ASSIGNMENT 3

ENGIN 242 - Applications in Data Analytics

Aditya Peshin
Enrollment Number: 3035280249

Q1. CART Models and Impurity Function

a) (10 points) Let $\Delta = C_{imp}(T_{old}) - C_{imp}(T_{new})$ be the absolute decrease in total impurity resulting from the split. Derive a formula for that can be computed locally at the bucket M , in other words it should only depend on the data points that fall in region R_M in the original tree T_{old}.
(Hint: we've discussed this concept in class, this question is asking for a more formal argument. You may assume that the two new buckets in T new resulting from the split are labeled as buckets M and M + 1 in T new.)

b) (10 points) Show that $\Delta >= 0$, hence splitting always reduces the total impurity cost. (Hint: you can use the fact that, given a sequence of real numbers $z_1, z_2, \ldots, z_n$ the mean $z = 1/n \Sigma z_i$ is the minimizer of the function $RSS(z) = \Sigma (z_i - z)^2$

c) (10 points) Let R^2_{old} be the training set R^2 value for the model denoted by T_{old}, and likewise let R^2_{new} be the training set R^2 value for the model denoted by T_{new}.
Let $SST = \Sigma (yi - y)^2$, $i \in \{1..N\}$ be the total sum of squared errors, where $y = 1/n \Sigma yi$ is the overall mean. For a given value of the complexity parameter (cp) >= 0, recall the modified cost function that is relevant in the pruning step:

$$C_\alpha(T) = C_{imp}(T) + \alpha * SST * |T|$$

Show that $C_\alpha(T_{new})$ $C_\alpha(T_{old})$ if and only if $R^2_{new} - R^2_{old} >= \alpha$. (Hence the choice of retaining a split if the increase in R^2 is at least is equivalent to retaining a split if the modified cost function is smaller after the split.)

Ans.

Part(a)

We know,

$$Q_m = (1/N_m) * \Sigma (y_i - \hat{y}_m)^2 \quad \text{where } i : x_i \in R_m$$

Also,

$$C = \Sigma N_m * Q_m \qquad \text{where } m \in \{ 1 \ldots M \}$$
$$= \Sigma N_m * (1/N_m) * \Sigma (y_i - \hat{y}_m)^2$$
$$= \Sigma\Sigma (y_i - \hat{y}_m)^2 \qquad \text{where } i : x_i \in R_m , m \in \{ 1 \ldots M \}$$

Assuming the split occurs at the M^{th} node, giving nodes ‘Ma’ and ‘Mb’, we have:
[Not using M, M+1 notation as it is confusing]

$$C_{imp}(T_{old}) = \sum_{i:\, xi \in R_1} (y_i - \hat{y}_1)^2 + \sum_{i:\, xi \in R_2} (y_i - \hat{y}_2)^2 + \ldots + \sum_{i:\, xi \in R_M} (y_i - \hat{y}_M)^2$$

$$C_{imp}(T_{new}) = \sum_{i:\, xi \in R_1} (y_i - \hat{y}_1)^2 + \sum_{i:\, xi \in R_2} (y_i - \hat{y}_2)^2 + \ldots + \sum_{i:\, xi \in R_{Ma}} (y_i - \hat{y}_{Ma})^2 + \sum_{i:\, xi \in R_{Mb}} (y_i - \hat{y}_{Mb})^2$$

Thus, the difference of the impurity costs of the new tree from the old becomes:

$$\Delta = C_{imp}(T_{old}) - C_{imp}(T_{new})$$

$$= \sum_{i:\, xi \in R_1} (y_i - \hat{y}_1)^2 + \sum_{i:\, xi \in R_2} (y_i - \hat{y}_2)^2 + \ldots + \sum_{i:\, xi \in R_M} (y_i - \hat{y}_M)^2$$

$$- \{ \sum_{i:\, xi \in R_1} (y_i - \hat{y}_1)^2 + \sum_{i:\, xi \in R_2} (y_i - \hat{y}_2)^2 + \ldots + \sum_{i:\, xi \in R_{Ma}} (y_i - \hat{y}_{Ma})^2 + \sum_{i:\, xi \in R_{Mb}} (y_i - \hat{y}_{Mb})^2 \}$$

$$= \sum_{i:\, xi \in R_M} (y_i - \hat{y}_M)^2 - \sum_{i:\, xi \in R_{Ma}} (y_i - \hat{y}_{Ma})^2 - \sum_{i:\, xi \in R_{Mb}} (y_i - \hat{y}_{Mb})^2$$

