ML Project: Biomechanical Patient Features

Lorenzo Luna

Introduction

This project will utilize and compare different machine learning techniques applied to a dataset containing orthopaedic patient data, with the goal of predicting the patient's condition. The dataset used has 310 observations, each corresponding to a patient. Each observation has 6 variables and one class identifier, which is the target of our prediction.

```
str(dataset)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 310 obs. of  7 variables:
##  $ pelvic_incidence        : num  63 39.1 68.8 69.3 49.7 ...
##  $ pelvic_tilt             : num  22.55 10.06 22.22 24.65 9.65 ...
##  $ lumbar_lordosis_angle   : num  39.6 25 50.1 44.3 28.3 ...
##  $ sacral_slope            : num  40.5 29 46.6 44.6 40.1 ...
##  $ pelvic_radius           : num  98.7 114.4 106 101.9 108.2 ...
##  $ degree_spondylolisthesis: num  -0.254 4.564 -3.53 11.212 7.919 ...
##  $ class                   : chr  "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()
##   .. )
```

We want the class variable to be a factor instead of a character.

```
#make class a factor variable
dataset$class = as.factor(dataset$class)
```

The dataset used has three classes: Normal, Hernia, and Spondylolisthesis. A version of this dataset merging Hernia and Spondylolisthesis into a single Abnormal class is also available, but I have chosen to work with three classes instead.

This is therefore a multinomial classification problem. Machine learning algorithms that only work on a binomial classification problem (like logistic regression) can be turned into multinomial classifiers by running the algorithm in a One-versus-All way, where the algorithm is trained to identify each class individually, then the class is decided by running all the individual classifiers and picking the prediction with the highest confidence.

It is important to note that the prevalence of Spondylolisthesis is high, while Hernia is the least frequent class. This can affect the overall accuracy of our model.

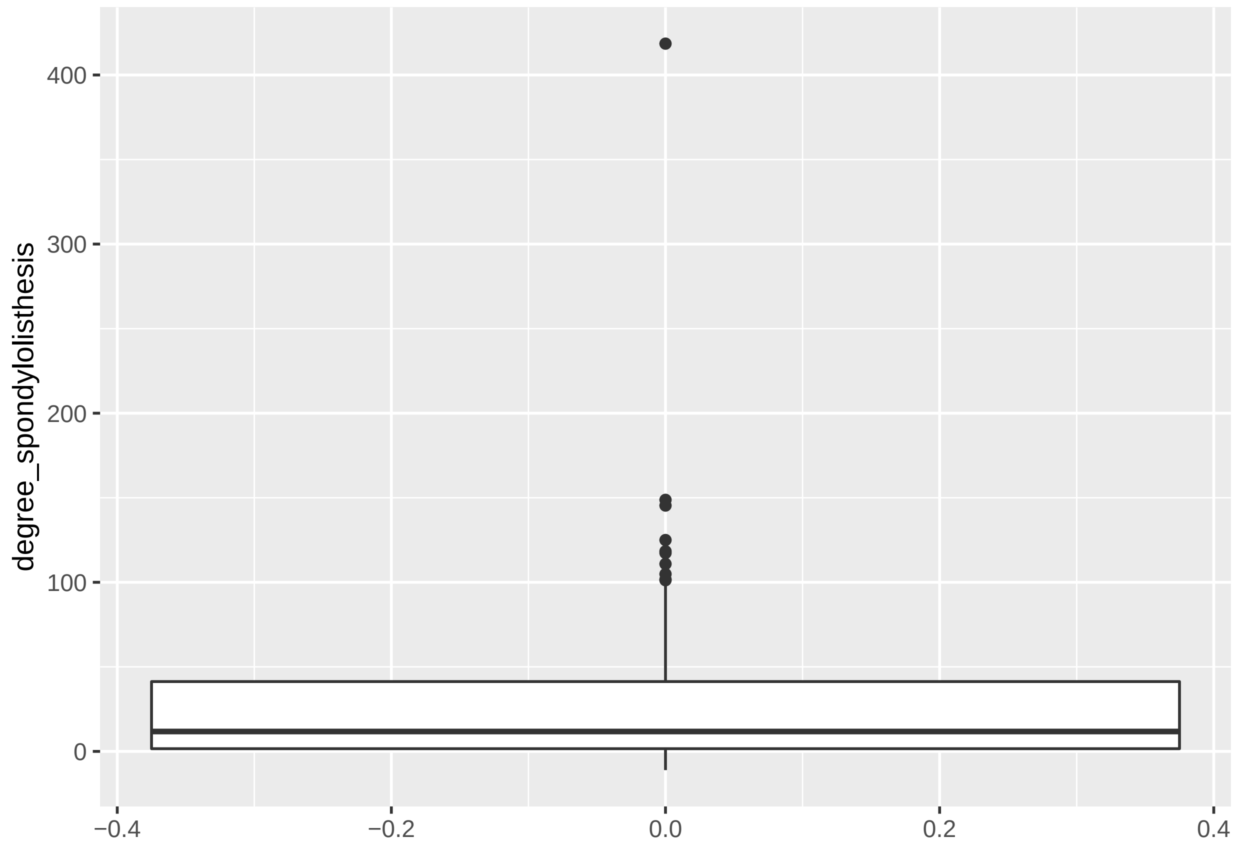
```
dataset %>% group_by(class) %>% summarize(count = n(), prevalence = count/310)
```

```
## # A tibble: 3 x 3
##   class              count prevalence
##   <fct>              <int>      <dbl>
## 1 Hernia               60      0.194
## 2 Normal              100      0.323
## 3 Spondylolisthesis   150      0.484
```

We will run popular machine learning algorithms suitable to solve this problem and compare their performance and stability.

Method

Exploratory data analysis shows that there is an outlier with an extreme value of degree_spondylolisthesis.



```
## # A tibble: 1 x 7
##   pelvic_incidence pelvic_tilt lumbar_lordosis~ sacral_slope pelvic_radius
##              <dbl>       <dbl>            <dbl>        <dbl>         <dbl>
## 1             130.        8.40             48.4         121.          108.
## # ... with 2 more variables: degree_spondylolisthesis <dbl>, class <fct>
```

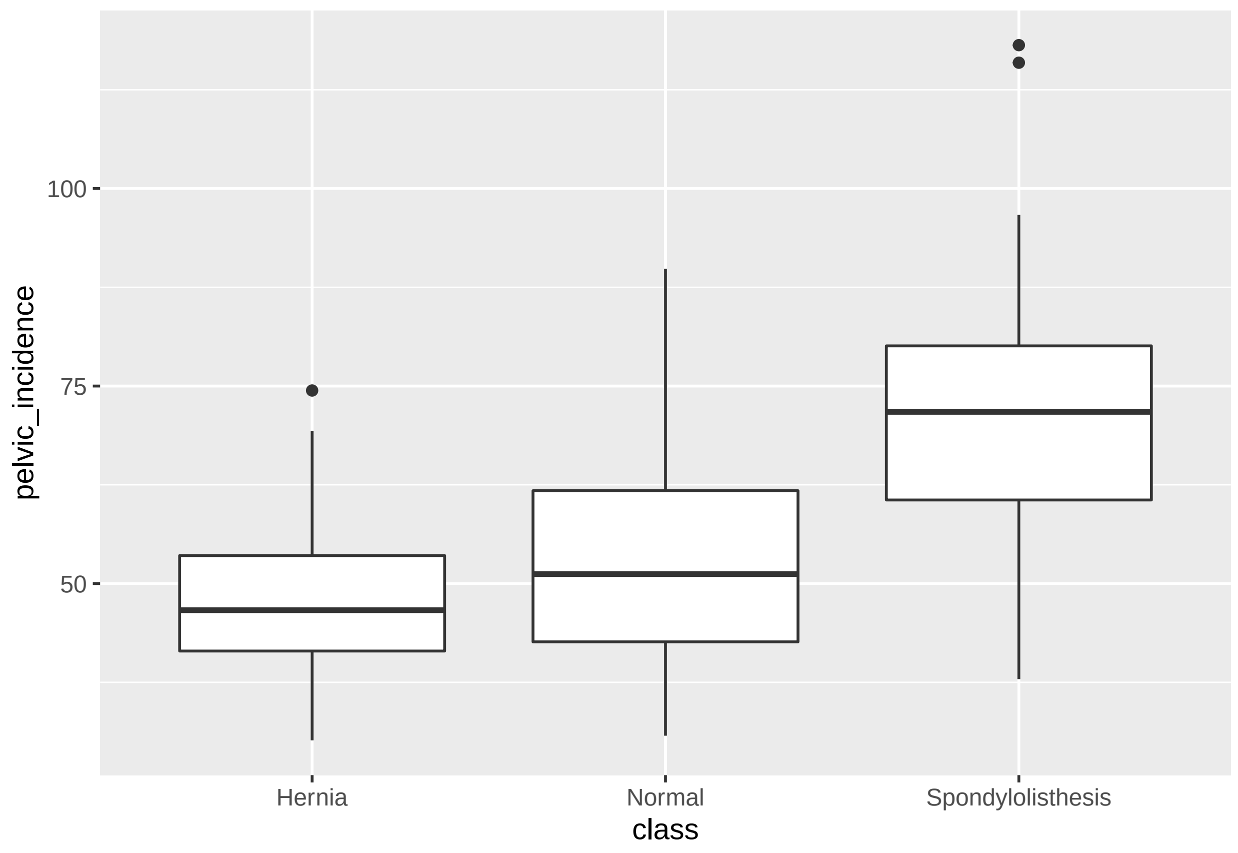
This observation will be discarded while performing future exploratory data analysis, for the sake of making plots more informative. Nonetheless, the observation will still be used as part of the training/test datasets as it still corresponds to a patient's data and must be accounted for.

