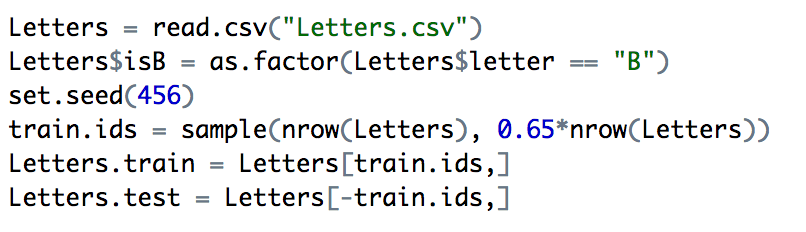
Nicolas Kardous

IND 242 HW3

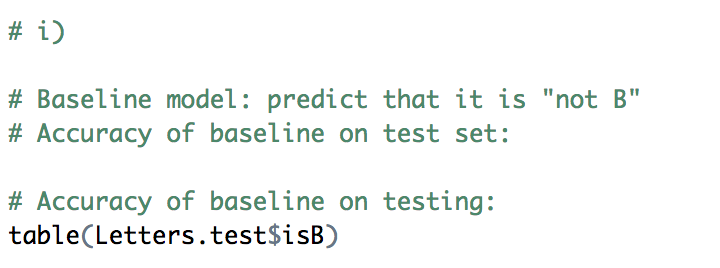
Paul Grigas

**Problem 2**

**Part a)**



**i)**



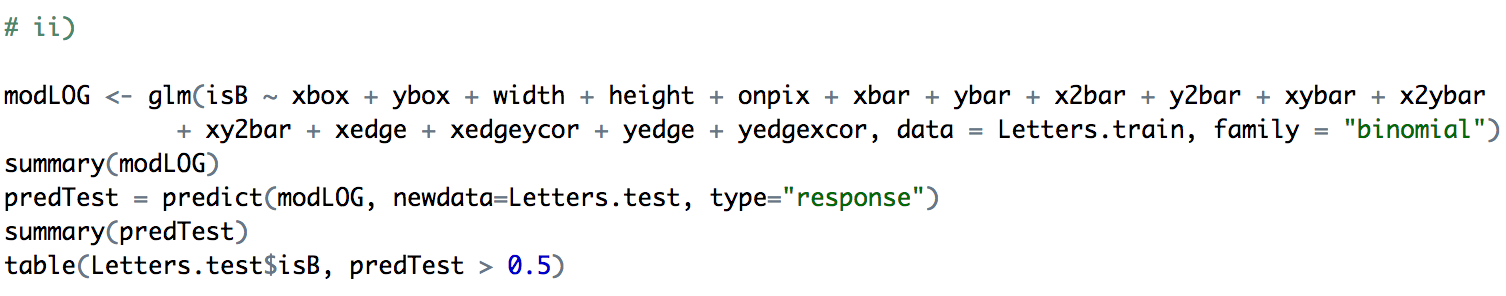
FALSE TRUE

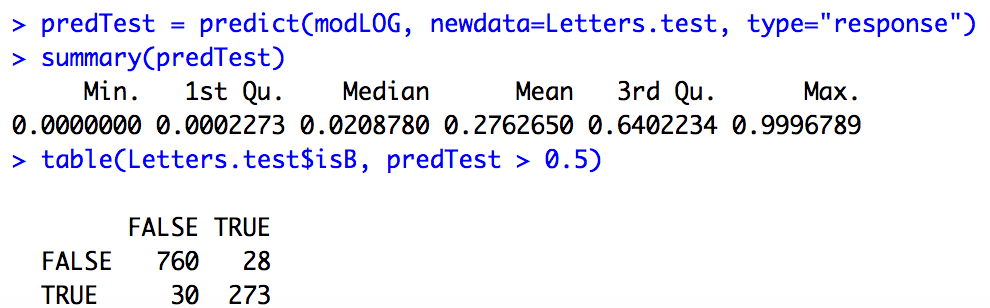
788 303

Model Accuracy: Number correct / Number in total = 788 / (788 + 303) = 0.7223

The accuracy is 72.23%

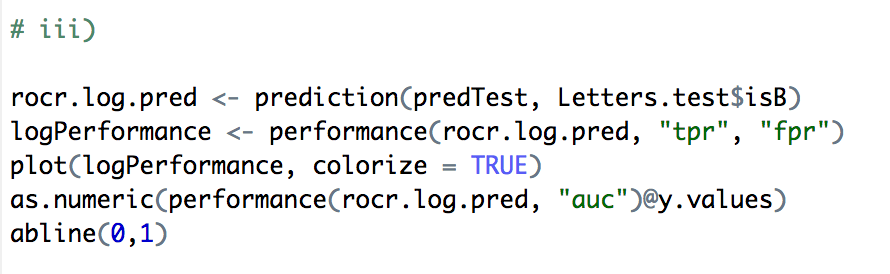
**ii)**

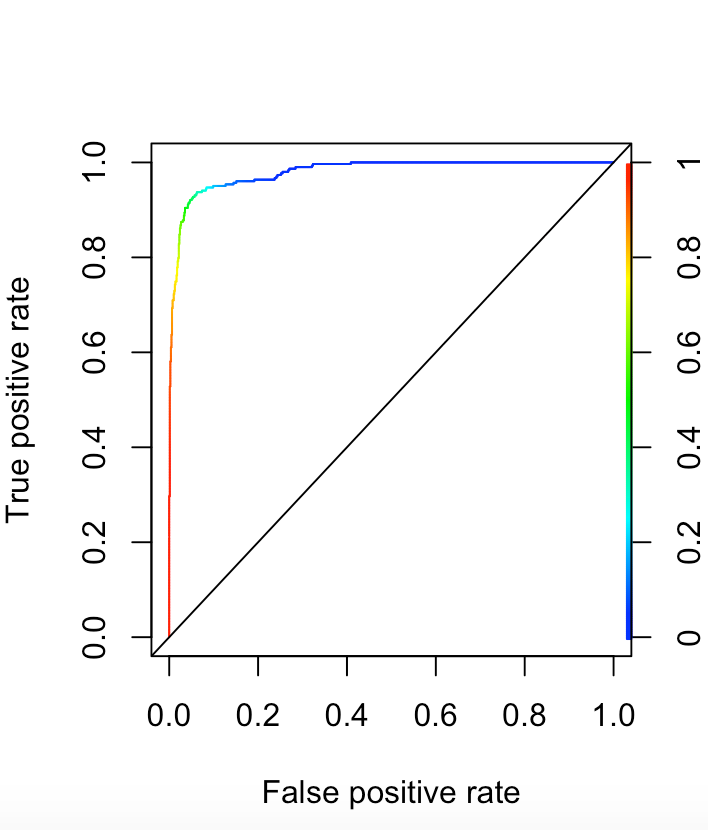




Using a threshold value of p = 0.5, we find that the accuracy is (760 + 273) / (760 + 28 + 30 + 273). Thus, the accuracy = 0.9468 = 94.68%

