Applied Data Science Using R - INF 2167H - Assignment #3

Faria Khandaker

12/02/2020

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com (http://rmarkdown.rstudio.com).

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document.

```
#Libraries
library(dplyr)
library(tidyverse)
library(ggplot2)
library(rcompanion)
library(MASS)
library(randomForest)
library(MASS) # stepwise regression
library(leaps) # all subsets regression
library(corrplot)
library(caret)
library(FNN)
library(mlbench)
library(lattice)
library(class)
library(gmodels)
```

Question 1 (Machine Learning using Linear Regression)

1. Load the Abalone dataset from the “AppliedPredictiveModeling” package

```
#install.packages("AppliedPredictiveModeling")
library(AppliedPredictiveModeling)
```

```
## Warning: package 'AppliedPredictiveModeling' was built under R version 4.0.3
```

```
data("abalone")
```

2. The age of Abalone is the number of rings + 1.5. Use the tidyverse library to create the dependent variable “age”. Display the structure of the dataset after creating the age column and removing the Rings column.

```
abalone<- abalone%>%mutate(age=Rings+1.5)

abalone<-abalone[-9]

str(abalone)
```

```
## 'data.frame':    4177 obs. of  9 variables:
##  $ Type         : Factor w/ 3 levels "F","I","M": 3 3 1 3 2 2 1 1 3 1 ...
##  $ LongestShell : num  0.455 0.35 0.53 0.44 0.33 0.425 0.53 0.545 0.475 0.55 ...
##  $ Diameter     : num  0.365 0.265 0.42 0.365 0.255 0.3 0.415 0.425 0.37 0.44 ...
##  $ Height       : num  0.095 0.09 0.135 0.125 0.08 0.095 0.15 0.125 0.125 0.15 ...
##  $ WholeWeight  : num  0.514 0.226 0.677 0.516 0.205 ...
##  $ ShuckedWeight: num  0.2245 0.0995 0.2565 0.2155 0.0895 ...
##  $ VisceraWeight: num  0.101 0.0485 0.1415 0.114 0.0395 ...
##  $ ShellWeight  : num  0.15 0.07 0.21 0.155 0.055 0.12 0.33 0.26 0.165 0.32 ...
##  $ age          : num  16.5 8.5 10.5 11.5 8.5 9.5 21.5 17.5 10.5 20.5 ...
```

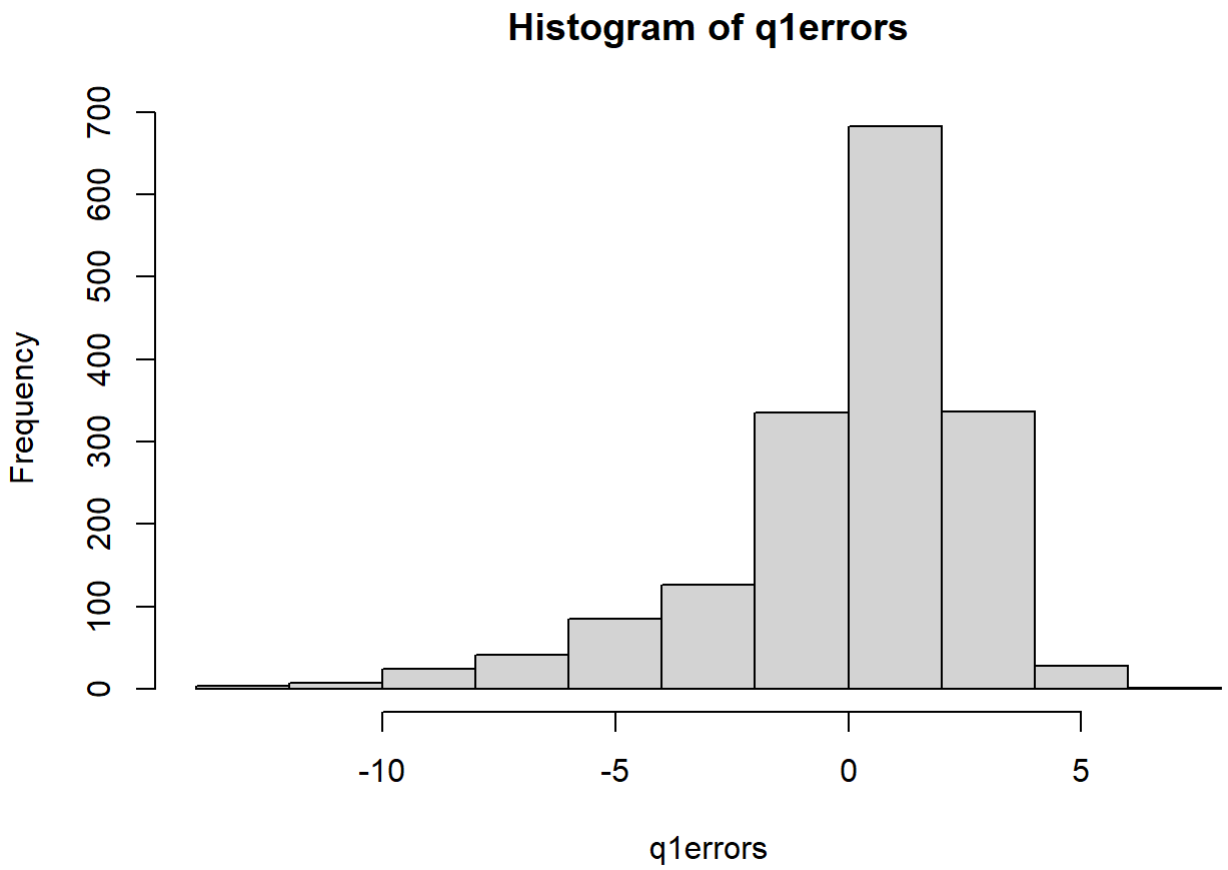
3. Use linear regression to train a model to predict abalone age from the abalone dataset (the one you updated in the previous question through creating age and removing rings). Create train and test sets in the ratio of 60:40. Set your seed to 123 for this question and from now on every time you set a seed in this assignment set it to 123. Display the summary of the model and interpret the coefficients and p-values of Type and ShuckedWeight variables.

```
#Type is a categorical variable consisting of female (considered the reference type), male, and infant(according to a Google search about the dataset). Shucked weight is the weight of the meat. The coefficients for the three variables shown in the summary table are all statistically significant because the p-values are all under 0.05. The coefficient for Type I or infant is -2.29 which mean that compared to females and which all other variables remaining constant, the age of the abalone is likely to decrease by 2.29 "units" for every 1 'unit' increase in type I. In other words, compared to females, infants are considered younger. And interpreting Type M in the same way, male abalones are usually younger than female abalones.

#For ShuckedWeight, every 1 unit increase in ShuckedWeight, or the weight of the meat, increases age by 3.88 units. So the older the abalone is, the more meat it is likely to have on it

set.seed(123)
#to get random indices
q1training<-sample(nrow(abalone),as.integer(nrow(abalone)*0.60))

q1train = abalone[q1training,]      ### training dataset, certain rows, all the columns
q1test = abalone[-q1training,]      ### test dataset

agemodel<-lm(age~Type+ShuckedWeight, data = q1train)
summary(agemodel)
```

```
##
## Call:
## lm(formula = age ~ Type + ShuckedWeight, data = q1train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.1821 -1.7759 -0.5713  1.0124 16.7940
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    10.9636     0.1694  64.720  < 2e-16 ***
## TypeI          -2.2853     0.1611 -14.186  < 2e-16 ***
## TypeM          -0.4598     0.1371  -3.354 0.000807 ***
## ShuckedWeight   3.8872     0.3050  12.744  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.809 on 2502 degrees of freedom
## Multiple R-squared:  0.2426, Adjusted R-squared:  0.2417
## F-statistic: 267.1 on 3 and 2502 DF,  p-value: < 2.2e-16
```

4. Predict the model, calculate the errors and the rmse. Interpret the model prediction.

```
#The root mean square error is calculated to be around 2.8 which means the predictions of the model are off by +/- 2.8. So t
he predicted ages are sometimes 2.8 years above or below the actual age. Calculating for relative change within values, it i
s found that 74.25% of the values predicted are within 25% of the actual value. When the errors are plotted in the form of a
histogram, they are found have left skew with the range of the errors being from negative 10 to six with most of the error c
oncentrating between negative two and positive two. The left skew of errors indicates that abalone age in this dataset does
 not follow a normal distribution.

agepred<- predict(agemodel,q1test)
q1errors <- agepred - q1test$age
q1rmse <- sqrt(mean((q1test$age - agepred)^2))
hist(q1errors)
```




```
q1rmse
```

```
## [1] 2.798687
```

```
q1rel_change <- abs(q1errors) / q1test$age
q1pred25 <- table(q1rel_change<0.25)["TRUE"] / nrow(q1test)
q1pred25
```

```
##      TRUE
## 0.7426691
```

##Question 2 (Feature Selection) ##### Use three methods of feature selection, name them and determine the best features to predict age of the abalone dataset. Use the same abalone dataset for which you created the age variable and removed rings.

Method 1: Forward Selection

```
fullaba <- lm(age~., data= abalone)  ## . means all the IVs
nullaba <- lm(age~1,data=abalone)
forward <- stepAIC(nullaba, scope=list(lower=nullaba, upper=fullaba), direction= "forward", trace=FALSE)
summary(forward)
```

```
##
## Call:
## lm(formula = age ~ ShellWeight + ShuckedWeight + Diameter + WholeWeight +
##     Type + VisceraWeight + Height, data = abalone)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.4644  -1.3041  -0.3427   0.8633  13.9571
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     5.37038    0.27536  19.503  < 2e-16 ***
## ShellWeight     8.75078    1.12405   7.785 8.73e-15 ***
## ShuckedWeight -19.80258    0.81490 -24.301  < 2e-16 ***
## Diameter       10.56951    0.98887  10.688  < 2e-16 ***
## WholeWeight     8.97751    0.72528  12.378  < 2e-16 ***
## TypeI          -0.82644    0.10220  -8.087 7.96e-16 ***
## TypeM           0.05755    0.08333   0.691     0.49
## VisceraWeight -10.61279    1.28782  -8.241 2.27e-16 ***
## Height         10.74911    1.53525   7.002 2.94e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.194 on 4168 degrees of freedom
## Multiple R-squared:  0.5379, Adjusted R-squared:  0.537
## F-statistic: 606.4 on 8 and 4168 DF,  p-value: < 2.2e-16
```

Method 2: Backward Selection

```
backward <- stepAIC(fullaba, direction= "backward", trace=FALSE)
summary(backward)
```

```
##
## Call:
## lm(formula = age ~ Type + Diameter + Height + WholeWeight + ShuckedWeight +
##     VisceraWeight + ShellWeight, data = abalone)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.4644  -1.3041  -0.3427   0.8633  13.9571
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     5.37038    0.27536  19.503  < 2e-16 ***
## TypeI          -0.82644    0.10220  -8.087 7.96e-16 ***
## TypeM           0.05755    0.08333   0.691     0.49
## Diameter       10.56951    0.98887  10.688  < 2e-16 ***
## Height         10.74911    1.53525   7.002 2.94e-12 ***
## WholeWeight     8.97751    0.72528  12.378  < 2e-16 ***
## ShuckedWeight -19.80258    0.81490 -24.301  < 2e-16 ***
## VisceraWeight -10.61279    1.28782  -8.241 2.27e-16 ***
## ShellWeight     8.75078    1.12405   7.785 8.73e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.194 on 4168 degrees of freedom
## Multiple R-squared:  0.5379, Adjusted R-squared:  0.537
## F-statistic: 606.4 on 8 and 4168 DF,  p-value: < 2.2e-16
```

Method 3: Stepwise Selection

```
stepwise<-stepAIC(forward, direction="both", trace=FALSE)
summary(stepwise)
```

```
##
## Call:
## lm(formula = age ~ ShellWeight + ShuckedWeight + Diameter + WholeWeight +
##     Type + VisceraWeight + Height, data = abalone)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.4644  -1.3041  -0.3427   0.8633  13.9571
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    5.37038    0.27536  19.503  < 2e-16 ***
## ShellWeight    8.75078    1.12405   7.785 8.73e-15 ***
## ShuckedWeight -19.80258    0.81490 -24.301  < 2e-16 ***
## Diameter      10.56951    0.98887  10.688  < 2e-16 ***
## WholeWeight    8.97751    0.72528  12.378  < 2e-16 ***
## TypeI         -0.82644    0.10220  -8.087 7.96e-16 ***
## TypeM          0.05755    0.08333   0.691     0.49
## VisceraWeight -10.61279    1.28782  -8.241 2.27e-16 ***
## Height        10.74911    1.53525   7.002 2.94e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.194 on 4168 degrees of freedom
## Multiple R-squared:  0.5379, Adjusted R-squared:  0.537
## F-statistic: 606.4 on 8 and 4168 DF,  p-value: < 2.2e-16
```

2. Determine computationally in two different ways which is the most important feature.

Method 1: NVMAX

```
# the most important feature shown through NVMAX is shell weight
subsets<-regsubsets(age~.,data=abalone, nbest=1,nvmax=3)   #nvmax is the number of attributes
sub.sum <- summary(subsets)
as.data.frame(sub.sum$outmat)
```

	TypeI	TypeM	LongestShell	Diameter	Height	WholeWeight	ShuckedWeight	VisceraWeight
	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>

	Typel <chr>	TypeM <chr>	LongestShell <chr>	Diameter <chr>	Height <chr>	WholeWeight <chr>	ShuckedWeight <chr>	VisceraWeight <chr>
1 (1)								
2 (1)							*	
3 (1)				*			*	

3 rows | 1-9 of 10 columns

Method 2: Random Forest

```
#importance tells you which variables have the most weight in estimating the target variable. Type is not there after featur
e selection

#install.packages("randomForest")
library(randomForest)
set.seed((123))
fullaba=randomForest(age~., data=abalone, ntree=500)
importance    <- importance(fullaba)
varImportance <- data.frame(Variables = row.names(importance),Importance = importance[ ,'IncNodePurity'])
x <- filter(varImportance, Importance>1500)   #keep only variables with importance >10
age <- abalone$age
features <- as.character(x$Variables)
cleanabalone <- cbind(age, abalone[,features])
```

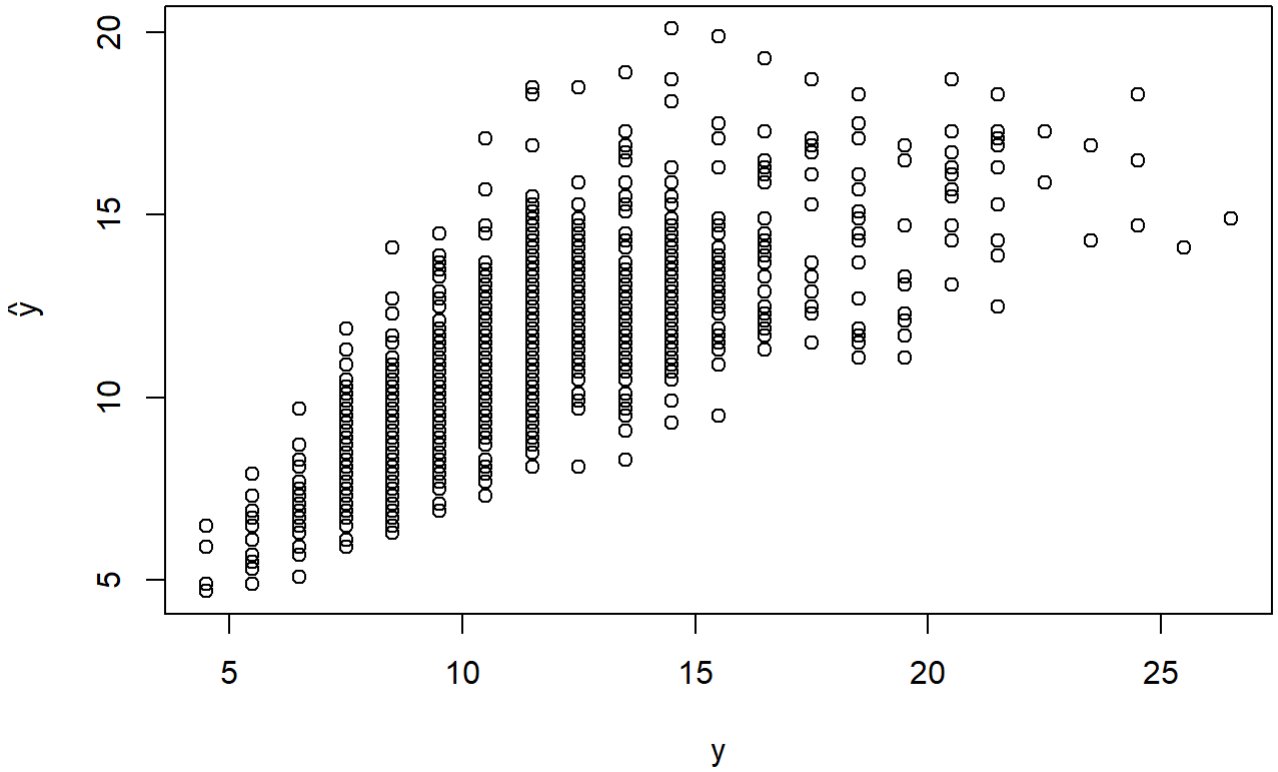
##Question 3 (Machine learning using KNN on a numeric dataset) ##### For this question, you will also use the abalone dataset for which you created the dependent variable age and removed rings in question 1.

1. Train a KNN model on the abalone dataset using a train/test ratio of 70-30 and a k of 5. Use all the variables in the abalone dataset to create the model.

```
abalone.numeric <- sapply( abalone[,2:9], as.numeric) #KNN only runs on numeric
set.seed(123)
q3trainindex <- sample(nrow(abalone.numeric), floor(nrow(abalone.numeric)*0.7))  #create train and test data sets
q3train <- abalone.numeric[q3trainindex,]
q3test <- abalone.numeric[-q3trainindex,]
q3train_labelsKNN <- q3train[,8]   ##DV in the training set
q3test_labelsKNN <- q3test[,8]     ##DV in the test set
q3train_KNN <- q3train[,-8]   ##Only keep IV in the training set
q3test_KNN <- q3test[,-8]     ##Only keep IV in the test set
set.seed(123)
q3KNNmodel <- knn.reg(train = q3train_KNN , test = q3test_KNN, y = q3train_labelsKNN , k = 5)
```

2. Calculate the performance measures of the KNN model you created in question 1 and interpret the performance measures

```
#the root mean square error for the KNN model predicting age of an abalone is 2.22. This means the predicted age and actual
 age is off by approximately 2.22 years. Considering that there are over 4000 entries in this dataset, a RMSE of 2.22 is can
be considered a good score.  The plotted graph which shows the predicted labels versus actual test labels have fairly overla
pping results. This indicates that our model is fairly good at predicting the age of abalone. A relative change percentage o
f 0.85 tells us that 86% of the time, the age prediction falls within 25% (higher or lower) of the real age. The RMSE  may b
e lower (which could lead to better relative change scores)if the dataset is normalized and run through the model again.

set.seed(123)
q3predicted <- q3KNNmodel$pred
plot(q3test_labelsKNN, q3predicted, xlab="y", ylab=expression(hat(y)))
```




```
q3errors <- q3predicted - q3test_labelsKNN
q3rmse <- sqrt(mean((q3test_labelsKNN-q3predicted)^2))
q3rmse
```

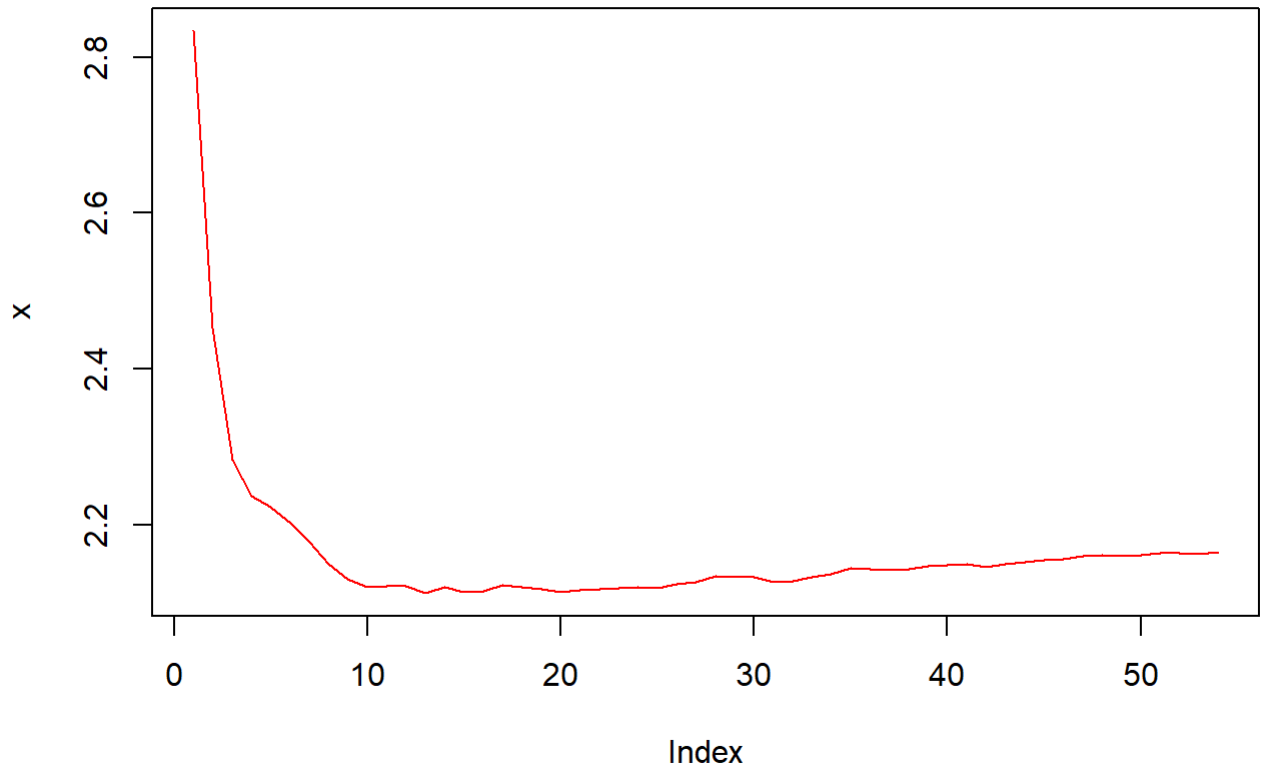
```
## [1] 2.222912
```

```
q3rel_change <- abs(q3errors) / q3test_labelsKNN
q3pred25 <- table(q3rel_change<0.25)["TRUE"] / nrow(q3test)
q3pred25
```

```
##      TRUE
## 0.8596491
```

