Assignment\_3\_rmd

Emily

10/10/2019

## Warning: package 'tidyverse' was built under R version 3.5.3

## -- Attaching packages --------------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.1.0 v purrr 0.2.5  
## v tibble 2.1.3 v dplyr 0.8.3  
## v tidyr 1.0.0 v stringr 1.3.1  
## v readr 1.3.1 v forcats 0.3.0

## Warning: package 'tibble' was built under R version 3.5.3

## Warning: package 'tidyr' was built under R version 3.5.3

## Warning: package 'dplyr' was built under R version 3.5.3

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

##   
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':  
##   
## select

## Warning: package 'caTools' was built under R version 3.5.3

## Warning: package 'rpart' was built under R version 3.5.3

## Warning: package 'rpart.plot' was built under R version 3.5.3

## Warning: package 'Rcpp' was built under R version 3.5.3

## Warning: package 'caret' was built under R version 3.5.3

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

## Warning: package 'randomForest' was built under R version 3.5.3

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':  
##   
## combine

## The following object is masked from 'package:ggplot2':  
##   
## margin

## Warning: package 'gbm' was built under R version 3.5.3

## Loaded gbm 2.1.5

## Warning: package 'ROCR' was built under R version 3.5.3

## Loading required package: gplots

## Warning: package 'gplots' was built under R version 3.5.3

##   
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':  
##   
## lowess

## Question 2 a

Letters <- read\_csv("Letters.csv")

## Parsed with column specification:  
## cols(  
## letter = col\_character(),  
## xbox = col\_double(),  
## ybox = col\_double(),  
## width = col\_double(),  
## height = col\_double(),  
## onpix = col\_double(),  
## xbar = col\_double(),  
## ybar = col\_double(),  
## x2bar = col\_double(),  
## y2bar = col\_double(),  
## xybar = col\_double(),  
## x2ybar = col\_double(),  
## xy2bar = col\_double(),  
## xedge = col\_double(),  
## xedgeycor = col\_double(),  
## yedge = col\_double(),  
## yedgexcor = col\_double()  
## )

Letters$isB = as.factor(Letters$letter == "B")  
  
train.ids = sample(nrow(Letters), .65\*nrow(Letters))  
Letters.train = Letters[train.ids,]  
Letters.test = Letters[-train.ids,]

### Question 2 ai

Letters.train.mod <- Letters.train %>%   
 dplyr::select(-letter)  
  
Letters.test.mod <- Letters.test %>%   
 dplyr::select(-letter)  
  
table(Letters.train.mod$isB)

##   
## FALSE TRUE   
## 1534 491

accuracy\_isb\_baseline = length(Letters.train.mod$isB[Letters.train.mod$isB== FALSE])/nrow(Letters.train.mod)  
accuracy\_isb\_baseline

## [1] 0.7575309

table(Letters.test.mod$isB)

##   
## FALSE TRUE   
## 816 275

accuracy\_isb\_baseline\_t = length(Letters.test.mod$isB[Letters.test.mod$isB== FALSE])/nrow(Letters.test.mod)  
accuracy\_isb\_baseline\_t

## [1] 0.7479377

The accuracy of the baseline model (a model assuming that none of the letters are B) is 0.7575309 on the training set.

The accuracy of the baseline model is 0.7479377 on the test set.

### Question 2 aii

mod1 <- glm(isB ~., data=Letters.train.mod, family="binomial")  
summary(mod1)

##   
## Call:  
## glm(formula = isB ~ ., family = "binomial", data = Letters.train.mod)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.0643 -0.1662 -0.0220 -0.0001 3.6049   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -14.00557 2.48066 -5.646 1.64e-08 \*\*\*  
## xbox -0.08359 0.12097 -0.691 0.489603   
## ybox 0.03118 0.08810 0.354 0.723418   
## width -1.15274 0.15584 -7.397 1.40e-13 \*\*\*  
## height -0.66299 0.14135 -4.691 2.72e-06 \*\*\*  
## onpix 0.88239 0.12831 6.877 6.10e-12 \*\*\*  
## xbar 0.49522 0.12624 3.923 8.75e-05 \*\*\*  
## ybar -0.61615 0.11546 -5.336 9.49e-08 \*\*\*  
## x2bar -0.38652 0.09392 -4.116 3.86e-05 \*\*\*  
## y2bar 1.26900 0.12411 10.225 < 2e-16 \*\*\*  
## xybar 0.31424 0.08914 3.525 0.000423 \*\*\*  
## x2ybar 0.53404 0.11936 4.474 7.67e-06 \*\*\*  
## xy2bar -0.40590 0.10557 -3.845 0.000121 \*\*\*  
## xedge -0.19544 0.09006 -2.170 0.029990 \*   
## xedgeycor 0.04724 0.09671 0.488 0.625204   
## yedge 1.64355 0.12412 13.242 < 2e-16 \*\*\*  
## yedgexcor 0.42224 0.07266 5.811 6.20e-09 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 2243.33 on 2024 degrees of freedom  
## Residual deviance: 663.26 on 2008 degrees of freedom  
## AIC: 697.26  
##   
## Number of Fisher Scoring iterations: 8

let.test\_b = predict(mod1, newdata=Letters.test.mod, type="response")  
  
summary(let.test\_b)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.000000 0.000308 0.012858 0.239453 0.422538 0.999364

t1 = table(Letters.test.mod$isB, let.test\_b > 0.5)  
t1

##   
## FALSE TRUE  
## FALSE 798 18  
## TRUE 39 236

table(Letters.test.mod$isB)

##   
## FALSE TRUE   
## 816 275

accuracy\_isb = (t1[1,1]+t1[2,2])/nrow(Letters.test.mod)  
accuracy\_isb

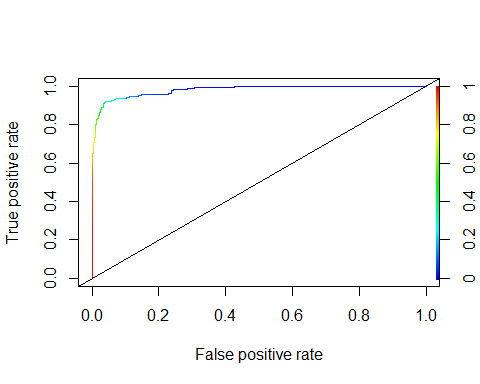
## [1] 0.9477544

The accuracy of the logistic model for predicting B on the test set is 0.9477544.

### Question 2 a iii

AUC of logistic regression model

rocr.log.pred <- prediction(let.test\_b, Letters.test.mod$isB)  
logPerformance <- performance(rocr.log.pred, "tpr", "fpr")  
plot(logPerformance, colorize = TRUE)  
abline(0, 1)



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

The Auc is 0.9807219.

