Capstone Orthopedic

Til Stein

13/05/2020

1. Introduction

In this analysis, I aim to find abnormalities in the spine from patient data such as the sacral slope or the pelvic tilt. I will start with single variable methods, before I move on to multivariate and ensemble analysis.

1.1 Data Source

The data I am using stems from the website kaggle. Here is the link: https://www.kaggle.com/uciml/biomechanical-features-of-orthopedic-patients/data?select=column_2C_weka.csv

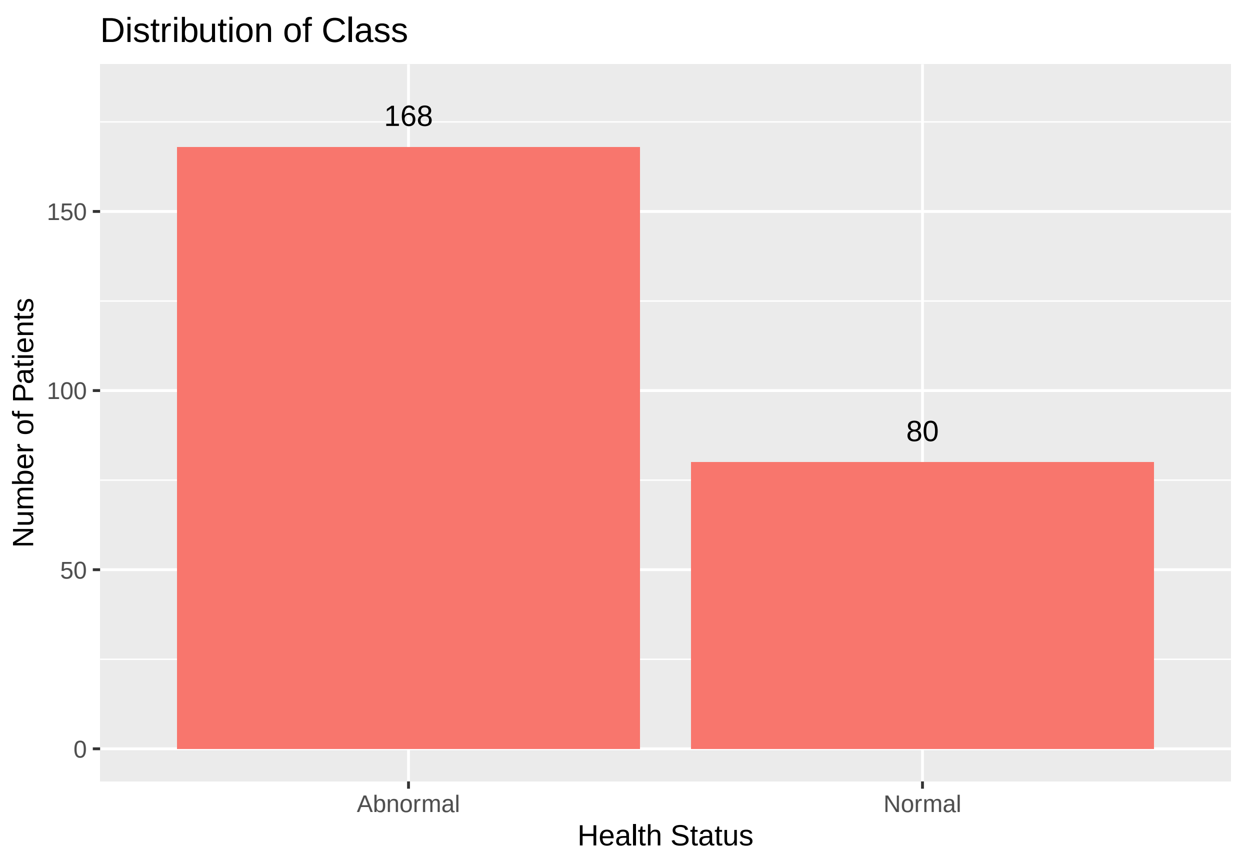
They originally downloaded it from the UCI ML repository. Credits go to: Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science

