CYO - Orthopedic BioMechanical features

Aravind Sankar

11/7/2020

Table Of Contents:

A) Introduction/Overview:

This document provides a detailed narrative of the Machine Learning model built on 'Biomechanical features of orthopedic patients' data-set as part of CYO project requirements in 'Data-Science Professional Certificate' course program offered by edX- HarvardX. The data-set contains 6 biomechanical attributes (predictors) derived from the shape and orientaton of the pelvis and lumbar spine; which, classify patients as belonging to one of the three categories : Normal, Disk Hernia or Spondylolisthesis (dependent variable). The aim of this project is to machine-learn the data and fit a model that can predict the patient classification based on the attributes as accurately as possible. As part of the project requirements, atleast 2 different models would be experimented to offer a variety in model selection.

Please Note : The data-set has been downloaded to my local machine and the R code basically operates on this. For your reference, have attached the data-set (CSV file-name : 'column_3C_weka') to the GitHub repository link https://github.com/AravindAmazon/CYO---Orthopedic-BioMechanical-features

B) Executive Summary:

The following two models have been deployed for this data-set:

a) Multinomial Logistic Regression
b) Random Forest (with one level of iteration to keep the error estimate least)
The accuracy levels and pros & cons of the individual models are detailed out in the sections below.

Please Note : The accuracy results are not documented in the 'Execute Summary' section because I observed that there is a fluctuation in accuracy at different instances of running the code. This is because the sample-size is very small (310 rows) and subsequently, the volume of the test set is as less as 62. So, a variation in the count of correct/incorrect predictions even by 1 or 2 causes a significant variation in overall model accuracy percentage. Also, there is fluctuation in the tuned version of Model 2 (Random Forest) increasing or decreasing the baseline accuracy due to the low sample size. Neverthless, when I ran the model on a set of dummy test data with a relatively bigger sample size, the fluctuation in accuracy was not seen which confirmed that the model logic was correct.

C) Methods/Analysis:

1) Download & Understand the data

```
#Download the required libraries
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(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(reshape2)) install.packages("reshape2", repos = "http://cran.us.r-project.org")
library(tidyverse)
library(caret)
library(data.table)
library(ggplot2)
library(reshape2)

#Open 'Biomechanical features of orthopedic patients' data-set from my local machine.
setwd("/Users/arsankar/Desktop/Knowledge/Data Science/Module 9 - Capstone/CYO")
#Please note that I have already downloaded the data-set from Kaggle to my local directory.
#Hence,the above line of code wouldnt work on another PC as this is specific to my local machine.
#So, for you to be able to run my overall code, request you to download the data-set from
#my GitHub Repository #https://github.com/AravindAmazon/CYO---Orthopedic-BioMechanical-features
#and update your local machine directory path above.



#Read CSV file and store in data-frame
datasetnew <- read.csv("column_3C_weka.csv", sep = "|", header = TRUE)
#Understand the data-set. Firstly, see a high-level view of all the fields
glimpse(datasetnew)
```

```
## Observations: 310
## Variables: 7
## $ pelvic_incidence         <dbl> 63.02782, 39.05695, 68.83202, 69.29701, 49...
## $ pelvic_tilt              <dbl> 22.552586, 10.060991, 22.218482, 24.652878...
## $ lumbar_lordosis_angle    <dbl> 39.60912, 25.01538, 50.09219, 44.31124, 28...
## $ sacral_slope             <dbl> 40.47523, 28.99596, 46.61354, 44.64413, 40...
## $ pelvic_radius            <dbl> 98.67292, 114.40543, 105.98514, 101.86850,...
## $ degree_spondylolisthesis <dbl> -0.2544000, 4.5642586, -3.5303173, 11.2115...
## $ class                    <fct> Hernia, Hernia, Hernia, Hernia, Hernia, He...
```

```
#Understand data-types and min-max values for each field
summary(datasetnew)
```

```
##  pelvic_incidence  pelvic_tilt     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          Hernia           : 60
##  1st Qu.:110.71   1st Qu.:  1.604          Normal           :100
##  Median :118.27   Median : 11.768          Spondylolisthesis:150
##  Mean   :117.92   Mean   : 26.297
##  3rd Qu.:125.47   3rd Qu.: 41.287
##  Max.   :163.07   Max.   :418.543
```

Key Inferences:

1) pelvic_tilt & degree_spondylolisthesis can have negative values
2) Spondylolisthesis (150) is more common than the other 2 classes.

```r
#Count the number of empty cells in the data-set
sum(!complete.cases(datasetnew))
```

```
## [1] 0
```