The dataset is randomly split into training/test datasets, with 30% of the data being kept for testing.

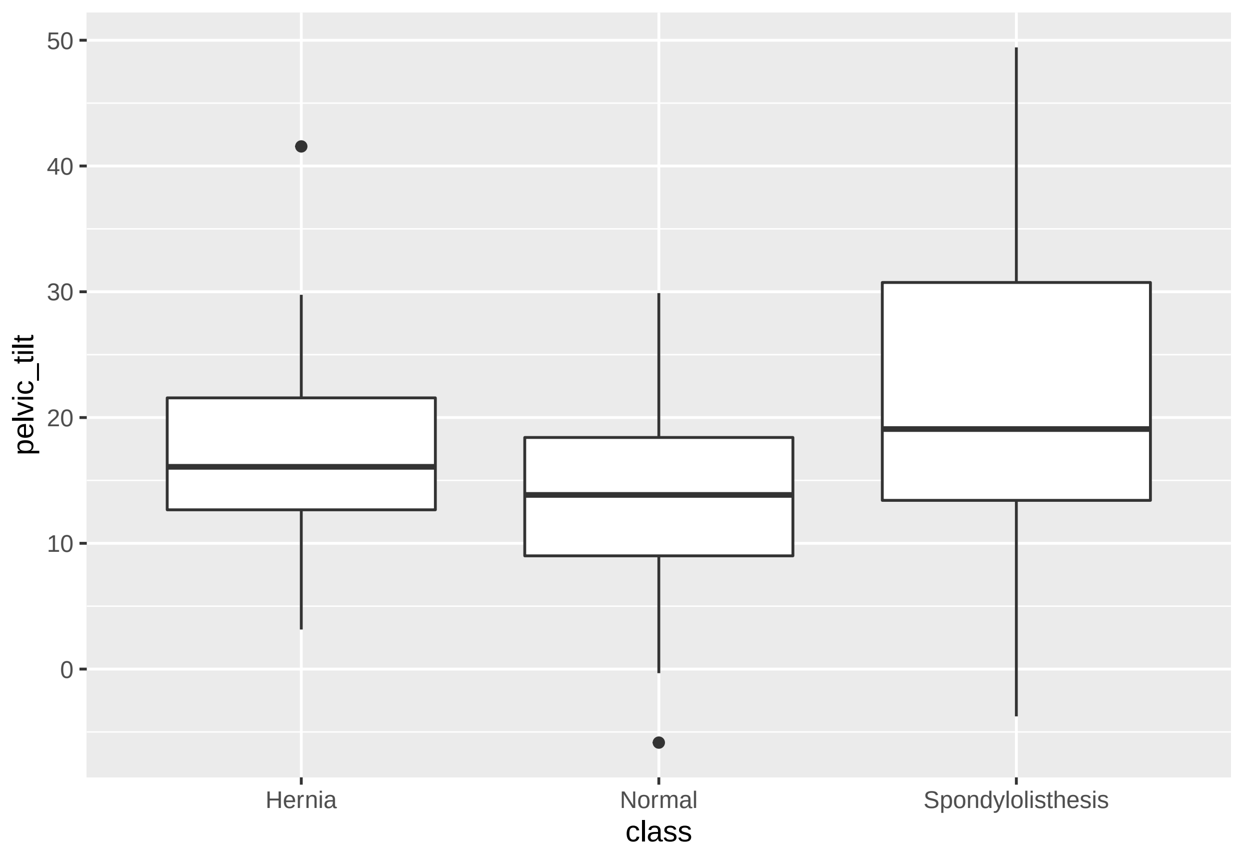
```
index = createDataPartition(dataset$class, p = 0.7, list = FALSE)
train = dataset[index,]
test = dataset[-index,]
```

To begin exploring the training dataset we plot boxplots of class versus all the variables available. We can notice some patterns. Notably, most Normal and Hernia cases have a value of degree_spondylolisthesis close to zero, while most cases of Spondylolisthesis have a much higher value.

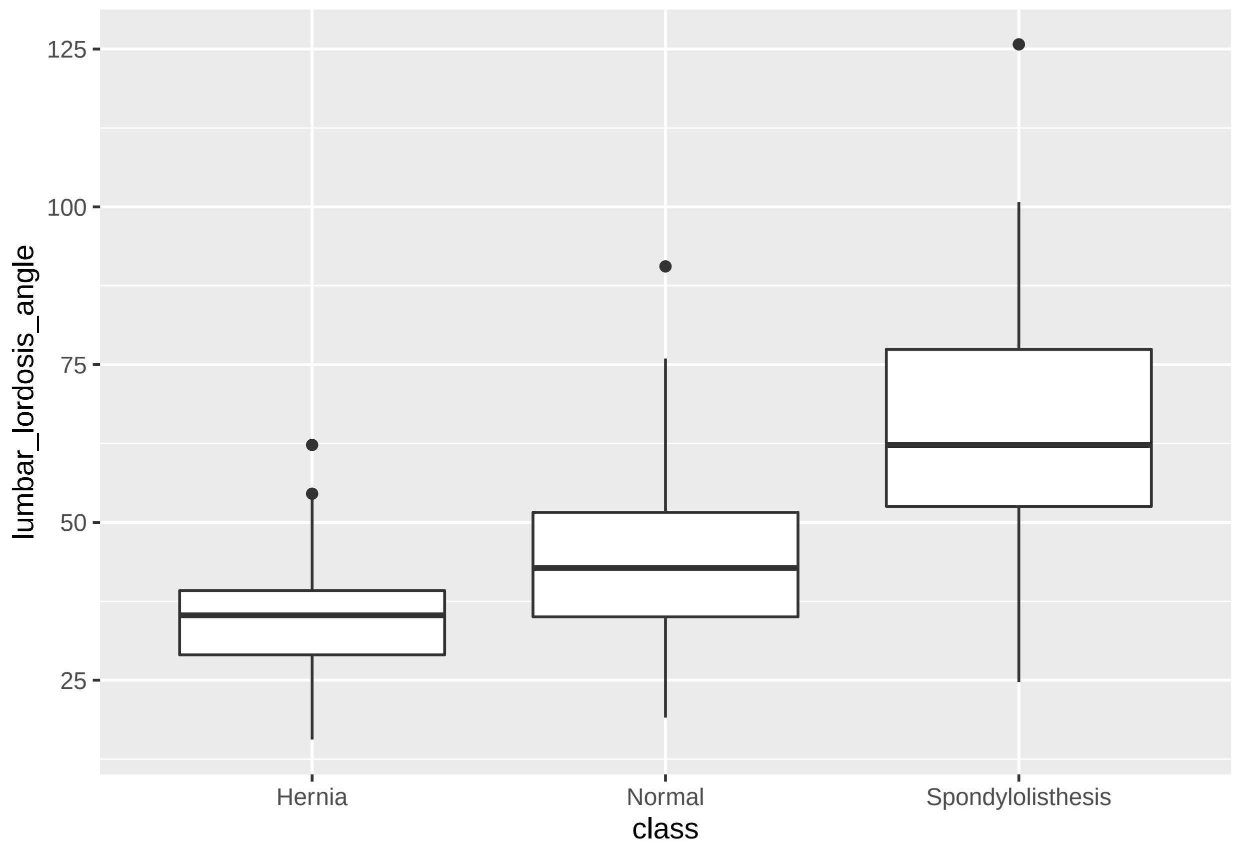
```r
train %>% filter(degree_spondylolisthesis < 400) %>%
  ggplot(aes(class , pelvic_incidence)) + geom_boxplot()
```



