Problem #1 Straightforward Training of DT

The below decision tree predicts whether a patient is likely to be diagnosed with breast cancer, regarding the characteristics of the cell nuclei present in the image. Since all predictors are quantitative variables, the model outputs a regression tree with recursive binary splitting. After dropping unnecessary attributes and rows with missing values and converting class label attribute into factor; the preprocessed dataset contains 683 observations across 10 attributes. The set of tuples in breast_cancer_updated.csv is associated with class labels, *Class*, with each taking one of two values, *malignant and benign*. As the goal of any supervised learning is to build models that perform well on unseen data, a random sampling within the levels of *Class*, with a 0.7 and 0.3 ratio respectively, is applied as a model validation process.

As the resulting *train_bc* dataset contains 479 observations, with 64% of them diagnosed as *benign*, stratified cross-validation is applied to reduce randomness and prevent overfitting. Based on the output, the model has an accuracy rate is 93.72% with Cp parameter set to 0.004, meaning 93.72% of the patients were correctly classified. The associated Kappa value of 0.86 indicates the instances classified by the DT classifier matched the data labeled as ground truth.

```
> # a. Apply decision tree & report the accuracy using 10-fold cross validation.
> ## splitting into train&test set
> set.seed(961)
> index= createDataPartition(y=bc$Class, p=0.7, list = FALSE)
> train_bc <- bc[index,]
> test_bc <- bc[-index,]
> idClass <- createFolds(train_bc$Class, k=10, returnTrain = TRUE)
> train_control<- trainControl(index=idClass, method = "cv", number = 10)
> ##Fit the model
> tree1<- train(Class ~., data=train_bc, method="rpart", trControl=train_control)
> tree1
CART

479 samples
  9 predictor
  2 classes: 'benign', 'malignant'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 431, 431, 431, 432, 431, 431, ...
Resampling results across tuning parameters:

  cp           Accuracy   Kappa
  0.003968254  0.9372304  0.8621837
  0.044642857  0.9246861  0.8336302
  0.803571429  0.7184976  0.2196649

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.003968254.
```

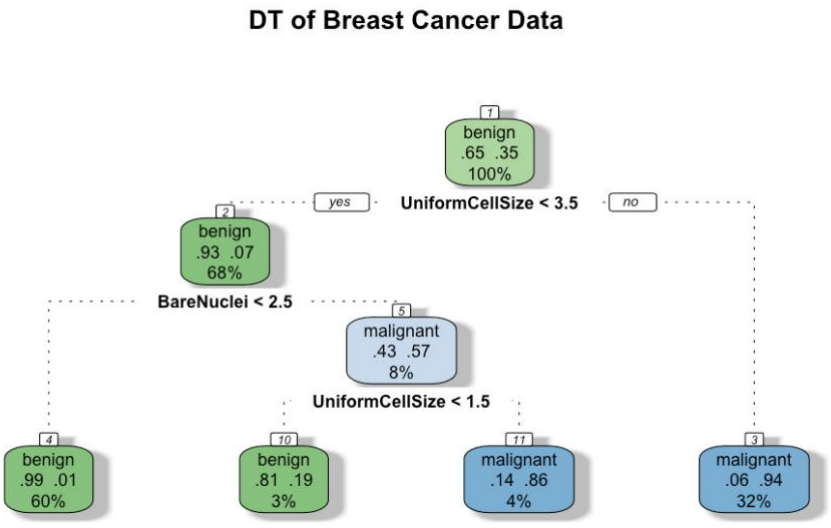
By checking the prediction accuracy on the testing dataset, the output accuracy rate is 97.06%, much higher than the no-information rate. Furthermore, the statistics by class and reference show the model is slightly biased, as incorrectly labeled data from the minority class *malignant* as *benign.* Worth noting, the test accuracy is slightly higher than both the train and model accuracy, indicating a high model quality, as shown below.

```
> confusionMatrix(test_bc$Class, pred_tree1)
Confusion Matrix and Statistics

          Reference
Prediction  benign malignant
  benign       128         5
  malignant      1        70

               Accuracy : 0.9706
                 95% CI : (0.9371, 0.9891)
    No Information Rate : 0.6324
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.936
```

```
> confusionMatrix(train_bc$Class, pred_tree1_train)
Confusion Matrix and Statistics

          Reference
Prediction  benign malignant
  benign       298        13
  malignant      5       163

               Accuracy : 0.9624
                 95% CI : (0.9413, 0.9776)
    No Information Rate : 0.6326
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.9184
```

Looking at the visualization of the decision tree model as below, each node box represents the classification, the probability of each class at that node, and the percentage of observations used at that node. The root node is listed as 1, the internal nodes are labeled as 2 and 5, and the rest boxes are considered as the leaf nodes. The green node boxes are cases where the majority of patients are benign, whereas the blue node boxes are cases where the majority are malignant. All terminating conditions are labeled under the boxes.

