Decision Tree and Random Forest on a Kaggle banking dataset

Introduction

In this report, we are looking at how a Decision Tree and Random Forest can be applied to a marketing banking campaign dataset to see how well it predicts a prospective customer's likelihood to open up a bank account.

Figure 1 - Original Banking Data 2
Figure 2 - Original Missing Data 2
Figure 3 - Decision Tree - minibucket = 7........ 3
Figure 4 - Decision Tree Confusion Matrix........ 3
Figure 5 - ROCs........ 4
Figure 6 - Random Forest Confusion Matrix 5
Figure 7 - Gini List........ 6

Decision Tree

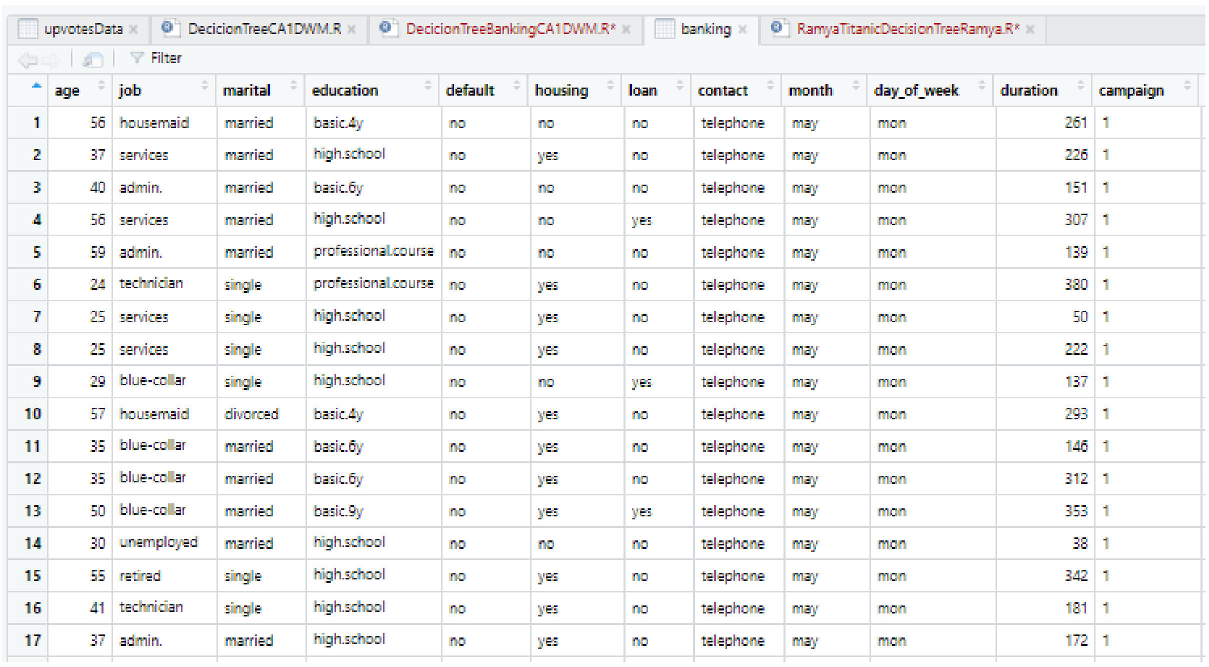
Decision trees are a supervised learning algorithm that uses a set of binary rules to arrive at a target value. It is used in the following type of scenarios:

- Credit scoring loan applicants
- Marketing campaign success
- Medical diagnosis based on symptoms

Decision trees divides data into smaller and smaller sections to identify patterns that can be applied to unseen data in a predictive way. Decision trees are widely used because they are easily understood and they have a high explanatory power. It works like a flow chart with a series of logical decisions. Decisions trees are made up of a series of nodes with the terminal nodes at the end of branches containing the classification/result. They can model almost any data and perform well despite their seemingly simplistic design. Decision trees use a heuristic called recursive partitioning which is also known as "divide and conquer". It works by repeatedly partitioning the data into multiple sub-spaces so that the outcomes in each sub-space is homogeneous. If is not unlike the children's 20 questions game. It starts off with: "is it an animal", "does it have more than 5 legs" and so on. That is how decision trees work (Paul Pandey: Medium, 2018)

The dataset used in this project comes from Kaggle
https://www.kaggle.com/volodymyrgavrysh/bank-marketing-campaigns-dataset

This dataset documents a marketing campaign with the aim of getting participants to open a deposit bank account.



