Quiz 2

mgestal

Thursday, May 28, 2015

Question 1

Load the Alzheimer's disease data using the commands:

```
library(AppliedPredictiveModeling)
library(caret)
data(AlzheimerDisease)
```

Which of the following commands will create training and test sets with about 50% of the observations assigned to each?

Solution

```
adData = data.frame(diagnosis,predictors)
trainIndex = createDataPartition(diagnosis, p = 0.50,list=FALSE)
training = adData[trainIndex,]
testing = adData[-trainIndex,]
```

Question 2

Load the cement data using the commands:

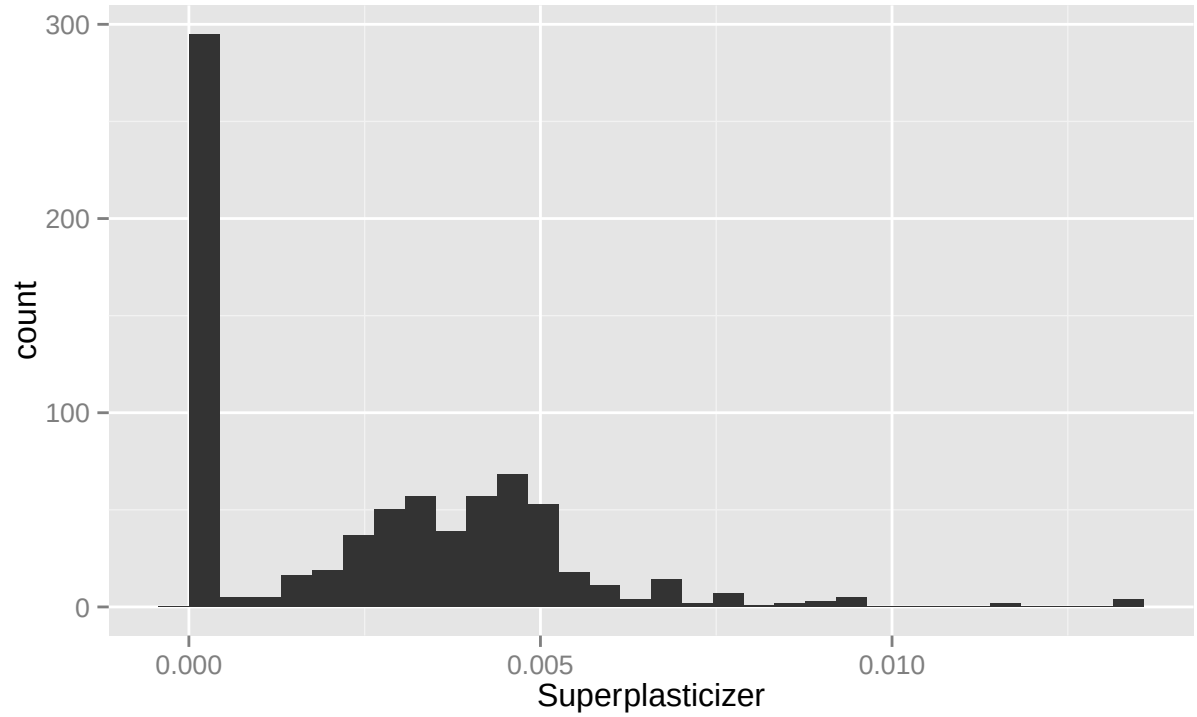
```
library(AppliedPredictiveModeling)
data(concrete)
library(caret)
set.seed(975)
inTrain = createDataPartition(mixtures$CompressiveStrength, p = 3/4)[[1]]
training = mixtures[ inTrain,]
testing = mixtures[-inTrain,]
```

Make a histogram and confirm the SuperPlasticizer variable is skewed. Normally you might use the log transform to try to make the data more symmetric. Why would that be a poor choice for this variable?

Solution

```
ggplot(data=training, aes(x=Superplasticizer)) + geom_histogram()
```

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



- **There are values of zero so when you take the log() transform those values will be -Inf.**

Question 3

Load the Alzheimer's disease data using the commands:

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

Find all the predictor variables in the training set that begin with IL. Perform principal components on these variables with the preProcess() function from the caret package. Calculate the number of principal components needed to capture 90% of the variance. How many are there?

