Orthopedic Patients Classification

Chhaya Mudgal

12/7/2020

Introduction and Project Outline

This script is the Capstone Own Project for HarvardX Data Science Professional Certificate. Dataset chosen to study classification of Orthopedic Patients based on Biomechanical Features. Data has been downloaded from Kaggle database.In this project KNN (Nearest Neighbour Algorithm) Supported Vector Machine (SVM) and Random Forest will be used for comparing accuracy of patients classification..

Machine Learning is being used in various medical fields to predict and classify diseases. Orthopedic health condition of a pateint can be detected from the biomechanical features. Application of machine learning algorithms in medical science helps in classification. Different algorithms are applied to detect diseases and classify patients accordingly. In this project various machine learning algorithms are applied to find out which one works most accurately to detect and classify orthopedic patients. Algorithms compared for accuracy are KNN,SVM and Random Forest. Each of the patients in the dataset is represented by six biomechanical attributes derived from the shape and orientation of pelvis and lumbar spine.

Why have i chose the data? I have chosen this dataset because: It is freely avaliable online on Kaggle. It is 'medium' sized. Not small or too big to be procecssed on my personal computer. There are a reasonable number of predictor columns, and easy to understand.

Load Data

Step1: Load Library Packages

```r
#Step1: Load Library Packages
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(corrplot)) install.packages("corrplot", repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(gmodels)) install.packages("gmodels", repos = "http://cran.us.r-project.org")
if(!require(knn)) install.packages("knn", repos = "http://cran.us.r-project.org")

library(gmodels)
library(tidyverse)
library(caret)
library(data.table)
library(corrplot)
library(ggplot2)
library(class)
```

Step2: Load Data

Data is downloaded from file stored in github repository Initial Step is to know the data set, hence the peek at the top 10 and last 10 lines for the dataset. Data set has 310 rows and 7 columns. Analyzing the structure of the dataset it is found that the six features have numeric values and the 7th column is the label that tells whether the patient falls under normal or abnormal category. This label is of type factor.

```
# Step2: Load Data
# Data file is stored in github for easy
# Getting data file from https://github.com/cmudgal/DataScienceHarvardX/tree/master/CapstoneProject/Orth


u="https://raw.githubusercontent.com/cmudgal/DataScienceHarvardX/master/CapstoneProject/OrthoPatientData
ortho_data <- read.csv(u)#"Column_2C_weka.csv")


# Peek at first 10 rows of data
head(ortho_data, n=10)
```

```
##    pelvic_incidence pelvic_tilt.numeric lumbar_lordosis_angle sacral_slope
## 1          63.02782           22.552586              39.60912     40.47523
## 2          39.05695           10.060991              25.01538     28.99596
## 3          68.83202           22.218482              50.09219     46.61354
## 4          69.29701           24.652878              44.31124     44.64413
## 5          49.71286            9.652075              28.31741     40.06078
## 6          40.25020           13.921907              25.12495     26.32829
## 7          53.43293           15.864336              37.16593     37.56859
## 8          45.36675           10.755611              29.03835     34.61114
## 9          43.79019           13.533753              42.69081     30.25644
## 10         36.68635            5.010884              41.94875     31.67547
##    pelvic_radius degree_spondylolisthesis    class
## 1       98.67292               -0.2544000 Abnormal
## 2      114.40543                4.5642586 Abnormal
## 3      105.98514               -3.5303173 Abnormal
## 4      101.86850               11.2115234 Abnormal
## 5      108.16872                7.9185006 Abnormal
## 6      130.32787                2.2306517 Abnormal
## 7      120.56752                5.9885507 Abnormal
## 8      117.27007              -10.6758708 Abnormal
## 9      125.00289               13.2890182 Abnormal
## 10      84.24142                0.6644371 Abnormal
```

```
# Peek at last 10 rows of data
tail(ortho_data, n=10)
```

```
##     pelvic_incidence pelvic_tilt.numeric lumbar_lordosis_angle sacral_slope
## 301         50.67668            6.461501              35.00000     44.21518
## 302         89.01488           26.075981              69.02126     62.93889
## 303         54.60032           21.488974              29.36022     33.11134
## 304         34.38230            2.062683              32.39082     32.31962
## 305         45.07545           12.306951              44.58318     32.76850
## 306         47.90357           13.616688              36.00000     34.28688
## 307         53.93675           20.721496              29.22053     33.21525
```

```
## 308         61.44660          22.694968              46.17035     38.75163
## 309         45.25279           8.693157              41.58313     36.55963
## 310         33.84164           5.073991              36.64123     28.76765
##     pelvic_radius degree_spondylolisthesis  class
## 301      116.5880               -0.2147106 Normal
## 302      111.4811                6.0615084 Normal
## 303      118.3433               -1.4710673 Normal
## 304      128.3002               -3.3655156 Normal
## 305      147.8946               -8.9417094 Normal
## 306      117.4491               -4.2453954 Normal
## 307      114.3658               -0.4210104 Normal
## 308      125.6707               -2.7078795 Normal
## 309      118.5458                0.2147502 Normal
## 310      123.9452               -0.1992491 Normal
```

```r
# Get number or rows and columns in data set
# Data set has 310 rows and 7 columns
dim(ortho_data)
```

```
## [1] 310   7
```

```r
# Get summary of the data
summary(ortho_data)
```

```
##  pelvic_incidence pelvic_tilt.numeric lumbar_lordosis_angle  sacral_slope
##  Min.   : 26.15   Min.   :-6.555      Min.   : 14.00        Min.   : 13.37
##  1st Qu.: 46.43   1st Qu.:10.667      1st Qu.: 37.00        1st Qu.: 33.35
##  Median : 58.69   Median :16.358      Median : 49.56        Median : 42.40
##  Mean   : 60.50   Mean   :17.543      Mean   : 51.93        Mean   : 42.95
##  3rd Qu.: 72.88   3rd Qu.:22.120      3rd Qu.: 63.00        3rd Qu.: 52.70
##  Max.   :129.83   Max.   :49.432      Max.   :125.74        Max.   :121.43
##  pelvic_radius    degree_spondylolisthesis      class
##  Min.   : 70.08   Min.   :-11.058          Abnormal:210
##  1st Qu.:110.71   1st Qu.:  1.604          Normal  :100
##  Median :118.27   Median : 11.768
##  Mean   :117.92   Mean   : 26.297
##  3rd Qu.:125.47   3rd Qu.: 41.287
##  Max.   :163.07   Max.   :418.543
```

```r
# Gives the structure of data info
# All non null data set with 7 variables
# 6 are numeric features
# class is the factor with levels "Abnormal" and "Normal"
str(ortho_data)
```