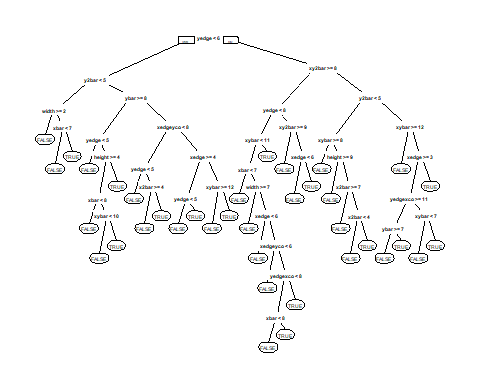
### Question 2 a iv

CART modeling B

modCART <- rpart(isB ~.,  
 data = Letters.train.mod, method="class",   
 minbucket=5, cp = 0.001)  
modCART

## n= 2025   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 2025 491 FALSE (0.75753086 0.24246914)   
## 2) yedge< 5.5 1369 120 FALSE (0.91234478 0.08765522)   
## 4) y2bar< 4.5 919 14 FALSE (0.98476605 0.01523395)   
## 8) width>=1.5 899 10 FALSE (0.98887653 0.01112347) \*  
## 9) width< 1.5 20 4 FALSE (0.80000000 0.20000000)   
## 18) xbar< 6.5 14 0 FALSE (1.00000000 0.00000000) \*  
## 19) xbar>=6.5 6 2 TRUE (0.33333333 0.66666667) \*  
## 5) y2bar>=4.5 450 106 FALSE (0.76444444 0.23555556)   
## 10) ybar>=7.5 236 18 FALSE (0.92372881 0.07627119)   
## 20) yedge< 4.5 158 2 FALSE (0.98734177 0.01265823) \*  
## 21) yedge>=4.5 78 16 FALSE (0.79487179 0.20512821)   
## 42) height>=3.5 66 8 FALSE (0.87878788 0.12121212)   
## 84) xbar< 7.5 37 1 FALSE (0.97297297 0.02702703) \*  
## 85) xbar>=7.5 29 7 FALSE (0.75862069 0.24137931)   
## 170) xybar< 9.5 20 2 FALSE (0.90000000 0.10000000) \*  
## 171) xybar>=9.5 9 4 TRUE (0.44444444 0.55555556) \*  
## 43) height< 3.5 12 4 TRUE (0.33333333 0.66666667) \*  
## 11) ybar< 7.5 214 88 FALSE (0.58878505 0.41121495)   
## 22) xedgeycor< 7.5 111 12 FALSE (0.89189189 0.10810811)   
## 44) yedge< 4.5 78 2 FALSE (0.97435897 0.02564103) \*  
## 45) yedge>=4.5 33 10 FALSE (0.69696970 0.30303030)   
## 90) x2bar>=3.5 20 2 FALSE (0.90000000 0.10000000) \*  
## 91) x2bar< 3.5 13 5 TRUE (0.38461538 0.61538462) \*  
## 23) xedgeycor>=7.5 103 27 TRUE (0.26213592 0.73786408)   
## 46) xedge>=3.5 22 5 FALSE (0.77272727 0.22727273)   
## 92) yedge< 4.5 13 0 FALSE (1.00000000 0.00000000) \*  
## 93) yedge>=4.5 9 4 TRUE (0.44444444 0.55555556) \*  
## 47) xedge< 3.5 81 10 TRUE (0.12345679 0.87654321)   
## 94) xybar>=11.5 5 1 FALSE (0.80000000 0.20000000) \*  
## 95) xybar< 11.5 76 6 TRUE (0.07894737 0.92105263) \*  
## 3) yedge>=5.5 656 285 TRUE (0.43445122 0.56554878)   
## 6) xy2bar>=7.5 250 59 FALSE (0.76400000 0.23600000)   
## 12) yedge< 7.5 195 30 FALSE (0.84615385 0.15384615)   
## 24) xybar< 10.5 183 20 FALSE (0.89071038 0.10928962)   
## 48) xbar< 6.5 74 0 FALSE (1.00000000 0.00000000) \*  
## 49) xbar>=6.5 109 20 FALSE (0.81651376 0.18348624)   
## 98) width>=6.5 51 3 FALSE (0.94117647 0.05882353) \*  
## 99) width< 6.5 58 17 FALSE (0.70689655 0.29310345)   
## 198) xedge< 5.5 23 2 FALSE (0.91304348 0.08695652) \*  
## 199) xedge>=5.5 35 15 FALSE (0.57142857 0.42857143)   
## 398) xedgeycor< 5.5 8 0 FALSE (1.00000000 0.00000000) \*  
## 399) xedgeycor>=5.5 27 12 TRUE (0.44444444 0.55555556)   
## 798) yedgexcor< 7.5 15 4 FALSE (0.73333333 0.26666667)   
## 1596) xbar< 7.5 8 0 FALSE (1.00000000 0.00000000) \*  
## 1597) xbar>=7.5 7 3 TRUE (0.42857143 0.57142857) \*  
## 799) yedgexcor>=7.5 12 1 TRUE (0.08333333 0.91666667) \*  
## 25) xybar>=10.5 12 2 TRUE (0.16666667 0.83333333) \*  
## 13) yedge>=7.5 55 26 TRUE (0.47272727 0.52727273)   
## 26) xy2bar>=8.5 20 1 FALSE (0.95000000 0.05000000) \*  
## 27) xy2bar< 8.5 35 7 TRUE (0.20000000 0.80000000)   
## 54) xedge< 5.5 10 4 FALSE (0.60000000 0.40000000) \*  
## 55) xedge>=5.5 25 1 TRUE (0.04000000 0.96000000) \*  
## 7) xy2bar< 7.5 406 94 TRUE (0.23152709 0.76847291)   
## 14) y2bar< 4.5 103 38 FALSE (0.63106796 0.36893204)   
## 28) xybar>=7.5 47 4 FALSE (0.91489362 0.08510638) \*  
## 29) xybar< 7.5 56 22 TRUE (0.39285714 0.60714286)   
## 58) height>=8.5 8 0 FALSE (1.00000000 0.00000000) \*  
## 59) height< 8.5 48 14 TRUE (0.29166667 0.70833333)   
## 118) x2bar>=6.5 11 3 FALSE (0.72727273 0.27272727) \*  
## 119) x2bar< 6.5 37 6 TRUE (0.16216216 0.83783784)   
## 238) x2bar< 3.5 6 1 FALSE (0.83333333 0.16666667) \*  
## 239) x2bar>=3.5 31 1 TRUE (0.03225806 0.96774194) \*  
## 15) y2bar>=4.5 303 29 TRUE (0.09570957 0.90429043)   
## 30) xybar>=12 8 0 FALSE (1.00000000 0.00000000) \*  
## 31) xybar< 12 295 21 TRUE (0.07118644 0.92881356)   
## 62) xedge>=2.5 109 21 TRUE (0.19266055 0.80733945)   
## 124) yedgexcor>=10.5 17 8 FALSE (0.52941176 0.47058824)   
## 248) ybar>=6.5 9 0 FALSE (1.00000000 0.00000000) \*  
## 249) ybar< 6.5 8 0 TRUE (0.00000000 1.00000000) \*  
## 125) yedgexcor< 10.5 92 12 TRUE (0.13043478 0.86956522)   
## 250) xybar< 6.5 13 6 FALSE (0.53846154 0.46153846) \*  
## 251) xybar>=6.5 79 5 TRUE (0.06329114 0.93670886) \*  
## 63) xedge< 2.5 186 0 TRUE (0.00000000 1.00000000) \*

prp(modCART)



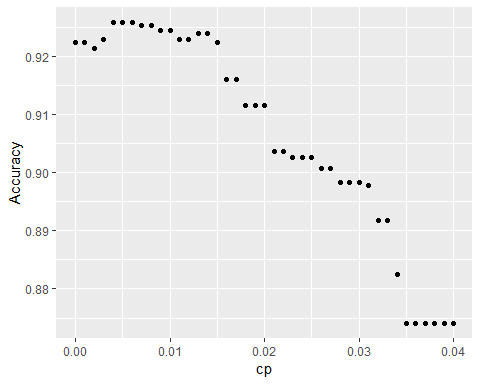
cpVals = data.frame(cp = seq(0, .04, by=.001))  
  
set.seed(123)  
train.cart <- train(isB ~.,  
 data = Letters.train.mod,  
 method = "rpart",  
 tuneGrid = cpVals,  
 trControl = trainControl(method = "cv", number=5),  
 metric = "Accuracy")  
  
# look at the cross validation results, stored as a data-frame  
# https://machinelearningmastery.com/machine-learning-evaluation-metrics-in-r/  
train.cart$results # please ignore kappa

