breastCancer3.r

setwd("C:/Users/Tony/Dropbox/Rowan/DM2/Lecture1/WisconsinBreastCancer")  
f=read.csv(file="wdbc\_data\_abclean.csv",header=F,stringsAsFactors = TRUE)  
  
dim(f)

## [1] 569 32

f[1,]

## V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11  
## 1 842302 1 17.99 10.38 122.8 1001 0.1184 0.2776 0.3001 0.1471 0.2419  
## V12 V13 V14 V15 V16 V17 V18 V19 V20  
## 1 0.07871 1.095 0.9053 8.589 153.4 0.006399 0.04904 0.05373 0.01587  
## V21 V22 V23 V24 V25 V26 V27 V28 V29 V30  
## 1 0.03003 0.006193 25.38 17.33 184.6 2019 0.1622 0.6656 0.7119 0.2654  
## V31 V32  
## 1 0.4601 0.1189

## We don't need the first column, it's just an id  
ff <-subset(f,select=-V1)  
dim(ff)

## [1] 569 31

library(plyr)  
count(ff,c("V2"))

## V2 freq  
## 1 0 357  
## 2 1 212

## 357 benign, 212 malignant  
  
  
set.seed(2)  
train=sample(1:nrow(ff),nrow(ff)\*(8/10))  
test=-train  
  
trainingData=ff[train,]  
testingData=ff[test,]  
dim(trainingData)

## [1] 455 31

dim(testingData)

## [1] 114 31

## We make V2 a factor because if it's a factor the randomforest call  
## solves as classification. Otherwise it attempts a regression  
## in other words it will return a 0 or 1 in this case rather than a range between 0 and 1  
formula = as.factor(V2)~V3+V4+V5+V6+V7+V8+V9+V10+V11+V12+V13+V14+V15+V16+V17+V18+V19+V20+V21+V22+V23+V24+V25+V26+V27+V28+V29+V30+V31+V32  
  
##install.packages("randomForest")  
library(randomForest)

## randomForest 4.6-12

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

fit <- randomForest(formula,  
 data=trainingData,   
 importance=TRUE,   
 ntree=2000)  
  
  
?randomForest

## starting httpd help server ...

## done

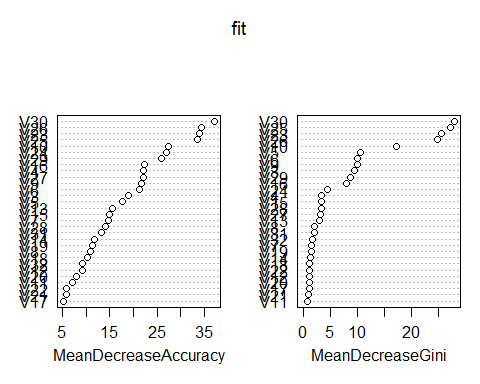
varImpPlot(fit)  
  
##install.packages("caret")  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