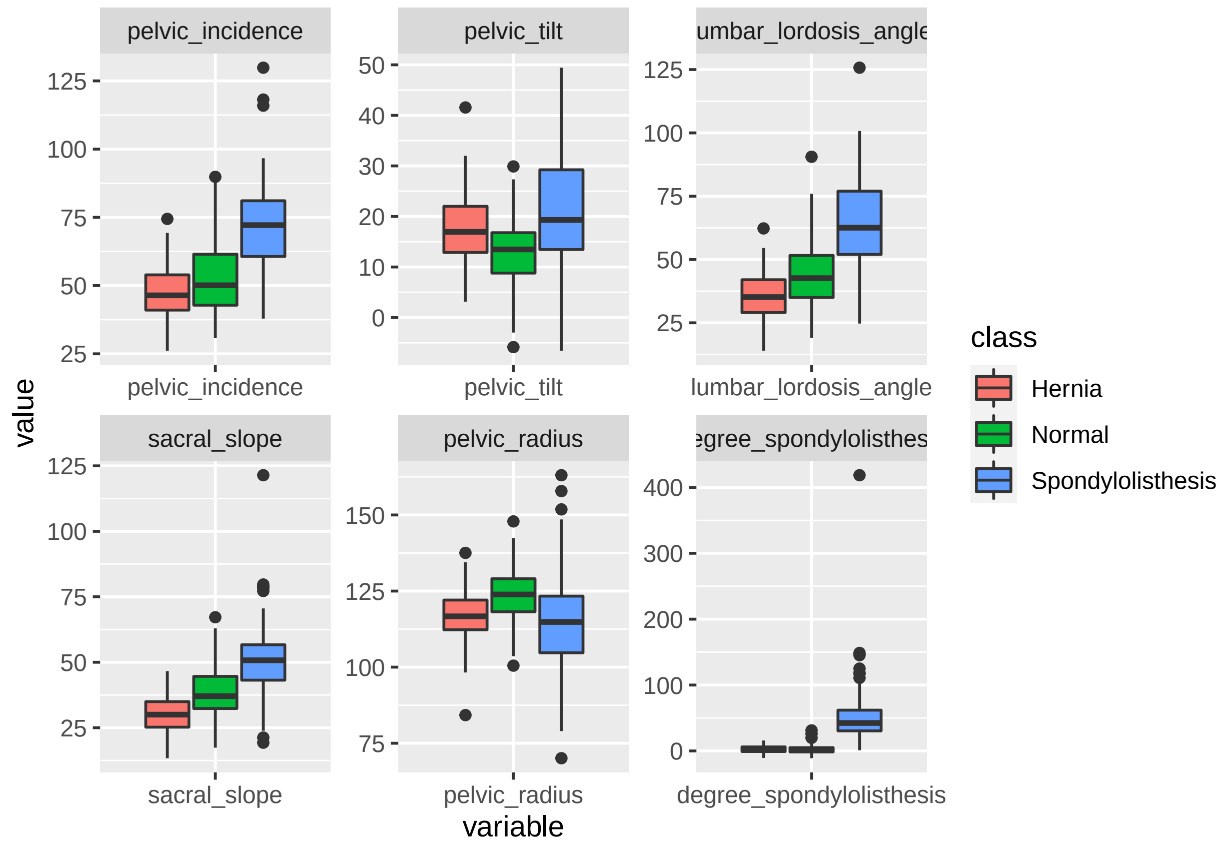
Inference:

The data-set is pretty clean already with no missing cells. There are 6 dbl variables (predictors) and 1 categorical variable (response variable). And, from the summary of these variables, it is seen that that there is no irrelevant/abnormal values in any of the variables.

So, the data-set is ready for analysis.

2) Exploratory Analysis

```r
#See a boxplot view of class vs. each feature
datasetnewbox<-melt(datasetnew, id.var="class")
bp<-ggplot(data=datasetnewbox, aes(x=variable, y=value)) + geom_boxplot(aes(fill=class))
bp+facet_wrap(~variable, scales = "free")
```



Key Inferences

1) Data-distributions for 3 features ('pelvic_incidence', 'lumbar_lordosis_angle' and 'degree_spondylolisthesis') are significantly different for 'Spondylolisthesis' class when compared to that of the other 2 classes.
2) Data-distribution for 'sacral_slope' is significantly different for each of the three classes.
3) Outliers (in all features except pelvic_tilt) are more for 'Spondylolisthesis' class than the other 2 classes.

With these significant differences, a regression-based model approach appears to be a good start.

