# FUNDAMENTALS OF DATA SCIENCE – DSC 441

## HOMEWORK 5 Final Project

### WINE DATASET

**Jashwanth Neeli - 2022-11-16**

(Technical Report Fundamentals of Data Science - Wine Dataset Jashwanth Neeli 2022-11-11

## Introduction:

To evaluate and compare 2 different classification models on Wine prediction dataset from the UCI dataset library. According to the source, the dataset is a *result of a chemical analysis of wines grown in the same region in Italy but* ***derived from three different cultivars.*** Hence, given a set of features, we need to determine from which of the **three cultivators the wine has come from.** Evidently, it is a classification problem.

## SUMMARY

The wine data set includes 16 different wine attributes, such as alcohol and Malic acid content, that were measured for 182 wine samples. These wines were grown in the same region of Italy but were derived from three different cultivars, resulting in three distinct wine classes. The goal here is to find a model that can predict the wine class based on the 16 measured parameters and determine the major differences between the three classes. This is a classification and clustering problem, and I will describe Three models and evaluate their accuracy. In addition, I will use principal component analysis to identify and investigate the differences between the three classes.

### Details of the Experiment

In this experiment I am following the seven data mining techniques Data gathering and integration, Data Exploration, Data Cleaning, Data Preprocessing, Clustering, Classification and Data Evaluation. I am using data Visualisation using ggplot to find the distributions between the variables. I am analysing by visualising all the numerical distributions. Finding the correlation among the variables and then in data mining techniques I am using clustering (Kmeans) and classification (SVM and KKNN). To determine which technique gives the best accuracy and which is the best fit model for the experiment. Finally, I am using data Evaluation to check or visualize the performance of the multi-class classification problem. I am using ROC curve metrics for checking any classification model's performance.

1. Data gathering and integration
2. Data Exploration

3. Data Preprocessing
4. Clustering
5. Classification
6. Data Evaluation

**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

*after Exploratory data analysis, that a feature is being too* noisy *or highly correlated to some other feature, simply remove it.*

## Data gathering and integration

I'll be using R Studio for this. It would be easy, however, to form a similar analysis in R.

## Let's import the requisite libraries:

#a. Data gathering and integration The first part is to get the data you will use. #You may use anything that has not been used in an assignment or tutorial. #It must have at least 100 data points and must include both numerical and categorial (or ordinal) variables. I recommend keeping this relatively straightforward because data cleaning can take a lot of time if you choose a large, messy dataset. Kaggle (https://www.kaggle.com/datasets) and the University of California at Irvine (UCI) (https://archive.ics.uci.edu/ml/index.php) maintain collections of datasets, some even telling you if they are good examples for testing specific machine learning techniques. You may also choose to join together more than one dataset, for example to merge data on health outcomes by US state with a dataset on food statistics per state. Merging data is not required and will earn you a bonus point in this step.

```
#install package called Party, other types of classification methods are available as well, rpart etc.
#party package allows us to do conditional inference tree(c-tree)
#call libraries
library(sandwich)

## Warning: package 'sandwich' was built under R version 4.2.2

library(zoo)

## Warning: package 'zoo' was built under R version 4.2.2

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

library(party)

## Warning: package 'party' was built under R version 4.2.2

## Loading required package: grid

## Loading required package: mvtnorm

## Loading required package: modeltools

## Loading required package: stats4

## Loading required package: strucchange

## Warning: package 'strucchange' was built under R version 4.2.2

#import the data set wine
#make sure you click yes for headings, strings as factors checkmark
 Winedata <-
read.csv("C:/Depaul_1st_quarter_subject/Fundamentals_of_DS/Week10/wine_dset1.csv", stringsAsFactors=TRUE)
```

### b. Data Exploration

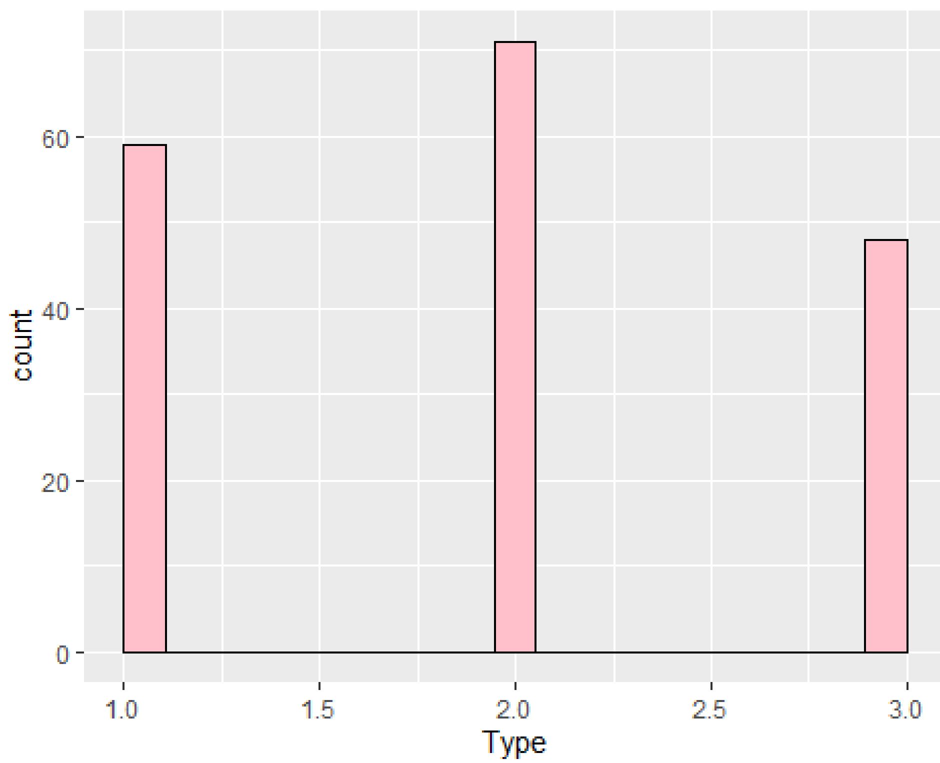
Using data exploration to understand what is happening is important throughout the pipeline, and is not limited to this step. However, it is important to use some exploration early on to make sure you understand your data. #You must at least consider the distributions of each variable and at least some of the relationships between pairs of variables.

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

## [1] 178  16

View( Winedata)

str(Winedata)

## 'data.frame':    178 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/ 2 levels "AA3V","CC5G": 1 2 2 2 2 2 2 2
2 2 ...

dim(Winedata)

## [1] 178  16
```

```
nrow(Winedata)

## [1] 178

ncol(Winedata)

## [1] 16

View(data.frame(sapply(Winedata, class)))
```

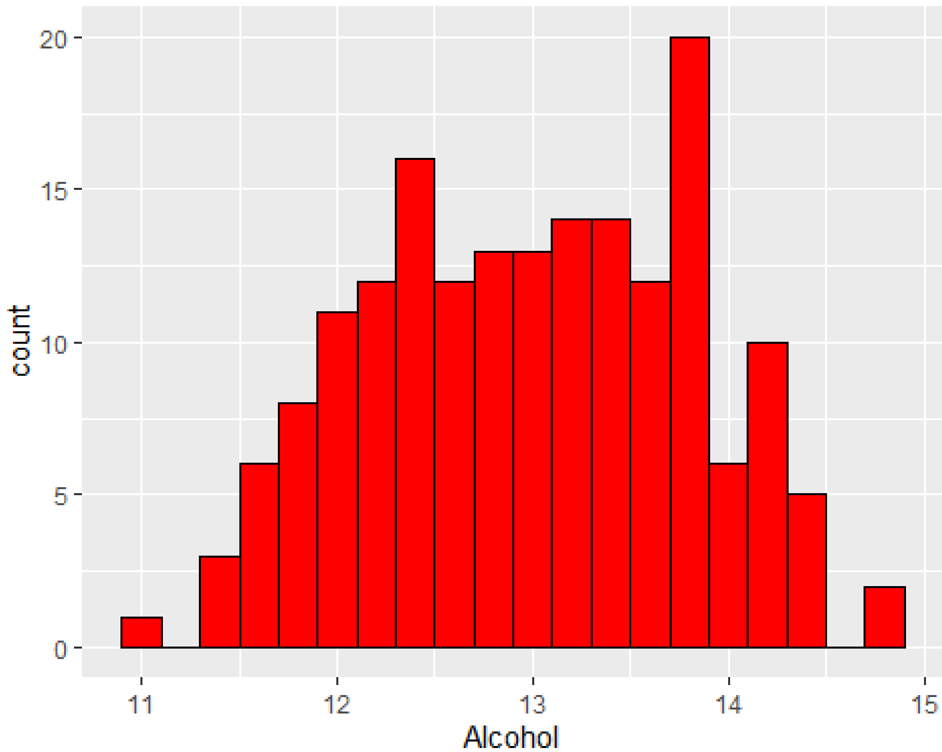
#VISUALIZING THE DATA: The following graphs visualizes each numerical column distributions :

```
#Visualization: Numerical data - Type
library(ggplot2)
summary(Winedata$Type)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   2.000   1.938   3.000   3.000

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

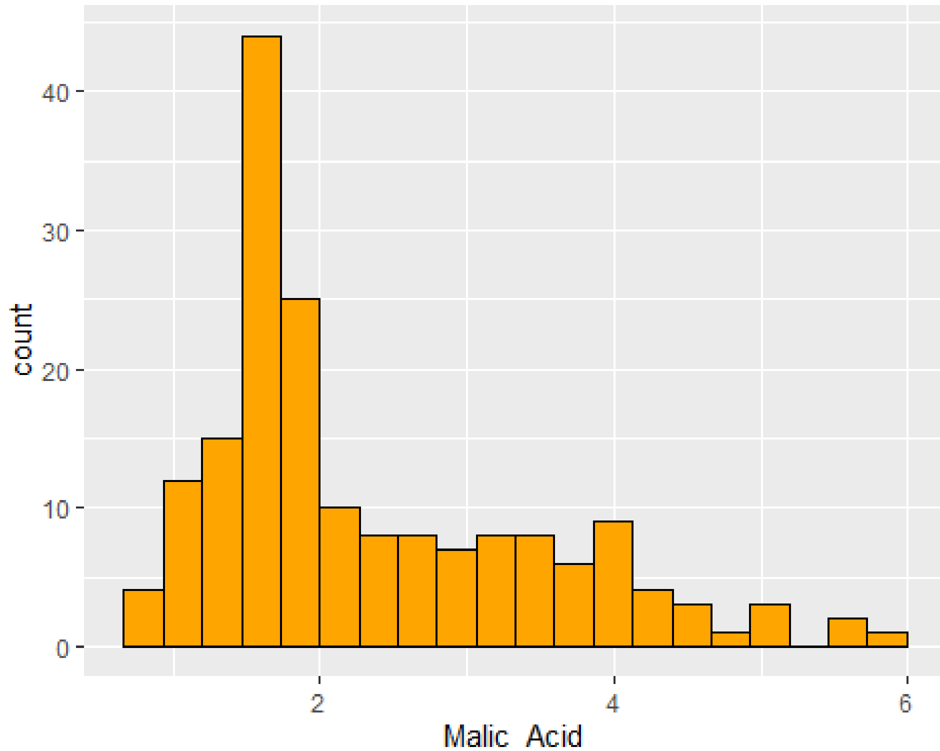



```
#Visualization: Numerical data - Alcohol
summary(Winedata$Alcohol)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   11.03   12.36   13.05   13.00   13.68   14.83
```

