* **Question to solve**

Q.19 Consider the other parameters that can be used to control the recursive partitioning

process. Read the documentation for them in the rpart.control() documentation. Also,

carry out an Internet search for more information on how to tweak the rpart() tuning

parameters. Experiment with values for these parameters. Do the trees that result make

sense with your understanding of how the parameters are used? Can you improve the

prediction using them?

* **Introduction**

At the very beginning of Internet, sending out commercial emails were not allowed. The first email spam was sent by Gary Thuerk in 1978 and 600 people received it. Email Spams has steadily grown since the early 1990s, and by 2014 spams has been estimated that it made up around 90% of email messages sent. Most of email spams are commercial in nature. Because of the expense of the spam is borne mostly by the person who receives it, it forms a perfect example of negative externality.

Every Email service provider has used anti-spam techniques to filter and refuse spam based on the content of the email. Spam detection is a classification problem, either Spam or not. In this project, we will build our own Spam filter with the dataset that has been utilized in the book of Data Science in R: A Case Studies Approach to Computational Reasoning and Problem Solving [1] (“the book” in the followings).

* **Background**

In order to build our own Spam filter, we use a dataset that has also been utilized in the book [1]. The dataset has more than 9000 emails that have been manually classified as Spam or not Spam by SpamAssassin[2] to make models. In the book [1], the dataset has been processed to add structure. We are using the same steps to clean up the data by regrouping them into header, body content and attachments (if any). This process will be automated by writing functions and then later being used on the whole dataset. Once the dataset has been processed, all unique words will be used to build a word dictionary which the frequency of words in Spam versus not Spam will be calculated. This process contributes to the establishment of the Spam detection classifier.

We are also required to explore **rpart** library in R so that we can build an improved decision tree model and understand how each parameter affects the decision tree outcome by tuning each of them. In the followings, we will tune one parameter at a time, visualize and analyze the change in the tree outcome, then finish the project by tuning all parameters to provide a better decision tree model.

* **Method**
  + **Classification Metrics**

Before we jump into the parameter tuning and modelling, we have to decide the performance metrics to evaluate how well the model performs. As we mentioned above Spam detection is a classification problem, a good performance is that the model can get as few as false positive and false negative as possible. Common classification metrics are accuracy, precision, recall and F1 score.

1. Accuracy: The ratio of number of correct predictions to the total number of predictions made. But it only works well when there are equal number of samples in each class.

Accuracy = the number of correct predictions/ total number of predictions = (True Positives + False Negatives)/ Total Number of Samples

1. Precision: The ratio of the number of correct positives to the number of positive results predicted by the model.

Precision = True Positives/ (True Positives + False Positives)

1. Recall: The ratio of the number of correct positives to the number of all samples that should have been tagged as positive)

Recall = True Positives/ (True Positives + False Negatives)

1. F1 Score: It is the harmonic mean between precision and recall. It is trying to find the balance between precision and recall. F1 score is a number that is between 0 and 1. It indicates how precise and robust our model is. A high F1 score should be preferred.

F1 Score = 2 ×

* + **Rpart Library**

In this project we are using Rpart to create a decision tree. In R, the Rpart library (Recursive Partitioning and Regression Trees) is the algorithm implementation of the CART algorithm (Classification and Regression Trees). Decision trees use gini index as metric to evaluate the splits and determine features that have the greatest significance. A Gini score gives an idea of how good a split is by how mixed the classes are in the two groups created by the split. Gini score is calculated as[3]

The rpart.control() is the control details of rpart algorithm which we have a list of various parameters that control aspects if rpart fit. There are 9 parameters that can be tuned to achieve different outcomes and we will talk about the most important ones that we have searched on the Internet.

1. **Minsplit**

The minimum number of observations in the parent node that must exist in a node in order for a split to be attempted. In other words, the number of observations in one child node. For example, in R, this value is 20 by default, if there is any node that has less than 20 observations, it will be tagged as a terminal node and no split could happen. [4][5] Thermionically, if nothing is given to minsplit at all, not even that default value 20, the tree will go on forever till there is no observation left in a node to split. It is a parameter that prevents the algorithm from overfitting. (Here I need some plot, something current cell 57 to show the relationship between minsplit and split). Maybe number of minsplits increases, number of splits decreases.

1. **Minbucket**

The smallest number of observations that are allowed in a terminal node. If a split decision breaks the data into a node with less than the given minbucket, it will not happen. It sounds very similar to minsplit. While minsplit informs the algorithm the number of observations the split could happen, the minbucket tells when to tag a node as terminal node. There is a relationship between minsplit and minbucket that If only one of minbucket or minsplit is specified, the code either sets minsplit to minbucket\*3 or minbucket to minsplit/3, as appropriate [4][5]. Again, this parameter is just like minsplit, it prevents the algorithm from overfitting by limiting the number of child nodes that can exist in a tree. (I also need some plot for this part)

1. **Cp**

Complexity parameter is a stopping parameter. It speeds up the search for splits since it locates the splits that do not meet up with the criteria. The complexity parameter (cp) in rpart is the minimum improvement in the model needed at each node. It’s based on the cost complexity of the model defined as

It adds up the misclassification at every terminal node in a tree. Then it multiplies the number of the splits with a penalty term lamda. After this, add it to the total misclassification. This lamda is achieved by cross-validation. In R, if we use printcp(), it will show a scaled version of lamda over the misclassification rate of overall data[6]. Therefore, a smaller value will make the tree more complex. A max value is 1 and it creates a tree with zero splits. (plots needed)

1. **maxdepth**

It is used to set the maximum depth of any node of the final tree. In other words, it tells the algorithm, how many levels from a root node it can go down. For example, if we have set the maxdepth as 2, the tree can go down at most two levels. (plots needed) Therefore, finding the balance of this value is needed. If a low maxdepth value is given to the algorithm, the tree will be relatively shallow and only identify the features that have greatest impact on the model. On the other hand, if a large value is given, the tree will be improved till the point that the tree will go too long and has overfitting problem. (plots needed to show the concept)

1. **xval**

This parameter is simply the number of cross validations we need to perform on the dataset. Cross validation is always preferred in any data science project. It is a great way to train the model on several datasets partitioned into k-folds and then test it on the test set. This process contributes to improve the model performance. Normally it is a value between 5 and 10, although we use 10 folds valiation most of the time.

* **References**

[1] Data Science in R: A Case Studies Approach to Computational Reasoning and Problem Solving', Part1 Chapter 3, Using Statistics to Identify Spam

[2] [http://spamassassin.apache.org](http://spamassassin.apache.org/)

[3] <https://medium.com/deep-math-machine-learning-ai/chapter-4-decision-trees-algorithms-b93975f7a1f1>

[4] <https://stat.ethz.ch/R-manual/R-devel/library/rpart/html/rpart.control.html>

[5] <https://www.gormanalysis.com/blog/decision-trees-in-r-using-rpart/>

[6] <http://www.learnbymarketing.com/tutorials/rpart-decision-trees-in-r/>