

Data Science Development Coursework

Desmond Dumebi Ojei, student no : (1410100)

3/14/2020

1 Data Description

Link : https://www.kaggle.com/uciml/biomechanical-features-of-orthopedic-patients#column_2C_weka.csv

The original dataset was downloaded from UCI ML repository: Lichman, M. (2013). UCI Machine Learning Repository [<http://archive.ics.uci.edu/ml>]. Irvine, CA: University of California, School of Information and Computer Science.

The medical dataset chosen consist of data from Biomechanical features of orthopedic patients, in which each patient is represented by six biomechanical attributes from the dataset, which was derived from the shape and orientation of the pelvis and lumbar spine. Also, based on these features, patients will be classified to be either (Normal or Abnormal).

Features:

pelvic incidence, pelvic.tilt _numeric,lumbar lordosis angle ,sacral slope, pelvic radius ,degree of spondylolisthesis ,Class : Normal or Abnormal

Rows (310)

Columns (7)

(See Appendix for a more detailed data description)

1.1 Objectives

- Visualizing the data in other to find correlation between the variables to check for possibilities of collinearity.
- Using feature box plot to show all the 6 variables as a function to the 2 classes(Normal and Abnormal) in other to spot any outlier observations in the variable as having outliers in your predictors may affect the performance of the model.
- The goal here is to establish a mathematical equation for Class(target) as a function of other 7 attributes, split the data into training and testing and building several Machine Learning models to predict the class of the patients to either be Normal or Abnormal.

2 Data Preprocessing and Exploratory Data Analysis

The steps carried out in this project:

- Checking for working directory
- Setting the correct working directory
- Loading the data using the `read_csv()` function as this is a “.csv” file
- Finding the dimensions and variables names in the dataset
- using the function “`str()`” to check if variables are in the correct structure
- using the function “`summary()`” to show the summary statistics of each variable in the dataset
- Check for missing values in the dataset with the aid of a plotted

2.1 loading packages

#Loading packages

library(tidyverse)

library(corrplot)

library(skimr)

library(caret)

library(dataMaid)

library(naniar)

library(ggplot2)

library(GGally)

library(DataExplorer)

library(xtable)

library(knitr)

library(ggplot2)

library(randomForest)

library(dplyr)

library(rpart)

library(rattle)

(Figure 1) shows that no missing values in the data set. Also, the statistical summary of the data set shows that there is an imbalance in the class distribution. This can result to the higher class to be favoured more during the Machine learning model process (Abnormal = 210, Normal = 100).

```
#Checking for working directory
getwd()

## [1] "C:/Users/hp/Desktop/Data Science/Second Semester/Data Science Develop
ment"

# Setting working directory
setwd('C:\\Users\\hp\\Desktop\\Datasets')

#Loading data
health_data = read.csv("column_2C_weka.csv")

#Snippet of the health_data data set
x <- xtable(head(health_data))
kable(x)
```

pelvic_inci dence	pelvic_tilt.n umeric	lumbar_lordos is_angle	sacral_s lope	pelvic_r adius	degree_spondyl olisthesis	class
63.02782	22.552586	39.60912	40.475	98.6729	-0.254400	Abnor mal
39.05695	10.060992	25.01538	28.995	114.405	4.564259	Abnor mal
68.83202	22.218482	50.09219	46.613	105.985	-3.530317	Abnor mal
69.29701	24.652878	44.31124	44.644	101.868	11.211523	Abnor mal
49.71286	9.652075	28.31741	40.060	108.168	7.918501	Abnor mal
40.25020	13.921907	25.12495	26.328	130.327	2.230652	Abnor mal

```
#Dimension of data
dim(health_data)

## [1] 310 7

#Ensuring data is in the right structure
str(health_data)

## 'data.frame': 310 obs. of 7 variables:
## $ pelvic_incidence : num 63 39.1 68.8 69.3 49.7 ...
## $ pelvic_tilt.numeric : 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                  : Factor w/ 2 levels "Abnormal","Normal": 1 1 1
1 1 1 1 1 1 1 ...

# Summary Statistics on each column
summary(health_data)

## pelvic_incidence pelvic_tilt.numeric 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      Abnormal:210
## 1st Qu.:110.71   1st Qu.: 1.604      Normal :100
## Median :118.27   Median : 11.768
## Mean   :117.92   Mean   : 26.297
## 3rd Qu.:125.47   3rd Qu.: 41.287
## Max.   :163.07   Max.   :418.543

#Plot to visually represent missing values
gg_miss_var(health_data)+ labs(title = "Original data set (No missing values)
")
```

