. There is no clear division between classes but we can see how they tend to behave differently. For instance, higher `totalProtein` values undoubtedly belong to horses that are either euthanized or alive.

Not much additional insight can be gained from a three dimensional plot, which we also computed. In figure 2 we can see the comparison between the labelled data on the left, with the clusters found using K-means with k=3. It is clear that the clusters found do not match any class from the data.

At this point of the study we have also investigated with higher k values in this clustering phase and some other variable analysis. Further examinations will be conducted and presented in the following section 3.2.

3.2 Pre-processing

The script for this section of the study can be found in `/code/preprocessing.ipynb`. The pre-processing step is a very important portion of every data mining project as it can deeply impact the performance of the proposed solutions. Furthermore, it is specific for each problem, which makes this task very interesting. As we have stated in the previous section 2, minimal work is done on this matter by other analysts, so we will try to dive further into this topic.

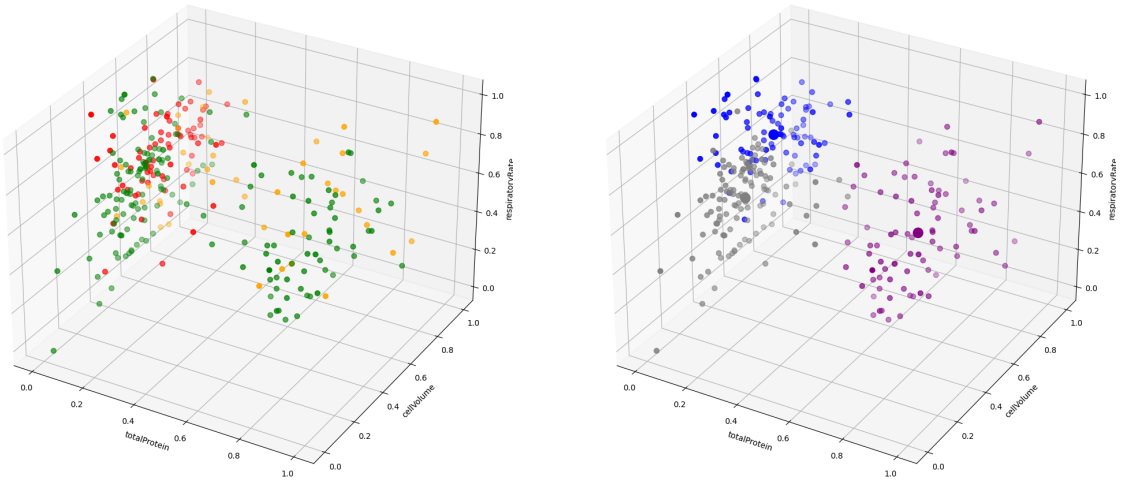
Figure 1: Numerical variable distribution comparison plots with `outcome` labels.

After loading the data, feature categories were added and a brief inspection of the data allowed us to already see some problems. We noticed there were a couple of missing labels in the target variable, one in each dataset (training and testing), which we deleted. After that, we checked for duplicate observation, which we did not have. Then, we decided to remove unneccessary variables `hospital`, `typeLesion2` and `typeLesion3`. The first one because it is an almost unique tag and the other two because the vast majority of observations have the same value (over 97%).

We carefully read the additional information provided with the dataset, which indicated us that `typeLesion` features actually coded four different features (`lesionSite`, `lesionType`, `lesionSubtype`, `lesionCode`), which we extracted. This information also helped us code the different categories of the factor variables.

The next step was dealing with outliers of the data, a treatment that we will describe in section 3.2.1. After that, a treatment for missing values, which will be discussed in section 3.2.2, was executed. Once the data was imputed, an additional refining process of multivariate outlier detection was computed, as described in section 3.2.1.

Once the data was clean, we checked the correlation between numerical variables to see if there was any highly correlated variables that we needed to examine but that was not the case. We also took a look at how the target variable was represented by the different categories of each factor and there was no clear sign of association.



(a) Target-labelled data.

(b) 3-Means clusters.

Figure 2: Three dimensional comparison of the data.

On top of that, we inspected numerical variable to see if they would benefit from some transformation in order to correct skewness/kurtosis and it will be discussed in section 3.2.3.

Finally, a feature extraction to discretize continuous data was performed and will be presented on section 3.2.4. Next, non-continuous variables were coded using one-hot encoding technique. Lastly, variables were normalized with two methods: standarization (centered around mean and one unit variance, *i.e.* $x_{\text{std}} = \frac{x-\mu}{\sigma}$) and min-max normalization (scaling of values to range zero to one, *i.e.* $x_{\text{minmax}} = \frac{x-\min(x)}{\max(x)-\min(x)}$).

Needless to say, this preprocessing was also conducted for the test data using only the information and the models trained with the training data.

3.2.1 Dealing with outliers

Before dealing with missing data we wanted to check if any observation included univariate outliers. We wanted to follow this particular order so they can be assigned as missing for a later imputation. Nonetheless, we need to keep in mind that we are going to use SVM and Random Forest to tackle this problem and these two methods are not particularly sensitive to outliers.

We decided to use the inter-quartile range technique to check for outliers. Considering knowledge acquired from previous subjects and the sparsity of this problem's data, we deemed the standard inter-quartile range factor of 3 would be fit enough to determine extreme outliers. With that factor we only consider four values as outliers in out whole data. As we are using the aforementioned methods, we will not consider mild outliers entitled to be discarded.

Two of these outliers were found in the `tempRectal` feature. Although they do not seem to be extremely unlikely data values if we take a look at the distribution in figure 3, we will decide to delete them.

The other technique we have contemplated for this study was checking multivariate outliers. Some datasets that we had previously worked on suffered from this issue which can also affect predictive performance. In order to detect them we have used Mahalanobis distance and we have

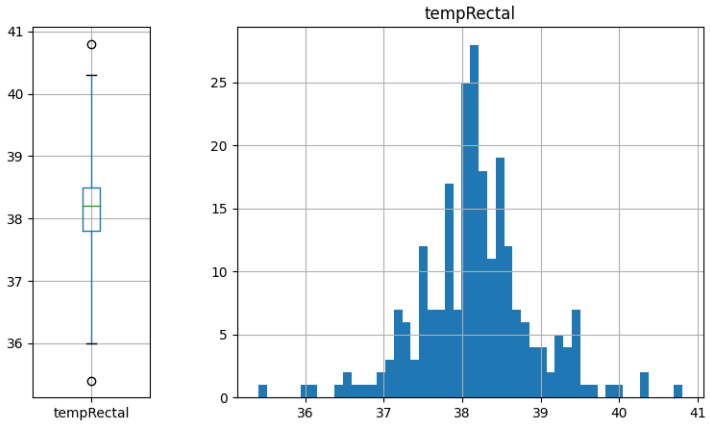


Figure 3: Boxplot and histogram of `tempRectal`.

decided to discard the ones surpassing a threshold. This threshold was established using a Chi-squared value with k-1 degrees of freedom, where k is the number of variables and a probability value of 0.95 to indentify outliers if present. In figure 4, we can see each ranked observation plotted versus their Mahalanobis distance and a red line which marks individuals to be removed (the ones with higher Mahalanobis distance than the threshold). No individual is tagged as they do not surpass the established threshold of approximately 55,76 and thus will not be discarded.

