**Homework Assignment 4**

*(due through Blackboard before class on 5/1/2019)*

Things you should please do, lest you should lose my tender affection:

* Somehow incorporate your last name into the name of the solution file you upload.
* Make sure your name is inside the document itself, at the beginning or in the header.
* All **code must be text inside of the document** that I can copy and paste – do not include screenshots of code.
* Make sure that your work is original – **do not copy your code from or share your code with anybody else in class.**

Prathima Mateti:

**1. Tree Models (50 points)**

For the exercises that follow, you will be using the Auto data set within the ISLR library.

a. Create a factor variable called *highmpg* within the Auto data set. The variable should have the value TRUE for autos with a *mpg* of greater than 30 and a value of FALSE otherwise. Then, **remove *mpg*** from the data set, because it would be trivial to predict *highmpg* from it. **Show your code**.

library(tree)

install.packages("ISLR")

library(ISLR)

attach(Auto)

?Auto

#make a categorical variable

Highmpg=ifelse(mpg<=30,"FALSE","TRUE")

Auto=data.frame(Auto,Highmpg)

summary(Auto)

Auto=subset(Auto, select= -c(mpg))

summary(Auto)

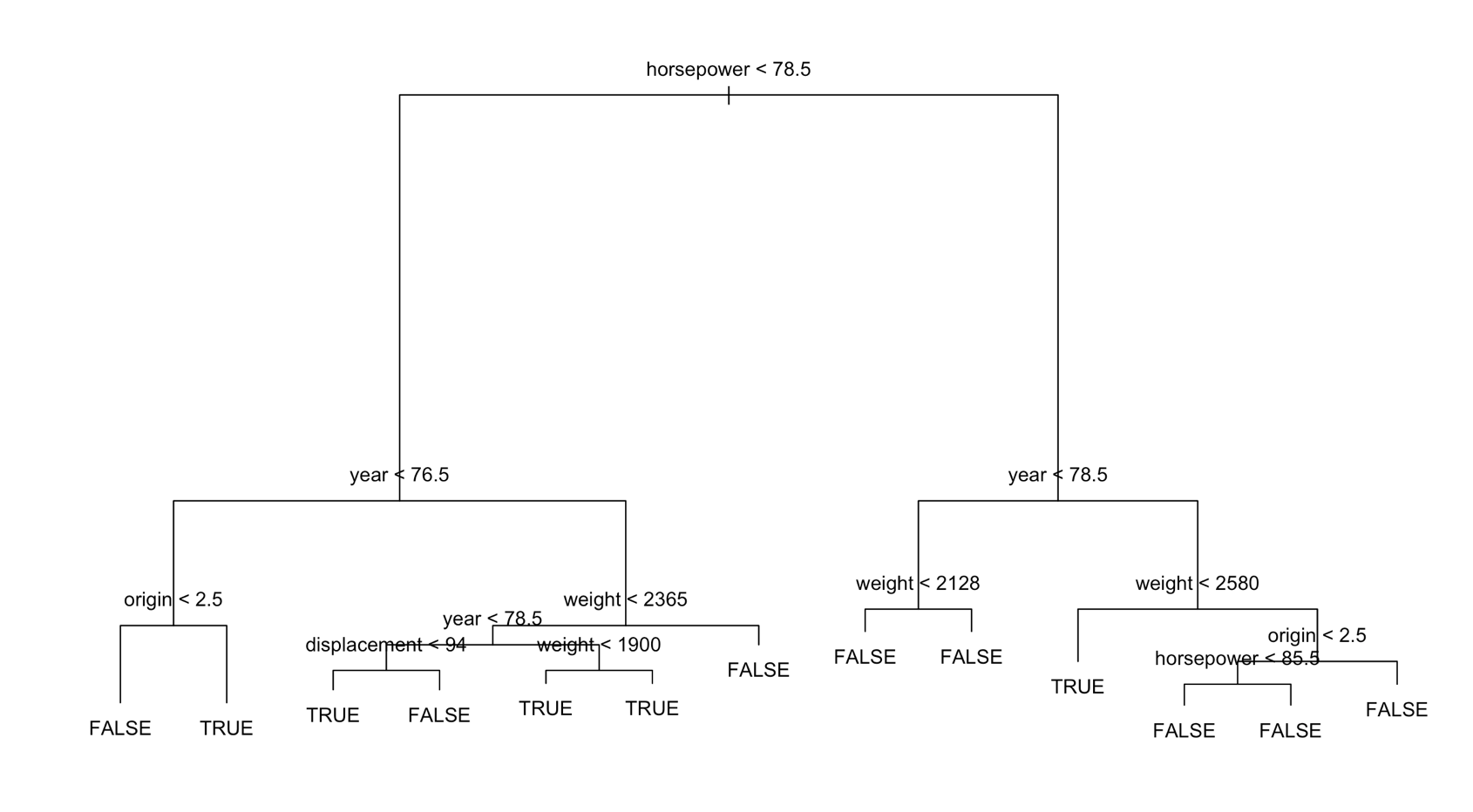
b. Build a decision tree model that can be used to predict whether an auto has a high mpg. The *name* variable can be safely excluded from the model as it has too many levels and won’t provide any predictive value, being unique to each row. **Show your code**, as well as a **plot** of the decision tree (with text), and also supply the **% of values that were misclassified** by the model.