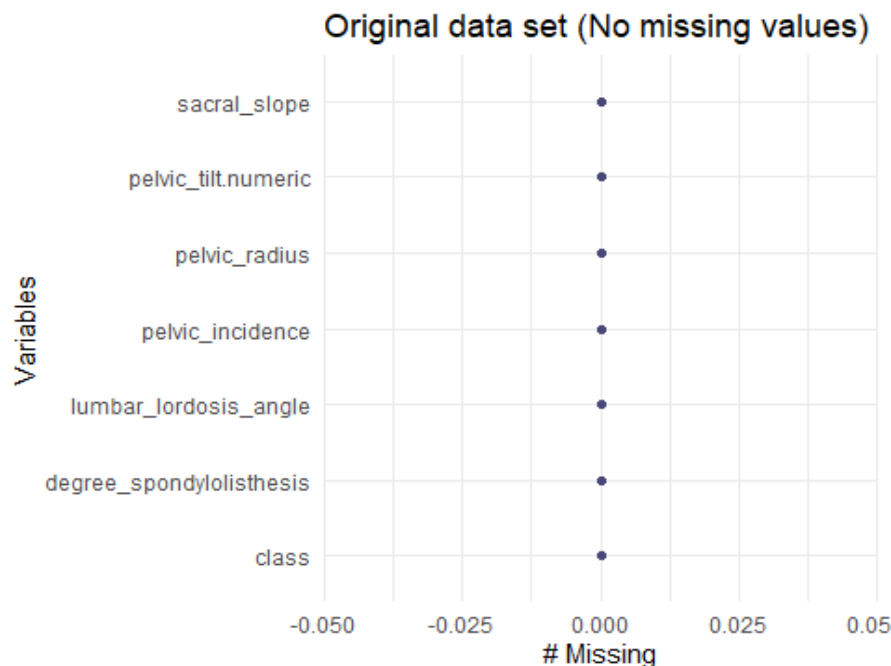


Figure 1: Number of Missing Values

2.2 Graphically creating plot to view distribution on dataset

The dataset is grouped by the class and as seen below, the mean value distribution for the Abnormal class is higher when compared side by side with the Normal class for most variables. However, the "pelvic_radius variable differs as this has a higher Normal class mean distribution than the Abnormal class.

```
# Exploratory Data Analysis

# Summary Statistics on each column
health_data %>%
  dplyr::group_by(class)%>%
  skim()
```

Data summary

Name	Piped data
Number of rows	310
Number of columns	7

















Column type frequency:

numeric	6
---------	---

Group variables	class
-----------------	-------

Variable type: numeric

skim_variable	class	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
pelvic_incidence	Abnormal	0	1	64.69	17.66	26.15	50.10	65.27	77.59	129.83	■■■
pelvic_incidence	Normal	0	1	51.69	12.37	30.74	42.82	50.12	61.47	89.83	■■■
pelvic_tilt.numeric	Abnormal	0	1	19.79	10.52	-6.55	13.05	18.80	24.82	49.43	■■■
pelvic_tilt.numeric	Normal	0	1	12.82	6.78	-5.85	8.80	13.48	16.79	29.89	■■■
lumbar_lordosis_angle	Abnormal	0	1	55.93	19.67	14.00	41.12	56.15	68.10	125.74	■■■

lumbar_lordo	Nor	0	1	43.	12.	19.	35.	42.	51.	90.	
sis_angle	mal			54	36	07	00	64	60	56	
sacral_slope	Abno	0	1	44.	14.	13.	34.	44.	55.	121	
	rma			90	52	37	38	64	15	.43	
sacral_slope	Nor	0	1	38.	9.6	17.	32.	37.	44.	67.	
	mal			86	2	39	34	06	61	20	
pelvic_radius	Abno	0	1	115	14.	70.	107	115	123	163	
	rma			.08	09	08	.31	.65	.13	.07	
pelvic_radius	Nor	0	1	123	9.0	100	118	123	129	147	
	mal			.89	1	.50	.18	.87	.04	.89	
degree_spond	Abno	0	1	37.	40.	-	7.2	31.	55.	418	
ylolisthesis	rma			78	70	10.	6	95	37	.54	
						68					
degree_spond	Nor	0	1	2.1	6.3	-	-	1.1	4.9	31.	
ylolisthesis	mal			9	1	11.	1.5	5	7	17	
						06	1				