3. Find the best value of K. Rerun the model and calculate its performance with the new k (if different from the original k=5)

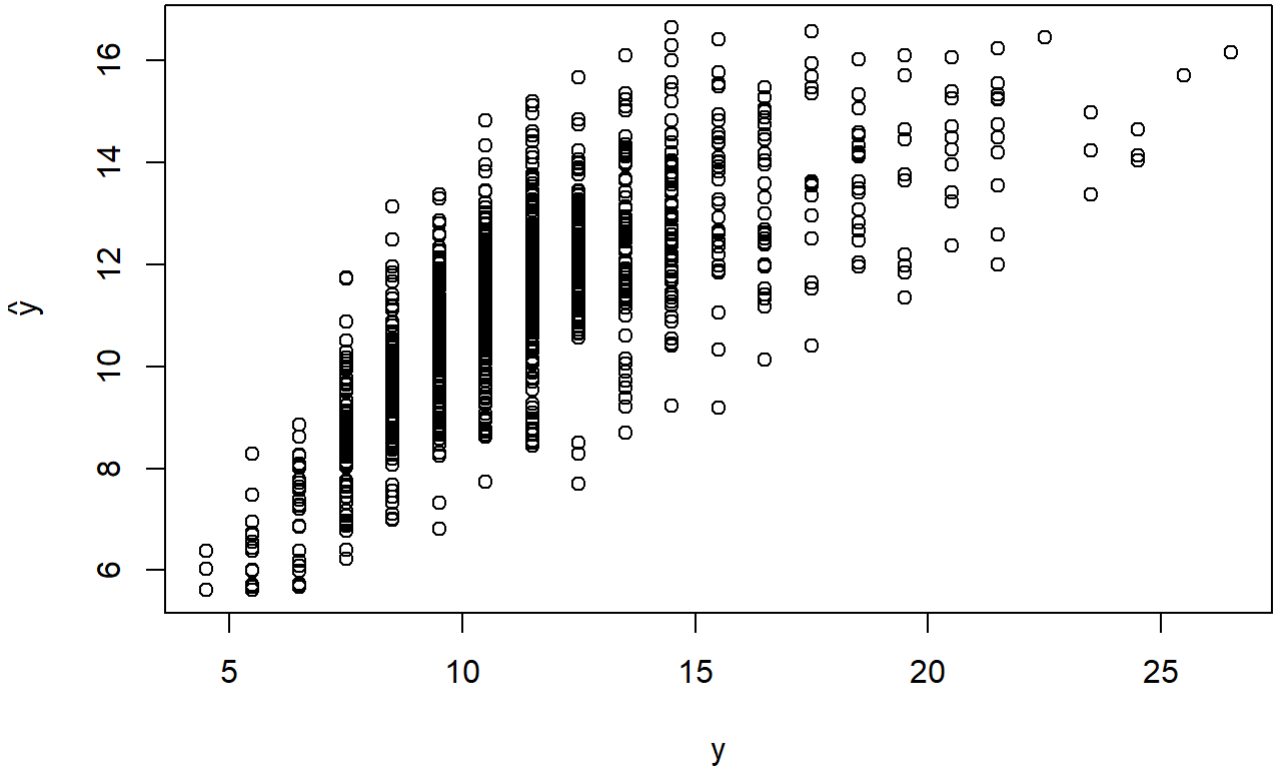
```
# this tells me that the best k is 13. I should re-run the model with k being 13

x <- 0
for (i in 1:sqrt(nrow(q3train)))   #try 1:11 to see the elbow clearly
{
set.seed(123)
KNNmodel <- knn.reg(train = q3train_KNN , test = q3test_KNN, y = q3train_labelsKNN, k = i)
predicted <- KNNmodel$pred
rmse <- sqrt(mean((q3test_labelsKNN-predicted)^2))
x[i] <- rmse
}
plot(x, type="l", col="red")
```




```
which.min(x)
```

```
## [1] 13
```

```
#After finding the best k to be 13 and using it to retrain the model, the performance of the model increased. The root mean
 square was found to be 2.11, which is a decrease of 0.11 units. The relative change of the values also changed from 86% to
 87%.

set.seed(123)
q3KNNmodel2 <- knn.reg(train = q3train_KNN , test = q3test_KNN, y = q3train_labelsKNN , k = 13)
q3predicted2 <- q3KNNmodel2$pred
plot(q3test_labelsKNN, predicted, xlab="y", ylab=expression(hat(y)))
```




```
q3errors2 <- q3predicted2 - q3test_labelsKNN
q3rmse2 <- sqrt(mean((q3test_labelsKNN-q3predicted2)^2))
q3rmse2
```

```
## [1] 2.112106
```

```
q3rel_change2 <- abs(q3errors2) / q3test_labelsKNN
q3pred252 <- table(q3rel_change2<0.25)["TRUE"] / nrow(q3test)
q3pred252
```

```
##      TRUE
## 0.8708134
```

Question 4 (Correlation)

Remove highly correlated (>=0.6) independent variables from the wine quality dataset. This is a dataset where wine quality is predicted from wine composition.

Import the dataset from http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv (http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv)

```
wine<-read.csv("http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv ",sep = ";")
```

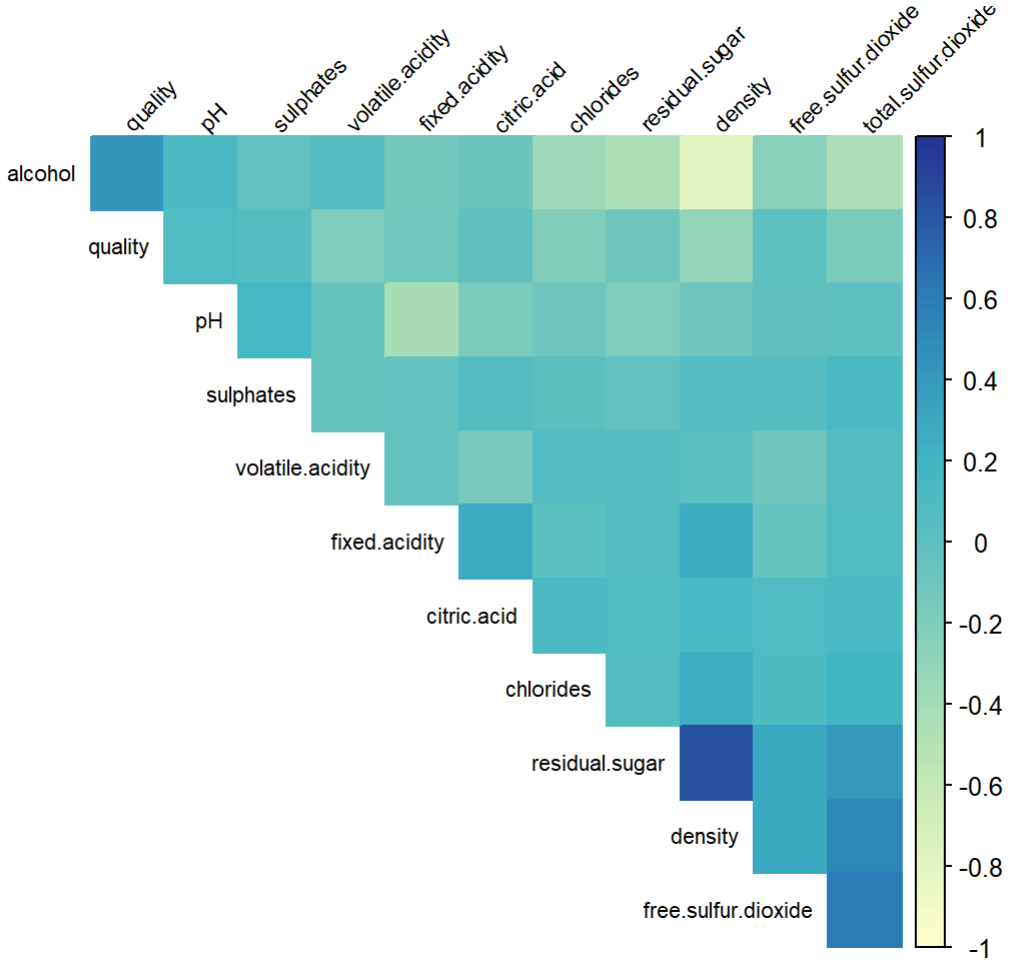
1. Display the correlation matrix and plot.

```
#This is the correlation matrix
cormat<- cor(wine, method = "pearson")
cormat   # a 6x6 matrix
```

```
##                       fixed.acidity volatile.acidity  citric.acid residual.sugar
## fixed.acidity           1.00000000      -0.02269729  0.289180698     0.08902070
## volatile.acidity       -0.02269729       1.00000000 -0.149471811     0.06428606
## citric.acid             0.28918070      -0.14947181  1.000000000     0.09421162
## residual.sugar          0.08902070       0.06428606  0.094211624     1.00000000
## chlorides               0.02308564       0.07051157  0.114364448     0.08868454
## free.sulfur.dioxide    -0.04939586      -0.09701194  0.094077221     0.29909835
## total.sulfur.dioxide    0.09106976       0.08926050  0.121130798     0.40143931
## density                 0.26533101       0.02711385  0.149502571     0.83896645
## pH                     -0.42585829      -0.03191537 -0.163748211    -0.19413345
## sulphates              -0.01714299      -0.03572815  0.062330940    -0.02666437
## alcohol                -0.12088112       0.06771794 -0.075728730    -0.45063122
## quality                -0.11366283      -0.19472297 -0.009209091    -0.09757683
##                        chlorides free.sulfur.dioxide total.sulfur.dioxide
## fixed.acidity         0.02308564       -0.0493958591          0.091069756
## volatile.acidity      0.07051157       -0.0970119393          0.089260504
## citric.acid           0.11436445        0.0940772210          0.121130798
## residual.sugar        0.08868454        0.2990983537          0.401439311
## chlorides             1.00000000        0.1013923521          0.198910300
## free.sulfur.dioxide   0.10139235        1.0000000000          0.615500965
## total.sulfur.dioxide  0.19891030        0.6155009650          1.000000000
## density               0.25721132        0.2942104109          0.529881324
## pH                   -0.09043946       -0.0006177961          0.002320972
## sulphates             0.01676288        0.0592172458          0.134562367
## alcohol              -0.36018871       -0.2501039415         -0.448892102
## quality              -0.20993441        0.0081580671         -0.174737218
##                          density            pH   sulphates     alcohol
## fixed.acidity         0.26533101 -0.4258582910 -0.01714299 -0.12088112
## volatile.acidity      0.02711385 -0.0319153683 -0.03572815  0.06771794
## citric.acid           0.14950257 -0.1637482114  0.06233094 -0.07572873
## residual.sugar        0.83896645 -0.1941334540 -0.02666437 -0.45063122
## chlorides             0.25721132 -0.0904394560  0.01676288 -0.36018871
## free.sulfur.dioxide   0.29421041 -0.0006177961  0.05921725 -0.25010394
## total.sulfur.dioxide  0.52988132  0.0023209718  0.13456237 -0.44889210
## density               1.00000000 -0.0935914935  0.07449315 -0.78013762
## pH                   -0.09359149  1.0000000000  0.15595150  0.12143210
## sulphates             0.07449315  0.1559514973  1.00000000 -0.01743277
## alcohol              -0.78013762  0.1214320987 -0.01743277  1.00000000
## quality              -0.30712331  0.0994272457  0.05367788  0.43557472
##                           quality
```

```
## fixed.acidity        -0.113662831
## volatile.acidity     -0.194722969
## citric.acid          -0.009209091
## residual.sugar       -0.097576829
## chlorides            -0.209934411
## free.sulfur.dioxide   0.008158067
## total.sulfur.dioxide -0.174737218
## density              -0.307123313
## pH                    0.099427246
## sulphates             0.053677877
## alcohol               0.435574715
## quality               1.000000000
```

```
#This is the correlation plot
col <- colorRampPalette(c("#FFFFCC","#C7E9B4","#7FCDBB","#40B6C4","#2C7FB8" ,"#253494"))
corrplot(cormat, method="color", col=col(200),
         type="upper", order="hclust",
         tl.col="black", tl.srt=45, tl.cex= 0.7, #Text label color and rotation
         # Combine with significance
         sig.level = 0.01,
         # hide correlation coefficient on the principal diagonal
         diag=FALSE
)
```



2. Interpret the correlation between alcohol and density, and between sulphates and free sulphur dioxide

```
#Interpreting correlation between alcohol and density, and between sulphates and free sulphur dioxide
cormat2<- wine[,c(6,8,10,11)]
correlationMatrix2 <- cor(cormat2, method = "pearson")
correlationMatrix2
```

```
##                      free.sulfur.dioxide     density   sulphates     alcohol
## free.sulfur.dioxide          1.00000000  0.29421041  0.05921725 -0.25010394
## density                      0.29421041  1.00000000  0.07449315 -0.78013762
## sulphates                    0.05921725  0.07449315  1.00000000 -0.01743277
## alcohol                     -0.25010394 -0.78013762 -0.01743277  1.00000000
```

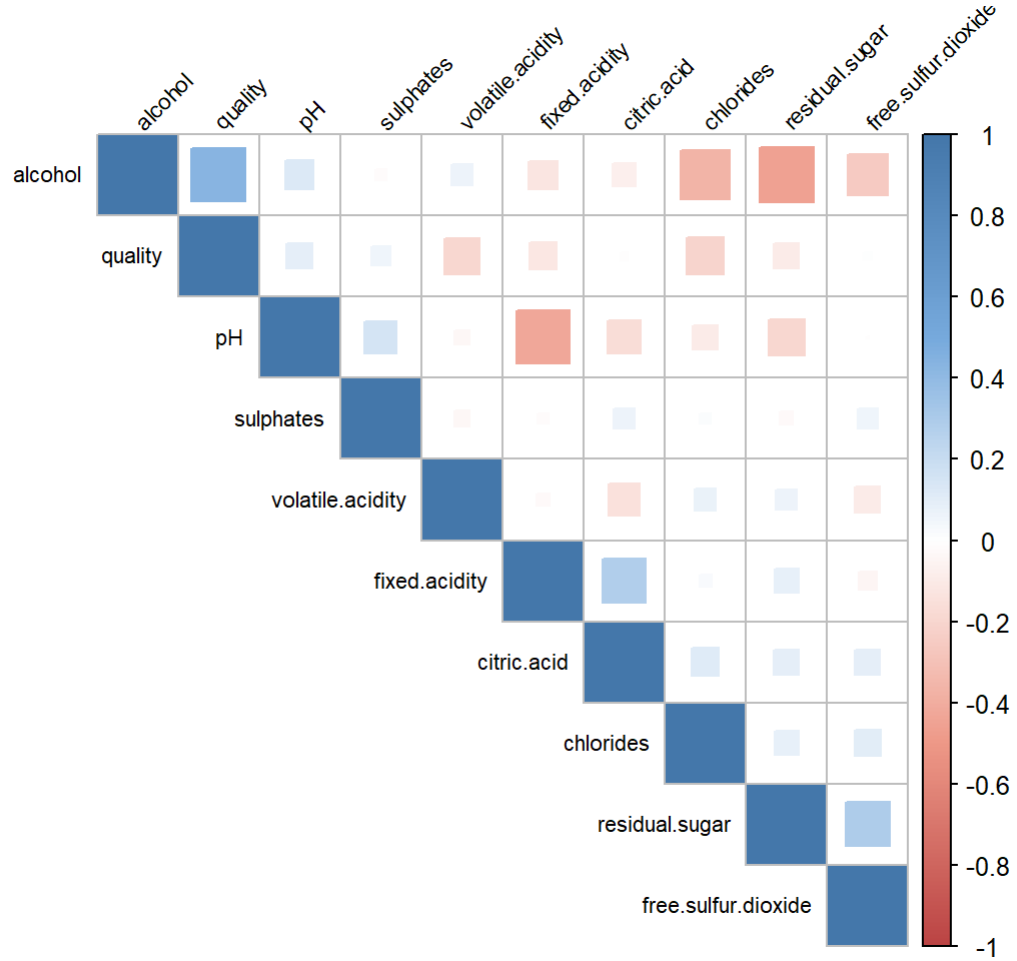
```
#There is a strong negative correlation between alcohol and density and a very weak positive correlation between sulphates a
nd free sulfur dioxide
```

3. Remove highly correlated variables.How many independent variables are left in the dataset? which variables were removed?

```
#there are 9 independent variables left in the dataset.
# the variables of density and total sulfur dioxide were removed
#Remove highly correlated
highlyCorrelated <- findCorrelation(cormat, cutoff = 0.6)
winenoncor <- wine[,-highlyCorrelated]  #keep only those not highly correlated

correlationMatrix2 <- cor(winenoncor, method = "pearson")  ### only numeric vars
correlationMatrix2
```

```
##                       fixed.acidity volatile.acidity  citric.acid residual.sugar
## fixed.acidity          1.00000000      -0.02269729  0.289180698     0.08902070
## volatile.acidity      -0.02269729       1.00000000 -0.149471811     0.06428606
## citric.acid            0.28918070      -0.14947181  1.000000000     0.09421162
## residual.sugar         0.08902070       0.06428606  0.094211624     1.00000000
## chlorides              0.02308564       0.07051157  0.114364448     0.08868454
## free.sulfur.dioxide   -0.04939586      -0.09701194  0.094077221     0.29909835
## pH                    -0.42585829      -0.03191537 -0.163748211    -0.19413345
## sulphates             -0.01714299      -0.03572815  0.062330940    -0.02666437
## alcohol               -0.12088112       0.06771794 -0.075728730    -0.45063122
## quality               -0.11366283      -0.19472297 -0.009209091    -0.09757683
##                        chlorides free.sulfur.dioxide            pH   sulphates
## fixed.acidity         0.02308564       -0.0493958591 -0.4258582910 -0.01714299
## volatile.acidity      0.07051157       -0.0970119393 -0.0319153683 -0.03572815
## citric.acid           0.11436445        0.0940772210 -0.1637482114  0.06233094
## residual.sugar        0.08868454        0.2990983537 -0.1941334540 -0.02666437
## chlorides             1.00000000        0.1013923521 -0.0904394560  0.01676288
## free.sulfur.dioxide   0.10139235        1.0000000000 -0.0006177961  0.05921725
## pH                   -0.09043946       -0.0006177961  1.0000000000  0.15595150
## sulphates             0.01676288        0.0592172458  0.1559514973  1.00000000
## alcohol              -0.36018871       -0.2501039415  0.1214320987 -0.01743277
## quality              -0.20993441        0.0081580671  0.0994272457  0.05367788
##                          alcohol      quality
## fixed.acidity        -0.12088112 -0.113662831
## volatile.acidity      0.06771794 -0.194722969
## citric.acid          -0.07572873 -0.009209091
## residual.sugar       -0.45063122 -0.097576829
## chlorides            -0.36018871 -0.209934411
## free.sulfur.dioxide  -0.25010394  0.008158067
## pH                    0.12143210  0.099427246
## sulphates            -0.01743277  0.053677877
## alcohol               1.00000000  0.435574715
## quality               0.43557472  1.000000000
```

```
col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))
corrplot(correlationMatrix2, method="square", col=col(200),
         type="upper", order="hclust",
         tl.col="black", tl.srt=45, tl.cex= 0.7, #Text label color and rotation
         sig.level = 0.01,
         diag=TRUE
)
```



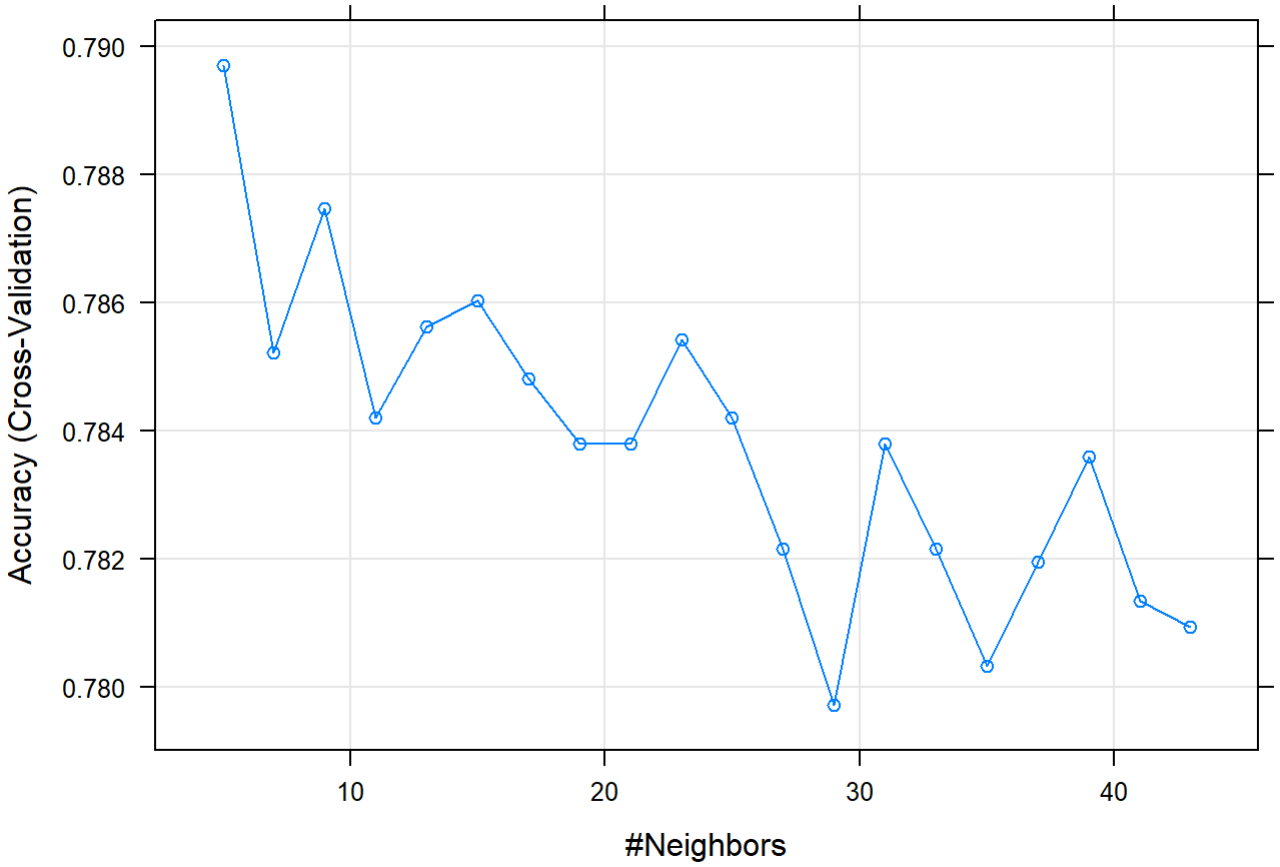
Question 5 (KNN classifier)

Reduce the levels of rating for wine quality to three levels as high, medium and low.

