Classifying patients by analyzing the biomechanical features of orthopedic patients

Sadia Boksh

Introduction

In this project we will use the data set from Kaggle to study the biochemical features of orthopedic patients and classify the patients based on these features. Using this data set, we will train few classification machine learning models to classify patients as belonging to one out of three categories: Normal, Disk Hernia or Spondylolisthesis. We will perform the model fitting on scaled raw data. We will choose the best performing model by analyzing their accuracy.

Methods

Data Exploration

The data set used in this project can be found in https://www.kaggle.com/uciml/biomechanical-features-of-orthopedic-patients. This data set has 310 rows and 7 columns. There are 6 features and 1 response variable. There are no missing values in the data set.

```
head(df)
```

```
##   pelvic_incidence pelvic_tilt lumbar_lordosis_angle sacral_slope pelvic_radius
## 1         63.02782   22.552586              39.60912     40.47523      98.67292
## 2         39.05695   10.060991              25.01538     28.99596     114.40543
## 3         68.83202   22.218482              50.09219     46.61354     105.98514
## 4         69.29701   24.652878              44.31124     44.64413     101.86850
## 5         49.71286    9.652075              28.31741     40.06078     108.16872
## 6         40.25020   13.921907              25.12495     26.32829     130.32787
##   degree_spondylolisthesis  class
## 1                -0.254400 Hernia
## 2                 4.564259 Hernia
## 3                -3.530317 Hernia
## 4                11.211523 Hernia
## 5                 7.918501 Hernia
## 6                 2.230652 Hernia
```

This data set has three categories in response variable. The patients are to be classified into these three categories:

```
## [1] "Hernia"            "Spondylolisthesis" "Normal"
```

Below is the table that shows the proportion of patients in each class:

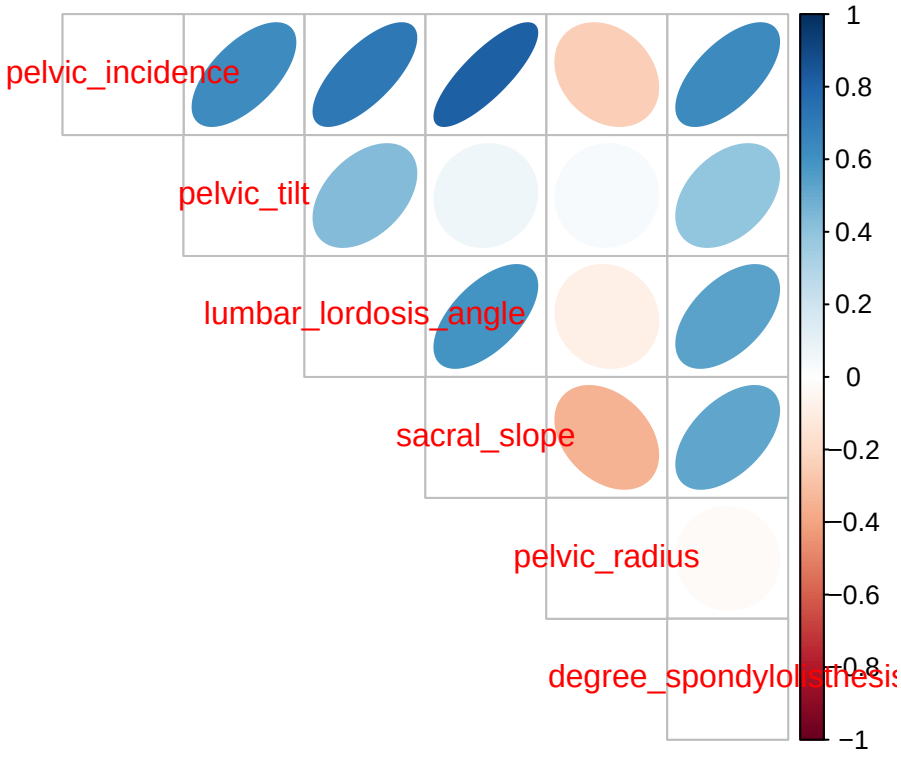
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 3 x 3
##   class                n  prop
##   <chr>            <int> <dbl>
## 1 Hernia              60 0.194
## 2 Normal             100 0.323
## 3 Spondylolisthesis  150 0.484
```

Plots

Below is the correlation plot:

```
##                           pelvic_incidence pelvic_tilt lumbar_lordosis_angle
## pelvic_incidence                 1.0000000  0.62919877            0.71728236
## pelvic_tilt                      0.6291988  1.00000000            0.43276386
## lumbar_lordosis_angle            0.7172824  0.43276386            1.00000000
## sacral_slope                     0.8149600  0.06234529            0.59838689
## pelvic_radius                   -0.2474672  0.03266781           -0.08034361
## degree_spondylolisthesis         0.6387427  0.39786228            0.53366701
##                           sacral_slope pelvic_radius degree_spondylolisthesis
## pelvic_incidence            0.81495999   -0.24746721               0.63874275
## pelvic_tilt                 0.06234529    0.03266781               0.39786228
## lumbar_lordosis_angle       0.59838689   -0.08034361               0.53366701
## sacral_slope                1.00000000   -0.34212835               0.52355746
## pelvic_radius              -0.34212835    1.00000000              -0.02606501
## degree_spondylolisthesis    0.52355746   -0.02606501               1.00000000
```



From the plot above we can see pelvic_incidence is highly correlated with sacral_slope.

Principle Component Analysis

We will apply PCA to explore the variable importance of each feature. Using the summary function we can see the variability explained by each PC:

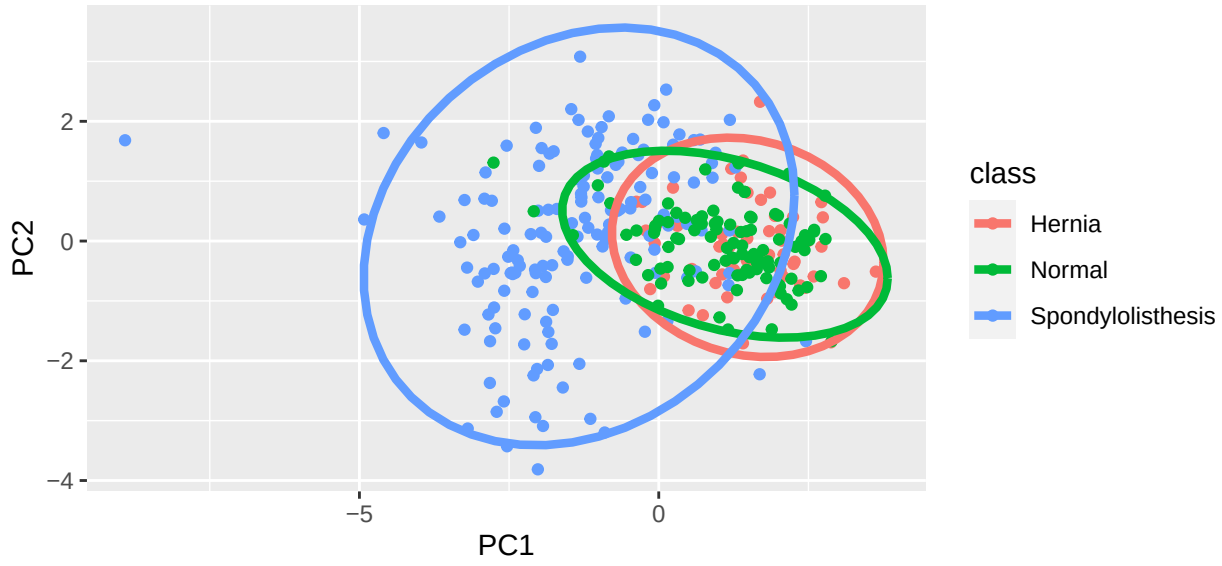
```
#transform to a matrix
x <- df[, 1:6] %>% as.matrix()

