**Lab Evaluation Report**

# UCS654 PREDICTIVE ANALYTICS USING STATISTICS

****

**Submitted By:**

**Ratish Jindal (101803004)**

**COE-1**

**Submitted To :**

**Mr. PS rana**

**Dataset Name:** Wine Data Set

**Dataset Source:** https://archive.ics.uci.edu/ml/datasets/wine

**Source:**

Original Owners:  
  
Forina, M. et al, PARVUS - An Extendible Package for Data Exploration, Classification and Correlation. Institute of Pharmaceutical and Food Analysis and Technologies, Via Brigata Salerno, 16147 Genoa, Italy.

**Data Set Information:**

These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.  
  
The attributes are (donated by Riccardo Leardi, riclea **'@'** anchem.unige.it )  
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

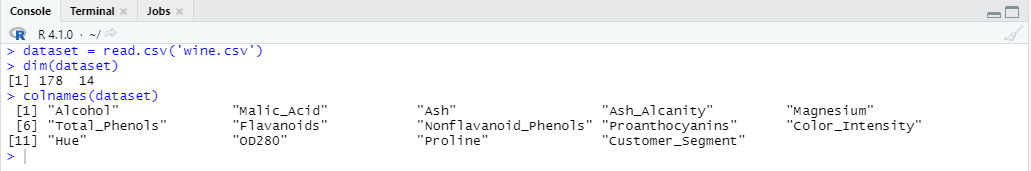
**CODE:**

**Part 1 – Analytics and Visualization**

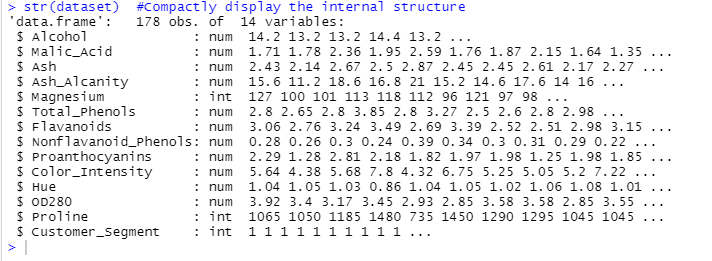
dataset = read.csv('wine.csv') #import datset

dim(dataset) #output the size/shape of the dataset

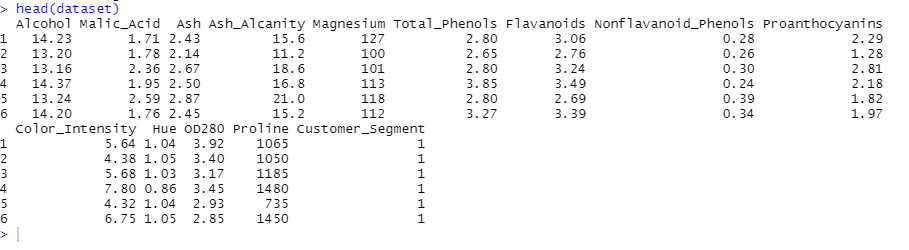
colnames(dataset) #list the names of the columns of the dataset



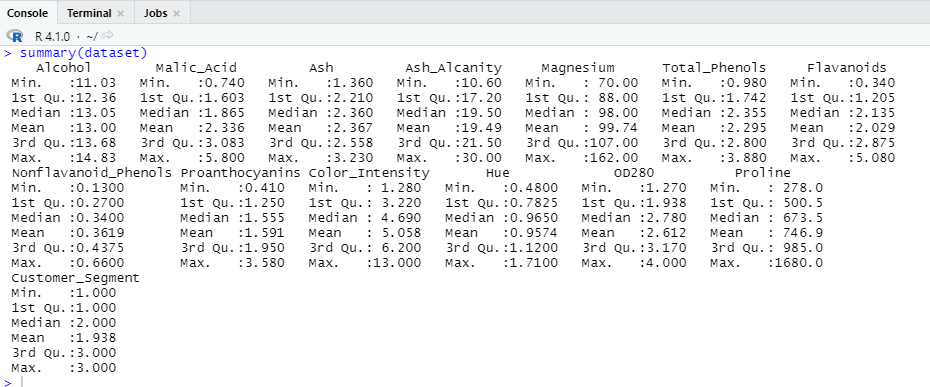
str(dataset) #Compactly display the internal structure



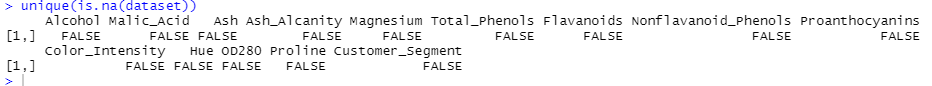
head(dataset) #Returns the first or last parts



summary(dataset)



unique(is.na(dataset)) #outputs true if there is a missing value in any column



data\_1=dataset %>% filter(dataset$Customer\_Segment == 1)

data\_2=dataset %>% filter(dataset$Customer\_Segment == 2)

data\_3=dataset %>% filter(dataset$Customer\_Segment == 3)

this creates a subset of the winery products collected from each winery.

