Michigan Technological University, Computer Science

MA 5790 Combined Section - Predictive Modeling Assignment 1

Anil Silwal

December 1, 2018

3.1 (a) Using visualizations, explore the predictor variables to understand their distributions as well as the relationships between predictors

Solution:

Approach for understanding distribution between predictors:
To understand the distribution, histogram/ box plot is used. I here used histogram to get the distribution of predictors.

Result:

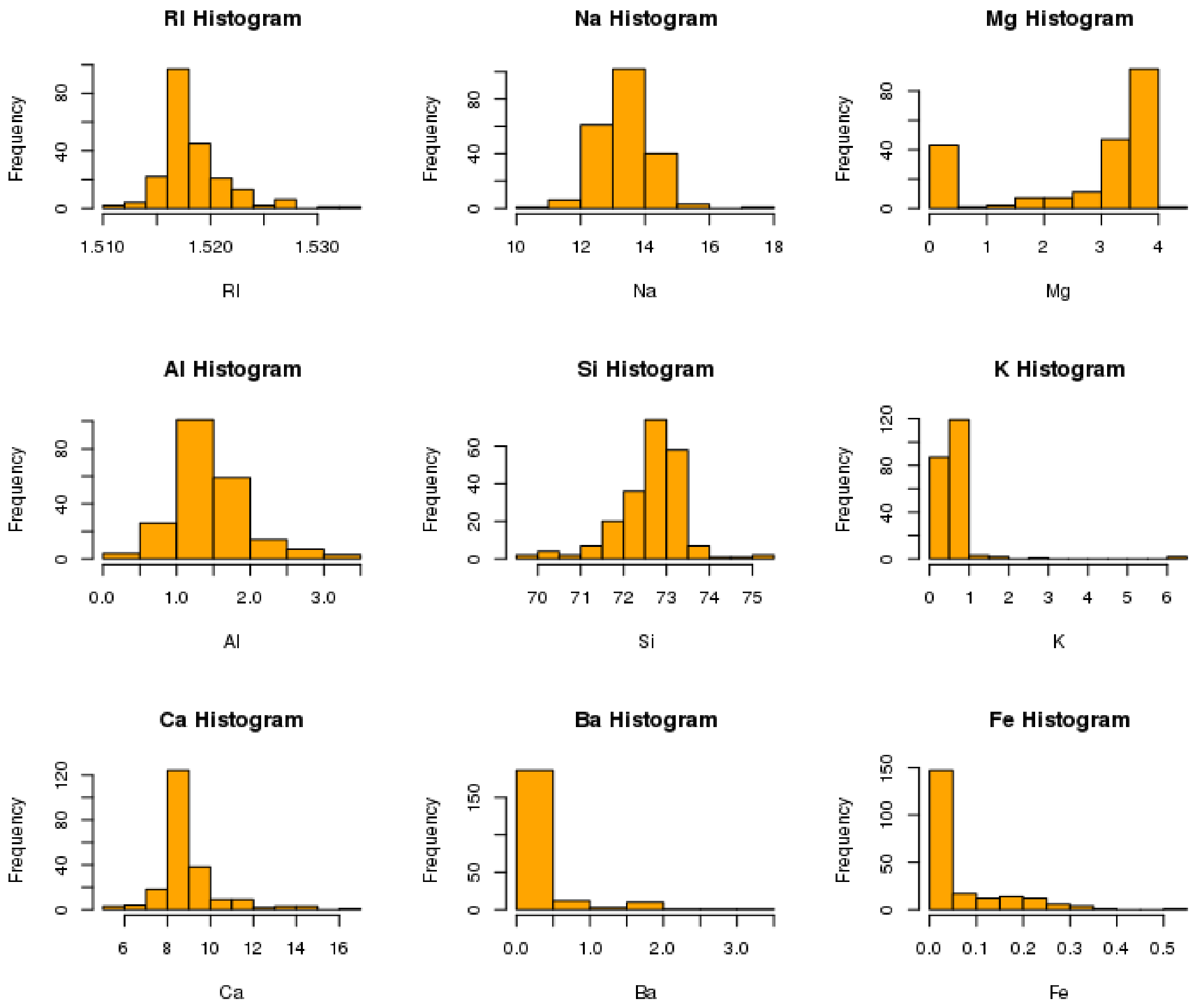


Figure 1: Histogram showing distribution of predictors

Analysis:
Comparing the histogram results for different 9 predictor, I found that RI, Na, Al, and Si are distributed normally while rest of others are not.

Approach for understanding relationships between predictors:
To understand the relationship between the predictors, scatter plot is used. So I have used the scatter plot to get the relationship between each of the predictors. But I used corrplot library to show the relationship of elements in even better way

Result:

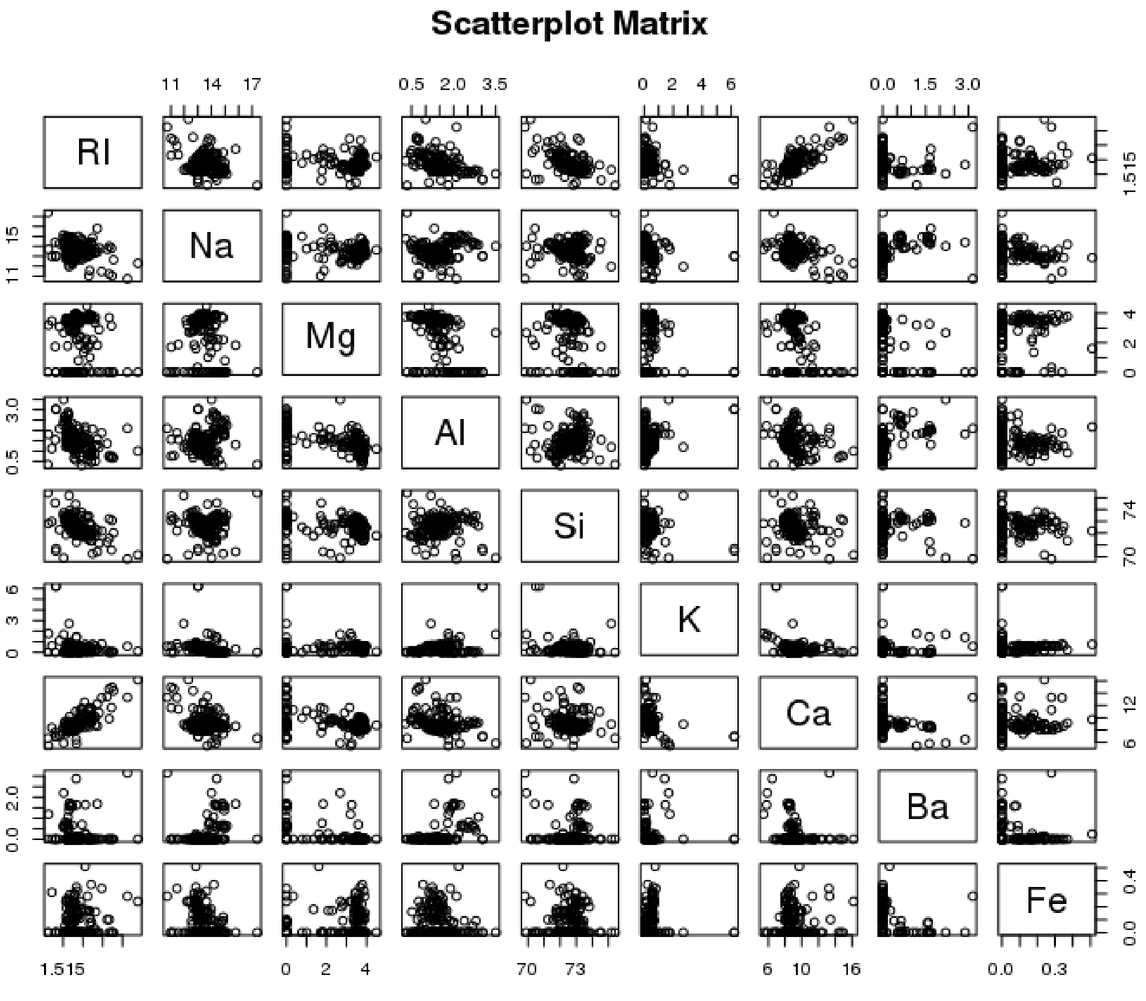


Figure 2: Scatterplot showing distribution of predictors

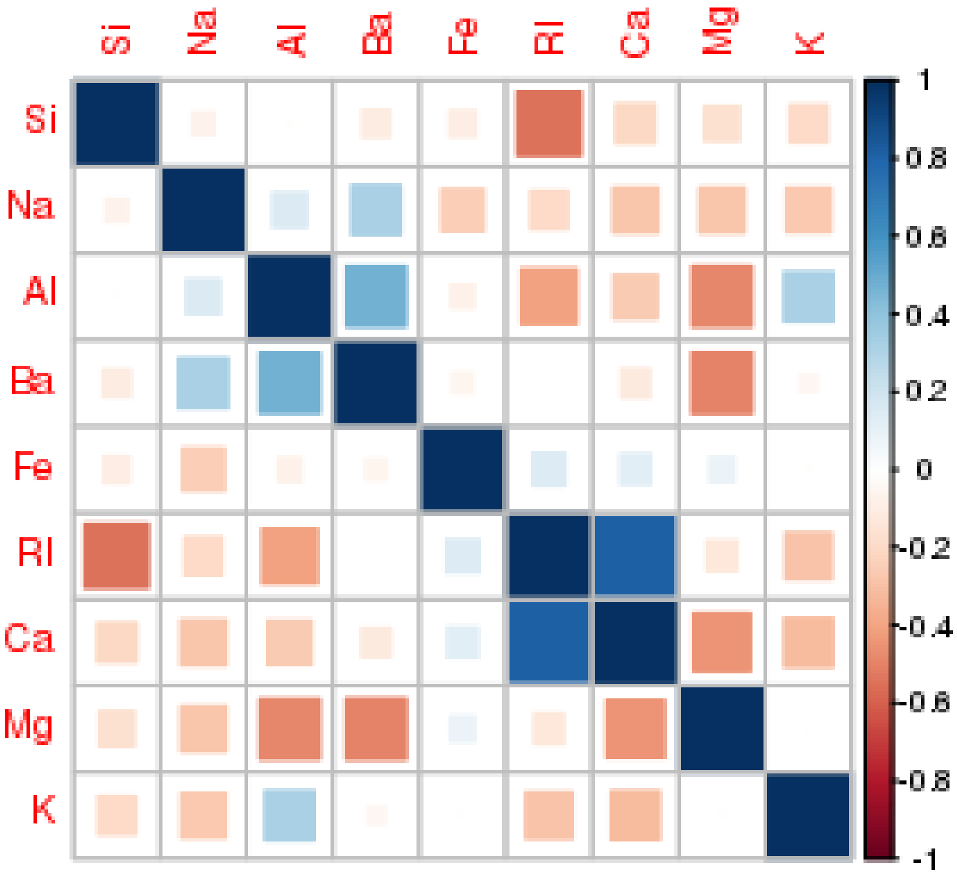


Figure 3: Heatmap showing correlation of independent variables

Analysis:
Comparing the **scatter plot** results for different nine predictor, I found that RI and Ca shows positive relationship. I found difficult to predict the relation of other variable rather than RI and Ca, so I used another clearly understanding visual tool, heatmap for this.
Comparing the **heatmap** of correlation of independent variables, I found even in clear visual representation that RI and Ca shows positive relation, indicated by **dark blue rectangular box**. And RI and Si, Mg and Al, Mg and Ba shows negative relation, indicated by **orange rectangular box**.

3.1 (b) Do there appear to be any outliers in the data? Are any predictors skewed?

Solution
Approach:
To know about the outliers, boxplot is a good data visualization tool. This boxplot has a box which contains 1st,2nd and 3rd quartile i.e 25 percent to 75 percent. Outside the box, there is a line called whisker and outside the whisker if any data appears, it is called **outliers**. The equal sized **whisker**, extended line from box on both sides, shows symmetric distribution. If one side has long whisker and another side has short, then those distribution are skewed.

Result:

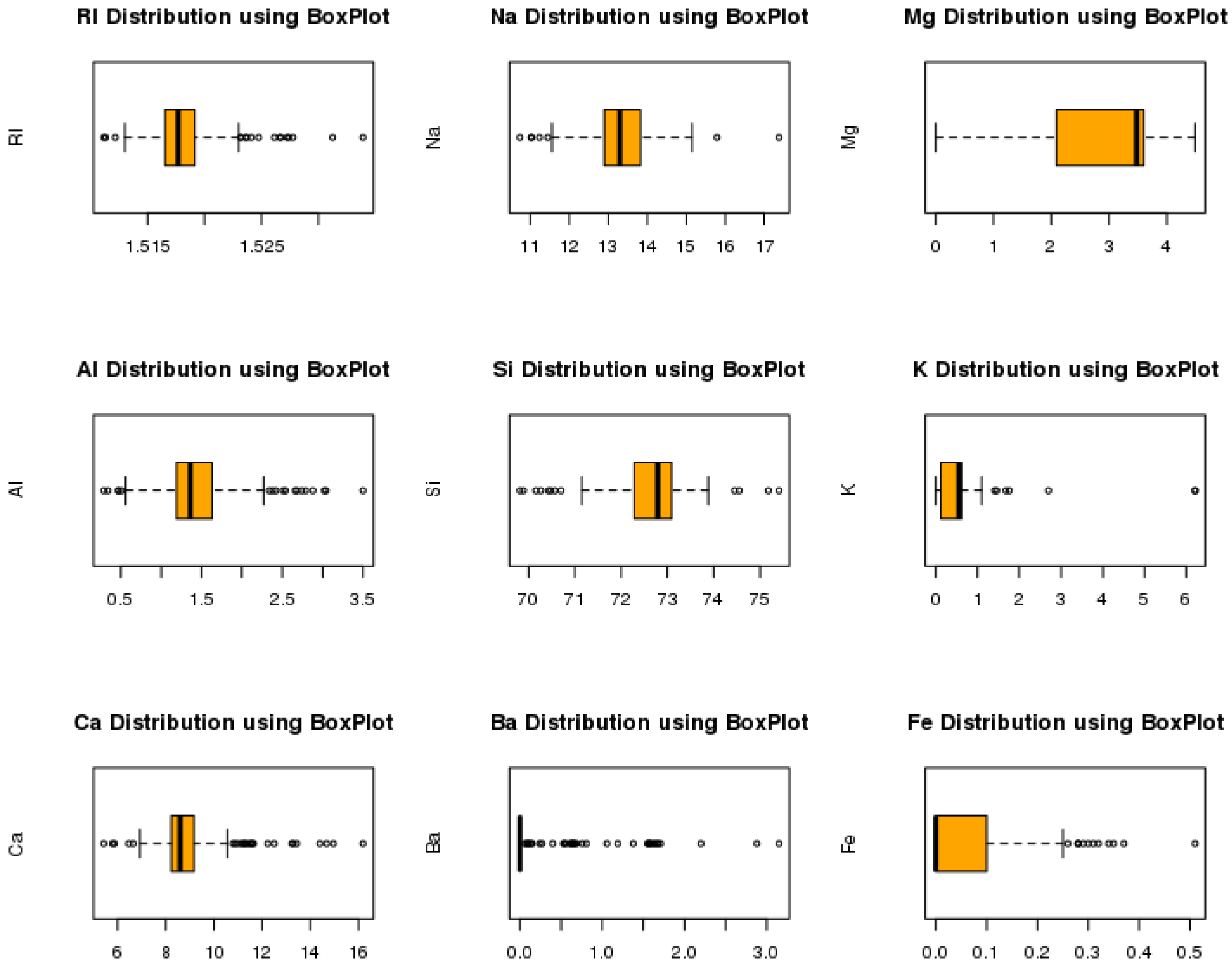


Figure 4: BoxPlot showing distribution of predictors

Analysis:
Considering **outliers**, when I compare the boxplots of each elements I found that Ba, and K has comparatively more outliers. Among these elements, I found Mg, K, Ba, Fe has skewed distribution.

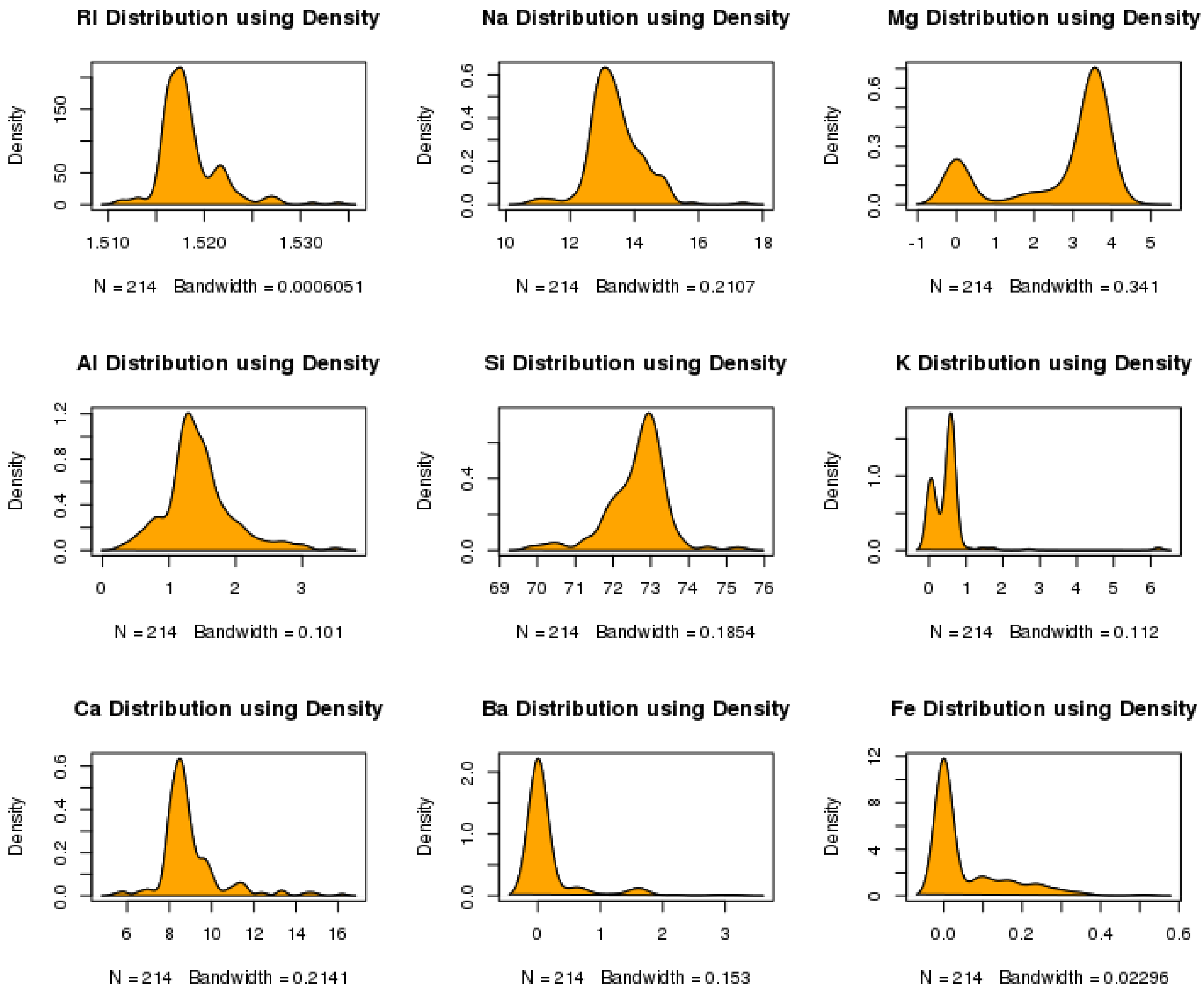
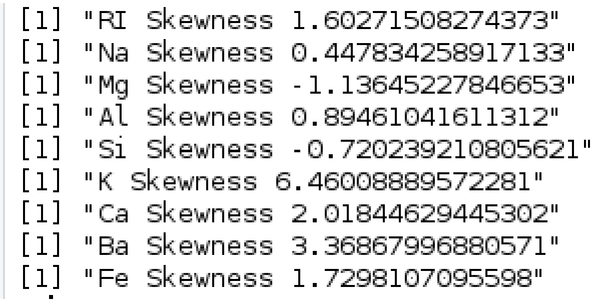


