Quiz2 Machine Learning

Fatemeh Abyarjoo

Question 1:

```
# install.packages("AppliedPredictiveModeling")
# install.packages("ggplot2")
# install.packages("lattice")
library(lattice)
library(ggplot2)
library(AppliedPredictiveModeling)
library(caret)
data(AlzheimerDisease)

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

Question 2: Load the cement data. 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?

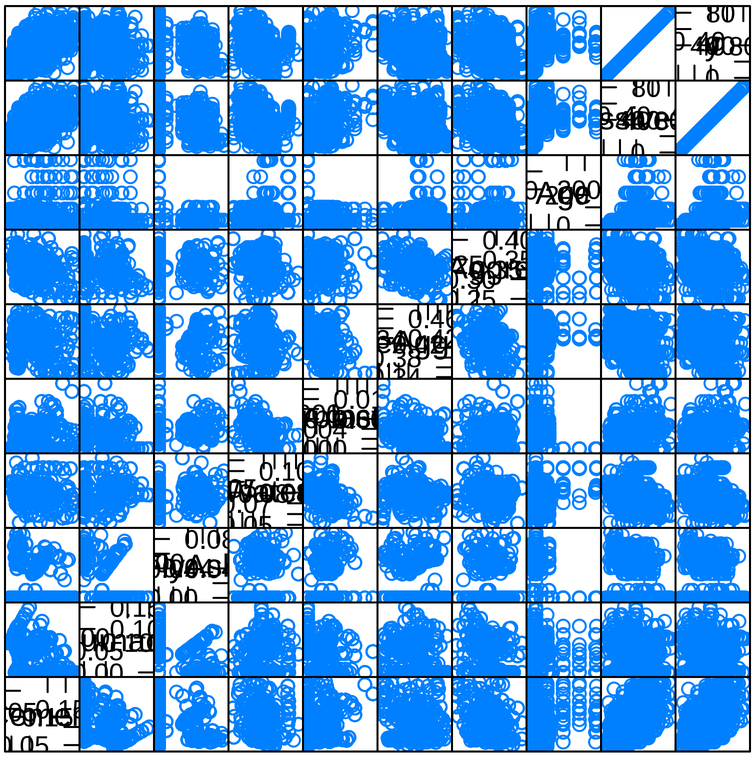
```
library(AppliedPredictiveModeling)
data(concrete)
library(caret)
set.seed(1000)
inTrain = createDataPartition(mixtures$CompressiveStrength, p = 3/4)[[1]]
training = mixtures[ inTrain,]
testing = mixtures[-inTrain,]

names(training)
```

```
## [1] "Cement"              "BlastFurnaceSlag"    "FlyAsh"
## [4] "Water"               "Superplasticizer"    "CoarseAggregate"
## [7] "FineAggregate"       "Age"                 "CompressiveStrength"
```

