Week 4 Quiz

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**(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)

Q1modRf <- train(y ~. , data = vowel.train, method = 'rf')

Q1modGbm <- train(y ~. , data = vowel.train, method = 'gbm', verbose = F)

Q1predRf <- predict(Q1modRf, newdata = vowel.test)

confusionMatrix(Q1predRf, vowel.test$y)$overall[1]

## Accuracy

## 0.5887446

Q1predGbm <- predict(Q1modGbm, newdata = vowel.test)

confusionMatrix(Q1predGbm, vowel.test$y)$overall[1]

## Accuracy

## 0.512987

agree <- Q1predRf == Q1predGbm

confusionMatrix(vowel.test$y[agree], Q1predRf[agree])$overall[1]

## Accuracy

## 0.621118

confusionMatrix(vowel.test$y[agree], Q1predGbm[agree])$overall[1]

## Accuracy

## 0.621118

* RF Accuracy = 0.6082  
  GBM Accuracy = 0.5152  
  Agreement Accuracy = 0.5152
* RF Accuracy = 0.9987  
  GBM Accuracy = 0.5152  
  Agreement Accuracy = 0.9985
* **RF Accuracy = 0.6082**  
  **GBM Accuracy = 0.5152**  
  **Agreement Accuracy = 0.6361**
* RF Accuracy = 0.9881  
  GBM Accuracy = 0.8371  
  Agreement Accuracy = 0.9983

1. 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)

Q2modRf <- train(diagnosis ~. , data = training, method = 'rf')

Q2modGbm <- train(diagnosis ~. , data = training, method = 'gbm', verbose = F)

Q2modLda <- train(diagnosis ~. , data = training, method = 'lda')

Q2predRf <- predict(Q2modRf, newdata = testing)

Q2predGbm <- predict(Q2modGbm, newdata = testing)

Q2predLda <- predict(Q2modLda, newdata = testing)

confusionMatrix(Q2predRf, testing$diagnosis)$overall[1]

## Accuracy

## 0.7926829

confusionMatrix(Q2predGbm, testing$diagnosis)$overall[1]

## Accuracy

## 0.7804878

confusionMatrix(Q2predLda, testing$diagnosis)$overall[1]

## Accuracy

## 0.7682927

Q2dataStack <- data.frame(Q2predRf, Q2predGbm, Q2predLda,

diagnosis = testing$diagnosis)

Q2modStack <- train(diagnosis ~., data = Q2dataStack, method = 'gam')

Q2predStack <- predict(Q2modStack, newdata = Q2dataStack)

confusionMatrix(Q2predStack, Q2dataStack$diagnosis)$overall[1]

## Accuracy

## 0.804878

* Stacked Accuracy: 0.80 is worse than all the other methods.
* *Stacked Accuracy: 0.80 is better than all three other methods*
* **Stacked Accuracy: 0.80 is better than random forests and lda and the same as boosting.**
* Stacked Accuracy: 0.88 is better than all three other methods
* NOTE: The difference in accuracy could be due to different versions of the package

1. 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,]

