Nirma university

Institute of technology

Department of mechanical engineering

Course: Basics of Machine Learning

Project Report

Prepared by

Shah Shrey (19BME134)

Alpesh Gohel (20BME503)

TABLE OF CONTENTS

ABSTRACT

As a part of the personal project in the basics of machine learning course, Biomechanical data is considered to perform the logistic regression, decision trees using party and rpart libraries and random forest classification, k nearest neighbour and lastly support vector machine.

We have used several different models to check which model gives the highest amount of accuracy and can be utilized in the real world.

The data is about the biomechanical features of orthopaedic patients which can be used to classify whether the patient's condition is normal or abnormal. The data set further includes that if the patient is classified with an abnormal condition, then which condition was the patient diagnosed with – Hernia or Spondylolisthesis.

Biomedical data consists of the following sub-data:

- pelvic incidence
- pelvic tilt
- lumbar lordosis angle
- sacral slope
- pelvic radius
- grade of spondylolisthesis

After training the models with the following data and testing it with testing dataset, it will be concluded which model will be best suitable to predict the same sub-data.

About Dataset

Context

The data have been organized in two different but related classification tasks.

- column*3C*weka.csv (file with three class labels)
    - The first task consists in classifying patients as belonging to one out of three categories: Normal (100 patients), Disk Hernia (60 patients) or Spondylolisthesis (150 patients).
- column*2C*weka.csv (file with two class labels)
    - For the second task, the categories Disk Hernia and Spondylolisthesis were merged into a single category labelled as 'abnormal'. Thus, the second task consists in classifying patients as belonging to one out of two categories: Normal (100 patients) or Abnormal (210 patients).

Content

Field Descriptions:

Each patient is represented in the data set by six biomechanical attributes derived from the shape and orientation of the pelvis and lumbar spine (each one is a column):

Dataset –

```
data <- read.csv("column_2C_weka.csv")

head(data)
##   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
##   pelvic_radius degree_spondylolisthesis    class
## 1      98.67292               -0.254400 Abnormal
## 2     114.40543                4.564259 Abnormal
## 3     105.98514               -3.530317 Abnormal
## 4     101.86850               11.211523 Abnormal
## 5     108.16872                7.918501 Abnormal
## 6     130.32787                2.230652 Abnormal


summary(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          Length:310
##  1st Qu.:110.71   1st Qu.:  1.604          Class :character
##  Median :118.27   Median : 11.768          Mode  :character
##  Mean   :117.92   Mean   : 26.297
##  3rd Qu.:125.47   3rd Qu.: 41.287
##  Max.   :163.07   Max.   :418.543
```

- Now we shall start applying different models to test the data set for classification -

