Instituto Superior Técnico

Departamento de Matemática

Mestrado em Matemática e Aplicações

Métodos Estatísticos em Data Mining

Computational Project 1

Authors:
Miguel Nascimento, 83646
Tiago Santos, 83657
Miguel Engrossa, 84142
Beatriz Santiago, 86592
Rodrigo Ribeiro, 87936

Professor:
Conceição Amado

April 30, 2020

Contents

1 Introduction

Cancer is the uncontrolled growth of abnormal cells in the body. Old cells do not die and instead grow out of control, forming new, abnormal cells. Cancer may occur anywhere in the body. In women, breast cancer and ovarian cancer are the most common. In men, it's prostate cancer. In recent years, data mining has become a process of growing importance in early cancer detection.

2 Problem Description

2.1 Dataset Description

The data was obtained from two sources: The National Cancer Institute (NCI) and the Eastern Virginia Medical School (EVMS). All the data consists of mass-spectra obtained with the SELDI technique. The samples include patients with cancer (ovarian or prostate cancer), and healthy or control patients [1]. From NCI, two datasets were obtained, one regarding ovarian cancer and another for prostate cancer. The third dataset has information with respect to prostate cancer given by the EVMS.

The three datasets were merged from those two different sources, and it resulted in 901 observations. The datasets at our disposal were already a split of this merge into a training, a validation and a test dataset (`arcene_train`, `arcene_valid` and `arcene_test`). Given that `arcene_test` had the response variable missing, we created a new dataset that had the training and validation dataset, which resulted in a total of 200 observations and 10000 features.

The variables are not labeled and therefore we have no way of knowing what they stand for.

According to [1], 3000 out of the 10000 variables were probes (i.e. random features), so we decided to remove them and work solely on the original dataset. Of the 7000 remaining features, some of those had the same constant value. Thus, these variables are of no relevance whatsoever to this study (given that no new information is added) and were deleted, leaving us with 6978 columns. Afterwards, we realized that some of the columns were repeated or were the result of the product of another column by a scalar. Although this may have effect on some classifiers, we took the decision to remove such columns. This was made by computing the correlation matrix of `arcene` and deleting variables with a correlation bigger than 0.99. This may have deleted some columns that were not strictly scalings of other columns, but we considered the lost information was worth it - we reduced the dataset from 6978 columns to 3525 columns, almost by half. When performing correlation analysis for 99%, we are removing variables that have close to a perfect increasing/decreasing linear relationship. This way, in the following sections we consider the "full dataset" to be this dataset with 3525 columns in which we deleted highly correlated variables.

2.2 Objectives

This is a two-class classification problem with continuous input variables.

The performed analysis aims to observe which features are the most helpful in determining if a given patient has cancer or not. The task of ARCENE is to distinguish cancer versus normal patterns from mass-spectrometric data.

2.3 Statistical Analysis

Even though we reduced by almost two thirds the number columns, there is still a big discrepancy between the number of independent variables (3525) and the number of observations (200). Therefore, our classifiers will probably be too well adjusted to the training dataset and most likely, not return great results regarding the test dataset.

Below, in figure 1, *1* indicates the presence of cancer cells and *-1* the absence of such cells. Therefore, 44% of the observations have cancer representing 88 sick patients, while 112 patients are cancer free accounting for 56% of the observations. Usually, when talking about cancerous diseases and when considering datasets of patient data, we are used to seeing datasets with very few observations where cancerous cells are present in comparison with healthy cases, which makes this dataset quite unusual for being composed by more than 40% of observations from cancerous cases.

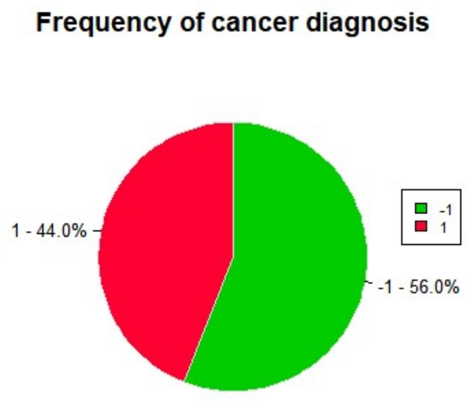

Figure 1: Pie chart of the cancer cases in the dataset

The matrix can be classified as sparse - even after removing all of the columns composed entirely by zeros, almost half of the entries (48%) are zeros. This may be a challenge for our classifiers.

It also worth to note that there is not a single missing value in the whole dataset.

Given that we have no information about variables, not much more could be done regarding exploratory data analysis.

3 Pre-Processing

In order to determine the best classification methods, it was required to perform the methods in question in different situations so that we could study the impact of each processing technique and thus, conclude which method had the best overall behaviour amongst the considered occasions. In this first part of the project, we only performed supervised methods.

We wanted to test not only the classifiers but also the impact that preprocessing may have in the classification. To achieve this, we used feature reduction based on correlation and principal component analysis. For feature reduction based on correlation, we used the thresholds of 75%, 85% and 95%, which will be referring as "Corr 0.75", "Corr 0.85" and "Corr 0.95". For PCA, we decided on keeping 75%, 85% and 95% of the variability, which we will refer to as "PCA 0.75", "PCA 0.85" and "PCA 0.95". This way, we tested all of our classifiers with 7 different cases - the full dataset (in which, as said before, we used feature reduction with a threshold of 99% for correlation), the three options for correlation and three options for PCA.

Each of these datasets was split into training and testing - 150 observations for training and 50 for testing.

4 Supervised Methods

First and foremost, we resorted to the `caret` package in `R` to cross validate our training datasets. We used k-fold cross validation with $k = 5$. The `train` function in this package allowed us to set up a grid of tuning parameters for each supervised method (if applicable) and, afterwards, to fit the models created with the tuning parameters to each of our datasets. The optimal model for each dataset was selected according to accuracy. Subsequently, we used the testing datasets to evaluate our optimal models according to the 5 measures that we believed to be the most relevant, considering that our classification problem is relative to the medical field. These five measures chosen, which we obtained by computing the confusion matrices of the models, were accuracy, true positive rate (TPR), false positive rate (FPR), precision and F1 score. We figured these would be the best measures to evaluate the performance of our classifiers - accuracy is the classical measure, number of correctly predicted cases. TPR and FPR are essential given that we are dealing with a problem from medicine. Precision measures the ratio of true positives over true positives plus false positives, that is, how many of the selected "positive" elements are actually positive. F1 Score is simply an harmonic mean of TPR and Precision.