Consider high quality wine is >=7, and low quality wine is <=4. Then, build a KNN classifier (75:25 train:test ratio) for the Wine dataset after normalizing it and choosing the best K (avoiding k=1).

```
#winenoncor is the dataset with highly correlated variables removed
winequal <- function(x){
  ifelse(wine$quality>=7, "high", ifelse(wine$quality <=4, "low", "medium"))
}
winenoncor$quality<-winequal(winenoncor)
winenoncor$quality<-as.factor(winenoncor$quality)

#normalize dataset:
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
#normalizing the dataset,including only the numerical features
winenormal <- sapply(winenoncor[,1:9], normalize)
```

```
#training test from different dataset, labels from different dataset
set.seed(123)
rn_train <- sample(nrow(winenoncor), floor(nrow(winenoncor)*0.75))
#create train and test data sets, normalized dataset doesn't contain labels
train <- winenormal[rn_train,]
test <- winenormal[-rn_train,]
train4label<-winenoncor[rn_train,10]
test4label<-winenoncor[-rn_train,10]

#i couldn't figure out how to create a for loop to plot accuracy vs the number of k.
#looking online, i was able to use the caret library and the knn method for the train function
set.seed(123)
model <- train(
  quality ~., data = winenoncor, method = "knn",
  trControl = trainControl("cv", number = 10),
  preProcess = c("center","scale"),
  tuneLength = 20
  )
plot(model)
```




```
model$bestTune
```

	k
	<int>
1	5

1 row

```
#the parameter bestTune gave me best k as 5.

#This is the KNN classifier with the best k which is 5
set.seed(123)
q5model<-knn(train =train , test = test, cl = train4label , k = 5)
crosstab<-CrossTable(x=test4label, y=q5model, prop.chisq=FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  1225
##
##
##              | q5model
##   test4label |      high |       low |    medium | Row Total |
## -------------|-----------|-----------|-----------|-----------|
##         high |       120 |         1 |       143 |       264 |
##              |     0.455 |     0.004 |     0.542 |     0.216 |
##              |     0.543 |     0.125 |     0.144 |           |
##              |     0.098 |     0.001 |     0.117 |           |
## -------------|-----------|-----------|-----------|-----------|
##          low |         3 |         4 |        32 |        39 |
##              |     0.077 |     0.103 |     0.821 |     0.032 |
##              |     0.014 |     0.500 |     0.032 |           |
##              |     0.002 |     0.003 |     0.026 |           |
## -------------|-----------|-----------|-----------|-----------|
##       medium |        98 |         3 |       821 |       922 |
##              |     0.106 |     0.003 |     0.890 |     0.753 |
##              |     0.443 |     0.375 |     0.824 |           |
##              |     0.080 |     0.002 |     0.670 |           |
## -------------|-----------|-----------|-----------|-----------|
## Column Total |       221 |         8 |       996 |      1225 |
##              |     0.180 |     0.007 |     0.813 |           |
## -------------|-----------|-----------|-----------|-----------|
##
##
```

Display all performance measures of the KNN classifier. What can you tell about the prediction of different categories of wine quality? Interpret why you got these results?

```
totalmed<-sum(winenoncor$quality=='medium')
totallow<-sum(winenoncor$quality=='low')
totalhigh<-sum(winenoncor$quality=='high')
totalwines<-nrow(winenoncor)

#This is the function that sums the true positives and all the values in the confusion matrix to provide the accuracy of the model
q5accuracy <- function(x){sum(diag(x$t))/(sum(x$t)) * 100}
q5accuracy(crosstab)
```

```
## [1] 77.14286
```

```
#recall is tp/tp+fn. This means
#This function extracts the value for each label which was obtained by dividing the true positive by the row total
q5recall<-diag(crosstab$prop.row)

#precision is tp/tp+fp.
#This function extracts the value for each label which was obtained by dividing the true positive by the column total
q5precision<-diag(crosstab$prop.col)

#f1 score, which is the balance between precision and recall is 2(precision x recall/precisiont+recall)
q5f1<-2*((q5recall*q5precision)/(q5recall+q5precision))

#According to the crosstable the column total(FP) for high=101, for medium=175 and for low=4
#According to the crosstable, the row total(FN) for high=144, for medium=101 and for low=35

#the accuracy of the model with k=5 is 77.14%.
#The Precision for each label is: high=0.54 , medium=0.825 , low=0.5
#The Recall for each label is:  high= 0.455, medium=0.89 , low= 0.103
#The F1 score for each label is: high=0.495 , medium= 0.86, low=0.17


#The KNN model overall has an accuracy of 77.17% so it can predict the correct quality of wine 77% of the time.The medium quality wines have the best performance measures for all three scores.  The low quality wines have the worst measures of all,  for all three scores. With a model accuracy of 77.14 percent and the low number of observations(183 out of 4898 total wines) for low quality wine, it makes sense that the performance measures are so low. Medium quality wine has the most number of observations at a total of 3655 and if we follow the same pattern as low quality wines, it makes sense that they have the higher performance measures has the model has more examples to train on for medium quality wines.
```

Question 6 (ML classifiers and ensemble models)

1.Use the wine dataset consider wine of quality less than or equal to 5 as low and greater than five as high.

```
#Read the dataset
q6wine<-read.csv("http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv ",sep = ";")
# label wine quality as high or low
q6winequal <- function(x){
  ifelse(q6wine$quality>=5, "high", "low")
}
q6wine$quality<-q6winequal(q6wine)
# i asked Deena and she said i didn't have to normalize or get rid of correlating variables for this question.
```

2.Build individual classifiers of random forest, support vector machines, Naive Bayes and logistic regression. Consider you are interested in whether the model correctly predicts high wine quality. Calculate performance measures and decide which is the best model. Use 70:30 training:test ratio and seed of 123. Use 3 repeats of 10-fold cross-validation.

```
#The random forest model has the highest accuracy in predicting wine quality at 96.87% so it is the best performing model ou
t of all tested models. RF model has 97.25% precision (or how correctly the model is able to identify the wine quality ), 9
9.58% recall(or how correctly the model is able to retrieve each wine label) and an F1 score (the balance between precision
 and recall) of 98.4%.  Below all the models and their performance measures can be found.
```

```
#Building training/test sets
set.seed(123)
q6wine_train <- sample(nrow(q6wine), floor(nrow(q6wine)*0.7))
q6train <- q6wine[q6wine_train,]#training set with labels
q6test <- q6wine[-q6wine_train,]#test set with labels
q6trainlabel<-q6train[,12]
q6testlabel<-q6test[,12]
q6test<-q6test[,-12]#test set no labels
```

Random Forest

```
ctrl <- trainControl(method="repeatedcv", number =10, repeats=3)  #cross-validation
set.seed(123)

q6RFmodel_wine <- train(quality ~ ., data= q6train, method="rf", ntree=500, trControl = ctrl)
testpredRF_wine <- predict(q6RFmodel_wine, q6test)#takes the model and the test data
cf_RF_wine <- confusionMatrix(as.factor(testpredRF_wine), as.factor(q6testlabel),mode = "everything")
print(cf_RF_wine)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction high  low
##       high 1417   40
##       low     6    7
##
##                Accuracy : 0.9687
##                  95% CI : (0.9585, 0.977)
##     No Information Rate : 0.968
##     P-Value [Acc > NIR] : 0.4797
##
##                   Kappa : 0.2226
##
##  Mcnemar's Test P-Value : 1.141e-06
##
##             Sensitivity : 0.9958
##             Specificity : 0.1489
##          Pos Pred Value : 0.9725
##          Neg Pred Value : 0.5385
##               Precision : 0.9725
##                  Recall : 0.9958
##                      F1 : 0.9840
##              Prevalence : 0.9680
##          Detection Rate : 0.9639
##    Detection Prevalence : 0.9912
##       Balanced Accuracy : 0.5724
##
##        'Positive' Class : high
##
```

Support Vector Machines

```
set.seed(123)
q6SVMmodel_wine <- train(quality ~ ., data= q6train, method="svmPoly", trControl = ctrl)
test_predSVM_wine <- predict(q6SVMmodel_wine, q6test)
cf_SVM_wine <- confusionMatrix(as.factor(test_predSVM_wine), as.factor(q6testlabel), mode = "everything")
print(cf_SVM_wine)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction high  low
##       high 1422   44
##       low     1    3
##
##                Accuracy : 0.9694
##                  95% CI : (0.9593, 0.9776)
##     No Information Rate : 0.968
##     P-Value [Acc > NIR] : 0.4207
##
##                   Kappa : 0.1132
##
##  Mcnemar's Test P-Value : 3.825e-10
##
##             Sensitivity : 0.99930
##             Specificity : 0.06383
##          Pos Pred Value : 0.96999
##          Neg Pred Value : 0.75000
##               Precision : 0.96999
##                  Recall : 0.99930
##                      F1 : 0.98442
##              Prevalence : 0.96803
##          Detection Rate : 0.96735
##    Detection Prevalence : 0.99728
##       Balanced Accuracy : 0.53156
##
##        'Positive' Class : high
##
```

Naive Bayes

```
set.seed(123)
q6NBmodel_wine <- train(quality ~ ., data= q6train, method="naive_bayes", trControl = ctrl)
test_predNB_wine <- predict(q6NBmodel_wine, q6test)
cf_NB_wine <- confusionMatrix(as.factor(test_predNB_wine), as.factor(q6testlabel),mode = "everything")
print(cf_NB_wine)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction high  low
##       high 1397   37
##       low    26   10
##
##                Accuracy : 0.9571
##                  95% CI : (0.9455, 0.9669)
##     No Information Rate : 0.968
##     P-Value [Acc > NIR] : 0.9905
##
##                   Kappa : 0.2193
##
##  Mcnemar's Test P-Value : 0.2077
##
##             Sensitivity : 0.9817
##             Specificity : 0.2128
##          Pos Pred Value : 0.9742
##          Neg Pred Value : 0.2778
##               Precision : 0.9742
##                  Recall : 0.9817
##                      F1 : 0.9779
##              Prevalence : 0.9680
##          Detection Rate : 0.9503
##    Detection Prevalence : 0.9755
##       Balanced Accuracy : 0.5972
##
##        'Positive' Class : high
##
```

Logistic Regression

```
set.seed(123)
LRmodel_wine <- train(quality ~ ., data= q6train, method="glm", trControl = ctrl)
test_predLR_wine <- predict(LRmodel_wine, q6test)
cf_LR_wine <- confusionMatrix(as.factor(test_predLR_wine), as.factor(q6testlabel),mode = "everything")
print(cf_LR_wine)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction high  low
##       high 1418   47
##       low     5    0
##
##                Accuracy : 0.9646
##                  95% CI : (0.9539, 0.9735)
##     No Information Rate : 0.968
##     P-Value [Acc > NIR] : 0.7949
##
##                   Kappa : -0.0062
##
##  Mcnemar's Test P-Value : 1.303e-08
##
##             Sensitivity : 0.9965
##             Specificity : 0.0000
##          Pos Pred Value : 0.9679
##          Neg Pred Value : 0.0000
##               Precision : 0.9679
##                  Recall : 0.9965
##                      F1 : 0.9820
##              Prevalence : 0.9680
##          Detection Rate : 0.9646
##    Detection Prevalence : 0.9966
##       Balanced Accuracy : 0.4982
##
##        'Positive' Class : high
##
```

3. Build an ensemble model of all the previous models and calculate and interpret its performance measures.

#Below are the performance measures for the ensemble model. All models have very high accuracy scores (all within 90%). Afte r creating an the first ensemble and checking for model correlations, it was found that Random Forest and Naive bayes had hi gher correlation than other models and since Naive Bayes had a slightly lower accuracy than Random Forest (RF=96.6% vs NB=9 5.5%), Naive Bayes was taken out and another ensemble model was trained.

#The Ensemble model contained SVM, RF and GLM. It had an accuracy score of 96.73% which is only slightly higher(0.13% highe r) than the RF model by itself . The ensemble model performed better in predicting wine quality than any single model alone.

```
library(caretEnsemble)
```

```
## Warning: package 'caretEnsemble' was built under R version 4.0.3
```

```
##
## Attaching package: 'caretEnsemble'
```

```
## The following object is masked from 'package:ggplot2':
##
##     autoplot
```

```
control <- trainControl(method="repeatedcv", number = 10, repeats=3, savePredictions="final", classProbs = TRUE)
algorithmList <- c('rf', 'svmPoly', 'glm', 'naive_bayes')
set.seed(123)
models_wine <- caretList(quality~., data=q6train, trControl=control, methodList=algorithmList)
```

```
## Warning in trControlCheck(x = trControl, y = target): indexes not defined in
## trControl. Attempting to set them ourselves, so each model in the ensemble will
## have the same resampling indexes.
```

```
## maximum number of iterations reached 0.002789176 0.002764103maximum number of iterations reached 0.01055842 0.01004022max
imum number of iterations reached 0.005197162 0.005035314maximum number of iterations reached 0.003504447 0.0034352maximum n
umber of iterations reached 0.003337715 0.003276872maximum number of iterations reached 0.001779945 0.001764353maximum numbe
r of iterations reached 0.007353188 0.007032763maximum number of iterations reached 0.0007774624 0.000758144maximum number o
f iterations reached 0.0039519 0.003870885maximum number of iterations reached 0.00453641 0.004398212maximum number of itera
tions reached 0.004404544 0.004301708maximum number of iterations reached 0.005065369 0.004811313maximum number of iteration
s reached 3.612712e-05 3.605609e-05maximum number of iterations reached 0.001273238 0.001263002maximum number of iterations
reached 0.003599158 0.003526215maximum number of iterations reached 0.002707517 0.002681011maximum number of iterations reac
hed 0.007850395 0.007633379maximum number of iterations reached 0.006223314 0.005957033maximum number of iterations reached
0.004672963 0.00455859maximum number of iterations reached 0.00188743 0.001864217maximum number of iterations reached 0.0040
8605 0.003993229maximum number of iterations reached 0.006753089 0.006468564maximum number of iterations reached 0.002047988
0.001957648maximum number of iterations reached 0.003943037 0.003860837maximum number of iterations reached 0.002638961 0.00
2585106maximum number of iterations reached 0.004181923 0.004127006maximum number of iterations reached 0.007068656 0.006624
777maximum number of iterations reached 0.0043372 0.004238322maximum number of iterations reached 0.001988043 0.001966057max
imum number of iterations reached 0.004634786 0.004537006maximum number of iterations reached 0.008950882 0.008676955maximum
number of iterations reached 0.002403048 0.002337109maximum number of iterations reached 0.003264901 0.003209577maximum numb
er of iterations reached 0.005141926 0.004991553maximum number of iterations reached 0.00396792 0.003911148maximum number of
iterations reached 0.005216248 0.004983645maximum number of iterations reached 0.00161149 0.00154653maximum number of iterat
ions reached 0.004137985 0.004023767maximum number of iterations reached 0.003773745 0.003692841maximum number of iterations
reached 0.004142151 0.00403147maximum number of iterations reached 0.002151073 0.002079081maximum number of iterations reach
ed 0.002985385 0.002940425maximum number of iterations reached 0.003349319 0.003262786maximum number of iterations reached
0.002202461 0.00218324maximum number of iterations reached 0.01026107 0.009797322maximum number of iterations reached 0.0090
38574 0.008290643maximum number of iterations reached 0.004844754 0.004732673maximum number of iterations reached 0.00346008
0.003395311maximum number of iterations reached 0.003699205 0.003652341maximum number of iterations reached 0.01143452 0.010
39115maximum number of iterations reached 0.002171453 0.002056633maximum number of iterations reached 0.004886732 0.00473038
2maximum number of iterations reached 0.003780064 0.003701941maximum number of iterations reached 0.004287122 0.004170519max
imum number of iterations reached 0.003737838 0.003531528maximum number of iterations reached 0.0009847783 0.0009808462maxim
um number of iterations reached 0.004068075 0.003977518maximum number of iterations reached 0.004453552 0.004344904maximum n
umber of iterations reached 0.009700855 0.0093128maximum number of iterations reached 0.01069464 0.01022783maximum number of
iterations reached 0.003965832 0.003875738maximum number of iterations reached 0.004092201 0.004002169maximum number of iter
ations reached 0.002810078 0.002772392maximum number of iterations reached 0.008696852 0.008224532maximum number of iteratio
ns reached 0.009208928 0.008695825maximum number of iterations reached 0.002296332 0.002258872maximum number of iterations r
eached 0.003255515 0.003189001maximum number of iterations reached 0.002980567 0.002912947maximum number of iterations reach
ed 0.009002725 0.008502053maximum number of iterations reached 0.002254009 0.002188195maximum number of iterations reached
0.004203043 0.004085031maximum number of iterations reached 0.003772983 0.003711764maximum number of iterations reached 0.00
3952807 0.00388291maximum number of iterations reached 0.01208124 0.01138305maximum number of iterations reached 0.006394153
0.005995813maximum number of iterations reached 0.00682729 0.006649723maximum number of iterations reached 0.004711993 0.004
641888maximum number of iterations reached 0.001974502 0.001964305maximum number of iterations reached 0.009616709 0.0092209
34maximum number of iterations reached 0.001083749 0.001059108maximum number of iterations reached 0.006460873 0.006336936ma
ximum number of iterations reached 0.006002482 0.005882788maximum number of iterations reached 0.006935356 0.006727551maximu
```

m number of iterations reached 0.003223132 0.003072655maximum number of iterations reached 0.005827062 0.005700216maximum nu
mber of iterations reached 0.008547903 0.00818777maximum number of iterations reached 0.002510959 0.002485493maximum number
of iterations reached 0.008045934 0.007842878maximum number of iterations reached 0.005527821 0.005257633maximum number of i
terations reached 0.001798204 0.001786087maximum number of iterations reached 0.00363236 0.003550835maximum number of iterat
ions reached 0.003800923 0.003720852maximum number of iterations reached 0.006555707 0.006257295maximum number of iterations
reached 0.001612885 0.001569809maximum number of iterations reached 0.003193779 0.003148835maximum number of iterations reac
hed 0.002627059 0.002603512maximum number of iterations reached 0.001921264 0.001905611maximum number of iterations reached
0.004211434 0.003984755maximum number of iterations reached 0.001888154 0.001866656maximum number of iterations reached 0.00
1799968 0.001789451maximum number of iterations reached 0.002008106 0.001994519maximum number of iterations reached 0.009394
7 0.008994613maximum number of iterations reached 0.006249752 0.005850473maximum number of iterations reached 0.003957682 0.
003873611maximum number of iterations reached 0.002729896 0.002688167maximum number of iterations reached 0.004630459 0.0045
16768maximum number of iterations reached 0.009684878 0.009076086maximum number of iterations reached 0.0001337268 0.0001330
383maximum number of iterations reached 0.003626624 0.003575884maximum number of iterations reached 0.004403767 0.004297871m
aximum number of iterations reached 0.004503582 0.004387497maximum number of iterations reached 0.003086069 0.002964362maxim
um number of iterations reached 0.00299989 0.002962669maximum number of iterations reached 0.003545708 0.003473696maximum nu
mber of iterations reached 0.0006701665 0.0006679449maximum number of iterations reached 0.009741893 0.009432743maximum numb
er of iterations reached 0.004983824 0.004800111maximum number of iterations reached 0.003375782 0.003306368maximum number o
f iterations reached 0.001911313 0.001888362maximum number of iterations reached 0.004039485 0.003927483maximum number of it
erations reached 0.007316168 0.007017992maximum number of iterations reached 0.002500871 0.002367505maximum number of iterat
ions reached 0.002658183 0.002624716maximum number of iterations reached 0.002800005 0.002743508maximum number of iterations
reached 0.003711934 0.003641948maximum number of iterations reached 0.004528416 0.004338961maximum number of iterations reac
hed 3.676717e-05 3.670295e-05maximum number of iterations reached 0.005661222 0.005498102maximum number of iterations reache
d 0.002407334 0.002369218maximum number of iterations reached 0.0015618 0.001553835maximum number of iterations reached 0.00
9033987 0.008627317maximum number of iterations reached 0.007053893 0.006753034maximum number of iterations reached 0.002059
488 0.002040991maximum number of iterations reached 0.003048768 0.003013154maximum number of iterations reached 0.004335248
0.004248695maximum number of iterations reached 0.00542332 0.005316378maximum number of iterations reached 0.001062762 0.001
043289maximum number of iterations reached 0.002517497 0.002474779maximum number of iterations reached 0.001666036 0.0016509
16maximum number of iterations reached 0.002440237 0.002406012maximum number of iterations reached 0.003667247 0.003527777ma
ximum number of iterations reached 0.004238275 0.00413775maximum number of iterations reached 0.002626282 0.002586276maximum
number of iterations reached 0.004942524 0.004797365maximum number of iterations reached 0.009685568 0.009394764maximum numb
er of iterations reached 0.004984919 0.00471405maximum number of iterations reached 0.005068506 0.004953143maximum number of
iterations reached 0.004576528 0.004457649maximum number of iterations reached 0.005279162 0.005152397maximum number of iter
ations reached 0.00795058 0.007503116maximum number of iterations reached 0.001562922 0.001523102maximum number of iteration
s reached 0.005227243 0.00510411maximum number of iterations reached 0.002859629 0.00280517maximum number of iterations reac
hed 0.006236313 0.00598671maximum number of iterations reached 0.002547582 0.002433175maximum number of iterations reached
0.00500854 0.004900769maximum number of iterations reached 0.003731909 0.003661032maximum number of iterations reached 0.002
895127 0.002868922maximum number of iterations reached 0.007995693 0.007732344maximum number of iterations reached 0.0070617
45 0.006617545maximum number of iterations reached 0.002564373 0.00253958maximum number of iterations reached 0.001755421 0.
001739131maximum number of iterations reached 0.005869237 0.00571463maximum number of iterations reached 0.007646692 0.00740
5524maximum number of iterations reached 0.0005008406 0.0004932376maximum number of iterations reached 0.002697347 0.0026677
05maximum number of iterations reached 0.001840461 0.001822788maximum number of iterations reached 0.002413806 0.002386795ma