As the *rpart* parameter uses the Gini index to make binary splits for all continuous attributes, the attribute *UniformCellSize* at a value of 3.5 is selected as the first split point with the maximized reduction in impurity. The leaf node 3 indicates there are 94% of patients with *UniformCellSize* is greater than 3.5 are diagnosed as malignant. On the other hand, the resulting internal node 2 indicates there are 68% of patients' *UniformCellSize* is smaller than 3.5, while 93% of those are diagnosed as benign. Meantime, the next split point selected is *BareNuclei* at a value of 2.5, in which the split results in one internal node labeled as 5 and one leaf node labeled as 4. The leaf node 4 indicates that 99% of patients with *BareNuclei*'s value smaller than 2.5 are diagnosed as benign. Lastly, the final split point at node 5 is *UniformCellSize* at a value of 1.5, indicating the 86% of patients with *UniformCellSize* greater than 1.5 are diagnosed as malignant.



IF-THEN Rule

- IF the *UniformCellSize* of the patient is greater than the threshold value of 3.5, THEN the patient has a 94% likeliness to be diagnosed as malignant.

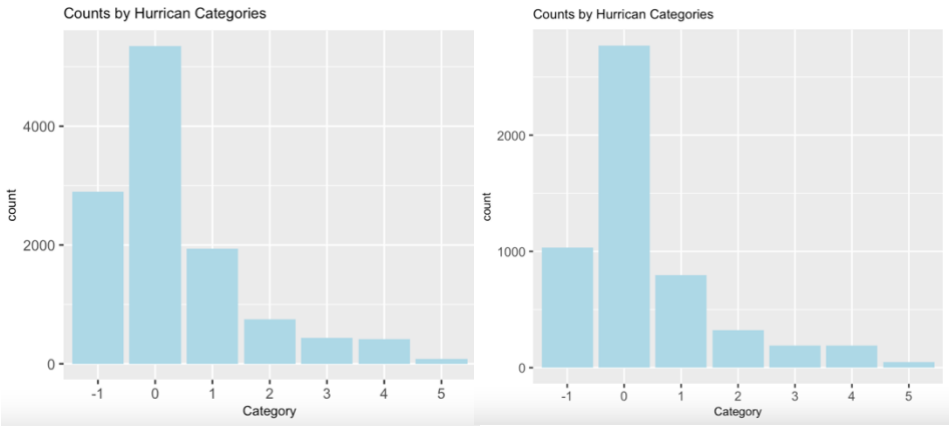
  IF *UniformCellSize >3.5, THEN Class = malignant*

- IF the *BareNuclei* of the patient is greater than the threshold value of 2.5, THEN the patient has a 57% likeliness to be diagnosed as malignant. Out of those patients, whose *UniformCellSize* is greater than 1.5; THEN the patient has an 86% likeliness to be diagnosed as malignant

  IF *BareNuclei >2.5 AND UniformCellSize >1.5 , THEN Class = malignant*

Problem #2 Decision Tree with Tuned Hyperparameter

In the *storms* subset of the NOAA Atlantic hurricane database, 198 tropical storms have been measured every six hours during the lifetime of a storm across 13 attributes. To predict the class label attribute- *category*, of an unseen storm, the decision tree model is built to classify based on the 12 predictors. Upon data exploration, the *storms* dataset has a low degree of completeness, as 55.89% of rows contain at least one missing value. After dropping rows with missing values and converting nominal attributes into categorical, the preprocessed dataset , *stormsDS*, contains 5,350 observations across 13 attributes. The set of tuples in *stormsDS dataset* is associated with class labels, *category*, with each taking one of seven values, ranging from -1 to 6.