2.2.1 Histogram and Density plot

As seen from the (Figure 2), notice that the health data variables contain a tail of data to the right ,therefore the distribution is said to be skewed to the right (positively skewed) except for pelvic_radius which seems to have a normal distribution as it is represented by a bell curve shape.The poitibvely skewed variables will be have the following characteristics (mean > median > mode).

```
#using the library nanair()
#Density and histogram plot
par(mfrow=c(2, 2))
colnames <- dimnames(health_data)[[2]]
for (i in 1:6) {
  hist(health_data[,i], main=colnames[i], probability=TRUE, col="blue", border="white")
  d <- density(health_data[,i])
  lines(d, col="red")
}
```

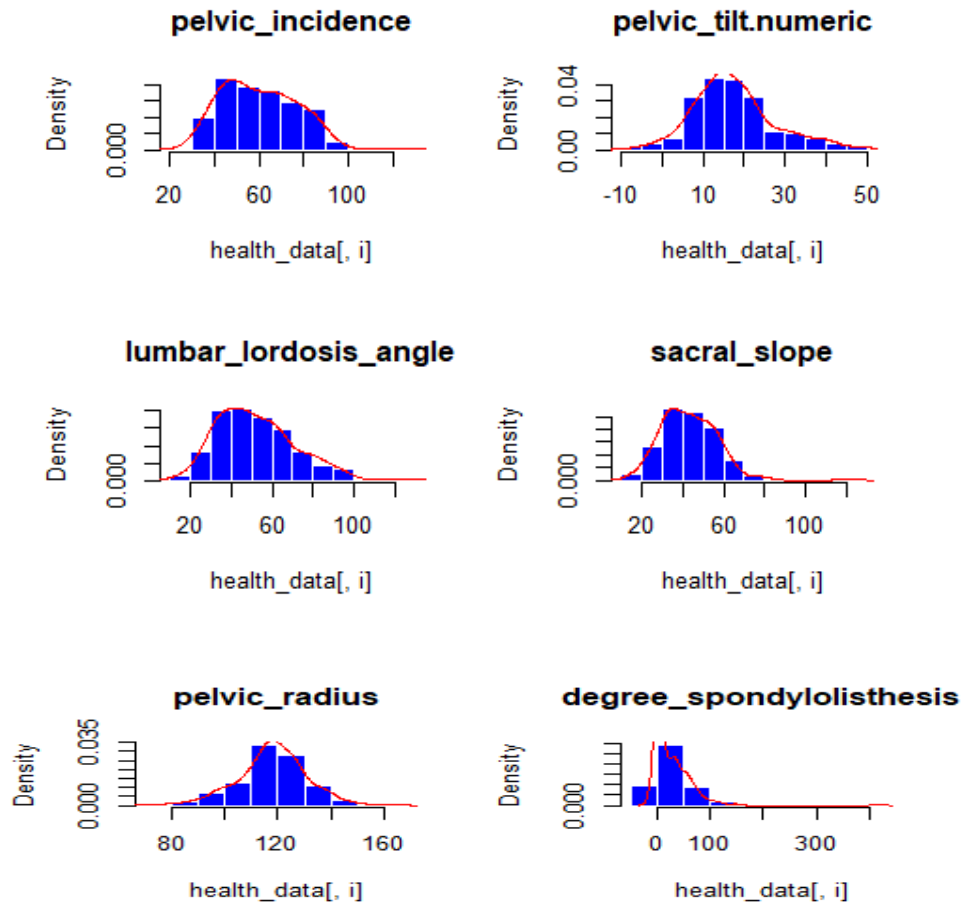


Figure 2: Density and Histogram Plot

2.2.2 Class Distribution

The (Figure 3) Class Distribution shows the number of distribution for the 2 classes (Normal and Abnormal). Also, this data is imbalanced which means that the conventional model evaluation techniques will not work efficiently to measure the model performance. This is as a result of the model being biased to favour the class with the highest percentage. In this case the model will tend to favour the Abnormal class more than the Normal class.

```
#Visualizing the class distribution on the dataset using the barplot() and table () function
barplot(table(health_data$class), main = "Class distribution")
```