```
summary(training)
```

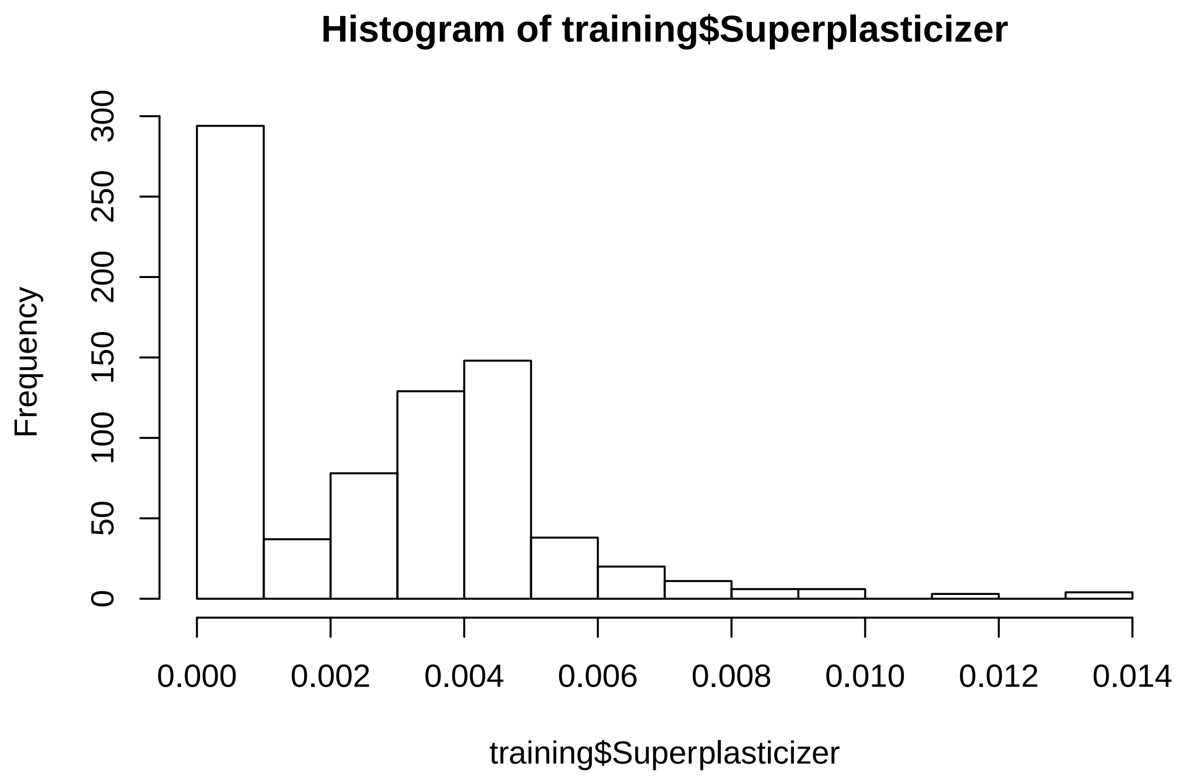
```
##      Cement        BlastFurnaceSlag       FlyAsh            Water
##  Min.   :0.04482   Min.   :0.000000   Min.   :0.00000   Min.   :0.05139
##  1st Qu.:0.08179   1st Qu.:0.000000   1st Qu.:0.00000   1st Qu.:0.06972
##  Median :0.11462   Median :0.009993   Median :0.00000   Median :0.07862
##  Mean   :0.11782   Mean   :0.032051   Mean   :0.02247   Mean   :0.07774
##  3rd Qu.:0.14793   3rd Qu.:0.061968   3rd Qu.:0.04999   3rd Qu.:0.08384
##  Max.   :0.22541   Max.   :0.150339   Max.   :0.08884   Max.   :0.11222
##  Superplasticizer   CoarseAggregate  FineAggregate        Age
##  Min.   :0.000000   Min.   :0.3459   Min.   :0.2480   Min.   :  1.00
##  1st Qu.:0.000000   1st Qu.:0.3986   1st Qu.:0.3113   1st Qu.: 14.00
##  Median :0.002726   Median :0.4213   Median :0.3305   Median : 28.00
##  Mean   :0.002608   Mean   :0.4167   Mean   :0.3306   Mean   : 47.46
##  3rd Qu.:0.004351   3rd Qu.:0.4389   3rd Qu.:0.3542   3rd Qu.: 56.00
##  Max.   :0.013149   Max.   :0.4798   Max.   :0.4141   Max.   :365.00
##  CompressiveStrength
##  Min.   : 2.33
##  1st Qu.:23.71
##  Median :34.48
##  Mean   :35.64
##  3rd Qu.:46.13
##  Max.   :82.60
```

```
# install.packages("ISLR")
library(ISLR)
featurePlot(x = training[,c("Cement","BlastFurnaceSlag","FlyAsh","Water",
                            "Superplasticizer","CoarseAggregate","FineAggregate",
                            "Age","CompressiveStrength")], y = training$CompressiveStrength,
            plot = "pairs")
```

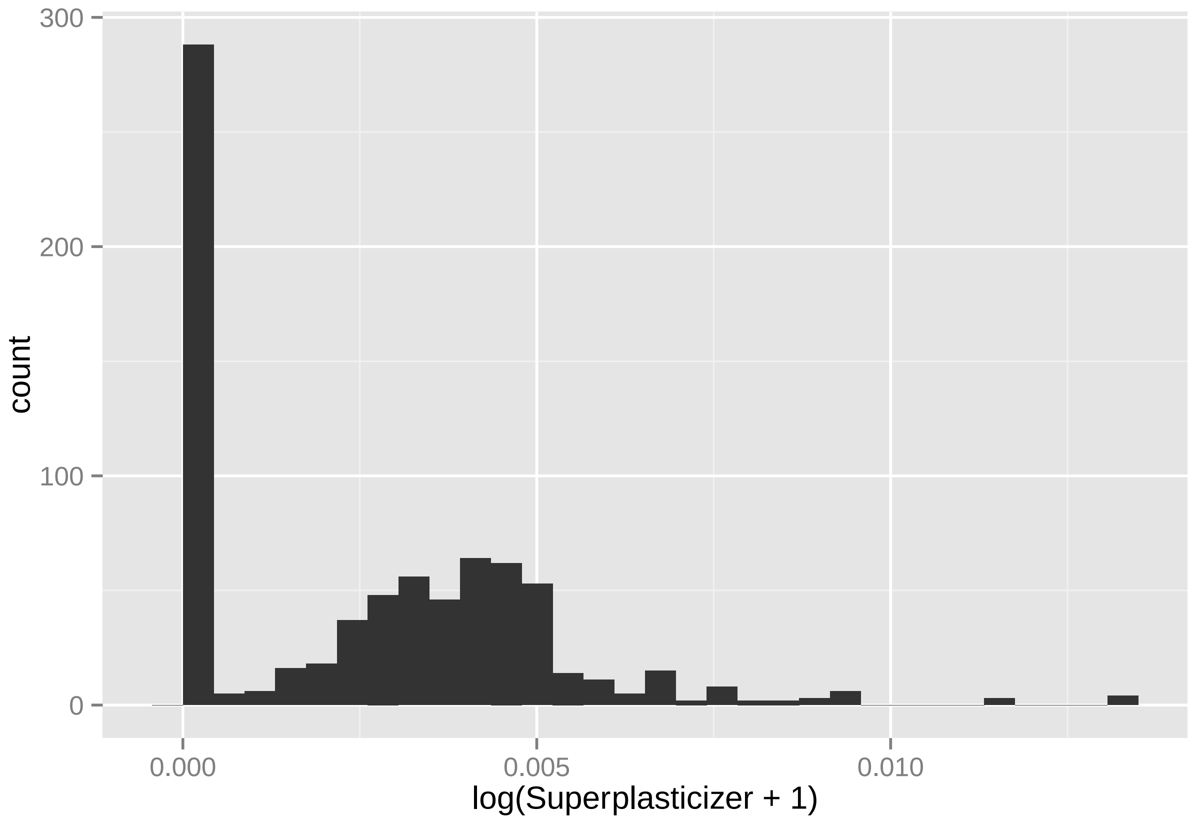


Scatter Plot Matrix

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