## cp Accuracy Kappa AccuracySD KappaSD  
## 1 0.000 0.9224714 0.7846613 0.009794868 0.02684034  
## 2 0.001 0.9224714 0.7846613 0.009794868 0.02684034  
## 3 0.002 0.9214837 0.7810012 0.009283880 0.02519108  
## 4 0.003 0.9229628 0.7859741 0.009139753 0.02367268  
## 5 0.004 0.9259245 0.7944927 0.009082250 0.02383468  
## 6 0.005 0.9259245 0.7935887 0.009082250 0.02431339  
## 7 0.006 0.9259245 0.7935832 0.010628754 0.02880816  
## 8 0.007 0.9254307 0.7920566 0.010685880 0.02899665  
## 9 0.008 0.9254307 0.7931948 0.010685880 0.02919228  
## 10 0.009 0.9244431 0.7908300 0.012669611 0.03399768  
## 11 0.010 0.9244431 0.7908300 0.012669611 0.03399768  
## 12 0.011 0.9229628 0.7861616 0.012395340 0.03405161  
## 13 0.012 0.9229628 0.7861616 0.012395340 0.03405161  
## 14 0.013 0.9239529 0.7891667 0.013557548 0.03756087  
## 15 0.014 0.9239529 0.7891667 0.013557548 0.03756087  
## 16 0.015 0.9224751 0.7845892 0.012627890 0.03510238  
## 17 0.016 0.9160529 0.7633194 0.012198947 0.03373642  
## 18 0.017 0.9160529 0.7633194 0.012198947 0.03373642  
## 19 0.018 0.9116096 0.7487537 0.017210108 0.05375565  
## 20 0.019 0.9116096 0.7467535 0.017210108 0.05511852  
## 21 0.020 0.9116096 0.7467535 0.017210108 0.05511852  
## 22 0.021 0.9037023 0.7221837 0.009735249 0.03619148  
## 23 0.022 0.9037023 0.7221837 0.009735249 0.03619148  
## 24 0.023 0.9027158 0.7206315 0.013019443 0.04411649  
## 25 0.024 0.9027158 0.7206315 0.013019443 0.04411649  
## 26 0.025 0.9027158 0.7206315 0.013019443 0.04411649  
## 27 0.026 0.9007430 0.7153045 0.012858486 0.04115080  
## 28 0.027 0.9007430 0.7153045 0.012858486 0.04115080  
## 29 0.028 0.8982738 0.7105473 0.011481414 0.03692879  
## 30 0.029 0.8982738 0.7105473 0.011481414 0.03692879  
## 31 0.030 0.8982738 0.7105473 0.011481414 0.03692879  
## 32 0.031 0.8977800 0.7105213 0.011508052 0.03691278  
## 33 0.032 0.8918394 0.6883426 0.010685004 0.04937375  
## 34 0.033 0.8918394 0.6883426 0.010685004 0.04937375  
## 35 0.034 0.8824567 0.6474960 0.019641461 0.07550464  
## 36 0.035 0.8740726 0.6114280 0.010779747 0.04162607  
## 37 0.036 0.8740726 0.6114280 0.010779747 0.04162607  
## 38 0.037 0.8740726 0.6114280 0.010779747 0.04162607  
## 39 0.038 0.8740726 0.6114280 0.010779747 0.04162607  
## 40 0.039 0.8740726 0.6114280 0.010779747 0.04162607  
## 41 0.040 0.8740726 0.6114280 0.010779747 0.04162607

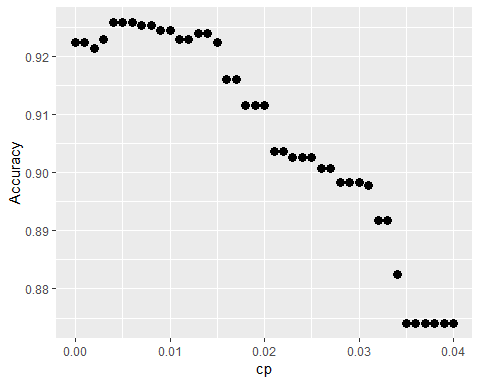
train.cart

## CART   
##   
## 2025 samples  
## 16 predictor  
## 2 classes: 'FALSE', 'TRUE'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 1621, 1620, 1620, 1619, 1620   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.000 0.9224714 0.7846613  
## 0.001 0.9224714 0.7846613  
## 0.002 0.9214837 0.7810012  
## 0.003 0.9229628 0.7859741  
## 0.004 0.9259245 0.7944927  
## 0.005 0.9259245 0.7935887  
## 0.006 0.9259245 0.7935832  
## 0.007 0.9254307 0.7920566  
## 0.008 0.9254307 0.7931948  
## 0.009 0.9244431 0.7908300  
## 0.010 0.9244431 0.7908300  
## 0.011 0.9229628 0.7861616  
## 0.012 0.9229628 0.7861616  
## 0.013 0.9239529 0.7891667  
## 0.014 0.9239529 0.7891667  
## 0.015 0.9224751 0.7845892  
## 0.016 0.9160529 0.7633194  
## 0.017 0.9160529 0.7633194  
## 0.018 0.9116096 0.7487537  
## 0.019 0.9116096 0.7467535  
## 0.020 0.9116096 0.7467535  
## 0.021 0.9037023 0.7221837  
## 0.022 0.9037023 0.7221837  
## 0.023 0.9027158 0.7206315  
## 0.024 0.9027158 0.7206315  
## 0.025 0.9027158 0.7206315  
## 0.026 0.9007430 0.7153045  
## 0.027 0.9007430 0.7153045  
## 0.028 0.8982738 0.7105473  
## 0.029 0.8982738 0.7105473  
## 0.030 0.8982738 0.7105473  
## 0.031 0.8977800 0.7105213  
## 0.032 0.8918394 0.6883426  
## 0.033 0.8918394 0.6883426  
## 0.034 0.8824567 0.6474960  
## 0.035 0.8740726 0.6114280  
## 0.036 0.8740726 0.6114280  
## 0.037 0.8740726 0.6114280  
## 0.038 0.8740726 0.6114280  
## 0.039 0.8740726 0.6114280  
## 0.040 0.8740726 0.6114280  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.006.

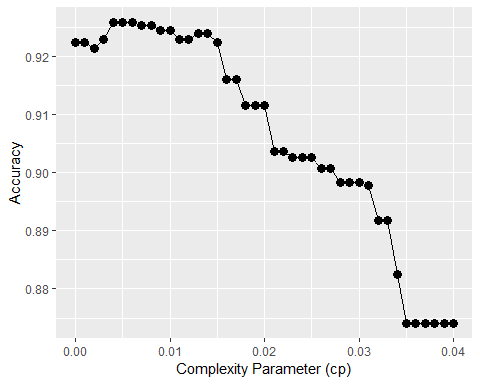
# plot the results  
ggplot(train.cart$results, aes(x=cp, y=Accuracy)) + geom\_point()



# We can increase the size of the points:  
ggplot(train.cart$results, aes(x=cp, y=Accuracy)) + geom\_point(size=3)



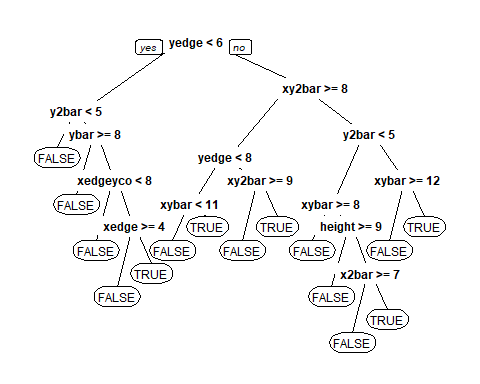
# We can change the default axis labels  
ggplot(train.cart$results, aes(x=cp, y=Accuracy)) + geom\_point(size=3) +  
 xlab("Complexity Parameter (cp)") + geom\_line()



# Extract the best model and make predictions  
train.cart$bestTune

## cp  
## 7 0.006

mod123 = train.cart$finalModel  
prp(mod123, digits=3)



Letters.test.mm = as.data.frame(model.matrix(isB~.+0, data=Letters.test.mod))  
pred\_cart = predict(mod123, newdata=Letters.test.mm, type="class")  
tcart = table(Letters.test.mod$isB, pred\_cart)  
  
accuracy\_isb\_cart = (tcart[1,1]+tcart[2,2])/nrow(Letters.test.mod)  
accuracy\_isb\_cart

## [1] 0.9257562

The accuracy of the CART model is 0.9257562. The cp value chosed to construct the CART model is 0.006. This value is chosen by cross validation; the model is run repeatedly with a set seed and one trial for each cp value from 0 to .04 increaing by .001 intervals, and the cp value which produces the highest training set accuracy is selected.