To examine possible class imbalance in the class label- *category*, bar plots are used to visualize the proportions that each class accounts for. Looking at the above bar plots of raw and cleaned dataset, storms that were labeled as 0 outnumbered the storms in other classes by a great percentage. In the *stormsDS* dataset especially, 51.79% of the storms are classified as class 0, whereas storms that are in class 5 only account for 0.87% of the entire dataset. The highly imbalanced distributions of classes suggest that, when applying cross-validation, random sampling would be prone toward imbalanced samples, resulting in a low model quality in

predicting items in minority classes. Hence, stratified cross-validation is applied for the following model-building steps.

I. Straightforward Decision Tree with CART

Using 10-fold stratified cross-validation with tuned hyperparameters, the resulting decision tree model on the basis of Gini's impurity index, has an accuracy rate of approximately 85.94%, with a kappa value of 0.78, indicating a high agreement between the predicted and actual label for the model. However, as class imbalance is presented in the dataset, the model quality is expected to be heavily biased.

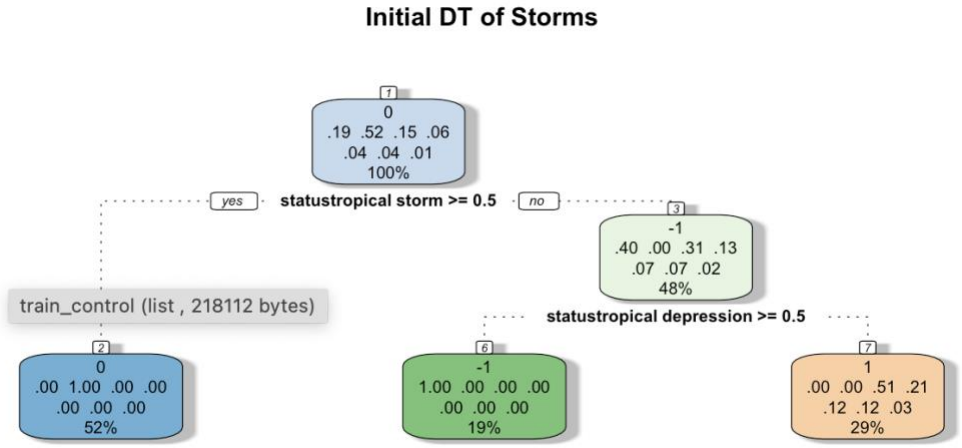
```
> idCategory <- createFolds(stormsDS$category, k=10, returnTrain = TRUE)
> train_control<- trainControl(index=idCategory, method = "cv", number = 10)
> hypers<- rpart.control(minsplit = 5, maxdepth = 2, minbucket = 3)
> ##Fit the model
> stormDT<- train(category ~., data=stormsDS, method="rpart1SE",
+                 control=hypers, trControl=train_control)
> stormDT
CART

5350 samples
  12 predictor
   7 classes: '-1', '0', '1', '2', '3', '4', '5'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 4816, 4814, 4816, 4813, 4814, 4815, ...
Resampling results:

  Accuracy   Kappa
  0.8594412  0.784229
```

The confusion matrix below indicates that, although the model performs accurately for Classes -1 and 0, it neglected the rest of the classes. The results indicates the *stormsDS* dataset is lacking data with the classification of Classes 2~5, which is leading to the NAs in the statistics by *Class*. Specifically, the specificity index of Class 2-5 is calculated as the number of correct non-Class predictions divided by the total number of non-Class items for each class, whereas their values greater than 0.9 indicates the model quality is high when predicting Classes -1 and 0. More importantly, as the prevalence indicates the number of true cases actually occurred out of all the observations, the value of 0 across all Classes 2~5 indicates there are no true cases.

```
Statistics by Class:

                     Class: -1 Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
Sensitivity             1.0000   0.9996   0.5145       NA       NA       NA       NA
Specificity             1.0000   1.0000   0.9997  0.93963  0.96449   0.9643 0.991215
Pos Pred Value          1.0000   1.0000   0.9987       NA       NA       NA       NA
Neg Pred Value          1.0000   0.9996   0.8351       NA       NA       NA       NA
Prevalence              0.1927   0.5181   0.2892  0.00000  0.00000   0.0000 0.000000
Detection Rate          0.1927   0.5179   0.1488  0.00000  0.00000   0.0000 0.000000
Detection Prevalence    0.1927   0.5179   0.1490  0.06037  0.03551   0.0357 0.008785
Balanced Accuracy       1.0000   0.9998   0.7571       NA       NA       NA       NA
```

With tuned hyperparameters, the minimum number of items in the parent node that could be split further is set at 5, the *maxdepth* parameter prevents the tree from growing past a depth of 2, and the *minbucket* of 3 provides the smallest number of items that are allowed in a terminal node. Using the Gini index method, the attribute *statustropical storm* at a value of 0.5 is selected as the first split point with the maximized reduction in impurity.