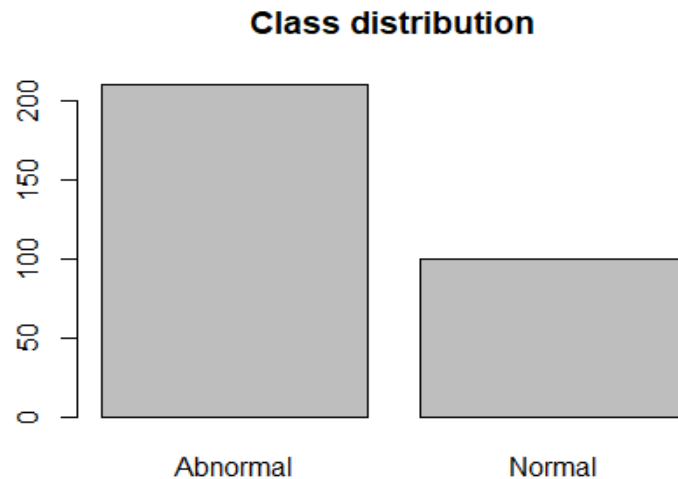


Figure 3: Class Distribution

```
table(health_data$class)

##
## Abnormal    Normal
##      210      100

#The functions prop.table() and table () is used to show the percentage of the
#class distribution.
#As see, the Abnormal distribution is approximately 68% of the dataset and the
#Normal distribution is 32% of the dataset
prop.table(table(health_data$class))

##
## Abnormal    Normal
## 0.6774194 0.3225806
```

2.2.3 Pairwise Correlation of the predictors

From (Figure 4) below, the “cor()” function produces a matrix that contains all of the pairwise correlations among the predictors in a data set.

```
# removing the last column and set the color (col=) to the class of the health_data
pairs(health_data, -ncol(health_data), col = health_data$class)
```

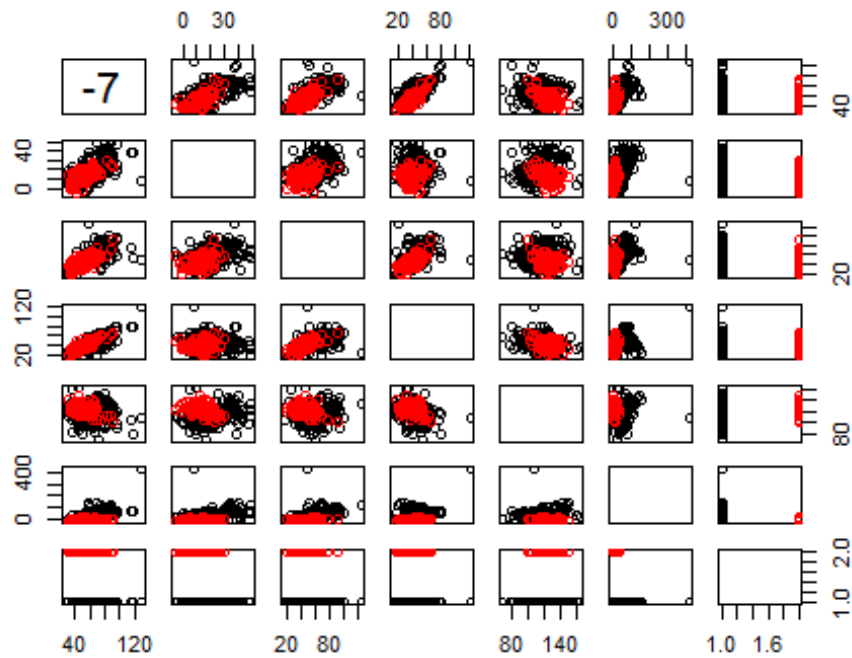



Figure 4: Pairwise Correlation Plot

```
health_without_class<- health_data[, -7]
cor(health_without_class)
```

	pelvic_incidence	pelvic_tilt.numeric	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisthesis
pelvic_incidence	1.0000000	0.62919877	0.71728236	0.81495999	-0.24746721	0.63874275
pelvic_tilt.numeric	0.6291988	1.00000000	0.43276386	0.06234529	0.03266781	0.39786228
lumbar_lordosis_angle	0.7172824	0.43276386	1.00000000	0.59838689	-0.08034361	0.53366701
sacral_slope	0.8149600	0.06234529	0.59838689	1.00000000	-0.34212835	0.52355746
pelvic_radius	-0.2474672	0.03266781	-0.08034361	-0.34212835	1.00000000	-0.02606501
degree_spondylolisthesis	0.6387427	0.39786228	0.53366701	0.52355746	-0.02606501	1.00000000

2.2.4 Correlation Matrix of the predictors

The `ggcorr()` function ignores the non-numeric variables automatically, which saves a lot of time by of the user from “subsetting-out” such variables prior to the matrix creation.