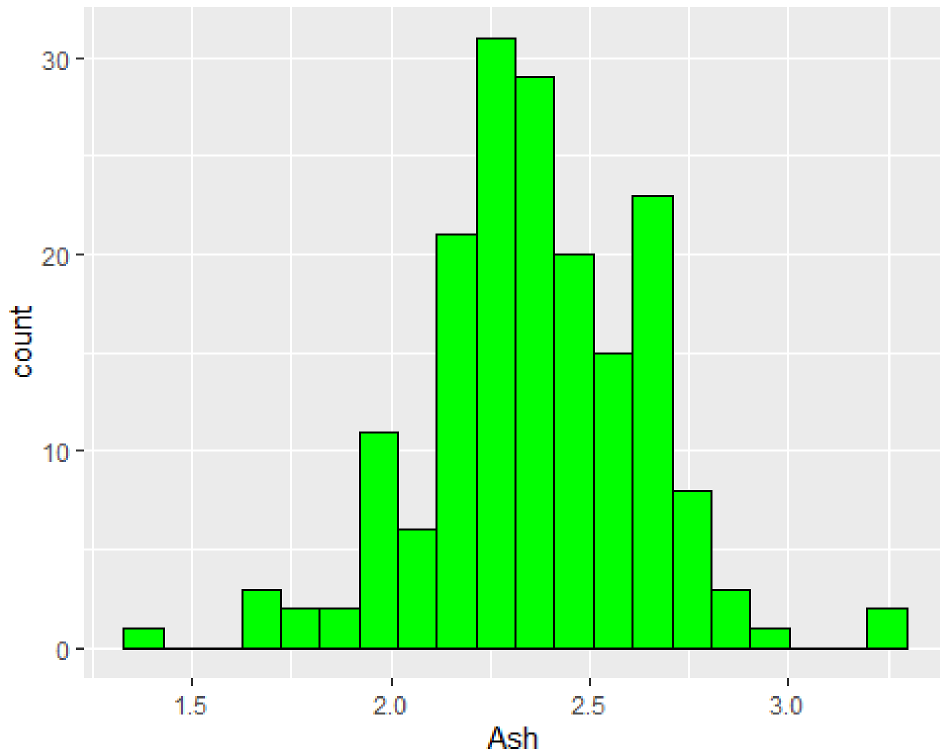
```r
ggplot (Winedata, aes (Alcohol)) + geom_histogram( fill='red',color="black" ,
bins = 20)
```



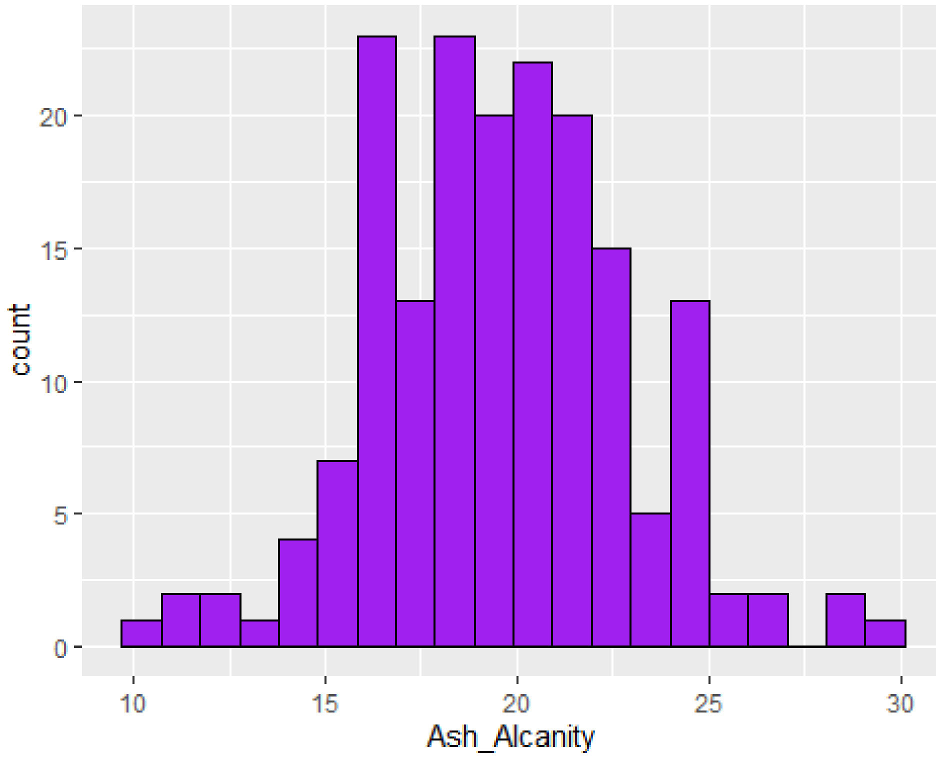

```r
#Visualization: Numerical data - Malic_Acid
summary(Winedata$Malic_Acid)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.740   1.603   1.865   2.336   3.083   5.800

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



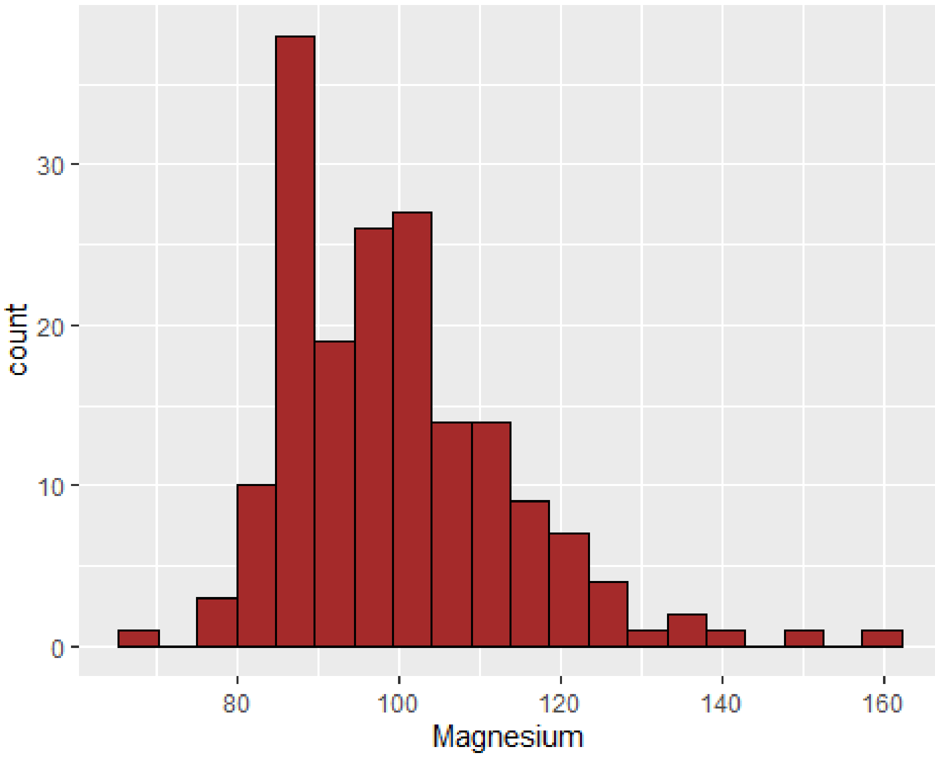

```
#Visualization: Numerical data - Ash
summary(Winedata$Ash)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.360   2.210   2.360   2.367   2.558   3.230

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.
##   10.60   17.20   19.50   19.49   21.50   30.00

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



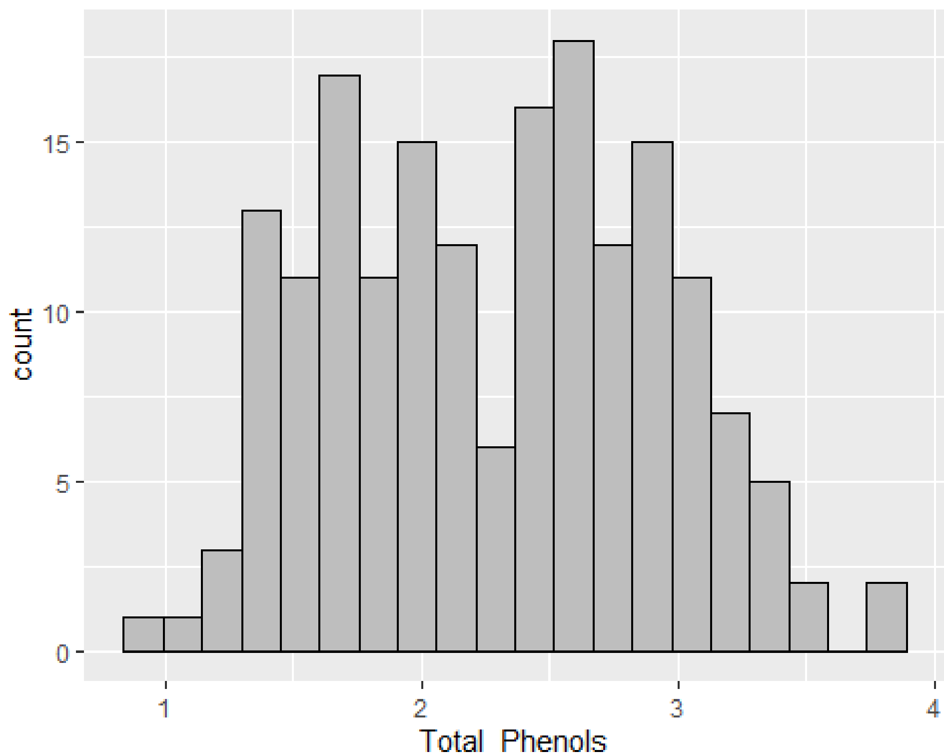

```
#Visualization: Numerical data - Magnesium
summary(Winedata$Magnesium)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   70.00   88.00   98.00   99.74  107.00  162.00

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



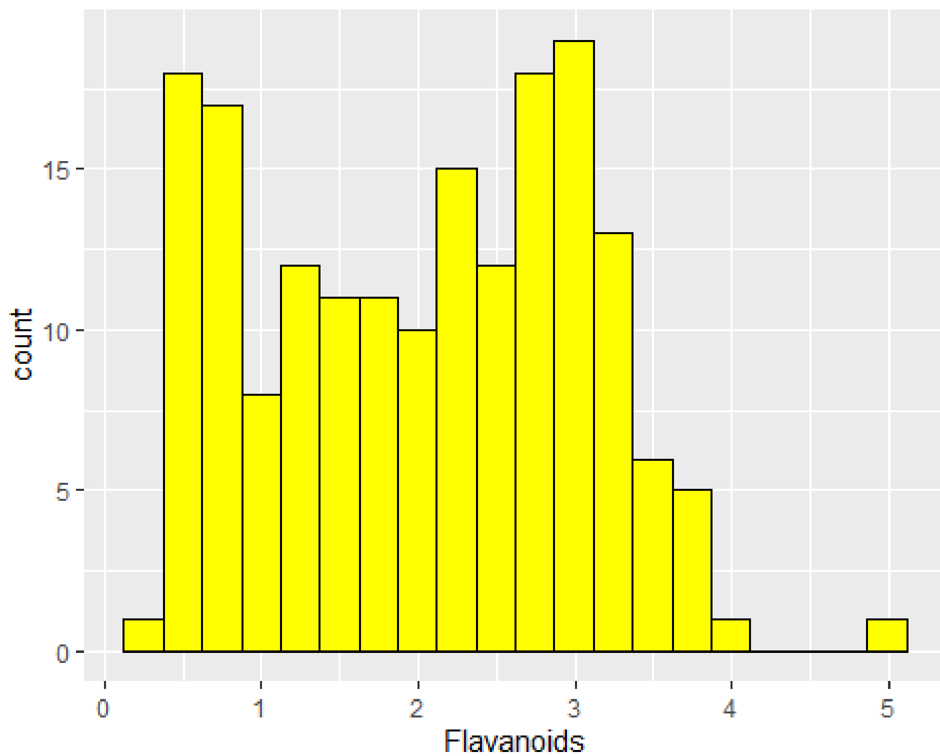