Since the other buckets are identical, they have the same impurity cost, and those terms get cancelled. The only terms that remain are the impurity cost of the unsplit bucket, and the impurity costs of the two new buckets formed. Thus, we are able to compute the value of Δ locally within the bucket that we are considering the split on.

Part - b

From part 'a', we have

$$\Delta = \sum_{i:\, xi \in R_M} (y_i - \hat{y}_M)^2 - \sum_{i:\, xi \in R_{Ma}} (y_i - \hat{y}_{Ma})^2 - \sum_{i:\, xi \in R_{Mb}} (y_i - \hat{y}_{Mb})^2$$

Splitting the original impurity cost according to the new buckets formed, and rearranging, we get

$$\Delta = \underbrace{\sum_{i:\, xi \in R_{Ma}} (y_i - \hat{y}_M)^2 - (y_i - \hat{y}_{Ma})^2}_{\text{----------Term 1----------}} + \underbrace{\sum_{i:\, xi \in R_{Mb}} (y_i - \hat{y}_M)^2 - (y_i - \hat{y}_{Mb})^2}_{\text{----------Term 2----------}}$$

Looking at the terms individually, we can see:

Term 1 - For the new bucket, since $\hat{y}_{Ma}$ is the minimizer for y in that bucket, we have the term $(y_i - \hat{y}_{Ma})^2$ which will always have a value less than or equal to $(y_i - \hat{y}_M)^2$. Thus, that summation term will always be non-negative.

Term 2 - For the new bucket, since $\hat{y}_{Mb}$ is the minimizer for y in that bucket, we have the term $(y_i - \hat{y}_{Mb})^2$ which will always have a value less than or equal to $(y_i - \hat{y}_M)^2$. Thus, that summation term will also always be non-negative.

Since both terms individually are non-negative, we can say that the summation of both terms is non-negative. Hence, we can say that $\Delta >= 0$.

Part - c

We have the impurity function of the tree for a given complexity parameter as,

$$C_\alpha(T) = C_{imp}(T) + \alpha * SST * |T| \qquad (1)$$

From the **Part - A**, we know that C_{imp} for this problem is the RSS error, because

$C_{imp} = \Sigma\, N_m * Q_m$ where $m \in \{ 1 \ldots M \}$

$= \Sigma N_m * (1/N_m) * \Sigma (y_i - \hat{y}_m)^2$
$= \Sigma\Sigma (y_i - \hat{y}_m)^2$
$= RSS$

Thus, the expression for $C_\alpha(T)$ becomes:

$$C_\alpha(T) = RSS + \alpha * SST * |T| \quad (2)$$

Dividing this expression by SST, we get:

$$C_\alpha(T) / SST = RSS / SST + \alpha * |T| \quad (3)$$

Subtracting 1 on both sides from this expression, we have:

$$(C_\alpha(T) / SST) - 1 = [(RSS / SST) - 1] + \alpha * |T| \quad (4)$$

----------↓----------

This is the expression for $-1 * R^2$ for that model

So, we can write this expression for T_{old} and T_{new} as:

$$(C_\alpha(T_{old}) / SST) - 1 = -1 * R^2_{old} + \alpha * |T_{old}| \quad (5)$$
$$(C_\alpha(T_{old}) / SST) - 1 = -1 * R^2_{new} + \alpha * |T_{new}| \quad (6)$$

Subtracting equation (5) and (6), we get:

$$(C_\alpha(T_{old}) - C_\alpha(T_{new})) / SST = R^2_{new} - R^2_{old} + \alpha * [|T_{old}| - |T_{new}|] \quad (7)$$

