PREDICTING DEATHS AND SURVIVALS OF THE TITANIC

DATA ANALYSIS, SECOND ASSIGNMENT

Keyla Guadalupe Muñoz Guilcapi

Sofía Pérez Pérez

Group 97

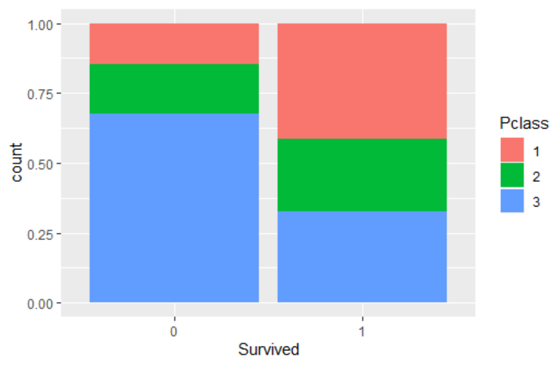
Introduction

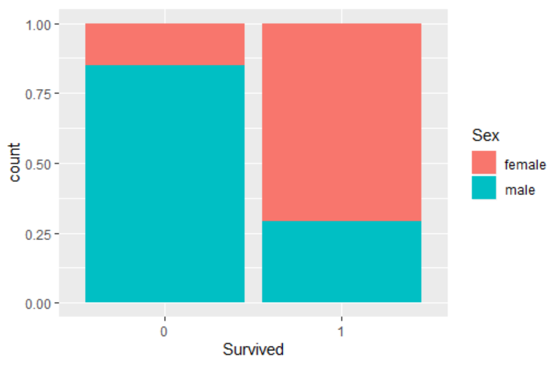
The aim of this project is to put into practice all the tools and methods learnt in class, to predict the survival or death of people in the Titanic. To do so, we count with several variables, ones with more importance than others, that will be used strategically for each of the classifiers that have been seen in class in order to get an accuracy for the prediction of the survival in each classifier and finally, choose the one which give us the best accuracy.

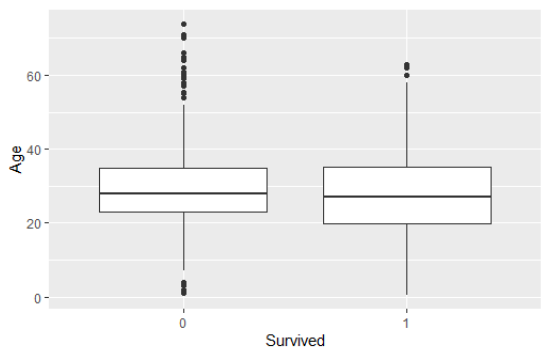
Our project is based on supervised learning. This type of Machine Learning consists of a target variable which is going to be predicted from a given set of predictors (independent variables) by generating a function that map inputs to desired outputs, in this case our desired output is the people that survived or not to the Titanic's sinking.

