**Homework Assignment 4**

*(due through Blackboard on 5/1/2020)*

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.
* Show the code for each step that requires code.
* Make sure that your work is original – **do not copy your code from or share your code with anybody else in class.**

**1. Car Shopping for Fuzzy (45 points)**

This is your friend Fuzzy.

****

Fuzzy is shopping for a new car. Fuzzy is having a hard time finding a car, and since Fuzzy is so busy, you have agreed to help identify cars that may be of interest. Thankfully, Fuzzy has been keeping records of cars that Fuzzy previously investigated, and whether those were deemed acceptable or not.

a. You will be working with the car\_data.csv file in Blackboard. Download that file to your local filesystem. Load it into a data frame called *fuzzycars* using the read.csv2 function, which is meant for files that use a semicolon as a column separator.

**fuzzycars <- read.csv2(file = 'D:/Data Mining & Data Base Systems/R/car\_data.csv')**

b. Break the *fuzzycars* data into two sets, a training data set called *cars.train* and a test data set called *cars.test*. The training data set should contain 80% of the records from *fuzzycars*, and the test data set should contain the remaining 20%. You should split the two data sets using a random sample, and you should use your own unique Villanova Id number to set the seed before taking your sample.

c. Build a decision tree model from your training data set. Your model should predict the value of *acceptability* on the basis of all the other variables. Call your model whatever you like.

**tree.cars.train = tree(acceptability ~ ., cars.train)**

d. Run the summary() function on your model and show the output here.

**summary(tree.cars.train)**

**Classification tree:**

**tree(formula = acceptability ~ ., data = cars.train)**

**Variables actually used in tree construction:**

**[1] "safety\_rating" "num\_persons" "purchase\_price"**

**[4] "trunk\_size" "maint\_cost"**

**Number of terminal nodes: 10**

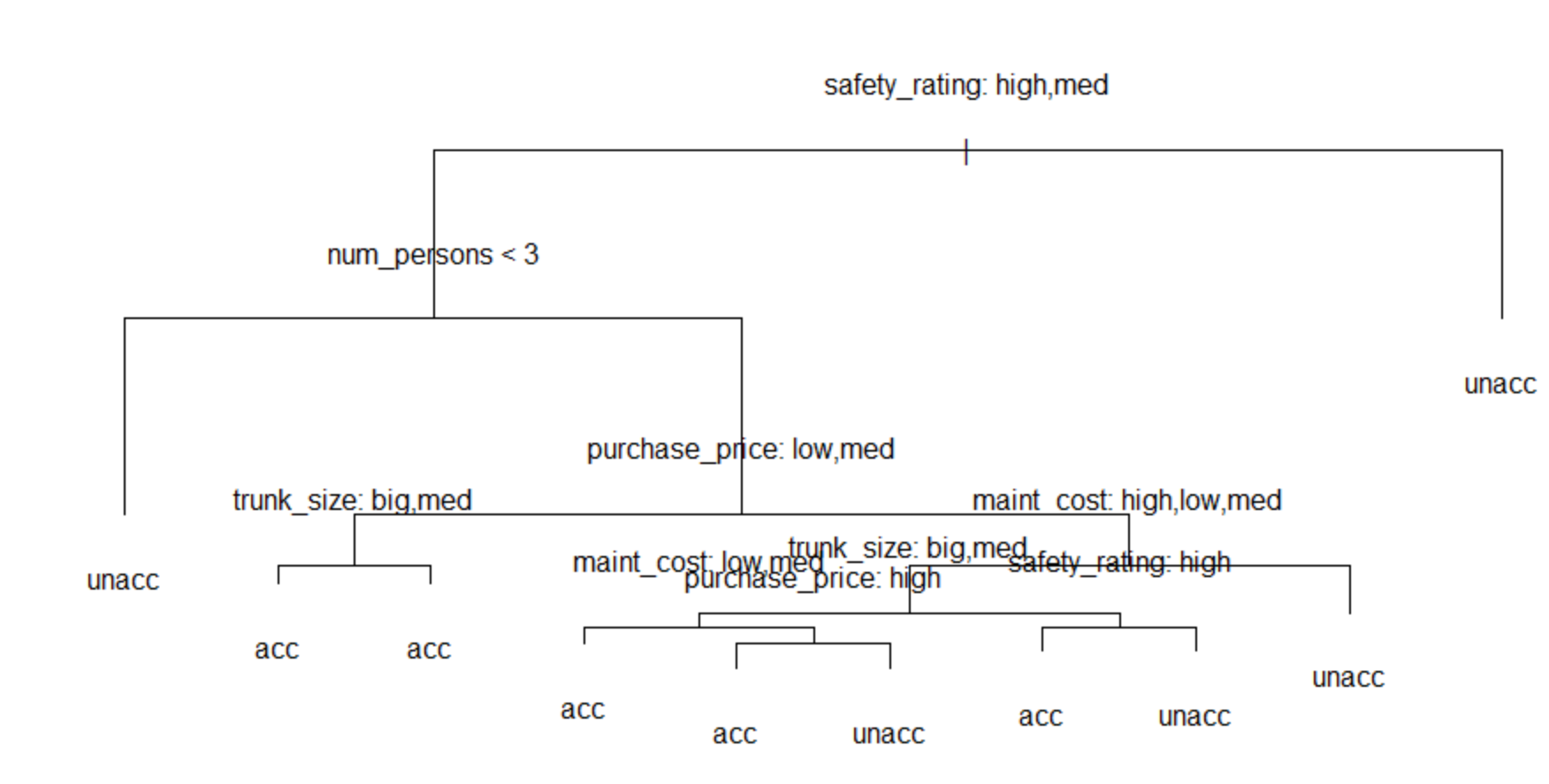
**Residual mean deviance: 0.2127 = 291.9 / 1372**