As seen from the (Figure 5) the highest correlated value is 0.8 which occurs between the biomechanical parameters (variables) `sacral_slope` and `pelvic_incidence`.

#In order to avoid guessing the correlation, this is a better presentation to see correlation between the variables

```
ggcorr(health_data, label = TRUE, label_alpha = TRUE)
```

```
## Warning in ggcorr(health_data, label = TRUE, label_alpha = TRUE): data in
## column(s) 'class' are not numeric and were ignored
```

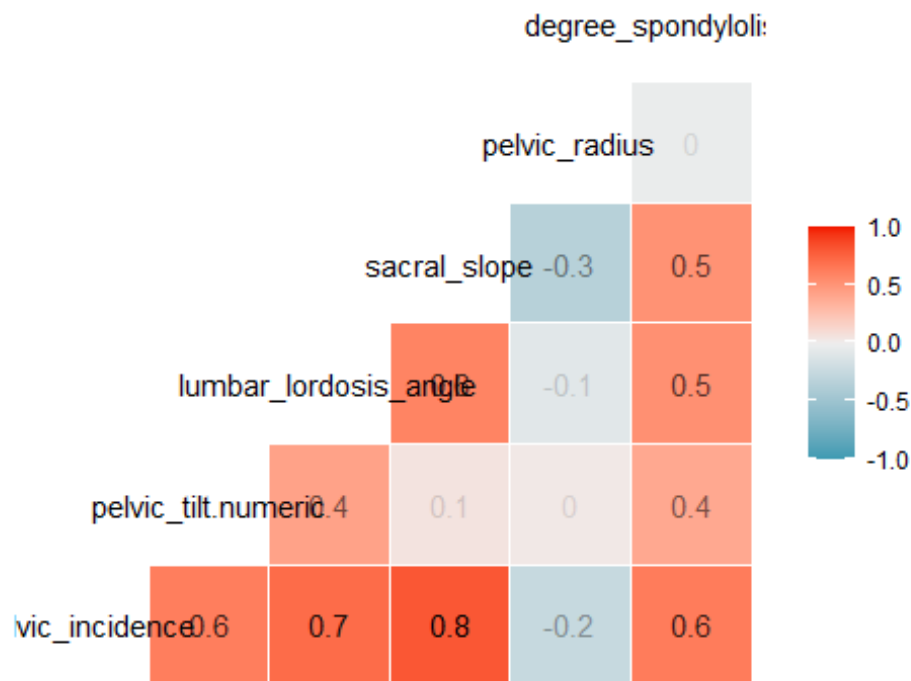


Figure 5 Correlation Matrix

#The ggcorr function ignores the non-numeric variables automatically, which saves a lot of time by of the user from “subsetting-out” such variables prior to the matrix creation

2.2.5 Box plot distributions for each variable

The (Figure 6) shows the boxplot of each attribute and the outliers. Also, in this report the outliers will be ignored, as these measurements are important for the analysis of the project. The attributes with the most outliers are the “`pelvic_radius`” and

“pelvic.tilt_numeric” ,these attributes are measured in angles, and angles can either be positive or negative.

```
# Box plot distributions for each variable
par(mfrow=(c(3,3)))
boxplot(health_data$pelvic_incidence,xlab = "pelvic_incidence", main = "Pelvic Incidence")
boxplot(health_data$pelvic_tilt.numeric, xlab = "pelvic_tilt.numeric", main = "Pelvic Tilt")
boxplot(health_data$lumbar_lordosis_angle, xlab = "lumbar_lordosis_angle", main = "Lordosis Angle")
boxplot(health_data$sacral_slope, xlab = "sacral_slope", main = "sacral Slope")
boxplot(health_data$pelvic_radius, xlab = "pelvic_radius", main = "Pelvic Radius")
boxplot(health_data$degree_spondylolisthesis, xlab = "degree_spondylolisthesis", main = "Degree Spondylolisthesis")
```

