Final Assessment - Biomechanical features of Orthopedic Patients

Maria Isabella Caldeira Henrique

2022-11-17

Introduction

This fal assessment consists on studying the Biomechanical features of orthopedic patients.

The analysis was divided of two tasks, being the first one: "Classify patients as belonging to one out of three categories: Normal (100 patients), Disk Hernia (60 patients) or Spondylolisthesis (150 patients)", and the second one: "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)."

The dataset is organized in two different csv files, being the "column3Cweka.csv" used for the first task and the "column2Cweka.csv" used for the second task.

Those tables contain the data of each patient by six biomechanical attributes derived from the shape and orientation of the pelvis and lumbar spine (each one is a column):

- Pelvic Incidence
- Pelvic Tilt
- Lumbar Lordosis Angle
- Sacral Slope
- Pelvic Radius
- Grade of Spondylolisthesis

The goal from this final assessment is to use these biomechanical features to classify patients according to their labels.

The development of both tasks are made by splitting the dataset in a training set and a test set, and using the training set to train the different machine learning algorithms and validate its accuracy with the test set. All those steps are described on further topics from this Final Assessment.

*1st. Task: **Classify patients as belonging to one out of three categories: Normal (100 patients), Disk Hernia (60 patients) or Spondylolisthesis (150 patients)***

1st. Task - Methods/Analysis:

In this study 5 Machine Learning Models are simulated and then compared which was the best one in order to predict correctly the type of disease/normal. The first one to be analyzed is the Classification Tree Model, the second one is the Random Forest, the third one is a Linear Discriminant Analysis (LDA) Model, the forth one is a K-nearest neighbors (KNN) Model and the fifth one is the K-nearest neighbors (KNN) with Cross-Validation.

The analysis can start with the accuracy of randomly guessing the outcome. Using the code below:

```
# guess with equal probability the class
guess <- sample(c("Hernia", "Normal","Spondylolisth"), nrow(test_set),
replace = TRUE)
mean(guess == test_set$class)

## [1] 0.1612903
```

The accuracy of randomly guessing will be equal to **0.1612903.**

That is not a good accuracy, therefore different machine learning models are going to be used in order to improve the accuracy and to make it able to predict the class based on six biomechanical attributes derived from the shape and orientation of the pelvis and lumbar spine.

Classification Tree Model

When trying to fit the Classification Tree Model I analyzed the count of rows on each class from the training dataset (train_set).

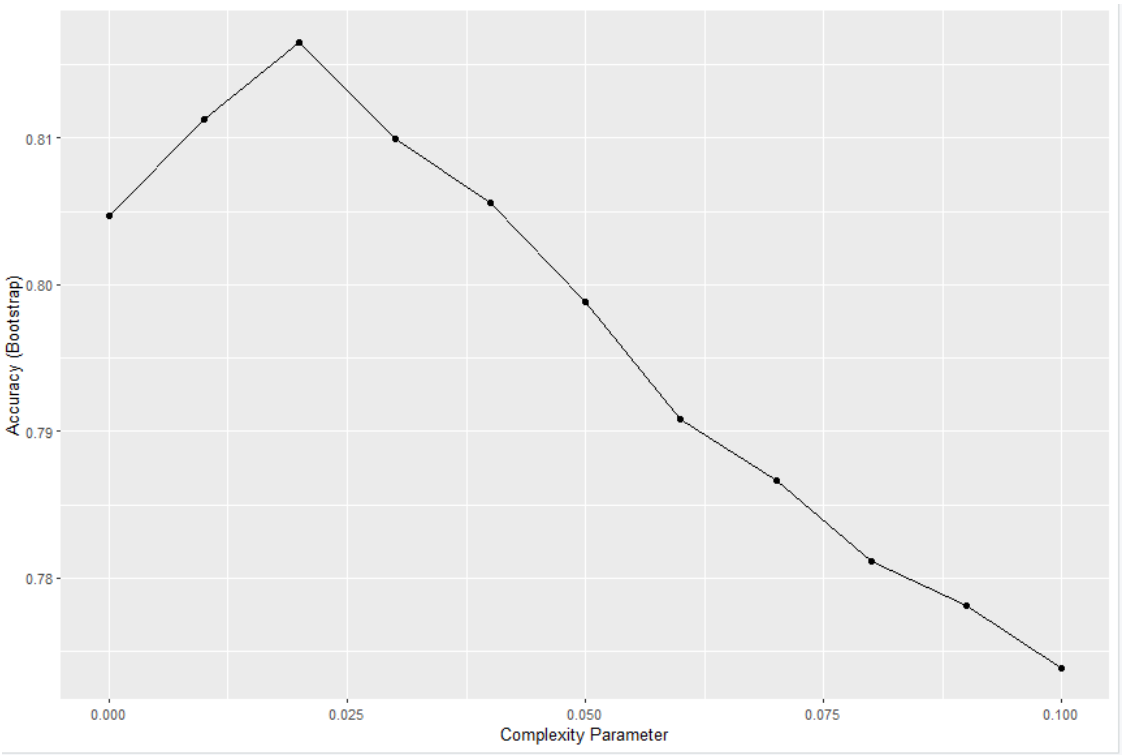
```
train_set  %>%  group_by(class)  %>%  summarize(count = n())
```

Class	Count
Hernia	48
Normal	80
Spondylolisthesis	120

In this table can be seen that the lowest count it has is 48 to Hernia class. By default, when using the rpart function it requires 20 observations before splitting a node. In this case, that wouldn't be a problem on using the default min split, but I simulated with the default number **(Method 1)** and using the argument of min split of 0 in order to split any node **(Method 2)**. Comparing the confusion matrix from both methods it can be evaluated if that increased or not the accuracy and with that, the model with the highest accuracy can be choosed.

