Practical Machine Learning - Week 2 Quiz

Ken Wood

9/16/2020

```
library(caret)
```

Question 1

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

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

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

What are the set of commands that will create non-overlapping training and test sets with about 50% of the observations assigned to each?

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

```
library(AppliedPredictiveModeling)
data(concrete)
set.seed(1000)
inTrain = createDataPartition(mixtures$CompressiveStrength, p = 3/4)[[1]]
training = mixtures[ inTrain,]
testing = mixtures[-inTrain,]
```

```
library(Hmisc)
```

Question 2

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

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

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

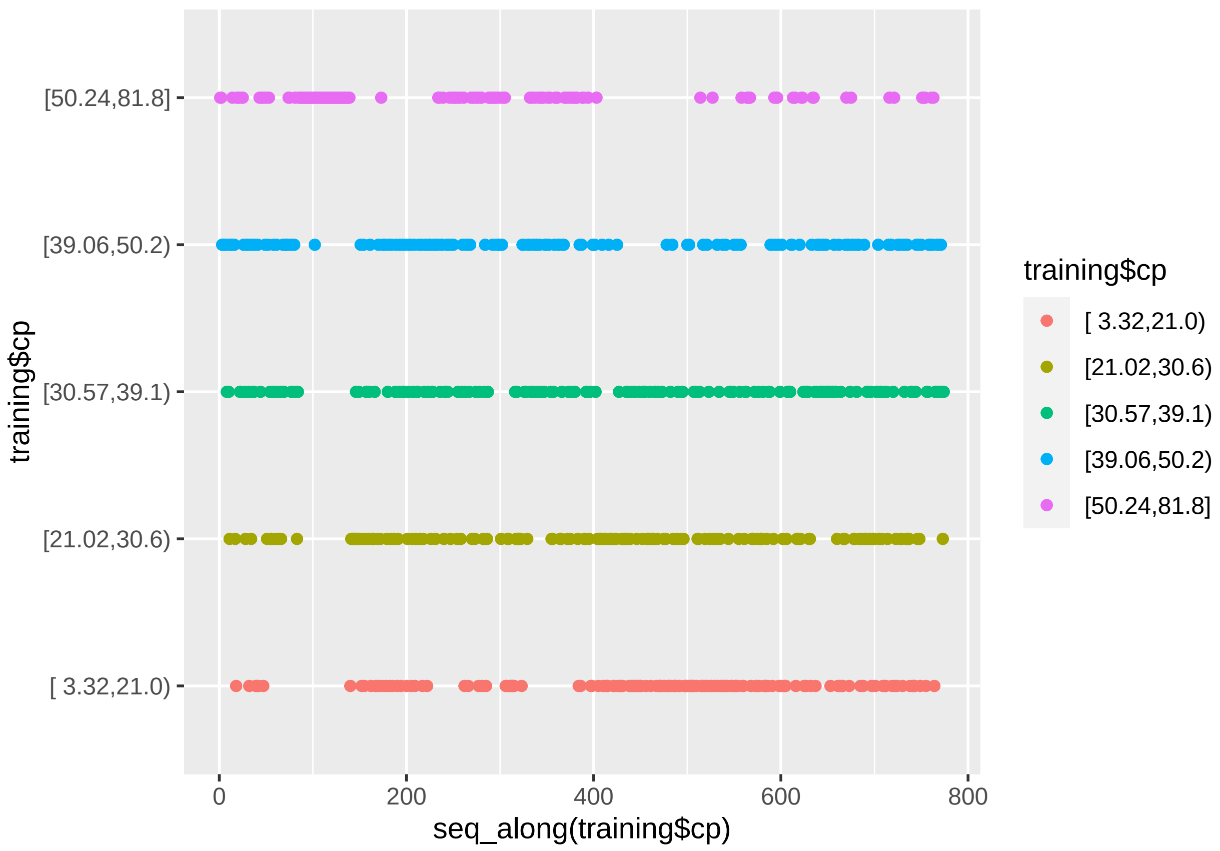
```
## Loading required package: Formula
```

```
##
## Attaching package: 'Hmisc'
```

```
## The following objects are masked from 'package:base':
```

```
##
##     format.pval, units
```

```
training$cp <- cut2(training$CompressiveStrength)
qplot(seq_along(training$cp), training$cp, color = training$cp)
```