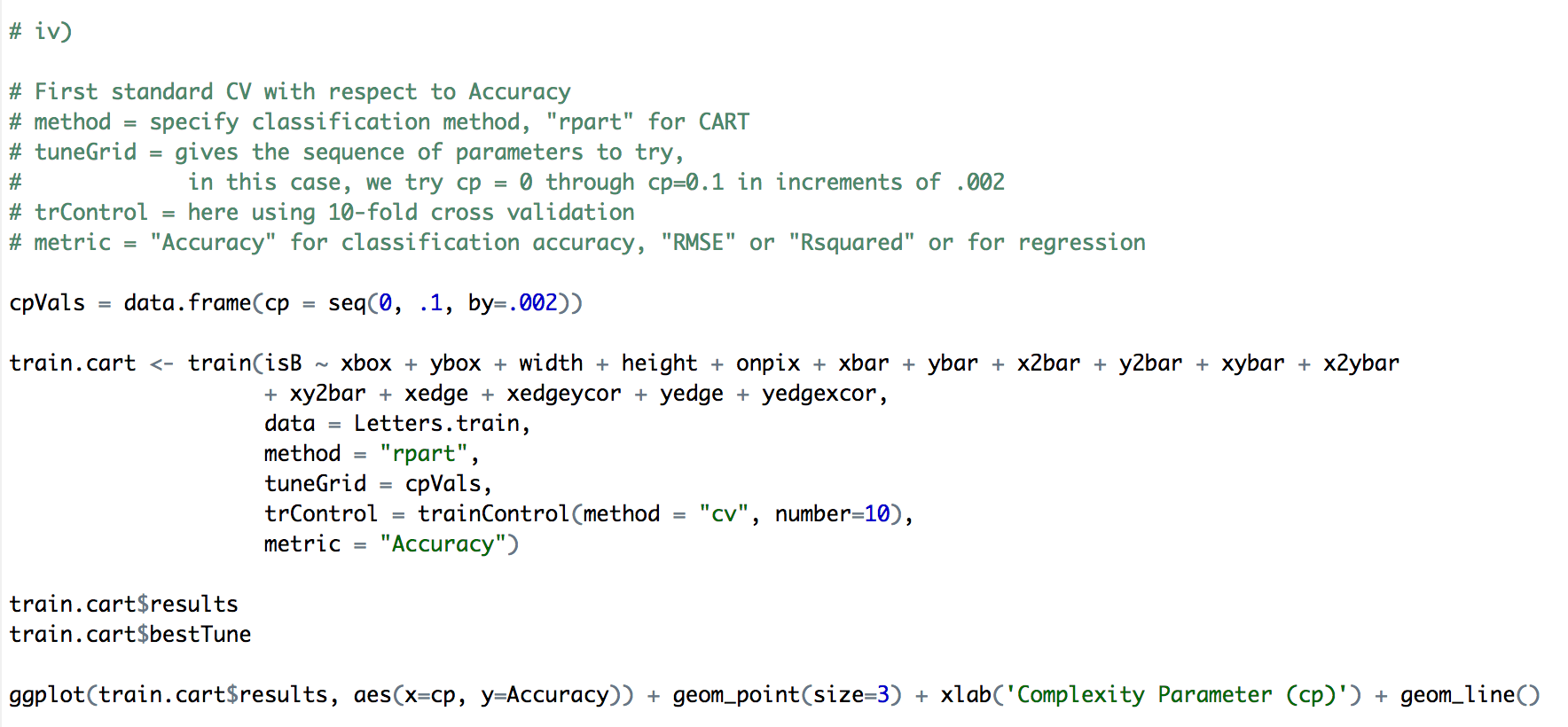
**iii)**

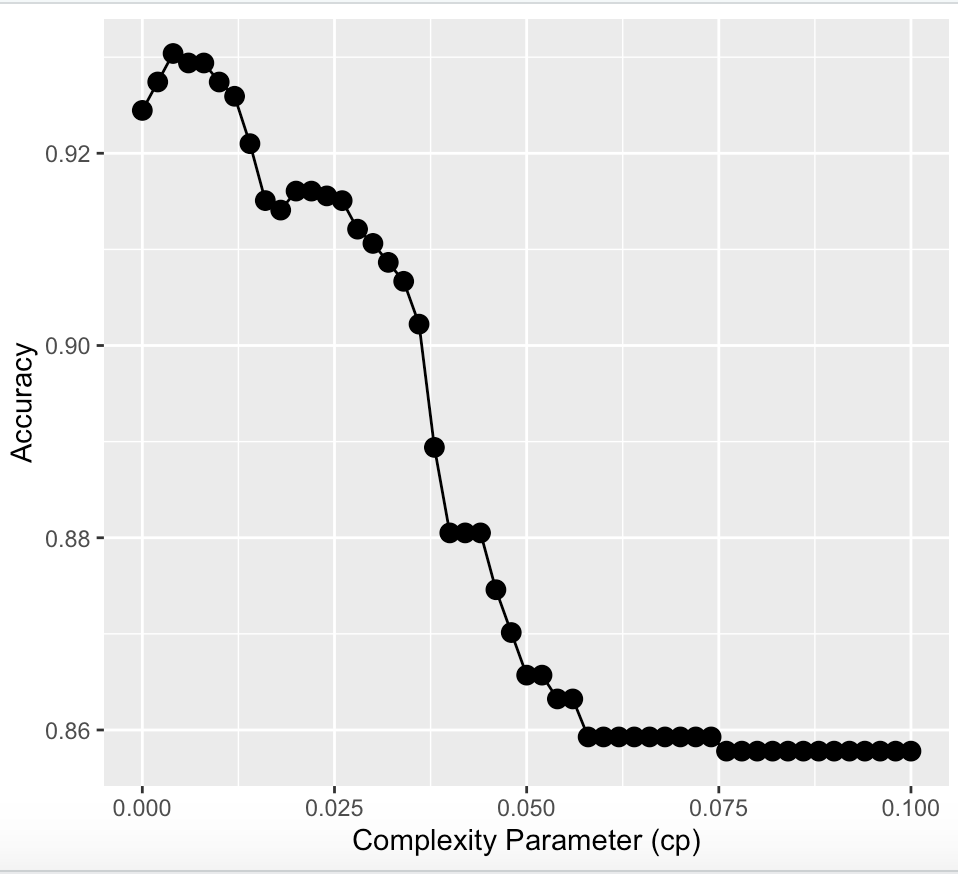




The AUC value we found is 0.97967, thus the model is very accurate.