Method 1:

cp



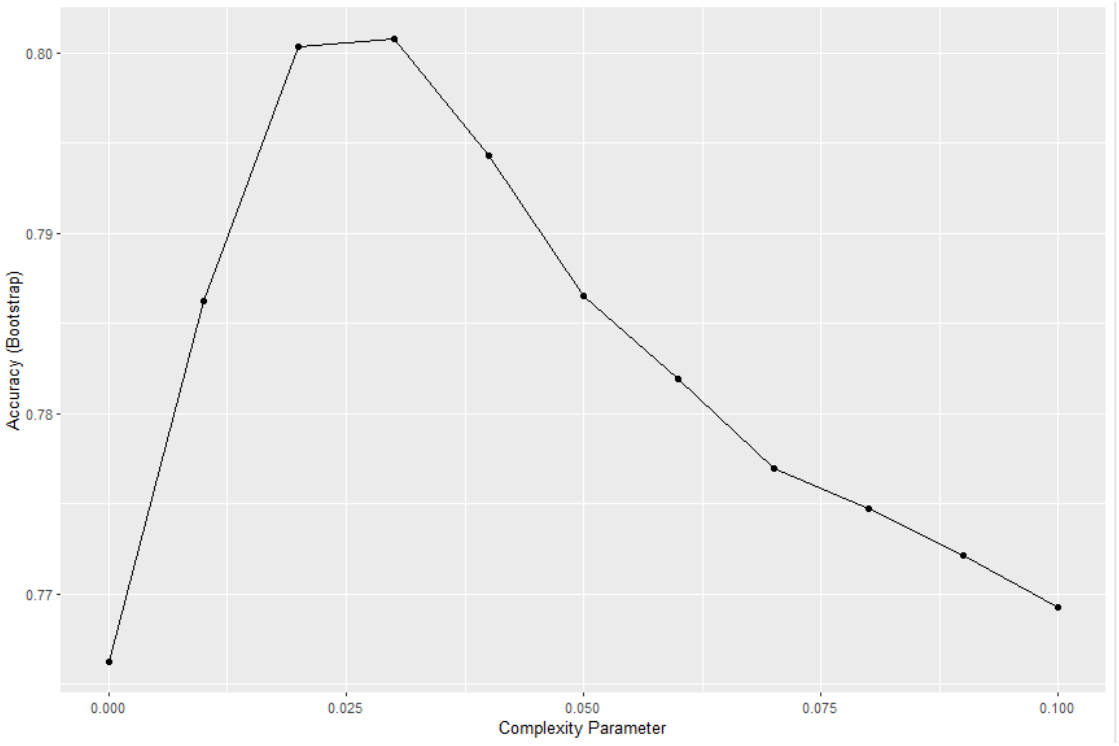
Confusion Matrix

Prediction	Hernia	Normal	Spondylolisthesis
Hernia	12.6	7.6	0.3
Normal	6.3	23.9	2.5
Spondylolisthesis	0.2	1.5	45.3

Model Training Accuracy (Average): **0.8168**

Method 2:

cp



Confusion Matrix

Prediction	Hernia	Normal	Spondylolisthesis
Hernia	11.9	7.4	0.3
Normal	6.9	23.9	2.5
Spondylolisthesis	0.3	1.7	45.3

Model Training Accuracy (Average): **0.8106**

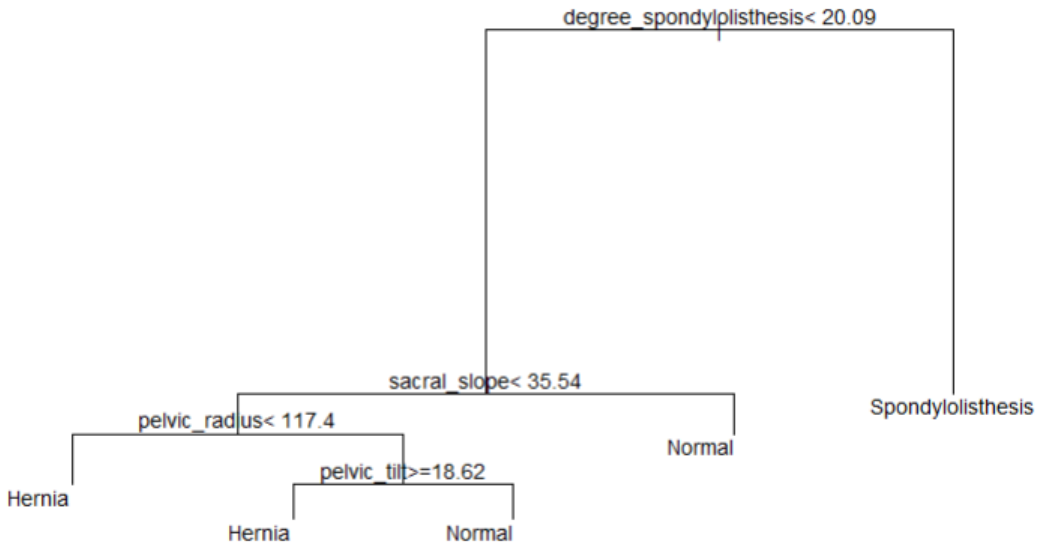
With that **Method 1** was the chosen one and the cp from the best tune was equal to 0.02

```
fit_rpart$bestTune
```

cp = 0.02

The tree from this model can be obtained by the following code and will look like the image below:

```
plot(fit_rpart$finalModel, margin = 0.1)
text(fit_rpart$finalModel)
```



From all those 6 biomechanical attributes we see that just **4** were used in order to build this model.

By testing this prediction in the test_set by using the following code we will get a **final accuracy** from this model of **0.7903226**

```
p_hat_rpart <- predict(fit_rpart, test_set)
y_hat_rpart <- factor(p_hat_rpart)

accuracy1 <- data.frame("res" = confusionMatrix(y_hat_rpart,
factor(test_set$class))$overall[["Accuracy"]])
accuracy1

##         res
## 1 0.7903226
```

Random Forest Model

This model was used in order to see if the disease class could be predicted with even fewer information. For this model it was setup the nodesize = 1 so it will allow small nodesize to grow and the parameter mtry was left to range from 50 – 200 with increments of 25.