The options taken and the results obtained, for each supervised method, are shown in the following subsections.

4.1 KNN

KNN - short for K-Nearest Neighbors - is a method of supervised learning in which an unknown instance is assigned the label most common amongst its k nearest neighbors. For example, for $k = 1$, the algorithm checks which of the instances on the training set is closer to the unknown instance and classifies that unknown instance in the same class as the known instance. For $k > 1$, a plurality vote is required, assigning the unknown instance the label most common amongst the neighbors. Ties may be a cumbersome issue in KNN - in the case of a binary classification, as we have, it is advisable to only choose odd values for k to avoid them.

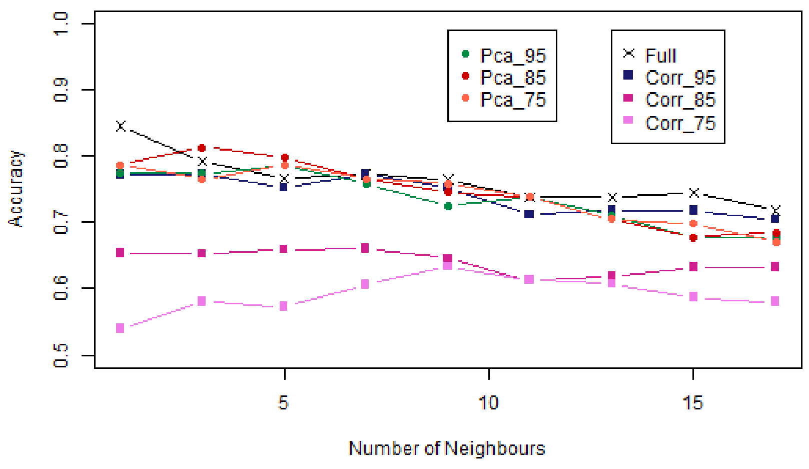

Figure 2: Accuracy values for the various preprocessing methods using KNN

We used `caret` to change the number of neighbors in KNN, changing from k=1 to k=17 in steps of two, to ensure we do not have ties. We did this for each of the preprocessing options (full dataset, three thresholds of correlation and three thresholds of PCA) and choose the best k using only accuracy. This gave us 7 different models for the KNN method. In Figure 2 we show the accuracy of the method for each of the seven preprocessing options and for each value of k. Clearly, the accuracy tends to decrease with as k increases in the case of PCAs. In the case of the correlation preprocessing, the 95% threshold can compare with PCA and the full dataset, but lower thresholds have significantly worse results. Perhaps the most interesting conclusion is that the full dataset does not give the best results for all values of k: PCA preprocessing gives us better results in some cases, even though it has less information than the full matrix. It is also worth to note that the best number of neighbors tend to be low, around 1,3 or 5.

	Accuracy	TPR	FPR	Precision	F1 Score
Full dataset (corr 0.99)	0.84	0.789	0.129	0.789	0.789
Corr 0.95	0.78	0.7	0.167	0.737	0.718
Corr 0.85	0.66	0.538	0.208	0.737	0.622
Corr 0.75	0.6	0.483	0.238	0.737	0.584
PCA 0.95	0.82	0.727	0.107	0.842	0.780
PCA 0.85	0.84	0.762	0.103	0.842	0.8
PCA 0.75	0.82	0.778	0.156	0.737	0.757

Table 1: KNN metrics scores

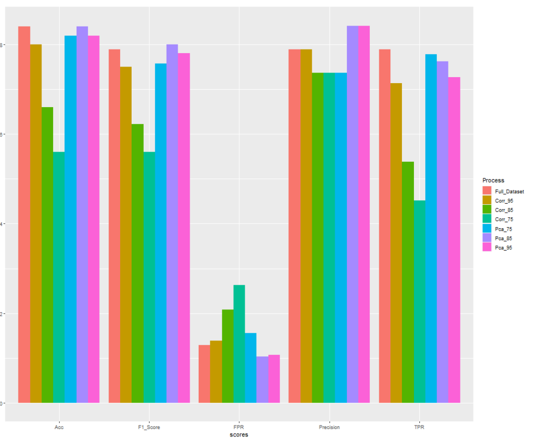

Figure 3: Metric scores for different datasets after applying KNN

In Table 1 we show the other measures for the best k choosen by `caret` for each of the preprocessing options. In Figure 3 we show the same information in a visual form. Given the medical nature of this dataset, perhaps the most important way to access the correctness of the classification would be TPR and FPR. To do this, we show in Figure 4 the ROC chart for the 7 options. There is a clear cluster with the PCAs, full dataset and the threshold of 95% for correlation. In this cluster, it is clear that the best results were obtained either with the full dataset or the PCA with 85% of variability explained. If we were forced to choose just one model, PCA 0.85 would be chosen given it has a higher precision.

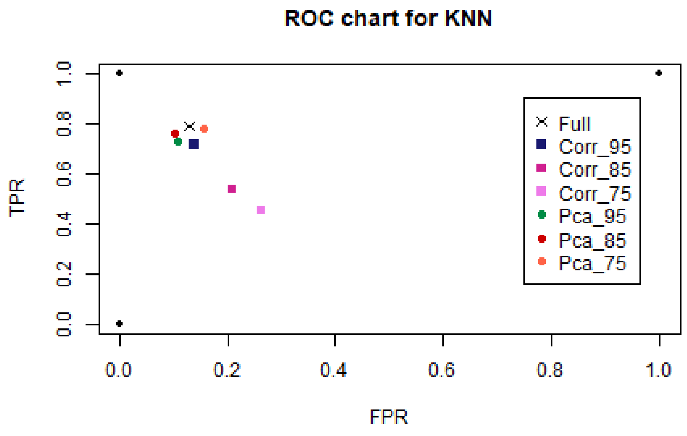

Figure 4: ROC for the various preprocessing methods using KNN

It is however important to keep in mind that we did not check everything there was to check. For example, the next step with these KNN models would be to use them with another distance metric, such as the Manhattan distance.