The leaf node 2 indicates there are 52% of storms with *statustropical storm* equal to or greater than 0.5 are classified as *Class1*. On the other hand, the resulting internal node 3 further selected *statustropical depression* at a value of 0.5 as the next split point. The results in two leaf nodes labeled as 6 and 7. The leaf node 6 indicates that storms with *statustropical* equal or greater than the threshold of 0.5 and *statustropical storm* smaller than 0.5 would be classified as *Class*1, whereas the other leaf node suggests there is a 51% likelihood that whose *statustropical depression* and *statustropical storm* are smaller than the threshold of 0.5 would be classified as *Class1*. Worth noting, the skips in indexes 4 and 5 might result from the hyperparameter tuning, that hides the pruned branches in the final tree.



II. Model Quality

The final model achieves an accuracy rate of approximately 85.94%, with a Kappa value of 0.78, indicating the classifier is stronger than expected by chance. The no information rate for both indicates the accuracy result is higher than NIR at the 0.05 significance level. Worth noting, the accuracy rate of the classifier on the testing set is similar to that on the train set; hence, there might not be any overfitting presented in the dataset at first galance. However, regarding to the large number of predictors, the hypothesis space might have many dimensions, so that the model ends up with meaningless regularity that is irrelevant to the true, important, distinguishing features for Class 2~5. Lastly, since there are too little training data for Class 2~5, the hypothesis space tends to overfit.

```
> ##evaluate the fit with train set
> pred_treeStorm_train <- predict(trainDT, train_storm)
> confusionMatrix(train_storm$category, pred_treeStorm_train)
Confusion Matrix and Statistics

          Reference
Prediction   -1    0    1    2    3    4    5
        -1  722    0    0    0    0    0    0
        0     0 1940    0    0    0    0    0
        1     0    0  558    0    0    0    0
        2     0    0  227    0    0    0    0
        3     0    0  133    0    0    0    0
        4     0    0  134    0    0    0    0
        5     0    0   33    0    0    0    0

Overall Statistics

               Accuracy : 0.8594
                 95% CI : (0.8478, 0.8703)
    No Information Rate : 0.5177
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7842

 Mcnemar's Test P-Value : NA
```

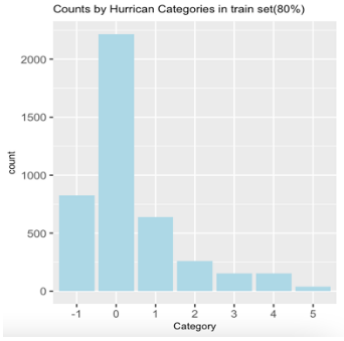
```
> ## evaluate the fit with test set
> pred_treeStorm <- predict(trainDT, test_storm)
> confusionMatrix(test_storm$category, pred_treeStorm)
Confusion Matrix and Statistics

          Reference
Prediction  -1   0   1   2   3   4   5
        -1 309   0   0   0   0   0   0
        0    0 831   0   0   0   0   0
        1    0   1 238   0   0   0   0
        2    0   0  96   0   0   0   0
        3    0   0  57   0   0   0   0
        4    0   0  57   0   0   0   0
        5    0   0  14   0   0   0   0

Overall Statistics

               Accuracy : 0.8596
                 95% CI : (0.8417, 0.8763)
    No Information Rate : 0.519
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7843

 Mcnemar's Test P-Value : NA
```