Logistic Regression

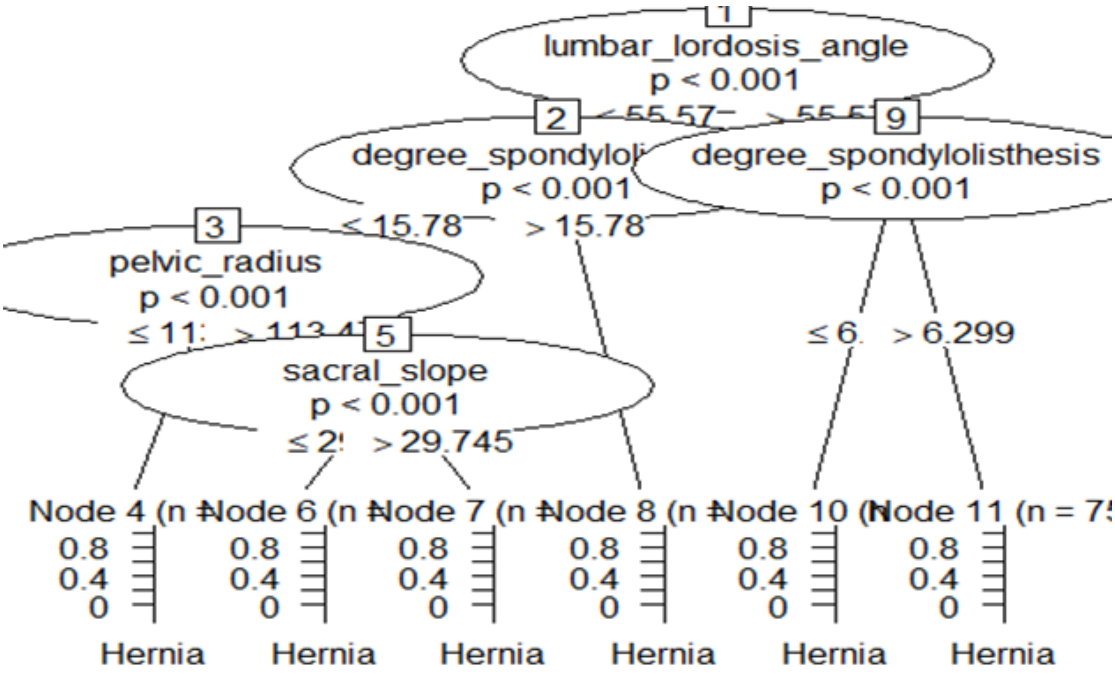
- Logistic regression is used for classification of only 2 class either 1 or 0. So we take the data as abnormal and normal instead of classifying diseases. The model will predict whether the patient has an abnormality or not.
- The data is read first then it is separated by the features and its response which is the classification. We convert the characters “Abnormal” and “Normal” to 1’s and 0’s for the sigmoid function to be used in regression. After converting we reattach the column to the features.
- The features are also normalized before reattaching for easier calculation and mapping. The data is then split into training and testing data. We use the “**glm**” function to train the model first. We can also manually train the model using the iterative method that we know.
- After the training process is complete, the predict function is used to use the model for the testing data which was left aside.
- We manually calculate the accuracy using the TP, TN, FP and FN values the accuracy turned out to be **84%**

```
pred <- predict(model, traindata, type = 'response')
j<-1; TP=0; TN=0; FP=0; FN=0;
for (i in pred)
{
  if (i>=0.5 && data_new[j,7]==1)
    TP <- TP + 1
  if (i>=0.5 && data_new[j,7]==0)
    FP <- FP + 1
  if (i<=0.5 && data_new[j,7]==0)
    TN <- TN + 1
  if (i<=0.5 && data_new[j,7]==1)
    FN <- FN + 1
  j <- j +1
}
accuracy <- (TP+TN)/(TN+TP+FP+FN)*100
accuracy

## [1] 84.83871
```

Using decision trees and Random forest

- We have already imported the party and rpart libraries at the start of program so here we shall use both of them to determine the accuracy and which one is better or is random forest better than both combined.
- We start with importing the data which has all the classes including the disease names and normality. Similar to last time we again split the data into training dataset and testing dataset

with split ratio as 70%. We first used the ctree function from the party library and created the tree model , the tree is shown below:

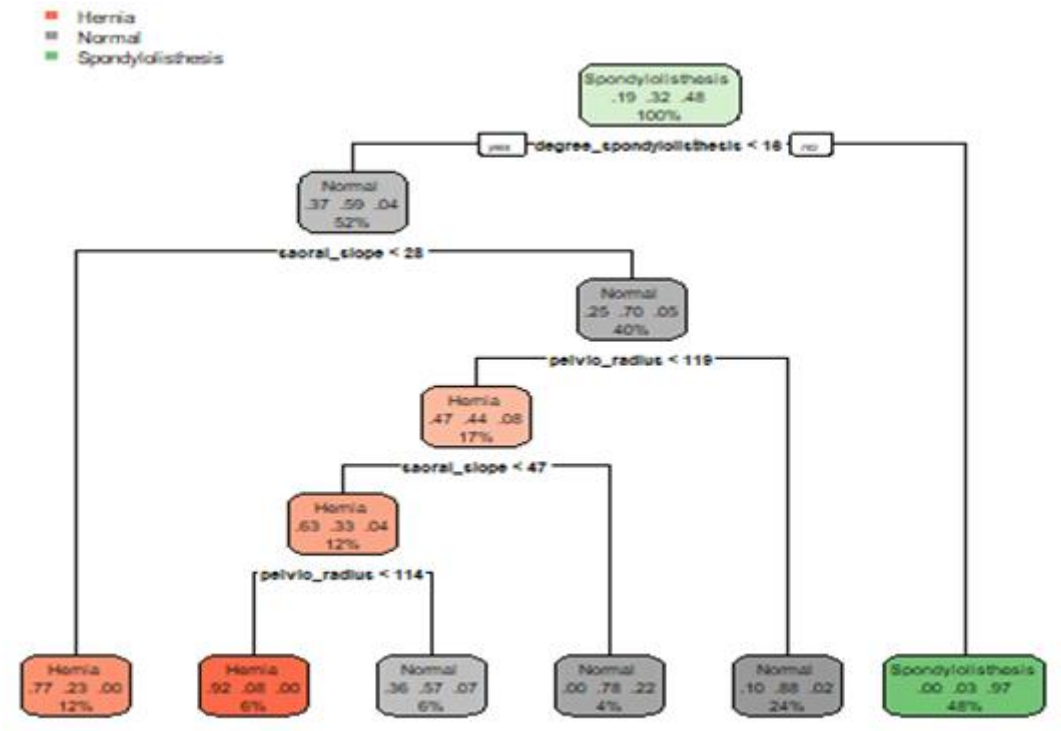


The training and testing accuracy was then found by predicting the testing data with this model which are also stated as below along with the confusion matrices of both of them. It can be seen that the accuracy for testing data is quite satisfactory given that this is a unpredictable area of studies (medical):

```
t1 <- table(Acutal = train$class, Predicted = train_pd_party)
t1

##                     Predicted
## Acutal               Hernia Normal Spondylolisthesis
##   Hernia                 35      7                 0
##   Normal                 13     53                 4
##   Spondylolisthesis       0      1               104

train_accuracy_party = sum(diag(t1)/sum(t1)) *100
train_accuracy_party

## [1] 88.47926

t1_tested <- table(Acutal = test$class, Predicted = test_pd1_party)
t1_tested

##                     Predicted
## Acutal               Hernia Normal Spondylolisthesis
##   Hernia                 10      8                 0
##   Normal                  5     25                 0
##   Spondylolisthesis       0      1                44

test_accuracy_party = sum(diag(t1_tested)/sum(t1_tested)) *100
test_accuracy_party
```

```
## [1] 84.94624
```

- Now we use the rpart library to create another tree which uses the rpart treeing method. The tree is also displayed below. The training and testing accuracy was then found for the tree2 also and it can be seen that the rpart package is quite similar to the the accuracy that party

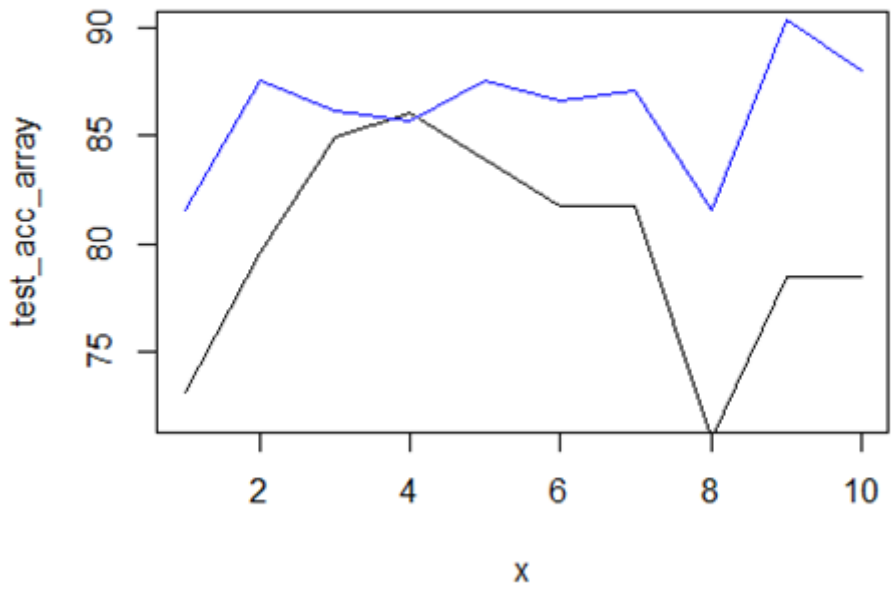


predicted

```
t_rpart_train <- table(Acutal = train$class, Predicted = pd_rpart_train)
t_rpart_train

##                     Predicted
## Acutal               Hernia Normal Spondylolisthesis
##   Hernia                 32     10                  0
##   Normal                  7     60                  3
##   Spondylolisthesis       0      4                101

accuracy1_train = sum (diag(t_rpart_train)/sum(t_rpart_train)) * 100
accuracy1_train

## [1] 88.94009

t_rpart_test <- table(Acutal = test$class, Predicted = pd_rpart_test)
t_rpart_test

##                     Predicted
## Acutal               Hernia Normal Spondylolisthesis
##   Hernia                 9      9                  0
##   Normal                 4     26                  0
##   Spondylolisthesis      0      1                 44
```

```
accuracy1_test = sum (diag(t_rpart_test)/sum(t_rpart_test))  * 100
accuracy1_test

## [1] 84.94624
```