1.2 Data Splitting

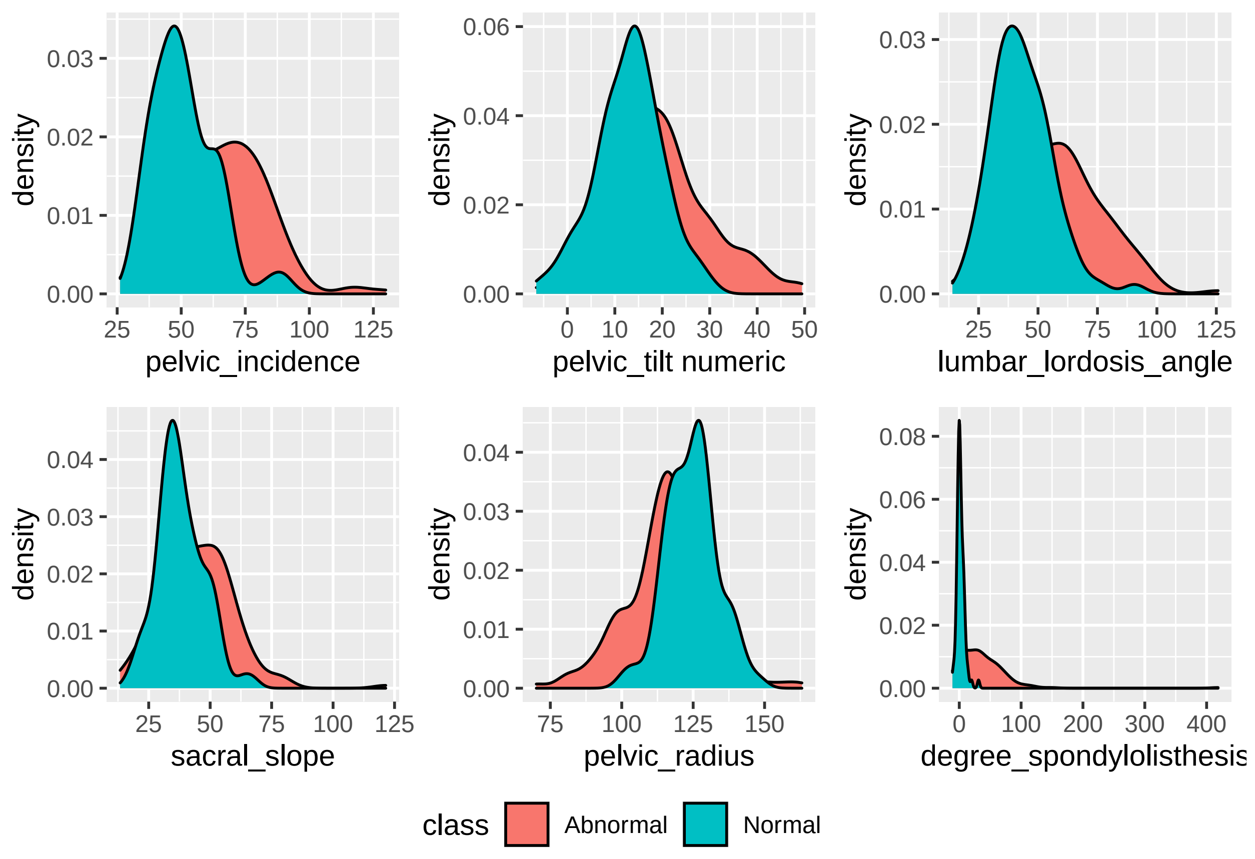
For this analysis, I randomly partitioned the data into a training set and a test set. I will use the training set to build my algorith and the test set only at the end, to test it. Therefore, all analysis of the data refers to the training set. Of the 310 patients, 248 (80%) are in the training set and 62 (20%) are in the test set. This is a very common way of splitting the data, as it gives me a good amount of entries to train my algorithm, as well as sufficient patients to test my algorithm. A split of 90/10 would have left me with only 31 patients to verify my finding, which could lead to unstable end results.

