Comparison of Classification and Clustering Algorithms on Glass Dataset Using R

Talha Hanif Butt

```r
library(caret)
```

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

```r
library(mclust)
```

```
## Package 'mclust' version 5.4.1
## Type 'citation("mclust")' for citing this R package in publications.
```

```r
library(fpc)
library(cluster)
library(clusteval)
library(factoextra)
```

```
## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ
```

```r
library(ggplot2)
library(kmed)
library(mlbench)
```

Loading Glass Dataset

```r
# attach the Glass Identification dataset to the environment
data("Glass")
# rename the dataset
dataset <- Glass
```

Partitioning Data for Validation

```r
# create a list of 80% of the rows inthe original dataset we can use for training
validation_index <- createDataPartition(dataset$Type, p=0.80, list=FALSE)
# select 20% of the data for validation
validation <- dataset[-validation_index,]
# use the remaining 80% of data to training and testing the models
dataset <- dataset[validation_index,]
```

Getting Insights from Data

```r
# dimensions of dataset
dim(dataset)
```

```
## [1] 174  10
```

```r
# list types for each attribute
sapply(dataset, class)
```

```
##        RI        Na        Mg        Al        Si         K        Ca
## "numeric" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric"
##        Ba        Fe      Type
## "numeric" "numeric"  "factor"
```

```r
# take a peek at the first 6 rows of the data
head(dataset)
```

```
##        RI    Na   Mg   Al    Si    K   Ca Ba Fe Type
## 1 1.52101 13.64 4.49 1.10 71.78 0.06 8.75  0  0    1
## 2 1.51761 13.89 3.60 1.36 72.73 0.48 7.83  0  0    1
## 3 1.51618 13.53 3.55 1.54 72.99 0.39 7.78  0  0    1
## 4 1.51766 13.21 3.69 1.29 72.61 0.57 8.22  0  0    1
## 5 1.51742 13.27 3.62 1.24 73.08 0.55 8.07  0  0    1
## 7 1.51743 13.30 3.60 1.14 73.09 0.58 8.17  0  0    1
```

```r
# list the levels for the class
levels(dataset$Type)
```

```
## [1] "1" "2" "3" "5" "6" "7"
```

```r
# summarize the class distribution
percentage <- prop.table(table(dataset$Type)) * 100
cbind(freq=table(dataset$Type), percentage=percentage)
```

```
##   freq percentage
## 1   56  32.183908
## 2   61  35.057471
## 3   14   8.045977
## 5   11   6.321839
## 6    8   4.597701
## 7   24  13.793103
```

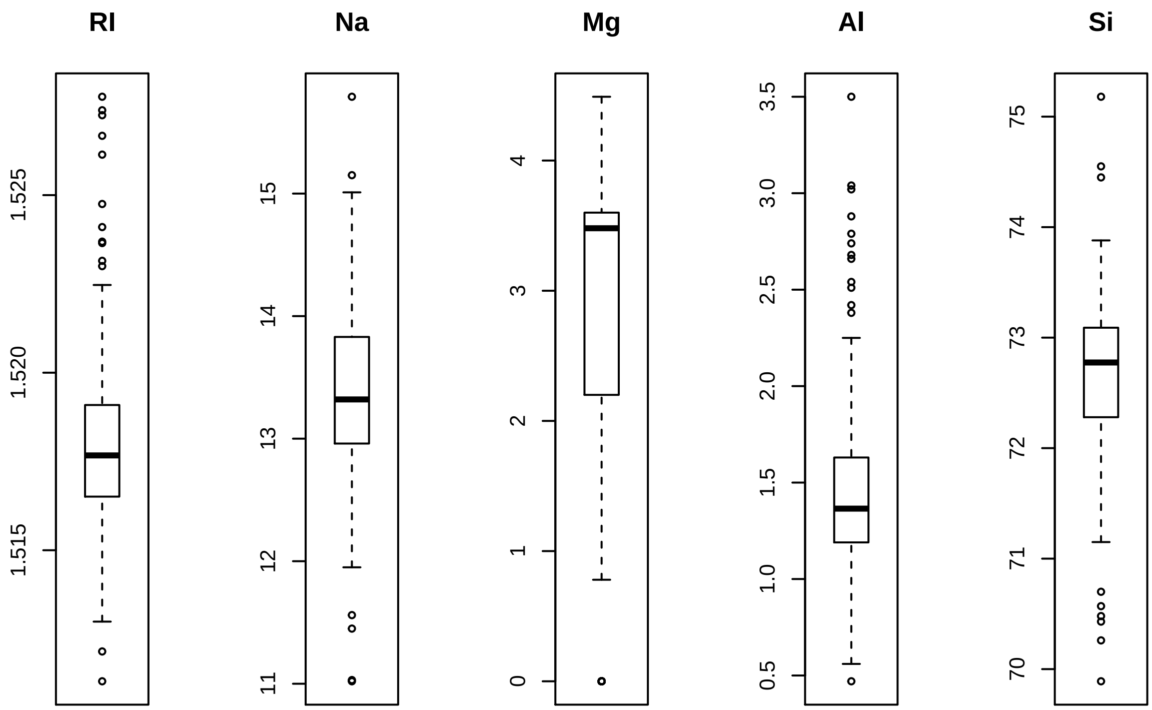
```r
# summarize attribute distributions
summary(dataset)
```

```
##        RI              Na              Mg              Al
##  Min.   :1.511   Min.   :11.02   Min.   :0.000   Min.   :0.470
##  1st Qu.:1.517   1st Qu.:12.96   1st Qu.:2.210   1st Qu.:1.190
##  Median :1.518   Median :13.32   Median :3.480   Median :1.365
##  Mean   :1.518   Mean   :13.42   Mean   :2.716   Mean   :1.462
##  3rd Qu.:1.519   3rd Qu.:13.82   3rd Qu.:3.600   3rd Qu.:1.627
##  Max.   :1.528   Max.   :15.79   Max.   :4.490   Max.   :3.500
##        Si              K                Ca                Ba
##  Min.   :69.89   Min.   :0.0000   Min.   : 5.430   Min.   :0.0000
##  1st Qu.:72.28   1st Qu.:0.1225   1st Qu.: 8.240   1st Qu.:0.0000
##  Median :72.78   Median :0.5500   Median : 8.590   Median :0.0000
##  Mean   :72.66   Mean   :0.5102   Mean   : 8.899   Mean   :0.1589
##  3rd Qu.:73.09   3rd Qu.:0.6000   3rd Qu.: 9.217   3rd Qu.:0.0000
##  Max.   :75.18   Max.   :6.2100   Max.   :14.960   Max.   :2.2000
##        Fe            Type
##  Min.   :0.00000   1:56
##  1st Qu.:0.00000   2:61
##  Median :0.00000   3:14
```

