VIT-AP UNIVERSITY, ANDHRA PRADESH

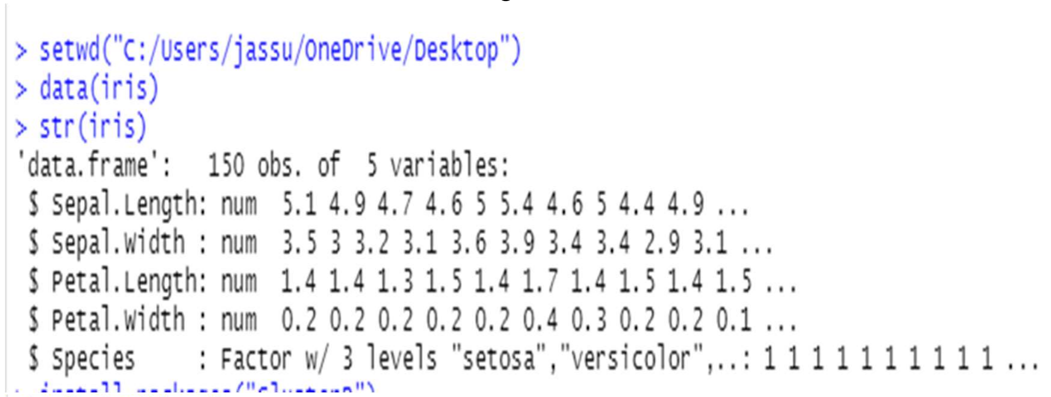
CSE4027 – Data Analytics - Lab Sheet :10

Academic year: 2022-2023 **Branch/ Class:** B.Tech
Semester: Fall **Date:23-11-22**
Faculty Name: Dr Syed Khasim **School:** SCOPE
Student name: MAJJIGA JASWANTH **Reg. no.:20BCD7171**

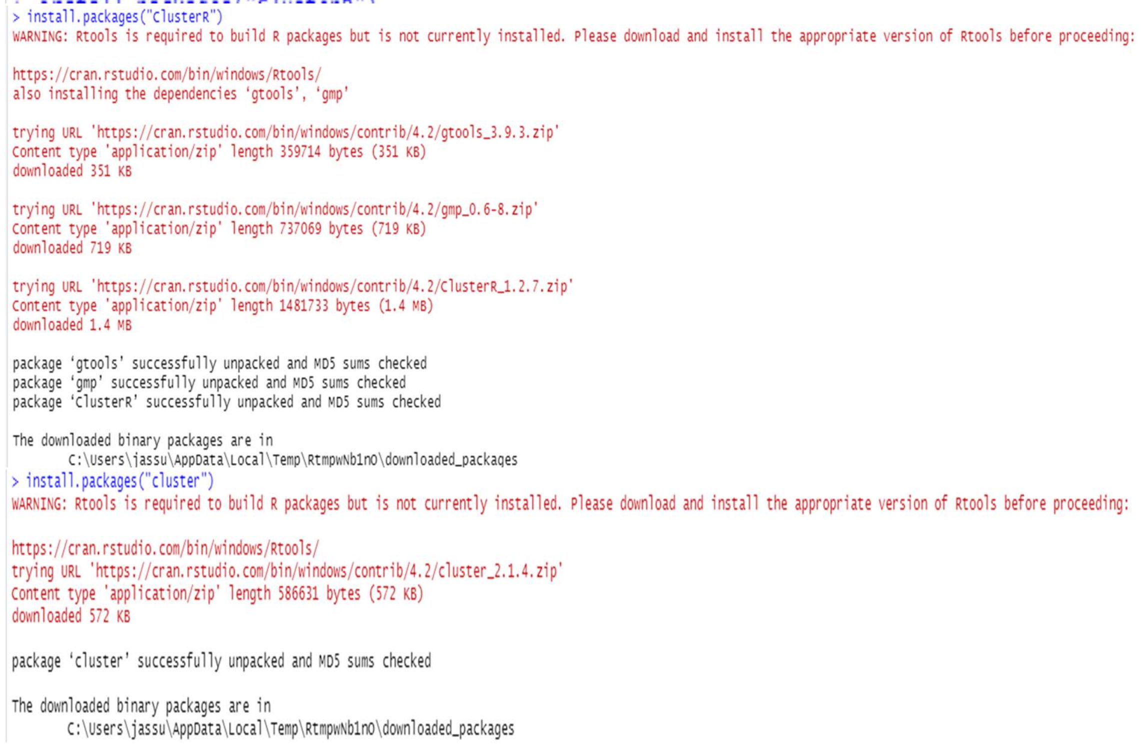
LAB 10

Clustering and Classification Algorithm implementation using R.

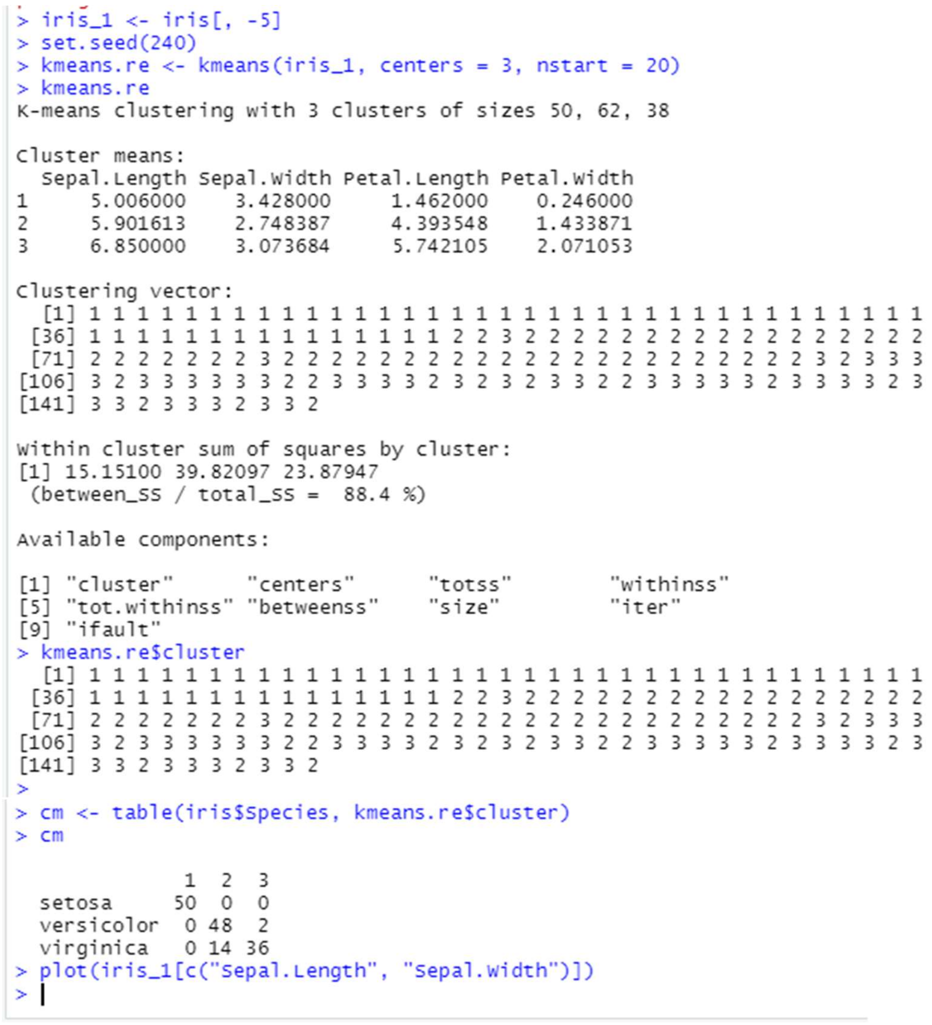
1. Use iris dataset for k means clustering in R



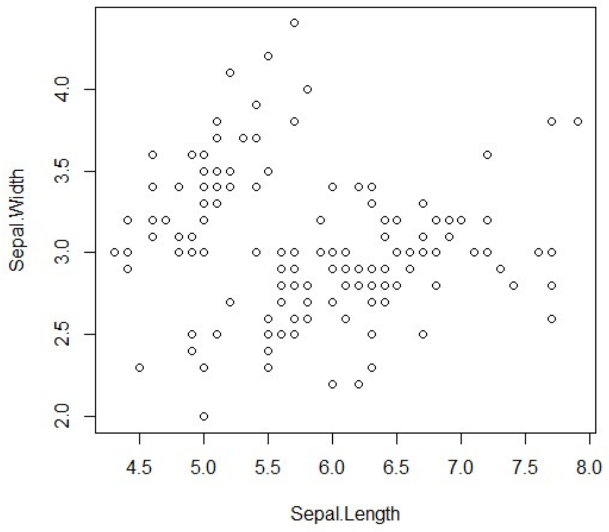
```
> setwd("C:/Users/jassu/OneDrive/Desktop")
> data(iris)
> str(iris)
'data.frame':	150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```



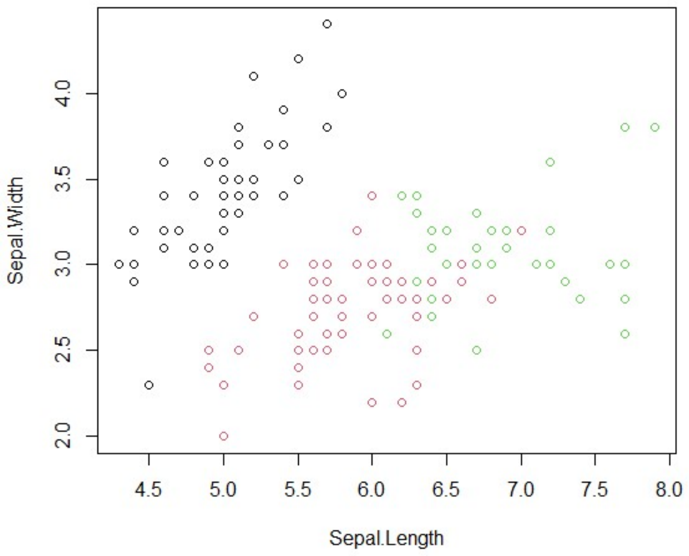
```
> install.packages("ClusterR")
WARNING: Rtools is required to build R packages but is not currently installed. Please download and install the appropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
also installing the dependencies ‘gtools’, ‘gmp’

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.2/gtools_3.9.3.zip'
Content type 'application/zip' length 359714 bytes (351 KB)
downloaded 351 KB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.2/gmp_0.6-8.zip'
Content type 'application/zip' length 737069 bytes (719 KB)
downloaded 719 KB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.2/ClusterR_1.2.7.zip'
Content type 'application/zip' length 1481733 bytes (1.4 MB)
downloaded 1.4 MB

package ‘gtools’ successfully unpacked and MD5 sums checked
package ‘gmp’ successfully unpacked and MD5 sums checked
package ‘ClusterR’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
	C:\Users\jassu\AppData\Local\Temp\RtmpwNb1n0\downloaded_packages
> install.packages("cluster")
WARNING: Rtools is required to build R packages but is not currently installed. Please download and install the appropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.2/cluster_2.1.4.zip'
Content type 'application/zip' length 586631 bytes (572 KB)
downloaded 572 KB

package ‘cluster’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
	C:\Users\jassu\AppData\Local\Temp\RtmpwNb1n0\downloaded_packages
```

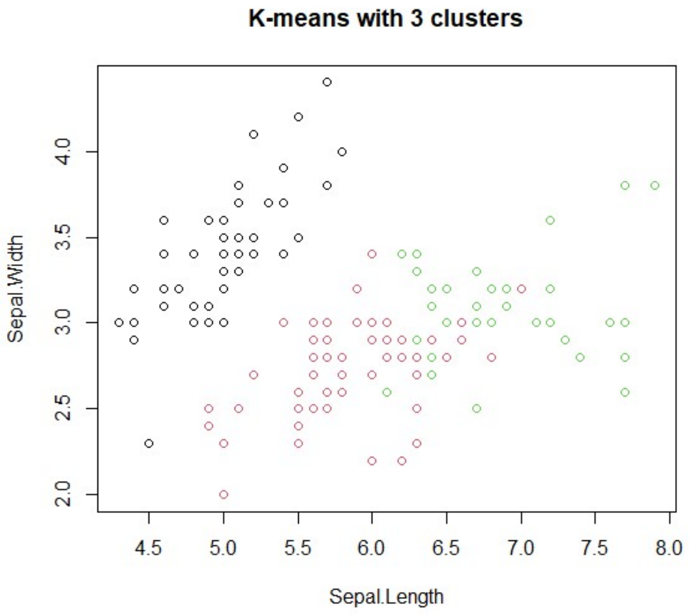


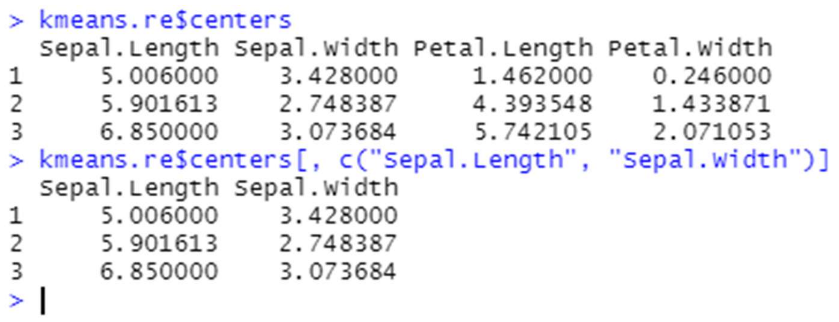
```
> iris_1 <- iris[, -5]
> set.seed(240)
> kmeans.re <- kmeans(iris_1, centers = 3, nstart = 20)
> kmeans.re
K-means clustering with 3 clusters of sizes 50, 62, 38

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     5.006000    3.428000     1.462000    0.246000
2     5.901613    2.748387     4.393548    1.433871
3     6.850000    3.073684     5.742105    2.071053

Clustering vector:
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [71] 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3
[106] 3 2 3 3 3 3 3 3 2 2 3 3 3 3 2 3 2 3 2 3 3 2 2 3 3 3 3 3 2 3 3 3 3 2 3
[141] 3 3 2 3 3 3 2 3 3 2

Within cluster sum of squares by cluster:
[1] 15.15100 39.82097 23.87947
 (between_SS / total_SS =  88.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"
[5] "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
> kmeans.re$cluster
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [71] 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3
[106] 3 2 3 3 3 3 3 3 2 2 3 3 3 3 2 3 2 3 2 3 3 2 2 3 3 3 3 3 2 3 3 3 3 2 3
[141] 3 3 2 3 3 3 2 3 3 2
>
> cm <- table(iris$Species, kmeans.re$cluster)
> cm

             1  2  3
  setosa     50  0  0
  versicolor  0 48  2
  virginica   0 14 36
> plot(iris_1[c("Sepal.Length", "Sepal.Width")])
> 
```



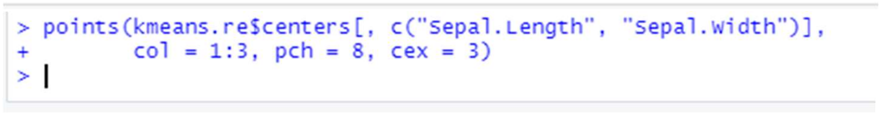
```
> plot(iris_1[c("Sepal.Length", "Sepal.Width")],col = kmeans.re$cluster)
> plot(iris_1[c("Sepal.Length", "Sepal.Width")],
+      col = kmeans.re$cluster,
+      main = "K-means with 3 clusters")
>
```



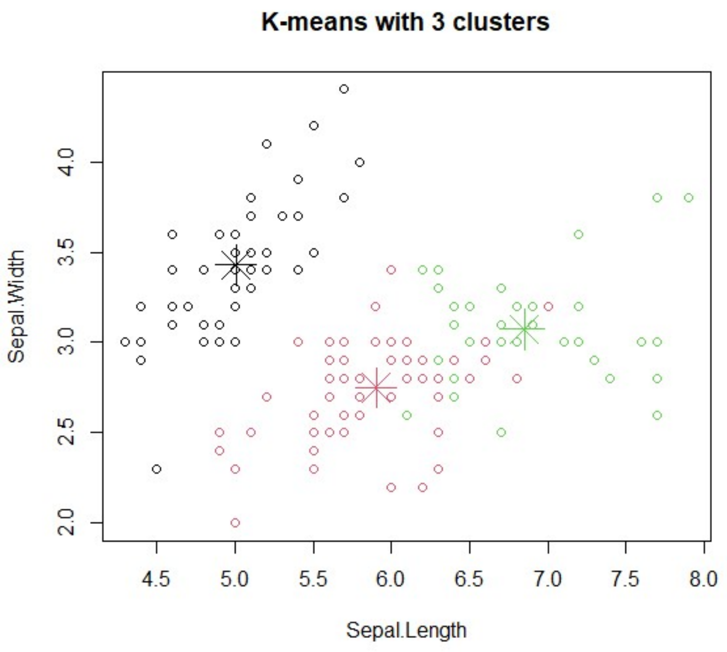


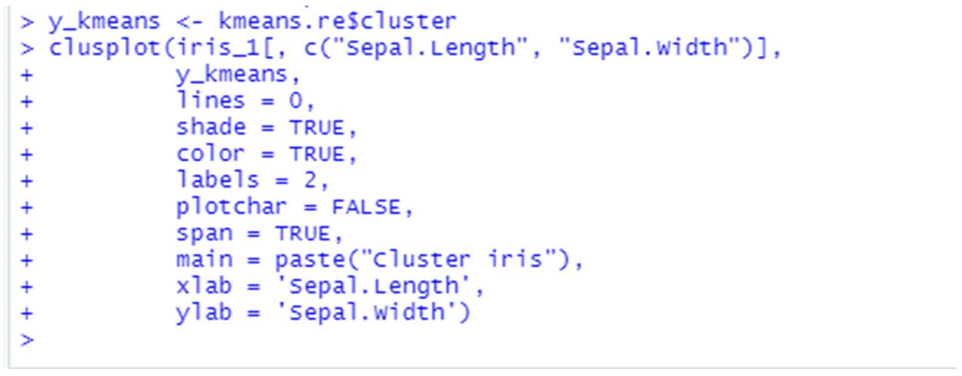


```
> kmeans.re$centers
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     5.006000    3.428000     1.462000    0.246000
2     5.901613    2.748387     4.393548    1.433871
3     6.850000    3.073684     5.742105    2.071053
> kmeans.re$centers[, c("Sepal.Length", "Sepal.Width")]
  Sepal.Length Sepal.Width
1     5.006000    3.428000
2     5.901613    2.748387
3     6.850000    3.073684
> |
```

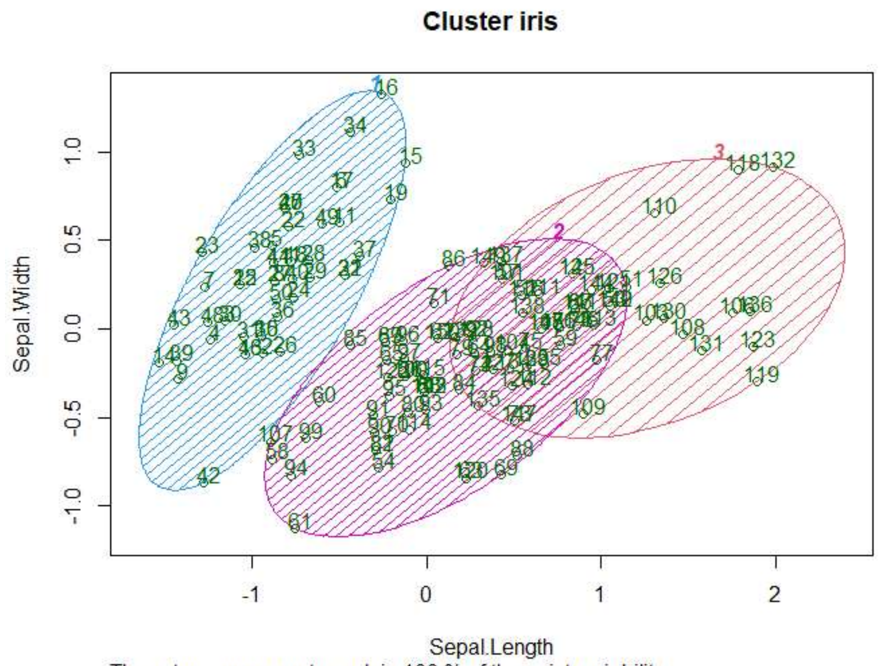


```
> points(kmeans.re$centers[, c("Sepal.Length", "Sepal.Width")],
+        col = 1:3, pch = 8, cex = 3)
> |
```



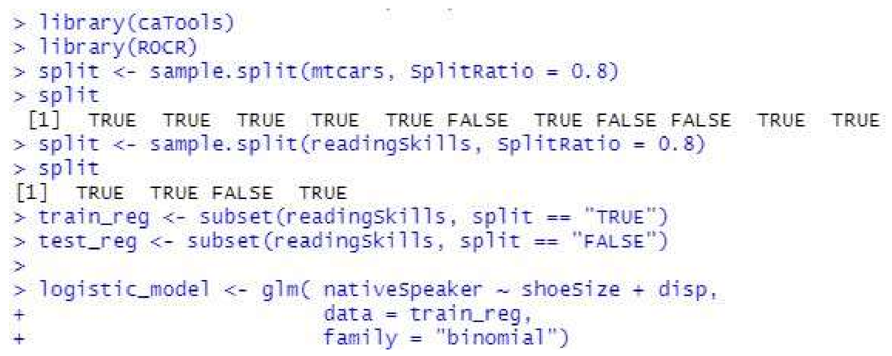