4.2 Decision Trees

Decision Trees are a type of Supervised Machine Learning where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final outcomes. And the decision nodes are where the data is split.

We took advantage of `caret` command with the method `rpart` where the tuning parameter is the complexity parameter (cp). This parameter is used to control the size of the decision tree and to select the optimal tree size. If the cost of adding another variable to the decision tree from the current node is above the value of cp, then tree building does not continue. We could also say that tree construction does not continue unless it would decrease the overall lack of fit by a factor of cp. By default, the gini impurity was used to select splits when performing classification.

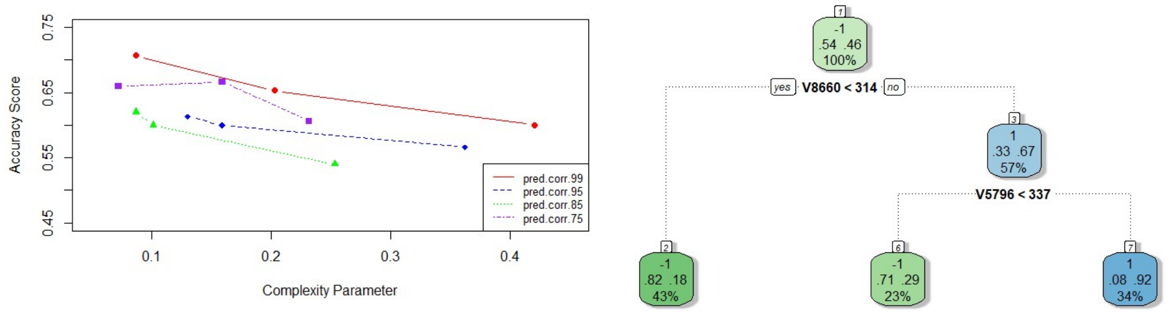

Figure 5: On the left, a plot representing the accuracy of each correlation dataset for different complexity parameter values. On the right, the decision tree with the best accuracy, Corr 0.99.

In figure 5, on the left we represented how each correlation dataset scored in terms of accuracy for different cp values. As expected, the dataset Corr 0.99 had the best result since it has the greater amount of variables, being the closest to the original dataset. On the right, we have the decision tree obtained by the mentioned dataset with an accuracy of 0.707 and $cp = 0.0870$. The tree has a depth of 2 and for the first decision node, variable **`V8660`** was used with a threshold of 314. For the second node, the chosen variable was **`V5796`** with a threshold equal to 337.

Similarly, this procedure was applied to the training datasets that were transformed under principal component analysis. As it can be seen below (figure 6), the best dataset was Corr 0.75 with $accuracy = 0.68$ and $cp = 0.0725$. The most relevant principal components were **`PC15`** (with a threshold of -1.3) and **`PC5`** (with a threshold of -7).

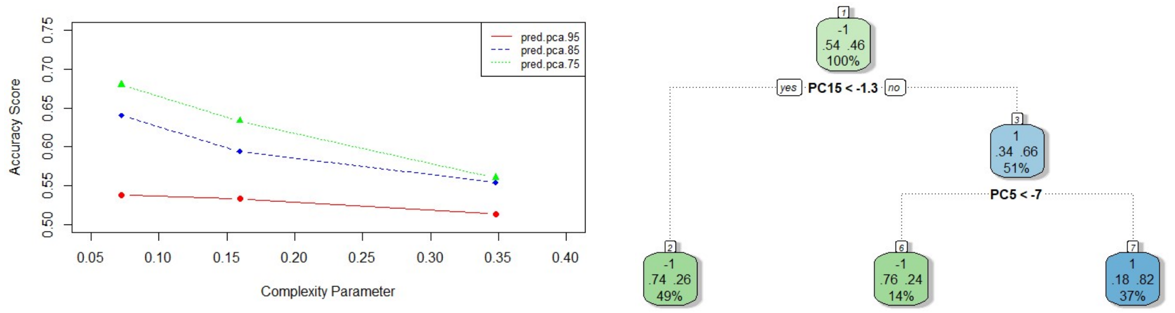

Figure 6: On the left, a plot representing the accuracy of each PCA dataset for different complexity parameter values. On the right, the decision tree with the best accuracy, Corr 0.75.

We also experimented with other tuning parameter by using the method `rpart2`, where the maximum tree depth was taken into account. Checking both figures below (8 and 7), the first thing that stands out is that in both cases (Correlation and PCA datasets), the optimal tree depth was 3, where Corr 0.99 and Corr 0.85 scored an accuracy of 0.727. For the correlation dataset, there are 4 decision nodes with the following variables **`V8660`**, **`V5796`**, **`V1193`** and **`V5451`**. Once again, variables `V8660` and `V5796` are necessary to build the decision tree with the same threshold as before. As for the PCA dataset a similar scenario occurs, where **`PC15`** and **`PC5`** are still in two decision nodes with the same threshold. We now have 4 more decision nodes when compared to the decision tree in figure 6, which are **`PC25`**, **`PC1`**, **`PC9`** and **`PC65`**.

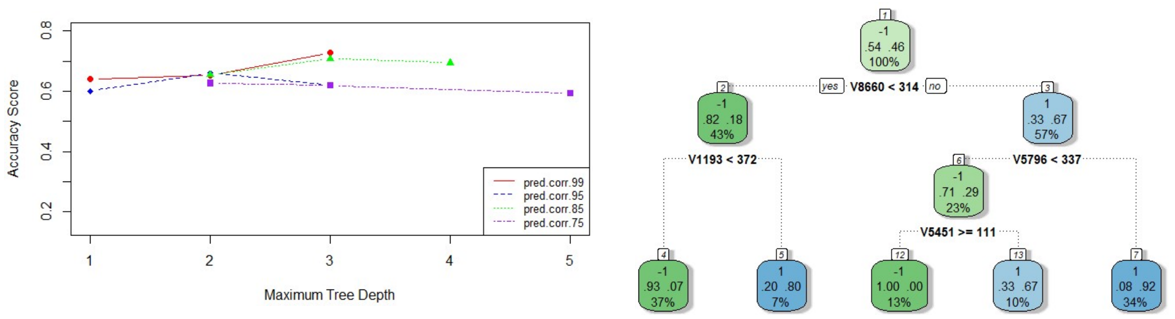

Figure 7: On the left, a plot representing the accuracy of each correlation dataset for different maximum tree depth. On the right, the decision tree with the best accuracy, Corr 0.99.

