Choose Your Own - Capstone

Yap Kim Thow

8/Jan/2021

1 INTRODUCTION

In this project, a data set of biomechanical features of orthopedic patients is used to create a classification model to predict patients as belonging to one of three categories: *Normal* (100 patients), *Disk Hernia* (60 patients) or *Spondylolisthesis* (150 patients). The latter two classes are considered *abnormal* classes.

Each patient(data point) in the data set has a total of seven features of *numeric* type (*pelvic incidence*, *pelvic tilt*, *lumbar lordosis angle*, *sacral slope*, *pelvic radius* and *grade of spondylolisthesis*).

This report first outlines the steps taken for data exploration to understand the data structure and gain insights about how we might use the data. Next, the data set is randomly sampled into an appropriate train-test split before model training.

The **Methods/Analysis** section contains brief descriptions of the three models considered in this report: *Classification tree*, *random forest*, and *support vector machine*, and the model training steps by using 10-fold cross-validations. The accuracies obtained from the cross-validations for the three models are compared, before testing their predictions against the test set. The *support vector machine* model is the best performing model based on its cross-validated result, and the accuracy of its predictions against the test set is presented.

The final section concludes the findings in this project, before describing some of the limitations and future work.

```
#load required packages
if(!require(tidyverse)) install.packages("tidyverse"); library(tidyverse)
if(!require(caret)) install.packages("caret"); library(caret)
if(!require(PerformanceAnalytics)) install.packages("PerformanceAnalytics");
    library(PerformanceAnalytics) #for correlation plot
if(!require(e1071)) install.packages("e1071"); library(e1071)
if(!require(rattle)) install.packages("rattle"); library(rattle) #for rpart plot
#set number of significant digits
options(digits = 4)
```

1.1 Data Exploration

We will first load the data set before conducting some basic data exploration steps, as well as checking for potential *NA*'s in the data set.

```
#read data from URL of my Github
dat <- read_csv("https://github.com/ktyap/HarvardX_ChooseYourOwn_Capstone/raw/master/column_3C_weka.csv
#basic data structure examination
head(dat)
```

```
## # A tibble: 6 x 7
##   pelvic_incidence pelvic_tilt lumbar_lordosis~ sacral_slope pelvic_radius
##              <dbl>       <dbl>            <dbl>        <dbl>         <dbl>
## 1             63.0       22.6              39.6         40.5          98.7
## 2             39.1       10.1              25.0         29.0         114.
## 3             68.8       22.2              50.1         46.6         106.
## 4             69.3       24.7              44.3         44.6         102.
## 5             49.7        9.65             28.3         40.1         108.
## 6             40.3       13.9              25.1         26.3         130.
## # ... with 2 more variables: degree_spondylolisthesis <dbl>, class <chr>
```

```
str(dat)
```

```
## tibble [310 x 7] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ pelvic_incidence        : num [1:310] 63 39.1 68.8 69.3 49.7 ...
##  $ pelvic_tilt             : num [1:310] 22.55 10.06 22.22 24.65 9.65 ...
##  $ lumbar_lordosis_angle   : num [1:310] 39.6 25 50.1 44.3 28.3 ...
##  $ sacral_slope            : num [1:310] 40.5 29 46.6 44.6 40.1 ...
##  $ pelvic_radius           : num [1:310] 98.7 114.4 106 101.9 108.2 ...
##  $ degree_spondylolisthesis: num [1:310] -0.254 4.564 -3.53 11.212 7.919 ...
##  $ class                   : chr [1:310] "Hernia" "Hernia" "Hernia" "Hernia" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   pelvic_incidence = col_double(),
##   ..   pelvic_tilt = col_double(),
##   ..   lumbar_lordosis_angle = col_double(),
##   ..   sacral_slope = col_double(),
##   ..   pelvic_radius = col_double(),
##   ..   degree_spondylolisthesis = col_double(),
##   ..   class = col_character()
##   .. )
```

```
#check for NAs
isNA <- apply(dat, 2, function(x){
            any(is.na(x))
})
isNA
```

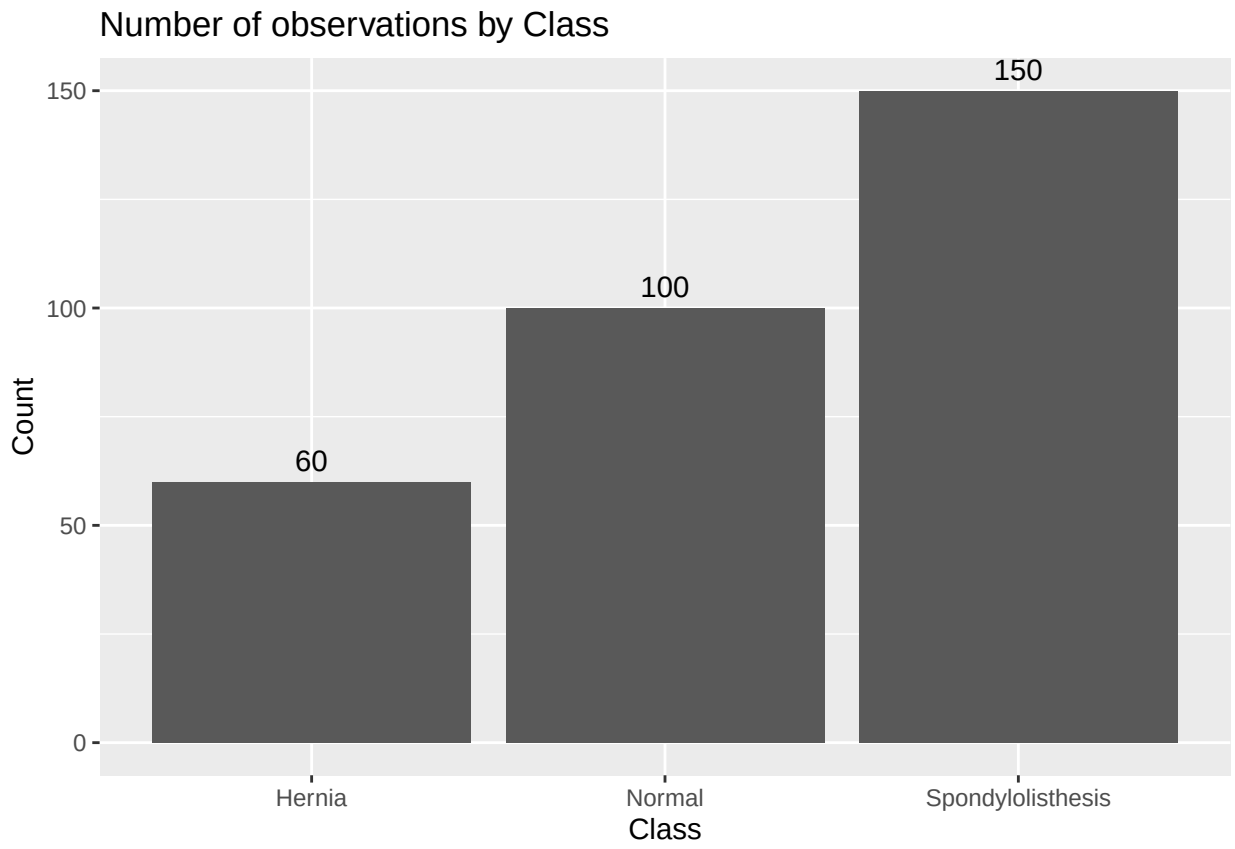
```
##         pelvic_incidence              pelvic_tilt    lumbar_lordosis_angle
##                    FALSE                    FALSE                    FALSE
##             sacral_slope           pelvic_radius degree_spondylolisthesis
##                    FALSE                    FALSE                    FALSE
##                    class
##                    FALSE
```

```
#number of distinct labels in "class"
levels(factor(dat$class))
```