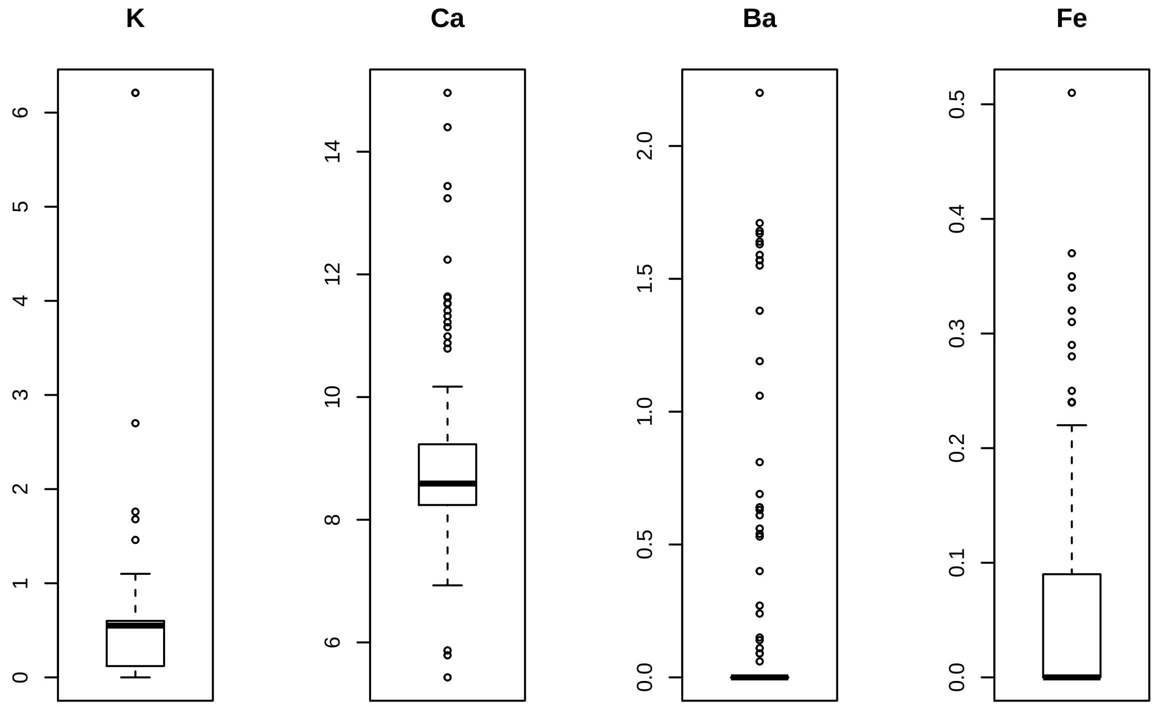
```
##  Mean   :0.05305   5:11
##  3rd Qu.:0.09000   6: 8
##  Max.   :0.51000   7:24
```

```
# split input and output
x <- dataset[,1:9]
y <- dataset[,10]
```

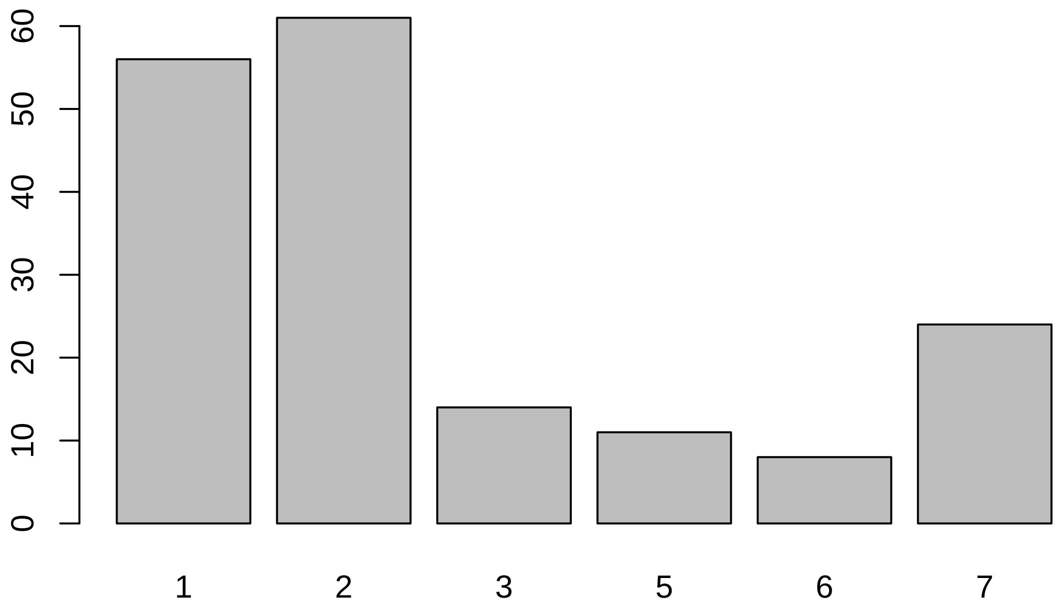
```
# boxplot for each attribute on one image
par(mfrow=c(1,5))
  for(i in 1:5) {
  boxplot(x[,i], main=names(Glass)[i])
}
```



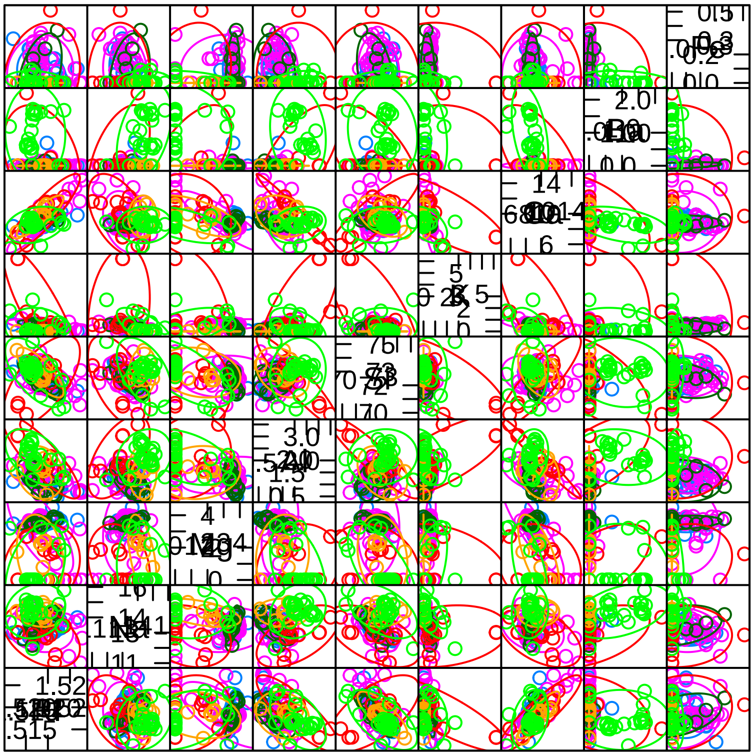

```
# boxplot for each attribute on one image
par(mfrow=c(1,4))
  for(i in 6:9) {
  boxplot(x[,i], main=names(Glass)[i])
}
```




```r
# barplot for class breakdown
plot(y)
```

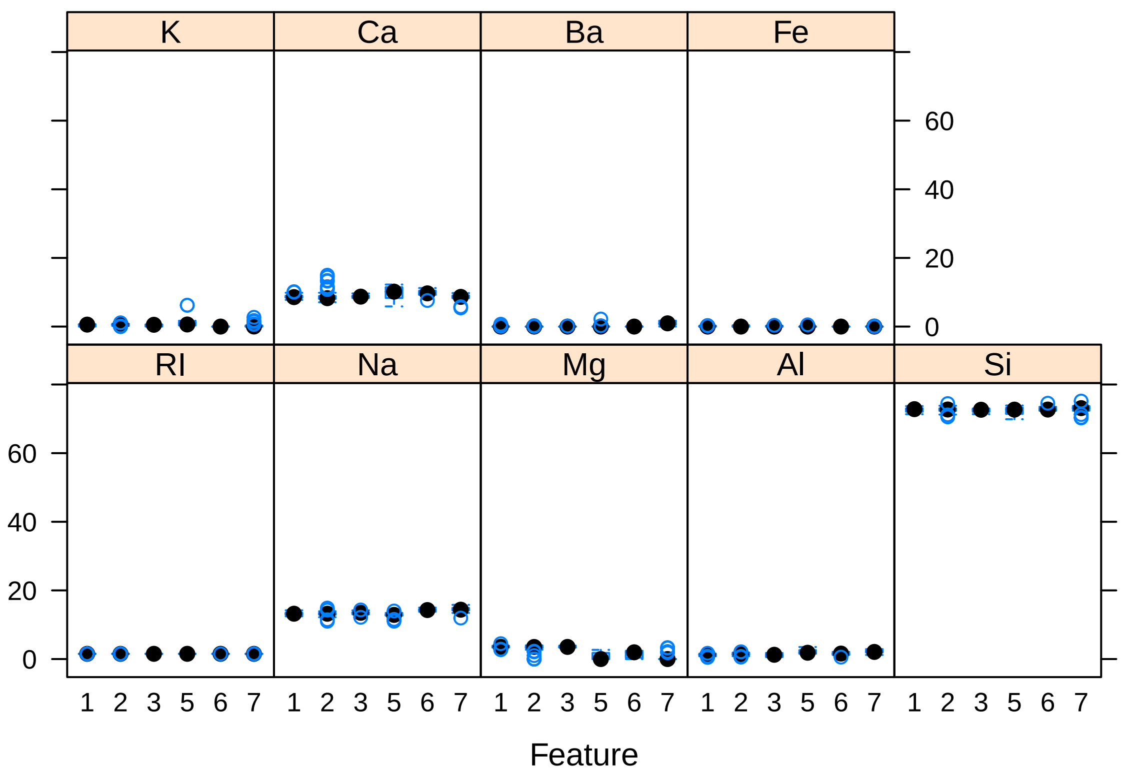


```r
# scatterplot matrix
featurePlot(x=x, y=y, plot="ellipse")
```