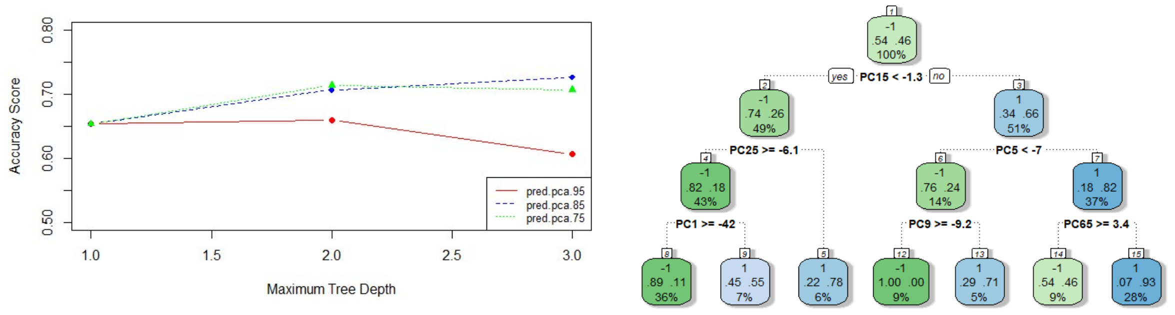

Figure 8: On the left, a plot representing the accuracy of each PCA dataset for different maximum tree depth. On the right, the decision tree with the best accuracy, Corr 0.85.

Now, using these models we predicted for each test dataset the response variable, obtaining a confusion matrix, which allowed us to use the measures described previously and get a better insight on the best model. By a medical point of view, TPR and FPR are the most important measures.

	Accuracy	TPR	FPR	Precision	F1 Score
Full dataset (corr 0.99)	0.76	0.733	0.229	0.579	0.647
Corr 0.95	0.76	0.64	0.12	0.842	0.727
Corr 0.85	0.74	0.875	0.286	0.368	0.518
Corr 0.75	0.66	0.528	0	1	0.691
PCA 0.95	0.68	0.588	0.273	0.526	0.555
PCA 0.85	0.68	0.588	0.273	0.526	0.555
PCA 0.75	0.68	0.588	0.273	0.526	0.555

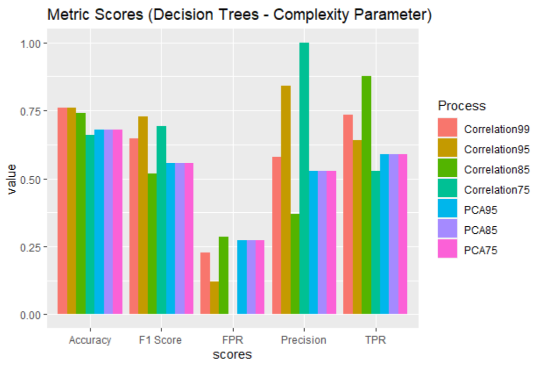

Figure 9: Metric scores for different datasets after applying decision trees with cp as a tuning parameter

The first thing that stands out is the dataset Corr 0.75 that scored a remarkable $FPR = 0$. This null result represents that no patient was classified as having cancerous cells when he was healthy. This is of the utmost importance given that it saves resources and money by not "wasting" time trying to cure healthy patients. However, this is probably not a very reliable result, given that this dataset is the one with the least explained variation of the data. Another relevant thing to point out is an extremely high TPR obtained from Corr 0.85 when compared to all the other values obtained throughout this paper. Also, as it can be seen above, there is absolutely no difference on applying PCA that explain 75%, 85% or 95% of the variation in the data, given that all measures return the same values regardless.

Similar results were obtained when predicting the response variable for the test datasets in which the method had the maximum tree depth as a parameter. Once again, the dataset Corr 0.75 scored $FPR = 0$. In general, the values for true positive rate were quite lower in comparison to the previous method.

	Accuracy	TPR	FPR	Precision	F1 Score
Full dataset (corr 0.99)	0.8	0.714	0.138	0.789	0.750
Corr 0.95	0.72	0.593	0.13	0.842	0.696
Corr 0.85	0.66	0.55	0.267	0.579	0.564
Corr 0.75	0.66	0.528	0	1	0.691
PCA 0.95	0.64	0.524	0.276	0.579	0.550
PCA 0.85	0.62	0.5	0.269	0.632	0.558
PCA 0.75	0.64	0.524	0.276	0.579	0.550

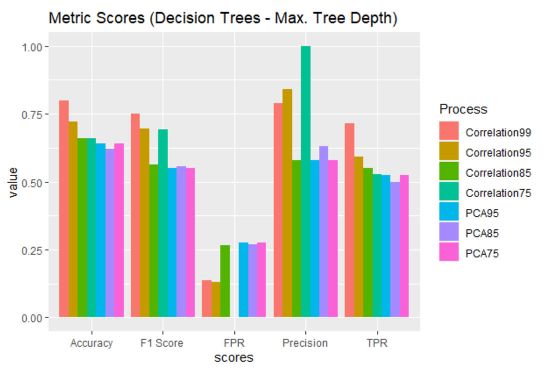

Figure 10: Metric scores for different datasets after applying decision trees with maximum tree depth as a tuning parameter

In either scenario, we were expecting that the greater the percentage for correlation or explained variation (PCA), the better the results. However, this wasn't the case at all. This might be explained by small number of observations when compared to the number of variables under study.

Overall, regarding the decision tree method, the datasets that were subject to correlation analysis performed far better than those who went through PCA transformations. This was verified with both tuning parameters, cp or maximum tree depth.

4.3 Random Forests

The fundamental idea behind a random forest is to combine many decision trees into a single model. Individually, predictions made by decision trees may not be accurate, but combined together, the predictions will be closer to the mark on average.