Figure 5: Density showing distribution of predictors

Analysis
Considering **skewness**, when I compare the density plots of each elements, it can be seen that Mg is left-skewed, while K, Ba, and Fe are right skewed. I used the caret library to get the skewness value of given elements and those are:



```
[1] "RI Skewness 1.60271508274373"
[1] "Na Skewness 0.447834258917133"
[1] "Mg Skewness -1.13645227846653"
[1] "Al Skewness 0.89461041611312"
[1] "Si Skewness -0.720239210805621"
[1] "K Skewness 6.46008889572281"
[1] "Ca Skewness 2.01844629445302"
[1] "Ba Skewness 3.36867996880571"
[1] "Fe Skewness 1.7298107095598"
```

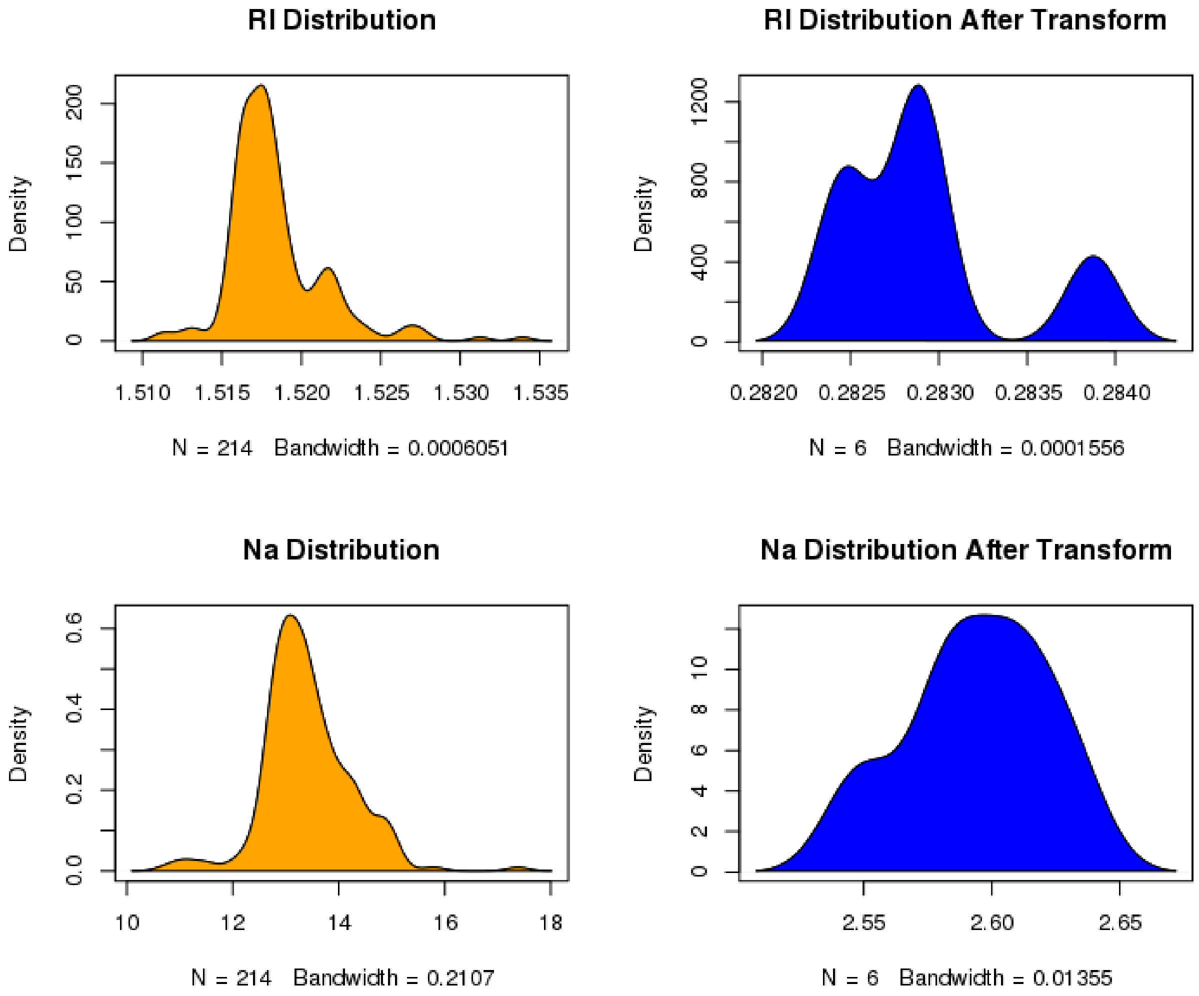
3.1(c) Are there any relevant transformations of one or more predictors that might improve the classification model?

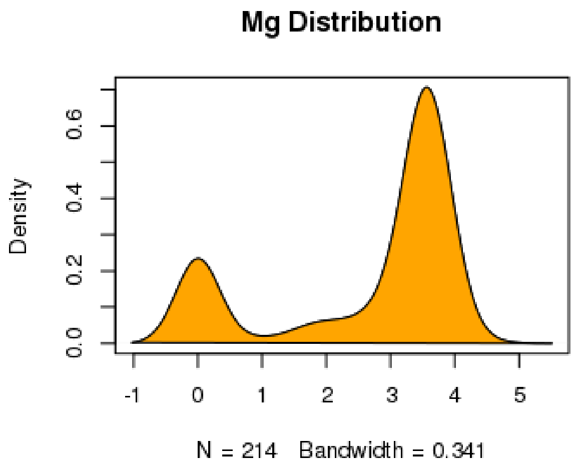
Solution:

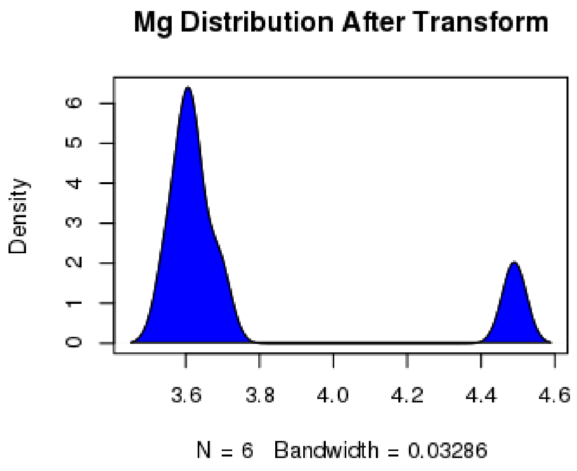
Approach

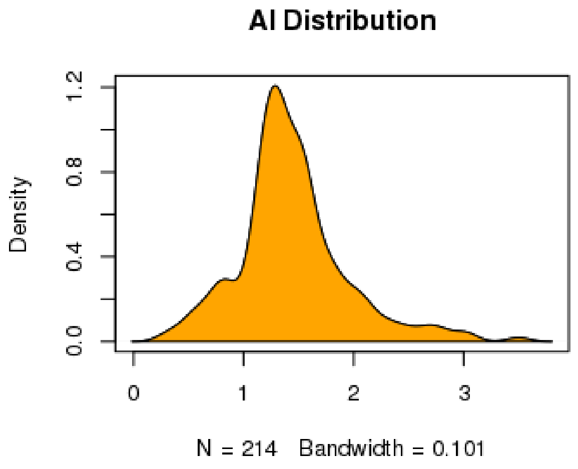
To remove the left skewness and right skewness, there are various approaches to do that. One approach is square root, cube root and log transform. Another approach is BoxCox transformation, which calculates the lamda λ and uses this to find the new transformed variable. I used BoxCox transformation because it normalizes the data and tries to distribute the data normally. For this, I am using library(caret) that has the boxcoxtransform function which calculates the λ and then I used $predict$ function which calculates the new transformed value of every untransformed value using the λ.