# scale and center the feature matrix
x_centered <- sweep(x, 2, colMeans(x))
scaled_X <- sweep(x_centered, 2, colSds(x), FUN = "/")


# principal components
pca <- prcomp(scaled_X)
summary(pca)$importance
```

```
##                             PC1     PC2      PC3       PC4       PC5
## Standard deviation     1.801605 1.09297 0.872405 0.6874067 0.5709795
## Proportion of Variance 0.540960 0.19910 0.126850 0.0787500 0.0543400
## Cumulative Proportion  0.540960 0.74006 0.866910 0.9456600 1.0000000
##                               PC6
## Standard deviation     1.935122e-10
```

```
## Proportion of Variance 0.000000e+00
## Cumulative Proportion  1.000000e+00
```

We can plot the first two PCS to see how they explain the variability:

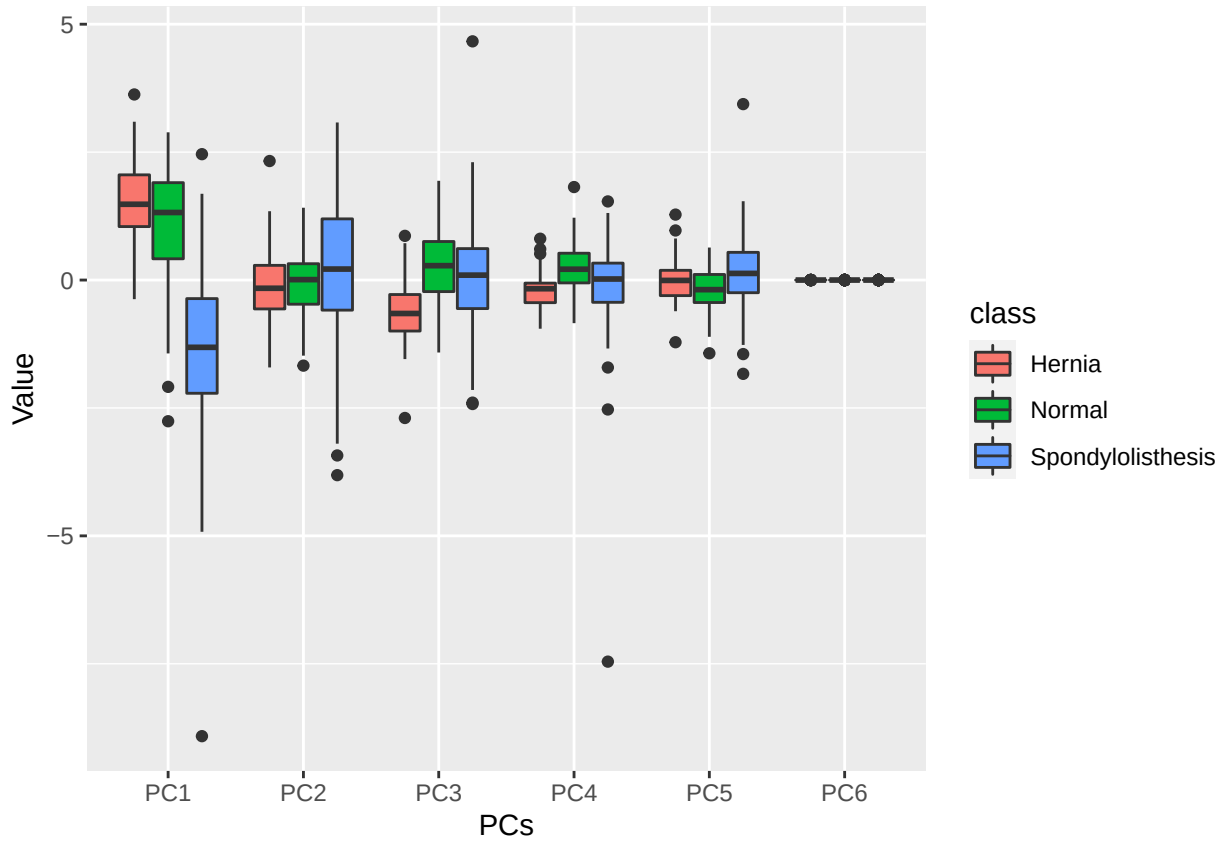
```
data.frame(pca$x[,1:2], class=df$class) %>%
    ggplot(aes(PC1,PC2, col = class))+
    geom_point() +
    coord_fixed(ratio = 1)+
    stat_ellipse(type="norm", lwd = 1.5)
```



We can see PC1 and PC2 has separated the patients into two categories: Spondylolisthesis and non Spondylolisthesis. Lower PC1 explains Spondylolisthesis and higher PC1 explains either Normal or Hernia.

We can also plot the first 10 PCs:

```
data.frame(pca$x[,1:6], class=df$class) %>%
    gather(PCs,Value, -class) %>%
    ggplot(aes(PCs,Value, fill = class))+
    geom_boxplot()
```



From the plot above we can see PC1 is not overlapping with other PCs.

Modeling

Now we will fit LDA, KNN and Random forest, SVM Linear models to the scaled data set and compare their accuracy.

First we will split the scaled data set to 80% train set and 20% test set. 80/20 split has been made to be able to train the model with as much data as possible at the same having a decent amount data for testing.

LDA

LDA makes predictions by estimating the probability that a new set of inputs belongs to each class. The class that gets the highest probability is the output class and a prediction is made. LDA model is used in this data set as it can handle continuous independent variable and a categorical dependent variable.

```
train_lda <- train(train_x, train_y, method = "lda")
pred_lda <-  predict(train_lda,  test_x)
acc_lda <- confusionMatrix(pred_lda,test_y)$overall['Accuracy']
acc_lda
```

```
##  Accuracy
## 0.8064516
```

K Nearest Neighbours

KNN algorithm can be used for classification where input consists of the k closest training examples in data set and output is a class membership. An object is classified by the majority vote it gets by its neighbors. The object is assigned to the class most common among its k nearest neighbors.

For KNN, I am using tuning parameter k from 15 to 40 and the default cross validation is performed by taking 25 bootstrap samples comprised of 25% of the observations

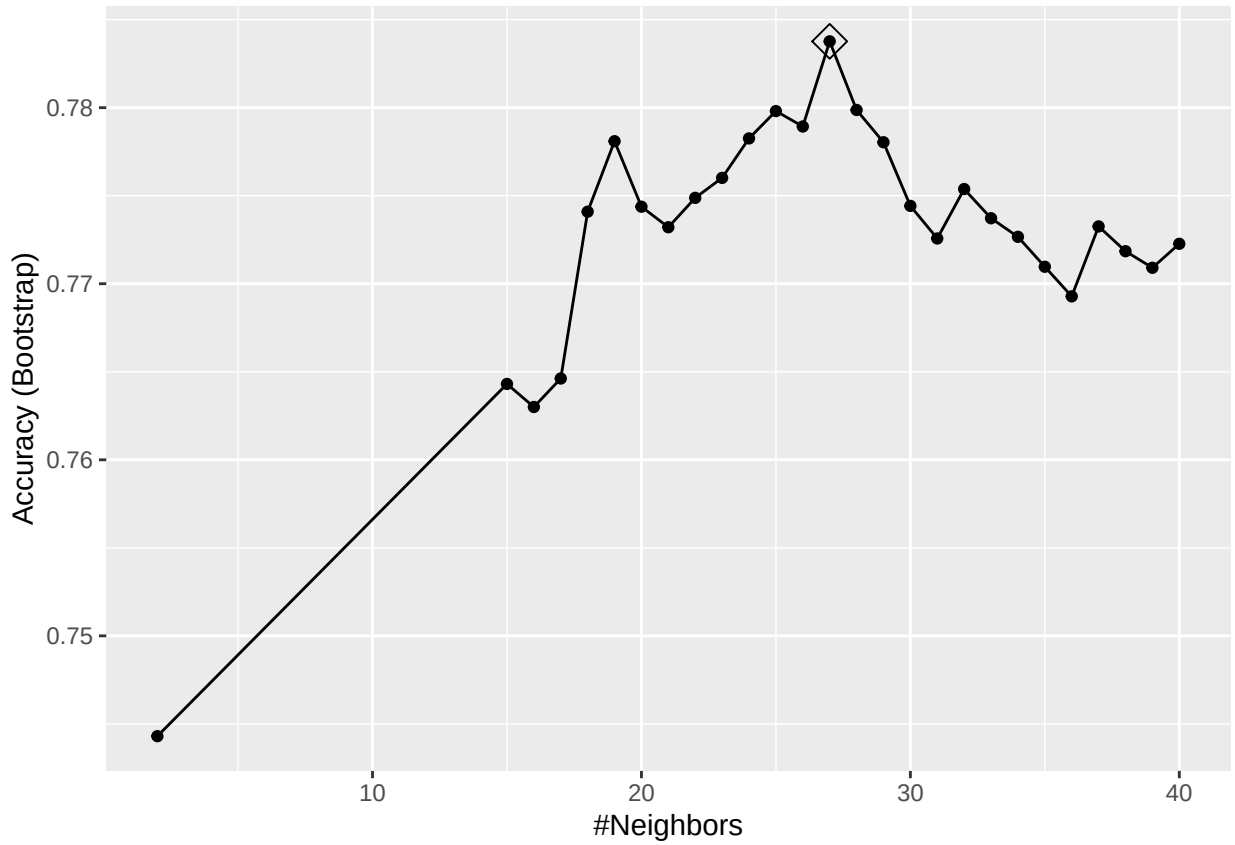
```
train_knn <- train(train_x, train_y, method = "knn",
                   tuneGrid = data.frame(k=c(15:40,2)))
pred_knn <-  predict(train_knn,  test_x)
acc_knn <- confusionMatrix(pred_knn,test_y)$overall['Accuracy']