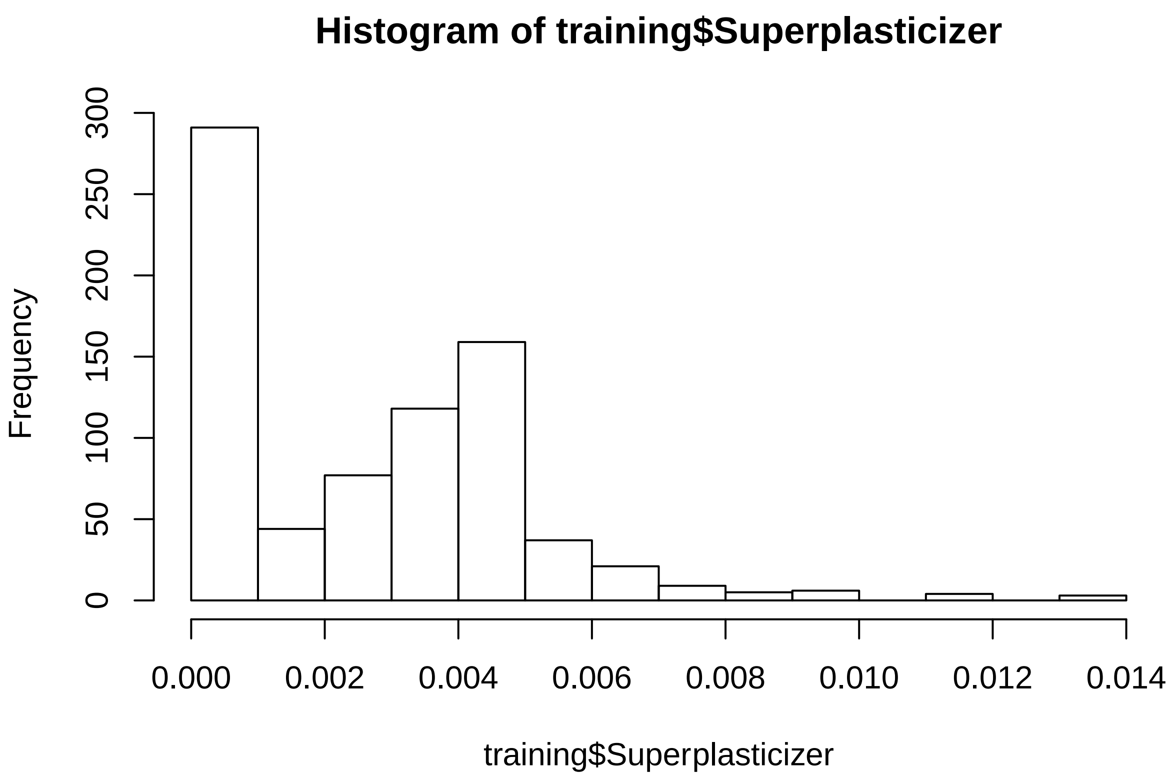


Answer: There is a non-random pattern in the plot of the outcome versus index that does not appear to be perfectly explained by any predictor suggesting a variable may be missing.

