HOMEWORK- 5

FUNDAMENTALS OF SCIENCE

WINE DATASET

Sadiya Amreen- 2079690

2022-11-16

AIM

To evaluate and compare Classification and Clustering models on Wine prediction dataset from the UCI dataset library.

DATA GATHERING AND INTEGRATION

PROBLEM DEFINITION

Data Set Characteristics:	Multivariate	**Number of Instances:**	182	**Area:**	Physical
Attribute Characteristics:	Integer, Real	**Number of Attributes:**	16	**Date Donated**	1991-07-01
Associated Tasks:	Classification Clustering	**Missing Values:**	161	**Number of Web Hits:**	2044123

The wine data set consists of 16 different parameters of wine such as Alcohol and Ash content which was measured for 178 wine samples. These wines were grown in the same region in Italy but derived from three different cultivars; therefore, there are three different classes of wine. The goal here is to find a model that can predict the class of wine, given 16 measured parameters and find out the major differences among the three different classes. This project will describe two Classification models – Decision Tree and KNN (K-Nearest Neighbor), and one Clustering model – K-Means, and asses the accuracy of each model. Furthermore, used principal component analysis to identify and explore the differences among the three classes.

The features are-
1)Alcohol
2) Malic acid
3) Ash
4) Alcalinity of ash
5) Magnesium
6) Total phenols
7) Flavanoids
8) Nonflavanoid phenols
9) Proanthocyanins
10)Color intensity
11)Hue
12)OD280/OD315 of diluted wines
13)Proline
14)Phosphoric Acid
15)Wine_Model

COLLECTING THE DATA:

Implementing the analysis in R.

Loading the Dataset:

```
#import the data set wine
 Winedata <- read.csv("C:/Sadiya Studies/Data Science/DS441-Fundamts DS/datasets/Wine_dset.csv", stringsAsFactors=TRUE)
head(Winedata)
```

```
##   Type Alcohol Malic_Acid  Ash Ash_Alcanity Magnesium Total_Phenols Flavanoids
## 1    1   14.23       1.71 2.43       15.6       127          2.80       3.06
## 2    1   13.20       1.78 2.14       11.2       100          2.65       2.76
## 3    1   13.16       2.36 2.67       18.6       101          2.80       3.24
## 4    1   14.37       1.95 2.50       16.8       113          3.85       3.49
## 5    1   13.24       2.59 2.87       21.0       118          2.80       2.69
## 6    1   14.20       1.76 2.45       15.2       112          3.27       3.39
##   Nonflavanoid_Phenols Proanthocyanins Color_Intensity  Hue OD280 Proline
## 1                 0.28            2.29            5.64 1.04  3.92    1065
## 2                 0.26            1.28            4.38 1.05  3.40    1050
```

```
## 3          0.30          2.81          5.68 1.03  3.17    1185
## 4          0.24          2.18          7.80 0.86  3.45    1480
## 5          0.39          1.82          4.32 1.04  2.93     735
## 6          0.34          1.97          6.75 1.05  2.85    1450
##   Phosphoric.acid Wine_Model
## 1            1.25       AA3V
## 2            3.26       CC5G
## 3            9.80       CR7D
## 4            1.23       CP9R
## 5            6.15       CF4K
## 6            9.50       CD3N
```

DATA EXPLORATION

In order to explore the Data, applying visualizations and summary statistics to evaluate individual distributions and relationships between pairs.

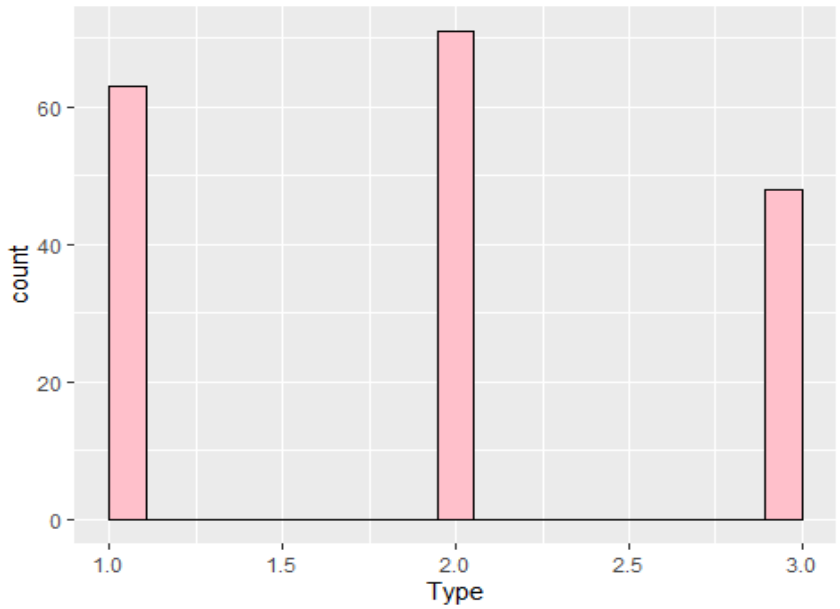
```
#data exploration
#Checking the dimension of data
dim(Winedata)

## [1] 182  16

#viewing the structure of the data

str(Winedata)

## 'data.frame':    182 obs. of  16 variables:
##  $ Type             : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Alcohol          : num  14.2 13.2 13.2 14.4 13.2 ...
##  $ Malic_Acid       : num  1.71 1.78 2.36 1.95 2.59 1.76 1.87 2.15 1.64 1.35 ...
##  $ Ash              : num  2.43 2.14 2.67 2.5 2.87 2.45 2.45 2.61 2.17 2.27 ...
##  $ Ash_Alcanity     : num  15.6 11.2 18.6 16.8 21 15.2 14.6 17.6 14 16 ...
##  $ Magnesium        : int  127 100 101 113 118 112 96 121 97 98 ...
##  $ Total_Phenols    : num  2.8 2.65 2.8 3.85 2.8 3.27 2.5 2.6 2.8 2.98 ...
##  $ Flavanoids       : num  3.06 2.76 3.24 3.49 2.69 3.39 2.52 2.51 2.98 3.15 ...
```

```
##  $ Nonflavanoid_Phenols: num  0.28 0.26 0.3 0.24 0.39 0.34 0.3 0.31 0.29 0.22
...
##  $ Proanthocyanins     : num  2.29 1.28 2.81 2.18 1.82 1.97 1.98 1.25 1.98 1.85 .
..
##  $ Color_Intensity     : num  5.64 4.38 5.68 7.8 4.32 6.75 5.25 5.05 5.2 7.22 ...
##  $ Hue                 : num  1.04 1.05 1.03 0.86 1.04 1.05 1.02 1.06 1.08 1.01 ...
##  $ OD280               : num  3.92 3.4 3.17 3.45 2.93 2.85 3.58 3.58 2.85 3.55 ...
##  $ Proline             : int  1065 1050 1185 1480 735 1450 1290 1295 1045 1045 ...
##  $ Phosphoric.acid     : num  1.25 3.26 9.8 1.23 6.15 9.5 2.2 2.5 66 25.3 ...
##  $ Wine_Model          : Factor w/ 3 levels "","AA3V","CC5G": 2 3 3 3 3 3 3 3 3
3 ...
```

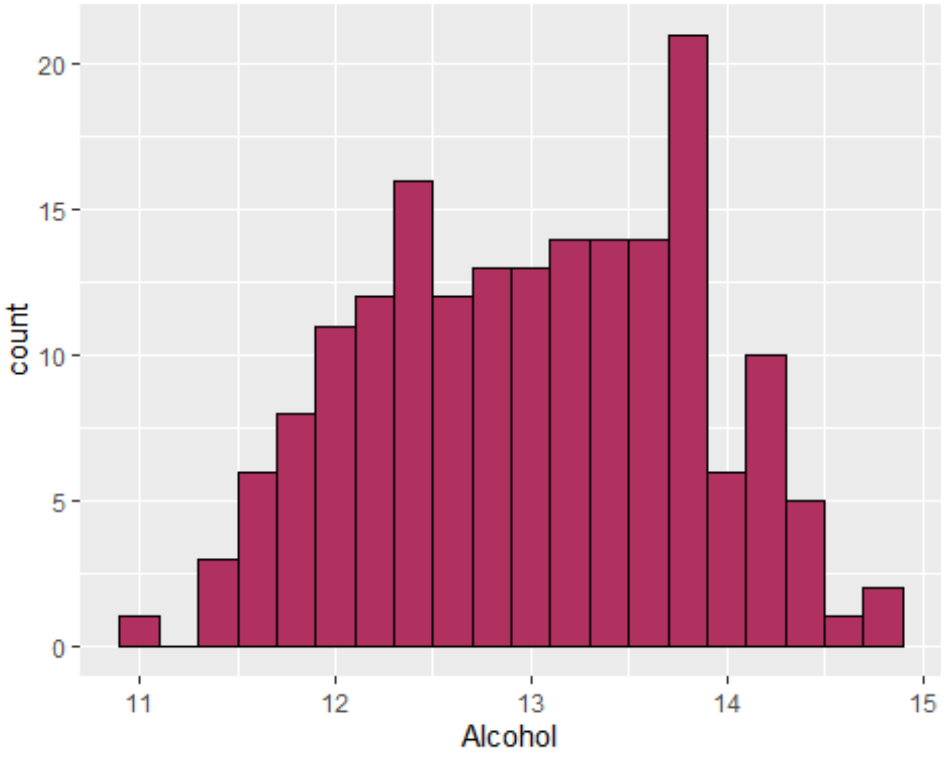
VISUALIZING THE DATA:

The following graphs visualizes the individual distributions

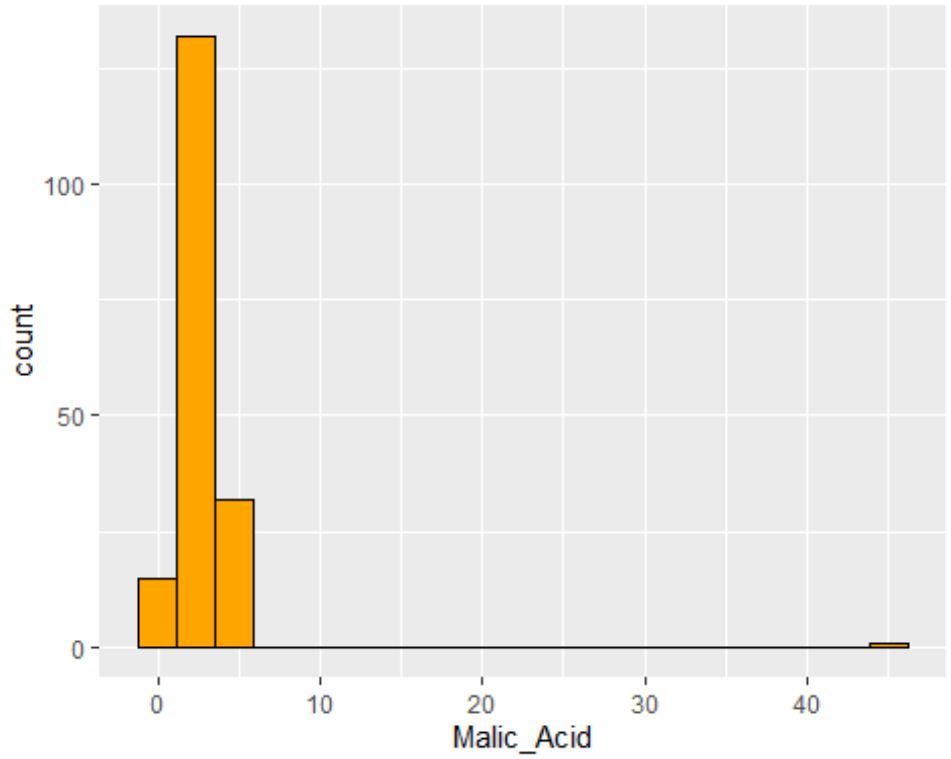
```
#Visualization: Numerical data - Type
summary(Winedata$Type)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   2.000   1.918   3.000   3.000

ggplot (Winedata, aes (Type)) + geom_histogram( fill='pink',color="black" , bins =
20)
```



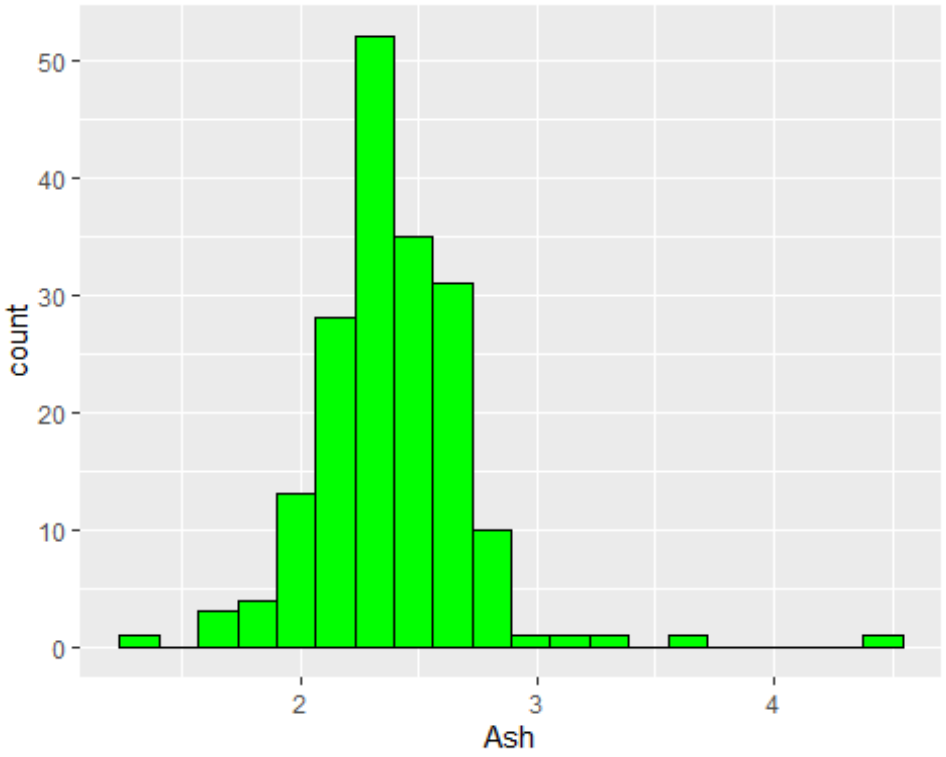
```
#Visualization: Numerical data - Alcohol
summary(Winedata$Alcohol)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   11.03   12.37   13.05   13.02   13.69   14.83

ggplot (Winedata, aes (Alcohol)) + geom_histogram( fill='maroon',color="black" , bins = 20)
```