### Question 2a 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?

set.seed(144)  
mod.let.rf <- randomForest(isB ~ ., data = Letters.train.mod, mtry = 5, nodesize = 5, ntree = 500)  
  
pred.let.rf <- predict(mod.let.rf, newdata = Letters.test.mod)  
  
t\_rf = table(Letters.test.mod$isB, pred.let.rf)  
t\_rf

## pred.let.rf  
## FALSE TRUE  
## FALSE 812 4  
## TRUE 22 253

accuracy\_isb\_rf = (t\_rf[1,1]+t\_rf[2,2])/nrow(Letters.test.mod)  
accuracy\_isb\_rf

## [1] 0.9761687

The accuracy of the random forest model is 0.9761687

### Question 2a vi

accuracy\_isb

## [1] 0.9477544

accuracy\_isb\_cart

## [1] 0.9257562

accuracy\_isb\_rf

## [1] 0.9761687

The accuracy of the random forest model is highest, the accuracy of the CART model is lowest, and the accuracy of the logistic model is in the middle. In this case, accuracy is more important than interpretability. Identifying letters based on text characteristics has limited moral implications and is not susceptible to biases in the input data, meaning that interpretability of the factors and decisions which cause one letter to be identified and not another is not critical. It is extremely important that models such as those which guide decisions on parole are interpretable, as these have significant implications for people’s lives and it is important that all stakeholders be able to identify why a decision is made. In the case of letter idenfitication, accuracy is more important than interpretability.

## Question 2b

### Question 2b i

baseline\_a = table(Letters.train$letter)  
baseline\_a["P"]

## P   
## 524

accuracy\_alla = baseline\_a["P"]/nrow(Letters.train)  
  
#based on this table the most common letter is P  
Letters.train.mod2 <- Letters.train %>%   
 mutate(letter.f = as.factor(letter)) %>%   
 dplyr::select(-letter) %>%   
 dplyr::select(-isB)  
  
Letters.test.mod2 <- Letters.test %>%   
 mutate(letter.f = as.factor(letter)) %>%   
 dplyr::select(-letter) %>%   
 dplyr::select(-isB)

The most common letter in the training set is P. The accuracy of the baseline set is 0.2587654. The baseline model predicts that all letters are P, therefore is correct for all P and incorrect for all other letters. ### Question 2b ii LDA modeling

LDA\_let <- lda(letter.f ~ ., Letters.train.mod2)  
LDA\_test\_let <- predict(LDA\_let, Letters.test.mod2)  
  
LDA\_t <- table(Letters.test.mod2$letter.f, LDA\_test\_let$class)  
LDA\_t

##   
## A B P R  
## A 252 1 1 13  
## B 0 245 1 29  
## P 1 14 260 4  
## R 0 20 0 250

accuracy\_LDA = (LDA\_t[1,1]+LDA\_t[2,2]+LDA\_t[3,3]+LDA\_t[4,4])/nrow(Letters.test.mod2)  
accuracy\_LDA

## [1] 0.9230064

The accuracy of the LDA model on the test set is 0.9230064.

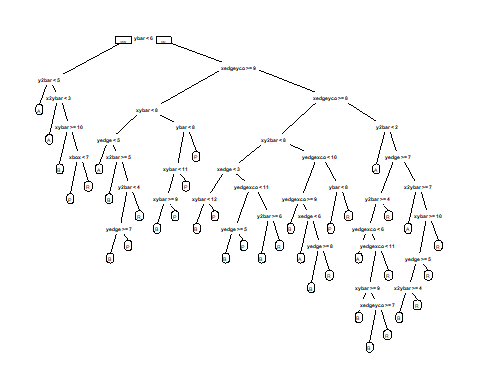
### Question 2b iii

CART modeling

modCART\_let <- rpart(letter.f ~.,  
 data = Letters.train.mod2, method="class",   
 minbucket=5, cp = 0.001)  
modCART\_let