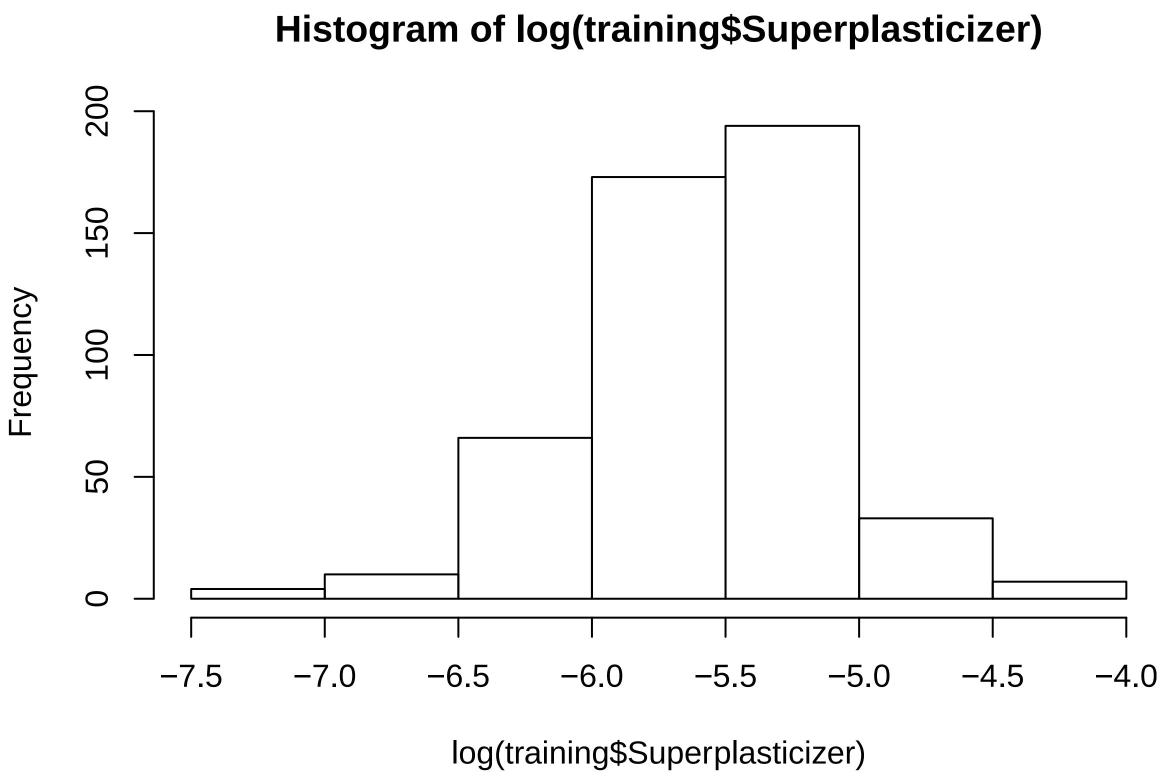
```
library(AppliedPredictiveModeling)
data(concrete)
library(caret)
set.seed(1000)
inTrain = createDataPartition(mixtures$CompressiveStrength, p = 3/4)[[1]]
training = mixtures[ inTrain,]
testing = mixtures[-inTrain,]
```

Question 3 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?

```
hist(training$Superplasticizer)
```



```r
hist(log(training$Superplasticizer))
```



There are 288 zero values in Superplasticizer. So we cannot apply a log transform to it.

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

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

Question 4 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?

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:Hmisc':
##
##     src, summarize

## The following objects are masked from 'package:stats':
##
##     filter, lag

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

```r
df <- training %>% dplyr:: select(starts_with("IL"))
result <- preProcess(df,method = 'pca', thresh = 0.9)
result$numComp
```

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

```r
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,]
```

Question 5 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?

```r
set.seed(3433)
trainingIL <- training[,grep("^IL|diagnosis", names(training))]
testingIL <- testing[,grep("^IL|diagnosis", names(testing))]
```

Non-PCA fit...

```r
set.seed(3433)
fit <- train(diagnosis~., data=trainingIL, method="glm")
pred <- predict(fit, testingIL)
cm <- confusionMatrix(pred,testingIL$diagnosis)
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Impaired Control
##   Impaired        4       2
##   Control        18      58
##
##                Accuracy : 0.7561
##                  95% CI : (0.6488, 0.8442)
##     No Information Rate : 0.7317
##     P-Value [Acc > NIR] : 0.3606488
##
##                   Kappa : 0.1929
##
##  Mcnemar's Test P-Value : 0.0007962
##
##             Sensitivity : 0.18182
##             Specificity : 0.96667
##          Pos Pred Value : 0.66667
##          Neg Pred Value : 0.76316
##              Prevalence : 0.26829
##          Detection Rate : 0.04878
##    Detection Prevalence : 0.07317
##       Balanced Accuracy : 0.57424
##
##        'Positive' Class : Impaired
##
```

PCA fit

```r
fitPC <- train(diagnosis~., method="glm", data=trainingIL, preProcess="pca", trControl = trainControl(p
predPC <- predict(fitPC, testingIL)
cmPCA <- confusionMatrix(predPC, testingIL$diagnosis)
cmPCA
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Impaired Control
##   Impaired        1       2
##   Control        21      58
##
##                Accuracy : 0.7195
##                  95% CI : (0.6094, 0.8132)
##     No Information Rate : 0.7317
##     P-Value [Acc > NIR] : 0.6517805
##
##                   Kappa : 0.0167
##
```

```
##  Mcnemar's Test P-Value : 0.0001746
##
##             Sensitivity : 0.04545
##             Specificity : 0.96667
##          Pos Pred Value : 0.33333
##          Neg Pred Value : 0.73418
##              Prevalence : 0.26829
##          Detection Rate : 0.01220
##    Detection Prevalence : 0.03659
##       Balanced Accuracy : 0.50606
##
##        'Positive' Class : Impaired
##
```