With that the train model will give an accuracy of 0.8194 and when predicting the test set the model will have the **final accuracy** of **0.8548387**

The variables of importance from this model can be obtained by this code and it is presented in the table below.

```
# set.seed(1991) # if using R 3.5 or earlier
set.seed(1991, sample.kind = "Rounding") # if using R 3.6 or later
```

```
## Warning in set.seed(1991, sample.kind = "Rounding"): non-uniform
'Rounding'
## sampler used

#Training Model
fit_rf <- with(column_3C_weka,
          train(x_train, y_train, method = "rf",
          nodesize = 1,
          tuneGrid = data.frame(mtry = seq(50, 200, 25))))

#Predicting test_set
p_hat_rf <- predict(fit_rf, test_set)
y_hat_rf <- factor(p_hat_rf)

accuracy_rf <- data.frame("res" = confusionMatrix(y_hat_rf,
factor(test_set$class))$overall[["Accuracy"]])
accuracy_rf

##         res
## 1 0.8548387

varImp(fit_rf)
```

Variable	Overall Importance
degree_spondylolisthesis	100.000
sacral_slope	16.359
pelvic_radius	12.270
pelvic_tilt	4.199
pelvic_incidence	1.235
lumbar_lordosis_angle	0.000

By ranking those parameters it is obtained the following table:

Term	Importance	Rank
degree_spondylolisthesis	100	1
sacral_slope	16.4	2
pelvic_radius	12.3	3
pelvic_tilt	4.20	4

Wich shows that the degree_spondylolisthesis is a very important biomechanical attribute in order to predict if the person land over the normal, Hernia or Spondylolisthesis category.

Just in case and in order to verify if there is any correlation between the two main variables, it can be used the following formula:

```
cor(column_3C_weka$degree_spondylolisthesis, column_3C_weka$sacral_slope)

## [1] 0.5235575
```

Which in this case will result in a value of 0.5235575, representing that those two variables are not correlated to each other.

LDA – Linear Discriminant Analysis Model

When trying to predict the class based in a linear discriminant analysis it is obtained an accuracy from the **final model** of **0.8064516** with the code below:

```
fit_lda <- train(x_train, y_train, method = "lda", data = train_set)

y_hat_lda <- predict(fit_lda, test_set)
accuracy_lda <- mean(y_hat_lda == test_set$class)
```

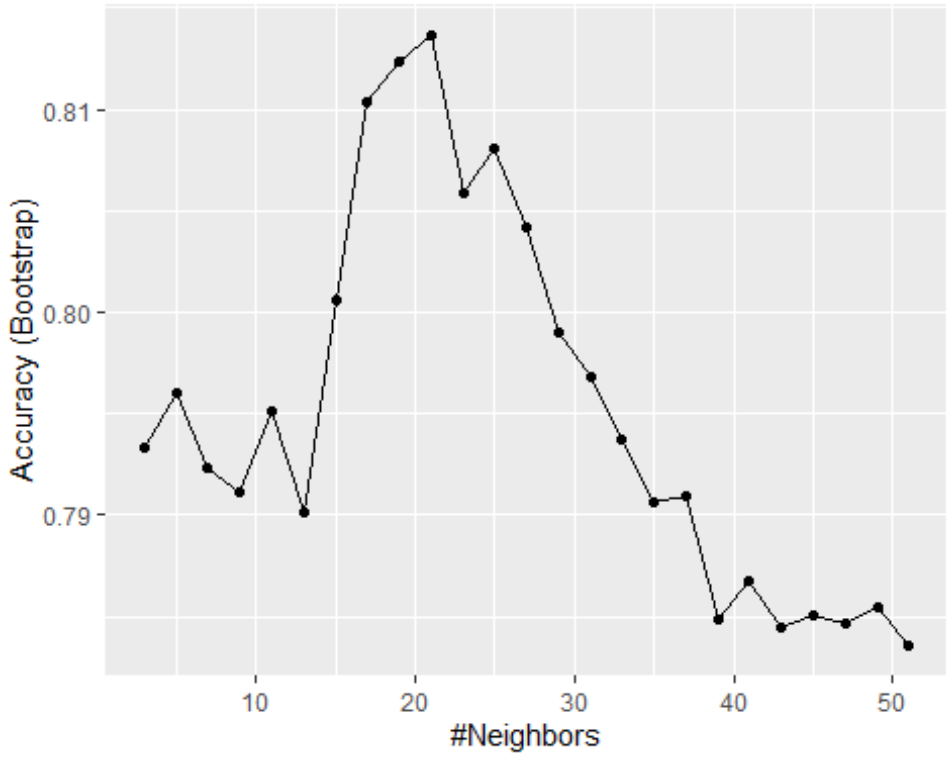
KNN – k-nearest neighbors Model

This model is a supervised machine learning algorithm that can be used to solve both classification and regression problems.

In this case it was left the parameter k (number of neighbors) to tune between 3 and 51 with increments of 2.

The best tune for the k parameter in this analysis was 21. With this model it is possible to predict the test set with an accuracy of **0.8709677**

```
#set.seed(1991) # if using R 3.5 or earlier
set.seed(1991, sample.kind = "Rounding") # if using R 3.6 or later

## Warning in set.seed(1991, sample.kind = "Rounding"): non-uniform
'Rounding'
## sampler used

fit_knn <- train(class ~ .,
                 method = "knn",
                 data = train_set,
                 tuneGrid = data.frame(k = seq(3, 51, 2)))
fit_knn$bestTune

##     k
## 10 21

ggplot(fit_knn)
```




```
knn_preds <- predict(fit_knn, test_set)
accuracy_knn <- mean(knn_preds == test_set$class)
fit_knn$results
```

KNN – k-nearest neighbors Model with Cross-validation

In this algorithm it is performed almost the same simulation as for the KNN Model, but it is used 10-fold cross-validation where each partition consists of 10% of the total. The k parameter (number of neighbors) will be tuned varying from 3 to 51 with increments of 2.