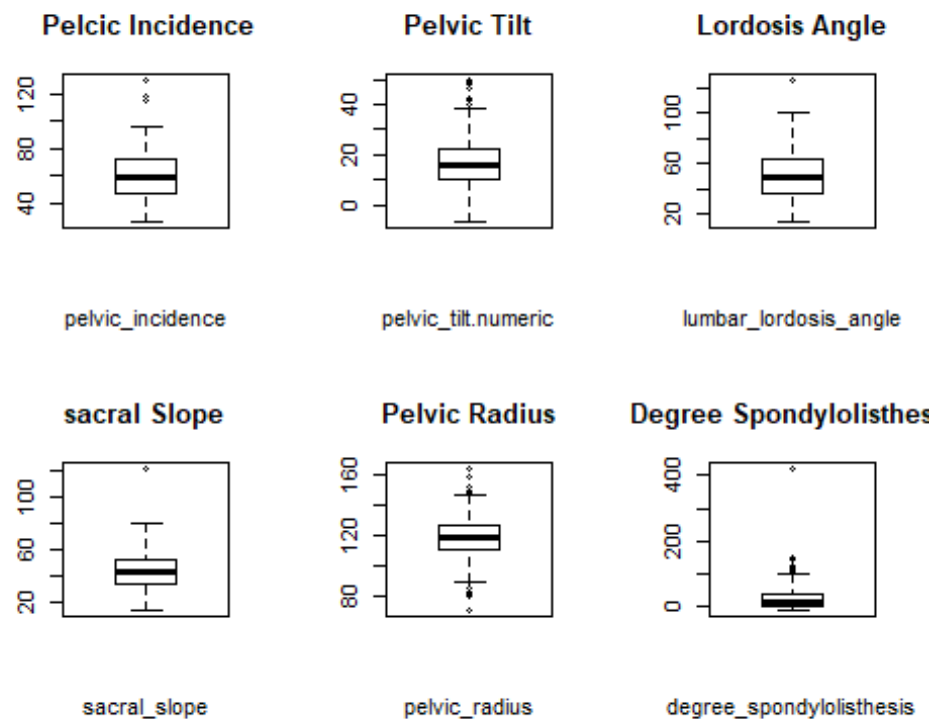


Figure 6: Box Plot Distribution of Attributes

2.2.6 Feature plot

(Figure 7) shows the boxplots of the independent variables as a function of the 2 classes.

Also, from the feature plot patients classified as Normal appears to be smaller than patients that are classified to be Abnormal. Furthermore, comparing the consistency of

the 2 classes, the Normal class is more consistent than the Abnormal class in most attributes, especially the “pelvic _ radius”, as the interquartile range spreads less for the Normal class than the Abnormal class. The Abnormal class has the lowest minimum value for each attributes. Also, the Abnormal class has the greatest range for each attributes with and without outliers .This simply means the Abnormal patients classified as Abnormal have the greatest variety of bio-mechanical measurements. Furthermore, the Abnormal class has the highest median regarding all attributes except the “pelvic_radius” which the median value is higher for the Normal Class. Analyzing the boxplot, 75% of the patients which are classified Abnormal for all the attributes have a higher measurement than the median measurement of patients classified as Normal for each attribute, except for the “pelvic_radius” attribute which differs. As seen, the median value of the boxplot are higher for the Abnormal class for each variable except for the “pelvic_radius” variable which differs. This plots also showed that the higher the values for each biomechanical variable the more likely the patient result will be Abnormal, except for where “pelvic_radius” measurement is about the 124degrees or over, chance of the patient to be classified Normal increases. This means that the pelvic _radius plays a huge factor for a patient to be classified as Normal.

```
#feature plot
#This shows the box plot for all the 6 variables as a function to the 2 classes
par(mfrow=c(2, 2))
featurePlot(x=health_data[,1:6], y=health_data$class, plot="box", scales=list(
x=list(relation="free"), y=list(relation="free")), col ="blue")
```

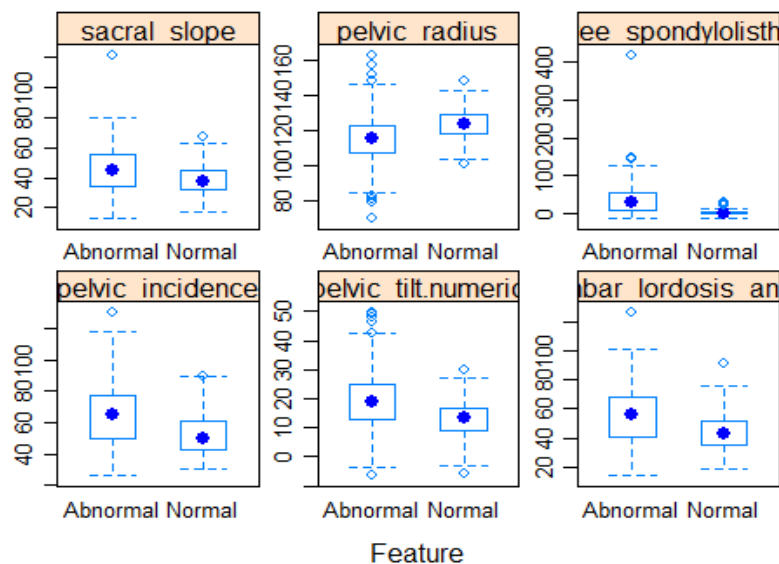


Figure 7: Feature Plot

3 Supervised learning experiments (Classification)

In this section the following classification algorithm below will be used to build a model which can successfully predict the condition of a patient from the bio-mechanical health dataset