Since the response variable has 3 levels ('Normal', 'Hernia' and 'Spondylolisthesis), Multinominal Logistic Regression is a good fit for this problem statement.

3) Model 1 - Multinominal Logistic Regression

```
#Deploy Model 1 - Multinomial Logistic Regression
#Partition training and test sets in 80-20 ratio
test_index <- createDataPartition(y = datasetnew$class, times = 1, p = 0.2, list = FALSE)
training_set <- datasetnew[-test_index,]
test_set <- datasetnew[test_index,]

#Build the model on the training set
library(nnet)
my_model <- multinom(class~., data = training_set)
```

```
## # weights:  24 (14 variable)
## initial  value 272.455848
## iter  10 value 138.578401
## iter  20 value 77.643379
## iter  30 value 75.905482
## iter  40 value 75.903326
## iter  50 value 75.898759
## final  value 75.898712
## converged
```

```
#Validate against the test set
my_model_results <- predict(my_model,test_set)

#Plot confusion matrix
confusionMatrix(data = my_model_results, reference = test_set$class)
```

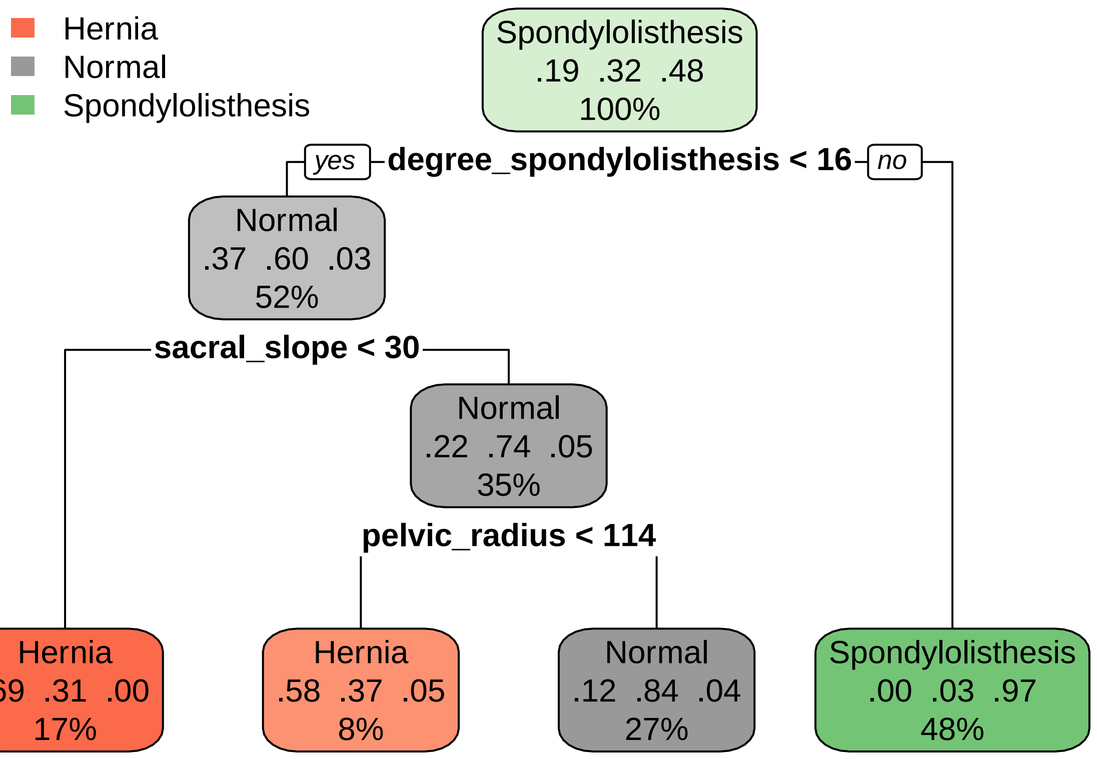
```
## Confusion Matrix and Statistics
##
##                     Reference
## Prediction            Hernia Normal Spondylolisthesis
##   Hernia                   7      0                 2
##   Normal                   5     20                 0
##   Spondylolisthesis        0      0                28
##
## Overall Statistics
##
##                Accuracy : 0.8871
##                  95% CI : (0.7811, 0.9534)
##     No Information Rate : 0.4839
##     P-Value [Acc > NIR] : 2.479e-11
##
##                   Kappa : 0.8189
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Hernia Class: Normal Class: Spondylolisthesis
## Sensitivity                 0.5833        1.0000                   0.9333
## Specificity                 0.9600        0.8810                   1.0000
## Pos Pred Value              0.7778        0.8000                   1.0000
## Neg Pred Value              0.9057        1.0000                   0.9412
## Prevalence                  0.1935        0.3226                   0.4839
## Detection Rate              0.1129        0.3226                   0.4516
## Detection Prevalence        0.1452        0.4032                   0.4516
## Balanced Accuracy           0.7717        0.9405                   0.9667
```

```
#Start tabling accuracy results for each iteration
resulttable1 <- table(observed = test_set$class, predicted = my_model_results)
accuracy1 <- sum(diag(resulttable1))/length(my_model_results)
#Accuracy from Multinomial Logistic Regression model is:
print(accuracy1)
```

```
## [1] 0.8870968
```

```
#Now, let us understand if a tree-structure can be a good alternative model fit for the data
if(!require(rpart)) install.packages("rpart", repos = "http://cran.us.r-project.org")
if(!require(rpart.plot)) install.packages("rpart.plot", repos = "http://cran.us.r-project.org")
library (rpart)
library(rpart.plot)
```

```
ctree_temp <- rpart(class~., data = training_set, method = 'class')
rpart.plot(ctree_temp)
```



```
# With all the predictors being continuous variables within a set range, a tree-structure can not only
#be a good fit but also show a decision-tree based visualization of the model that is
#easily interpretable. Aggregation of many such decision trees can help with a variety of
#options to tune and enhance accuracy. Hence, a RandomForest model can be deployed in this regard.
```