a=table(dataset$Customer\_Segment)

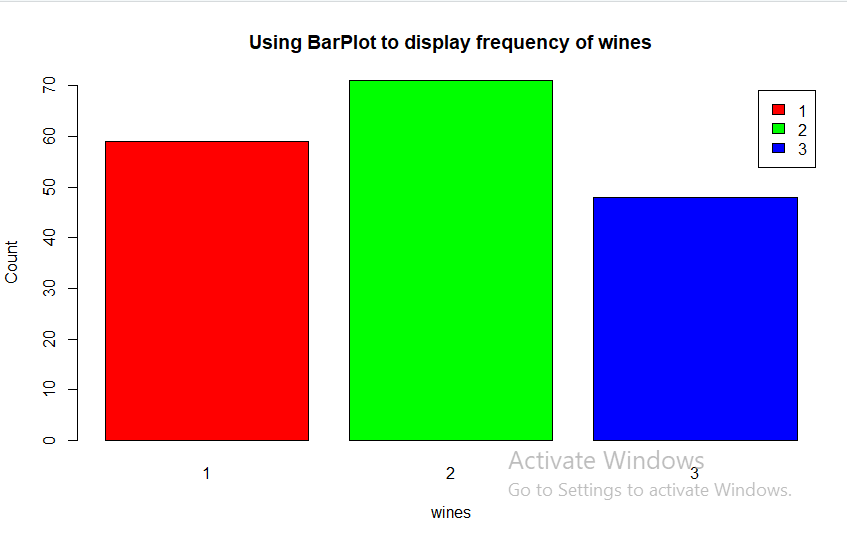
barplot(a,main="Using BarPlot to display frequency of wines",

ylab="Count",

xlab="wines",

col=rainbow(3),

legend=rownames(a))

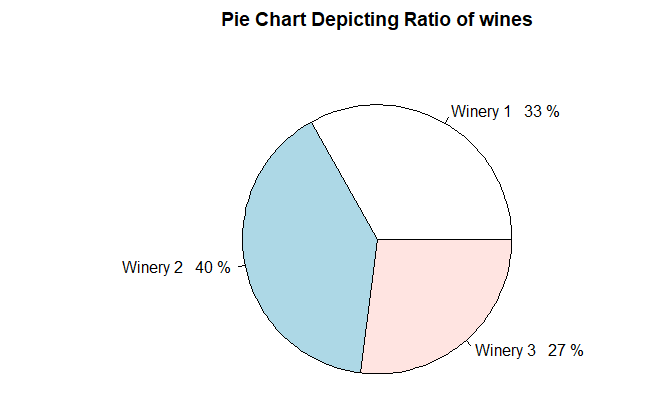


pct=round(a/sum(a)\*100)

lbs=paste(c("Winery 1","Winery 2","Winery 3")," ",pct,"%",sep=" ")

library(plotrix)

pie(a,labels=lbs,main="Pie Chart Depicting Ratio of wines")



a=table(data\_1$Alcohol)

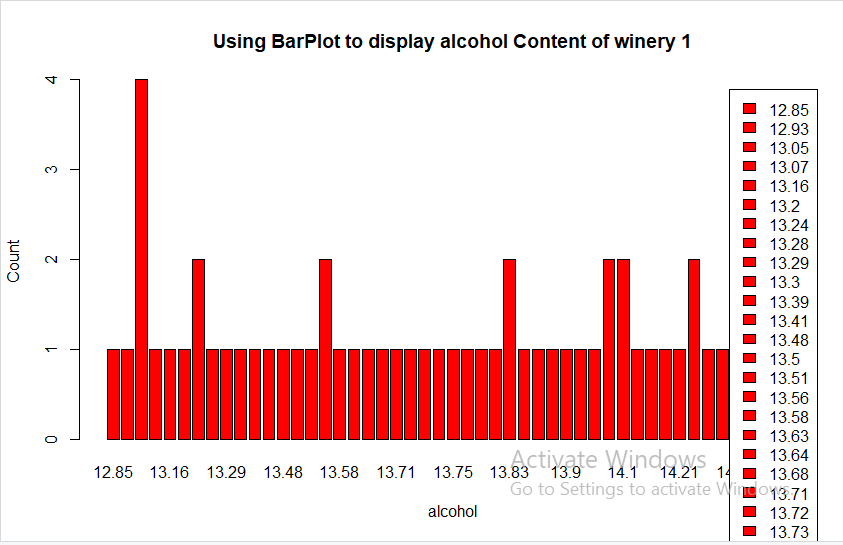
barplot(a,main="Using BarPlot to display alcohol Content of winery 1",