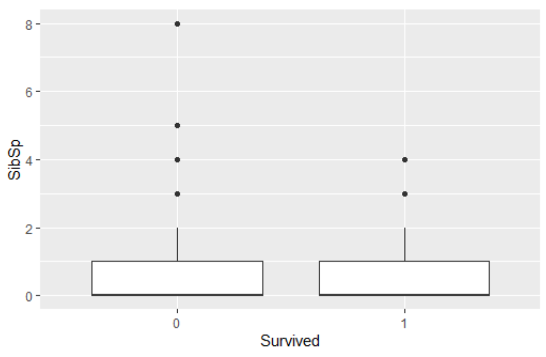
Exploratory Data Alanalysis

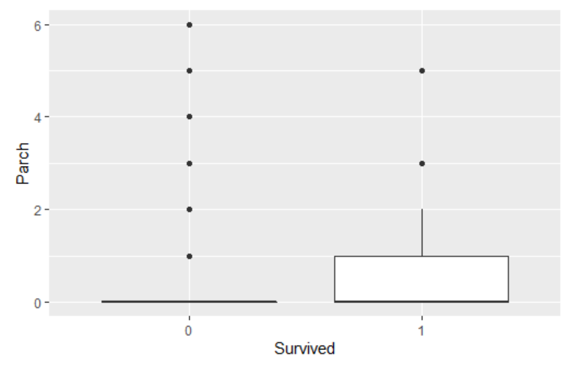
To understand better our data, we have started by doing an exploratory (unsupervised learning) data analysis. First of all, we computed four boxplots, one per variable, and for the factor variables two barcharts, to identify which variables were useful. In all the cases, we have compared them with survived, since it is our label data, and we have come up with these results.

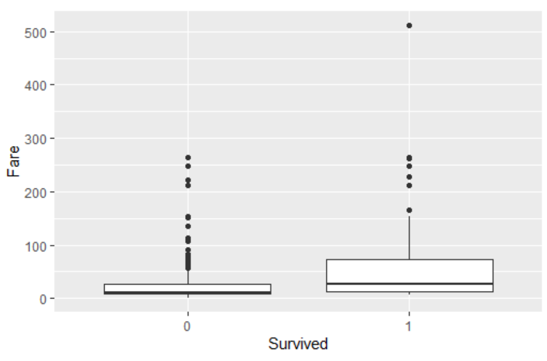






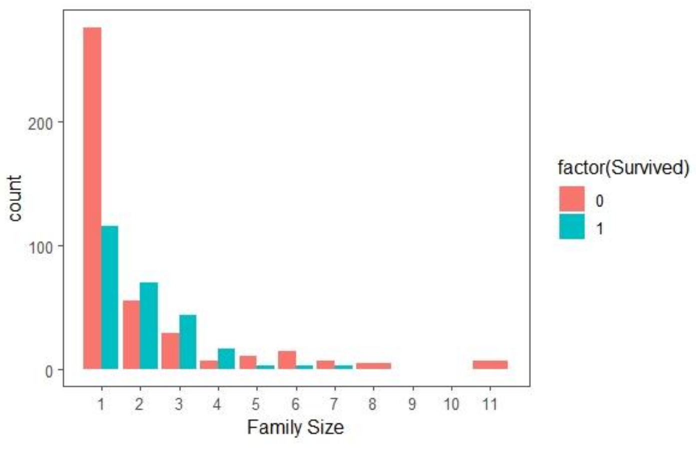






As we can observe, there are variables that are more useful than others. For example, sex, age, Parch, SibSp, are important, and also Pclass since it has outliers that could be interesting. But on the other side, variables like Fare is not going to be taken into account since we consider it is irrelevant, and also Embarked, since it is not complete.

We noticed that SibSp (Siblings/Spouse) and Parch (Parent/Children) are important variables to take into account, therefore, we want to get an insight of the relationship between families and survival. To do so, we created a variable called FamilySize which represent the number of the family members for each passenger, then to make it more visual we plotted a graph using ggplot.



From this plot we can notice that single people are those who mostly did not survive. On the other hand, the rate of survival in families with 2,3 and 4 members is greater than "not survived", and families with more than 4 members mostly did not survived. We considered this information very relevant.

Data arrangement

After having done the exploratory data analysis, we start to prepare our data. First of all, we delete the columns of the variables that we didn't consider very important, those were: Ticket, Fare, Cabin, and Embarked. Also, since Sex is a factor, to turn it into numeric, we used dummy cols, to obtain 0's and 1's, for men and women respectively, so it is easier to work with. Lastly, we considered as part of our data the variable that we created before “FamilySize” and we used this one instead of SibSp and Parch.

Another important work to do before starting to implement our classifiers, is preparing our data for the cross-validation. So, we develop the repeated k-fold crossvalidation by randomly splitting our data set into k-folds (being k an integer number) to make groups of train and test. By doing this crossvalidation we are assessing how the result of a model will generalize to an independent data set, in other words, we get an estimation of the accuracy of our model.