ximum number of iterations reached 0.003977925 0.003763698maximum number of iterations reached 0.0006564256 0.000654696maxim
um number of iterations reached 0.002090819 0.002068905maximum number of iterations reached 0.002355547 0.002331708maximum n
umber of iterations reached 0.009377373 0.009196972maximum number of iterations reached 0.007483449 0.007090299maximum numbe
r of iterations reached 0.004737309 0.004608952maximum number of iterations reached 0.004733591 0.004614944maximum number of
iterations reached 0.002900242 0.002867119maximum number of iterations reached 0.008446866 0.007894206maximum number of iter
ations reached 0.0006074023 0.0006014776maximum number of iterations reached 0.004369554 0.004271982maximum number of iterat
ions reached 0.004791197 0.004648694maximum number of iterations reached 0.003515547 0.00343159maximum number of iterations
reached 0.004561977 0.004294957maximum number of iterations reached 0.002931036 0.002869469maximum number of iterations reac
hed 0.003436142 0.003376203maximum number of iterations reached 0.004518109 0.00441439maximum number of iterations reached
0.01028604 0.009853872maximum number of iterations reached 0.004763536 0.004562389maximum number of iterations reached 0.003
343868 0.003299668maximum number of iterations reached 0.003193013 0.003169167maximum number of iterations reached 0.0042041
71 0.00410831maximum number of iterations reached 0.009903619 0.009163031maximum number of iterations reached 0.0008341973
0.0008138748maximum number of iterations reached 0.00493404 0.004789531maximum number of iterations reached 0.004968471 0.00
4819578maximum number of iterations reached 0.004209469 0.004102786maximum number of iterations reached 0.002256582 0.002164
734maximum number of iterations reached 0.00601128 0.005821504maximum number of iterations reached 0.004020066 0.003933105ma
ximum number of iterations reached 0.002748684 0.002712011maximum number of iterations reached 0.01110414 0.01060753maximum
number of iterations reached 0.01037114 0.009836393maximum number of iterations reached 0.002030018 0.002007298maximum numbe
r of iterations reached 0.003290175 0.003226852maximum number of iterations reached 0.001865269 0.001854157maximum number of
iterations reached 0.004330601 0.00424922maximum number of iterations reached 0.007881252 0.007371816maximum number of itera
tions reached 0.003563987 0.003497026maximum number of iterations reached 0.004191243 0.004069643maximum number of iteration
s reached 0.004602432 0.004503621maximum number of iterations reached 0.006915237 0.006668664maximum number of iterations re
ached 0.00338093 0.003234749maximum number of iterations reached 0.004441397 0.004317572maximum number of iterations reached
0.001430186 0.001420745maximum number of iterations reached 0.00137515 0.001368122maximum number of iterations reached 0.008
777705 0.008480995maximum number of iterations reached 0.007476487 0.00710635maximum number of iterations reached 0.00263539
0.002590524maximum number of iterations reached 0.004733775 0.004598901maximum number of iterations reached 0.005330689 0.00
5207673maximum number of iterations reached 0.006853929 0.006685285maximum number of iterations reached 0.001485457 0.001433
577maximum number of iterations reached 0.003067337 0.003012921maximum number of iterations reached 0.002153497 0.002118339m
aximum number of iterations reached 0.002804321 0.002763648maximum number of iterations reached 0.002818192 0.002667391maxim
um number of iterations reached 0.003049927 0.00300752maximum number of iterations reached 0.004437135 0.004345525maximum nu
mber of iterations reached 0.003360428 0.003297357maximum number of iterations reached 0.009344521 0.008940323maximum number
of iterations reached 0.006763462 0.006449058maximum number of iterations reached 0.003854923 0.003758808maximum number of i
terations reached 0.002603261 0.00257926maximum number of iterations reached 0.0022905 0.00225938maximum number of iteration
s reached 0.007830266 0.007530698maximum number of iterations reached 0.001544657 0.001497465maximum number of iterations re
ached 0.003208765 0.00316031maximum number of iterations reached 0.002325165 0.002292769maximum number of iterations reached
0.003016649 0.002934272maximum number of iterations reached 0.004305341 0.004116962maximum number of iterations reached 0.00
3531823 0.003488287maximum number of iterations reached 0.003345618 0.003272869maximum number of iterations reached 0.003579
476 0.003542138maximum number of iterations reached 0.01136971 0.01066383maximum number of iterations reached 0.00732897 0.0
06768551maximum number of iterations reached 0.004555626 0.004494187maximum number of iterations reached 0.006114218 0.00597
6042maximum number of iterations reached 0.004215141 0.004152504maximum number of iterations reached 0.008124695 0.007689337
maximum number of iterations reached 0.0008716459 0.0008451512maximum number of iterations reached 0.00352441 0.003494058max
imum number of iterations reached 0.005406898 0.005278518maximum number of iterations reached 0.006643699 0.006513282maximum

number of iterations reached 0.005851941 0.005477835maximum number of iterations reached 2.931911e-05 2.928465e-05maximum nu
mber of iterations reached 0.005460149 0.005417056maximum number of iterations reached 0.007895314 0.007679921maximum number
of iterations reached 0.004234427 0.00413772maximum number of iterations reached 0.009306361 0.008926801maximum number of it
erations reached 0.005337513 0.005098294maximum number of iterations reached 0.003402633 0.003339638maximum number of iterat
ions reached 0.004430718 0.004323454maximum number of iterations reached 0.004681467 0.00457063maximum number of iterations
reached 0.007248206 0.00693162maximum number of iterations reached 0.002317067 0.002225703maximum number of iterations reach
ed 0.002693204 0.002659714maximum number of iterations reached 0.004681235 0.004564515maximum number of iterations reached
0.003018139 0.002965559maximum number of iterations reached 0.001860499 0.001796461maximum number of iterations reached 0.00
3753455 0.003657419maximum number of iterations reached 0.001659781 0.001644662maximum number of iterations reached 0.002099
929 0.002080052maximum number of iterations reached 0.0110984 0.01057173maximum number of iterations reached 0.004301798 0.0
0417637maximum number of iterations reached 0.003916472 0.003840959maximum number of iterations reached 0.00425508 0.0041509
72maximum number of iterations reached 0.006142813 0.0059097maximum number of iterations reached 0.009589673 0.009031318maxi
mum number of iterations reached 0.000749243 0.0007342996maximum number of iterations reached 0.005205308 0.00507588maximum
number of iterations reached 0.00522335 0.005028549maximum number of iterations reached 0.003499618 0.003418316maximum numbe
r of iterations reached 0.002104128 0.002034405maximum number of iterations reached 0.003929286 0.003850007maximum number of
iterations reached 0.004363404 0.004274115maximum number of iterations reached 0.003506728 0.003455573maximum number of iter
ations reached 0.01077092 0.01027453maximum number of iterations reached 0.006256814 0.005856157maximum number of iterations
reached 0.004376001 0.004286815maximum number of iterations reached 0.005157632 0.005017302maximum number of iterations reac
hed 0.001404441 0.001397366maximum number of iterations reached 0.009586163 0.008916991maximum number of iterations reached
0.0005178681 0.0005101567maximum number of iterations reached 0.004081637 0.004005536maximum number of iterations reached 0.
001466749 0.001456829maximum number of iterations reached 0.002596915 0.002559078maximum number of iterations reached 0.0057
49889 0.005376133maximum number of iterations reached 0.003443189 0.003379936maximum number of iterations reached 0.00258222
6 0.002547135maximum number of iterations reached 0.004351351 0.004261334maximum number of iterations reached 0.008733244 0.
00838496maximum number of iterations reached 0.006857684 0.006353113maximum number of iterations reached 0.004475639 0.00435
8452maximum number of iterations reached 0.005665421 0.005496628maximum number of iterations reached 0.002360753 0.002342572
maximum number of iterations reached 0.006167292 0.005886221maximum number of iterations reached 0.0006417587 0.0006346554ma
ximum number of iterations reached 0.004137481 0.004065584maximum number of iterations reached 0.003165928 0.00317005maximum
number of iterations reached 0.003758516 0.003659041maximum number of iterations reached 0.002238718 0.002169498maximum numb
er of iterations reached 0.003606379 0.003540698maximum number of iterations reached 0.005869165 0.005697122maximum number o
f iterations reached 0.001750474 0.001730248maximum number of iterations reached 0.007842974 0.007577741maximum number of it
erations reached 0.006873472 0.006390644maximum number of iterations reached 0.004340938 0.00423222maximum number of iterati
ons reached 0.002996296 0.002948941maximum number of iterations reached 0.005673961 0.005533246maximum number of iterations
reached 0.009767259 0.009158023maximum number of iterations reached 0.000869069 0.000847017maximum number of iterations reac
hed 0.004392148 0.004262849maximum number of iterations reached 0.004524327 0.004413745maximum number of iterations reached
0.005625618 0.0054765maximum number of iterations reached 0.003797342 0.003623059maximum number of iterations reached 0.0035
67921 0.003507105maximum number of iterations reached 0.003466288 0.003392974maximum number of iterations reached 0.0047341
0.004678916maximum number of iterations reached 0.01074796 0.0100676maximum number of iterations reached 0.007983394 0.00750
154maximum number of iterations reached 0.007289754 0.007130557maximum number of iterations reached 0.006047187 0.005902362m
aximum number of iterations reached 0.003279246 0.003236741maximum number of iterations reached 0.008454731 0.007984947maxim
um number of iterations reached 0.0002879769 0.0002848626maximum number of iterations reached 0.009001487 0.008562333maximum
number of iterations reached 0.007013 0.006777847maximum number of iterations reached 0.007204201 0.007035538maximum number

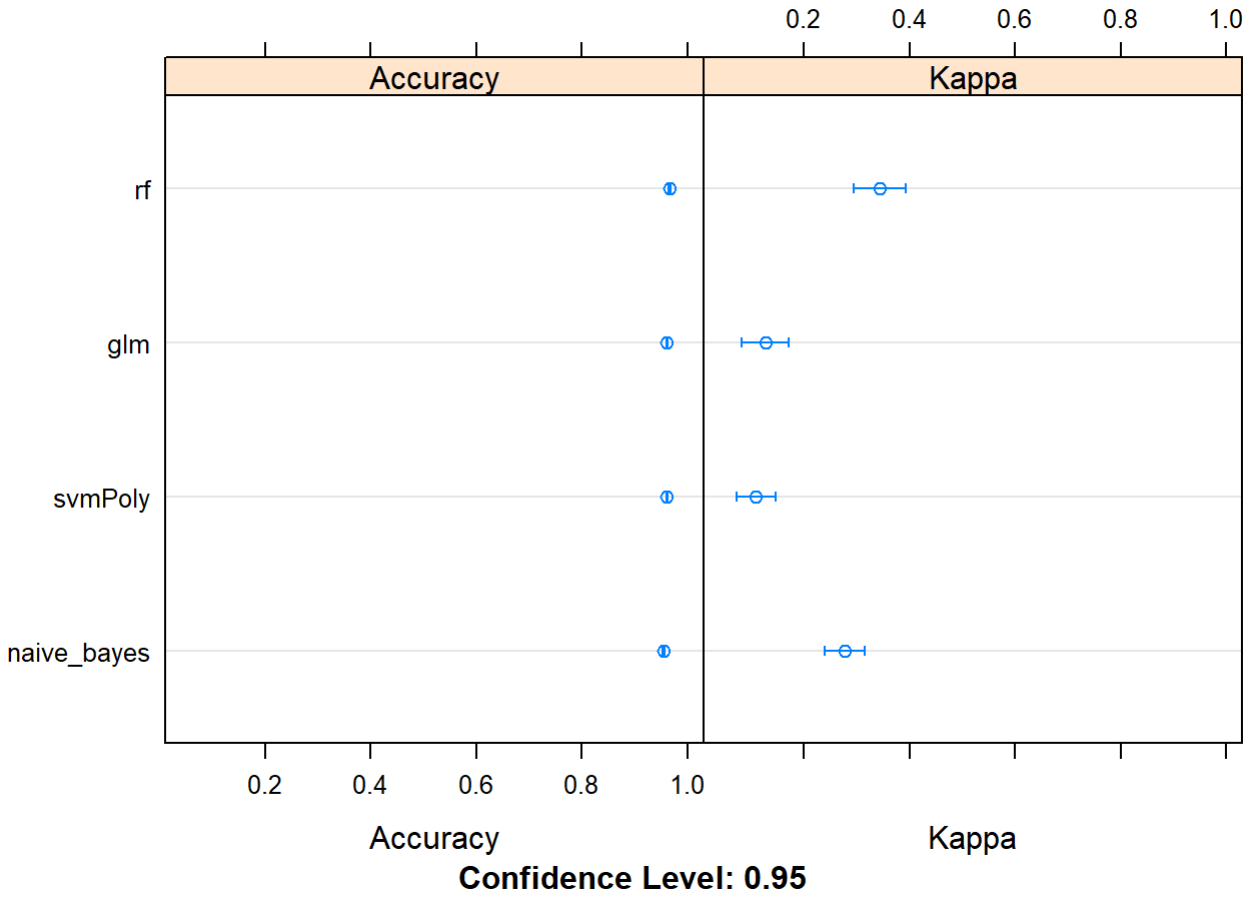
of iterations reached 0.003944891 0.003765467maximum number of iterations reached 0.006748169 0.006587513maximum number of i
terations reached 0.00449918 0.004434718maximum number of iterations reached 0.002992006 0.002959374maximum number of iterat
ions reached 0.009382282 0.00905086maximum number of iterations reached 0.006483126 0.005976964maximum number of iterations
reached 0.00606002 0.00589198maximum number of iterations reached 0.004186044 0.004055782maximum number of iterations reache
d 0.003764431 0.003692163maximum number of iterations reached 0.008743194 0.008344548maximum number of iterations reached 0.
001635954 0.001559492maximum number of iterations reached 0.003775858 0.003715822maximum number of iterations reached 0.0037
63575 0.003676693maximum number of iterations reached 0.003341016 0.003280379maximum number of iterations reached 0.00315170
7 0.002987786maximum number of iterations reached 0.00612804 0.005841632maximum number of iterations reached 0.002964701 0.0
02913587maximum number of iterations reached 0.003692681 0.003614555maximum number of iterations reached 0.008812191 0.00852
9364maximum number of iterations reached 0.007148605 0.006722723maximum number of iterations reached 0.004826209 0.004693915
maximum number of iterations reached 0.002049251 0.002020538maximum number of iterations reached 0.004460888 0.00436526maxim
um number of iterations reached 0.007447044 0.00719999maximum number of iterations reached 0.001330806 0.001295907maximum nu
mber of iterations reached 0.004499944 0.004414871maximum number of iterations reached 0.004408555 0.004282214maximum number
of iterations reached 0.001763541 0.001749692maximum number of iterations reached 0.001469525 0.001433032maximum number of i
terations reached 1.390471e-05 1.390198e-05maximum number of iterations reached 0.003049216 0.002992445maximum number of ite
rations reached 0.002773008 0.002737856maximum number of iterations reached 0.003265293 0.003206826maximum number of iterati
ons reached 0.008813664 0.008745318maximum number of iterations reached 0.006355991 0.006087174maximum number of iterations
reached 0.00453266 0.004437241maximum number of iterations reached 0.003027164 0.002971186maximum number of iterations reach
ed 0.003004536 0.00297479maximum number of iterations reached 0.01039497 0.009730407maximum number of iterations reached 0.0
003058092 0.000303059maximum number of iterations reached 0.002683509 0.002644807maximum number of iterations reached 0.0049
75851 0.00482157maximum number of iterations reached 0.001111551 0.001105992maximum number of iterations reached 0.003660012
0.003428322maximum number of iterations reached 0.004907275 0.004770178maximum number of iterations reached 0.002430958 0.00
2379459maximum number of iterations reached 0.0008288943 0.0008269206maximum number of iterations reached 0.007778755 0.0075
65258maximum number of iterations reached 0.007436513 0.007025775maximum number of iterations reached 0.002464178 0.00243891
6maximum number of iterations reached 0.001957702 0.001945606maximum number of iterations reached 0.003443314 0.003387241max
imum number of iterations reached 0.009328439 0.008838778maximum number of iterations reached 0.007332777 0.007043303maximum
number of iterations reached 0.003014997 0.002971574maximum number of iterations reached 0.002347648 0.002324876maximum numb
er of iterations reached 0.003003771 0.002941406maximum number of iterations reached 0.007822067 0.007479511maximum number o
f iterations reached 0.002247459 0.002167321maximum number of iterations reached 0.001596807 0.0015882maximum number of iter
ations reached 0.00404082 0.003975567maximum number of iterations reached 0.004406322 0.004342469maximum number of iteration
s reached 0.01212159 0.01149102maximum number of iterations reached 0.003456912 0.003278187maximum number of iterations reac
hed 0.004793862 0.004699012maximum number of iterations reached 0.003013879 0.002955842maximum number of iterations reached
0.001594912 0.001577482maximum number of iterations reached 0.008394836 0.007987189maximum number of iterations reached 0.00
06157777 0.0006029889maximum number of iterations reached 0.004633261 0.004507831maximum number of iterations reached 0.0041
19015 0.004055224maximum number of iterations reached 0.001810198 0.001795755maximum number of iterations reached 0.00397253
7 0.003799667maximum number of iterations reached 0.00333588 0.003274272maximum number of iterations reached 0.002296159 0.0
02272182maximum number of iterations reached 0.001925036 0.00191398maximum number of iterations reached 0.009373285 0.009257
046maximum number of iterations reached 0.005986239 0.005737263maximum number of iterations reached 0.003935498 0.003862316m
aximum number of iterations reached 0.00551955 0.005321314maximum number of iterations reached 0.003363283 0.00331702maximum
number of iterations reached 0.006958098 0.006699827maximum number of iterations reached 0.001411191 0.001379378maximum numb
er of iterations reached 0.003340372 0.003276417maximum number of iterations reached 0.00328007 0.003215035maximum number of

```
iterations reached 0.002082675 0.002063586maximum number of iterations reached 0.003000339 0.002895222maximum number of iter
ations reached 0.003959262 0.003887532maximum number of iterations reached 0.004451806 0.004345843
```

```
set.seed(123)
results_wine <- resamples(models_wine)
summary(results_wine)
```

```
##
## Call:
## summary.resamples(object = results_wine)
##
## Models: rf, svmPoly, glm, naive_bayes
## Number of resamples: 30
##
## Accuracy
##                  Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rf          0.9561404 0.9600015 0.9664254 0.9660639 0.9708455 0.9738372    0
## svmPoly     0.9475219 0.9591837 0.9619883 0.9614006 0.9642365 0.9707602    0
## glm         0.9533528 0.9591837 0.9619883 0.9615921 0.9649123 0.9680233    0
## naive_bayes 0.9416910 0.9475600 0.9561404 0.9547845 0.9592727 0.9651163    0
##
## Kappa
##                     Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rf           0.104750305 0.2649746 0.3434892 0.3437564 0.4609457 0.5315904    0
## svmPoly     -0.005471956 0.0000000 0.1285910 0.1095305 0.1381772 0.3659622    0
## glm         -0.010309278 0.0000000 0.1285910 0.1269126 0.2200614 0.3435115    0
## naive_bayes  0.075398017 0.2032935 0.2807671 0.2770020 0.3390001 0.4824473    0
```

```
dotplot(results_wine)
```




```
modelCor(results_wine)
```

```
##                    rf   svmPoly       glm naive_bayes
## rf          1.0000000 0.4938575 0.5323977   0.5624734
## svmPoly     0.4938575 1.0000000 0.5049027   0.3197998
## glm         0.5323977 0.5049027 1.0000000   0.5459277
## naive_bayes 0.5624734 0.3197998 0.5459277   1.0000000
```