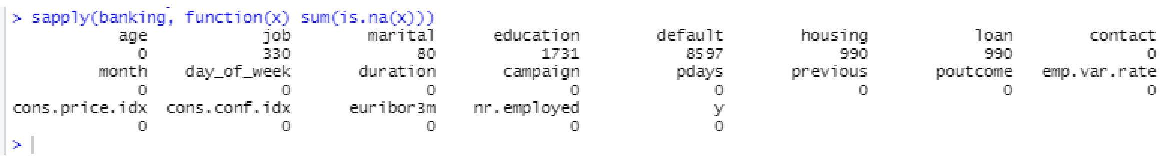
upvotesData | DecicionTreeCA1DWM.R | DecicionTreeBankingCA1DWM.R* | banking | RamyaTitanicDecisionTreeRamya.R*

Filter

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign
1	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	261	1
2	37	services	married	high.school	no	yes	no	telephone	may	mon	226	1
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	151	1
4	56	services	married	high.school	no	no	yes	telephone	may	mon	307	1
5	59	admin.	married	professional.course	no	no	no	telephone	may	mon	139	1
6	24	technician	single	professional.course	no	yes	no	telephone	may	mon	380	1
7	25	services	single	high.school	no	yes	no	telephone	may	mon	50	1
8	25	services	single	high.school	no	yes	no	telephone	may	mon	222	1
9	29	blue-collar	single	high.school	no	no	yes	telephone	may	mon	137	1
10	57	housemaid	divorced	basic.4y	no	yes	no	telephone	may	mon	293	1
11	35	blue-collar	married	basic.6y	no	yes	no	telephone	may	mon	146	1
12	35	blue-collar	married	basic.6y	no	yes	no	telephone	may	mon	312	1
13	50	blue-collar	married	basic.9y	no	yes	yes	telephone	may	mon	353	1
14	30	unemployed	married	high.school	no	no	no	telephone	may	mon	38	1
15	55	retired	single	high.school	no	yes	no	telephone	may	mon	342	1
16	41	technician	single	high.school	no	yes	no	telephone	may	mon	181	1
17	37	admin.	married	high.school	no	yes	no	telephone	may	mon	172	1

Figure 1 - Original Banking Data

Figure 1 shows a snapshot of the data. In Figure 2 - Original Missing Data, we can see the result of filtering the data for missing data. Clearly, there is a considerable amount of missing data.



```
> sapply(banking, function(x) sum(is.na(x)))
           age            job        marital      education        default        housing           loan        contact 
             0            330             80           1731           8597            990            990              0 
         month    day_of_week       duration       campaign          pdays       previous       poutcome   emp.var.rate 
             0              0              0              0              0              0              0              0 
cons.price.idx  cons.conf.idx      euribor3m    nr.employed              y 
             0              0              0              0              0 
> 
```

Figure 2 - Original Missing Data

As there are 41,188 observations in the dataset and we know that the random forest algorithm does not work with NAs, we will use the tidyverse filter to remove the unwanted NA observations. This action brings the number of observations down to 30,488. The success of the marketing campaign is a customer opening up a deposit account. However, the successful outcome of opening up a deposit account in the trimmed down dataset is only 13.45% so the data is skewed towards not opening up a bank account. This skewness is a limitation for predicting the success of other campaigns based on this data.

When using the decision tree model, the data was split between train and test in an 80:20 split using random sampling.