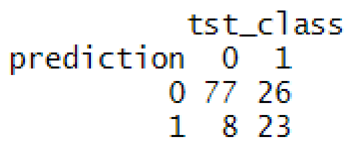
Creating predictive models by supervised learning

Support Vector Machines (SVM)

The first classifier we will work with is **Support Vector Machines**, that is a supervised learning model, that splits the data, dividing it by a line (straight or curve) or by a plane, depending on if our data is linearly separable or not. In this case, we have taken into account that our data is not linearly separable, so we have used the kernel radial and not linear. What we have done is to do a loop to try all the possible values of gamma (from 1 to 100), in all the different five combinations of train and test set, and each value has been stored in a matrix, and then we have computed the mean of each row, that means, the mean of the accuracy of the five different experiments. At the end we have taken as gamma the value of the vector parametergamma in the index with the value of the higher mean.

In our experience, we have done the experiment several times with different values of gamma (200 possible values) and the most repeated accuracy has been 74,62%, when gamma=1, so that is the reason why we have chosen that value for gamma. Anyways, since our data is randomly splitted, each time we obtain a different result and accuracy, so we have named finalgamma to the highest mean, so in case it is not gamma=1, the program computes automatically the best mean and the svm of that gamma in particullary.

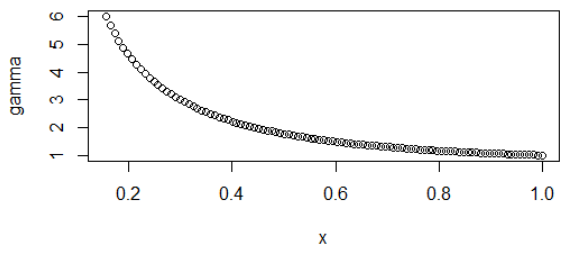
With gamma=1, we have obtained this confusion matrix:



```
          tst_class
prediction  0  1
         0 77 26
         1  8 23
```

This means that if there really were 85 people that died, but in the prediction 8 of those have been classified as survivals. And on the other hand, if there were 49 people that survived, 26 of them have been classified as deaths. As we can see, the error is much more committed in classifying survivals rather than deaths.

Apart from that, what we can see in this plot, is that gamma is going to decrease if the x gets bigger, this means that the best accuracy is obtained when gamma=1 and the worst when gamma= 6 approximately.