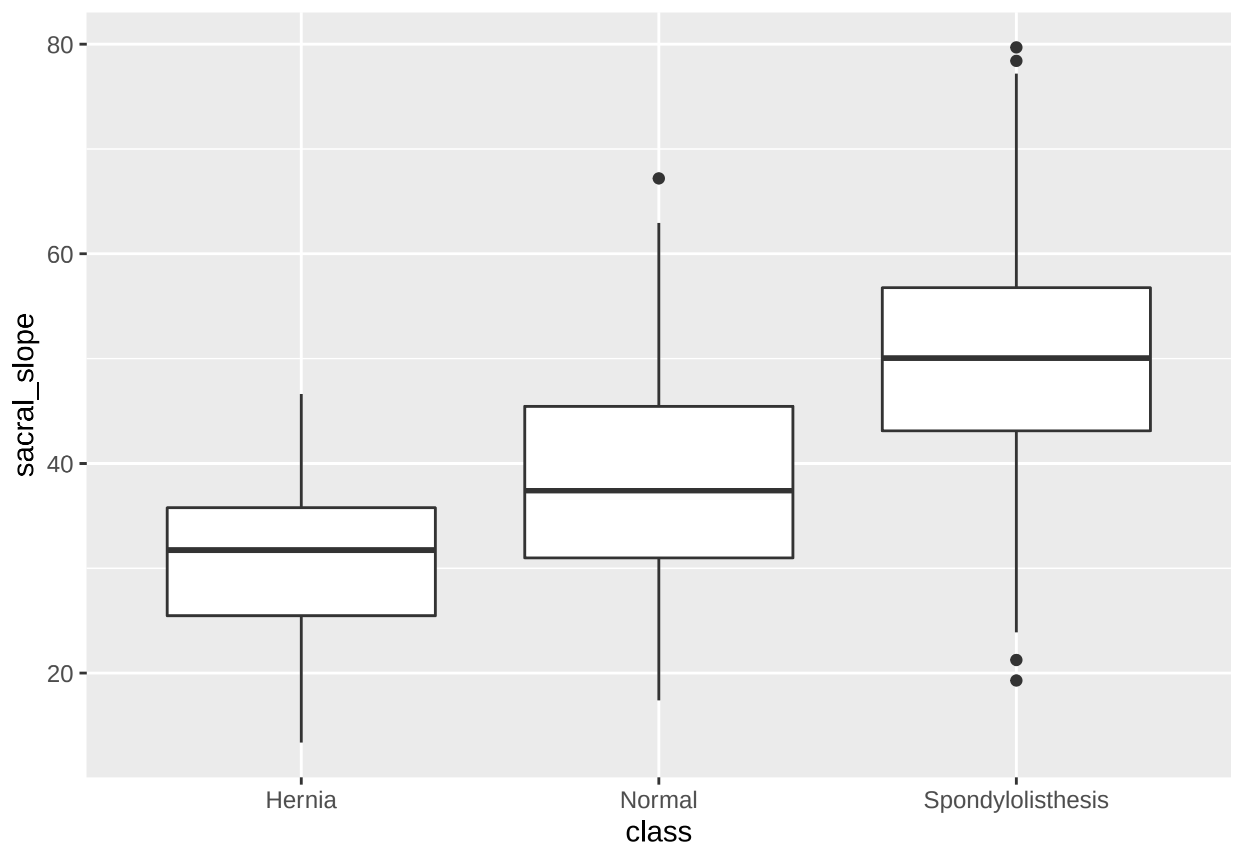
```r
train %>% filter(degree_spondylolisthesis < 400) %>%
  ggplot(aes(class , pelvic_tilt)) + geom_boxplot()
```



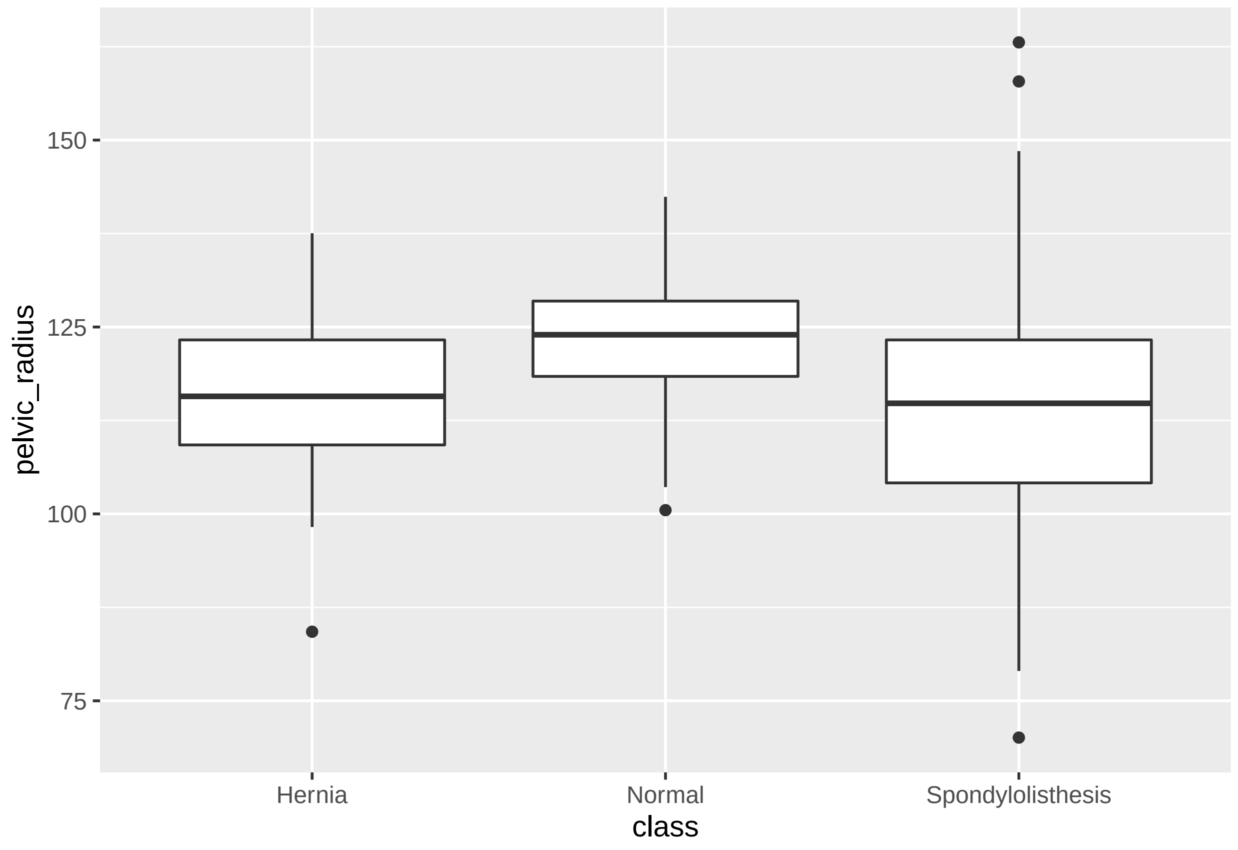
```
train %>% filter(degree_spondylolisthesis < 400) %>%
  ggplot(aes(class , lumbar_lordosis_angle)) + geom_boxplot()
```