In Figure 3 - Decision Tree - minibucket = 7, we can see that the algorithm has chosen nr.employed as the root node and duration and pdays as the most relevant nodes.

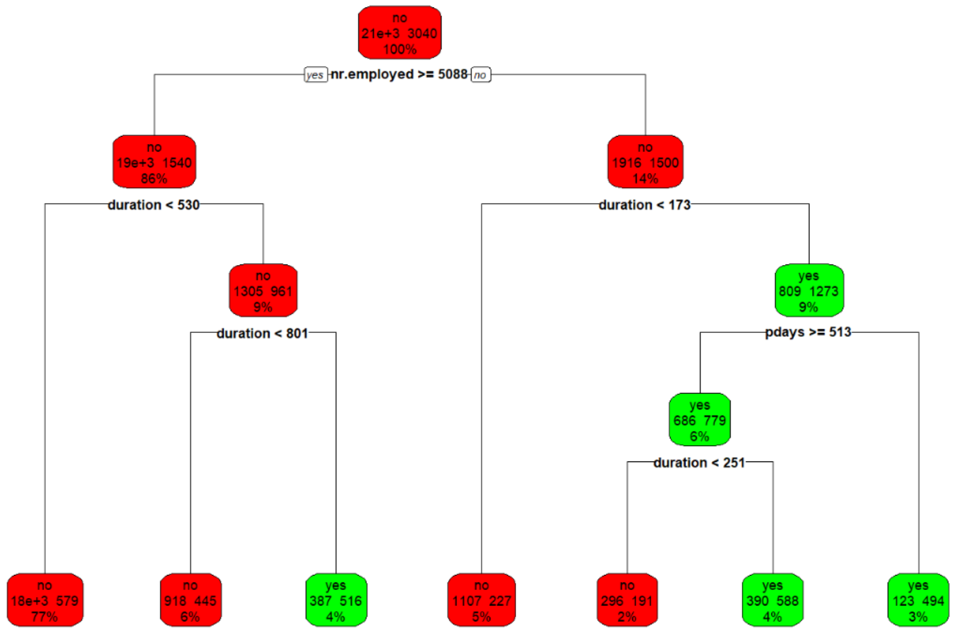
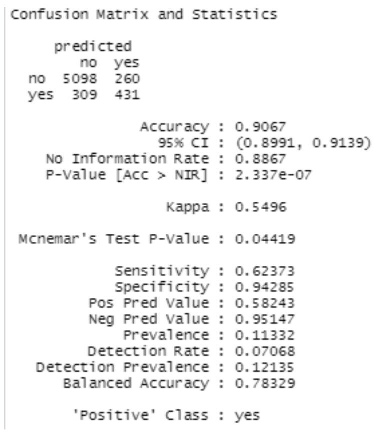


Figure 3 - Decision Tree - minibucket = 7

If a minbucket setting is too small, more splits are generated and overfitting can occur. If too large, the model is simpler but accuracy suffers. (Paul Pandey: Medium, 2018).

You can also prune the decision tree based on best cp value. Pruning the tree helps avoid overfitting because you end up with a smaller tree with the resultant less cross validation error (Asansha Sharma: Edureka, 2019).



```
Confusion Matrix and Statistics

     predicted
        no  yes
  no  5098  260
  yes  309  431

               Accuracy : 0.9067
                 95% CI : (0.8991, 0.9139)
    No Information Rate : 0.8867
    P-Value [Acc > NIR] : 2.337e-07

                  Kappa : 0.5496

 Mcnemar's Test P-Value : 0.04419

            Sensitivity : 0.62373
            Specificity : 0.94285
         Pos Pred Value : 0.58243
         Neg Pred Value : 0.95147
             Prevalence : 0.11332
         Detection Rate : 0.07068
   Detection Prevalence : 0.12135
      Balanced Accuracy : 0.78329

       'Positive' Class : yes
```

Figure 4 - Decision Tree Confusion Matrix

The result we are most interested in is Yes and that is the positive value and the prediction value for the positive Yes result (Sensitivity) is poor at only 62%, so that is not a good result. The Kappa is only 55% and this is a better indicator of accuracy than the general Accuracy of 91%, because the data is heavily skewed towards No (Jason Browsnlee: Machine Learning Mastery, 2016).