1.3 Data Insight

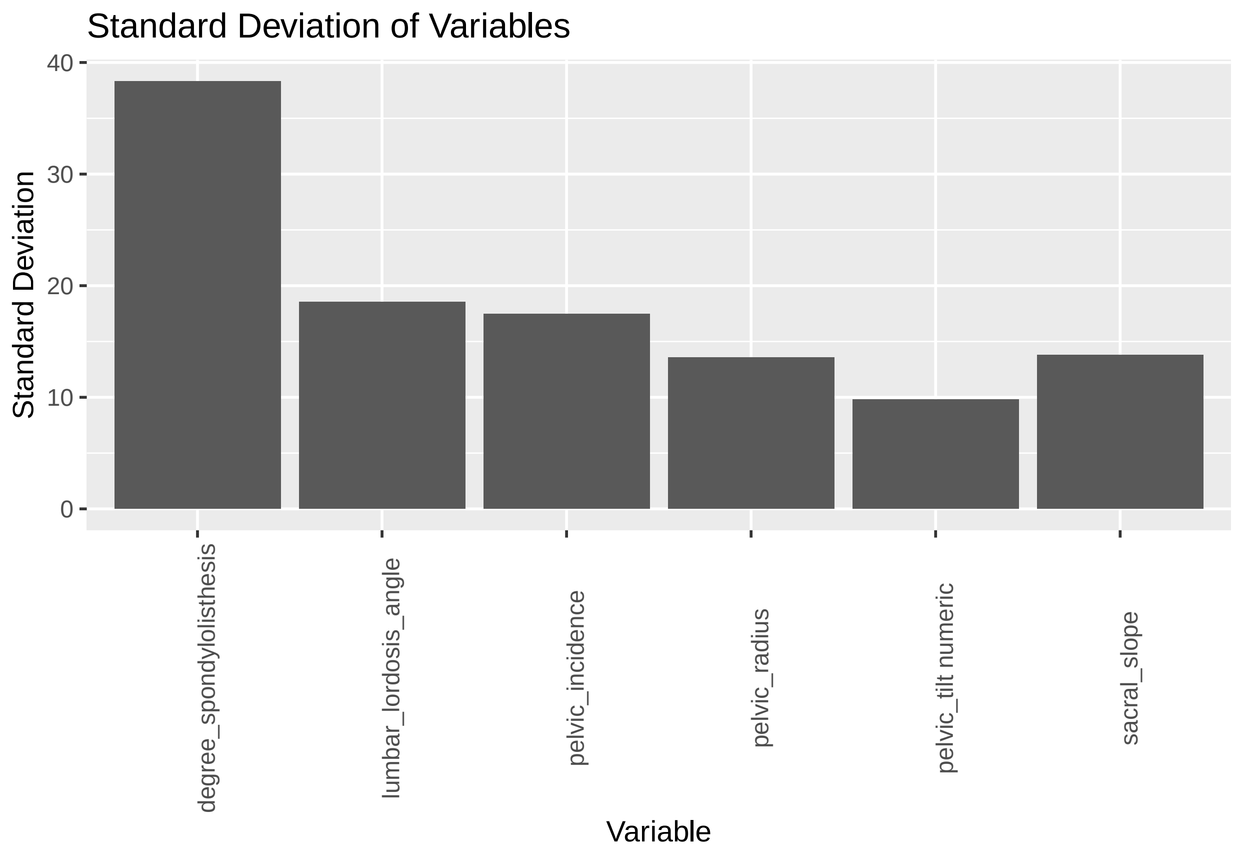
1.3.1 Prediction Variable In the data, orthopedic patients are classified as "normal" or "abnormal". Patients that are classified as "abnormal" either have a disk hernia or a spondylolisthesis. Patients that are classified as "Normal", have a healthy spine. Within the data are body measurements of the patients. These body measurements are: pelvic incidence, pelvic tilt, lumbar lordosis angle, sacral slope, pelvic radius and degree of spondylolisthesis. In the first plot we can see that there are more patients with abnormalities (168) than healthy patients (80). This also expalins, why I haven't divided the set into a training, testing and validation set.



1.3.2 Predictors



Above, we can see the distribution of our predictors. Furthermore, the classes are encoded by colour. We can see that most variables are somewhat normally distributed. However, the degree of spondylolisthesis predictor has most of its data at around 0, with extreme offshoots.



If we look at the variance within our predictors, we can see that degree of spondylolisthesis has the highest standard deviation. Nevertheless, all other variables also do have a considerable variance. For this reason, it does not make sense to cut remove a variable from our algorithm. Especially due to the fact that our data set is relatively small anyways and we do not need to free up computing power.

2 Analysis

In this analysis we will always look at three factors. The accuracy, sensitivity and specificity of a model. The accuracy describes how often our model correctly predicts the class (health status) of a patient. The sensitivity describes how many people that are not healthy have been detected, while the specificity describes how many healthy people have correctly been predicted as such. In this example, it is most important than people with abnormalities (sensitivity) are correctly identified as such. However overall accuracy is also very important.

2.1 Single Variable

2.1.1 Method First, we want to try predicting the health status of a patient with only one predictor. For this, we will try finding a value, below or above which we predict a patient to be abnormal. Below, I have included exemplary code for the first predictor. If you want to see the code for all predictors, it is provided in the RMarkdown file.