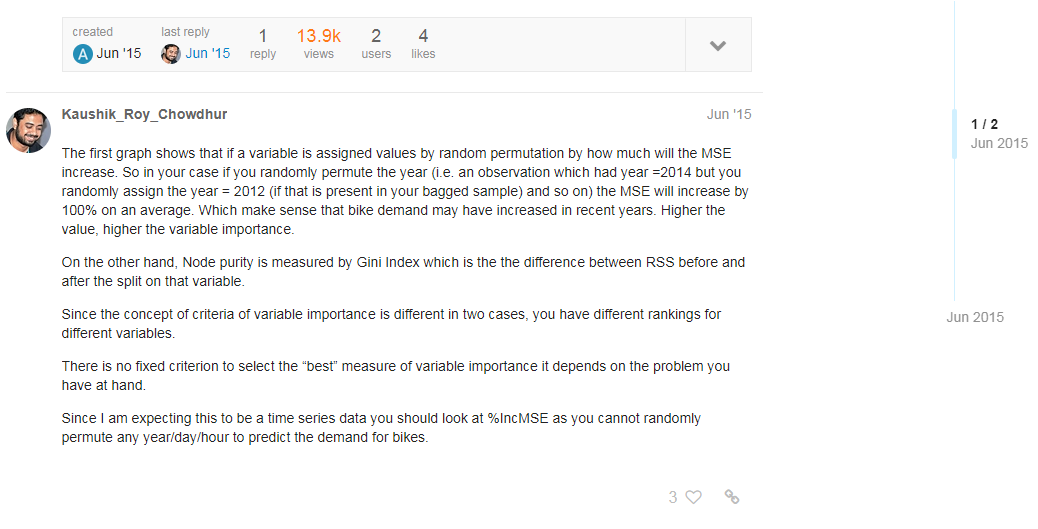
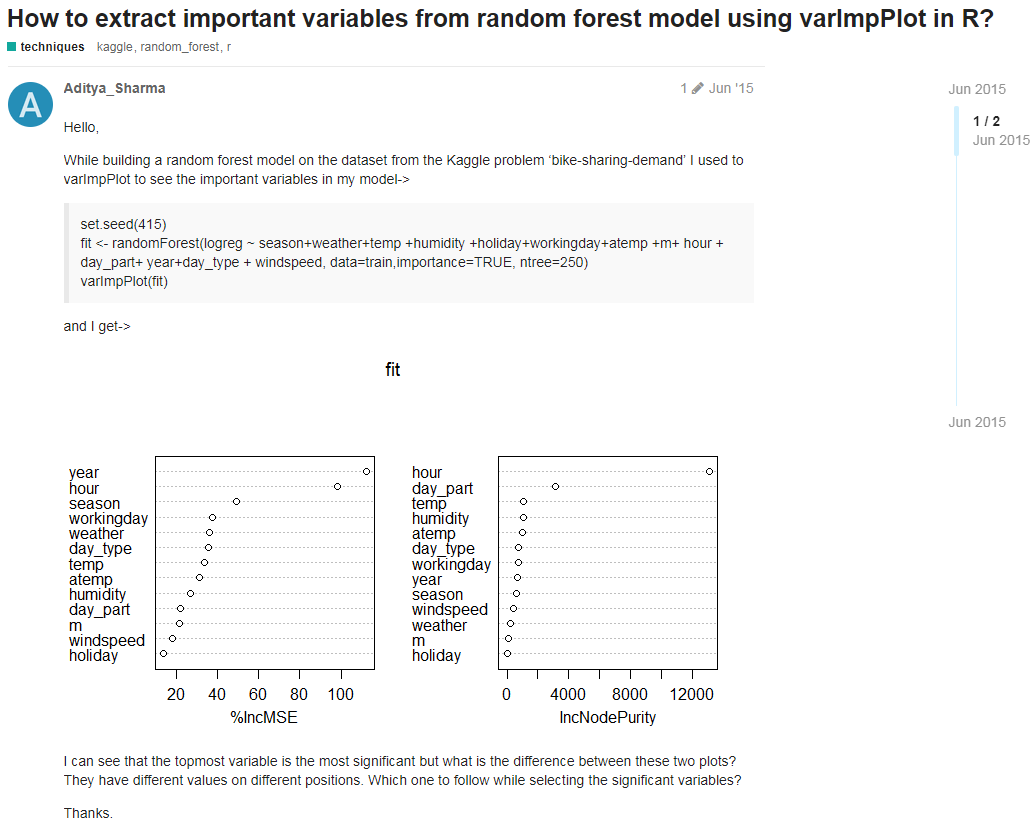
##   
## Attaching package: 'ggplot2'

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



varImp(fit)

