Mark Hession

A00239393

Data Mining & Machine Learning Module Assignment

Table of Contents

Contents

[Regression: 2](#_Toc25306306)

[1.1 2](#_Toc25306307)

[1.2 2](#_Toc25306308)

[1.3 3](#_Toc25306309)

[1.4 3](#_Toc25306310)

[1.5 4](#_Toc25306311)

[1.6 4](#_Toc25306312)

[Decision Tree:   
2.1 5](#_Toc25306313)

[2.2 5](#_Toc25306314)

[2.3 6](#_Toc25306315)

[2.4 6](#_Toc25306316)

[2.5 7](#_Toc25306317)

[2.6 8](#_Toc25306318)

[KNN: 9](#_Toc25306319)

[3.1 9](#_Toc25306320)

[3.2 9](#_Toc25306321)

[3.3 11](#_Toc25306322)

[3.4 11](#_Toc25306323)

[3.5 12](#_Toc25306324)

[3.6 13](#_Toc25306325)

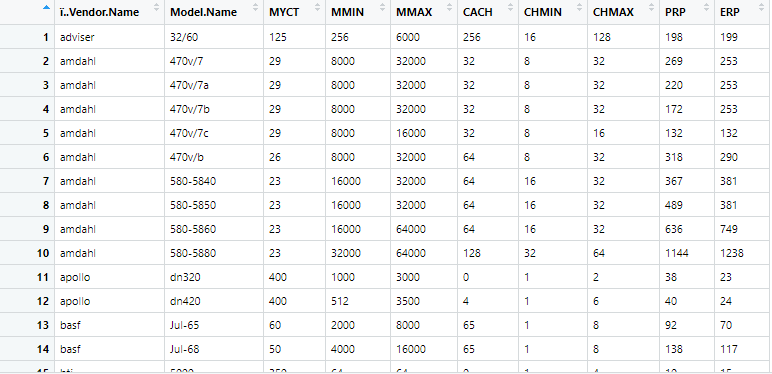
# Regression:

# 1.1

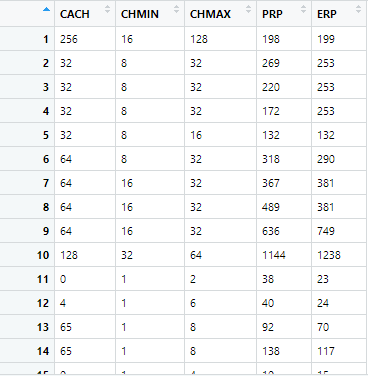
This dataset is about relative CPU performance data, described in terms of its cycle time, memory size, etc. The data we will be using from the data set is CACH: cache memory in kilobytes (integer)CHMIN: minimum channels in units (integer), CHMAX: maximum channels in units (integer), PRP: published relative performance (integer), ERP: estimated relative performance from the original article (integer). We will be predicting the ERP of a system based on its CACH/CHMIN/CHMAX/PRP.

# 1.2

This is my dataset prior to any alterations.

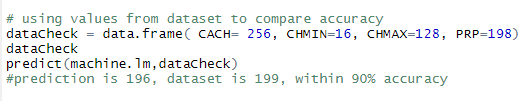


This is my dataset after it is altered to suit the regression process better.



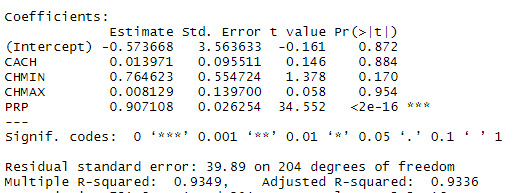
# 1.3

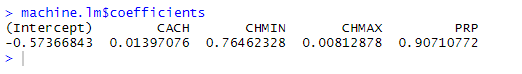
There is no training/testing data in the way there is for decision tree and knn. Instead, I used data from my dataset to compare to my predicted outcome to compare the accuracy. This is explained in more detail later.