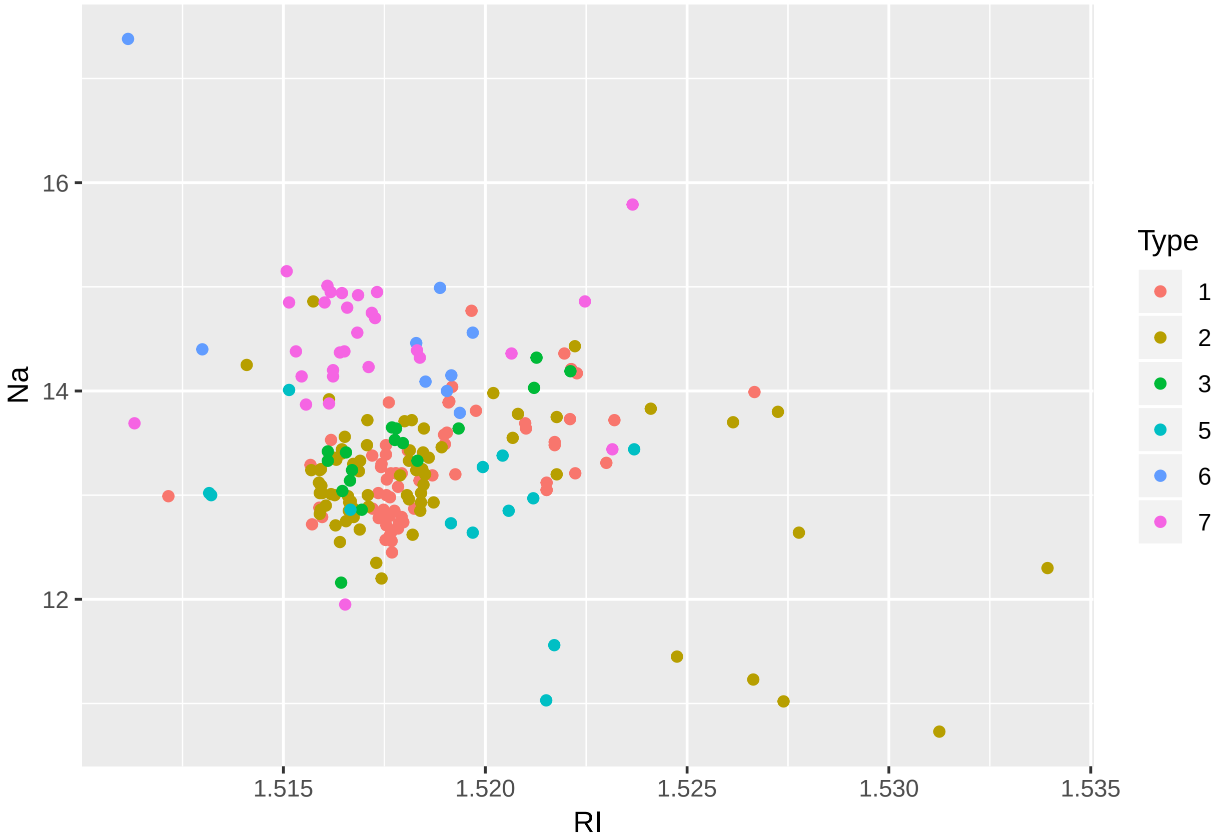


Scatter Plot Matrix

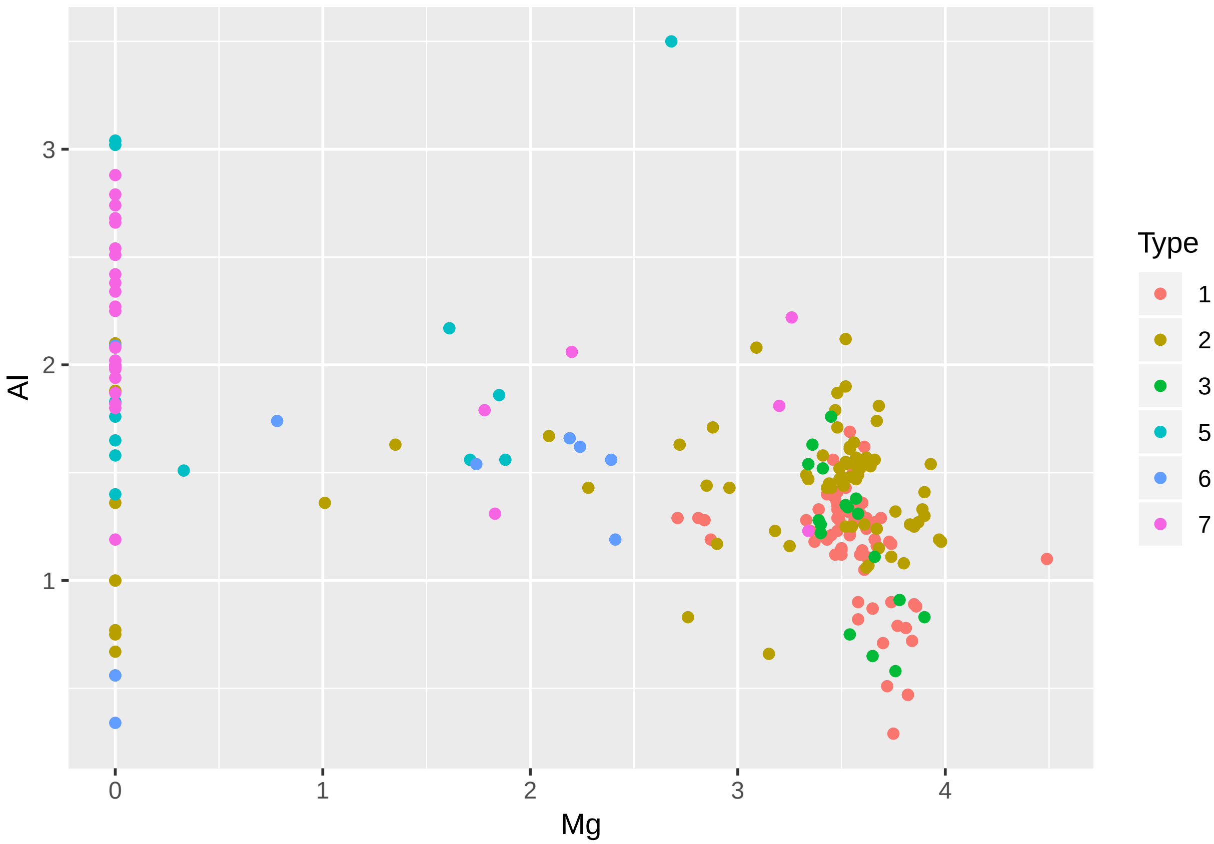
```r
# box and whisker plots for each attribute
featurePlot(x=x, y=y, plot="box")
```