This method consists in selecting a pre-determined number of randomly chosen variables and with the help of measures (gini impurity, in our case), the variable that scores the best will be chosen to the decision node, and so on. This procedure is repeated, creating n decision trees, which will be used to determine the response of a new observation. Here, we chose to vary the number of randomly selected variables at each iteration.

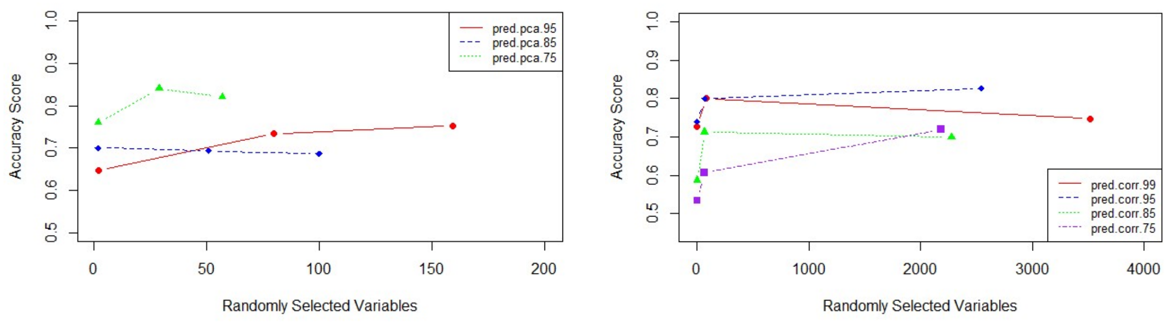

Figure 11: Plots of accuracy vs. number of randomly selected variables for all training datasets

On figure 11, we can see how changing the number of randomly selected variables affects the accuracy of each dataset. The dataset transformed via principal component analysis that scored the higher was Corr 0.75 obtaining 0.840 on *accuracy* when the number of randomly selected variables was 29. As for correlation datasets, Corr 0.95 had the best result with *accuracy* = 0.826 but with a way larger number of selected variables at each step, 2544.

Finally, we predicted what the outcome would be for the observations in the test dataset in order to check how well this method adapts.

	Accuracy	TPR	FPR	Precision	F1 Score
Full dataset (corr 0.99)	0.78	0.7	0.167	0.737	0.718
Corr 0.95	0.78	0.7	0.167	0.737	0.718
Corr 0.85	0.82	0.778	0.156	0.737	0.757
Corr 0.75	0.8	0.8	0.2	0.632	0.706
PCA 0.95	0.76	0.769	0.243	0.526	0.625
PCA 0.85	0.68	0.615	0.297	0.421	0.500
PCA 0.75	0.82	0.812	0.176	0.684	0.743

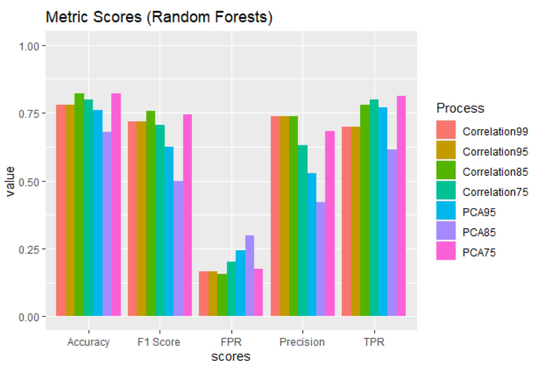

Figure 12: Metric scores for different datasets after applying random forests

For the PCA's datasets, Corr 0.75 is still the one with the best results out of the three datasets. These results are a great improvement when compared to the ones obtained by the Decision Tree Method, which was expected, taking into account how Random Forests are built. However, it is odd since this is the dataset with least explained variation of the data. This could be possibly explained by what was mentioned in a previous section: the ratio between the number of observations and the number of variables is extremely low when it should be greater than 10. Thus, it results in a bad scenario to apply some methods.

The datasets built under correlation analysis performed a bit better in terms of FPR and TPR.

As it has been referred on multiple occasions, the huge difference between the number of observations and variables is a great concern of ours. Thus, we resorted to *Out-of-Bag* error (OOB), which analyses the errors made when building random forests. The great thing about it is that it allows to analyse the data without the need of train and test dataset. That way, we are able to perform this method with 200 observations, instead of 150 on the training dataset. It might not seem much, but given shortage of observations, it could make a significant difference.

Below, we present the error rates for the training datasets and the full datasets after the transformations. Note that these values were obtained when the number of trees was 500.

Method	**Correlation**		**PCA**	
Dataset	*Training*	*Full*	*Training*	*Full*
75%	24%	19.5%	22.67%	20%
85%	26.67%	19%	31.3%	20.5%
95%	22%	21%	26.67%	23.5%
99%	16.67%	20.5%	–	–

Table 2: OOB estimate of error rate for each dataset

As it can be seen on table 2, with the exception of Corr 0.99, all datasets improved when the full dataset was used. Some had quite the significant difference which corroborates what was mentioned previously. The following two plots represent how this error rate changes when the number of trees in the random forests increases. These two plots are the ones with the lowest OOB error when the random forest was formed 500 trees.

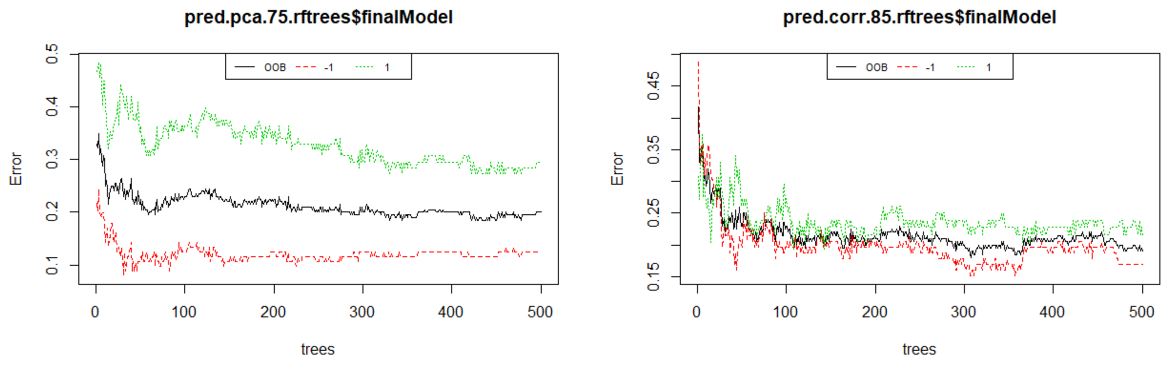

Figure 13: Plots of Out-of-Bag error for the datasets with best accuracy, Corr 0.75 and Corr 0.85

4.4 Naive Bayes

Naive Bayes classifiers are probabilistic machine learning classifiers based on applying Bayes's theorem, while assuming strong independence between the features. It is due to these assumptions that these classifiers are said to be naive.