For $C_\alpha(T_{old}) >= C_\alpha(T_{new})$, or for $C_\alpha(T_{old}) - C_\alpha(T_{new})$ to be positive,
The RHS of equation (7) must be positive,

$$R^2_{new} - R^2_{old} + \alpha * [|T_{old}| - |T_{new}|] >= 0 \quad (8)$$

Since we know that only one split happens between trees, we know that

$$[|T_{old}| - |T_{new}|] = -1 \quad (9)$$

Substituting equation (9) in equation (8), we have:

$$R^2_{new} - R^2_{old} + \alpha * (-1) >= 0 \quad (10)$$

Or,

$$R^2_{new} - R^2_{old} >= \alpha \quad (11)$$

Which is what we wanted to prove. Thus, the model keeps the split only if the difference in the R^2 values is greater than α.

Q2. Letter Recognition (Adapted from Bertsimas 22.3)

Part A.

A.i) Before building any models, first consider a baseline method that always predicts the most frequent outcome, which is “not B". What is the accuracy of this baseline method on the test set?

Ans) The baseline model code accuracy is 0.7222

A.ii) Construct a logistic regression model to predict whether or not the letter is a B, using the training set to build your model. (Remember to not use the variable letter as one of the independent variables in any of the models in this part, since it is not a valid feature.) What is the accuracy of your logistic regression model on the test set, using a threshold of p = 0:5?

Ans) The accuracy of the logistic regression model on the test set, using a threshold of p=0.5 is, accuracy (Logistic Regression) = 0.9468

A.iii) What is the AUC of your logistic regression model?

Ans) The AUC of the logistic regression model is = 0.97966

A.iv) Construct a CART tree to predict whether or not a letter is a B, using the training set to build your model. Select the cp value for the tree through cross-validation, and explain how you did the cross-validation and how you selected the cp value. What is the accuracy of your CART model on the test set?

Ans) In order to do cross validation, I first defined a dataframe of possible cp values, 0 to 0.04 with a 0.002 increment in each iteration. Using the train function to train the model and examining the results, I observed that as the value of cp increased from 0.000 to 0.004 the accuracy increased, and thereon decreased as cp increased to 0.040. Thus, I determined that 0.004 would be the optimal value for cp, to get maximum accuracy.
The accuracy of the CART model on the test set (cp at 0.004) is = 0.9248

A.v) Now construct a Random Forest model to predict whether or not the letter is a B. Just leave the Random Forest parameters at their default values (i.e., leave them out of the function call). What is the accuracy of this Random Forest model on the test set?

Ans) The accuracy of the Random Forest model on the test set is = 0.9743

A.vi) Compare the accuracy of your logistic regression, CART, and Random Forest models. Which one performs best on the test set? For this application, which do you think is more important: interpretability or accuracy?

Ans) Among the various models, the Random Forest model has the best test set accuracy, and should be preferred in this situation. In this specific application, we are interested in the ability of the program to accurately classify the input into the right class/letter, which is why accuracy is a lot more important than interpretability. Having a program that gives us a higher accuracy is of more value than understanding how the various features like xbox/ybox impact the final outcome(this carries almost zero value).

Part B

B.i) In a multi-class classification problem, a simple baseline model is to predict (for every observation) the most frequent class over all of the classes. For this problem, what does the baseline method predict, and what is the baseline accuracy on the test set?

Ans) The number of observations of the various classes in the training set are:

	A	B	P	R
Frequency	546	463	520	496

Here, the most frequent observation is 'A', so this will be the prediction of the baseline for all observations.
In the test set, the observations are in the following classes with frequency:

	A	B	P	R
Frequency	243	303	283	262

The baseline model always predicts 'A' on the test set, and the accuracy on the test set then becomes = 243 / (243 + 303 + 283 + 262) = 0.2227

B.ii) Construct an LDA model to predict letter, using the training set to build your model. (Throughout remember not to use the variable isB in the model, as it is not a valid feature.) What is the test set accuracy of your LDA model?