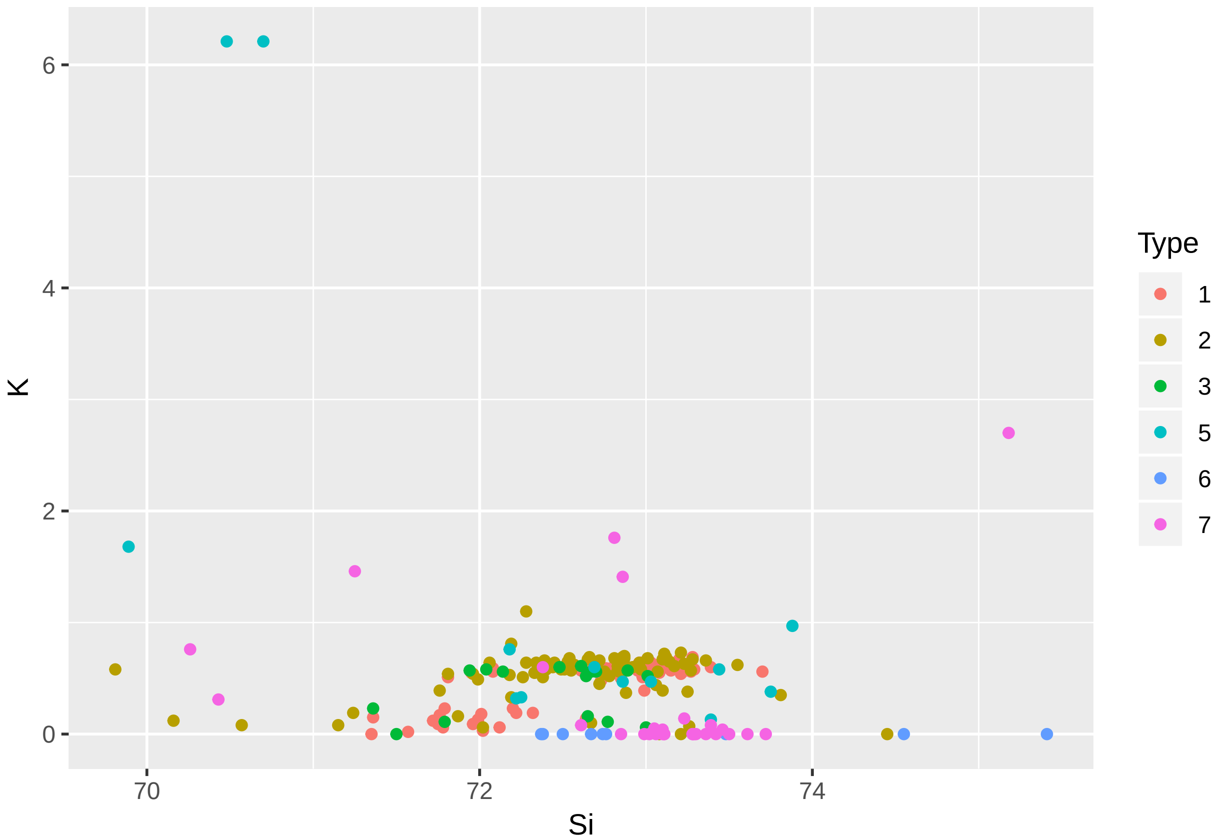
```r
ggplot(Glass, aes(RI,Na, color = Type)) + geom_point()
```



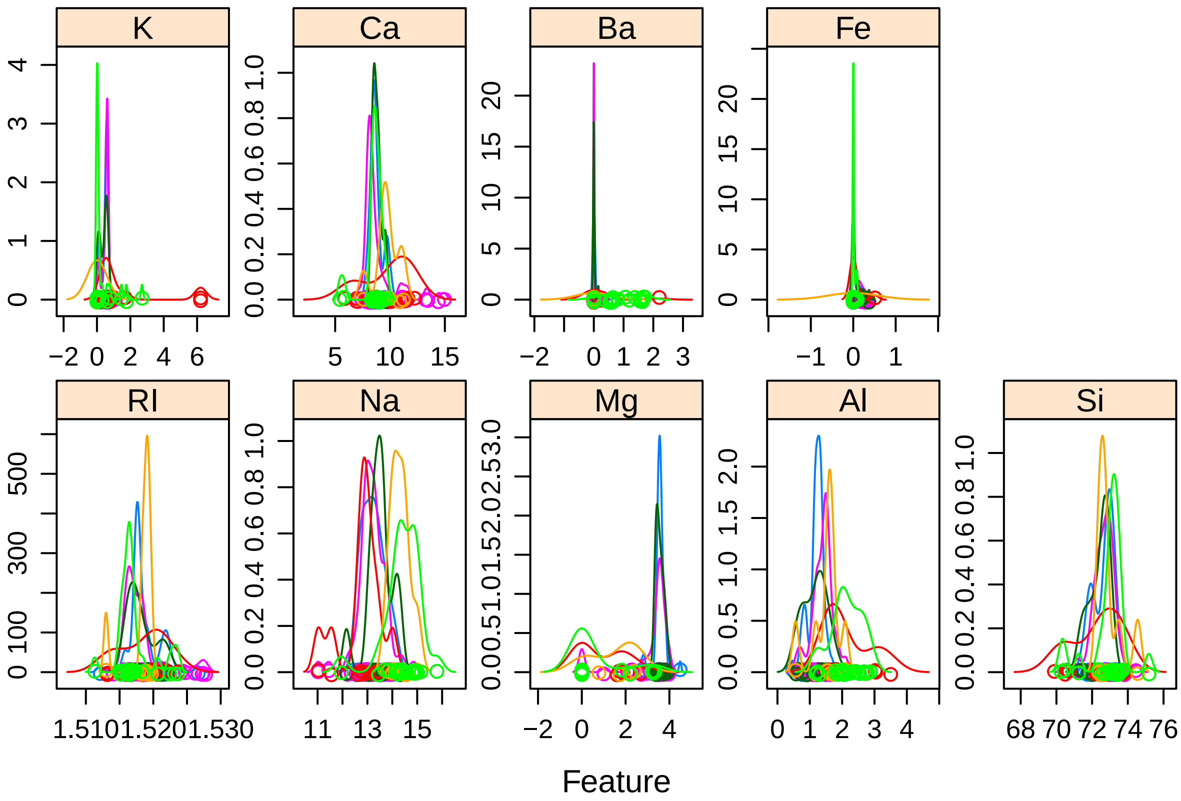

```r
ggplot(Glass, aes(Mg,Al, color = Type)) + geom_point()
```




```r
ggplot(Glass, aes(Si,K, color = Type)) + geom_point()
```




```r
# density plots for each attribute by class value
scales <- list(x=list(relation="free"), y=list(relation="free"))
featurePlot(x=x, y=y, plot="density", scales=scales)
```



Applying Classification Algorithms

```
# Run algorithms using 10-fold cross validation
control <- trainControl(method="cv", number=10)
metric <- "Accuracy"
```

```
# kNN
set.seed(7)
fit.knn <- train(Type~., data=dataset, method="knn", metric=metric, trControl=control)
# SVM
set.seed(7)
fit.svm <- train(Type~., data=dataset, method="svmRadial", metric=metric, trControl=control)
# Random Forest
set.seed(7)
fit.rf <- train(Type~., data=dataset, method="rf", metric=metric, trControl=control)
```