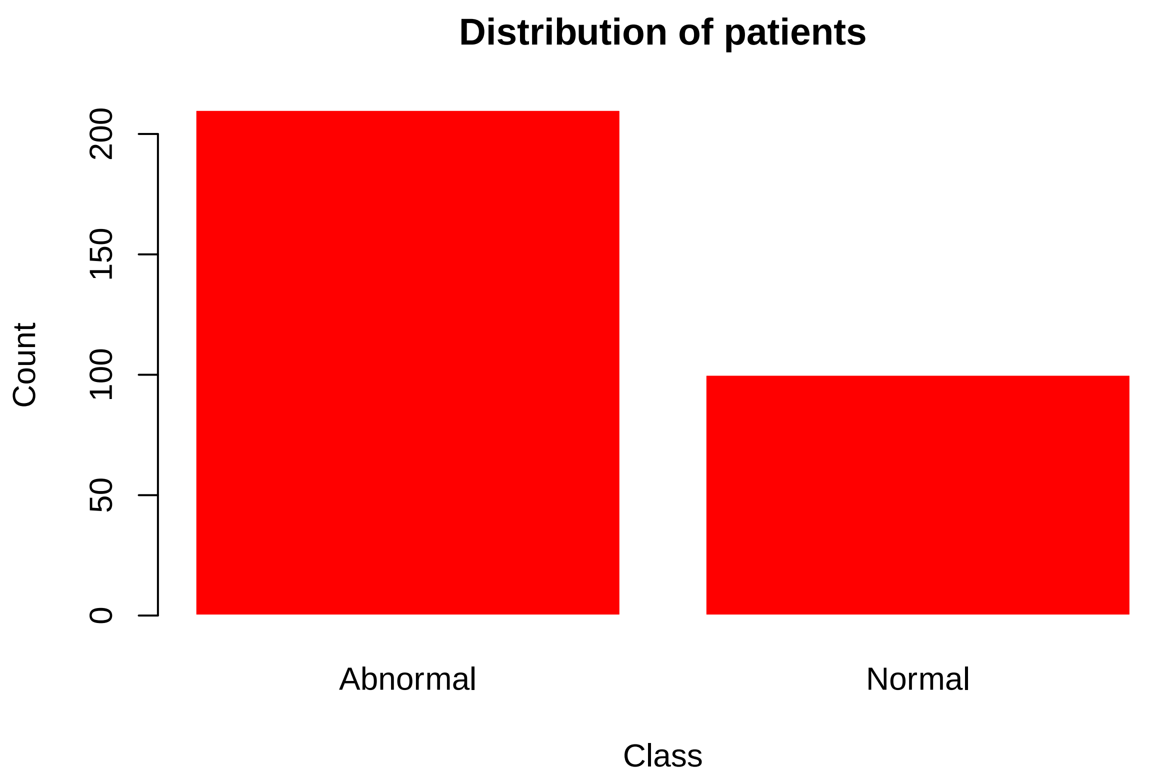
```
## 'data.frame':    310 obs. of  7 variables:
##  $ pelvic_incidence        : num  63 39.1 68.8 69.3 49.7 ...
##  $ pelvic_tilt.numeric     : num  22.55 10.06 22.22 24.65 9.65 ...
##  $ lumbar_lordosis_angle   : num  39.6 25 50.1 44.3 28.3 ...
##  $ sacral_slope            : num  40.5 29 46.6 44.6 40.1 ...
##  $ pelvic_radius           : num  98.7 114.4 106 101.9 108.2 ...
##  $ degree_spondylolisthesis: num  -0.254 4.564 -3.53 11.212 7.919 ...
##  $ class                   : Factor w/ 2 levels "Abnormal","Normal": 1 1 1 1 1 1 1 1 1 1 ...
```

Explore Data

There are highly positive correlations between Pelvic Incidence and Sacral Slope , also, between Pelvic Incidence and Lumbar Lordosis Angle as can be seen by scatter plots below. From the figure, as it seems Normal class values are smaller than Abnormal values; therefore narrowed with selecting some correlated features. Lets look at correlation matrix of the select features: pelvic_raduis, pelvic_incidence and lumbar_lordosis_angle.

There are 210 patients in Abnormal and 100 in Normal category

```
# Explore Data for Orthopedic Patients
plot(ortho_data$class, freq=TRUE, col="red", border="white",
     main="Distribution of patients", xlab="Class", ylab="Count")
```




```
#
# Divison of 'class' attribute of the patients
table(ortho_data$class)
```

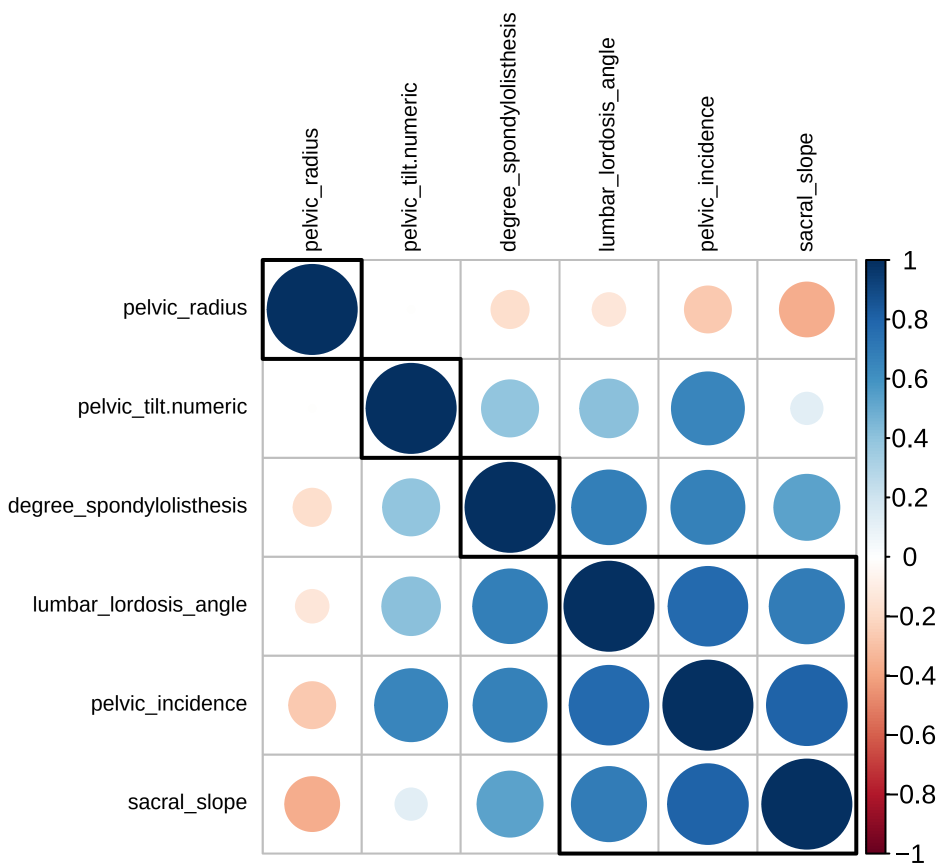
```
##
## Abnormal   Normal
##      210      100
```

```
# Percentual division of patients using the  `class` attribute
round(prop.table(table(ortho_data$class)) * 100, digits = 1)
```