- Lastly 10 fold cross validation was done and the training and testing accuracies were plotted. It can be clearly seen that the training accuracy was always higher than the testing accuracy but the accuracy of around 80% is still satisfactory for rudimentary purposes.



Random forests

- As we know that random forests are always a help in classifying more accurately than a single tree because the higher number of trees the higher will be the true prediction. In this section of the report we have used random forest with the number of trees equalling to 500 trees.
- But after plotting the graph with the number o trees, we concluded that 200 trees would suffice for this dataset rather than using 500 trees.
- The training accuracy was found to be 100% with random forests and the testing accuracy turned out to be 81% which is better than most of the models in the report. Below we have attached the accuracies and the graph for the number of trees w.r.t the classification.

```
table_RF_train <- table(Acutal = train$class, Predicted = train_pred_RF)
table_RF_train

##                    Predicted
## Acutal              Hernia Normal Spondylolisthesis
##   Hernia                42      0                 0
```

```
##   Normal                 0     70                 0
##   Spondylolisthesis      0      0               105
```

```
train_accuracy_RF = sum(diag(table_RF_train)/sum(table_RF_train)) *100
train_accuracy_RF
```

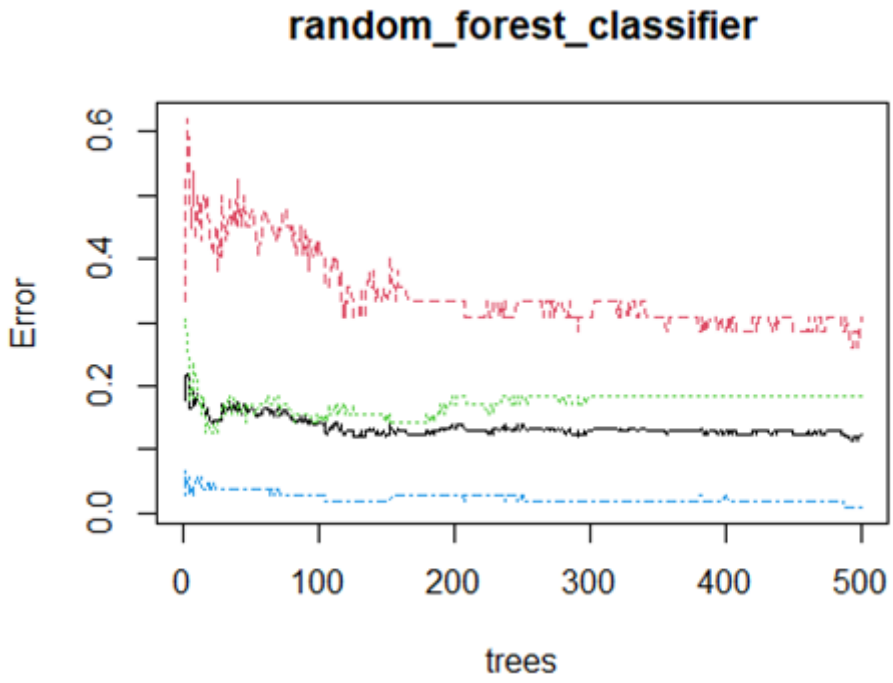
```
## [1] 100
```

```
table_RF_test <- table(Acutal = test$class, Predicted = test_pred_RF)
table_RF_test
```

```
##                     Predicted
## Acutal               Hernia Normal Spondylolisthesis
##   Hernia                11      6                 1
##   Normal                 6     21                 3
##   Spondylolisthesis      0      1                44
```

```
test_accuracy_RF = sum(diag(table_RF_test)/sum(table_RF_test)) *100
test_accuracy_RF
```