Problem #3 Decision Tree Model Comparison

I. Parallel Hyperparameter Tuning

The goal of the configuration is to tackle complex hyperparameter optimization tasks with higher dimensions, and gain intuition about the tradeoffs of model quality and model complexity for the imbalanced dataset. The algorithm below firstly allocate a grid of 3 hyperparameters – minbucket (5,20,35), minsplit (20,60,100), and maxdepth(3,5,7,10), for 10-fold stratified cross-validated decision tree models, which results in a total of 36 configurations. For the most complex model, the minimum number of items in the parent node that could be split further is set at 20, with a depth of 10, and the smallest number of items that are allowed in a terminal node is 5. On the other hand, the simplest model has a depth of 3, a minsplit of 100, and a minbucket of 35. A glimpse of the grid and class distribution plot are shown as below.

	maxdepth	minsplit	minbucket
1	3	20	5
2	5	20	5
3	7	20	5
4	10	20	5
5	3	60	5



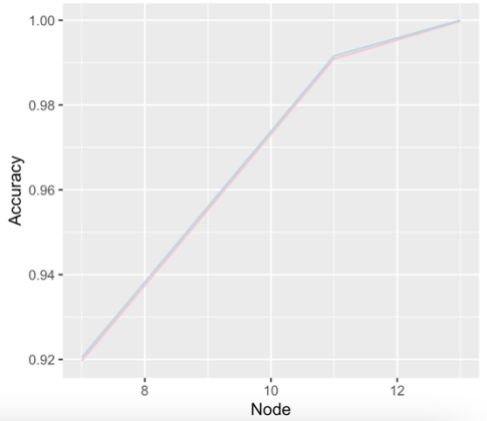
The foreach function returns the number of nodes, and the accuracies for both training and test set as a data frame, whereas each row corresponding to a combination of defined hyperparameter per iteration. Finally, the data frame *comp* merges the *results* and *hyperparms* data frame

together. The simplest decision tree that correctly classifies all the classes in the imbalanced dataset, especially for *Class2~5* is selected as the most optimal model.

```
##initialize stratified cross validation
idCategory <- createFolds(train_storm$category, k=10, returnTrain = TRUE)
train_control<- trainControl(index=idCategory, method = "cv", number = 10)

##create hyperparameter matrix
maxdepth<- c(3,5,7,10)
minsplit<- c(20,60,100)
minbucket<- c(5,20,35)
hyperparms= expand.grid(maxdepth=maxdepth, minsplit=minsplit,minbucket=minbucket)
#loop through parms values
library(doParallel)
results=foreach(i=1:nrow(hyperparms),.combine=rbind)%dopar%{
  d=hyperparms[i,]$maxdepth
  s=hyperparms[i,]$minsplit
  b=hyperparms[i,]$minbucket
  fit= train(category ~., data=train_storm, method="rpart1SE",
             control=rpart.control(minsplit = s, maxdepth = d, minbucket = b),
             trControl=train_control)
  pred_train<- predict(fit, train_storm)
  accuracy_train<- (confusionMatrix(train_storm$category, pred_train))$overall[1]
  pred=predict(fit, test_storm)
  accuracy_test<- (confusionMatrix(test_storm$category, pred))$overall[1]
  node<- nrow(fit$finalModel$frame)
  data.frame(Node=node, AccuracyTrain=accuracy_train,
             AccuracyTestn=accuracy_test)
}
hyperparms[which.min(results$AccuracyTrain),]
comp<-cbind(hyperparms,results)
```