```
> y_kmeans <- kmeans.re$cluster
> clusplot(iris_1[, c("Sepal.Length", "Sepal.Width")],
+          y_kmeans,
+          lines = 0,
+          shade = TRUE,
+          color = TRUE,
+          labels = 2,
+          plotchar = FALSE,
+          span = TRUE,
+          main = paste("Cluster iris"),
+          xlab = 'Sepal.Length',
+          ylab = 'Sepal.Width')
>
```

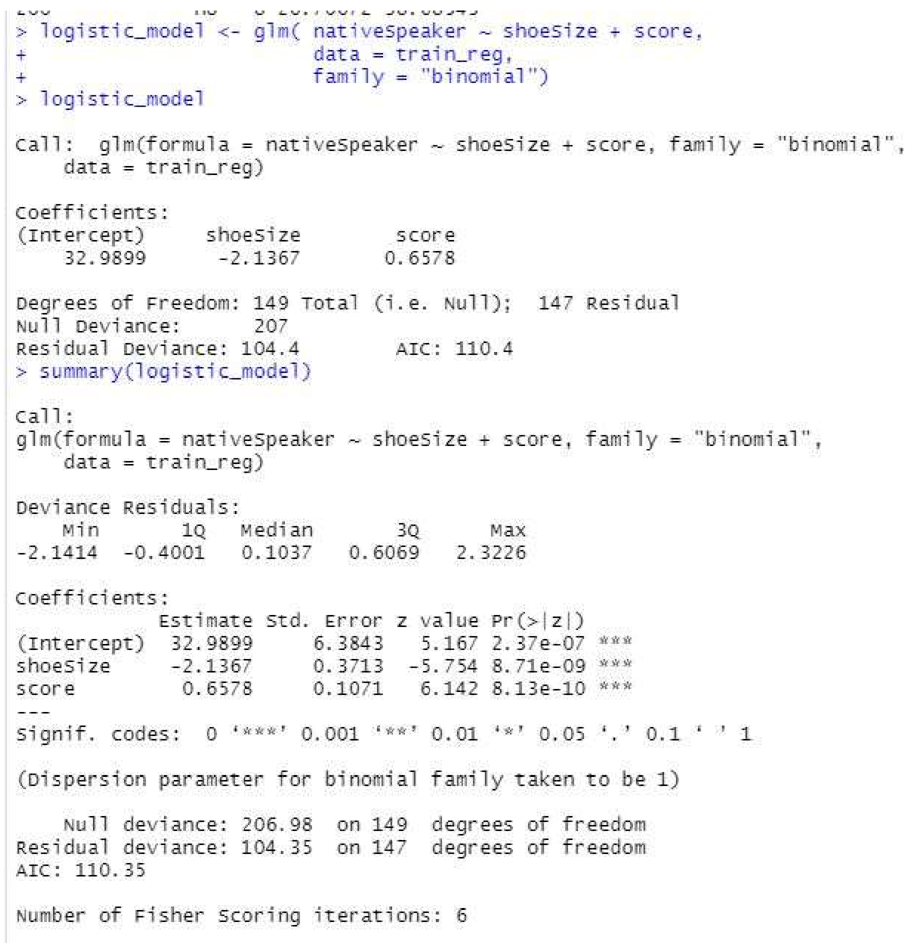


These two components explain 100 % of the point variability.

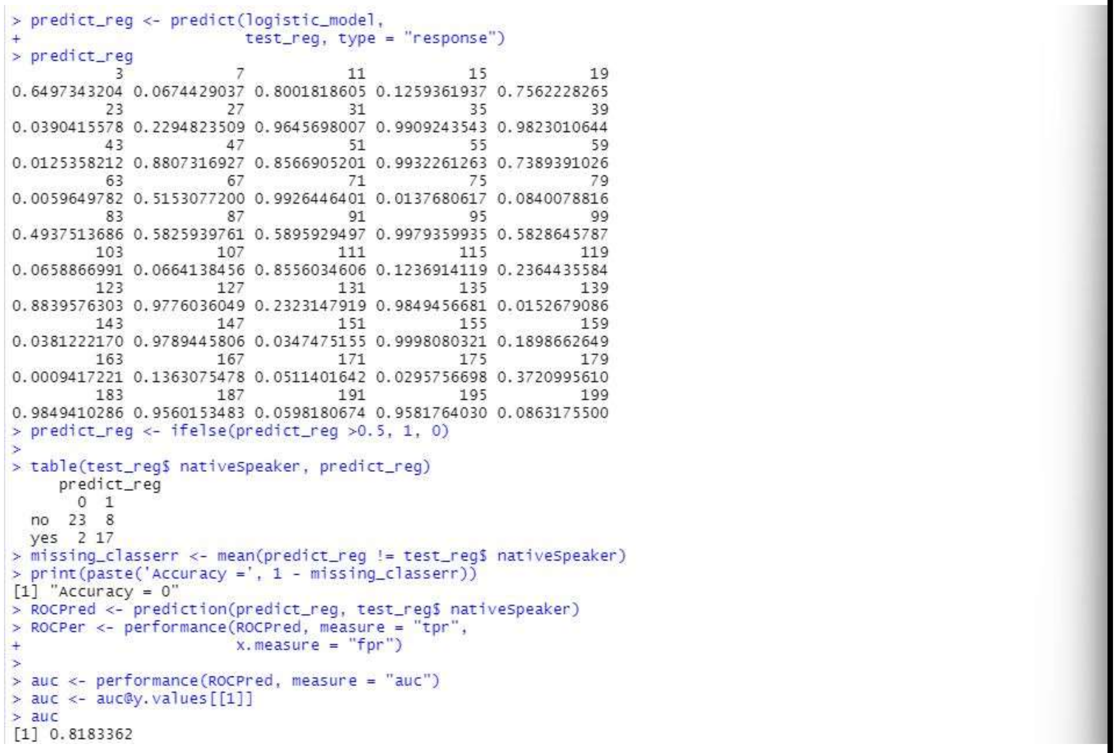
2. Use readingSkillsdataset for all classification practice which is default in party package
   a. Logistic regression



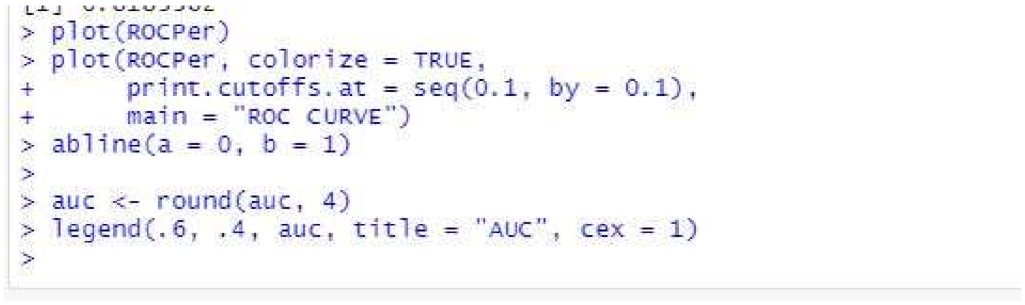
```
> library(caTools)
> library(ROCR)
> split <- sample.split(mtcars, SplitRatio = 0.8)
> split
 [1]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE
> split <- sample.split(readingSkills, SplitRatio = 0.8)
> split
[1]  TRUE  TRUE FALSE  TRUE
> train_reg <- subset(readingSkills, split == "TRUE")
> test_reg <- subset(readingSkills, split == "FALSE")
>
> logistic_model <- glm( nativeSpeaker ~ shoeSize + disp,
+                        data = train_reg,
+                        family = "binomial")
```



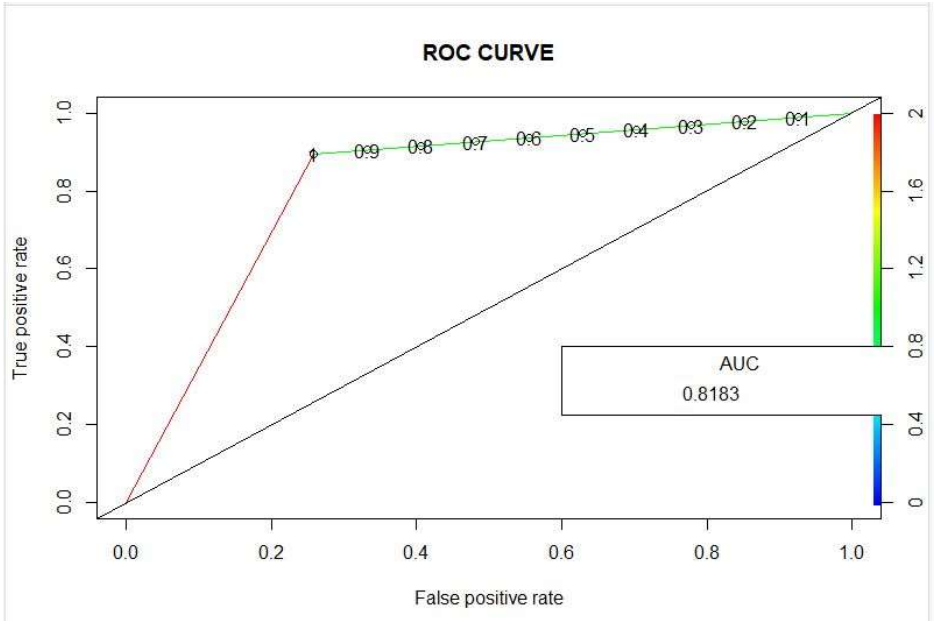
```
> logistic_model <- glm( nativeSpeaker ~ shoeSize + score,
+                        data = train_reg,
+                        family = "binomial")
> logistic_model

Call:  glm(formula = nativeSpeaker ~ shoeSize + score, family = "binomial",
    data = train_reg)

Coefficients:
(Intercept)     shoeSize        score
    32.9899      -2.1367       0.6578

Degrees of Freedom: 149 Total (i.e. Null);  147 Residual
Null Deviance:      207
Residual Deviance: 104.4        AIC: 110.4
> summary(logistic_model)

Call:
glm(formula = nativeSpeaker ~ shoeSize + score, family = "binomial",
    data = train_reg)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1414  -0.4001   0.1037   0.6069   2.3226

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  32.9899     6.3843   5.167 2.37e-07 ***
shoeSize     -2.1367     0.3713  -5.754 8.71e-09 ***
score         0.6578     0.1071   6.142 8.13e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 206.98  on 149  degrees of freedom
Residual deviance: 104.35  on 147  degrees of freedom
AIC: 110.35

Number of Fisher Scoring iterations: 6
```



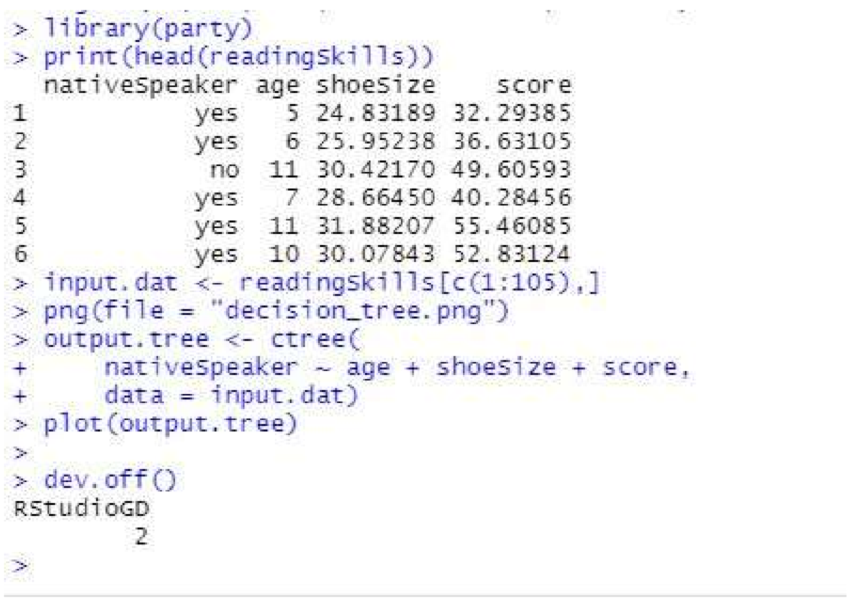
```
> predict_reg <- predict(logistic_model,
+                        test_reg, type = "response")
> predict_reg
           3            7           11           15           19
0.6497343204 0.0674429037 0.8001818605 0.1259361937 0.7562228265
          23           27           31           35           39
0.0390415578 0.2294823509 0.9645698007 0.9909243543 0.9823010644
          43           47           51           55           59
0.0125358212 0.8807316927 0.8566905201 0.9932261263 0.7389391026
          63           67           71           75           79
0.0059649782 0.5153077200 0.9926446401 0.0137680617 0.0840078816
          83           87           91           95           99
0.4937513686 0.5825939761 0.5895929497 0.9979359935 0.5828645787
         103          107          111          115          119
0.0658866991 0.0664138456 0.8556034606 0.1236914119 0.2364435584
         123          127          131          135          139
0.8839576303 0.9776036049 0.2323147919 0.9849456681 0.0152679086
         143          147          151          155          159
0.0381222170 0.9789445806 0.0347475155 0.9998080321 0.1898662649
         163          167          171          175          179
0.0009417221 0.1363075478 0.0511401642 0.0295756698 0.3720995610
         183          187          191          195          199
0.9849410286 0.9560153483 0.0598180674 0.9581764030 0.0863175500
> predict_reg <- ifelse(predict_reg >0.5, 1, 0)
>
> table(test_reg$ nativeSpeaker, predict_reg)
     predict_reg
       0  1
  no  23  8
  yes  2 17
> missing_classerr <- mean(predict_reg != test_reg$ nativeSpeaker)
> print(paste('Accuracy =', 1 - missing_classerr))
[1] "Accuracy = 0"
> ROCPred <- prediction(predict_reg, test_reg$ nativeSpeaker)
> ROCPer <- performance(ROCPred, measure = "tpr",
+                       x.measure = "fpr")
>
> auc <- performance(ROCPred, measure = "auc")
> auc <- auc@y.values[[1]]
> auc
[1] 0.8183362
```



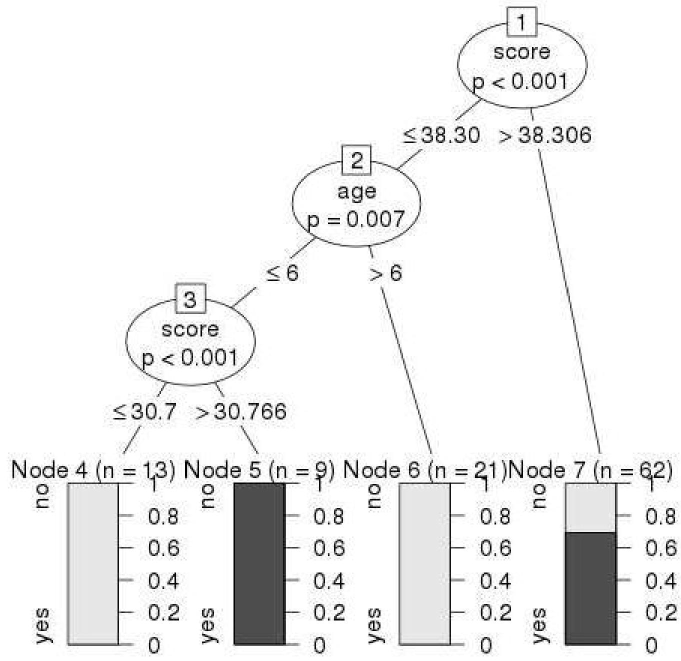
```
> plot(ROCPer)
> plot(ROCPer, colorize = TRUE,
+      print.cutoffs.at = seq(0.1, by = 0.1),
+      main = "ROC CURVE")
> abline(a = 0, b = 1)
>
> auc <- round(auc, 4)
> legend(.6, .4, auc, title = "AUC", cex = 1)
>
```



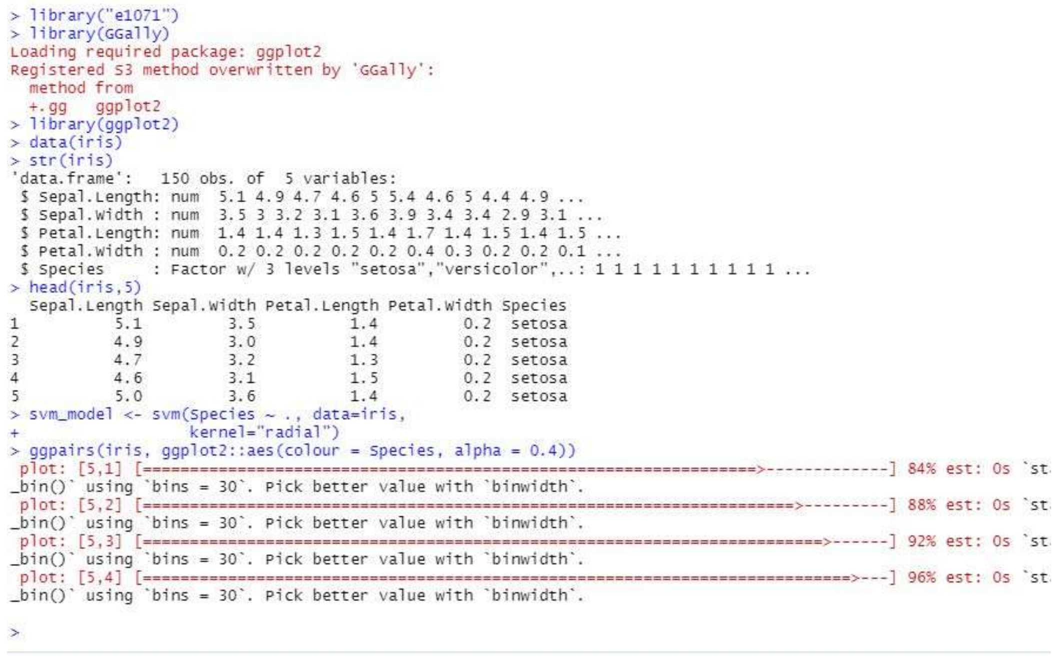
b. Decision trees



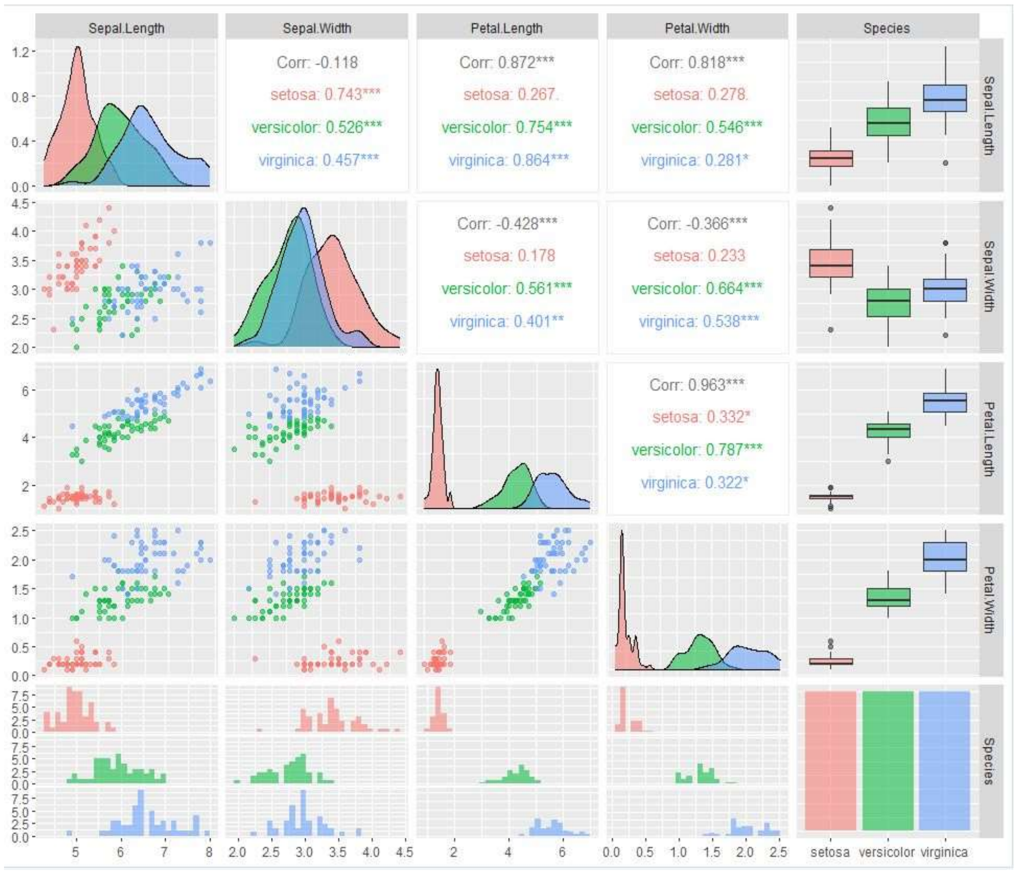
```
> library(party)
> print(head(readingSkills))
  nativeSpeaker age shoeSize    score
1           yes   5 24.83189 32.29385
2           yes   6 25.95238 36.63105
3            no  11 30.42170 49.60593
4           yes   7 28.66450 40.28456
5           yes  11 31.88207 55.46085
6           yes  10 30.07843 52.83124
> input.dat <- readingSkills[c(1:105),]
> png(file = "decision_tree.png")
> output.tree <- ctree(
+     nativeSpeaker ~ age + shoeSize + score,
+     data = input.dat)
> plot(output.tree)
>
> dev.off()
RStudioGD
        2
>
```

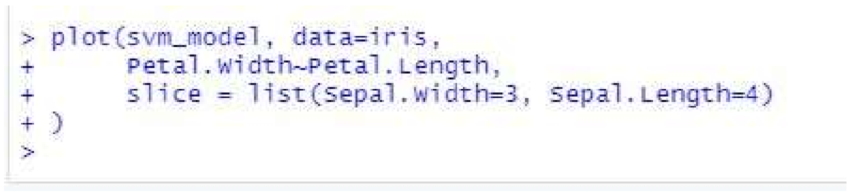


c. Support Vector Machines (iris data – default)

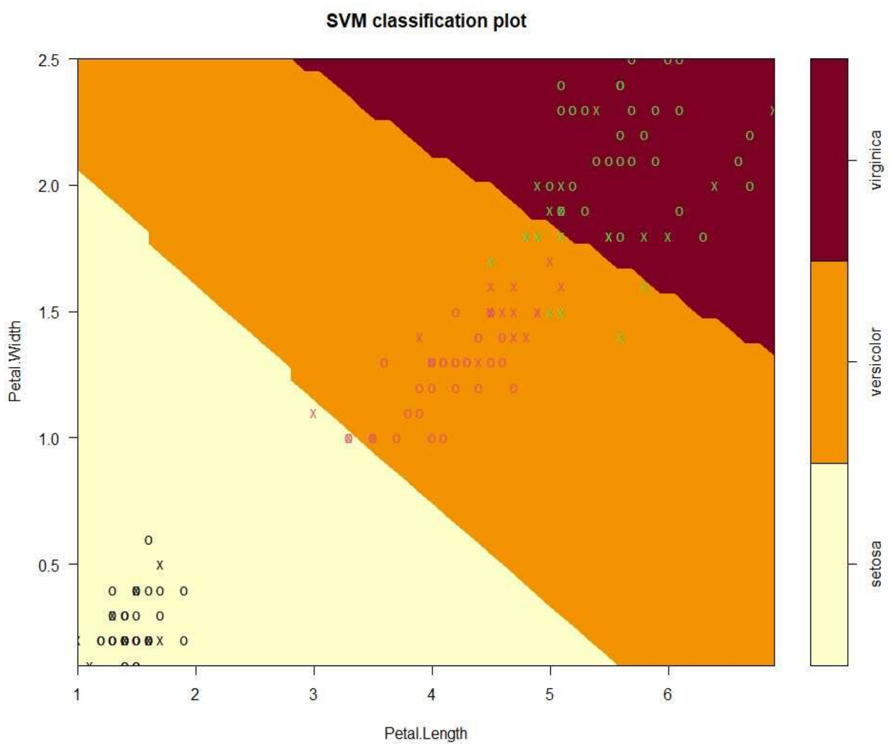


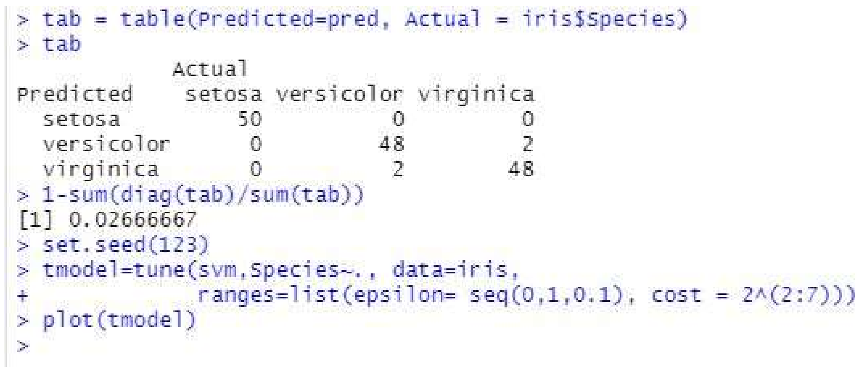
```
> library("e1071")
> library(GGally)
Loading required package: ggplot2
Registered S3 method overwritten by 'GGally':
  method from
  +.gg   ggplot2
> library(ggplot2)
> data(iris)
> str(iris)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
> head(iris,5)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
> svm_model <- svm(Species ~ ., data=iris,
+                  kernel="radial")
> ggpairs(iris, ggplot2::aes(colour = Species, alpha = 0.4))
 plot: [5,1] [==================================================================>-------------] 84% est: 0s `st
_bin()` using `bins = 30`. Pick better value with `binwidth`.
 plot: [5,2] [======================================================================>---------] 88% est: 0s `st
_bin()` using `bins = 30`. Pick better value with `binwidth`.
 plot: [5,3] [=========================================================================>------] 92% est: 0s `st
_bin()` using `bins = 30`. Pick better value with `binwidth`.
 plot: [5,4] [============================================================================>---] 96% est: 0s `st
_bin()` using `bins = 30`. Pick better value with `binwidth`.