The method `nb`, in the `caret` package, for Naive Bayes, has 3 tuning parameters. The following choices for the parameters were made:

- We kept `fL` (the factor for Laplace correction) as 0, since out datasets only contain continuous features, and thus we did not need Laplace correction;
- The `usekernel` parameter was either `True` of `False`, meaning that the model would either use a kernel density estimate or a normal density estimate, respectively;
- The `adjust` parameter (used to adjust the kernel density estimate, if `usekernel` is defined as `True`) took integer values from 1 to 4, since we noticed that the optimal value for `adjust` for these datasets was often contained in the real interval $[1, 4]$, and experimenting with more values could require heavy computational power or become very time-consuming, especially for the correlation datasets.

It is important to mention that the `train` function was not able to create models with `usekernel` set to `False` for the datasets built through correlation analysis (including the "full" dataset, where we eliminated variables with correlation higher than 99%). The error message provided us with a clue as to why this was the case: in variables with an extremely small number of observations different from zero, the `nb` method may calculate the variance of one of the classes as zero, and thus a normal density estimate cannot be used. For the PCA datasets, the same did not occur, as there were no variables in this conditions.

Another relevant thing to point out is that during the process of cross validation, and afterwards, in the testing of the method, we got the warning message "Numerical 0 probability for all classes with observation X" from `R`, for several observations. We suspect that this happened because, during classification using Naive Bayes, when calculating the product of the conditional probabilities, `R` obtained probabilities extremely close to 0 for both classes with these observations (floating-point underflow). Nevertheless, `R` was still able to achieve results. After a quick analysis of the code for the `nb` method in `R`, we believe that this was possible because the method also performs the calculations by summing the logarithms of the probabilities, to avoid underflow.

A plot representing the accuracy scores for each dataset for the different values of the tuning parameters is shown in Figure 14.

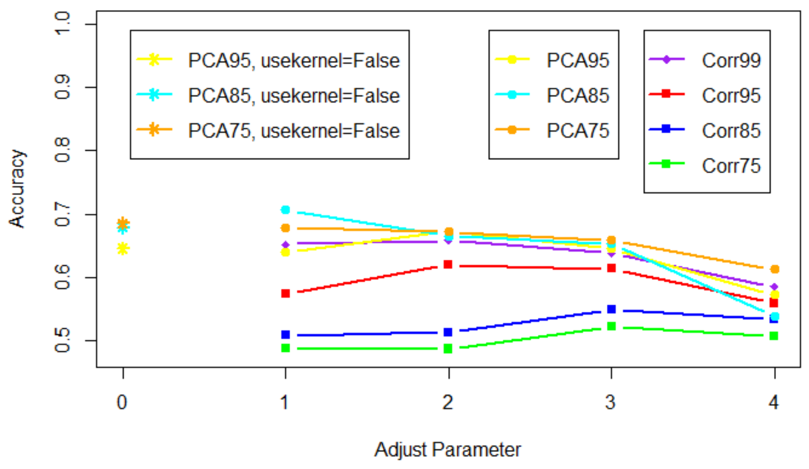

Figure 14: Plot representing the accuracy obtained in cross validation for each dataset for the different values of the tuning parameters when applying Naive Bayes (the accuracy scores of the models with `usekernel` set to `False` are shown in the points where `adjust` is 0, since a normal density estimate does not have the `adjust` parameter, or in other words, `adjust` in the method `nb` is set to 0)

By analyzing Figure 14, we can first see that the only dataset where a normal density estimate was optimal was the PCA dataset that can explain 75% of the data. For the rest of the datasets, a kernel density estimate was preferred. Also, for the datasets where a this type of density estimate was preferred, our choice of values for the `adjust` parameter appears to be adequate, since for the majority of the datasets, the optimal value seems to be in the real interval $[1, 4]$ (with the only exception being possibly the PCA dataset that explains 85% of the data). As for the accuracy scores, the models for the PCA datasets were amongst the best. The model for the "full" dataset (Corr 0.99) was also relatively good in comparison, while the worst ones were clearly the models for the Corr 0.75 and Corr 0.85 datasets.

	Accuracy	TPR	FPR	Precision	F1 Score
Full dataset (corr 0.99)	0.7	0.571	0.136	0.842	0.681
Corr 0.95	0.68	0.556	0.174	0.789	0.652
Corr 0.85	0.46	0.357	0.409	0.526	0.425
Corr 0.75	0.46	0.333	0.423	0.421	0.372
PCA 0.95	0.68	0.56	0.2	0.737	0.636
PCA 0.85	0.7	0.591	0.214	0.684	0.634
PCA 0.75	0.76	0.652	0.148	0.789	0.714

Table 3: Naive Bayes metrics scores

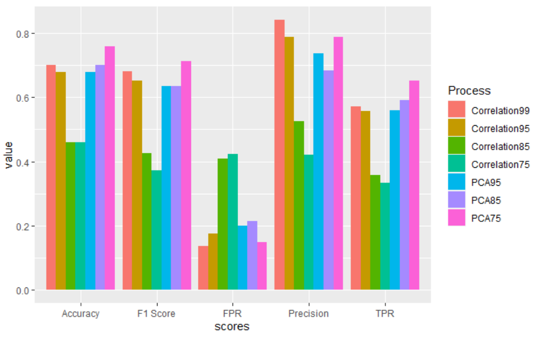

Figure 15: Metric scores for the different datasets after applying Naive Bayes

The metric scores for the different datasets, obtained after testing, can be seen in Table 4 and Figure 16, and they allow us to draw similar conclusions. The scores for the PCA datasets were amongst the best, with the PCA 0.75 dataset achieving the best scores in Accuracy, TPR and F1 Score. This makes sense, as these datasets are created by generating non-correlated combinations of the original variables, which is beneficial for Naive Bayes classifiers. However, it is odd that the best PCA dataset was the one with the least explained variation of the data, something that also happened for some of the other classifiers we used in this project. One possible reason for this may be the extremely low ratio between the number of observations and the number of variables.