**iv)**

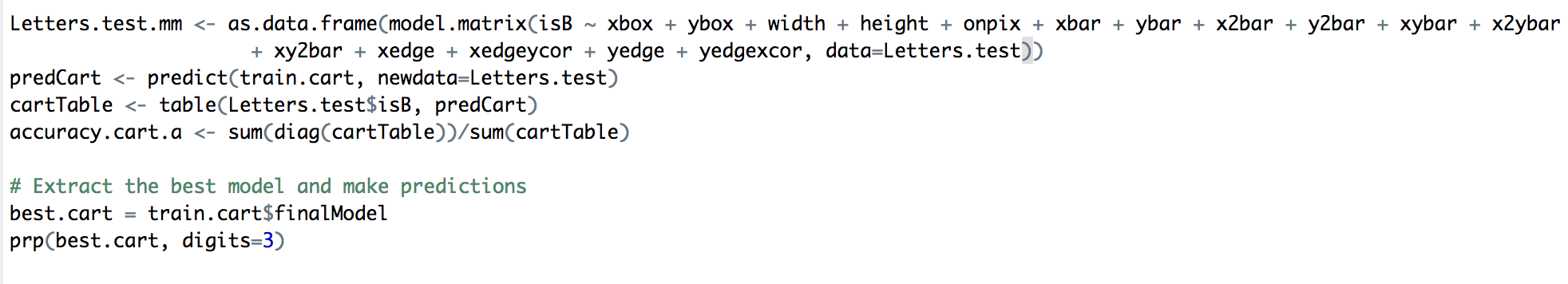


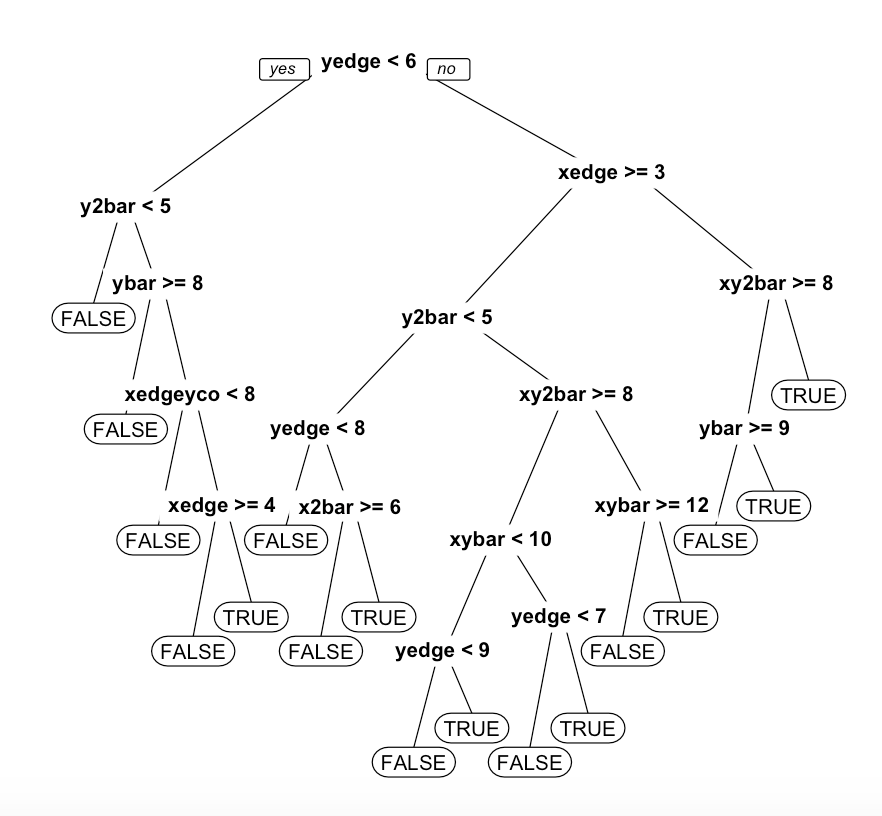


I did the cross validation utilizing the training mode and specifying the method, tuneGrid, trControl and metric. I specificed the method to be rpart for CART. For tuneGrid, I could change the cp values that are run through the model. In this case, I chose my cp values to be from 0 to 0.1 and incremented by 0.002. For trControl, I chose to use a 10-fold cross validation, and we based our metric on "Accuracy"

Given this, we found that we had the best accuracy with a cp value of 0.006, with an accuracy of 0.9249593

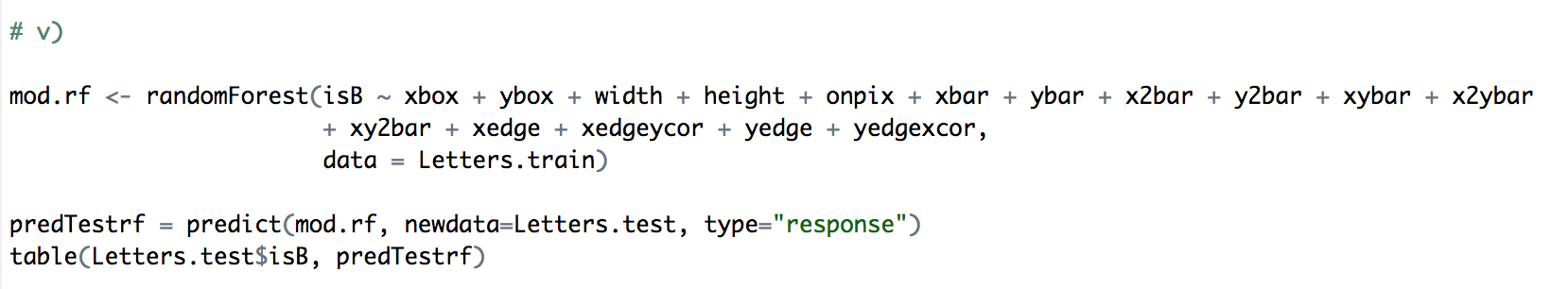
Once we have the best model, we then use it to make predictions on the test set.

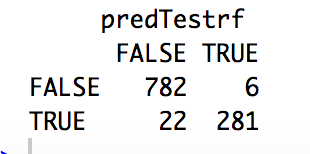




The accuracy on the test set is found to be 0.9248 which is very close to the accuracy of the model on the train data.

**v)**





The accuracy of the random forest model on the test set is (781 + 283)/(781+283+20+7) = 0.9753. Thus, the accuracy is 97.53%

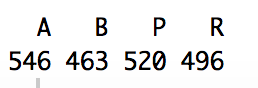
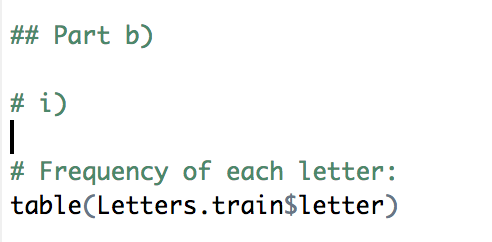
**vi)**

We had the highest accuracy with our random forest model. This is because we got 97.53% for random forest, 92.49% for CART, and 94.68% for the logistic regression.

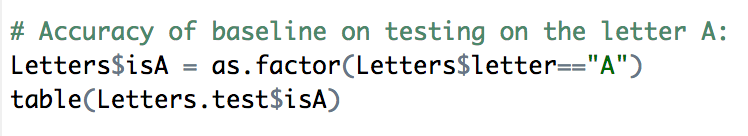
Regarding this application, accuracy is more important than interpretability. This is because we are looking at letter recognition, and our letter recognition has to be right. When identifying these letters, interpretability is less important because the code is not communicated to any people, and the people involve only worry if the results are correct rather than if they could interpret it.

**Part b)**

**i)**



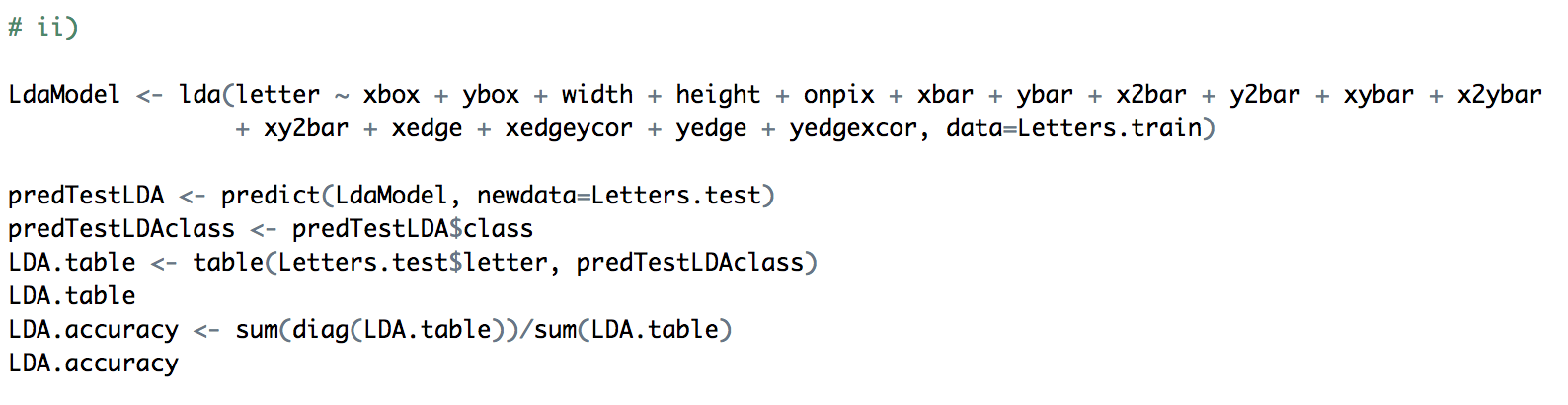
We see that ‘A’ is the most frequent letter in the train dataset. Thus, we create our baseline model to predict that all the letters are ‘A’. We then test this baseline model on the test dataset.