- RandomForest
- CART(Classification And Regression Trees)
- KNN(K Nearest Neighbour)
- SVM(Support Vector Machines)

These classification models all have their individual strengths depending on the dataset provided.

Evaluation metrics used in this project:

Confusion Matrix: Recall - This metric judges models by how few false negatives are predicted, Accuracy - Accurate predictions of the model, Kappa - This measures the agreement between two raters, in the context of classification, observed (or actual) class and predicted class.

3.1 Data Partition

After split, Training = 248 observations and 7 variables and Testing = 62 observation and 7 variables.

```
TrainingIndex <- createDataPartition(health_data$class, p=0.8, list = FALSE)
Training <- health_data[TrainingIndex,] # Training Set
Testing <- health_data[-TrainingIndex,] # Test Set
```

3.2 Random Forest

Random forest is developed by aggregating trees, which can be used for both classification and regression. This Classification model was used in this project, as it avoids overfitting. In addition, in order for this algorithm to make a prediction, every tree makes a decision about new instances and we use a voting scheme to classify them.

3.2.1 Methodology

- The caret package are used in this section.
- Set seed to 1 for all models to ensure consistent reproducibility of results
- The data will be split training and testing set (~ 80/20 split). Testing set will be used to for final experiment and compared with the training set.
- For this classification model the “cv” K-fold cross-validation will be used to evaluate the model and to avoid overfitting
- Setting method to “rf”

- Using tuneGrid argument to tune the mtry value to optimize the model as the mtry has an effect on the performance of the model *Validating with the Confusion Matrix
- On the confusion matrix The positive class is set to “Abnormal” and negative class set to “Normal”.

```
#setting seed
set.seed(1)
#Train control function
tr <- trainControl(method="cv", number = 10)
#Fitting the model in the train function and setting method to "rf"
#
RF.model <- train(class ~ ., data = Training,
                  method = "rf",
                  trcontrol = tr,
                  preProcess=c("center", "scale"),
                  tuneGrid = expand.grid(.mtry=c(2,4,6)))
```

As seen from the (Figure 8) below the model reaches its optimal performance with the mtry value of 2 with an accuracy of 85.2% and a Kappa value of 66.24%. The mtry value (Randomly selected number of trees) is inversely proportional to the accuracy of the model, as seen when the mtry value increases the accuracy reduces. In this project, the tune grid parameter was preferably used, as it searches over the entire grid.

```
set.seed(1)
print(RF.model)

## Random Forest
##
## 248 samples
## 6 predictor
## 2 classes: 'Abnormal', 'Normal'
##
## Pre-processing: centered (6), scaled (6)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 248, 248, 248, 248, 248, 248, ...
## Resampling results across tuning parameters:
##
##  mtry  Accuracy  Kappa
##  2     0.8521184 0.6624330
##  4     0.8493814 0.6572480
##  6     0.8405899 0.6409477
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.

plot(RF.model)
```

