HW10

Jingshi

4/14/2017

Quetsion 1

Code with improvements from class work:

```
# Read the data
airfoil<-read.table(
  "http://archive.ics.uci.edu/ml/machine-learning-databases/00291/airfoil_self_noise.dat")
#head(airfoil)

#nrow(airfoil)
# Split into train and test

set.seed(6)
train <- sample(nrow(airfoil), nrow(airfoil) *0.7)
#length(train)
mytrain <- airfoil[train, ]
mytest <- airfoil[-train, ]

######################## Linear model ###########################
lm.model<-lm(V6~. ,data = mytrain)

# Calculate rms of linear model
prediction.lm <- predict(lm.model, mytest)
rms.lm<- sqrt(mean((mytest$V6 - prediction.lm)^2))

######################### Decision Tree model ############################
#install.packages("tree")
require(tree)
```

```
## Loading required package: tree
```

```
tree.model<-tree(V6~. ,data=mytrain)
#summary(tree.model)
# plot tree
#plot(tree.model)
#text(tree.model, pretty=2)
# Calculate rms
prediction.tree <- predict(tree.model, newdata=mytest)
rms.tree<- sqrt(mean((mytest$V6 - prediction.tree)^2))

######################### Bagging ##############################
ntrain = nrow(mytrain)
N = 100 # number of bootstrap samples
mybag = as.list(rep(NA,N))
for (j in 1:N){
  bootstrap = sample(ntrain, replace = TRUE)
  mybag[[j]] = tree(V6 ~ ., data = mytrain[bootstrap,])
```

```
}

# Prediction
pred.bag = 0*airfoil$V6[-train]
for (j in 1:N){
  pred.bag = pred.bag + predict(mybag[[j]], newdata = mytest)
}
pred.bag = pred.bag/N
# Calculate rms
rms.bag<- sqrt(mean((mytest$V6 - pred.bag)^2))

###################### Boosted Tree################################
#install.packages("gbm")
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.3.2

## Loading required package: survival

## Loading required package: lattice

## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.3
```

```
boost.1 = gbm(V6 ~., data = mytrain, distribution = "gaussian", n.trees = 5000,
              shrinkage =0.01, interaction.depth =2)
#summary(boost.1)
boost.pred.1 = predict(boost.1, newdata=mytest,n.trees = 5000, type ="response")
# Calculate rms
rms.boost<- sqrt(mean((mytest$V6 - boost.pred.1)^2))

################## Neural Networks ############################
library(nnet)
nnet.model <- nnet(V6 ~.,
                   size = 3,  data=mytrain,skip = F,
                   trace = F,maxit = 1000, linout = T)


prediction.nnet <- predict(nnet.model, newdata=mytest)
rms.nnet<- sqrt(mean((mytest$V6 - prediction.nnet)^2))
```

Report:

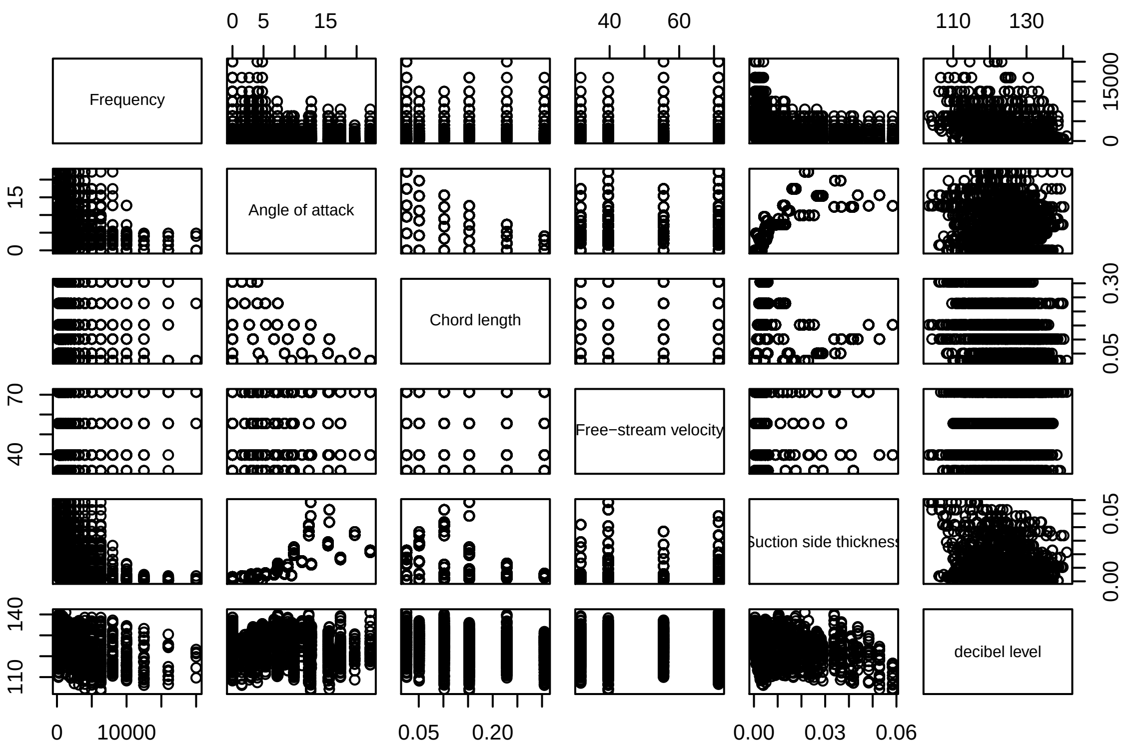
In the class work, our group was assigned to work on Airfoil self noise data. The general goal was to fit models to predict decible lever/scaled sound pressure level (in decibels) by using all the other attributes (Frequency in Hertz, Angle of attack in degrees, Chord length in meters, Free-stream velocity in meters per second and Suction side displacement thickness in meters).