# 1.4

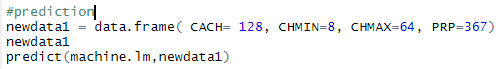




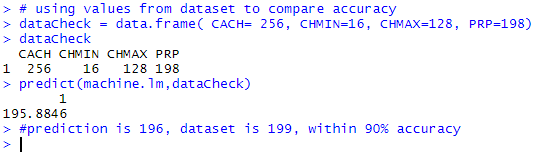
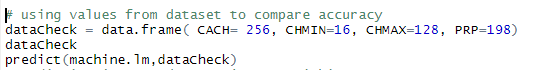
From our model generation, where we will try to predict the ERP, we can see our R-Squared value is 0.93, giving us a 93% accuracy for our prediction, which we will test later.

# 1.5

We can see that based on our new data frame inserts of newdata1, we have a predicted ERP of 341.



When we test our accuracy of our regression, we can take values from our original dataset and see how the ERP compares to our prediction.



Our dataset had an ERP of 199, our predicted ERP for the same values was 196, which is within the 93% accuracy bounds from before.

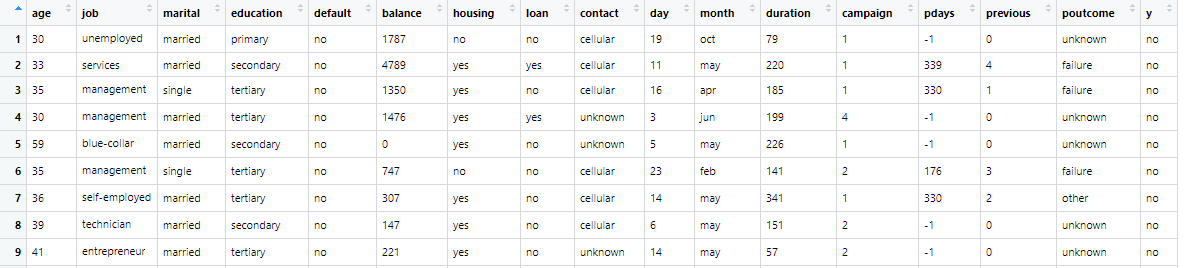
# 1.6

From our results based on our model, we can see when passing in the values CACH, CHMIN, CHMAX, PRP, we have a 93% accurate result, as proved when we inserted data from our dataset and compared it to our prediction result. For our dataset, to be able to predict the ERP based on these values is beneficial to the consumer as it gives the Estimated Real Performance of a CPU, based off the CPUs cache information, and you can compare it to the PRP of the CPU. This gives consumers an accurate estimate of what type of performance they can expect for the CPU they will buy. The biggest take away is the connection between the PRP and the ERP, which seem to have a difference which will scale, depending on the values inserted. The more modest and realistic a value you insert for PRP like in our original dataset you get a more accurate representation of what the ERP will be and how they compare as marketing vs actual performance.

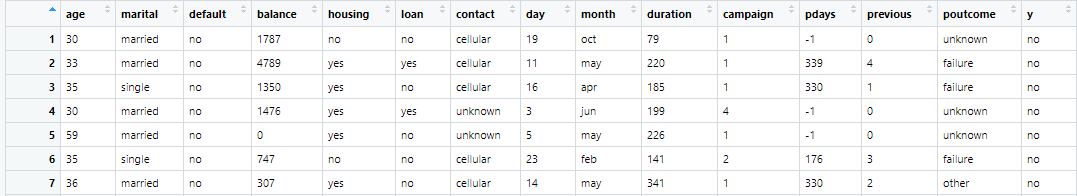
Decision Tree:  
2.1 – For this analysis, I have chosen a dataset which is related with direct marketing campaigns (phone calls) of a Portuguese banking institution. The goal is to predict if a client will say yes or no to a term deposit subscription. The dataset includes several details such as the client’s age/job/education along with banking details such as if they have a loan and what type of contact, they have. The decision tree will show the branching paths based on the client’s information and predict how likely they are to say yes or no based on that information.