The best tune for the k parameter in this analysis was also 21. And this model showed the same accuracy as the KNN Model being equal to **0.8709677**

```
#set.seed(1991)
set.seed(1991, sample.kind = "Rounding")    # simulate R 3.5

## Warning in set.seed(1991, sample.kind = "Rounding"): non-uniform
'Rounding'
## sampler used

fit_knn_cv <- train(class ~ .,
                    method = "knn",
                    data = train_set,
                    tuneGrid = data.frame(k = seq(3, 51, 2)),
                    trControl = trainControl(method = "cv", number = 10, p
= 0.9))
fit_knn_cv$bestTune
```

```
##     k
## 10 21

knn_cv_preds <- predict(fit_knn_cv, test_set)
accuracy_knn_cv <- mean(knn_cv_preds == test_set$class)
accuracy_knn_cv

## [1] 0.8709677
```

1st. Task - Results:

Compiling all those accuracies from those machine learning models it is noticed that the best model will be the KNN with k parameter as 21, Kappa of 0.6994152, AccuracySD of 0.03409978 and KappaSD of 0.05617487.

The training results from the model are shown below:

```
fit_knn$results
```

k	Accuracy	Kappa	AccuracySD	KappaSD
21	0.8136531	0.6994152	0.03409978	0.05617487

More info from the model can be obtained by using the following code:

```
fit_knn$modelInfo
```

The accuracy from all models on the test dataset are showed below:

Classification Tree Accuracy	Random Forest Accuracy	LDA Accuracy	KNN Accuracy	KNN with Cross-Validation Accuracy
0.7903226	0.8548387	0.8064516	0.8709677	0.8709677

1st. Task - Conclusion:

In conclusion, a series of machine learning algorithms can be used in multi-classification challenges. The best model simulated (KNN) shows an accuracy of 87% on predicting the right class (Normal, Hernia or Spondylolisth).

Since this analysis consists of a multiclass classification model the logistic regression method was not simulated, per by default, logistic regression cannot be used for classification tasks that have more than 2 class labels.

Other analysis that can be done is to re-run the models using just the first 2 important variables (degree_spondylolisthesis and sacral_slope) and compare the accuracy from those models with the previous ones with all variables. If they are similar, then the simple models with the best accuracy should be picked.

2nd. Task: Classify patients as belonging to one out of two categories: Normal (100 patients), Abnormal (210 patients)

2nd. Task - Methods/Analysis:

In this study, 7 Machine Learning Models are simulated and then compared which was the best one in order to predict correctly two classes Normal/Abnormal. The first one to be analyzed is the Classification Tree Model, the second one is the Random Forest, the third one is a Linear Discriminant Analysis (LDA) Model, the forth one is a Logistic Regression Model, the fifth one is a K-nearest neighbors (KNN) Model and the final model is the K-nearest neighbors (KNN) with Cross-Validation.

The analysis can be started with the accuracy of randomly guessing the outcome. Using the code below:

```
# guess with equal probability the classes
guess <- sample(c("Abnormal", "Normal"), nrow(test_set2), replace = TRUE)
mean(guess == test_set2$class)

## [1] 0.516129
```

The accuracy of randomly guessing will be equal to **0.516129**

That is not a really good accuracy, therefore different machine learning models are going to be used in order to improve the accuracy and to make it able to predict the class based on six biomechanical attributes derived from the shape and orientation of the pelvis and lumbar spine.

Classification Tree Model

When trying to fit the Classification Tree Model I analyzed the count of rows on each class from the training dataset (train_set2).

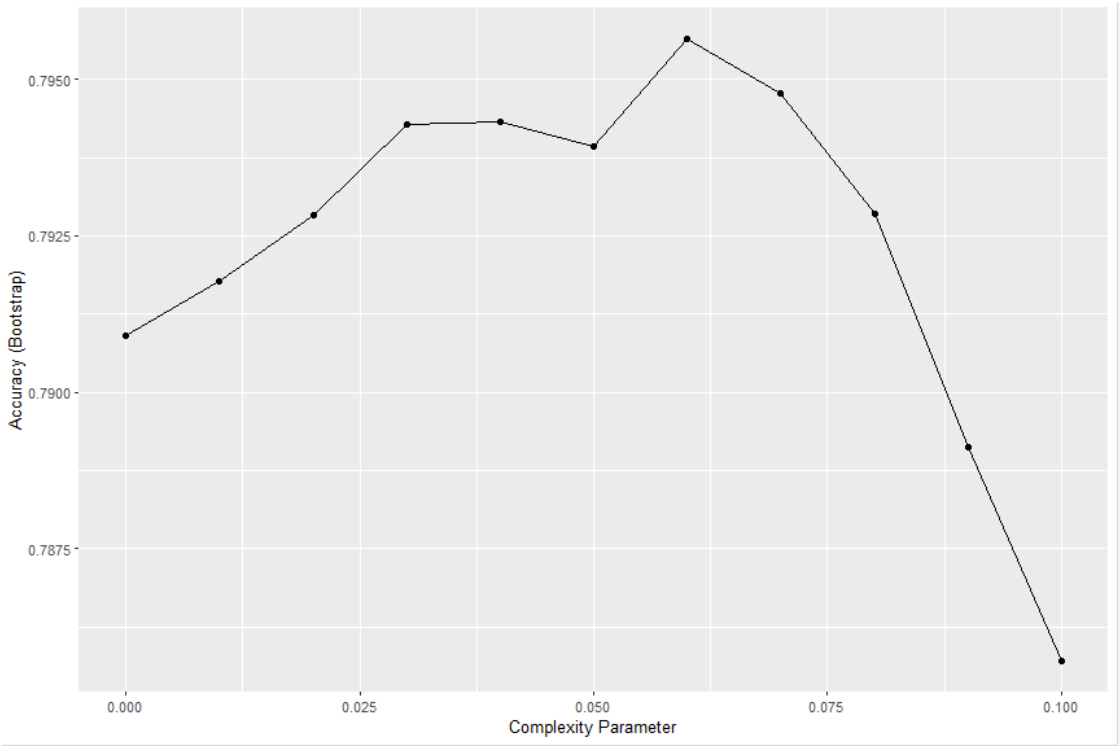
```
train_set2 %>%  group_by(class)  %>%  summarize(count = n())
```

Class	Count
Abnormal	168
Normal	80

In this table can be seen that the lowest count is 80 for Normal class. By default, when using the rpart function it requires 20 observations before splitting a node. In this case, that wouldn't be a problem on using the default min split, but I simulated with the default number (Method 1) and using the argument of min split of 0 in order to split any node (Method 2). Comparing the confusion matrix from both methods it can evaluated if that increased or not the accuracy and with that, the model with the highest accuracy can be kept.