```
#Visualization: Numerical data - Total_Phenols
summary(Winedata$Total_Phenols)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.980   1.742   2.355   2.295   2.800   3.880
```

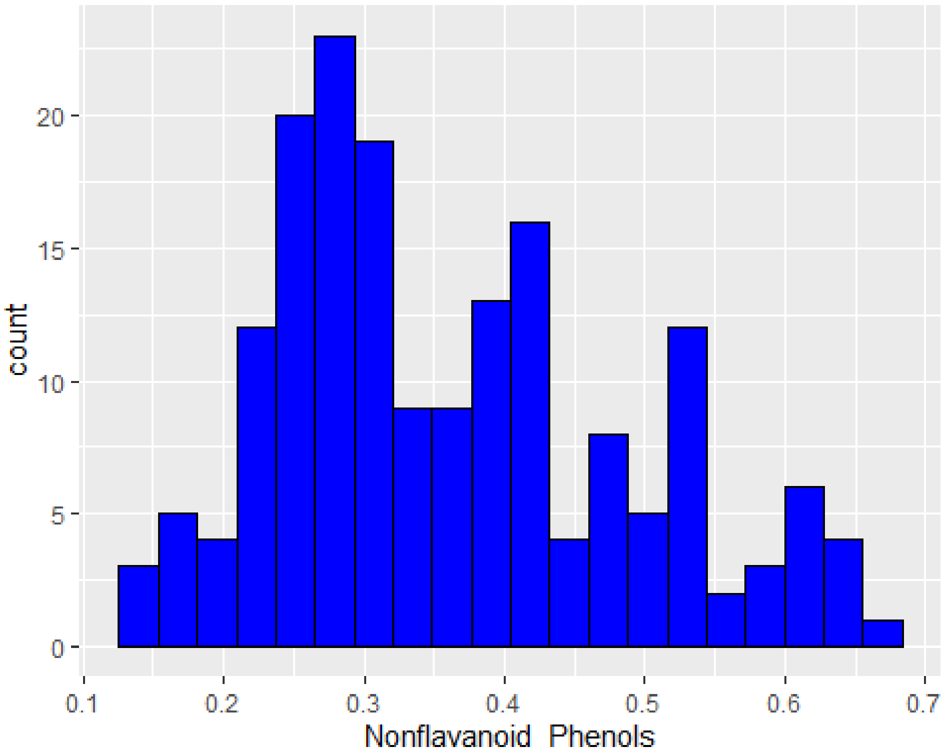
```
ggplot (Winedata, aes (Total_Phenols)) + geom_histogram(
fill='grey',color="black" , bins = 20)
```



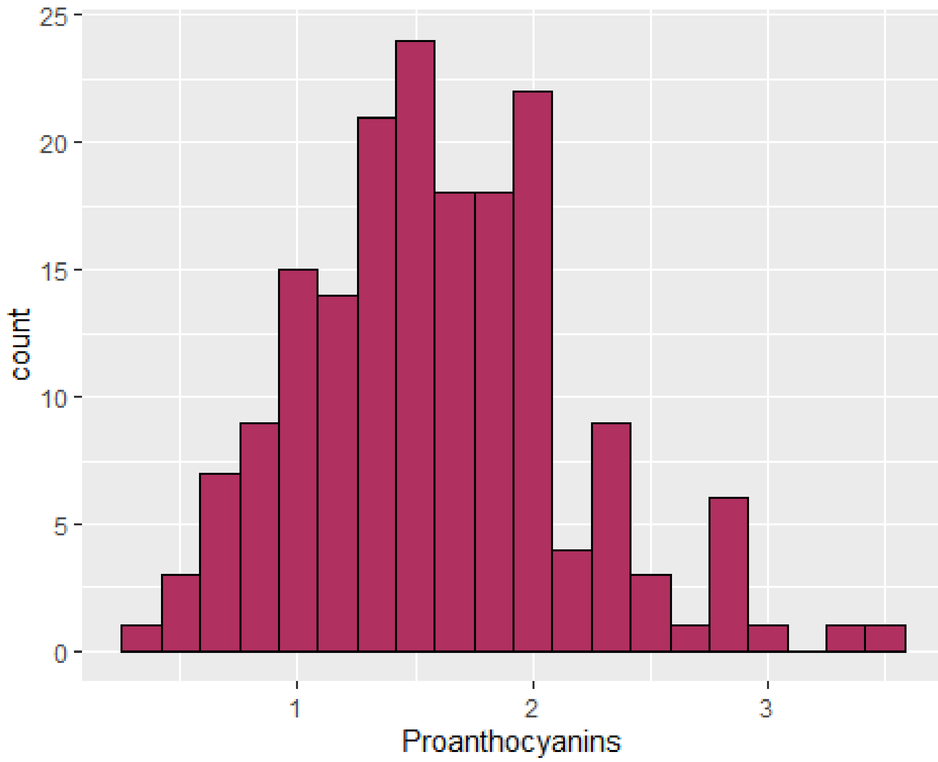

```
#Visualization: Numerical data - Flavanoids
summary(Winedata$Flavanoids)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.340   1.205   2.135   2.029   2.875   5.080

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



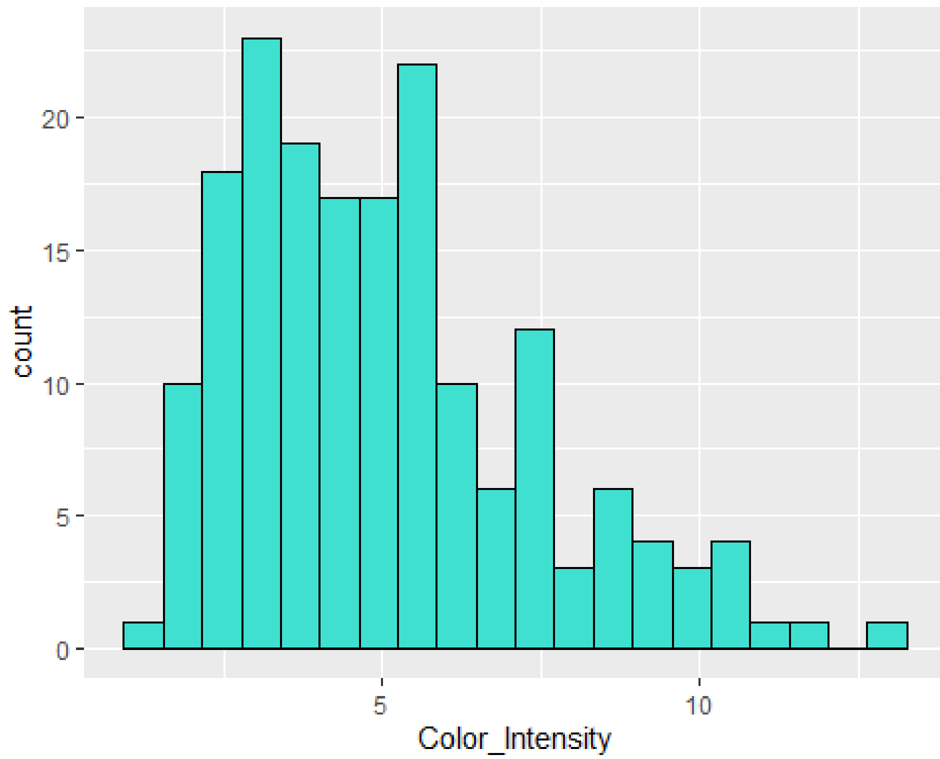

```
#Visualization: Numerical data - Nonflavanoid_Phenols
summary(Winedata$Nonflavanoid_Phenols)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1300  0.2700  0.3400  0.3619  0.4375  0.6600

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



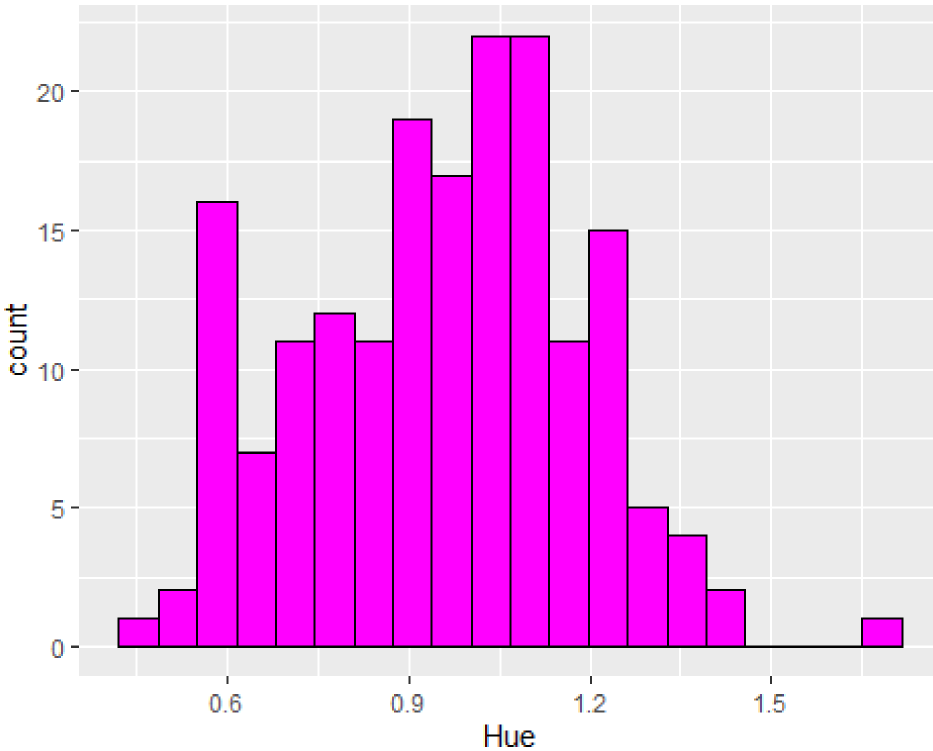

```
#Visualization: Numerical data - Proanthocyanins
summary(Winedata$Proanthocyanins)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.410   1.250   1.555   1.591   1.950   3.580

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



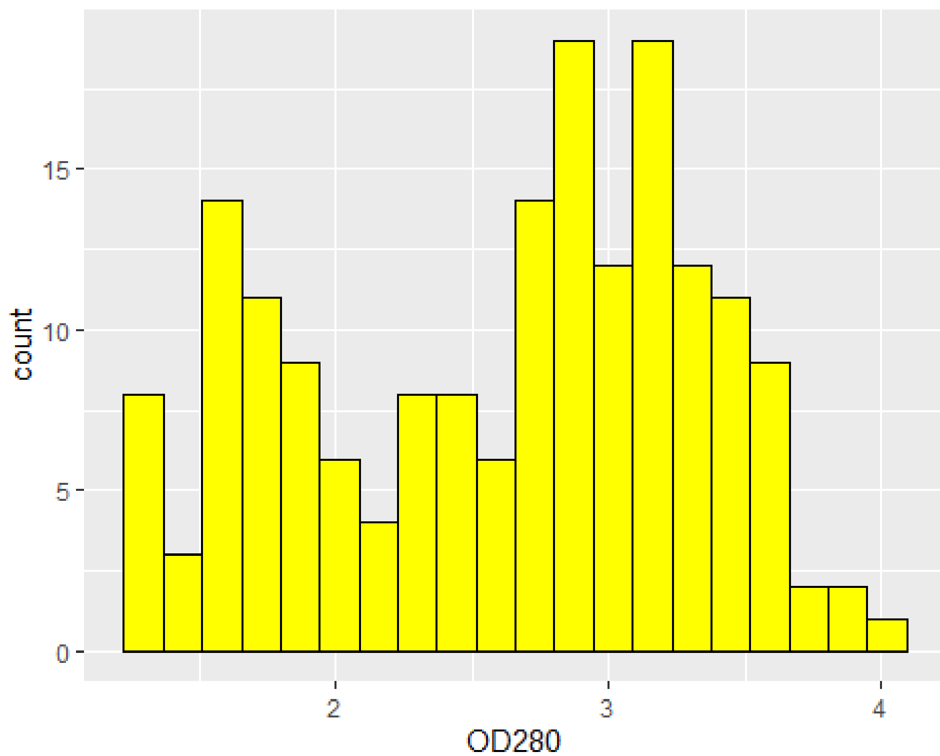

```
#Visualization: Numerical data - Color_Intensity
summary(Winedata$Color_Intensity)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   1.280   3.220   4.690   5.058   6.200  13.000

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



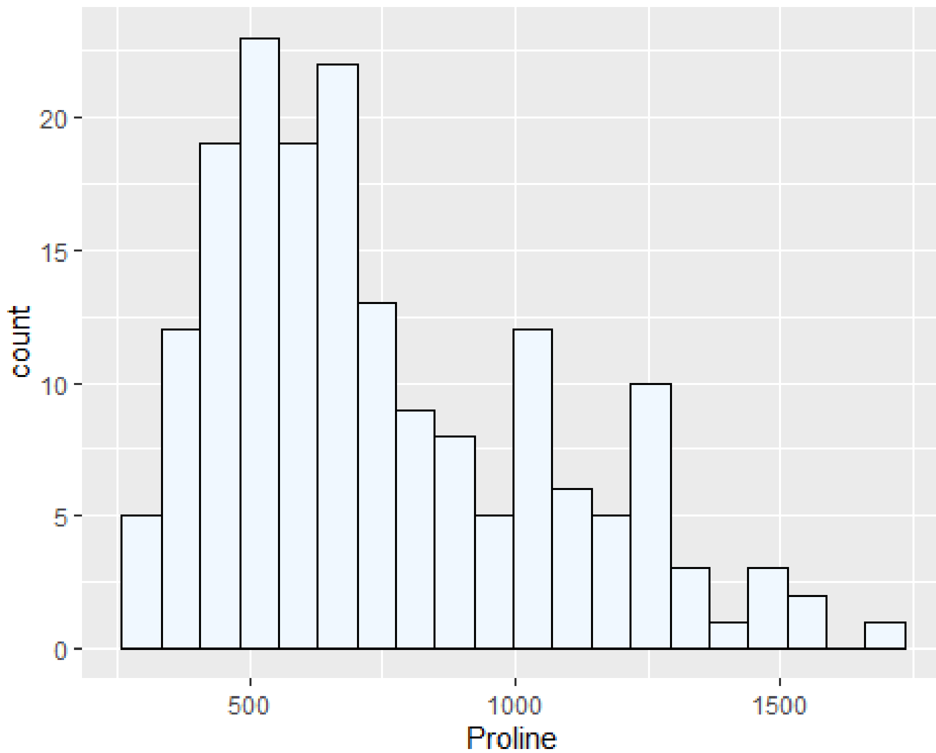