## n= 2025   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 2025 1501 P (0.257777778 0.242469136 0.258765432 0.240987654)   
## 2) ybar< 5.5 467 31 A (0.933618844 0.029978587 0.017130621 0.019271949)   
## 4) y2bar< 4.5 433 4 A (0.990762125 0.000000000 0.004618938 0.004618938) \*  
## 5) y2bar>=4.5 34 20 B (0.205882353 0.411764706 0.176470588 0.205882353)   
## 10) x2ybar< 2.5 7 1 A (0.857142857 0.000000000 0.000000000 0.142857143) \*  
## 11) x2ybar>=2.5 27 13 B (0.037037037 0.518518519 0.222222222 0.222222222)   
## 22) xybar>=9.5 9 0 B (0.000000000 1.000000000 0.000000000 0.000000000) \*  
## 23) xybar< 9.5 18 12 P (0.055555556 0.277777778 0.333333333 0.333333333)   
## 46) xbox< 6.5 11 5 P (0.090909091 0.363636364 0.545454545 0.000000000) \*  
## 47) xbox>=6.5 7 1 R (0.000000000 0.142857143 0.000000000 0.857142857) \*  
## 3) ybar>=5.5 1558 1042 P (0.055198973 0.306161746 0.331193838 0.307445443)   
## 6) xedgeycor>=8.5 634 136 P (0.033123028 0.118296530 0.785488959 0.063091483)   
## 12) xybar< 7.5 128 72 B (0.164062500 0.437500000 0.156250000 0.242187500)   
## 24) yedge< 4.5 23 3 A (0.869565217 0.000000000 0.130434783 0.000000000) \*  
## 25) yedge>=4.5 105 49 B (0.009523810 0.533333333 0.161904762 0.295238095)   
## 50) x2bar>=4.5 49 7 B (0.020408163 0.857142857 0.040816327 0.081632653) \*  
## 51) x2bar< 4.5 56 29 R (0.000000000 0.250000000 0.267857143 0.482142857)   
## 102) y2bar< 3.5 23 9 P (0.000000000 0.347826087 0.608695652 0.043478261)   
## 204) yedge>=6.5 9 2 B (0.000000000 0.777777778 0.111111111 0.111111111) \*  
## 205) yedge< 6.5 14 1 P (0.000000000 0.071428571 0.928571429 0.000000000) \*  
## 103) y2bar>=3.5 33 7 R (0.000000000 0.181818182 0.030303030 0.787878788) \*  
## 13) xybar>=7.5 506 28 P (0.000000000 0.037549407 0.944664032 0.017786561)   
## 26) ybar< 7.5 44 24 P (0.000000000 0.386363636 0.454545455 0.159090909)   
## 52) xybar< 10.5 33 16 B (0.000000000 0.515151515 0.272727273 0.212121212)   
## 104) xybar>=8.5 21 5 B (0.000000000 0.761904762 0.095238095 0.142857143) \*  
## 105) xybar< 8.5 12 5 P (0.000000000 0.083333333 0.583333333 0.333333333) \*  
## 53) xybar>=10.5 11 0 P (0.000000000 0.000000000 1.000000000 0.000000000) \*  
## 27) ybar>=7.5 462 4 P (0.000000000 0.004329004 0.991341991 0.004329004) \*  
## 7) xedgeycor< 8.5 924 485 R (0.070346320 0.435064935 0.019480519 0.475108225)   
## 14) xedgeycor>=7.5 424 105 B (0.047169811 0.752358491 0.042452830 0.158018868)   
## 28) xy2bar< 7.5 348 45 B (0.014367816 0.870689655 0.037356322 0.077586207)   
## 56) xedge< 2.5 247 10 B (0.004048583 0.959514170 0.024291498 0.012145749)   
## 112) xybar< 11.5 242 6 B (0.004132231 0.975206612 0.008264463 0.012396694) \*  
## 113) xybar>=11.5 5 1 P (0.000000000 0.200000000 0.800000000 0.000000000) \*  
## 57) xedge>=2.5 101 35 B (0.039603960 0.653465347 0.069306931 0.237623762)   
## 114) yedgexcor< 10.5 83 23 B (0.048192771 0.722891566 0.084337349 0.144578313)   
## 228) yedge>=4.5 73 13 B (0.027397260 0.821917808 0.027397260 0.123287671) \*  
## 229) yedge< 4.5 10 5 P (0.200000000 0.000000000 0.500000000 0.300000000) \*  
## 115) yedgexcor>=10.5 18 6 R (0.000000000 0.333333333 0.000000000 0.666666667)   
## 230) y2bar>=5.5 8 2 B (0.000000000 0.750000000 0.000000000 0.250000000) \*  
## 231) y2bar< 5.5 10 0 R (0.000000000 0.000000000 0.000000000 1.000000000) \*  
## 29) xy2bar>=7.5 76 36 R (0.197368421 0.210526316 0.065789474 0.526315789)   
## 58) yedgexcor< 9.5 42 27 A (0.357142857 0.333333333 0.000000000 0.309523810)   
## 116) yedgexcor>=8.5 9 0 B (0.000000000 1.000000000 0.000000000 0.000000000) \*  
## 117) yedgexcor< 8.5 33 18 A (0.454545455 0.151515152 0.000000000 0.393939394)   
## 234) xedge< 5.5 16 3 A (0.812500000 0.000000000 0.000000000 0.187500000) \*  
## 235) xedge>=5.5 17 7 R (0.117647059 0.294117647 0.000000000 0.588235294)   
## 470) yedge>=7.5 6 1 B (0.166666667 0.833333333 0.000000000 0.000000000) \*  
## 471) yedge< 7.5 11 1 R (0.090909091 0.000000000 0.000000000 0.909090909) \*  
## 59) yedgexcor>=9.5 34 7 R (0.000000000 0.058823529 0.147058824 0.794117647)   
## 118) ybar< 7.5 9 4 P (0.000000000 0.222222222 0.555555556 0.222222222) \*  
## 119) ybar>=7.5 25 0 R (0.000000000 0.000000000 0.000000000 1.000000000) \*  
## 15) xedgeycor< 7.5 500 128 R (0.090000000 0.166000000 0.000000000 0.744000000)   
## 30) y2bar< 1.5 27 0 A (1.000000000 0.000000000 0.000000000 0.000000000) \*  
## 31) y2bar>=1.5 473 101 R (0.038054968 0.175475687 0.000000000 0.786469345)   
## 62) yedge>=6.5 94 46 B (0.095744681 0.510638298 0.000000000 0.393617021)   
## 124) y2bar>=3.5 77 29 B (0.103896104 0.623376623 0.000000000 0.272727273)   
## 248) yedgexcor< 5.5 7 0 A (1.000000000 0.000000000 0.000000000 0.000000000) \*  
## 249) yedgexcor>=5.5 70 22 B (0.014285714 0.685714286 0.000000000 0.300000000)   
## 498) yedgexcor< 10.5 58 13 B (0.017241379 0.775862069 0.000000000 0.206896552)   
## 996) xybar>=8.5 33 2 B (0.000000000 0.939393939 0.000000000 0.060606061) \*  
## 997) xybar< 8.5 25 11 B (0.040000000 0.560000000 0.000000000 0.400000000)   
## 1994) xedgeycor>=6.5 17 3 B (0.058823529 0.823529412 0.000000000 0.117647059) \*  
## 1995) xedgeycor< 6.5 8 0 R (0.000000000 0.000000000 0.000000000 1.000000000) \*  
## 499) yedgexcor>=10.5 12 3 R (0.000000000 0.250000000 0.000000000 0.750000000) \*  
## 125) y2bar< 3.5 17 1 R (0.058823529 0.000000000 0.000000000 0.941176471) \*  
## 63) yedge< 6.5 379 44 R (0.023746702 0.092348285 0.000000000 0.883905013)   
## 126) x2ybar>=6.5 7 1 A (0.857142857 0.142857143 0.000000000 0.000000000) \*  
## 127) x2ybar< 6.5 372 37 R (0.008064516 0.091397849 0.000000000 0.900537634)   
## 254) xybar>=9.5 89 25 R (0.000000000 0.280898876 0.000000000 0.719101124)   
## 508) yedge>=4.5 37 14 B (0.000000000 0.621621622 0.000000000 0.378378378)   
## 1016) x2ybar>=3.5 23 2 B (0.000000000 0.913043478 0.000000000 0.086956522) \*  
## 1017) x2ybar< 3.5 14 2 R (0.000000000 0.142857143 0.000000000 0.857142857) \*  
## 509) yedge< 4.5 52 2 R (0.000000000 0.038461538 0.000000000 0.961538462) \*  
## 255) xybar< 9.5 283 12 R (0.010600707 0.031802120 0.000000000 0.957597173) \*

prp(modCART\_let)



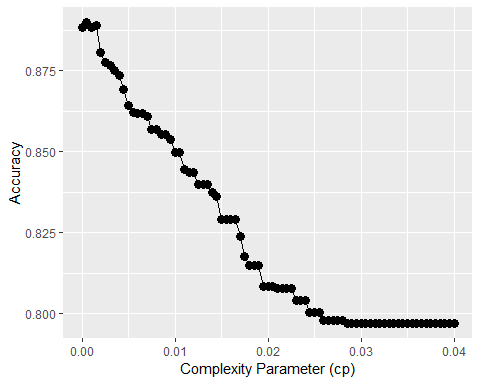
cpVals = data.frame(cp = seq(0, .04, by=.0005))  
  
set.seed(123)  
train.cart2 <- train(letter.f ~.,  
 data = Letters.train.mod2,  
 method = "rpart",  
 tuneGrid = cpVals,  
 trControl = trainControl(method = "cv", number=5),  
 metric = "Accuracy")  
  
# look at the cross validation results, stored as a data-frame  
# https://machinelearningmastery.com/machine-learning-evaluation-metrics-in-r/  
train.cart2$results # please ignore kappa