ylab="Count",

xlab="alcohol",

col='red',

legend=rownames(a))



hist(data\_1$Alcohol,

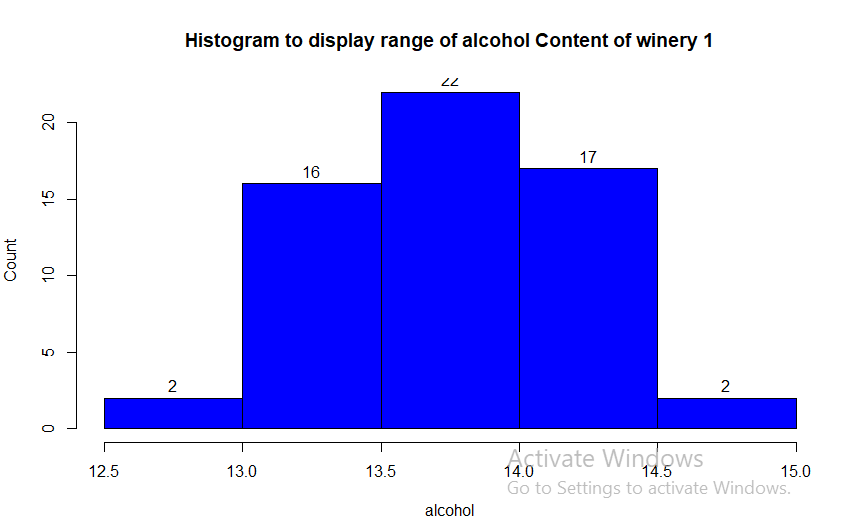
col="blue",

main="Histogram to display range of alcohol Content of winery 1",

xlab="alcohol",

ylab="Count",

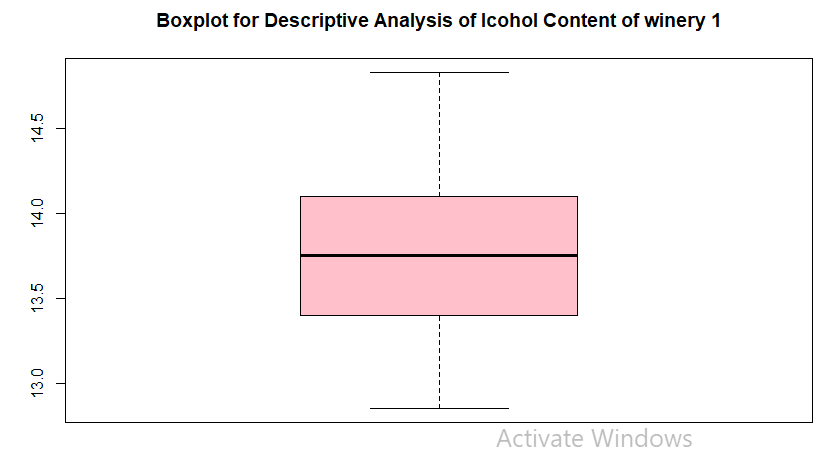
labels=TRUE)



boxplot(data\_1$Alcohol,

col="pink",

main="Boxplot for Descriptive Analysis of lcohol Content of winery 1")



Similarly, we can plot other variables of our dataset and subset to analyze its value range and frequency.

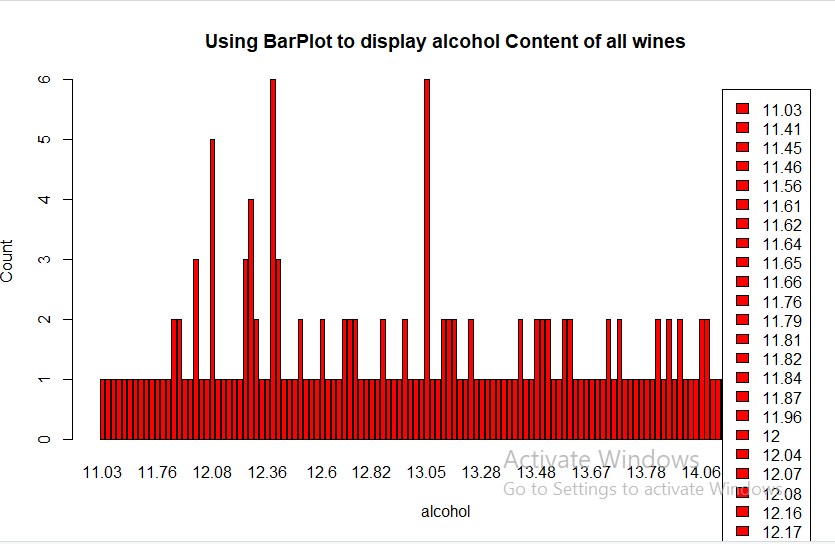
a=table(dataset$Alcohol)