```
#Visualization: Numerical data - Hue
summary(Winedata$Hue)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.4800  0.7825  0.9650  0.9574  1.1200  1.7100

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



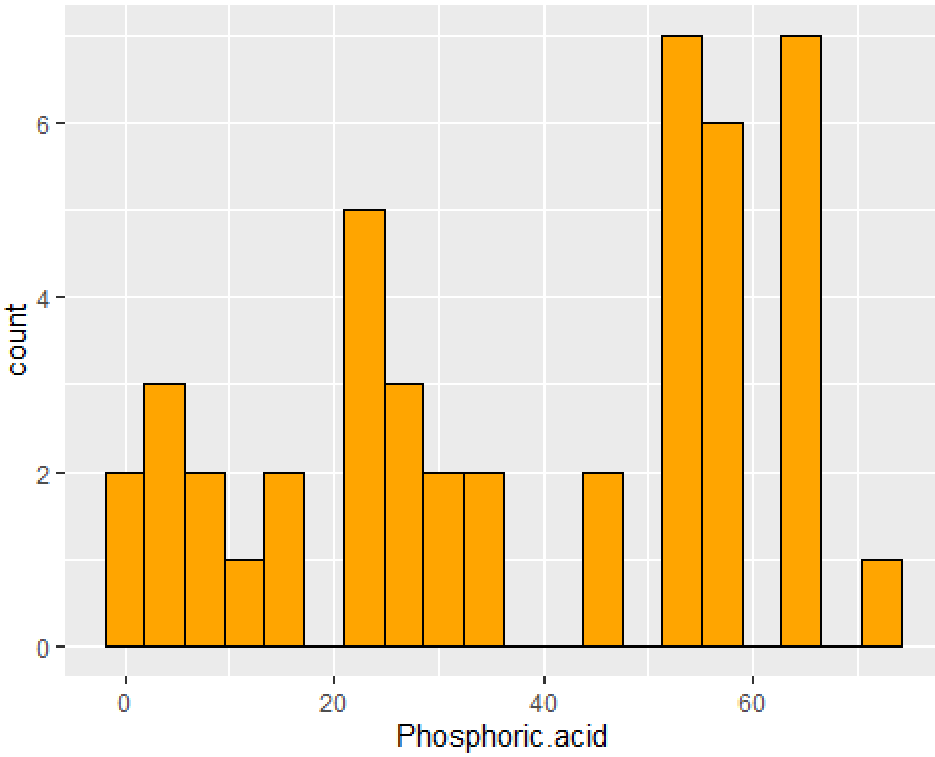

```
#Visualization: Numerical data -OD280
summary(Winedata$OD280)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   1.270   1.938   2.780   2.612   3.170   4.000

ggplot (Winedata, aes (OD280)) + geom_histogram( fill='yellow',color="black"
, bins = 20)
```




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

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

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

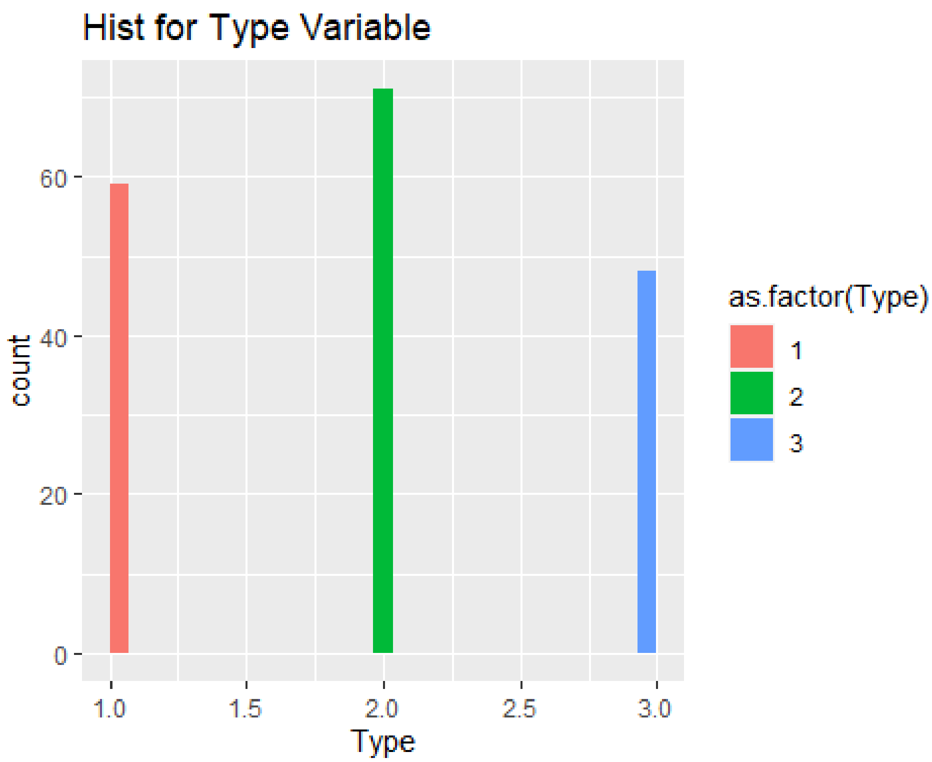



```
#Visualization: Numerical data -Phosphoric.acid

summary(Winedata$Phosphoric.acid)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##    1.23   22.50   45.80   38.27   55.90   73.50     133

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

## Warning: Removed 133 rows containing non-finite values (stat_bin).
```



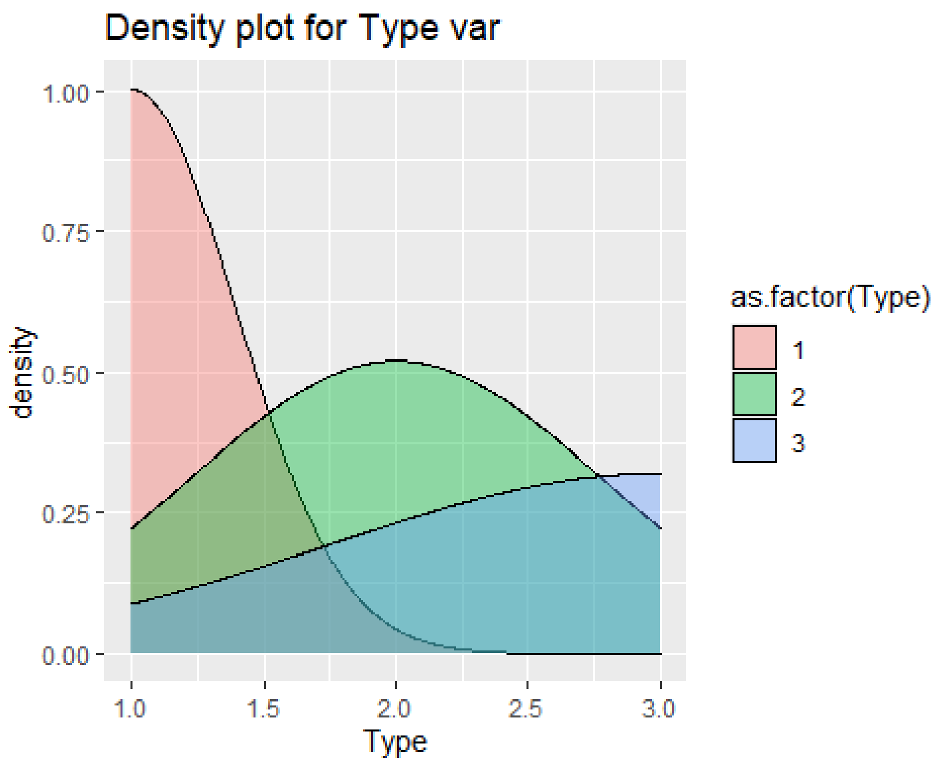

#ANALYSIS OF DATA USING VISUALIZATION This is a histogram that gives the occurrences of the “Type”:

```
ggplot(data=Winedata,aes(x=Type,fill=as.factor(Type)))+geom_histogram()+
labs(title = "Hist for Type Variable")

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



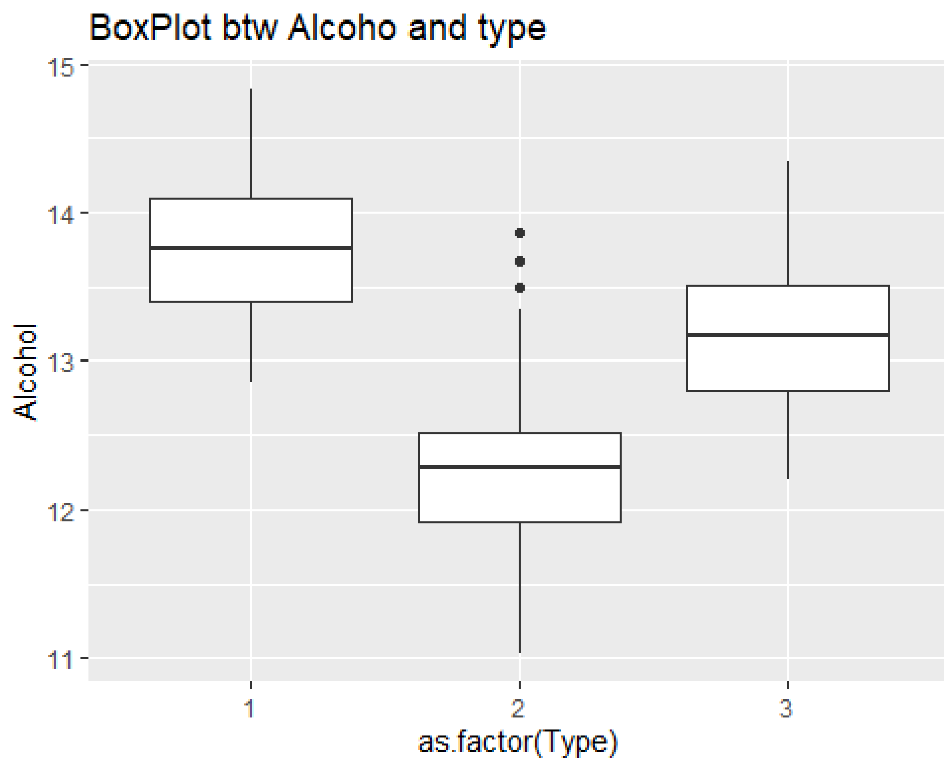

This is a density plot that tells us again about the Type Factor visually:

```
ggplot(data=Winedata,aes(x=Type,fill=as.factor(Type)))+geom_density(alpha=0.4
)+ labs(title = "Density plot for Type var")
```



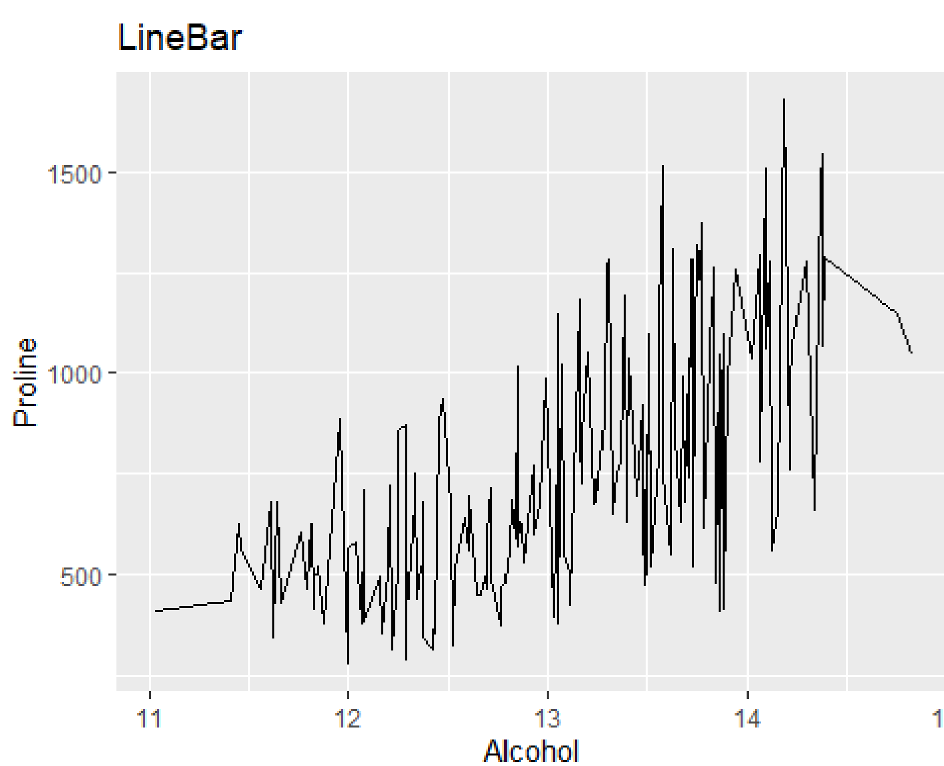

This is a box-plot that generates a graph between Type and Alcohol factors in the data set:

```
ggplot(data=Winedata,aes(x=as.factor(Type),y=Alcohol))+geom_boxplot()+labs(title = "BoxPlot btw Alcoho and type")
```



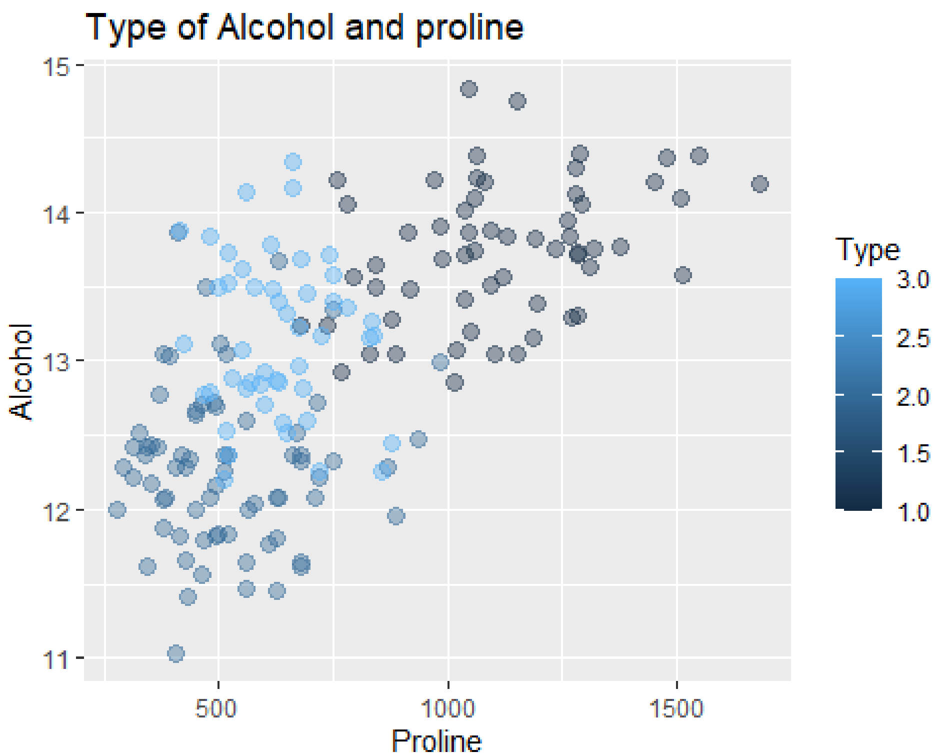

This is a line bar that generates a graph between Alcohol and Proline from our dataset:

```
ggplot(data=Winedata,aes(x=Alcohol,y=Proline))+geom_line()+ labs(title =
"LineBar")
```



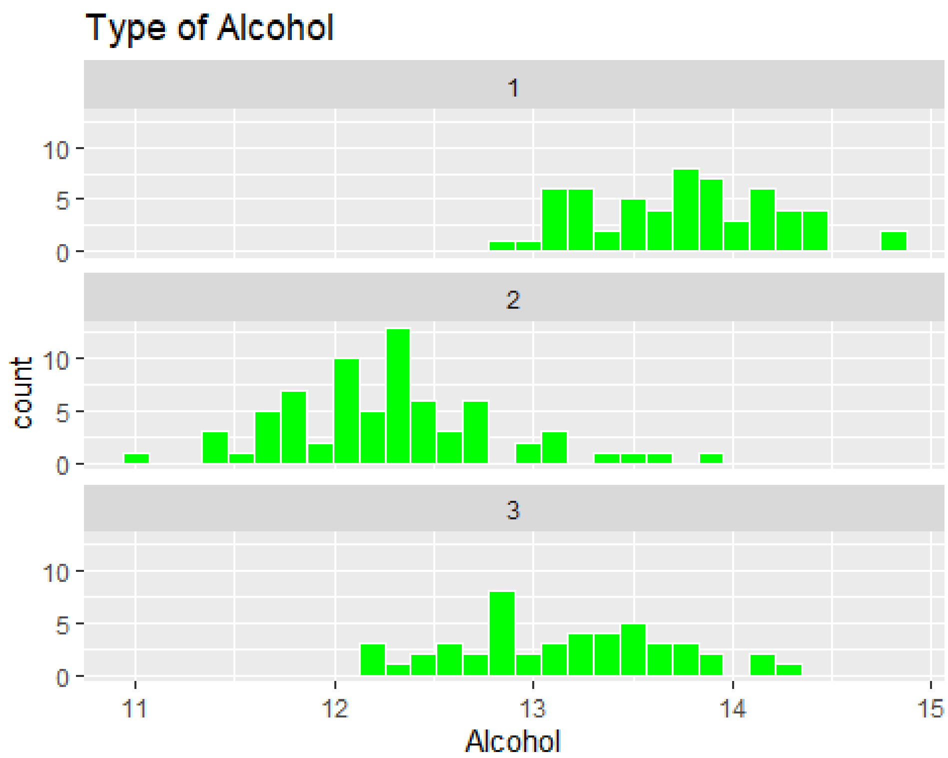
Now we make a scatterplot between Proline , Alcohol and Output factors of our dataset :

```
ggplot(data =
Winedata,aes(x=Proline,y=Alcohol,color=Type))+geom_point(alpha=0.4,size=3)+la
bs(title = "Type of Alcohol and proline")
```




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

```
ggplot(Winedata, aes(x = Alcohol)) + geom_histogram(fill = "green", color =
"white") + facet_wrap(~Type, ncol = 1)+ labs(title = "Type of Alcohol")

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




```
wine_df <- Winedata
```

#c. Data Cleaning and PreProcessing

as per plan

1)create an extra columns in the dataset done

2)pollute the data with NAs and factors done

3)remove NAs

4)remove column - 70% is null hence removing the column

5)bin/smoothing

6)num to categorical

7)summarize after cleaning

8)vizualizaing after cleaning

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

## [1] 133

View(wine_df)

summary(wine_df)
```

```
##       Type          Alcohol        Malic_Acid         Ash       
##  Min.   :1.000   Min.   :11.03   Min.   :0.740   Min.   :1.360  
##  1st Qu.:1.000   1st Qu.:12.36   1st Qu.:1.603   1st Qu.:2.210  
##  Median :2.000   Median :13.05   Median :1.865   Median :2.360  
##  Mean   :1.938   Mean   :13.00   Mean   :2.336   Mean   :2.367  
##  3rd Qu.:3.000   3rd Qu.:13.68   3rd Qu.:3.083   3rd Qu.:2.558  
##  Max.   :3.000   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       Phosphoric.acid Wine_Model
##  Min.   :1.270   Min.   : 278.0   Min.   : 1.23   AA3V:  1  
##  1st Qu.:1.938   1st Qu.: 500.5   1st Qu.:22.50   CC5G:177  
##  Median :2.780   Median : 673.5   Median :45.80             
##  Mean   :2.612   Mean   : 746.9   Mean   :38.27             
##  3rd Qu.:3.170   3rd Qu.: 985.0   3rd Qu.:55.90             
##  Max.   :4.000   Max.   :1680.0   Max.   :73.50             
##                                   NA's   :133               
```

```
dim(wine_df)
```

```
## [1] 178  16
```

```
wine_df$Wine_Model <- NULL

#str(Winedata$Phosphoric.acid)
wine_df$Phosphoric.acid <- as.numeric(wine_df$Phosphoric.acid)

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

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

```r
complete.cases(wine_df)
```

```
##   [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
TRUE TRUE
##  [16] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
TRUE TRUE
##  [31] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
TRUE TRUE
##  [46] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
TRUE TRUE
##  [61] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
TRUE TRUE
##  [76] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
TRUE TRUE
##  [91] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
TRUE TRUE
## [106] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
TRUE TRUE
## [121] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
TRUE TRUE
## [136] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
TRUE TRUE
## [151] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
TRUE TRUE
## [166] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```r
wine_df <- wine_df[complete.cases(wine_df),]
sum(is.na(wine_df))
```

```
## [1] 0
```