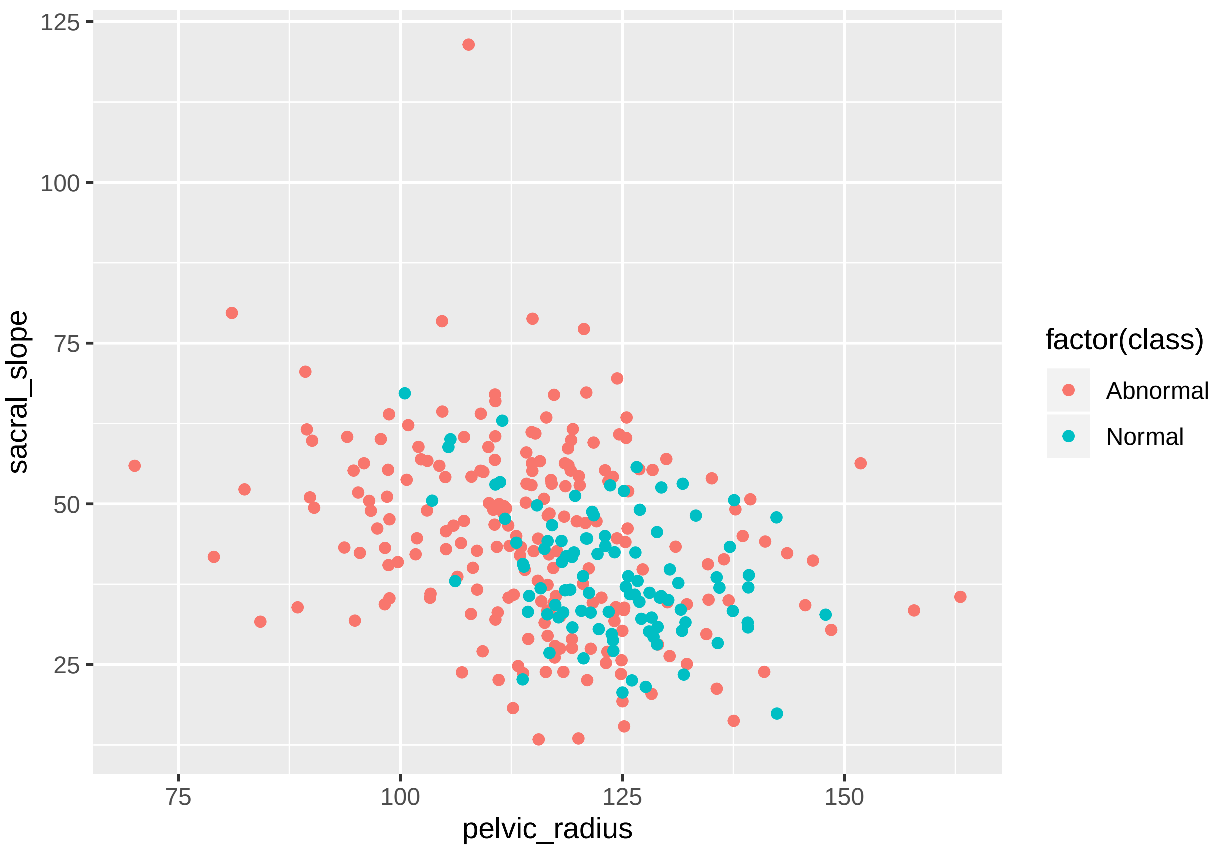
```
##
## Abnormal    Normal
##      67.7      32.3
```

```r
# Correlation plot

com = ortho_data[,1:6]
cc = cor(com, method = "spearman")
corrplot(cc, tl.col = "black", order = "hclust", hclust.method = "average", addrect = 4, tl.cex = 0.7)
```

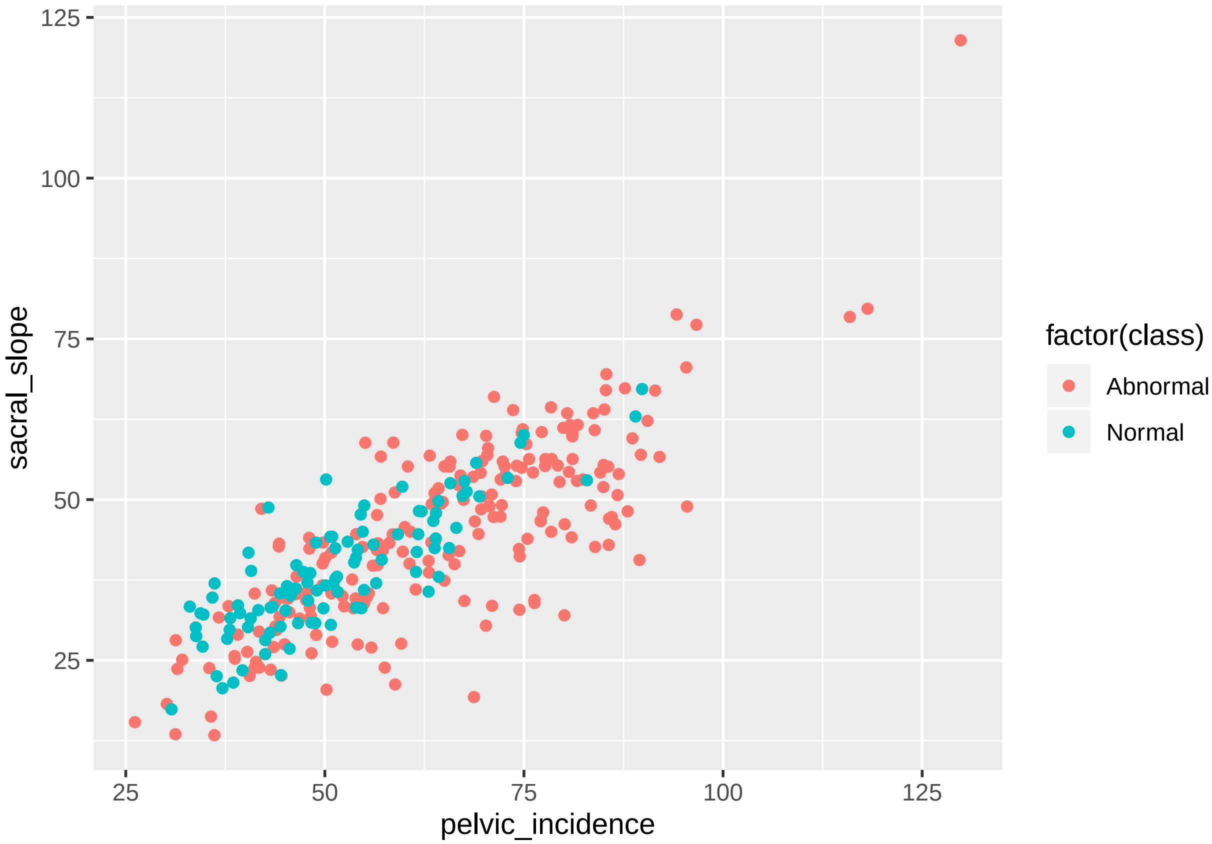



```r
# Scatter plot for pelvic_radius and sacral_slope for distribution of patients
ggplot(ortho_data, aes(x = pelvic_radius, y = sacral_slope)) +
  geom_point(aes(color = factor(class)))
```