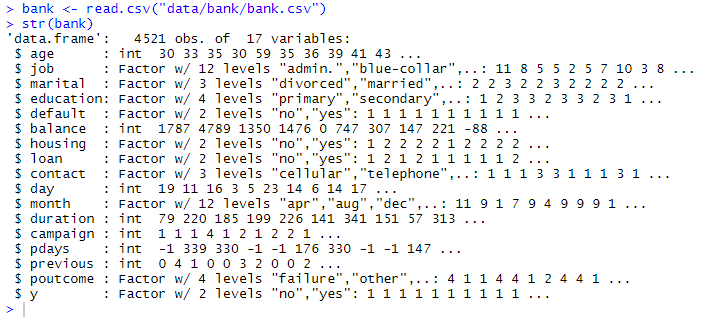
# 2.2

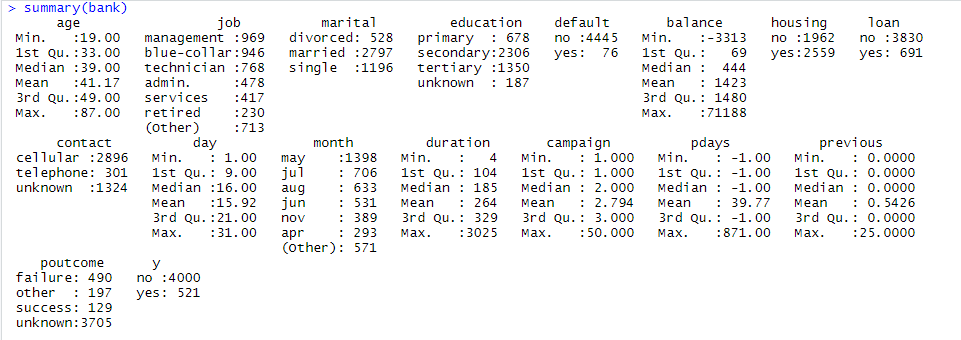
Bank Dataset before alterations



Bank dataset after alterations



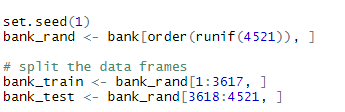




# 2.3

Splitting the data frames.

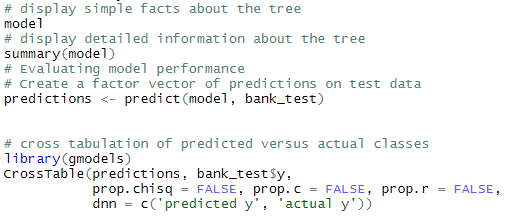
For the training and testing, I split it 80/20, 80% for training and 20% for testing.