```r
qplot(log(Superplasticizer+1),data=training)
```



There are a large number of values that are the same and even if you took the log(SuperPlasticizer + 1) they would still all be identical so the distribution would not be symmetric.

Question 3 Load the Alzheimer's disease data.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?

```
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,]

names(training)
```

```
##   [1] "diagnosis"                          "ACE_CD143_Angiotensin_Converti"
##   [3] "ACTH_Adrenocorticotropic_Hormon"  "AXL"
##   [5] "Adiponectin"                        "Alpha_1_Antichymotrypsin"
##   [7] "Alpha_1_Antitrypsin"                "Alpha_1_Microglobulin"
##   [9] "Alpha_2_Macroglobulin"              "Angiopoietin_2_ANG_2"
##  [11] "Angiotensinogen"                    "Apolipoprotein_A_IV"
##  [13] "Apolipoprotein_A1"                  "Apolipoprotein_A2"
##  [15] "Apolipoprotein_B"                   "Apolipoprotein_CI"
##  [17] "Apolipoprotein_CIII"               "Apolipoprotein_D"
##  [19] "Apolipoprotein_E"                  "Apolipoprotein_H"
##  [21] "B_Lymphocyte_Chemoattractant_BL"  "BMP_6"
##  [23] "Beta_2_Microglobulin"              "Betacellulin"
##  [25] "C_Reactive_Protein"                "CD40"
##  [27] "CD5L"                              "Calbindin"
##  [29] "Calcitonin"                        "CgA"
##  [31] "Clusterin_Apo_J"                   "Complement_3"
##  [33] "Complement_Factor_H"               "Connective_Tissue_Growth_Factor"
##  [35] "Cortisol"                          "Creatine_Kinase_MB"
##  [37] "Cystatin_C"                        "EGF_R"
##  [39] "EN_RAGE"                           "ENA_78"
##  [41] "Eotaxin_3"                         "FAS"
##  [43] "FSH_Follicle_Stimulation_Hormon"  "Fas_Ligand"
##  [45] "Fatty_Acid_Binding_Protein"        "Ferritin"
##  [47] "Fetuin_A"                          "Fibrinogen"
##  [49] "GRO_alpha"                         "Gamma_Interferon_induced_Monokin"
##  [51] "Glutathione_S_Transferase_alpha"  "HB_EGF"
##  [53] "HCC_4"                             "Hepatocyte_Growth_Factor_HGF"
##  [55] "I_309"                             "ICAM_1"
##  [57] "IGF_BP_2"                          "IL_11"
##  [59] "IL_13"                             "IL_16"
##  [61] "IL_17E"                            "IL_1alpha"
##  [63] "IL_3"                              "IL_4"
##  [65] "IL_5"                              "IL_6"
##  [67] "IL_6_Receptor"                     "IL_7"
##  [69] "IL_8"                              "IP_10_Inducible_Protein_10"
##  [71] "IgA"                               "Insulin"
##  [73] "Kidney_Injury_Molecule_1_KIM_1"   "LOX_1"
##  [75] "Leptin"                            "Lipoprotein_a"
##  [77] "MCP_1"                             "MCP_2"
##  [79] "MIF"                               "MIP_1alpha"
##  [81] "MIP_1beta"                         "MMP_2"
##  [83] "MMP_3"                             "MMP10"
##  [85] "MMP7"                              "Myoglobin"
##  [87] "NT_proBNP"                         "NrCAM"
##  [89] "Osteopontin"                       "PAI_1"
##  [91] "PAPP_A"                            "PLGF"
##  [93] "PYY"                               "Pancreatic_polypeptide"
##  [95] "Prolactin"                         "Prostatic_Acid_Phosphatase"
##  [97] "Protein_S"                         "Pulmonary_and_Activation_Regulat"
##  [99] "RANTES"                            "Resistin"
## [101] "S100b"                             "SGOT"
## [103] "SHBG"                              "SOD"
## [105] "Serum_Amyloid_P"                   "Sortilin"
## [107] "Stem_Cell_Factor"                  "TGF_alpha"
## [109] "TIMP_1"                            "TNF_RII"
## [111] "TRAIL_R3"                          "TTR_prealbumin"
## [113] "Tamm_Horsfall_Protein_THP"         "Thrombomodulin"
## [115] "Thrombopoietin"                    "Thymus_Expressed_Chemokine_TECK"
## [117] "Thyroid_Stimulating_Hormone"       "Thyroxine_Binding_Globulin"
## [119] "Tissue_Factor"                     "Transferrin"
## [121] "Trefoil_Factor_3_TFF3"             "VCAM_1"
## [123] "VEGF"                              "Vitronectin"
## [125] "von_Willebrand_Factor"            "age"
## [127] "tau"                              "p_tau"
## [129] "Ab_42"                            "male"
## [131] "Genotype"
```