>
```



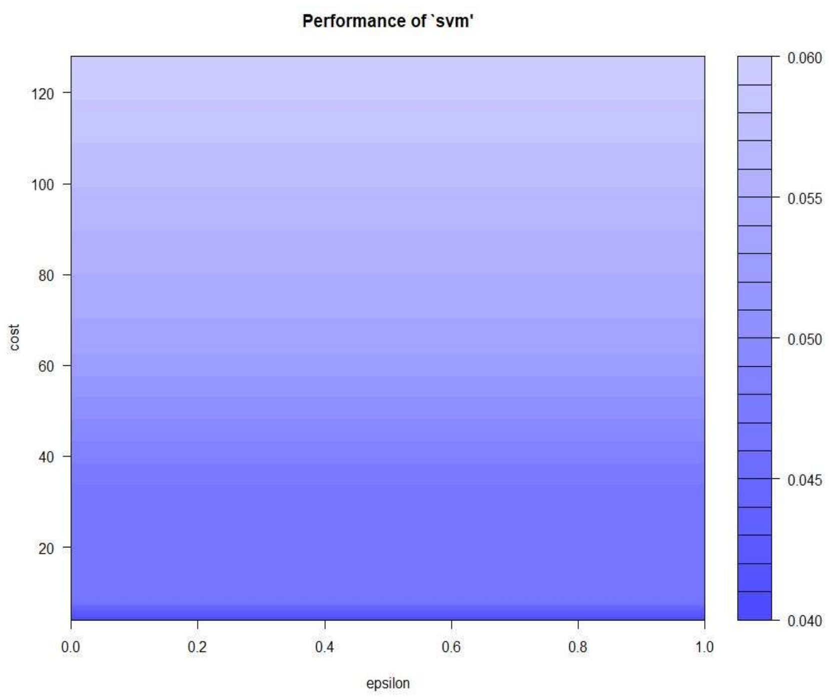


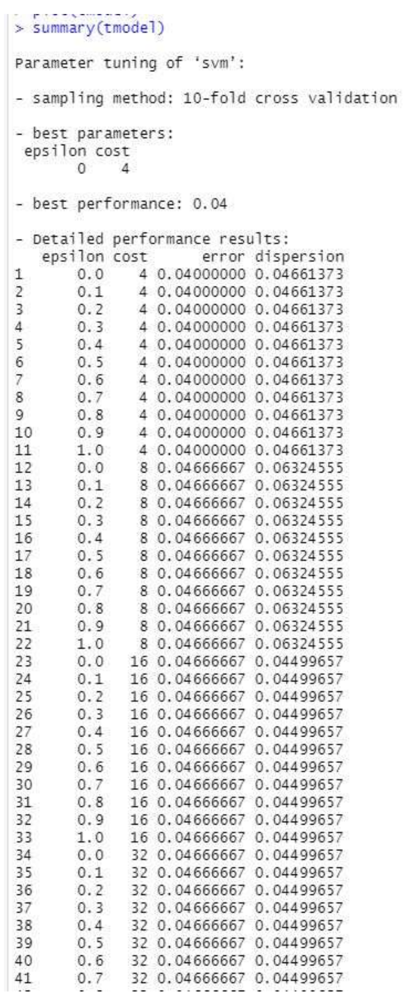
```
> plot(svm_model, data=iris,
+       Petal.Width~Petal.Length,
+       slice = list(Sepal.Width=3, Sepal.Length=4)
+ )
>
```



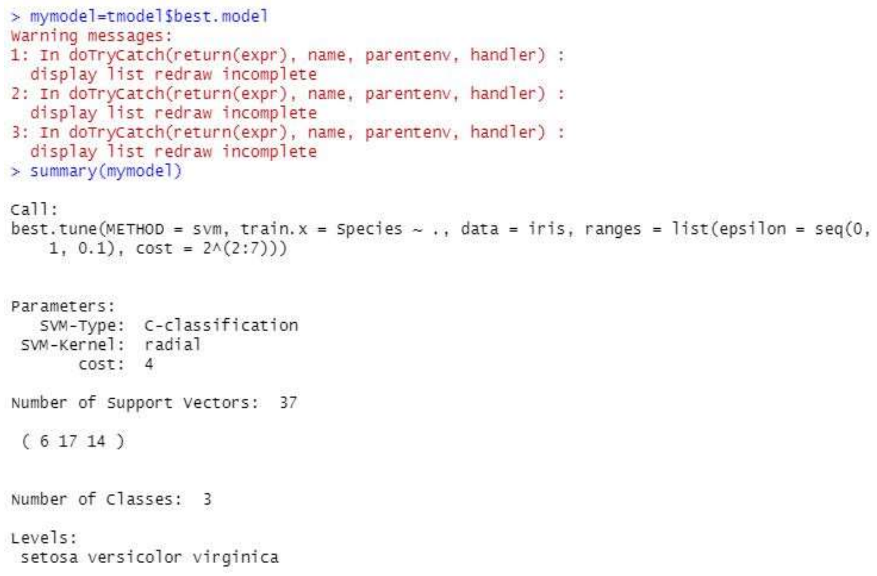


```
> tab = table(Predicted=pred, Actual = iris$Species)
> tab
            Actual
Predicted    setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         48         2
  virginica       0          2        48
> 1-sum(diag(tab)/sum(tab))
[1] 0.02666667
> set.seed(123)
> tmodel=tune(svm,Species~., data=iris,
+             ranges=list(epsilon= seq(0,1,0.1), cost = 2^(2:7)))
> plot(tmodel)
>
```

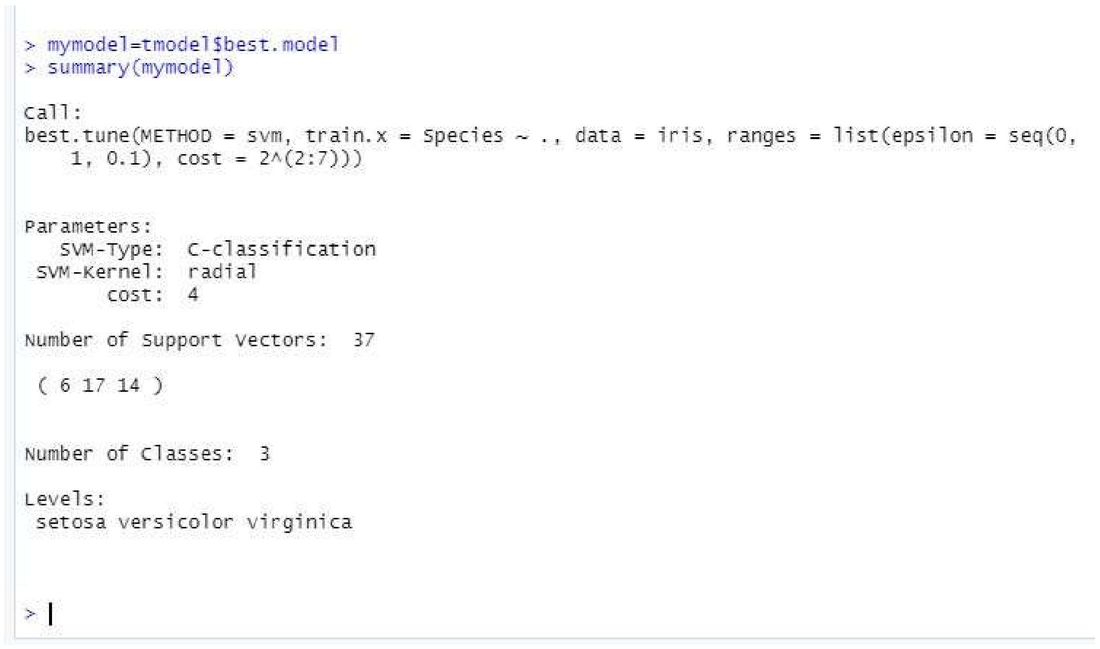




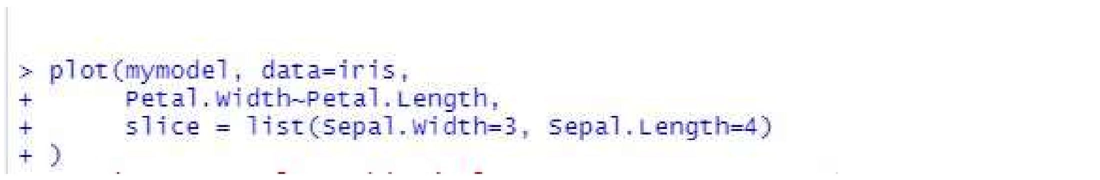
```
> summary(tmodel)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 epsilon cost
       0    4

- best performance: 0.04

- Detailed performance results:
   epsilon cost      error dispersion
1      0.0    4 0.04000000 0.04661373
2      0.1    4 0.04000000 0.04661373
3      0.2    4 0.04000000 0.04661373
4      0.3    4 0.04000000 0.04661373
5      0.4    4 0.04000000 0.04661373
6      0.5    4 0.04000000 0.04661373
7      0.6    4 0.04000000 0.04661373
8      0.7    4 0.04000000 0.04661373
9      0.8    4 0.04000000 0.04661373
10     0.9    4 0.04000000 0.04661373
11     1.0    4 0.04000000 0.04661373
12     0.0    8 0.04666667 0.06324555
13     0.1    8 0.04666667 0.06324555
14     0.2    8 0.04666667 0.06324555
15     0.3    8 0.04666667 0.06324555
16     0.4    8 0.04666667 0.06324555
17     0.5    8 0.04666667 0.06324555
18     0.6    8 0.04666667 0.06324555
19     0.7    8 0.04666667 0.06324555
20     0.8    8 0.04666667 0.06324555
21     0.9    8 0.04666667 0.06324555
22     1.0    8 0.04666667 0.06324555
23     0.0   16 0.04666667 0.04499657
24     0.1   16 0.04666667 0.04499657
25     0.2   16 0.04666667 0.04499657
26     0.3   16 0.04666667 0.04499657
27     0.4   16 0.04666667 0.04499657
28     0.5   16 0.04666667 0.04499657
29     0.6   16 0.04666667 0.04499657
30     0.7   16 0.04666667 0.04499657
31     0.8   16 0.04666667 0.04499657
32     0.9   16 0.04666667 0.04499657
33     1.0   16 0.04666667 0.04499657
34     0.0   32 0.04666667 0.04499657
35     0.1   32 0.04666667 0.04499657
36     0.2   32 0.04666667 0.04499657
37     0.3   32 0.04666667 0.04499657
38     0.4   32 0.04666667 0.04499657
39     0.5   32 0.04666667 0.04499657
40     0.6   32 0.04666667 0.04499657
41     0.7   32 0.04666667 0.04499657
```



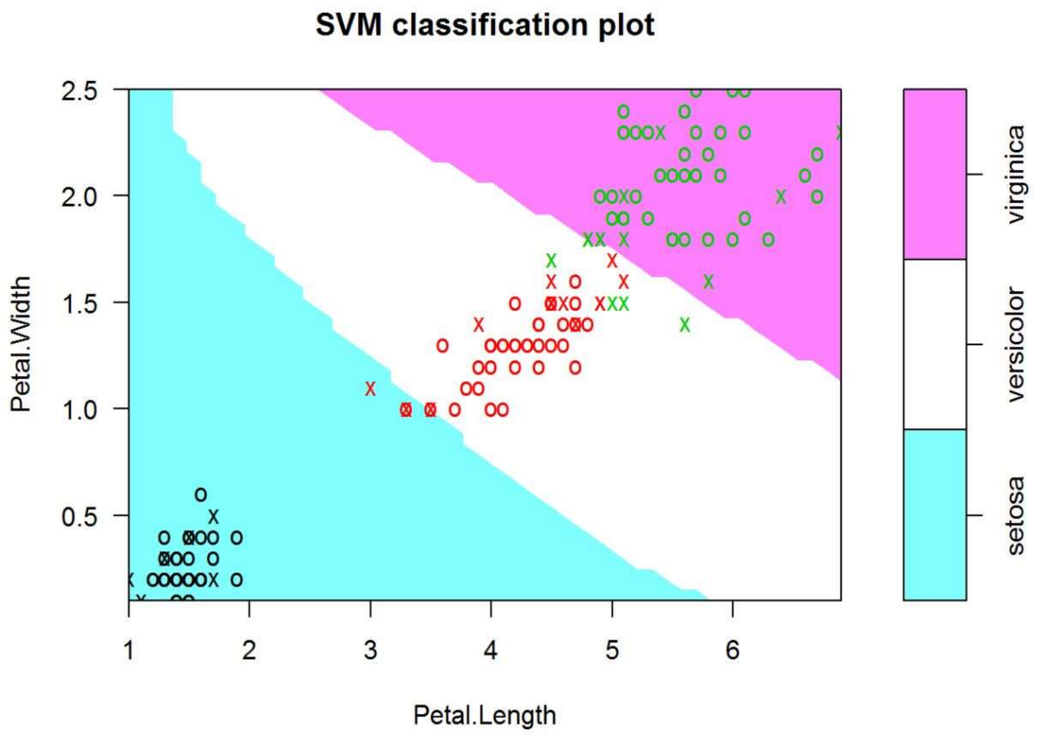
```
> mymodel=tmodel$best.model
Warning messages:
1: In doTryCatch(return(expr), name, parentenv, handler) :
  display list redraw incomplete
2: In doTryCatch(return(expr), name, parentenv, handler) :
  display list redraw incomplete
3: In doTryCatch(return(expr), name, parentenv, handler) :
  display list redraw incomplete
> summary(mymodel)

Call:
best.tune(METHOD = svm, train.x = Species ~ ., data = iris, ranges = list(epsilon = seq(0,
    1, 0.1), cost = 2^(2:7)))


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  4

Number of Support Vectors:  37

 ( 6 17 14 )


Number of Classes:  3

Levels:
 setosa versicolor virginica
```

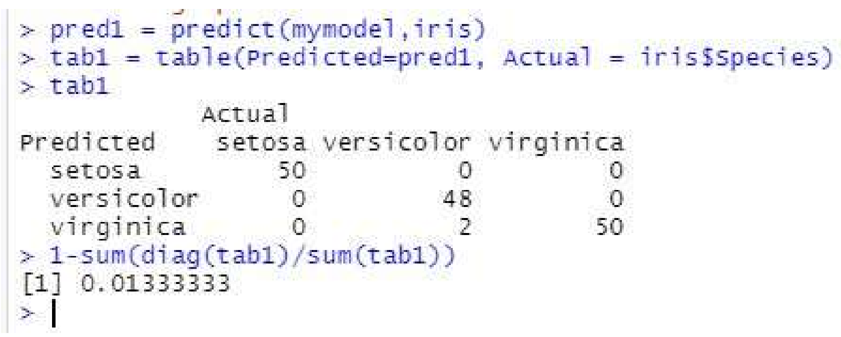


```
> mymodel=tmodel$best.model
> summary(mymodel)

Call:
best.tune(METHOD = svm, train.x = Species ~ ., data = iris, ranges = list(epsilon = seq(0,
    1, 0.1), cost = 2^(2:7)))


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  4

Number of Support Vectors:  37

 ( 6 17 14 )


Number of Classes:  3

Levels:
 setosa versicolor virginica



> |
```



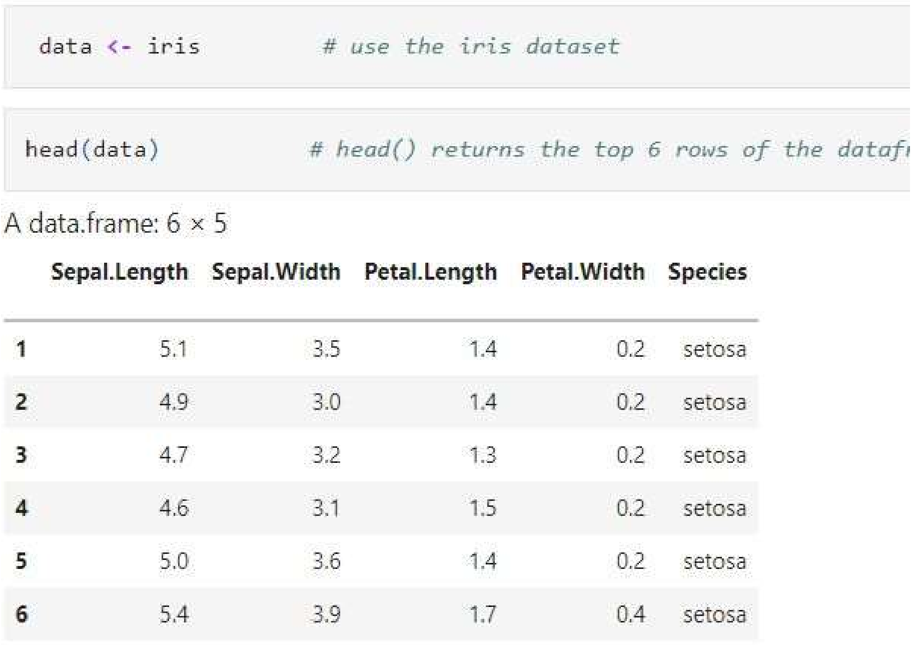
```
> plot(mymodel, data=iris,
+       Petal.Width~Petal.Length,
+       slice = list(Sepal.Width=3, Sepal.Length=4)
+ )
```





```
> pred1 = predict(mymodel,iris)
> tab1 = table(Predicted=pred1, Actual = iris$Species)
> tab1
            Actual
Predicted    setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         48         0
  virginica       0          2        50
> 1-sum(diag(tab1)/sum(tab1))
[1] 0.01333333
> 
```

d. Naive Bayes Classifier (use hsbdata.csv)

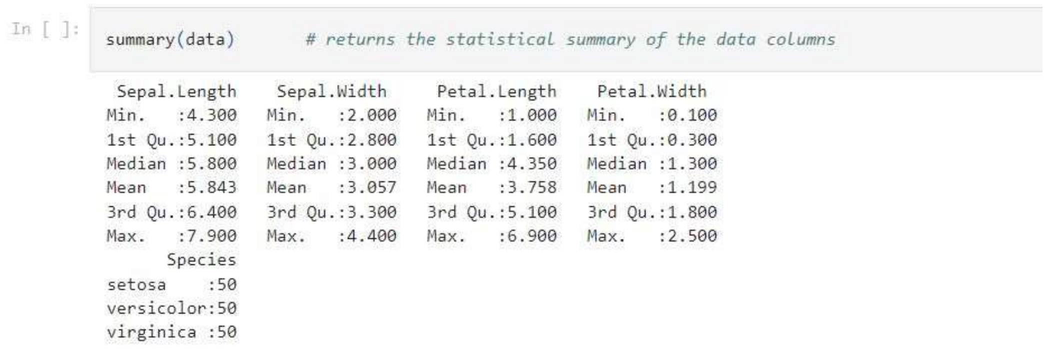


```
data <- iris          # use the iris dataset
```

```
head(data)          # head() returns the top 6 rows of the datafr
```

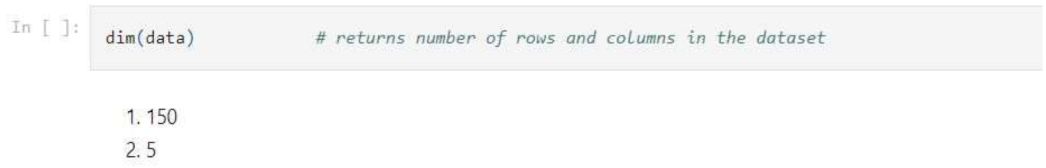
A data.frame: 6 × 5

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa



```
In [ ]: summary(data)       # returns the statistical summary of the data columns
```