Ans) The test set accuracy of the LDA model is = 0.9184

B.iii) Now construct a CART model to predict letter using the training data. Again, select the cp value for the tree through cross-validation. Again, explain how you did the cross-validation and how you selected the cp value. What is the test set accuracy of your CART model?

Ans) In order to do cross validation, I first defined a dataframe of possible cp values, 0 to 0.04 with a 0.002 increment in each iteration. Using the train function to train the model and

examining the results, I observed that the value of accuracy decreased as cp increased from 0.000 to 0.040. Thus, I determined that I needed to further refine the search for cp, and then changed the upper and lower bounds to (0.000,0.002) and changed the 'by' value to 0.0001, and repeated the experiment. I found that even in this increased fine tuning of cp, the maximum value of accuracy was still at 0.0, and thus, 0 would be the optimal value of cp to get maximum accuracy.

The test set accuracy of the CART model is = 0.89092

B.iv) Next build a model for predicting letter using "vanilla" bagging of CART models. This can be achieved, for example, by setting mtry equal to the total number of features (i.e., set m = p) in the randomForest package in R. What is the test set accuracy of your bagging model?

Ans) The test set accuracy of the bagging model is = 0.9477

B.v) Now build a Random Forest model, and this time use cross-validation to select the mtry value for the Random Forests method. Explain how you did the cross validation and how you selected the mtry value. What is the test set accuracy for your Random Forest model?

Ans) In order to do the cross validation model, I defined a 'mtry' dataframe to contain the values from 1 to 16, as I wanted to vary mtry over all the integer values between 1 and p(which is 16 in this case). Upon cross validation, I found that a value of mtry = 6 led to the maximum value of random forest accuracy, and thus finalized this value of mtry for the final model.

The test set accuracy of the Random Forest model for optimal value of mtry is = 0.9651

B.vi) Train a "gradient boosting machine (gbm)" model to predict letter using the training data. Set the interaction depth to 10, run the method for 3300 iterations, and leave all other parameters at their default values. (The values for the interaction depth and the number of iterations were obtained via a 5-fold cross validation procedure.) What is the test set accuracy of your gradient boosting model?

Ans) The test set accuracy of the gradient boosting model is = 0.9798

B.vii) Compare the accuracy of your LDA, CART, bagging, Random Forest, and boosting models for this problem. Which one would you recommend for this problem? Is your choice different from the model you recommended in part (a)? Why or why not?

Ans) The most accurate model among all the models for this problem is the Gradient Boosting Machine, with a test set accuracy of 97.98%

I would recommend GBM, as it provides the highest accuracy among the models. My choice is different from part (A), as in this situation, GBM provides a marginal advantage over random forests. However, it is important to consider that this difference might just be due to the choice of seed, and Random Forests may give better performance on a different seed / test set.