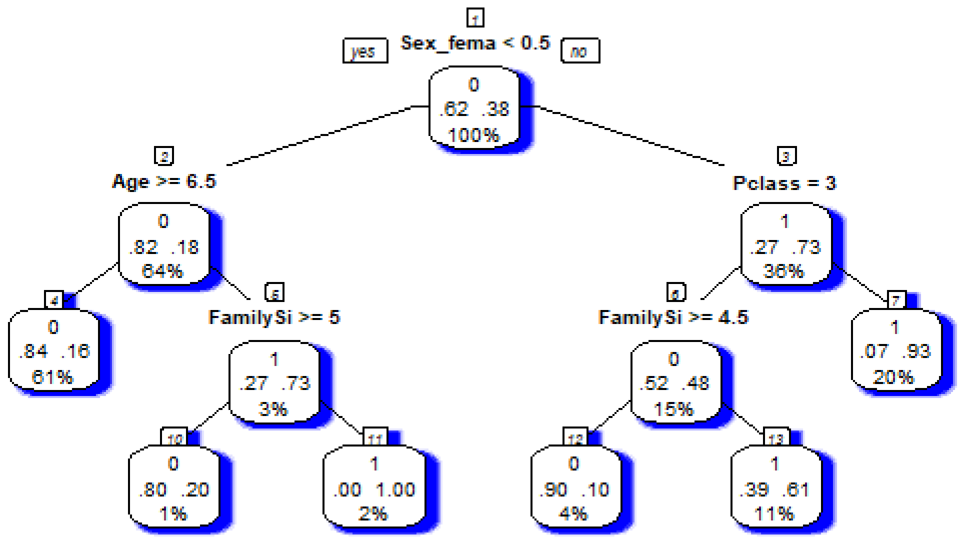
Decision Trees

Another classifier that we have used is **Decision Trees**. It used a tree to represent all de decisions that the classifier has to make, for example if the person is a girl or a boy. So at the end we obtain a tree with numerous branches. We have used the option extra=104 that shows the % of error and the % of success of each solution, taking into account the train and the test set. It also shows the % of people of our sample that belongs to that class (for example being a girl or a boy).

To compute the decision trees classifier, we had two parameters to take into account, minsplit and minbucket. To decide the optimal value for each one, we have created a vector of length the length of the parameterminsplit, and by combining 250 possible values of minsplit and 250 for minbucket, in each position of the vector we have added the mean of each of the five different experiments with different train and test sets, with the corresponded minsplit and minbucket. Apart from that, we have also created a matrix with two columns (one for the values of minsplit and the other one for the values of minbucket), and we have been storing there the values taken by parameterminsplit and parameterminbucket. We have considered this an easy way to obtain the values of the two parameters, referring to one specific mean. To find the highest mean we have followed the same process as in SVM, using the function which.max to obtain the higher mean, and in which row it was located. Then just by finding the values of the two parameters in the matrix in that specific row, we could obtain the value for minsplit and minbucket that maximizes the accuracy of our model.

We have done the experiment a lot of times, and thought we were conscious that the data is randomly splitted and that each time the most accurate values of the parameters would change, we consider that the most repeated one was minsplit=1 and minbucket=5, obtaining with those an 85,82% of accuracy for our model.

Our tree took into account all our variables when making decisions, for example the variable Sex, to see if a person was a boy or a girl, and also the Age to check if the person was smaller than 6.5 or not.



For this classifier we also created a confusion matrix to understand better our data. This shows that 81 people really died, but 8 of them were classified as survivals. And also, that 53 people survived, but 11 were classified as a dead. As we can see again is that the error is much more committed when classifying survivals rather than deaths.

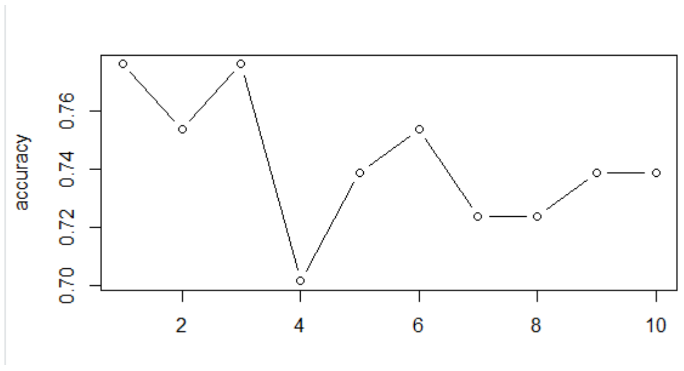
```
        predicted
class    0  1
      0 73 11
      1  8 42
```

K – Nearest Neighbors (KNN).

The third classifier we have worked with is **K – Nearest Neighbors** (KNN). This supervised learning algorithm is one of the simplest to use and to work with. Knn classifies coordinates into groups that have something in common, the similarity of its attributes.

To run this algorithm, we have one very important parameter to take into account, and it is the value of our k i.e. the number of nearest neighbors that we want to have. We say that it is important because the value of our k will be decisive in determining the efficacy of our model.

For this reason, the first thing that we do when working with knn is choosing the appropriate value for k. In order to do so, we created a “for loop” in which the program runs knn with different values of k, and then we plot a graph comparing the accuracy obtained according to the value of k. We obtained this plot.



