Practical Machine Learning - Quiz 4

Marcela Castro Leon

March 24, 2018

Question 1

For this quiz we will be using several R packages. R package versions change over time, the right answers have been checked using the following versions of the packages. AppliedPredictiveModeling: v1.1.6 caret: v6.0.47 ElemStatLearn: v2012.04-0 pgmm: v1.1 rpart: v4.1.8 gbm: v2.1 lubridate: v1.3.3 forecast: v5.6 e1071: v1.6.4 If you aren't using these versions of the packages, your answers may not exactly match the right answer, but hopefully should be close.

Load the vowel.train and vowel.test data sets:

```
library(AppliedPredictiveModeling)  #1.1-6
library(caret)  #packageDescription("caret") 6.0-78
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(ElemStatLearn) #packageDescription("ElemStatLearn") #2015.6.26
library(pgmm)  #1.2.2
library(rpart) #4.1-13
library(gbm)
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: splines
```

```
## Loading required package: parallel
```

```
## Loaded gbm 2.1.3
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
library(forecast)
library(e1071)
library(ElemStatLearn)
suppressMessages(library(caret))
data(vowel.train)
data(vowel.test)
```

Set the variable y to be a factor variable in both the training and test set. Then set the seed to 33833. Fit (1) a random forest predictor relating the factor variable y to the remaining variables and (2) a boosted predictor using the “gbm” method. Fit these both with the train() command in the caret package. What are the accuracies for the two approaches on the test data set? What is the accuracy among the test set samples where the two methods agree?

```
vowel.test_f<-cbind(as.factor(vowel.test$y), vowel.test[,-1])
vowel.train_f<-cbind(as.factor(vowel.train$y), vowel.train[,-1])
names(vowel.test_f)<-names(vowel.test)
names(vowel.train_f)<-names(vowel.train)
set.seed(33833)
modelFit_boost<-train(y~., method="gbm", data=vowel.train_f,verbose=FALSE)
modelFit_rf<-train(y~., method="rf", data=vowel.train_f)
predic_boost<-predict(modelFit_boost,newdata=vowel.test_f)
predic_rf<-predict(modelFit_rf,newdata=vowel.test_f)
cm_boost<-confusionMatrix(predic_boost, vowel.test_f$y)
cm_rf<-confusionMatrix(predic_rf, vowel.test_f$y)
cm_boost$overall['Accuracy']
```

```
##  Accuracy
## 0.5324675
```

```
cm_rf$overall['Accuracy']
```

```
##  Accuracy
## 0.5887446
```

```
cm_agree<-confusionMatrix(predic_rf,predic_boost)
cm_agree$overall['Accuracy']
```

```
##  Accuracy
## 0.6861472
```

Answer: (by aprox) RF Accuracy = 0.6082 GBM Accuracy = 0.5152 Agreement Accuracy = 0.6361

Question 2

Load the Alzheimer’s data using the following commands

```
library(caret)
library(gbm)
set.seed(3433)
library(AppliedPredictiveModeling)
data(AlzheimerDisease)
adData = data.frame(diagnosis,predictors)
inTrain = createDataPartition(adData$diagnosis, p = 3/4)[[1]]
training = adData[ inTrain,]
testing = adData[-inTrain,]
```

Set the seed to 62433 and predict diagnosis with all the other variables using a random forest ("rf"), boosted trees ("gbm") and linear discriminant analysis ("lda") model. Stack the predictions together using random forests ("rf"). What is the resulting accuracy on the test set? Is it better or worse than each of the individual predictions?

```
set.seed(62433)
modelFit_rf<-train(diagnosis~., method="rf", data=training)
modelFit_gbm<-train(diagnosis~., method="gbm", data=training,verbose=FALSE)
modelFit_lda<-train(diagnosis~., method="lda", data=training)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
predict_rf<-predict(modelFit_rf,newdata=testing)
predict_gbm<-predict(modelFit_gbm,newdata=testing)
predict_lda<-predict(modelFit_lda,newdata=testing)
confusionMatrix(predict_rf, testing$diagnosis)$overall['Accuracy']
```

```
##  Accuracy
## 0.7682927
```

```
confusionMatrix(predict_gbm, testing$diagnosis)$overall['Accuracy']
```

```
##  Accuracy
## 0.8170732
```

```
confusionMatrix(predict_lda, testing$diagnosis)$overall['Accuracy']
```

```
##  Accuracy
## 0.7682927
```

```
#create a new dataframe with the predictions
predDF <- data.frame(predict_rf, predict_gbm, predict_lda, diagnosis = testing$diagnosis)
#create a new model using the new data frame and rf method
combModFit <- train(diagnosis ~.,method="rf",data=predDF)
```

```
## note: only 2 unique complexity parameters in default grid. Truncating the grid to 2 .
```

```
#predict values and calculate the confusion matrix to check the accuracy
combPred <- predict(combModFit, predDF)
confusionMatrix(combPred, testing$diagnosis)$overall['Accuracy']
```

```
##  Accuracy
## 0.8170732
```

Answer: Stacked Accuracy: 0.80 is better than random forests and lda and the same as boosting.