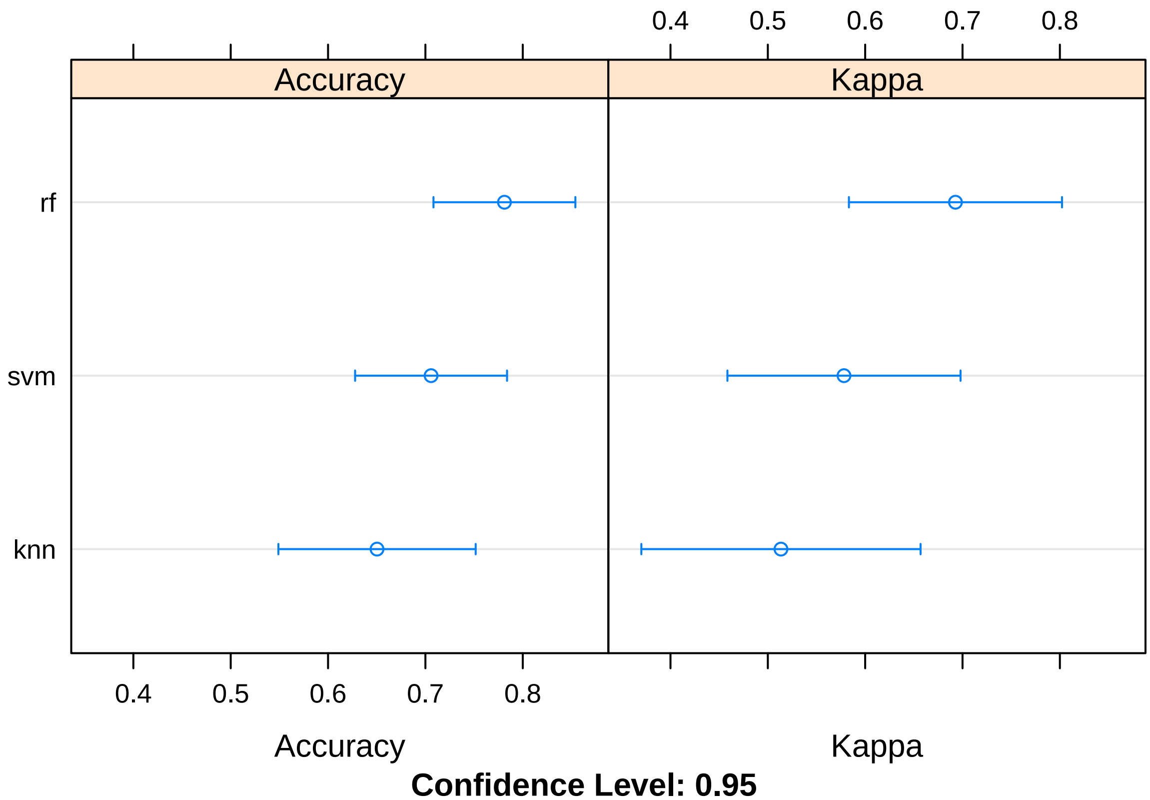
Comparison of the Classification Algorithms

```
# summarize accuracy of models
results <- resamples(list(knn=fit.knn, svm=fit.svm, rf=fit.rf))
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
```

```
## Models: knn, svm, rf
## Number of resamples: 10
##
## Accuracy
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## knn 0.5294118 0.5666118 0.5835913 0.6503074 0.7152778 0.9411765    0
## svm 0.5294118 0.6519608 0.6858553 0.7058243 0.7796053 0.8823529    0
## rf  0.5882353 0.7136223 0.7951389 0.7811791 0.8347039 0.9411765    0
##
## Kappa
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## knn 0.2727273 0.3962010 0.4512828 0.5134822 0.5985102 0.9174757    0
## svm 0.2727273 0.4861779 0.5610477 0.5782246 0.7031814 0.8308458    0
## rf  0.3928571 0.6040660 0.7184943 0.6927617 0.7842742 0.9154229    0
```

```r
# compare accuracy of models
dotplot(results)
```



Insights from the best model

```r
# summarize Best Model
print(fit.rf)
```

```
## Random Forest
##
## 174 samples
##   9 predictors
##   6 classes: '1', '2', '3', '5', '6', '7'
```

```
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 157, 158, 155, 156, 158, 157, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.7811791  0.6927617
##   5     0.7336365  0.6311314
##   9     0.7287001  0.6266981
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
# estimate skill of Random Forest on the validation dataset
predictions <- predict(fit.rf, validation)
confusionMatrix(predictions, validation$Type)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3  5  6  7
##          1 13  1  0  0  0  0
##          2  1 13  1  1  0  0
##          3  0  0  2  0  0  0
##          5  0  1  0  1  0  0
##          6  0  0  0  0  1  0
##          7  0  0  0  0  0  5
##
## Overall Statistics
##
##                Accuracy : 0.875
##                  95% CI : (0.732, 0.9581)
##     No Information Rate : 0.375
##     P-Value [Acc > NIR] : 8.429e-11
##
##                   Kappa : 0.8227
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 5 Class: 6 Class: 7
## Sensitivity            0.9286   0.8667   0.6667   0.5000    1.000    1.000
## Specificity            0.9615   0.8800   1.0000   0.9737    1.000    1.000
## Pos Pred Value         0.9286   0.8125   1.0000   0.5000    1.000    1.000
## Neg Pred Value         0.9615   0.9167   0.9737   0.9737    1.000    1.000
## Prevalence             0.3500   0.3750   0.0750   0.0500    0.025    0.125
## Detection Rate         0.3250   0.3250   0.0500   0.0250    0.025    0.125
## Detection Prevalence   0.3500   0.4000   0.0500   0.0500    0.025    0.125
## Balanced Accuracy      0.9451   0.8733   0.8333   0.7368    1.000    1.000
```

Applying Clustering Algorithms

```r
# K-means
set.seed(20)
fit.kmeans <- kmeans(Glass[, 1:9], 7, nstart = 20)
# Hierarchical Agglomerative
set.seed(20)
d <- dist(Glass[,1:9], method = "euclidean") # distance matrix
fit.ha <- hclust(d, method="ward.D")
# K-Medoids Clustering
num <- as.matrix(Glass[,1:9])
mrwdist <- distNumeric(num, num, method = "mrw")
fit.kmedoids <- fastkmed(mrwdist, ncluster = 7, iterate = 50)
```

Getting insights from Hierarchical Agglomerative Clustering

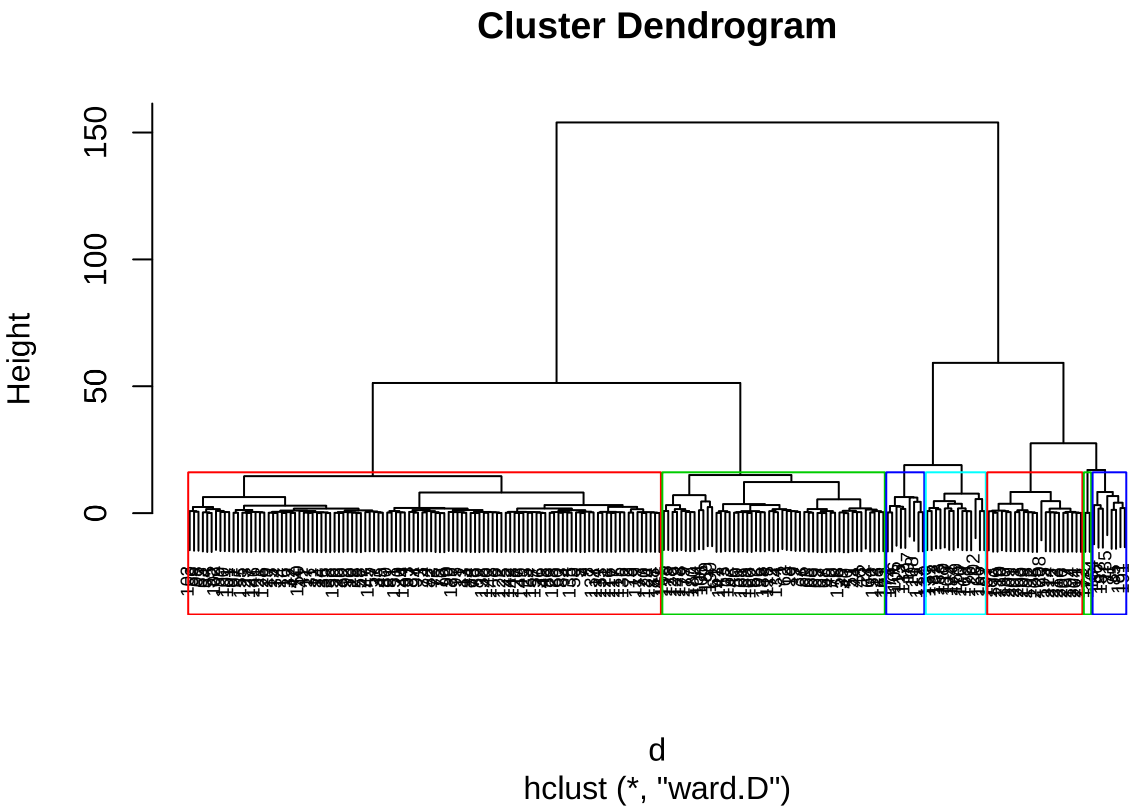
```r
# Cut tree into 4 groups
sub_grp <- cutree(fit.ha, k = 7)

# Number of members in each cluster
table(sub_grp)
```