```
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width   
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100  
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300  
 Median :5.800   Median :3.000   Median :4.350   Median :1.300  
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199  
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800  
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500  
       Species  
 setosa    :50  
 versicolor:50  
 virginica :50  
```



```
In [ ]: dim(data)          # returns number of rows and columns in the dataset
```

1. 150
2. 5

```
In [ ]:
# split the data into train-test with a ratio 80:20
split <- sample.split(iris, SplitRatio = 0.8)
train_data <- subset(data, split == "TRUE")
test_data <- subset(data, split == "FALSE")
```

```
In [ ]:
# Feature Scaling
train_scale <- scale(train_data[, 1:4])
test_scale <- scale(test_data[, 1:4])
```

```
In [ ]:
dim(train_data)
```

1. 120
2. 5

```
In [ ]:
dim(test_data)
```

1. 30
2. 5

```
set.seed(1)  # Setting Seed
classifier_naive <- naiveBayes(Species~ ., data = train_data)
classifier_naive
```

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
    setosa versicolor  virginica 
 0.3333333  0.3333333  0.3333333 

Conditional probabilities:
            Sepal.Length
Y             [,1]      [,2]
  setosa     5.020 0.3131314
  versicolor 5.965 0.5206259
  virginica  6.625 0.6581832

            Sepal.Width
Y              [,1]      [,2]
  setosa     3.4375 0.3739515
  versicolor 2.8000 0.3137858
  virginica  3.0075 0.3237501
```

```
            Petal.Length
Y                [,1]      [,2]
  setosa     1.4625 0.1734824
  versicolor 4.2625 0.4204317
  virginica  5.5650 0.5404414

            Petal.Width
Y                [,1]       [,2]
  setosa     0.2400 0.09001424
  versicolor 1.3275 0.20253205
  virginica  2.0400 0.27623847
```

In [ ]:

```
summary(classifier_naive)
```

```
          Length Class  Mode
apriori   3      table  numeric
tables    4      -none- list
levels    3      -none- character
isnumeric 4      -none- logical
call      4      -none- call
```

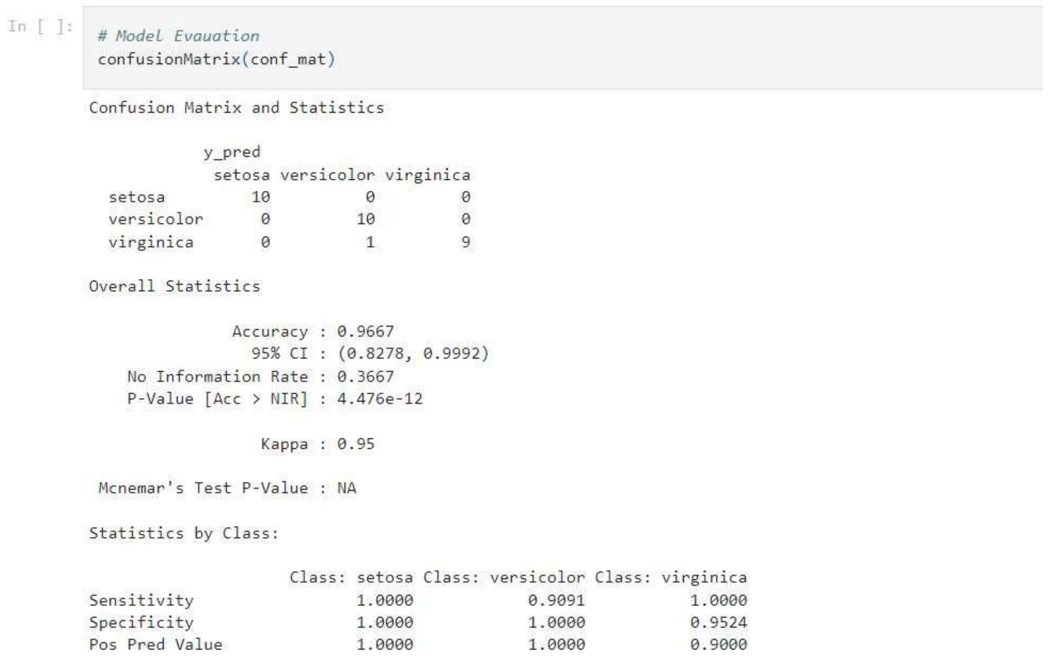
In [ ]:

```
# Predicting on test data'
y_pred <- predict(classifier_naive, newdata = test_data)
```

In [ ]:

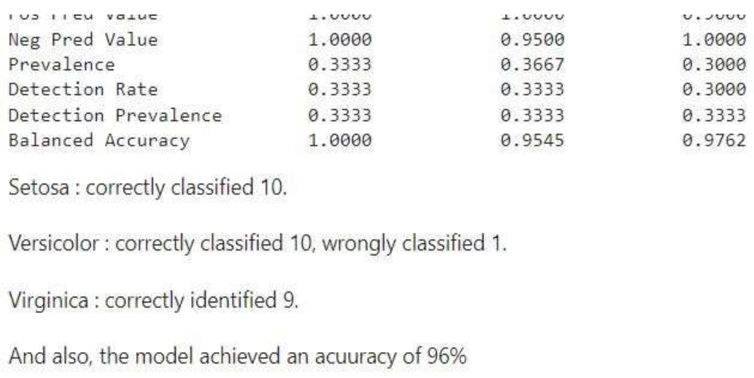
```
# Confusion Matrix
conf_mat <- table(test_data$Species,y_pred)
print(conf_mat)
```

```
            y_pred
             setosa versicolor virginica
  setosa         10          0         0
  versicolor      0         10         0
  virginica       0          1         9
```



```
In [ ]: # Model Evauation
        confusionMatrix(conf_mat)
```

```
Confusion Matrix and Statistics

            y_pred
             setosa versicolor virginica
  setosa         10          0         0
  versicolor      0         10         0
  virginica       0          1         9

Overall Statistics

               Accuracy : 0.9667
                 95% CI : (0.8278, 0.9992)
    No Information Rate : 0.3667
    P-Value [Acc > NIR] : 4.476e-12

                  Kappa : 0.95

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            0.9091           1.0000
Specificity                 1.0000            1.0000           0.9524
Pos Pred Value              1.0000            1.0000           0.9000
```



```
Neg Pred Value              1.0000            0.9500           1.0000
Prevalence                  0.3333            0.3667           0.3000
Detection Rate              0.3333            0.3333           0.3000
Detection Prevalence        0.3333            0.3333           0.3333
Balanced Accuracy           1.0000            0.9545           0.9762
```

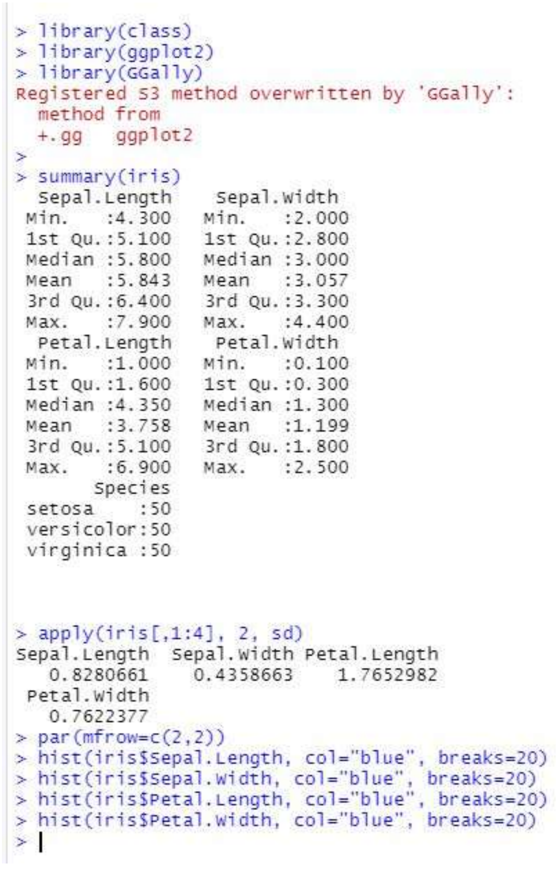
Setosa : correctly classified 10.

Versicolor : correctly classified 10, wrongly classified 1.

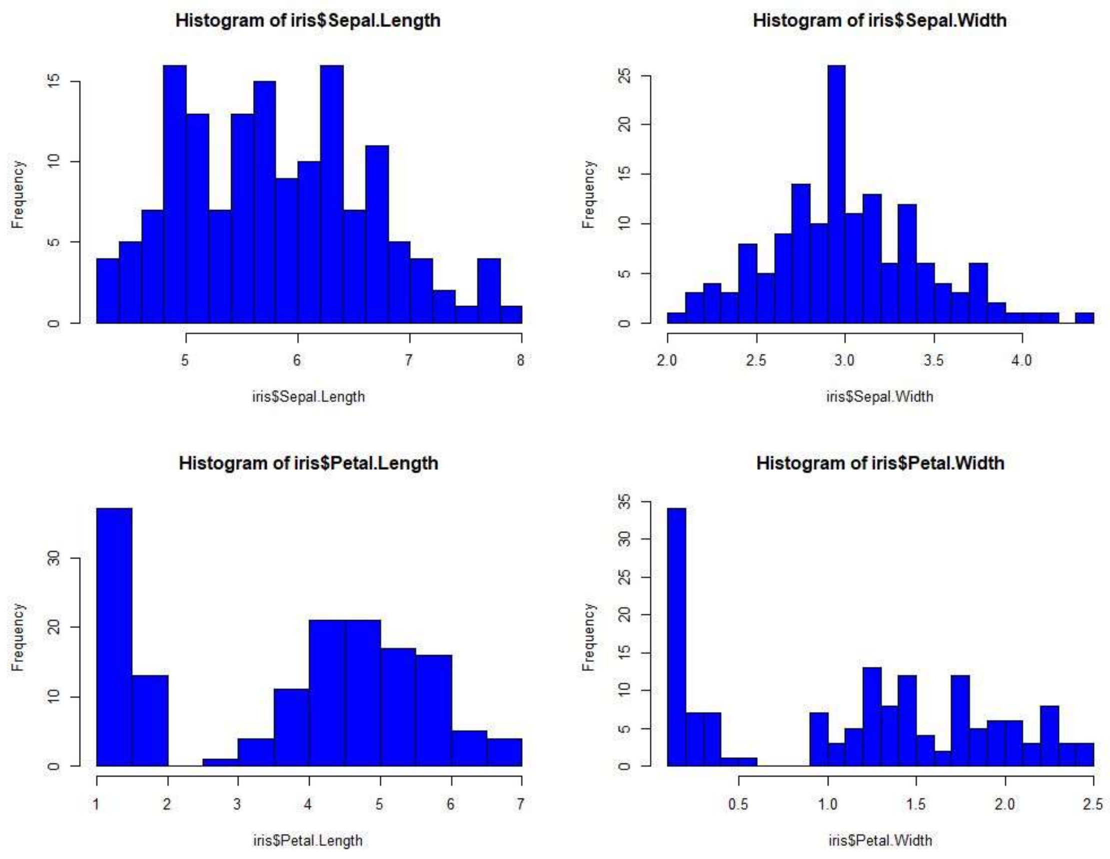
Virginica : correctly identified 9.

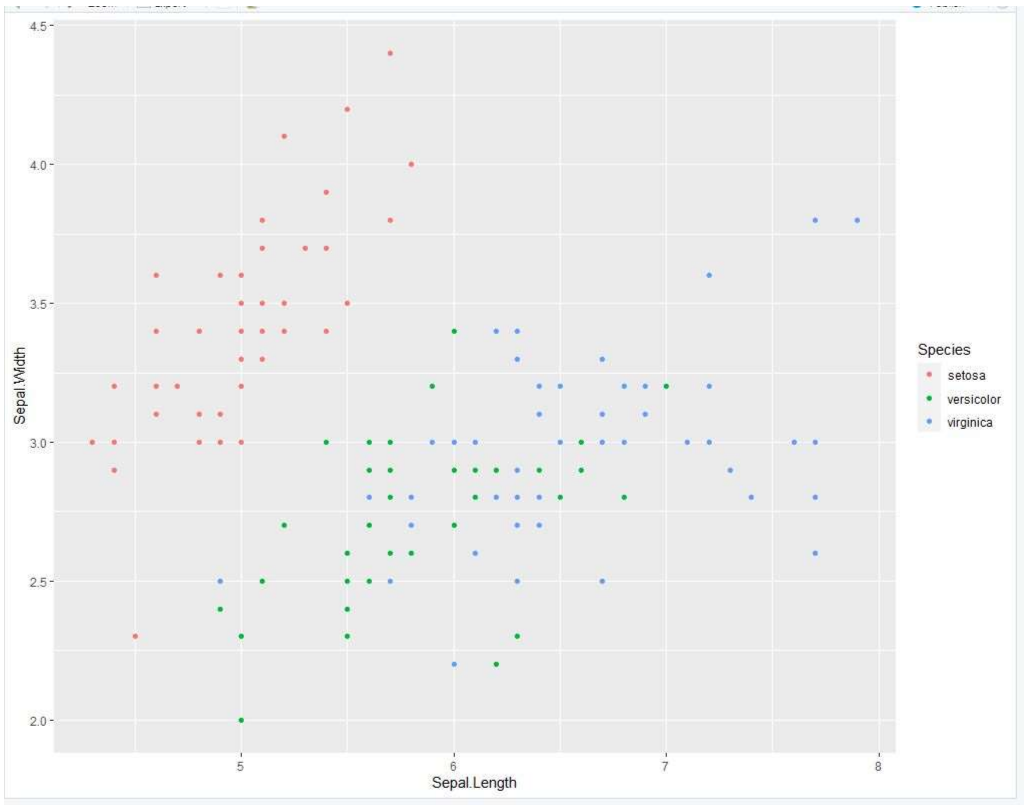
And also, the model achieved an acuuracy of 96%

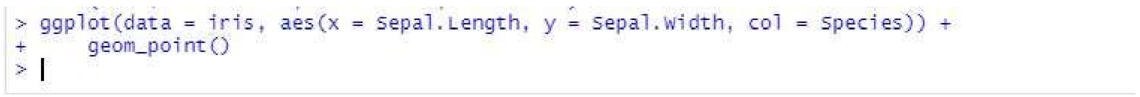
e. k-Nearest Neighbour (iris data)



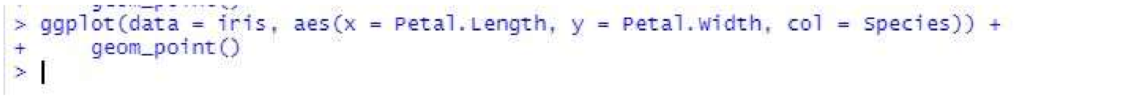
```
> library(class)
> library(ggplot2)
> library(GGally)
Registered S3 method overwritten by 'GGally':
  method from   
  +.gg   ggplot2
> 
> summary(iris)
  Sepal.Length    Sepal.Width   
 Min.   :4.300   Min.   :2.000  
 1st Qu.:5.100   1st Qu.:2.800  
 Median :5.800   Median :3.000  
 Mean   :5.843   Mean   :3.057  
 3rd Qu.:6.400   3rd Qu.:3.300  
 Max.   :7.900   Max.   :4.400  
  Petal.Length    Petal.Width   
 Min.   :1.000   Min.   :0.100  
 1st Qu.:1.600   1st Qu.:0.300  
 Median :4.350   Median :1.300  
 Mean   :3.758   Mean   :1.199  
 3rd Qu.:5.100   3rd Qu.:1.800  
 Max.   :6.900   Max.   :2.500  
       Species  
 setosa    :50  
 versicolor:50  
 virginica :50  



> apply(iris[,1:4], 2, sd)
Sepal.Length  Sepal.Width Petal.Length 
   0.8280661    0.4358663    1.7652982 
 Petal.Width 
   0.7622377 
> par(mfrow=c(2,2))
> hist(iris$Sepal.Length, col="blue", breaks=20)
> hist(iris$Sepal.Width, col="blue", breaks=20)
> hist(iris$Petal.Length, col="blue", breaks=20)
> hist(iris$Petal.Width, col="blue", breaks=20)
> |
```


Histogram of iris$Sepal.Length
Frequency
iris$Sepal.Length
Histogram of iris$Sepal.Width
Frequency
iris$Sepal.Width
Histogram of iris$Petal.Length
Frequency
iris$Petal.Length
Histogram of iris$Petal.Width
Frequency
iris$Petal.Width

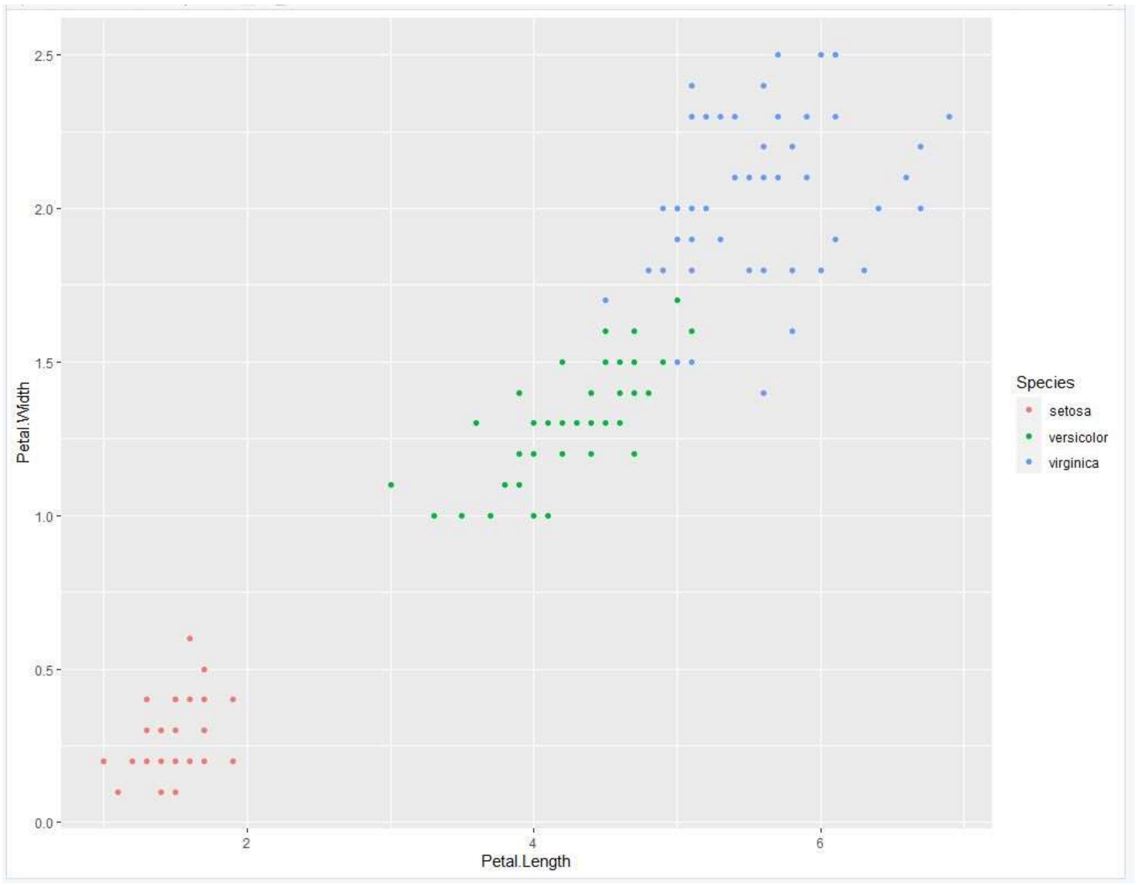


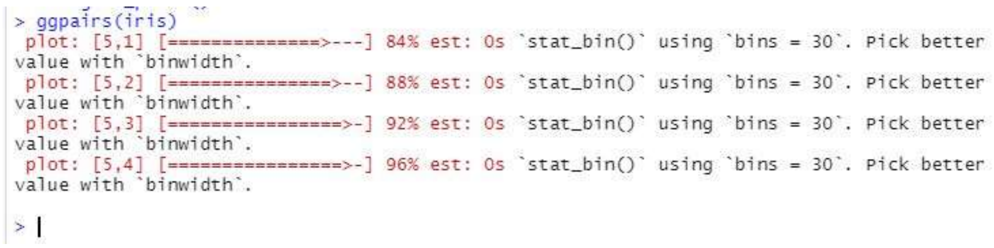


```
> ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width, col = Species)) +
+     geom_point()
>
```