tree.Auto=tree(Highmpg~.-name,Auto)

summary(tree.Auto)

plot(tree.Auto)

text(tree.Auto,pretty=0)



> summary(tree.Auto)

Classification tree:

tree(formula = Highmpg ~ . - name, data = Auto)

Variables actually used in tree construction:

[1] "horsepower" "year" "origin" "weight" "displacement"

Number of terminal nodes: 13

Residual mean deviance: 0.2311 = 87.59 / 379

Misclassification error rate: 0.05867 = 23 / 392

c. Based on the model you built, what 3-4 variables seem to be most important in determining whether an auto has a high mpg? **List the variables.**

These variables seem to be most important in determining whether an auto has a high mpg

horsepower

year

origin

weight

d. Set a seed using your 8-digitVillanova number. Break the Auto data into training and test data sets using random sampling. Let the training data set be 70% of the records and the test data set be the remaining 30%. Train a new model on the training data, and then test the predictions of the new model against your test data. **Show your code** (including your Villanova number), as well as a table that **compares the predicted test values with the actuals**. State what **% of the test data was misclassified**.

set.seed(02080075)

train=sample(1:nrow(Auto), nrow(Auto)\*.7)

Auto.test=Auto[-train,]

Highmpg.test=Highmpg[-train]

tree.Auto=tree(Highmpg~.-name,Auto,subset=train)

tree.pred=predict(tree.Auto,Auto.test,type="class")

table(tree.pred,Highmpg.test)

> set.seed(02080075)

> train=sample(1:nrow(Auto), nrow(Auto)\*.7)

> Auto.test=Auto[-train,]

> Highmpg.test=Highmpg[-train]

> tree.Auto=tree(Highmpg~.-name,Auto,subset=train)

> tree.pred=predict(tree.Auto,Auto.test,type="class")

> table(tree.pred,Highmpg.test)

Highmpg.test

tree.pred FALSE TRUE

FALSE 89 4

TRUE 7 18

% of the test data was misclassified = ((7+4)/(89+4+7+18))\*100=9.322

e. Determine whether your tree should be pruned. Use cross-validation to determine the ideal size of your tree. **Show your code**, along with the **output** that tells you how large your tree should be. Add a few sentences that **explain how large your tree should ideally be and how you know** that this is the best size.

Yes, my tree should be pruned.

cv.Auto=cv.tree(tree.Auto,FUN=prune.misclass)

names(cv.Auto)

cv.Auto

> cv.Auto=cv.tree(tree.Auto,FUN=prune.misclass)

> names(cv.Auto)

[1] "size" "dev" "k" "method"