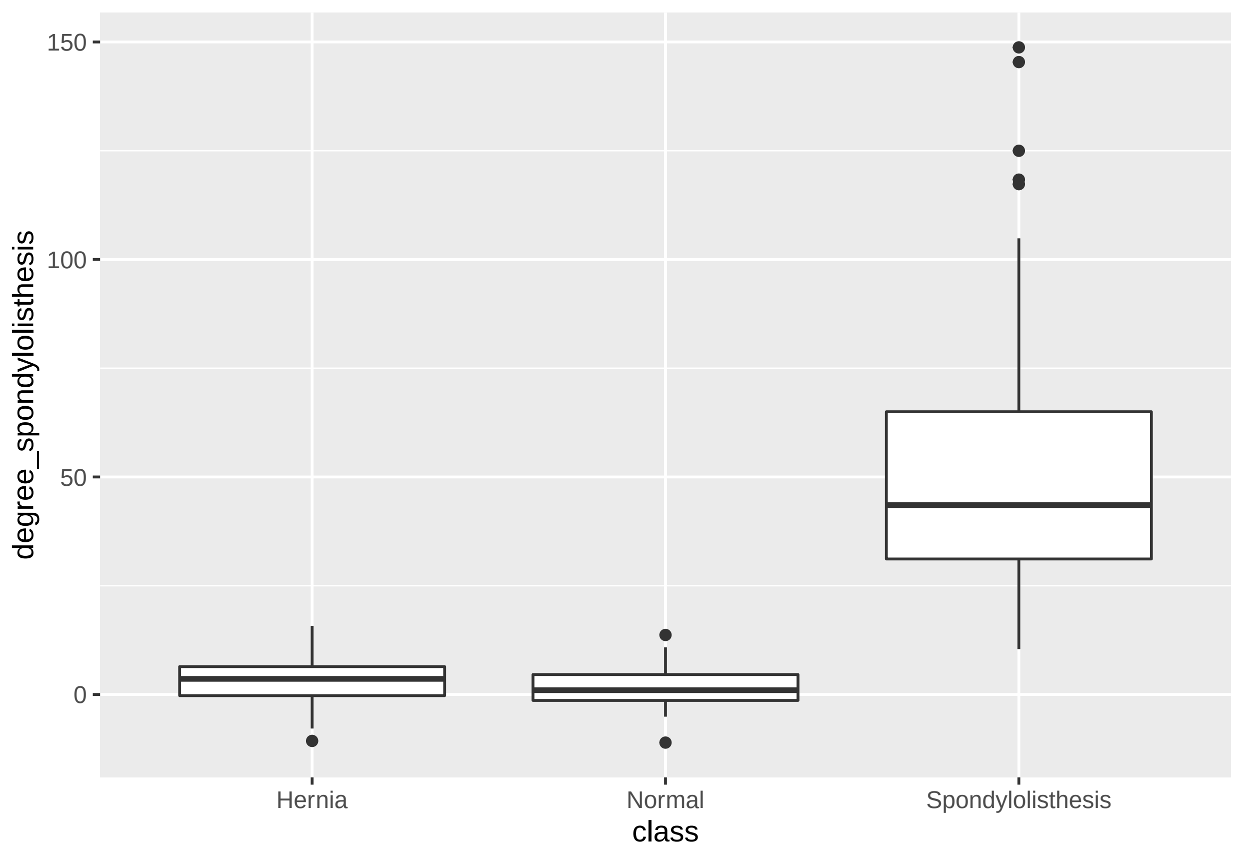
```
train %>% filter(degree_spondylolisthesis < 400) %>%
  ggplot(aes(class , sacral_slope)) + geom_boxplot()
```



```
train %>% filter(degree_spondylolisthesis < 400) %>%
  ggplot(aes(class , pelvic_radius)) + geom_boxplot()
```



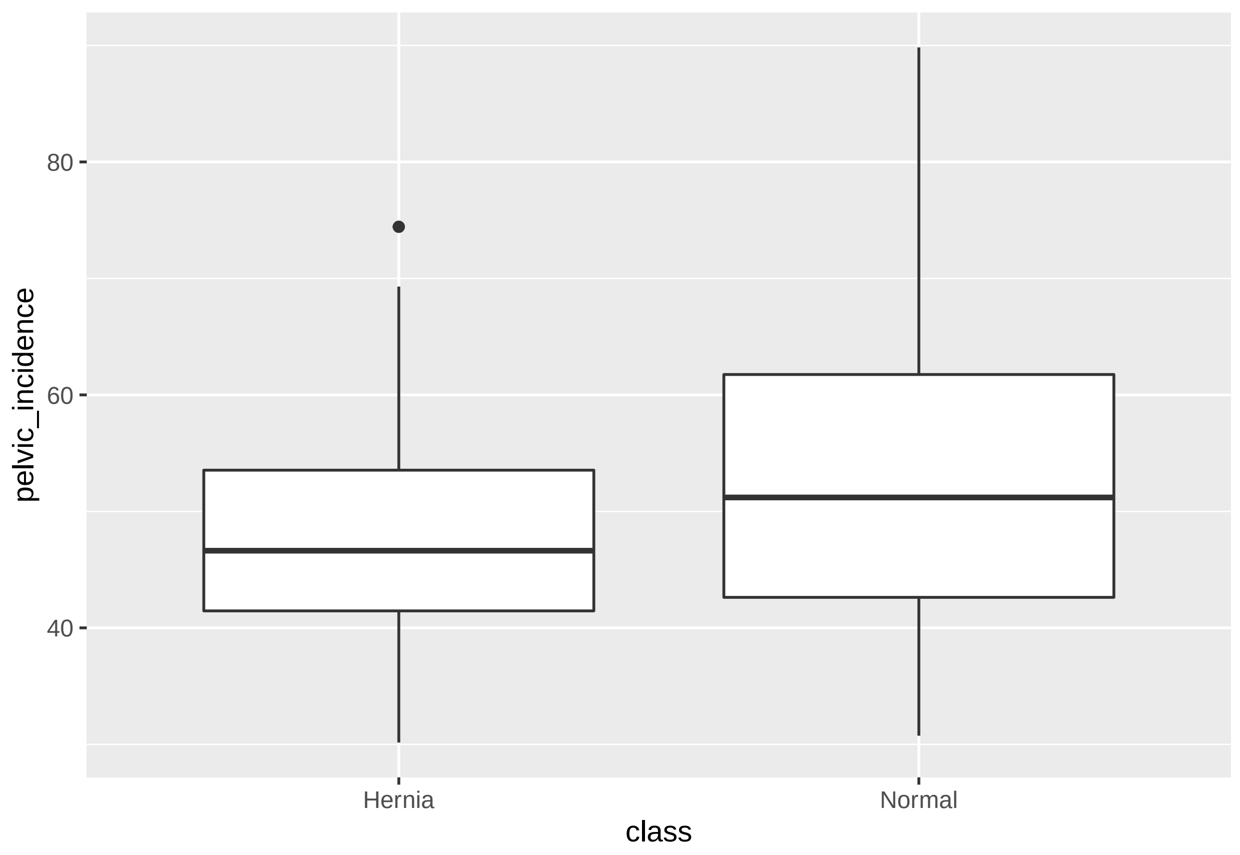
```r
train %>% filter(degree_spondylolisthesis < 400) %>%
  ggplot(aes(class , degree_spondylolisthesis)) + geom_boxplot()
```