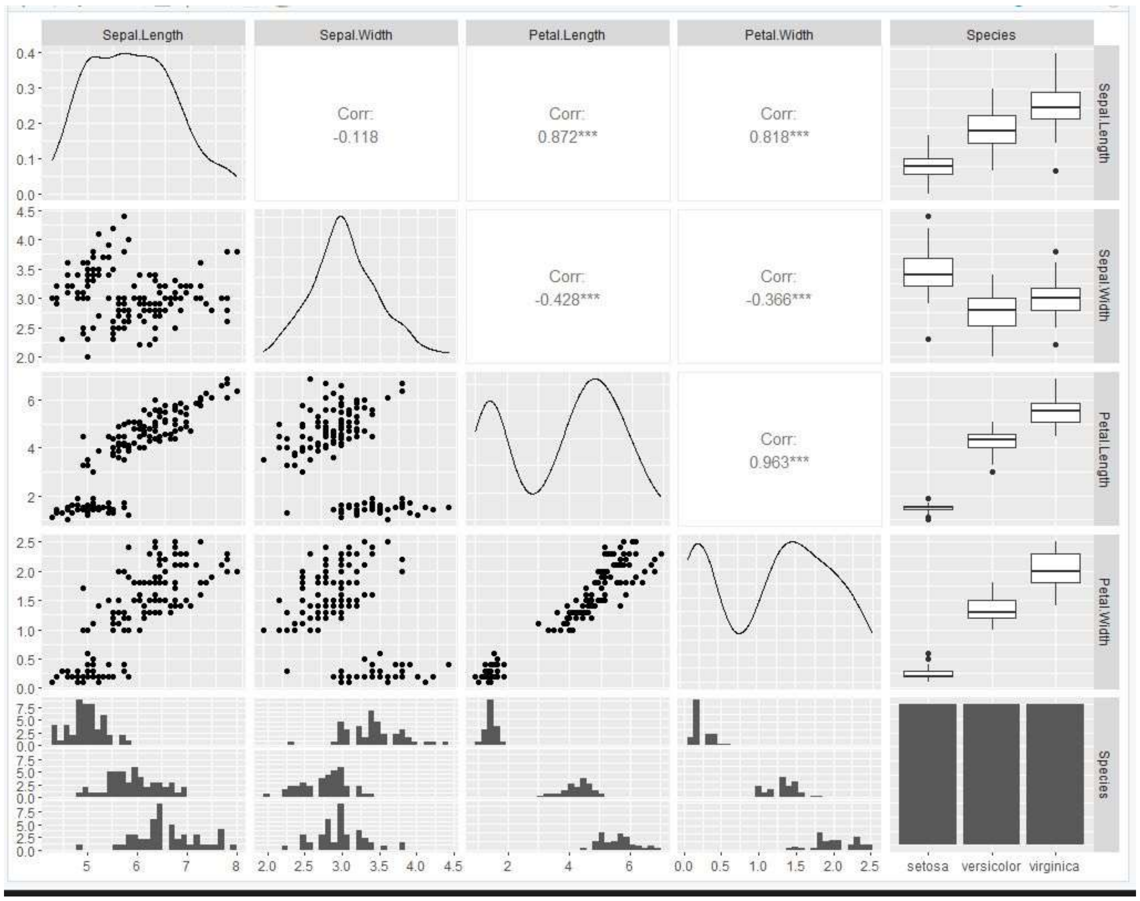


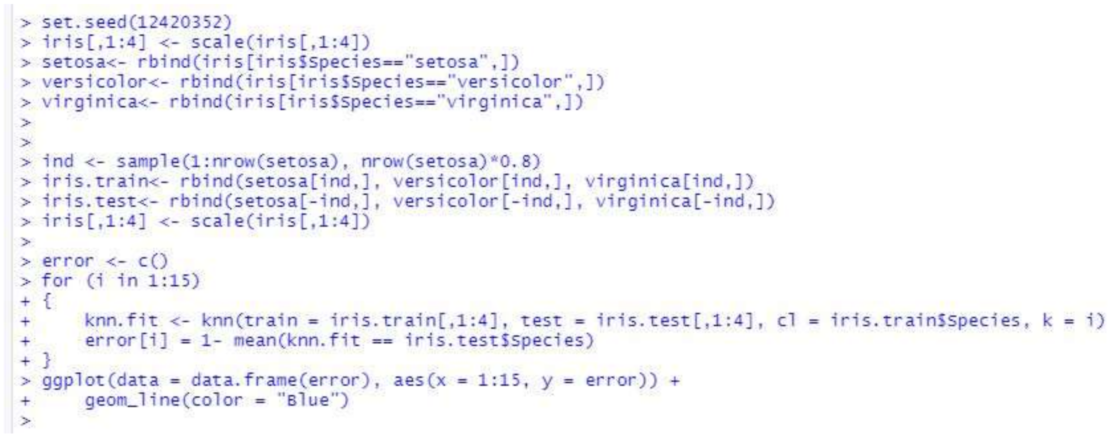
```
> ggplot(data = iris, aes(x = Petal.Length, y = Petal.Width, col = Species)) +
+     geom_point()
>
```



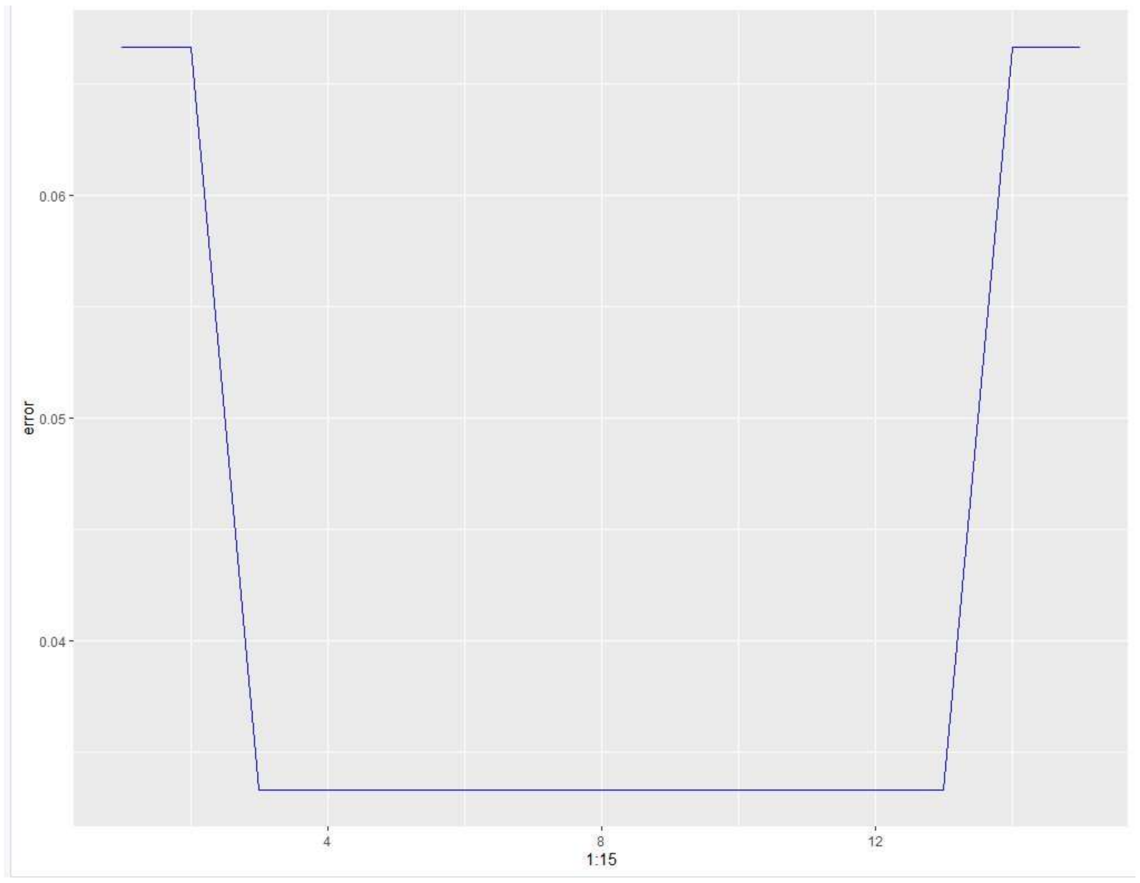


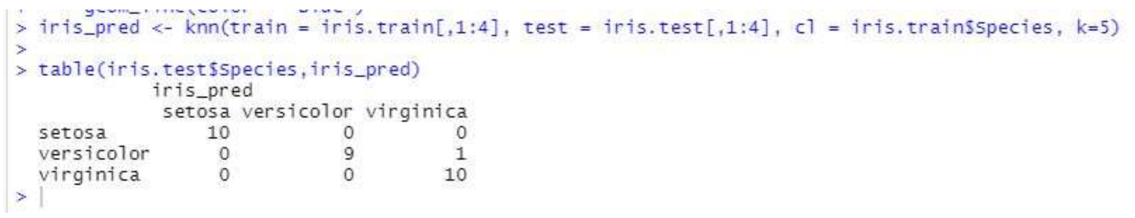
```
> ggpairs(iris)
 plot: [5,1] [==============>---] 84% est: 0s `stat_bin()` using `bins = 30`. Pick better
value with `binwidth`.
 plot: [5,2] [===============>--] 88% est: 0s `stat_bin()` using `bins = 30`. Pick better
value with `binwidth`.
 plot: [5,3] [================>-] 92% est: 0s `stat_bin()` using `bins = 30`. Pick better
value with `binwidth`.
 plot: [5,4] [================>-] 96% est: 0s `stat_bin()` using `bins = 30`. Pick better
value with `binwidth`.

> |
```





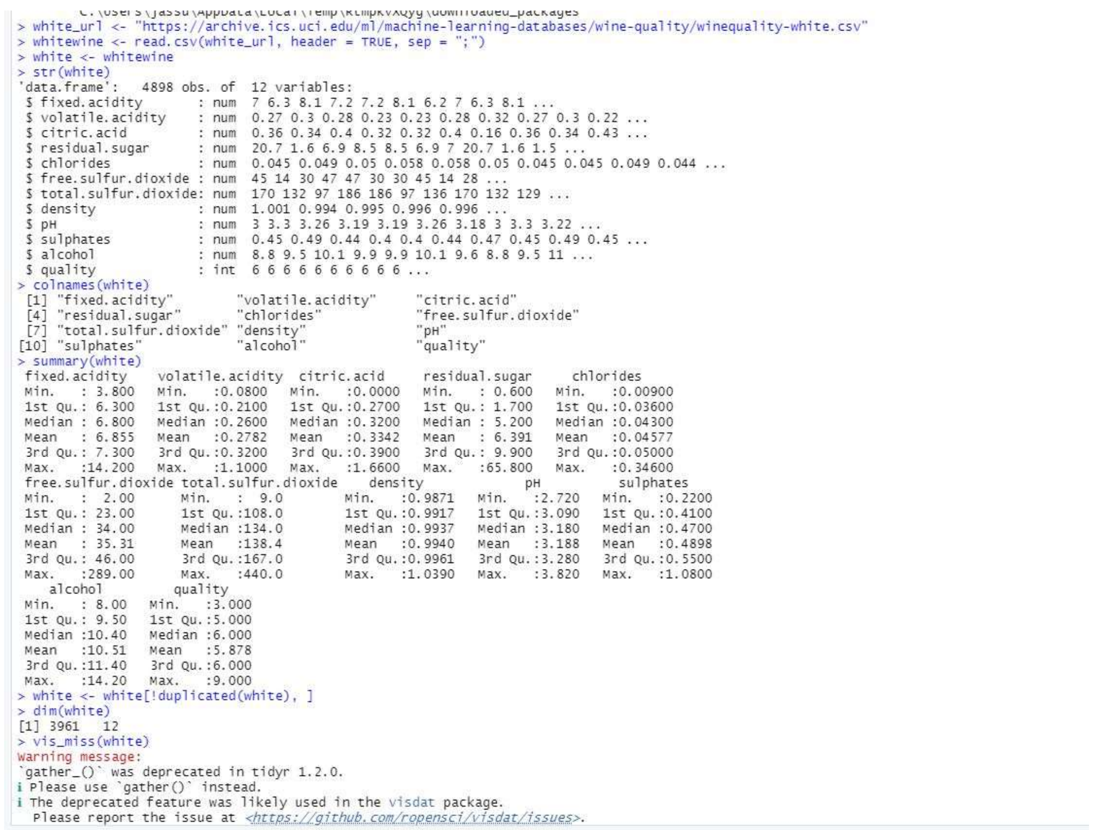
```
> set.seed(12420352)
> iris[,1:4] <- scale(iris[,1:4])
> setosa<- rbind(iris[iris$Species=="setosa",])
> versicolor<- rbind(iris[iris$Species=="versicolor",])
> virginica<- rbind(iris[iris$Species=="virginica",])
>
>
> ind <- sample(1:nrow(setosa), nrow(setosa)*0.8)
> iris.train<- rbind(setosa[ind,], versicolor[ind,], virginica[ind,])
> iris.test<- rbind(setosa[-ind,], versicolor[-ind,], virginica[-ind,])
> iris[,1:4] <- scale(iris[,1:4])
>
> error <- c()
> for (i in 1:15)
+ {
+     knn.fit <- knn(train = iris.train[,1:4], test = iris.test[,1:4], cl = iris.train$Species, k = i)
+     error[i] = 1- mean(knn.fit == iris.test$Species)
+ }
> ggplot(data = data.frame(error), aes(x = 1:15, y = error)) +
+     geom_line(color = "Blue")
>
```



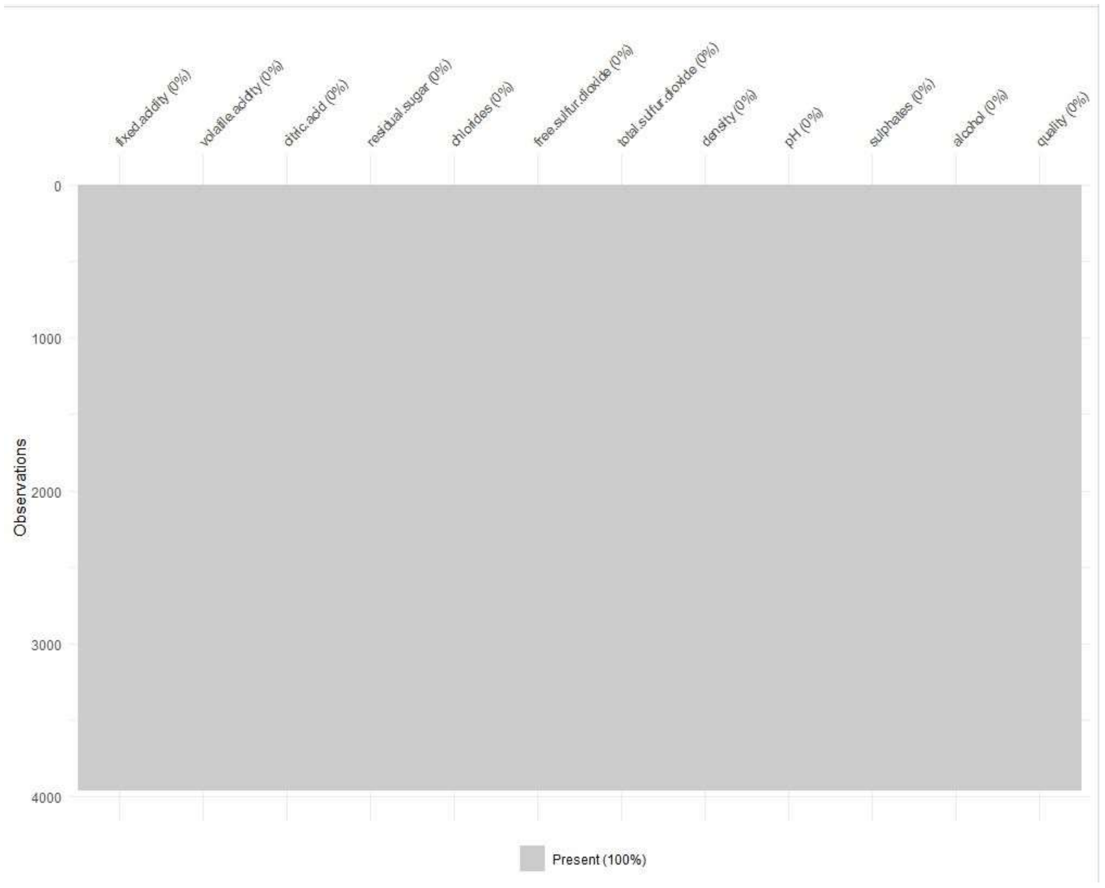


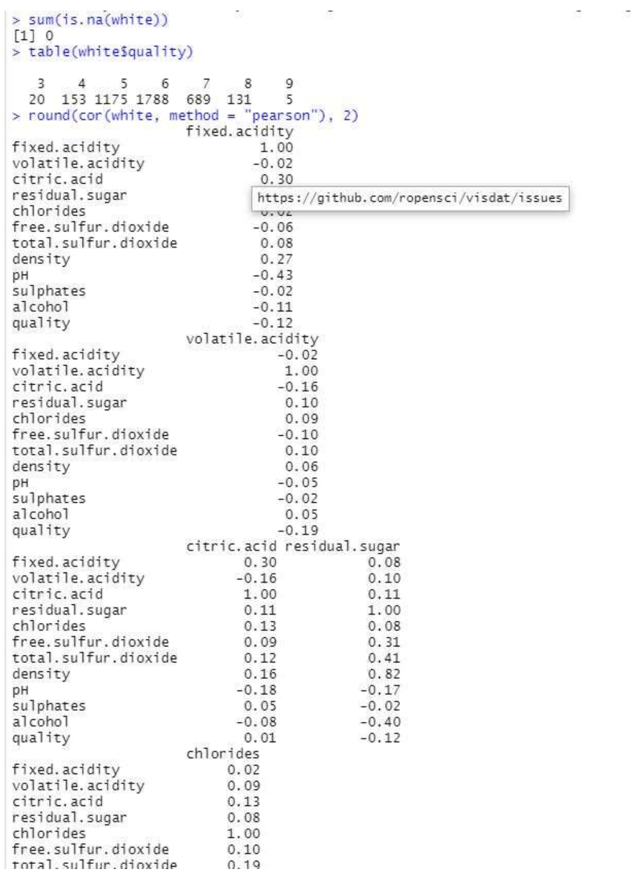
```
> iris_pred <- knn(train = iris.train[,1:4], test = iris.test[,1:4], cl = iris.train$Species, k=5)
>
> table(iris.test$Species,iris_pred)
            iris_pred
             setosa versicolor virginica
  setosa         10          0         0
  versicolor      0          9         1
  virginica       0          0        10
> |
```

3. Use win equality dataset for all classification practice and use quality as predictor variable
   a. Logistic regression



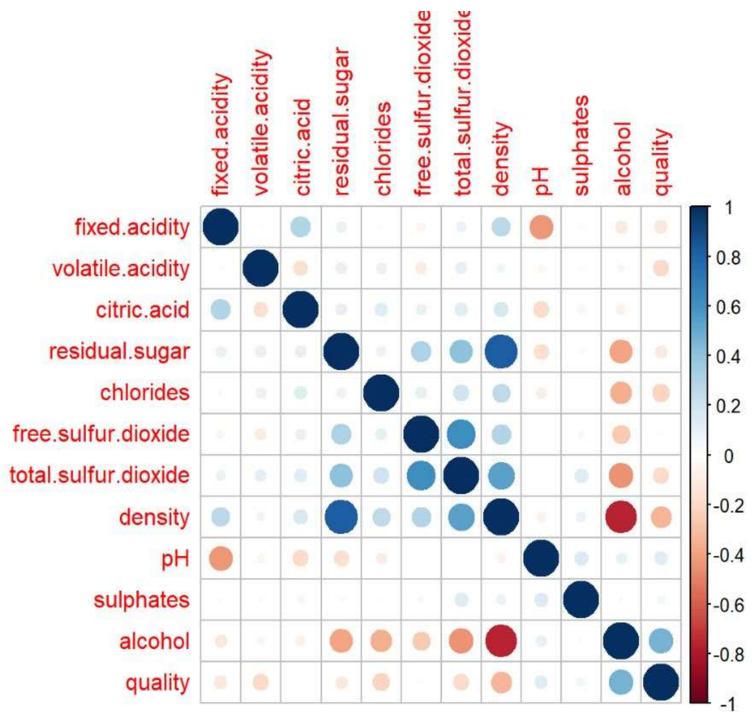
```
        C:\Users\jassu\AppData\Local\Temp\RtmpkvXQyg\downloaded_packages
> white_url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv"
> whitewine <- read.csv(white_url, header = TRUE, sep = ";")
> white <- whitewine
> str(white)
'data.frame':	4898 obs. of  12 variables:
 $ fixed.acidity       : num  7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
 $ volatile.acidity    : num  0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3 0.22 ...
 $ citric.acid         : num  0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34 0.43 ...
 $ residual.sugar      : num  20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
 $ chlorides           : num  0.045 0.049 0.05 0.058 0.058 0.05 0.045 0.045 0.049 0.044 ...
 $ free.sulfur.dioxide : num  45 14 30 47 47 30 30 45 14 28 ...
 $ total.sulfur.dioxide: num  170 132 97 186 186 97 136 170 132 129 ...
 $ density             : num  1.001 0.994 0.995 0.996 0.996 ...
 $ pH                  : num  3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22 ...
 $ sulphates           : num  0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45 ...
 $ alcohol             : num  8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
 $ quality             : int  6 6 6 6 6 6 6 6 6 6 ...
> colnames(white)
 [1] "fixed.acidity"        "volatile.acidity"     "citric.acid"         
 [4] "residual.sugar"       "chlorides"            "free.sulfur.dioxide" 
 [7] "total.sulfur.dioxide" "density"              "pH"                  
[10] "sulphates"            "alcohol"              "quality"             
> summary(white)
 fixed.acidity    volatile.acidity  citric.acid     residual.sugar     chlorides      
 Min.   : 3.800   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600   Min.   :0.00900  
 1st Qu.: 6.300   1st Qu.:0.2100   1st Qu.:0.2700   1st Qu.: 1.700   1st Qu.:0.03600  
 Median : 6.800   Median :0.2600   Median :0.3200   Median : 5.200   Median :0.04300  
 Mean   : 6.855   Mean   :0.2782   Mean   :0.3342   Mean   : 6.391   Mean   :0.04577  
 3rd Qu.: 7.300   3rd Qu.:0.3200   3rd Qu.:0.3900   3rd Qu.: 9.900   3rd Qu.:0.05000  
 Max.   :14.200   Max.   :1.1000   Max.   :1.6600   Max.   :65.800   Max.   :0.34600  
 free.sulfur.dioxide total.sulfur.dioxide    density             pH          sulphates     
 Min.   :  2.00      Min.   :  9.0        Min.   :0.9871   Min.   :2.720   Min.   :0.2200  
 1st Qu.: 23.00      1st Qu.:108.0        1st Qu.:0.9917   1st Qu.:3.090   1st Qu.:0.4100  
 Median : 34.00      Median :134.0        Median :0.9937   Median :3.180   Median :0.4700  
 Mean   : 35.31      Mean   :138.4        Mean   :0.9940   Mean   :3.188   Mean   :0.4898  
 3rd Qu.: 46.00      3rd Qu.:167.0        3rd Qu.:0.9961   3rd Qu.:3.280   3rd Qu.:0.5500  
 Max.   :289.00      Max.   :440.0        Max.   :1.0390   Max.   :3.820   Max.   :1.0800  
    alcohol         quality     
 Min.   : 8.00   Min.   :3.000  
 1st Qu.: 9.50   1st Qu.:5.000  
 Median :10.40   Median :6.000  
 Mean   :10.51   Mean   :5.878  
 3rd Qu.:11.40   3rd Qu.:6.000  
 Max.   :14.20   Max.   :9.000  
> white <- white[!duplicated(white), ]
> dim(white)
[1] 3961   12
> vis_miss(white)
Warning message:
`gather_()` was deprecated in tidyr 1.2.0.
i Please use `gather()` instead.
i The deprecated feature was likely used in the visdat package.
  Please report the issue at <https://github.com/ropensci/visdat/issues>.
```