Method 1:

cp



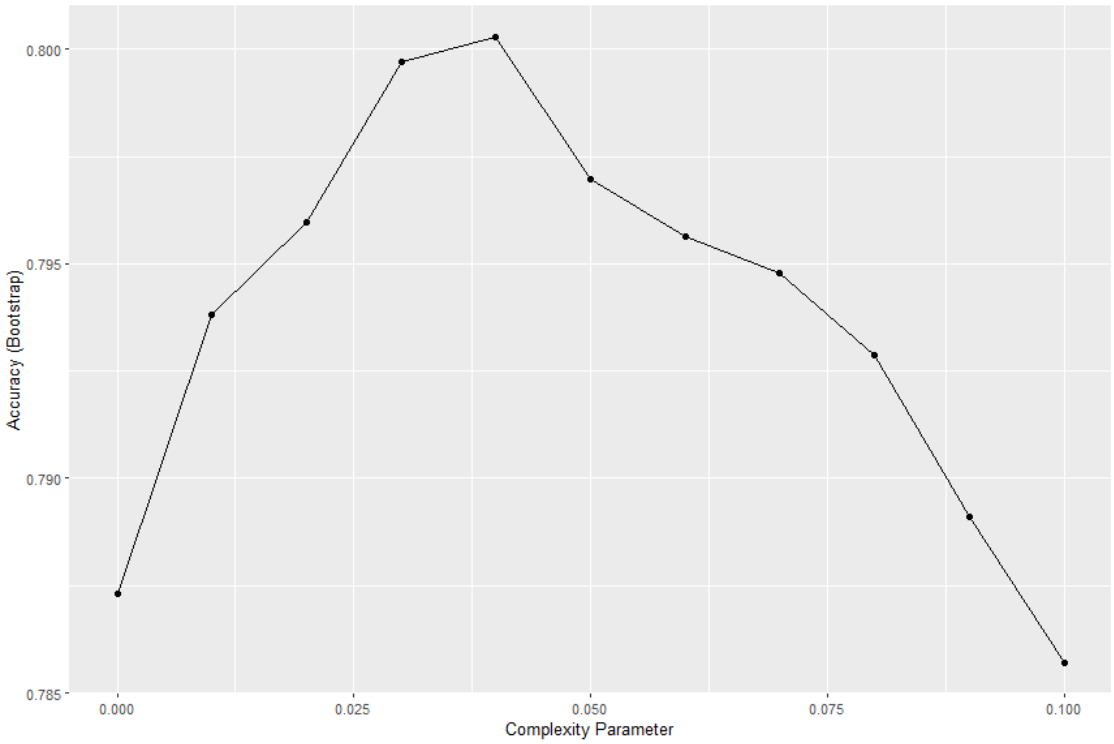
Confusion Matrix

Prediction	Abnormal	Normal
Abnormal	57.1	10.5
Normal	10	22.5

Model Training Accuracy (Average): **0.7957**

Method 2:

cp

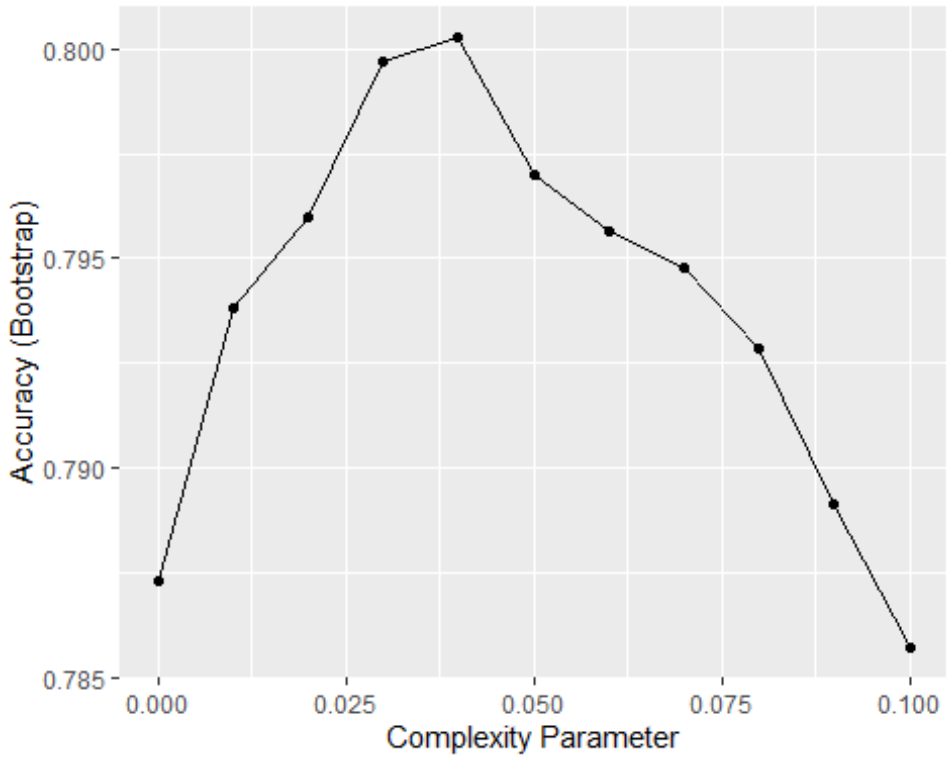


Confusion Matrix

Prediction	Abnormal	Normal
Abnormal	57.6	10.5
Normal	9.4	22.4

Model Training Accuracy (Average): **0.8005**

With that **Method 2** was the chosen one and the cp from the best tune was equal to 0.04