There is an AUC (area under the curve) measurement that is useful measure of an algorithm's effectiveness. It tells us how well we discriminate between two groups. In Figure 5 - ROCs, we can see the the TPR(sensitivity) ROC (receiver operating characteristic) and TNR(specificity) ROC curves. We are most concerned about the high TPR False negative rate.

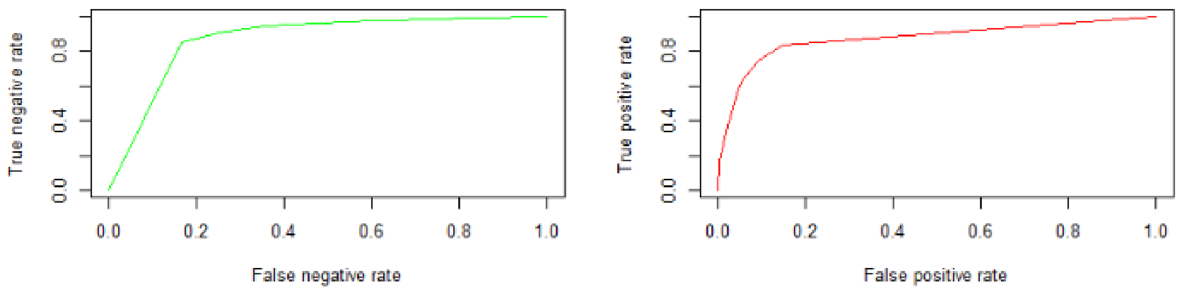
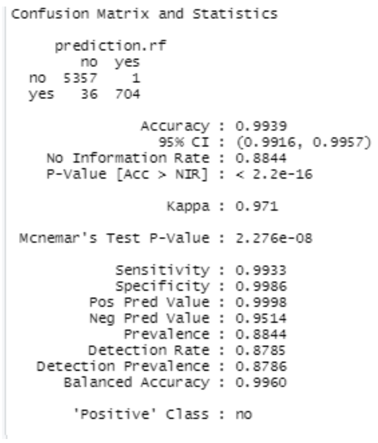


Figure 5 - ROCs

Random Forest

The Random forest is a large number of individual decision trees operating as an ensemble (Tony Yiu: Towardsdatascience.com, 2019). Each tree is a random forest and the tree with the best accuracy becomes the model's predictor. Crowd sourcing is the main idea behind Random Forest based on the principle "A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models" (Tony Yiu: Towardsdatascience.com, 2019). Random Forest adds diversity to an algorithm.

In Figure 6 - Random Forest Confusion Matrix, we can see that the Positive Prediction level is 99% and the Kappa value is similar at 97%. This clearly shows that this dataset is a good predictor for predicting customers opening up a deposit account and this is good news.



```
Confusion Matrix and Statistics

     prediction.rf
        no  yes
  no  5357    1
  yes   36  704

               Accuracy : 0.9939
                 95% CI : (0.9916, 0.9957)
    No Information Rate : 0.8844
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.971

 Mcnemar's Test P-Value : 2.276e-08

            Sensitivity : 0.9933
            Specificity : 0.9986
         Pos Pred Value : 0.9998
         Neg Pred Value : 0.9514
             Prevalence : 0.8844
         Detection Rate : 0.8785
   Detection Prevalence : 0.8786
      Balanced Accuracy : 0.9960

       'Positive' Class : no
```

Figure 6 - Random Forest Confusion Matrix

The Gini Index measures the degree or probability that a particular variable being wrongly classified when it is randomly chosen. It refers to its "impurity". A gini index of .05 denotes equally distributed elements across classes. When building a decision tree, ideally, you want the root node having a least gini index, see Figure 7 - Gini List where you can see that the duration factor has the biggest gini index.