II. Model Evaluating

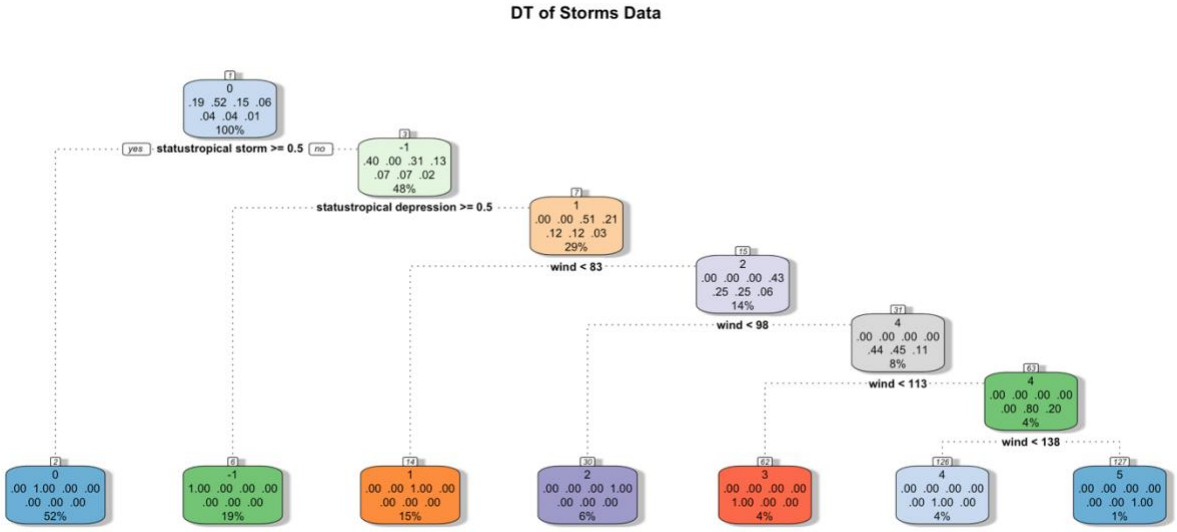
To find the most optimal hyperparameter, the inflection point locates at node valued of 11, where the tree tends to get extremely complex. At the inflection point, the maxdepth is set at 5 while other parameters vary. However, based on the confusion matrix, although the model achieves an accuracy rate of approximately 99.16%, with a Kappa value of 0.98, indicating the classifier is stronger than expected; the model ends up with meaningless regularity that is irrelevant to the true, important, distinguishing features for Class 5.




```
> fit2= train(category ~., data=train_storm, method="rpart1SE",
+             control=rpart.control(minsplit = 20, maxdepth = 5, minbucket = 5),
+             trControl=train_control,metric="Accuracy")
> predFinal2=predict(fit2, test_storm)
> accuracy_test2<- confusionMatrix(test_storm$category, predFinal2)
> accuracy_test2
Confusion Matrix and Statistics

          Reference
Prediction  -1   0   1   2   3   4   5
        -1 206   0   0   0   0   0   0
        0    0 554   0   0   0   0   0
        1    0   0 159   0   0   0   0
        2    0   0   0  64   0   0   0
        3    0   0   0   0  38   0   0
        4    0   0   0   0   0  38   0
        5    0   0   0   0   0   9   0
```

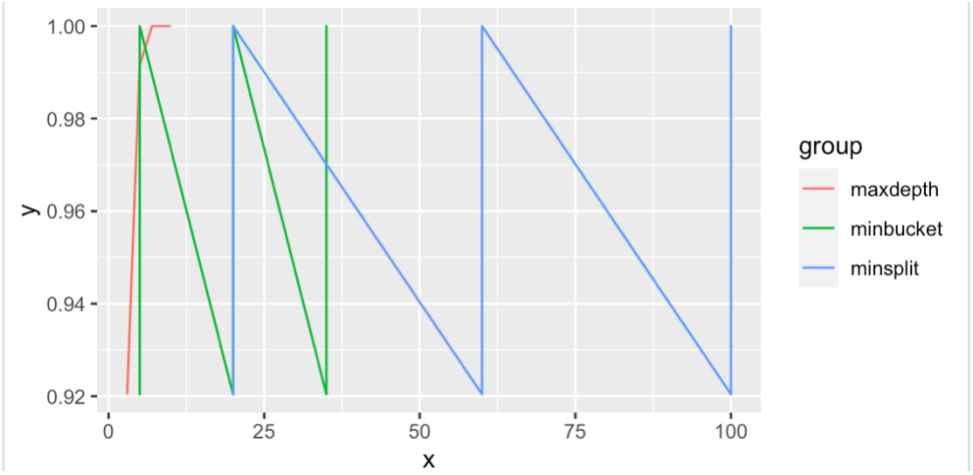
Hence, by checking the model with the largest accuracy rate on test set, the parameter set returned as in below. By increasing the maxdepth parameter by 1, the below model successfully captures meaningful regularity that is relevant to important distinguish features for all classes.

```
> hyperparms[which.max(results$AccuracyTestn),]
  maxdepth minsplit minbucket
3        7       20         5

> fit= train(category ~., data=train_storm, method="rpart1SE",
+           control=rpart.control(minsplit = 20, maxdepth = 7, minbucket = 5),
+           trControl=train_control,metric="Accuracy")
> predFinal=predict(fit, test_storm)
> accuracy_test<- confusionMatrix(test_storm$category, predFinal)
> accuracy_test
Confusion Matrix and Statistics

          Reference
Prediction  -1   0   1   2   3   4   5
        -1 206   0   0   0   0   0   0
        0    0 554   0   0   0   0   0
        1    0   0 159   0   0   0   0
        2    0   0   0  64   0   0   0
        3    0   0   0   0  38   0   0
        4    0   0   0   0   0  38   0
        5    0   0   0   0   0   0   9

Overall Statistics

               Accuracy : 1
                 95% CI : (0.9966, 1)
    No Information Rate : 0.5187
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1
```