Solution

```
IL_cols <- grep("^IL", colnames(training), value="TRUE")
preProc <- preProcess(training[, IL_cols], method="pca", thresh=0.9)
preProc$rotation
```

```
##                       PC1           PC2           PC3          PC4
## IL_11         -0.06529786  0.5555956867  0.2031317937 -0.050389599
## IL_13          0.27529157  0.3559427297 -0.0399010765  0.265076920
## IL_16          0.42079000  0.0007224953  0.0832211446 -0.082097273
## IL_17E        -0.01126118  0.5635958176  0.3744707126  0.302512329
## IL_1alpha      0.25078195 -0.0687043488 -0.3008366900  0.330945942
## IL_3           0.42026485 -0.0703352892 -0.1049647272 -0.065352774
## IL_4           0.33302031  0.0688495706 -0.1395450144  0.165631691
## IL_5           0.38706503 -0.0039619980  0.0005616126 -0.224448981
## IL_6           0.05398185 -0.4248425653  0.6090821756  0.417591202
## IL_6_Receptor  0.21218980  0.1005338329  0.2920341087 -0.659953479
## IL_7           0.32948731  0.0806070090 -0.1966471906  0.165544952
## IL_8           0.29329723 -0.1883039842  0.4405255221  0.002811187
##                       PC5          PC6         PC7         PC8
## IL_11          0.73512798 -0.102014559  0.20984151 -0.08402367
## IL_13         -0.25796332 -0.068927711  0.58942516 -0.06839401
## IL_16          0.04435883 -0.007094672 -0.06581741  0.02665034
## IL_17E        -0.38918707  0.221149380 -0.46462692  0.02185290
## IL_1alpha      0.16992452  0.742391473  0.12787035 -0.19555207
## IL_3           0.02352819 -0.165587911 -0.09006656 -0.15062164
## IL_4          -0.14268797 -0.297421293  0.19661173  0.57346657
## IL_5           0.08426042  0.153835977 -0.16425757 -0.02917286
## IL_6          -0.00165066 -0.166089521  0.21895103 -0.34568186
## IL_6_Receptor -0.29654048  0.138000448  0.22657846 -0.26274531
## IL_7           0.11373532 -0.405698338 -0.42065832 -0.40841984
## IL_8           0.28608600  0.184321013 -0.14833779  0.49101347
##                       PC9
## IL_11          0.183359387
## IL_13         -0.512677898
## IL_16         -0.225338083
## IL_17E         0.117769681
## IL_1alpha      0.256874424
## IL_3           0.014565029
## IL_4           0.591849422
## IL_5           0.003418637
## IL_6           0.221816813
## IL_6_Receptor  0.276527746
## IL_7           0.002607462
## IL_8          -0.311002624
```

```
ncol(preProc$rotation)
```

```
## [1] 9
```

Question 4

Load the Alzheimer's disease data using the commands:

```
library(caret)
library(AppliedPredictiveModeling)
set.seed(3433)
data(AlzheimerDisease)
```

```
adData = data.frame(diagnosis,predictors)
inTrain = createDataPartition(adData$diagnosis, p = 3/4)[[1]]
training = adData[ inTrain,]
testing = adData[-inTrain,]
```

Create a training data set consisting of only the predictors with variable names beginning with IL and the diagnosis. Build two predictive models, one using the predictors as they are and one using PCA with principal components explaining 80% of the variance in the predictors. Use method=“glm” in the train function. What is the accuracy of each method in the test set? Which is more accurate?

Solution

First model: GLM Second model: GLM with PCA preProc

```
IL_cols <- grep("^IL", colnames(training), value="TRUE")

training <- training[, c('diagnosis', IL_cols)]
testing <- testing[, c('diagnosis', IL_cols)]

# GLM with IL_* predictors

modelGLM <- train(diagnosis ~ ., method="glm", data=training)

predictionsGLM <- predict(modelGLM, newdata=testing)
confusionMatrix(predictionsGLM, testing$diagnosis)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Impaired Control
##   Impaired        2       9
##   Control        20      51
##
##                Accuracy : 0.6463
##                  95% CI : (0.533, 0.7488)
##     No Information Rate : 0.7317
##     P-Value [Acc > NIR] : 0.96637
##
##                   Kappa : -0.0702
##  Mcnemar's Test P-Value : 0.06332
##
##             Sensitivity : 0.09091
##             Specificity : 0.85000
##          Pos Pred Value : 0.18182
##          Neg Pred Value : 0.71831
##              Prevalence : 0.26829
##          Detection Rate : 0.02439
##    Detection Prevalence : 0.13415
##       Balanced Accuracy : 0.47045
##
##        'Positive' Class : Impaired
##
```

```r
# Non-PCA Accuracy
confusionMatrix(predictionsGLM, testing$diagnosis)$overall[1]
```

```
##  Accuracy
## 0.6463415
```

```r
# GLM with PCA to explain the 80% variance of the predictors

modelGLM_PCA <- train(diagnosis ~ ., method="glm", data=training,
                      preProcess = "pca",
                      trControl = trainControl(preProcOptions = list(thresh = 0.8)))

predictionsGLM_PCA <- predict(modelGLM_PCA, newdata=testing)
confusionMatrix(predictionsGLM_PCA, testing$diagnosis)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Impaired Control
##   Impaired        3       4
##   Control        19      56
##
##                Accuracy : 0.7195
##                  95% CI : (0.6094, 0.8132)
##     No Information Rate : 0.7317
##     P-Value [Acc > NIR] : 0.651780
##
##                   Kappa : 0.0889
##  Mcnemar's Test P-Value : 0.003509
##
##             Sensitivity : 0.13636
##             Specificity : 0.93333
##          Pos Pred Value : 0.42857
##          Neg Pred Value : 0.74667
##              Prevalence : 0.26829
##          Detection Rate : 0.03659
##    Detection Prevalence : 0.08537
##       Balanced Accuracy : 0.53485
##
##        'Positive' Class : Impaired
##
```

```r
# PCA Accuracy
confusionMatrix(predictionsGLM_PCA, testing$diagnosis)$overall[1]
```

```
##  Accuracy
## 0.7195122
```