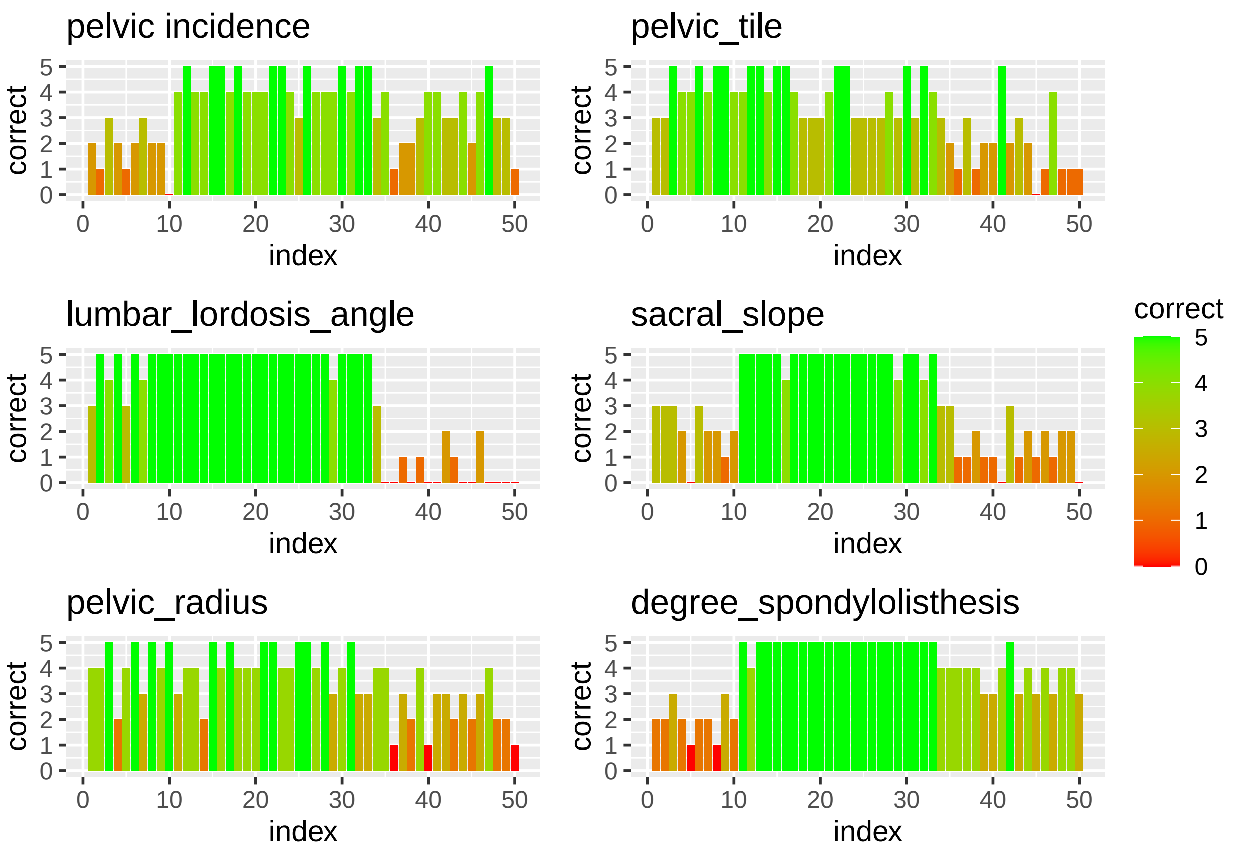
```r
#create a vector of possible threshold values
pi_vector <- c(50:100)
#create a function that tests decision accuracy for value x
pi_func <- function(x) {
  dec_pi <- ifelse(train_set$pelvic_incidence > x,"Abnormal","Normal")
  mean(dec_pi ==train_set$class)
  }
#apply our vector of possible values to the function
pi_res <- sapply(pi_vector,pi_func)
#save x value that maximizes accuracy
pi_max <- pi_vector[which.max(pi_res)]
#calculate the model with maximum accuracy x value
y_hat_pi <- ifelse(train_set$pelvic_incidence > pi_max,"Abnormal","Normal")
#Creating Confusion Matrix
cm <- confusionMatrix(factor(y_hat_pi),factor(train_set$class))
#Storing ensemble accuracy
acc_pi <- cm$overall["Accuracy"]
#Storing Ensemble Sensitivity and Specificity
ss_pi <- cm$byClass[1:2]
```

2.1.2 Results The degree of spondylolisthesis was the most powerful single predictor with an accuracy of close to 80% and a sensitivity of 81%. This could've been expected, if you look at the distribution as well as the variance of the variable. While the lumbar lordosis angle had an accuracy of only about 68%, its sensitivity was at close to 96%.

model	Accuracy	Sensitivity	Specificity
Only Pelvic Incidence	0.6854839	0.7202381	0.6125
Only Pelvic Tilt	0.6733871	0.8095238	0.3875
Only Lumbar Lordosis Angle	0.6774194	0.9583333	0.0875
Only Sacral Slope	0.6411290	0.8035714	0.3000
Only Pelvic Radius	0.7217742	0.8154762	0.5250
Only Degree of Spondylolisthesis	0.7943548	0.8154762	0.7500

From the graphs below, we can see the correct predictions a method made in a region of the data set. For the graphs, the dataset was sliced into lists of 5. Here, we can see how many of those 5 predictions were correct within each list. While the index of the list has no relation to the variable value, every list includes a different group of patients of unique values. If different predictors/methods are good or bad at different parts of the graph, it might be worth combining these predictors/methods.

From the graphs below, we can see that this is the case. For example, the degree_spondylolisthesis variable is very good at predicting the lists in the middle, while pelvic_tie is much better at predicting the lists closer to zero.