## cp Accuracy Kappa AccuracySD KappaSD  
## 1 0.0000 0.8884404 0.8512271 0.019632357 0.02615577  
## 2 0.0005 0.8899146 0.8531979 0.016528361 0.02200728  
## 3 0.0010 0.8884368 0.8512360 0.018385568 0.02447321  
## 4 0.0015 0.8889258 0.8518999 0.016110906 0.02142909  
## 5 0.0020 0.8805173 0.8407009 0.015537036 0.02068963  
## 6 0.0025 0.8775470 0.8367507 0.013509316 0.01798980  
## 7 0.0030 0.8765617 0.8354352 0.014288432 0.01902529  
## 8 0.0035 0.8750681 0.8334314 0.010888653 0.01451612  
## 9 0.0040 0.8735866 0.8314489 0.012308864 0.01640036  
## 10 0.0045 0.8691446 0.8254903 0.008190488 0.01090836  
## 11 0.0050 0.8642063 0.8189046 0.010036890 0.01337797  
## 12 0.0055 0.8622407 0.8162862 0.010808377 0.01439668  
## 13 0.0060 0.8617457 0.8156278 0.010036109 0.01336621  
## 14 0.0065 0.8617457 0.8156278 0.010036109 0.01336621  
## 15 0.0070 0.8607665 0.8143352 0.012419228 0.01652507  
## 16 0.0075 0.8568061 0.8090558 0.011335626 0.01510679  
## 17 0.0080 0.8568061 0.8090558 0.011335626 0.01510679  
## 18 0.0085 0.8553319 0.8070950 0.013154819 0.01751400  
## 19 0.0090 0.8553319 0.8070950 0.013154819 0.01751400  
## 20 0.0095 0.8538455 0.8051252 0.009727583 0.01294288  
## 21 0.0100 0.8498925 0.7998599 0.010091086 0.01342278  
## 22 0.0105 0.8498925 0.7998599 0.010091086 0.01342278  
## 23 0.0110 0.8444701 0.7926441 0.016067140 0.02135549  
## 24 0.0115 0.8434824 0.7913270 0.015646205 0.02079650  
## 25 0.0120 0.8434824 0.7913270 0.015646205 0.02079650  
## 26 0.0125 0.8400171 0.7867159 0.014098062 0.01874511  
## 27 0.0130 0.8400171 0.7867159 0.014098062 0.01874511  
## 28 0.0135 0.8400171 0.7867159 0.014098062 0.01874511  
## 29 0.0140 0.8375601 0.7834577 0.016958838 0.02252945  
## 30 0.0145 0.8360786 0.7814883 0.016246033 0.02158616  
## 31 0.0150 0.8291552 0.7722552 0.015696583 0.02090802  
## 32 0.0155 0.8291552 0.7722552 0.015696583 0.02090802  
## 33 0.0160 0.8291552 0.7722552 0.015696583 0.02090802  
## 34 0.0165 0.8291552 0.7722552 0.015696583 0.02090802  
## 35 0.0170 0.8237183 0.7650234 0.014565307 0.01939696  
## 36 0.0175 0.8177777 0.7571327 0.011901550 0.01589665  
## 37 0.0180 0.8148293 0.7532223 0.013495677 0.01797155  
## 38 0.0185 0.8148293 0.7532223 0.013495677 0.01797155  
## 39 0.0190 0.8148293 0.7532223 0.013495677 0.01797155  
## 40 0.0195 0.8083936 0.7445583 0.023582331 0.03162573  
## 41 0.0200 0.8083936 0.7445583 0.023582331 0.03162573  
## 42 0.0205 0.8083936 0.7445583 0.023582331 0.03162573  
## 43 0.0210 0.8078998 0.7439014 0.023112262 0.03100166  
## 44 0.0215 0.8078998 0.7439014 0.023112262 0.03100166  
## 45 0.0220 0.8078998 0.7439014 0.023112262 0.03100166  
## 46 0.0225 0.8078998 0.7439014 0.023112262 0.03100166  
## 47 0.0230 0.8039686 0.7386323 0.025598463 0.03427607  
## 48 0.0235 0.8039686 0.7386323 0.025598463 0.03427607  
## 49 0.0240 0.8039686 0.7386323 0.025598463 0.03427607  
## 50 0.0245 0.8005118 0.7339556 0.023150435 0.03097956  
## 51 0.0250 0.8005118 0.7339556 0.023150435 0.03097956  
## 52 0.0255 0.8005118 0.7339556 0.023150435 0.03097956  
## 53 0.0260 0.7980366 0.7305950 0.019330928 0.02580436  
## 54 0.0265 0.7980366 0.7305950 0.019330928 0.02580436  
## 55 0.0270 0.7980366 0.7305950 0.019330928 0.02580436  
## 56 0.0275 0.7980366 0.7305950 0.019330928 0.02580436  
## 57 0.0280 0.7980366 0.7305950 0.019330928 0.02580436  
## 58 0.0285 0.7970489 0.7291915 0.018852696 0.02512314  
## 59 0.0290 0.7970489 0.7291915 0.018852696 0.02512314  
## 60 0.0295 0.7970489 0.7291915 0.018852696 0.02512314  
## 61 0.0300 0.7970489 0.7291915 0.018852696 0.02512314  
## 62 0.0305 0.7970489 0.7291915 0.018852696 0.02512314  
## 63 0.0310 0.7970489 0.7291915 0.018852696 0.02512314  
## 64 0.0315 0.7970489 0.7291915 0.018852696 0.02512314  
## 65 0.0320 0.7970489 0.7291915 0.018852696 0.02512314  
## 66 0.0325 0.7970489 0.7291915 0.018852696 0.02512314  
## 67 0.0330 0.7970489 0.7291915 0.018852696 0.02512314  
## 68 0.0335 0.7970489 0.7291915 0.018852696 0.02512314  
## 69 0.0340 0.7970489 0.7291915 0.018852696 0.02512314  
## 70 0.0345 0.7970489 0.7291915 0.018852696 0.02512314  
## 71 0.0350 0.7970489 0.7291915 0.018852696 0.02512314  
## 72 0.0355 0.7970489 0.7291915 0.018852696 0.02512314  
## 73 0.0360 0.7970489 0.7291915 0.018852696 0.02512314  
## 74 0.0365 0.7970489 0.7291915 0.018852696 0.02512314  
## 75 0.0370 0.7970489 0.7291915 0.018852696 0.02512314  
## 76 0.0375 0.7970489 0.7291915 0.018852696 0.02512314  
## 77 0.0380 0.7970489 0.7291915 0.018852696 0.02512314  
## 78 0.0385 0.7970489 0.7291915 0.018852696 0.02512314  
## 79 0.0390 0.7970489 0.7291915 0.018852696 0.02512314  
## 80 0.0395 0.7970489 0.7291915 0.018852696 0.02512314  
## 81 0.0400 0.7970489 0.7291915 0.018852696 0.02512314

train.cart2

## CART   
##   
## 2025 samples  
## 16 predictor  
## 4 classes: 'A', 'B', 'P', 'R'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 1621, 1621, 1620, 1618, 1620   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.0000 0.8884404 0.8512271  
## 0.0005 0.8899146 0.8531979  
## 0.0010 0.8884368 0.8512360  
## 0.0015 0.8889258 0.8518999  
## 0.0020 0.8805173 0.8407009  
## 0.0025 0.8775470 0.8367507  
## 0.0030 0.8765617 0.8354352  
## 0.0035 0.8750681 0.8334314  
## 0.0040 0.8735866 0.8314489  
## 0.0045 0.8691446 0.8254903  
## 0.0050 0.8642063 0.8189046  
## 0.0055 0.8622407 0.8162862  
## 0.0060 0.8617457 0.8156278  
## 0.0065 0.8617457 0.8156278  
## 0.0070 0.8607665 0.8143352  
## 0.0075 0.8568061 0.8090558  
## 0.0080 0.8568061 0.8090558  
## 0.0085 0.8553319 0.8070950  
## 0.0090 0.8553319 0.8070950  
## 0.0095 0.8538455 0.8051252  
## 0.0100 0.8498925 0.7998599  
## 0.0105 0.8498925 0.7998599  
## 0.0110 0.8444701 0.7926441  
## 0.0115 0.8434824 0.7913270  
## 0.0120 0.8434824 0.7913270  
## 0.0125 0.8400171 0.7867159  
## 0.0130 0.8400171 0.7867159  
## 0.0135 0.8400171 0.7867159  
## 0.0140 0.8375601 0.7834577  
## 0.0145 0.8360786 0.7814883  
## 0.0150 0.8291552 0.7722552  
## 0.0155 0.8291552 0.7722552  
## 0.0160 0.8291552 0.7722552  
## 0.0165 0.8291552 0.7722552  
## 0.0170 0.8237183 0.7650234  
## 0.0175 0.8177777 0.7571327  
## 0.0180 0.8148293 0.7532223  
## 0.0185 0.8148293 0.7532223  
## 0.0190 0.8148293 0.7532223  
## 0.0195 0.8083936 0.7445583  
## 0.0200 0.8083936 0.7445583  
## 0.0205 0.8083936 0.7445583  
## 0.0210 0.8078998 0.7439014  
## 0.0215 0.8078998 0.7439014  
## 0.0220 0.8078998 0.7439014  
## 0.0225 0.8078998 0.7439014  
## 0.0230 0.8039686 0.7386323  
## 0.0235 0.8039686 0.7386323  
## 0.0240 0.8039686 0.7386323  
## 0.0245 0.8005118 0.7339556  
## 0.0250 0.8005118 0.7339556  
## 0.0255 0.8005118 0.7339556  
## 0.0260 0.7980366 0.7305950  
## 0.0265 0.7980366 0.7305950  
## 0.0270 0.7980366 0.7305950  
## 0.0275 0.7980366 0.7305950  
## 0.0280 0.7980366 0.7305950  
## 0.0285 0.7970489 0.7291915  
## 0.0290 0.7970489 0.7291915  
## 0.0295 0.7970489 0.7291915  
## 0.0300 0.7970489 0.7291915  
## 0.0305 0.7970489 0.7291915  
## 0.0310 0.7970489 0.7291915  
## 0.0315 0.7970489 0.7291915  
## 0.0320 0.7970489 0.7291915  
## 0.0325 0.7970489 0.7291915  
## 0.0330 0.7970489 0.7291915  
## 0.0335 0.7970489 0.7291915  
## 0.0340 0.7970489 0.7291915  
## 0.0345 0.7970489 0.7291915  
## 0.0350 0.7970489 0.7291915  
## 0.0355 0.7970489 0.7291915  
## 0.0360 0.7970489 0.7291915  
## 0.0365 0.7970489 0.7291915  
## 0.0370 0.7970489 0.7291915  
## 0.0375 0.7970489 0.7291915  
## 0.0380 0.7970489 0.7291915  
## 0.0385 0.7970489 0.7291915  
## 0.0390 0.7970489 0.7291915  
## 0.0395 0.7970489 0.7291915  
## 0.0400 0.7970489 0.7291915  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 5e-04.