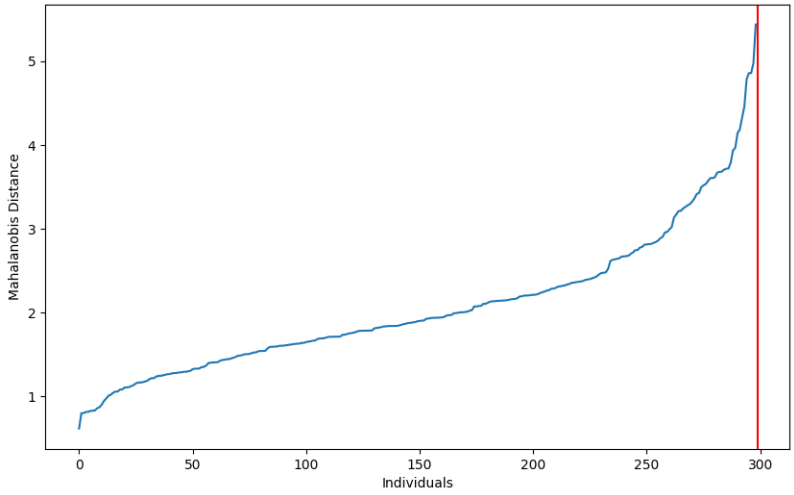


Figure 4: Plot of the individuals ranked with their Mahalanobis distance.

3.2.2 Dealing with missing data

As we have previously commented, one of the challenges of working with this dataset is the amount of missing values present. Take a look at figure 5, this basic inspection of the variables allows us to see the ratio of missing data among features. After seeing this, we made the decision to delete features with high missing data. We decided to do so because the expected distribution of this variables is previously unknown and we did not feel confident inputting most of the information. As we want to be safe, we are going to set a really high threshold of 40% missing values on the feature to be deleted. This ratio seems very high compared to previous work we have developed so, just to be safe, we will also consider a lower threshold to see if this decision pays off or if it has any impact at all. In figure 5, the dotted red line represents the 40% threshold and the three features surpassing it will be discarded from our study.

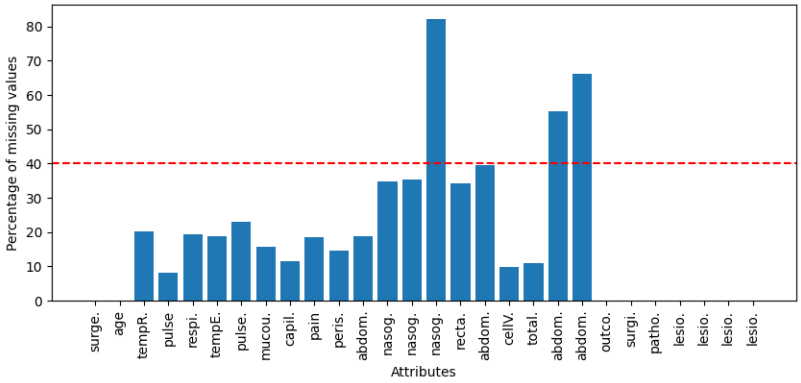


Figure 5: Missing value ratio for each attribute.

The information we are losing because of this decision is the PH of the reflux of the horse and two measures that determine if the horse's gut is compromised. Unfortunately, we have no expert knowledge on the significance of these measures for this subject. However, unlike with `abdominalDistension` feature, there was no comment on the importance of these variables for the outcome of the horse in the additional attribute information of the dataset. Once we had discarded these problematic features, we carefully inspected all of our options to deal with missing data on instances. Some of which will be discussed followingly, together with our final approach.

The first discarded thought was to delete the observations with missing values, but this would just leave us with approximately one third of the original data and we would lose meaningful information that could positively affect the task at hand. So, instead of following that path, we opted for imputing the data.

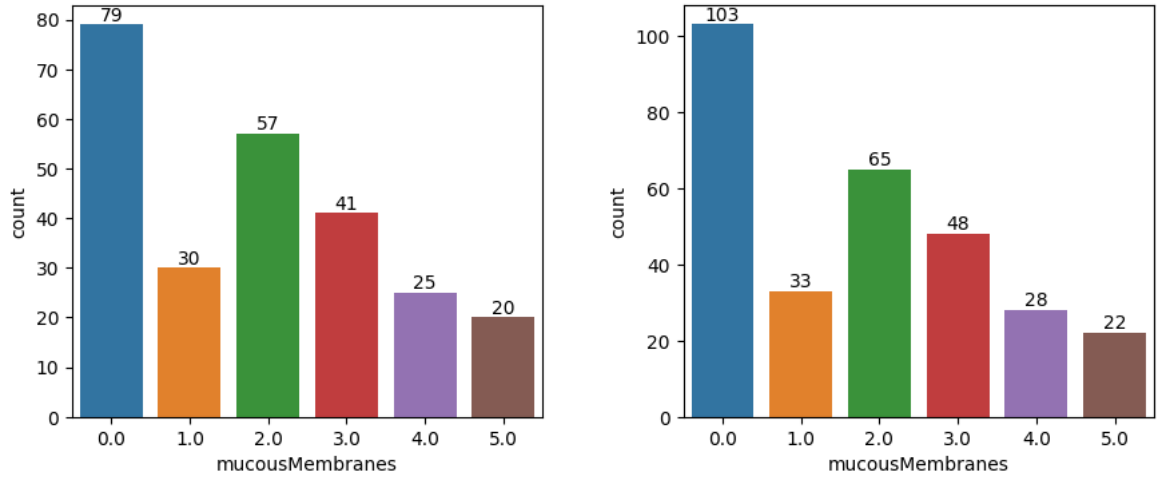
For categorical data we considered replacing absent values with the mode (the most frequent category). However, as our amount of missing values were that high, this could entail introducing bias as it not only changes the feature distribution (over-representing the mode category in our case) but also does not take into account relationships with other features. The same argument can be said for systematic random sampling, which we also took into account. Another approach we considered was creating a new category for the missing values. A good side effect of this decision is that we would not be making any assumption on the nature of the missing data (whether they are MAR, MCAR or MNAR cases). However, meaningless conclusions could be extracted from this analysis, specially with the high representation of the new category.