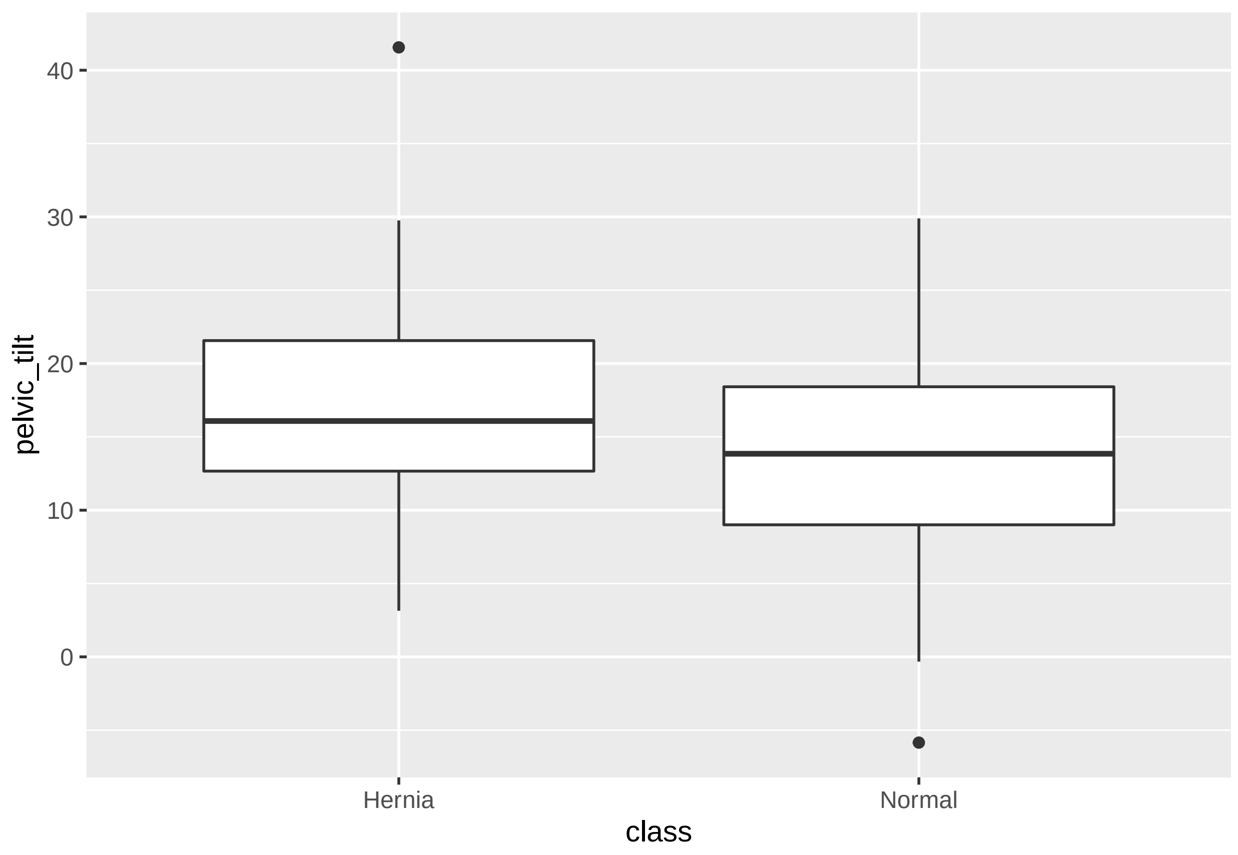
There seems to be a sharp cutoff value after which most cases are Spondylolisthesis. Therefore, any model dividing the decision regions linearly will be sufficiently accurate for classifying these cases. We will try using a classification tree model and analyze its precision depending on the three classes.

To further study the difference between Hernia and Normal cases, exploratory data analysis is performed on a subset of the training dataset only including those cases. Plotting the boxplots of class versus variable shows that it is harder to spot distinctions as clear as the previously found one. There still are some patterns which might be captured by non-linear models like k-nearest neighbors, random forests, and more advanced methods like boosted logistic regression and support vector machines, although it is unreasonable to expect high precision. One such pattern is the fact that Normal cases on average have a higher value of sacral_slope than Hernia cases.

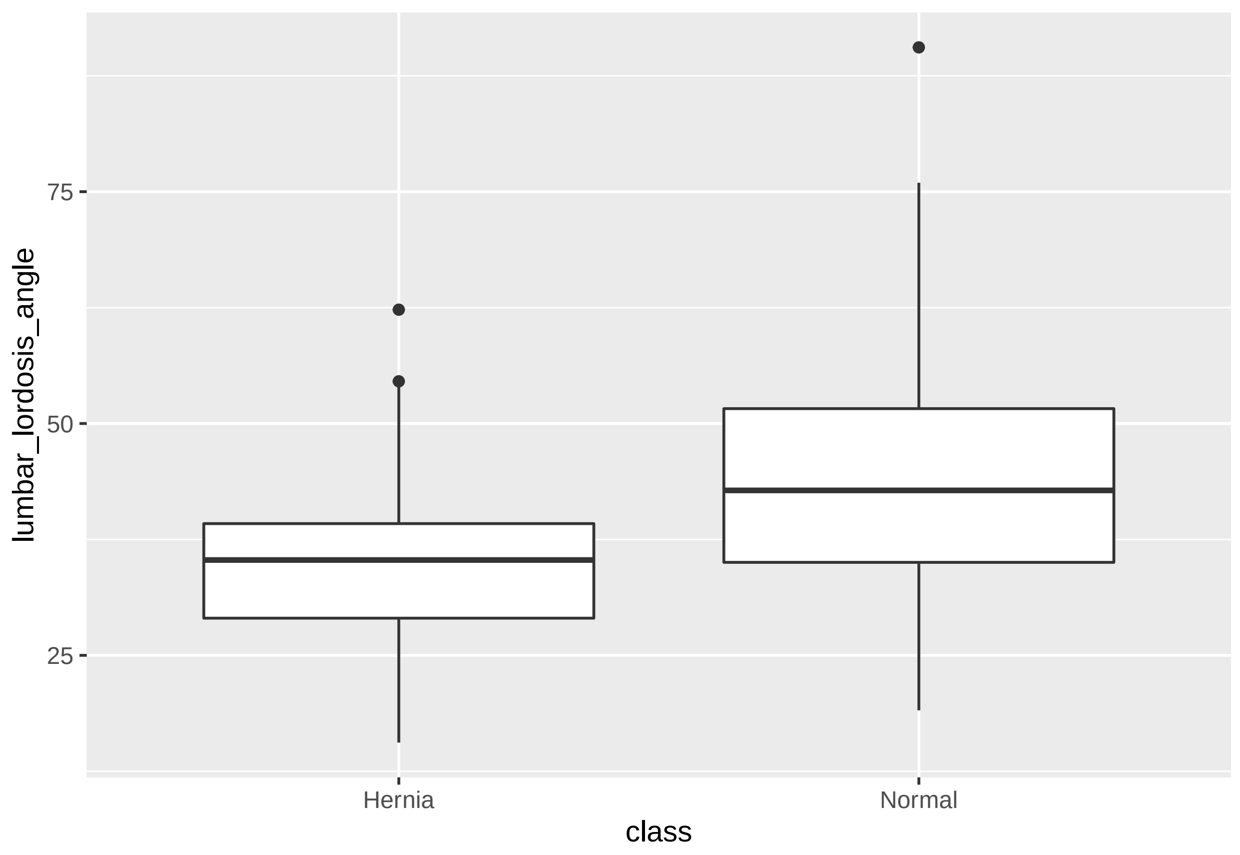
```
train_no_spondy = train %>% filter(class != "Spondylolisthesis")

train_no_spondy %>% filter(degree_spondylolisthesis < 400) %>%
  ggplot(aes(class , pelvic_incidence)) + geom_boxplot()
```