```
ILset=grep("^IL", names(training), value = TRUE)

Nofcom =preProcess(training[, ILset], method = "pca", thresh = 0.9)
Nofcom
```

```
##
## Call:
## preProcess.default(x = training[, ILset], method = "pca", thresh = 0.9)
##
## Created from 251 samples and 12 variables
## Pre-processing: principal component signal extraction, scaled, centered
##
## PCA needed 9 components to capture 90 percent of the variance
```

Question 4: Load the Alzheimer's disease data.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?

```
library(lattice)
library(ggplot2)
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,]

set.seed(3433)
ILset=grep("^IL", names(training), value = TRUE)
ILpredictor= predictors[, ILset]
dataset= data.frame(diagnosis, ILpredictor)
inTrain = createDataPartition(dataset$diagnosis, p = 3/4)[[1]]
training = dataset[inTrain, ]
testing = dataset[-inTrain, ]

# install.packages("Hmisc")
# install.packages("survival")
# install.packages("gridExtra")
# install.packages("dplyr")
# install.packages('e1071', dependencies=TRUE)

library(gridExtra)
```

```
## Loading required package: grid
```

```r
library(survival)
```

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

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
##
## The following object is masked from 'package:stats':
##
##     filter
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(Hmisc)
```

```
## Loading required package: Formula
##
## Attaching package: 'Hmisc'
##
## The following objects are masked from 'package:dplyr':
##
##     combine, src, summarize
##
## The following objects are masked from 'package:base':
##
##     format.pval, round.POSIXt, trunc.POSIXt, units
```

```r
Mod1=train(diagnosis ~ ., method = "glm", data = training)
predictions=predict(Mod1, newdata = testing)
Confusionmat1= confusionMatrix(predictions, testing$diagnosis)
print(Confusionmat1)
```

```
## 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
##
```

```
Mod2=train(training$diagnosis ~ ., method = "glm", preProcess = "pca",
   data = training, trControl = trainControl(preProcOptions = list(thresh = 0.8)))
Confusionmat2=confusionMatrix(testing$diagnosis, predict(Mod2, testing))
print(Confusionmat2)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Impaired Control
##   Impaired        3      19
##   Control         4      56
##
##                Accuracy : 0.7195
##                  95% CI : (0.6094, 0.8132)
##     No Information Rate : 0.9146
##     P-Value [Acc > NIR] : 1.000000
##
##                   Kappa : 0.0889
##  Mcnemar's Test P-Value : 0.003509
##
##             Sensitivity : 0.42857
##             Specificity : 0.74667
##          Pos Pred Value : 0.13636
##          Neg Pred Value : 0.93333
##              Prevalence : 0.08537
##          Detection Rate : 0.03659
##    Detection Prevalence : 0.26829
##       Balanced Accuracy : 0.58762
##
##        'Positive' Class : Impaired
##
```