barplot(a,main="Using BarPlot to display alcohol Content of all wines",

ylab="Count",

xlab="alcohol",

col='red',

legend=rownames(a)) 

hist(dataset$Alcohol,

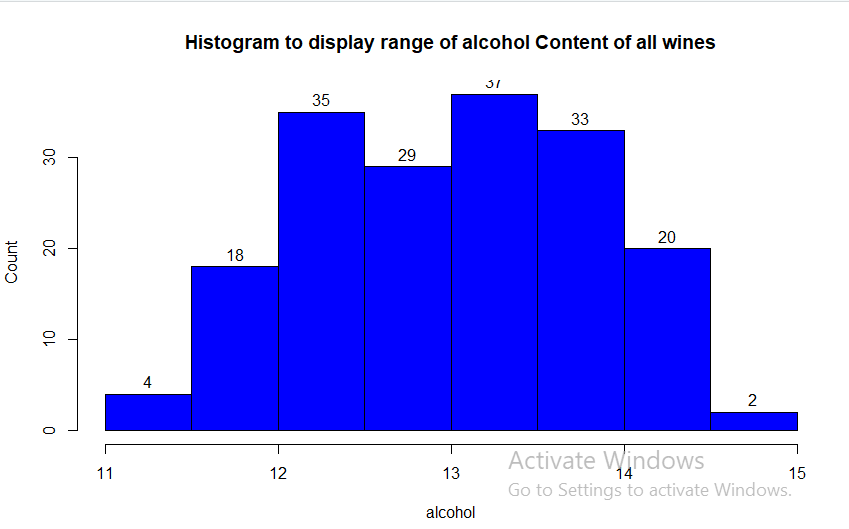
col="blue",

main="Histogram to display range of alcohol Content of all wines",

xlab="alcohol",

ylab="Count",

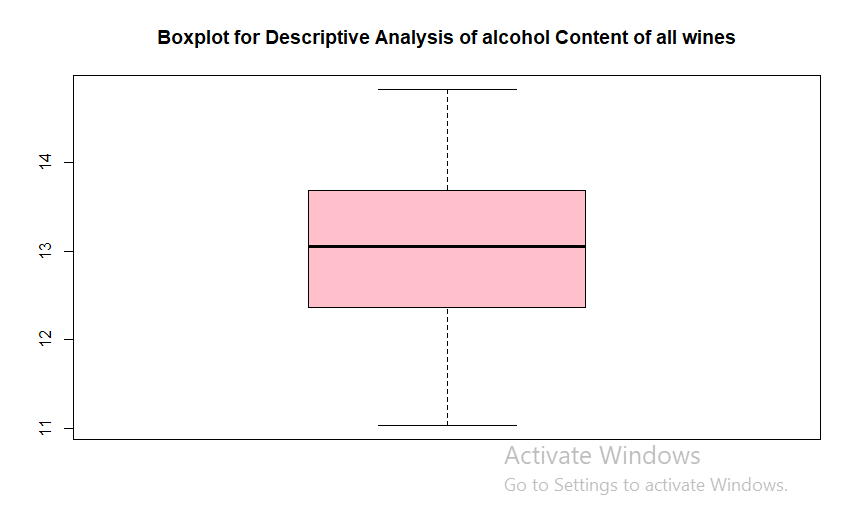
labels=TRUE)



boxplot(dataset$Alcohol,

col="pink",

main="Boxplot for Descriptive Analysis of alcohol Content of all wines")

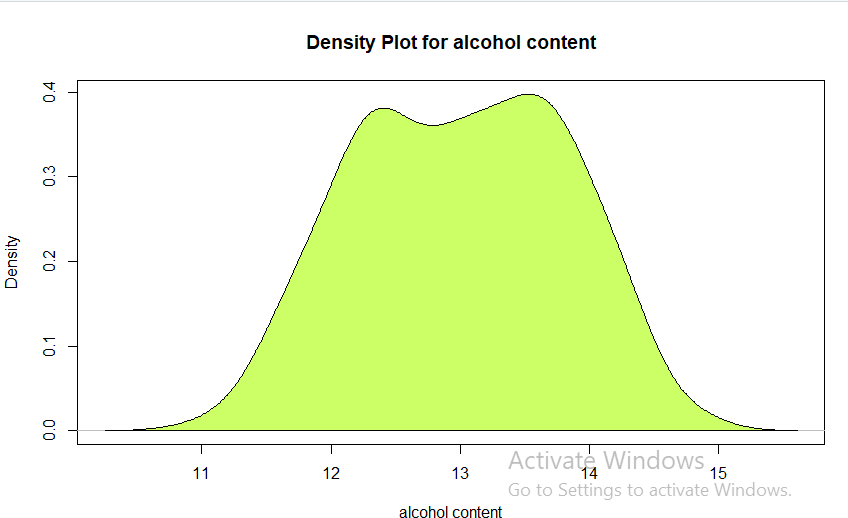


plot(density(dataset$Alcohol),

main="Density Plot for alcohol content",

xlab="alcohol content",ylab="Density")

polygon(density(dataset$Alcohol),col="#ccff66")