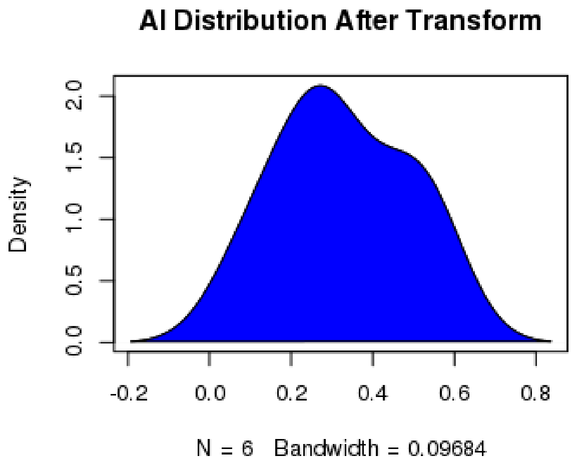
Result

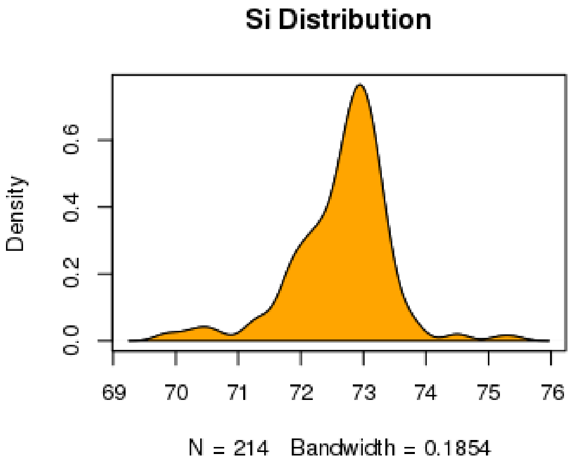


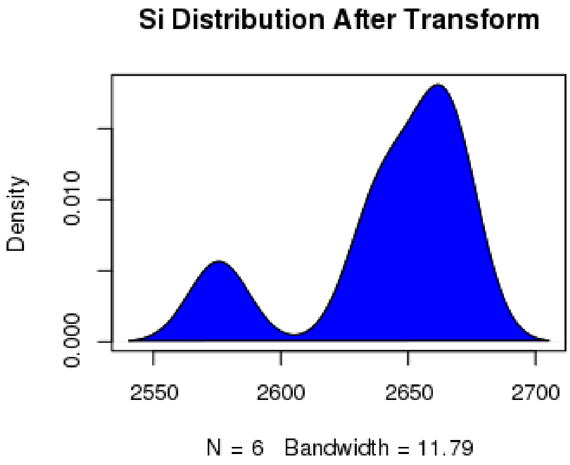

Mg Distribution
Density
0.0
0.2
0.4
0.6
-1
0
1
2
3
4
5
N = 214 Bandwidth = 0.341

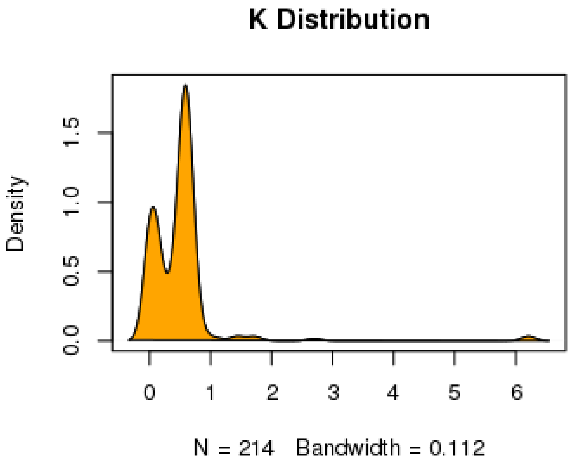

Mg Distribution After Transform
Density
0
1
2
3
4
5
6
3.6
3.8
4.0
4.2
4.4
4.6
N = 6 Bandwidth = 0.03286

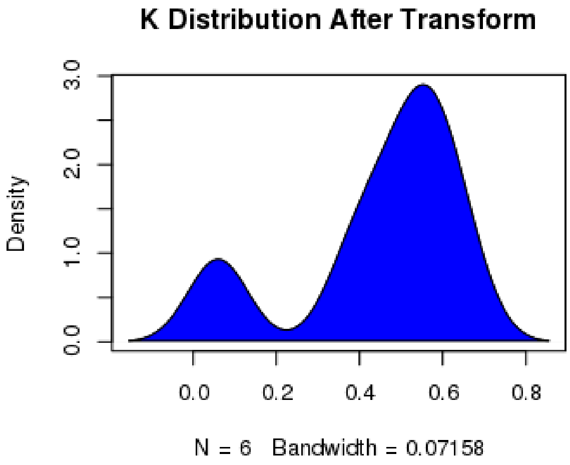

Al Distribution
Density
0.0
0.4
0.8
1.2
0
1
2
3
N = 214 Bandwidth = 0.101

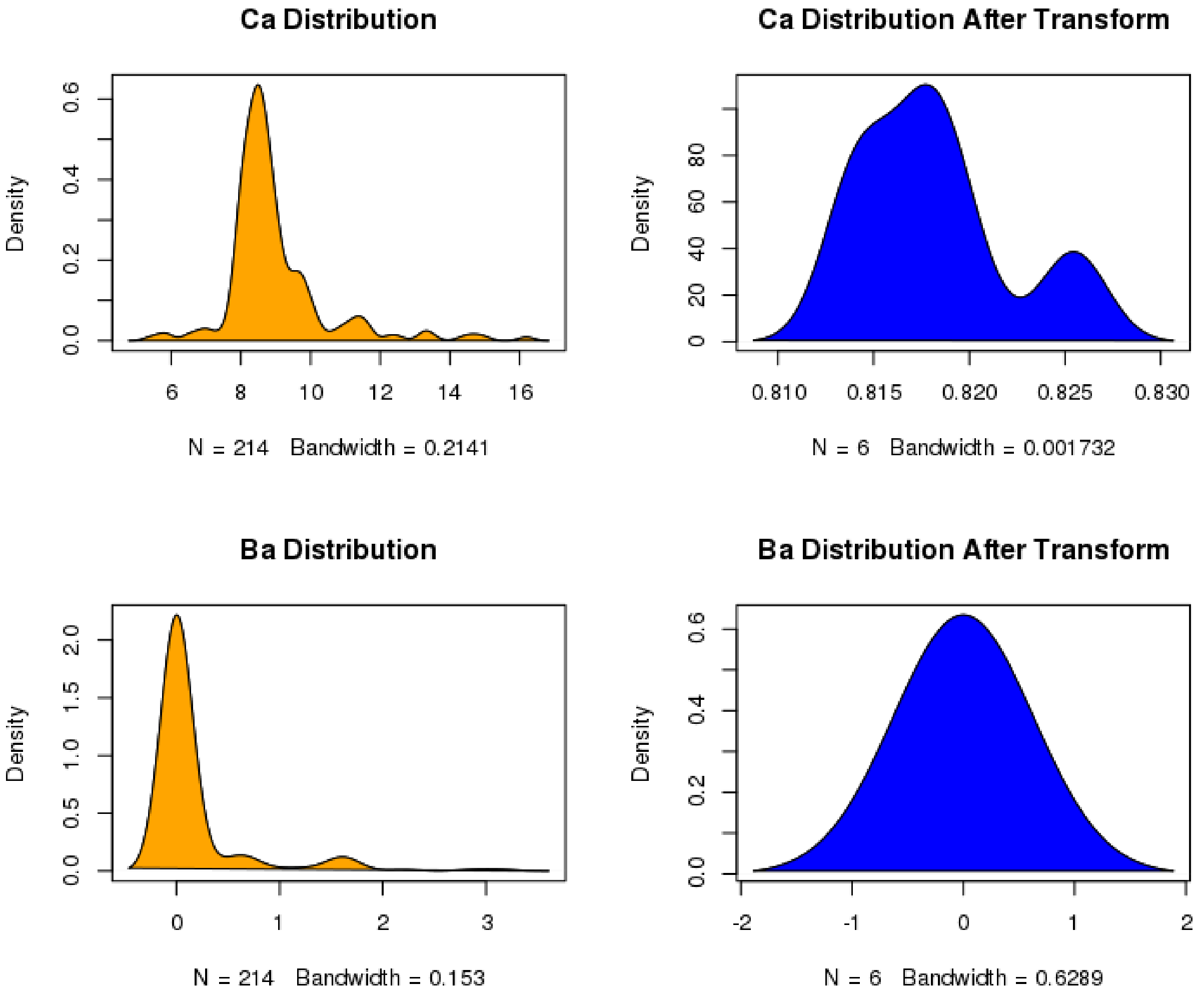

Al Distribution After Transform
Density
0.0
0.5
1.0
1.5
2.0
-0.2
0.0
0.2
0.4
0.6
0.8
N = 6 Bandwidth = 0.09684


Si Distribution
Density
0.0
0.2
0.4
0.6
69
70
71
72
73
74
75
76
N = 214 Bandwidth = 0.1854


Si Distribution After Transform
Density
0.000
0.010
2550
2600
2650
2700
N = 6 Bandwidth = 11.79


K Distribution
Density
0.0
0.5
1.0
1.5
0
1
2
3
4
5
6
N = 214 Bandwidth = 0.112


K Distribution After Transform
Density
0.0
1.0
2.0
3.0
0.0
0.2
0.4
0.6
0.8
N = 6 Bandwidth = 0.07158