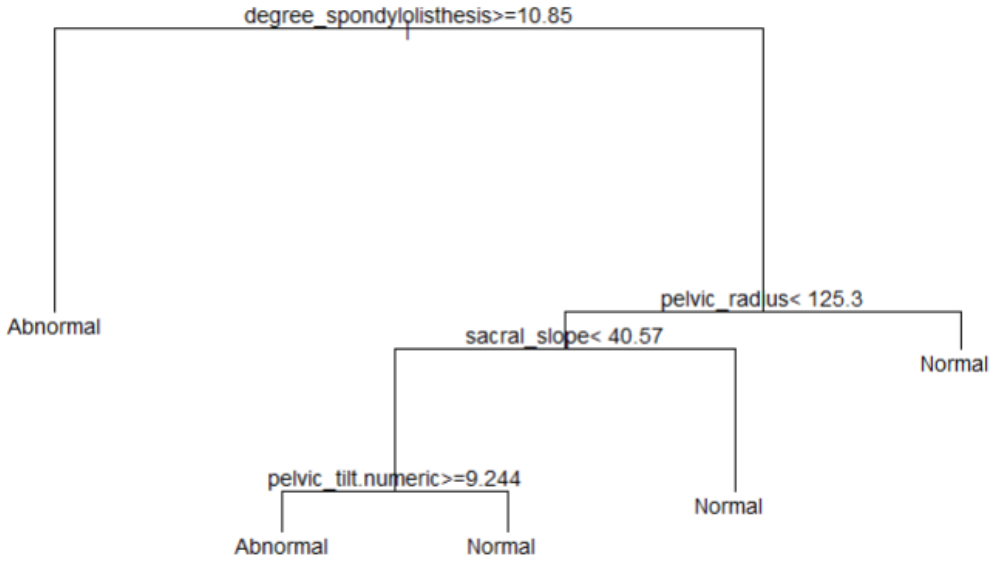



```
fit_rpart2_2$bestTune
```

```
##     cp
##   0.04
```

The tree from this model can be obtained by the following code and will look like the image below:

```
plot(fit_rpart2_2$finalModel, margin = 0.1)
text(fit_rpart2_2$finalModel)
```



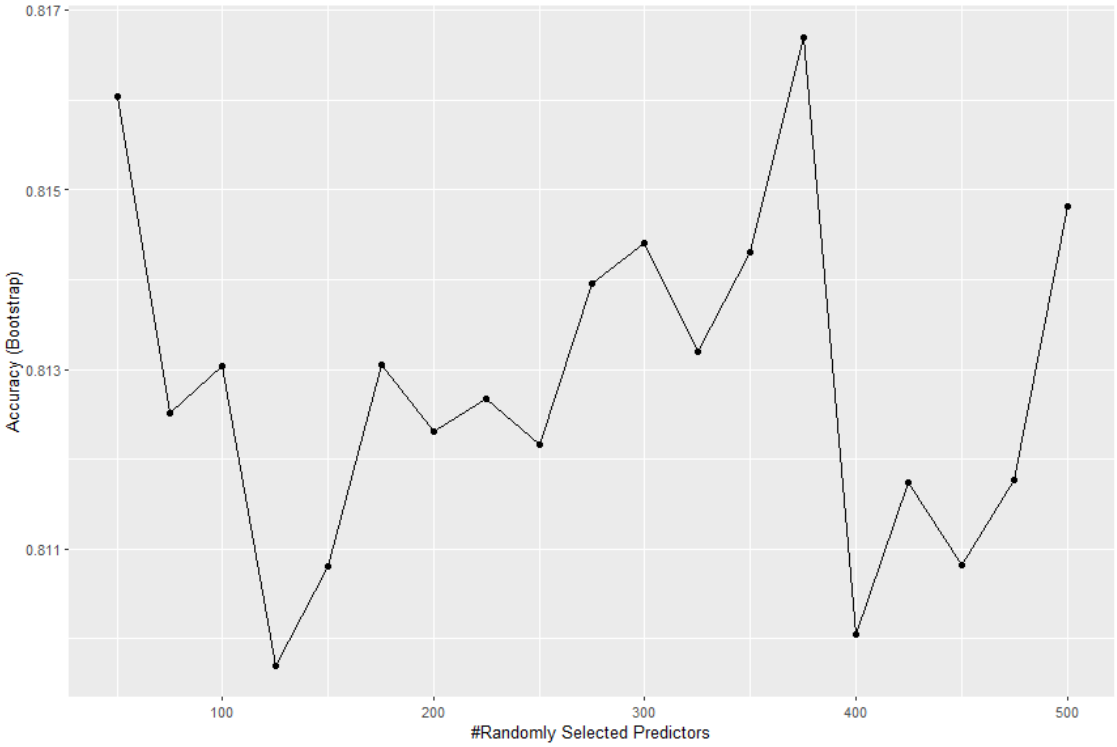
From all those 6 biomechanical attributes just **4** were used in order to build this model.

By testing this prediction in the test_set2 by using the following code it will be obtained a **final accuracy** from this model of **0.8870968**