On the other hand, the “full” dataset achieved the best score in terms of FPR and Precision, and it got good results all-round, in comparison with the rest. Although the Corr 0.95 dataset obtained slightly worse scores than the “full” dataset, it still was on par with the rest. In turn, the worst datasets were clearly Corr 0.85 and Corr 0.75. This shows us that, for the Naive Bayes classifier, eliminating more variables through correlation analysis does not necessarily lead to a more efficient and correct model, even though we decreased the correlation between the variables in the dataset.

4.5 Logistic Regression

We decided to use as a classification method a linear method as well. Since our output variable is binary (either 1 or -1), we thought of using the Logistic Regression method. As it is a linear method, the decision boundaries are linear and the goal is to describe the response variable as a multiple linear regression where the independent (explanatory) variables can be categorical or continuous. The motivation to use this method is model the posterior probabilities of the different classes through linear functions and ensuring that these probabilities sum to 1 and remain in $[0, 1]$. Introducing the sigmoid/logistic function:

$$f(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

where t represents the linear function composed of the explanatory variable(s). This function takes values between 0 and 1 and its result represents the probability of the observation to be classified as 0 or 1 (in our case).

Since this classification approach has no tuning parameters, we just use the `caret` package with default settings for the logistic regression. To properly use this package, the labels identified as -1 were transformed into zero in order to have a binary output composed of zeros and ones.

Splitting the base datasets into train and test sets, some metrics were evaluated to be able to compare the performance of this method between sets. Defining in our model a 5-Fold Cross-Validation and defining the accuracy as our metric to guarantee better results the different datasets were tested and put into analysis. These results are shown in the table below:

	Accuracy	TPR	FPR	Precision	F1 Score
Full dataset (corr 0.99)	0.48	0.348	0.407	0.421	0.381
Corr 0.95	0.48	0.379	0.381	0.579	0.458
Corr 0.85	0.54	0.433	0.3	0.684	0.53
Corr 0.75	0.4	0.28	0.48	0.368	0.318
PCA 0.95	0.66	0.542	0.231	0.684	0.605
PCA 0.85	0.78	0.7	0.167	0.737	0.718
PCA 0.75	0.8	0.737	0.161	0.737	0.737

Table 4: Logistic Regression metrics scores

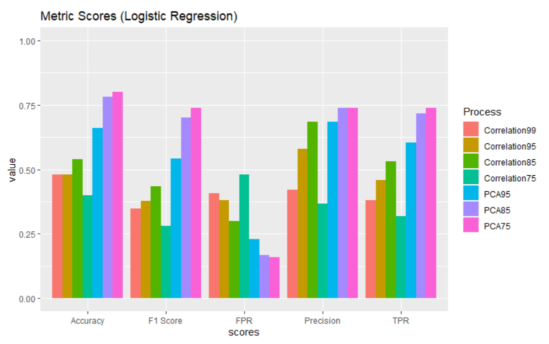

Figure 16: Metric scores for the different datasets after applying Logistic Regression

Analysing these results, we can observe that the datasets where variables were removed based on their correlation have very bad results regarding the classification through this method, always having an accuracy below 54%. Regarding the PCA datasets, we can point out that the results were not extraordinary, since the best result we got was regarding **PCA 0.75** with an accuracy of 80%. This shows us that having more variables is not equal to having a more efficient and correct model for this classifier. This dataset also has the highest TPR, Precision and F1 Score, and the lowest FPR, among all others, meaning it is indeed the best out of all the tested datasets.

Having few observations for a big amount of explanatory variables can be one of the causes of this "not so good" classification method, even though cross-validation was used. This heavily contrasts with what is recommended to apply this method: having 10 times more observations than variables (one in ten rule). Another reason for these results can come from the fact that this method assumes that all variables are independent from each other. In our case, this can't be verified since we have a lot of different variables (hence why we tested several different datasets with different number of variables), and it seems almost impossible that all of these predictors are orthogonal.

4.6 LDA

LDA - short for Linear Discriminant Analysis - is a method of supervised learning where a linear combination of features that characterizes or separates classes of objects or events. When a new observation arrives, the linear combination is computed and it is assigned to the class with a closest score. This method is used when groups are known a priori.

By taking advantage of the `Caret` package and its `lda` method we were able to perform Linear Discriminant Analysis. In the construction of the model, first we define a `trainControl` using 5-Fold Cross Validation. Since we are using this package we don't need to give information about prior probabilities, i.e. we do not need to inform the function about the actual frequency of each label in the training dataset in question because it will be calculated by the `train` function. The best model for each dataset was found using `Accuracy` as metric, which was also done by the function. Then, we predicted the results on the test dataset and computed a confusion matrix and from it we were able to calculate the true positive rate (TPR), the false positive rate (FPR), the Precision and the F1 Score.

We performed discriminant analysis in different settings in order to see if any of the pre-processing techniques impacted the final results. We could not do this to all the datasets that we had in the beginning because here were some problems in the computations. For the datasets that were obtained using correlation technique, some variables were constant for at least one of the labels which unabled `lda` method to perform. This might have happened because a lot of variables had more than 50% of zeros and when applying cross validation, at least one label must have been full of zeros. We tried to fixed this using leave one out cross validation instead of 5-Fold but, unfortunately, the results were the same.

The most extreme measure would be to take those variables out of the dataset. However, when we attempted to do it, there were too many variables with this problem.