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

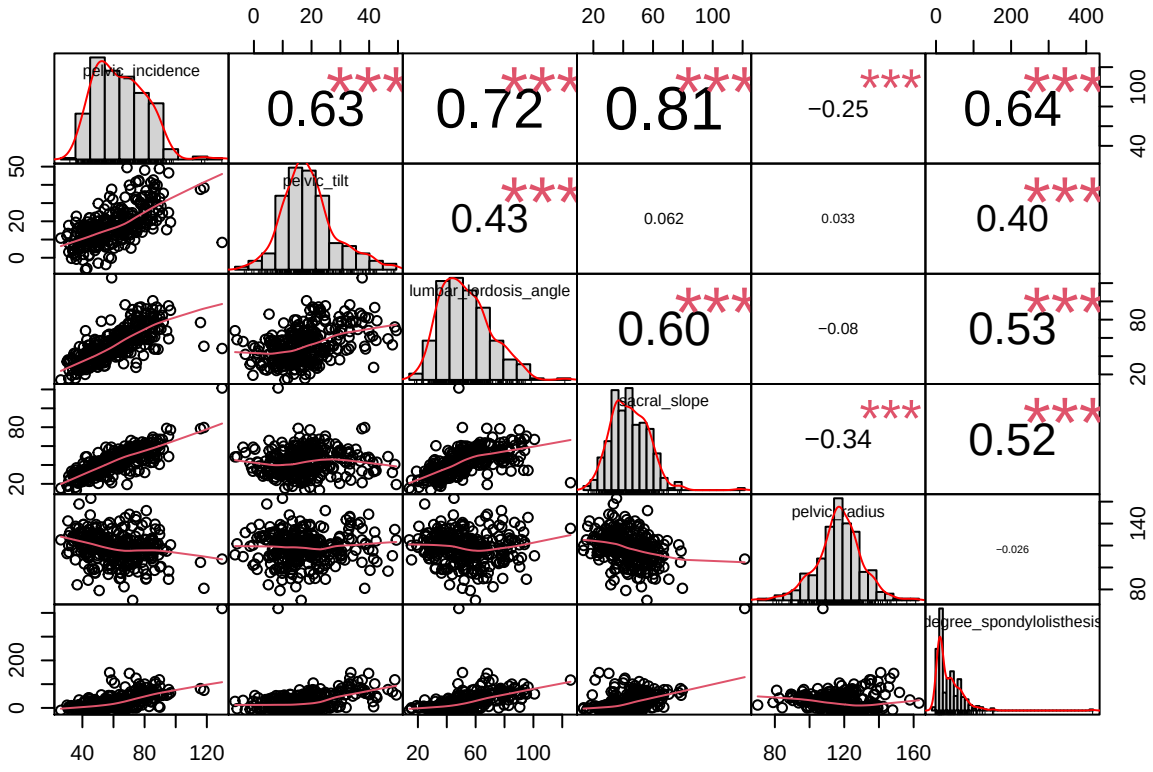
The data set has a total of seven variables and 310 observations, with a response or outcome *class* of three labels. We will change the *class* variable to categorical before plotting the number of observations that fall into these classes.

```r
#mutate class variable to factor
dat <- dat %>% mutate(class = factor(class))
#count of observations in the 3 classes
dat %>% ggplot(aes(x = class)) + geom_bar() +
            geom_text(stat = "count", aes(label = ..count..), vjust = -0.5) +
            ggtitle("Number of observations by Class") +
            xlab("Class") + ylab("Count")
```



The above shows that the number of observations in the classes are different, but the imbalance may not be severe enough to cause major issues in the classification task in this project. So no action is taken to correct for data imbalance here.

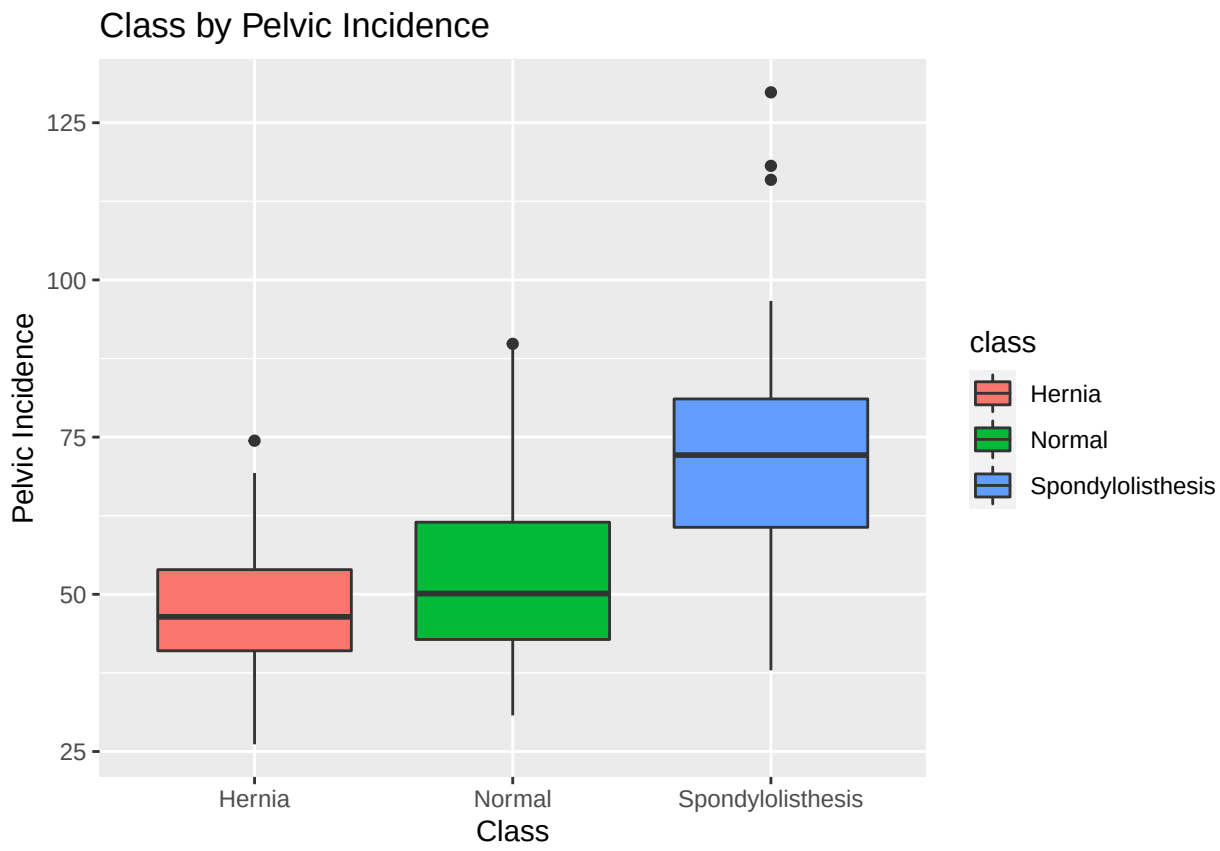
```r
#correlation plot of variables
chart.Correlation(dat[,-7])
```