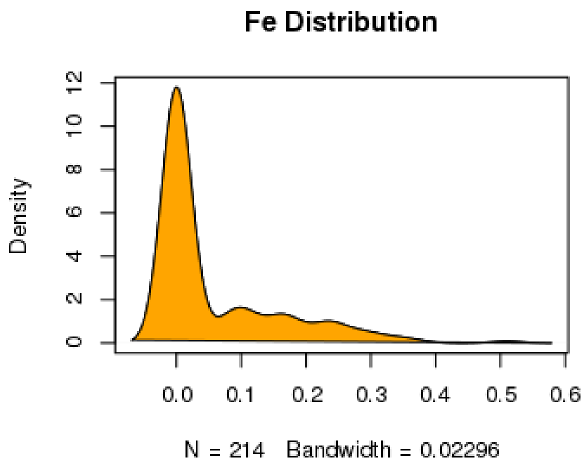


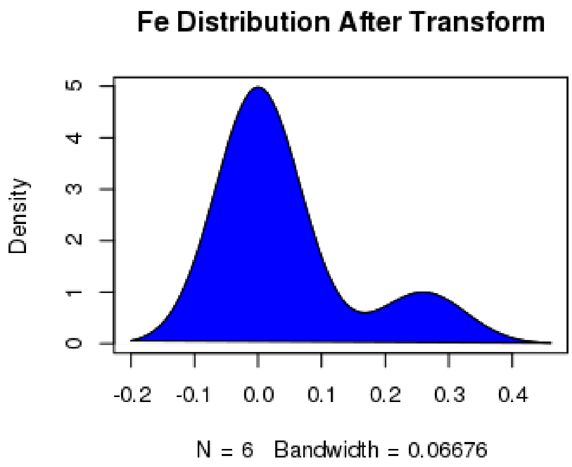
Ca Distribution

Ca Distribution After Transform

Ba Distribution

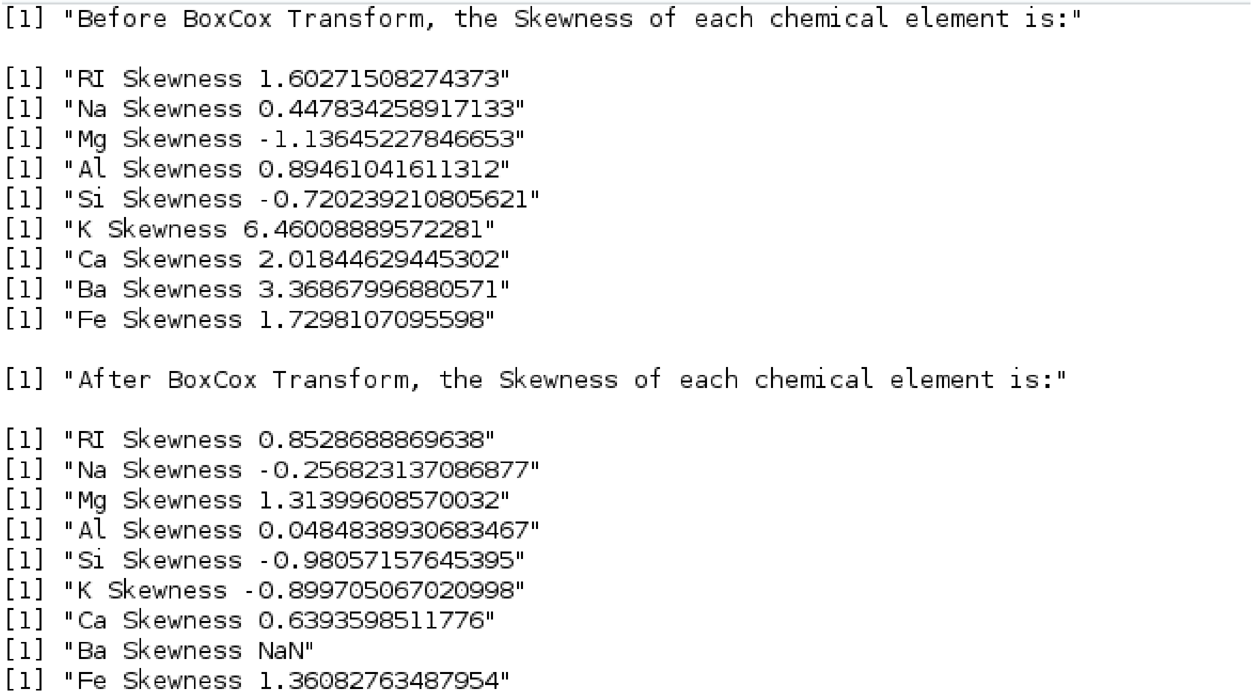
Ba Distribution After Transform





Analysis after boxcox transformation

Considering transformation, I found data of each chemical element normally distributed. The comparitive skewness value of given elements after box cox transformation are quite impressive and I find these as the improvement of classification model. The comparative skewness value before and after boxcox transformation is given below:



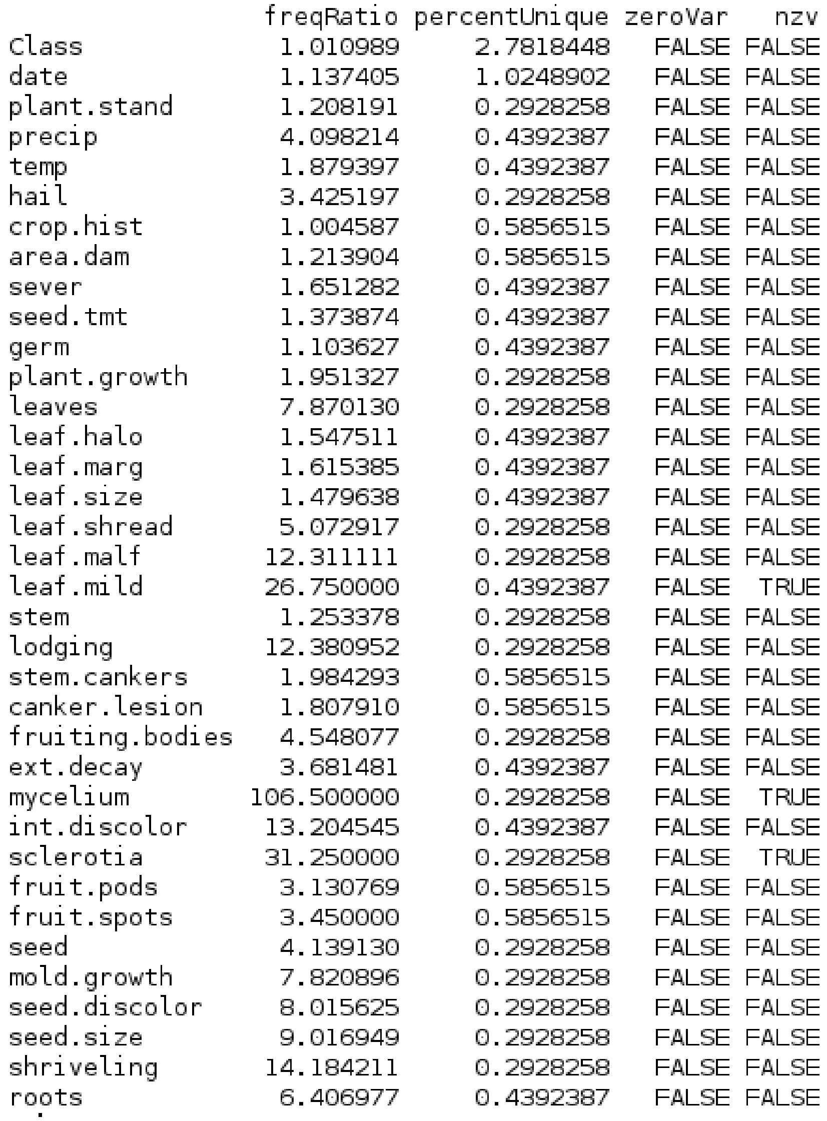
```
[1] "Before BoxCox Transform, the Skewness of each chemical element is:"

[1] "RI Skewness 1.60271508274373"
[1] "Na Skewness 0.447834258917133"
[1] "Mg Skewness -1.13645227846653"
[1] "Al Skewness 0.89461041611312"
[1] "Si Skewness -0.720239210805621"
[1] "K Skewness 6.46008889572281"
[1] "Ca Skewness 2.01844629445302"
[1] "Ba Skewness 3.36867996880571"
[1] "Fe Skewness 1.7298107095598"

[1] "After BoxCox Transform, the Skewness of each chemical element is:"

[1] "RI Skewness 0.8528688869638"
[1] "Na Skewness -0.256823137086877"
[1] "Mg Skewness 1.31399608570032"
[1] "Al Skewness 0.0484838930683467"
[1] "Si Skewness -0.98057157645395"
[1] "K Skewness -0.899705067020998"
[1] "Ca Skewness 0.6393598511776"
[1] "Ba Skewness NaN"
[1] "Fe Skewness 1.36082763487954"
```

3.2 (a) Investigate the frequency distributions for the categorical predictors. Are any of the distributions degenerate in the ways discussed earlier in this chapter?

Solution:

Approach:
If the predictor variables has either zero variance or near-zero-variance then the distribution of these variables are degenerated distributions. I have used nearZeroVar() function to find the uniqueness of data that will return the zero-variance or near-zero-variance value of each vaiable.