> cv.Auto

$size

[1] 13 7 5 4 3 1

$dev

[1] 24 24 23 35 32 59

$k

[1] -Inf 0.0 1.0 4.0 7.0 16.5

$method

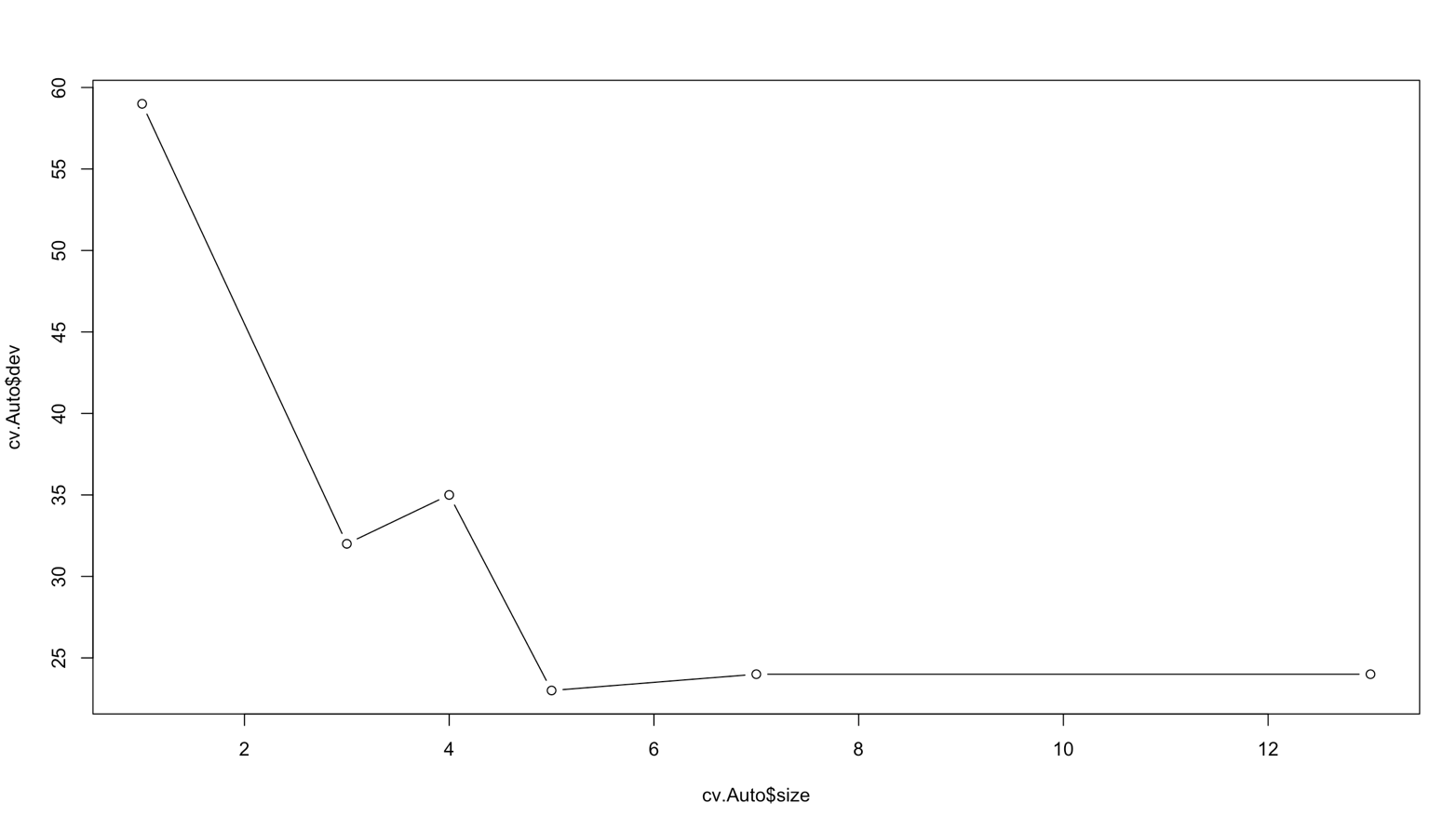
[1] "misclass"

attr(,"class")

[1] "prune" "tree.sequence"

From the cross-validation the ideal size of the tree should be 13. As the size decreases the misclassification rate is increasing. Hence, the best size of the tree is 13 where the misclassification rate is 9.322%

plot(cv.Auto$size,cv.Auto$dev,type="b")



f. Create a pruned version of your tree based on the ideal size you determined in the previous step. Use this pruned version to make predictions for your test data set. **Show your code,** as well as a table that **compares the predicted test values with the actuals**. State what **% of the test data was misclassified**. Explain **whether or not the pruning improved** your predictive accuracy.