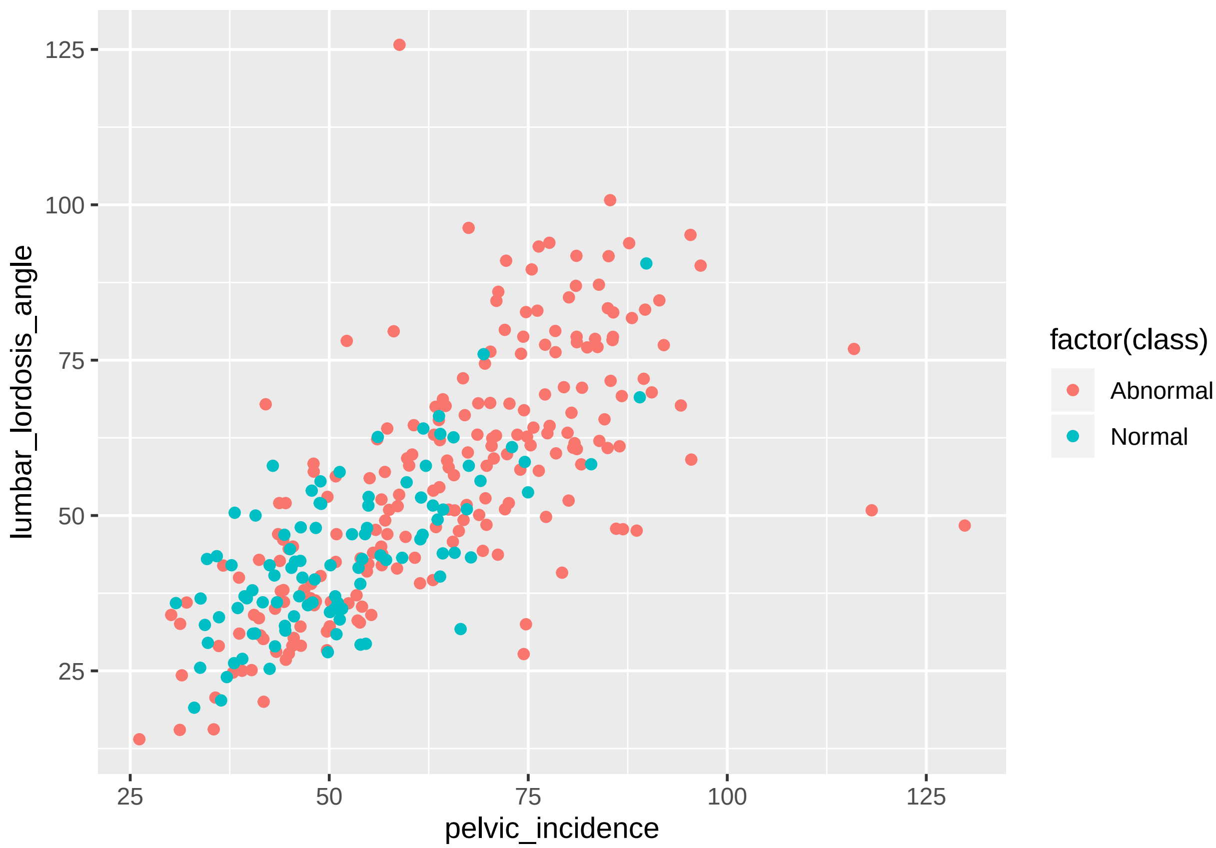

```
# Scatter plot for pelvic_incidence and sacral_slope for distribution of patients

ggplot(ortho_data, aes(x = pelvic_incidence, y = sacral_slope)) +
  geom_point(aes(color = factor(class)))
```