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



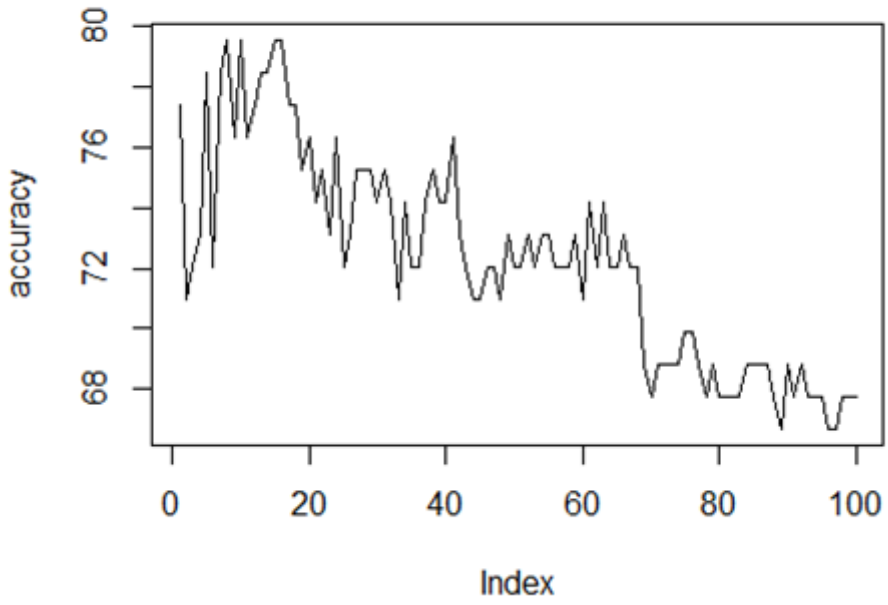

```
importance(random_forest_classifier)
```

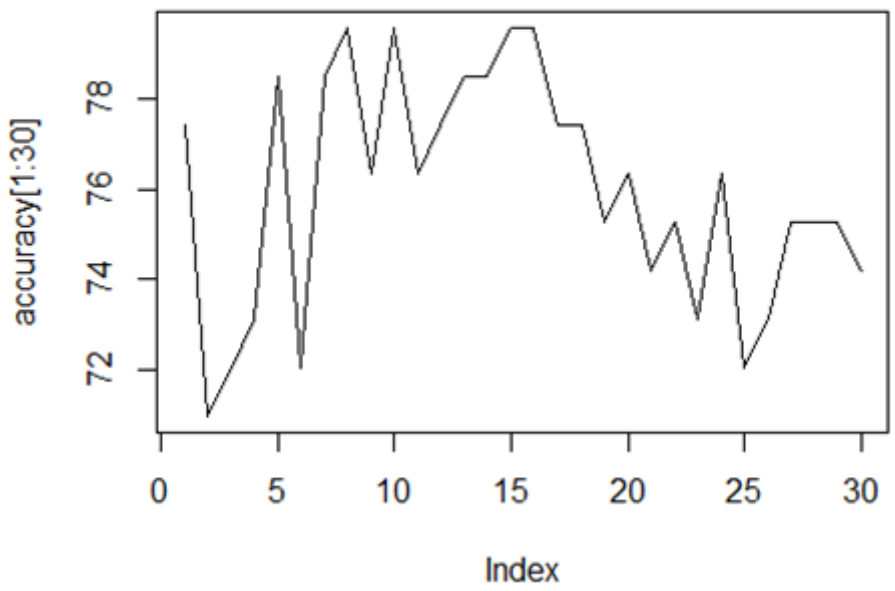
```
##                           MeanDecreaseGini
## pelvic_incidence                  15.09326
## pelvic_tilt                       13.43869
## lumbar_lordosis_angle             18.03142
## sacral_slope                      15.98425
## pelvic_radius                     15.76036
## degree_spondylolisthesis          56.38590
```

Using k nearest neighbour

- In K nearest neighbours we know that the new data is classified based on the k nearest neighbours. All we need to do is specify the k and the function calculated the Euclidian distance between the points and selects the nearest k points for the model to classify.
- The data is again read from the file and then the features and response are again separated to normalize the features. After using the minmax function to normalize the data. The response is attached back to the features and knn function is used to start classifying based on training an testing data.
- The accuracy first time turned out to be 70%. After some thinking we decided to loop over the values of k from 1 to 100 to find the best k. the graphs and code snippet is given below. From the graph we can see the maximum value of k lies somewhere in between the 15-20 region

```
for (i in 1:100)
{
  predict = knn(train[,-7],test[,-7],train[,7],k=i)
  table1 = table(actual = test[,7],predicted = predict)
  accuracy1 = sum (diag(table1)/sum(table1))
  accuracy[i]=accuracy1*100
}

plot(accuracy, t="l")
plot(accuracy[1:30],t="l")
```