## 0 1  
## V3 13.821789 13.821789  
## V4 16.388836 16.388836  
## V5 12.927214 12.927214  
## V6 15.432730 15.432730  
## V7 9.711773 9.711773  
## V8 7.325517 7.325517  
## V9 15.241131 15.241131  
## V10 19.709392 19.709392  
## V11 5.025004 5.025004  
## V12 5.782743 5.782743  
## V13 11.167927 11.167927  
## V14 8.827998 8.827998  
## V15 10.895317 10.895317  
## V16 16.216383 16.216383  
## V17 3.764316 3.764316  
## V18 5.715620 5.715620  
## V19 8.198247 8.198247  
## V20 5.162944 5.162944  
## V21 3.804464 3.804464  
## V22 3.720051 3.720051  
## V23 25.177445 25.177445  
## V24 21.228930 21.228930  
## V25 25.488600 25.488600  
## V26 26.012613 26.012613  
## V27 16.665759 16.665759  
## V28 10.068708 10.068708  
## V29 18.014252 18.014252  
## V30 27.262956 27.262956  
## V31 9.225810 9.225810  
## V32 6.662864 6.662864



## Both measures suggest V30 and V26 are the most important variables

testingData$result <- predict(fit, testingData)  
count(testingData,c("V2","result"))

## V2 result freq  
## 1 0 0 77  
## 2 0 1 1  
## 3 1 0 2  
## 4 1 1 34

err = 3/(77+3+34)  
err

## [1] 0.02631579

## 2 false negatives, 3 errors overall

## Let's compare this to a single tree using   
library(rpart)  
treeFit <- rpart(formula,  
 data=trainingData,method="class")  
  
testingData$result2 <- predict(treeFit, testingData,type="class")  
  
  
count(testingData,c("V2","result2"))

## V2 result2 freq  
## 1 0 0 76  
## 2 0 1 2  
## 3 1 0 2  
## 4 1 1 34

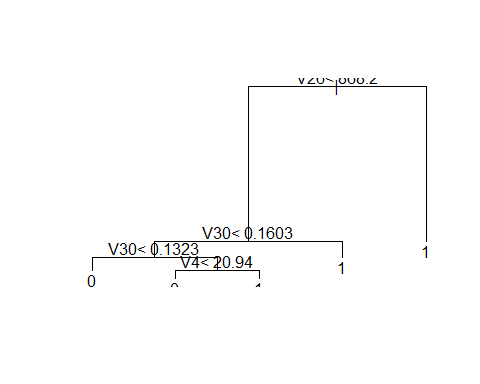
err = 4/(76+4+34)  
err

## [1] 0.03508772

## One more error than forest  
  
  
count(testingData,c("V2","result","result2"))

## V2 result result2 freq  
## 1 0 0 0 76  
## 2 0 0 1 1  
## 3 0 1 1 1  
## 4 1 0 0 2  
## 5 1 1 1 34

## single tree gives one more false positive than forest  
  
plot(treeFit); text(treeFit)



imp<-varImp(treeFit)  
imp$rn<-rownames(imp)  
imp[order(-imp$Overall),]

## Overall rn  
## V30 175.350310 V30  
## V10 156.039399 V10  
## V26 151.469955 V26  
## V23 151.112247 V23  
## V25 149.803220 V25  
## V29 17.861797 V29  
## V28 16.855686 V28  
## V9 15.526844 V9  
## V24 11.445008 V24  
## V4 8.400000 V4  
## V14 3.000000 V14  
## V17 2.700000 V17  
## V18 2.526316 V18  
## V3 0.000000 V3  
## V5 0.000000 V5  
## V6 0.000000 V6  
## V7 0.000000 V7  
## V8 0.000000 V8  
## V11 0.000000 V11  
## V12 0.000000 V12  
## V13 0.000000 V13  
## V15 0.000000 V15  
## V16 0.000000 V16  
## V19 0.000000 V19  
## V20 0.000000 V20  
## V21 0.000000 V21  
## V22 0.000000 V22  
## V27 0.000000 V27  
## V31 0.000000 V31  
## V32 0.000000 V32