4) **Model 2 - Random Forest**

```
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")
library(randomForest)
```

```
rfmodel <- randomForest(class~., data = training_set, ntree=10)
#See model summary
rfmodel
```

```
##
```

```
## Call:
##  randomForest(formula = class ~ ., data = training_set, ntree = 10)
##                Type of random forest: classification
##                      Number of trees: 10
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 21.9%
## Confusion matrix:
##                   Hernia Normal Spondylolisthesis class.error
## Hernia                25     20                 3  0.47916667
## Normal                19     54                 6  0.31645570
## Spondylolisthesis      2      3               110  0.04347826
```

```
#Validate on the test set
rfp <- predict(rfmodel, test_set)
#See overall accuracy
resulttable2a <- table(observed = test_set$class, predicted = rfp)
resulttable2a
```

```
##                    predicted
## observed             Hernia Normal Spondylolisthesis
##   Hernia                  4      8                 0
##   Normal                  0     19                 1
##   Spondylolisthesis       0      0                30
```

```
accuracy2a <- sum(diag(resulttable2a))/length(rfp)
#Accuracy from the baseline model of RandomForest is:
print(accuracy2a)
```

```
## [1] 0.8548387
```

Inference

The estimate of error rate seen in the baseline model summary can be optimized by fine-tuning. The model is defined for 10 trees (nTree) and the default number of variables to be sampled at each split(mTry). Let us tune mTry for an optimum value where the estimate of error is the least