As we have to put an integer value for our k, we can't take the first point of our graph. However, we can see that k = 3 takes also the highest value of accuracy. Therefore, we choose k to be equal to 3.

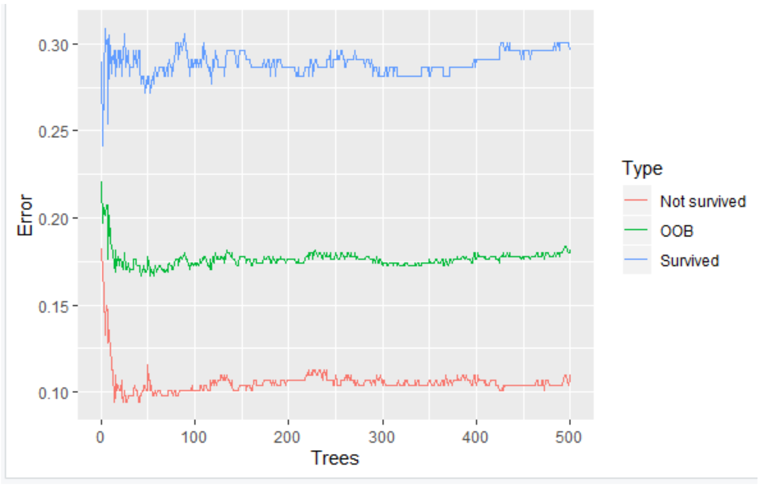
One important feature in knn is that we do not first create a model with the trainning set and then validate it with the testing set. Instead of that, we enter both sets at the same time in our model. Finally, with this model we obtained a 76.86% of accuracy.

Random Forest

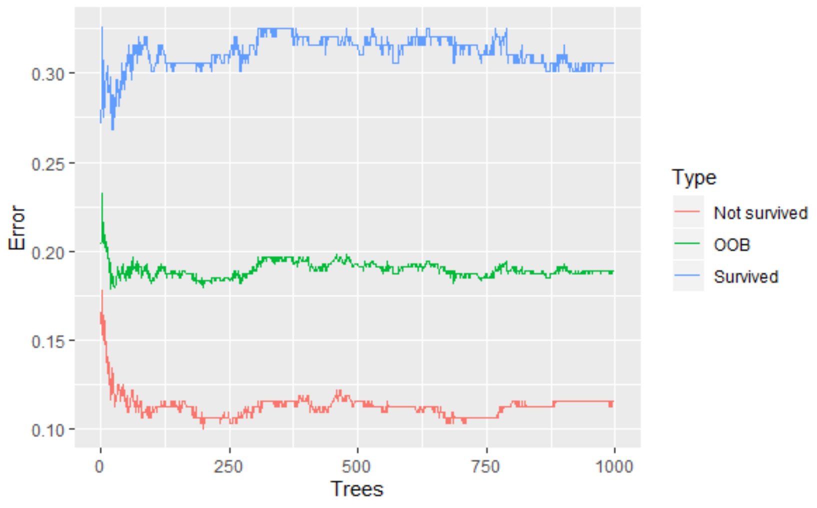
The last classifier that we used is **Random Forest**. This supervised learning algorithm works as a "strong learner" classifier by aggregating multiple outputs made by a diverse set of predictors. That is why it is said that Random Forest is built as an ensemble of Decision Trees.

To work with this classifier we have to take into account two important parameters, the number of trees, and the number of variables tried at each split.

The default number of trees is equal to 500 and the number of variables used is equal to 2. Now, we have to check if those default values are good enough for our model to be as much accurate as possible. To do so, we plot a graph of our random forest with the default values and we get this:

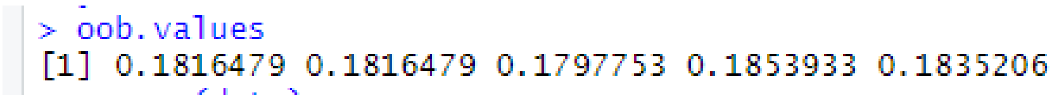


The first thing we can observe in this plot is that the error rate when classifying "survived" is greater than the error rate of "not survived". If we added more trees, would the error rate go down further?