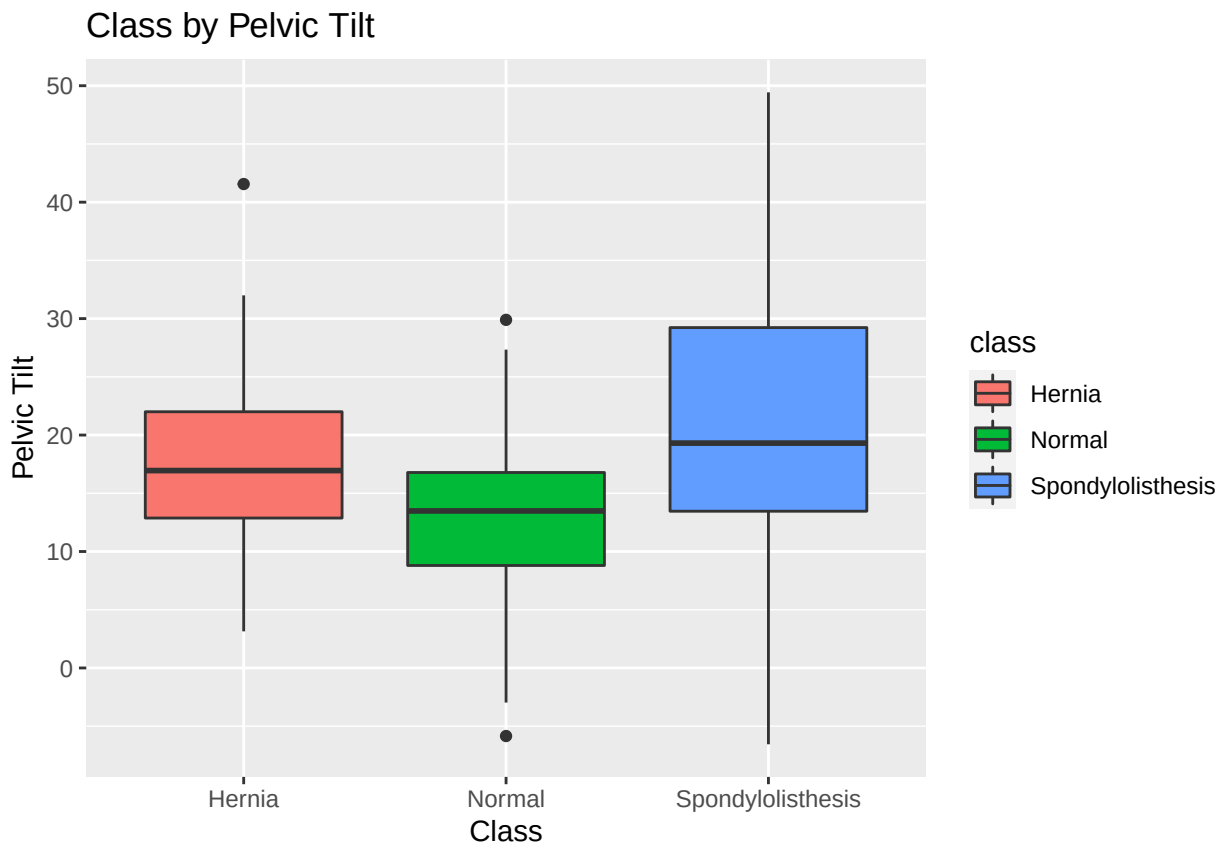
From the correlation plot, we can see that the first variable *pelvic_incidence* is either highly or moderately correlated with the rest of the variables. This suggests that it may be removed from model training or prediction.

We will now move on to explore more details of the data set by using boxplots.

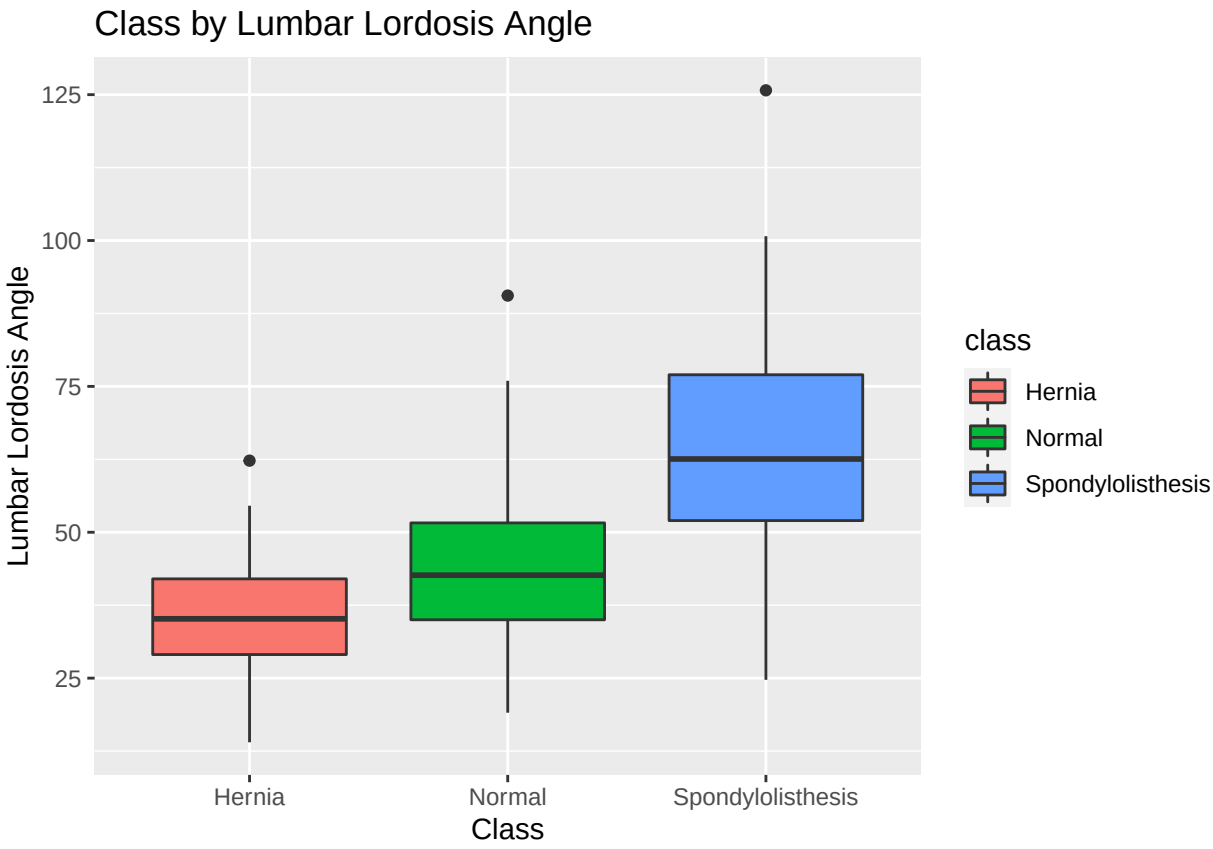
```
#Boxplot for "pelvic_incidence"
dat %>% ggplot(aes(x = class, y = pelvic_incidence, fill = class)) + geom_boxplot() +
          ggtitle("Class by Pelvic Incidence") + xlab("Class") +
        ylab("Pelvic Incidence")
```