Random Forest Model

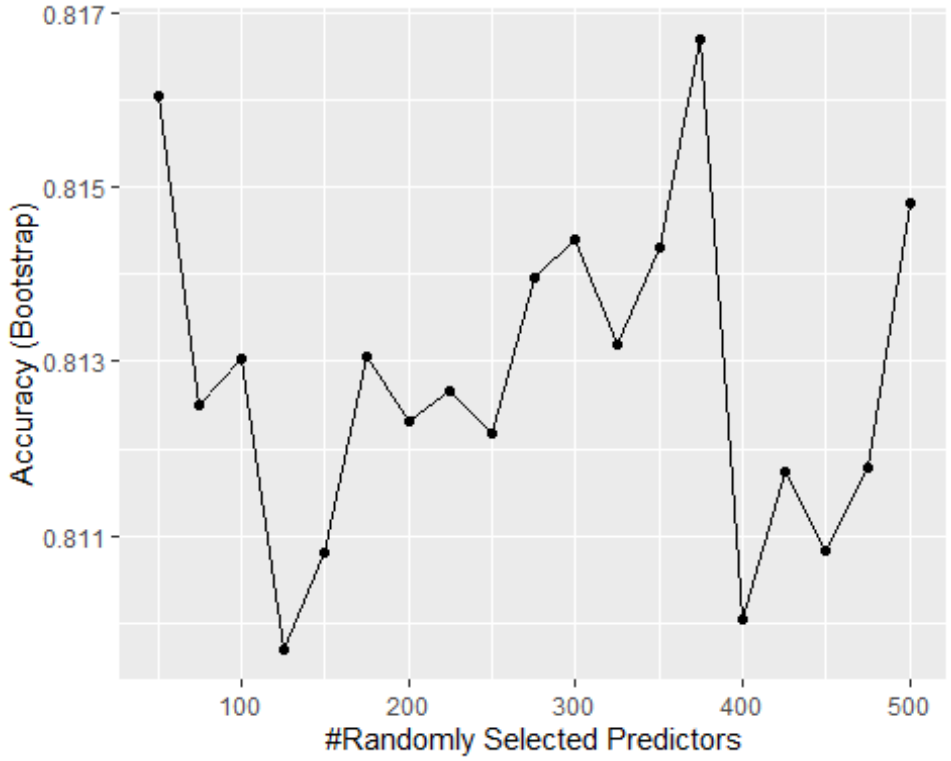
This model was used in order to see if the Normal/Abnormal class could be predicted with even fewer information. For this model it was setup the nodesize = 1 so it will allow small nodesize to grow and the parameter mtry was left to range from 50 – 500 with increments of 25.

The best tune for the mtry parameter is equal to 375.



With that the train model will give an accuracy of 0.8168 and when predicting the test set the model will have the **final accuracy** of **0.8709677**

The variables of importance from this model can be obtained by this code and it is presented in the table below.




```
fit_rf2$bestTune

##    mtry
## 14  375

varImp(fit_rf2)
```

Variable	Overall Importance
degree_spondylolisthesis	100.000
pelvis_radius	22.2980
sacral_slope	16.8140
pelvic_tilt	5.5201
lumbar_lordosis_angle	0.2004
pelvic_incidence	0.000

By ranking those parameters the following table is obtained:

Variable	Importance	Rank
degree_spondylolisthesis	100	1
pelvic_radius	22.3	2
sacral_slope	16.8	3
pelvic_tilt.numeric	5.52	4

Wich shows that the degree_spondylolisthesis is a very important biomechanical attribute in order to predict if the person land over the Normal/Abnormal category.

Just in case and in order to verify if there is any correlation between the three main variables, it can be used the following formula:

```
cor(column_2C_weka$degree_spondylolisthesis, column_2C_weka$pelvic_radius)

## [1] -0.02606501

cor(column_2C_weka$degree_spondylolisthesis, column_2C_weka$sacral_slope)

## [1] 0.5235575

cor(column_2C_weka$pelvic_radius, column_2C_weka$sacral_slope)

## [1] -0.3421283
```

Which in this case will result on the following values respectively, -0.02606501, 0.5235575, -0.3421283, representing that those two variables are not correlated to each other.

LDA – Linear Discriminant Analysis Model

When trying to predict the class based in a linear discriminant analysis it is obtained an accuracy from the **final model** of **0.8548387** with the code below:

```
#set.seed(1991) # if using R 3.5 or earlier
set.seed(1991, sample.kind = "Rounding") # if using R 3.6 or later

fit_lda2 <- train(x_train2, y_train2, method = "lda", data = train_set2)

y_hat_lda2 <- predict(fit_lda2, test_set2)
accuracy_lda2 <- mean(y_hat_lda2 == test_set2$class)
```

Linear Regression Model

When trying to predict the class based in a linear regression model is it is obtained an accuracy from the **final model** of **0.9032258** with the code below:

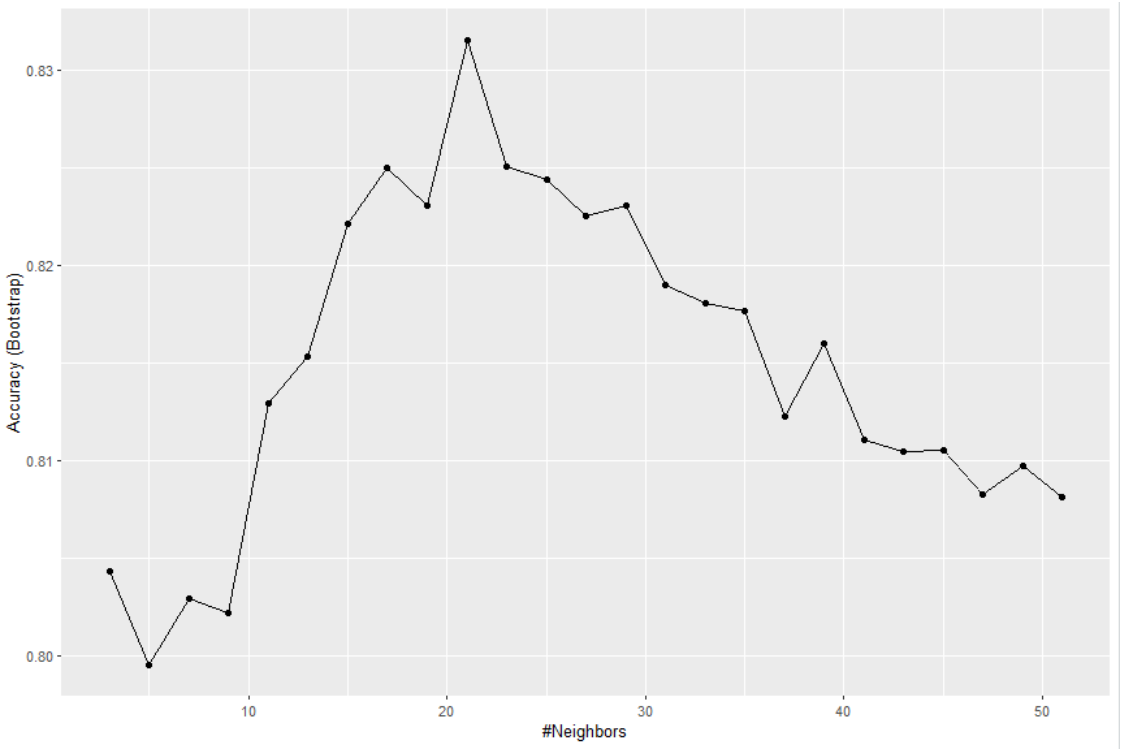
```
# set.seed(1991) # if using R 3.5 or earlier
set.seed(1991, sample.kind = "Rounding") # if using R 3.6 or later

fit_glm2 <- train(class ~ ., method = "glm", data = train_set2)

glm_preds2 <- predict(fit_glm2, test_set2)
accuracy_glm <- mean(glm_preds2 == test_set2$class)
```

KNN – k-nearest neighbors Model

This model is a supervised machine learning algorithm that can be used to solve both classification and regression problems.