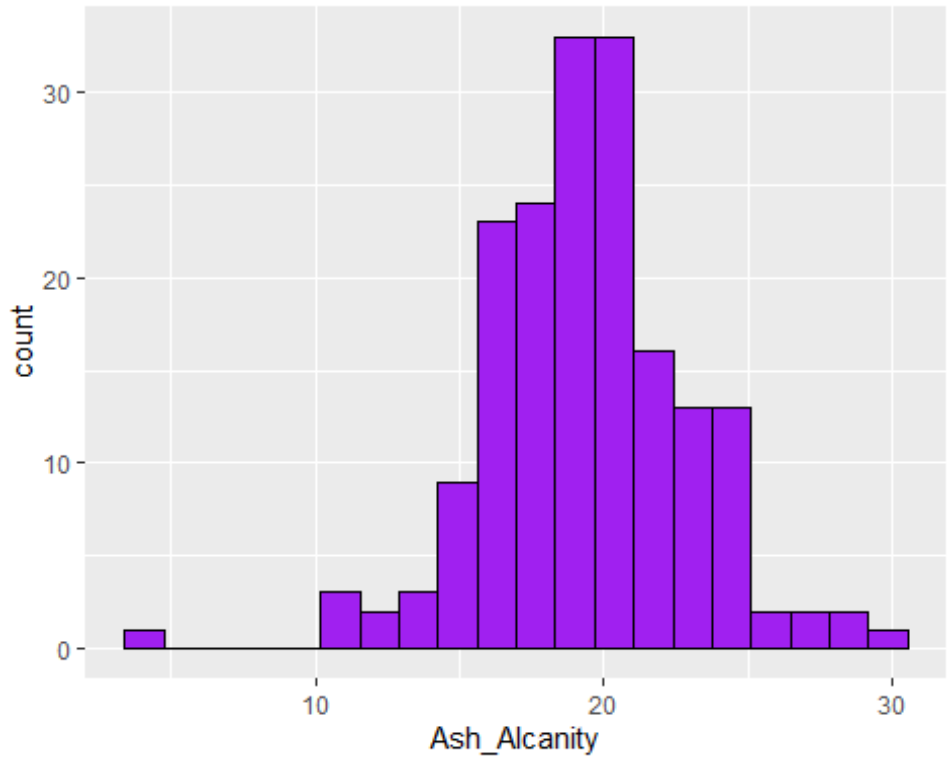
```
#Visualization: Numerical data - Malic_Acid
summary(Winedata$Malic_Acid)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.740   1.607   1.865   2.574   3.105  45.800       2

ggplot (Winedata, aes (Malic_Acid)) + geom_histogram( fill='orange',color="black
" , bins = 20)
```



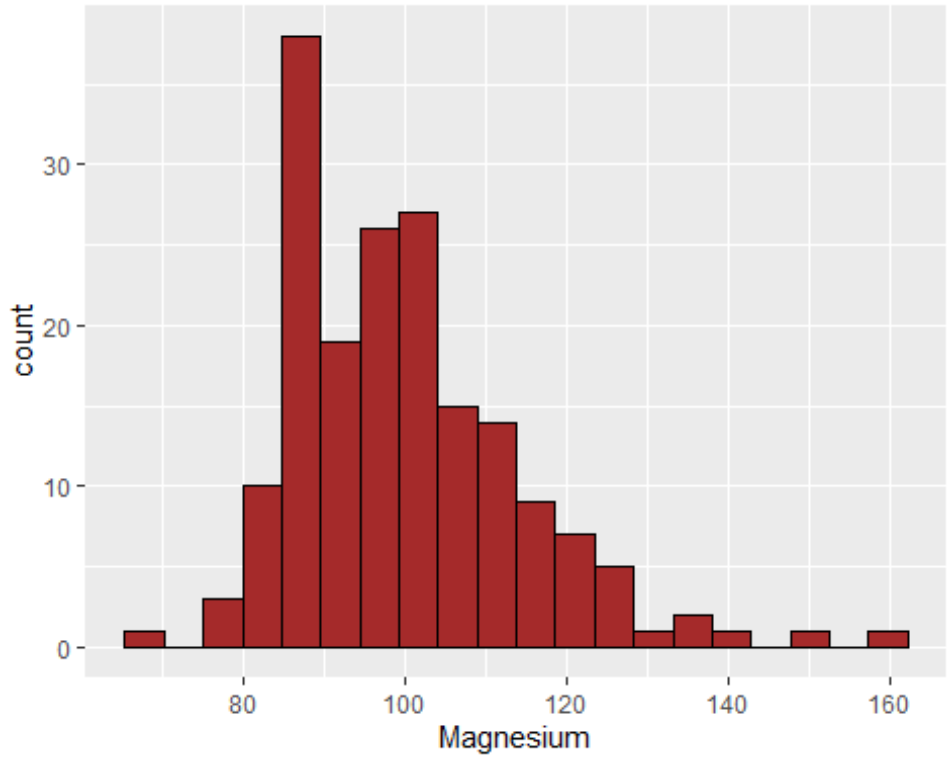
```
#Visualization: Numerical data - Ash
summary(Winedata$Ash)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.360   2.212   2.360   2.387   2.575   4.500

ggplot (Winedata, aes (Ash)) + geom_histogram( fill='green',color="black" , bins = 20)
```



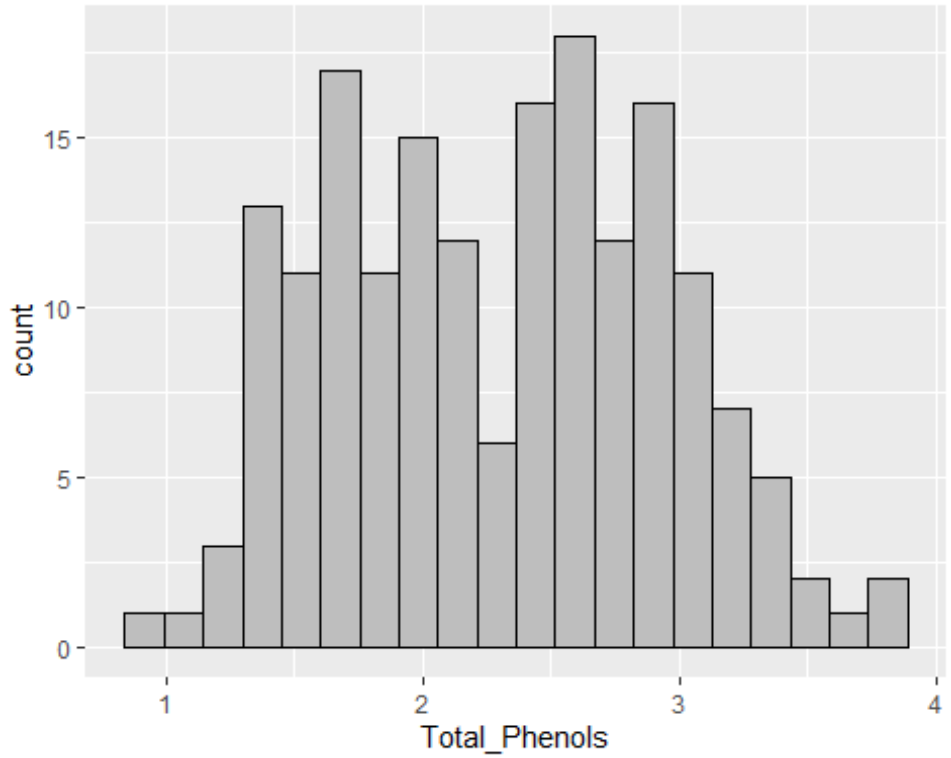
```
#Visualization: Numerical data - Ash_Alcanity
summary(Winedata$Ash_Alcanity)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    4.20   17.18   19.45   19.41   21.50   30.00       2

ggplot (Winedata, aes (Ash_Alcanity)) + geom_histogram( fill='purple',color="black" , bins = 20)
```



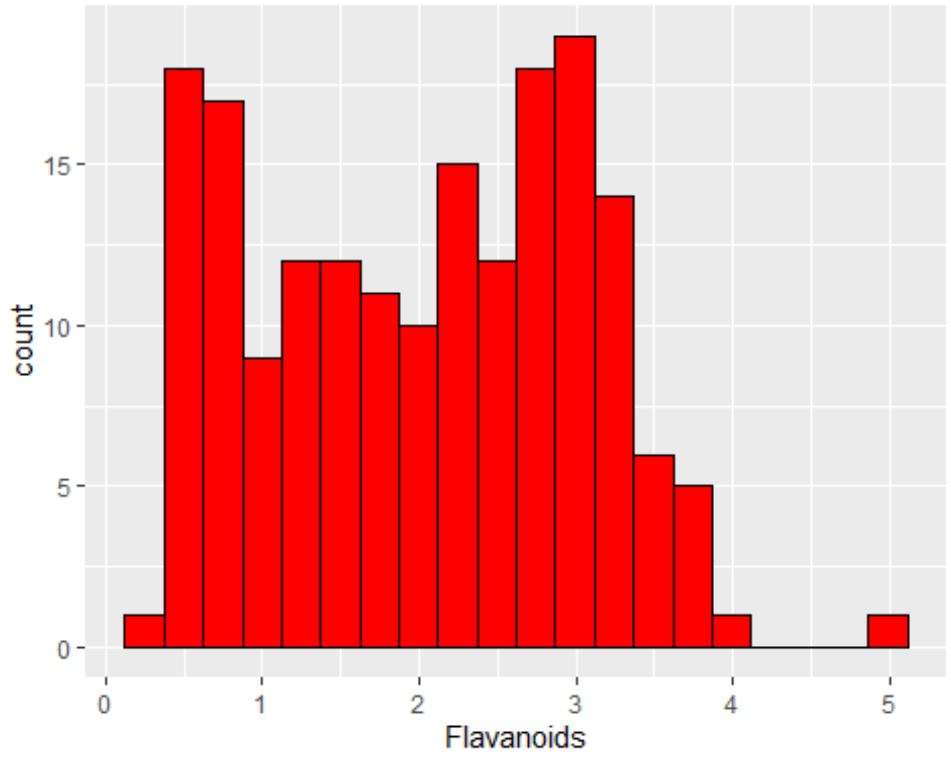
```
#Visualization: Numerical data - Magnesium
summary(Winedata$Magnesium)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   70.00   88.00   98.00   99.93  107.25  162.00       2

ggplot (Winedata, aes (Magnesium)) + geom_histogram( fill='brown',color="black
" , bins = 20)
```



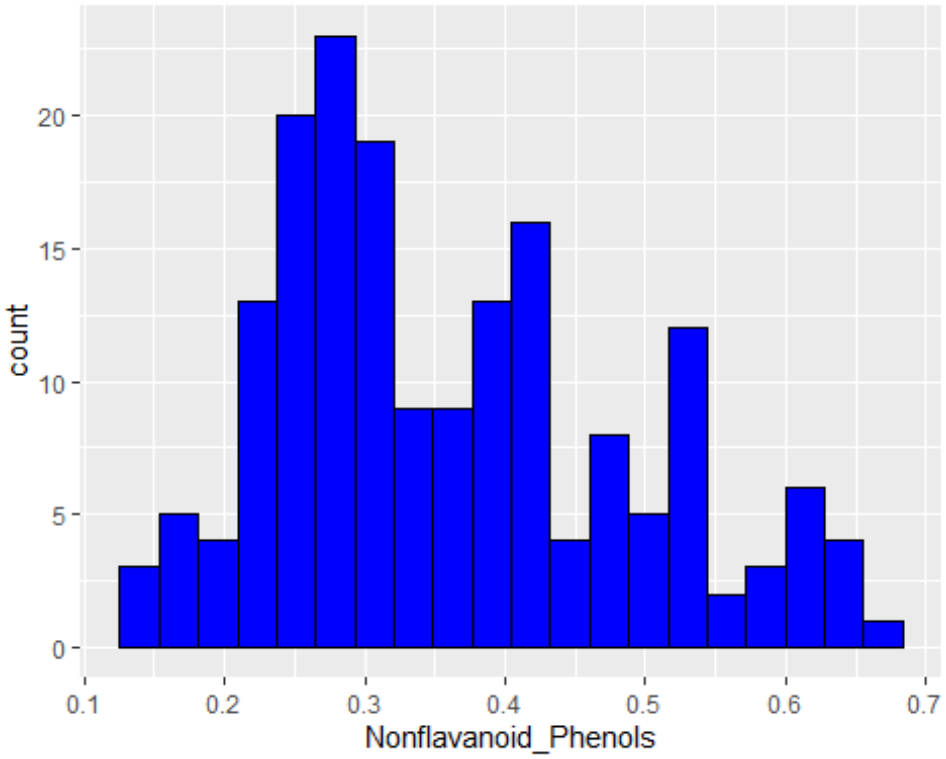
```
#Visualization: Numerical data - Total_Phenols
summary(Winedata$Total_Phenols)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.980   1.748   2.380   2.305   2.800   3.880       2

ggplot (Winedata, aes (Total_Phenols)) + geom_histogram( fill='grey',color="blac
k" , bins = 20)
```



```
#Visualization: Numerical data - Flavanoids
summary(Winedata$Flavanoids)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.340   1.200   2.130   2.028   2.880   5.080       1

ggplot (Winedata, aes (Flavanoids)) + geom_histogram( fill='red',color="black" , bins = 20)
```



```
#Visualization: Numerical data - Nonflavanoid_Phenols
summary(Winedata$Nonflavanoid_Phenols)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  0.1300  0.2650  0.3400  0.3611  0.4350  0.6600       3

ggplot (Winedata, aes (Nonflavanoid_Phenols)) + geom_histogram( fill='blue',colo
r="black" , bins = 20)
```