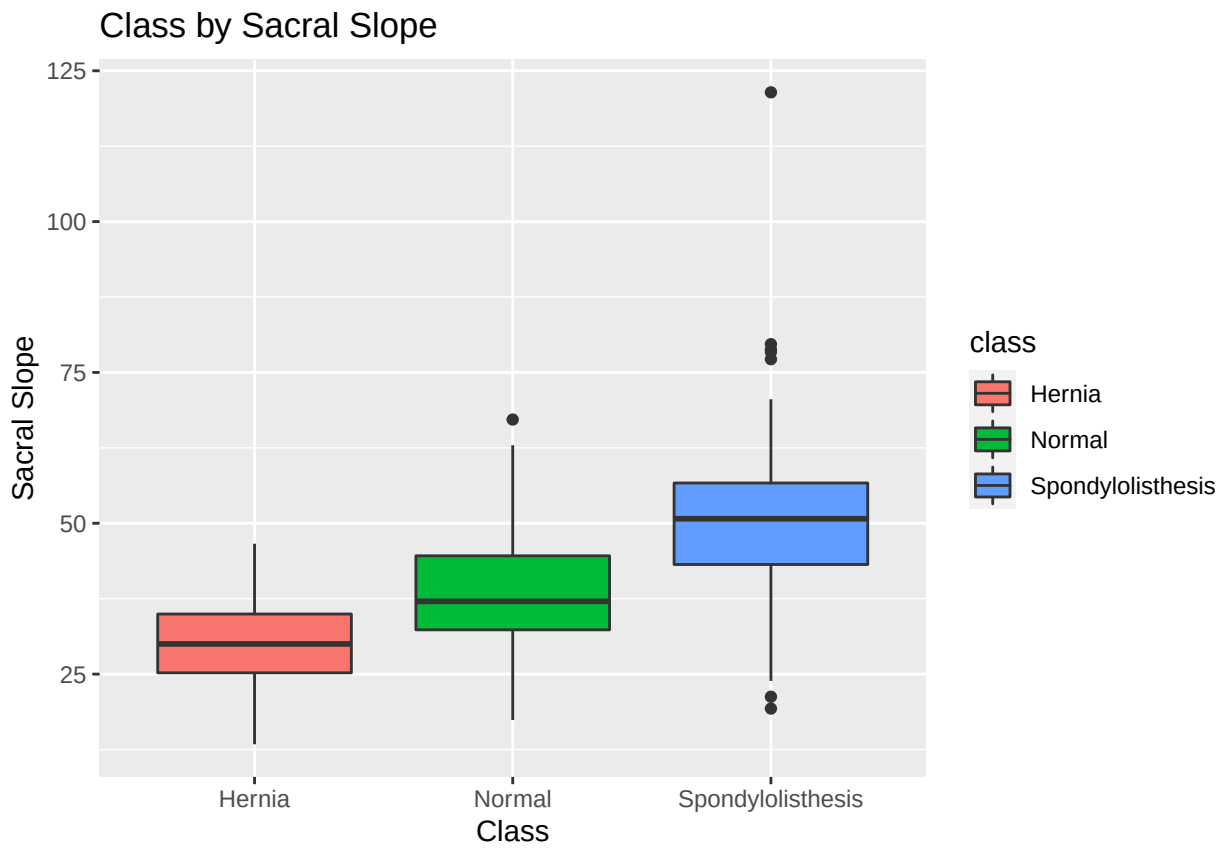

```
#Boxplot for "pelvic_tilt"
dat %>% ggplot(aes(x = class, y = pelvic_tilt, fill = class)) + geom_boxplot() +
          ggtitle("Class by Pelvic Tilt") + xlab("Class") + ylab("Pelvic Tilt")
```



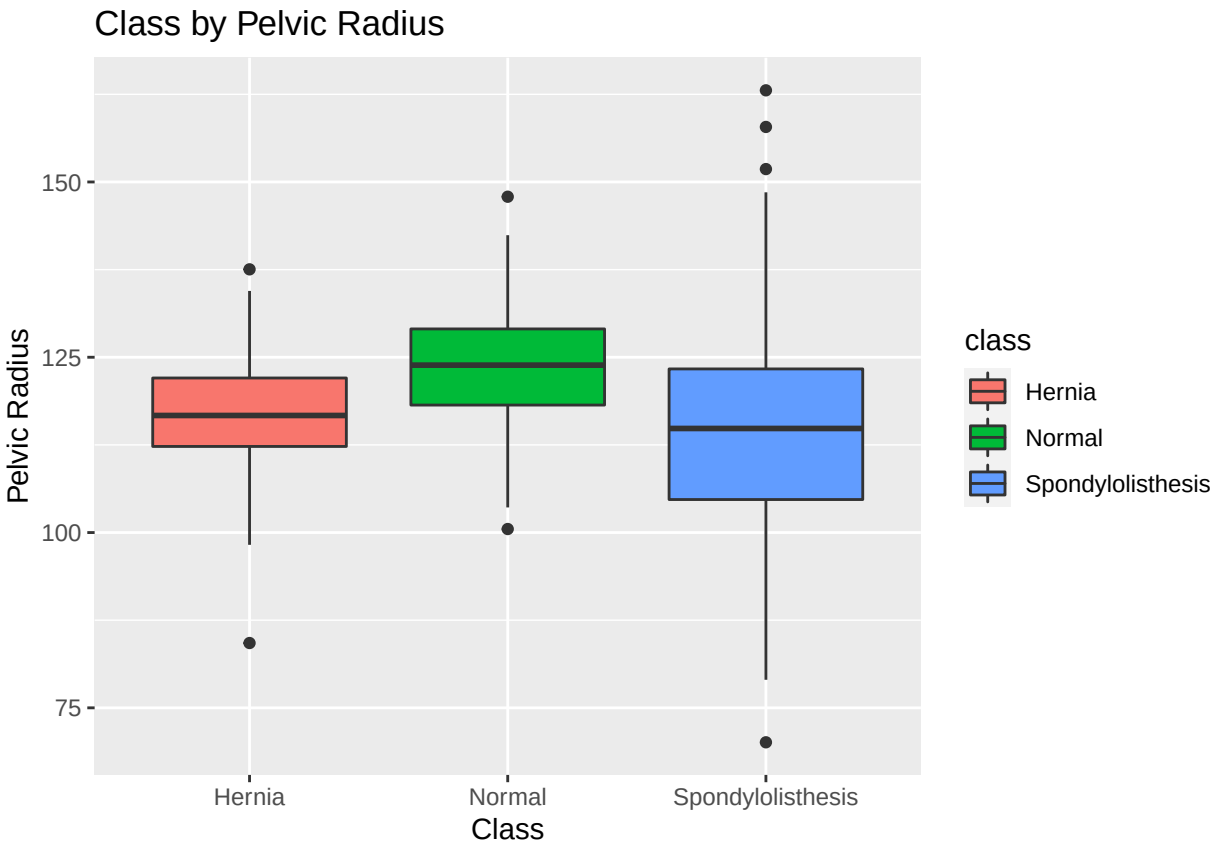

```
#Boxplot for "lumbar_loardosis_angle"
dat %>% ggplot(aes(x = class, y = lumbar_lordosis_angle, fill = class)) +
          geom_boxplot() +
          ggtitle("Class by Lumbar Lordosis Angle") + xlab("Class") +
          ylab("Lumbar Lordosis Angle")
```



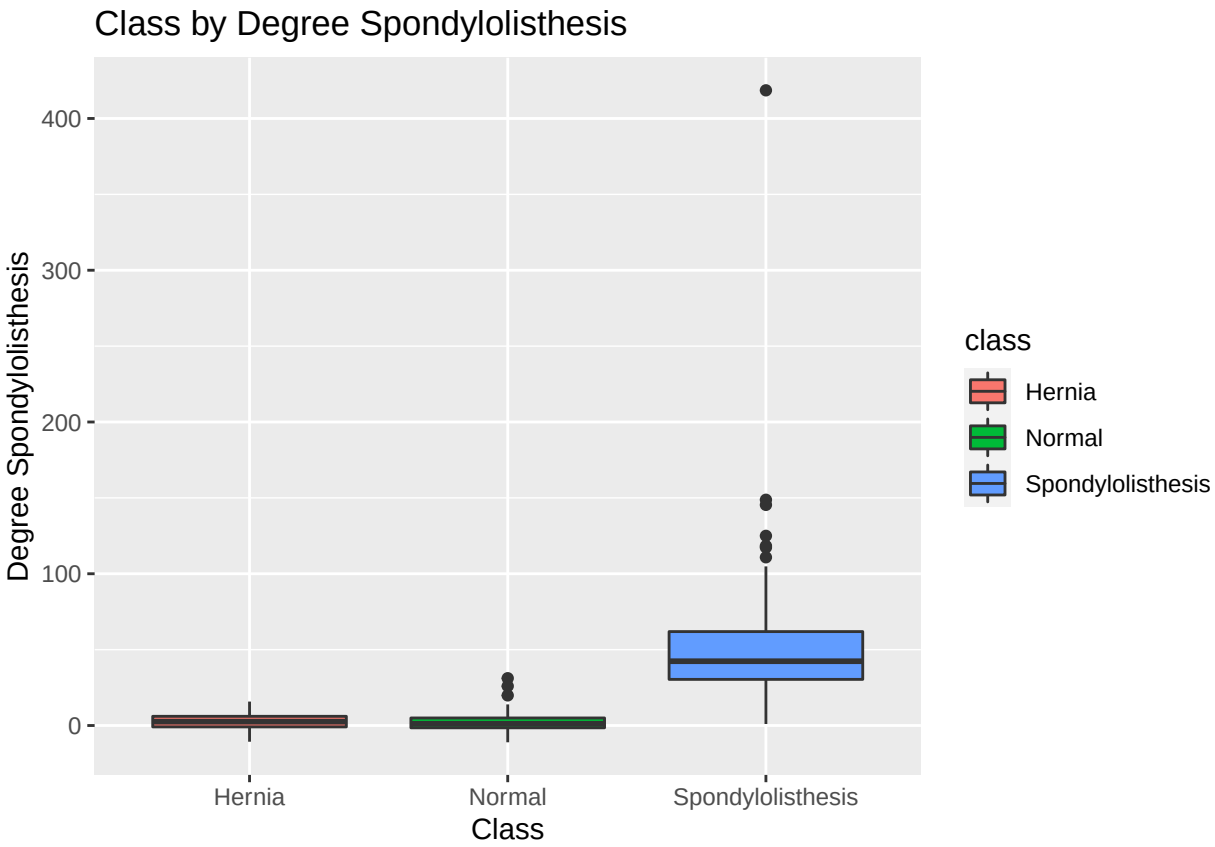

```
#Boxplot for "sacral_slope"
dat %>% ggplot(aes(x = class, y = sacral_slope, fill = class)) + geom_boxplot() +
          ggtitle("Class by Sacral Slope") + xlab("Class") + ylab("Sacral Slope")
```




```
#Boxplot for "pelvic_radius"
dat %>% ggplot(aes(x = class, y = pelvic_radius, fill = class)) + geom_boxplot() +
          ggtitle("Class by Pelvic Radius") + xlab("Class") + ylab("Pelvic Radius")
```




```
#Boxplot for "degree_spondylolishesis"
dat %>% ggplot(aes(x = class, y = degree_spondylolisthesis, fill = class)) +
          geom_boxplot() +
          ggtitle("Class by Degree Spondylolisthesis") + xlab("Class") +
          ylab("Degree Spondylolisthesis")
```