III. Parameter Evaluating

As the previous section explore different manually selected hyperparameters, the plot below suggests that maxdepth parameter affects the model quality the most heavily, compared to the other two parameters. Worth noting, increasing the thresholds of minbucket and minsplit tend to cause overfitting as instability in model accuracy has shown below.



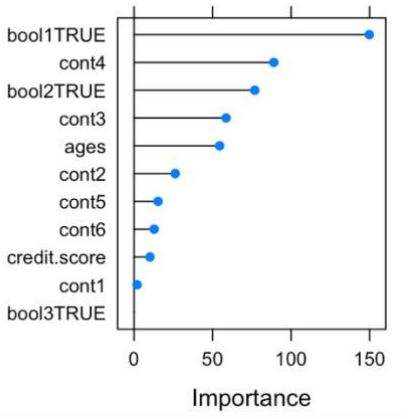
Problem #4 Feature Selection

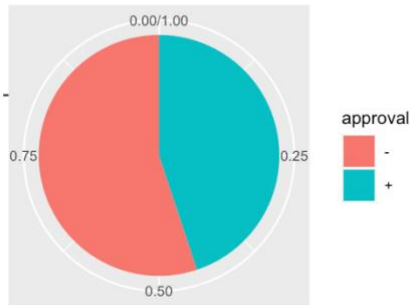
I. Relevance Analysis

The following section evaluates the significance of each predictor, in an attempt to reduce model complexity. After removing all incomplete cases and converting the class label into factor in *Bank_Modified* dataset; the cleaned dataset contains 666 observations. The set of tuples is associated with class labels, *approval*, with each taking one of two values, corresponding to the classes including loan approval and loan denial, respectively.

Since the approval rate for loans is approximately 44.5%, the random sampling would be prone toward imbalanced samples. Hence, stratifying the sampling is applied. Meantime, as many of the predictor attributes are dominated by outliers that heavily skewed the data distribution; hence, data points are centered and scaled during the cross-validation.

```
> ##Fit the model
> initFit<- train(approval ~., data=trainBank, method="rpart1SE",
+                 control=hypers, trControl=train_control,preProcess=c("center","scale"))
> var_imp <- varImp(initFit,scale = FALSE)
> var_imp
rpart1SE variable importance

               Overall
bool1TRUE      149.779
cont4           88.997
bool2TRUE       76.858
cont3           58.622
ages            54.510
cont2           26.212
cont5           15.293
cont6           12.775
credit.score    10.127
cont1            1.907
bool3TRUE        0.000
```





Based on the output, variables including *bool1, cont4,and bool2,* produce a high level of generalization, whereas their reductions are the most significant when each predictor is added to the model. On the other hand, *bool3 and, cont1* provide little information to the tree model.

II. Model Tuning

After rebuilding the tree model with the top 6 most relevant variables, the *tunedFit* model performs worse than the *initFit* model on the test set, although the returned model accuracy is slightly higher than the initial model. The reason behind this might be the lower estimation variance in the initial model, hence better predictive performance on unseen data.