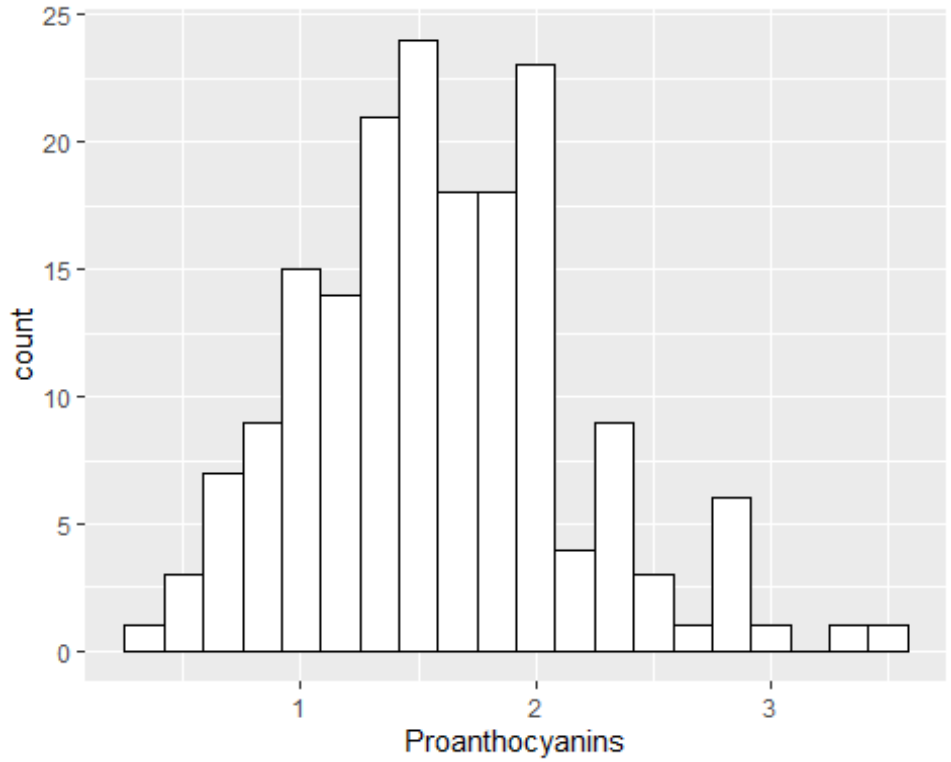


#Visualization: Numerical data - Proanthocyanins

```
summary(Winedata$Proanthocyanins)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.410   1.250   1.560   1.593   1.950   3.580       3
```

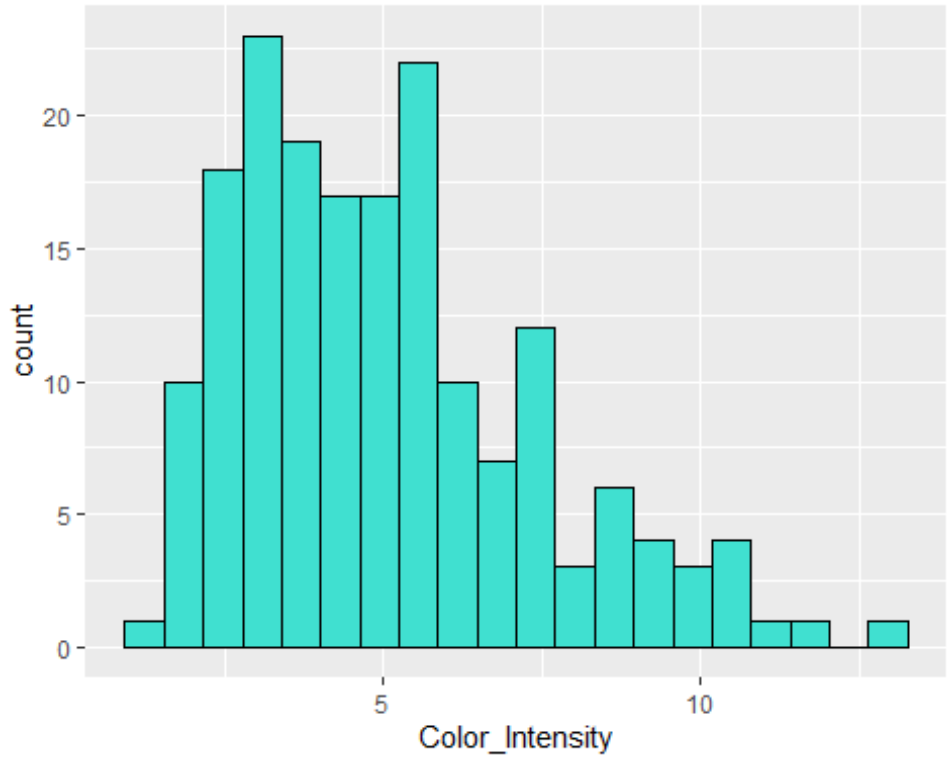
```
ggplot (Winedata, aes (Proanthocyanins)) + geom_histogram( fill='white',color="black" , bins = 20)
```



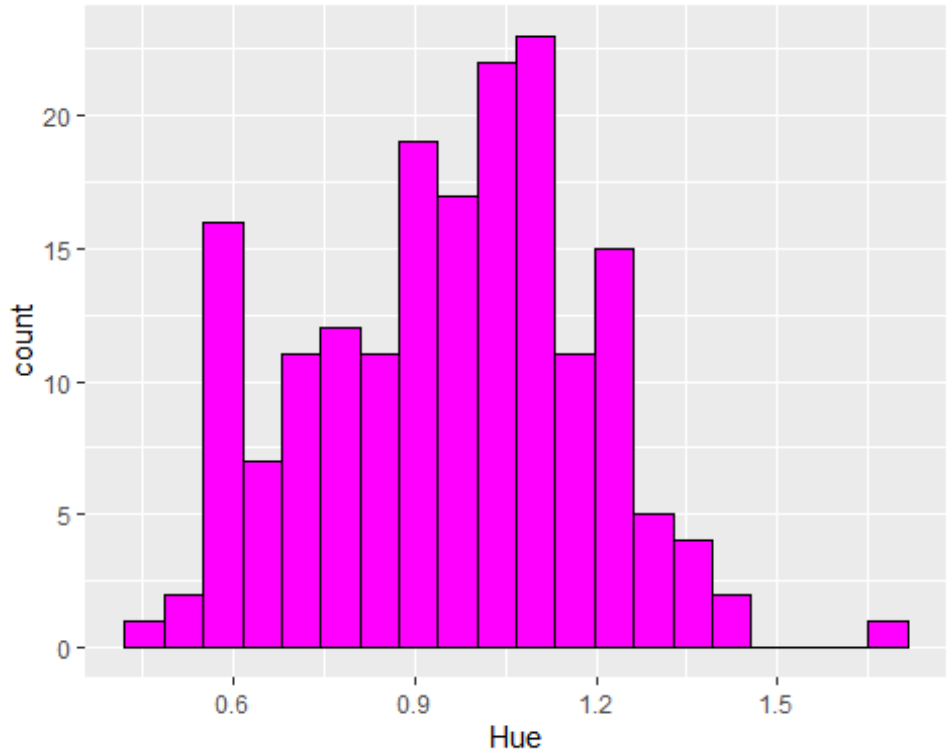
```
#Visualization: Numerical data - Color_Intensity
summary(Winedata$Color_Intensity)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   1.280   3.230   4.700   5.068   6.225  13.000       3
```

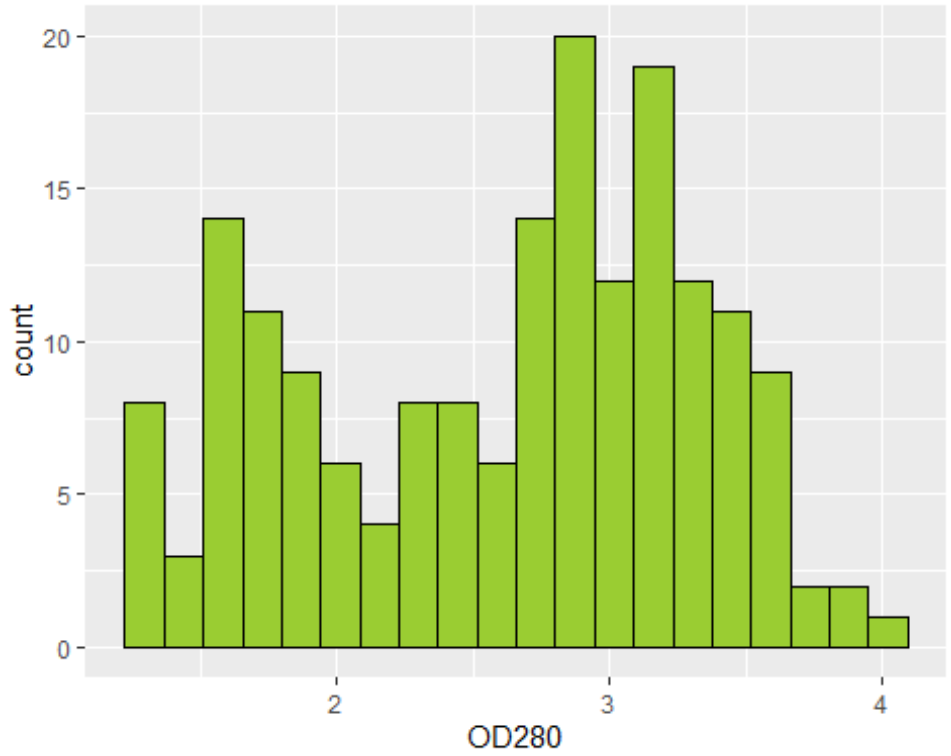
```
ggplot (Winedata, aes (Color_Intensity)) + geom_histogram( fill='turquoise',color="black" , bins = 20)
```



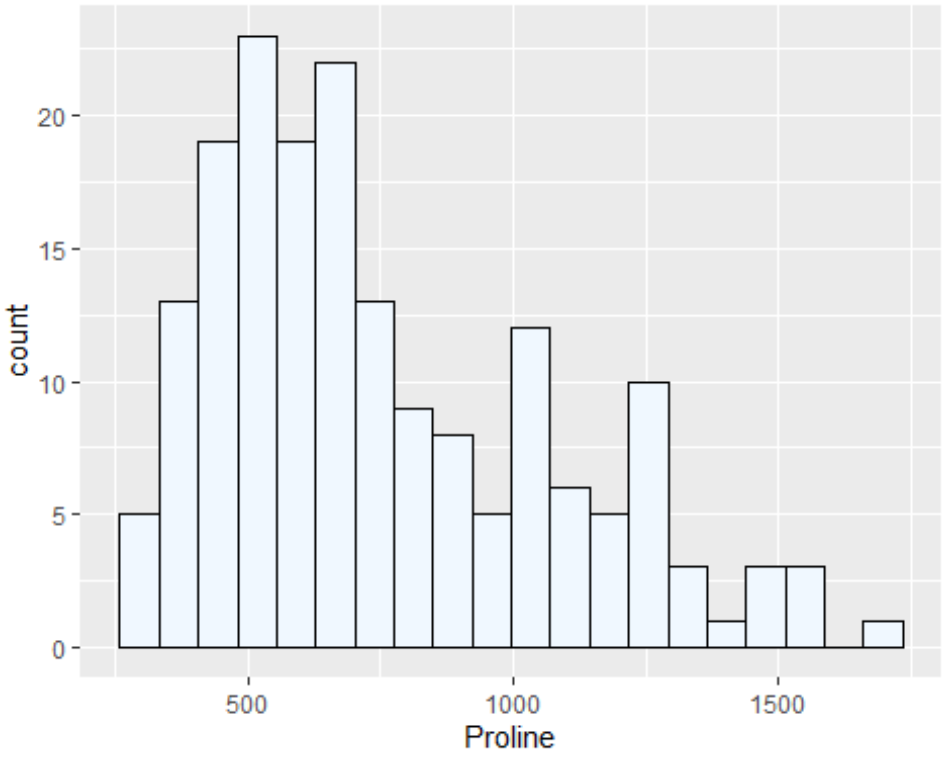
```
#Visualization: Numerical data - Hue
summary(Winedata$Hue)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  0.4800  0.7850  0.9700  0.9582  1.1200  1.7100       3

ggplot (Winedata, aes (Hue)) + geom_histogram( fill='magenta',color="black" , bins = 20)
```