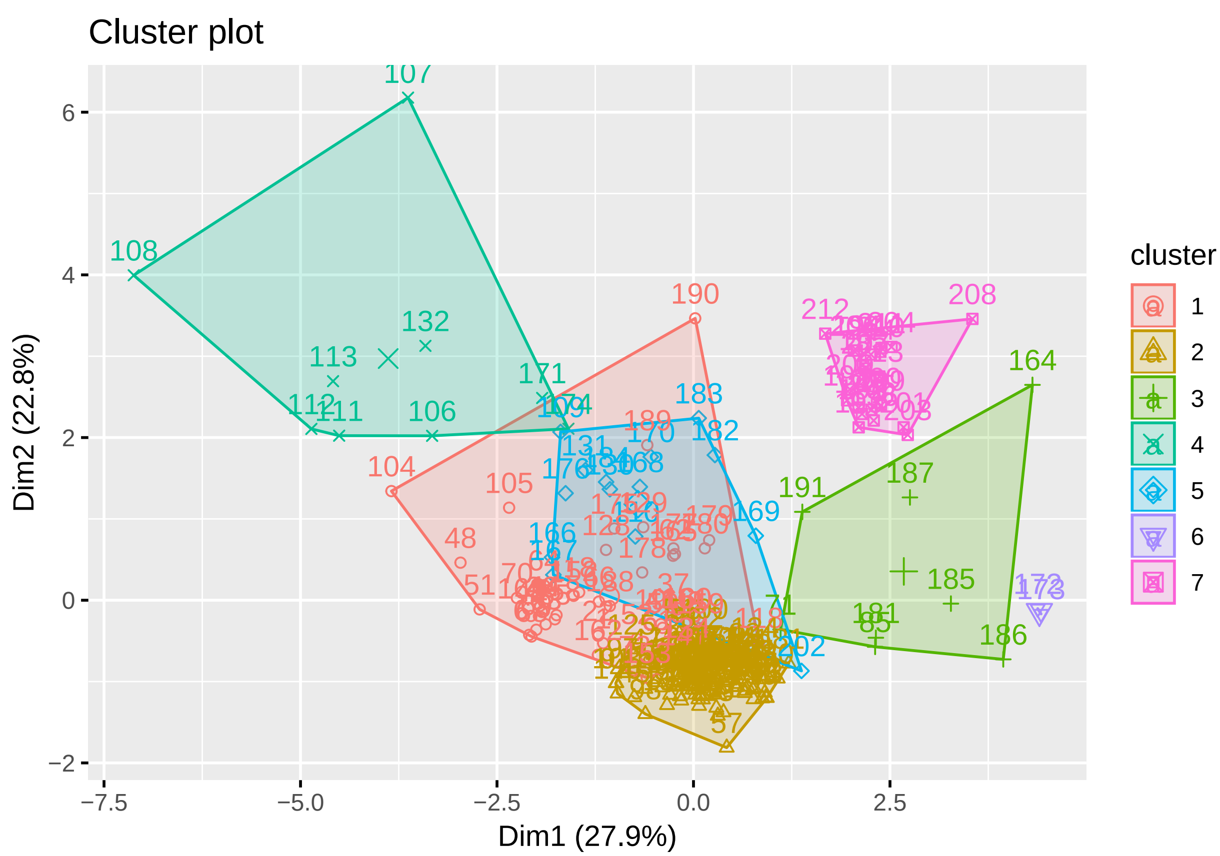
```
## sub_grp
##   1   2   3   4   5   6   7
##  51 108   8   9  14   2  22
```

```
## sub_grp
```

```r
plot(fit.ha, cex = 0.6)
rect.hclust(fit.ha, k = 7, border = 2:5)
```




```
fviz_cluster(list(data = Glass[,1:9], cluster = sub_grp))
```