In this case it was left the parameter k (number of neighbors) to tune between 3 and 51 with increments of 2.

The best tune for the k parameter in this analysis was 21. With this model it is possible to predict the test set with an accuracy of **0.9032258**



KNN – k-nearest neighbors Model with Cross-validation

In this algorithm it is performed almost the same simulation as for the KNN Model, but it is used 10-fold cross-validation where each partition consists of 10% of the total. The k parameter (number of neighbors) will be tuned varying from 3 to 51 with increments of 2.

The best tune for the k parameter in this analysis was 25. And this model showed the same accuracy as the KNN Model being equal to **0.8548387**

2nd. Task - RESULTS:

Compiling all those accuracies from those machine learning models it can be seen that the best models are the Linear Regression and the KNN and that they improved a lot from the result provided by the random guessing.

The Linear Regression model has the following parameters:

```
fit_glm2$results
```

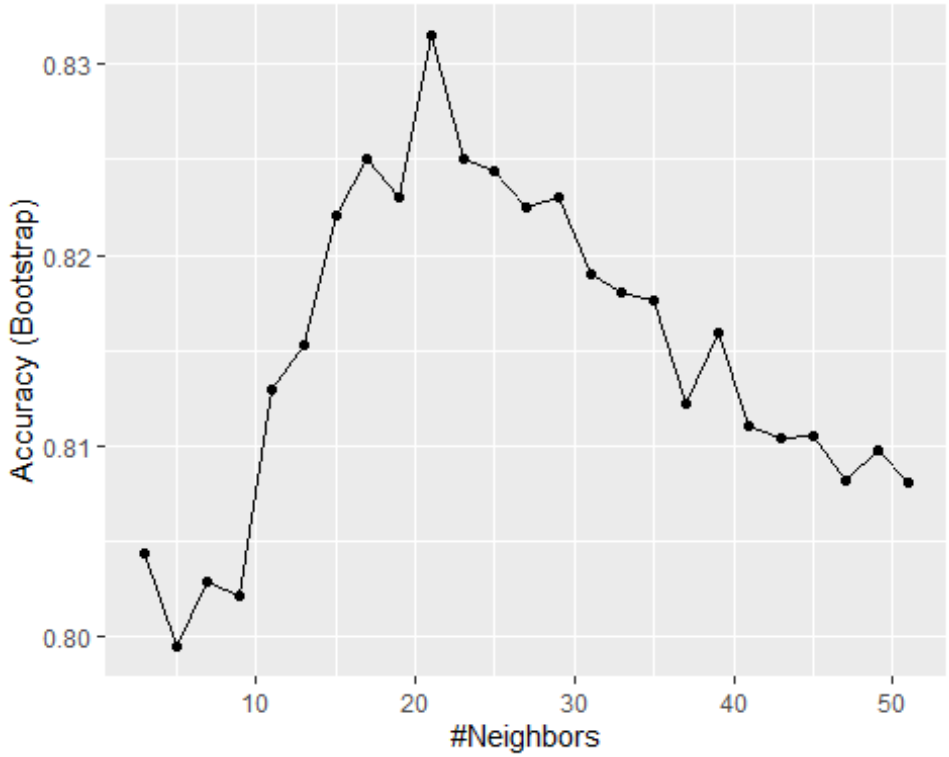
Parameter	Accuracy	Kappa	AccuracySD	KappaSD
none	0.8419787	0.6412655	0.03759098	0.09031056

More info from the Linear Regression model can be obtained by using the following code:

```
fit_glm2$modelInfo
```

The KNN model has the following parameters:

```
#set.seed(1991) # if using R 3.5 or earlier
set.seed(1991, sample.kind = "Rounding") # if using R 3.6 or later

fit_knn2 <- train(class ~ .,
                  method = "knn",
                  data = train_set2,
                  tuneGrid = data.frame(k = seq(3, 51, 2)))
fit_knn2$bestTune

ggplot(fit_knn2)
```




```
knn_preds2 <- predict(fit_knn2, test_set2)
accuracy_knn2 <- mean(knn_preds2 == test_set2$class)
fit_knn2$results
```

k	Accuracy	Kappa	AccuracySD	KappaSD
21	0.8315074	0.6280538	0.03996250	0.08820394

More info from the KNN model can be obtained by using the following code:

```
fit_knn2$modelInfo
```

The accuracy from all models on the test dataset are showed below:

Classification Tree Model Accuracy	Random Forest Accuracy	LDA Accuracy	Linear Regression Accuracy	KNN Accuracy	KNN with Cross-Validation Accuracy
0.8870968	0.8709677	0.8548387	0.9032258	0.9032258	0.8548387

2nd. Task - Conclusion:

In conclusion, a series of machine learning algorithms can be used in classification challenges. The best models simulated show an accuracy of 90% on predicting the right class (Normal or Abnormal).

Since this analysis consists of a 2 class model the logistic regression method was simulated, per by default, logistic regression can be used for classification tasks that have 2 class labels.

Other analysis that can be done is to re-run the models using just the first 3 important variables (degree_spondylolisthesis, pelvic_radius and sacral_slope) and compare the accuracy from those models with the previous ones with all variables. If they are similar, then the simple models with the best accuracy should be picked.