In the data preperation step, we have diveded the data into 70% training set and 30% testing set.

Then, we have fitted a linear model with all predictors as a baseline case. Then, we fitted a decision tree, applied bagging, applied a boosted tree model. All These models are fitted with all the predictors.

Before looking at the models, let's take a look at the scatter plots of all the variables in order to get an overview of the data set:

```r
airfoil2<-airfoil
colnames(airfoil2)<-c("Frequency","Angle of attack","Chord length",
                      "Free-stream velocity", "Suction side thickness", "decibel level")
plot(airfoil2)
```



It seems that there is not obvious relationship between decible level and other variables. There seems to have a weak negative linear relationship between decible level and frequency. There also seems to have a weak negative linear relationship between decibel level and suction side displacement thickness. Interestingly, there seems to have a moderate to strong positive curved relationship between angle of attack and suction side displacement thickness, as you can see from the scatter plots.

Now, let's take a look at the fitted models and their evaluations.

Firstly, let's take a look at the linear model (which is our baseline model):

```r
summary(lm.model)
```

```
##
## Call:
## lm(formula = V6 ~ ., data = mytrain)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17.0108  -2.9870  -0.1887   2.9711  15.6765
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.330e+02  6.503e-01 204.591   <2e-16 ***
## V1          -1.301e-03  4.875e-05 -26.694   <2e-16 ***
## V2          -4.181e-01  4.737e-02  -8.827   <2e-16 ***
```

```
## V3          -3.796e+01  1.945e+00 -19.519   <2e-16 ***
## V4           1.051e-01  9.564e-03  10.989   <2e-16 ***
## V5          -1.568e+02  1.788e+01  -8.769   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.76 on 1046 degrees of freedom
## Multiple R-squared:  0.5435, Adjusted R-squared:  0.5414
## F-statistic: 249.1 on 5 and 1046 DF,  p-value: < 2.2e-16
```

```r
cat ("Root mean squared error of linear model:", rms.lm)
```

```
## Root mean squared error of linear model: 4.939083
```