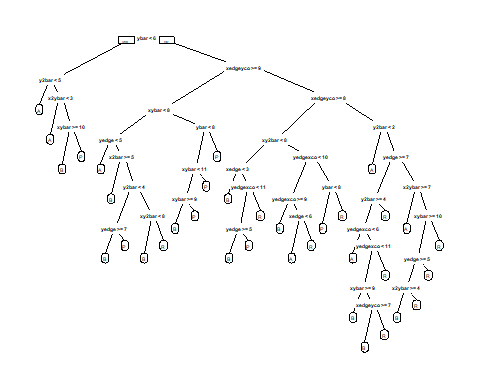
# plot the results  
ggplot(train.cart2$results, aes(x=cp, y=Accuracy)) + geom\_point(size=3) +  
 xlab("Complexity Parameter (cp)") + geom\_line()



# Extract the best model and make predictions  
train.cart2$bestTune

## cp  
## 2 5e-04

mod123 = train.cart2$finalModel  
prp(mod123, digits=3)



train.cart2

## CART   
##   
## 2025 samples  
## 16 predictor  
## 4 classes: 'A', 'B', 'P', 'R'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 1621, 1621, 1620, 1618, 1620   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.0000 0.8884404 0.8512271  
## 0.0005 0.8899146 0.8531979  
## 0.0010 0.8884368 0.8512360  
## 0.0015 0.8889258 0.8518999  
## 0.0020 0.8805173 0.8407009  
## 0.0025 0.8775470 0.8367507  
## 0.0030 0.8765617 0.8354352  
## 0.0035 0.8750681 0.8334314  
## 0.0040 0.8735866 0.8314489  
## 0.0045 0.8691446 0.8254903  
## 0.0050 0.8642063 0.8189046  
## 0.0055 0.8622407 0.8162862  
## 0.0060 0.8617457 0.8156278  
## 0.0065 0.8617457 0.8156278  
## 0.0070 0.8607665 0.8143352  
## 0.0075 0.8568061 0.8090558  
## 0.0080 0.8568061 0.8090558  
## 0.0085 0.8553319 0.8070950  
## 0.0090 0.8553319 0.8070950  
## 0.0095 0.8538455 0.8051252  
## 0.0100 0.8498925 0.7998599  
## 0.0105 0.8498925 0.7998599  
## 0.0110 0.8444701 0.7926441  
## 0.0115 0.8434824 0.7913270  
## 0.0120 0.8434824 0.7913270  
## 0.0125 0.8400171 0.7867159  
## 0.0130 0.8400171 0.7867159  
## 0.0135 0.8400171 0.7867159  
## 0.0140 0.8375601 0.7834577  
## 0.0145 0.8360786 0.7814883  
## 0.0150 0.8291552 0.7722552  
## 0.0155 0.8291552 0.7722552  
## 0.0160 0.8291552 0.7722552  
## 0.0165 0.8291552 0.7722552  
## 0.0170 0.8237183 0.7650234  
## 0.0175 0.8177777 0.7571327  
## 0.0180 0.8148293 0.7532223  
## 0.0185 0.8148293 0.7532223  
## 0.0190 0.8148293 0.7532223  
## 0.0195 0.8083936 0.7445583  
## 0.0200 0.8083936 0.7445583  
## 0.0205 0.8083936 0.7445583  
## 0.0210 0.8078998 0.7439014  
## 0.0215 0.8078998 0.7439014  
## 0.0220 0.8078998 0.7439014  
## 0.0225 0.8078998 0.7439014  
## 0.0230 0.8039686 0.7386323  
## 0.0235 0.8039686 0.7386323  
## 0.0240 0.8039686 0.7386323  
## 0.0245 0.8005118 0.7339556  
## 0.0250 0.8005118 0.7339556  
## 0.0255 0.8005118 0.7339556  
## 0.0260 0.7980366 0.7305950  
## 0.0265 0.7980366 0.7305950  
## 0.0270 0.7980366 0.7305950  
## 0.0275 0.7980366 0.7305950  
## 0.0280 0.7980366 0.7305950  
## 0.0285 0.7970489 0.7291915  
## 0.0290 0.7970489 0.7291915  
## 0.0295 0.7970489 0.7291915  
## 0.0300 0.7970489 0.7291915  
## 0.0305 0.7970489 0.7291915  
## 0.0310 0.7970489 0.7291915  
## 0.0315 0.7970489 0.7291915  
## 0.0320 0.7970489 0.7291915  
## 0.0325 0.7970489 0.7291915  
## 0.0330 0.7970489 0.7291915  
## 0.0335 0.7970489 0.7291915  
## 0.0340 0.7970489 0.7291915  
## 0.0345 0.7970489 0.7291915  
## 0.0350 0.7970489 0.7291915  
## 0.0355 0.7970489 0.7291915  
## 0.0360 0.7970489 0.7291915  
## 0.0365 0.7970489 0.7291915  
## 0.0370 0.7970489 0.7291915  
## 0.0375 0.7970489 0.7291915  
## 0.0380 0.7970489 0.7291915  
## 0.0385 0.7970489 0.7291915  
## 0.0390 0.7970489 0.7291915  
## 0.0395 0.7970489 0.7291915  
## 0.0400 0.7970489 0.7291915  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 5e-04.

# extract the "model matrix" for letter csv before we can make predictions  
# This is because caret does not work with factors, instead it creates dummy variables   
Letters.test.all.mm = as.data.frame(model.matrix(letter.f~.+0, data=Letters.test.mod2))  
pred\_cartisb = predict(mod123, newdata=Letters.test.all.mm, type="class")  
  
  
tcart\_all = table(Letters.test.mod2$letter.f, pred\_cartisb)  
tcart\_all

## pred\_cartisb  
## A B P R  
## A 259 3 1 4  
## B 4 229 17 25  
## P 3 16 260 0  
## R 9 16 8 237

accuracy\_cart\_all = (tcart\_all[1,1]+tcart\_all[2,2]+tcart\_all[3,3]+tcart\_all[4,4])/nrow(Letters.test.mod2)  
accuracy\_cart\_all

## [1] 0.9028414

The cross validation technique utilized in this model sets a seed and then runs the CART model with every possible cp value between 0 and .04 increasing by an increment of .0005. The cp value that produces the highest acuracy is selected as the cp value that will be used in the final CART model. The optimal cp value determined by cross validation for this model is 510^{-4}.