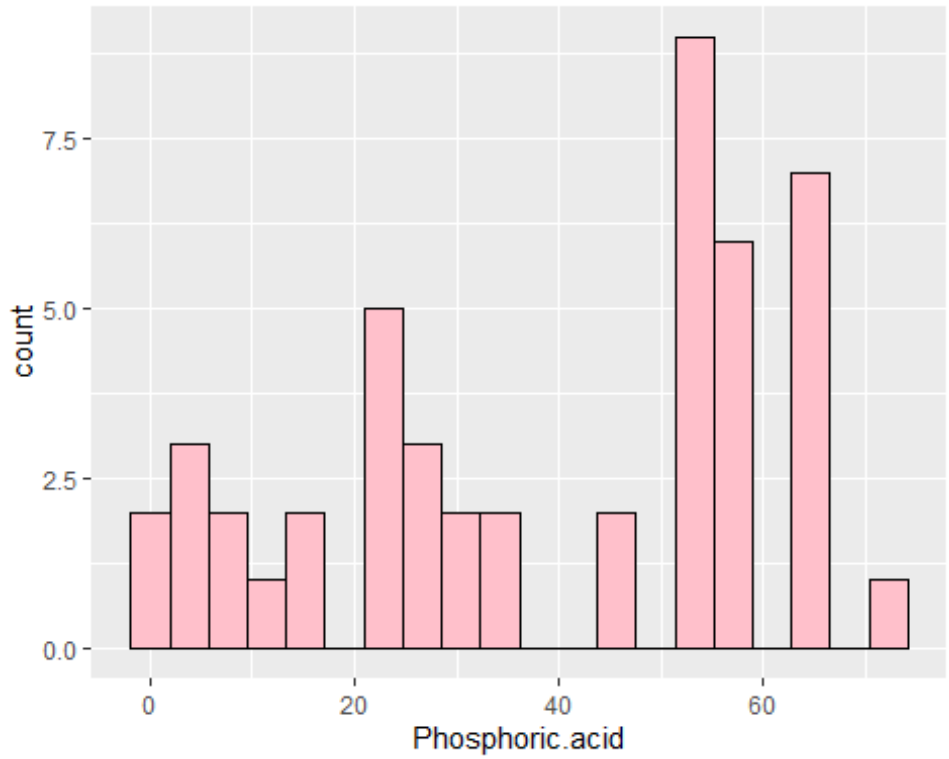
```
#Visualization: Numerical data -OD280
summary(Winedata$OD280)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   1.270   1.945   2.780   2.613   3.170   4.000       3

ggplot (Winedata, aes (OD280)) + geom_histogram( fill='yellowgreen',color="blac
k" , bins = 20)
```



```
#Visualization: Numerical data -Proline
summary(Winedata$Proline)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   278.0   500.0   673.5   748.9   986.2  1680.0       2

ggplot (Winedata, aes (Proline)) + geom_histogram( fill='aliceblue',color="black" ,
bins = 20)
```



```
#Visualization: Numerical data -Phosphoric.acid
summary(Winedata$Phosphoric.acid)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    1.23   22.50   45.80   38.84   55.90   73.50     135

ggplot (Winedata, aes (Phosphoric.acid)) + geom_histogram( fill='pink',color="black" , bins = 20)
```

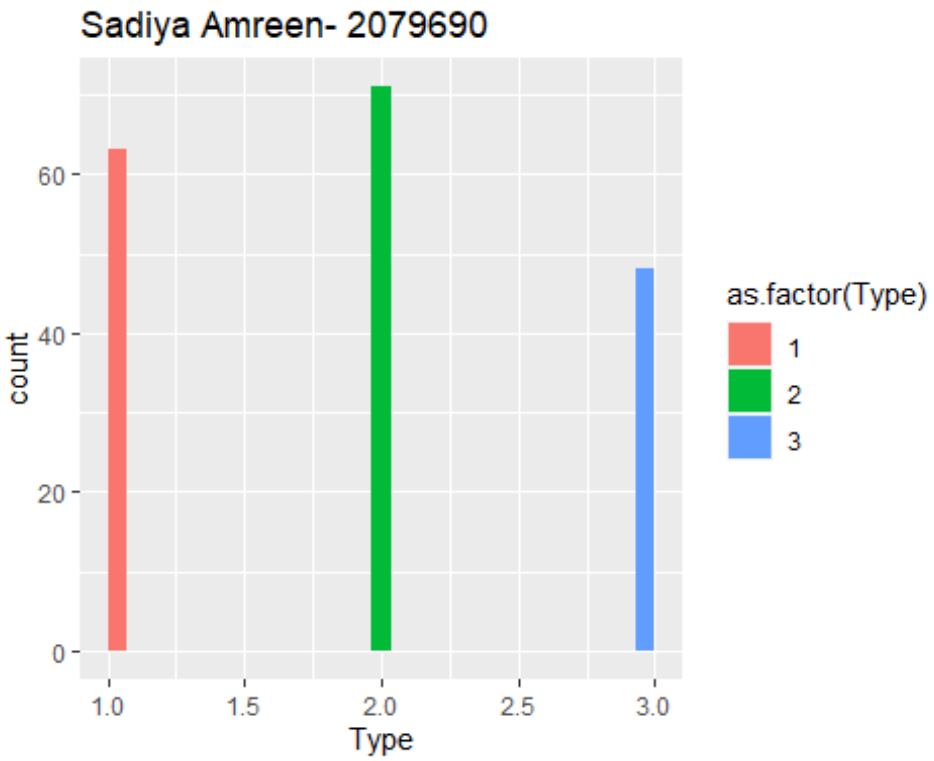


ANALYSIS OF DATA USING VISUALIZATION

This is a histogram that gives the occurrences of the “Type”:

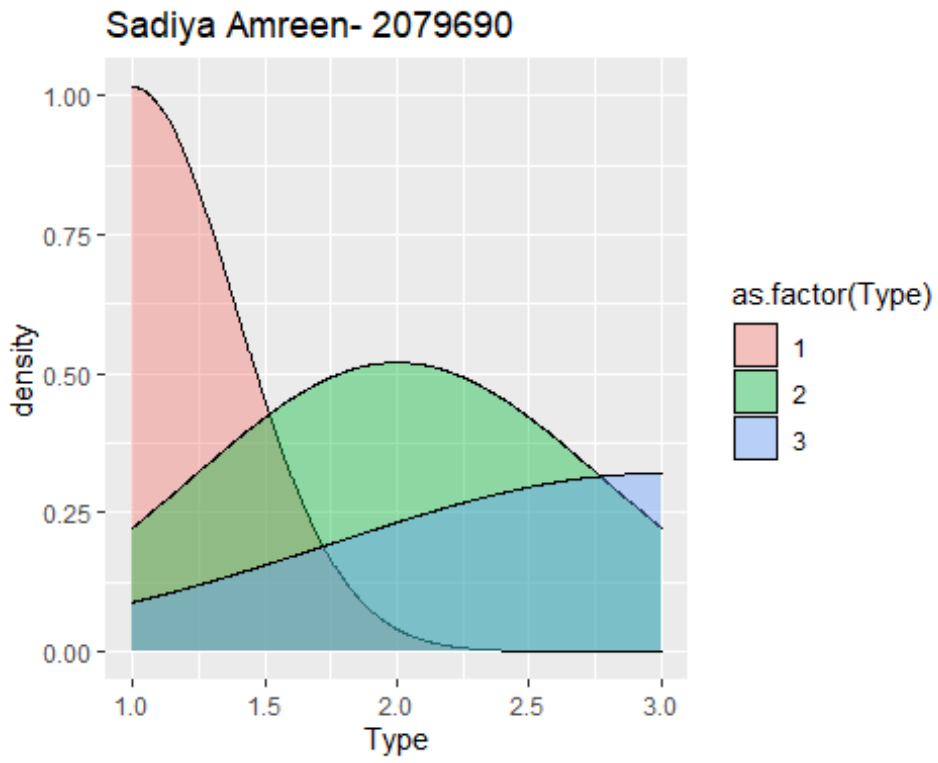
```
ggplot(data=Winedata,aes(x=Type,fill=as.factor(Type)))+geom_histogram()+ labs(
title = "Sadiya Amreen- 2079690")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



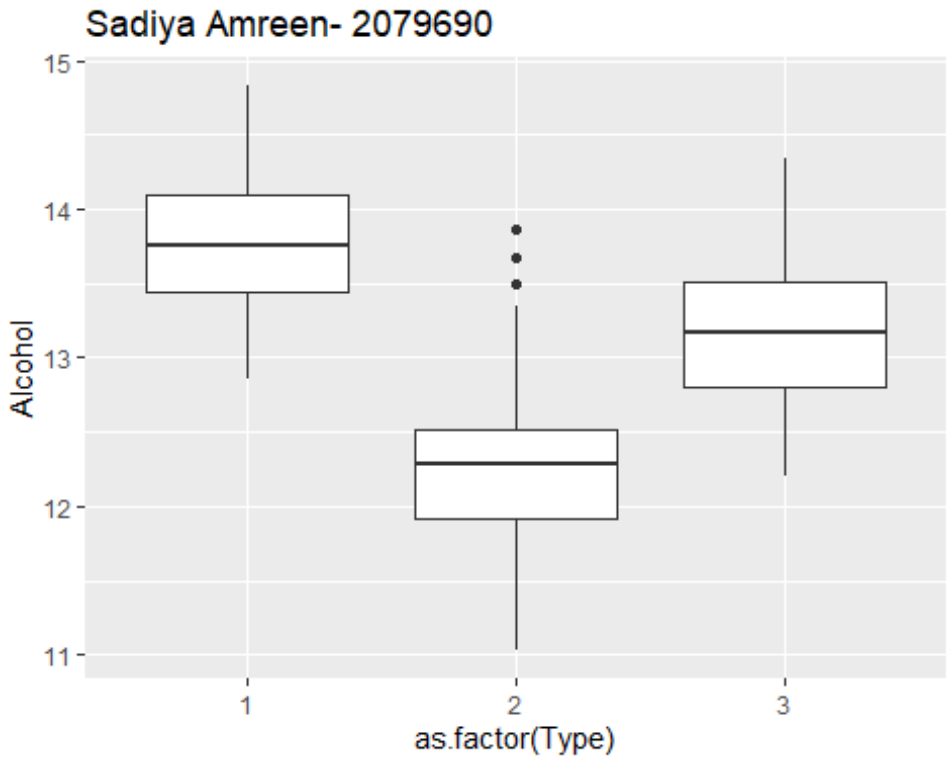
This is a density plot that tells us again about the Type Factor visually:

```
ggplot(data=Winedata,aes(x=Type,fill=as.factor(Type)))+geom_density(alpha=0.4
)+ labs(title = "Sadiya Amreen- 2079690")
```



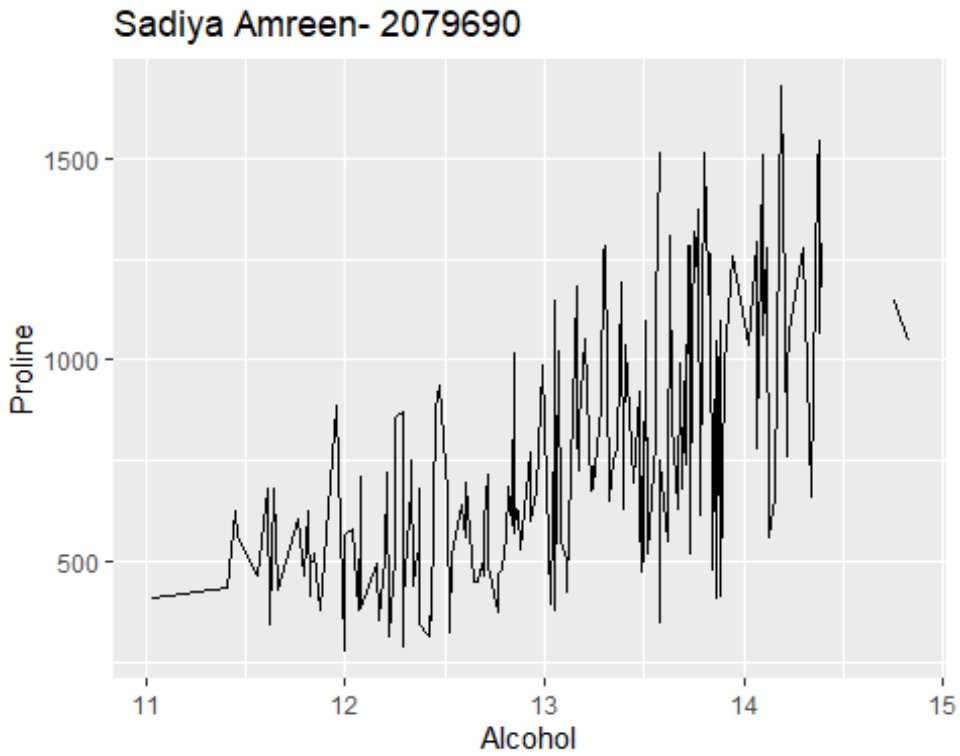
This is a box-plot that generates a graph between Type and Alcohol factors in the data set:

```
ggplot(data=Winedata,aes(x=as.factor(Type),y=Alcohol))+geom_boxplot()+labs(title = "Sadiya Amreen- 2079690")
```



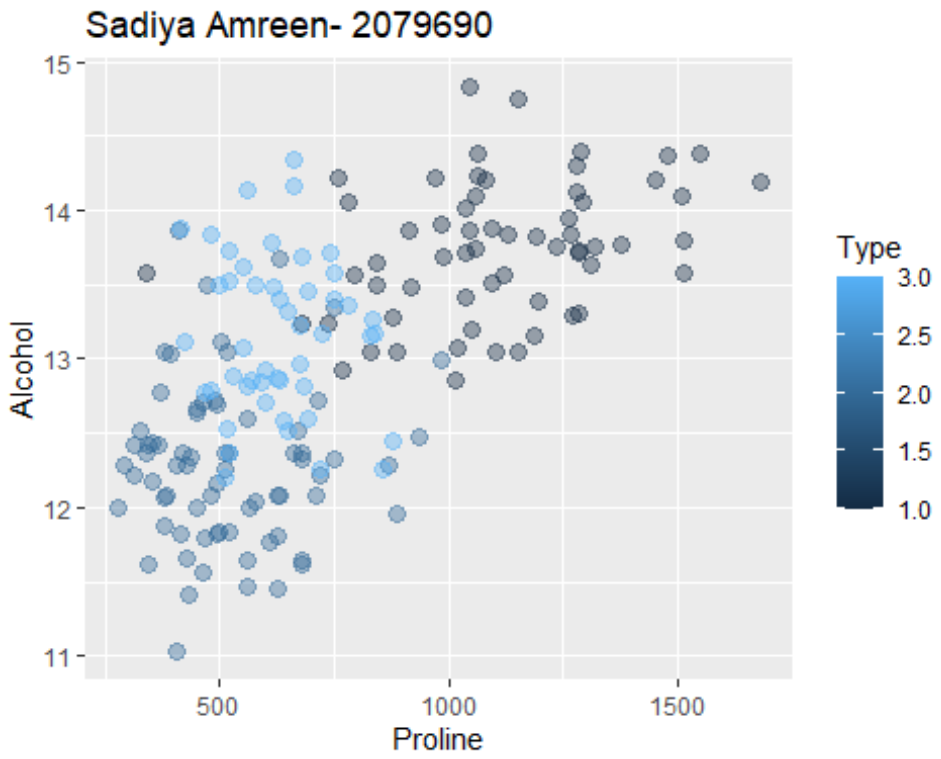
This is a line bar that generates a graph between Alcohol and Proline from our dataset:

```
ggplot(data=Winedata,aes(x=Alcohol,y=Proline))+geom_line()+ labs(title = "Sadiya Amreen- 2079690")
```



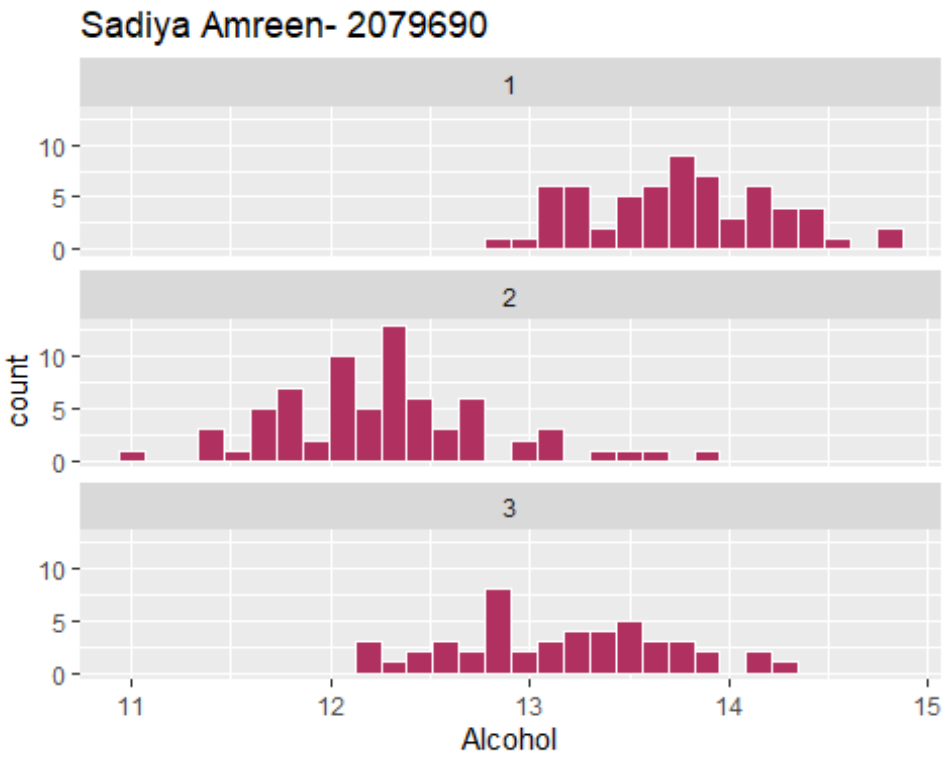
Now we make a scatterplot between Proline , Alcohol and Output factors of our dataset :

```
ggplot(data = Winedata,aes(x=Proline,y=Alcohol,color=Type))+geom_point(alpha =0.4,size=3)+labs(title = "Sadiya Amreen- 2079690")
```



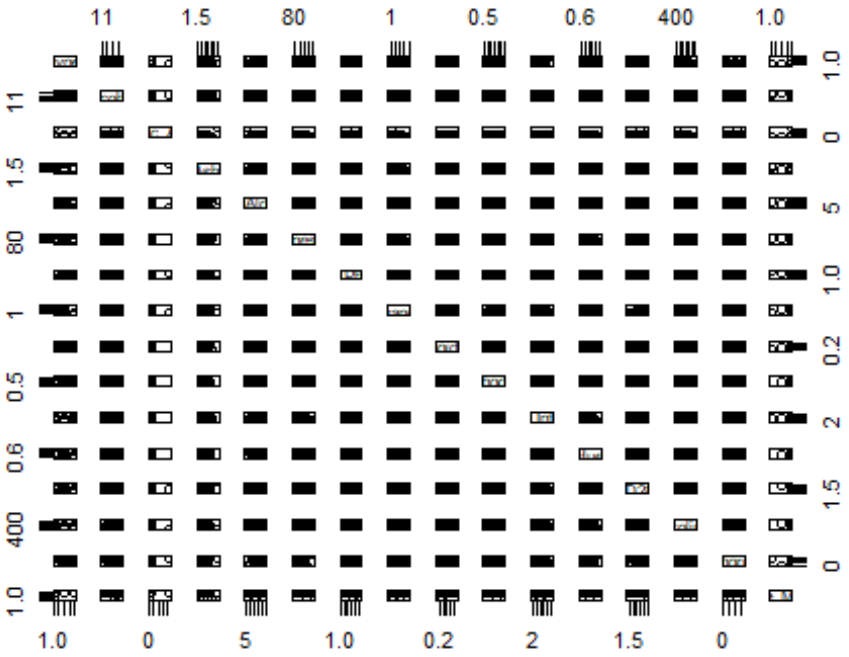
This is a Histogram that consist various columns of our dataset :

```
ggplot(Winedata, aes(x = Alcohol)) + geom_histogram(fill = "maroon", color = "white") + facet_wrap(~Type, ncol = 1)+ labs(title = "Sadiya Amreen- 2079690")

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



The following graph explains how the dataset is correlated to one other:

```
pairs(Winedata)
```



DATA CLEANING :

The Data Cleaning section includes:
a) counting the number of missing values.
b) removing missing values.
c) removing an attribute(Phosphoric acid) as it contains 70% of missing values.
d) removing an attribute (Wine_Model) as it is irrelevant to the Analysis.
e) change variable type of attribute 'Type' from numeric to categorical.
f) showing resulting summary statistics and dimensions of the data stating the data is clean enough to continue with the analysis.

```
dim(Winedata) #dimensions of data before cleaning

## [1] 182  16
summary(Winedata) #summary before Data Cleaning

#Counting number of NA's
sum(is.na(Winedata))

## [1] 161

#Removing NA's

Winedata <- na.omit(Winedata)

#removing Wine_Model column (because its irrelevant to the Analysis)
Winedata$Wine_Model <- NULL

#removing Phosphoric.acid column (70% of data is NA)
Winedata$Phosphoric.acid <- NULL

# change variable type of attribute 'Type' from numeric to categorical

Winedata <- within(Winedata, {
  Type[Type == 1] <- "A"
  Type[Type == 2] <- "B"
  Type[Type == 3] <- "C"
   } )
```

```
summary(Winedata) #summary after Data Cleaning
```

```
##      Type            Alcohol        Malic_Acid         Ash
##  Length:178         Min.   :11.03   Min.   :0.740   Min.   :1.360
##  Class :character   1st Qu.:12.36   1st Qu.:1.603   1st Qu.:2.210
##  Mode  :character   Median :13.05   Median :1.865   Median :2.360
##                     Mean   :13.00   Mean   :2.336   Mean   :2.367
##                     3rd Qu.:13.68   3rd Qu.:3.083   3rd Qu.:2.558
##                     Max.   :14.83   Max.   :5.800   Max.   :3.230
##  Ash_Alcanity     Magnesium      Total_Phenols     Flavanoids
##  Min.   :10.60   Min.   : 70.00   Min.   :0.980   Min.   :0.340
##  1st Qu.:17.20   1st Qu.: 88.00   1st Qu.:1.742   1st Qu.:1.205
##  Median :19.50   Median : 98.00   Median :2.355   Median :2.135
##  Mean   :19.49   Mean   : 99.74   Mean   :2.295   Mean   :2.029
##  3rd Qu.:21.50   3rd Qu.:107.00   3rd Qu.:2.800   3rd Qu.:2.875
##  Max.   :30.00   Max.   :162.00   Max.   :3.880   Max.   :5.080
##  Nonflavanoid_Phenols Proanthocyanins Color_Intensity       Hue
##  Min.   :0.1300       Min.   :0.410   Min.   : 1.280   Min.   :0.4800
##  1st Qu.:0.2700       1st Qu.:1.250   1st Qu.: 3.220   1st Qu.:0.7825
##  Median :0.3400       Median :1.555   Median : 4.690   Median :0.9650
##  Mean   :0.3619       Mean   :1.591   Mean   : 5.058   Mean   :0.9574
##  3rd Qu.:0.4375       3rd Qu.:1.950   3rd Qu.: 6.200   3rd Qu.:1.1200
##  Max.   :0.6600       Max.   :3.580   Max.   :13.000   Max.   :1.7100
##      OD280          Proline
##  Min.   :1.270   Min.   : 278.0
##  1st Qu.:1.938   1st Qu.: 500.5
##  Median :2.780   Median : 673.5
##  Mean   :2.612   Mean   : 746.9
##  3rd Qu.:3.170   3rd Qu.: 985.0
##  Max.   :4.000   Max.   :1680.0
```

```
dim(Winedata) #dimensions after Data Cleaning
```

```
## [1] 178  14
```

DATA PRE-PROCESSING:

Pre-Processing the data by applying normalization so as to scale the attributes for better comparison and performances. Z-Score normalization technique implemented.

```
#summary before normalization
summary(Winedata$Proline)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   278.0   500.5   673.5   746.9   985.0  1680.0

SD= sd(Winedata$Proline)
SD

## [1] 314.9075

#applying z score normalization
z_normfile <- Winedata[c(14)]
m_z_normfile<- mean(z_normfile$Proline)
sd_z_normfile <- sd(z_normfile$Proline)
final_z <- (z_normfile-m_z_normfile)/sd_z_normfile
#summary after normalization
summary(final_z)

##     Proline
##  Min.   :-1.4890
##  1st Qu.:-0.7824
##  Median :-0.2331
##  Mean   : 0.0000
##  3rd Qu.: 0.7561
##  Max.   : 2.9631
```

CLASSIFICATION:

Applying two classification Models on the chosen dataset.
1) Decision tree Analysis.
2)KNN Analysis.

CLASSIFICATION 1:
DESCION TREE ANALYSIS

1) loading required libraries.
2) spliting the data into partitions for Training and testing.
3)Setting of hyperparameters.
4)Creating a Model.
5)Prediction of Data.
6) Applying confusion Matrix on testdata and checking the Accuracy.
7)Visualizing the constructed Decision tree.

```
#classification 1 Decision tree
#call libraries
library(sandwich)
library(zoo)
library(party)
library(tidyverse)
library(ggplot2)
library(caret)
library(mlbench)
library(stats)
library(factoextra)
library(dplyr)
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
```

```r
#DECISION TREE (CLASSIFIER-1)
#Creating Partition of data into 70% and 30%
mainindex = createDataPartition(y=Winedata$Type, p=0.7, list=FALSE)
traindata = Winedata[mainindex,]
testdata = Winedata[-mainindex,]

#Setting hyper parameters
train_control = trainControl(method = "cv", number = 10)
hyper4a = rpart.control(minsplit = 10, maxdepth = 10)

#Applying Model
tree1 <- train(Type ~ .,control = hyper4a, data = traindata , method = "rpart1SE", tr
Control = train_control)
tree1
```

```
## CART
##
## 126 samples
##  13 predictor
##   3 classes: 'A', 'B', 'C'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 114, 114, 112, 114, 113, 114, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9037546  0.8521924
```

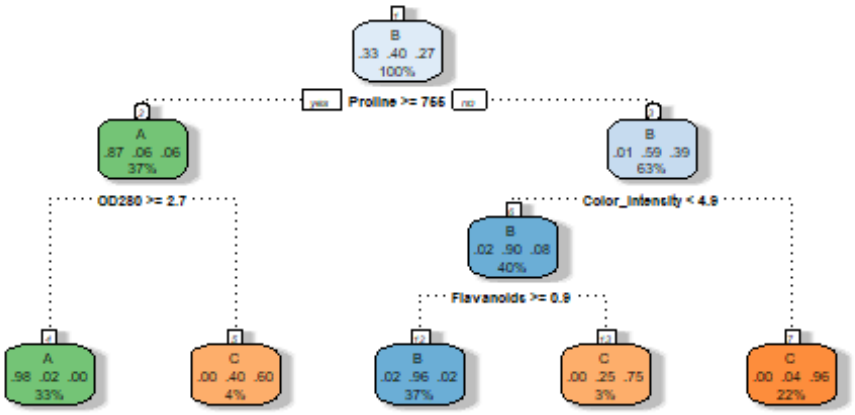
```r
#data predicting
pred_tree <- predict(tree1, testdata)
#Applying Confusion Matrix on test data
confusionMatrix(as.factor(testdata$Type), pred_tree)
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction  A  B  C
##          A 13  1  3
##          B  1 18  2
##          C  0  0 14
##
```

```
## Overall Statistics
##
##                Accuracy : 0.8654
##                  95% CI : (0.7421, 0.9441)
##     No Information Rate : 0.3654
##     P-Value [Acc > NIR] : 1.279e-13
##
##                   Kappa : 0.7979
##
##  Mcnemar's Test P-Value : 0.1718
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C
## Sensitivity            0.9286   0.9474   0.7368
## Specificity            0.8947   0.9091   1.0000
## Pos Pred Value         0.7647   0.8571   1.0000
## Neg Pred Value         0.9714   0.9677   0.8684
## Prevalence             0.2692   0.3654   0.3654
## Detection Rate         0.2500   0.3462   0.2692
## Detection Prevalence   0.3269   0.4038   0.2692
## Balanced Accuracy      0.9117   0.9282   0.8684
```

```
#confusionMatrix(pred,as.factor(testing$Final))
```

```
fancyRpartPlot(tree1$finalModel, caption = "")
```



<u>Inference For Decision Tree Analysis</u>:
"rPart1SE" method has been implemented and with Cross validation of 10 folds applied on Train data, the Accuracy of train data is 90% and for Test data the Confusion Matrix was applied to check the Accuracy of test data which is 86%.