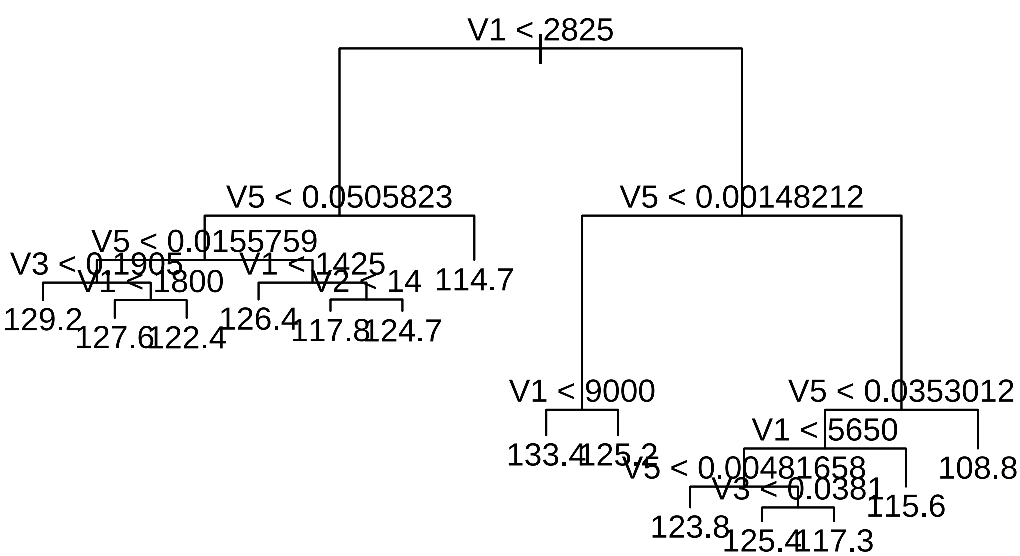
The summary shows that every parametes are significant because their p-values are all less than $\alpha = 0.05$. The model is also statistically significant because its p-value ($< 2.2 \times 10^{-16}$) from F-test is much smaller than $\alpha = 0.05$. The model's adjusted R-squared is high at 0.5414. And it has a root mean squared error of 4.939. Overall, the linear model fits well.

Secondly, let's take a look at the decision tree model:

```r
summary(tree.model)
```

```
##
## Regression tree:
## tree(formula = V6 ~ ., data = mytrain)
## Variables actually used in tree construction:
## [1] "V1" "V5" "V3" "V2"
## Number of terminal nodes:  14
## Residual mean deviance:  18.7 = 19410 / 1038
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -13.9000  -2.7060   0.1481   0.0000   2.5110  14.6400
```

```r
# plot tree
plot(tree.model)
text(tree.model, pretty=1)
```




```r
cat ("Root mean squared error of decision tree model:", rms.tree)
```

```
## Root mean squared error of decision tree model: 4.633032
```

As the summary shows, the decision tree model actually used 4 variables in tree construction. They are V1 (Frequency, in Hertzs), V5 (Suction side displacement thickness, in meters), V3 (Chord length, in meters) and V2 (Angle of attack, in degrees). The above tree graph also shows how the decision tree is constructed. It has a root mean squared of 4.633032 which is close to the root mean squared error of the linear model.

Thirdly, we have fitted a bagging model with all the predictors and 100 as the number of bootstrap samples.

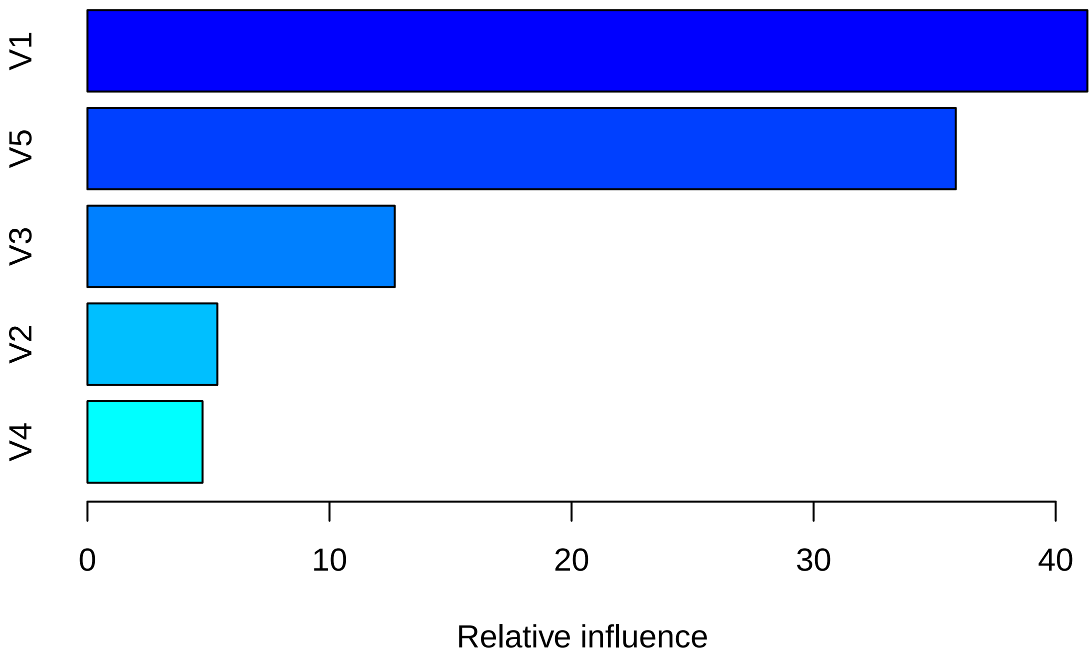
```r
cat ("Root mean squared error of bagging model:", rms.bag)
```

```
## Root mean squared error of bagging model: 3.648248
```

It has a root mean squared error of 3.648248 which is smaller than the previous two models. This indicates that the model fits even better than the previous two models.