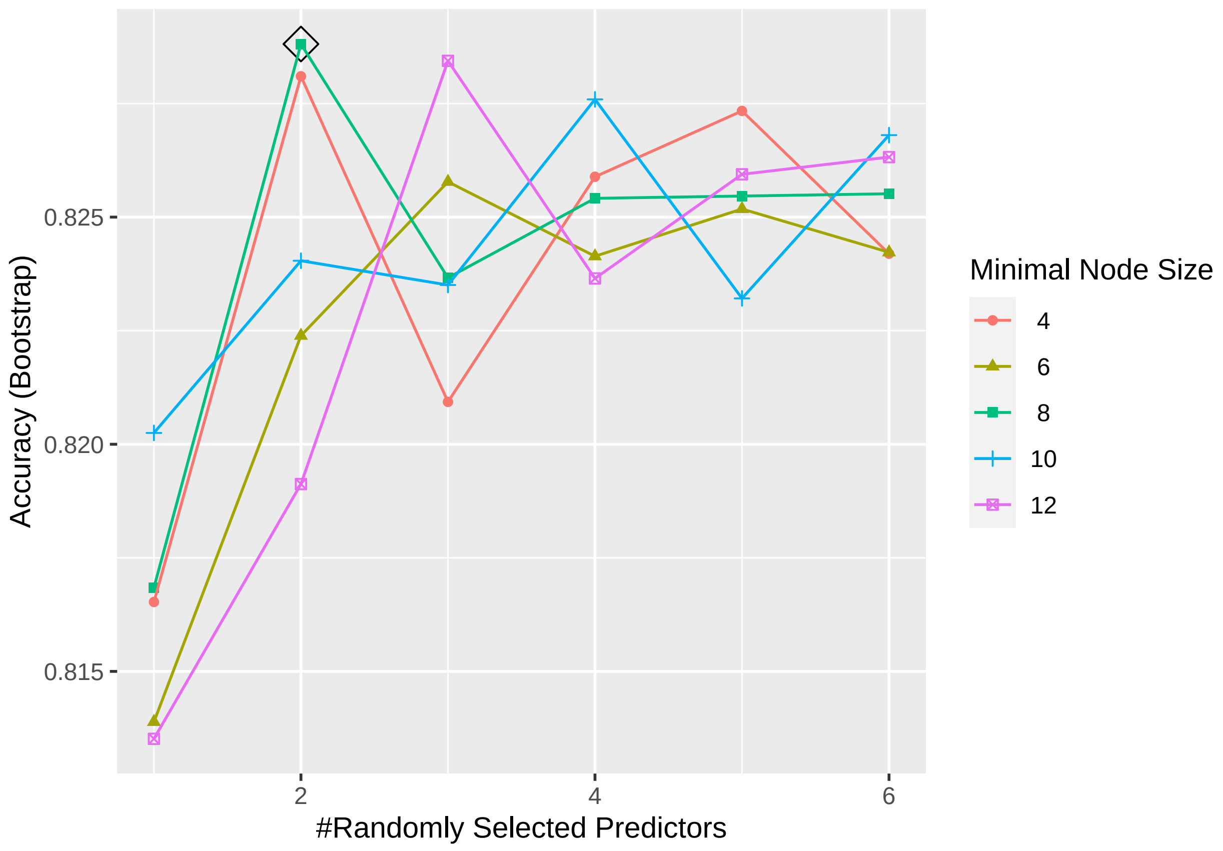
2.2 Multivariable Models

2.2.1 Method Now we will look at multiple predictors. We will use the train function from the caret package to train or models with different methods. We will look at the methods "Random Forest", "knn" and "svmLinear". All three are popular machine learning algorithms. We will start by finding the best tuning parameters for the algorithm, before we run the algorithm with optimised paramters.

Below, you will find an example code from the random forest method. If you want to see the full code, you can find it in the RMarkdown file.