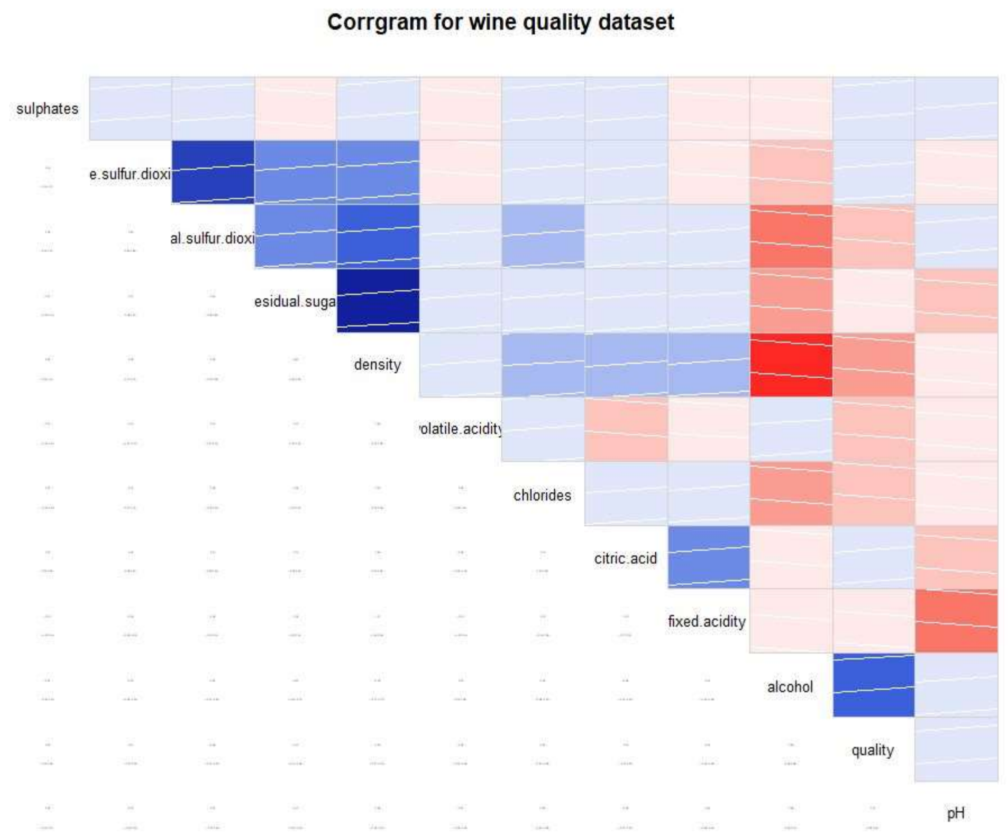


```
> sum(is.na(white))
[1] 0
> table(white$quality)

   3    4    5    6    7    8    9 
  20  153 1175 1788  689  131    5 
> round(cor(white, method = "pearson"), 2)
                     fixed.acidity
fixed.acidity                 1.00
volatile.acidity             -0.02
citric.acid                   0.30
residual.sugar         https://github.com/ropensci/visdat/issues
chlorides                     0.02
free.sulfur.dioxide          -0.06
total.sulfur.dioxide          0.08
density                       0.27
pH                           -0.43
sulphates                    -0.02
alcohol                      -0.11
quality                      -0.12
                     volatile.acidity
fixed.acidity                   -0.02
volatile.acidity                 1.00
citric.acid                     -0.16
residual.sugar                   0.10
chlorides                        0.09
free.sulfur.dioxide             -0.10
total.sulfur.dioxide             0.10
density                          0.06
pH                              -0.05
sulphates                       -0.02
alcohol                          0.05
quality                         -0.19
                     citric.acid residual.sugar
fixed.acidity               0.30           0.08
volatile.acidity           -0.16           0.10
citric.acid                 1.00           0.11
residual.sugar              0.11           1.00
chlorides                   0.13           0.08
free.sulfur.dioxide         0.09           0.31
total.sulfur.dioxide        0.12           0.41
density                     0.16           0.82
pH                         -0.18          -0.17
sulphates                   0.05          -0.02
alcohol                    -0.08          -0.40
quality                     0.01          -0.12
                     chlorides
fixed.acidity             0.02
volatile.acidity          0.09
citric.acid               0.13
residual.sugar            0.08
chlorides                 1.00
free.sulfur.dioxide       0.10
total.sulfur.dioxide      0.19
```

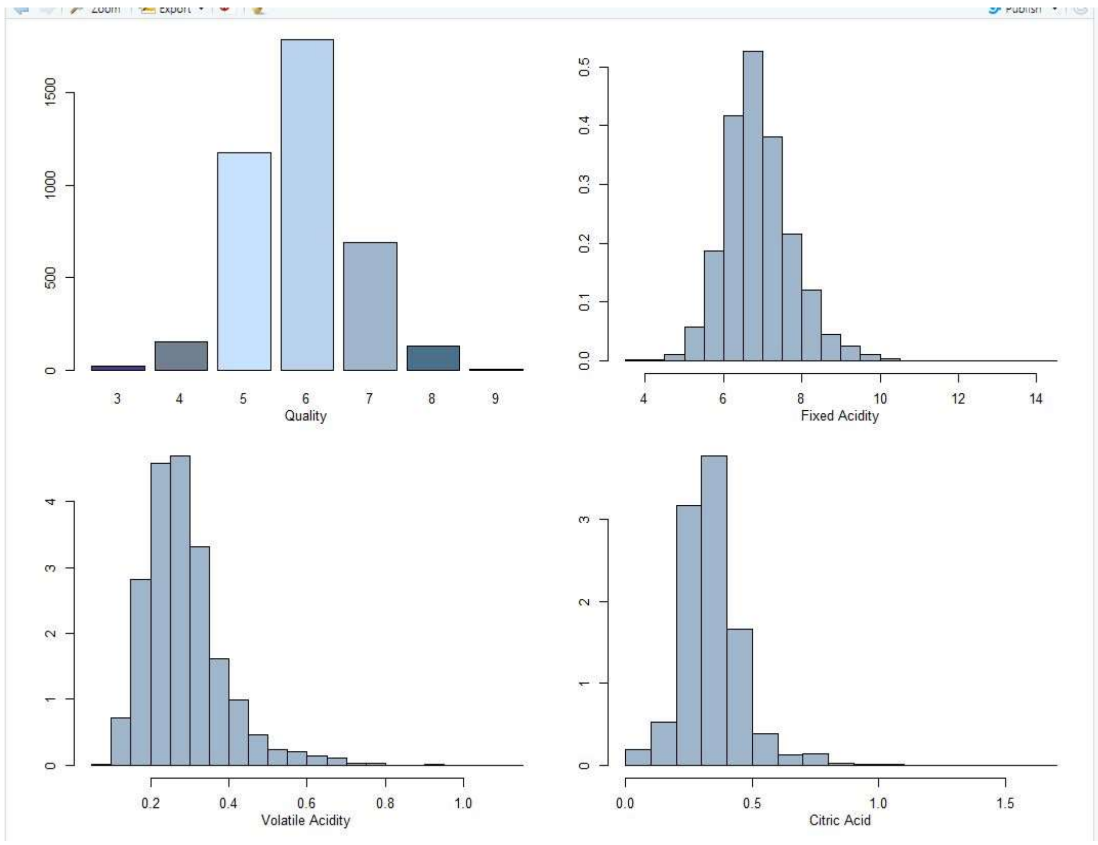
```
                    free.sulfur.dioxide
fixed.acidity                     -0.06
volatile.acidity                  -0.10
citric.acid                        0.09
residual.sugar                     0.31
chlorides                          0.10
free.sulfur.dioxide                1.00
total.sulfur.dioxide               0.62
density                            0.29
pH                                -0.01
sulphates                          0.04
alcohol                           -0.25
quality                            0.01
                    total.sulfur.dioxide
fixed.acidity                       0.08
volatile.acidity                    0.10
citric.acid                         0.12
residual.sugar                      0.41
chlorides                           0.19
free.sulfur.dioxide                 0.62
total.sulfur.dioxide                1.00
density                             0.54
pH                                  0.01
sulphates                           0.14
alcohol                            -0.45
quality                            -0.18
                    density    pH sulphates
fixed.acidity          0.27 -0.43     -0.02
volatile.acidity       0.06 -0.05     -0.02
citric.acid            0.16 -0.18      0.05
residual.sugar         0.82 -0.17     -0.02
chlorides              0.25 -0.09      0.02
free.sulfur.dioxide    0.29 -0.01      0.04
total.sulfur.dioxide   0.54  0.01      0.14
density                1.00 -0.06      0.08
pH                    -0.06  1.00      0.14
sulphates              0.08  0.14      1.00
alcohol               -0.76  0.09     -0.02
quality               -0.34  0.12      0.05
                    alcohol quality
fixed.acidity         -0.11   -0.12
volatile.acidity       0.05   -0.19
citric.acid           -0.08    0.01
residual.sugar        -0.40   -0.12
chlorides             -0.36   -0.22
free.sulfur.dioxide   -0.25    0.01
total.sulfur.dioxide  -0.45   -0.18
density               -0.76   -0.34
pH                     0.09    0.12
sulphates             -0.02    0.05
alcohol                1.00    0.46
quality                0.46    1.00
> corrplot(cor(white))
```



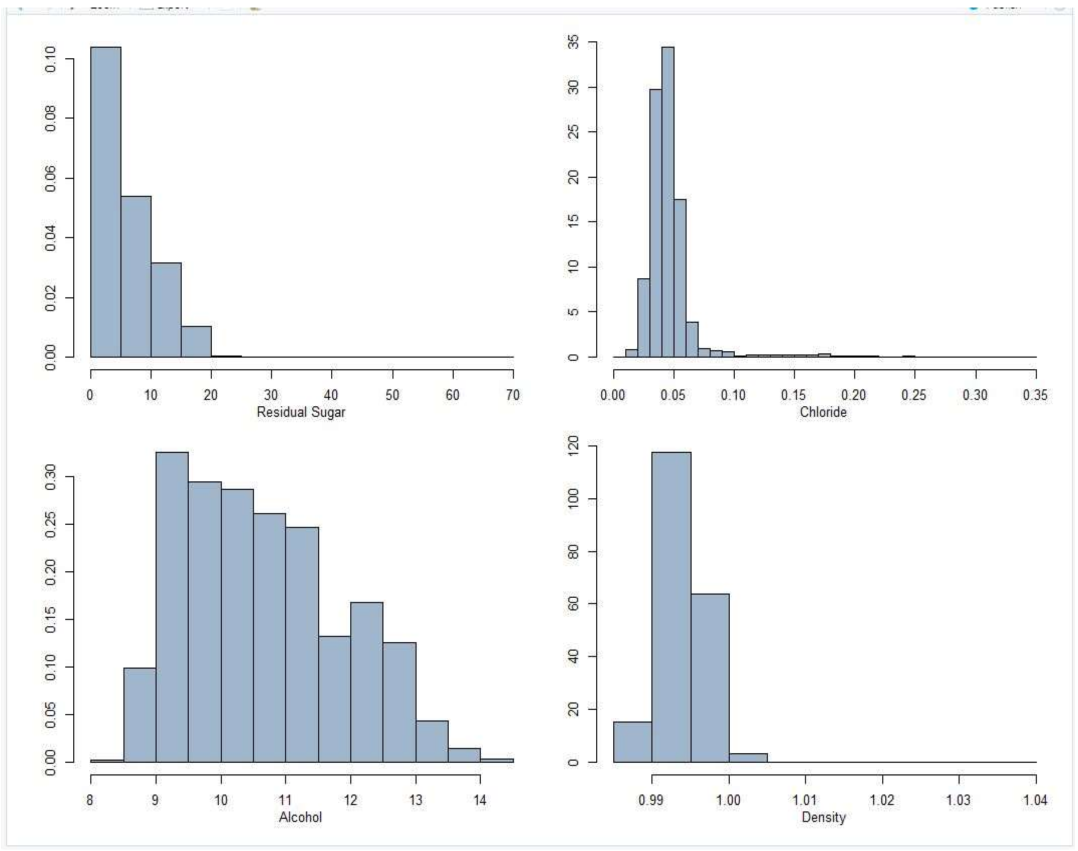
```
> corrgram(white, type="data", lower.panel=panel.conf,
+          upper.panel=panel.shade, main= "Corrgram for wine quality dataset", order=T, cex.labels=1.2)
There were 50 or more warnings (use warnings() to see the first 50)
>
```



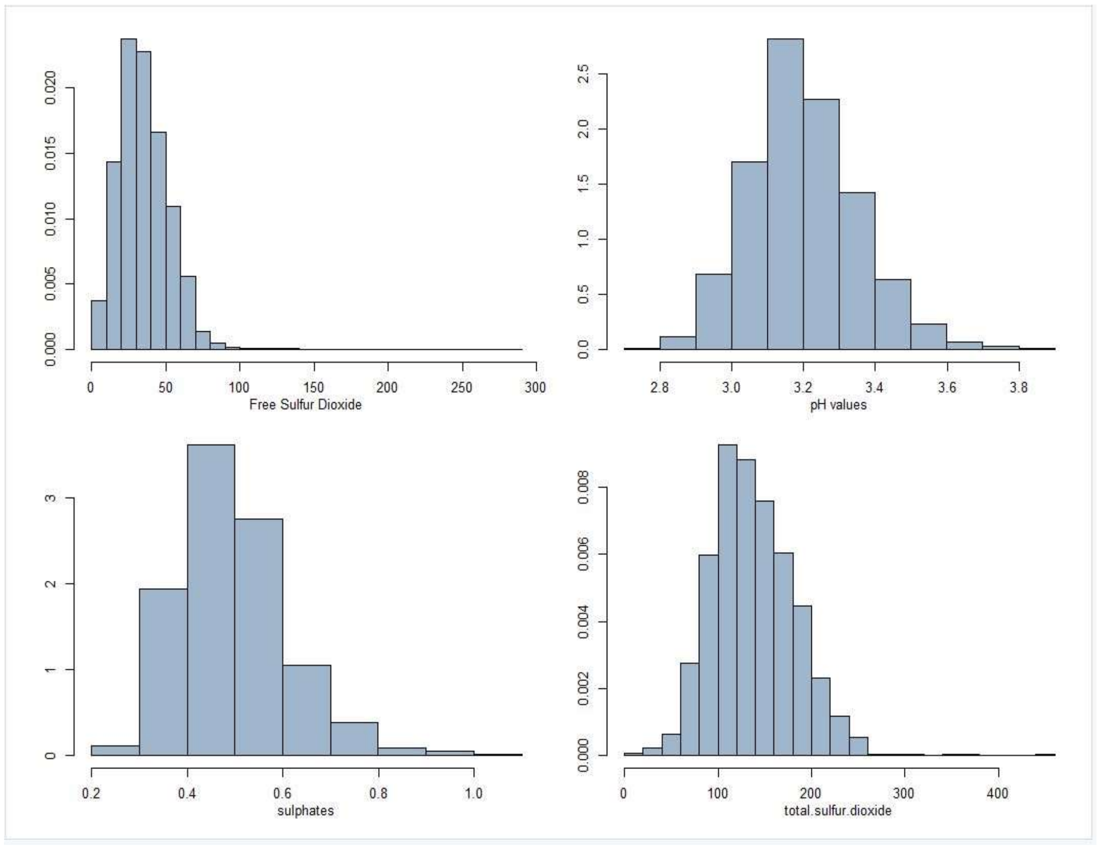
```
> attach(white)
>
> par(mfrow=c(2,2), oma = c(1,1,0,0) + 0.1, mar = c(3,3,1,1) + 0.1)
> barplot((table(quality)), col=c("slateblue4", "slategray", "slategray1", "slategray2", "slategray3", "skyblue4"))
> mtext("Quality", side=1, outer=F, line=2, cex=0.8)
>
>
> truehist(fixed.acidity, h = 0.5, col="slategray3")
> mtext("Fixed Acidity", side=1, outer=F, line=2, cex=0.8)
>
> truehist(volatile.acidity, h = 0.05, col="slategray3")
> mtext("Volatile Acidity", side=1, outer=F, line=2, cex=0.8)
>
> truehist(citric.acid, h = 0.1, col="slategray3")
> mtext("Citric Acid", side=1, outer=F, line=2, cex=0.8)
>
```



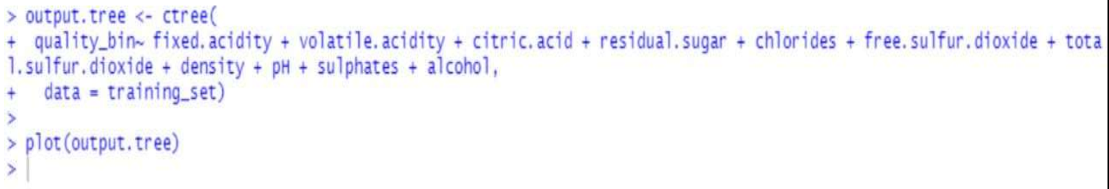
```
> par(mfrow=c(2,2), oma = c(1,1,0,0) + 0.1, mar = c(3,3,1,1) + 0.1)
>
>
>
>
> truehist(residual.sugar, h = 5, col="slategray3")
> mtext("Residual Sugar", side=1, outer=F, line=2, cex=0.8)
>
> truehist(chlorides, h = 0.01, col="slategray3")
> mtext("Chloride", side=1, outer=F, line=2, cex=0.8)
>
> truehist(alcohol, h = 0.5, col="slategray3")
> mtext("Alcohol", side=1, outer=F, line=2, cex=0.8)
>
>
> truehist(density, h = 0.005, col="slategray3")
> mtext("Density", side=1, outer=F, line=2, cex=0.8)
>
```



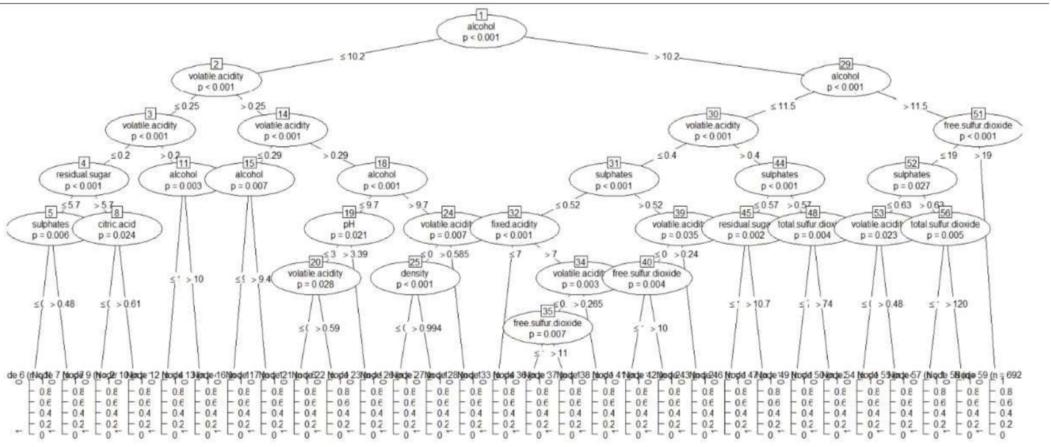
```
>
>
> par(mfrow=c(2,2), oma = c(1,1,0,0) + 0.1, mar = c(3,3,1,1) + 0.1)
>
> truehist(free.sulfur.dioxide, h = 10, col="slategray3")
> mtext("Free Sulfur Dioxide", side=1, outer=F, line=2, cex=0.8)
>
> truehist(pH, h = 0.1, col="slategray3")
> mtext("pH values", side=1, outer=F, line=2, cex=0.8)
>
> truehist(sulphates, h = 0.1, col="slategray3")
> mtext("sulphates", side=1, outer=F, line=2, cex=0.8)
>
>
> truehist(total.sulfur.dioxide, h = 20, col="slategray3")
> mtext("total.sulfur.dioxide", side=1, outer=F, line=2, cex=0.8)
> 
```