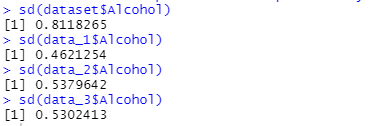


sd(dataset$Alcohol) # computes the standard deviation

sd(data\_1$Alcohol)

sd(data\_2$Alcohol)

sd(data\_3$Alcohol)



**Part 2 – Prediction**

I will use principal component analysis (PCA) on our dataset. I am using PCA for predicting values because goal of PCA is to identify and detect correlation between variables, if there's a strong correlation and it’s found, then you could reduce the dimensionality, which really what PCA is intended for.

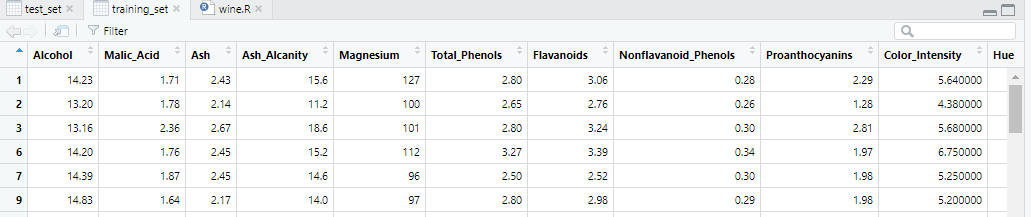
library(caTools)

set.seed(123)

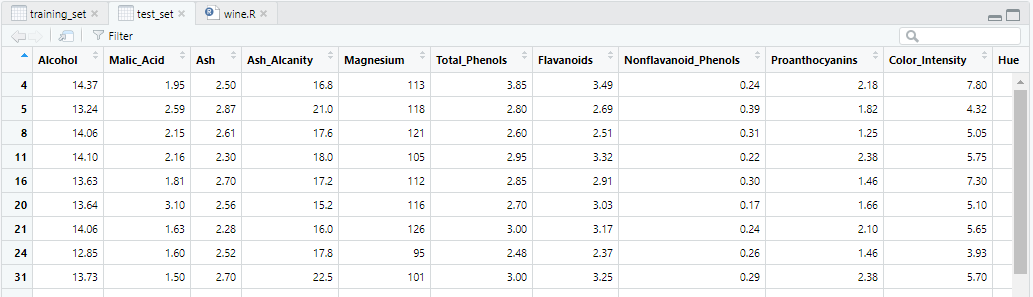
split = sample.split(dataset$Customer\_Segment, SplitRatio = 0.8) #spliting the dataset

training\_set = subset(dataset, split == TRUE)

test\_set = subset(dataset, split == FALSE)



Training set

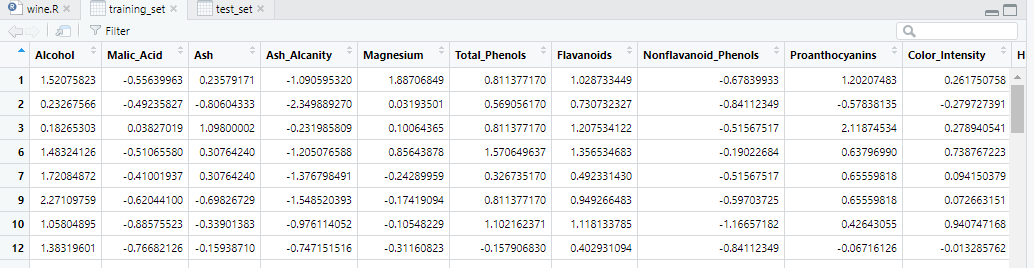


Test set

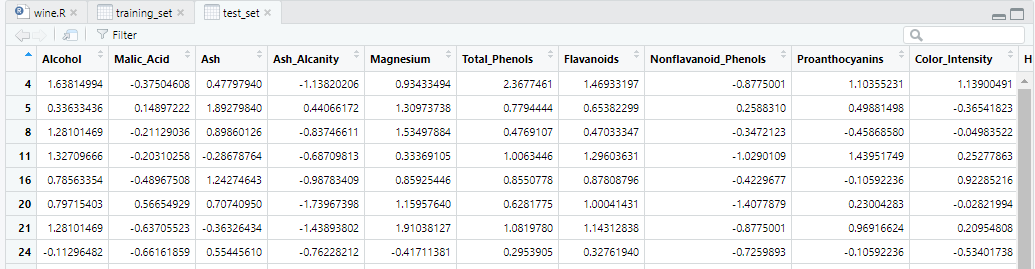
# Feature Scaling

training\_set[-14] = scale(training\_set[-14])

test\_set[-14] = scale(test\_set[-14])