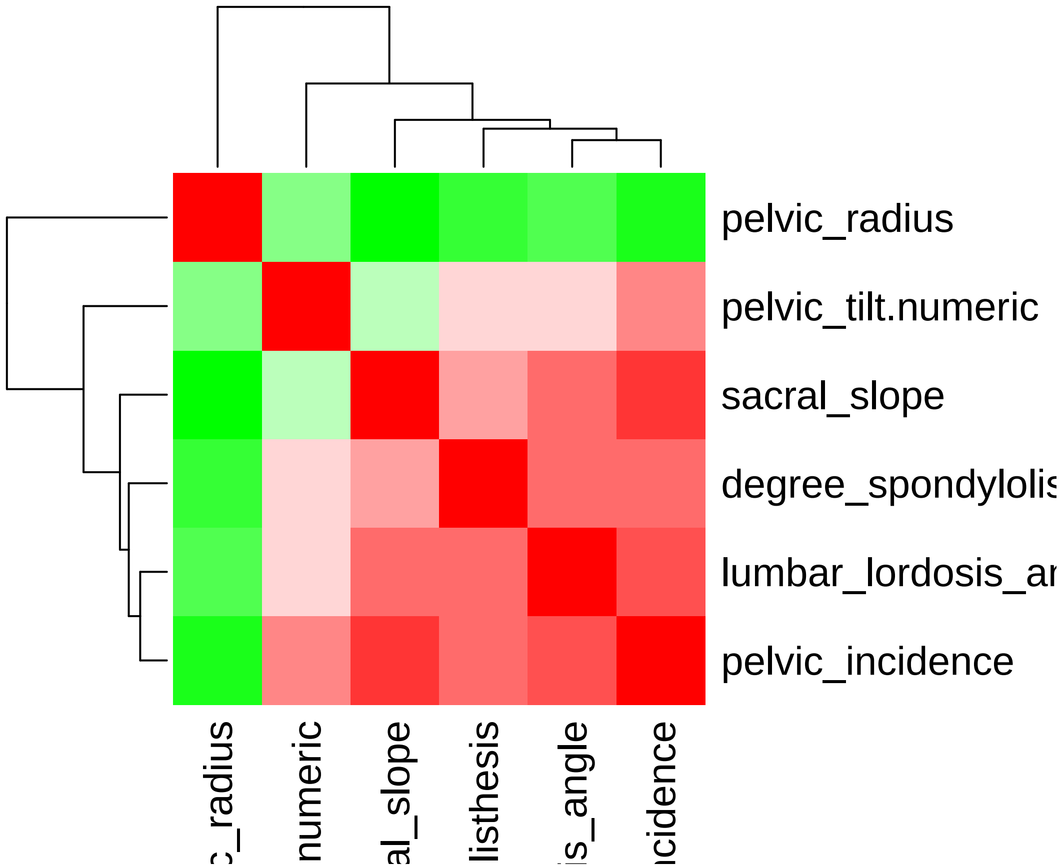



```
# Scatter plot for pelvic_incidence and Lumbar Lordosis Angle for distribution of patients

ggplot(ortho_data, aes(x = pelvic_incidence, y = lumbar_lordosis_angle)) +
  geom_point(aes(color = factor(class)))
```




```r
#heat map
palette = colorRampPalette(c("green", "white", "red")) (20)
heatmap(x = cc, col = palette, symm = TRUE)
```



Prepare Data

This step involves, cleaning, normalizing and splicing of data. The Biomechanical Orthopedic data set will be used for classification, which is an example of predictive modeling. The last attribute of the data set, class, will be the target variable or the variable that I want to predict. 1) Check of null values 2) normalize data 3) Split data in test and train data sets

Normalize data: Looking at the summary output it is seen that all the features are not in consistent range.Look at the minimum and maximum values of all the (numerical) attributes. If one attribute has a wide range of values,need to normalize the dataset, because this means that the distance will be dominated by this feature. In the current dataset it is degree_spondylolosthesis that has wide range from -11.058 to 418.543

In order to assess the performance of the mode data set is divided into two parts: a training set and a test set. The first is used to train the system, while the second is used to evaluate the learned or trained system. 70% of the original data set is as the training set, while the 10% that remains will compose the test set.

```
# check if isna

data_na<-apply(ortho_data, 2, function(x) any(is.na(x)))
data_na
```

```
##         pelvic_incidence      pelvic_tilt.numeric    lumbar_lordosis_angle
##                    FALSE                    FALSE                    FALSE
##             sacral_slope            pelvic_radius degree_spondylolisthesis
##                    FALSE                    FALSE                    FALSE
```

```
##                    class
##                    FALSE
```

```r
# There is no na in the dataset

# Build normalize function
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x))) }

# Apply normalize function to the orthopedic data set

ortho_data.norm<-as.data.frame(lapply(ortho_data[,c(1,2,3,4,5,6)], normalize))

# Summary for normalized data
summary(ortho_data.norm)
```

```
##  pelvic_incidence pelvic_tilt.numeric lumbar_lordosis_angle  sacral_slope
##  Min.   :0.0000   Min.   :0.0000      Min.   :0.0000        Min.   :0.0000
##  1st Qu.:0.1956   1st Qu.:0.3076      1st Qu.:0.2058        1st Qu.:0.1849
##  Median :0.3139   Median :0.4093      Median :0.3183        Median :0.2687
##  Mean   :0.3313   Mean   :0.4304      Mean   :0.3394        Mean   :0.2738
##  3rd Qu.:0.4507   3rd Qu.:0.5122      3rd Qu.:0.4385        3rd Qu.:0.3639
##  Max.   :1.0000   Max.   :1.0000      Max.   :1.0000        Max.   :1.0000
##  pelvic_radius    degree_spondylolisthesis
##  Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.4369   1st Qu.:0.02947
##  Median :0.5182   Median :0.05313
##  Mean   :0.5145   Mean   :0.08695
##  3rd Qu.:0.5956   3rd Qu.:0.12185
##  Max.   :1.0000   Max.   :1.00000
```

```r
# Split Data into Training and Test Sets

# To make training and test sets,  set a seed. This is a number of R's random number generator.
# The major advantage of setting a seed is that it gives same sequence of random numbers.
set.seed(123)
ortho_data.ind <- sample(1:nrow(ortho_data.norm),size=nrow(ortho_data.norm)*0.7,replace = FALSE) #rando

ortho_data.train <- ortho_data.norm[ortho_data.ind,] # 70% training data

# Inspect training set
head(ortho_data.train)
```

```
##     pelvic_incidence pelvic_tilt.numeric lumbar_lordosis_angle sacral_slope
## 179        0.5256865           0.5876264             0.4196986    0.3788822
## 14         0.2644930           0.4825382             0.1709289    0.1827127
## 195        0.4484296           0.4551798             0.4832544    0.3733742
## 306        0.2098221           0.3602926             0.1968814    0.1935909
## 118        0.5776410           0.8092195             0.3031205    0.3139261
## 299        0.3892446           0.4903408             0.1586459    0.2983693
##     pelvic_radius degree_spondylolisthesis
## 179     0.5379260               0.14787138
## 14      0.4396688               0.04213903
```

```
## 195     0.5041576              0.08482849
## 306     0.5093802              0.01585839
## 118     0.5593208              0.17003315
## 299     0.6325551              0.02927222
```

```
ortho_data.test <- ortho_data.norm[-ortho_data.ind,] # remaining 30% test data

# Inspect test set
head(ortho_data.test)
```

```
##    pelvic_incidence pelvic_tilt.numeric lumbar_lordosis_angle sacral_slope
## 2        0.12450104           0.2967831            0.09857833  0.144629352
## 3        0.41166648           0.5139323            0.32299466  0.307660537
## 8        0.18535588           0.3091900            0.13458053  0.196591648
## 12       0.04903709           0.4335087            0.01342373  0.001384728
## 15       0.30044817           0.5491263            0.29532214  0.182712658
## 18       0.04945783           0.1732483            0.16612314  0.136628277
##    pelvic_radius degree_spondylolisthesis
## 2      0.4766489             0.0363649708
## 3      0.3860969             0.0175229033
## 8      0.5074553             0.0008899132
## 12     0.5374088             0.0269038552
## 15     0.5024710             0.0391645164
## 18     0.6337221             0.0341740122
```

```
# Compose 'class' training labels
ortho_data.trainLabels <- ortho_data[ortho_data.ind,7]

# Inspect result
#print(ortho_data.trainLabels)

# Compose 'class' test labels
ortho_data.testLabels <- ortho_data[-ortho_data.ind, 7]

# Inspect result
#print(ortho_data.testLabels)
```