Taking out Naive Bayes and rerunning ensemble model with rest of the algorithms

```
library(caretEnsemble)
control <- trainControl(method="repeatedcv", number = 10, repeats=3, savePredictions="final", classProbs = TRUE)
algorithmList2 <- c('rf', 'svmPoly', 'glm')
set.seed(123)
models_wine2 <- caretList(quality~., data=q6train, trControl=control, methodList=algorithmList2)
```

```
## Warning in trControlCheck(x = trControl, y = target): indexes not defined in
## trControl. Attempting to set them ourselves, so each model in the ensemble will
## have the same resampling indexes.
```

```
## maximum number of iterations reached 0.002789176 0.002764103maximum number of iterations reached 0.01055842 0.01004022max
imum number of iterations reached 0.005197162 0.005035314maximum number of iterations reached 0.003504447 0.0034352maximum n
umber of iterations reached 0.003337715 0.003276872maximum number of iterations reached 0.001779945 0.001764353maximum numbe
r of iterations reached 0.007353188 0.007032763maximum number of iterations reached 0.0007774624 0.000758144maximum number o
f iterations reached 0.0039519 0.003870885maximum number of iterations reached 0.00453641 0.004398212maximum number of itera
tions reached 0.004404544 0.004301708maximum number of iterations reached 0.005065369 0.004811313maximum number of iteration
s reached 3.612712e-05 3.605609e-05maximum number of iterations reached 0.001273238 0.001263002maximum number of iterations
reached 0.003599158 0.003526215maximum number of iterations reached 0.002707517 0.002681011maximum number of iterations reac
hed 0.007850395 0.007633379maximum number of iterations reached 0.006223314 0.005957033maximum number of iterations reached
0.004672963 0.00455859maximum number of iterations reached 0.00188743 0.001864217maximum number of iterations reached 0.0040
8605 0.003993229maximum number of iterations reached 0.006753089 0.006468564maximum number of iterations reached 0.002047988
0.001957648maximum number of iterations reached 0.003943037 0.003860837maximum number of iterations reached 0.002638961 0.00
2585106maximum number of iterations reached 0.004181923 0.004127006maximum number of iterations reached 0.007068656 0.006624
777maximum number of iterations reached 0.0043372 0.004238322maximum number of iterations reached 0.001988043 0.001966057max
imum number of iterations reached 0.004634786 0.004537006maximum number of iterations reached 0.008950882 0.008676955maximum
number of iterations reached 0.002403048 0.002337109maximum number of iterations reached 0.003264901 0.003209577maximum numb
er of iterations reached 0.005141926 0.004991553maximum number of iterations reached 0.00396792 0.003911148maximum number of
iterations reached 0.005216248 0.004983645maximum number of iterations reached 0.00161149 0.00154653maximum number of iterat
ions reached 0.004137985 0.004023767maximum number of iterations reached 0.003773745 0.003692841maximum number of iterations
reached 0.004142151 0.00403147maximum number of iterations reached 0.002151073 0.002079081maximum number of iterations reach
ed 0.002985385 0.002940425maximum number of iterations reached 0.003349319 0.003262786maximum number of iterations reached
0.002202461 0.00218324maximum number of iterations reached 0.01026107 0.009797322maximum number of iterations reached 0.0090
38574 0.008290643maximum number of iterations reached 0.004844754 0.004732673maximum number of iterations reached 0.00346008
0.003395311maximum number of iterations reached 0.003699205 0.003652341maximum number of iterations reached 0.01143452 0.010
39115maximum number of iterations reached 0.002171453 0.002056633maximum number of iterations reached 0.004886732 0.00473038
2maximum number of iterations reached 0.003780064 0.003701941maximum number of iterations reached 0.004287122 0.004170519max
imum number of iterations reached 0.003737838 0.003531528maximum number of iterations reached 0.0009847783 0.0009808462maxim
um number of iterations reached 0.004068075 0.003977518maximum number of iterations reached 0.004453552 0.004344904maximum n
umber of iterations reached 0.009700855 0.0093128maximum number of iterations reached 0.01069464 0.01022783maximum number of
iterations reached 0.003965832 0.003875738maximum number of iterations reached 0.004092201 0.004002169maximum number of iter
ations reached 0.002810078 0.002772392maximum number of iterations reached 0.008696852 0.008224532maximum number of iteratio
ns reached 0.009208928 0.008695825maximum number of iterations reached 0.002296332 0.002258872maximum number of iterations r
eached 0.003255515 0.003189001maximum number of iterations reached 0.002980567 0.002912947maximum number of iterations reach
ed 0.009002725 0.008502053maximum number of iterations reached 0.002254009 0.002188195maximum number of iterations reached
0.004203043 0.004085031maximum number of iterations reached 0.003772983 0.003711764maximum number of iterations reached 0.00
3952807 0.00388291maximum number of iterations reached 0.01208124 0.01138305maximum number of iterations reached 0.006394153
0.005995813maximum number of iterations reached 0.00682729 0.006649723maximum number of iterations reached 0.004711993 0.004
641888maximum number of iterations reached 0.001974502 0.001964305maximum number of iterations reached 0.009616709 0.0092209
34maximum number of iterations reached 0.001083749 0.001059108maximum number of iterations reached 0.006460873 0.006336936ma
ximum number of iterations reached 0.006002482 0.005882788maximum number of iterations reached 0.006935356 0.006727551maximu
```

m number of iterations reached 0.003223132 0.003072655maximum number of iterations reached 0.005827062 0.005700216maximum nu
mber of iterations reached 0.008547903 0.00818777maximum number of iterations reached 0.002510959 0.002485493maximum number
of iterations reached 0.008045934 0.007842878maximum number of iterations reached 0.005527821 0.005257633maximum number of i
terations reached 0.001798204 0.001786087maximum number of iterations reached 0.00363236 0.003550835maximum number of iterat
ions reached 0.003800923 0.003720852maximum number of iterations reached 0.006555707 0.006257295maximum number of iterations
reached 0.001612885 0.001569809maximum number of iterations reached 0.003193779 0.003148835maximum number of iterations reac
hed 0.002627059 0.002603512maximum number of iterations reached 0.001921264 0.001905611maximum number of iterations reached
0.004211434 0.003984755maximum number of iterations reached 0.001888154 0.001866656maximum number of iterations reached 0.00
1799968 0.001789451maximum number of iterations reached 0.002008106 0.001994519maximum number of iterations reached 0.009394
7 0.008994613maximum number of iterations reached 0.006249752 0.005850473maximum number of iterations reached 0.003957682 0.
003873611maximum number of iterations reached 0.002729896 0.002688167maximum number of iterations reached 0.004630459 0.0045
16768maximum number of iterations reached 0.009684878 0.009076086maximum number of iterations reached 0.0001337268 0.0001330
383maximum number of iterations reached 0.003626624 0.003575884maximum number of iterations reached 0.004403767 0.004297871m
aximum number of iterations reached 0.004503582 0.004387497maximum number of iterations reached 0.003086069 0.002964362maxim
um number of iterations reached 0.00299989 0.002962669maximum number of iterations reached 0.003545708 0.003473696maximum nu
mber of iterations reached 0.0006701665 0.0006679449maximum number of iterations reached 0.009741893 0.009432743maximum numb
er of iterations reached 0.004983824 0.004800111maximum number of iterations reached 0.003375782 0.003306368maximum number o
f iterations reached 0.001911313 0.001888362maximum number of iterations reached 0.004039485 0.003927483maximum number of it
erations reached 0.007316168 0.007017992maximum number of iterations reached 0.002500871 0.002367505maximum number of iterat
ions reached 0.002658183 0.002624716maximum number of iterations reached 0.002800005 0.002743508maximum number of iterations
reached 0.003711934 0.003641948maximum number of iterations reached 0.004528416 0.004338961maximum number of iterations reac
hed 3.676717e-05 3.670295e-05maximum number of iterations reached 0.005661222 0.005498102maximum number of iterations reache
d 0.002407334 0.002369218maximum number of iterations reached 0.0015618 0.001553835maximum number of iterations reached 0.00
9033987 0.008627317maximum number of iterations reached 0.007053893 0.006753034maximum number of iterations reached 0.002059
488 0.002040991maximum number of iterations reached 0.003048768 0.003013154maximum number of iterations reached 0.004335248
0.004248695maximum number of iterations reached 0.00542332 0.005316378maximum number of iterations reached 0.001062762 0.001
043289maximum number of iterations reached 0.002517497 0.002474779maximum number of iterations reached 0.001666036 0.0016509
16maximum number of iterations reached 0.002440237 0.002406012maximum number of iterations reached 0.003667247 0.003527777ma
ximum number of iterations reached 0.004238275 0.00413775maximum number of iterations reached 0.002626282 0.002586276maximum
number of iterations reached 0.004942524 0.004797365maximum number of iterations reached 0.009685568 0.009394764maximum numb
er of iterations reached 0.004984919 0.00471405maximum number of iterations reached 0.005068506 0.004953143maximum number of
iterations reached 0.004576528 0.004457649maximum number of iterations reached 0.005279162 0.005152397maximum number of iter
ations reached 0.00795058 0.007503116maximum number of iterations reached 0.001562922 0.001523102maximum number of iteration
s reached 0.005227243 0.00510411maximum number of iterations reached 0.002859629 0.00280517maximum number of iterations reac
hed 0.006236313 0.00598671maximum number of iterations reached 0.002547582 0.002433175maximum number of iterations reached
0.00500854 0.004900769maximum number of iterations reached 0.003731909 0.003661032maximum number of iterations reached 0.002
895127 0.002868922maximum number of iterations reached 0.007995693 0.007732344maximum number of iterations reached 0.0070617
45 0.006617545maximum number of iterations reached 0.002564373 0.00253958maximum number of iterations reached 0.001755421 0.
001739131maximum number of iterations reached 0.005869237 0.00571463maximum number of iterations reached 0.007646692 0.00740
5524maximum number of iterations reached 0.0005008406 0.0004932376maximum number of iterations reached 0.002697347 0.0026677
05maximum number of iterations reached 0.001840461 0.001822788maximum number of iterations reached 0.002413806 0.002386795ma

ximum number of iterations reached 0.003977925 0.003763698maximum number of iterations reached 0.0006564256 0.000654696maxim
um number of iterations reached 0.002090819 0.002068905maximum number of iterations reached 0.002355547 0.002331708maximum n
umber of iterations reached 0.009377373 0.009196972maximum number of iterations reached 0.007483449 0.007090299maximum numbe
r of iterations reached 0.004737309 0.004608952maximum number of iterations reached 0.004733591 0.004614944maximum number of
iterations reached 0.002900242 0.002867119maximum number of iterations reached 0.008446866 0.007894206maximum number of iter
ations reached 0.0006074023 0.0006014776maximum number of iterations reached 0.004369554 0.004271982maximum number of iterat
ions reached 0.004791197 0.004648694maximum number of iterations reached 0.003515547 0.00343159maximum number of iterations
reached 0.004561977 0.004294957maximum number of iterations reached 0.002931036 0.002869469maximum number of iterations reac
hed 0.003436142 0.003376203maximum number of iterations reached 0.004518109 0.00441439maximum number of iterations reached
0.01028604 0.009853872maximum number of iterations reached 0.004763536 0.004562389maximum number of iterations reached 0.003
343868 0.003299668maximum number of iterations reached 0.003193013 0.003169167maximum number of iterations reached 0.0042041
71 0.00410831maximum number of iterations reached 0.009903619 0.009163031maximum number of iterations reached 0.0008341973
0.0008138748maximum number of iterations reached 0.00493404 0.004789531maximum number of iterations reached 0.004968471 0.00
4819578maximum number of iterations reached 0.004209469 0.004102786maximum number of iterations reached 0.002256582 0.002164
734maximum number of iterations reached 0.00601128 0.005821504maximum number of iterations reached 0.004020066 0.003933105ma
ximum number of iterations reached 0.002748684 0.002712011maximum number of iterations reached 0.01110414 0.01060753maximum
number of iterations reached 0.01037114 0.009836393maximum number of iterations reached 0.002030018 0.002007298maximum numbe
r of iterations reached 0.003290175 0.003226852maximum number of iterations reached 0.001865269 0.001854157maximum number of
iterations reached 0.004330601 0.00424922maximum number of iterations reached 0.007881252 0.007371816maximum number of itera
tions reached 0.003563987 0.003497026maximum number of iterations reached 0.004191243 0.004069643maximum number of iteration
s reached 0.004602432 0.004503621maximum number of iterations reached 0.006915237 0.006668664maximum number of iterations re
ached 0.00338093 0.003234749maximum number of iterations reached 0.004441397 0.004317572maximum number of iterations reached
0.001430186 0.001420745maximum number of iterations reached 0.00137515 0.001368122maximum number of iterations reached 0.008
777705 0.008480995maximum number of iterations reached 0.007476487 0.00710635maximum number of iterations reached 0.00263539
0.002590524maximum number of iterations reached 0.004733775 0.004598901maximum number of iterations reached 0.005330689 0.00
5207673maximum number of iterations reached 0.006853929 0.006685285maximum number of iterations reached 0.001485457 0.001433
577maximum number of iterations reached 0.003067337 0.003012921maximum number of iterations reached 0.002153497 0.002118339m
aximum number of iterations reached 0.002804321 0.002763648maximum number of iterations reached 0.002818192 0.002667391maxim
um number of iterations reached 0.003049927 0.00300752maximum number of iterations reached 0.004437135 0.004345525maximum nu
mber of iterations reached 0.003360428 0.003297357maximum number of iterations reached 0.009344521 0.008940323maximum number
of iterations reached 0.006763462 0.006449058maximum number of iterations reached 0.003854923 0.003758808maximum number of i
terations reached 0.002603261 0.00257926maximum number of iterations reached 0.0022905 0.00225938maximum number of iteration
s reached 0.007830266 0.007530698maximum number of iterations reached 0.001544657 0.001497465maximum number of iterations re
ached 0.003208765 0.00316031maximum number of iterations reached 0.002325165 0.002292769maximum number of iterations reached
0.003016649 0.002934272maximum number of iterations reached 0.004305341 0.004116962maximum number of iterations reached 0.00
3531823 0.003488287maximum number of iterations reached 0.003345618 0.003272869maximum number of iterations reached 0.003579
476 0.003542138maximum number of iterations reached 0.01136971 0.01066383maximum number of iterations reached 0.00732897 0.0
06768551maximum number of iterations reached 0.004555626 0.004494187maximum number of iterations reached 0.006114218 0.00597
6042maximum number of iterations reached 0.004215141 0.004152504maximum number of iterations reached 0.008124695 0.007689337
maximum number of iterations reached 0.0008716459 0.0008451512maximum number of iterations reached 0.00352441 0.003494058max
imum number of iterations reached 0.005406898 0.005278518maximum number of iterations reached 0.006643699 0.006513282maximum

number of iterations reached 0.005851941 0.005477835maximum number of iterations reached 2.931911e-05 2.928465e-05maximum nu
mber of iterations reached 0.005460149 0.005417056maximum number of iterations reached 0.007895314 0.007679921maximum number
of iterations reached 0.004234427 0.00413772maximum number of iterations reached 0.009306361 0.008926801maximum number of it
erations reached 0.005337513 0.005098294maximum number of iterations reached 0.003402633 0.003339638maximum number of iterat
ions reached 0.004430718 0.004323454maximum number of iterations reached 0.004681467 0.00457063maximum number of iterations
reached 0.007248206 0.00693162maximum number of iterations reached 0.002317067 0.002225703maximum number of iterations reach
ed 0.002693204 0.002659714maximum number of iterations reached 0.004681235 0.004564515maximum number of iterations reached
0.003018139 0.002965559maximum number of iterations reached 0.001860499 0.001796461maximum number of iterations reached 0.00
3753455 0.003657419maximum number of iterations reached 0.001659781 0.001644662maximum number of iterations reached 0.002099
929 0.002080052maximum number of iterations reached 0.0110984 0.01057173maximum number of iterations reached 0.004301798 0.0
0417637maximum number of iterations reached 0.003916472 0.003840959maximum number of iterations reached 0.00425508 0.0041509
72maximum number of iterations reached 0.006142813 0.0059097maximum number of iterations reached 0.009589673 0.009031318maxi
mum number of iterations reached 0.000749243 0.0007342996maximum number of iterations reached 0.005205308 0.00507588maximum
number of iterations reached 0.00522335 0.005028549maximum number of iterations reached 0.003499618 0.003418316maximum numbe
r of iterations reached 0.002104128 0.002034405maximum number of iterations reached 0.003929286 0.003850007maximum number of
iterations reached 0.004363404 0.004274115maximum number of iterations reached 0.003506728 0.003455573maximum number of iter
ations reached 0.01077092 0.01027453maximum number of iterations reached 0.006256814 0.005856157maximum number of iterations
reached 0.004376001 0.004286815maximum number of iterations reached 0.005157632 0.005017302maximum number of iterations reac
hed 0.001404441 0.001397366maximum number of iterations reached 0.009586163 0.008916991maximum number of iterations reached
0.0005178681 0.0005101567maximum number of iterations reached 0.004081637 0.004005536maximum number of iterations reached 0.
001466749 0.001456829maximum number of iterations reached 0.002596915 0.002559078maximum number of iterations reached 0.0057
49889 0.005376133maximum number of iterations reached 0.003443189 0.003379936maximum number of iterations reached 0.00258222
6 0.002547135maximum number of iterations reached 0.004351351 0.004261334maximum number of iterations reached 0.008733244 0.
00838496maximum number of iterations reached 0.006857684 0.006353113maximum number of iterations reached 0.004475639 0.00435
8452maximum number of iterations reached 0.005665421 0.005496628maximum number of iterations reached 0.002360753 0.002342572
maximum number of iterations reached 0.006167292 0.005886221maximum number of iterations reached 0.0006417587 0.0006346554ma
ximum number of iterations reached 0.004137481 0.004065584maximum number of iterations reached 0.003165928 0.00317005maximum
number of iterations reached 0.003758516 0.003659041maximum number of iterations reached 0.002238718 0.002169498maximum numb
er of iterations reached 0.003606379 0.003540698maximum number of iterations reached 0.005869165 0.005697122maximum number o
f iterations reached 0.001750474 0.001730248maximum number of iterations reached 0.007842974 0.007577741maximum number of it
erations reached 0.006873472 0.006390644maximum number of iterations reached 0.004340938 0.00423222maximum number of iterati
ons reached 0.002996296 0.002948941maximum number of iterations reached 0.005673961 0.005533246maximum number of iterations
reached 0.009767259 0.009158023maximum number of iterations reached 0.000869069 0.000847017maximum number of iterations reac
hed 0.004392148 0.004262849maximum number of iterations reached 0.004524327 0.004413745maximum number of iterations reached
0.005625618 0.0054765maximum number of iterations reached 0.003797342 0.003623059maximum number of iterations reached 0.0035
67921 0.003507105maximum number of iterations reached 0.003466288 0.003392974maximum number of iterations reached 0.0047341
0.004678916maximum number of iterations reached 0.01074796 0.0100676maximum number of iterations reached 0.007983394 0.00750
154maximum number of iterations reached 0.007289754 0.007130557maximum number of iterations reached 0.006047187 0.005902362m
aximum number of iterations reached 0.003279246 0.003236741maximum number of iterations reached 0.008454731 0.007984947maxim
um number of iterations reached 0.0002879769 0.0002848626maximum number of iterations reached 0.009001487 0.008562333maximum
number of iterations reached 0.007013 0.006777847maximum number of iterations reached 0.007204201 0.007035538maximum number