**Misclassification error rate: 0.0398 = 55 / 1382**

e. Plot your tree on a graph and add text to it. Don’t forget the “pretty” flag when you add your text.

**plot(tree.cars.train)**

**text(tree.cars.train, pretty = 0)**



f. Now, generate a set of predictions for *acceptability* on your test data set, and create a confusion matrix to show the performance of your model. How did your model score for accuracy, precision, and recall?

**cars.pred = predict(tree.cars.train, cars.test, type="class")**

**confusionMatrix(cars.pred, cars.test$acceptability, mode = "prec\_recall" )**

**Confusion Matrix and Statistics**

**Reference**

**Prediction acc unacc**

**acc 98 19**

**unacc 0 229**

**Accuracy : 0.9451**

**95% CI : (0.9156, 0.9666)**

**No Information Rate : 0.7168**

**P-Value [Acc > NIR] : < 2.2e-16**

**Kappa : 0.8722**

**Mcnemar's Test P-Value : 3.636e-05**

**Precision : 0.8376**

**Recall : 1.0000**

**F1 : 0.9116**

**Prevalence : 0.2832**

**Detection Rate : 0.2832**

**Detection Prevalence : 0.3382**

**Balanced Accuracy : 0.9617**

**'Positive' Class : acc**

**The accuracy of the model is 94.51%.**

**The precision is 0.8376 and recall is 1.00**

g. How does the misclassification rate on your test data set compare to the misclassification rate you saw when you ran the summary() function on your model? Do the respective misclassification rates make sense? Explain.

**Misclassification rate by running the summary() function on the model is** **0.0398 = 55 / 1382**

**Test data**

str(cars.test)

'data.frame': 346 obs. of 7 variables:

$ purchase\_price: Factor w/ 4 levels "high","low","med",..: 4 4 4 4 4 4 4 4 4 4 ...

$ maint\_cost : Factor w/ 4 levels "high","low","med",..: 4 4 4 4 4 4 4 4 4 4 ...

$ num\_doors : int 2 2 2 2 2 3 3 3 3 3 ...

$ num\_persons : int 2 2 4 4 4 2 2 2 4 4 ...

$ trunk\_size : Factor w/ 3 levels "big","med","small": 2 2 3 2 1 2 2 1 3 3 ...

$ safety\_rating : Factor w/ 3 levels "high","low","med": 2 3 2 3 1 2 1 1 3 1 ...

$ acceptability : Factor w/ 2 levels "acc","unacc": 2 2 2 2 2 2 2 2 2 2 ...

**Reference**

**Prediction acc unacc**

**acc 98 19**

**unacc 0 229**

**Misclassification rate on test data is 19/346 = 0.054**

h. Your other friend, Stinky, says that he has a great car that Fuzzy will love. The car has the following characteristics:

purchase\_price: high

maint\_cost: high

num\_doors: 2

num\_persons: 4

trunk\_size: small

safety\_rating: high

Using your decision tree model, predict whether Stinky’s car will be acceptable to Fuzzy.