For numerical data, imputation with the mean or median was evidently considered, but, as we know, this messes the prior distribution. Additionally, we wanted to use a method that kept some information between variables. The first method that came to mind was KNN, this approach is very sensitive to outliers, which we do not seem to suffer from, as we can see from section 3.2.1. Nonetheless, we need to keep in mind that this algorithm would only take into account numerical data and we wanted to select a technique that would allow us to impute mixed-data types. In addition, it was a method we had previously used so we decided to do some more research on the subject. From the experimentation of this article [13], which considered all three possible missing patterns and coincidentally our 30% missing data amount, we were led to contemplate approaches based on the Random Forest method. We were considering two different options which were MissForest [14] and MICE using a Random Forest-based approach [15]. The later has a random component which translates into randomly choosing one of the predictions of the generated models, while the first one aims a selecting the best prediction. Evidently both options

have their upsides and downsides and exhaustive research on the topic could be performed. For the proposed task, we have chosen to work with the first one.

The MissForest technique works in the following way. A initialization phase replaces the missing values of the features with their mean or mode depending on the nature of the variable (continuous or categorical). Then, a imputation phase starts and builds a Random Forest model where the response variable is the feature to be imputed. Those observations where the data is present for that feature in the original dataset will become the training set while those where it was missing will become the prediction set. This is sequentially done for all variables that need imputation. Once all variables have been imputed, an iteration of this algorithm is finished. The stopping condition of these iterations is met when the difference between the new imputed data and the one from the previous iteration (or initialization) increases with respect to both variable types.

After the data imputation it is very important to check if the distributions of the features were significantly affected by this technique. In figures 6 we can see how the categorical data with imputed values on the right plot did not seem to disturb the existing distribution of the original data (left plot). The same reasoning can be reached for the numerical variables. In figures 7a and 7b we can see how the imputed data from feature `pulse` (left plot) seems to have changed minimally the distribution of the original data. Although the same can not be said for instance for `tempRectal` feature (right plot), these variations seem adequate enough considering the tremendous amount of missing data some of these variables had.

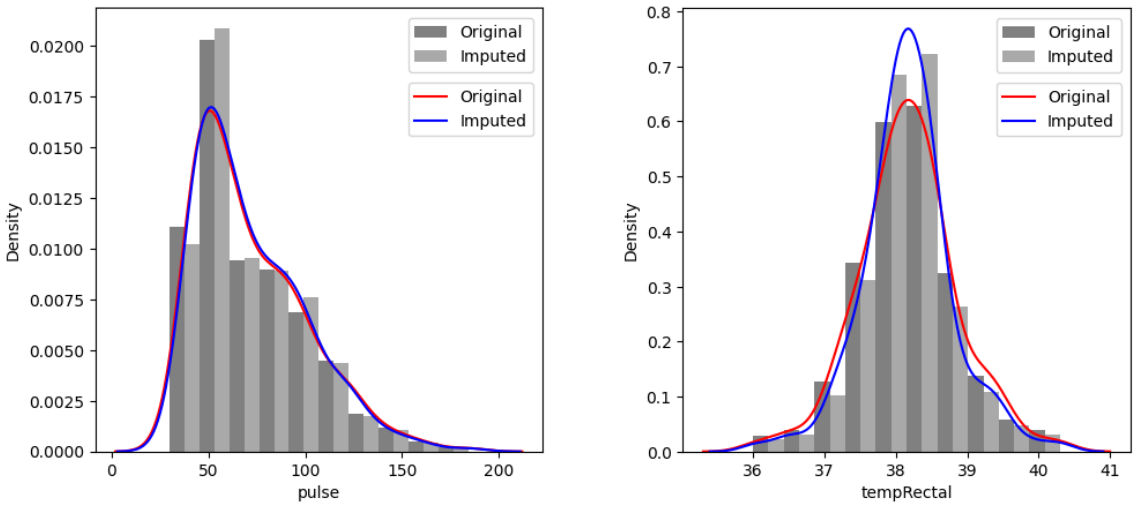


(a) Original data.

(b) Imputed data.

Figure 6: Missing data imputation distribution comparison for `mucousMembranes` variable.

When we thought about imputing data an idea came across about adding an additional binary feature for each of the original ones that captures whether data was missing on that particular observation. Although this idea was not original, we have never used it or seen its effects, so we discussed its strengths and weaknesses and after deliberation we decided to implement it. Once again, to be safe, we will save this feature on the dataset but we may decide to use it or not depending on the performance of the models. One downside of this technique is that it increases the dimensionality of the problem, which is not an issue in our case. It may also add multi-collinearity between the imputed feature and the extra one but, as we plan on using SVM and Random Forest it is not an issue either.



(a) `pulse` feature.

(b) `tempRectal` feature.

Figure 7: Missing data imputation distribution comparison for two numerical variables.

3.2.3 Variable transformation

At this point of the pre-processing task we wondered if our continuous data would benefit from any transformation. By looking at the plotted distributions of each numerical variable, we noticed some of them were asymmetric or had small tails, so we computed the skewness and kurtosis measure for each of them.

The yielded values deferred our thoughts about these variables following a normal distribution. In order to correct some of these asymmetries we tried using three different transformations and comparing their results for each of the numerical features. In figure 8 we can take a look at the four different probability plots of the `respiratoryRate` variable. In this case it is clear that the log and Box-Cox transformation seem to work better on changing this feature into a normal distribution.

Although they may seem normal, after using Shapiro-tests for every outcome, this hypothesis is rejected. Either way, the skewness and kurtosis values have changed and some features are less skewed. Upon further inspection we have decided to keep the Box-Cox transformations in two variables (`respiratoryRate` and `cellVolume`). In figures 9 we can see how the previously examined variable was changed.

3.2.4 Feature extraction

In addition to using one-hot encoding on categorical data to transform them into numerical one, we will extract categorical features from out already existing continuous data. To do this we will use a K-means approach to discretize each of the variables with a set K of three. We have chosen this value as it has yielded good results for us in previous projects and although each problem is different, the point of this study is not to find an optimum of categories that properly represent our data. We also considered using knowledge on the subject to create these categories, as the expected normal values of many measures are described in additional information of the dataset. However, this declared normal values do not seem to adjust to our data, a reasonable possibility for this deviation could be that at the time of measuring the horses were not in standard conditions (for instance, the pulse would inevitable be higher because of stress).