The above boxplots are plotted with whiskers of 1.5 * interquantile range (IQR). From the boxplots above, three variables (*pelvic_incidence*, *lumbar_lordosis_angle* and *sacral_slope*) display a similar trend with the range of values and medians increasing from *Hernia* to *Normal*, and to *Spondylolisthesis*. There are in general more overlaps of the IQR's between the first two classes, *Hernia* and *Normal*. **It should be noted that these three variable should be useful in separating the *Hernia* and *Spondylolisthesis* classes since the IQR's are entirely separable.**

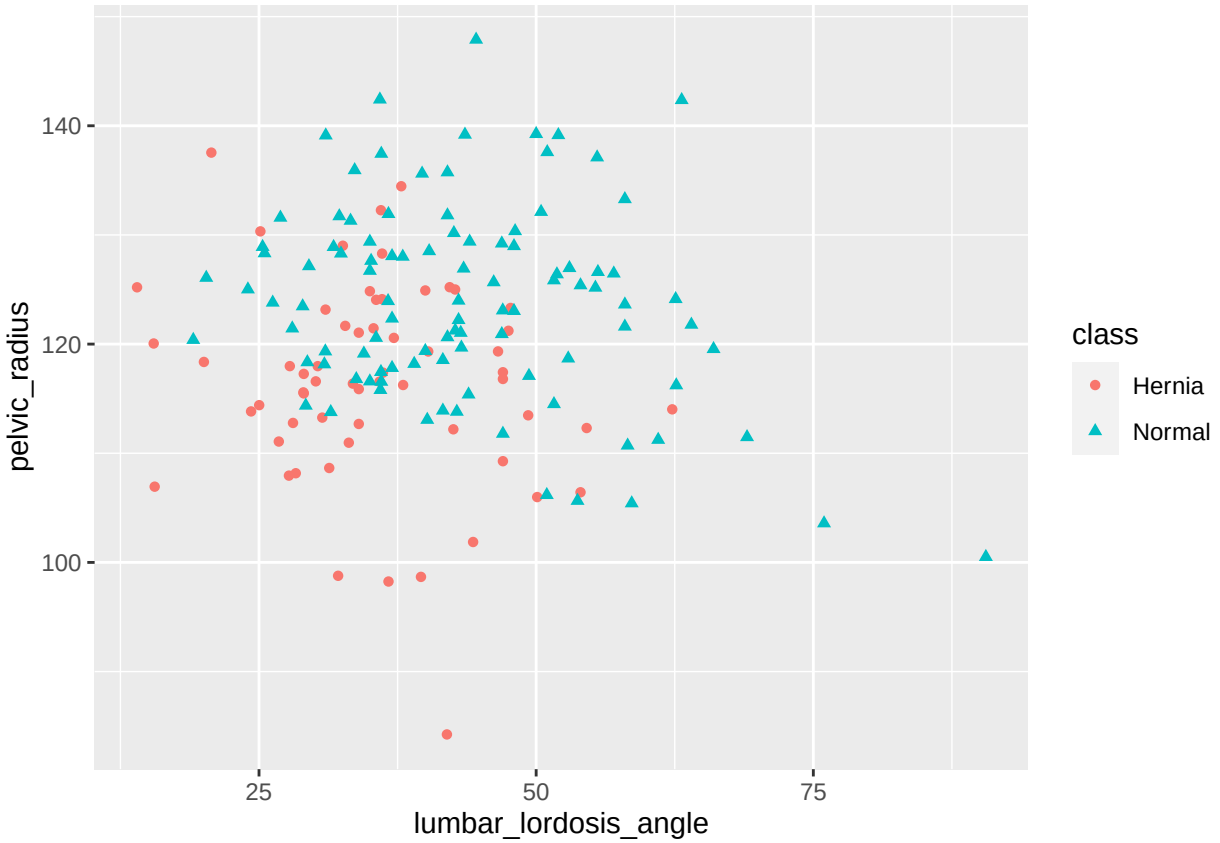
For the variable *pelvic_tilt*, the medians and IQR's for the *Hernia* and *Spondylolisthesis* classes roughly coincide/overlap entirely, with the median and IQR for the *Normal* class having lower values but still with some overlaps with the two other classes. For the *pelvic_radius* variable, the medians and IQR's for the *Hernia* and *Spondylolisthesis* classes also roughly coincide/overlap entirely, with the median and IQR for the *Normal* class having higher values but with some overlaps with the other two classes. **These two variables may be useful in separating the *Normal* class from the other two abnormal classes.**

For the last variable *degree_spondylolisthesis*, the IQR for the *spondylolisthesis* class is entirely separable from the other two classes. This suggests that this variable will be an important (if not the most important) variable in deciding if a data point falls into the *spondylolisthesis* class. The values for the other two classes largely coincide for this variable.

There are a handful of outliers as can be seen from the boxplots. But in this report all data points will be included in the analysis.

Knowing that the *spondylolisthesis* class could be predicted by the *degree_spondylolisthesis* variable, we may now try and see if we could tell the *Hernia* class from the *Normal* class by using the other variables. Based on the boxplots and the correlation plot, it appears that the variables *pelvic_radius* and *lumbar_lordosis_angle* (they are less correlated to the other variables) may perform reasonably well in predicting the two classes, as seen below:

```r
#examine distribution of Hernia and Normal classes (lumbar_lordosis_angle vs pelvic_radius)
dat %>% filter(class == "Hernia" | class == "Normal") %>%
          ggplot(aes(x = lumbar_lordosis_angle, y = pelvic_radius, shape = class,
          color = class)) + geom_point()
```



From the above plot, it appears that the two classes could indeed be separable, albeit not with a clear boundary but with an overlap of data points.