Pruning based on the ideal size

prune.Auto=prune.misclass(tree.Auto,best=13)

plot(prune.Auto)

text(prune.Auto,pretty=0)

tree.pred=predict(prune.Auto,Auto.test,type="class")

table(tree.pred,Highmpg.test)

mean(tree.pred != Highmpg.test)

> prune.Auto=prune.misclass(tree.Auto,best=13)

> text(prune.Auto,pretty=0)

> plot(prune.Auto)

> text(prune.Auto,pretty=0)

> tree.pred=predict(prune.Auto,Auto.test,type="class")

> table(tree.pred,Highmpg.test)

Highmpg.test

tree.pred FALSE TRUE

FALSE 89 4

TRUE 7 18

> mean(tree.pred != Highmpg.test)

[1] 0.09322034

With the ideal size 13, % of the test data misclassified: 0.09322034 \* 100 = 9.322

prune.Auto=prune.misclass(tree.Auto,best=7)

plot(prune.Auto)

text(prune.Auto,pretty=0)

tree.pred=predict(prune.Auto,Auto.test,type="class")

table(tree.pred,Highmpg.test)

mean(tree.pred != Highmpg.test)

> prune.Auto=prune.misclass(tree.Auto,best=7)

> plot(prune.Auto)

> text(prune.Auto,pretty=0)

> tree.pred=predict(prune.Auto,Auto.test,type="class")

> table(tree.pred,Highmpg.test)

Highmpg.test

tree.pred FALSE TRUE

FALSE 87 3

TRUE 9 19

> mean(tree.pred != Highmpg.test)

[1] 0.1016949

>

With size 7, % of the test data misclassified: 0.1016949\*100=10.169

Therefore from the above misclassification rate values, we can say that the pruning improved the predictive accuracy.

g. Try to improve your results using a random forest. **Show your code**, your **OOB error rate,** and what **% of the test data was misclassified**. Note again that *name* should not be part of your analysis.

install.packages("randomForest")

library(randomForest) #can use for bagging too

?Auto

summary(Auto)

Auto$Highmpg = as.factor(Auto$Highmpg)

Auto=Auto[,-8]

bag.Auto=randomForest(Highmpg~.,data=Auto,subset=train,importance=TRUE)

bag.Auto

> bag.Auto=randomForest(Highmpg~.,data=Auto,subset=train,importance=TRUE)

> bag.Auto

Call:

randomForest(formula = Highmpg ~ ., data = Auto, importance = TRUE, subset = train)

Type of random forest: classification

Number of trees: 500

No. of variables tried at each split: 2

OOB estimate of error rate: 7.3%

Confusion matrix:

FALSE TRUE class.error

FALSE 204 9 0.04225352

TRUE 11 50 0.18032787

>

OOB estimate of error rate: 7.3%

Misclassification rate=((11+9)/(204+9+11+50))\*100= 7.29 %

h. What variables ended up being most important in the approach you used in the previous step? **Show the code and the output** that tell you this? How does this list **compare** to the list you got in step 1.c?

importance(bag.Auto)

> importance(bag.Auto)

FALSE TRUE MeanDecreaseAccuracy MeanDecreaseGini

cylinders 2.026815 7.477705 7.821785 3.487120

displacement 8.772762 19.829010 19.623609 16.409496

horsepower 8.474623 22.779070 21.380756 19.669769

weight 8.523383 21.297573 21.647776 20.564838

acceleration 2.252279 7.864582 7.120090 6.460805

year 19.246266 35.750158 35.443885 21.264190

origin 4.681302 16.868698 15.533296 5.416104

MeanDecreaseGini is higher for year so year is the most important variable in this approach.

varImpPlot(bag.Auto)

Auto.bag.pred=predict(bag.Auto,Auto.test,type="class")

table(Auto.bag.pred,Auto.test$Highmpg)

mean(Auto.bag.pred != Auto.test$Highmpg)

> varImpPlot(bag.Auto)

> Auto.bag.pred=predict(bag.Auto,Auto.test,type="class")

> table(Auto.bag.pred,Auto.test$Highmpg)

