**HOMEWORK 4 -Q1**

**Sadiya Amreen**

Problem 1 (25 points):

For this problem, you will tune and apply kNN and compare it to other classifiers. We will use the wine quality data, which has a number of measurements about chemical components in wine, plus a quality rating. There are separate files for red and white wines, so the first step is some data preparation. a. Load the two provided wine quality datasets and prepare them by (1) ensuring that all the variables have the right type (e.g., what is numeric vs. factor), (2) adding a type column to each that indicates if it is red or white wine and (2) merging the two tables together into one table (hint: try full\_join()). You now have one table that contains the data on red and white wine, with a column that tells if the wine was from the red or white set (the type column you made).

# Preprocessing data:

getwd()

## [1] "C:/Sadiya Studies/Data Science/DS441-Fundamts DS/homework"

print(setwd("C:/Sadiya Studies/Data Science/DS441-Fundamts DS/homework"))

## [1] "C:/Sadiya Studies/Data Science/DS441-Fundamts DS/homework"

library(astsa)  
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(e1071)  
library(ggplot2)  
library(tidyverse)

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

## ✔ tibble 3.1.8 ✔ purrr 0.3.4  
## ✔ tidyr 1.2.1 ✔ stringr 1.4.1  
## ✔ readr 2.1.2 ✔ forcats 0.5.2  
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ✖ purrr::lift() masks caret::lift()

library(rpart)  
library(rattle)

## Loading required package: bitops  
##   
## Attaching package: 'bitops'  
##   
## The following object is masked from 'package:astsa':  
##   
## %^%  
##   
## Rattle: A free graphical interface for data science with R.  
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.

#Creating two tables and joining them  
whitewine1 <- read.csv("1-winequality-WHITE.csv", sep = ";", header = TRUE) #create excel table by spearating ';'  
redwine1 <- read.csv("2-winequality-RED.csv", sep = ";", header = TRUE) #create excel table by spearating ';'  
type1 <- rep(c('red'), each = 1599) # Create type coloumn with lable red  
redwine1$type <- type1 # # add type coloumn with lable red using $ sign  
summary(redwine1)

## fixed.acidity volatile.acidity citric.acid residual.sugar   
## Min. : 4.60 Min. :0.1200 Min. :0.000 Min. : 0.900   
## 1st Qu.: 7.10 1st Qu.:0.3900 1st Qu.:0.090 1st Qu.: 1.900   
## Median : 7.90 Median :0.5200 Median :0.260 Median : 2.200   
## Mean : 8.32 Mean :0.5278 Mean :0.271 Mean : 2.539   
## 3rd Qu.: 9.20 3rd Qu.:0.6400 3rd Qu.:0.420 3rd Qu.: 2.600   
## Max. :15.90 Max. :1.5800 Max. :1.000 Max. :15.500   
## chlorides free.sulfur.dioxide total.sulfur.dioxide density   
## Min. :0.01200 Min. : 1.00 Min. : 6.00 Min. :0.9901   
## 1st Qu.:0.07000 1st Qu.: 7.00 1st Qu.: 22.00 1st Qu.:0.9956   
## Median :0.07900 Median :14.00 Median : 38.00 Median :0.9968   
## Mean :0.08747 Mean :15.87 Mean : 46.47 Mean :0.9967   
## 3rd Qu.:0.09000 3rd Qu.:21.00 3rd Qu.: 62.00 3rd Qu.:0.9978   
## Max. :0.61100 Max. :72.00 Max. :289.00 Max. :1.0037   
## pH sulphates alcohol quality   
## Min. :2.740 Min. :0.3300 Min. : 8.40 Min. :3.000   
## 1st Qu.:3.210 1st Qu.:0.5500 1st Qu.: 9.50 1st Qu.:5.000   
## Median :3.310 Median :0.6200 Median :10.20 Median :6.000   
## Mean :3.311 Mean :0.6581 Mean :10.42 Mean :5.636   
## 3rd Qu.:3.400 3rd Qu.:0.7300 3rd Qu.:11.10 3rd Qu.:6.000   
## Max. :4.010 Max. :2.0000 Max. :14.90 Max. :8.000   
## type   
## Length:1599   
## Class :character   
## Mode :character   
##   
##   
##

type <- rep(c('white'), each = 4898) # create type coloumn with lable white  
whitewine1$type <- type # add type coloumn with lable white using $ sign  
summary(whitewine1)