acc_knn
```

```
##  Accuracy
## 0.8064516
```

```
train_knn$bestTune
```

```
##     k
## 14 27
```

```
ggplot(train_knn, highlight = TRUE)
```



SVM Linear Model

Support vector machine algorithm creates a line or a hyperplane which separates the data into classes. For SVM Linear model, I have used tuning parameter C from 1 to 10 and 10 fold cross validation.

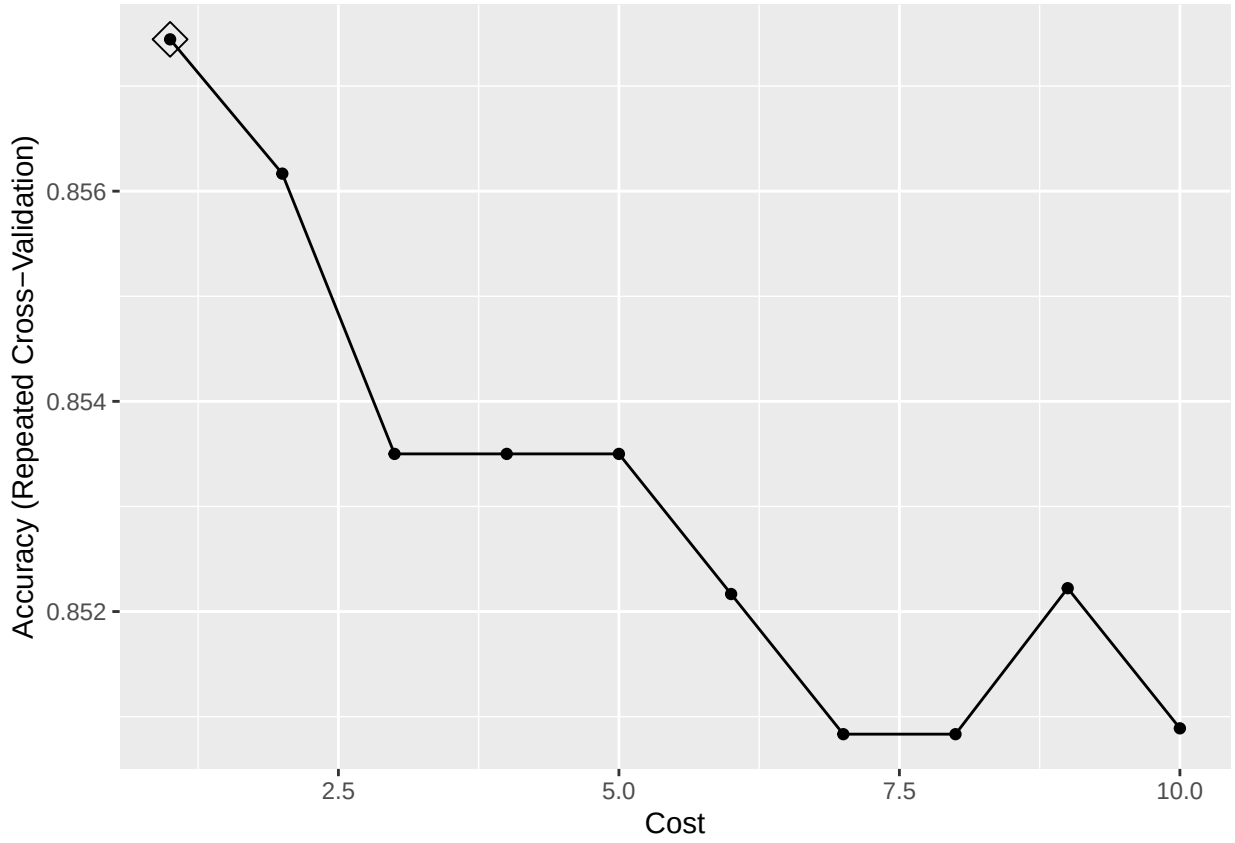
```
train_control <- trainControl(method="repeatedcv", number=10, repeats=3)

train_svm <- train(train_x, train_y, method = "svmLinear",
                   tuneGrid = data.frame(C=c(1:10,2)),
                   trControl = train_control)
pred_svm <-  predict(train_svm,  test_x)
acc_svm <- confusionMatrix(pred_svm,test_y)$overall['Accuracy']