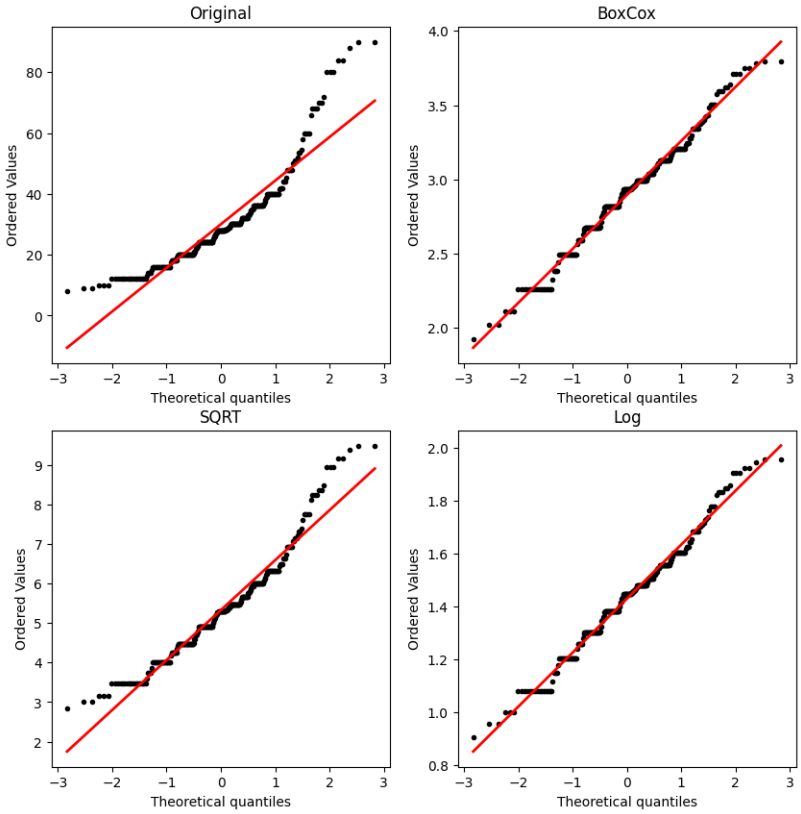


Figure 8: Probability plots for different transformations of `respiratoryRate` feature.

On figure 10 we can see on the left plot the original data of the `pulse` feature which will be categorized into three categories that can be seen on the right plot following the aforementioned methodology.

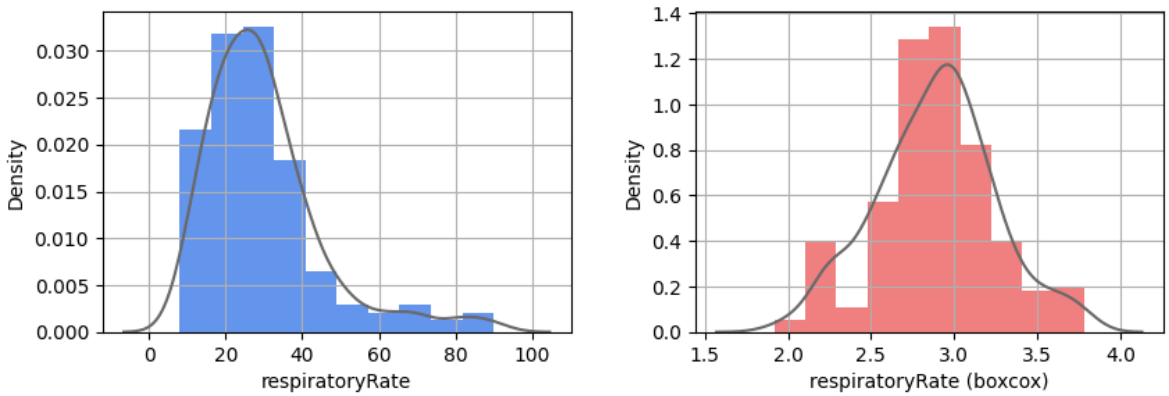
4 Experimentation methodology

In this section we will comment on the experimentation methodology that we will follow in order to solve the problem that we have stated in previous sections. At this point, we have already been able to work in depth with the data and to know its peculiarities, as well as some of its advantages and disadvantages. Now, we will comment on the methods considered to solve the problem. We will also study which is the most suitable resampling protocol in order to be able to analyze and compare the results.

4.1 Methods considered

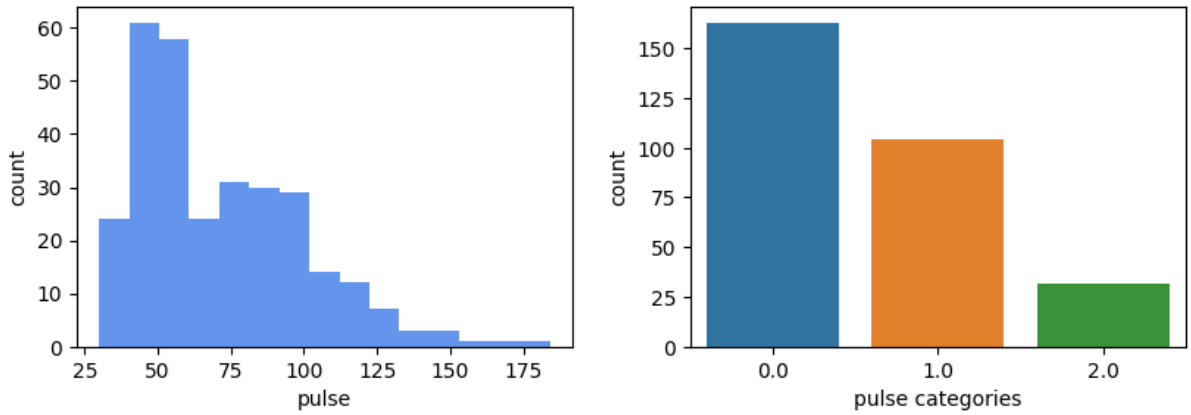
In this project we have decided to limit the scope of study to discriminative models. These types of models can be used to solve both classification and regression problems. Unlike the generative models, the discriminative ones try to model the decision boundaries between the classes of the data. In practice, the goal of these models is to learn the conditional probability distribution $P(y|x)$, where y are the data labels (classes) and x are the training data. In this way, when analyzing unobserved variables, they can use this distribution to determine whether they belong to one class or another.

In order to solve the problem we have carried out a preliminary study of the different discriminative models found in the literature. Finally, we have chosen a subset of methods that we found interesting and that have distinct characteristics and peculiarities. We believe that by



(a) Original data. (b) Transformed data.

Figure 9: Data transformation for the `respiratoryRate` feature.



(a) Original continuous data. (b) Extracted categorical data.

Figure 10: Discretization of the `pulse` feature.

following this procedure we will be able to further deepen our knowledge in the field of study of this project.

4.1.1 Support Vector Machines

Support Vector Machines (SVM) are a set of algorithms based on supervised learning that allow us to solve classification and regression problems by analyzing a set of data. The general idea of how SVMs work is based on identifying the optimal hyperplane that separates the data and that is also equidistant from the closest examples of each class. In this way, the margin on each side of the hyperplane is maximized. The samples that define this margin are called *support vectors*.

A peculiarity of SVMs is the fact that they use what are called *kernel functions* in order to separate the data. In many cases, the original dimensionality of the data does not allow discerning a hyperplane that clearly separates them. Kernel functions facilitate the detection of these hyperplanes in a space of higher dimensionality than the original. This new space is called the *feature space*. Kernel functions can also be interpreted as a measure of similarity between individuals in our data.

In the current literature we find multiple kernel functions for different types of data. However, it is difficult to find one for mixed data such as we have in our problem. One of the most used functions is the Radial Basis Function, which is defined as:

$$k_{rbf}(x, x') = \exp(-\gamma||x - x'||^2) \tag{1}$$

Where $x, z \in \mathcal{X}$ and $\gamma = \frac{1}{2\sigma^2}$. This function uses the squared Euclidean distance to determine the similarity between two individuals. A curiosity related to the RBF kernel is that its feature space has an infinite number of dimensions [16]. However, since —thanks to the kernel trick— we do not need to explicitly visit this infinite-dimensional space, we can simply use the expression of the RBF function to compute the similarity between individuals.