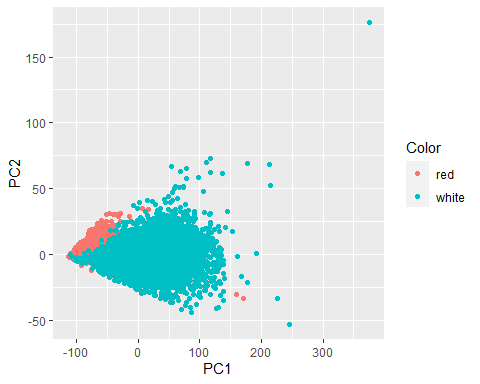
## fixed.acidity volatile.acidity citric.acid residual.sugar   
## Min. : 3.800 Min. :0.0800 Min. :0.0000 Min. : 0.600   
## 1st Qu.: 6.300 1st Qu.:0.2100 1st Qu.:0.2700 1st Qu.: 1.700   
## Median : 6.800 Median :0.2600 Median :0.3200 Median : 5.200   
## Mean : 6.855 Mean :0.2782 Mean :0.3342 Mean : 6.391   
## 3rd Qu.: 7.300 3rd Qu.:0.3200 3rd Qu.:0.3900 3rd Qu.: 9.900   
## Max. :14.200 Max. :1.1000 Max. :1.6600 Max. :65.800   
## chlorides free.sulfur.dioxide total.sulfur.dioxide density   
## Min. :0.00900 Min. : 2.00 Min. : 9.0 Min. :0.9871   
## 1st Qu.:0.03600 1st Qu.: 23.00 1st Qu.:108.0 1st Qu.:0.9917   
## Median :0.04300 Median : 34.00 Median :134.0 Median :0.9937   
## Mean :0.04577 Mean : 35.31 Mean :138.4 Mean :0.9940   
## 3rd Qu.:0.05000 3rd Qu.: 46.00 3rd Qu.:167.0 3rd Qu.:0.9961   
## Max. :0.34600 Max. :289.00 Max. :440.0 Max. :1.0390   
## pH sulphates alcohol quality   
## Min. :2.720 Min. :0.2200 Min. : 8.00 Min. :3.000   
## 1st Qu.:3.090 1st Qu.:0.4100 1st Qu.: 9.50 1st Qu.:5.000   
## Median :3.180 Median :0.4700 Median :10.40 Median :6.000   
## Mean :3.188 Mean :0.4898 Mean :10.51 Mean :5.878   
## 3rd Qu.:3.280 3rd Qu.:0.5500 3rd Qu.:11.40 3rd Qu.:6.000   
## Max. :3.820 Max. :1.0800 Max. :14.20 Max. :9.000   
## type   
## Length:4898   
## Class :character   
## Mode :character   
##   
##   
##

rtable <- full\_join(redwine1, whitewine1, all = TRUE) #join two table with full join function

## Joining, by = c("fixed.acidity", "volatile.acidity", "citric.acid",  
## "residual.sugar", "chlorides", "free.sulfur.dioxide", "total.sulfur.dioxide",  
## "density", "pH", "sulphates", "alcohol", "quality", "type")

1. Use PCA to create a projection of the data to 2D and show a scatterplot with color showing the wine type.

#Use PCA and create scatter plot with wine type.  
data\_1 = rtable %>% select(-c(type)) #Remove type col  
pca = prcomp(data\_1) #Saving pca as data frame  
rotated\_data = as.data.frame(pca$x)  
rotated\_data$Color <- rtable$type  
ggplot(data = rotated\_data, aes(x = PC1, y = PC2, col = Color)) + geom\_point() #Create scatter plot

 c.We are going to try kNN, SVM and decision trees on this data. Based on the ‘shape’ of the data in the visualization from (b), which do you think will do best and why?

##Inference:  
#We use KNN because SVM has a lot of points and they get overlapped and hyperboundary dosent coincide.   
#In Decision tree, the categorical variable is not being used Hence choosing KNN.

1. Use kNN (tune k), use decision trees (basic rpart method is fine), and SVM (tune C) to predict type from the rest of the variables. Compare the accuracy values – is this what you expected? Can you explain it? Note: you will need to fix the columns names for rpart because it is not able to handle the underscores. This code will do the trick (assuming you called your data wine\_quality): colnames(wine\_quality) <- make.names(colnames(wine\_quality))

# KNN  
ctrl <- trainControl(method = "cv", number = 10)  
knn1 <- train(type~., data = rtable, method = "knn", trControl = ctrl, preProcess= c("center", "scale"), tuneLength = 15)  
knn1

## k-Nearest Neighbors   
##   
## 6497 samples  
## 12 predictor  
## 2 classes: 'red', 'white'   
##   
## Pre-processing: centered (12), scaled (12)   
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 5847, 5847, 5847, 5847, 5848, 5847, ...   
## Resampling results across tuning parameters:  
##   
## k Accuracy Kappa   
## 5 0.9926118 0.9800547  
## 7 0.9930734 0.9812993  
## 9 0.9933808 0.9821334  
## 11 0.9924575 0.9796914  
## 13 0.9933811 0.9821530  
## 15 0.9930731 0.9813499  
## 17 0.9930729 0.9813584  
## 19 0.9930731 0.9813516  
## 21 0.9930729 0.9813391  
## 23 0.9929190 0.9809306  
## 25 0.9927652 0.9805204  
## 27 0.9927652 0.9805204  
## 29 0.9926114 0.9801102  
## 31 0.9919955 0.9784498  
## 33 0.9919955 0.9784516  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was k = 13.