## Let's actually see if we can get the same accuracy just using V26, V30, and V4  
formula

## as.factor(V2) ~ V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10 + V11 +   
## V12 + V13 + V14 + V15 + V16 + V17 + V18 + V19 + V20 + V21 +   
## V22 + V23 + V24 + V25 + V26 + V27 + V28 + V29 + V30 + V31 +   
## V32

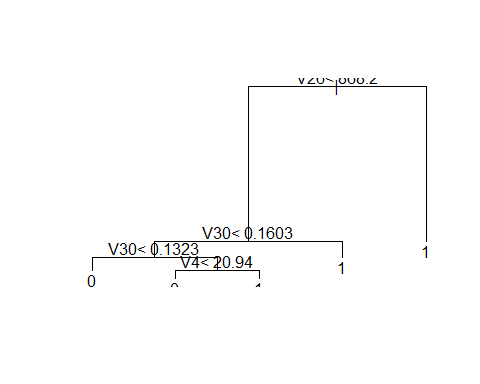
formula <- as.factor(v2)~V26+V30+V4  
treeFit <- rpart(formula,  
 data=trainingData,method="class")  
  
testingData$result2 <- predict(treeFit, testingData,type="class")  
count(testingData,c("V2","result2"))

## V2 result2 freq  
## 1 0 0 76  
## 2 0 1 2  
## 3 1 0 2  
## 4 1 1 34

err = 4/(76+4+34)  
err

## [1] 0.03508772

plot(treeFit); text(treeFit)



## Not surprising, but let's see what happens if we change our training and testing set  
  
set.seed(563)  
train=sample(1:nrow(ff),nrow(ff)\*(8/10))  
test=-train  
  
trainingData=ff[train,]  
testingData=ff[test,]  
treeFit <- rpart(formula,  
 data=trainingData,method="class")  
  
testingData$result2 <- predict(treeFit, testingData,type="class")  
count(testingData,c("V2","result2"))

## V2 result2 freq  
## 1 0 0 69  
## 2 0 1 2  
## 3 1 0 3  
## 4 1 1 40

err = 5/(69+5+40)  
err

## [1] 0.04385965

## Pretty good.  
  
## Now let's see if these same 3 variables work for our SVM from last week.  
library(e1071)  
svm\_model <- svm(formula, data=trainingData)  
summary(svm\_model)

##   
## Call:  
## svm(formula = formula, data = trainingData)  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: radial   
## cost: 1   
## gamma: 0.03333333   
##   
## Number of Support Vectors: 112  
##   
## ( 55 57 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## 0 1

testingData$result <- predict(svm\_model,testingData)  
count(testingData,c("V2","result"))

## V2 result freq  
## 1 0 0 70  
## 2 0 1 1  
## 3 1 0 1  
## 4 1 1 42

err = 2/(2+70+42)  
err

## [1] 0.01754386

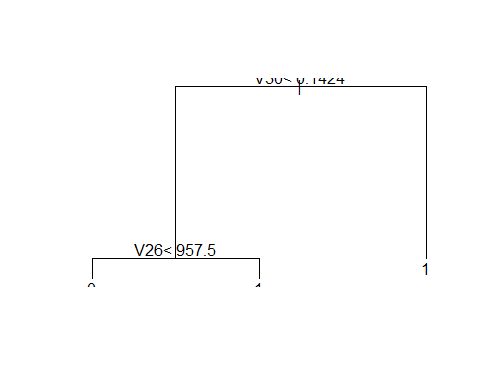
## Excellent. Let's see what happens if we just take the supposed 5 most important variables  
## V30, V10, V26, V23, V25 and leaving out V4  
formuala <- as.factor(v2)~V30+V10+V26+v23+v25  
treeFit <- rpart(formula,  
 data=trainingData,method="class")  
  
testingData$result2 <- predict(treeFit, testingData,type="class")  
count(testingData,c("V2","result2"))