Result:



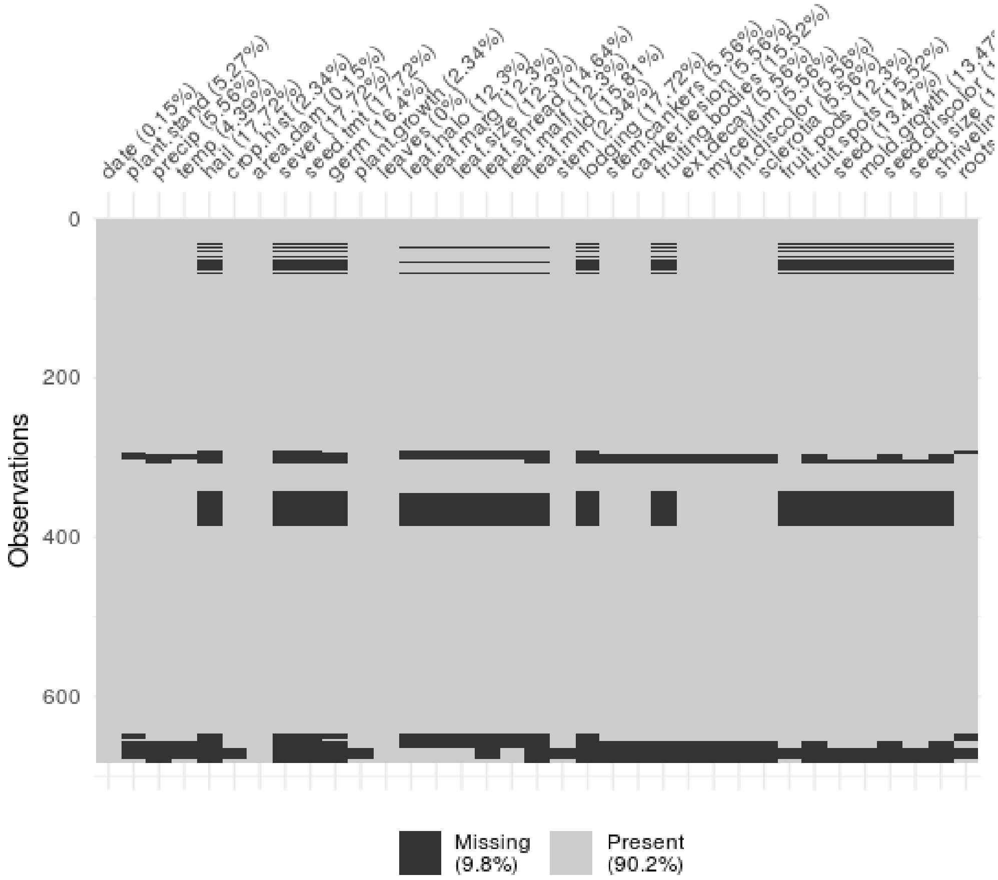
```
                  freqRatio percentUnique zeroVar   nzv
Class              1.010989     2.7818448   FALSE FALSE
date               1.137405     1.0248902   FALSE FALSE
plant.stand        1.208191     0.2928258   FALSE FALSE
precip             4.098214     0.4392387   FALSE FALSE
temp               1.879397     0.4392387   FALSE FALSE
hail               3.425197     0.2928258   FALSE FALSE
crop.hist          1.004587     0.5856515   FALSE FALSE
area.dam           1.213904     0.5856515   FALSE FALSE
sever              1.651282     0.4392387   FALSE FALSE
seed.tmt           1.373874     0.4392387   FALSE FALSE
germ               1.103627     0.4392387   FALSE FALSE
plant.growth       1.951327     0.2928258   FALSE FALSE
leaves             7.870130     0.2928258   FALSE FALSE
leaf.halo          1.547511     0.4392387   FALSE FALSE
leaf.marg          1.615385     0.4392387   FALSE FALSE
leaf.size          1.479638     0.4392387   FALSE FALSE
leaf.shread        5.072917     0.2928258   FALSE FALSE
leaf.malf         12.311111     0.2928258   FALSE FALSE
leaf.mild         26.750000     0.4392387   FALSE  TRUE
stem               1.253378     0.2928258   FALSE FALSE
lodging           12.380952     0.2928258   FALSE FALSE
stem.cankers       1.984293     0.5856515   FALSE FALSE
canker.lesion      1.807910     0.5856515   FALSE FALSE
fruiting.bodies    4.548077     0.2928258   FALSE FALSE
ext.decay          3.681481     0.4392387   FALSE FALSE
mycelium         106.500000     0.2928258   FALSE  TRUE
int.discolor      13.204545     0.4392387   FALSE FALSE
sclerotia         31.250000     0.2928258   FALSE  TRUE
fruit.pods         3.130769     0.5856515   FALSE FALSE
fruit.spots        3.450000     0.5856515   FALSE FALSE
seed               4.139130     0.2928258   FALSE FALSE
mold.growth        7.820896     0.2928258   FALSE FALSE
seed.discolor      8.015625     0.2928258   FALSE FALSE
seed.size          9.016949     0.2928258   FALSE FALSE
shriveling        14.184211     0.2928258   FALSE FALSE
roots              6.406977     0.4392387   FALSE FALSE
```

Analysis: By running the near-zero-variance function, it is seen that **leaf.mild**, **mycelium** and **sclerotia** predictors are having near-zero-variance. While none of the predictors are having zero variance.

3.2 (b) Roughly 18 % of the data are missing. Are there particular predictors that are more likely to be missing? Is the pattern of missing data related to the classes?
Solution:

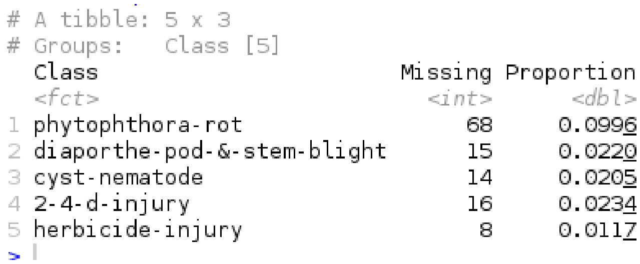
Approach:
To find the missing value in each predictor, I have use library(visdat) where function vis_miss() calculates the missing values in each variable, by length. I have used the library(dplyr) to find the pattern of missing data related to classes.

Result:



Analysis:
After seeing this result in above figure, it is seen that one predictor "hail" has roughly 18% missing value and another predictor "date" has 1% missing value. The total missing value is 9.8% in overall data.

Result:



```
# A tibble: 5 x 3
# Groups:   Class [5]
  Class                        Missing Proportion
  <fct>                          <int>      <dbl>
1 phytophthora-rot                  68     0.0996
2 diaporthe-pod-&-stem-blight       15     0.0220
3 cyst-nematode                     14     0.0205
4 2-4-d-injury                      16     0.0234
5 herbicide-injury                   8     0.0117
> 
```

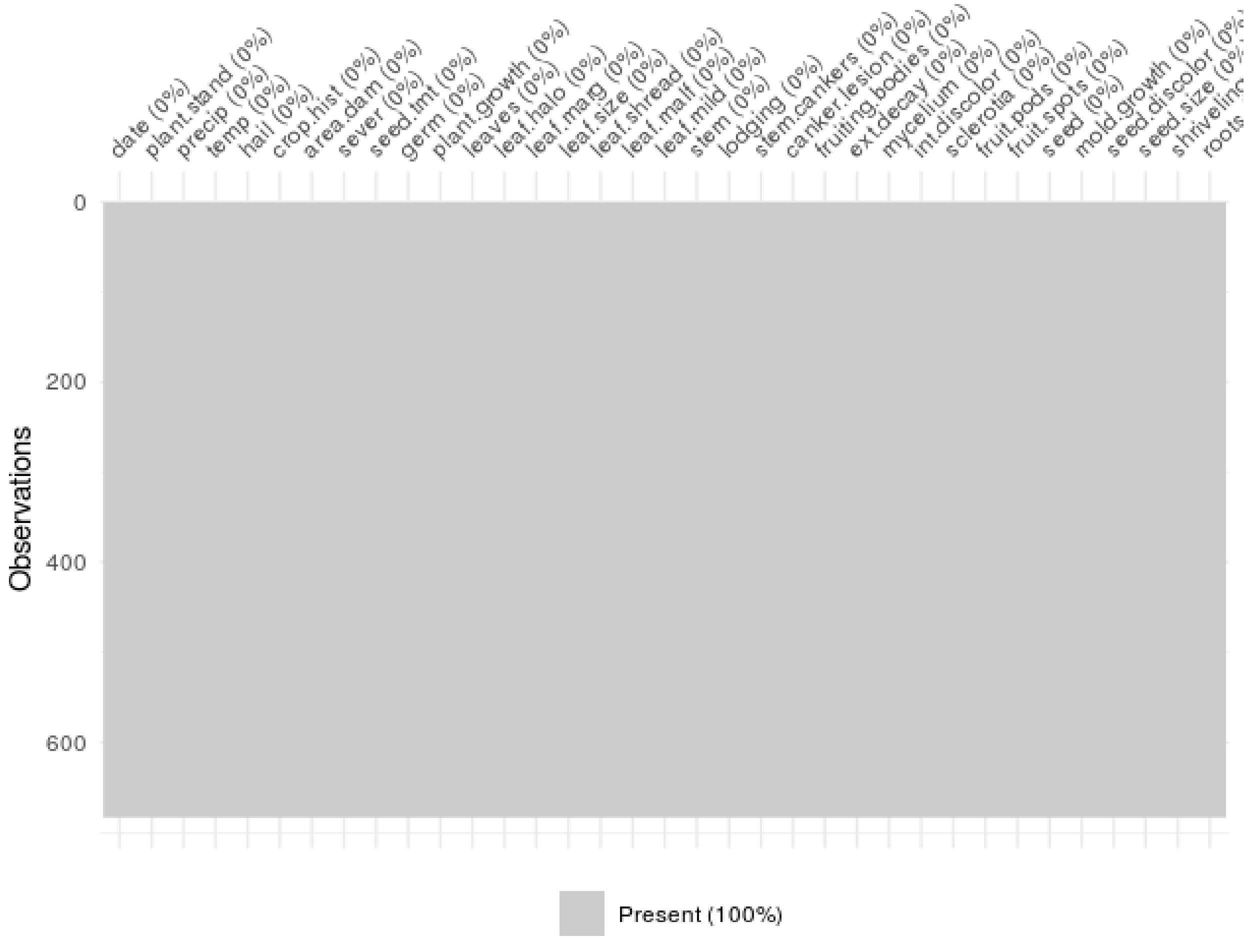
Analysis:

After seeing the result from the table, I found that there are 5 classes having missing data and among them ”phytophthora-rot” class having nearly 10 percent missing/incomplete cases. And remaining ”diaporthe-pod--stem-blight”, ”cyst-nematode”, ”2-4-d-injury”, and ”herbicide-injury” has the range of (1-2) percent missing/incomplete cases.