1.2 Data Partitioning

The data set will be split into train-test sets of 80% and 20%, respectively. A 70/30 split may not be ideal given the somewhat limited number of observations since we would want to have more data for training, and a 90/10 split may be at risk of under-representing any given class in the test set since the data set is somewhat imbalanced.

```r
set.seed(5, sample.kind="Rounding")
#80/20 split of training and test sets
index <- createDataPartition(y = dat$class, times = 1, p = 0.8, list = FALSE)
train <- dat[index,]
test <- dat[-index,]
```

2 METHODS/ANALYSIS

2.1 Models

The *caret* package will be used for training of the models and making predictions in this report. Before we proceed to the models, the training control is defined here based on a 10-fold cross-validation:

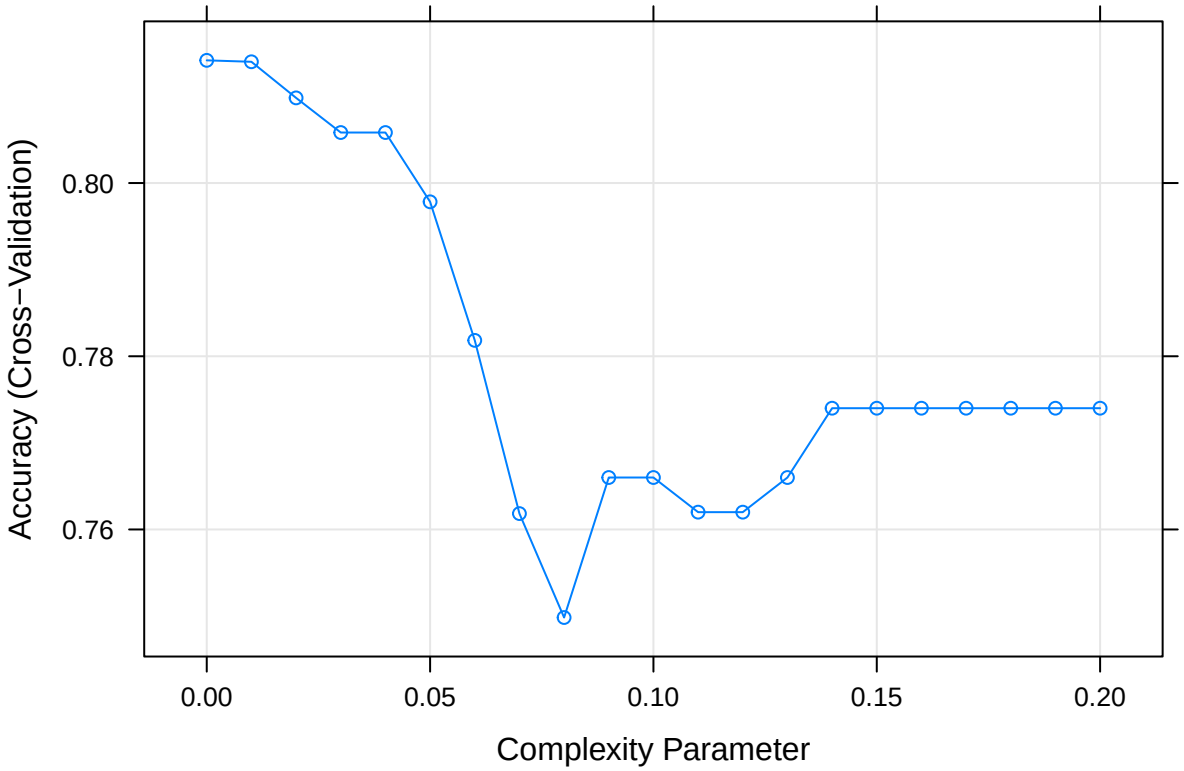
```
#train control for later model training (10-fold cross-validation)
trainControl <- trainControl(method = "cv", number = 10, savePredictions = "all")
```

2.1-1 Classification Tree

The classification tree is an obvious choice for a classification model. Classification trees are easy to visualize and they can help us better interpret how the different variables affect the outcomes. They are also less affected by outliers (since we have a few of those). However, they are prone to over-fitting and may not be the best performing model since they are inflexible and highly unstable to changes in training data (*Introduction to Data Science*).

```
set.seed(5, sample.kind="Rounding")
#classificaiton tree using rpart
train_rpart <- train(class ~ ., method = "rpart",
                     tuneGrid = data.frame(cp = seq(0.0, 0.2, by = 0.01)),
                     trControl = trainControl,
                     data = train)
```

```
#plot accuracy based on cp values
plot(train_rpart)
```




```
#first 5 results of cp values
train_rpart$results[1:5,]
```

```
##      cp Accuracy  Kappa AccuracySD KappaSD
## 1 0.00   0.8142 0.7006    0.04934 0.08597
## 2 0.01   0.8140 0.6994    0.05345 0.09272
## 3 0.02   0.8098 0.6917    0.06332 0.11189
## 4 0.03   0.8058 0.6843    0.06246 0.11082
## 5 0.04   0.8058 0.6837    0.06246 0.11009
```

The best performing complexity parameter (cp) value is 0, with an accuracy of 0.8142 based on the cross-validated results.

```
#confusion matrix for cross-validated rpart model
confusionMatrix(train_rpart)
```

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##                    Reference
## Prediction          Hernia Normal Spondylolisthesis
##   Hernia              10.9    6.9               0.0
##   Normal               8.5   24.2               2.0
##   Spondylolisthesis    0.0    1.2              46.4
```

```
##
##  Accuracy (average) : 0.8145
```

As observed earlier, the model in generally does well in predicting *Spondylolisthesis* class (average accuracy of 0.9587). But it does substantially worse on the first two classes, with accuracies of 0.5619 and 0.7492, respectively, for the *Hernia* and *Normal* classes.

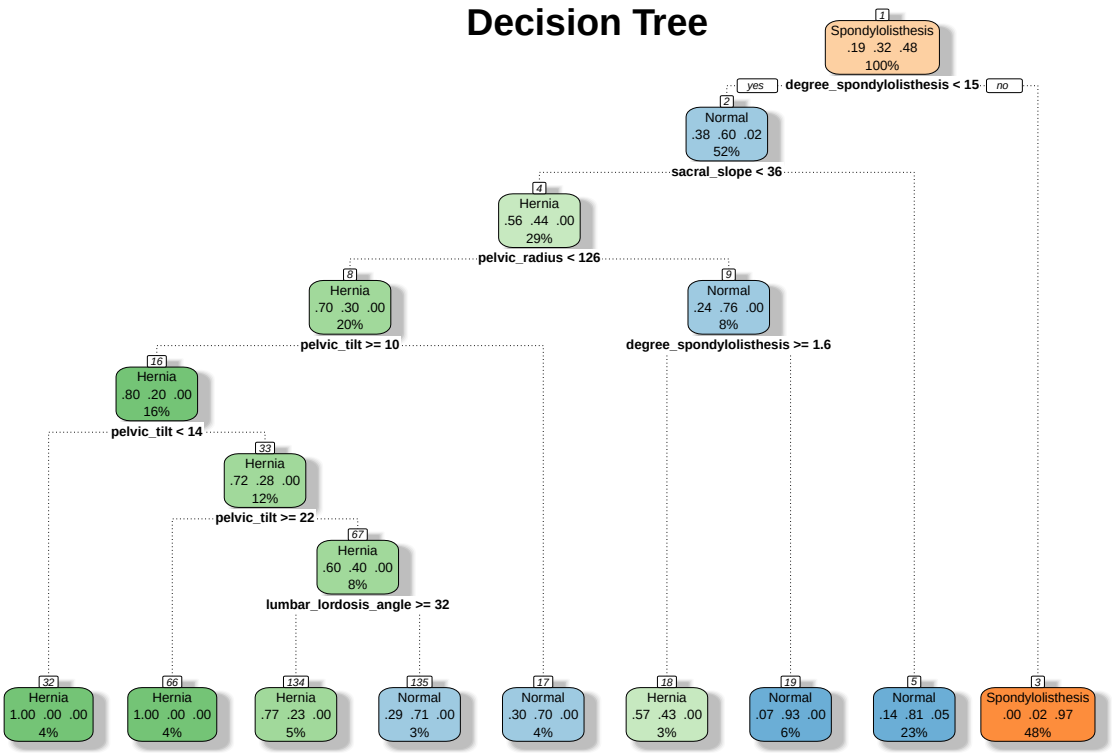
If we examine the variable importance in the model, we will find that the most important variable is indeed *degree_spondylolisthesis*, with the other variables (apart from *pelvic_tilt*) having varying degrees of importance. The zero importance of *pelvic_tilt* suggests that it may be left out in the analysis (we saw earlier that it is highly correlated with the other variables).