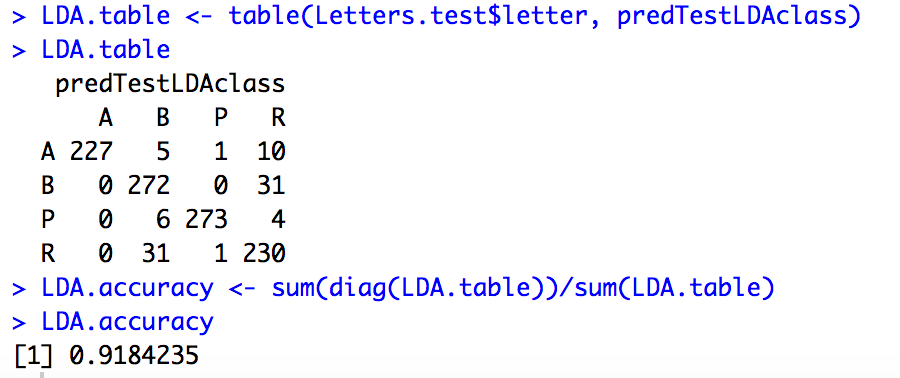


From this, we find that the baseline accuracy on the test set is 243/(243+848) = 0.223

Thus, we get that the accuracy is 22.3%

**ii)**

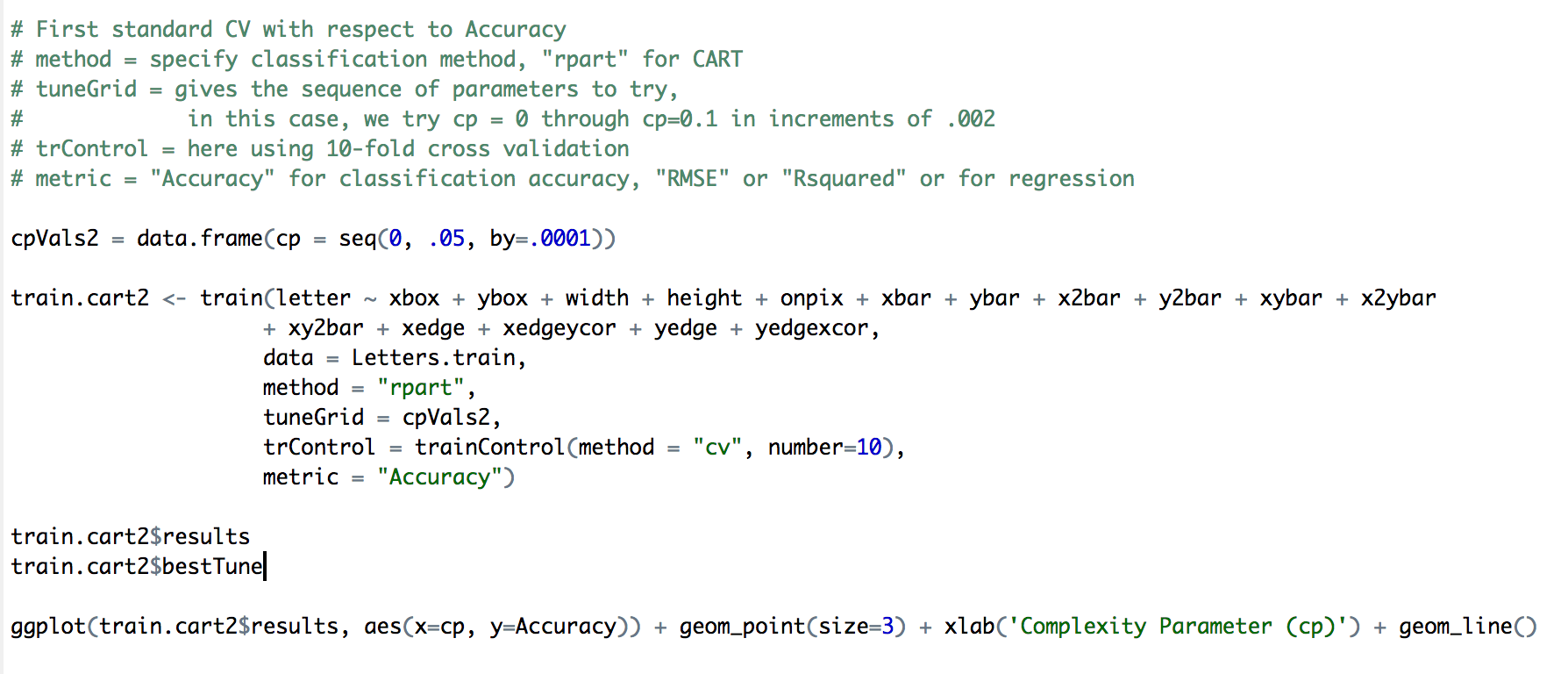




The table above is our contingency table, where the numbers in the diagonal are our true positives. The code above also calculated our accuracy to be 0.9184 = 91.84%

Accuracy = (227 + 272 + 273 + 230) / (227 + 5 + 1 + 10 + 272 + 31 + 6 + 273 + 4 + 31 + 1 + 230)