Training set after scaling



Test set after scaling

library(caret)

library(e1071)

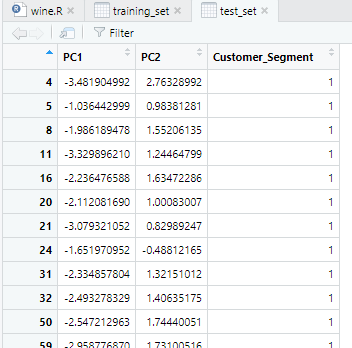
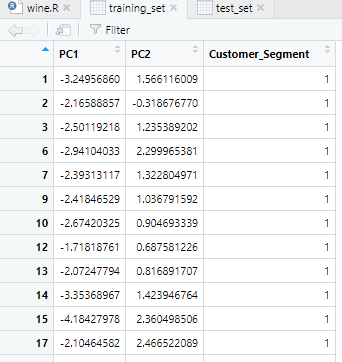
pca = preProcess(x = training\_set[-14], method = 'pca', pcaComp = 2) #training the model

training\_set = predict(pca, training\_set)

training\_set = training\_set[c(2, 3, 1)]

test\_set = predict(pca, test\_set)

test\_set = test\_set[c(2, 3, 1)]



Training and Test set after feature extraction

# Fitting SVM to the Training set (I chose svm model)

classifier = svm(formula = Customer\_Segment ~ .,

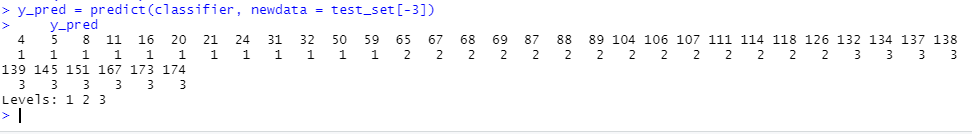
data = training\_set,

type = 'C-classification',

kernel = 'linear')

# Predicting the Test set results

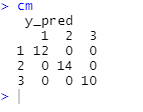
y\_pred = predict(classifier, newdata = test\_set[-3])



# Making the Confusion Matrix

cm = table(test\_set[, 3], y\_pred)

cm



# Visualising the Training set results

library(ElemStatLearn)

set = training\_set

X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)

X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)

grid\_set = expand.grid(X1, X2)

colnames(grid\_set) = c('PC1', 'PC2')

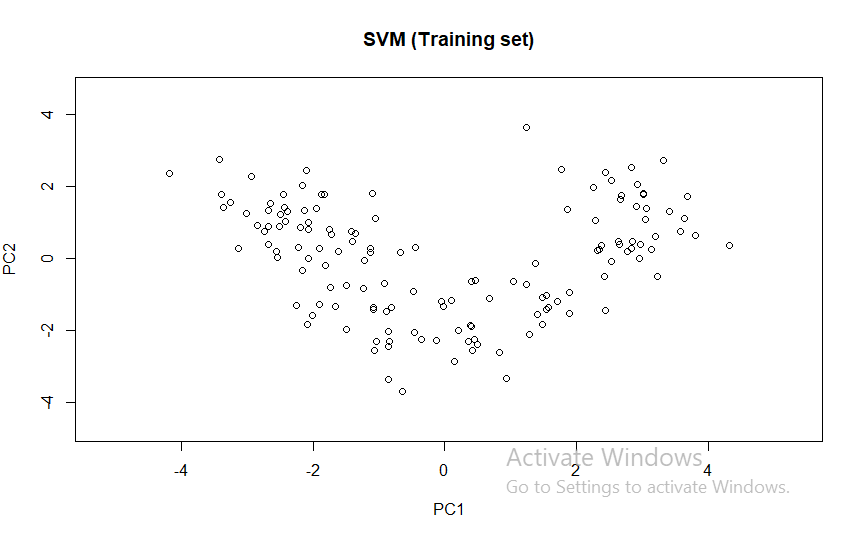
y\_grid = predict(classifier, newdata = grid\_set)