## V2 result2 freq  
## 1 0 0 69  
## 2 0 1 2  
## 3 1 0 3  
## 4 1 1 40

err = 5/(69+5+40)  
err

## [1] 0.04385965

## Same error  
plot(treeFit); text(treeFit)



## Much simpler tree.  
## See if we can get away with just using V30 and V26  
formula <- as.factor(V2)~V30+V26  
treeFit <- rpart(formula,  
 data=trainingData,method="class")  
  
testingData$result2 <- predict(treeFit, testingData,type="class")  
count(testingData,c("V2","result2"))

## V2 result2 freq  
## 1 0 0 69  
## 2 0 1 2  
## 3 1 0 3  
## 4 1 1 40

## Same error. See if it works with a different training and testing set  
  
set.seed(1017)  
train=sample(1:nrow(ff),nrow(ff)\*(8/10))  
test=-train  
  
trainingData=ff[train,]  
testingData=ff[test,]  
treeFit <- rpart(formula,  
 data=trainingData,method="class")  
  
testingData$result2 <- predict(treeFit, testingData,type="class")  
count(testingData,c("V2","result2"))

## V2 result2 freq  
## 1 0 0 75  
## 2 1 0 3  
## 3 1 1 36

formula

## as.factor(V2) ~ V30 + V26

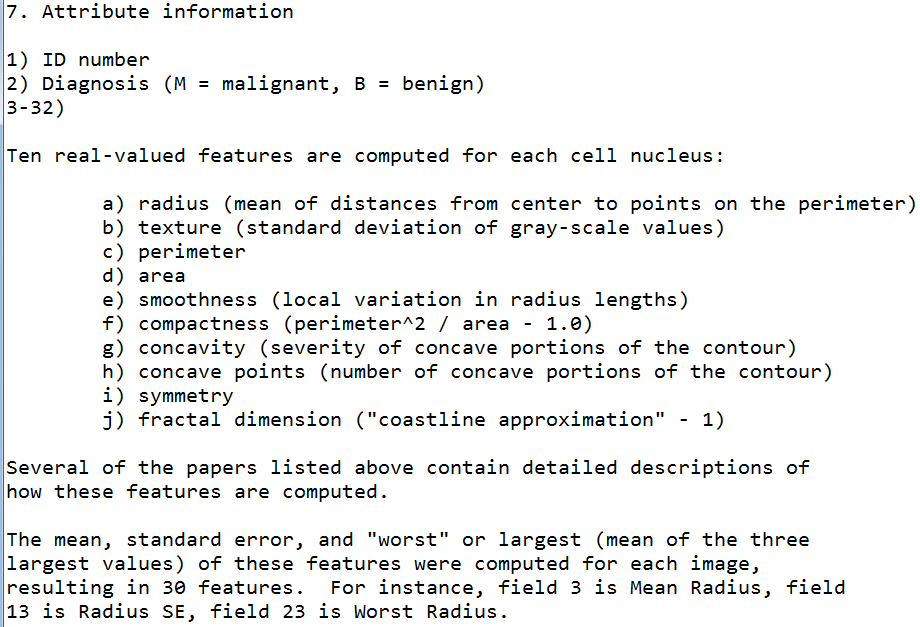
## See if the SVM performs as well with just 2 variables  
svm\_model <- svm(formula, data=trainingData)  
summary(svm\_model)

##   
## Call:  
## svm(formula = formula, data = trainingData)  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: radial   
## cost: 1   
## gamma: 0.5   
##   
## Number of Support Vectors: 82  
##   
## ( 39 43 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## 0 1

testingData$result <- predict(svm\_model,testingData)  
count(testingData,c("V2","result"))

## V2 result freq  
## 1 0 0 75  
## 2 1 0 2  
## 3 1 1 37

## Astonishing. What we have shown is that even though we have 33 variables; We can  
## do an excellent job of predicting malignancy with just 2 variables  
## V30 and V26. Let's look those up and see what they are.



## V30 must be worst concave points  
## V26 must be worst area