= 0.9184235

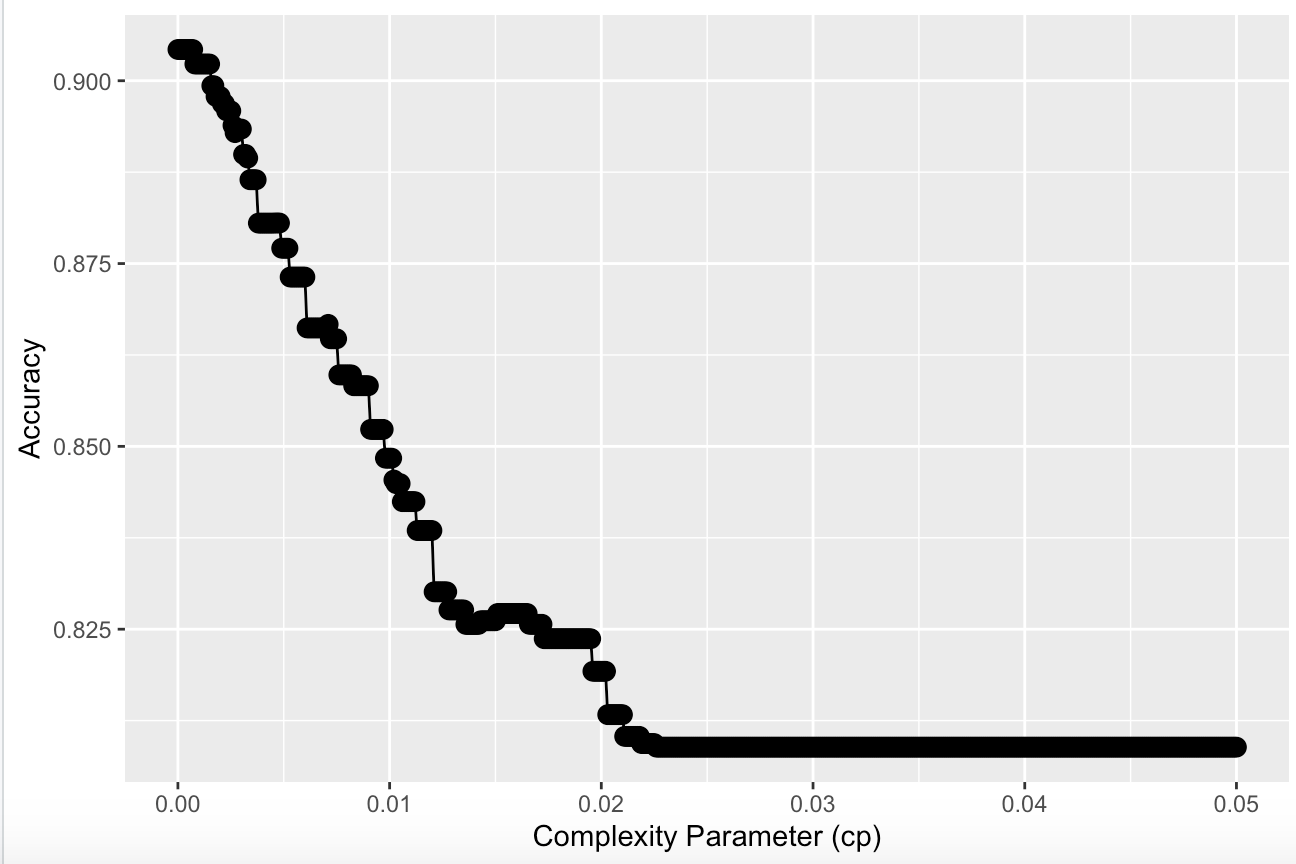


**iii)**

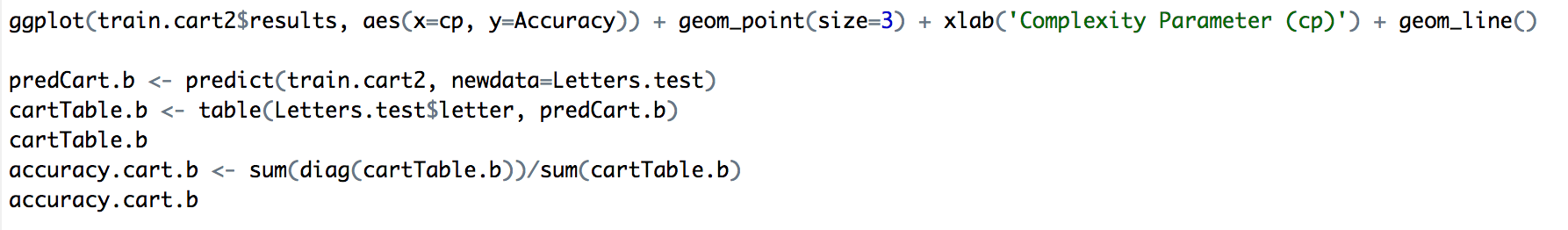
I did the cross validation utilizing the training mode and specifying the method, tuneGrid, trControl and metric. I specificed the method to be rpart for CART. For tuneGrid, I could change the cp values that are run through the model. In this case, I chose my cp values to be from 0 to 0.05 and incremented by 0.0001. For trControl, I chose to use a 10-fold cross validation, and we based our metric on "Accuracy"

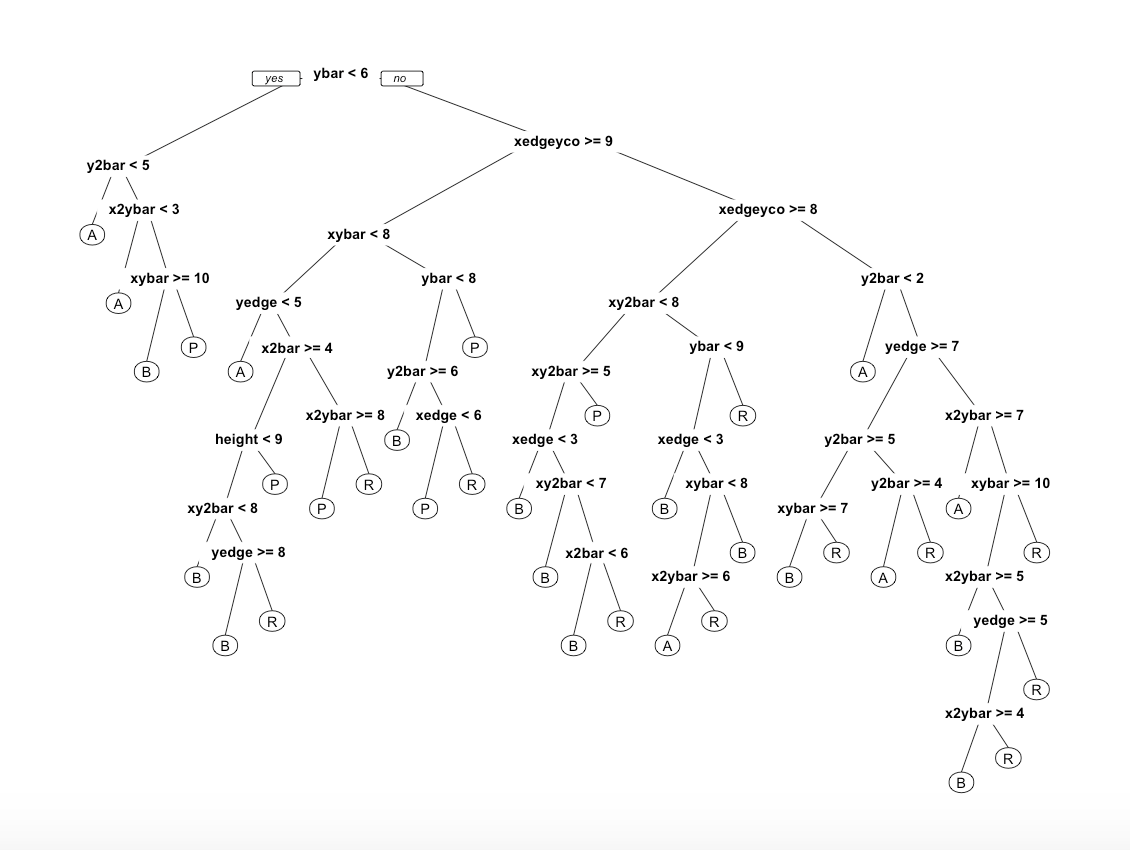
Given this, we found that we had the best accuracy with a cp value of 0.0007, with an accuracy of 0.9106 = 91.06%

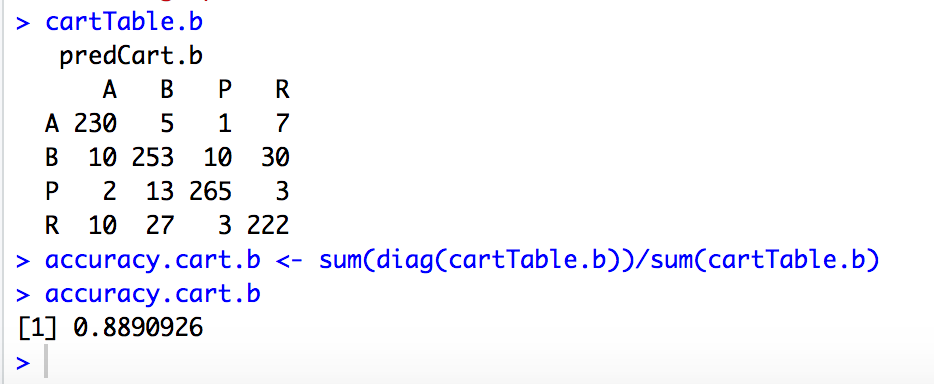
We get the following plot to see the relationship between the accuracy and the complexity parameter.



Once we have the best model, we then use it to make predictions on the test set.