```
#variable importance based on cross-validated rpart
varImp(train_rpart)
```

```
## rpart variable importance
##
##                          Overall
## degree_spondylolisthesis   100.0
## sacral_slope                38.7
## lumbar_lordosis_angle       32.7
## pelvic_incidence            29.2
## pelvic_radius               22.5
## pelvic_tilt                  0.0
```

The decision tree based on this model is:

```
#plot decision tree based on rpart
fancyRpartPlot(train_rpart$finalModel, main = "Decision Tree", sub = "")
```



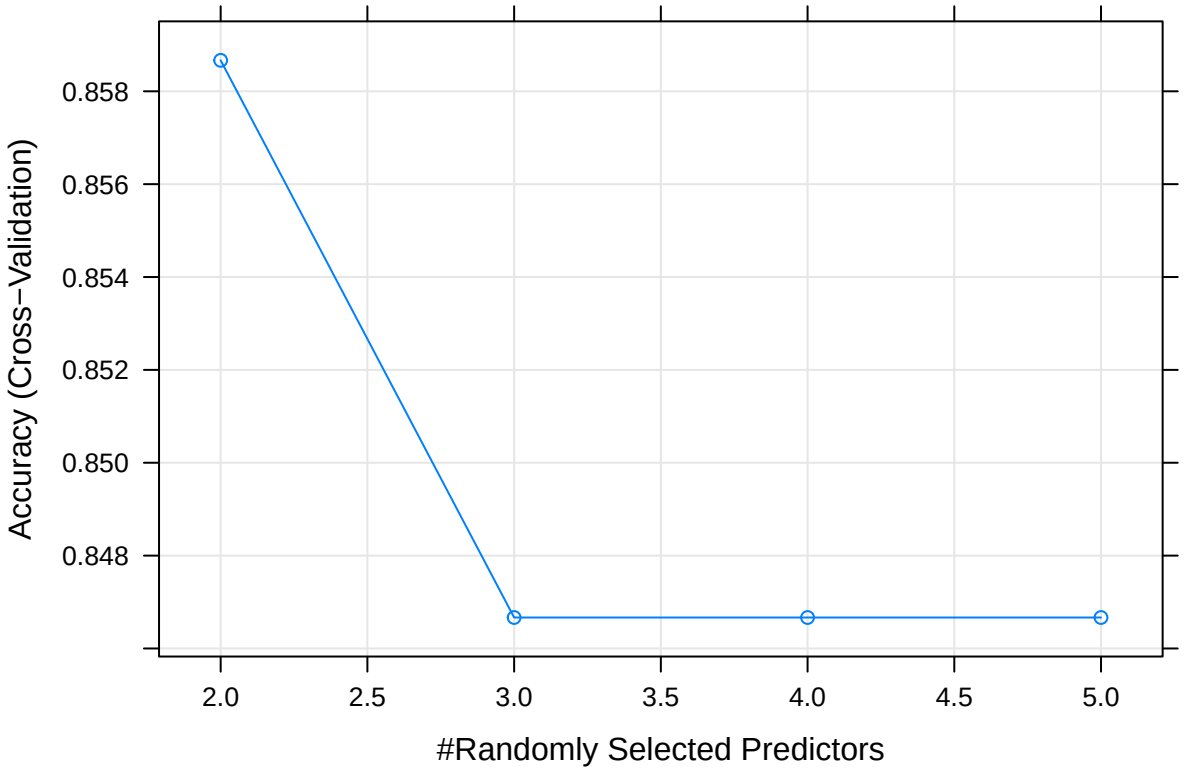
2.1-2 Random Forest

Random forests can address the shortcomings of decision trees (overfitting and susceptible to training data changes), improving prediction performance and reducing instability by averaging many decision trees constructed with randomness (*Introduction to Data Science*). However, random forests are difficult to interpret since we will lose the interpretability of a decision tree.

```
set.seed(5, sample.kind="Rounding")
#random forest using rf package
train_rf <- train(class ~ ., method = "rf",
                  ntree = 1000, tuneGrid = data.frame(mtry = seq(2, 5)),
                  trControl = trainControl,
                  data = train)
```

The *ntree* parameter of 1000 is chosen which is sufficient for the current problem (e.g. re-running with *ntree = 2000* would result in a similar accuracy).

```
#plot accuracies (10-fold cross-validation) for mtry
plot(train_rf)
```




```
#confusion matrix for cross-validated rf model
confusionMatrix(train_rf)
```

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##                     Reference
## Prediction           Hernia Normal Spondylolisthesis
##   Hernia               12.9    4.0               0.0
##   Normal                6.5   25.4               0.8
##   Spondylolisthesis     0.0    2.8              47.6
##
##  Accuracy (average) : 0.8589
```

The random forest model managed to produce a better fit to the cross-validated data compared to the decision tree model as expected.

```
#variable importance based on cross-validated rf
varImp(train_rf)
```

```
## rf variable importance
##
##                           Overall
## degree_spondylolisthesis   100.0
```

```
## sacral_slope               16.9
## lumbar_lordosis_angle      13.4
## pelvic_incidence           12.3
## pelvic_radius              11.9
## pelvic_tilt                 0.0
```

The most important variable for the random forest model is *degree_spondylolisthesis* as expected, but the subsequent four variables have some, but less importance, when compared to the earlier classification tree. This should be due to the averaging effect of the random forest, which also gives it a better fit.

2.1-3 Support Vector Machine

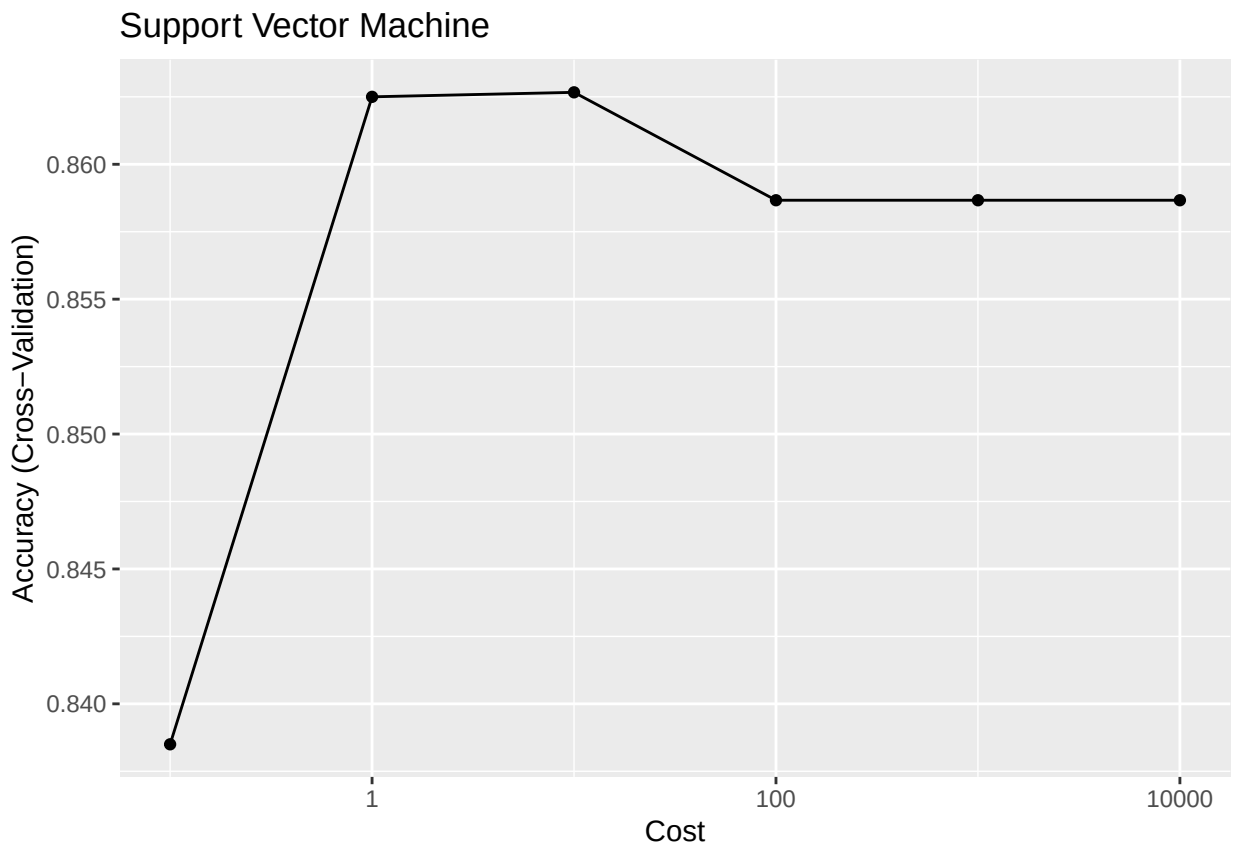
The last model fitted in this report is the support vector machine (SVM). It can be used for classification, by creating a hyperplane that separate the data into categories, keeping the margin between different categories as wide as possible.