	Accuracy	TPR	FPR	Precision	F1 Score
PCA 0.75	0.8	0.737	0.161	0.737	0.737
PCA 0.85	0.86	0.75	0.038	0.947	0.837
PCA 0.95	0.64	0.524	0.276	0.579	0.55

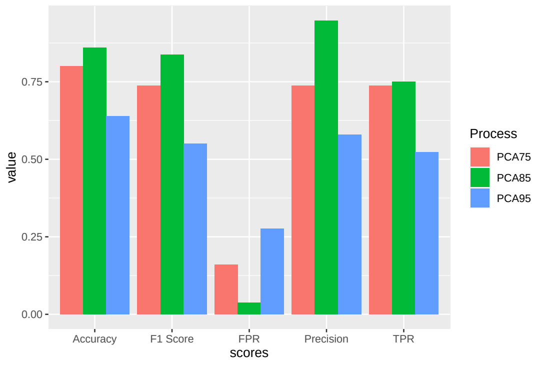

Figure 17: Metric scores for different datasets after applying LDA

In figure 17, we can see the results of the multiple measures for the 3 PCA's settings tested. As we said before these measures were computed making use of the confusion matrix for each dataset.

Looking to this figure we can clearly see that the best results were obtained using PCA 0.85 for all the measures used and the worst results were obtained using PCA 0.95 even though it explains more variability of the data. During the computation using this last dataset some warnings showed up saying that some variables were collinear. This can explain the poor results for this set.

5 Analysis

To sum up the work done and show in a clear way which classification methods show the best outcome in terms of being able to correctly identify new observations.

The best results of each technique and the associated dataset are displayed in Table 5.

	Accuracy	TPR	FPR	Precision	F1 Score
KNN (PCA 0.85)	0.84	0.762	0.103	0.842	0.8
Decision Trees (Max Tree - Corr 99)	0.8	0.714	0.138	0.789	0.75
Decision Trees (cp - Corr 0.85)	0.74	0.875	0.286	0.368	0.518
Random Forests (PCA 0.75)	0.82	0.812	0.176	0.684	0.743
Naive Bayes (PCA 0.75)	0.76	0.652	0.148	0.789	0.714
Logistic Regression (PCA 0.75)	0.8	0.737	0.161	0.737	0.737
LDA (PCA 0.85)	0.86	0.75	0.038	0.947	0.837

Table 5: Best scores for each classifier

With these results, we can see see that the highest accuracy is obtained using LDA (with the PCA 0.85 dataset), the highest TPR can be found when using Decision Trees with the cp as tuning parameter (with the Corr 0.85 dataset), the lowest FPR, the highest Precision and the highest F1 Score are observed in the LDA method (using the PCA 0.85 dataset).

Since 4 of the best metrics are found when we use the LDA technique, we conclude that the best results are thus obtained when combining this method with the PCA 0.85 dataset.

Another observation we can make is the fact that almost all of the best results for each classifier (apart from decision trees and random forests) are obtained when using datasets obtained through the PCA 0.75 and PCA 0.85 datasets. In other words, choosing one of these two datasets seems to have an important influence in the results for the methods we chose to evaluate.

If we were to talk with a research team, our contribution would be to understand what are the medical needs for a test like we have - that is, what is the (medical) research team trying to find? We cannot be confident enough that one of our classifiers can be a overall good test for cancer, because this field, in which human lifes are at stake, requires very good results. However, some information can be gained when we look at TPR and FPR.

Using decision trees with the Corr 0.85 we achieved 0.875 TPR and using LDA with PCA 0.85 we achieved 0.038 FPR, our best values for those measures. We would discuss with the researcher that those two different classifiers (using different preprocessing) may have some good results reducing the number of Type I or Type II errors. A test with such a low FPR would be somewhat good identifying the true positives - that is, the researcher or medical team could be quite sure that a person had cancer (reducing Type I errors), while maybe a second test would have to be made with the people who tested negative. A similar argument could be made for a test with high TPR, it would be helpful to check whether or not we are commiting Type II errors.

However, we would not advice any medical team to use any of these classifiers blindly, since our results, while somewhat good for other areas, are not up to the high standards of medicine.

6 Conclusion

These results could be heavily improved if we had a different dataset with more observations than variables, so that the models for each classifier had more information to make more suitable and accurate predictions.

Also, the fact that we do not know what the variables represent, means we cannot exclude any of them by order of importance. In other words, we cannot tell the difference between a variable that is extremely relevant to the problem in question, and another one that has nothing to do with it. So, we must work with all of the explanatory variables with the same order of significance, which can lead to some unwanted results. In a real world setting, we would be working with a multidisciplinary team with mathematicians, doctors, biologists and other experts from relevant areas that could guide us with knowledge that we do not have.

Another issue that can be related to these results may derive from the fact that some important features are excluded when the original dataset is transformed. Since we are not told what each of these features represent, it could happen that some relevant information is lost during the process of creating new datasets with less variables. Even though this can happen, we observed that in this case, for most of the methods, better results are usually obtained when datasets with less variables than the original one (Corr 0.99) are used.

It also worth to mention that our own work is somewhat limited and there could be much more to add. Outlier detection was not made at all, which could improve our results; other classifiers could be used or even other learning techniques, such as unsupervised learning (clusters, association rules, etc). Concerning the classifiers used, other parameters could be explored - for example, different distance metrics in KNN or different ways to split data in decision trees. However, our time and resources are limited and given those constrains we are somewhat satisfied with the results presented.

We think that future work should analyse other parameters of the classifiers used, use unsupervised learning and outlier detection.

7 References

[1] The National Cancer Institute (NCI) and the Eastern Virginia Medical School (EVMS). “Arcene Data Set”. In: (Feb. 2008). URL: http://archive.ics.uci.edu/ml/datasets/Arcene.

[2] Isabelle Guyon et al. “Result Analysis of the NIPS 2003 Feature Selection Challenge”. In: *Advances in Neural Information Processing Systems 17*. Ed. by L. K. Saul, Y. Weiss, and L. Bottou. MIT Press, 2005, pp. 545–552. URL: http://papers.nips.cc/paper/2728-result-analysis-of-the-nips-2003-feature-selection-challenge.pdf.

[3] Max Kuhn. “The caret Package”. In: (Mar. 2019). URL: http://topepo.github.io/caret/.

[4] James Le. “Decision Trees in R”. In: (June 2018). URL: https://www.datacamp.com/community/tutorials/decision-trees-R.

[5] Zulaikha Lateef. “A Comprehensive Guide To Random Forest In R”. In: (May 2019). URL: https://www.edureka.co/blog/random-forest-classifier/.