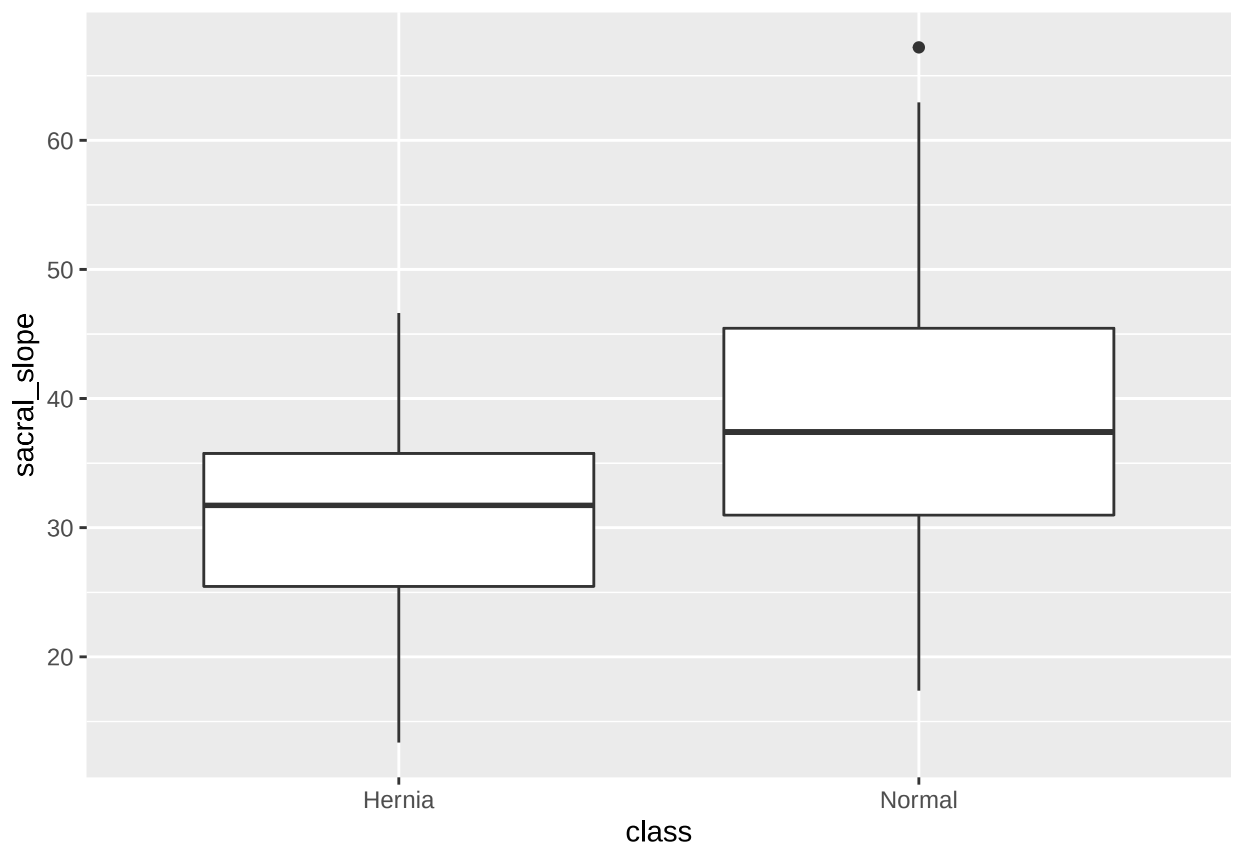
```
train_no_spondy %>% filter(degree_spondylolisthesis < 400) %>%
  ggplot(aes(class , pelvic_tilt)) + geom_boxplot()
```



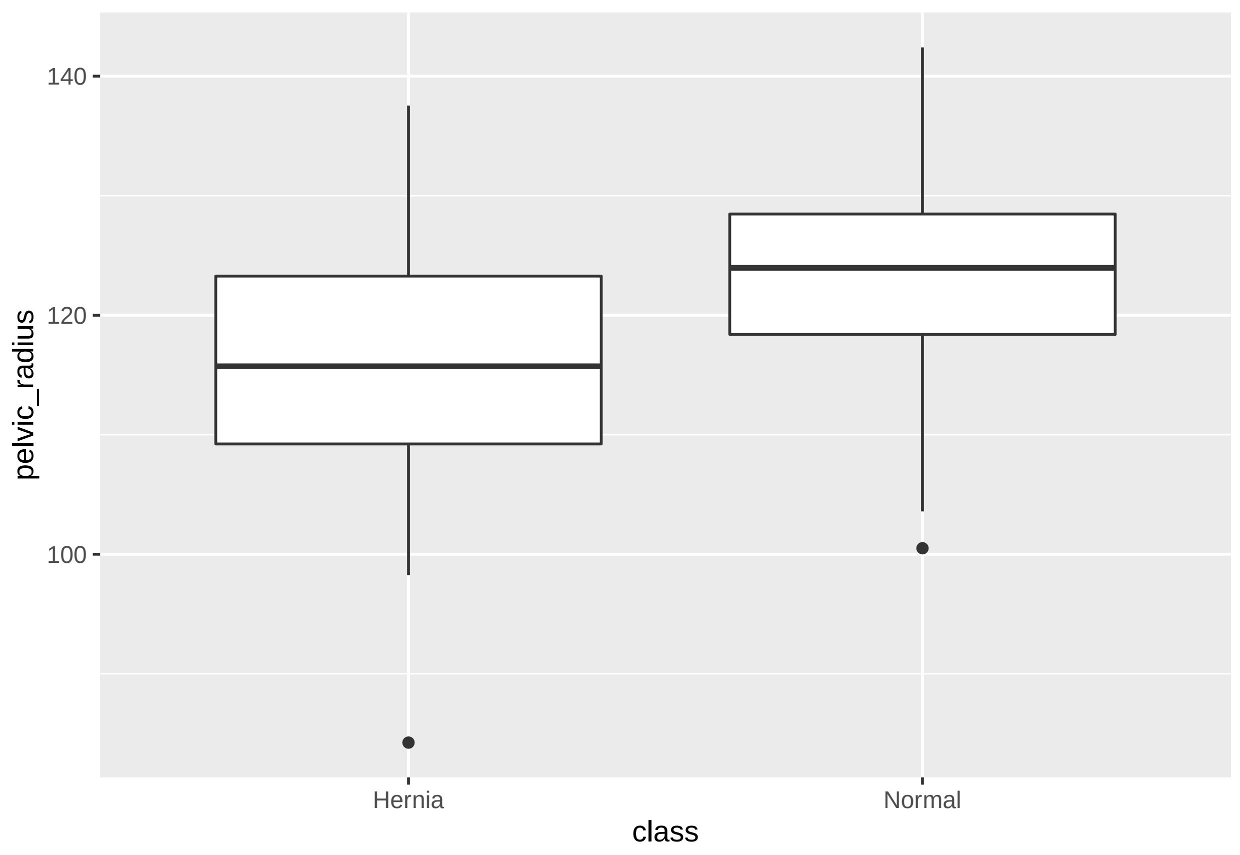
```
train_no_spondy %>% filter(degree_spondylolisthesis < 400) %>%
  ggplot(aes(class , lumbar_lordosis_angle)) + geom_boxplot()
```