3.2 (c) Develop a strategy for handling missing data, either by eliminating predictors or imputation.
Solution:

Approach:
For handling the missing data, I used imputation strategy for the missing values using library DMwR. I did imputation using KNN classifier with value 10 i.e k=10. Then, I used same vis_miss to visualize the finite result data after imputation.

Result:



Analysis: In figure, we can see that the KNN method solves the missing value problem predictors after imputation. After applying this imputation using the knnImputation() function from the DMwR library, there are no missing data left in the data.

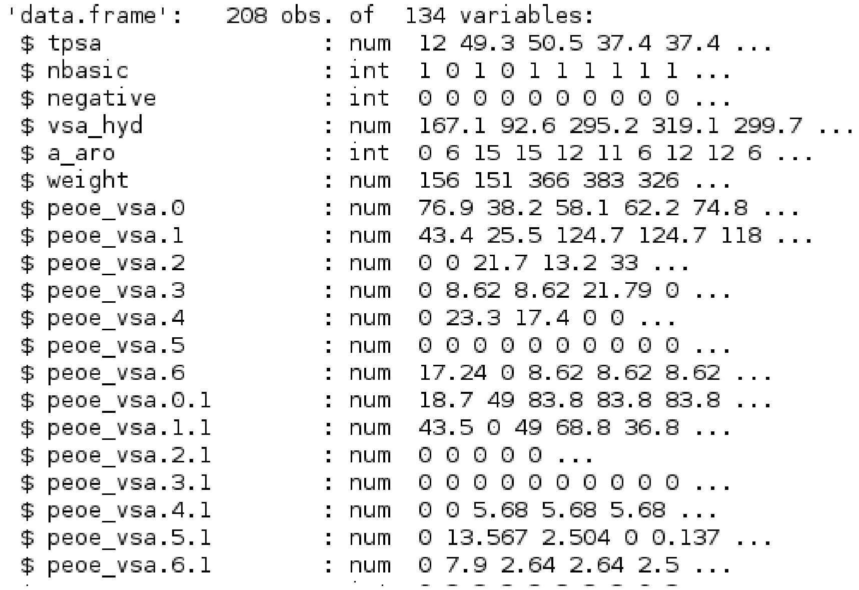
3.3.(a) Start R and use these commands to load the data:

Solution:

Approach:
I used the library(caret) and used the str(logBBB) and str(bbbDescr) to know what these logBBB and bbbDescr values of BloodBrain have using R.

Result:
str(logBBB) gives
num [1 : 208] 1.08 -0.4 0.22 0.14 0.69 0.44 -0.43 1.38 0.75 0.88 ...

str(bbbDescr) gives



```
'data.frame':   208 obs. of  134 variables:
 $ tpsa               : num  12 49.3 50.5 37.4 37.4 ...
 $ nbasic             : int  1 0 1 0 1 1 1 1 1 1 ...
 $ negative           : int  0 0 0 0 0 0 0 0 0 0 ...
 $ vsa_hyd            : num  167.1 92.6 295.2 319.1 299.7 ...
 $ a_aro              : int  0 6 15 15 12 11 6 12 12 6 ...
 $ weight             : num  156 151 366 383 326 ...
 $ peoe_vsa.0         : num  76.9 38.2 58.1 62.2 74.8 ...
 $ peoe_vsa.1         : num  43.4 25.5 124.7 124.7 118 ...
 $ peoe_vsa.2         : num  0 0 21.7 13.2 33 ...
 $ peoe_vsa.3         : num  0 8.62 8.62 21.79 0 ...
 $ peoe_vsa.4         : num  0 23.3 17.4 0 0 ...
 $ peoe_vsa.5         : num  0 0 0 0 0 0 0 0 0 0 ...
 $ peoe_vsa.6         : num  17.24 0 8.62 8.62 8.62 ...
 $ peoe_vsa.0.1       : num  18.7 49 83.8 83.8 83.8 ...
 $ peoe_vsa.1.1       : num  43.5 0 49 68.8 36.8 ...
 $ peoe_vsa.2.1       : num  0 0 0 0 0 ...
 $ peoe_vsa.3.1       : num  0 0 0 0 0 0 0 0 0 0 ...
 $ peoe_vsa.4.1       : num  0 0 5.68 5.68 5.68 ...
 $ peoe_vsa.5.1       : num  0 13.567 2.504 0 0.137 ...
 $ peoe_vsa.6.1       : num  0 7.9 2.64 2.64 2.5 ...
```

Analysis
The vector of assay results logBBB has variable range from 1 to 208 which is the actual output. And, the data frame bbbDescr has 134 variables with 208 observations, few variables screenshot above.

3.3.(b) Do any of the individual predictors have degenerate distributions?
Solution:

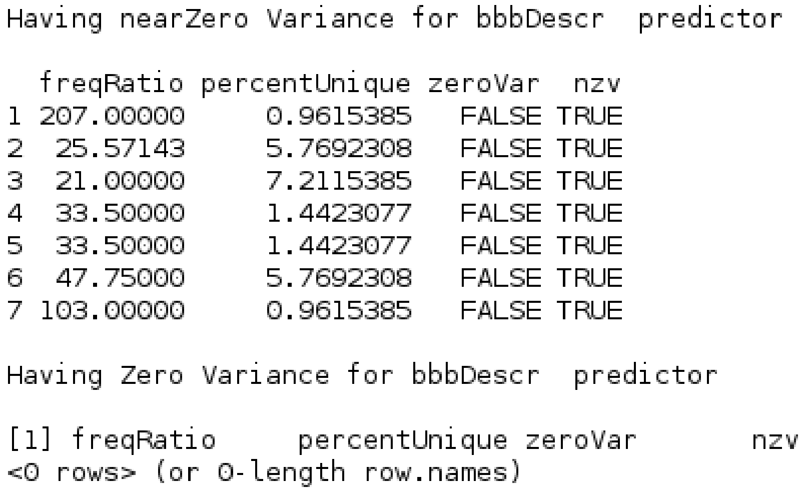
Approach:
I used the function nearZeroVar() to calculate if there is zero-variance or near-zero variance i.e degenerate distribution. I applied that function to ata frame bbbDescr using R. In order to get the data having only near-zero-variance and zero-variance, I used the filter() function of library(dplyr).

Result:

nearZeroVar(logBBB) gives

Class	freqRatio	percentUnique	zeroVar	nzv
negative	207.000000	0.9615385	FALSE	TRUE
$peoe\_vsa.2.1$	25.571429	5.7692308	FALSE	TRUE
$peoe\_vsa.3.1$	21.000000	7.2115385	FALSE	TRUE
a_acid	33.500000	1.4423077	FALSE	TRUE
vsa_acid	33.500000	1.4423077	FALSE	TRUE
$frac.anion7.$	47.750000	5.7692308	FALSE	TRUE
alert	103.000000	0.9615385	FALSE	TRUE

ScreenShot of Result:



```
Having nearZero Variance for bbbDescr  predictor

  freqRatio percentUnique zeroVar  nzv
1 207.00000     0.9615385   FALSE TRUE
2  25.57143     5.7692308   FALSE TRUE
3  21.00000     7.2115385   FALSE TRUE
4  33.50000     1.4423077   FALSE TRUE
5  33.50000     1.4423077   FALSE TRUE
6  47.75000     5.7692308   FALSE TRUE
7 103.00000     0.9615385   FALSE TRUE

Having Zero Variance for bbbDescr  predictor

[1] freqRatio     percentUnique zeroVar       nzv
<0 rows> (or 0-length row.names)
```

Analysis

After performing the function, I found that data frame bbbDescr have 7 variables that shows TRUE for near-zero-variance which means the data frame have degenerated distribution. And the table presented in the result clearly depicts the reason behind the degenerated distribution in dataframe bbbDescr.

3.3 (c) Generally speaking, are there strong relationships between the predic- tor data? If so, how could correlations in the predictor set be reduced? Does this have a dramatic effect on the number of predictors available for modeling?