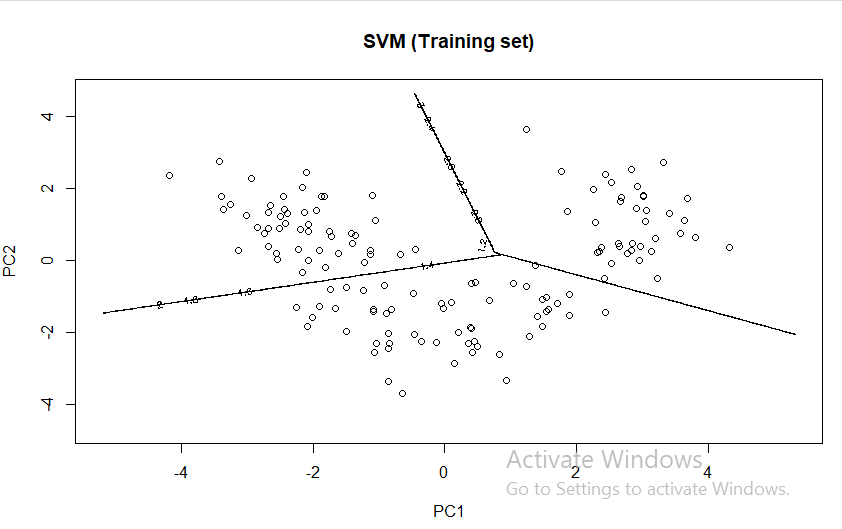
plot(set[, -3],

main = 'SVM (Training set)',

xlab = 'PC1', ylab = 'PC2',

xlim = range(X1), ylim = range(X2))



contour(X1, X2, matrix(as.numeric(y\_grid), length(X1), length(X2)), add = TRUE)

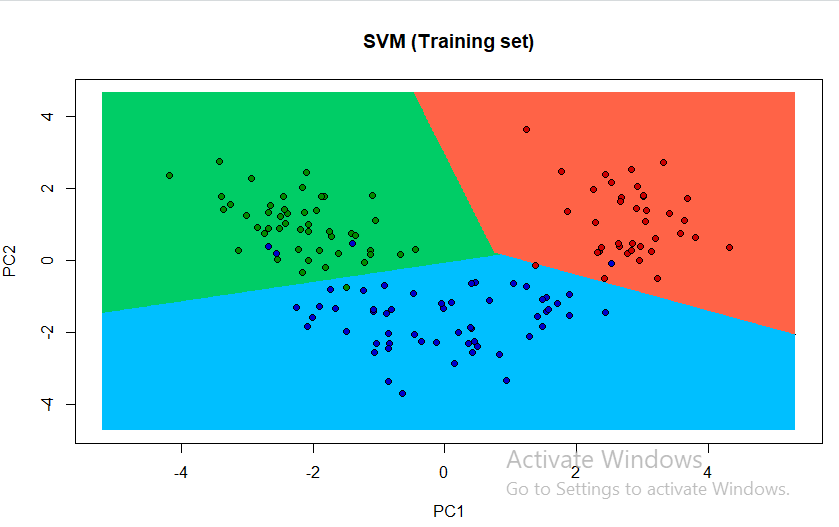
points(grid\_set,

pch = '.',

col = ifelse(y\_grid == 2, 'deepskyblue', ifelse(y\_grid == 1, 'springgreen3', 'tomato')))

points(set, pch = 21,

bg = ifelse(set[, 3] == 2, 'blue3', ifelse(set[, 3] == 1, 'green4', 'red3')))



# Visualising the Test set results

library(ElemStatLearn)

set = test\_set

X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)

X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)

grid\_set = expand.grid(X1, X2)

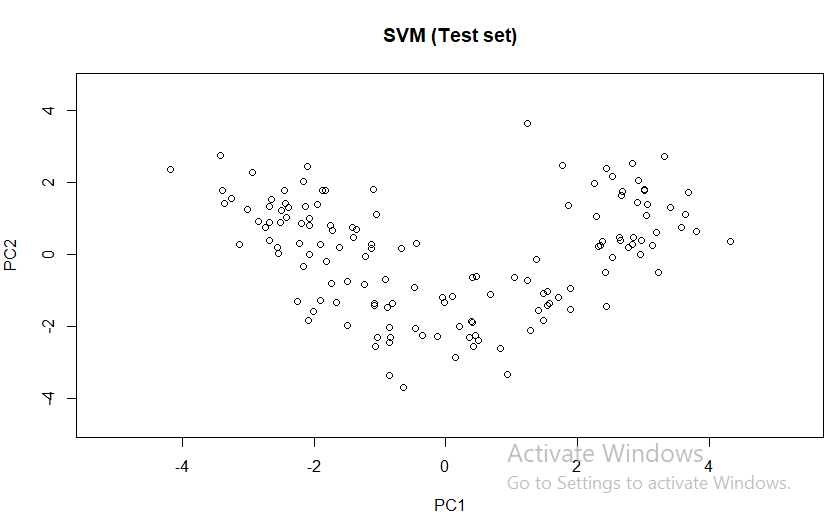
colnames(grid\_set) = c('PC1', 'PC2')

y\_grid = predict(classifier, newdata = grid\_set)