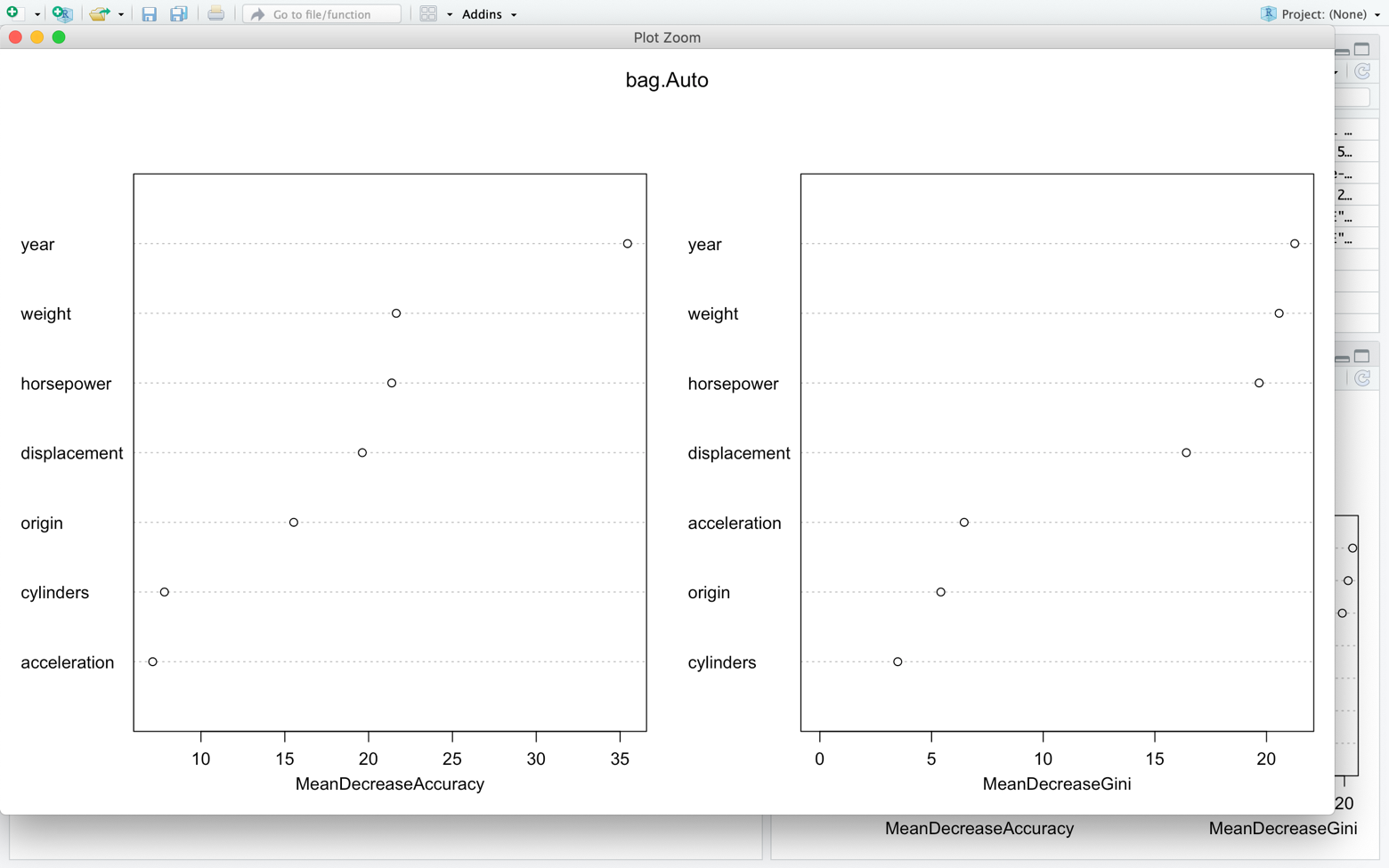
Auto.bag.pred FALSE TRUE

FALSE 90 5

TRUE 6 17

> mean(Auto.bag.pred != Auto.test$Highmpg)

[1] 0.09322034



**2. Bayes’ Theorem (10 points)**

In XYZ Company, 5% of laptops fail within their warranty period. 8% of the people in the company work in the Sales department. However, the people from Sales travel often, and so are rougher on their laptops than the average employee. 15% of the laptops that fail within their warranty period belonged to Sales employees.

