Practical Machine Learning - Week 4 Quiz

Ken Wood

9/18/2020

Clear out all previous R sessions...

```
rm(list = ls())
```

Question 1 Load the vowel.train and vowel.test data sets:

```
library(caret)
```

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

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

```
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```
library(ElemStatLearn)
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.train$y <- as.factor(vowel.train$y)
vowel.test$y <- as.factor(vowel.test$y)
set.seed(33833)
mod_rf <- train(y ~ ., method = "rf", data = vowel.train)
mod_gbm <- train(y ~ ., method = "gbm", data = vowel.train,verbose=FALSE)
pred_rf <- predict(mod_rf, vowel.test)
pred_gbm <- predict(mod_gbm, vowel.test)
```

Get accuracy of Random Forest model...

```
confusionMatrix(pred_rf, vowel.test$y)$overall[1]
```

```
##  Accuracy
## 0.5974026
```

Get accuracy of Boost model...

```
confusionMatrix(pred_gbm, vowel.test$y)$overall[1]
```

```
## Accuracy
## 0.530303
```

Create dataframe of rf & gbm predictions along with true y_test values.

```
predDF <- data.frame(pred_rf, pred_gbm, y = vowel.test$y)
```

Calculate accuracy among the test set samples where the two methods agree.

```
sum(pred_rf[predDF$pred_rf == predDF$pred_gbm] ==
        predDF$y[predDF$pred_rf == predDF$pred_gbm]) /
  sum(predDF$pred_rf == predDF$pred_gbm)
```

```
## [1] 0.640625
```

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)
mod_rf <- train(diagnosis ~ ., method = "rf",data = training)
mod_gbm <- train(diagnosis ~ .,  method = "gbm",data = training, verbose=FALSE)
mod_lda <- train(diagnosis ~ ., method = "lda", data = training,verbose=FALSE)
```

```
## 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: model fit failed for Resample23: parameter=none Error in lda.default(x, grouping, ...) :
##   variable 131 appears to be constant within groups

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```
pred_rf <- predict(mod_rf, testing)
pred_gbm <- predict(mod_gbm, testing)
pred_lda <- predict(mod_lda, testing)
predDF <- data.frame(pred_rf, pred_gbm, pred_lda, diagnosis = testing$diagnosis)
```

Accuracy of RF

```
confusionMatrix(pred_rf, testing$diagnosis)$overall[1]
```

```
## Accuracy
## 0.902439
```

Accuracy of GBM

```
confusionMatrix(pred_gbm, testing$diagnosis)$overall[1]
```

```
## Accuracy
## 0.902439
```

Accuracy of LDA

```
confusionMatrix(pred_lda, testing$diagnosis)$overall[1]
```

```
##  Accuracy
## 0.9146341
```

```
combModFit <- train(diagnosis ~ ., method = "rf", data = predDF)
```

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

```
combPred <- predict(combModFit, predDF)
confusionMatrix(combPred,testing$diagnosis)$overall[1]
```