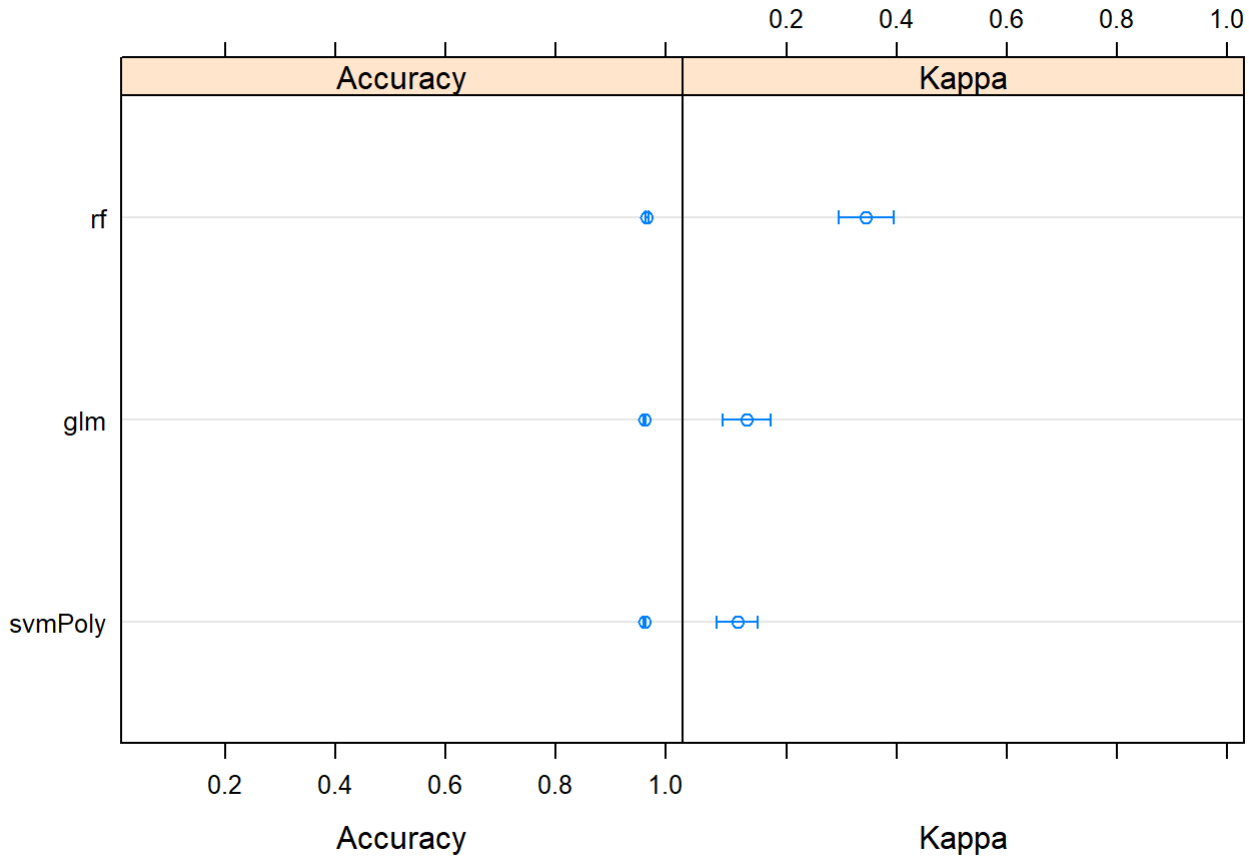
of iterations reached 0.003944891 0.003765467maximum number of iterations reached 0.006748169 0.006587513maximum number of i
terations reached 0.00449918 0.004434718maximum number of iterations reached 0.002992006 0.002959374maximum number of iterat
ions reached 0.009382282 0.00905086maximum number of iterations reached 0.006483126 0.005976964maximum number of iterations
reached 0.00606002 0.00589198maximum number of iterations reached 0.004186044 0.004055782maximum number of iterations reache
d 0.003764431 0.003692163maximum number of iterations reached 0.008743194 0.008344548maximum number of iterations reached 0.
001635954 0.001559492maximum number of iterations reached 0.003775858 0.003715822maximum number of iterations reached 0.0037
63575 0.003676693maximum number of iterations reached 0.003341016 0.003280379maximum number of iterations reached 0.00315170
7 0.002987786maximum number of iterations reached 0.00612804 0.005841632maximum number of iterations reached 0.002964701 0.0
02913587maximum number of iterations reached 0.003692681 0.003614555maximum number of iterations reached 0.008812191 0.00852
9364maximum number of iterations reached 0.007148605 0.006722723maximum number of iterations reached 0.004826209 0.004693915
maximum number of iterations reached 0.002049251 0.002020538maximum number of iterations reached 0.004460888 0.00436526maxim
um number of iterations reached 0.007447044 0.00719999maximum number of iterations reached 0.001330806 0.001295907maximum nu
mber of iterations reached 0.004499944 0.004414871maximum number of iterations reached 0.004408555 0.004282214maximum number
of iterations reached 0.001763541 0.001749692maximum number of iterations reached 0.001469525 0.001433032maximum number of i
terations reached 1.390471e-05 1.390198e-05maximum number of iterations reached 0.003049216 0.002992445maximum number of ite
rations reached 0.002773008 0.002737856maximum number of iterations reached 0.003265293 0.003206826maximum number of iterati
ons reached 0.008813664 0.008745318maximum number of iterations reached 0.006355991 0.006087174maximum number of iterations
reached 0.00453266 0.004437241maximum number of iterations reached 0.003027164 0.002971186maximum number of iterations reach
ed 0.003004536 0.00297479maximum number of iterations reached 0.01039497 0.009730407maximum number of iterations reached 0.0
003058092 0.000303059maximum number of iterations reached 0.002683509 0.002644807maximum number of iterations reached 0.0049
75851 0.00482157maximum number of iterations reached 0.001111551 0.001105992maximum number of iterations reached 0.003660012
0.003428322maximum number of iterations reached 0.004907275 0.004770178maximum number of iterations reached 0.002430958 0.00
2379459maximum number of iterations reached 0.0008288943 0.0008269206maximum number of iterations reached 0.007778755 0.0075
65258maximum number of iterations reached 0.007436513 0.007025775maximum number of iterations reached 0.002464178 0.00243891
6maximum number of iterations reached 0.001957702 0.001945606maximum number of iterations reached 0.003443314 0.003387241max
imum number of iterations reached 0.009328439 0.008838778maximum number of iterations reached 0.007332777 0.007043303maximum
number of iterations reached 0.003014997 0.002971574maximum number of iterations reached 0.002347648 0.002324876maximum numb
er of iterations reached 0.003003771 0.002941406maximum number of iterations reached 0.007822067 0.007479511maximum number o
f iterations reached 0.002247459 0.002167321maximum number of iterations reached 0.001596807 0.0015882maximum number of iter
ations reached 0.00404082 0.003975567maximum number of iterations reached 0.004406322 0.004342469maximum number of iteration
s reached 0.01212159 0.01149102maximum number of iterations reached 0.003456912 0.003278187maximum number of iterations reac
hed 0.004793862 0.004699012maximum number of iterations reached 0.003013879 0.002955842maximum number of iterations reached
0.001594912 0.001577482maximum number of iterations reached 0.008394836 0.007987189maximum number of iterations reached 0.00
06157777 0.0006029889maximum number of iterations reached 0.004633261 0.004507831maximum number of iterations reached 0.0041
19015 0.004055224maximum number of iterations reached 0.001810198 0.001795755maximum number of iterations reached 0.00397253
7 0.003799667maximum number of iterations reached 0.00333588 0.003274272maximum number of iterations reached 0.002296159 0.0
02272182maximum number of iterations reached 0.001925036 0.00191398maximum number of iterations reached 0.009373285 0.009257
046maximum number of iterations reached 0.005986239 0.005737263maximum number of iterations reached 0.003935498 0.003862316m
aximum number of iterations reached 0.00551955 0.005321314maximum number of iterations reached 0.003363283 0.00331702maximum
number of iterations reached 0.006958098 0.006699827maximum number of iterations reached 0.001411191 0.001379378maximum numb
er of iterations reached 0.003340372 0.003276417maximum number of iterations reached 0.00328007 0.003215035maximum number of

```
iterations reached 0.002082675 0.002063586maximum number of iterations reached 0.003000339 0.002895222maximum number of iter
ations reached 0.003959262 0.003887532maximum number of iterations reached 0.004451806 0.004345843
```

```
set.seed(123)
results_wine2 <- resamples(models_wine2)
summary(results_wine2)
```

```
##
## Call:
## summary.resamples(object = results_wine2)
##
## Models: rf, svmPoly, glm
## Number of resamples: 30
##
## Accuracy
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rf      0.9561404 0.9600015 0.9664254 0.9660639 0.9708455 0.9738372    0
## svmPoly 0.9475219 0.9591837 0.9619883 0.9614006 0.9642365 0.9707602    0
## glm     0.9533528 0.9591837 0.9619883 0.9615921 0.9649123 0.9680233    0
##
## Kappa
##                 Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rf       0.104750305 0.2649746 0.3434892 0.3437564 0.4609457 0.5315904    0
## svmPoly -0.005471956 0.0000000 0.1285910 0.1095305 0.1381772 0.3659622    0
## glm     -0.010309278 0.0000000 0.1285910 0.1269126 0.2200614 0.3435115    0
```

```
dotplot(results_wine2)
```




```
modelCor(results_wine2)
```

```
##               rf   svmPoly       glm
## rf      1.0000000 0.4938575 0.5323977
## svmPoly 0.4938575 1.0000000 0.5049027
## glm     0.5323977 0.5049027 1.0000000
```

```
stackControl <- trainControl(method="repeatedcv", number=10, repeats=5, savePredictions=TRUE, classProbs=TRUE)
set.seed(123)
winestack.rf <- caretStack(models_wine2, method="rf", metric="Accuracy", trControl=stackControl)
```

```
## note: only 2 unique complexity parameters in default grid. Truncating the grid to 2 .
```

```
print(winestack.rf)  ##ensemble model
```

```
## A rf ensemble of 3 base models: rf, svmPoly, glm
##
## Ensemble results:
## Random Forest
##
## 10284 samples
##     3 predictor
##     2 classes: 'high', 'low'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 9255, 9255, 9256, 9256, 9255, 9257, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.9659667  0.3782592
##   3     0.9657335  0.3837443
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
winestack.pred <- predict(winestack.rf , q6test)
cf_ensemble_wine <- confusionMatrix(as.factor(winestack.pred), as.factor(q6testlabel) , positive="high", mode = "everything"
)
print(cf_ensemble_wine)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction high  low
##       high 1414   39
##       low     9    8
##
##                Accuracy : 0.9673
##                  95% CI : (0.9569, 0.9758)
##     No Information Rate : 0.968
##     P-Value [Acc > NIR] : 0.5965
##
##                   Kappa : 0.237
##
##  Mcnemar's Test P-Value : 2.842e-05
##
##             Sensitivity : 0.9937
##             Specificity : 0.1702
##          Pos Pred Value : 0.9732
##          Neg Pred Value : 0.4706
##               Precision : 0.9732
##                  Recall : 0.9937
##                      F1 : 0.9833
##              Prevalence : 0.9680
##          Detection Rate : 0.9619
##    Detection Prevalence : 0.9884
##       Balanced Accuracy : 0.5819
##
##        'Positive' Class : high
##
```

Question 7

1.Use the USDA dataset. Consider calories less than 200 as low, and 200 and more as high.

```
USDA<-read.csv(file.choose())
caloriefactor<- function(x){ ifelse(USDA$Calories<200,"low","high")
}

USDA$Calories<-caloriefactor(USDA)
```

2.Build individual classifiers of random forest, support vector machines, Naive Bayes and logistic regression. Consider you are interested in whether the model correctly predicts high calories. Calculate performance measures and decide which is the best model. Use 70:30 training:test ratio and seed of 123. Use 3 repeats of 5-fold cross-validation.

```
#Below are the performance measures for the models.
#Based on the results of the models, Logistic regression is the best algorithm to classify high versus low calories. It has
 an accuracy of 98.78% with a confidence interval between 0.9817-0.9923, precision of 98.47%, recall of 99% and an f1 score
 of 98.74%. It may seem as if the model is overfitting but these metrics are the results of feeding the model with test dat
a. This means the Logistic regression model is performing very well.
```

Creating Training/Test set

```
set.seed(123)
q7usda_train <- sample(nrow(USDA), floor(nrow(USDA)*0.7))
q7train <- USDA[q7usda_train,]#training set with labels
q7test <- USDA[-q7usda_train,]#test set with labels
q7trainlabel<-q7train[,1]
q7testlabel<-q7test[,1]
q7test<-q7test[,-1]#test set no labels
```

Random Forest

```
ctrl <- trainControl(method="repeatedcv", number=5, repeats=3)  #cross-validation
set.seed(123)

q7RFmodel_usda <- train(Calories ~ ., data= q7train, method="rf", ntree=500, trControl = ctrl)
testpredRF_usda <- predict(q7RFmodel_usda, q7test)#takes the model and the test data
cf_RF_usda <- confusionMatrix(as.factor(testpredRF_usda), as.factor(q7testlabel),mode = "everything")
print(cf_RF_usda)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction high low
##       high  894  22
##       low    15 961
##
##                Accuracy : 0.9804
##                  95% CI : (0.9731, 0.9862)
##     No Information Rate : 0.5196
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9608
##
##  Mcnemar's Test P-Value : 0.3239
##
##             Sensitivity : 0.9835
##             Specificity : 0.9776
##          Pos Pred Value : 0.9760
##          Neg Pred Value : 0.9846
##               Precision : 0.9760
##                  Recall : 0.9835
##                      F1 : 0.9797
##              Prevalence : 0.4804
##          Detection Rate : 0.4725
##    Detection Prevalence : 0.4841
##       Balanced Accuracy : 0.9806
##
##        'Positive' Class : high
##
```

Support Vector Machine

```
set.seed(123)
q7SVMmodel_usda <- train(Calories ~ ., data= q7train, method="svmPoly", trControl = ctrl)
test_predSVM_usda <- predict(q7SVMmodel_usda, q7test)
cf_SVM_usda <- confusionMatrix(as.factor(test_predSVM_usda), as.factor(q7testlabel), mode = "everything")
print(cf_SVM_usda)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction high low
##       high  895  13
##       low    14 970
##
##                Accuracy : 0.9857
##                  95% CI : (0.9793, 0.9906)
##     No Information Rate : 0.5196
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9714
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9846
##             Specificity : 0.9868
##          Pos Pred Value : 0.9857
##          Neg Pred Value : 0.9858
##               Precision : 0.9857
##                  Recall : 0.9846
##                      F1 : 0.9851
##              Prevalence : 0.4804
##          Detection Rate : 0.4730
##    Detection Prevalence : 0.4799
##       Balanced Accuracy : 0.9857
##
##        'Positive' Class : high
##
```

Naive Bayes

```
set.seed(123)
q7NBmodel_usda <- train(Calories ~ ., data= q7train, method="naive_bayes", trControl = ctrl)
test_predNB_usda <- predict(q7NBmodel_usda, q7test)
cf_NB_usda <- confusionMatrix(as.factor(test_predNB_usda), as.factor(q7testlabel),mode = "everything")
print(cf_NB_usda)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction high low
##       high  731  16
##       low   178 967
##
##                Accuracy : 0.8975
##                  95% CI : (0.8829, 0.9108)
##     No Information Rate : 0.5196
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7932
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8042
##             Specificity : 0.9837
##          Pos Pred Value : 0.9786
##          Neg Pred Value : 0.8445
##               Precision : 0.9786
##                  Recall : 0.8042
##                      F1 : 0.8829
##              Prevalence : 0.4804
##          Detection Rate : 0.3864
##    Detection Prevalence : 0.3948
##       Balanced Accuracy : 0.8940
##
##        'Positive' Class : high
##
```

Logistic Regression

```
set.seed(123)
LRmodel_usda <- train(Calories ~ ., data= q7train, method="glm", trControl = ctrl)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
test_predLR_usda <- predict(LRmodel_usda, q7test)
cf_LR_usda <- confusionMatrix(as.factor(test_predLR_usda), as.factor(q7testlabel),mode = "everything")
print(cf_LR_usda)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction high low
##       high  900  14
##       low     9 969
##
##                Accuracy : 0.9878
##                  95% CI : (0.9818, 0.9923)
##     No Information Rate : 0.5196
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9757
##
##  Mcnemar's Test P-Value : 0.4042
##
##             Sensitivity : 0.9901
##             Specificity : 0.9858
##          Pos Pred Value : 0.9847
##          Neg Pred Value : 0.9908
##               Precision : 0.9847
##                  Recall : 0.9901
##                      F1 : 0.9874
##              Prevalence : 0.4804
##          Detection Rate : 0.4757
##    Detection Prevalence : 0.4831
##       Balanced Accuracy : 0.9879
##
##        'Positive' Class : high
##
```

3. Build an ensemble model of all the previous models and calculate and interpret its performance measures.

```
#Below are the ensemble models. There were 2 ensemble models that were ran. After running the first one, it was shown that t
he models Support Vector Machines and Logistic Regression had a high correlation at 94%.
#SVM has lower accuracy than Logistic regression (SVM accuracy=98.17%, GLM accuracy=98.4%) and therefore another ensemble mo
del was created without it
#The second Ensemble model contained NB, RF and GLM. It had an accuracy score of 98.9% which is only slightly higher than th
e GLM model by itself (0.5% higher). The ensemble model performed better in predicting high and low calories than any single
model alone.
```

Ensemble Model

```
library(caretEnsemble)
control2 <- trainControl(method="repeatedcv", number = 5, repeats=3, savePredictions="final", classProbs = TRUE)
algorithmList <- c('rf', 'svmPoly', 'glm', 'naive_bayes')
set.seed(123)
models_usda <- caretList(Calories~., data=q7train, trControl=control2, methodList=algorithmList)
```

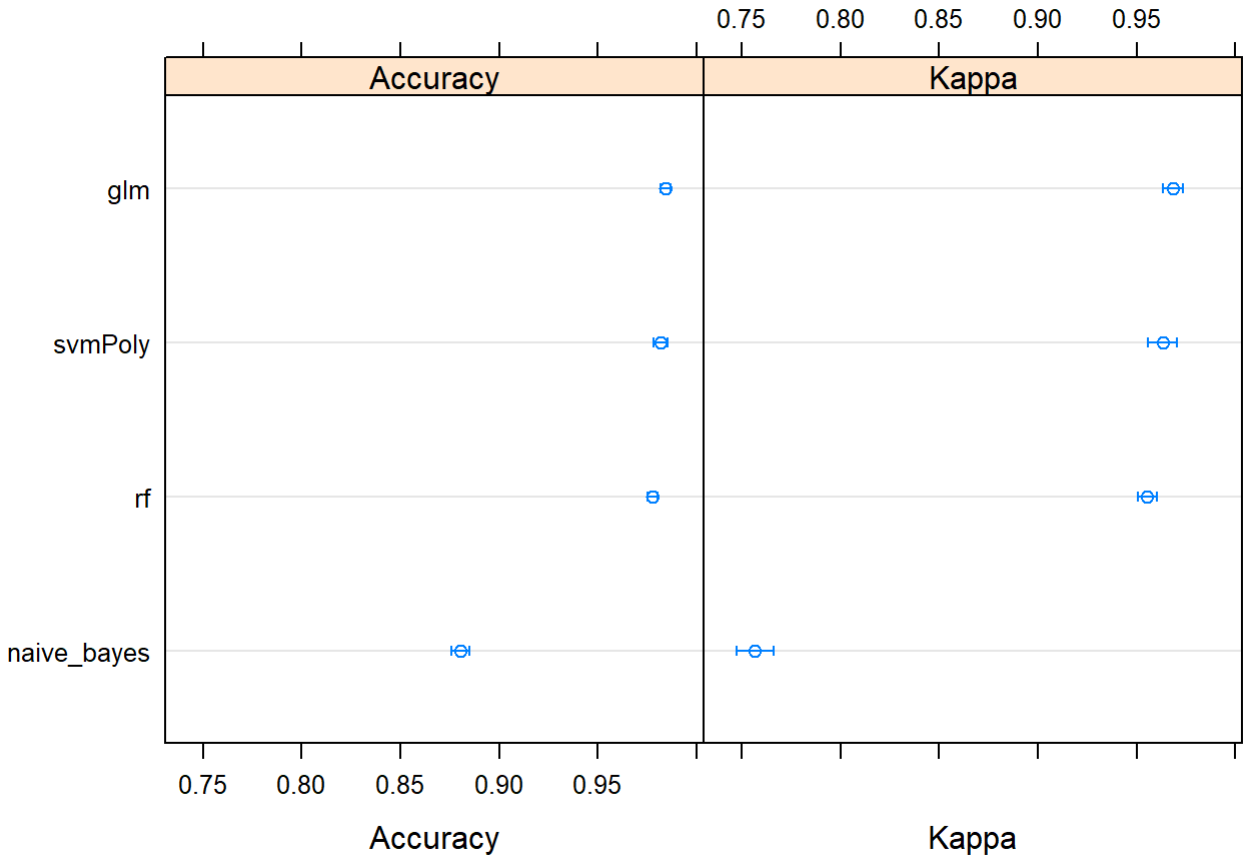
```
## Warning in trControlCheck(x = trControl, y = target): indexes not defined in
## trControl. Attempting to set them ourselves, so each model in the ensemble will
## have the same resampling indexes.
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
set.seed(123)
results_usda <- resamples(models_usda)
summary(results_usda)
```

```
##
## Call:
## summary.resamples(object = results_usda)
##
## Models: rf, svmPoly, glm, naive_bayes
## Number of resamples: 15
##
## Accuracy
##                  Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rf          0.9705549 0.9750567 0.9784824 0.9777932 0.9813137 0.9841270    0
## svmPoly     0.9660249 0.9779034 0.9829932 0.9817209 0.9852775 0.9909400    0
## glm         0.9728199 0.9829932 0.9841270 0.9842890 0.9869762 0.9909400    0
## naive_bayes 0.8673469 0.8742922 0.8788222 0.8804292 0.8872524 0.8924122    0
##
## Kappa
##                  Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rf          0.9408637 0.9498507 0.9568047 0.9554128 0.9624742 0.9681261    0
## svmPoly     0.9317658 0.9556419 0.9658540 0.9632861 0.9704360 0.9817987    0
## glm         0.9453961 0.9658231 0.9681064 0.9684371 0.9738458 0.9817932    0
## naive_bayes 0.7298601 0.7440736 0.7535225 0.7568103 0.7709973 0.7814339    0
```

```
dotplot(results_usda)
```




```
modelCor(results_usda)
```

```
##                    rf    svmPoly        glm naive_bayes
## rf          1.0000000  0.1258074  0.2281525   0.2883902
## svmPoly     0.1258074  1.0000000  0.9383481  -0.3808597
## glm         0.2281525  0.9383481  1.0000000  -0.2446110
## naive_bayes 0.2883902 -0.3808597 -0.2446110   1.0000000
```

```
library(caretEnsemble)
control2 <- trainControl(method="repeatedcv", number = 5, repeats=3, savePredictions="final",classProbs = TRUE)
algorithmListq7 <- c('rf', 'glm', 'naive_bayes')
set.seed(123)
models_usda2 <- caretList(Calories~., data=q7train, trControl=control2, methodList=algorithmListq7)
```