Using support vector machine

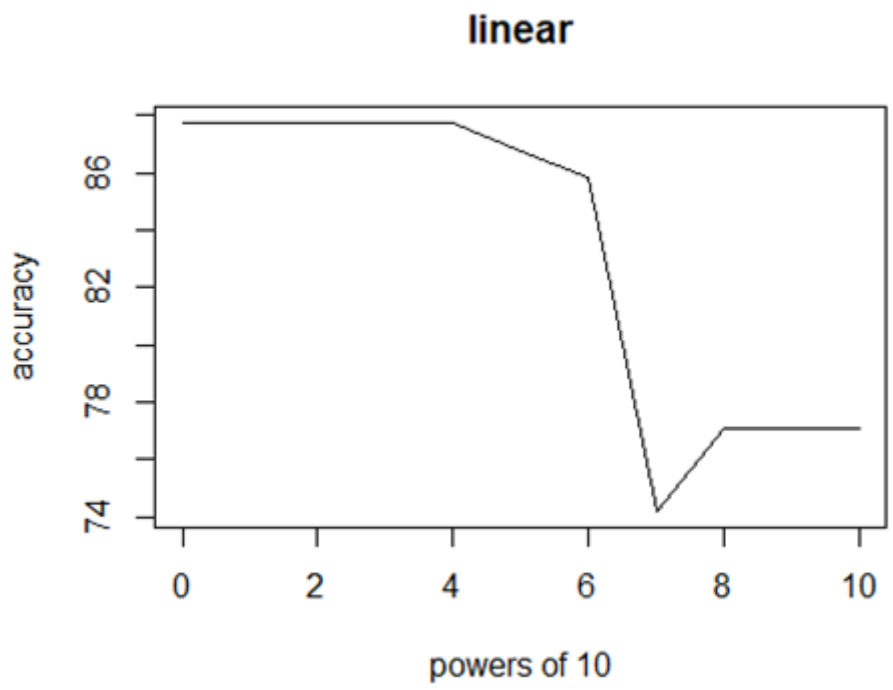
- Support vector machine is also a very very useful model where the classification is done based on error margins. We used the SVM classifier for 4 different types of kernels to classify on – Linear, Radial, Polynomial and Sigmoid.
- From the initial accuracy calculations, it can be seen that the Radial function has the highest accuracy being around 86% compared to other functions. The code snippet and accuracies are shown below.

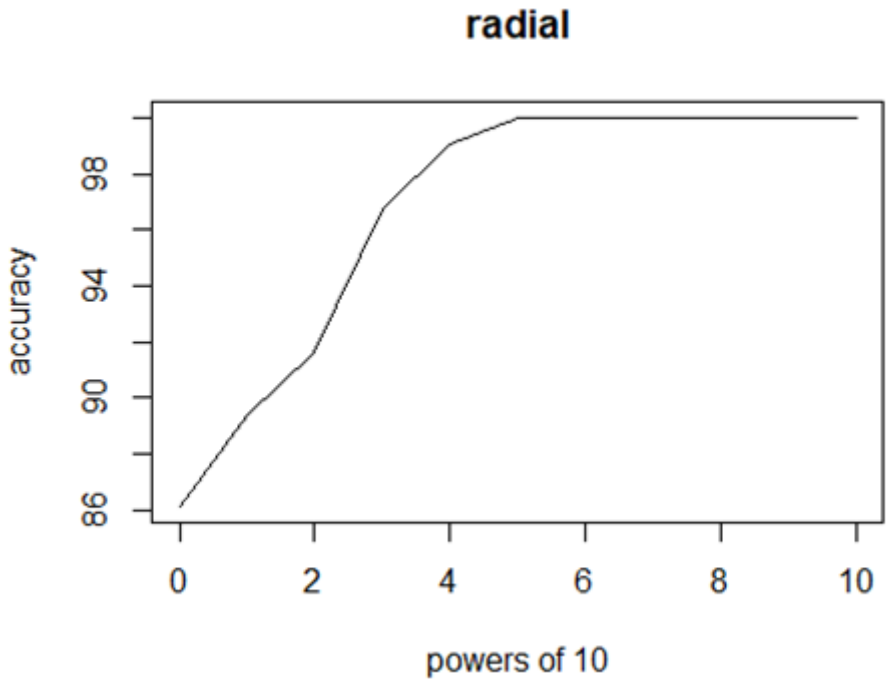
```
kernellist <- c("linear","radial","polynomial","sigmoid")

#using different kernels to see which one gives best model
accuracy_kernel <- c()
for( i in 1:4)
{
  model <- svm(class~.,
               data = data,kernel = kernellist[i])
  summary(model)
  pred <- predict(model,data=data)
  t <- table(actual = data$class, Predicted = pred)
  accuracy_kernel[i] = sum(diag(t))/sum(t) *100

}

accuracy_kernel

## [1] 87.74194 86.12903 82.58065 82.90323
```