```
##  Accuracy
## 0.9268293
```

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

Question 3

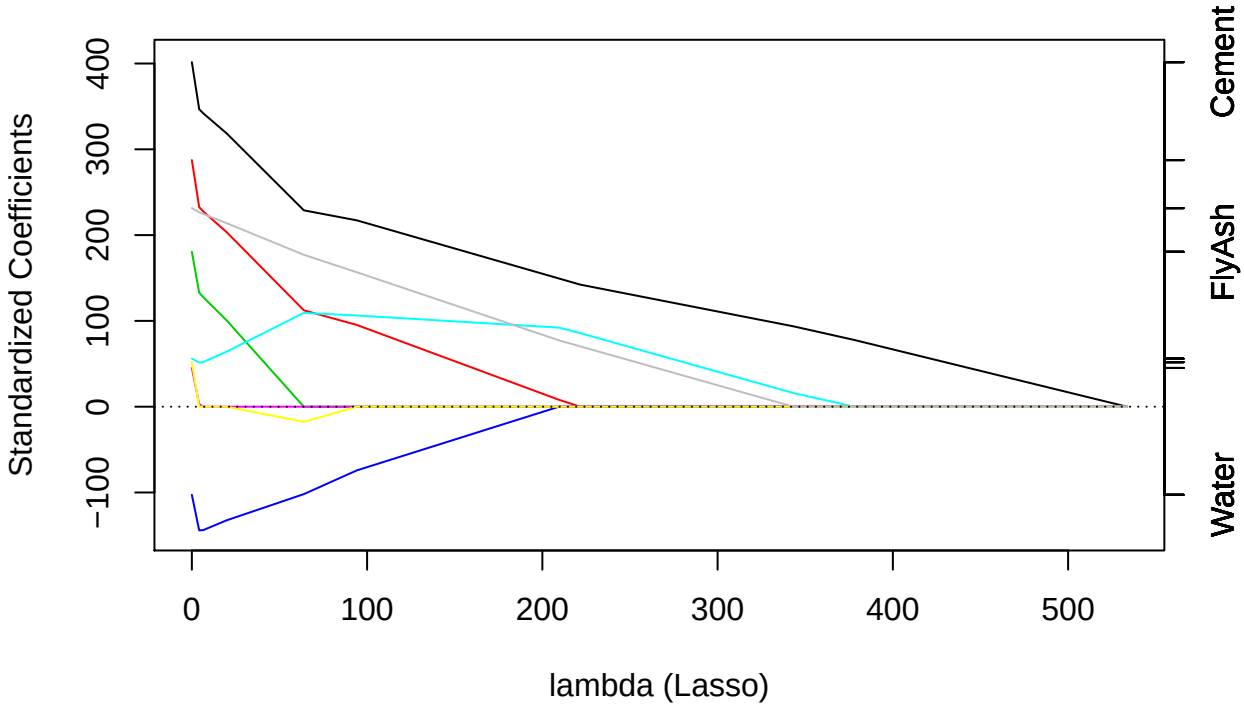
```
## Loading required package: lars
```

```
## 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)
mod_lasso <- train(CompressiveStrength~.,method="lasso",data=concrete)
plot.enet(mod_lasso$finalModel,xvar="penalty",use.color=TRUE)
```



The coefficient path shows that the variable Cement is the last coefficient to be set to zero as the penalty increases.

Question 4 Load the data on the number of visitors to the instructors blog from here:

https://d396qusza40orc.cloudfront.net/predmachlearn/gaData.csv

```
library(lubridate) # For year() function below
```

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

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
dat = read.csv("~/Desktop/gaData.csv")
training = dat[year(dat$date) < 2012,]
```

```
## Warning: tz(): Don't know how to compute timezone for object of class factor;
## returning "UTC". This warning will become an error in the next major version of
## lubridate.
```

```
testing = dat[(year(dat$date)) > 2011,]
```

```
## Warning: tz(): Don't know how to compute timezone for object of class factor;
## returning "UTC". This warning will become an error in the next major version of
## lubridate.
```

```
tstrain = ts(training$visitsTumblr)
```

```
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
```

```
##   as.zoo.data.frame zoo
```

```
mod_ts <- bats(tstrain)
fcast <- forecast(mod_ts, level = 95, h = dim(testing)[1])
sum(fcast$lower < testing$visitsTumblr & testing$visitsTumblr < fcast$upper) / dim(testing)[1]
```

```
## [1] 0.9617021
```

Question 5 Load the concrete data with the commands:

```
set.seed(3523)
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)
library(e1071)
model_svm <- svm(CompressiveStrength ~ ., data=training)
pred_svm <- predict(model_svm,testing)
accuracy(pred_svm, testing$CompressiveStrength)
```

```
##                 ME     RMSE      MAE       MPE     MAPE
## Test set 0.3113479 7.962075 5.515605 -6.845664 20.31935
```