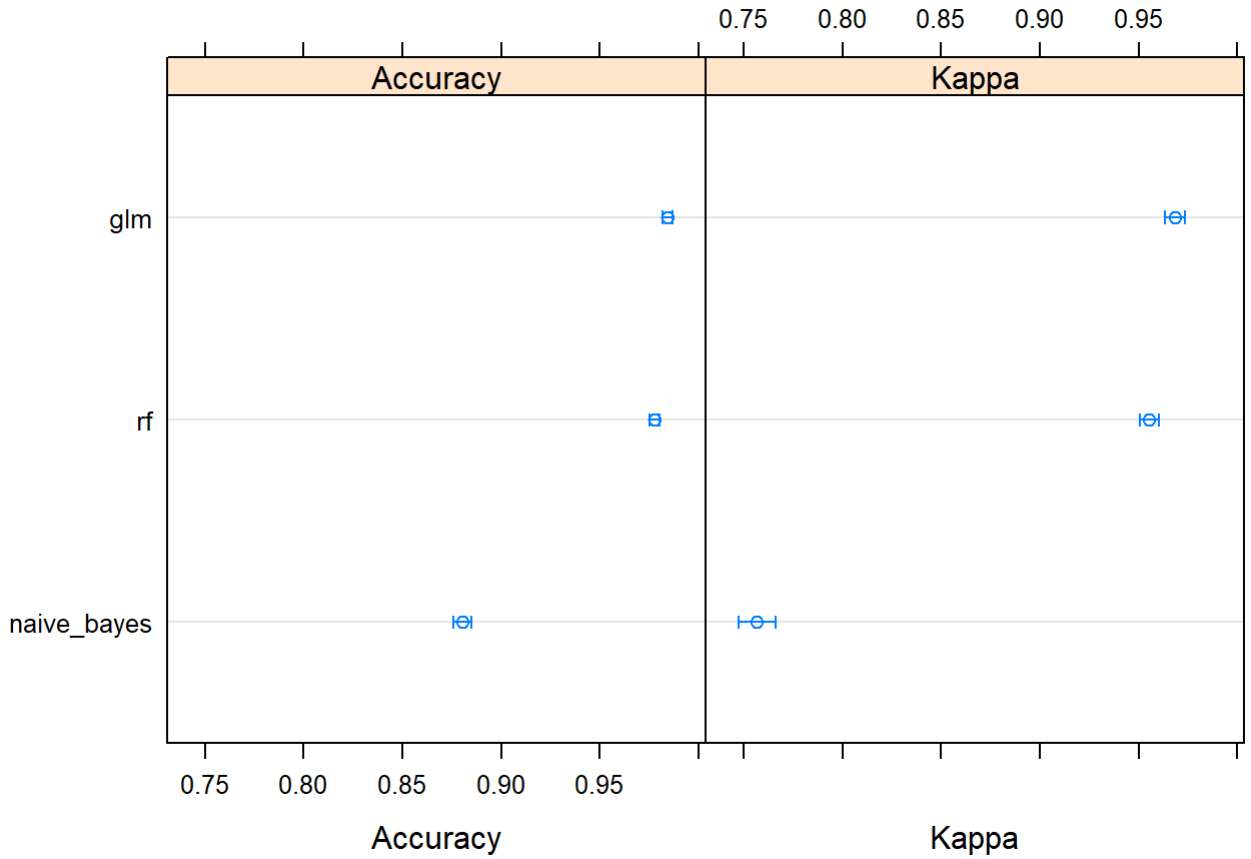
```
## Warning in trControlCheck(x = trControl, y = target): indexes not defined in
## trControl. Attempting to set them ourselves, so each model in the ensemble will
## have the same resampling indexes.
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
set.seed(123)
results_usda2 <- resamples(models_usda2)
summary(results_usda2)
```

```
##
## Call:
## summary.resamples(object = results_usda2)
##
## Models: rf, glm, naive_bayes
## Number of resamples: 15
##
## Accuracy
##                  Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rf          0.9705549 0.9750567 0.9784824 0.9777932 0.9813137 0.9841270    0
## glm         0.9728199 0.9829932 0.9841270 0.9842890 0.9869762 0.9909400    0
## naive_bayes 0.8673469 0.8742922 0.8788222 0.8804292 0.8872524 0.8924122    0
##
## Kappa
##                  Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rf          0.9408637 0.9498507 0.9568047 0.9554128 0.9624742 0.9681261    0
## glm         0.9453961 0.9658231 0.9681064 0.9684371 0.9738458 0.9817932    0
## naive_bayes 0.7298601 0.7440736 0.7535225 0.7568103 0.7709973 0.7814339    0
```

```
dotplot(results_usda2)
```




```
modelCor(results_usda2)
```

```
##                    rf        glm naive_bayes
## rf          1.0000000  0.2281525   0.2883902
## glm         0.2281525  1.0000000  -0.2446110
## naive_bayes 0.2883902 -0.2446110   1.0000000
```

```
stackControl2 <- trainControl(method="repeatedcv", number=10, repeats=3, savePredictions=TRUE, classProbs=TRUE)
set.seed(123)
usdastack.rf <- caretStack(models_usda2, method="rf", metric="Accuracy", trControl=stackControl2)
```

```
## note: only 2 unique complexity parameters in default grid. Truncating the grid to 2 .
```

```
print(usdastack.rf)  ##ensemble model
```

```
## A rf ensemble of 3 base models: rf, glm, naive_bayes
##
## Ensemble results:
## Random Forest
##
## 13239 samples
##     3 predictor
##     2 classes: 'high', 'low'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 11915, 11914, 11915, 11916, 11915, 11916, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.9872850  0.9744549
##   3     0.9865801  0.9730390
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
stack.pred2 <- predict(usdastack.rf , q7test)
cf_ensemble_usda <- confusionMatrix(as.factor(stack.pred2), as.factor(q7testlabel) , positive="high", mode = "everything")
print(cf_ensemble_usda)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction high low
##       high  901  13
##       low     8 970
##
##                Accuracy : 0.9889
##                  95% CI : (0.9831, 0.9931)
##     No Information Rate : 0.5196
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9778
##
##  Mcnemar's Test P-Value : 0.3827
##
##             Sensitivity : 0.9912
##             Specificity : 0.9868
##          Pos Pred Value : 0.9858
##          Neg Pred Value : 0.9918
##               Precision : 0.9858
##                  Recall : 0.9912
##                      F1 : 0.9885
##              Prevalence : 0.4804
##          Detection Rate : 0.4762
##    Detection Prevalence : 0.4831
##       Balanced Accuracy : 0.9890
##
##        'Positive' Class : high
##
```

Question 8 (Kmeans clustering)

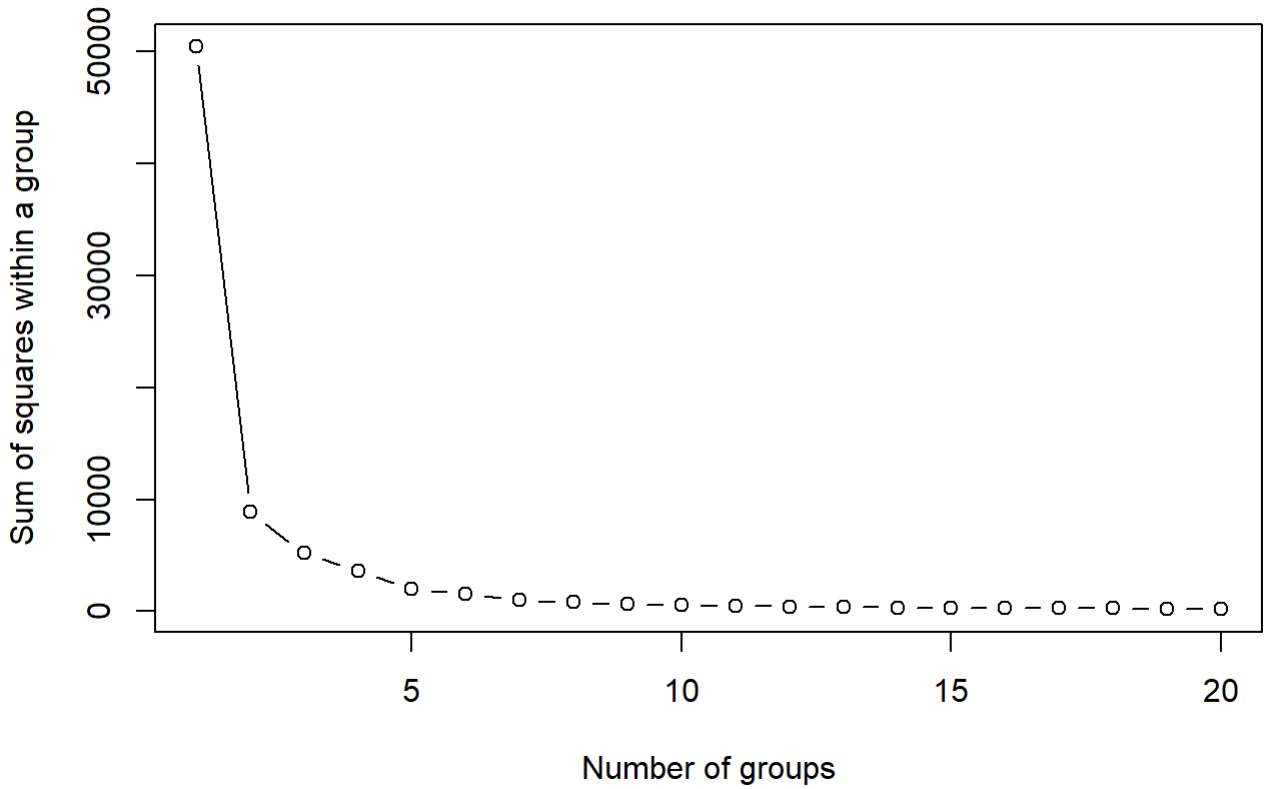
1.Apply a kmeans clustering to the geyser dataset “faithful” embedded in R. What is the best value of k to cluster this dataset? Interpret how did you determine this value? What is the compactness of the kmeans clustering?

```
#The best value of k for this dataset was found to be three as the largest decrease of the within sum of squares was seen to be between 2 and 3. The within sums of squares is a metric that shows how dissimilar are the members of a cluster are. The more dissimilar they are, the less compact they are. Kmeans clustering aims to minimize the differences within the cluster and maximize the differences between the clusters.The compactness of the clusters is 89.7; this means the members of a cluster are 89.7% similar to each other.


data<-faithful

kmeans(data, centers = 3, nstart = 20)
```

```
## K-means clustering with 3 clusters of sizes 94, 84, 94
##
## Cluster means:
##   eruptions  waiting
## 1  4.131351 75.21277
## 2  4.369012 84.91667
## 3  2.056734 54.05319
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   1   3   1   3   2   3   2   2   3   2   3   2   1   3   2   3   3   2   3   1
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##   3   3   1   1   1   2   3   1   1   1   1   1   1   1   1   3   3   1   3   2
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##   1   3   2   3   1   2   3   3   2   3   1   2   3   1   3   2   1   3   1   2
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
##   3   2   3   2   3   2   1   1   1   1   2   3   1   1   3   1   3   1   1   2
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##   1   2   1   1   1   2   1   1   3   2   3   2   3   1   3   1   2   1   3   2
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
##   3   2   3   2   2   3   2   3   2   2   1   3   2   1   3   2   3   2   3   2
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
##   3   1   1   3   2   2   3   2   3   2   3   2   3   2   3   2   3   2   3   1
## 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160
##   2   3   2   1   1   3   1   3   2   3   1   1   1   2   1   1   2   2   3   2
## 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
##   3   2   3   1   1   1   3   2   3   2   3   3   1   1   2   2   1   3   2   1
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200
##   3   1   2   2   3   1   2   3   2   3   2   3   1   2   1   2   2   1   3   1
## 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220
##   3   2   2   3   1   3   1   2   3   2   1   1   3   1   3   1   3   2   3   1
## 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240
##   3   2   3   1   1   1   1   1   1   1   1   3   2   3   2   3   3   1   1   3
## 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260
##   1   3   2   3   2   2   3   2   1   1   3   2   1   1   2   1   1   2   3   1
## 261 262 263 264 265 266 267 268 269 270 271 272
##   1   2   3   2   3   3   1   2   3   2   3   1
##
## Within cluster sum of squares by cluster:
## [1] 1450.793 1074.427 2688.048
```

```
##  (between_SS / total_SS =  89.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

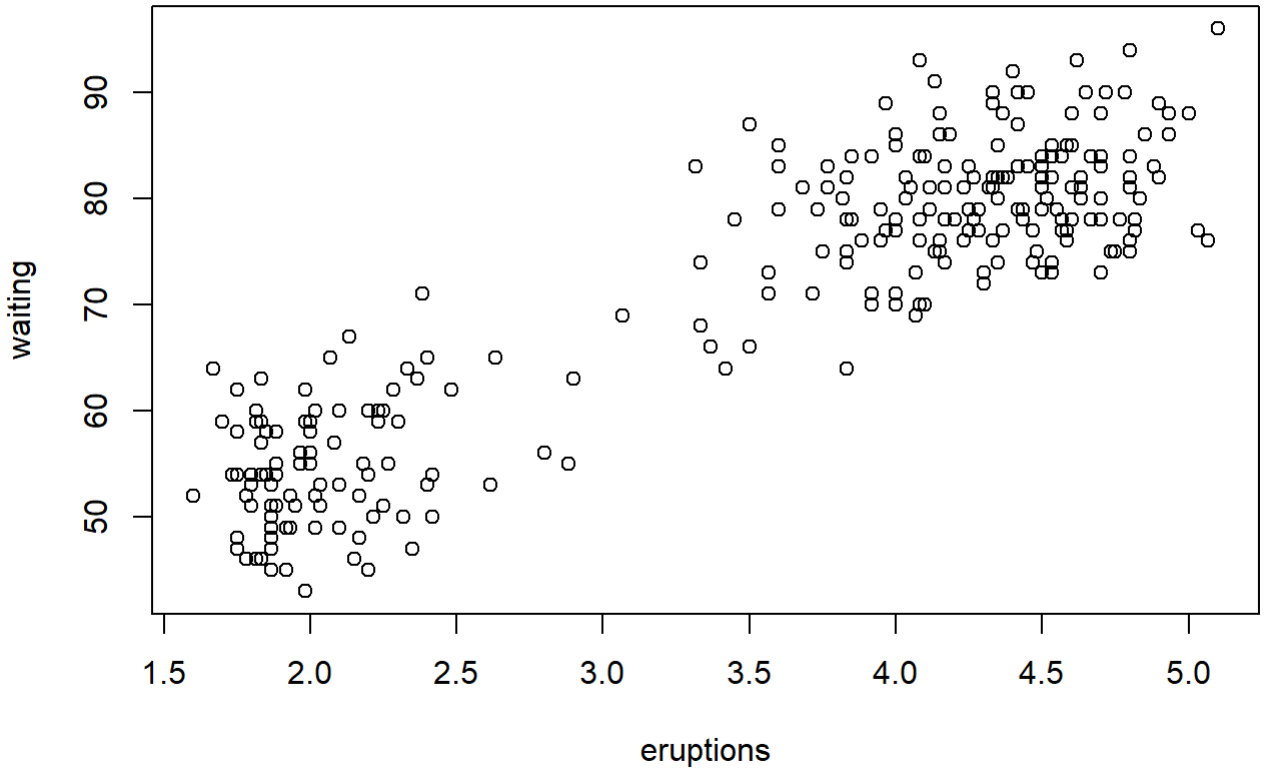
```
wssplot <- function(data, nc, seed=123){
               wss <- (nrow(data)-1)*sum(apply(data,2,var))
               for (i in 2:nc){  #at least 2 clusters
                    set.seed(seed)
                    wss[i] <- sum(kmeans(data, centers=i)$withinss)}
                plot(1:nc, wss, type="b", xlab="Number of groups",
                     ylab="Sum of squares within a group")
                }


wssplot(data, nc = 20)
```



2. Plot the original faithful

dataset (eruptions on the x-axis vs waiting on the y-axis). From the plot, can you explain why did you get the best value of K that you got in question 1?

```
plot(data)
```




```
# after plotting the data, it can be seen that the data can be clustered into 2 large groups and upon closer inspection, the largest group can infact be divided into 2 clusters for a total of 3. This explains why the best k value was found to be 3.
```

3. Validate your clustering and interpret the results of your validation

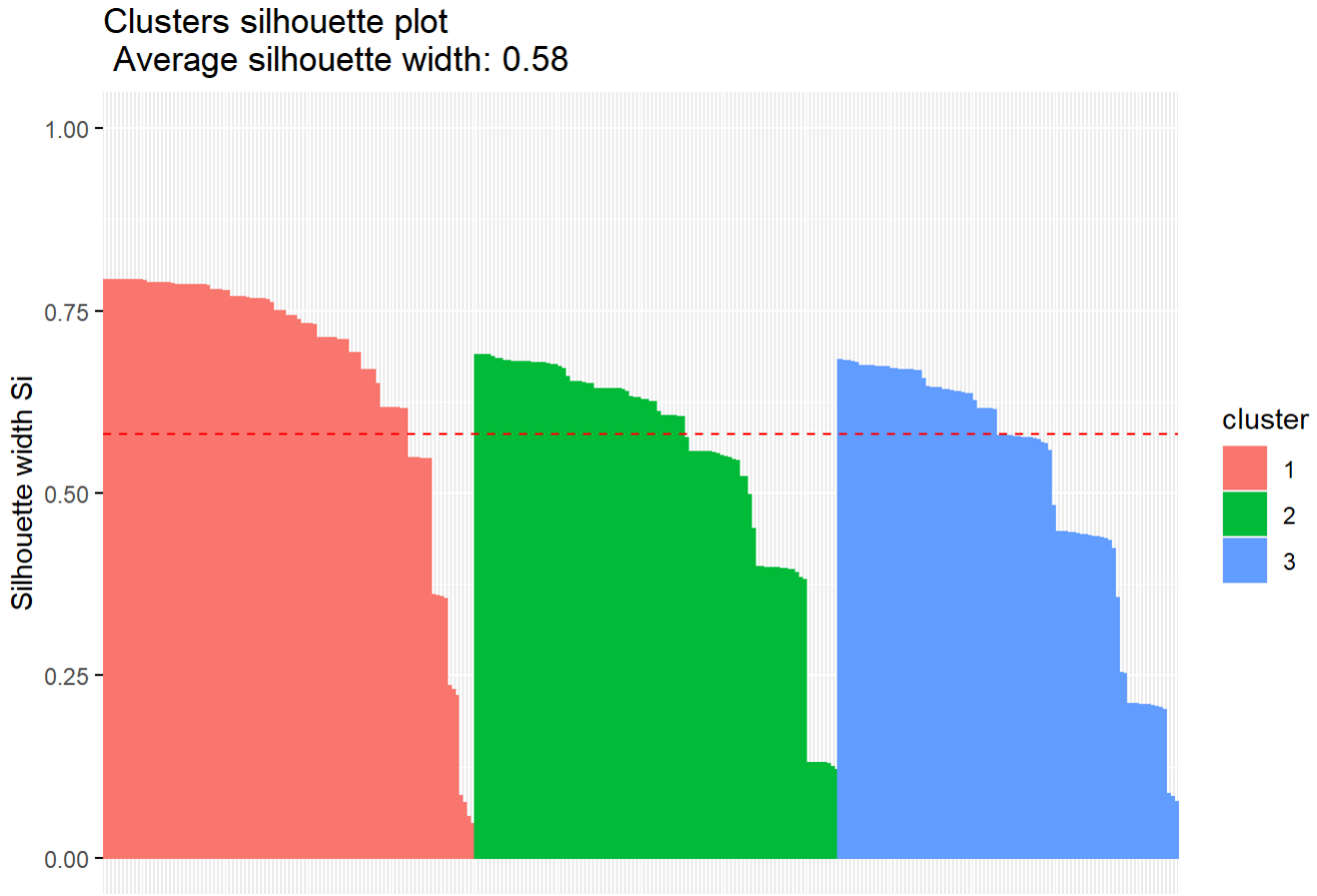
```
datacluster<-kmeans(data, centers = 3, nstart = 20)
clusterlabels<-table(data$eruptions, datacluster$cluster)
library(cluster)
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 4.0.3
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
sil <- silhouette(datacluster$cluster, dist(data))
fviz_silhouette(sil)
```

```
##   cluster size ave.sil.width
## 1       1   94          0.67
## 2       2   92          0.55
## 3       3   86          0.52
```



```
# This dataset has 2 columns eruptions and waiting. Both are measured in minutes and provide information on the duration of
 both events respectively. Therefore no legible label for each cluster exists. Based on the visualization below, we can see
 that the third  cluster has the best silhouette width.  The closer the Si coefficient is to one, the more intracluster simi
larity.There are some points negative silhouette widths
```

4. Find trends in the dataset. Based on these trends, which varibale (eruptions or waiting) is a better variable to divide the dataset into categories? What will be the cutpoints at which the clusters are drawn? Interpret you answer.

```
library(GGally)   ## extension to ggplot
```

```
## Warning: package 'GGally' was built under R version 4.0.3
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
#install.packages("plotly")
library(plotly)
```

```
## Warning: package 'plotly' was built under R version 4.0.3
```

```
##
## Attaching package: 'plotly'
```

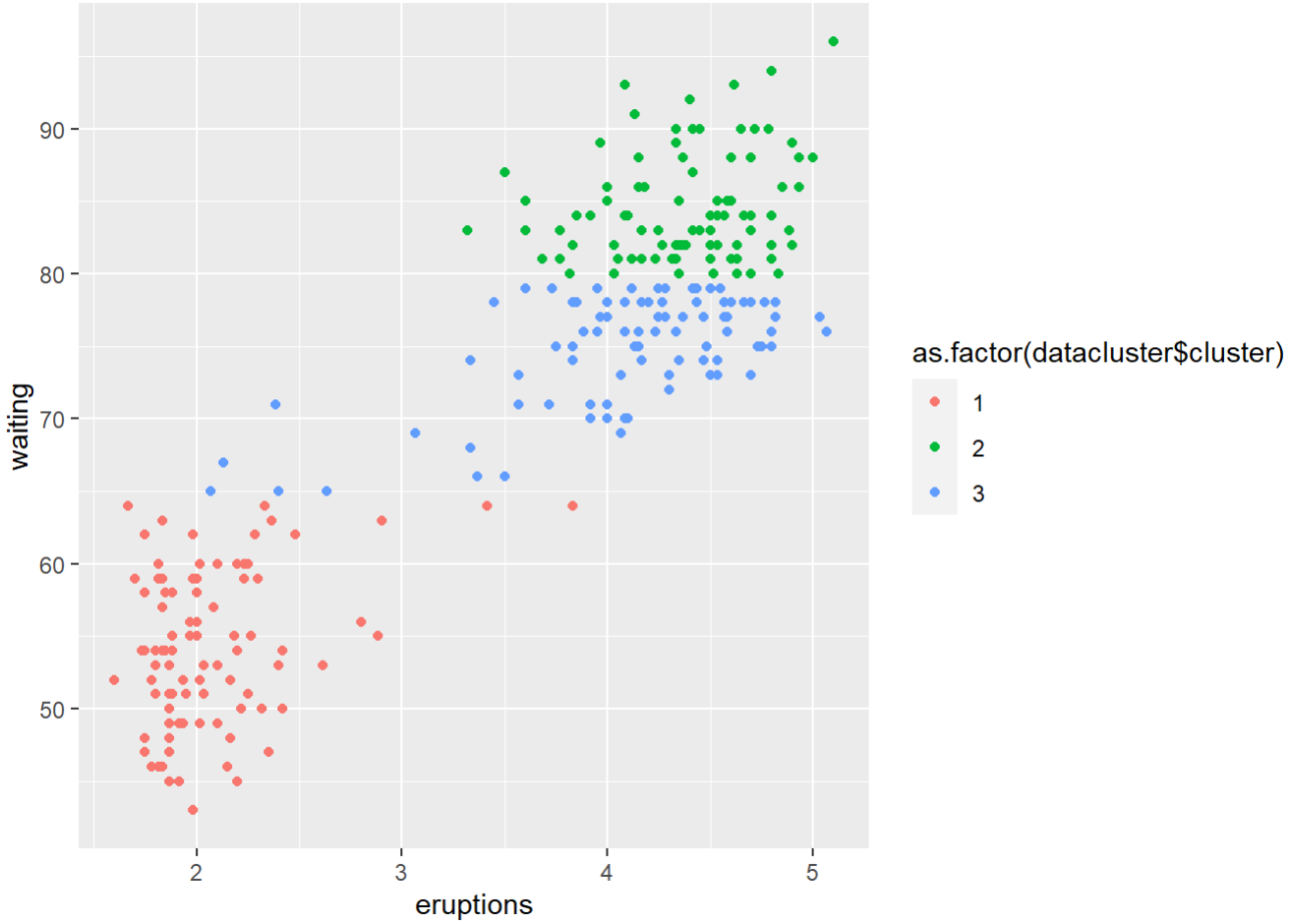
```
## The following object is masked from 'package:MASS':
##
##     select
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