Code:

```
# Assignment 3 Question 2

library(rpart)
library(rpart.plot)
library(caret)
library(randomForest)
library(gbm)
library(caTools)
library(dplyr)
library(ggplot2)
library(GGally)
library(ROCR)
library(e1071)
library(tidyverse)
library(MASS)

# Part A
letters_orig <- read.csv("Letters.csv")
letters_orig$isB = as.factor(letters_orig$letter == "B")

#setting seed
set.seed(456)

#splitting into training and testing set
train.ids = sample(nrow(letters_orig), 0.65*nrow(letters_orig))
letters_orig.train = letters_orig[train.ids,]
letters_orig.test = letters_orig[-train.ids,]

# Part A.i [ Baseline Model ]
set.seed(456)
table (letters_orig.test$isB)
   # False True
   # 788   303
baseline_acc <- (788) / nrow(letters_orig.test)
baseline_acc
   # This value is 0.7222
```

```
# Part A.ii [ Logistic Regression ]
set.seed(456)
mod1 <- glm(isB ~ . - letter, data = letters_orig.train, family = "binomial" )
summary(mod1)
   # making the prediction on the test data
pred1 <- predict(mod1, newdata = letters_orig.test, type = "response")
table(letters_orig.test$isB, pred1>= 0.5)
   #       FALSE TRUE
   # FALSE   760   28
   # TRUE     30  273
logistic_acc <- (273 + 760)/(273+760+30+28)
logistic_acc
   # this accuracy is 0.9468

# Part A.iii  [ Area under the Curve ]
set.seed(456)
pred_logistic <- prediction(pred1, letters_orig.test$isB)
auc_logistic <- performance(pred_logistic, measure = "auc")
auc_logistic <- auc_logistic@y.values[[1]]
   # The value of AUC is 0.97966

# Part A.iv   [ CART Tree to predict letter B]
set.seed(456)
cpVals = data.frame(cp = seq(0, .04, by=.002))
model_cart <- train(isB ~ . - letter,
                    data = letters_orig.train,
                    method = "rpart",
                    tuneGrid = cpVals,
                    trControl = trainControl(method = "cv", number=10),
                    metric = "Accuracy" )
model_cart$results
model_cart
   # for cp = 0.004, we get maximum accuracy of 0.9323 on the training set

# Now, we predict the test set values and calculate test set accuracy
set.seed(456)
pred2 <- predict(model_cart$finalModel, newdata = letters_orig.test, type = "prob")
table(letters_orig.test$isB, pred2[,2] >= 0.5)
   #         FALSE TRUE
   # FALSE     768   20
   # TRUE       62  241
cart_acc <- (241 + 768)/(241+20+62+768)
cart_acc
```

```
   # this accuracy is 0.924839

# Part A.v  [ Random Forest to predict letter B]
   # Making the model
set.seed(456)
model_randomforest <- randomForest(as.factor(isB) ~.-letter,
                                   data = letters_orig.train)
   # Predicting the outcomes
pred_rf <- predict(model_randomforest, newdata = letters_orig.test)
   # Confusion Matrix
table(pred_rf, letters_orig.test$isB)
 # pred_rf   FALSE TRUE
 #    FALSE   781   21
 #    TRUE      7  282

randomforest_acc <- (781 + 282)/ (781+281+7+22)
randomforest_acc
   # This value is 0.9743355

# Part B

# Part B.i    [ Baseline Model ]
set.seed(456)
table(letters_orig.train$letter)
     #  A   B   P   R
     # 546 463 520 496
  # Thus, the prediction for every observation is 'A'
table(letters_orig.test$letter)
     #    A   B   P   R
     #  243 303 283 262
baseline_acc_part_b <- 243 / (243 + 303 + 283 + 262)
baseline_acc_part_b
     # The value of accuracy is 0.2227 for the baseline model

# Part B.ii   [ Linear Discriminant Analysis ]
set.seed(456)
model_lda <- lda(letter~.-isB,
                 data = letters_orig.train)
pred_lda <- predict(model_lda, newdata = letters_orig.test)
pred_lda_class <- pred_lda$class
pred_lda_probs <- pred_lda$posterior

tab <- table(letters_orig.test$letter, pred_lda_class)
```

```
tab
#         pred_lda_class
#           A     B     P     R
#     A   227     5     1    10
#     B     0   272     0    31
#     P     0     6   273     4
#     R     0    31     1   230
lda_accuracy_part_b <- sum(diag(tab))/ sum(tab)
lda_accuracy_part_b
   # The value of model accuracy for this case of LDA is 0.9184


# Part B.iii    [ CART Model]
set.seed(456)
cpVals_part_b = data.frame(cp = seq(0, .002, by=.0001))
model_cart_part_b <- train(letter ~ . - isB,
                           data = letters_orig.train,
                           method = "rpart",