acc_svm
```

```
##  Accuracy
## 0.8709677
```

```
train_svm$bestTune
```

```
##   C
## 1 1
```

```
ggplot(train_svm, highlight = TRUE)
```



Random Forest

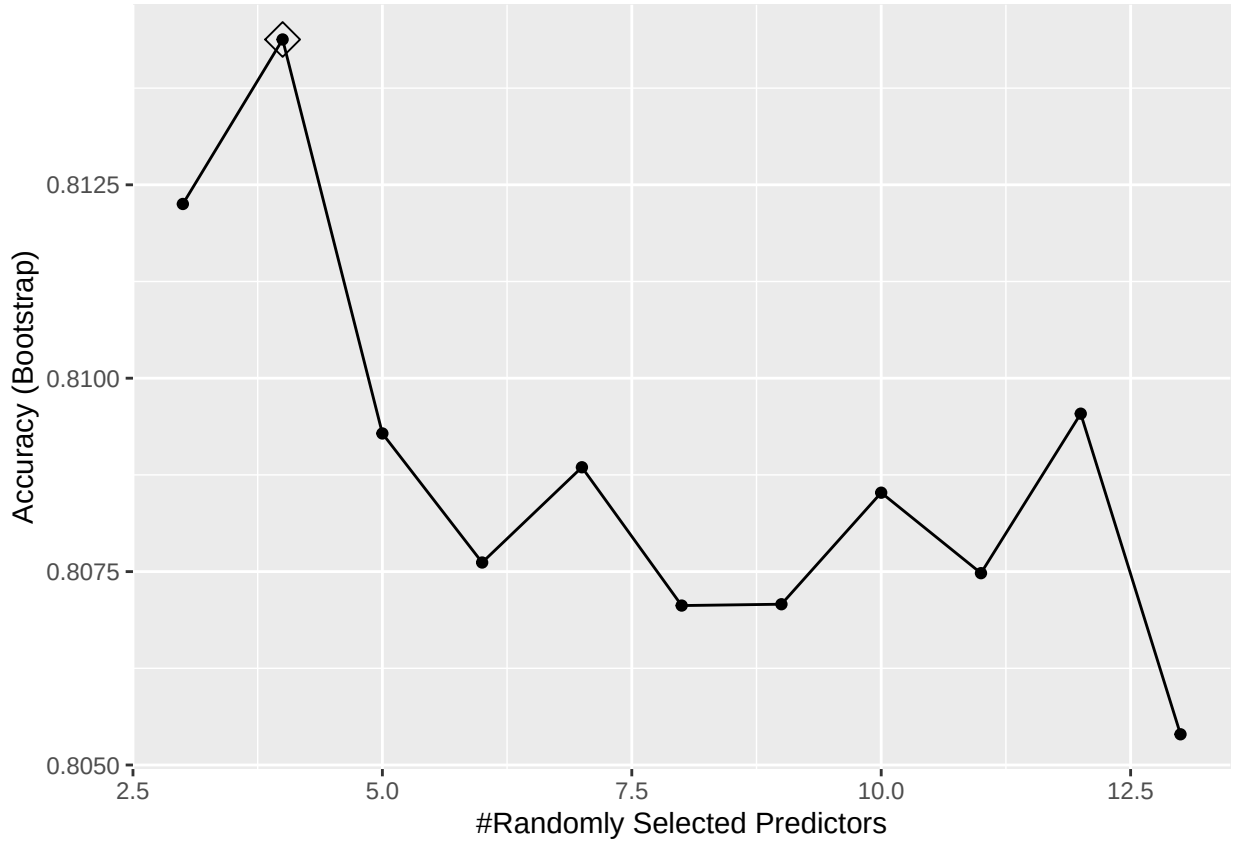
Random forest consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction. Random forest allows each individual tree to randomly sample from the dataset with replacement, resulting in different trees. This process is known as bagging.

For Random forest, tune grid parameter is mtry (number of variables randomly sampled as candidates at each split) with values from 3 to 13.

```
train_rf <- train(train_x, train_y, method = "rf",
                  tuneGrid = data.frame(mtry=c(3:13)), importance=TRUE)
pred_rf <-  predict(train_rf,  test_x)
acc_rf <- confusionMatrix(pred_rf,test_y)$overall['Accuracy']
acc_rf
```

```
##  Accuracy
## 0.8548387
```

```
ggplot(train_rf, highlight = TRUE)
```




```
## rf variable importance
##
##   variables are sorted by maximum importance across the classes
##                           Hernia Normal Spondylolisthesis
```

```
## degree_spondylolisthesis 50.290 67.761           100.000
## pelvic_radius             5.004 33.335            11.844
## sacral_slope             26.756  4.483             6.845
## pelvic_tilt               5.823 21.067             9.119
## pelvic_incidence          9.669 12.366            14.015
## lumbar_lordosis_angle    13.175  0.000             8.231
```

Results

Now we can compare the results of different models and their accuracy.

Below is the accuracy table summary:

```
##          Method  Accuracy
## 1           LDA 0.8064516
## 2           KNN 0.8064516
## 3 Random Forest 0.8548387
## 4           SVM 0.8709677
```

Conclusion

In summary, this analysis shows it is possible to classify the orthopedic patients by analyzing their biochemical features. SVM Linear is the highest performing model with accuracy around 87%. Future work can be done to improve the accuracy above 87%.