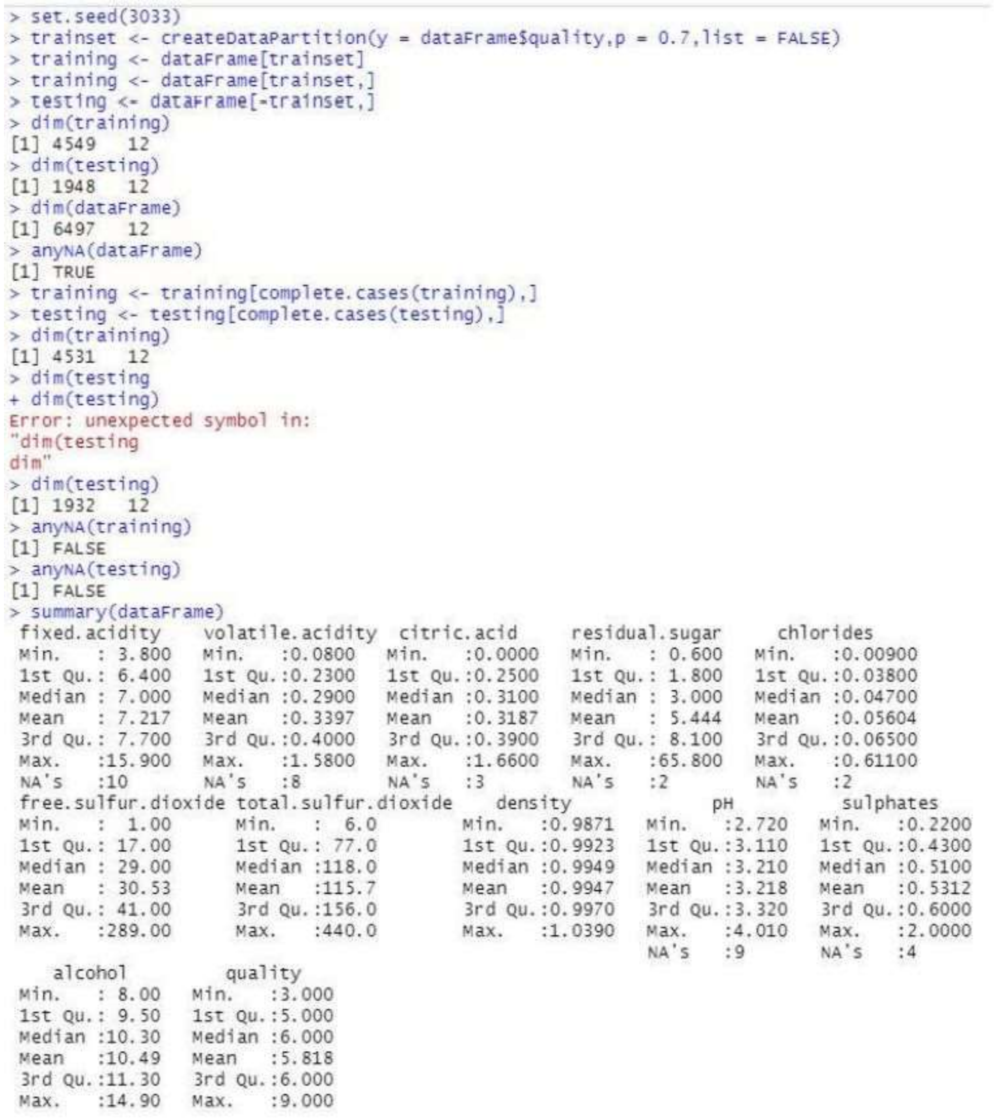
b. Decision trees



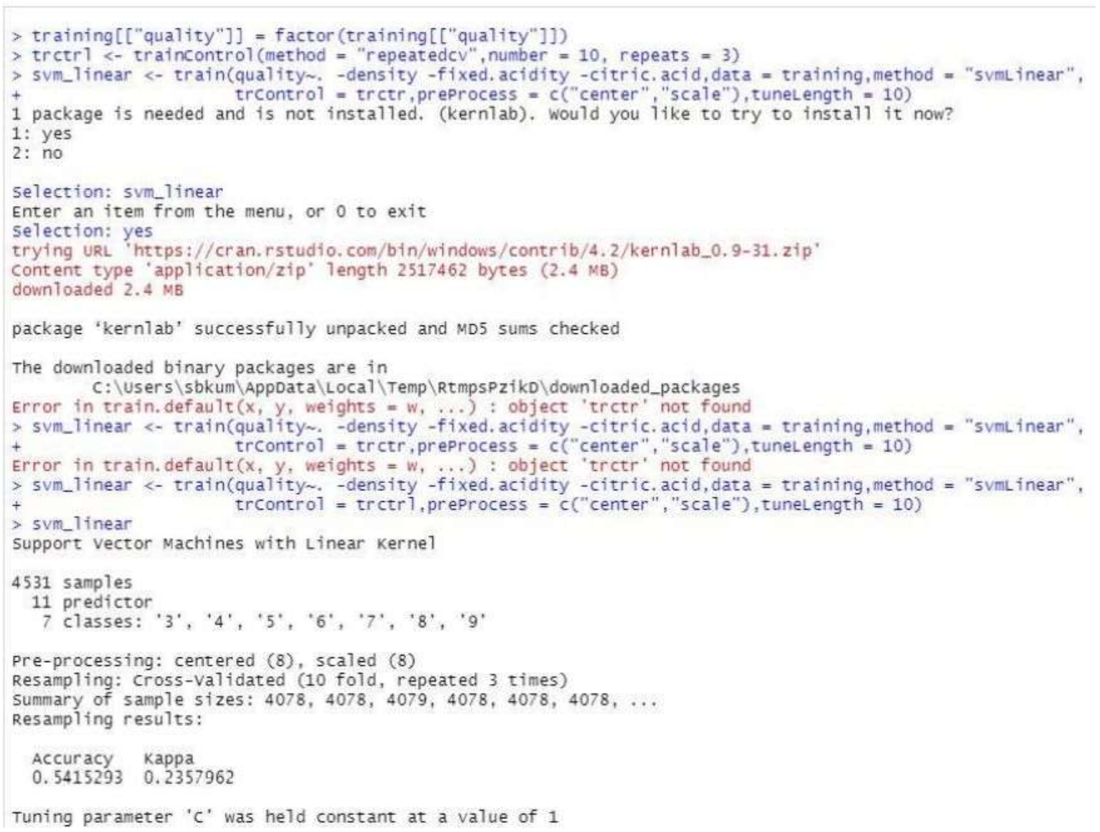
```
> output.tree <- ctree(
+  quality_bin~ fixed.acidity + volatile.acidity + citric.acid + residual.sugar + chlorides + free.sulfur.dioxide + tota
l.sulfur.dioxide + density + pH + sulphates + alcohol,
+   data = training_set)
>
> plot(output.tree)
>
```


1 alcohol p < 0.001
≤ 10.2
> 10.2
2 volatile.acidity p < 0.001
29 alcohol p < 0.001
3 volatile.acidity p < 0.001
14 volatile.acidity p < 0.001
30 volatile.acidity p < 0.001
51 free.sulfur.dioxide p < 0.001
4 residual.sugar p < 0.001
11 alcohol p = 0.003
15 alcohol p = 0.007
18 alcohol p < 0.001
31 sulphates p < 0.001
44 sulphates p < 0.001
52 sulphates p = 0.027
5 sulphates p = 0.006
8 citric.acid p = 0.024
19 pH p = 0.021
24 volatile.acidity p = 0.007
32 fixed.acidity p < 0.001
39 volatile.acidity p = 0.035
45 residual.sugar p = 0.002
48 total.sulfur.dioxide p = 0.004
53 volatile.acidity p = 0.023
56 total.sulfur.dioxide p = 0.005
20 volatile.acidity p = 0.028
25 density p < 0.001
34 volatile.acidity p = 0.003
40 free.sulfur.dioxide p = 0.004
35 free.sulfur.dioxide p = 0.007

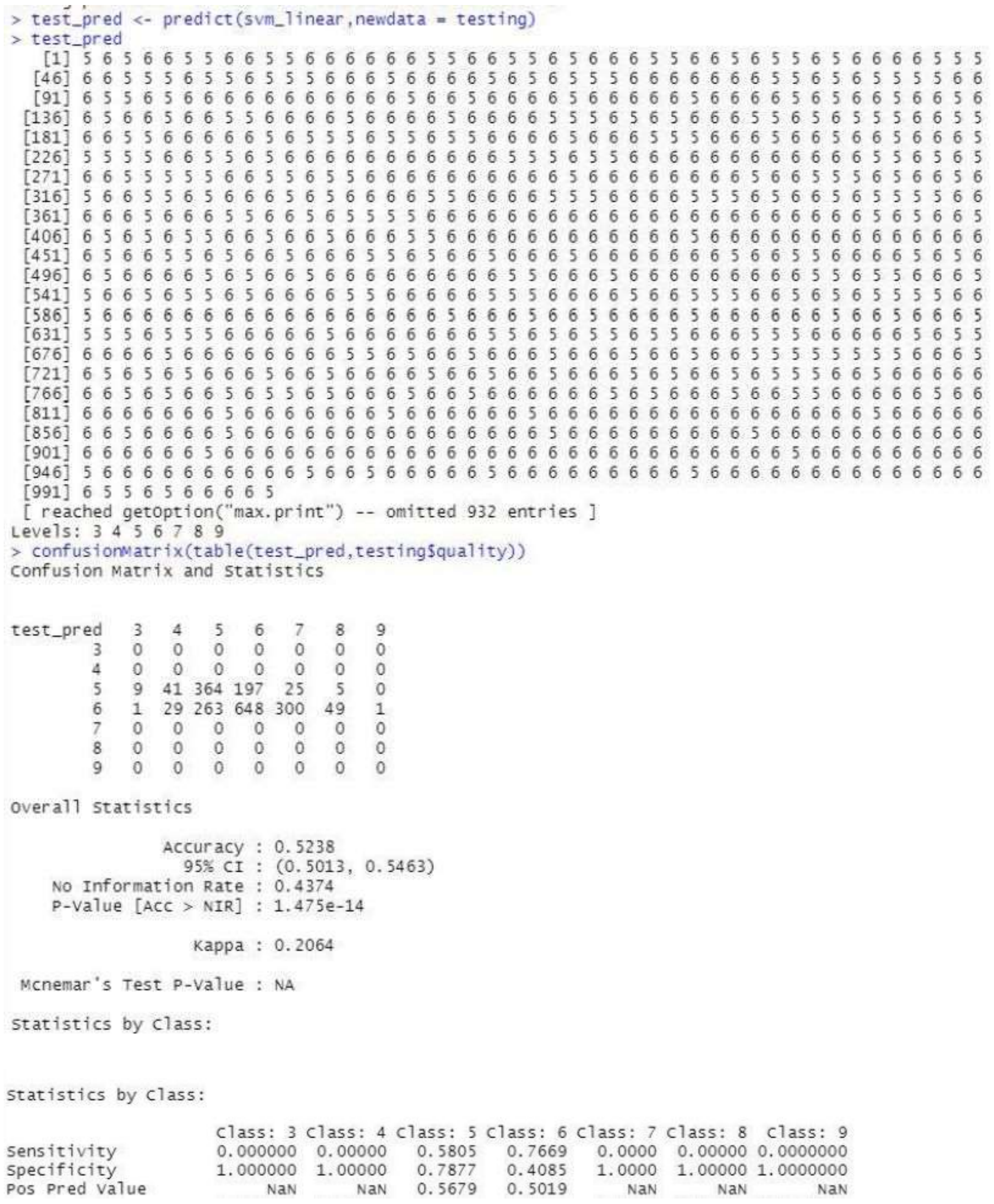
c. Support Vector Machines



```
> set.seed(3033)
> trainset <- createDataPartition(y = dataFrame$quality,p = 0.7,list = FALSE)
> training <- dataFrame[trainset]
> training <- dataFrame[trainset,]
> testing <- dataFrame[-trainset,]
> dim(training)
[1] 4549   12
> dim(testing)
[1] 1948   12
> dim(dataFrame)
[1] 6497   12
> anyNA(dataFrame)
[1] TRUE
> training <- training[complete.cases(training),]
> testing <- testing[complete.cases(testing),]
> dim(training)
[1] 4531   12
> dim(testing
+ dim(testing)
Error: unexpected symbol in:
"dim(testing
dim"
> dim(testing)
[1] 1932   12
> anyNA(training)
[1] FALSE
> anyNA(testing)
[1] FALSE
> summary(dataFrame)
 fixed.acidity    volatile.acidity  citric.acid     residual.sugar     chlorides     
 Min.   : 3.800   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600   Min.   :0.00900  
 1st Qu.: 6.400   1st Qu.:0.2300   1st Qu.:0.2500   1st Qu.: 1.800   1st Qu.:0.03800  
 Median : 7.000   Median :0.2900   Median :0.3100   Median : 3.000   Median :0.04700  
 Mean   : 7.217   Mean   :0.3397   Mean   :0.3187   Mean   : 5.444   Mean   :0.05604  
 3rd Qu.: 7.700   3rd Qu.:0.4000   3rd Qu.:0.3900   3rd Qu.: 8.100   3rd Qu.:0.06500  
 Max.   :15.900   Max.   :1.5800   Max.   :1.6600   Max.   :65.800   Max.   :0.61100  
 NA's   :10       NA's   :8        NA's   :3        NA's   :2        NA's   :2        
 free.sulfur.dioxide total.sulfur.dioxide    density             pH          sulphates     
 Min.   :  1.00      Min.   :  6.0        Min.   :0.9871   Min.   :2.720   Min.   :0.2200  
 1st Qu.: 17.00      1st Qu.: 77.0        1st Qu.:0.9923   1st Qu.:3.110   1st Qu.:0.4300  
 Median : 29.00      Median :118.0        Median :0.9949   Median :3.210   Median :0.5100  
 Mean   : 30.53      Mean   :115.7        Mean   :0.9947   Mean   :3.218   Mean   :0.5312  
 3rd Qu.: 41.00      3rd Qu.:156.0        3rd Qu.:0.9970   3rd Qu.:3.320   3rd Qu.:0.6000  
 Max.   :289.00      Max.   :440.0        Max.   :1.0390   Max.   :4.010   Max.   :2.0000  
                                                           NA's   :9       NA's   :4       
    alcohol         quality     
 Min.   : 8.00   Min.   :3.000  
 1st Qu.: 9.50   1st Qu.:5.000  
 Median :10.30   Median :6.000  
 Mean   :10.49   Mean   :5.818  
 3rd Qu.:11.30   3rd Qu.:6.000  
 Max.   :14.90   Max.   :9.000  
```

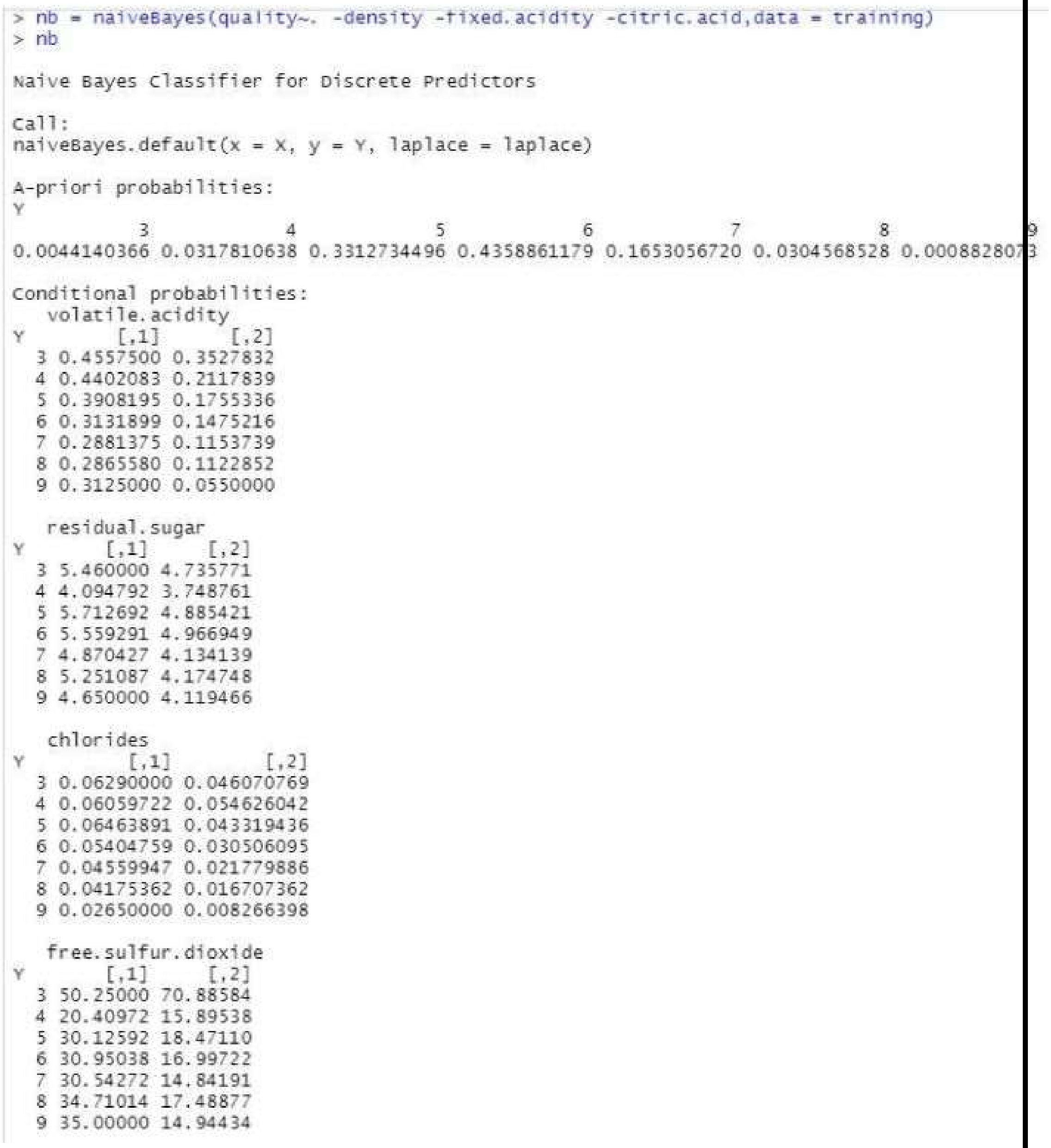


```
> training[["quality"]] = factor(training[["quality"]])
> trctrl <- trainControl(method = "repeatedcv",number = 10, repeats = 3)
> svm_linear <- train(quality~. -density -fixed.acidity -citric.acid,data = training,method = "svmLinear",
+                     trControl = trctr,preProcess = c("center","scale"),tuneLength = 10)
1 package is needed and is not installed. (kernlab). Would you like to try to install it now?
1: yes
2: no

Selection: svm_linear
Enter an item from the menu, or 0 to exit
Selection: yes
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.2/kernlab_0.9-31.zip'
Content type 'application/zip' length 2517462 bytes (2.4 MB)
downloaded 2.4 MB

package ‘kernlab’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\sbkum\AppData\Local\Temp\RtmpsPzikD\downloaded_packages
Error in train.default(x, y, weights = w, ...) : object 'trctr' not found
> svm_linear <- train(quality~. -density -fixed.acidity -citric.acid,data = training,method = "svmLinear",
+                     trControl = trctr,preProcess = c("center","scale"),tuneLength = 10)
Error in train.default(x, y, weights = w, ...) : object 'trctr' not found
> svm_linear <- train(quality~. -density -fixed.acidity -citric.acid,data = training,method = "svmLinear",
+                     trControl = trctrl,preProcess = c("center","scale"),tuneLength = 10)
> svm_linear
Support Vector Machines with Linear Kernel

4531 samples
  11 predictor
   7 classes: '3', '4', '5', '6', '7', '8', '9'

Pre-processing: centered (8), scaled (8)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 4078, 4078, 4079, 4078, 4078, 4078, ...
Resampling results:

  Accuracy   Kappa
  0.5415293  0.2357962

Tuning parameter 'C' was held constant at a value of 1
```

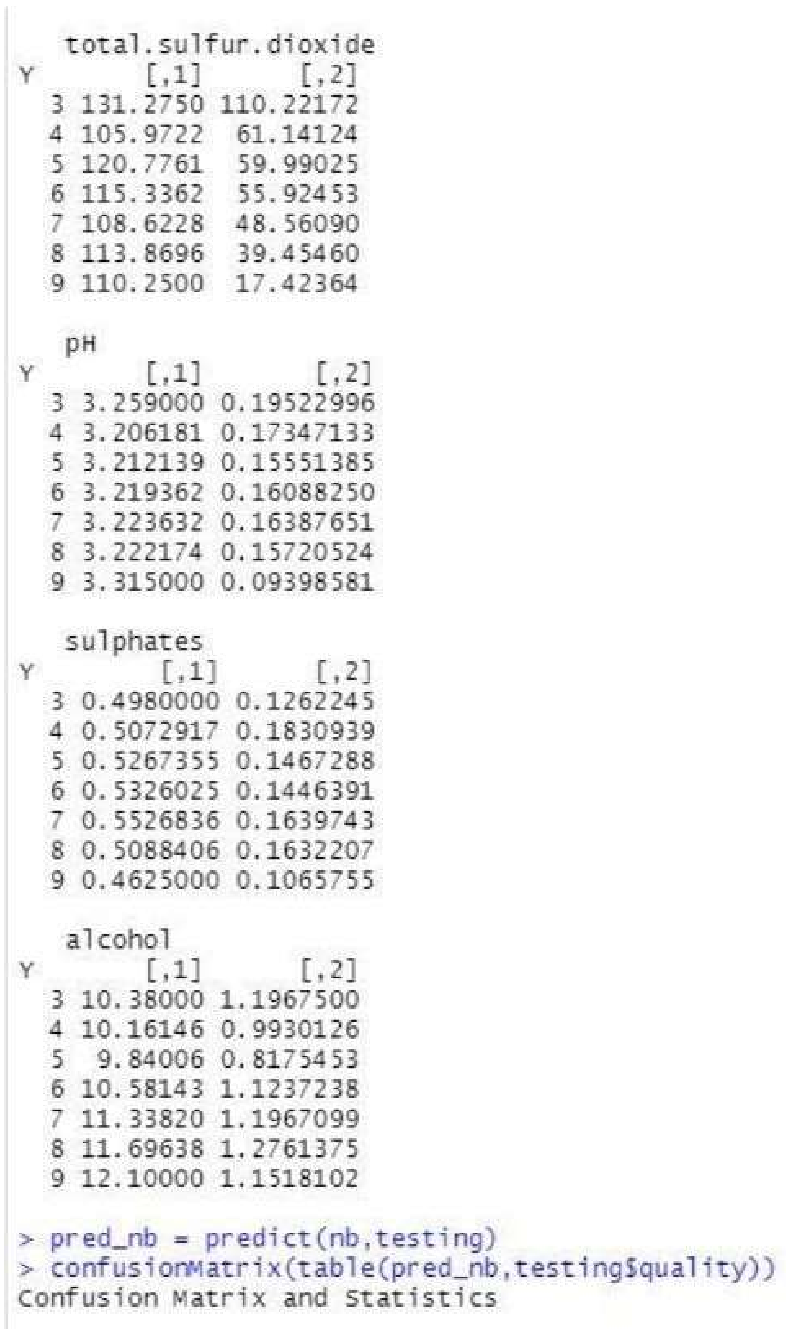


```
> test_pred <- predict(svm_linear,newdata = testing)
> test_pred
   [1] 5 6 5 6 6 5 5 6 6 5 5 6 6 6 6 6 6 5 5 6 6 5 5 6 5 6 6 6 5 5 6 6 5 6 5 5 6 5 6 6 6 6 5 5 5
  [46] 6 6 5 5 5 6 5 5 6 5 5 5 6 6 6 5 6 6 6 6 5 6 5 6 5 5 5 6 6 6 6 6 6 6 5 5 6 5 6 5 5 5 5 6 6
  [91] 6 5 5 6 5 6 6 6 6 6 6 6 6 6 6 5 6 6 5 6 6 6 6 5 6 6 6 6 6 5 6 6 6 6 5 6 5 6 6 5 6 6 5 6 6 5 6
 [136] 6 5 6 6 5 6 6 5 5 6 6 6 6 5 6 6 6 6 5 6 6 6 6 5 5 5 6 5 6 5 6 6 6 5 5 6 5 6 5 5 5 6 6 5 5
 [181] 6 6 5 5 6 6 6 6 6 5 6 5 5 5 6 5 5 6 5 5 6 6 6 6 5 6 6 6 5 5 5 6 6 6 5 6 6 5 6 6 5 6 6 6 5
 [226] 5 5 5 5 6 6 5 5 6 5 6 6 6 6 6 6 6 6 6 6 5 5 5 6 5 5 6 6 6 6 6 6 6 6 6 6 6 5 5 6 5 6 5
 [271] 6 6 5 5 5 5 5 6 6 5 5 6 5 5 6 6 6 6 6 6 6 6 6 5 6 6 6 6 6 6 6 5 6 6 5 5 5 6 5 6 6 5 6
 [316] 5 6 6 5 5 6 5 6 6 6 5 6 5 6 6 6 6 5 5 6 6 6 6 5 5 5 6 6 6 6 5 5 5 6 5 6 6 5 6 5 5 5 5 6 6
 [361] 6 6 6 5 6 6 6 5 5 6 6 5 6 5 5 5 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 5 6 5 6 6 5
 [406] 6 5 6 5 6 5 5 6 6 5 6 6 5 6 6 6 5 5 6 6 6 6 6 6 6 6 6 6 6 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
 [451] 6 5 6 6 5 5 6 5 6 6 5 6 6 6 5 5 6 5 6 6 5 6 6 6 5 6 6 6 6 6 6 6 5 6 6 5 5 6 6 6 6 5 6 5 6
 [496] 6 5 6 6 6 6 5 6 5 6 6 5 6 6 6 6 6 6 6 6 5 5 6 6 6 5 6 6 6 6 6 6 6 6 5 5 6 5 5 6 6 6 5
 [541] 5 6 6 5 6 5 5 6 5 6 6 6 6 5 5 6 6 6 6 6 5 5 5 6 6 6 6 5 6 6 5 5 5 6 6 5 6 5 6 5 5 5 5 6 6
 [586] 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 5 6 6 6 5 6 6 5 6 6 6 6 5 6 6 6 6 6 6 5 6 6 5 6 6 6 5
 [631] 5 5 5 6 5 5 5 6 6 6 6 6 5 6 6 6 6 6 6 5 5 6 5 6 5 5 6 5 5 6 6 6 5 5 5 6 6 6 6 6 5 6 5 5
 [676] 6 6 6 6 5 6 6 6 6 6 6 6 6 5 5 6 5 6 6 5 6 6 6 5 6 6 6 5 6 6 5 6 6 5 5 5 5 5 5 5 5 6 6 6 5
 [721] 6 5 6 5 6 5 6 6 6 5 6 6 5 6 6 6 6 5 6 6 5 6 6 5 6 6 6 5 6 5 6 6 5 6 5 5 5 6 6 5 6 6 6 6 6
 [766] 6 6 5 6 5 6 6 5 6 5 5 6 5 6 6 6 5 6 6 5 6 6 6 6 6 6 5 6 5 6 6 6 5 6 6 5 5 6 6 6 6 6 5 6 6
 [811] 6 6 6 6 6 6 6 5 6 6 6 6 6 6 6 5 6 6 6 6 6 6 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 5 6 6 6 6 6 6
 [856] 6 6 5 6 6 6 6 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6 5 6 6 6 6 6 6 6 6 6 6 5 6 6 6 6 6 6 6 6 6 6 6
 [901] 6 6 6 6 6 6 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 5 6 6 6 6 6 6 6 6 6
 [946] 5 6 6 6 6 6 6 6 6 6 6 5 6 6 5 6 6 6 6 6 5 6 6 6 6 6 6 6 6 6 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6
 [991] 6 5 5 6 5 6 6 6 6 5
 [ reached getOption("max.print") -- omitted 932 entries ]
Levels: 3 4 5 6 7 8 9
> confusionMatrix(table(test_pred,testing$quality))
Confusion Matrix and Statistics


test_pred   3   4   5   6   7   8   9
        3   0   0   0   0   0   0   0
        4   0   0   0   0   0   0   0
        5   9  41 364 197  25   5   0
        6   1  29 263 648 300  49   1
        7   0   0   0   0   0   0   0
        8   0   0   0   0   0   0   0
        9   0   0   0   0   0   0   0

Overall Statistics

               Accuracy : 0.5238
                 95% CI : (0.5013, 0.5463)
    No Information Rate : 0.4374
    P-Value [Acc > NIR] : 1.475e-14

                  Kappa : 0.2064

 Mcnemar's Test P-Value : NA

Statistics by Class:


Statistics by Class:

                     Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8  Class: 9
Sensitivity          0.000000  0.00000   0.5805   0.7669   0.0000  0.00000 0.0000000
Specificity          1.000000  1.00000   0.7877   0.4085   1.0000  1.00000 1.0000000
Pos Pred Value            NaN      NaN   0.5679   0.5019      NaN      NaN       NaN
```