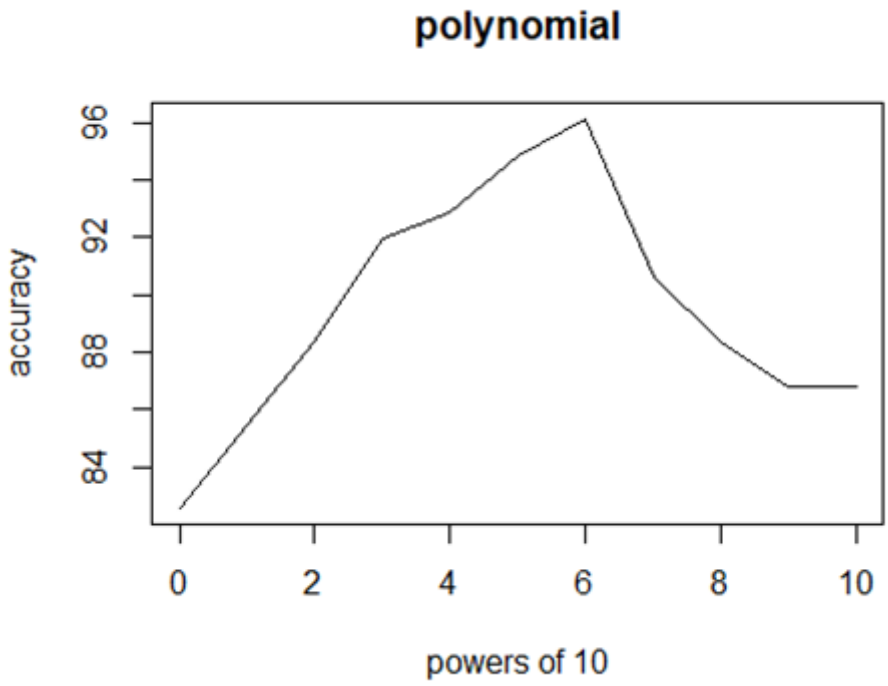
Checking the best cost

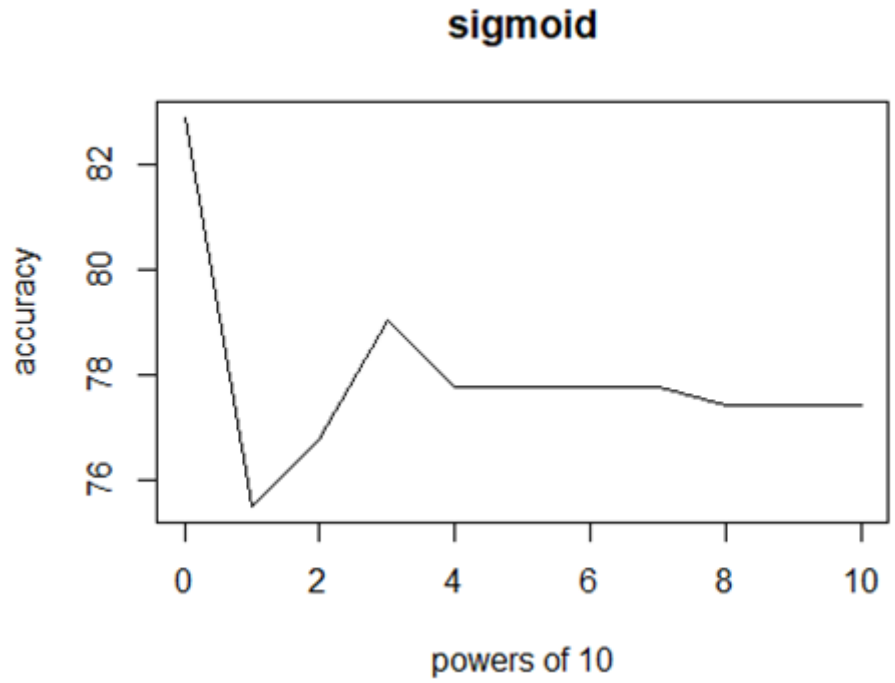
- After some thinking we realized we haven't included the cost at all being set at default cost as 1. So, we ran the loo again but this time created another loop inside of the kernel loop which iterates over different values of cost going by the geometric progression from 1 to 10,00,00,00,000. After each iteration of kernel the plot for accuracy vs the cost was plotted to see which cost is best for which kernel. The code snippet and the various graphs for various kernels against the different costs are displayed below.

```
gp <- c(10^seq(0,10))
gp_length <- c(0:10)

j=1
for (k in 1:4)
{
  for( i in gp)
  {
    model <- svm(class~.,
                 data = data,kernel = kernellist[k],cost = i)
    pred <- predict(model,data=data)
    t <- table(actual = data$class, Predicted = pred)
    accuracy_cost[j] = sum(diag(t))/sum(t) *100
    j <- j+1
    #plot(model,data,Petal.Length~Petal.Width)
  }

  plot(gp_length,accuracy_cost[(j-11):(j-1)], t="l",
       xlab="powers of 10",ylab="accuracy",main=kernellist[k])

}
```




radial
accuracy
86
90
94
98
0
2
4
6
8
10
powers of 10


polynomial
accuracy
84
88
92
96
0
2
4
6
8
10
powers of 10




```
accuracy_cost

##  [1]  87.74194  87.74194  87.74194  87.74194  87.74194  86.77419  85.80645
##  [8]  74.19355  77.09677  77.09677  77.09677  86.12903  89.35484  91.61290
## [15]  96.77419  99.03226 100.00000 100.00000 100.00000 100.00000 100.00000
## [22] 100.00000  82.58065  85.48387  88.38710  91.93548  92.90323  94.83871
## [29]  96.12903  90.64516  88.38710  86.77419  86.77419  82.90323  75.48387
## [36]  76.77419  79.03226  77.74194  77.74194  77.74194  77.74194  77.41935
## [43]  77.41935  77.41935
```

- From the graphs we can see that the worst function to use is the sigmoid function which is also a given because it is the simplest classifier out there. After sigmoid comes the linear function as it also decreases as the cost increases to tremendous value.