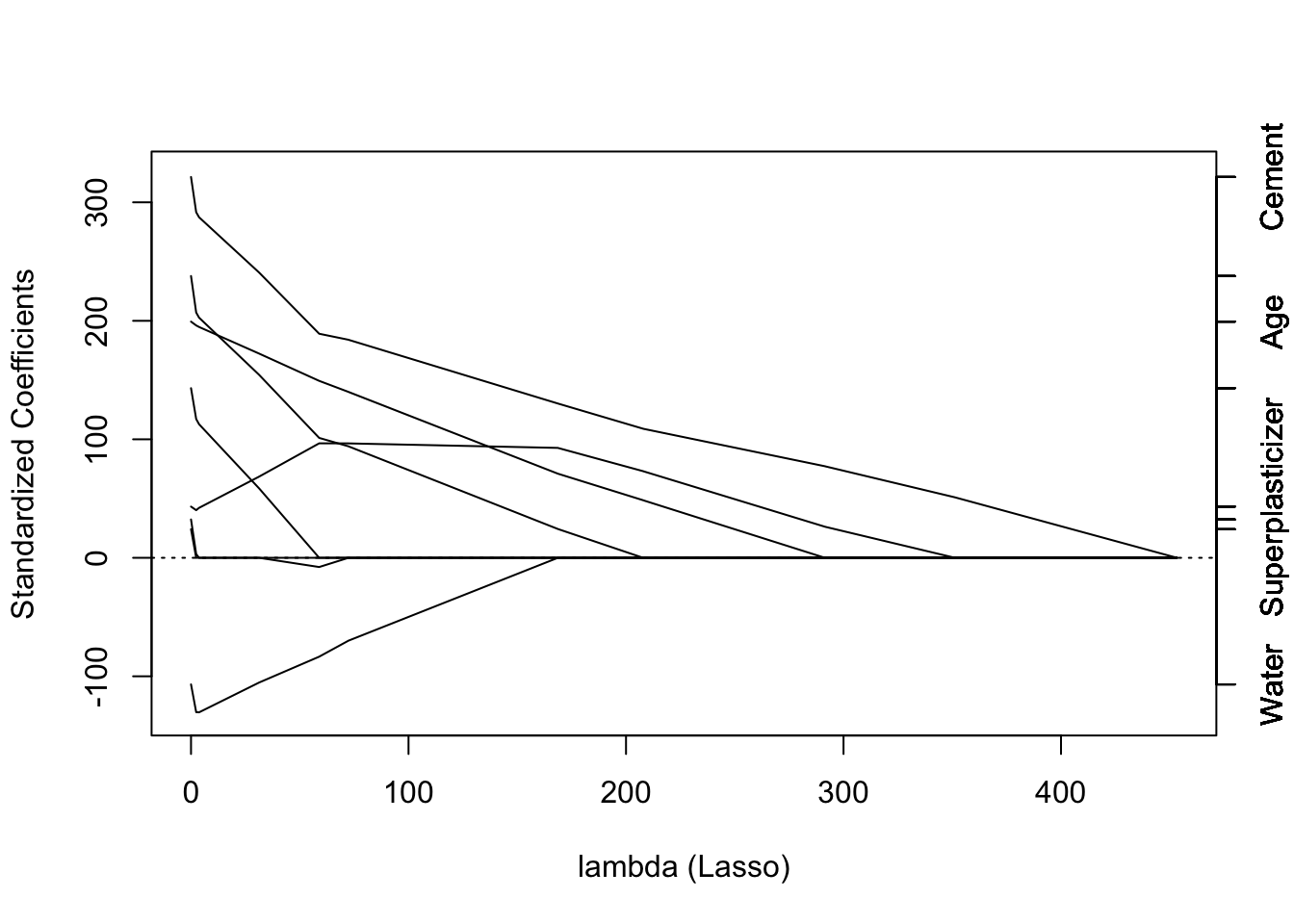
Set the seed to 233 and fit a lasso model to predict Compressive Strength. Which variable is the last coefficient to be set to zero as the penalty increases? (Hint: it may be useful to look up ?plot.enet).

set.seed(233)

Q3modLas <- train(CompressiveStrength ~., data = training, method = 'lasso')

**library**(elasticnet)

plot.enet(Q3modLas$finalModel, xvar = 'penalty')



* CoarseAggregate
* Age
* **Cement**
* Water

1. 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*

**if** (!file.exists('gaData.csv')) {

download.file('https://d396qusza40orc.cloudfront.net/predmachlearn/gaData.csv',

destfile = './gaData.csv',

method = 'curl', quiet = T)

}

dat = read.csv("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?

**library**(forecast)

Q4modFc <- bats(training$visitsTumblr)

fcast <- forecast(Q4modFc, h = nrow(testing), level = 0.95)

accuracy(fcast, testing$visitsTumblr)

## ME RMSE MAE MPE MAPE MASE

## Training set 10.49059 258.7813 40.17169 -Inf Inf 0.8425522

## Test set 174.66089 294.9901 199.92648 22.73707 40.045 4.1932146

## ACF1

## Training set 0.01240395

## Test set NA

(sum((fcast$lower < testing$visitsTumblr) & (fcast$upper > testing$visitsTumblr)))/nrow(testing)

## [1] 0.9617021

* 95%
* 94%
* **96%**
* 93%

1. 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?

**library**(e1071)

set.seed(325)

Q5modSvm <- svm(CompressiveStrength ~., data = training)

Q5predSvm <- predict(Q5modSvm, testing)

accuracy(Q5predSvm, testing$CompressiveStrength)

## ME RMSE MAE MPE MAPE

## Test set 0.1682863 6.715009 5.120835 -7.102348 19.27739

* 6.93
* 107.44
* **6.72**
* 35.59