We will be using with a linear kernel *svmLinear* from the *kernlab* package. In addition to the tuning and training control parameters, we would also need to center and scale the training data before running the model, since SVM is a distance-based method.

```
set.seed(5, sample.kind="Rounding")
#svm using svmLinear
train_svm <- train(class ~ ., method = "svmLinear",
                   preProcess = c("center", "scale"), #centering and scaling
                   trControl = trainControl,
                   #cost parameter
                   tuneGrid = data.frame(C = c(0.1, 1, 10, 100, 1000, 10000)),
                   data = train)
```

The 10-fold cross-validated accuracies are plotted in the following for the different values of tuning parameter C:

```
#plot accuracies (10-fold cross-validation) for C
train_svm %>% ggplot(aes(x = C, y = Accuracy)) + scale_x_log10() +
                ggtitle("Support Vector Machine")
```



The SVM has resulted in an improvement in the cross-validated accuracy (0.8627) compared to random forest.

2.2 Results

Based on the 10-fold cross-validated accuracies from the fitted models in the previous section, the best result is obtained from the SVM model, which should be the chosen model to be tested against the test set. However, for educational purposes in this report, we will be testing all the three models against the test set to see if indeed the SVM model would have the best performance, or if any of these would have problems like over-fitting.

2.2-1 Classification Tree Prediction

The predictions from the classification tree model as well as the accuracy obtained against the test set are given in the following:

```
#predictions from rpart
pred_rpart <- predict(train_rpart, test)
#accuracy of rpart prediction against test set
acc_rpart <- confusionMatrix(pred_rpart, test$class)$overall["Accuracy"]
acc_rpart
```

```
## Accuracy
##   0.7903
```

The prediction accuracy obtained is close to but only slightly worse than its earlier cross-validated accuracy, which suggests that the model is not over-fitted. We will be keeping track of the accuracy results:

```
#keep track of model results
acc_results <- tibble(method = "Classification Tree",
                  "10-fold CV" = as.character(round(train_rpart$results[1,2],4)),
                  "Test Set" = as.character(round(acc_rpart,4)))
```

2.2-2 Random Forest Prediction

The predictions from the random forest as well as the accuracy obtained against the test set are given in the following:

```
#predictions from rf
pred_rf <- predict(train_rf, test)
#accuracy of rf prediction against test set
acc_rf <- confusionMatrix(pred_rf, test$class)$overall["Accuracy"]
acc_rf
```

```
## Accuracy
##   0.8065
```

The accuracy of the predictions from the random forest model is only a slight improvement over the classification tree's. The predictions accuracy is lower than its cross-validated accuracy (0.8065 vs 0.8587), but it should not be a cause for concern of possible over-fitting (since random forest is less prone to over-fitting, and considering the limited number of observations and the somewhat imbalanced data, it should be expected that there would be some discrepancies between the cross-validated results and test set results).

```
#keep track of model results
acc_results <- acc_results %>% bind_rows(tibble(method = "Random Forest",
                      "10-fold CV" = as.character(round(train_rf$results[1,2],4)),
                      "Test Set" = as.character(round(acc_rf,4))))
```

2.2-3 Support Vector Machine Prediction

The predictions from the support vector machine as well as the accuracy obtained against the test set are given in the following:

```
#predictions from SVM
pred_svm <- predict(train_svm, test)
#accuracy of SVM prediction against test set
acc_svm <- confusionMatrix(pred_svm, test$class)$overall["Accuracy"]
acc_svm
```

```
## Accuracy
##    0.871
```

The predictions accuracy obtained by the SVM is close to but slightly better than its cross-validated accuracy (0.871 vs 0.8627). It is also the best performing based on the test set results. The results table is given in the following:

```r
#keep track of model results
acc_results <- acc_results %>% bind_rows(tibble(method = "Support Vector Machine",
                      "10-fold CV" = as.character(round(train_svm$results[3,2],4)),
                      "Test Set" = as.character(round(acc_svm,4))))
acc_results
```

```
## # A tibble: 3 x 3
##   method                 `10-fold CV` `Test Set`
##   <chr>                  <chr>        <chr>
## 1 Classification Tree    0.8142       0.7903
## 2 Random Forest          0.8587       0.8065
## 3 Support Vector Machine 0.8627       0.871
```

3 CONCLUSIONS

A total of three machine learning models (classification tree, random forest and support vector machine) are trained and tested with a data set of biomechanical features of orthopedic patients. These are classification models which can be used to classify patients based on six features, with each patient belonging to one of three categories: *Normal* (100 patients), *Disk Hernia* (60 patients) or *Spondylolisthesis* (150) patients.

Based on the somewhat limited number of observations and the slight imbalanced classes, the train-set split is chosen to be 80%/20%, in order to find a balance between having sufficient training data and guarding against having an imbalanced test set. The support vector machine (SVM) model has the highest cross-validated accuracy (0.8627), and correspondingly the predictions accuracy of the SVM model against the test set is 0.871. The following is a summary of the results.

```r
#results summary
acc_results
```

```
## # A tibble: 3 x 3
##   method                 `10-fold CV` `Test Set`
##   <chr>                  <chr>        <chr>
## 1 Classification Tree    0.8142       0.7903
## 2 Random Forest          0.8587       0.8065
## 3 Support Vector Machine 0.8627       0.871
```

The predictions accuracy obtained above is reasonably good. With more data points, better fine-tuned and/or more advanced models, this could be a valuable tool to make quick predictions whether a patient could be suffering from orthopedic issues based on the values of the features, potentially helping medical specialists in the diagnostics.

3.1 Limitations

One limitation for the current classification task is the somewhat limited number of observations (310), which limits the number of training data available for model training. The other limitation is that the class labels are somewhat imbalanced. These can cause discrepancies between the cross-validated accuracy and test set accuracy, e.g. if we vary the random seed number we could obtain a better or slightly worse test set accuracy based on chance. However, the effects caused by these limitations should be, and appear to be, acceptable for the current classification task.

3.2 Future Work

For data sets with even more limited observations, or if we would like to address some of the effects caused by this in the current task, we could consider using repeated k-fold cross-validations or bootstrapping in the cross-validation step. With regards to the imbalanced labels, we could consider to employ techniques such as oversampling to try and alleviate the impacts.

To further improve upon the performance of a classification model for this task, we could include other models and investigate their performances. In addition, an ensemble of multiple models could be tried to see if any meaningful improvements could be gained.