Solution

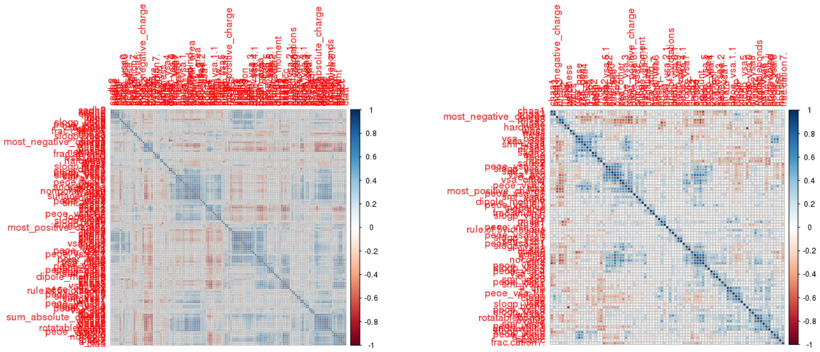
Approach

To get those answers, I used the findCorrelation() function for data frame: bbbDescr and plotted the data using corrplot(). The data if are highly correlated, then either

(a) we can convert into Principal Component Scores(PCA-Scores) and perform multiple regression ,OR

(b) we eliminate those highly correlated predictor variables.

Among these options,I used second approach to reduce the correlations between the predictors.



(a) Before Reducing Correlations

(b) After Reducing Correlations

Figure 6: Comparing the result before and after correlation reduction.

Analysis

Yes, there is a strong relationship between the predictor data of data frame: bbbDescr, as shown on left-side of above figure.

After performing the findCorrelation() with cutoff value=0.85 for data frame bbbDescr, I find that length of highly correlated predictor data are 49. The figure on left side shows data are highly correlated explained by more blue dots.

While after removing those highly correlated datas, the result I find the data are less likely correlated as represented by the less blue dots on the right side of above figure.

Reducing correlated variable means reducing reduntant datas or reducing data having similar characteristics. This means if we remove these datas, we might get a little less accuracy but it is ok. Because for modeling important thing is to collect all the important data which we will not miss even if we reduce the correlated datas. And having important data without redundancy will be more precise, accurate and result driven. If we consider every correlated datas, then all output will dominant for these correlated datas and creating a model with those data might end up with no good results.

This is why, the reduction of correlated variable will have not any dramatic effect on the number of predictors available for modeling.

3.1 R - CODE

```
library(mlbench)
library(caret)
library(e1071)
library(corrplot)
library(AppliedPredictiveModeling)

# Load the data Glass
data(Glass)


####################################################################
############################# 3.1 (a)################################
####################################################################

# this is done to divide the graph in 3 columns
par(mfrow = c(3, 3))

# histogram that shows the distribution of glass elements.
for (i in 1:9)-
    hist(Glass[ ,i], xlab = names(Glass[i]), main = paste(names(Glass[i]), ”Histogram”), col=”orange”)


# scatterplot to show the correlation of chemical elements
pairs(Glass[,1:9], main=”Scatterplot Matrix Representation for 9 Elements”) # equivalent to plot(X)

#resetting the par to get plot on whole page rather than 3 columns in a page.
par(mfrow = c())

#
# to find the correlation structure of the data, the corrplot package contains an
# excellent function to find the positive and negative relation between the elements
correlation¡-cor(Glass[,1:9])
dim(correlation)
correlation[1:9,1:9]
corrplot(correlation,order=”hclust”,method=”square”)




####################################################################
###################### End of 3.1(a)################################
####################################################################





####################################################################
############################# 3.1 (b)################################
####################################################################

# this is done to divide the graph in 3 columns
par(mfrow = c(3, 3))
# 3.1(b): To find if there appear any outlier in a given data
# boxplot that shows the distribution of glass elements.
for (i in 1:9)-
  boxplot(Glass[ ,i], ylab = names(Glass[i]), horizontal=T,main = paste(names(Glass[i]),
  ”Distribution using BoxPlot”), col=”orange”)
```

```
# density plot
# to determine the left-skewed and right-skewed distribution of graph
for (i in 1:9) –
  d ¡- density(Glass[,i], na.rm = TRUE)
  plot(d,main = paste(names(Glass[i]), ”Distribution using Density”))
  polygon(d, col=”orange”)


cat(”“n”)
print(”Before BoxCox Transform, the Skewness of each chemical element is:”)
cat(”“n”)

# table with values of skewness
# to determine the left-skewed and right-skewed distribution of graph
for (i in 1:9) –
  print(paste(names(Glass[i]), ”Skewness”, skewness(Glass[,i])))



####################################################################
###################### End of 3.1(b)################################
####################################################################



####################################################################
############################ 3.1 (c)################################
####################################################################


# 3.1(c) To find any relevant transformation of element
# boxcox transformation is used
# I used the boxcox transformation for transformation so that each elements
# distributes normally

# this is done to divide the graph in 2 columns
par(mfrow = c(2, 2))

# density plot
# showing the difference after applying the boxcox transformation
for(i in 1:9)–
  d ¡- density(Glass[,i], na.rm = TRUE)
  plot(d,main = paste(names(Glass[i]), ”Distribution”))
  polygon(d, col=”orange”)

  transformed¡-predict(BoxCoxTrans(Glass[,i]),head(Glass[,i]))
  d2 ¡- density(transformed, na.rm = TRUE)
  plot(d2,main = paste(names(Glass[i]), ”Distribution After Transform”))
  polygon(d2, col=”blue”)

cat(”“n”)
print(”After BoxCox Transform, the Skewness of each chemical element is:”)
cat(”“n”)

# table with values of skewness
# to determine the left-skewed and right-skewed distribution of graph
for (i in 1:9) –
  transformed¡-predict(BoxCoxTrans(Glass[,i]),head(Glass[,i]))
  print(paste(names(Glass[i]), ”Skewness”, skewness(transformed)))



####################################################################
###################### End of 3.1(c)################################
####################################################################
```

3.2 R - CODE

```
# load library
library(mlbench)
library(caret)
library(e1071)
library(AppliedPredictiveModeling)
library(dplyr)
library(visdat)
library(DMwR)

# load data
data(Soybean)

#########################################################################################
# 3.2(a) Investigate the frequency distributions for the categorical predictors.
# Are any of the distributions degenerate in the ways discussed earlier in this chapter?
#########################################################################################

cat(”Near-Zero-Variance of 36 variables“n”)
print(nearZeroVar(Soybean, names = TRUE, saveMetrics=T))

##########################################################################################
##########################################################################################

#########################################################################################
# 3.2(b) Roughly 18 % of the data are missing. Are there particular predictors that are
# more likely to be missing? Is the pattern of missing data related to the classes?
#########################################################################################

# missing value in each predictor using vis miss()
missedValue ¡- Soybean[, -1]
plot(vis miss(missedValue))


cat(”Pattern of missing date related to the classes“n”)
print(Soybean %¿%
  mutate(Total = n()) %¿%
  filter(!complete.cases(.)) %¿%
  group by(Class) %¿%
  mutate(Missing = n(), Proportion=Missing/Total) %¿%
  select(Class, Missing, Proportion) %¿%
  unique())

##########################################################################################
##########################################################################################

# 3.2 (c) ) Develop a strategy for handling missing data, either by eliminating predictors
# or imputation

# need -DMwR  package
# # ## Apply Knn-Imputation
Im¡-knnImputation(missedValue)
plot(vis miss(Im))

##########################################################################################
##########################################################################################
```

3.3 R - CODE:

```
# loading the library

library(caret)
library(dplyr)
library(mlbench)
library(corrplot)


#loading the variables
data(BloodBrain)

####################################################################
####################################################################

# 3.3(a) The Blood Brain Barrier Data has
# vector logBBB that contains the concentration ration and
# data frame bbbDescr that contains the descriptor values


# to get the output returned by logBBB
str(logBBB)

# to get the variables returned by bbbDescr
str(bbbDescr)

####################################################################
###################################################################

# 3.3(b) Data Degeneration
#  to view the overall data having nearzerovariance or zero variance
bbbDescrVariance¡-nearZeroVar(bbbDescr, names = TRUE, saveMetrics=T)
cat(”“n”)
cat(” overall degenerated distribution of data frame bbbdescr “n“n”)
#
print(bbbDescrVariance)
cat(”“n”)


#  to view only those data that have nearzerovariance
cat(”Having nearZero Variance for bbbDescr  predictor “n“n”)

bbbDescrVarianceNZV¡-filter(bbbDescrVariance,nzv==”TRUE”)
print(bbbDescrVarianceNZV)
cat(”“n”)


#  to view only those data that have zerovariance
cat(”Having Zero Variance for bbbDescr  predictor “n“n”)

bbbDescrVarianceZeroVar¡-filter(bbbDescrVariance,zeroVar==”TRUE”)
print(bbbDescrVarianceZeroVar)
cat(”“n”)

####################################################################
###################################################################


# 3.3 (c) Generally speaking, are there strong relationships between the  predictor data?
# If so, how could correlations in the predictor set be reduced?
# Does this have a dramatic effect on the number of predictors available for modeling?

correlations¡-cor(bbbDescr)
```

```
corrplot(correlations,order=”hclust”)

highcorr¡- findCorrelation(correlations,cutoff = 0.85)
print(length(highcorr))

# #  By using following method, I removed the highly correlated data
filteredCor ¡- bbbDescr[,-highcorr]

filteredHighCor¡-cor(filteredCor)
corrplot(filteredHighCor,order=”hclust”)



####################################################################
###################################################################
```

References:

1. Applied Predictive Modeling : @authors Max Kuhn. Kjell Johnson
2. https://archive.ics.uci.edu/ml/index.php