                           tuneGrid = cpVals_part_b,
                           trControl = trainControl(method = "cv", number=10),
                           metric = "Accuracy" )
model_cart_part_b$results
model_cart_part_b

# Now, for this model, we predict the test set values and calculate test set accuracy
set.seed(456)
pred_cart_part_b <- predict(model_cart_part_b$finalModel, newdata = letters_orig.test, type =
"prob")
pred_cart_part_b
pred_cart_part_b.extended <- mutate(as.data.frame(pred_cart_part_b), Pred = ifelse(A>=0.5,
'A', ifelse(B>=0.5, "B", ifelse(P>=0.5, "P" , "R"))))

tab_cart_part_b <- table(letters_orig.test$letter, pred_cart_part_b.extended$Pred)
tab_cart_part_b
   #         A     B     P     R
   #   A   230     5     0     8
   #   B    10   253     6    34
   #   P     2    13   265     3
   #   R    10    27     1   224

cart_acc_part_b <- (sum(diag(tab_cart_part_b)))/ sum(tab_cart_part_b)
cart_acc_part_b
   # The accuracy for this model is 0.89092
```

```
# Part B.iv     [ Bagging of CART Models]
set.seed(456)
ncol(letters_orig.test)
   # The number of columns are 18
model_bagging <- randomForest(letter ~ . - isB,
                         data = letters_orig.train,
                         mtry = 16 )
model_bagging
   # Predicting the outcomes
set.seed(456)
pred_bagging <- predict(model_bagging, newdata = letters_orig.test)
pred_bagging
   # Confusion Matrix
bagging_tab <- table(pred_bagging, letters_orig.test$letter)
bagging_tab
#  pred_bagging
#       A   B   P   R
#    A 236   2   0   1
#    B   4 280   2  19
#    P   1   2 278   2
#    R   2  19   3 240

bagging_acc <- sum(diag(bagging_tab))/sum(bagging_tab)
bagging_acc
   # The value for bagging accuracy is 0.9477

# Part B.v    [ Random forest with cross validation ]
set.seed(456)
mtry_part_b = data.frame(mtry = seq(1, 16, by=1))
model_randomforests_part_b <- train(letter ~ . - isB,
                               data = letters_orig.train,
                               method = "rf",
                               tuneGrid = mtry_part_b,
                               trControl = trainControl(method = "cv", number=10),
                               metric = "Accuracy" )

model_randomforests_part_b$results
# This table shows the best value accuracy at mtry = 6

final_rf_part_b <- model_randomforests_part_b$finalModel
final_rf_part_b
```

```
# Making the predictions on the test set
set.seed(456)
pred_rf_part_b <- predict(final_rf_part_b, letters_orig.test)
pred_rf_part_b

tab_rf_part_b <- table(pred_rf_part_b, letters_orig.test$letter)
tab_rf_part_b

#  pred_rf_part_b   A    B    P    R
#           A    238    1    0    0
#           B     2   286    2   10
#           P     1    0   279    2
#           R     2   16    2   250

#Now, calculating the accuracy of the model, we get:
rf_acc_part_b <- sum(diag(tab_rf_part_b))/sum(tab_rf_part_b)
rf_acc_part_b
# The value of accuracy for the best model using cross validation
# on Random Forests is 0.9651

# Part B.vi   [ Boosting ]
set.seed(456)
model_boosting <- gbm(letter ~ . -isB,
                      data = letters_orig.train,
                      distribution = "multinomial",
                      n.trees = 3300,
                      interaction.depth = 10)

# Making the Predictions on the boosting model
set.seed(456)
pred_boost <- predict(model_boosting, newdata = letters_orig.test, n.trees = 3300, type =
"response")
pred_boost
# Converting the posterior probabilities into a prediction
final_pred_part_b = apply(pred_boost, 1, which.max)
final_pred_part_b = factor(final_pred_part_b, levels = c(1,2,3,4), labels = c("A", "B", "P", "R"))

# The Confusion Matrix for Boosting
tab_boosting_part_b <- table(final_pred_part_b, letters_orig.test$letter)
tab_boosting_part_b

#  final_pred_part_b    A    B    P    R
#                A  240    0    0    0
```

```
#             B   2  295    1    5
#             P   0    0  277    0
#             R   1    8    5  257
```

```
# Calculating the accuracy of the model, we get
boosting_acc <- sum(diag(tab_boosting_part_b))/sum(tab_boosting_part_b)
boosting_acc
# The value of prediction accuracy in this case is 0.979835
```