One of the limitations of the RBF kernel is that it can only deal with numeric data. We can use it to make predictions as long as we use the numeric and numerized —in preprocessing stage— categorical variables of our data. However, there are other types of kernel functions that would allow us to work with mixed data. For example, the aggregation kernel function:

$$k_{inner}(x, x') = \frac{1}{d}\sum_{i=1}^{d} k_i(x_i, x'_i) \tag{2}$$

$$k_{aggr}(x, x') = \frac{\exp(\gamma \cdot k_{inner}(x, x')) - 1}{\exp(\gamma) - 1} \tag{3}$$

Where $\gamma \in \mathbb{R}^+$ is a hyperparameter that allows us to adjust to what extent the training samples influence the generation of the hyperplane that separates the data. The formula of equation 3 is actually a kernel function function since the arithmetic mean of kernel functions is also a kernel[2].

Considering the performance and characteristics of these algorithms, we have considered it appropriate to employ SVMs to provide a further solution to our problem. More specifically, we will use two different approaches: first, we will use the numerical data (original and numerized categorical variables with one-hot encoding, see section 3.2) and the RBF kernel. Secondly we will use the original mixed data using the aggregation kernel and the following kernels:

$$k_{uni}(x_i, x'_i) = \begin{cases} h_\alpha(P_Z(x_i)) & \text{if } x_i = x'_i \\ 0 & \text{if } x_i \neq x'_i \end{cases} \tag{4}$$

This kernel function is called Univariate and is useful for determining the similarity between two values of the same categorical variable Z. $P_Z(x_i)$ is a probability function that indicates the probability that the categorical variable Z takes value x_i. Finally, $h_\alpha(x) = (1 - x^\alpha)^{1/\alpha}$, P_Z is an inversion function with a hyperparameter α. The effect of this function is to introduce a nonlinear behavior to the computation of the similarity index.

$$k_{jac}(x, x') = \frac{N_{11}}{N_{01} + N_{10} + N_{11}} \tag{5}$$

This kernel function (equation 5) is based on the Jaccard similarity index and is meant to evaluate the similarity between two vectors (instances) x, x' of binary data. The notation used defines $N_{01} \equiv$ Num. of attributes with value 0 in x and value 1 in x', $N_{10} \equiv$ Num. of attributes with value 1 in x and value 0 in x' and $N_{11} \equiv$ Num. of attributes with value 1 in both x and x'.

These functions (equations 4 and 5), together with that of the RBF kernel (equation 1), will be used as k_i kernel functions in the *inner* kernel defined in the equation 2. However, they will be applied depending on the domain of our data (*i.e.* for example, the jaccard kernel will only be applied on the subset of binary variables, while the univariate kernel will be applied on each categorical variable of the instances).

[2]We will not prove this statement as we consider it to be outside the scope of the project.

4.1.2 KNN

The KNN is one of the most known methods of machine learning. It is founded on the principle of proximity of Gestalt which enunciates that elements that are closer together are perceived to be more related than elements that are farther apart. This is a particular assumption of this model and there are different approaches to calculate this proximity that we will have to specify as a hyperparameter of the model.

To calculate an observation label, the distance between all training points is calculated and the mode of the k closest observations will suffice for prediction.

4.1.3 Logistic Regression

Contrary to what its name suggests, Logistic Regression is a classification algorithm that measures the relationship between a dependent variable (also known as target) and other independent ones. As we have seen in class it aims at minimizing the following log loss function 7.

$$\hat{p}(X_i) = expit(X_i w + w_0) = \frac{1}{1 + exp(-X_i w - w_0} \tag{6}$$

$$\min_w C \sum_{i=1}^{n} (-y_i log(\hat{p}(X_i)) - (1 - y_i) log(1 - \hat{p}(X_i))) + r(w) \tag{7}$$

Where y_i is the target which takes a value zero or one for a data point i and $r(w)$ is the regularization term 8 (we will only consider l1 and l2 for this task as we are already familiar with them). This classifier has multiple assumptions like the lack of multicollinearity, the independent variables being linearly related to the defined logit of the response variable or the independence between observations.

$$l_1 = ||w||_1 \qquad l_2 = \frac{1}{2}||w||_2^2 = \frac{1}{2} w^T w \tag{8}$$

In this method, a Sigmoid function is used to assign data to a class and the hyperplane which separates the two categories is used as a decision boundary. However, as in our case we are training a multiclass task the classifier will need some modifications. In the implementation we will are going to use, the minimization problem will change to 10 (probability for each class is computed) if the optimization algorithm used is `lbfgs`. If the solver is `liblinear` instead, a one-versus-rest implementation approach is executed (fitting one classifier per class) [17].

$$\hat{p}_k(X_i) = \frac{exp(X_i W_k + W_{0,k}}{\sum_{l=0}^{K-1}} \tag{9}$$

$$\min_W -C \sum_{i=1}^{n} \sum_{k=0}^{K-1} [y_i = k] log(\hat{p}_k(X_i)) + r(W) \tag{10}$$

Where $y_i \in 1, ..., K$ is the enconded label of the target variable for observation i and the matrix of coefficients W, the rows of which (W_k) correspond to class k. The expression $[P]$ evaluates to 0 if P is false and 1 otherwise while $r(W)$ is the regularization term once again 11.

$$l_1 = ||W||_{1,1} = \sum_{i=1}^{n} \sum_{j=1}^{K} |W_{i,j}| \qquad l_2 = \frac{1}{2}||W||_F^2 = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{K} W_{i,j}^2 \tag{11}$$

4.1.4 Decision Tree & Random Forest

The goal of a Decision Tree is the same as any other classification method: to create a model that predicts the value of a target variable. In particular, this method accomplishes it by learning simple decision rules inferred from the data features.

The equations 13 are the loss functions implemented in the used library [18] to compute the quality of a candidate split $\theta = (j, t_m)$ consisting of a feature j and threshold t_m at node m with data Q_m with n_m samples. Two partitions (equations 12) are computed. The algorithm will choose the arguments that minimize the the aforementioned loss function and it will be recursively computed updating the next θ as θ^* until a depth stoping condition.

$$Q_m^{left}(\theta) = \{(x, y)|x_j \leq t_m\}, \qquad Q_m^{right}(\theta) = Q_m Q_m^{left}(\theta), \qquad \theta^* = argmin_\theta G(Q_m, \theta) \tag{12}$$

$$Gini : H(Q_m) = \sum_k p_{mk}(1 - p_{mk}) \qquad Entropy : H(Q_m) = -\sum_k p_{mk} log(p_{mk}) \tag{13}$$

$$G(Q_m, \theta) = \frac{n_m^{left}}{n_m} H(Q_m^{left}(\theta)) + \frac{n_m^{right}}{n_m} H(Q_m^{right}(\theta)), \qquad p_{mk} = \frac{1}{n_m} \sum_{y \in Q_m} I(y = k) \tag{14}$$

Where $x_i \in R^n, i = 1, ..., l$ are the training vector with label $y \in R^l$, the target take values $0, 1, ..., k - 1$ for node m.