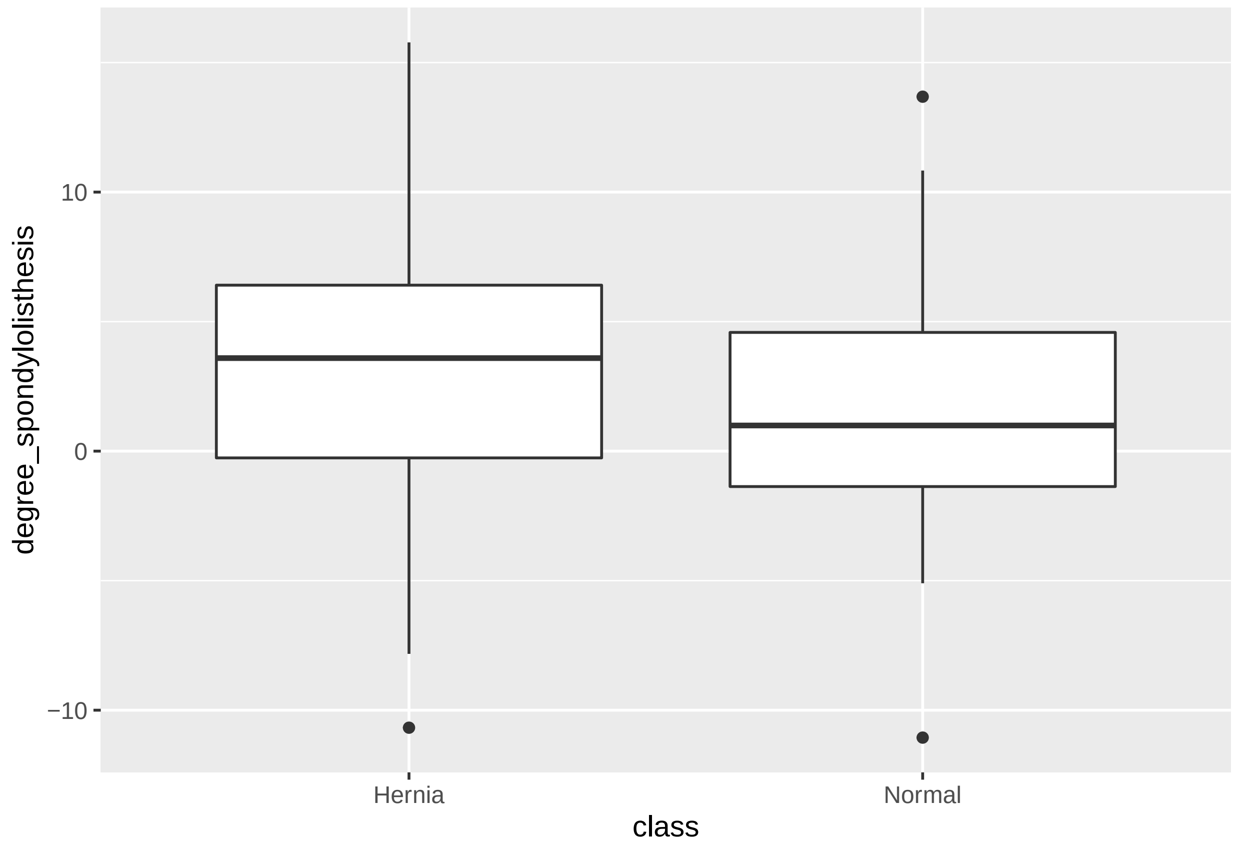
```
train_no_spondy %>% filter(degree_spondylolisthesis < 400) %>%
  ggplot(aes(class , sacral_slope)) + geom_boxplot()
```



```r
train_no_spondy %>% filter(degree_spondylolisthesis < 400) %>%
  ggplot(aes(class , pelvic_radius)) + geom_boxplot()
```



```r
train_no_spondy %>% filter(degree_spondylolisthesis < 400) %>%
  ggplot(aes(class , degree_spondylolisthesis)) + geom_boxplot()
```



Given the high degree of intersectionality between the Normal and Hernia classes, it should be noted that clustering techniques would be ineffective in this situation, although they might be effective at defining a cluster including the Spondylolisthesis class only.

Also note that a Multilayer Perceptron model was trained on the dataset but it achieved less than 50% accuracy. MLP classifiers are useful when trained on higher-dimensional data, like the MNIST dataset.

Results

We now fit the mentioned machine learning models to the data. At this point it should be noted that, given the relatively low amount of data available, the accuracy of the models has some random variability depending on how we randomly pick the training and test sets. We will later perform a Monte Carlo simulation to better estimate the accuracy of these models. Due to this, my commentary on the model's performance might not be consistent with the output displayed after knitting this report's corresponding R markdown file.

```r
#fit a classification decision tree
fit_CART = caret::train(class ~ ., data = train, method = "rpart")

#fit a knn model
fit_knn = caret::train(class ~ ., data = train, method = "knn",
                       tuneGrid = data.frame(k = seq(3,33,2)))

#fit a random forest model
fit_rf = caret::train(class ~ ., data = train, method = "rf")

#fit boosted logistic regression model, a logistic variant of AdaBoost
fit_logitBoost = caret::train(class ~ ., data = train, method = "LogitBoost")
```

```r
#fit support vector machine model
fit_svm = caret::train(class ~ ., data = train, method = "svmLinear",
                       tuneGrid = data.frame(C = seq(0.01,10,1)))
```

As expected, using a classification tree model we achieve high sensitivity and specificity to Spondylolisthesis cases. On the other hand, the model suffers from low sensitivity to Hernia and Normal cases which negatively impacts the overall accuracy.

```r
#predict using classification tree
y_CART = predict(fit_CART, newdata = test)
caret::confusionMatrix(y_CART, reference = test$class)
```

```
## Confusion Matrix and Statistics
##
##                    Reference
## Prediction          Hernia Normal Spondylolisthesis
##   Hernia                 9      7                 2
##   Normal                 9     20                 1
##   Spondylolisthesis      0      3                42
##
## Overall Statistics
##
##                Accuracy : 0.7634
##                  95% CI : (0.664, 0.8454)
##     No Information Rate : 0.4839
##     P-Value [Acc > NIR] : 3.327e-08
##
##                   Kappa : 0.6211
##
##  Mcnemar's Test P-Value : 0.3547
##
## Statistics by Class:
##
##                      Class: Hernia Class: Normal Class: Spondylolisthesis
## Sensitivity                0.50000        0.6667                   0.9333
## Specificity                0.88000        0.8413                   0.9375
## Pos Pred Value             0.50000        0.6667                   0.9333
## Neg Pred Value             0.88000        0.8413                   0.9375
## Prevalence                 0.19355        0.3226                   0.4839
## Detection Rate             0.09677        0.2151                   0.4516
## Detection Prevalence       0.19355        0.3226                   0.4839
## Balanced Accuracy          0.69000        0.7540                   0.9354
```

It is interesting to display the variable importance determined by the model. As shown, degree_spondylolisthesis is the most important variable for predicting classes.

```r
#display variable importance
fit_CART$finalModel$variable.importance
```

```
## degree_spondylolisthesis         pelvic_incidence    lumbar_lordosis_angle
##                 79.29551                 47.37649                 45.42846
##             sacral_slope             pelvic_radius              pelvic_tilt
##                 41.00662                 27.31034                 24.28533
```

Using a random forest model improves performance. Sensitivity to Hernia and Normal cases remains low, while Spondylolisthesis is predicted with good accuracy.

```r
#predict using random forest
y_rf = predict(fit_rf, newdata = test)
caret::confusionMatrix(y_rf, reference = test$class)
```

```
## Confusion Matrix and Statistics
##
##                    Reference
## Prediction          Hernia Normal Spondylolisthesis
##   Hernia                12      3                 0
##   Normal                 6     23                 2
##   Spondylolisthesis      0      4                43
##
## Overall Statistics
##
##                Accuracy : 0.8387
##                  95% CI : (0.748, 0.9068)
##     No Information Rate : 0.4839
##     P-Value [Acc > NIR] : 1.193e-12
##
##                   Kappa : 0.7385
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Hernia Class: Normal Class: Spondylolisthesis
## Sensitivity                 0.6667        0.7667                   0.9556
## Specificity                 0.9600        0.8730                   0.9167
## Pos Pred Value              0.8000        0.7419                   0.9149
## Neg Pred Value              0.9231        0.8871                   0.9565
## Prevalence                  0.1935        0.3226                   0.4839
## Detection Rate              0.1290        0.2473                   0.4624
## Detection Prevalence        0.1613        0.3333                   0.5054
## Balanced Accuracy           0.8133        0.8198                   0.9361
```

Again, degree_spondylolisthesis is the most important variable.

```r
#display variable importance
fit_rf$finalModel$importance
```

```
##                          MeanDecreaseGini
## pelvic_incidence                16.468401
## pelvic_tilt                      9.845646
## lumbar_lordosis_angle           18.028108
## sacral_slope                    16.674129
## pelvic_radius                   15.523061
## degree_spondylolisthesis        58.056581
```

The K-Nearest Neighbors algorithm achieves results similar to the random forest model. Cross-validation is used to tune the K parameter.

```r
#predict using knn
y_knn = predict(fit_knn, newdata = test)
caret::confusionMatrix(y_knn, reference = test$class)
```

```
## Confusion Matrix and Statistics
```

```
##
##                    Reference
## Prediction          Hernia Normal Spondylolisthesis
##   Hernia                 8      1                 1
##   Normal                10     27                 4
##   Spondylolisthesis      0      2                40
##
## Overall Statistics
##
##                Accuracy : 0.8065
##                  95% CI : (0.7115, 0.8811)
##     No Information Rate : 0.4839
##     P-Value [Acc > NIR] : 1.424e-10
##
##                   Kappa : 0.687
##
##  Mcnemar's Test P-Value : 0.02889
##
## Statistics by Class:
##
##                      Class: Hernia Class: Normal Class: Spondylolisthesis
## Sensitivity                0.44444        0.9000                   0.8889
## Specificity                0.97333        0.7778                   0.9583
## Pos Pred Value             0.80000        0.6585                   0.9524
## Neg Pred Value             0.87952        0.9423                   0.9020
## Prevalence                 0.19355        0.3226                   0.4839
## Detection Rate             0.08602        0.2903                   0.4301
## Detection Prevalence       0.10753        0.4409                   0.4516
## Balanced Accuracy          0.70889        0.8389                   0.9236
```