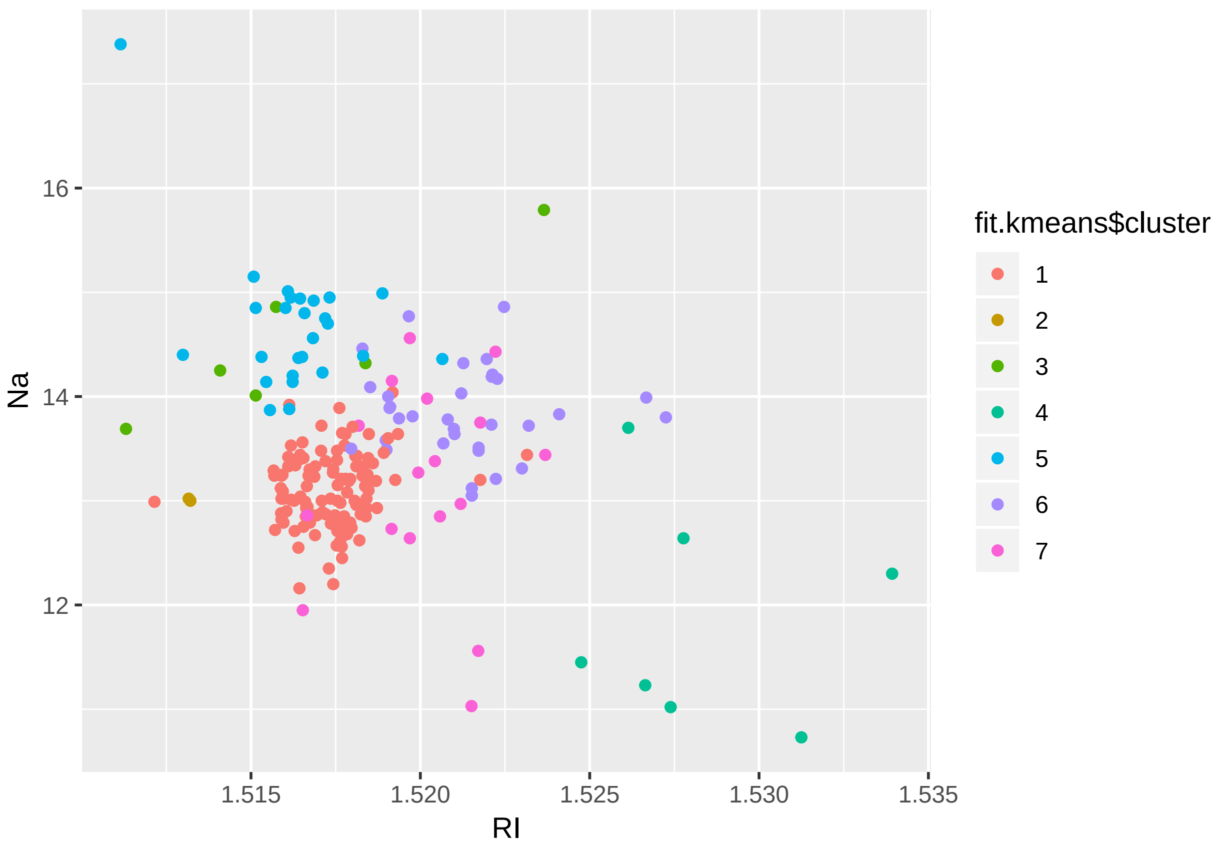


Getting insights from K-Means Clustering

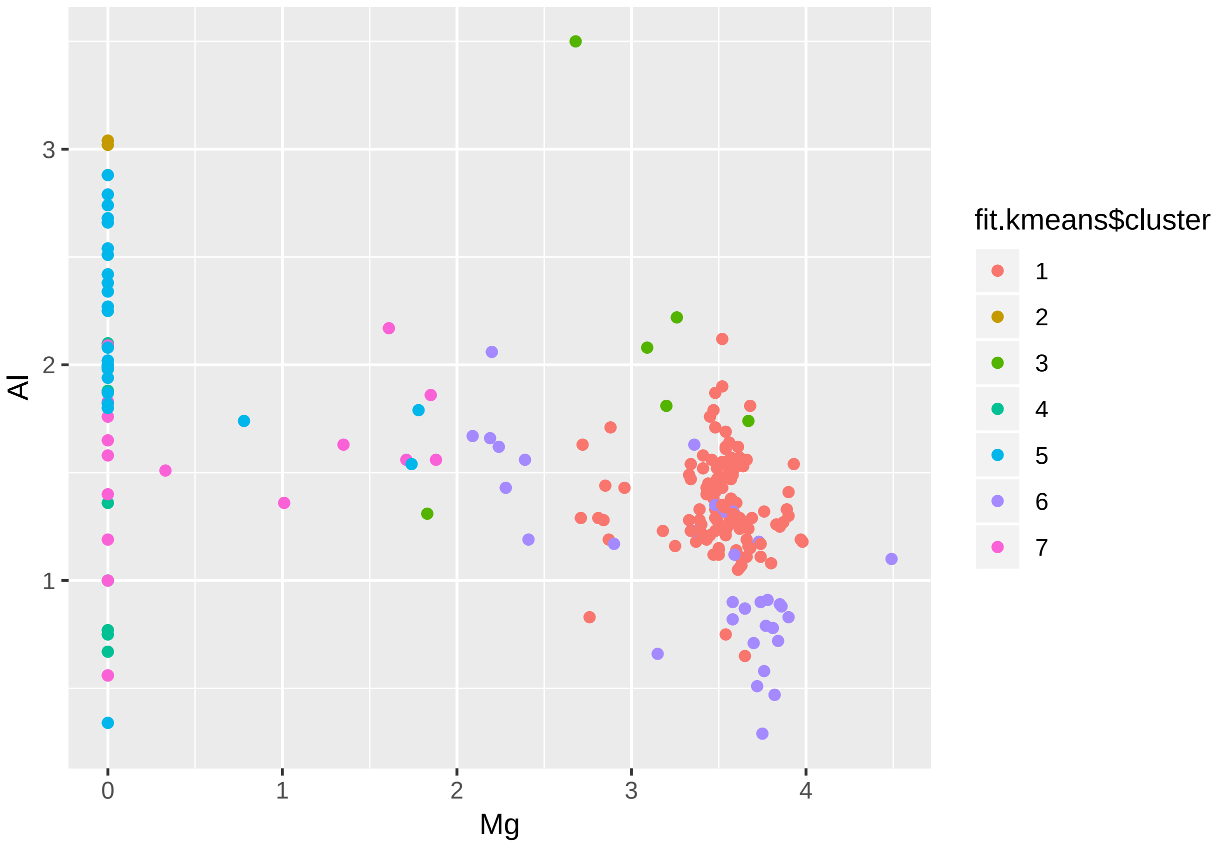
```
table(fit.kmeans$cluster, Glass$Type)
```

```
##
##      1  2  3  5  6  7
##   1 48 59 13  0  0  1
##   2  0  0  0  2  0  0
##   3  0  2  0  1  0  3
##   4  0  7  0  0  0  0
##   5  0  0  0  0  3 23
##   6 22  4  4  0  4  1
##   7  0  4  0 10  2  1
```

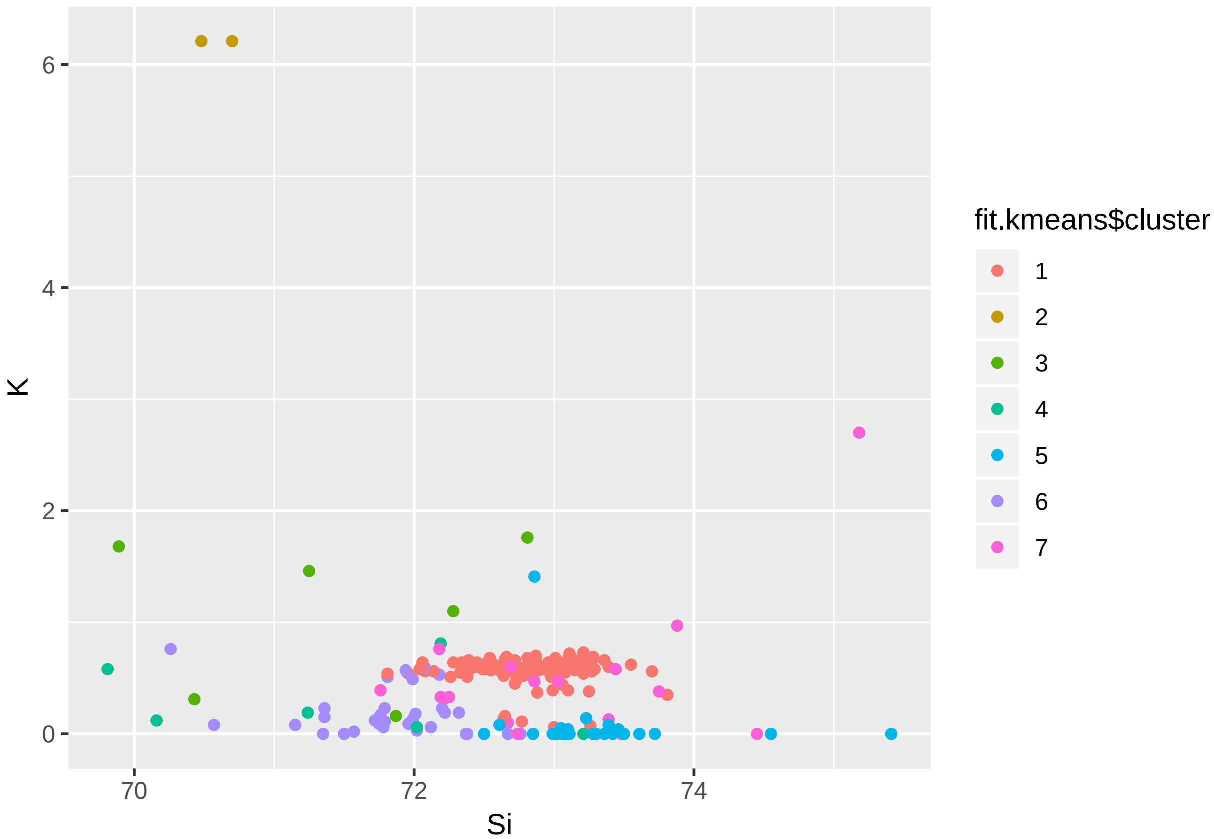
```
fit.kmeans$cluster <- as.factor(fit.kmeans$cluster)
ggplot(Glass, aes(RI, Na, color = fit.kmeans$cluster)) + geom_point()
```




```
fit.kmeans$cluster <- as.factor(fit.kmeans$cluster)
ggplot(Glass, aes(Mg, Al, color = fit.kmeans$cluster)) + geom_point()
```




```
fit.kmeans$cluster <- as.factor(fit.kmeans$cluster)
ggplot(Glass, aes(Si, K, color = fit.kmeans$cluster)) + geom_point()
```