- The polynomial function is also good but only up to a certain extent as after some fixed cost point the accuracy starts declining.
- the radial function turns to to be the best as expected even from the loop that did not include cost at all. As the cost increases in the radial function, we can see that after some point the accuracy stops increases but the same time it does not start dropping as in polynomial function.
- So, we can conclude that the kernel “Radial” is the best kernel when using SVM for classification purposes for this data set.

CONCLUSION

From the series of operations performed above we can conclude that SVM function with the kernels: “Radial” and “Polynomial” with the appropriate cost values assigned to them. This model is followed by random forests which also have a significant accuracy considering the fact that the observational points were considerably less. From these tests we can successfully use these modes in everyday prediction of such diseases. But the models are only used ass a basis for decision making by doctors. Further consultation would still be advised. The table of accuracies is given below.

MODEL	TRAINING ACCURACY (%)	TESTING ACCURACY (%)
Logistic regression	-	84.83
Trees using party	88.47	84.94
Trees using rpart	88.94	84.94
Random forest	100	81.72
KNN	-	78.43
SVM linear (best cost)	-	87.74
SVM radial (best cost)	-	100
SVM polynomial (best cost)	-	96.12
SVM sigmoid (best cost)	-	82.90

Libraries used:

```
library(ClusterR)
library(cluster)
library(party)
library(rpart)
library(rpart.plot)
library(randomForest)
library(ggplot2)
library(e1071)
```

Acknowledgements

- The original dataset was downloaded from UCI ML repository:
- Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science
- Files were converted to CSV.