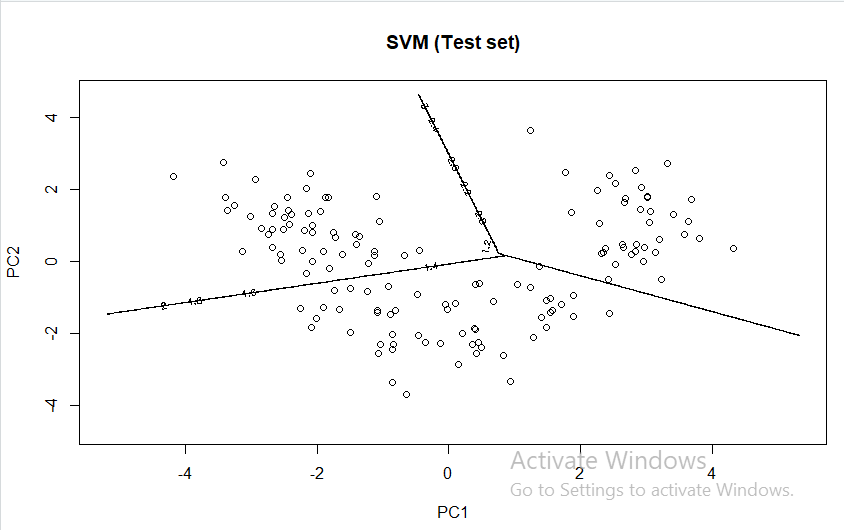
plot(set[, -3], main = 'SVM (Test set)',

xlab = 'PC1', ylab = 'PC2',

xlim = range(X1), ylim = range(X2))



contour(X1, X2, matrix(as.numeric(y\_grid), length(X1), length(X2)), add = TRUE)



points(grid\_set,

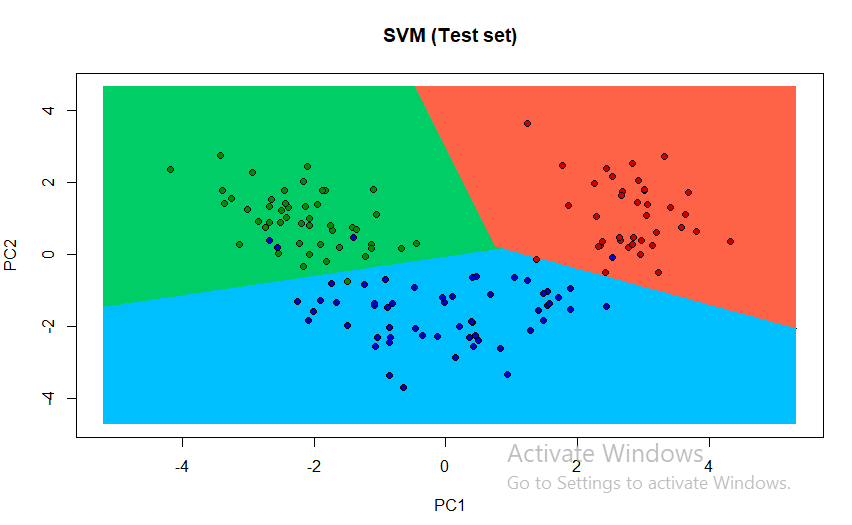
pch = '.',

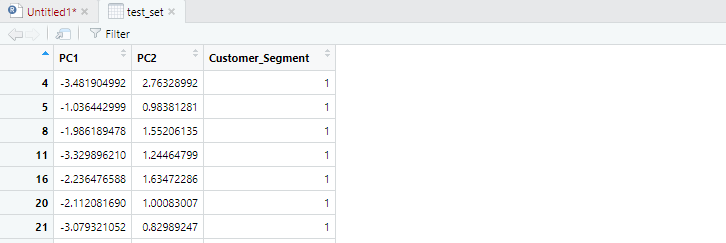
col = ifelse(y\_grid == 2, 'deepskyblue', ifelse(y\_grid == 1, 'springgreen3', 'tomato')))

points(set,

pch = 21,

bg = ifelse(set[, 3] == 2, 'blue3', ifelse(set[, 3] == 1, 'green4', 'red3')))





Test Set Predictions

**SUMMARY**

In this data science project, we worked on a highly distributed Multivariate dataset. With help of various functions we analyzed our dataset’s component, how the alcohol level and other measures of wines from each winery re distributed, their frequency, count etc. due to high correlation between the dataset, we used PCA model to reduce their dimensionality and then plot its graph and at last predicted the values. I used SVM model for classification and it has given me an accuracy of 100 % on the test set and it could be seen in confusion matrix.

I can use other classification models like naïve bayes alse and similarly other dimension reduction models like lda or kernel PCA instead of PCA.

.