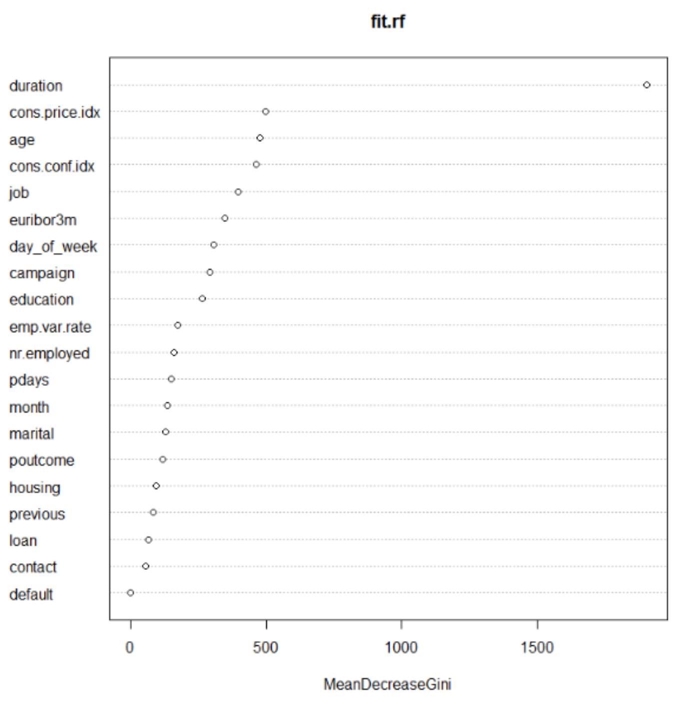


Figure 7 - Gini List

In Figure 7 - Gini List, the duration factor is highly biased as a variable, so it should not be a decision tree root and there might be a case to remove the duration field completely from the decision tree. Decision Tree analysis is trial and error process and a balance act removing variables that are considered irrelevant or are skewed in favour of one result such as duration variable.

References

Anon., 2020. *Package: ROCR.* [Online]
Available at: https://cran.r-project.org/web/packages/ROCR/ROCR.pdf
[Accessed 25 July 2020].

Asansha Sharma: Edureka, 2019. *Creating, Validating and Pruning Decision Tree in R.* [Online]
Available at: https://www.edureka.co/blog/implementation-of-decision-tree/#:~:text=Syntax%20%3A%20printcp%20(%20x%20)%20where,()%20function%20i.e.%20'xerror'.
[Accessed 25 July 2020].

Jason Browsnlee: Machine Learning Mastery, 2016. *Machine Learning Evaluation Metrics in R.* [Online]
Available at: https://machinelearningmastery.com/machine-learning-evaluation-metrics-in-r/
[Accessed 25 July 2020].

Paul Pandey: Medium, 2018. *A Guide to Machine Learning in R for Beginners: Decision Trees.* [Online]
Available at: https://medium.com/analytics-vidhya/a-guide-to-machine-learning-in-r-for-beginners-decision-trees-c24dfd490abb
[Accessed 25 July 2020].

Tony Yiu: Towardsdatascience.com, 2019. *Understanding Random Forest.* [Online]
Available at: https://towardsdatascience.com/understanding-random-forest-58381e0602d2
[Accessed 24 July 2020].

Wickham, H., 2007. *Reshaping Data with the reshape Package.* [Online]
Available at: https://vita.had.co.nz/papers/reshape.pdf
[Accessed 23 July 2020].

Wickham, H., 2011. *The Split-Apply-Combine Strategy for Data Analysis.* [Online]
Available at: https://www.jstatsoft.org/article/view/v040i01/v40i01.pdf
[Accessed 23 July 2020].

Wickham, H., 2014. *Tidy Data: Journal of Statistical Software.* [Online]
Available at: https://vita.had.co.nz/papers/tidy-data.pdf
[Accessed 23 July 2020].