John Smith has just joined the Sales department and gotten a laptop. He asks you how likely it is that his laptop will fail within the warranty period. What do you tell him?

Use Bayes’ Theorem to answer John. **State** your answer, and **show the calculations** you used to derive it.

We have

% of laptops fail within their warranty period= P(W)=5

% of the people in the company work in the Sales department=P(S)=8

% of the laptops that fail within their warranty period belonged to Sales employees=P(W/S)=15

Using Bayes’ theorem

P(S/W)=(P(W/S)\*P(S))/P(W)

= (15\*8)/5

=24

Therefore, John Smith being from Sales department, the probability that his laptop that will fail within the warranty period is 24%

**3. Twitter Predictions (40 points)**

I’d like you to predict whether a tweet is more likely to be sent during the day or at night. Download the “nova\_tweets.csv” file from Blackboard. It represents 5000 tweets about Villanova from April of 2018. The tweet\_txt column contains the text of the tweet. The tod column stands for “time of day” and tells you the hour of the day in which the tweet was sent in 24-hour format (values 0-23).

For each tweet, please categorize it as a night tweet if it is sent between 8 PM (20:00) and 5:59 AM (05:59). Note that only the hour is represented in the tod column, so your night tweets have values 20-23 and 0-5 in the tod column. Otherwise, it is a day tweet.

Break your data into test (20%) and training (80%) sets using random sampling (check that the proportion of day and night tweets is roughly similar between the two sets after you create them), and then use your training set to build a model that predicts day or night based on the words in the tweets. Apply that model to your test data set to predict whether the tweets in the test set were sent during the day or during the night.

**Show all of your code**, and **answer the following question**:

* Were the predictions made by your model more accurate than just guessing the dominant class? That is, if you had just guessed “Day” or “Night” (whichever is more prominent in the data) for all examples, would you have gotten a higher percentage correct than trying to predict based on the words in the tweets?

Note: There will be many ways to tackle this problem. The code I gave you in the slides should be enough to get it done, but feel free to use other methods as well. Just be sure to use the naiveBayes() function to build your model.

tweets\_raw = read.csv("C:\\Data Mining softwares\\nova\_tweets.csv")

str(tweets\_raw)

tweetstime=ifelse(tweets\_raw$tod>5&tweets\_raw$tod<20,"DAY","NIGHT")

tweets\_raw=data.frame(tweets\_raw,tweetstime)

tweets\_raw$tweetstime <- factor(tweets\_raw$tweetstime)

str(tweets\_raw$tweetstime)

table(tweets\_raw$tweetstime)

install.packages("tm")

library(tm)

tweets\_corpus<- VCorpus(VectorSource(tweets\_raw$tweet\_txt))

print(tweets\_corpus)

inspect(tweets\_corpus[1:2])

as.character(tweets\_corpus[[1]])

lapply(tweets\_corpus[1:2], as.character)

tweets\_corpus\_clean <- tm\_map(tweets\_corpus, content\_transformer(tolower))

as.character(tweets\_corpus[[1]])

as.character(tweets\_corpus\_clean[[1]])

tweets\_corpus\_clean <- tm\_map(tweets\_corpus\_clean, removeNumbers)

tweets\_corpus\_clean <- tm\_map(tweets\_corpus\_clean, removeWords, stopwords())

tweets\_corpus\_clean <- tm\_map(tweets\_corpus\_clean, removePunctuation)

install.packages("SnowballC")

library(SnowballC)

tweets\_corpus\_clean <- tm\_map(tweets\_corpus\_clean, stemDocument)

tweets\_corpus\_clean <- tm\_map(tweets\_corpus\_clean, stripWhitespace)

lapply(tweets\_corpus[1:3], as.character)

lapply(tweets\_corpus\_clean[1:3], as.character)

tweets\_dtm <- DocumentTermMatrix(tweets\_corpus\_clean)

tweets\_dtm

tweets\_dtm\_train <- tweets\_dtm[1:4000, ]

tweets\_dtm\_test  <- tweets\_dtm[4001:5000, ]

tweets\_train\_labels <- tweets\_raw[1:4000, ]$tweetstime

tweets\_test\_labels  <- tweets\_raw[4001:5000, ]$tweetstime

prop.table(table(tweets\_train\_labels))

prop.table(table(tweets\_test\_labels))