The error rate varies a lot. However, we can see that from 875 trees more or less, the error rate stabilizes. Therefore, 1000 would be an optimal number for the trees.

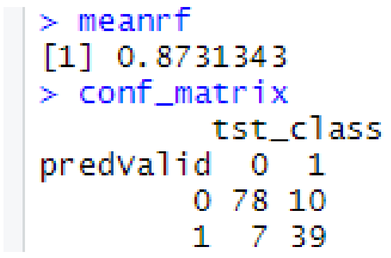
Now to make sure that we are considering the optimal number of variables, we create a “for loop” in which we assign to “mtry” different values from 1 to 10.



```
> oob.values
[1] 0.1816479 0.1816479 0.1797753 0.1853933 0.1835206
```

We can see that the first and the second values are the same. However, as 2 correspond to the default value, we are going to leave the same 2 because we can see that it is the value with which we obtain the lowest error.

Once we have chosen our parameters we create a model for our random forest. With our final model we have obtained this confussion matrix and our final accuracy which is equal to 87.31%.



```
> meanrf
[1] 0.8731343
> conf_matrix
          tst_class
predvalid  0  1
        0 78 10
        1  7 39
```

For our testing set, we have 134 passengers. Here we can see that 117 out of those 134 passengers were well classified in survived or not survived, whereas the other 17 passengers were missclassified.

Conclusions

To approach this assigment, we used the classifiers showed before, because we considered them the most appropiated for this project. Taking into consideration the four classifiers we have used to test our data, we have come up with the conclusion that the four of them were very realistic and accurate because our goal in the whole project was to obtain a percentage of survived people at least of 256/668=38%, because that is the amount of people that really survived (according to our data set), so it would have been bad to obtain in our prediction a lower percentage.

Comparing the performance of the models that we created, the ones with the lowest accuracy were k- Nearest Neighbours and Support Vector Machine models. Knn is said to be a lazy learner because there is no training involved in knn, however we can notice that it performed better than SVM, and we could attribute this result to the fact that our training data was much larger than the number of features, and in this cases most of the time knn performs better than svm. On the other hand, SVM performs really good at handling outliers, but one thing that we have to be careful with is with the tunning of its hyperparameters and kernels.

Now, the classifiers that gave us the best accuracy in the models for this project were Decision Trees and Random Forest. These classifiers work in a similar concept, and as we know Random Forest is essentially a collection of decision trees, in which the variance and bias are reduced and therefore it gives a better accuracy and a more generalized solution. This is what we could see in our project because Random Forest gave us the best accuracy for this assignment being it equal to 87.31%

Choosing the best classifier always depends on the data set that we are working with, and for this case we chose Random Forest as our preferred model to predict on this data. Firstly, because it is evident that the highest accuracy was given by this classifier. Furthermore, Random Forest is less prone to over-fitting because each tree of the ‘forest’ is based on a random sample of the training data and therefore they are diverse.

We consider that our model will perform very good at predicting on future data, because as we mentioned before, we have gotten an accuracy of 87.31% which is really good to make predictions, and to make a quick estimation, for example if someone enter a data with let say 500 passengers, the model will definitely make an accurate prediction on 437 of them.

Finally, to make it easier to test our classifier with your secret data, we have created at the end a function called “my model” that allows you to introduce your data set, and it automatically selects the variables we have been working with, and calculates the prediction, considering our data as the train set, and yours as the test set. We have implemented in that function the random forest method because it was the one with the higher accuracy in our experiment. This function will return a prediction of the people that survived or not to the Titanic’s sink, based on our data and on our classifier with the parameters that we have stablished.