Random Forests is an ensemble method and thus its purpose is to combine predictions of some estimators in order to improve generalizability over a single one. In this particular method an ensemble of trees from a sample with replacement (extracted from the training set) is built. In addition, the best split (found by the tree) may be chosen by a random subset of features.

The implementation we are using [19] averages the probabilistic prediction of classifiers instead of the original approach of single-class voting.

4.2 Resampling protocol

At this point we have developed a state of the art of the considered methods established in previous sections. In addition, we have justified its choice and studied its main characteristics. However, given that we want to apply all them to our data set, it is necessary that we establish a resampling protocol.

For each considered method, we will have a set of parameters or hyperparameters that must be estimated before carrying out the training process. To do this, we will establish —separately, for each model— a set Q of values that we want to test for each parameter $i = 1, \ldots, n$ that we want to estimate. Therefore, we will have $Q_{1\ldots n}$ different sets of parameterization values. For each combination of the values of these sets we will carry out a k-fold Cross Validation process (preferably, with $k \geq 10$). As a way to analyze the performance of the parameterization, we will compute the average validation error obtained in the consecutive iterations of CV and select the parameterization that gives us the minimum error.

In order to carry out the performance comparison between the different models we will use the accuracy metric (*i.e.* $\frac{TP+TN}{TP+TN+FP+FN}$). We know that accuracy is not a good measure for

classification, especially with unbalanced data but we will use it in order to be able to compare our performance to other solutions. However, we will also consider other relevant metrics, such as macro average F1 score, in order to properly undertand our models behaviour.

5 Results

At this point, we have already implemented and executed the code of each of the methods described in previous sections. In this section we will analyze the results obtained for each method and make a brief discussion of them.

5.1 KNN

In `./code/knn-classification.ipynb`, the code for the experimentation with these models can be found. As KNN computes distances we have decided to use both normalized versions of the dataset. In addition, we will try with different input data to see if there is a noticeable difference between using all numeric variables, a reduced subset where high missing percentage (more than 25%) features are removed and a minimal subset where 5 continuous variables are used.

Table 1: KNN best parameter solutions.

Features	Scaling	n_neighbours	metric	algorithm	weights	Test Accuracy
Numeric	MinMax	10	manhattan	ball_tree	distance	73.13%
Numeric	Std	6	manhattan	ball_tree	distance	70.15%
Reduced	MinMax	15	minkowski	ball_tree	distance	68.66%
Reduced	Std	15	minkowski	ball_tree	distance	70.15%
Minimal	MinMax	10	minkowski	ball_tree	distance	71.64%
Minimal	Std	6	minkowski	ball_tree	distance	68.66%

On table 1 we can see the best hyperparameters found for every considered combination. The `n_neighbours` sets the k (number of clusters) in the algorithm, which is specified with the `algorithm` parameter. `metric` specifies which metric to compute distance with. The value `minkowski` refers to the standard Euclidean distance $||x_a, x_b|| = sqrt(|x_{a1} - x_{b1}|^2 + |x_{a2} - x_{b2}|^2)$ while `manhattan` computes the distance $||x_a, x_b|| = |x_{a1} - x_{b1}| + |x_{a2} - x_{b2}|$. `weigths` specifies the weights to use, in the case of the value `distance`, the inverse of the distance is used (closer neighbours have greater influence).

The best results for classifying the given test dataset were found using all numerical features (including one-hot encoding with the categorical ones as specified in 3.2) and the MinMax normalization. Nonetheless, the difference in accuracies found does not make us believe that there is huge impact on the provided data. It is interesting to see how the classification with the minimal dataset (features: `tempRectal`, `pulse`, `respiratoryRate`, `cellVolume` and `totalProtein`) yields competent results.

5.2 Logistic Regression

In `./code/lr-classification.ipynb`, the code for the experimentation with these models can be found. This time, we have used numerical variables but we decided not to include the extra ones that determined if a particular feature was missing on the original data. Furthermore, we

will use the different generated normalized datasets with the commented modifications from last section and expose the results for all combinations.

On table 2 the best hyperparameters found for these combinations following our methodology are presented. The parameter `penalty` specifies which regularization penalty to use in the regularizer `l1` for the absolute value of the coefficient's magnitude or `l2` for its squared counterpart. The parameter `C` weights this mentioned regularization (smaller values for stronger regularization). The `class_weight` argument specifies the weights to use for the metric computation. With value `None`, weights of one are used while with value `balanced` the weight $n\_samples/(n\_classes * np.bincount(y))$ is computed to adjust them inversely proportional to class frequencies. `solver` value determines which algorithm to use for the optimization problem.

Table 2: Logistic Regression best parameter solutions.

Features	Scaling	penalty	C	class_weight	solver	Test Accuracy
Numeric	MinMax	l1	1	balanced	liblinear	65.67%
Numeric	Std	l2	0.001	balanced	lbfgs	67.16%
Reduced	MinMax	l1	0.1	balanced	liblinear	67.16%
Reduced	Std	l2	0.01	balanced	liblinear	74.62%
Minimal	MinMax	l2	1	balanced	liblinear	71.64%
Minimal	Std	l2	0.01	balanced	liblinear	73.13%

'missing' variables not included in this analysis as an attempt to exclude multicollinearity.

From table 2 we can see how the best accuracy value for the test data of this problem is found when using a reduced subset with standarization. From the results obtained it seems reasonable to believe that including a larger set of features might perform slighly worse for this classification task where the test set is given.

5.3 Decision Tree & Random Forest

On `./code/dt-rf-classification.ipynb`, the code for the experimentation with these models can be found. For the classification task regarding these methods we have used the non-normalized version of the dataset because no distances need to be computed this time. Nonetheless, we have used two different datasets once again to check if there is any noticeable impact about using features with highly imputed data.

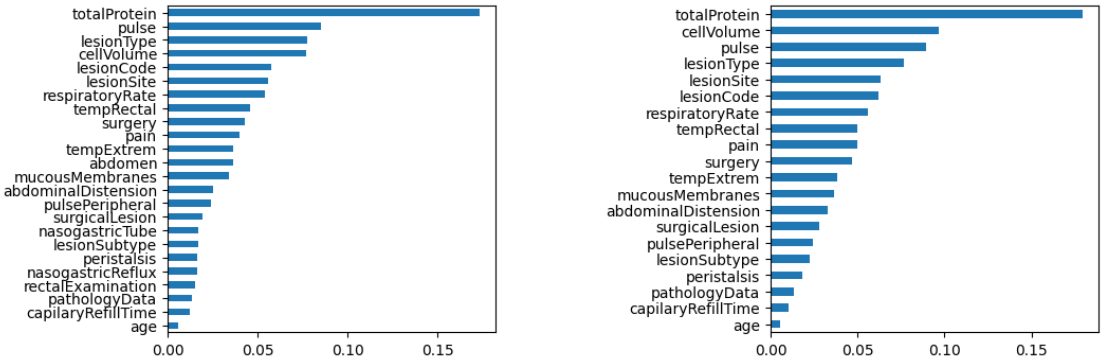
Table 3: Decision Trees and Random Forest best parameter solutions.