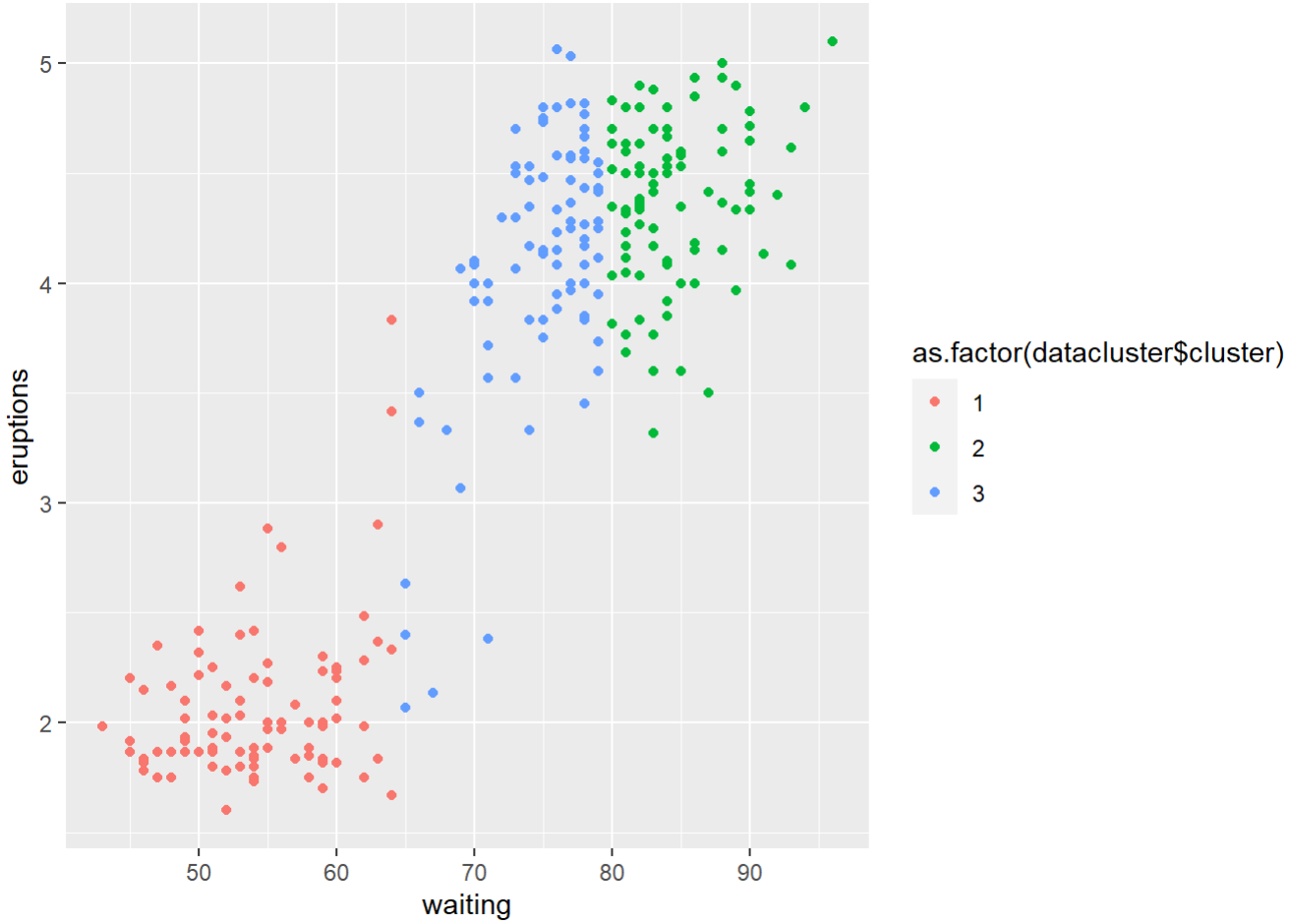
```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
#adding the clusters to the faithful dataset
data$cluster <- as.factor(datacluster$cluster)

ggplot(data, aes(x=eruptions, y = waiting, color = as.factor(datacluster$cluster))) + geom_point()
```




```
ggplot(data, aes(x=waiting, y = eruptions, color = as.factor(datacluster$cluster))) + geom_point()
```

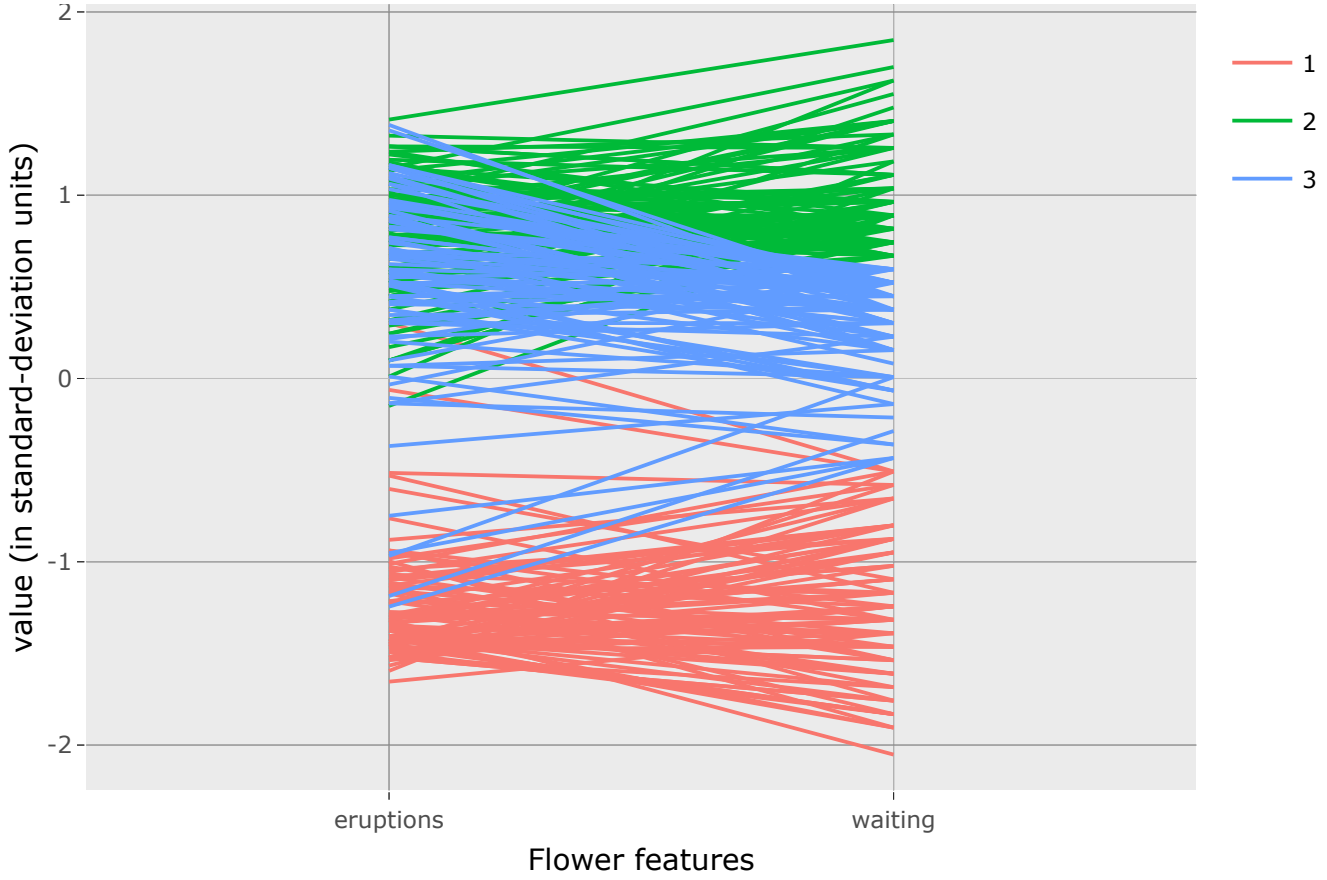



```
p <- ggparcoord(data = data, columns = c(1:2), groupColumn = "cluster", scale = "std") + labs(x = "Flower features", y = "va
lue (in standard-deviation units)", title = "Clustering")
ggplotly(p)
```

```
## Warning: `group_by_()` is deprecated as of dplyr 0.7.0.
## Please use `group_by()` instead.
## See vignette('programming') for more help
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

Clustering

cluster




```
#Based on the scatterplots, it is better to divide the dataset by waiting time
#There is a general trend showing that the longer the waiting time, the more eruptions that take place. there are outlier pr
esent but this is the general trend.
#Some of the cutpoints to appropriately determine clusters are waiting times of  65 and 80 minutes.
```

Question 9 (Text analytics)

For this question, we will do text and sentiment analysis of Martin Luther King's speech "I have a dream". The speech is available on Quercus in file "Dream_Speech.docx"

1. Draw 20 most frequent words in Martin Luther King's speech "I have a dream" and show their counts?

```
library(tidytext)
```

```
## Warning: package 'tidytext' was built under R version 4.0.3
```

```
library(sentimentr)
```

```
## Warning: package 'sentimentr' was built under R version 4.0.3
```

```
##
## Attaching package: 'sentimentr'
```

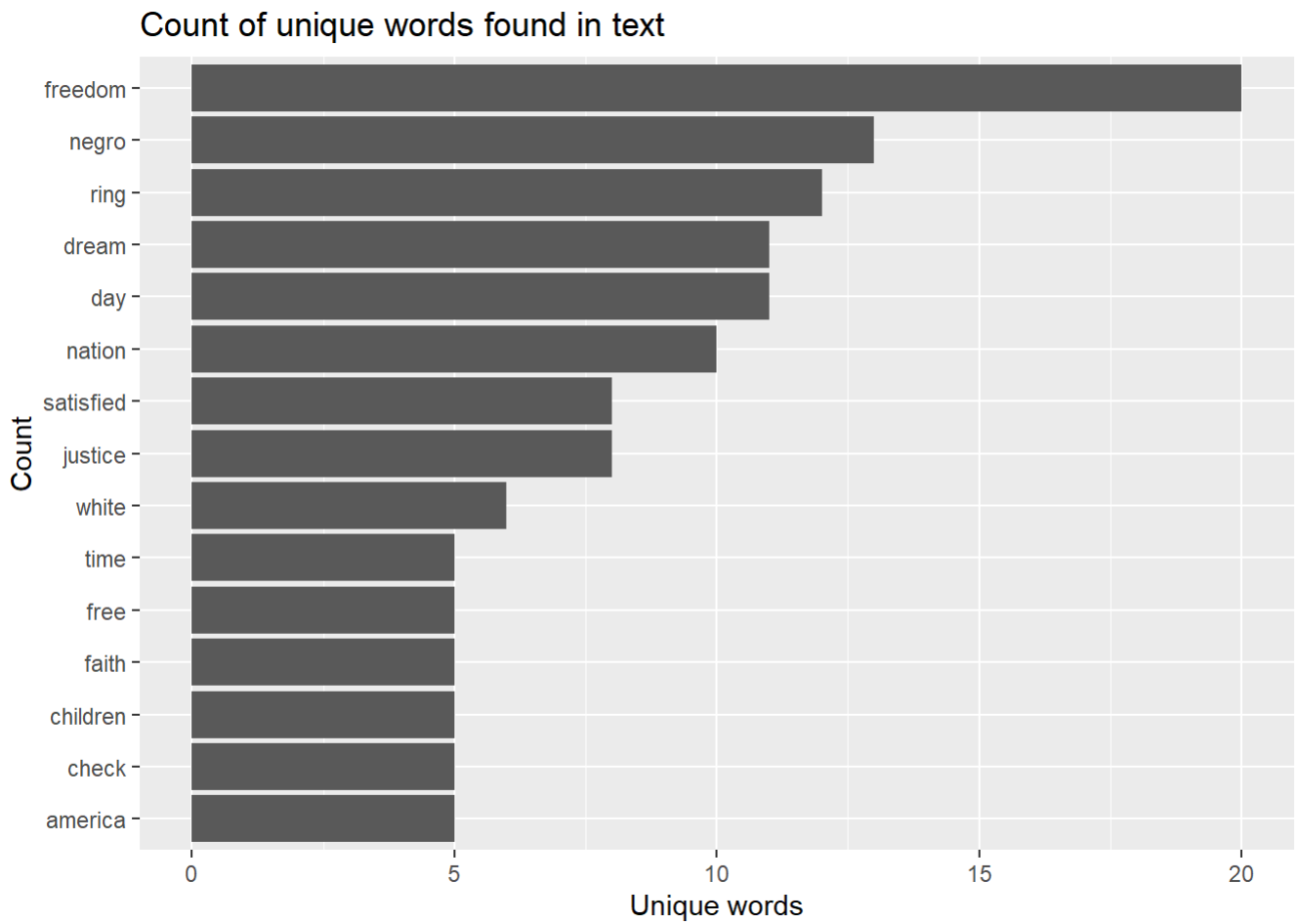
```
## The following object is masked from 'package:plotly':
##
##     highlight
```

```
library(readtext)
```

```
## Warning: package 'readtext' was built under R version 4.0.3
```

```
library(tidyverse)
library(stringr)
dream<-readtext(paste0(file.choose()))
tidydream <- unnest_tokens(dream,word,text)
tidyfiltered1 <- filter(tidydream,!tidydream$word %in% stop_words$word)
tidyfiltered2 <- filter(tidyfiltered1,!tidyfiltered1$word %in% stop_words$word)

library(ggplot2)
tidyfiltered2 %>%
  count(word, sort = TRUE) %>%
  top_n(15) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(x = word, y = n)) +
  geom_col() +
  coord_flip() +
  labs(x = "Count",
       y = "Unique words",
       title = "Count of unique words found in text")
```

```
## Selecting by n
```



2. What are the 20 most common bigrams. Show their counts.

```
dreambigrams <- dream %>%
    unnest_tokens(bigram, text, token = "ngrams", n = 2) %>%
    separate(bigram, c("word1", "word2"), sep = " ") %>%
    filter(!word1 %in% stop_words$word) %>%
    filter(!word2 %in% stop_words$word) %>%
    na.omit() %>%
    count(word1, word2, sort = TRUE)
head(dreambigrams,20L)
```

	word1 <chr>	word2 <chr>	n <int>
1	freedom	ring	10
2	god’s	children	3
3	insufficient	funds	2
4	join	hands	2
5	police	brutality	2
6	promissory	note	2
7	american	dream	1
8	american	society	1
9	bad	check	1
10	basic	mobility	1

1-10 of 20 rows Previous **1** 2 Next

3. Do a sentiment analysis of the speech using a chart of nrc lexicon.

```
#install.packages("readtext")
library(readtext)
library(tidyverse)
library(tidytext)
library(stringr)


dreamtokens <- tidyfiltered2

dreamtokens %>%
  inner_join(get_sentiments("nrc")) %>% # pull out only sentiment words
  count(sentiment) %>% # count the # of positive & negative words
  spread(sentiment, n, fill = 0) %>% # made data wide rather than narrow
  mutate(sentiment = positive - negative) # # of positive words - # of negative words
```

```
## Joining, by = "word"
```

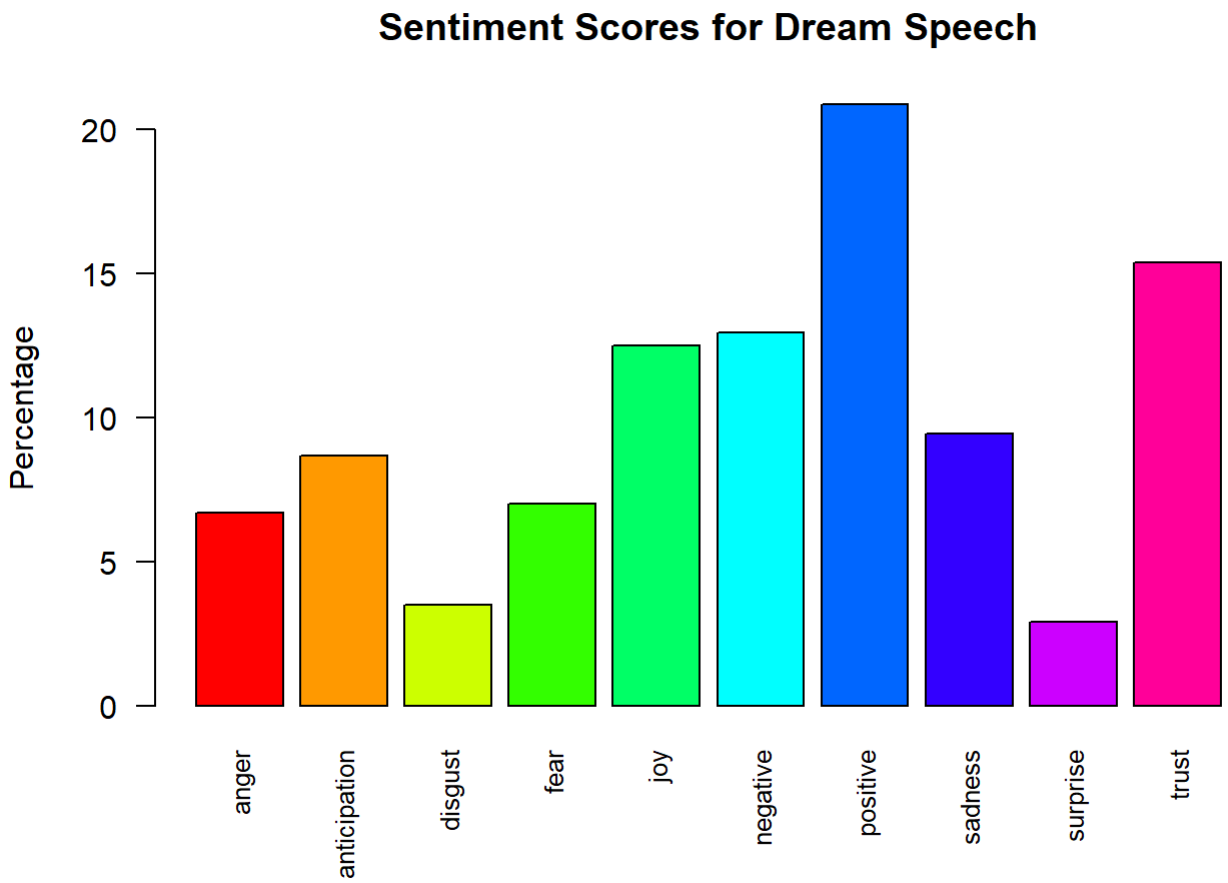
anger <dbl>	anticipation <dbl>	disgust <dbl>	fear <dbl>	joy <dbl>	negative <dbl>	positive <dbl>	sadness <dbl>	surprise <dbl>	trust <dbl>
44	57	23	46	82	85	137	62	19	101

1 row | 1-10 of 11 columns

```
dreamnrc_senti <- get_sentiments("nrc")
dreamsenti <- inner_join(dreamtokens, dreamnrc_senti)
```

```
## Joining, by = "word"
```

```
dreamcount_senti <- dreamsenti %>% group_by(sentiment) %>% tally()

barplot(100*dreamcount_senti$n/sum(dreamcount_senti$n),
        names.arg = dreamcount_senti$sentiment,
        las=2,  #vertical orientation of text
        cex.names = 0.8,
        col=rainbow(10),
        ylab='Percentage',
        main= 'Sentiment Scores for Dream Speech')
```



4. What are the four most common sentiments? Display them with their counts.

```
#The four most common sentiments are positive, trust, negative, joy.
ordered<-dreamcount_senti[order(-dreamcount_senti$n),c(1,2)]
head(ordered,4)
```

sentiment <chr>	n <int>
positive	137
trust	101
negative	85

sentiment	n
<chr>	<int>
joy	82

4 rows

5. Find the top six words associated with each of the 4 sentiments you identified in the previous question. Why do you think these four sentiments are most common?

```
dreamword_proportions <- count(dreamsenti,word,sentiment)
dreamword_proportions <- group_by(dreamword_proportions,n)
dreampositive<-dreamword_proportions[dreamword_proportions$sentiment=="positive",]
dreamtrust<-dreamword_proportions[dreamword_proportions$sentiment=="trust",]
dreamanticipation<-dreamword_proportions[dreamword_proportions$sentiment=="anticipation",]
dreamnegative<-dreamword_proportions[dreamword_proportions$sentiment=="negative",]

DreamPosOrdered<-dreampositive[order(-dreampositive$n),c(1,2,3)]
DreamNegOrdered<-dreamnegative[order(-dreamnegative$n),c(1,2,3)]
DreamTrustOrdered<-dreamtrust[order(-dreamtrust$n),c(1,2,3)]
DreamAnticipationOrdered<-dreamanticipation[order(-dreamanticipation$n),c(1,2,3)]

#Top 6 words associated with Positive Sentiment
head(DreamPosOrdered,6)
```

word	sentiment	n
<chr>	<chr>	<int>
freedom	positive	20
justice	positive	8
satisfied	positive	8
white	positive	6
faith	positive	5
hope	positive	4

6 rows

```
#Top 6 words associated with Trust Sentiment
head(DreamTrustOrdered,6)
```

word <chr>	sentiment <chr>	n <int>
freedom	trust	20
nation	trust	10
justice	trust	8
white	trust	6
faith	trust	5
hope	trust	4

6 rows

```
#Top 6 words associated with Anticipation Sentiment
head(DreamAnticipationOrdered,6)
```

word <chr>	sentiment <chr>	n <int>
white	anticipation	6
faith	anticipation	5
time	anticipation	5
hope	anticipation	4
mountain	anticipation	4
sing	anticipation	3

6 rows

```
#Top 6 words associated with Negative Sentiment
head(DreamNegOrdered,6)
```

word <chr>	sentiment <chr>	n <int>
negro	negative	13
black	negative	4
injustice	negative	3
words	negative	3
brutality	negative	2
despair	negative	2

6 rows