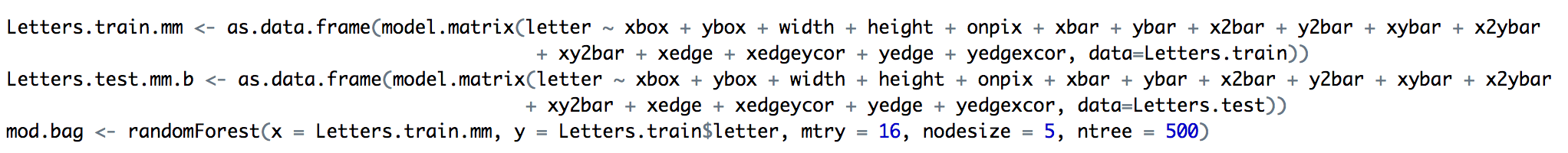




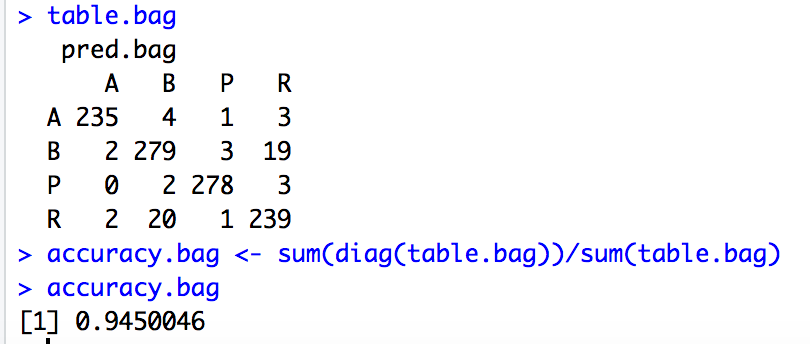
The contingency table is shown above, and the accuracy is found to be 0.88909 = 88.91%

**iv)**

We want to predict the letter by using “vanilla” bagging on the cart models. The parameter mtry is set to 16, because we have 16 features.

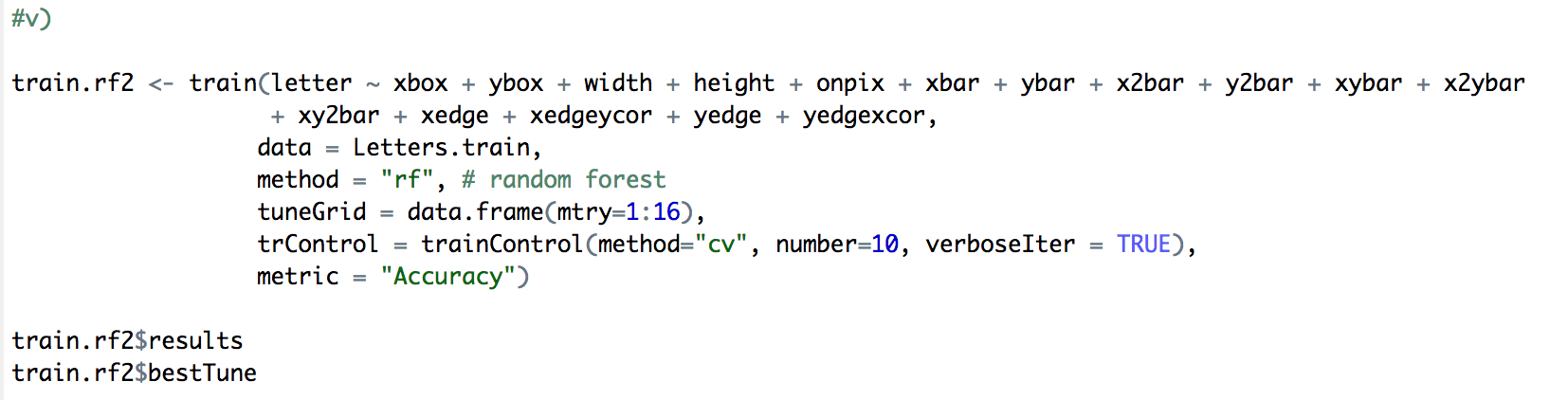


We have a prediction of the model from the code shown above, we then test this model on the test set model and find the accuracy. This is shown in the code below.



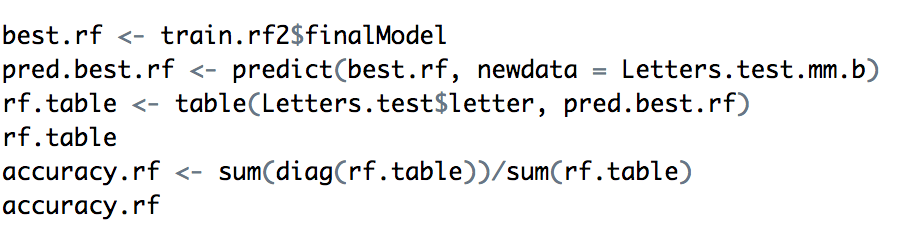
Thus, from this, we find that the accuracy is equal to 0.945 = 94.5%. The accuracy is calculated by summing the numbers in the diagonal, divided by the sum of all the numbers in the table.

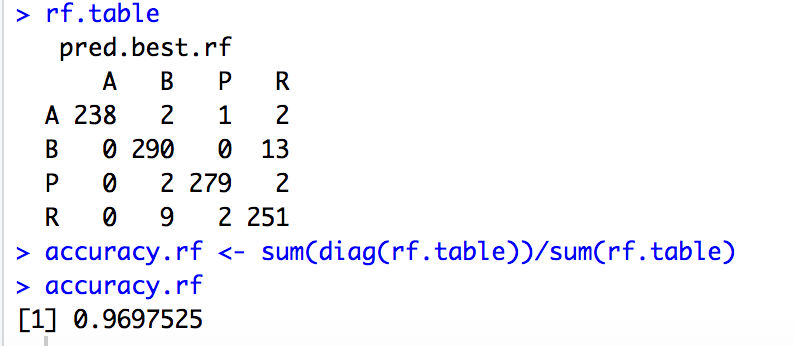
**v)**



We use tuneGrid and we give it a dataframe of mtry values ranging from 1 to 16, we set our method as 'cv' or cross validation = 10, and from this, we found that the mtry value with the highest accuracy is a mtry value of 4. Additionally, train.rf2$bestTune gives us a mtry value of 4.

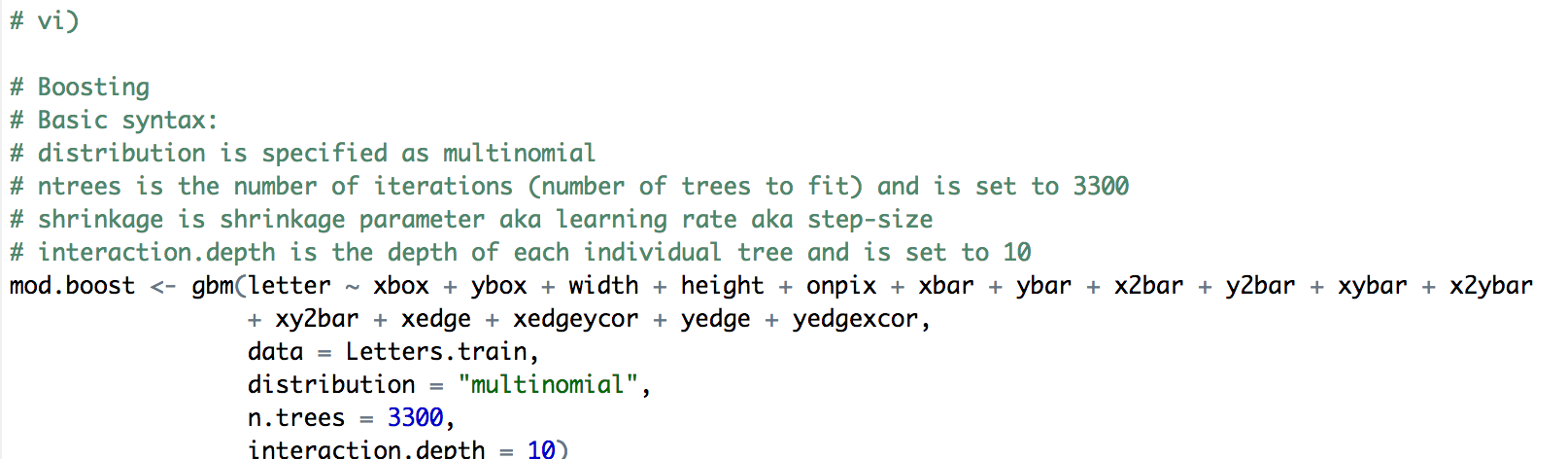
With our best model, we then implement it on the test data. This is shown in the code below.