Models/ Algorithms and Evaluation

KNN K Nearest Neighbour Algorithm

Build Classifier to find the k nearest neighbour for the training set using the knn() function, which uses the Euclidian distance measure to find the k-nearest neighbours to the new instance. KNN model is done in 2 ways using caret and class package. In Class package, We have to decide on the number of neighbors (k). There are several rules of thumb, one being the square root of the number of observations in the training set. In this case, we select 16 as the number of neighbors, which is approximately the square root of our sample size N = 217.Infact the model was run for both k=16 and K=17.

In caret package, the function picks the optimal number of neighbors (k) for you.

```
# To find the k parameter for the knn function
nr<-NROW(ortho_data.trainLabels)
```

```r
# sqrt of 217 is 14.7
sqrt(nr)
```

```
## [1] 14.73092
```

```r
dim(ortho_data.trainLabels)
```

```
## NULL
```

```r
dim(ortho_data.train)
```

```
## [1] 217   6
```

```r
ortho_data.knn.15 <- knn(train=ortho_data.train, test=ortho_data.test, cl=ortho_data.trainLabels, k=14)
ortho_data.knn.16 <- knn(train=ortho_data.train, test=ortho_data.test, cl=ortho_data.trainLabels, k=15)

# Inspect
ortho_data.knn.15
```

```
##  [1] Abnormal Abnormal Abnormal Abnormal Abnormal Normal   Abnormal Abnormal
##  [9] Abnormal Normal   Abnormal Abnormal Normal   Abnormal Abnormal Abnormal
## [17] Abnormal Abnormal Abnormal Abnormal Abnormal Abnormal Normal   Abnormal
## [25] Abnormal Abnormal Abnormal Abnormal Abnormal Normal   Abnormal Abnormal
## [33] Abnormal Normal   Abnormal Abnormal Abnormal Abnormal Abnormal Abnormal
## [41] Abnormal Abnormal Abnormal Abnormal Abnormal Abnormal Abnormal Abnormal
## [49] Normal   Abnormal Abnormal Abnormal Abnormal Abnormal Abnormal Abnormal
## [57] Abnormal Abnormal Abnormal Abnormal Normal   Abnormal Normal   Normal
## [65] Abnormal Normal   Abnormal Normal   Normal   Normal   Normal   Abnormal
## [73] Normal   Abnormal Normal   Normal   Normal   Normal   Normal   Abnormal
## [81] Normal   Normal   Normal   Normal   Normal   Abnormal Normal   Abnormal
## [89] Normal   Abnormal Abnormal Abnormal Normal
## Levels: Abnormal Normal
```

```r
# ortho_data.knn.15 stores the knn() function that takes as arguments
# the training set, the test set, the train labels and the amount of
# neighbours to find with this algorithm. The result of this function
# is a factor vector with the predicted classes for each row of the test data.

# Note that the test labels will be used to see if the model is good at prediction.

# Model Evaluation
# An essential next step in machine learning is the evaluation
# of the model's performance. Analyze the degree of correctness of the model's predictions.

# Put `ortho_data.testLabels` in a data frame
ortho_dataTestLabels <- data.frame(ortho_data.testLabels)

# Merge `ortho_data.knn.15` and `ortho_data.testLabels`
merge <- data.frame(ortho_data.knn.16, ortho_data.testLabels)
```

```r
# Inspect 'merge'
#merge


CrossTable(x = ortho_data.testLabels, y = ortho_data.knn.16, prop.chisq=FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  93
##
##
##                       | ortho_data.knn.16
## ortho_data.testLabels |  Abnormal |    Normal | Row Total |
## ----------------------|-----------|-----------|-----------|
##              Abnormal |        54 |         8 |        62 |
##                       |     0.871 |     0.129 |     0.667 |
##                       |     0.857 |     0.267 |           |
##                       |     0.581 |     0.086 |           |
## ----------------------|-----------|-----------|-----------|
##                Normal |         9 |        22 |        31 |
##                       |     0.290 |     0.710 |     0.333 |
##                       |     0.143 |     0.733 |           |
##                       |     0.097 |     0.237 |           |
## ----------------------|-----------|-----------|-----------|
##          Column Total |        63 |        30 |        93 |
##                       |     0.677 |     0.323 |           |
## ----------------------|-----------|-----------|-----------|
##
##
```

```r
##create confusion matrix
tab <- table(ortho_data.knn.16, ortho_data.testLabels)

##this function divides the correct predictions by total number of predictions that tell us how accurat

accuracy <- function(x){sum(diag(x)/(sum(rowSums(x))) * 100}
accuracy(tab)
```

```
## [1] 81.72043
```

```r
#Calculate the proportion of correct classification for k = 15,16
ACC.15 <- 100 * sum( ortho_data.testLabels == ortho_data.knn.15)/NROW( ortho_data.testLabels)
ACC.16 <- 100 * sum( ortho_data.testLabels == ortho_data.knn.16)/NROW( ortho_data.testLabels)
ACC.15
```

```
## [1] 80.64516
```

```r
ACC.16
```

```
## [1] 81.72043
```

```r
# confusion Matrix
confusionMatrix(table(ortho_data.knn.16, ortho_data.testLabels))
```

```
## Confusion Matrix and Statistics
##
##                   ortho_data.testLabels
## ortho_data.knn.16 Abnormal Normal
##          Abnormal       54      9
##          Normal          8     22
##
##                Accuracy : 0.8172
##                  95% CI : (0.7235, 0.8898)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 0.0009547
##
##                   Kappa : 0.5854
##
##  Mcnemar's Test P-Value : 1.0000000
##
##             Sensitivity : 0.8710
##             Specificity : 0.7097
##          Pos Pred Value : 0.8571
##          Neg Pred Value : 0.7333
##              Prevalence : 0.6667
##          Detection Rate : 0.5806
##    Detection Prevalence : 0.6774
##       Balanced Accuracy : 0.7903
##
##        'Positive' Class : Abnormal
##
```

```r
################### Using Caret Package
# Create index to split based on labels
index <- createDataPartition(ortho_data$class, p=0.7, list=FALSE)
# Subset training set with index
ortho.training <- ortho_data[index,]

# Subset test set with index
ortho.test <- ortho_data[-index,]

# Overview of algos supported by caret
```

```r
#names(getModelInfo())

# Train a model
model_knn <- train(ortho.training[, 1:6], ortho.training[, 7], method='knn')

# Predict the labels of the test set
predictions_knn<-predict(object=model_knn,ortho.test[,1:6])

# Evaluate the predictions
table(predictions_knn)
```

```
## predictions_knn
## Abnormal   Normal
##       63       30
```

```r
# Confusion matrix kNN
confusionMatrix(predictions_knn,ortho.test[,7])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Abnormal Normal
##   Abnormal       54      9
##   Normal          9     21
##
##                Accuracy : 0.8065
##                  95% CI : (0.7115, 0.8811)
##     No Information Rate : 0.6774
##     P-Value [Acc > NIR] : 0.004057
##
##                   Kappa : 0.5571
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.8571
##             Specificity : 0.7000
##          Pos Pred Value : 0.8571
##          Neg Pred Value : 0.7000
##              Prevalence : 0.6774
##          Detection Rate : 0.5806
##    Detection Prevalence : 0.6774
##       Balanced Accuracy : 0.7786
##
##        'Positive' Class : Abnormal
##
```

