# PCA

#it is one feature extraction technique.

#we dont consider dependent variable that is why it is unsupervised model

# it will extract one which have most variance in detaset therefore we can reduce no of independent variable.

#it is basically classification problem it which ,here we will use logestic regression model and will apply PCA in classification

#----------------------------problem

#dependent variable is customer\_segment which have three type of wine and each wine corresponding to each customer

#each independent variable have different type of chemical for three type of customer

#we can not visulize lot of independent variable at once so we will apply dimensionlaity to limited variable which can explain maximum variance.

#----------------------------

# Importing the dataset

dataset = read.csv('Wine.csv')

#http://archive.ics.uci.edu/ml/datasets/wine dataset information

# Splitting the dataset into the Training set and Test set

install.packages('caTools')

library(caTools)

set.seed(123)

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

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

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

# Feature Scaling

#scaling must be apply when we are using dimensionality

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

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

# Applying PCA

# install.packages('caret')

library(caret)

# install.packages('e1071')

library(e1071)

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

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)]

# Fitting SVM to the Training set

# install.packages('e1071')

library(e1071)

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)

# Visualising the Training set results

#install.packages("ElemStatLearn")

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')))