Model	Features	trees	crit	md	mf	msl	mss	weights	Test Accuracy
RF	All	100	gini	None	5	2	3	balanced	73.13%
RF	Reduced	50	gini	150	log2	2	5	balanced	71.64%
DT	All	-	gini	61	None	25	11	None	73.13%
DT	Reduced	-	gini	51	None	25	19	None	73.13%

On table 3 we can see the four proposed solutions with tunned hyperparameters using the described methodology. The parameter `trees` refers to the number trees in the forest while the parameter `mf` indicates the maximum number of features to consider for selecting the best split.

`crit` specifies the function to measure the quality of a split. `md` defines the maximum depth of the tree and the value `None` forces it to expand nodes until all leaves are pure or they contain less than `mss` samples (the parameter which determines the amount of samples to split an internal node. `msl` sets the minimum limit of samples a node needs to be a leaf. Lastly, the `weights` parameter sets the weights as commented in section 5.2.

From the results of table 3 it seems reasonable to assume that once again there is no noticeable difference between using all considered variables on the preprocessing phase or using a subset of the ones with less amount of missings. In figures 11 we can take a look at the features deemed most important by the Random Forest executions and check that they are very similar regardless of the dataset used.



(a) All dataset.

(b) Reduced dataset.

Figure 11: Ranked feature importance for the Random Forest.

5.4 Support Vector Machines

As we discussed in later sections, we will use two different SVM approaches. First, we will train the SVM using the numeric variables (original and categorical ones numerized by means of the one-hot encoding technique, NUM) and the RBF kernel. Secondly, we will use the aggregate kernel to work directly with the original data (MIXED). In both approaches, we will us a C-SVM. You can find the code associated with this section in the script `./code/svm.Rmd`.

The programming language we have chosen for the implementation is R. We have considered it suitable, since it has several libraries to work with SVM that are also widely used in the literature. More specifically, we will use the `kernlab` library. The main disadvantage is that it does not have the aggregation kernel implemented by default. Nonetheless, we have implemented it through several objects of the `kernel` class provided by the same library. Due to space limitations in this report, we cannot comment in depth on the implementation of these kernels, but we encourage the reader to take a look at the code of the kernel function located in the script mentioned in previous paragraphs.

In order to estimate the parameters of the first approach (*i.e.* $\gamma \in (0, 1]$ of the RBF kernel) we used 10-fold Cross Validation (CV) using the training set and the values $\gamma \in \{0.0001 : 1 : 0.05\}$[3]. We have also estimated the parameter C of the C-SVM with values $C \in \{10^i \mid i \in -2 : 5 : 0.5\}$ at the same time as γ (*i.e.* with each combination of $\gamma \times C$). In this approach we used two different numerical subsets of data: (1) without considering the *missings* variables (NUM) and (2) considering them (NUM-MISS).

[3] $\{i : j : k\}$ indicates a sequence of values from i to j by sets of k units.

In the case of the second approach, we had to estimate the γ_{aggr} parameter of the aggregate kernel and the C parameter of the C-SVM. For efficiency reasons we have decided to set the parameters $\gamma = 1$ and $\alpha = 1$ of the RBF and univariate kernel, respectively. To estimate γ_{aggr} we created a list containing the values of the squared distances (in feature space, $||\Phi(x) - \Phi(x')||^2 = 2 \cdot (1 - k_{inner}(x, x'))$) between all the individuals and we sorted it in increasing order. Then, we took the first and third quantile of the list and set $\gamma = \frac{Q_1+Q_3}{2}$. Finally, in order to estimate the value of C we used 5-fold CV and the values $C \in \{10^i \mid i \in -1 : 1 : 0.5\}$.

You can find the results obtained in the parameter estimation and the subsequent test accuracy in table 4. After analyzing the results obtained, we can draw some preliminary conclusions. First, we see that the best option seems to be to use the RBF kernel with the *missing* and one-hot encoded variables included and with a parametrization of $\gamma = 0.1001$ and $C = 3.1623$.

The worst solution has been the one obtained with the aggregation kernel. It is worth saying that this result has surprised us in many ways. We consider that —under normal conditions— the aggregation kernel should perform better than the RBF, since it uses custom similarity measures for each type of variable. Nevertheless, we had many problems with the training execution time (for example, estimating the value of C took $\approx$ 2hrs.). This is why it has been very difficult for us to calibrate the parameters of this kernel, which we believe has had a significant impact on the quality of the result. Furthermore, —as we discussed before, for simplicity— we set the values of $\gamma_{rbf} = \alpha_{uni} = 1$. It seems reasonable to think that if we had estimated them (with a more efficient algorithm than the one provided by `kernlab`) we could have improved the accuracy value.

Table 4: SVM best parameter solutions.

Data	Scaling	Kernel	γ_{rbf}	α_{uni}	γ_{aggr}	C	TR Accuracy (%)	TE Accuracy (%)
NUM	Std	k_{rbf}	0.1001	–	–	3.1623	76.25	70.15
NUM-MISS	Std	k_{rbf}	0.1501	–	–	3.1623	74.92	73.15
MIXED	Std	k_{aggr}	1.0000	1.0000	0.8476	3.1623	–	64.18

6 Final model chosen

During the experimentation stage we ran the KNN, Logistic Regression (LR), Decision Trees (DT), Random Forests (RF) and Support Vector Machines (SVM) models. The best accuracies obtained for each method were, respectively, 73.13%, 74.62%, 73.13%, 73.13% and 73.15%.

It is not difficult to realize that all the discriminative models mentioned above have obtained very similar metrics. In spite of that, the one which has provided us with a better performance in terms of the accuracy for the given test observations is the LR model. That is why we have decided to select it as the final chosen model. Next, we will analyze in depth the parameterization used and how the results were obtained.

The model has been generated from the Reduced Standardized dataset. In other words, the data has been normalized based on the standardization method and does not include those variables that, in the original dataset, exceeded 25% of missing values. We used as regularization term (penalty, equation 8) `l2`. The cost parameter (C) is 0.01, we have set a `balanced` class weight and the `liblinear` solver algorithm. This algorithm, as we discussed previously in section 4.1.3, follows a one-versus-rest approach.

As neither the context nor the additional information provided did not specify the relevance of each class' predictions, we have made no assumptions about it and we have decided to stick with the proposed accuracy metric. The results obtained were an accuracy of 74.63% and a macro average F1 score of 61.8%.