#SVM  
train\_control = trainControl(method = 'cv', number =10)  
grid <- expand.grid(C = 10^seq(-5,2,0.5))  
svm1 <- train(type~., data = rtable, method = "svmLinear", trControl = train\_control, tuneGrid = grid)  
svm1

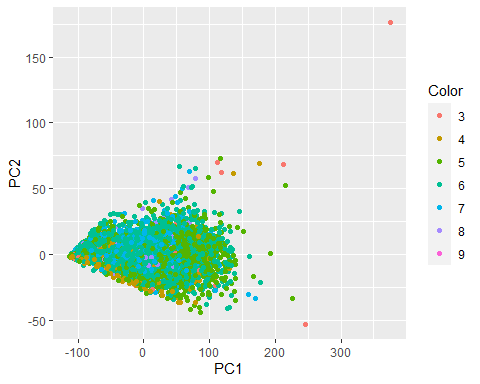
## Support Vector Machines with Linear Kernel   
##   
## 6497 samples  
## 12 predictor  
## 2 classes: 'red', 'white'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 5847, 5848, 5847, 5847, 5847, 5847, ...   
## Resampling results across tuning parameters:  
##   
## C Accuracy Kappa   
## 1.000000e-05 0.7538865 0.000000000  
## 3.162278e-05 0.7541941 0.001872731  
## 1.000000e-04 0.9301249 0.791996531  
## 3.162278e-04 0.9846106 0.957932795  
## 1.000000e-03 0.9903051 0.973760734  
## 3.162278e-03 0.9915361 0.977175635  
## 1.000000e-02 0.9924596 0.979673868  
## 3.162278e-02 0.9935368 0.982555995  
## 1.000000e-01 0.9943060 0.984631403  
## 3.162278e-01 0.9950753 0.986701432  
## 1.000000e+00 0.9950753 0.986698013  
## 3.162278e+00 0.9953830 0.987523582  
## 1.000000e+01 0.9953830 0.987523582  
## 3.162278e+01 0.9953830 0.987523582  
## 1.000000e+02 0.9952291 0.987104644  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was C = 3.162278.

#decision tree  
train\_control = trainControl(method = 'cv', number =10)  
tree1 <- train(type~., data = rtable, method = "rpart", trControl = train\_control)  
tree1

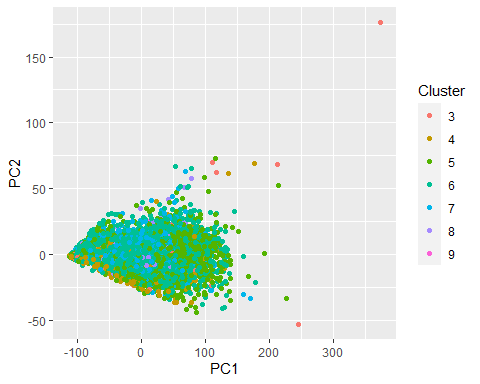
## CART   
##   
## 6497 samples  
## 12 predictor  
## 2 classes: 'red', 'white'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 5848, 5848, 5847, 5848, 5847, 5847, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.06253909 0.9513588 0.8637129  
## 0.06754221 0.9307365 0.8048305  
## 0.70043777 0.8351670 0.3816070  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.06253909.

1. Use the same already computed PCA again to show a scatter plot of the data and to visualize the labels for kNN, decision tree and SVM. Note that you do not need to recreate the PCA projection, you have already done this in 1b. Here, you just make a new visualization for each classifier using its labels for color (same points but change the color). Map the color results to the classifier, that is use the “predict” function to predict the class of your data, add it to your data frame and use it as a color. This is done for KNN in the tutorial, it should be similar for the others. Consider and explain the differences in how these classifiers performed.

#knn  
rotated\_data$Color = as.factor(knn1$trainingData$quality)  
ggplot(data = rotated\_data, aes(x = PC1, y = PC2, col = Color))+ geom\_point()



#tree  
rotated\_data$Cluster = as.factor(tree1$trainingData$quality)  
ggplot(data = rotated\_data, aes(x = PC1, y = PC2, col = Cluster))+ geom\_point()



#svm  
rotated\_data$Color = as.factor(svm1$trainingData$quality)  
ggplot(data = rotated\_data, aes(x = PC1, y = PC2, col = Color)) + geom\_point()