Fourthly, we have fitted a boosted tree model. Let's take a look at its summary:

```r
summary(boost.1)
```




```
##    var   rel.inf
## V1  V1 41.309733
## V5  V5 35.873254
## V3  V3 12.695569
## V2  V2  5.365919
## V4  V4  4.755526
```

```r
cat ("Root mean squared error of boosted tree model:", rms.boost)
```

```
## Root mean squared error of boosted tree model: 2.401303
```

As you can see from the output, V1 (Frequency, in Hertzs) has the largest relative influence. V5 (Suction side displacement thickness, in meters) has the second largest relative influence. The other 3 variables (V3-chord length, V2-angle of attack and V4-free-stream velocity) have comparatively lower relative influence. Moreover, root mean squred error of boosted tree model is 2.401303 which indicates that this model fits better than the previous models.

Fifthly, there are other methods for predicting the noise level. Such as neural networks. I also fitted a neural networks model as shown below:

```
summary(nnet.model)
```

```
## a 5-3-1 network with 22 weights
## options were - linear output units
##  b->h1 i1->h1 i2->h1 i3->h1 i4->h1 i5->h1
##   0.72  13.97   0.35   0.34   2.01   0.04
##  b->h2 i1->h2 i2->h2 i3->h2 i4->h2 i5->h2
##   0.25  -0.33  -0.60  -0.67  -0.17  -0.47
##  b->h3 i1->h3 i2->h3 i3->h3 i4->h3 i5->h3
##   0.20   0.39  -0.40  -0.24   0.23  -0.14
##   b->o  h1->o  h2->o  h3->o
## 102.53 -80.17  -0.44 102.45
```

```
cat ("Root mean squared error of neural networks model:", rms.nnet)
```

```
## Root mean squared error of neural networks model: 6.585085
```

As the output shows, neural networks model has a higher root mean squred error than decision tree model, bagging model, boosted tree model and linear model. This implies that this neural networks model is not as good as the previous models in terms of model quality.

Last but not least, let's take a look at the root mean squared errors for all the above fiited models:

```
as.table(matrix(c("rms_linear_model", rms.lm, "rms_decision_tree_model", rms.tree,
                  "rms_bagging",rms.bag, "rms_boosted_tree_model", rms.boost,
                  "rms_nnet",rms.nnet),nrow = 2))
```

```
##   A                B                       C
## A rms_linear_model rms_decision_tree_model rms_bagging
## B 4.93908254829089 4.63303212261244        3.6482475631992
##   D                      E
## A rms_boosted_tree_model rms_nnet
## B 2.40130261644975       6.58508523294135
```

As the output shows, boosted tree model has the lowest root mean squared error (about 2.40) so it is considered as the best model over all the fitted models.

Question 2

```
load("ex0408.rdata")
#mydf.test
#mydf.train

################## Fit a tree to the training data ###################
require(tree)
set.seed(1)
mytree<-tree(z~. ,data=mydf.train)


# Evaluate the model on the training data
pred=predict(mytree, newdata =mydf.train)
#pred
p.pred <- apply(pred, 1, which.max)
```

```r
#length(p.pred)
#p.pred
tab=table(mydf.test$z,p.pred)
tab
```

```
##        p.pred
##            1    2
##   FALSE  437 2544
##   TRUE  1112 5907
```

```r
print(paste('Accuracy rate on the training data:', sum(diag(tab))/sum(tab)))
```

```
## [1] "Accuracy rate on the training data: 0.6344"
```

```r
# Evaluate the model on the test data
pred=predict(mytree, newdata =mydf.test)
#pred
p.pred <- apply(pred, 1, which.max)
#length(p.pred)
#p.pred
tab=table(mydf.test$z,p.pred)
tab
```

```
##        p.pred
##            1    2
##   FALSE 1047 1934
##   TRUE   545 6474
```

```r
print(paste('Test accuracy rate:', sum(diag(tab))/sum(tab)))
```

```
## [1] "Test accuracy rate: 0.7521"
```