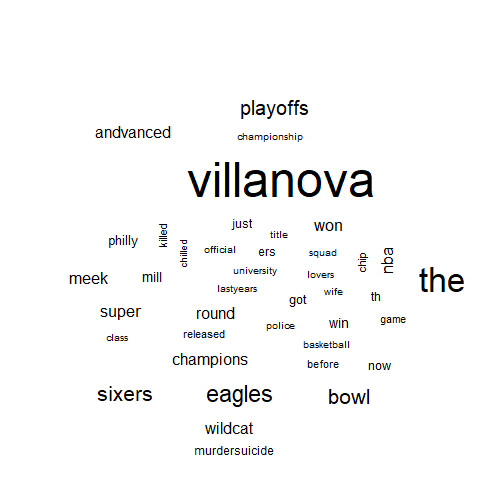
DAY <- subset(tweets\_raw, tweetstime == "DAY")

NIGHT <- subset(tweets\_raw, tweetstime == "NIGHT")

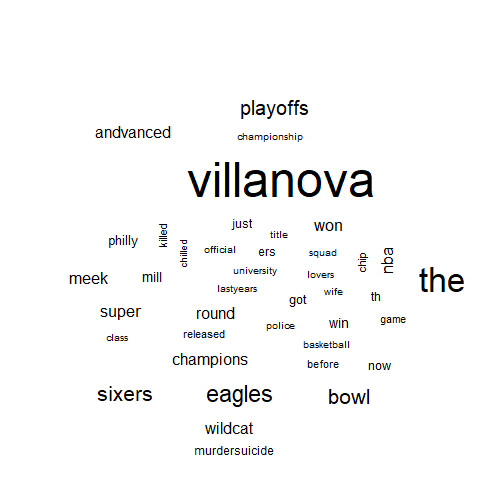
install.packages("wordcloud")

library(wordcloud)

wordcloud(DAY$tweet\_txt, max.words = 40, scale = c(3, 0.5))



wordcloud(NIGHT$tweet\_txt, max.words = 40, scale = c(3, 0.5))



findFreqTerms(tweets\_dtm\_train, 5)

tweets\_freq\_words <- findFreqTerms(tweets\_dtm\_train, 5)

str(tweets\_freq\_words)

tweets\_dtm\_freq\_train <- tweets\_dtm\_train[ , tweets\_freq\_words]

tweets\_dtm\_freq\_test <- tweets\_dtm\_test[ , tweets\_freq\_words]

sink("dtm\_train\_sample.txt")

inspect(tweets\_dtm\_freq\_train)

sink()

convert\_counts <- function(x) {

  x <- ifelse(x > 0, "Yes", "No")

}

tweets\_train <- apply(tweets\_dtm\_freq\_train, MARGIN = 2, convert\_counts)

tweets\_test  <- apply(tweets\_dtm\_freq\_test, MARGIN = 2, convert\_counts)

install.packages("e1071")

library(e1071)

tweets\_classifier <- naiveBayes(tweets\_train, tweets\_train\_labels)

tweets\_test\_pred <- predict(tweets\_classifier, tweets\_test)

install.packages("gmodels")

library(gmodels)

CrossTable(tweets\_test\_pred, tweets\_test\_labels,

           prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE,

           dnn = c('predicted', 'actual'))

Cell Contents

|-------------------------|

|                       N |

|-------------------------|

Total Observations in Table:  1000

                 | tweets\_test\_labels

tweets\_test\_pred |     NIGHT | Row Total |

-----------------|-----------|-----------|

             DAY |       485 |       485 |

-----------------|-----------|-----------|

           NIGHT |       515 |       515 |

-----------------|-----------|-----------|

    Column Total |      1000 |      1000 |

-----------------|-----------|-----------|

Predictions made by the model are more accurate. i.e., the tweets during Nights are more which is predicted accurately by the model We got a higher percentage correct than trying to predict based on the words in the tweets.