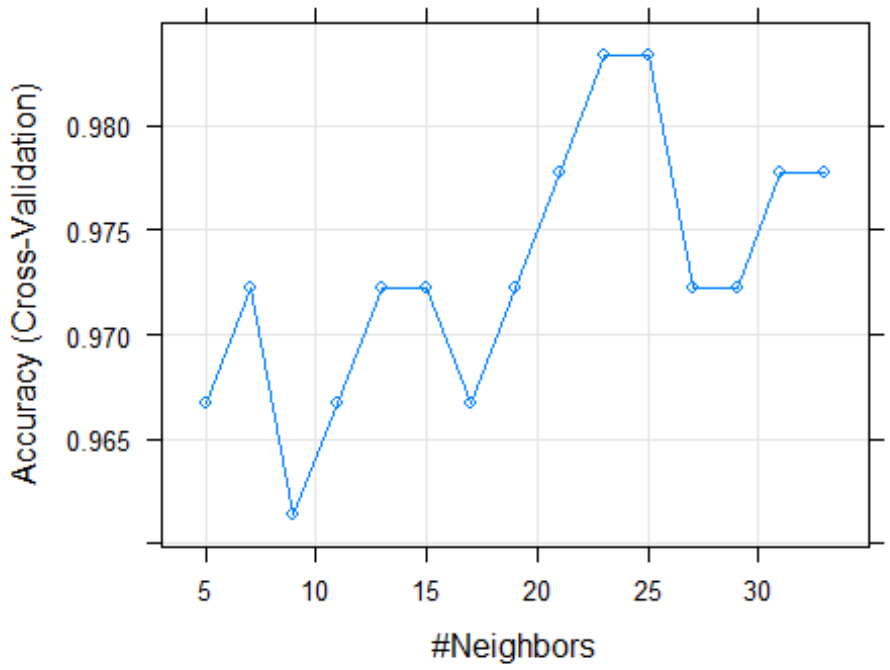
Classification 2:
<u>K-Nearest Neighbors (KNN):</u>

KNN is a non-parametric approach in which an observation is classified based on the class of its K-nearest neighbors. It's a useful model when the decision boundary is non-linear but it will not tell us about which predictors are important. KNN model uses Euclidean distance to measure the distance between two points and if features have different scales it can impact the model. As each of the 13 features have different scales, it is important to normalize data so that all features have the same range of values.

```
#classification 2 KNN
head(Winedata)

##   Type Alcohol Malic_Acid  Ash Ash_Alcanity Magnesium Total_Phenols Flav
anoids
## 1    A   14.23       1.71 2.43         15.6       127          2.80    3.06
## 2    A   13.20       1.78 2.14         11.2       100          2.65    2.76
## 3    A   13.16       2.36 2.67         18.6       101          2.80    3.24
## 4    A   14.37       1.95 2.50         16.8       113          3.85    3.49
## 5    A   13.24       2.59 2.87         21.0       118          2.80    2.69
## 6    A   14.20       1.76 2.45         15.2       112          3.27    3.39
##   Nonflavanoid_Phenols Proanthocyanins Color_Intensity  Hue OD280 Proline
## 1                 0.28             2.29            5.64 1.04  3.92    1065
## 2                 0.26             1.28            4.38 1.05  3.40    1050
## 3                 0.30             2.81            5.68 1.03  3.17    1185
## 4                 0.24             2.18            7.80 0.86  3.45    1480
## 5                 0.39             1.82            4.32 1.04  2.93     735
## 6             0.34         1.97          6.75 1.05  2.85    1450

set.seed(123)

# scaling is crucial for KNN
ctrl <- trainControl(method="cv", number = 10)
knnFit <- train(Type ~ ., data = Winedata,
                method = "knn",
                trControl = ctrl,
                preProcess = c("center","scale"))

#Output of kNN fit
knnFit

## k-Nearest Neighbors
##
## 178 samples
##  13 predictor
##   3 classes: 'A', 'B', 'C'
##
## Pre-processing: centered (13), scaled (13)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 160, 160, 161, 160, 159, 160, ...
## Resampling results across tuning parameters:
```

```
##
##   k  Accuracy   Kappa
##   5  0.9666667  0.9501149
##   7  0.9666667  0.9501149
##   9  0.9614035  0.9421317
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 7.
```

```
set.seed(123)
ctrl <- trainControl(method="cv", number = 10)
knnFit <- train(Type ~ ., data = Winedata,
                method = "knn",
                trControl = ctrl,
                preProcess = c("center","scale"),
                tuneLength = 15)

# Show a plot of accuracy vs k
plot(knnFit)
```



```
library(kknn)

# setup a tuneGrid with the tuning parameters
tuneGrid <- expand.grid(kmax = 3:7,                    # test a range of k values 3 to 7
                        kernel = c("rectangular", "cos"),  # regular and cosine-based distan
ce functions
                        distance = 1:3)                # powers of Minkowski 1 to 3

# tune and fit the model with 10-fold cross validation,
# standardization, and our specialized tune grid
kknn_fit <- train(Type ~ .,
                  data = Winedata,
                  method = 'kknn',
                  trControl = ctrl,
                  preProcess = c('center', 'scale'),
                  tuneGrid = tuneGrid)

# Printing trained model provides report
kknn_fit

## k-Nearest Neighbors
##
## 178 samples
##  13 predictor
##   3 classes: 'A', 'B', 'C'
##
## Pre-processing: centered (13), scaled (13)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 160, 160, 160, 161, 160, 161, ...
## Resampling results across tuning parameters:
##
##   kmax  kernel      distance  Accuracy   Kappa
##   3     rectangular  1        0.9774510  0.9659832
##   3     rectangular  2        0.9551944  0.9324404
##   3     rectangular  3        0.9604231  0.9402841
##   3     cos          1        0.9774510  0.9659832
##   3     cos          2        0.9551944  0.9324404
##   3     cos          3        0.9604231  0.9402841
##   4     rectangular  1        0.9774510  0.9659832
##   4     rectangular  2        0.9551944  0.9324404
##   4     rectangular  3        0.9604231  0.9402841
```

```
##   4     cos         1      0.9774510  0.9659832
##   4     cos         2      0.9551944  0.9324404
##   4     cos         3      0.9604231  0.9402841
##   5     rectangular 1          0.9774510  0.9659832
##   5     rectangular 2          0.9551944  0.9325179
##   5     rectangular 3          0.9604231  0.9402841
##   5     cos         1      0.9774510  0.9659832
##   5     cos         2      0.9551944  0.9324404
##   5     cos         3      0.9604231  0.9402841
##   6     rectangular 1          0.9774510  0.9659832
##   6     rectangular 2          0.9551944  0.9325179
##   6     rectangular 3          0.9604231  0.9402841
##   6     cos         1      0.9774510  0.9659832
##   6     cos         2      0.9551944  0.9324404
##   6     cos         3      0.9604231  0.9402841
##   7     rectangular 1          0.9774510  0.9659832
##   7     rectangular 2          0.9551944  0.9325179
##   7     rectangular 3          0.9545408  0.9314763
##   7     cos         1      0.9774510  0.9659832
##   7     cos         2      0.9551944  0.9324404
##   7     cos         3      0.9604231  0.9402841
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were kmax = 7, distance = 1 and kernel
##  = rectangular.
```

```
# Predict
pred_knn <- predict(kknn_fit, Winedata)

# Generate confusion matrix
confusionMatrix(as.factor(Winedata$Type), pred_knn)
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction  A  B  C
##          A 59  0  0
##          B  0 71  0
##          C  0  0 48
##
```

```
## Overall Statistics
##
##               Accuracy : 1
##                 95% CI : (0.9795, 1)
##    No Information Rate : 0.3989
##    P-Value [Acc > NIR] : < 2.2e-16
##
##                  Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C
## Sensitivity            1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000
## Prevalence             0.3315   0.3989   0.2697
## Detection Rate         0.3315   0.3989   0.2697
## Detection Prevalence   0.3315   0.3989   0.2697
## Balanced Accuracy      1.0000   1.0000   1.0000
```

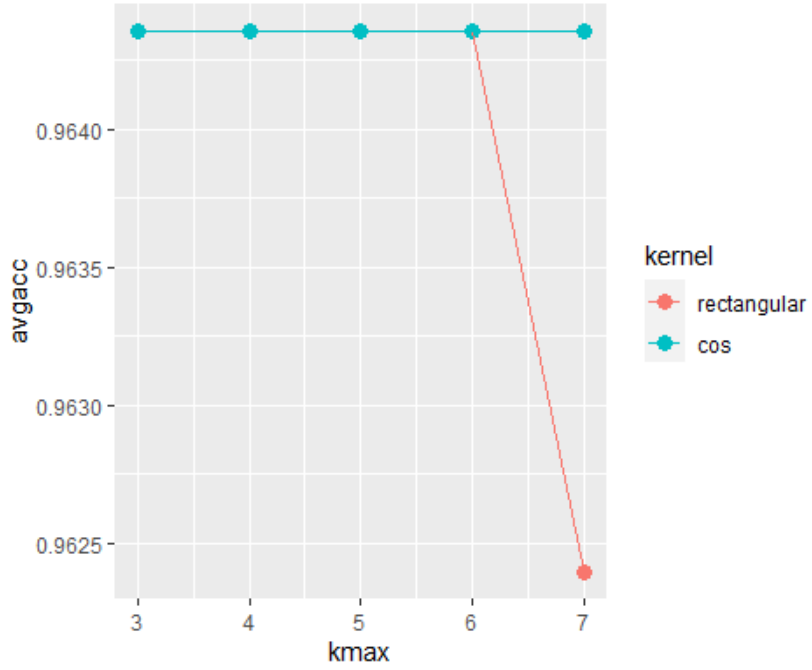
```
knn_results = kknn_fit$results # gives just the table of results by parameter
head(knn_results)
```

```
##    kmax        kernel distance  Accuracy     Kappa AccuracySD     KappaSD
## 1     3   rectangular        1 0.9774510 0.9659832 0.02912594 0.04394199
## 4     3           cos        1 0.9774510 0.9659832 0.02912594 0.04394199
## 2     3   rectangular        2 0.9551944 0.9324404 0.02368108 0.03570393
## 5     3           cos        2 0.9551944 0.9324404 0.02368108 0.03570393
## 3     3   rectangular        3 0.9604231 0.9402841 0.02737937 0.04131121
## 6    3           cos        3 0.9604231 0.9402841 0.02737937 0.04131121
```

```
# group by k and distance function, create an aggregation by averaging
knn_results <- knn_results %>%
  group_by(kmax, kernel) %>%
  mutate(avgacc = mean(Accuracy))
head(knn_results)
```

```
## # A tibble: 6 × 8
## # Groups:   kmax, kernel [2]
##    kmax kernel      distance Accuracy Kappa AccuracySD KappaSD avgacc
##   <int> <fct>          <int>    <dbl> <dbl>      <dbl>   <dbl>  <dbl>
## 1     3 rectangular        1    0.977 0.966     0.0291  0.0439  0.964
## 2     3 cos                1    0.977 0.966     0.0291  0.0439  0.964
## 3     3 rectangular        2    0.955 0.932     0.0237  0.0357  0.964
## 4     3 cos                2    0.955 0.932     0.0237  0.0357  0.964
## 5     3 rectangular        3    0.960 0.940     0.0274  0.0413  0.964
## 6     3 cos                3    0.960 0.940     0.0274  0.0413  0.964
```

```
# plot aggregated (over Minkowski power) accuracy per k, split by distance functio
n
ggplot(knn_results, aes(x=kmax, y=avgacc, color=kernel)) +
  geom_point(size=3) + geom_line()
```



Summary

The accuracy of KNN classification model on the wine data set is calculated and the highest accuracy is 97% by (k=3) model.