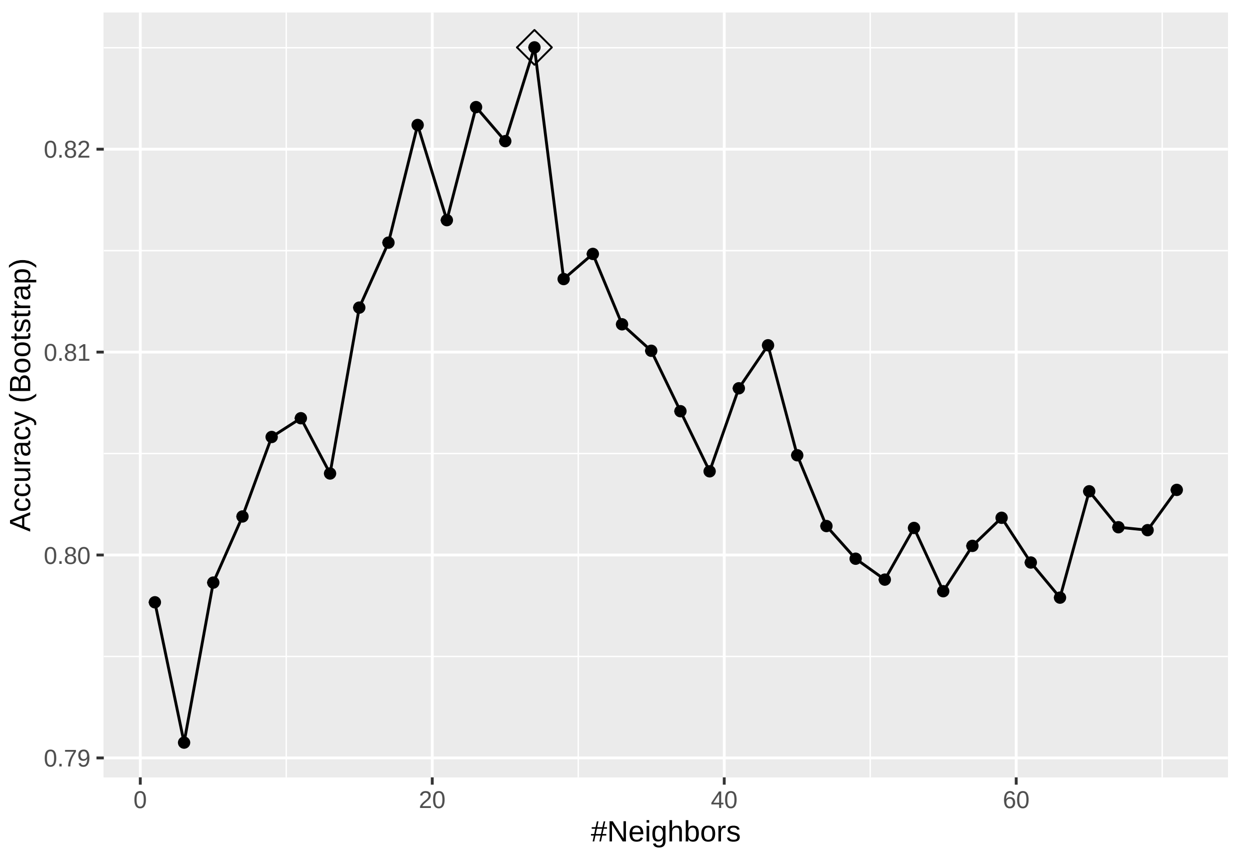
```
#Create a df with tuneGrid options
rf_grid = expand.grid(mtry = seq(1,6,1),splitrule = "extratrees", min.node.size = seq(4,12,2))
#Setting the seed
set.seed(1, sample.kind = "Rounding")
#Find best tuneGrid options for train method
train_rf <- train(class ~ .,
                  method = "ranger",
                  data = train_set,
                 tuneGrid = rf_grid)

#Create a df with best tuneGrid options
rf_values <- data.frame(mtry = c(2), splitrule = c("extratrees"),
                        min.node.size = c(8))
#Setting the seed
set.seed(1, sample.kind = "Rounding")
#Train Model with best variables
rf_fit <- train(class ~ .,
                  method = "ranger",
                  data = train_set,
                 tuneGrid = rf_values)
#Predict y_hat
y_hat_rf <- predict(rf_fit, train_set, type = "raw")
#Create confusion matrix
cm <- confusionMatrix(y_hat_rf,
                 factor(train_set$class))
#Store accuracy
acc_rf <- cm$overall["Accuracy"]
#Store Sensitivity and specificity
ss_rf <- cm$byClass[1:2]
```



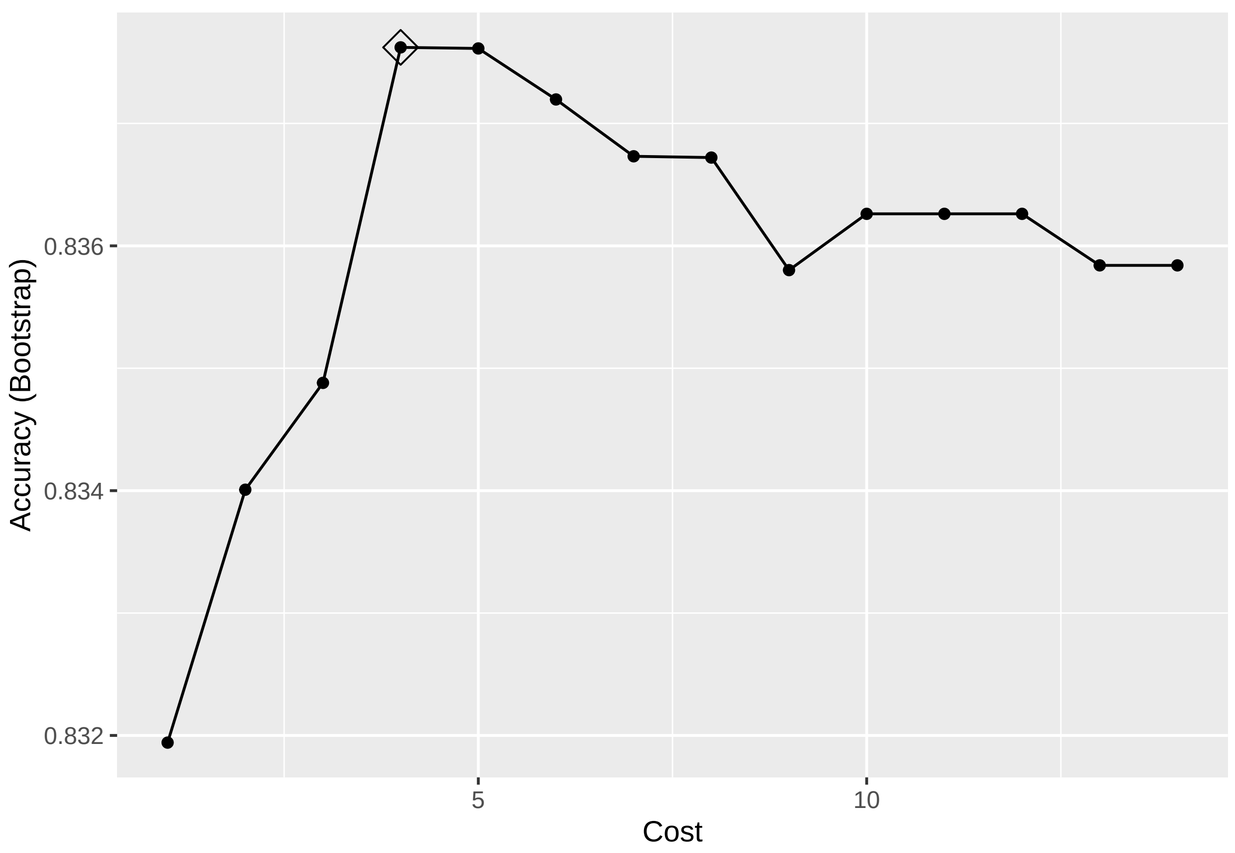
```
##   mtry  splitrule min.node.size
## 8    2 extratrees             8
```