SVM Linear (Support Vector Machine)

support vector machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. They are mostly used in classification problems.

```r
# Train a model

model_svm <- train(ortho.training[, 1:6], ortho.training[, 7],method='svmLinear',trControl=trainControl


# Predict the labels of the test set
predictions_svm<-predict(object=model_svm,ortho.test[,1:6])

# Evaluate the predictions
table(predictions_svm)
```

```
## predictions_svm
## Abnormal   Normal
##       65       28
```

```r
# Confusion matrix SVM
confusionMatrix(predictions_svm,ortho.test[,7])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Abnormal Normal
##   Abnormal       56      9
##   Normal          7     21
##
##                Accuracy : 0.828
##                  95% CI : (0.7357, 0.8983)
##     No Information Rate : 0.6774
##     P-Value [Acc > NIR] : 0.0008401
##
##                   Kappa : 0.5994
##
##  Mcnemar's Test P-Value : 0.8025873
##
##             Sensitivity : 0.8889
##             Specificity : 0.7000
##          Pos Pred Value : 0.8615
##          Neg Pred Value : 0.7500
##              Prevalence : 0.6774
##          Detection Rate : 0.6022
##    Detection Prevalence : 0.6989
##       Balanced Accuracy : 0.7944
##
##        'Positive' Class : Abnormal
##
```

Random Forest Algorithm

Random Forest is one such very powerful ensembling machine learning algorithm. It works by creating multiple decision trees and combining the output generated by each of the decision trees. Decision tree is a classification model which works on the concept of information gain at every node.

```r
# Train a model
model_rf <- train(ortho.training[, 1:6], ortho.training[, 7], method='rf')

# Predict the labels of the test set
predictions_rf<-predict(object=model_rf,ortho.test[,1:6])

# Evaluate the predictions
table(predictions_rf)
```

```
## predictions_rf
## Abnormal   Normal
##       62       31
```

```r
# Confusion matrix Random Forest
confusionMatrix(predictions_rf,ortho.test[,7])
```

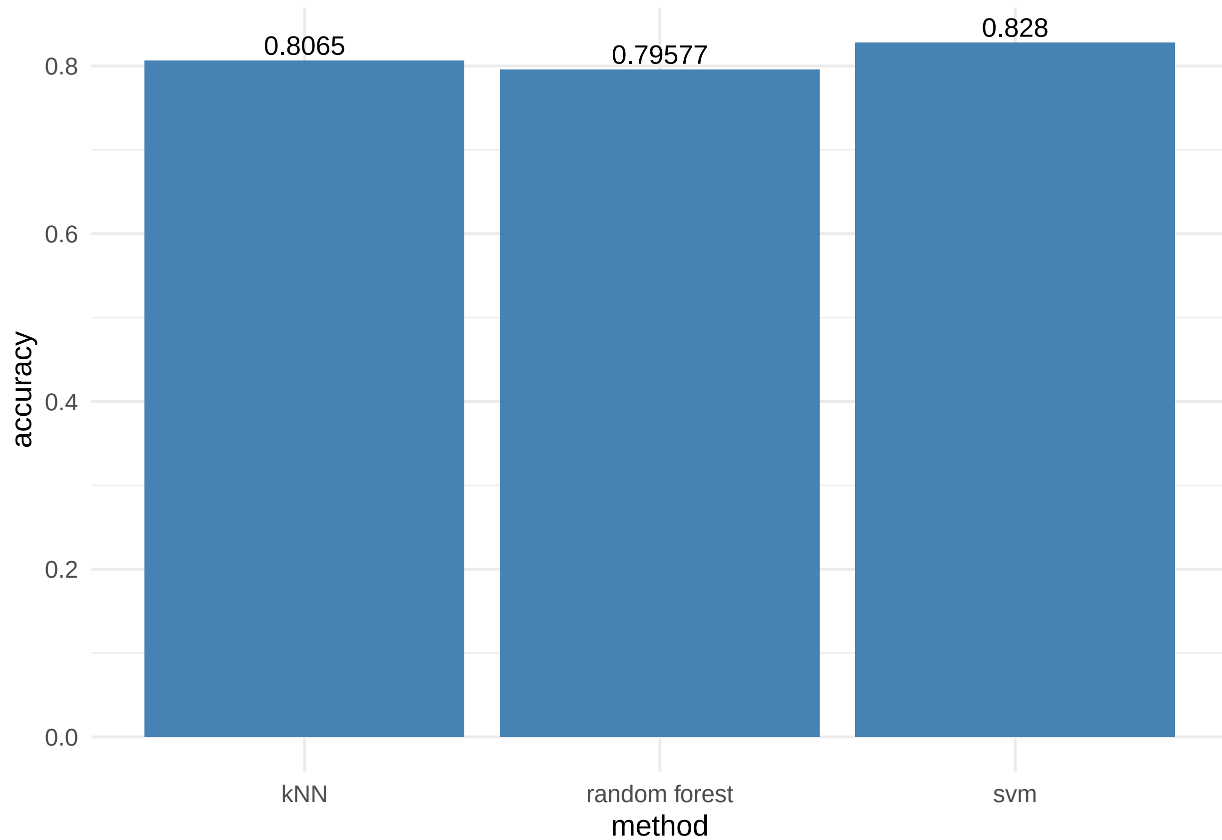
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Abnormal Normal
##   Abnormal       53      9
##   Normal         10     21
##
##                Accuracy : 0.7957
##                  95% CI : (0.6995, 0.8723)
##     No Information Rate : 0.6774
##     P-Value [Acc > NIR] : 0.008095
##
##                   Kappa : 0.5366
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.8413
##             Specificity : 0.7000
##          Pos Pred Value : 0.8548
##          Neg Pred Value : 0.6774
##              Prevalence : 0.6774
##          Detection Rate : 0.5699
##    Detection Prevalence : 0.6667
##       Balanced Accuracy : 0.7706
##
##        'Positive' Class : Abnormal
##
```

Analysis and Conclusion

After comparing KNN, Supported Vector Machine (SVM) and Random Forest Algorithms the prediction for this data set is highest for KNN algorithm. The results of the Cross Table indicate that our model did not predict mother's job very well. To read the Cross Table, we begin by examining the top-left to bottom-right diagonal of the matrix. The diagonal of the matrix represents the number of cases that were correctly classified for each category. If the model correctly classified all cases, the matrix would have zeros everywhere but the diagonal. In this case, we see that the numbers are quite high in the off-diagonals, indicating that our model did not successfully classify our outcome based on our predictors.

Confusion matrix or error matrix is used for summarizing the performance of a classification algorithm. Calculating a confusion matrix gives an idea of where the classification model is right and what types of errors it is making. A confusion matrix is used to check the performance of a classification model on a set of test data for which the true values are known. It can be seen that random forest performed with 79%, knn with 80% accuracy and svn linear with 82% accuracy.