CLUSTERING:

Correct use of clustering and choice of parameters. Necessary preprocessing justified and executed properly.
For applying the clustering model, the Wine Dataset have been preprocessed and the Type attribute is removed(The Class Label).
Since clustering relies on distances and dissimilarities, normalizing the Wine Data Set for obtaining scaled values.
#K Means algorithm will be covered for Clustering model. Caret library doesn't provide us clustering functions, so for clustering we will use the stats library instead. Although stats package gives us the clustering functions, it lacks good visualizations and some other useful clustering functions. To use a series of different clusters to determine the optimal number of K we can use the factoextra package.

The kmeans function requires to determine the number of K beforehand. fviz_nbclust function gives access to the 2 most common ones.

The very first method is identifying the knee in a plot where you compare the wss distance.

```
#clustering - grouping of objects into diff groups of similar characteristics.
# View dataset
dfw <- Winedata
head(dfw)

##   Type Alcohol Malic_Acid  Ash Ash_Alcanity Magnesium Total_Phenols Flav
anoids
## 1    A   14.23       1.71 2.43         15.6       127          2.80    3.06
## 2    A   13.20       1.78 2.14         11.2       100          2.65    2.76
## 3    A   13.16       2.36 2.67         18.6       101          2.80    3.24
## 4    A   14.37       1.95 2.50         16.8       113          3.85    3.49
## 5    A   13.24       2.59 2.87         21.0       118          2.80    2.69
## 6    A   14.20       1.76 2.45         15.2       112          3.27    3.39
##   Nonflavanoid_Phenols Proanthocyanins Color_Intensity  Hue OD280 Proline
## 1                 0.28             2.29            5.64 1.04  3.92    1065
## 2                 0.26             1.28            4.38 1.05  3.40    1050
## 3                 0.30             2.81            5.68 1.03  3.17    1185
## 4                 0.24             2.18            7.80 0.86  3.45    1480
```

```
## 5                    0.39                1.82              4.32 1.04 2.93  735
## 6          0.34      1.97       6.75 1.05 2.85 1450
```

```
dim(Winedata)
```

```
## [1] 178  14
```

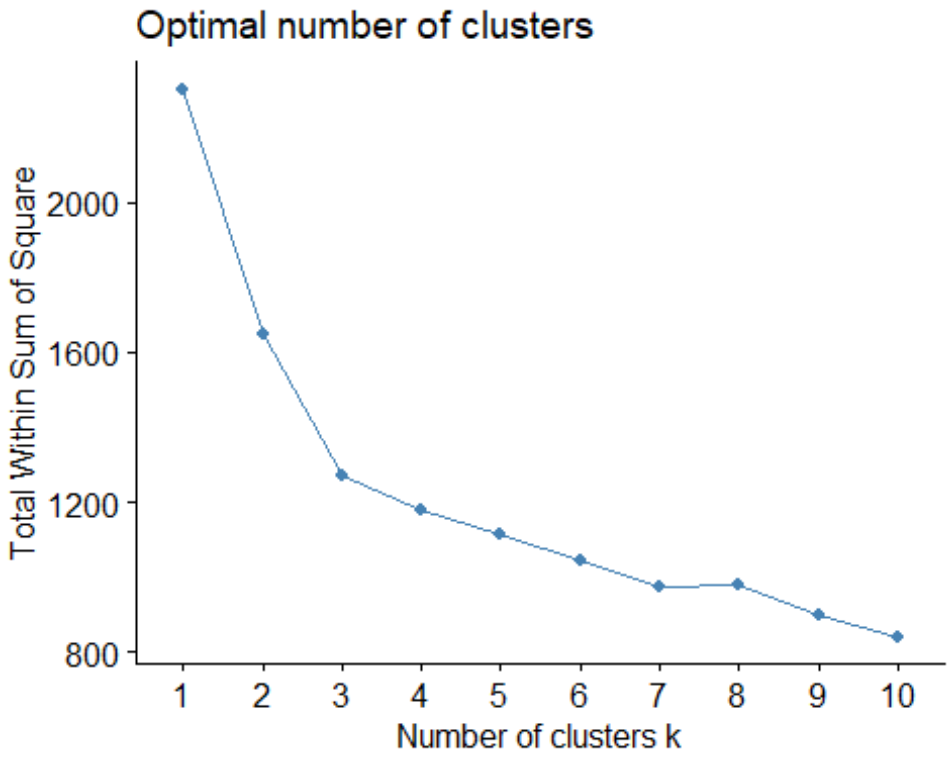
```
# Remove class labels
predictors <- dfw %>% select(-c(Type))
head(predictors)
```

```
##   Alcohol Malic_Acid  Ash Ash_Alcanity Magnesium Total_Phenols Flavanoids
## 1   14.23       1.71 2.43         15.6       127          2.80       3.06
## 2   13.20       1.78 2.14         11.2       100          2.65       2.76
## 3   13.16       2.36 2.67         18.6       101          2.80       3.24
## 4   14.37       1.95 2.50         16.8       113          3.85       3.49
## 5   13.24       2.59 2.87         21.0       118          2.80       2.69
## 6   14.20       1.76 2.45         15.2       112          3.27       3.39
##   Nonflavanoid_Phenols Proanthocyanins Color_Intensity  Hue OD280 Proline
## 1                 0.28            2.29            5.64 1.04  3.92    1065
## 2                 0.26            1.28            4.38 1.05  3.40    1050
## 3                 0.30            2.81            5.68 1.03  3.17    1185
## 4                 0.24            2.18            7.80 0.86  3.45    1480
## 5                 0.39            1.82            4.32 1.04  2.93     735
## 6          0.34      1.97       6.75 1.05 2.85 1450
```

```
# Set seed
set.seed(123)
```

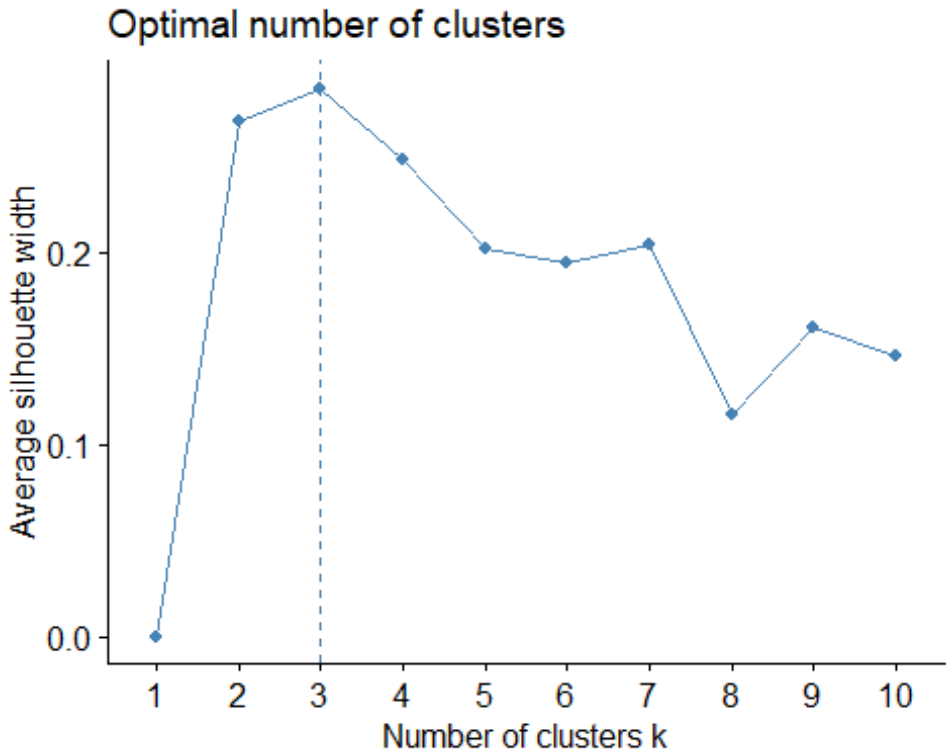
```
# Center scale allows us to standardize the data
preproc <- preProcess(predictors, method=c("center", "scale"))
# We have to call predict to fit our data based on preprocessing
predictors <- predict(preproc, predictors)
```

```
# Find the knee
fviz_nbclust(predictors, kmeans, method = "wss")
```



According to the plot there are 2 good options here. K = 2 represents the last non flat slope, or some argue that we should use K = 3 because that is the last point before the slope goes flat.The other option is comparing the average silhoutte scores of different K values. This one is more straightforward because its just expected to select the highest value.

```
fviz_nbclust(predictors, kmeans, method = "silhouette")
```



The knee suggests a K of 3 and of the silhoutte score suggests K = 3. Hence k=3.Centers parameter sets the K value and nstart determines the number of random sets to use.

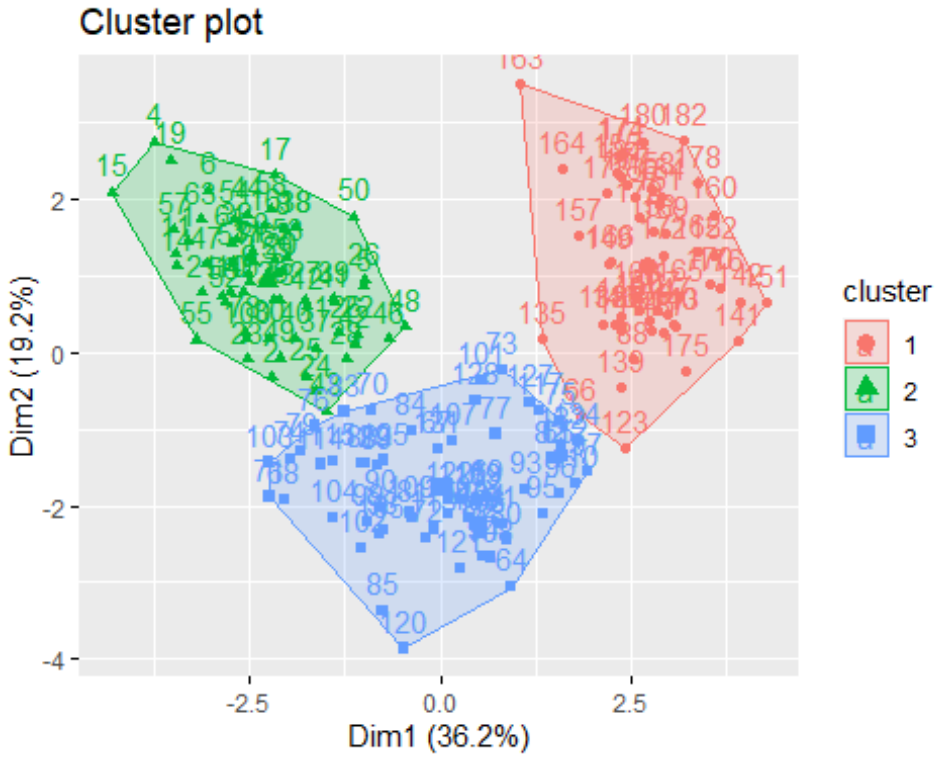
```
# Fit the data
fit <- kmeans(predictors, centers = 3, nstart = 25)
# Display the kmeans object information
fit

## K-means clustering with 3 clusters of sizes 51, 62, 65
##
## Cluster means:
##      Alcohol Malic_Acid        Ash Ash_Alcanity   Magnesium Total_Phenols
## 1  0.1644436  0.8690954  0.1863726    0.5228924 -0.07526047   -0.97657548
## 2  0.8328826 -0.3029551  0.3636801   -0.6084749  0.57596208    0.88274724
## 3 -0.9234669 -0.3929331 -0.4931257    0.1701220 -0.49032869   -0.07576891
##    Flavanoids Nonflavanoid_Phenols Proanthocyanins Color_Intensity        Hue
## 1 -1.21182921          0.72402116     -0.77751312       0.9388902 -1.1615122
## 2  0.97506900         -0.56050853      0.57865427       0.1705823  0.4726504
## 3  0.02075402         -0.03343924      0.05810161      -0.8993770  0.4605046
##       OD280    Proline
```

```
## 1 -1.2887761 -0.4059428
## 2  0.7770551  1.1220202
## 3  0.2700025 -0.7517257
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
##  21  22  23  24  25  26  27  28  29  30  31  32  37  38  39  40  41  42  43  44
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
##  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   3
##  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84
##   3   1   3   3   3   3   3   3   3   3   3   3   3   2   3   3   3   3   3   3
##  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102 103 104
##   3   3   3   1   3   3   3   3   3   3   3   3   3   3   3   2   3   3   3   3
## 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 1
23 124
##   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   1   3
## 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 1
43 144
##   3   2   3   3   3   3   3   3   3   3   1   1   1   1   1   1   1   1   1   1
## 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 1
63 164
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##
## Within cluster sum of squares by cluster:
## [1] 326.3537 385.6983 558.6971
##  (between_SS / total_SS =  44.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Using the fviz_cluster function, visualization is made on how the clusters are formed. For high dimensional data, this function uses PCA to generate two dimensions where it visualizes the wine data on the 2-dimensional plane. The plot

is formed according to the first two principal components that explain the majority of the variance.

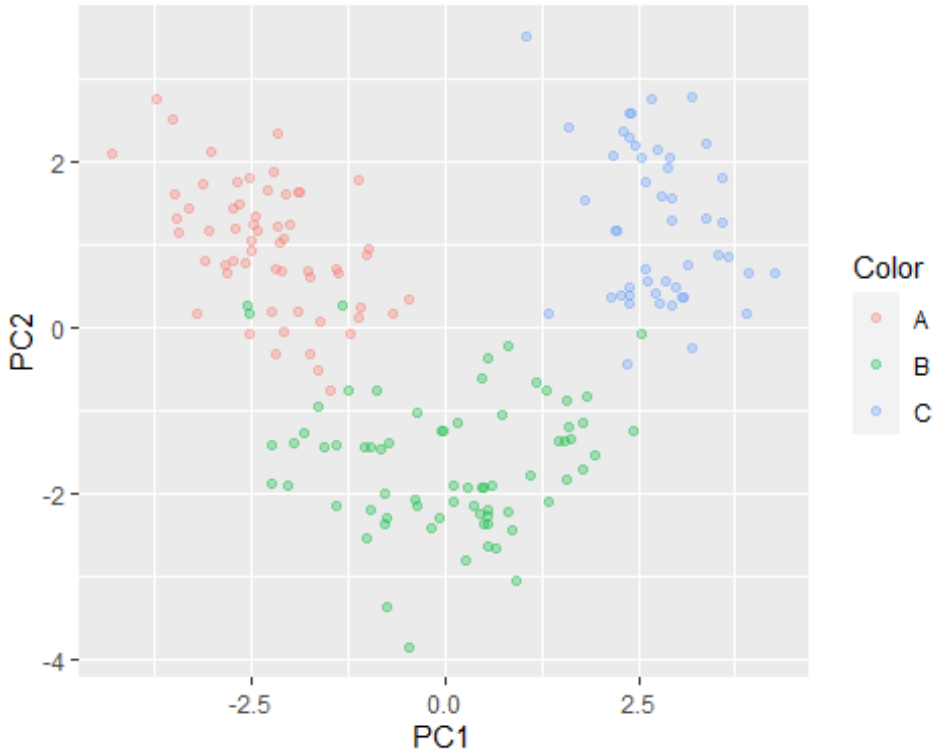
```
# Display the cluster plot
fviz_cluster(fit, data = predictors)
```



For comparison it generated a unique PCA plot and colored the points based on the respective labels.