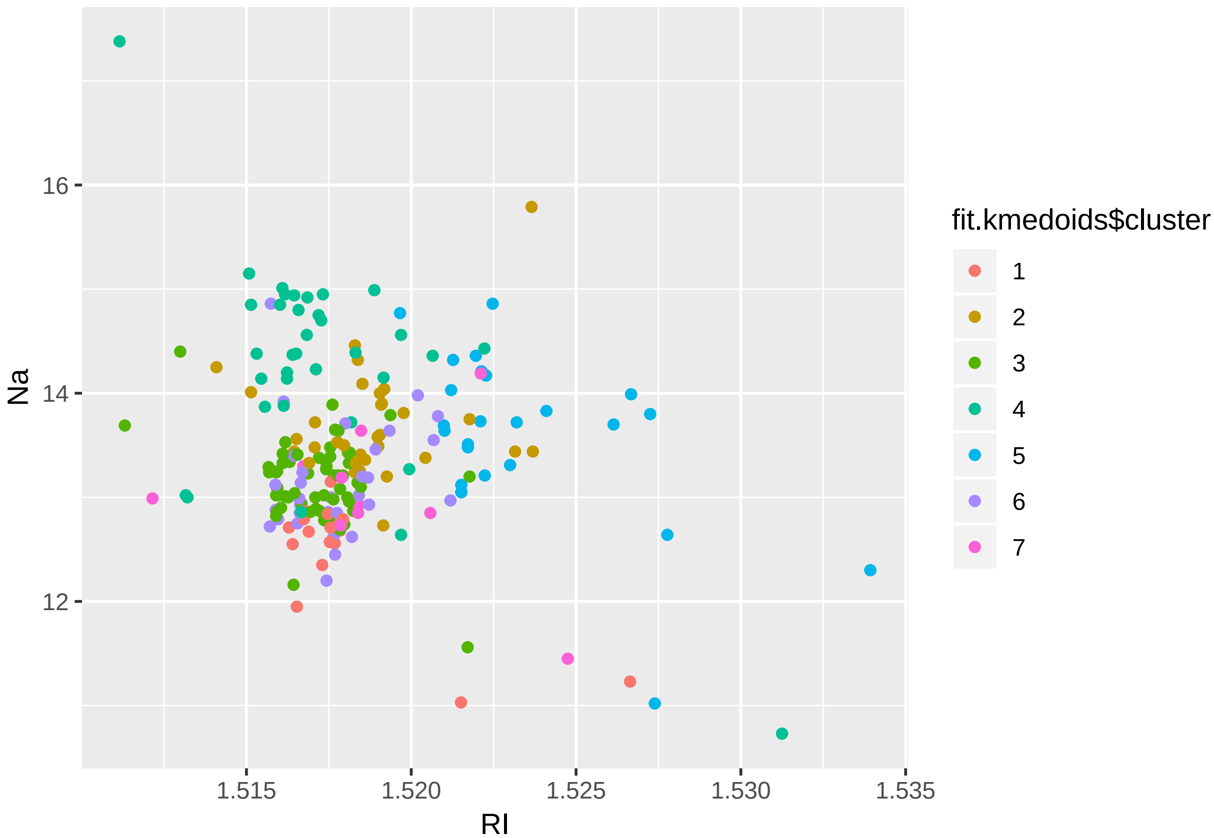


Getting insights from K-Medoids Clustering

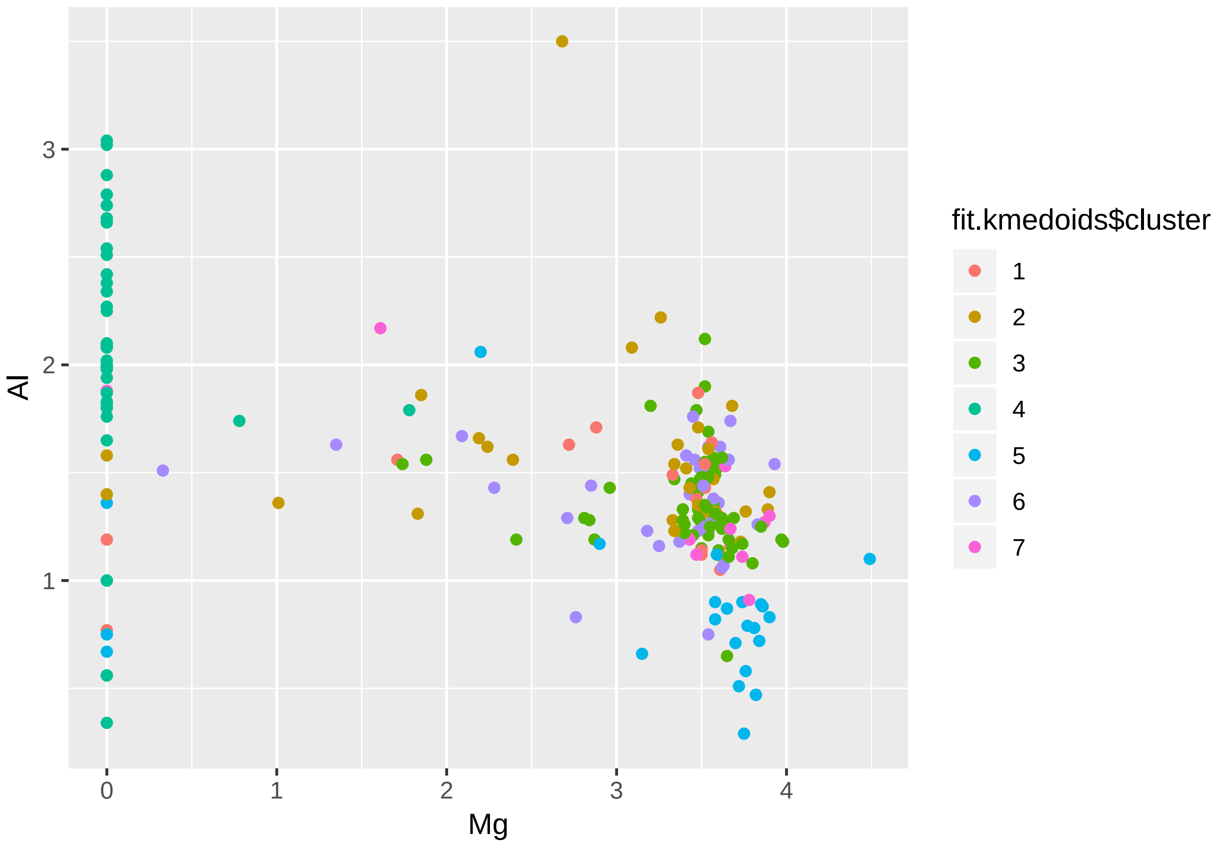
```
(fastiris <- table(fit.kmedoids$cluster, Glass[,10]))
```

```
##
##       1  2  3  5  6  7
##    1  9  7  0  1  0  1
##    2  8 11  3  4  3  3
##    3 24 25  8  1  2  1
##    4  0  3  0  5  4 23
##    5 17  6  2  0  0  1
##    6 10 18  3  1  0  0
##    7  2  6  1  1  0  0
```

```
fit.kmedoids$cluster <- as.factor(fit.kmedoids$cluster)
ggplot(Glass, aes(RI, Na, color = fit.kmedoids$cluster)) + geom_point()
```




```r
fit.kmedoids$cluster <- as.factor(fit.kmedoids$cluster)
ggplot(Glass, aes(Mg, Al, color = fit.kmedoids$cluster)) + geom_point()
```




```
fit.kmedoids$cluster <- as.factor(fit.kmedoids$cluster)
ggplot(Glass, aes(Si, K, color = fit.kmedoids$cluster)) + geom_point()
```

Conclusion

With better accuracy and kappa measures, Random Forest has outperformed other competitors on Glass Dataset while Hierarchical Agglomerative Clustering is the winner when compared with K-Means and K-Medoids Clustering on Glass Dataset as it has clustered data better evident from the Cluster Plot and Cluster Dendrogram.