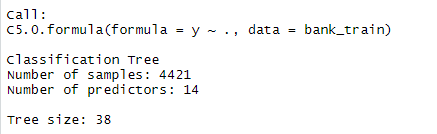




# 2.4

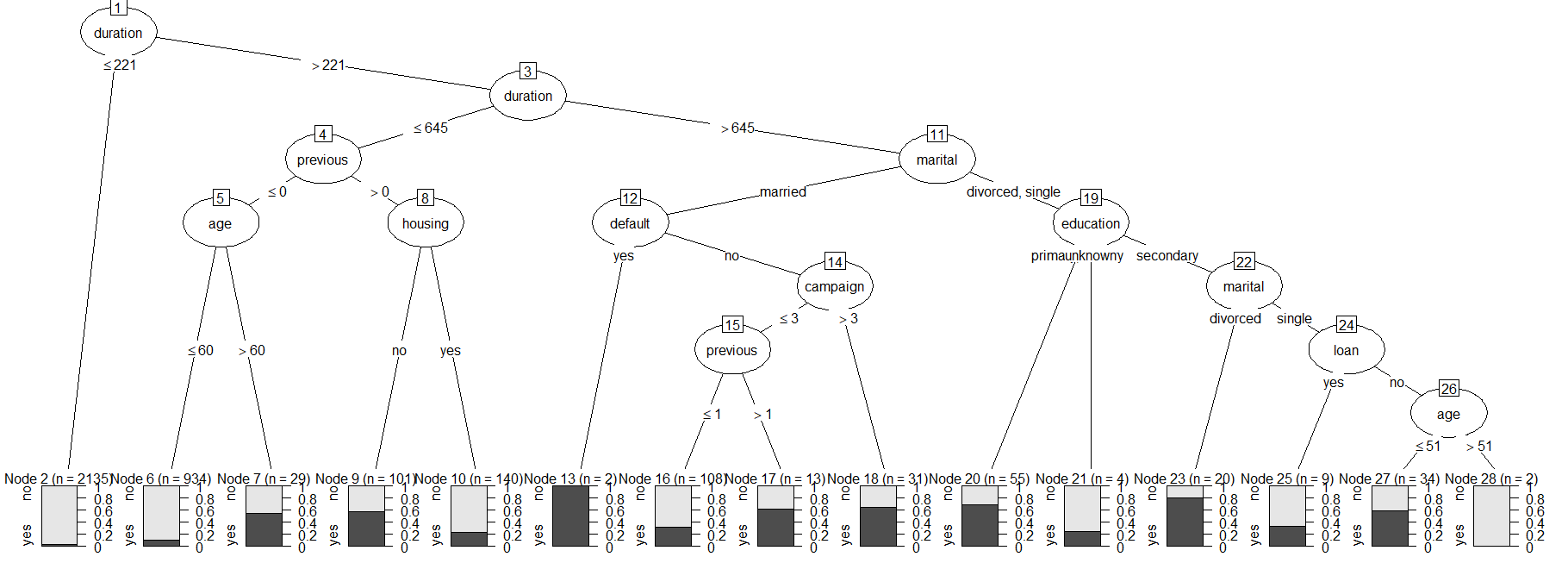
Model

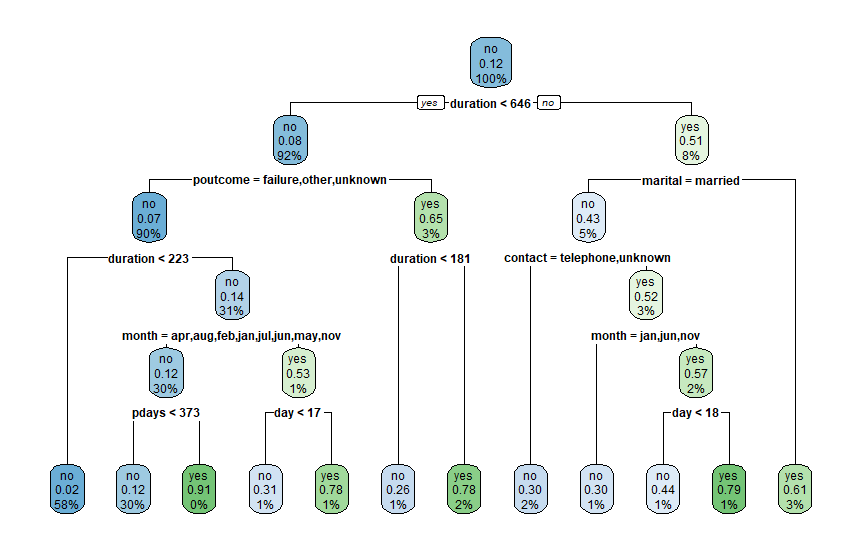


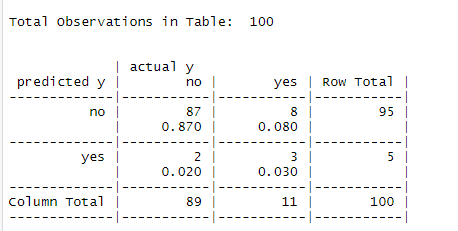


# 2.5

Predictions







# 2.6

Evaluation

From our prediction, we can see the branching paths on the decision tree, depending on certain data per client we can see how that influences the outcome. An example is people who had a contract duration over 646 and were married said yes and they made up 3% of the total amount of people in the dataset, but those that were unmarried could be split into two groups, those who had a campaign lower or higher than 4, being 3% no and 1% yes.