Question 3

Load the concrete data with the commands:

```
set.seed(3523)
library(AppliedPredictiveModeling)
library(elasticnet)
```

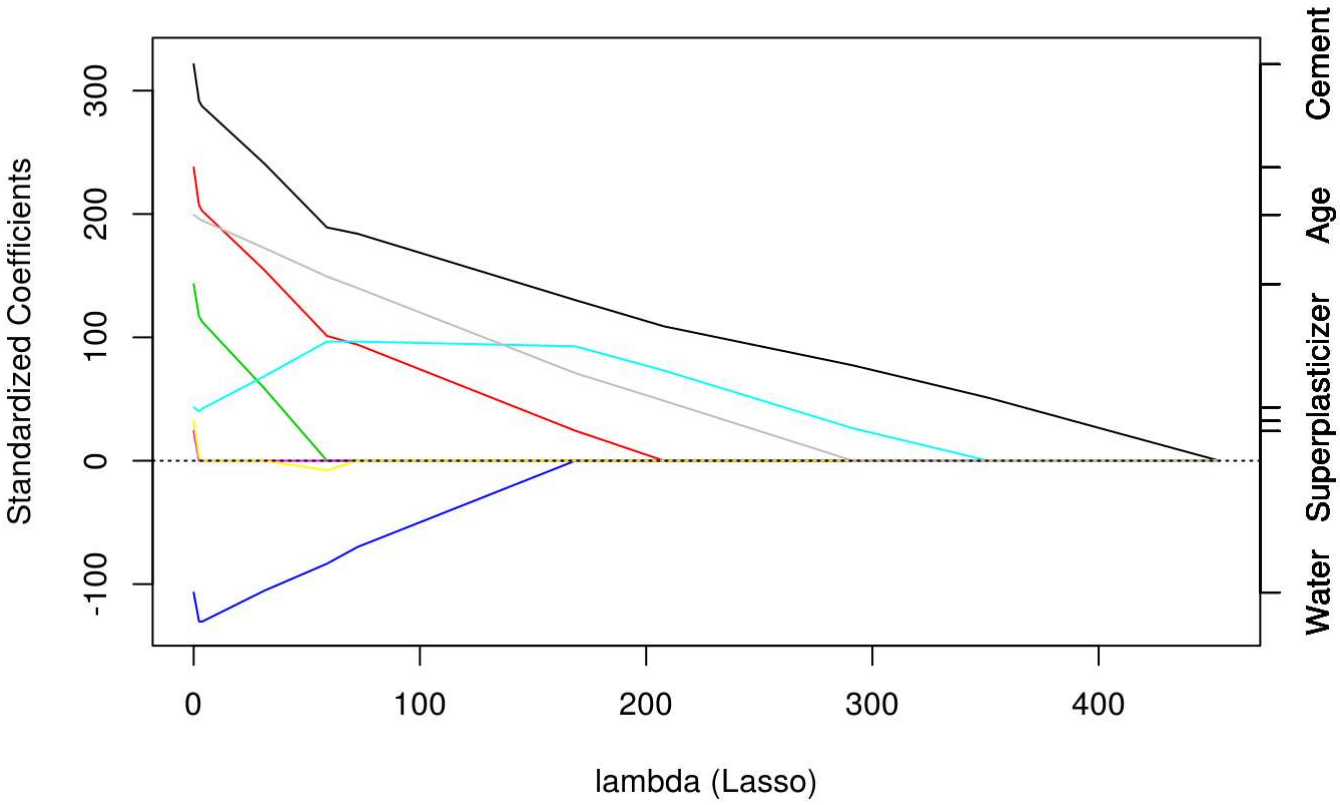
```
## Loading required package: lars
```

```
## Loaded lars 1.2
```

```
data(concrete)
inTrain = createDataPartition(concrete$CompressiveStrength, p = 3/4)[[1]]
training = concrete[ inTrain,]
testing = concrete[-inTrain,]
```