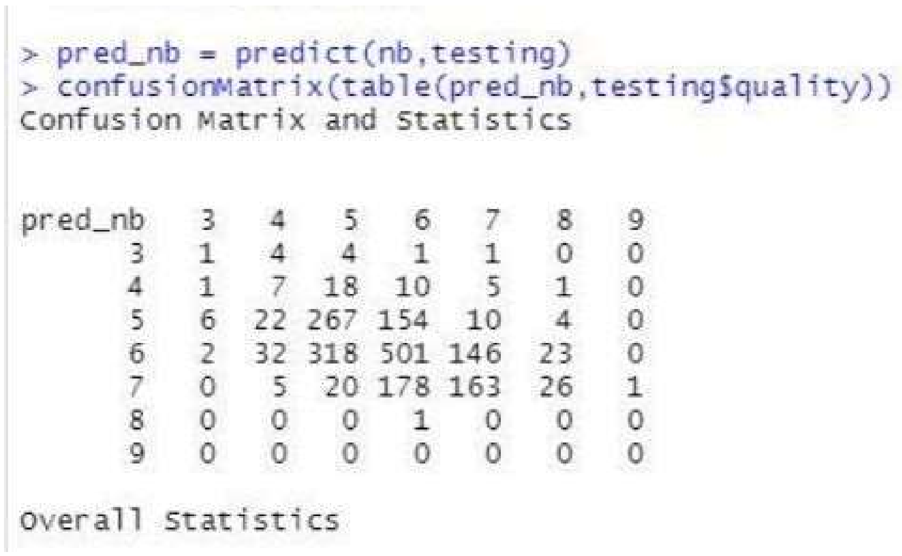
d. Naive Bayes Classifier



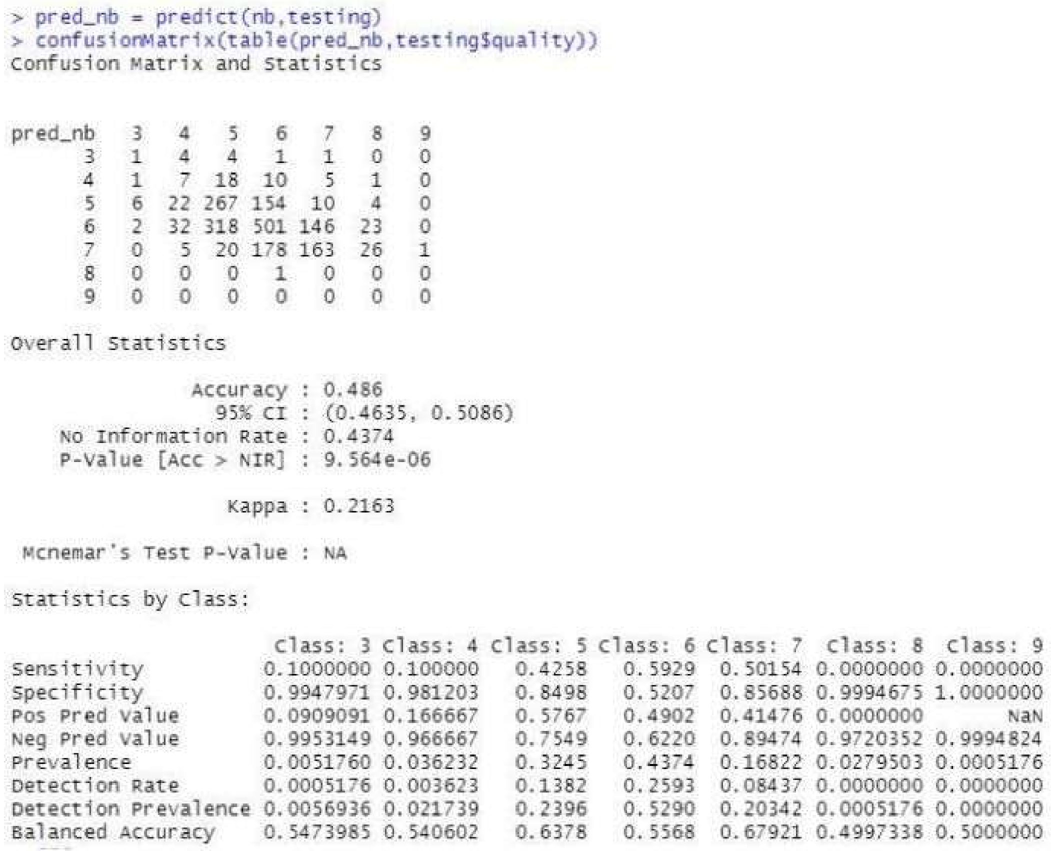
```
> nb = naiveBayes(quality~. -density -fixed.acidity -citric.acid,data = training)
> nb

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
           3            4            5            6            7            8            9
0.0044140366 0.0317810638 0.3312734496 0.4358861179 0.1653056720 0.0304568528 0.0008828073

Conditional probabilities:
   volatile.acidity
Y        [,1]      [,2]
  3 0.4557500 0.3527832
  4 0.4402083 0.2117839
  5 0.3908195 0.1755336
  6 0.3131899 0.1475216
  7 0.2881375 0.1153739
  8 0.2865580 0.1122852
  9 0.3125000 0.0550000

   residual.sugar
Y       [,1]     [,2]
  3 5.460000 4.735771
  4 4.094792 3.748761
  5 5.712692 4.885421
  6 5.559291 4.966949
  7 4.870427 4.134139
  8 5.251087 4.174748
  9 4.650000 4.119466

   chlorides
Y         [,1]        [,2]
  3 0.06290000 0.046070769
  4 0.06059722 0.054626042
  5 0.06463891 0.043319436
  6 0.05404759 0.030506095
  7 0.04559947 0.021779886
  8 0.04175362 0.016707362
  9 0.02650000 0.008266398

   free.sulfur.dioxide
Y       [,1]     [,2]
  3 50.25000 70.88584
  4 20.40972 15.89538
  5 30.12592 18.47110
  6 30.95038 16.99722
  7 30.54272 14.84191
  8 34.71014 17.48877
  9 35.00000 14.94434
```



```
   total.sulfur.dioxide
Y       [,1]      [,2]
  3 131.2750 110.22172
  4 105.9722  61.14124
  5 120.7761  59.99025
  6 115.3362  55.92453
  7 108.6228  48.56090
  8 113.8696  39.45460
  9 110.2500  17.42364

   pH
Y       [,1]       [,2]
  3 3.259000 0.19522996
  4 3.206181 0.17347133
  5 3.212139 0.15551385
  6 3.219362 0.16088250
  7 3.223632 0.16387651
  8 3.222174 0.15720524
  9 3.315000 0.09398581

   sulphates
Y        [,1]      [,2]
  3 0.4980000 0.1262245
  4 0.5072917 0.1830939
  5 0.5267355 0.1467288
  6 0.5326025 0.1446391
  7 0.5526836 0.1639743
  8 0.5088406 0.1632207
  9 0.4625000 0.1065755

   alcohol
Y       [,1]      [,2]
  3 10.38000 1.1967500
  4 10.16146 0.9930126
  5  9.84006 0.8175453
  6 10.58143 1.1237238
  7 11.33820 1.1967099
  8 11.69638 1.2761375
  9 12.10000 1.1518102

> pred_nb = predict(nb,testing)
> confusionMatrix(table(pred_nb,testing$quality))
Confusion Matrix and Statistics
```



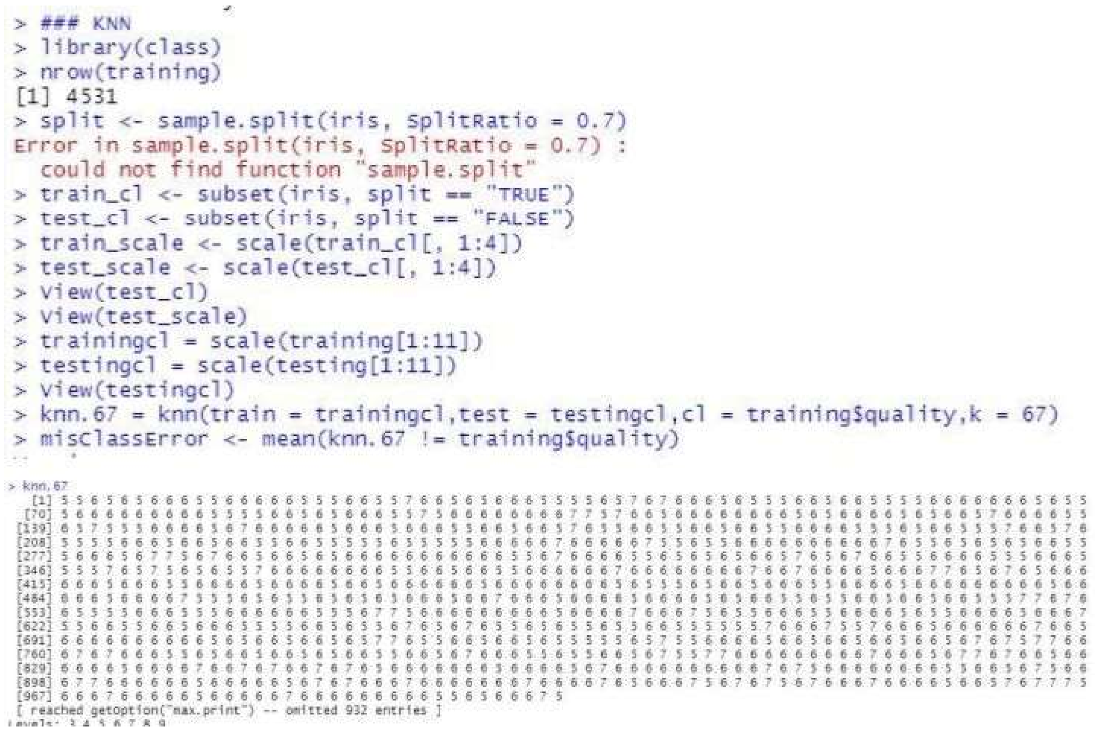
```
> pred_nb = predict(nb,testing)
> confusionMatrix(table(pred_nb,testing$quality))
Confusion Matrix and Statistics


pred_nb   3   4   5   6   7   8   9
      3   1   4   4   1   1   0   0
      4   1   7  18  10   5   1   0
      5   6  22 267 154  10   4   0
      6   2  32 318 501 146  23   0
      7   0   5  20 178 163  26   1
      8   0   0   0   1   0   0   0
      9   0   0   0   0   0   0   0

Overall Statistics
```



```
> pred_nb = predict(nb,testing)
> confusionMatrix(table(pred_nb,testing$quality))
Confusion Matrix and Statistics


pred_nb   3   4   5   6   7   8   9
      3   1   4   4   1   1   0   0
      4   1   7  18  10   5   1   0
      5   6  22 267 154  10   4   0
      6   2  32 318 501 146  23   0
      7   0   5  20 178 163  26   1
      8   0   0   0   1   0   0   0
      9   0   0   0   0   0   0   0

Overall Statistics
```

```
               Accuracy : 0.486
                 95% CI : (0.4635, 0.5086)
    No Information Rate : 0.4374
    P-Value [Acc > NIR] : 9.564e-06

                  Kappa : 0.2163

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 3 Class: 4 Class: 5 Class: 6 Class: 7  Class: 8  Class: 9
Sensitivity          0.1000000 0.100000   0.4258   0.5929  0.50154 0.0000000 0.0000000
Specificity          0.9947971 0.981203   0.8498   0.5207  0.85688 0.9994675 1.0000000
Pos Pred Value       0.0909091 0.166667   0.5767   0.4902  0.41476 0.0000000       NaN
Neg Pred Value       0.9953149 0.966667   0.7549   0.6220  0.89474 0.9720352 0.9994824
Prevalence           0.0051760 0.036232   0.3245   0.4374  0.16822 0.0279503 0.0005176
Detection Rate       0.0005176 0.003623   0.1382   0.2593  0.08437 0.0000000 0.0000000
Detection Prevalence 0.0056936 0.021739   0.2396   0.5290  0.20342 0.0005176 0.0000000
Balanced Accuracy    0.5473985 0.540602   0.6378   0.5568  0.67921 0.4997338 0.5000000
```

e. k-Nearest Neighbour



```
> ### KNN
> library(class)
> nrow(training)
[1] 4531
> split <- sample.split(iris, SplitRatio = 0.7)
Error in sample.split(iris, SplitRatio = 0.7) :
  could not find function "sample.split"
> train_cl <- subset(iris, split == "TRUE")
> test_cl <- subset(iris, split == "FALSE")
> train_scale <- scale(train_cl[, 1:4])
> test_scale <- scale(test_cl[, 1:4])
> View(test_cl)
> View(test_scale)
> trainingcl = scale(training[1:11])
> testingcl = scale(testing[1:11])
> View(testingcl)
> knn.67 = knn(train = trainingcl,test = testingcl,cl = training$quality,k = 67)
> misClassError <- mean(knn.67 != training$quality)
```

```
> knn.67
   [1] 5 5 6 5 6 5 6 6 6 5 5 6 6 6 6 6 5 5 5 6 6 5 5 7 6 6 5 6 5 6 6 6 5 5 5 5 6 5 7 6 7 6 6 6 5 6 5 5 5 6 6 5 6 6 5 5 5 5 6 6 6 6 6 6 6 5 6 5 5
  [70] 5 6 6 6 6 6 6 6 6 6 5 5 5 5 6 6 5 6 5 6 6 6 5 5 7 5 6 6 6 6 6 6 6 6 7 7 5 7 6 6 5 6 6 6 6 6 6 6 6 5 6 5 6 6 6 6 5 6 5 6 6 5 7 6 6 6 6 5 5
 [139] 6 5 7 5 5 5 6 6 6 6 5 6 7 6 6 6 6 6 5 6 6 6 5 6 6 6 5 5 6 6 5 6 6 5 7 6 5 5 6 6 5 5 6 6 5 6 6 5 5 6 6 6 6 5 5 5 6 5 6 6 5 5 5 7 6 6 5 7 6
 [208] 5 5 5 5 6 6 6 5 6 6 5 6 6 5 5 6 6 5 5 5 5 5 6 5 5 5 5 5 6 6 6 6 6 7 6 6 6 6 6 7 5 5 6 5 5 6 6 6 6 6 6 6 6 6 6 7 6 5 5 6 5 6 5 6 5 6 6 5 5
 [277] 5 6 6 6 5 6 7 7 5 6 7 6 6 5 6 6 5 6 5 6 6 6 6 6 6 6 6 6 6 6 5 5 6 7 6 6 6 6 5 5 6 5 6 5 6 5 6 6 5 7 6 5 6 7 6 6 5 5 6 6 6 6 5 5 5 6 6 6 5
 [346] 5 5 5 7 6 5 7 5 6 5 6 5 5 7 6 6 6 6 6 6 6 6 5 5 6 6 5 6 6 5 5 6 6 6 6 6 6 7 6 6 6 6 6 6 6 6 7 6 6 7 6 6 6 6 5 6 6 6 7 7 6 5 6 7 6 5 6 6 6
 [415] 6 6 6 5 6 6 6 5 5 6 6 6 6 5 6 6 6 6 5 6 6 5 6 6 6 6 6 6 5 6 6 6 6 6 6 6 6 5 6 5 5 5 6 5 6 6 5 6 6 6 5 5 6 6 6 6 5 6 6 6 6 6 6 6 6 6 5 6 6
 [484] 6 6 6 5 6 6 6 6 7 5 5 5 6 5 6 5 5 6 5 6 5 6 5 6 6 6 5 6 6 7 6 6 6 5 6 6 6 6 5 6 6 6 6 5 6 5 6 6 5 5 6 5 5 6 6 5 6 6 5 6 6 5 5 5 7 7 6 7 6
 [553] 6 5 5 5 5 6 6 6 5 5 5 6 6 6 6 6 6 5 5 5 6 7 7 5 6 6 6 6 6 6 6 6 6 5 6 6 6 6 7 6 6 6 7 5 6 5 5 6 6 6 5 5 6 6 6 6 5 6 5 5 6 6 6 6 5 6 6 6 7
 [622] 5 5 6 6 5 5 6 6 5 6 6 6 5 5 5 5 6 6 5 6 5 5 6 7 6 5 6 7 6 5 5 6 5 6 5 5 6 5 5 6 6 5 5 5 5 5 5 7 6 6 6 7 5 5 7 6 6 6 5 6 6 6 6 6 6 7 6 6 5
 [691] 6 6 6 6 6 6 6 6 6 6 5 6 5 6 6 5 6 6 5 6 5 7 7 6 5 5 6 6 5 6 6 5 6 5 5 5 5 5 6 5 7 5 5 6 6 6 6 5 6 6 5 6 6 5 6 6 5 6 6 5 6 7 6 7 5 7 7 6 6
 [760] 6 7 6 7 6 6 6 5 5 6 5 6 6 5 6 6 5 6 5 6 6 5 5 6 6 5 6 7 6 6 6 5 5 6 5 5 6 6 5 6 7 5 5 7 7 6 6 6 6 6 6 6 6 6 7 6 6 6 5 6 7 7 6 7 6 6 5 6 6
 [829] 6 6 6 6 5 6 6 6 6 7 6 6 7 6 7 6 6 7 6 7 6 5 6 6 6 6 6 6 6 5 6 6 6 6 5 6 7 6 6 6 6 6 6 6 6 6 6 7 6 7 5 6 6 6 6 6 6 6 6 5 5 6 6 5 6 7 5 6 6
 [898] 6 7 7 6 6 6 6 6 6 5 6 6 6 6 6 5 6 7 6 7 6 6 6 7 6 6 6 6 6 6 6 7 6 6 6 6 7 6 5 6 6 6 7 5 6 7 6 7 5 6 7 6 6 6 7 6 6 6 6 5 6 6 5 7 6 7 7 7 5
 [967] 6 6 6 7 6 6 6 6 6 5 6 6 6 6 6 7 6 6 6 6 6 6 6 6 6 5 5 6 5 6 6 6 7 5
 [ reached getOption("max.print") -- omitted 932 entries ]
Levels: 3 4 5 6 7 8 9
```

Ref:

Clustering: https://www.geeksforgeeks.org/k-means-clustering-in-r-programming/

Classification:

https://www.javatpoint.com/r-classification

https://www.geeksforgeeks.org/classification-in-r-programming/

Ref: naïve bayes - https://rpubs.com/riazakhan94/naive_bayes_classifier_e1071

Ref: SVM - https://rpubs.com/dimensionless/svm

Ref: K-nearest - https://rpubs.com/beane/n4_3