```
# Calculate PCA
pca = prcomp(predictors)
# Save as dataframe
rotated_data = as.data.frame(pca$x)
# Add original labels as a reference
rotated_data$Color <- dfw$Type
```

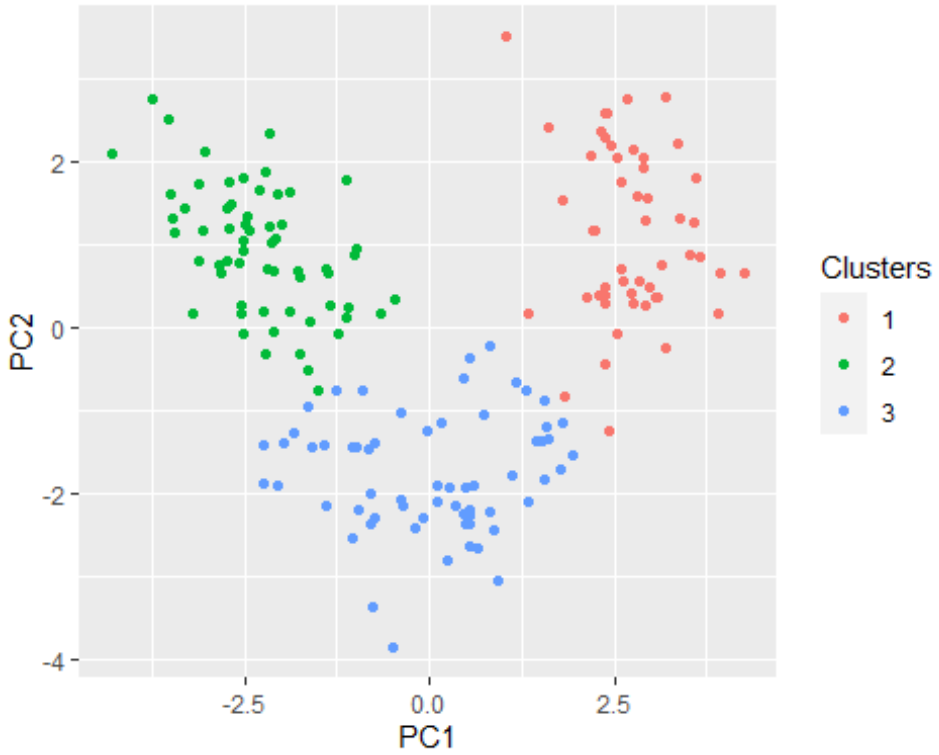
```
# Plot and color by labels
ggplot(data = rotated_data, aes(x = PC1, y = PC2, col = Color)) + geom_point(alpha = 0.3)
```



The cluster plot can also be done on ggplot based on the cluster results got from the algorithm. For K Means this is achieved by calling $cluster on the fit. This way it will get the coloring for individual points and get rid of the area markers.

```
# Assign clusters as a new column
rotated_data$Clusters = as.factor(fit$cluster)
```

```
# Plot and color by labels
ggplot(data = rotated_data, aes(x = PC1, y = PC2, col = Clusters)) + geom_point()
```



To compare the models it created a table of predictions and analyzed it.

```
# Create a dataframe
result <- data.frame(Type = dfw$Type, Kmeans = fit$cluster)
# View the first 100 cases one by one
head(result, n = 100)
```

```
##    Type Kmeans
## 1     A      2
## 2     A      2
## 3     A      2
## 4     A      2
## 5     A      2
## 6     A      2
## 7     A      2
## 8     A      2
## 9     A      2
## 10    A      2
```

```
## 11    A    2
## 12    A    2
## 13    A    2
## 14    A    2
## 15    A    2
## 16    A    2
## 17    A    2
## 18    A    2
## 19    A    2
## 20    A    2
## 21    A    2
## 22    A    2
## 23    A    2
## 24    A    2
## 25    A    2
## 26    A    2
## 27    A    2
## 28    A    2
## 29    A    2
## 30    A    2
## 31    A    2
## 32    A    2
## 37    A    2
## 38    A    2
## 39    A    2
## 40    A    2
## 41    A    2
## 42    A    2
## 43    A    2
## 44    A    2
## 45    A    2
## 46    A    2
## 47    A    2
## 48    A    2
## 49    A    2
## 50    A    2
## 51    A    2
## 52    A    2
## 53    A    2
## 54    A    2
```

```
## 55    A    2
## 56    A    2
## 57    A    2
## 58    A    2
## 59    A    2
## 60    A    2
## 61    A    2
## 62    A    2
## 63    A    2
## 64    B    3
## 65    B    3
## 66    B    1
## 67    B    3
## 68    B    3
## 69    B    3
## 70    B    3
## 71    B    3
## 72    B    3
## 73    B    3
## 74    B    3
## 75    B    3
## 76    B    3
## 77    B    3
## 78    B    2
## 79    B    3
## 80    B    3
## 81    B    3
## 82    B    3
## 83    B    3
## 84    B    3
## 85    B    3
## 86    B    3
## 87    B    3
## 88    B    1
## 89    B    3
## 90    B    3
## 91    B    3
## 92    B    3
## 93    B    3
## 94    B    3
```

```
## 95     B      3
## 96     B      3
## 97     B      3
## 98     B      3
## 99     B      3
## 100    B      2
## 101    B      3
## 102    B      3
## 103    B      3
## 104    B      3
```

Used group_by in tidyverse to create crosstab (cross tabulation of two (or more) fa ctors). Using the select pipe we subset our new dataframe to include only the colu mns we are interested in. Next we call the table function on our selection to get the table form display.

```
# Crosstab for K Means
result %>% group_by(Kmeans) %>% select(Kmeans, Type) %>% table()
```

```
##       Type
## Kmeans  A  B  C
##      1  0  3 48
##      2 59  3  0
##      3  0 65  0
```

Here K Means seemed to get relatively better clusters on the data according to both cross tabulation and the one-by-one comparison on the table. When comparison wit h the original type attribute A,B,C are done with cluster numbers 1,2,3 , it tries to fi nd out how accurate it is.
so 48 of Cs have been shown as 1 and has misidentified B as 3 , its pretty accurate, 59 of Bs have been shown as 2 and has misidentified A an Cs 2 and 3 respectively , 65 of Bs have been shown as and no misidentifications for this cluster.

The overall accuracy by Clustering is 97%

Comparison of Accuracies for both Classification and Clustering

The accuracy of KNN classification model on the wine data set is calculated and the highest accuracy is 97% by (k=3) model.

The overall accuracy by Clustering is 97%
Hence any of the above two models can be used based on the requirement of Class label.

Evaluation:

In this section calculating Correct 2x2 Confusion Matrix, Precision Recall and ROC with complete explanation of the difference.

Confusion Matrix

```
#Creating Confusion Matrix
pred_tree_ev <- predict(tree1, testdata)
cm1 <- confusionMatrix(as.factor(testdata$Type), pred_tree_ev)
cm1
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction  A  B  C
##          A 13  1  3
##          B  1 18  2
##          C  0  0 14
##
## Overall Statistics
##
##               Accuracy : 0.8654
##                 95% CI : (0.7421, 0.9441)
##    No Information Rate : 0.3654
##    P-Value [Acc > NIR] : 1.279e-13
##
```

```
##                Kappa : 0.7979
##
##  Mcnemar's Test P-Value : 0.1718
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C
## Sensitivity            0.9286   0.9474   0.7368
## Specificity            0.8947   0.9091   1.0000
## Pos Pred Value         0.7647   0.8571   1.0000
## Neg Pred Value         0.9714   0.9677   0.8684
## Prevalence             0.2692   0.3654   0.3654
## Detection Rate         0.2500   0.3462   0.2692
## Detection Prevalence   0.3269   0.4038   0.2692
## Balanced Accuracy      0.9117   0.9282   0.8684
```

```
# Store the byClass object of confusion matrix as a dataframe
metrics <- as.data.frame(cm1$byClass)
# View the object
metrics
```

```
##          Sensitivity Specificity Pos Pred Value Neg Pred Value Precision
## Class: A   0.9285714   0.8947368      0.7647059      0.9714286 0.7647059
## Class: B   0.9473684   0.9090909      0.8571429      0.9677419 0.8571429
## Class: C   0.7368421   1.0000000      1.0000000      0.8684211 1.0000000
##             Recall       F1 Prevalence Detection Rate Detection Prevalence
## Class: A 0.9285714 0.8387097  0.2692308      0.2500000            0.3269231
## Class: B 0.9473684 0.9000000  0.3653846      0.3461538            0.4038462
## Class: C 0.7368421 0.8484848  0.3653846      0.2692308            0.2692308
##          Balanced Accuracy
## Class: A         0.9116541
## Class: B         0.9282297
## Class: C         0.8684211
```

Precision and Recall

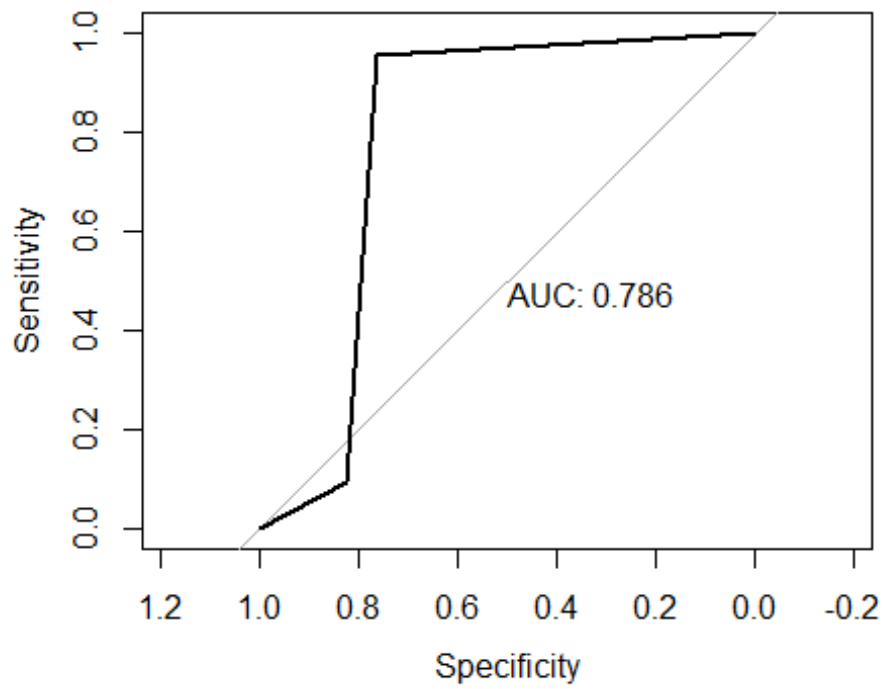
```
# Get the precision value for each class
metrics %>% select(c(Precision))
```

```
##          Precision
## Class: A 0.7647059
```

```
## Class: B 0.8571429
## Class: C 1.0000000

# Get the recall value for each class
metrics %>% select(c(Recall))

##              Recall
## Class: A 0.9285714
## Class: B 0.9473684
## Class: C 0.7368421
```

Ploting ROC

```
library(pROC)

roctree<-roc(response=testdata$Type,predictor=as.numeric(pred_tree_ev))

plot(roctree, print.auc=TRUE)
```



REFLECTION:

The most useful lessons from this course are investigating various sorts of datasets, data preprocessing, various methods for cleaning and addressing missing values with numerous techniques, including binning, smoothing, normalizing, and many others. Additionally, Predicting the values of labels with known values using various "classification" techniques Machine learning with SVM, Decision Tree, and KNN parameters. Moreover, learnt to employ "clustering" and models like k-means and other techniques, one can forecast the values of unknown labels. To deal with prejudice and variance in class imbalance, I learned advance evaluation. In addition, with Knowing the error rate of a model and using accuracy, recall, and ROC can be highly beneficial.

Applying Data Science Using R Programming helped me gain more exposure on the precision of the field and elevated my interest in the course further.

Resources:

Dataset obtained : UCI Machine Learning Repository: Wine Data Set