**stinky\_car <- data.frame(purchase\_price= "high",**

**maint\_cost= "high",**

**num\_doors= c(2),**

**num\_persons= c(4),**

**trunk\_size= "small",**

**safety\_rating= "high"**

**)**

**predict(tree.cars.train, stinky\_car, type="class")**

**[1] acc**

**Levels: acc unacc**

**Stinky’s car is acceptable to Fuzzy**

i. Try to improve your test set results using a random forest. Show your code, your OOB error rate**,** and your accuracy, precision, and recall

**install.packages("randomForest")**

**library(randomForest)**

**set.seed(02137024)**

**rf.cars = randomForest(acceptability ~ .,data = cars.test, importance = TRUE)**

**rf.cars**

**Call:**

**randomForest(formula = acceptability ~ ., data = cars.test, importance = TRUE)**

**Type of random forest: classification**

**Number of trees: 500**

**No. of variables tried at each split: 2**

**OOB estimate of error rate: 3.76%**

**Confusion matrix:**

**acc unacc class.error**

**acc 90 8 0.08163265**

**unacc 5 243 0.02016129**

**OOB Rate is 3.76%**

**cars.rf\_pred = predict(rf.cars, cars.test, type="class")**

**confusionMatrix(cars.rf\_pred, cars.test$acceptability, mode = "prec\_recall" )**

**Confusion Matrix and Statistics**

**Reference**

**Prediction acc unacc**

**acc 98 0**

**unacc 0 248**

**Accuracy : 1**

**95% CI : (0.9894, 1)**

**No Information Rate : 0.7168**

**P-Value [Acc > NIR] : < 2.2e-16**

**Kappa : 1**

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

**Precision : 1.0000**

**Recall : 1.0000**

**F1 : 1.0000**

**Prevalence : 0.2832**

**Detection Rate : 0.2832**

**Detection Prevalence : 0.2832**

**Balanced Accuracy : 1.0000**

**'Positive' Class : acc**

**Accuracy is 1**

**Precision is 1**

**Recall is 1**

j. 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.

**Safety rating and num\_persons are the important variables**

**importance(rf.cars)**

acc unacc MeanDecreaseAccuracy MeanDecreaseGini

purchase\_price 26.853075 26.578648 34.72783 19.033430

maint\_cost 15.964747 20.552284 23.73623 13.511116

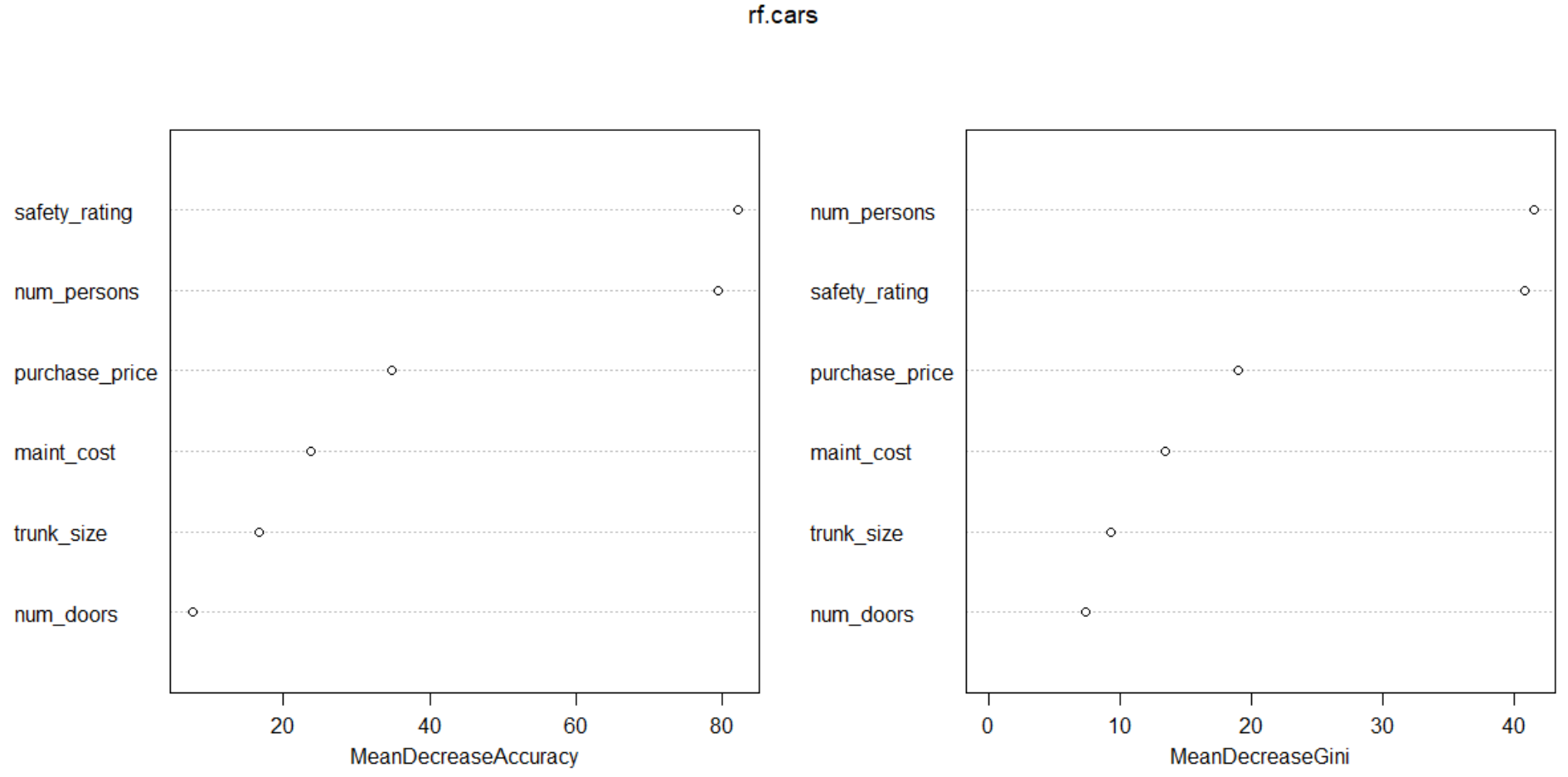
num\_doors 6.391681 5.105495 7.59226 7.350737