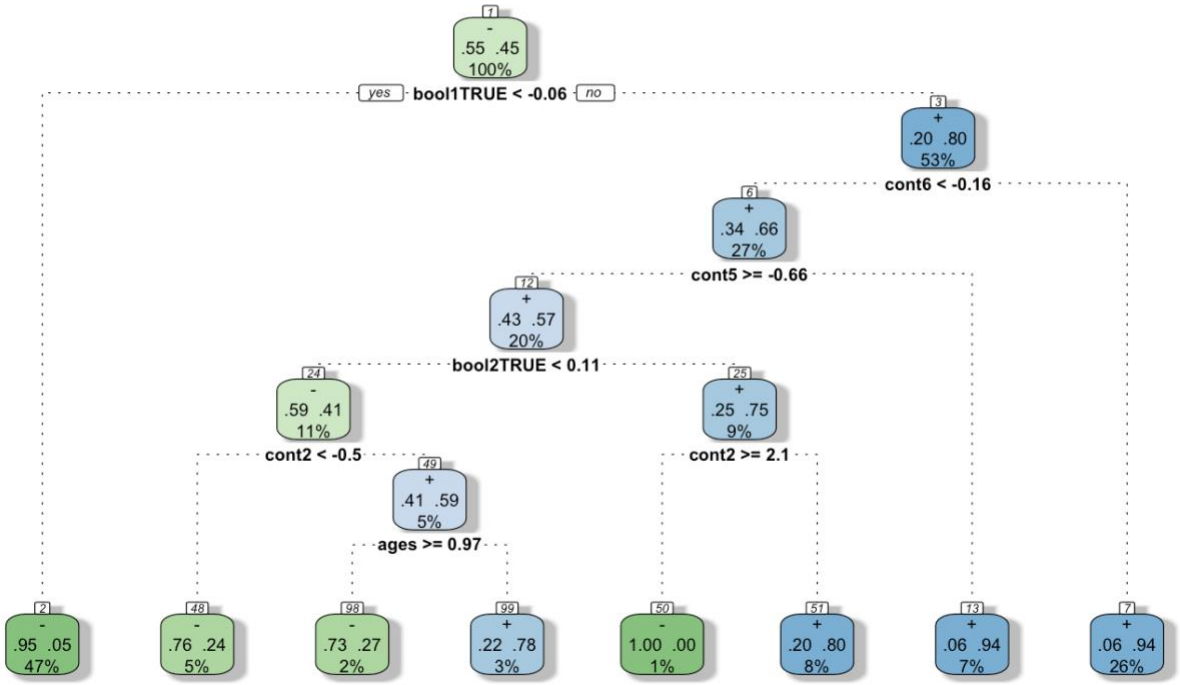
```
> ##Fit the model
> initFit<- train(approval ~., data=trainBank, method="rpart1SE",
+                 control=hypers, trControl=train_control,preProc
ess=c("center","scale"))
> initFit
CART

534 samples
 11 predictor
  2 classes: '-', '+'

Pre-processing: centered (11), scaled (11)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 481, 481, 481, 480, 480, 480, ...
Resampling results:

  Accuracy   Kappa
  0.8725017  0.7433342

> confusionMatrix(testBank$approval, predinit)
Confusion Matrix and Statistics

          Reference
Prediction  -  +
         - 62 11
         + 10 49

               Accuracy : 0.8409
                 95% CI : (0.7672, 0.8987)
    No Information Rate : 0.5455
    P-Value [Acc > NIR] : 6.195e-13

                  Kappa : 0.6787

 Mcnemar's Test P-Value : 1
```

```
> tunedFit<- train(approval ~., data=tunedBank, method="rpart1SE",
+                  control=hypers, trControl=train_controlTuned,prePr
ocess=c("center","scale"))
> tunedFit
CART

534 samples
  6 predictor
  2 classes: '-', '+'

Pre-processing: centered (6), scaled (6)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 481, 481, 480, 481, 481, 480, ...
Resampling results:

  Accuracy   Kappa
  0.8766247  0.7516569

> confusionMatrix(tunedTest$approval, predT)
Confusion Matrix and Statistics

          Reference
Prediction  -  +
         - 59 14
         +  9 50

               Accuracy : 0.8258
                 95% CI : (0.7501, 0.8862)
    No Information Rate : 0.5152
    P-Value [Acc > NIR] : 8.989e-14

                  Kappa : 0.6504

 Mcnemar's Test P-Value : 0.4042
```

III. Tree Visualization

Based on the visualizations, the tuned tree is much simplified with fewer variables. While the model quality is not much worsened, the irrelevant variables introduce mostly the noises to the initial model. Looking at the *DT of Initial Bank Data*, the tree is extremely complex to interpret, as there are only two classes to classify. Hence, feature selection or dimensionality reduction is necessary for tree pruning.

DT of Initial Bank Data



DT of Tuned Bank Data