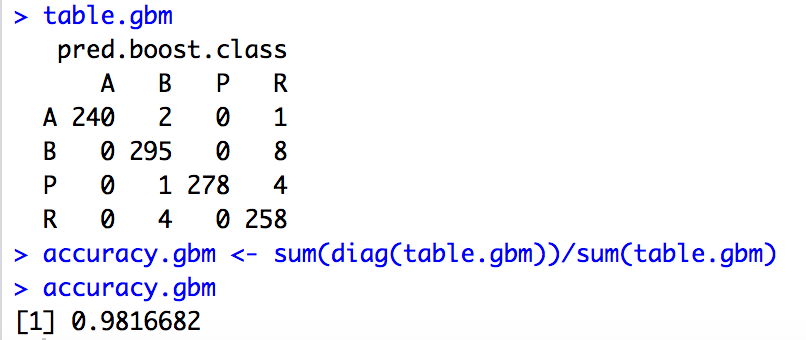
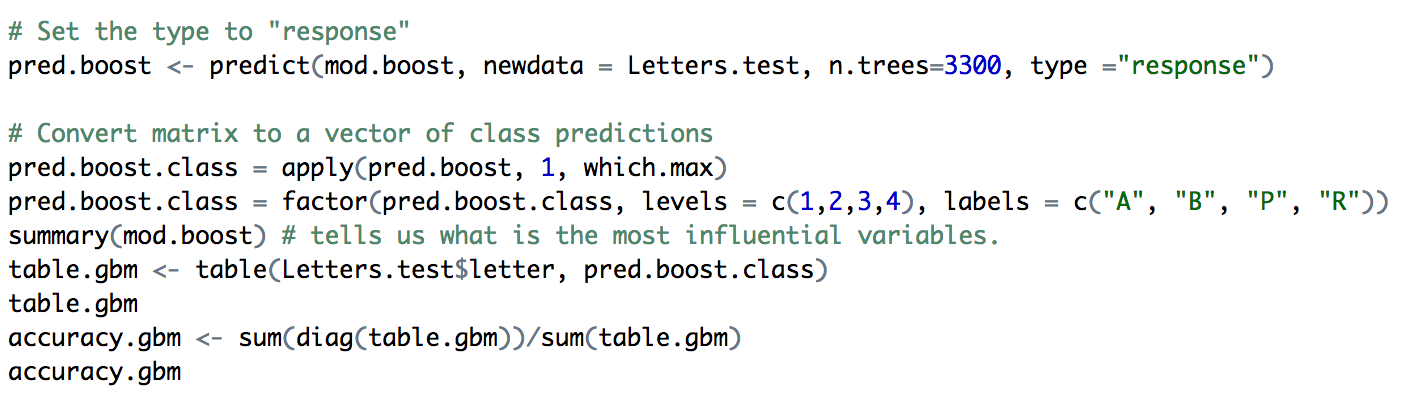


From this, we see that the accuracy we get is equal to 0.9698 = 96.98%

**vi)**



Once we have a prediction, we then apply to the test set data. This is shown in the code below.



The accuracy of this model is found to be. 0.9816682 or 98.17%.

**vii)**

|  |  |
| --- | --- |
| **Model** | **Accuracy** |
| Baseline | 0.223 |
| LDA | 0.9184235 |
| CART | 0.8890926 |
| Bagging | 0.9450046 |
| Random Forest | 0.9697525 |
| Boosting | 0.9816682 |

The highest accuracy we have is 0.9816682 and that is from the boosting to multi-classification. The lowest accuracy we. Have is from the CART model with an accuracy of 0.8890926. The model we recommended form part a) is the random forest model with an accuracy of 97.53%. Because we got a higher accuracy value for the boosting method, the boosting method is the optimal method to use. Thus, our choice is different from the model we recommended in part a). I would choose the boosting method because though it ls le less interpretable than the simple random forest model from part a), the accuracy is higher.

**CODE**

# IND242HW3 Problem 2

# Nicolas Kardous

library(dplyr)

library(ggplot2)

library(caTools) # splits

library(rpart) # CART

library(rpart.plot) # CART plotting

library(caret) # cross validation

library(randomForest)

library(ROCR)

library(MASS)

library(gbm)

# Part a)

Letters = read.csv("Letters.csv")

Letters$isB = as.factor(Letters$letter == "B")

set.seed(456)

train.ids = sample(nrow(Letters), 0.65\*nrow(Letters))

Letters.train = Letters[train.ids,]

Letters.test = Letters[-train.ids,]

# i)

# Baseline model: predict that it is "not B"

# Accuracy of baseline on test set:

# Accuracy of baseline on testing:

table(Letters.test$isB)

# Model Accuracy: Number correct / Number in total = 788 / (788 + 303) = 0.7223

# The accuracy is 72.23%

# ii)

modLOG <- glm(isB ~ xbox + ybox + width + height + onpix + xbar + ybar + x2bar + y2bar + xybar + x2ybar

+ xy2bar + xedge + xedgeycor + yedge + yedgexcor, data = Letters.train, family = "binomial")

summary(modLOG)

predTest = predict(modLOG, newdata=Letters.test, type="response")

summary(predTest)

table(Letters.test$isB, predTest > 0.5)

# Using a threshold value of p = 0.5, we find that the accuracy is (760 + 273) / (760 + 28 + 30 + 273)

# Thus, the accuracy = 0.9468 = 94.68%

# iii)

rocr.log.pred <- prediction(predTest, Letters.test$isB)

logPerformance <- performance(rocr.log.pred, "tpr", "fpr")

plot(logPerformance, colorize = TRUE)

as.numeric(performance(rocr.log.pred, "auc")@y.values)

abline(0,1)

# The AUC value we found is 0.97967

# iv)

# First standard CV with respect to Accuracy

# method = specify classification method, "rpart" for CART

# tuneGrid = gives the sequence of parameters to try,

# in this case, we try cp = 0 through cp=0.1 in increments of .002

# trControl = here using 10-fold cross validation

# metric = "Accuracy" for classification accuracy, "RMSE" or "Rsquared" or for regression

cpVals = data.frame(cp = seq(0, .1, by=.002))

train.cart <- train(isB ~ xbox + ybox + width + height + onpix + xbar + ybar + x2bar + y2bar + xybar + x2ybar

+ xy2bar + xedge + xedgeycor + yedge + yedgexcor,

data = Letters.train,

method = "rpart",

tuneGrid = cpVals,

trControl = trainControl(method = "cv", number=10),

metric = "Accuracy")

train.cart$results

train.cart$bestTune

ggplot(train.cart$results, aes(x=cp, y=Accuracy)) + geom\_point(size=3) + xlab('Complexity Parameter (cp)') + geom\_line()

# I did the cross validation utilizing the training mode and specifying the method, tuneGrid, trControl and metric. I specificed the method to be rpart for CART. For tuneGrid, I could change the cp values that are run through the model. In this case, I chose my cp values to be from 0 to 0.1 and incremented by 0.002. For trControl, I chose to use a 10-fold cross validation, and we based our metric on "Accuracy"