**num\_persons 72.928820 58.247007 79.40131 41.532989**

trunk\_size 12.755014 13.044804 16.73043 9.348449

**safety\_rating 73.646210 59.791576 82.07111 40.801434**

**varImpPlot(rf.cars)**



**2. Bayes’ Theorem (5 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.

**The chance of smith laptop to fail within the warranty period is 9.4%.**

**P(H|E) = ( p(E|H) \* p(H) ) / p(E)**

**= (0.15 \* 0.05) / 0.08**

**= 0.094**

**3. Gini in the Bottle (5 points)**

What is the gini impurity of the contents of this bottle? Show your calculations.



**Gini Impurity = 1 – ((7/9)^2 + (2/9)^2)**

**= 1 – ((0.78)^2 + (0.22)^2)**

**= 1 – (0.61 + 0.048)**

**= 1 – 0.658**

**= 0.342**

**4. Twitter Predictions (45 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 your code, and answer the following questions:

**Code:**

**tweets <- read.csv("D:/Data Mining & Data Base Systems/R/nova\_tweets.csv")**

**#install.packages("dplyr")**

**library(dplyr)**

**tweets = mutate(tweets, tod = ifelse(tod %in% 20:23, "Night",**

**ifelse(tod %in% 0:5, "Night", "Day")))**

**tweets$tod <- factor(tweets$tod)**

**str(tweets$tod)**

**#install.packages("tm")**

**library(tm)**

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

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

**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)**

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

**train\_pct <- .8**

**set.seed(100)**

**train = sample(1:nrow(tweets),train\_pct \* nrow(tweets))**

**tweets\_dtm\_train <- tweets\_dtm[train, ]**

**tweets\_dtm\_test <- tweets\_dtm[-train, ]**

**tweets\_train\_labels <- tweets[train, ]$tod**

**tweets\_test\_labels <- tweets[-train, ]$tod**

**day <- subset(tweets, tod == "Day")**

**night <- subset(tweets, tod == "Night")**

**tweets\_freq\_words <- findFreqTerms(tweets\_dtm\_train, 7)**

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

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

**convert\_counts <- function(x) {**

**return(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("caret")**

**library(caret)**

**confusionMatrix(as.factor(tweets\_test\_pred), tweets\_test\_labels, positive = "Day",**

**mode = "prec\_recall")**

* 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?

**Yes predictions made by the model are accurate than guessing the dominant class because the percentage of day tweets by guessing is 54.5% but the percentage of day tweets predicted by the model is 66.8%.**

* What was the precision and recall of your model?

**Confusion Matrix and Statistics**

**Reference**

**Prediction Day Night**

**Day 361 148**

**Night 184 307**

**Accuracy : 0.668**

**95% CI : (0.6378, 0.6972)**

**No Information Rate : 0.545**

**P-Value [Acc > NIR] : 1.659e-15**

**Kappa : 0.3349**

**Mcnemar's Test P-Value : 0.05475**

**Precision : 0.7092**

**Recall : 0.6624**

**F1 : 0.6850**

**Prevalence : 0.5450**

**Detection Rate : 0.3610**

**Detection Prevalence : 0.5090**

**Balanced Accuracy : 0.6686**

**'Positive' Class : Day**

**Precision: 0.7092**

**Recall: 0.6624**

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.