2.2.1.1 Random Forest Ideal parameters are mtry = 2, splitrule = "extratrees" and min.node.size = 8.



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

2.2.1.2 knn Ideal parameter is k = 27.



```
##   C
## 4 4
```

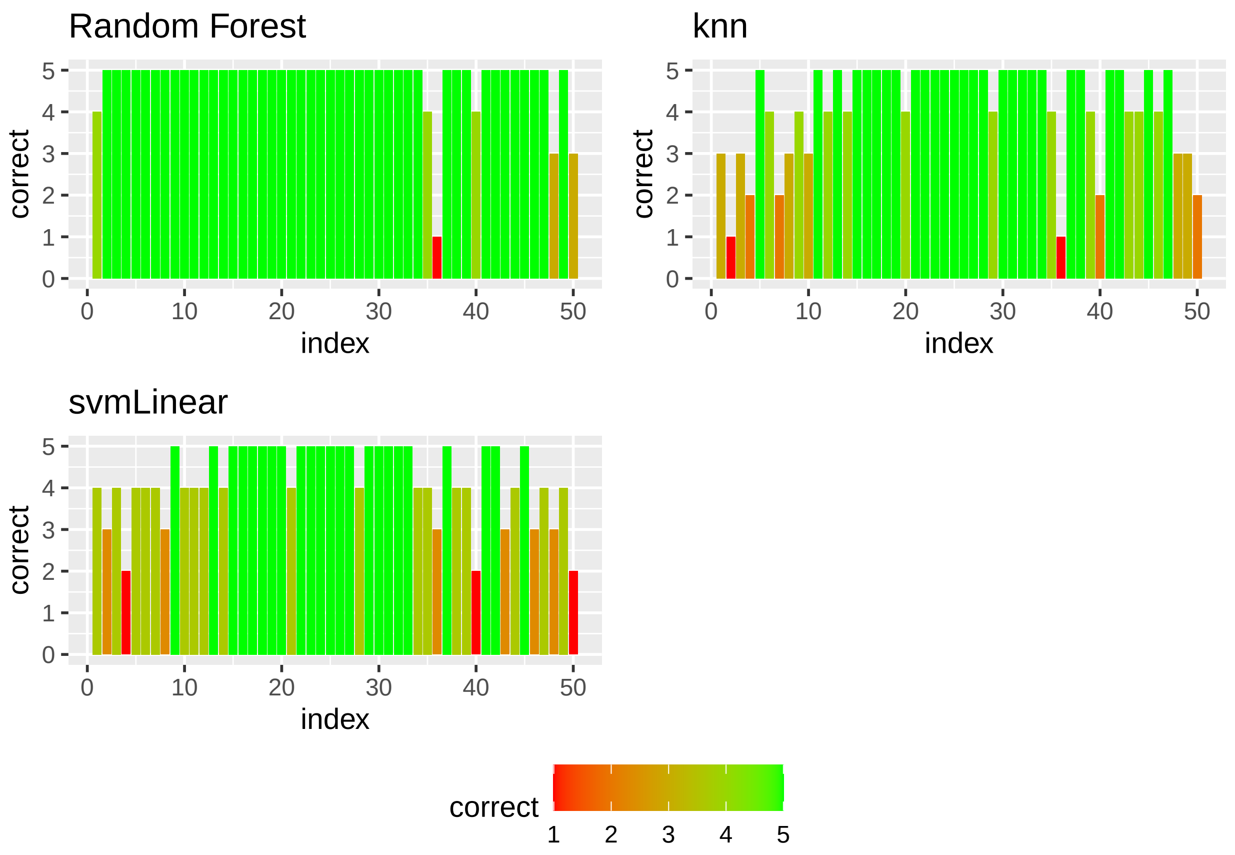
2.2.1.3 svmLinear Ideal parameter is C = 4.