# Given this, we found that we had the best accuracy with a cp value of 0.006, with an accuracy of 0.9249593

Letters.test.mm <- as.data.frame(model.matrix(isB ~ xbox + ybox + width + height + onpix + xbar + ybar + x2bar + y2bar + xybar + x2ybar

+ xy2bar + xedge + xedgeycor + yedge + yedgexcor, data=Letters.test))

predCart <- predict(train.cart, newdata=Letters.test)

cartTable <- table(Letters.test$isB, predCart)

accuracy.cart.a <- sum(diag(cartTable))/sum(cartTable)

# Extract the best model and make predictions

best.cart2 = train.cart$finalModel

prp(best.cart2, digits=3)

# v)

mod.rf <- randomForest(isB ~ xbox + ybox + width + height + onpix + xbar + ybar + x2bar + y2bar + xybar + x2ybar

+ xy2bar + xedge + xedgeycor + yedge + yedgexcor,

data = Letters.train)

predTestrf = predict(mod.rf, newdata=Letters.test, type="response")

table(Letters.test$isB, predTestrf)

# The accuracy of the random forest model on the test set is (781 + 283)/(781+283+20+7) = 0.9753

# Thus, the accuracy is 97.53%

# vi)

# We had the highest accuracy with our random forest model. This is because we got 97.53% for random forest, 92.49% for CART, and 94.68% for the logistic regression.

# Because all these accuracy values are relatively similar, it seems like interpretabiltiy if more important than accuracy. This is because we set our metric for the CART model to be based on accuracy, and its accuracy is not as high as our other two models. Additionally, the random forest model was very easy to implement and was much more simple and interpretable.

## Part b)

# i)

# Frequency of each letter:

table(Letters.train$letter)

# Accuracy of baseline on testing on the letter A:

Letters$isA = as.factor(Letters$letter=="A")

table(Letters.test$isA)

# ii)

LdaModel <- lda(letter ~ xbox + ybox + width + height + onpix + xbar + ybar + x2bar + y2bar + xybar + x2ybar

+ xy2bar + xedge + xedgeycor + yedge + yedgexcor, data=Letters.train)

predTestLDA <- predict(LdaModel, newdata=Letters.test)

predTestLDAclass <- predTestLDA$class

LDA.table <- table(Letters.test$letter, predTestLDAclass)

LDA.table

LDA.accuracy <- sum(diag(LDA.table))/sum(LDA.table)

LDA.accuracy

# iii)

# First standard CV with respect to Accuracy

# method = specify classification method, "rpart" for CART

# tuneGrid = gives the sequence of parameters to try,

# in this case, we try cp = 0 through cp=0.1 in increments of .002

# trControl = here using 10-fold cross validation

# metric = "Accuracy" for classification accuracy, "RMSE" or "Rsquared" or for regression

cpVals2 = data.frame(cp = seq(0, .05, by=.0001))

train.cart2 <- train(letter ~ xbox + ybox + width + height + onpix + xbar + ybar + x2bar + y2bar + xybar + x2ybar

+ xy2bar + xedge + xedgeycor + yedge + yedgexcor,

data = Letters.train,

method = "rpart",

tuneGrid = cpVals2,

trControl = trainControl(method = "cv", number=10),

metric = "Accuracy")

train.cart2$results

train.cart2$bestTune

# I did the cross validation utilizing the training mode and specifying the method, tuneGrid, trControl and metric. I specificed the method to be rpart for CART. For tuneGrid, I could change the cp values that are run through the model. In this case, I chose my cp values to be from 0 to 0.1 and incremented by 0.002. For trControl, I chose to use a 10-fold cross validation, and we based our metric on "Accuracy"

# Given this, we found that we had the best accuracy with a cp value of 0, with an accuracy of 0.90716

ggplot(train.cart2$results, aes(x=cp, y=Accuracy)) + geom\_point(size=3) + xlab('Complexity Parameter (cp)') + geom\_line()

predCart.b <- predict(train.cart2, newdata=Letters.test)

cartTable.b <- table(Letters.test$letter, predCart.b)

cartTable.b

accuracy.cart.b <- sum(diag(cartTable.b))/sum(cartTable.b)

accuracy.cart.b

# Extract the best model and make predictions

best.cart2 = train.cart2$finalModel

prp(best.cart2, digits=3)

# iv)

# bagging can be done just by setting mtry = p = 16

# "randomForest" doesn't break categorical variables for us, it treat them as one numeric column.

Letters.train.mm <- as.data.frame(model.matrix(letter ~ xbox + ybox + width + height + onpix + xbar + ybar + x2bar + y2bar + xybar + x2ybar

+ xy2bar + xedge + xedgeycor + yedge + yedgexcor, data=Letters.train))

Letters.test.mm.b <- as.data.frame(model.matrix(letter ~ xbox + ybox + width + height + onpix + xbar + ybar + x2bar + y2bar + xybar + x2ybar

+ xy2bar + xedge + xedgeycor + yedge + yedgexcor, data=Letters.test))

mod.bag <- randomForest(x = Letters.train.mm, y = Letters.train$letter, mtry = 16, nodesize = 5, ntree = 500)

pred.bag <- predict(mod.bag, newdata = Letters.test.mm.b)

table.bag <- table(Letters.test$letter, pred.bag)

table.bag

accuracy.bag <- sum(diag(table.bag))/sum(table.bag)

accuracy.bag

# Accuracy

#v)

train.rf2 <- train(letter ~ xbox + ybox + width + height + onpix + xbar + ybar + x2bar + y2bar + xybar + x2ybar

+ xy2bar + xedge + xedgeycor + yedge + yedgexcor,

data = Letters.train,

method = "rf", # random forest

tuneGrid = data.frame(mtry=1:16),

trControl = trainControl(method="cv", number=10, verboseIter = TRUE),

metric = "Accuracy")

train.rf2$results

train.rf2$bestTune

best.rf <- train.rf2$finalModel

pred.best.rf <- predict(best.rf, newdata = Letters.test.mm.b)

rf.table <- table(Letters.test$letter, pred.best.rf)

rf.table

accuracy.rf <- sum(diag(rf.table))/sum(rf.table)

accuracy.rf

# We use tuneGrid and we give it a dataframe of mtry values ranging from 1 to 16, we set our method as 'cv' or cross validation, and from this, we found that the mtry value with the highest accuracy is a mtry value of 4. Additionally, train.rf2$bestTune gives us a mtry value of 4

# Accuracy:

# vi)

# Boosting

# Basic syntax:

# distribution is specified as multinomial

# ntrees is the number of iterations (number of trees to fit) and is set to 3300

# shrinkage is shrinkage parameter aka learning rate aka step-size

# interaction.depth is the depth of each individual tree and is set to 10

set.seed(456)

mod.boost <- gbm(letter ~ xbox + ybox + width + height + onpix + xbar + ybar + x2bar + y2bar + xybar + x2ybar

+ xy2bar + xedge + xedgeycor + yedge + yedgexcor,

data = Letters.train,

distribution = "multinomial",

n.trees = 3300,

interaction.depth = 10)

# Set the type to "response"

pred.boost <- predict(mod.boost, newdata = Letters.test, n.trees=3300, type ="response")

# Convert matrix to a vector of class predictions

pred.boost.class = apply(pred.boost, 1, which.max)

pred.boost.class = factor(pred.boost.class, levels = c(1,2,3,4), labels = c("A", "B", "P", "R"))

summary(mod.boost) # tells us what is the most influential variables.

table.gbm <- table(Letters.test$letter, pred.boost.class)

table.gbm

accuracy.gbm <- sum(diag(table.gbm))/sum(table.gbm)

accuracy.gbm

# vii)

hh