The accuracy of the resulting CART model on the test data is 0.9028414. ### Question 2b iv Vanilla bagging of CART models- random forest using all features (16 features to guess the letter)

set.seed(144)  
mod.let.rf.all <- randomForest(letter.f ~ ., data = Letters.train.mod2, mtry = 16, nodesize = 5, ntree = 500)  
  
pred.let.bag.all <- predict(mod.let.rf.all, newdata = Letters.test.mod2)  
  
t\_bag\_all = table(Letters.test.mod2$letter.f, pred.let.bag.all)  
t\_bag\_all

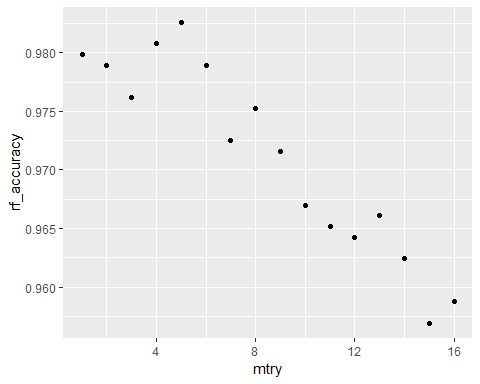
## pred.let.bag.all  
## A B P R  
## A 263 1 1 2  
## B 3 250 4 18  
## P 2 2 273 2  
## R 4 6 0 260

accuracy\_bagging = (t\_bag\_all[1,1]+t\_bag\_all[2,2]+t\_bag\_all[3,3]+t\_bag\_all[4,4])/nrow(Letters.test.mod2)  
accuracy\_bagging

## [1] 0.9587534

### Question 2b v

mtryVals = data.frame(mtry = seq(1, 16, by=1), rf\_accuracy = seq(1, 16, by = 1))  
  
#Cross validation looking at accuracy  
  
for(i in 1:16){  
 set.seed(144)  
 mod.let.rf.all <- randomForest(letter.f ~ ., data = Letters.train.mod2, mtry = i)  
  
 pred.let.rf.all <- predict(mod.let.rf.all, newdata = Letters.test.mod2)  
  
 t\_rf\_all = table(Letters.test.mod2$letter.f, pred.let.rf.all)  
 t\_rf\_all  
  
 accuracy\_let\_rf = (t\_rf\_all[1,1]+t\_rf\_all[2,2]+t\_rf\_all[3,3]+t\_rf\_all[4,4])/nrow(Letters.test.mod)  
 accuracy\_let\_rf  
   
 mtryVals$rf\_accuracy[i]= accuracy\_let\_rf  
  
}  
  
mtryVals %>% ggplot(aes(x = mtry, y = rf\_accuracy))+  
 geom\_point()



ideal\_mtry = mtryVals$mtry[which.is.max(mtryVals$rf\_accuracy)]  
  
mod.let.rf.all.f <- randomForest(letter.f ~ ., data = Letters.train.mod2, mtry = ideal\_mtry)  
  
pred.let.rf.all <- predict(mod.let.rf.all.f, newdata = Letters.test.mod2)  
  
t\_rf\_all = table(Letters.test.mod2$letter.f, pred.let.rf.all)  
t\_rf\_all

## pred.let.rf.all  
## A B P R  
## A 267 0 0 0  
## B 0 262 2 11  
## P 0 3 276 0  
## R 0 5 0 265

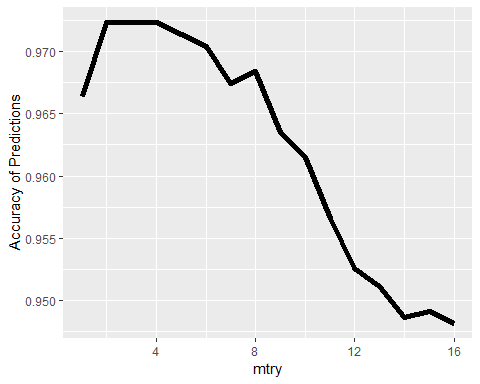
accuracy\_let\_rf = (t\_rf\_all[1,1]+t\_rf\_all[2,2]+t\_rf\_all[3,3]+t\_rf\_all[4,4])/nrow(Letters.test.mod)  
accuracy\_let\_rf

## [1] 0.9807516

set.seed(144)  
train.rf = train(letter.f~., data = Letters.train.mod2, method = "rf", tuneGrid = data.frame(mtry=seq(1, 16, 1)), trControl = trainControl(method = "cv", number = 5), metric = "Accuracy")  
  
best.rf = train.rf$finalModel  
best.rf

##   
## Call:  
## randomForest(x = x, y = y, mtry = param$mtry)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 3  
##   
## OOB estimate of error rate: 2.27%  
## Confusion matrix:  
## A B P R class.error  
## A 520 1 0 1 0.003831418  
## B 0 476 2 13 0.030549898  
## P 0 6 514 4 0.019083969  
## R 0 19 0 469 0.038934426

rf.plot <- ggplot(train.rf$results, aes(x=mtry, y=Accuracy)) + geom\_line(lwd=2) +  
 ylab("Accuracy of Predictions")  
rf.plot



Letters.test.mm = as.data.frame(model.matrix(letter.f ~. +0, data = Letters.test.mod2))  
  
set.seed(144)  
pred.best.rf = predict(best.rf, newdata = Letters.test.mm, type = "class")  
  
t\_rf\_all = table(Letters.test.mod2$letter.f, pred.best.rf)  
t\_rf\_all

## pred.best.rf  
## A B P R  
## A 267 0 0 0  
## B 0 263 3 9  
## P 0 4 273 2  
## R 0 4 0 266

accuracy.rf = (t\_rf\_all[1,1]+t\_rf\_all[2,2]+t\_rf\_all[3,3]+t\_rf\_all[4,4])/nrow(Letters.test.mod)  
accuracy.rf

## [1] 0.979835

For this random forest model, cross validation is employed to determine the ideal mtry value to use. This is achieved by setting a seed and repeatedly running the fandom forest model for every value of mtry from 1 to 16. The mtry value which produces the greatest accuracy is 2. The accuracy of the random forest model with mtry=2 applied to the test set is 0.979835.

### Queation 2b Vi

Boosting

set.seed(144)  
mod.boost <- gbm(letter.f ~ .,  
 data = Letters.train.mod2,  
 distribution = "multinomial",  
 n.trees = 3300,  
 interaction.depth = 10)  
  
set.seed(144)  
pred.boost <- predict(mod.boost, newdata = Letters.test.mod2, n.trees=3300, type = "response")  
  
pred\_fixed = apply(pred.boost, 1, which.max)   
pred = factor(pred\_fixed, levels = c(1,2,3,4), labels = c("A", "B", "P", "R"))  
  
t\_rf\_all = table(Letters.test.mod2$letter.f, pred)  
  
accuracy.boost = (t\_rf\_all[1,1]+t\_rf\_all[2,2]+t\_rf\_all[3,3]+t\_rf\_all[4,4])/nrow(Letters.test.mod)  
accuracy.boost

## [1] 0.9780018

The accuracy of the boosted model is 0.9780018.  
### Question 2b vii

accuracy\_LDA

## [1] 0.9230064

accuracy\_cart\_all

## [1] 0.9028414

accuracy\_bagging

## [1] 0.9587534

accuracy.rf

## [1] 0.979835

accuracy.boost

## [1] 0.9780018

The accuracy order of the models are, from least to most accurate, Cart model, LDA model, vanilla-bagged model, random forest model, boosting model. I would select the boosting model for this problem, because it is the most accurate. As in the case of the isB modeling, accuracy is more important than interpretability.

Note that the echo = FALSE parameter was added to the code chunk to prevent printing of the R code that generated the plot.