The interesting data in my opinion, is seeing the difference between those who are married/divorced/single. Married people typically said yes, especially those who already had credit. Those without credit weren’t as overwhelmingly responsive to the term deposit but even with factoring in their campaign and previous contact amount, they were just over 50% likely to say yes. This is presumably new couples looking for finance for a home together or maybe raise and fund having children.

There is also a correlation between the age of the client and their likelihood of taking on a term deposit. Those who were above or equal to 60 years were over 90% likely to say no, while those under 60 were approximately likely to say no 45% of the time, though this is dived into deeper when we factor in their marital status and if they already have a loan. Those who were single, had no personal loan and were older than 51 years of age were 99% likely to say no but those under or equal to 51% were around 55% likely to say yes.

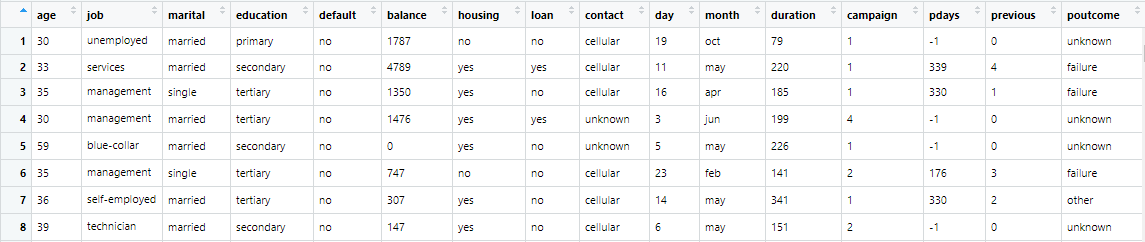
# KNN:

# 3.1

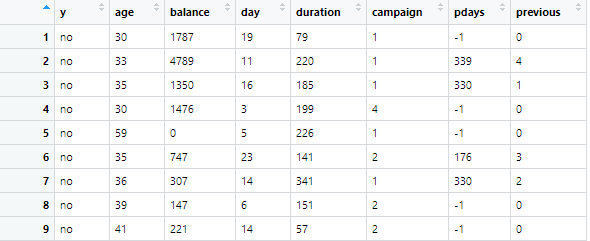
For my KNN prediction, I used the same dataset as decision trees, to compare the results given for each. Since there is no model for KNN, I will only be able to compare the reported number of success/failures and compare them to the paths of the decision tree. We are still trying to predict the likelihood of a yes or no response from a banks client on if they would like a term deposit or not. To predict this, we use the data from the dataset of clients, a mix of yes and no responses.

# 3.2

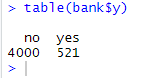
Dataset before alterations

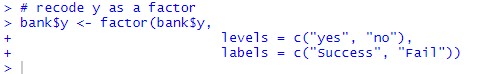


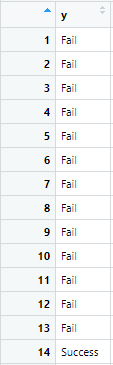
Dataset after alterations



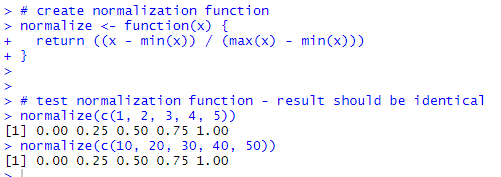
I removed all non-numerical columns and reordered so the “y” column, which is soon to be my success/failure column, is at the front, for simplicity sake down the line.