2.2.2 Results

We can see that the Random Forest model has performed the best out of the three with an accuracy of 96% and a sensitivity of 94%, while the specificity is also quiet high at 90%. Nevertheless, the other two models performed better than all of the single variable predictions.

model	Accuracy	Sensitivity	Specificity
Random Forest	0.9637097	0.9940476	0.9000
knn	0.8346774	0.8571429	0.7875
svmLinear	0.8508065	0.8928571	0.7625

Looking at our graphs, we can see that Random Forest perfectly predicted almost all lists. However, we might be able to slightly improve upon that by building an ensemble of the three methods.



2.3 Ensemble

Below, you can find the code for a simple ensemble of the three training models. It always assigns the health status, of which the majority of models are in favour.

```
#Storing the y_hats
y_hat_stored <- cbind(y_hat_knn,y_hat_rf,y_hat_svm)
#Creating average prediction
y_hat_means <- rowMeans(y_hat_stored == "1")
#Creating ensemble y_hat
y_hat_ensemble <- ifelse(y_hat_means > 0.5,"Abnormal","Normal")
#Creating Confusion Matrix
cm <- confusionMatrix(factor(y_hat_ensemble),factor(train_set$class))
#Storing ensemble accuracy
acc_ensemble <- cm$overall["Accuracy"]
#Storing Ensemble Sensitivity and Specificity
ss_ensemble <- cm$byClass[1:2]
```

We can see that the ensemble model could not improve upon the Random Forest model, with an accuracy of 89% and a sensitivity of about 93%.

model	Accuracy	Sensitivity	Specificity
Train Ensemble	0.8870968	0.9285714	0.8

2.4 Choosing our Method

Looking at all our models, we can clearly see that the Random Forest model has performed the best out of all models. The next closest was the ensemble model, with the pelvic incidence proving to be the worst predictor, judging by sensitivity.

model	Accuracy	Sensitivity	Specificity
Only Pelvic Incidence	0.6854839	0.7202381	0.6125
Only Pelvic Tilt	0.6733871	0.8095238	0.3875
Only Lumbar Lordosis Angle	0.6774194	0.9583333	0.0875
Only Sacral Slope	0.6411290	0.8035714	0.3000
Only Pelvic Radius	0.7217742	0.8154762	0.5250
Only Degree of Spondylolisthesis	0.7943548	0.8154762	0.7500
Random Forest	0.9637097	0.9940476	0.9000
knn	0.8346774	0.8571429	0.7875
svmLinear	0.8508065	0.8928571	0.7625
Train Ensemble	0.8870968	0.9285714	0.8000

3 Result

Now, we will run our model on the test set. Below, you find the code for our final model.

```
##1. RF Model

#Create a df with best tuneGrid options
rf_values <- data.frame(mtry = c(2), splitrule = c("extratrees"),
                        min.node.size = c(8))
#Setting the seed
set.seed(1, sample.kind = "Rounding")
#Train Model with best variables
rf_fit <- train(class ~ .,
                 method = "ranger",
                 data = train_set,
                tuneGrid = rf_values)
#Predict y_hat
y_hat_rf_final <- predict(rf_fit, test_set, type = "raw")
#Creating Confusion Matrix
cm <- confusionMatrix(y_hat_rf_final,factor(test_set$class))
#Storing ensemble accuracy
acc_final <- cm$overall["Accuracy"]
#Storing Ensemble Sensitivity and Specificity
ss_final <- cm$byClass[1:2]
```

We can see that our final model performed at an accuracy of 87.1 % and a sensitivity of 95%. This means that out of 20 patients with abnormalities regarding their spine, 19 are correctly identified. With a specificiy of 0.7, there is a considerable amount of "false positives". However, this is okay in terms of this model, as we wanted to make sure that we would identified all patients with a problem.

model	Accuracy	Sensitivity	Specificity
Final Model	0.8709677	0.952381	0.7

4 Conclusion

During this report, I managed to create a prediction method that correctly identifies patients with spine abnormalities 95% of the time. This is a rate with which I am satisfied. However, the specificity is quiet low at 70%. The limiting factor of this analysis quiet clearly was the relatively small dataset, which forced me to decide between the two. Overall the accuracy of 87.1% is also good. In the future, I might try more machine learning models on this data set and implement a better working ensemble. Furthermore, I trained my models aiming for highest accuracy. Alternatively, I could've also trained for better Sensitivity. Nevertheless, it was an amazing first project and I can't wait to build upon it.