This is the cross-validated value picked for K.

```
#display best k
fit_knn$bestTune
```

```
##     k
## 12 25
```

The Support Vector Machine model provides slightly improved performance. Accuracy for the Spondylolisthesis class remains high. Cross-validation is used to tune the cost parameter C.

```
#predict using support vector machine
y_svm = predict(fit_svm, newdata = test)
caret::confusionMatrix(y_svm, reference = test$class)
```

```
## Confusion Matrix and Statistics
##
##                    Reference
## Prediction          Hernia Normal Spondylolisthesis
##   Hernia                12      1                 1
##   Normal                 6     26                 2
##   Spondylolisthesis      0      3                42
##
## Overall Statistics
##
##                Accuracy : 0.8602
##                  95% CI : (0.7728, 0.9234)
##     No Information Rate : 0.4839
##     P-Value [Acc > NIR] : 3.374e-14
##
##                   Kappa : 0.7741
##
##  Mcnemar's Test P-Value : 0.1893
##
## Statistics by Class:
##
##                      Class: Hernia Class: Normal Class: Spondylolisthesis
## Sensitivity                 0.6667        0.8667                   0.9333
## Specificity                 0.9733        0.8730                   0.9375
## Pos Pred Value              0.8571        0.7647                   0.9333
## Neg Pred Value              0.9241        0.9322                   0.9375
## Prevalence                  0.1935        0.3226                   0.4839
## Detection Rate              0.1290        0.2796                   0.4516
## Detection Prevalence        0.1505        0.3656                   0.4839
## Balanced Accuracy           0.8200        0.8698                   0.9354
```

This is the cross-validated value picked for C.

```
#display best C
fit_svm$bestTune
```

```
##      C
## 2 1.01
```

The Boosted Logistic Regression (LogitBoost) meta-learning model provides a performance similar to the SVM model.

```
#predict using logitBoost
y_logit = predict(fit_logitBoost, newdata = test)
caret::confusionMatrix(y_logit, reference = test$class)
```

```
## Confusion Matrix and Statistics
##
##                     Reference
## Prediction          Hernia Normal Spondylolisthesis
##   Hernia                 7      2                 0
##   Normal                 8     21                 1
##   Spondylolisthesis      0      4                43
##
## Overall Statistics
##
##                Accuracy : 0.8256
##                  95% CI : (0.7287, 0.899)
##     No Information Rate : 0.5116
##     P-Value [Acc > NIR] : 1.287e-09
##
##                   Kappa : 0.7057
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Hernia Class: Normal Class: Spondylolisthesis
## Sensitivity                 0.4667        0.7778                   0.9773
## Specificity                    0.9718        0.8475                   0.9048
## Pos Pred Value                 0.7778        0.7000                   0.9149
## Neg Pred Value                 0.8961        0.8929                   0.9744
## Prevalence                     0.1744        0.3140                   0.5116
## Detection Rate                 0.0814        0.2442                   0.5000
## Detection Prevalence           0.1047        0.3488                   0.5465
## Balanced Accuracy              0.7192        0.8126                   0.9410
```

As explained before, these accuracy estimates are unstable because the available sample size is low.

To get a better estimate of the accuracy of the models, we will be running a Monte Carlo simulation replicating 100 times the model fitting and prediction. This is a sort of cross-validation, in that we are training and validating the models on random splits of the original dataset. The following function takes in input the dataset and outputs the overall accuracy of the models tested.

```
#this function returns a vector containing the accuracy of the machine learning algorithms
accuracies = function(dataset){
  index = createDataPartition(dataset$class, p = 0.7, list = FALSE)

  train = dataset[index,]
  test = dataset[-index,]

  fit_logitBoost = caret::train(class ~ ., data = train, method = "LogitBoost")
  fit_CART = caret::train(class ~ ., data = train, method = "rpart")
  fit_knn = caret::train(class ~ ., data = train, method = "knn", tuneGrid = data.frame(k = seq(3,33,2))
  fit_rf = caret::train(class ~ ., data = train, method = "rf")
  fit_svm = caret::train(class ~ ., data = train, method = "svmLinear")

  y_logit = predict(fit_logitBoost, newdata = test)
  y_CART = predict(fit_CART, newdata = test)
  y_knn = predict(fit_knn, newdata = test)
  y_rf = predict(fit_rf, newdata = test)
  y_svm = predict(fit_svm, newdata = test)
  c(
    caret::confusionMatrix(y_logit, reference = test$class)$overall[[1]],
    caret::confusionMatrix(y_CART, reference = test$class)$overall[[1]],
    caret::confusionMatrix(y_knn, reference = test$class)$overall[[1]],
    caret::confusionMatrix(y_rf, reference = test$class)$overall[[1]],
    caret::confusionMatrix(y_svm, reference = test$class)$overall[[1]]
  )
}
```

We now run the Monte Carlo simulation. Please note that this takes a while to run.

```
aggregated_acc = replicate(100, accuracies(dataset))
```

After running the simulation, we compute and display the mean and standard deviation of the models' accuracy.

```
aggregated_acc = t(aggregated_acc)
colnames(aggregated_acc) = c("LogitBoost", "Classification Tree",
                             "K-NN", "Random Forest", "Support Vector Machine")
#report mean accuracy for each model
mean_acc = t(as.matrix(colMeans(aggregated_acc)))
rownames(mean_acc) = c("Mean accuracy")

#report standard deviation of accuracy for each model
```

```
acc_sd = t(as.matrix(c(sd(aggregated_acc[,1]), sd(aggregated_acc[,2]),
                       sd(aggregated_acc[,3]), sd(aggregated_acc[,4]),
                       sd(aggregated_acc[,5]))))
colnames(acc_sd) = c("LogitBoost", "Classification Tree", "K-NN",
                     "Random Forest", "Support Vector Machine")
rownames(acc_sd) = c("Standard deviation of accuracy")

mean_acc
```

```
##               LogitBoost Classification Tree      K-NN Random Forest
## Mean accuracy  0.8602782            0.791828 0.8337634      0.837957
##               Support Vector Machine
## Mean accuracy              0.8627957
```

```
acc_sd
```

```
##                                LogitBoost Classification Tree       K-NN
## Standard deviation of accuracy 0.02790907          0.03660064 0.03058245
##                                Random Forest Support Vector Machine
## Standard deviation of accuracy    0.02916871             0.03303153
```

The means show that the SVM model performs best, with LogitBoost closely following. The random forest model has a performance slightly better than k-nearest neighbors, but somewhat far from LogitBoost and SVM. The classification tree model performs poorly compared to the others.

As shown, the standard deviation of these estimates is around 0.03 for all the models, therefore our accuracy is relatively unstable. This could be fixed by having more data available.

Conclusion

We have explored the dataset and identified models capable of predicting the patient's condition given their biomechanical features, while discarding other models which would be ineffective. Given the structure of the data, distinguishing between the Normal and Hernia classes is much harder than detecting the Spondylolisthesis class. The machine learning methods we've used achieve reasonable accuracy, but in medical applications a much higher accuracy is needed for the model to be effective and safely usable because of Bayes' Theorem. The accuracy of our models is unstable due to the low amount of data available and it would be very useful to have more data on Hernia cases, which is the least prevalent class of the dataset.

In the future I aim to continue working on this dataset.

First of all, merging the Normal and Hernia classes and training the same models to only predict Spondylolisthesis should result in a machine learning algorithm with around 95% accuracy, which is much more suited for use in medical applications.

Second of all, to improve overall accuracy, building an ensemble algorithm might be useful, as well as including additional models in such ensemble.