```
method <- c('kNN','svm','random forest')
accuracy <- c(0.8065, 0.828, 0.79577)
df <- data.frame(method,accuracy)
ggplot(data=df, aes(x=method, y=accuracy)) +
  geom_bar(stat="identity", fill="steelblue")+
  geom_text(aes(label=accuracy), vjust=-0.3, size=3.5)+
  theme_minimal()
```



Next Steps.. Optimization:

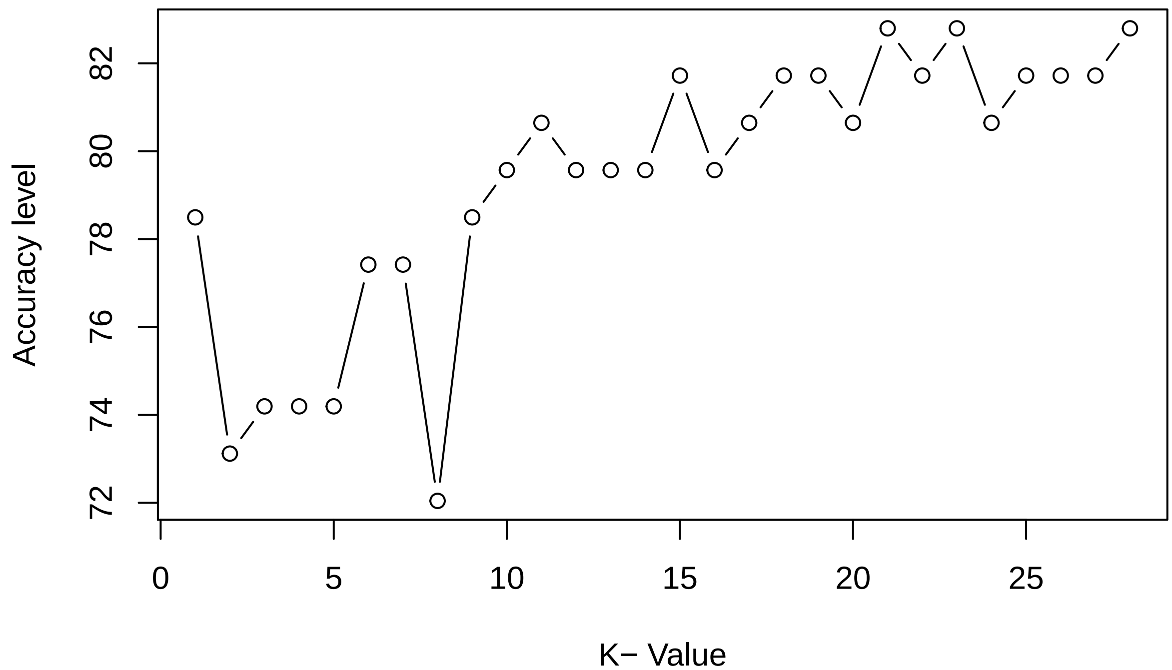
For kNN algorithm, the tuning parameters are 'k' value and number of 'features/attributes selection. Optimum 'k' value can be found using graph below. It was found that max accuracy in knn is at k=15.

```
# Optimization
i=1
k.optm=1
```

```r
for (i in 1:28){
  knn.mod <- knn(train=ortho_data.train, test=ortho_data.test, cl=ortho_data.trainLabels, k=i)
  k.optm[i] <- 100 * sum( ortho_data.testLabels == knn.mod)/NROW( ortho_data.testLabels)
  k=i
  cat(k,'=',k.optm[i],'
')
}
```

```
## 1 = 78.49462
## 2 = 73.11828
## 3 = 74.19355
## 4 = 74.19355
## 5 = 74.19355
## 6 = 77.41935
## 7 = 77.41935
## 8 = 72.04301
## 9 = 78.49462
## 10 = 79.56989
## 11 = 80.64516
## 12 = 79.56989
## 13 = 79.56989
## 14 = 79.56989
## 15 = 81.72043
## 16 = 79.56989
## 17 = 80.64516
## 18 = 81.72043
## 19 = 81.72043
## 20 = 80.64516
## 21 = 82.7957
## 22 = 81.72043
## 23 = 82.7957
## 24 = 80.64516
## 25 = 81.72043
## 26 = 81.72043
## 27 = 81.72043
## 28 = 82.7957
```

```r
#Accuracy plot
plot(k.optm, type="b", xlab="K- Value",ylab="Accuracy level")
```


Accuracy level
72
74
76
78
80
82
0
5
10
15
20
25
K− Value