The accuracy rate on the training data is 0.6344. The test accuracy rate is 0.7521 which indicates that the tree model fits well.

```r
########################## Bagging ########################

#install.packages("ipred")
library(ipred)
```

```
## Warning: package 'ipred' was built under R version 3.3.2
```

```r
set.seed(1)
bag.2 <- bagging(z ~ ., data = mydf.train)

# See the accuracy rate on the training data
pred.bag.2 <- predict(bag.2, newdata = mydf.train)
tab = table(mydf.train$z,pred.bag.2)
tab
```

```
##        pred.bag.2
##         FALSE TRUE
##   FALSE  2976   12
##   TRUE      3 7009
```

```r
print(paste('Accuracy rate on the training data:', sum(diag(tab))/sum(tab)))
```

```
## [1] "Accuracy rate on the training data: 0.9985"
```

```r
# test accuracy rate
pred.bag.2 <- predict(bag.2, newdata = mydf.test)
tab = table(mydf.test$z,pred.bag.2)
tab
```

```
##        pred.bag.2
##         FALSE TRUE
##   FALSE  1469 1512
##   TRUE   1054 5965
```

```r
print(paste('Test accuracy rate:', sum(diag(tab))/sum(tab)))
```

```
## [1] "Test accuracy rate: 0.7434"
```

As the output shows, the accuracy rate on the training data is 0.9985 which is very close to 1. So it is possible to obtain a perfect fit on the training data. The test accuracy rate is 0.7434 which is slightly lower than the tree model but still fits well.

```r
#################### Random Forest #########################
library(randomForest)
```

```
## randomForest 4.6-12
```

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

```r
set.seed(1)
myrf = randomForest(z ~., data=mydf.train, mtry = ceiling(sqrt(10)), importance = TRUE)

# See the accuracy rate on the training data
pred <- predict(myrf, newdata = mydf.train)
tab = table(mydf.train$z,pred)
tab
```

```
##        pred
##         FALSE TRUE
##   FALSE  2988    0
##   TRUE      0 7012
```

```r
print(paste('Accuracy rate on the training data:', sum(diag(tab))/sum(tab)))
```

```
## [1] "Accuracy rate on the training data: 1"
```

```r
# test accuracy rate
pred<- predict(myrf, newdata = mydf.test)
tab = table(mydf.test$z,pred)
tab
```

```
##        pred
##         FALSE TRUE
##   FALSE  1468 1513
##   TRUE    919 6100
```

```r
print(paste('Test accuracy rate:', sum(diag(tab))/sum(tab)))
```

```
## [1] "Test accuracy rate: 0.7568"
```

As the output shows, accuracy rate on the training data is 1 which means it is possible to obtain a perfect fit on the training data.

The test accurcy rate is 0.7568 which is about the same as the previous 2 models. The model fits well overall.

```r
########### Boosted tree ###################
library(gbm)
set.seed(1)
mydf.train$z<-as.numeric(mydf.train$z)-1
mydf.test$z<-as.numeric(mydf.test$z)-1

boosting<-gbm(z~.,data=mydf.train, distribution='bernoulli',
              n.trees=5000,shrinkage=0.002,interaction.depth = 10)

# See the accuracy rate on the training data
pred <- predict(boosting, newdata = mydf.train,n.trees = 5000, type = "response")
tab = table(mydf.train$z,pred>0.5)
tab
```

```
##
##     FALSE TRUE
##   0  1653 1335
##   1   601 6411
```

```r
print(paste('Accuracy rate on the training data:', sum(diag(tab))/sum(tab)))
```

```
## [1] "Accuracy rate on the training data: 0.8064"
```

```r
# test accuracy rate
pred<- predict(boosting, newdata = mydf.test,n.trees = 5000, type = "response")
tab = table(mydf.test$z,pred>0.5)
tab
```

```
##
##     FALSE TRUE
##   0  1453 1528
##   1   887 6132
```

```r
print(paste('Test accuracy rate:', sum(diag(tab))/sum(tab)))
```

```
## [1] "Test accuracy rate: 0.7585"
```

As the output shows, the accuracy rate on the training data is 0.8064. The training data has been fitted well but not as perfect as bagging and random forest. The test accuracy rate is 0.7585 which means this model performs well in the test data and the test accuracy rate is about the same level as the previous models.