As we can see in the confusion matrix (table 5), the model has classified significantly well the cases of horses that have lived (82.99%). However, the result has not been that satisfactory in the cases of classifying whether it dies (66.67%) or, even more, is euthanized (37.5%). This behavior is not irrational considering that the Died and Euthanised classes are underrepresented.

Table 5: Confusion matrix of the results obtained with the best model chosen.

	Died	Euthanised	Lived	Total
Died	8	1	3	12
Euthanised	1	3	4	8
Lived	5	3	39	47
Total	14	7	46	–

Colums ≡ Predicted, Rows ≡ True .

7 Conclusions

During this project we have been working with different discriminative models such as KNN, Random Forests or Support Vector Machines, among others. We started by carrying out an introduction and a state of the art towards the previous work related to the project and the dataset chosen. After that, we performed a data exploration process in which we preprocessed, visualized and applied clustering techniques to our data, among other procedures. Then, we performed an experimentation phase by making use of the different methods mentioned above. Finally, we analyzed our results and proposed a definitive final model.

This experimentation step allowed us to grasp some knowledge about the behaviour of this dataset. As we have previously commented on the results section 5, different datasets with distinct features and normalization approaches were tried for every considered method. The purpose of these executions was to check if there was a noticeable difference regarding these initial conditions (for this specific task).

Although some differences can be seen about the feature subset used, the impact was not as harsh as we expected. In previous projects we would have never considered keeping variables with that amount of missing values. Although it did not pay off, it did not ruin the perfomance either. This could very possibly be due to these specific features relevance, but we will keep this knowledge into account for further studies. We also found remarkable that the dataset with the five numerical features was able to yield competent results in terms of accuracy. It is also worth to notice that these variables were highly ranked in Random Forest feature importance.

A similar argument to the previous experiment can be extracted for the selected normalization approach. The standarization data seemed to yield slightly better accuracy results in the best solutions of some methods. However, when comparing all other reviewed models, the performances were matched.

After working with this dataset, we have confirmed the missing data from it to be a challenging factor that we believe could have an effect, in addition to the subjective nature of the data, on the upper bound of predictive performance. Extensive research was done to investigate related work and all solutions found seemed to be in the same scope. Nevertheless, the results obtained are more than competent and matched our expectation as we have considered many additional valuable approaches.

On a personal level, we believe we have done a good job. We deem this project, as well as the contents of the subject, to be very interesting. All of this has helped us to delve deeper into many of the essential concepts in the field of Machine Learning. We are satisfied with the applied methodology and the insights we have gained from the models we have selected. Also, the members of the group have maintained a very good work dynamic and we have been able to discuss and analyze each of the steps and tasks in a critical, careful and respectful way.

8 Future work

To conclude the project, we would like to comment on some aspects that we consider appropriate for future work associated with the problem we have addressed. First, we believe it would be interesting to conduct a thorough and in-depth analysis of the variables in the data set. In some cases (see table 2) we have been able to see that the Minimal data set (*i.e.* the one that considered only the original variables with real domain) has provided us with up to 73% of accuracy. Therefore, although it is true that it has not been the best result, we believe that it is necessary to analyze whether the result has been caused by the parameterization of the model applied or by the used variables.

In relation to the data set, we believe that it would be appropriate to carry out a more active, complete (with fewer missing values) and objective (without subjective measures of health personnel) update. In addition, it would also be interesting to expand the data with other metrics that may have an impact on the survival of the animal.

To conclude, in relation to the methods and algorithms used, we have been able to see how the programming languages used (R, for SVMs and Python for the other methods) have not been too efficient. Therefore, it would be interesting to use a programming language capable of managing efficiency with finer granularity (such as, for example, C++) and to use algorithms that are more focused in computational efficiency.

References

[1] D. Dua and C. Graff, *UCI machine learning repository*, 2017. [Online]. Available: `http://archive.ics.uci.edu/ml`.

[2] M. McLeish and M. Cecile, *Horse colic data set*, 1989. [Online]. Available: `http://archive.ics.uci.edu/ml/datasets/Horse+Colic`.

[3] R. Nock, M. Sebban, and D. Bernard, "A simple locally adaptive nearest neighbor rule with application to pollution forecasting," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 17, no. 08, pp. 1369–1382, 2003.

[4] M. A. Hall, "Correlation-based feature selection for machine learning," Ph.D. dissertation, The University of Waikato, 1999.

[5] K. M. Ting and I. H. Witten, "Issues in stacked generalization," *Journal of artificial intelligence research*, vol. 10, pp. 271–289, 1999.

[6] E. Frank and I. H. Witten, "Generating accurate rule sets without global optimization," 1998.

[7] H. A. Guvenir and A. Akkus, "Weighted k nearest neighbor classification on feature projections," in *Proceedings of the 12-th International Symposium on Computer and Information Sciences*, 1997.

[8] J. J. Liu and J. T.-Y. Kwok, "An extended genetic rule induction algorithm," in *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No. 00TH8512)*, IEEE, vol. 1, 2000, pp. 458–463.

[9] M. Deshpande and G. Karypis, "Using conjunction of attribute values for classification," in *Proceedings of the eleventh international conference on information and knowledge management*, 2002, pp. 356–364.

[10] R. K. Singh, *Visualization, analysis, insights & predictions*, 2020. [Online]. Available: `https://www.kaggle.com/code/chandrarajsingh/visualization-analysis-insights-predictions`.

[11] N. H. Mallampalli, *Hyper parameter tuning : Svm*, 2020. [Online]. Available: `https://www.kaggle.com/code/himakund/hyper-parameter-tuning-svm`.

[12] S. Siddiqi, *Data examination and cleaning*, 2018. [Online]. Available: `https://www.kaggle.com/code/sabasiddiqi/data-examination-and-cleaning`.

[13] S. Jäger, A. Allhorn, and F. Bießmann, "A benchmark for data imputation methods," *Frontiers in big Data*, p. 48, 2021.

[14] D. J. Stekhoven and P. Bühlmann, "Missforest—non-parametric missing value imputation for mixed-type data," *Bioinformatics*, vol. 28, no. 1, pp. 112–118, 2012.

[15] L. L. Doove, S. Van Buuren, and E. Dusseldorp, "Recursive partitioning for missing data imputation in the presence of interaction effects," *Computational statistics & data analysis*, vol. 72, pp. 92–104, 2014.

[16] A. Shashua, "Introduction to machine learning: Class notes 67577," *arXiv preprint arXiv:0904.3664*, 2009.

[17] scikit-learn developers, *User guide sklearn - linear models*, 2007. [Online]. Available: `https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression`.

[18] scikit-learn developers, *User guide sklearn - decision trees*, 2007. [Online]. Available: `https://scikit-learn.org/stable/modules/tree.html`.

[19] scikit-learn developers, *User guide sklearn - ensemble methods*, 2007. [Online]. Available: `https://scikit-learn.org/stable/modules/ensemble.html#forests-of-randomized-trees`.