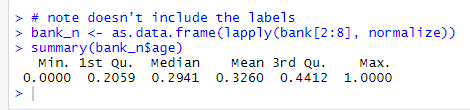






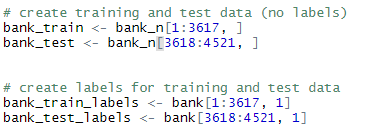
Creating the normalize function

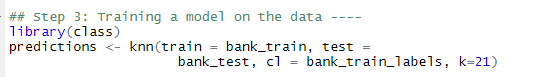




# 3.3

For the training and testing split, I decided to go 80/20 again like for decision tree so I can more accurately compare the results.

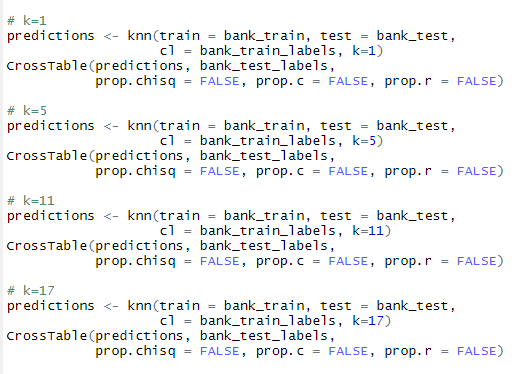




# 3.4

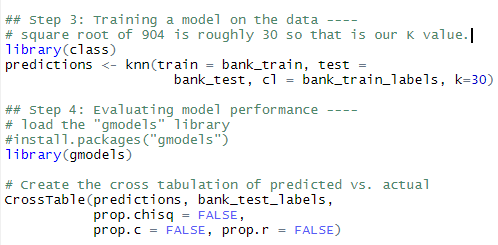
Model

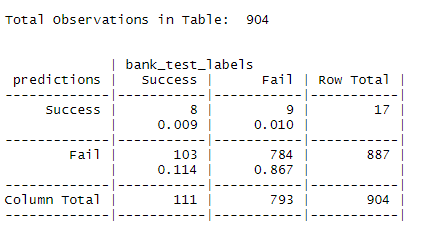
Knn does not have a model. It is about finding the best K value. I tried out multiple K values before deciding on 30, which is roughly the square root of 904 ( the number of observations), which is typically the best value to use.



# 3.5

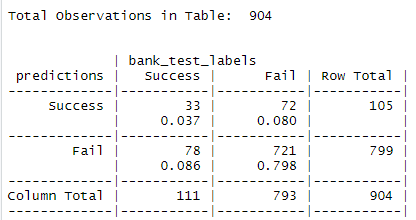
K=30 gave me the highest accuracy with 87.2%



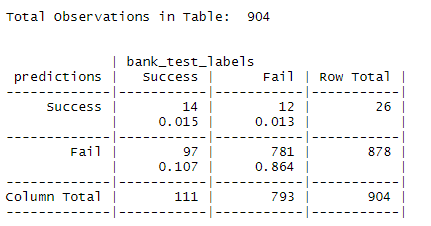


Some other cross table results with various K values:

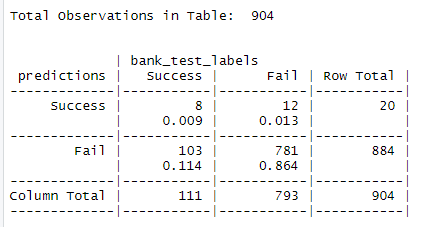
K=1



K=11



K=27



# 3.6

Evaluation.

With KNN, it is difficult to define the concepts, you just must take the result you get and see what you can observe from it. When we look at the cross table result for this dataset, we can see a significant amount of failures compared to successes. This is in line with what we expected based on our decision tree prediction, which had far more failures than successes and we could path what factors lead to choosing one over the other. When we calculate our accuracy by adding (8+781)/904 and multiply that by 100, we can see we have 87.2%, which is a very good accuracy rating.