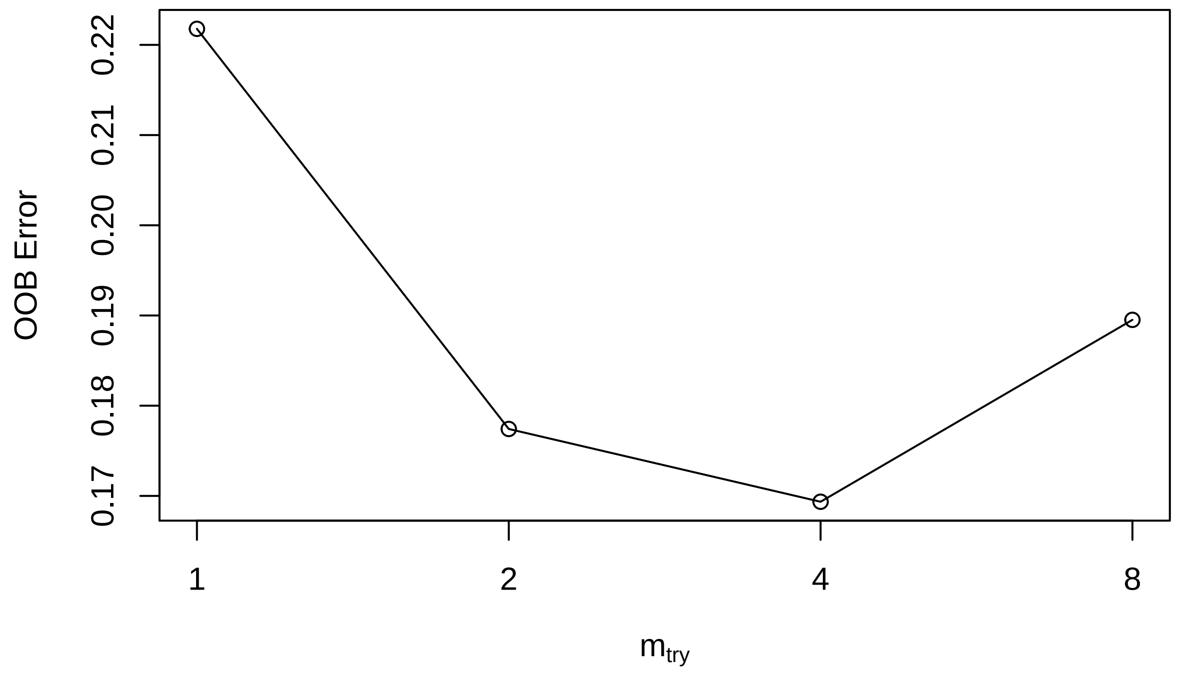
5) Train/tune Model 2 (Random Forest)

```
# Tune for optimum mtry
mtry <- tuneRF(training_set[-7],training_set$class,ntreeTry=100, stepFactor = 0.5, improve = 0.01, trac
```

```
## mtry = 2  OOB error = 17.74%
## Searching left ...
## mtry = 4     OOB error = 16.94%
## 0.04545455 0.01

## Warning in randomForest.default(x, y, mtry = mtryCur, ntree = ntreeTry, :
## invalid mtry: reset to within valid range

## mtry = 8     OOB error = 18.95%
## -0.1190476 0.01
## Searching right ...
## mtry = 1     OOB error = 22.18%
## -0.3095238 0.01
```



```
best.m <- mtry[mtry[,2]==min(mtry[,2]),1]
mtry
```

```
##       mtry  OOBError
## 1.OOB    1 0.2217742
## 2.OOB    2 0.1774194
## 4.OOB    4 0.1693548
## 8.OOB    8 0.1895161
```

```
best.m
```

```
## [1] 4
```

```
# Applying optimum mtry value and also increasing tree-size to 100
rfmodel_tuned <- randomForest(class~., data = training_set, ntree=100, mtry = best.m)
rfp_tuned <- predict(rfmodel_tuned, test_set)
resulttable2b <-table(observed = test_set$class, predicted = rfp_tuned)
resulttable2b
```

```
##                     predicted
## observed             Hernia Normal Spondylolisthesis
##   Hernia                  6      6                 0
##   Normal                  1     19                 0
##   Spondylolisthesis       0      0                30
```

```r
accuracy2b <- sum(diag(resulttable2b))/length(rfp_tuned)
# Accuracy after optimizing mTry for least error estimate is :
print(accuracy2b)
```

```
## [1] 0.8870968
```

D) Results:

Summary of the two models built for 'Biomechanical features of orthopedic patients' is as follows:

```r
Model <- c("Multinomial Logistic Regression","Random Forest","Random Forest(tuned to optimize mtry)")
Accuracy <- c(accuracy1, accuracy2a, accuracy2b)
data.frame(Model, Accuracy)
```

```
##                                   Model  Accuracy
## 1       Multinomial Logistic Regression 0.8870968
## 2                         Random Forest 0.8548387
## 3 Random Forest(tuned to optimize mtry) 0.8870968
```

Models comparison: While Multinomial Logistic Regression (MLR) might produce a better overall accuracy for this problem statement, Random Forest (RF) brings other benefits in the form of visualization (tree-structure) easily interpretable by customers. The randomForestExplainer package offers a variety of metrics to understand (and subsequently iterate) the depth and breadth of the model. Further, there is a lot of tuning parameters (mtry, ntree etc.) in scope to consider for iterating the model to improve accuracy in the long-term.

On the flip side, the chances for over-fitting are more in RF when compared to MLR. Selection of the best model for this problem statement could be based on what is critical to the customer - Timeline, Interpretability, Precision etc.

E) Conclusion

Aiming an 'accuracy closer to 100%' is seen more as a journey than an accomplishment since there could be very many iterations to be followed further (reduce sampling bias errors, optimize tuning parameters etc.). Also, the confusion matrices clearly indicate that the correct & incorrect predictions vary across the models. So, there is a huge opportunity for 'Ensemble' as well in order to optimize the results. Enhanced system features in the form of memory management and deeper code execution are key factors to train the model further and attain better accuracy. The learning experience as part of this project submission was great. The training materials were effective enough to help us pick a problem statement on our own and also independently deploy an end-to-end Machine Learning solution.