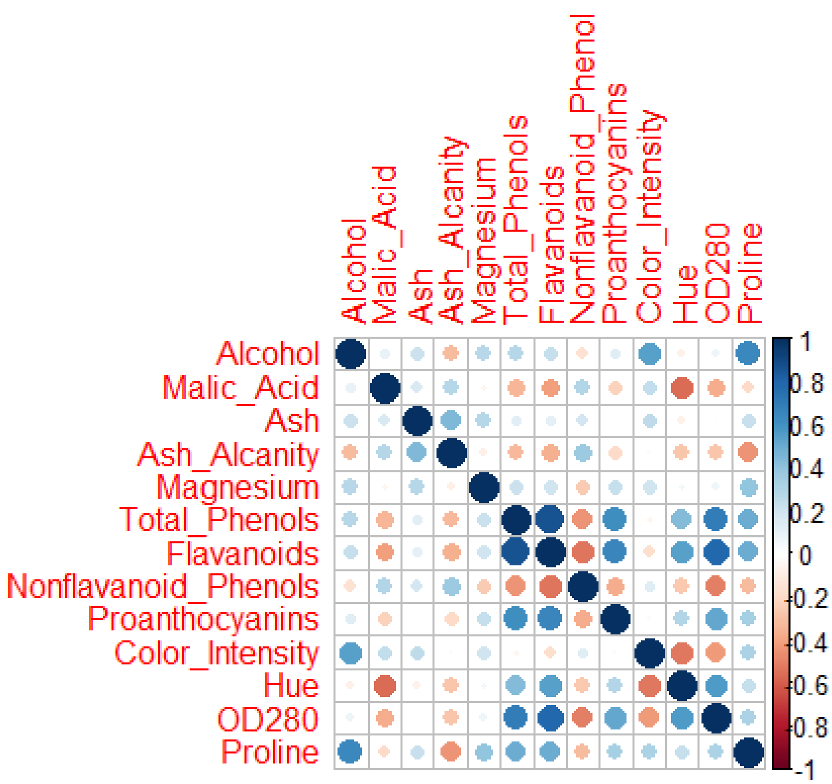
```r
summary(wine_df)
```

```
##      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
```

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

```
library(corrplot)

## Warning: package 'corrplot' was built under R version 4.2.2

## corrplot 0.92 loaded

#visualize correlation matrix

corr_var <- dplyr::select(wine_df, -Type)
#View(corr_var)
corrplot(cor(corr_var))
```



#**Data PreProcessing**

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

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

str(wine_df)

## 'data.frame':    178 obs. of  14 variables:
##  $ Type                : chr  "A" "A" "A" "A" ...
##  $ 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 ...

library(caret)

## Loading required package: lattice

#min max
#summary before normalization
summary(wine_df$Proline)

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

##     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##      0.0       0.0       5.0     998.6     399.0 100000.0
#applying normalization
file_mm <- wine_df[c(14)]
```

```r
preproc_mm <- preProcess(file_mm, method=c("range"))
norm_mm <- predict(preproc_mm, file_mm)
# the values range [0-1]
#summary after normalization
summary(norm_mm)
```

```
##     Proline
##  Min.   :0.0000
##  1st Qu.:0.1587
##  Median :0.2821
##  Mean   :0.3344
##  3rd Qu.:0.5043
##  Max.   :1.0000
```

```r
summary(wine_df)
```

```
##      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
```

```r
summary(Winedata)
```

```
##       Type          Alcohol        Malic_Acid         Ash
##  Min.   :1.000   Min.   :11.03   Min.   :0.740   Min.   :1.360
##  1st Qu.:1.000   1st Qu.:12.36   1st Qu.:1.603   1st Qu.:2.210
##  Median :2.000   Median :13.05   Median :1.865   Median :2.360
```

```
## Mean   :1.938   Mean   :13.00   Mean   :2.336   Mean   :2.367
## 3rd Qu.:3.000   3rd Qu.:13.68   3rd Qu.:3.083   3rd Qu.:2.558
## Max.   :3.000   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       Phosphoric.acid Wine_Model
## Min.   :1.270   Min.   : 278.0   Min.   : 1.23   AA3V:  1
## 1st Qu.:1.938   1st Qu.: 500.5   1st Qu.:22.50   CC5G:177
## Median :2.780   Median : 673.5   Median :45.80
## Mean   :2.612   Mean   : 746.9   Mean   :38.27
## 3rd Qu.:3.170   3rd Qu.: 985.0   3rd Qu.:55.90
## Max.   :4.000   Max.   :1680.0   Max.   :73.50
##                                  NA's   :133
```

***clustering*** ***- grouping of objects into diff groups of similar*** `characteristics`.

```
library(stats)
library(factoextra)

## Warning: package 'factoextra' was built under R version 4.2.2

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

library(ggplot2)
library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.2.2

## ── Attaching packages
## ───────────────────────────────────────
## tidyverse 1.3.2 ──

## ✓ tibble  3.1.8      ✓ dplyr   1.0.10
## ✓ tidyr   1.2.1      ✓ stringr 1.4.1
```

Correct clustering and parameter selection. Preprocessing is required.

justified and carried out correctly The Wine Dataset was preprocessed and prepared for use with the clustering model. The Type attribute has been removed ( The Class Label ). Because clustering is based on  distances and dissimilarities, normalizing the Wine Data is necessary. Set in order to obtain scaled values.

For the Clustering model, the #K Means algorithm will be covered. There is no Caret library. provide us with clustering functions, so we will use the stats library for clustering instead. Although the stats package provides clustering functions, it lacks good visualizations, as well as a few other useful clustering functions to employ a series.

We can use the factoextra to determine the optimal number of K for each cluster package.

```
## ✓ readr   2.1.3      ✓ forcats 0.5.2
## ✓ purrr   0.3.4

## Warning: package 'readr' was built under R version 4.2.2

## Warning: package 'forcats' was built under R version 4.2.2

## ── Conflicts ──────────────────────────────────────────
tidyverse_conflicts() ──
## ✕ stringr::boundary() masks strucchange::boundary()
## ✕ dplyr::filter()     masks stats::filter()
## ✕ dplyr::lag()        masks stats::lag()
## ✕ purrr::lift()       masks caret::lift()
```

```
library(caret)
library(dplyr)
# View dataset
w_df <- wine_df
head(w_df)
```

```
##   Type Alcohol Malic_Acid  Ash Ash_Alcanity Magnesium Total_Phenols
Flavanoids
## 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(w_df)
```

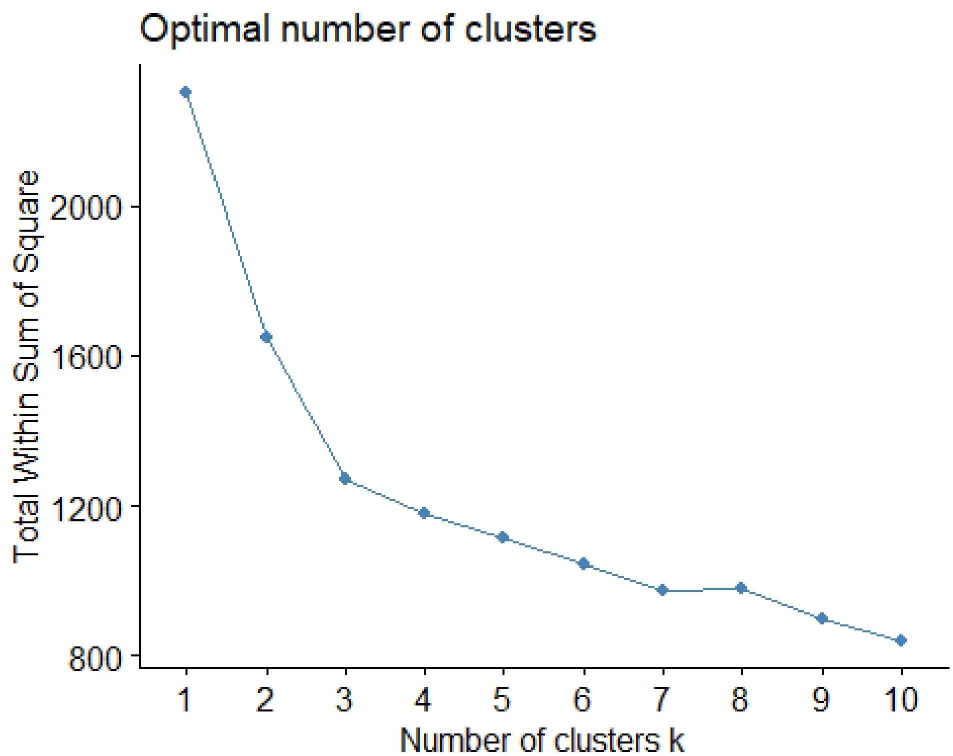
```
## [1] 178  14
```

```
# Remove class labels
predictors <- w_df %>% 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
```

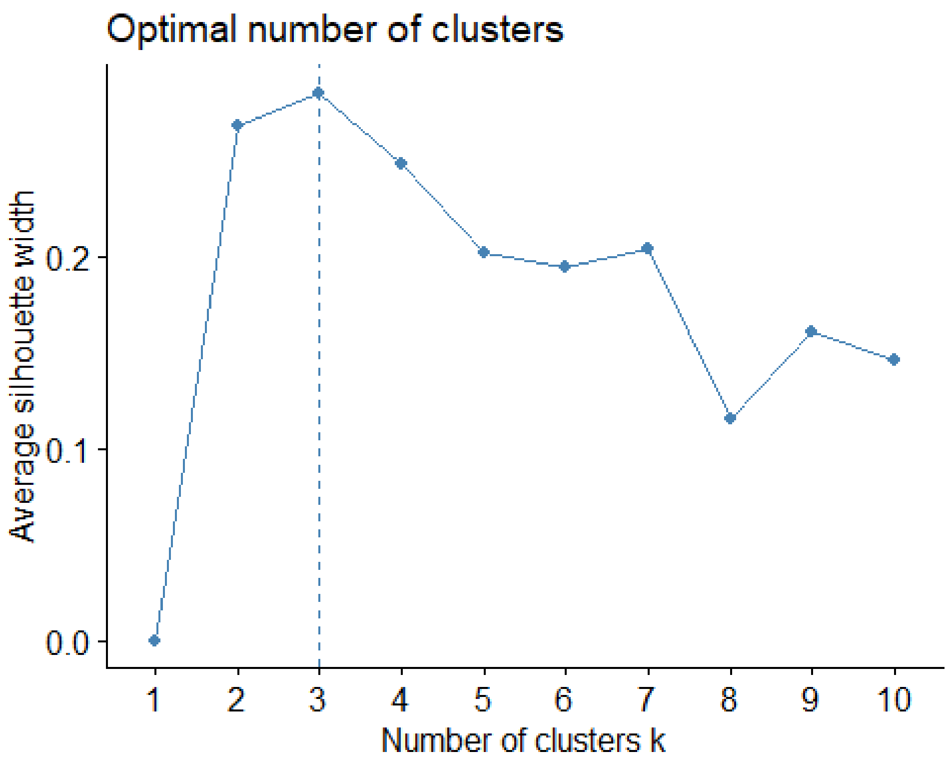
```
# Set seed
set.seed(123)

# 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)

library(stats)
library(factoextra)
library(ggplot2)
library(tidyverse)
library(caret)
library(dplyr)
# Find the knee
fviz_nbclust(predictors, kmeans, method = "wss")
```

There are two viable options here, according to the plot. Some argue that K = 2 represents the last non-flat slope, while others argue that K = 3 represents the last point before the slope flattens. The average silhoutte scores of different K values can also be compared. This one is simpler because the user is only expected to select the highest value.




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

The knee indicates a K of 3, and the silhoutte score indicates a K of 3. Hence,  The K value is set by the k=3.Centers parameter, and the number of random starts is determined by nstart.

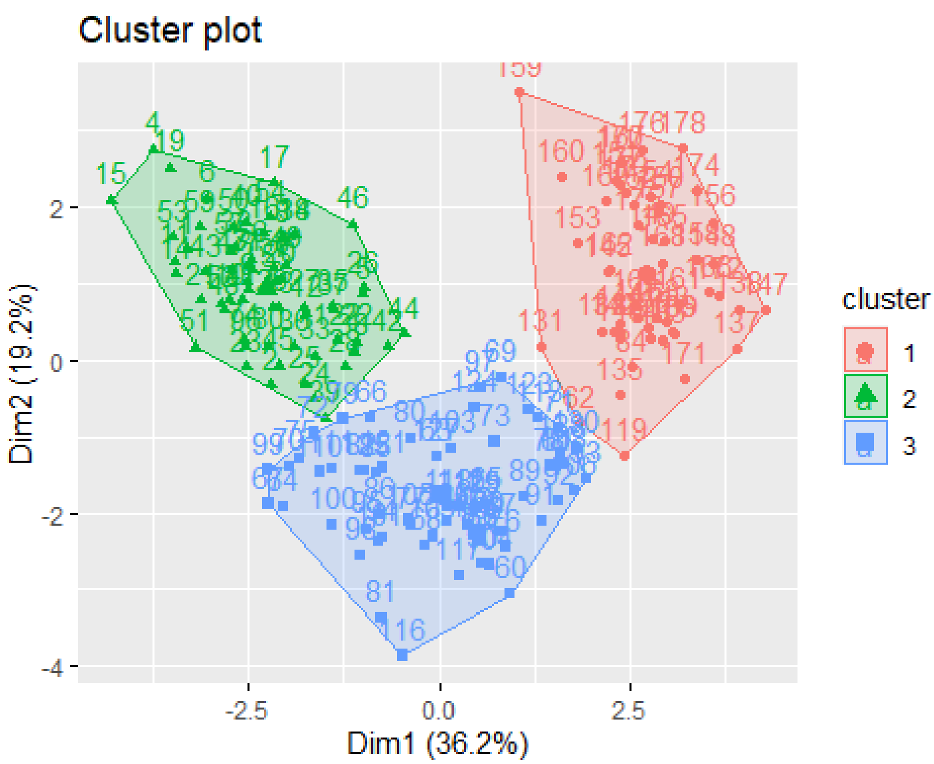