Set the seed to 233 and fit a lasso model to predict Compressive Strength. Which variable is the last coefficient to be set to zero as the penalty increases? (Hint: it may be useful to look up ?plot.enet).

```
set.seed(233)
modFit<-train(CompressiveStrength ~ ., method="lasso", data=training)
plot.enet(modFit$finalModel, xvar = "penalty", use.color = TRUE)
```



Answer: Cement

Question 4

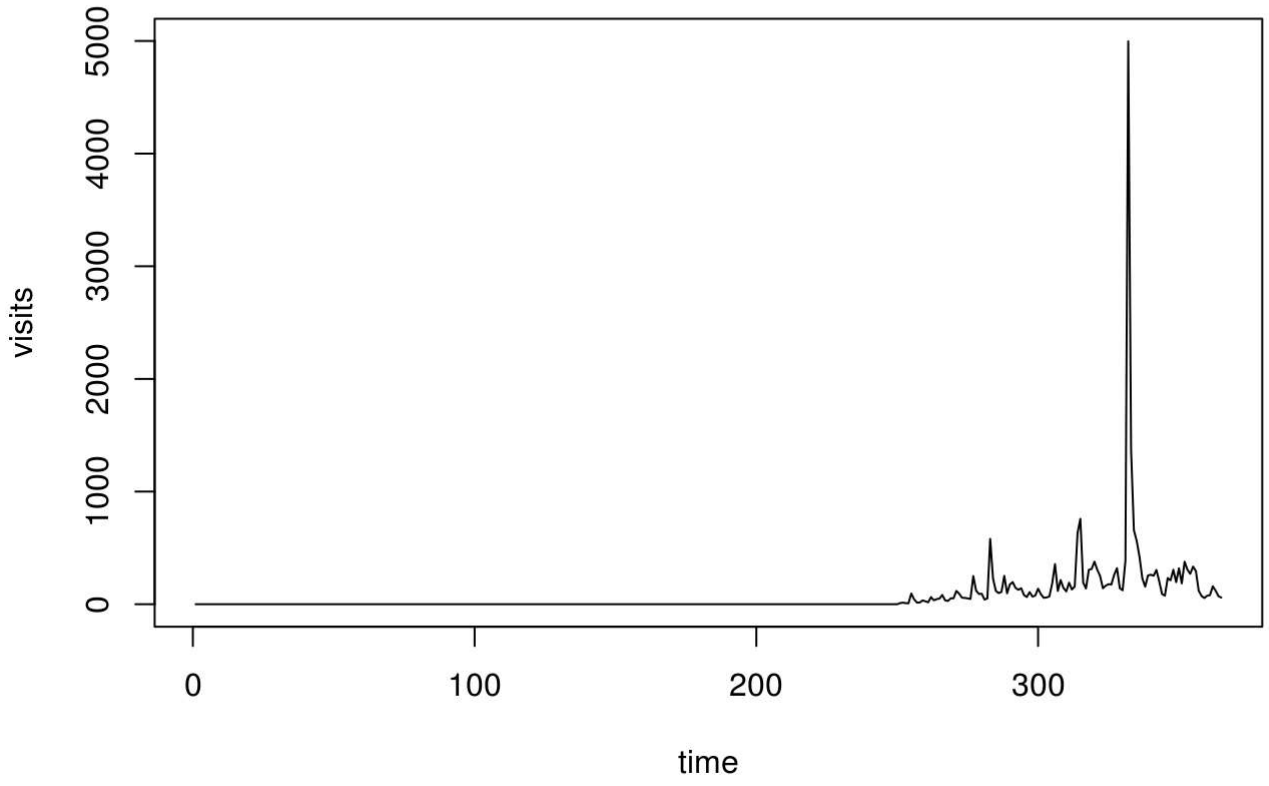
Load the data on the number of visitors to the instructors blog from here: https://d396qusza40orc.cloudfront.net/predmachlearn/gaData.csv

Using the commands:

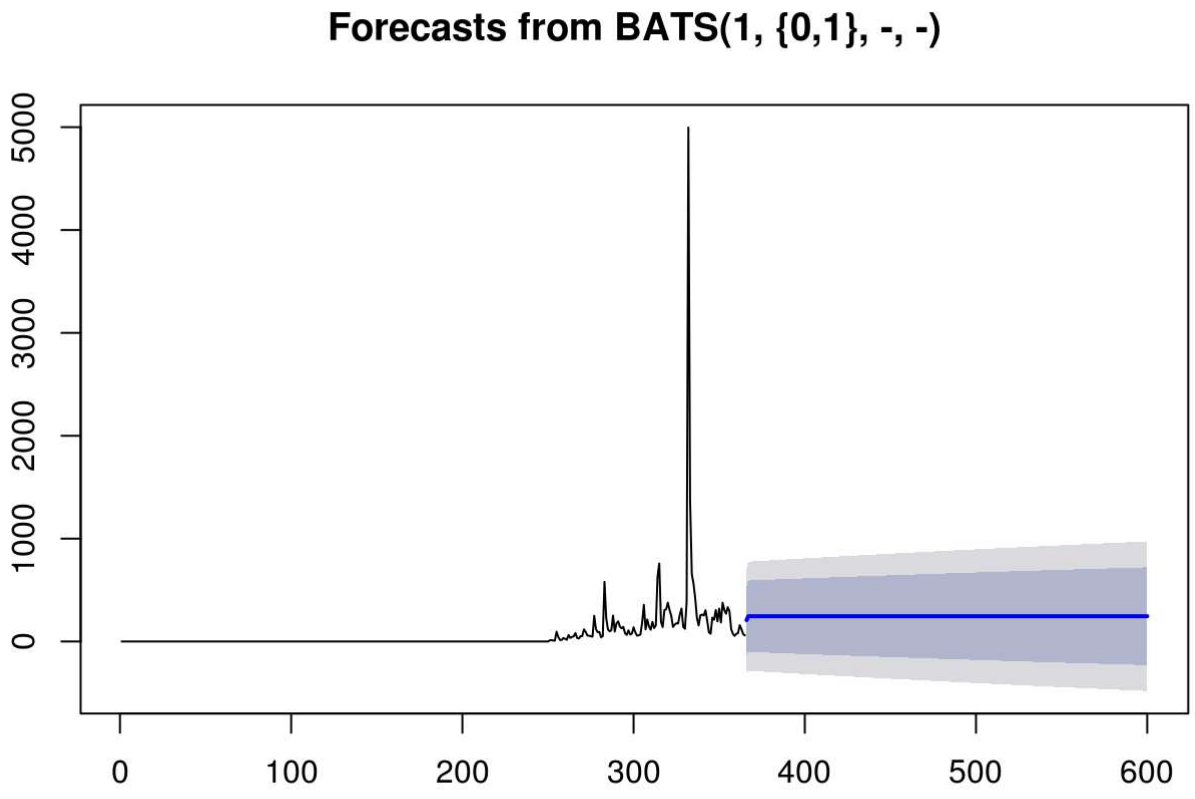
```
library(lubridate) # For year() function below
dat = read.csv("~/Desktop/gaData.csv")
training = dat[year(dat$date) < 2012,]
testing = dat[(year(dat$date)) > 2011,]
tstrain = ts(training$visitsTumblr)
```

Fit a model using the bats() function in the forecast package to the training time series. Then forecast this model for the remaining time points. For how many of the testing points is the true value within the 95% prediction interval bounds?

```
modFit<-bats(tstrain)
plot(tstrain,xlab="time", ylab="visits")
```




```
fcast <- forecast(modFit, h=nrow(testing))
plot(fcast)
```



```
fcast_lower95 = fcast$lower[,2]
fcast_upper95 = fcast$upper[,2]
table( (testing$visitsTumblr>fcast_lower95) & (testing$visitsTumblr<fcast_upper95) )
```

```
##
## FALSE  TRUE
##     9   226
```

Answer: 0.96

Question 5

Load the concrete data with the commands:

```
set.seed(3523)
library(e1071)
library(AppliedPredictiveModeling)
data(concrete)
inTrain = createDataPartition(concrete$CompressiveStrength, p = 3/4)[[1]]
training = concrete[ inTrain,]
testing = concrete[-inTrain,]
```

Set the seed to 325 and fit a support vector machine using the e1071 package to predict Compressive Strength using the default settings. Predict on the testing set. What is the RMSE?

```
set.seed(325)
modFit<-svm(CompressiveStrength~., data=training)
predict_svm<-predict(modFit, testing)
accuracy(predict_svm, testing$CompressiveStrength)
```

```
##                 ME     RMSE      MAE       MPE     MAPE
## Test set 0.1682863 6.715009 5.120835 -7.102348 19.27739
```

Answer: RMSE: 6.71