```
# Fit the data
k_fit <- kmeans(predictors, centers = 3, nstart = 25)
# Display the kmeans object information
k_fit
```

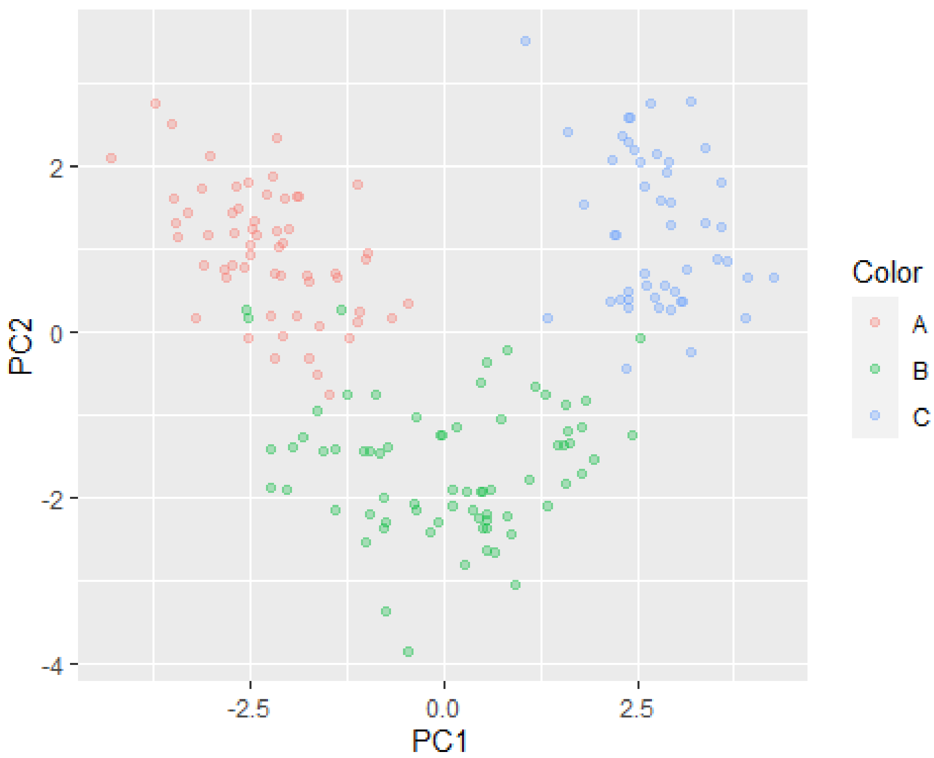
```
## K-means clustering with 3 clusters of sizes 51, 62, 65
##
## Cluster means:
##      Alcohol Malic_Acid        Ash Ash_Alcanity   Magnesium Total_Phenols
## 1  0.1644436  0.8690954  0.1863726    0.5228924 -0.07526047   -0.97657548
## 2  0.8328826 -0.3029551  0.3636801   -0.6084749  0.57596208    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  33  34  35  36  37  38
39  40
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
2   2
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58
59  60
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
2   3
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78
79  80
##   3   1   3   3   3   3   3   3   3   3   3   3   3   2   3   3   3   3
3   3
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98
99 100
##   3   3   3   1   3   3   3   3   3   3   3   3   3   3   3   2   3   3
3   3
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118
119 120
##   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
1   3
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138
139 140
##   3   2   3   3   3   3   3   3   3   3   1   1   1   1   1   1   1   1
1   1
## 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158
159 160
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
1   1
## 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178
##   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"

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

The fviz cluster function is used to visualize how the clusters are formed. formed. This function generates two-dimensional data from high-dimensional data using PCA. dimensions in which it visualizes wine data on a two-dimensional plane The storyline is composed of the first two major components that explain the majority of the variation

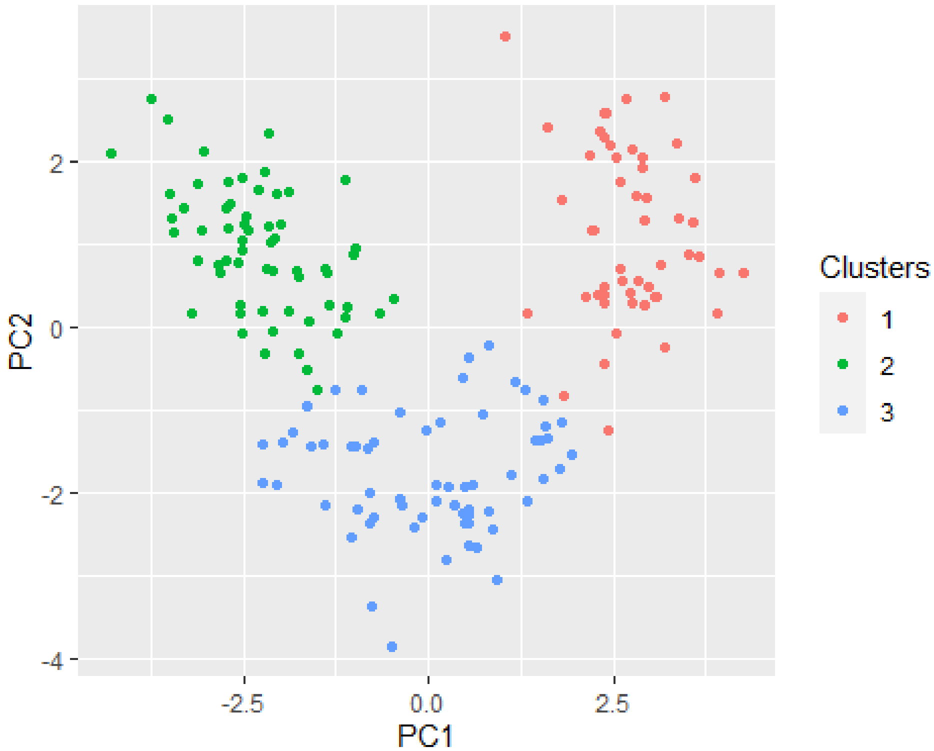



```
# Calculate PCA
pca = prcomp(predictors)
# Save as dataframe
rotated_data = as.data.frame(pca$x)
# Add original labels as a reference
rotated_data$Color <- w_df$Type
# Plot and color by labels
ggplot(data = rotated_data, aes(x = PC1, y = PC2, col = Color)) +
geom_point(alpha = 0.3)
```




The cluster plot can also be done on ggplot using the algorithm's cluster results. This is accomplished in K Means by calling $cluster on the fit. This way, it will obtain the coloring for individual points while removing the area markers.

```
# Assign clusters as a new column
rotated_data$Clusters = as.factor(k_fit$cluster)
# Plot and color by labels
ggplot(data = rotated_data, aes(x = PC1, y = PC2, col = Clusters)) +
geom_point()
```




```
# Create a dataframe
result <- data.frame(Type = w_df$Type, Kmeans = k_fit$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
## 33      A       2
## 34      A       2
## 35      A       2
## 36      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      B       3
## 61      B       3
## 62      B       1
## 63      B       3
## 64      B       3
## 65      B       3
## 66      B       3
## 67      B       3
## 68      B       3
## 69      B       3
## 70      B       3
## 71      B       3
```

```
## 72     B      3
## 73     B      3
## 74     B      2
## 75     B      3
## 76     B      3
## 77     B      3
## 78     B      3
## 79     B      3
## 80     B      3
## 81     B      3
## 82     B      3
## 83     B      3
## 84     B      1
## 85     B      3
## 86     B      3
## 87     B      3
## 88     B      3
## 89     B      3
## 90     B      3
## 91     B      3
## 92     B      3
## 93     B      3
## 94     B      3
## 95     B      3
## 96     B      2
## 97     B      3
## 98     B      3
## 99     B      3
## 100    B      3
```

```
# 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
```

According to both, K Means produced relatively better clusters on the data. On the table, use cross tabulation and one-by-one comparison. When compared to When the original type attributes A,B,C have been completed with cluster numbers 1,2,3, it attempts to Discover how accurate it is. So, 48 Cs have been shown as 1 and B has been misidentified as 3, which is fairly accurate. 59 of the Bs have been shown as 2 and A and Cs 2 and 3 have been misidentified. There have been 65 Bs identified as and no misidentifications for this cluster. Clustering has a 97% overall accuracy. Comparison of Classification and Clustering Accuracy

The precision of the KNN classification model on wine

**CLASSIFICATION:** Applying two classification Models on the chosen dataset.

1) SVM. 2)KNN Analysis.

```
library(stats)
library(factoextra)
library(ggplot2)
library(tidyverse)
library(caret)
library(dplyr)
library(mlbench)

## Warning: package 'mlbench' was built under R version 4.2.2

library(sandwich)
library(zoo)
library(party)
library(rpart)
library(rpart.plot)
library(caret)
library(e1071)
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       CC5G
## 4             1.23       CC5G
## 5             6.15       CC5G
## 6             9.50       CC5G
```

```
set.seed(123)
```

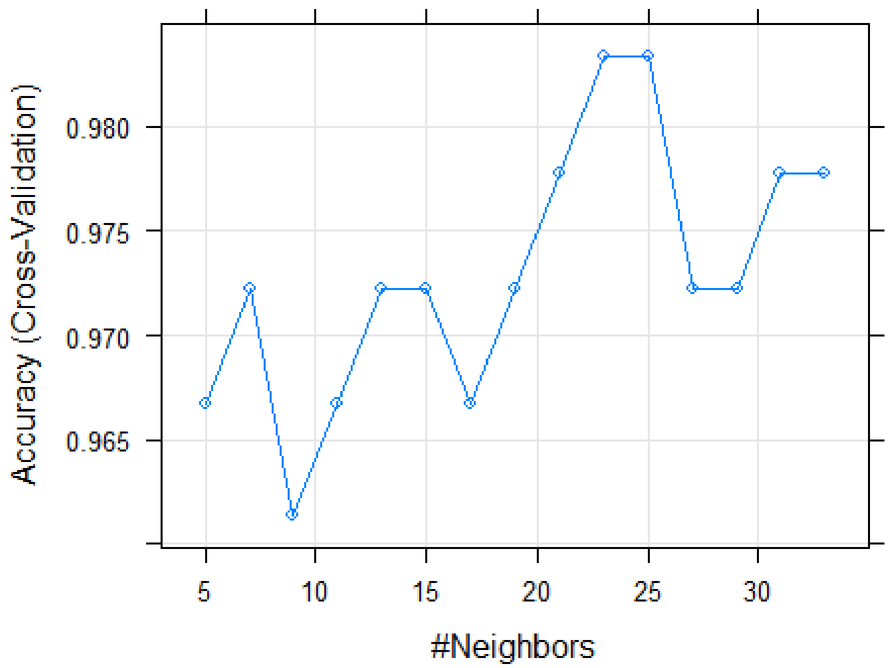
KNN is a non-parametric approach that classifies observations based on their class of its closest K neighbors When the decision boundary is ambiguous, this model is useful. It is non-linear, but it does not tell us which predictors are important. The KNN model measures the distance between two points using Euclidean distance. The model may be influenced if features have different scales. As each of the 13 characteristics have different scales, it is critical to normalize data so that all features have the same scale the same value range

```
# Remember scaling is crucial for KNN
ctrl <- trainControl(method="cv", number = 10)
knnFit <- train(Type ~ ., data = wine_df,
                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 = wine_df,
```

```
                method = "knn",
                trControl = ctrl,
                preProcess = c("center","scale"),
                tuneLength = 15)

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




```
#done the first time you use it
library(kknn)

## Warning: package 'kknn' was built under R version 4.2.2

##
## Attaching package: 'kknn'

## The following object is masked from 'package:caret':
##
##     contr.dummy

# 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 distance 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 = wine_df,
                  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(wine_df$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
```

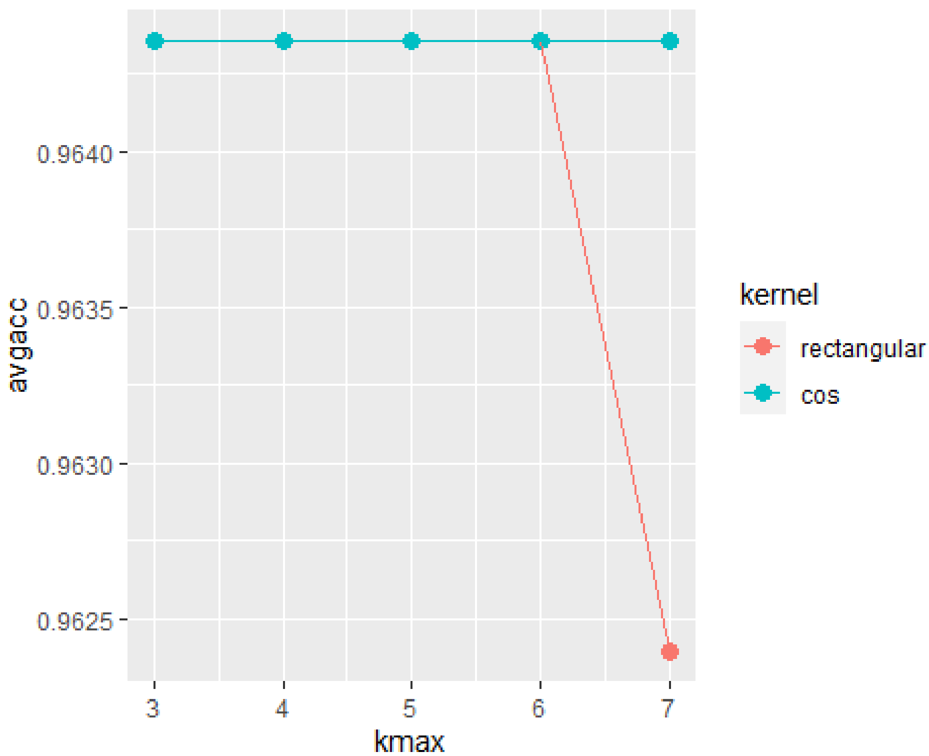
```
knn_results = kknn_fit$results # gives just the table of results by parameter
head(knn_results)
```

```
##   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
function
ggplot(knn_results, aes(x=kmax, y=avgacc, color=kernel)) +
  geom_point(size=3) + geom_line()
```




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

#SVM #SVM (CLASSIFIER-2) #Setting parameters

```
grid <- expand.grid(C = 10^seq(-5,2,0.5))
train_control = trainControl(method = "cv", number = 10)

library(stats)
library(factoextra)
library(ggplot2)
library(tidyverse)
library(caret)
library(dplyr)
library(mlbench)
library(sandwich)
library(zoo)
library(party)
library(rpart)
library(rpart.plot)
library(caret)
```

```
library(e1071)
#Applying Model
```

We can set all sorts of parameters for *trainControl* to tell caret how we want to evaluate our classifier. This is how we specify the different sampling methods explained in the lecture. We also give the paramters of those sampling methods, so in this case we will use cross validation (specified as *cv*) and specify we want 10 folds (using the *number* argument).

```
svm_grid <- train(Type ~., data = traindata, method = "svmLinear",
 trControl = train_control, tuneGrid = grid)
svm_grid
```

```
## Support Vector Machines with Linear Kernel
##
## 126 samples
##  13 predictor
##   3 classes: 'A', 'B', 'C'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 114, 113, 114, 114, 114, 112, ...
## Resampling results across tuning parameters:
##
##   C             Accuracy   Kappa
##   1.000000e-05  0.3978938  0.0000000
##   3.162278e-05  0.3978938  0.0000000
##   1.000000e-04  0.3978938  0.0000000
##   3.162278e-04  0.3978938  0.0000000
##   1.000000e-03  0.3978938  0.0000000
##   3.162278e-03  0.9511905  0.9249624
##   1.000000e-02  0.9833333  0.9748684
##   3.162278e-02  0.9750000  0.9625000
##   1.000000e-01  0.9500000  0.9247368
##   3.162278e-01  0.9500000  0.9248351
##   1.000000e+00  0.9500000  0.9251155
##   3.162278e+00  0.9500000  0.9251155
##   1.000000e+01  0.9500000  0.9251155
##   3.162278e+01  0.9500000  0.9251155
##   1.000000e+02  0.9500000  0.9251155
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.01.
```

we can generate a confusion matrix to show a breakdown of exactly where our classifier is making mistakes. the *confusionMatrix* function needs two arguments: the labels and the predictions. A typical usage is to take that a step further and get a confusion matrix for the performance on the train set and, on the test, set separately so they can be compared.

```
#Applying Confusion Matrix on Test data
pred_svm <- predict(svm_grid, testdata)
confusionMatrix(as.factor(testdata$Type), pred_svm)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C
##          A 17  0  0
##          B  1 20  0
##          C  0  0 14
##
## Overall Statistics
##
##                Accuracy : 0.9808
##                  95% CI : (0.8974, 0.9995)
##     No Information Rate : 0.3846
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9708
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C
## Sensitivity            0.9444   1.0000   1.0000
## Specificity            1.0000   0.9688   1.0000
## Pos Pred Value         1.0000   0.9524   1.0000
## Neg Pred Value         0.9714   1.0000   1.0000
## Prevalence             0.3462   0.3846   0.2692
## Detection Rate         0.3269   0.3846   0.2692
## Detection Prevalence   0.3269   0.4038   0.2692
## Balanced Accuracy      0.9722   0.9844   1.0000
```

# #Data Evaluation

```
#Creating Confusion Matrix
predsvm_ev <- predict(svm_grid, testdata)
con_mat <- confusionMatrix(as.factor(testdata$Type), predsvm_ev)
con_mat

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C
##          A 17  0  0
##          B  1 20  0
```

```
##          C  0  0 14
##
## Overall Statistics
##
##                Accuracy : 0.9808
##                  95% CI : (0.8974, 0.9995)
##     No Information Rate : 0.3846
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9708
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C
## Sensitivity            0.9444   1.0000   1.0000
## Specificity            1.0000   0.9688   1.0000
## Pos Pred Value         1.0000   0.9524   1.0000
## Neg Pred Value         0.9714   1.0000   1.0000
## Prevalence             0.3462   0.3846   0.2692
## Detection Rate         0.3269   0.3846   0.2692
## Detection Prevalence   0.3269   0.4038   0.2692
## Balanced Accuracy      0.9722   0.9844   1.0000
```

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

```
##          Sensitivity Specificity Pos Pred Value Neg Pred Value Precision
## Class: A   0.9444444     1.00000       1.000000      0.9714286  1.000000
## Class: B   1.0000000     0.96875       0.952381      1.0000000  0.952381
## Class: C   1.0000000     1.00000       1.000000      1.0000000  1.000000
##             Recall        F1 Prevalence Detection Rate Detection
Prevalence
## Class: A 0.9444444 0.9714286  0.3461538      0.3269231
0.3269231
## Class: B 1.0000000 0.9756098  0.3846154      0.3846154
0.4038462
## Class: C 1.0000000 1.0000000  0.2692308      0.2692308
0.2692308
##          Balanced Accuracy
## Class: A         0.9722222
## Class: B         0.9843750
## Class: C         1.0000000
```

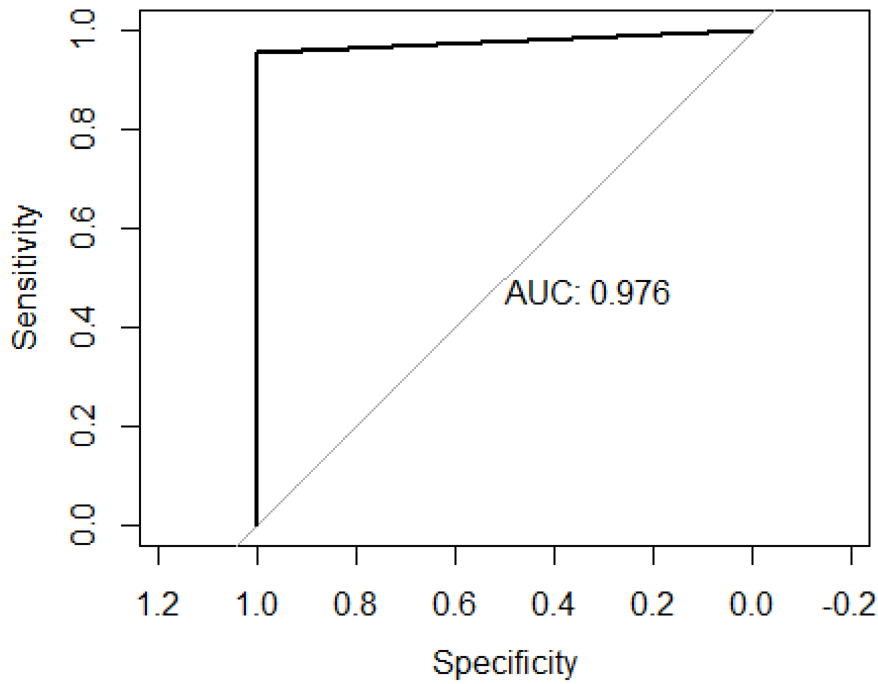
#Getting Precision and Recall

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

```
##          Precision
## Class: A  1.000000
## Class: B  0.952381
## Class: C  1.000000

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

##             Recall
## Class: A 0.9444444
## Class: B 1.0000000
## Class: C 1.0000000
```

#Ploting ROC

```
library(pROC)

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var

roccurve<-roc(response=testdata$Type,predictor=as.numeric(predsvm_ev))

## Warning in roc.default(response = testdata$Type, predictor =
## as.numeric(predsvm_ev)): 'response' has more than two levels. Consider
setting
## 'levels' explicitly or using 'multiclass.roc' instead

## Setting levels: control = A, case = B

## Setting direction: controls < cases

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




Here the SVM model has a higher AUC and balanced accuracy score and therefore it would be desired if we want to have a more balanced classifier. The overall accuracy scores here only vary by a small margin and we might think that they are really similar. However, the SVM classifier is actually better at predicting the non-dominant class (positives) as seen in the confusion matrices.

## REFLECTION:

The most useful lessons from this course are investigating various types of datasets, data mining, and data visualization. preprocessing, various cleaning methods, and addressing missing values with numerous techniques such as binning, smoothing, normalizing, and many more Furthermore, Predicting Machine Learning uses various "classification" techniques to compare the values of labels with known values. SVM, Decision Tree, and KNN parameters were used. Furthermore, I learned how to use "clustering" and Models such as k-means and other techniques can be used to predict the values of unknown labels. to deal with I learned advance evaluation while dealing with prejudice and class imbalance. Furthermore, with Knowing a model's error rate and employing accuracy, recall, and ROC can be extremely beneficial.