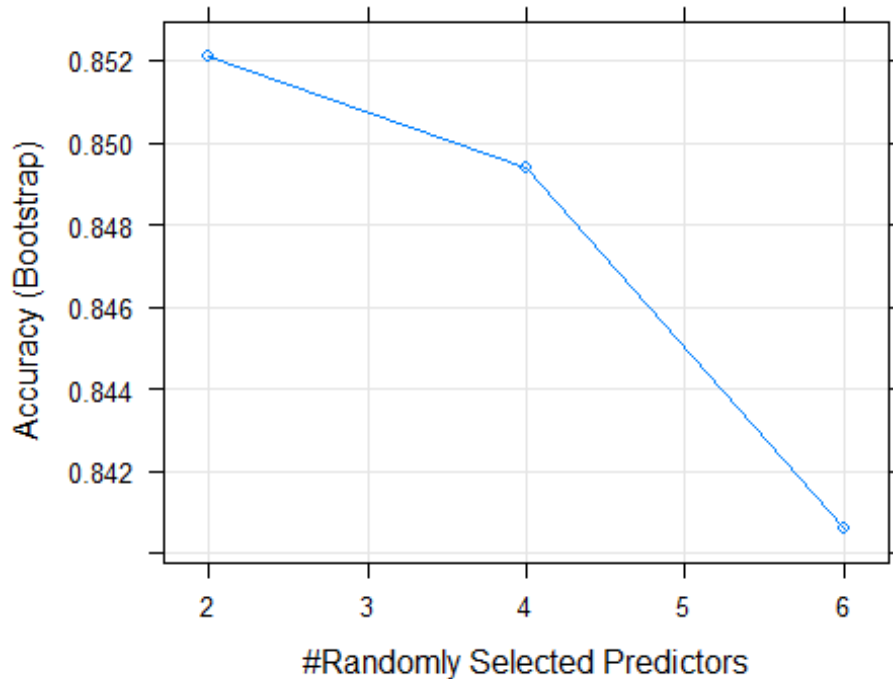


Figure 8: Accuracy(Bootstrap) vs Randomly Selected Predictors

3.2.2 Validating Random Forest Model

3.2.3 Observations

Training - Accuracy : 100% with 95% CI : (0.9852, 1). This section shows the performance of both the Training and Testing set using the Random forest model. As seen below, The training model performs perfectly well with an accuracy of 100%.Analysing the confusion matrix of the model, this model has a total of 248 observations,the actual model has 168 patients classified as Abnormal and the model correctly predicted 168 as Abnormal (True Positive) with 0 patients classified incorrectly (False Negative) .Also, the model predicted 80 patients to be normal (True Negative) which is exactly the same actual model.

Testing - Accuracy : 0.8226 % with 95% CI : (0.7047, 0.908) The model correctly predicted 34 as Abnormal and incorrectly predicted 8 to be Normal. Also, correctly predicted 17 patients to be Normal and incorrectly predicted 3 to be Abnormal which were actually Normal

95% CI : (0.7047, 0.908) which does not overlap with 95% CI : (0.9852, 1)

So there is a significant difference between the confidence interval of the training and testing sets.

```

set.seed(1)

#Predicting the training and the test
trainResultsRF <- predict(RF.model ,Training)
testResultsRF <- predict(RF.model ,Testing)

#Confusion matrix
confusionMatrix(trainResultsRF, Training$class,positive = "Abnormal",mode="everything")

## Confusion Matrix and Statistics
##
##              Reference
## Prediction Abnormal Normal
##   Abnormal      168      0
##   Normal         0      80
##
##              Accuracy : 1
##              95% CI : (0.9852, 1)
##   No Information Rate : 0.6774
##   P-Value [Acc > NIR] : < 2.2e-16
##
##              Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##              Sensitivity : 1.0000
##              Specificity : 1.0000
##              Pos Pred Value : 1.0000
##              Neg Pred Value : 1.0000
##              Precision : 1.0000
##              Recall : 1.0000
##              F1 : 1.0000
##              Prevalence : 0.6774
##              Detection Rate : 0.6774
##   Detection Prevalence : 0.6774
##       Balanced Accuracy : 1.0000
##
##       'Positive' Class : Abnormal
##

confusionMatrix(testResultsRF, Testing$class,positive = "Abnormal",mode="everything")

## Confusion Matrix and Statistics
##
##              Reference
## Prediction Abnormal Normal
##   Abnormal       34      3

```



```

##      Normal          8      17
##
##              Accuracy : 0.8226
##              95% CI : (0.7047, 0.908)
##      No Information Rate : 0.6774
##      P-Value [Acc > NIR] : 0.008074
##
##              Kappa : 0.619
##
##      McNemar's Test P-Value : 0.227800
##
##              Sensitivity : 0.8095
##              Specificity : 0.8500
##      Pos Pred Value : 0.9189
##      Neg Pred Value : 0.6800
##              Precision : 0.9189
##              Recall : 0.8095
##              F1 : 0.8608
##              Prevalence : 0.6774
##      Detection Rate : 0.5484
##      Detection Prevalence : 0.5968
##      Balanced Accuracy : 0.8298
##
##      'Positive' Class : Abnormal
##

```

3.3 CART(Classification And Regression Trees)

Applying the CART to both the Training and Testing dataset for the health dataset. The CART is a decision tree algorithm which uses a set of binary rules to calculate a target class. This algorithm is applied because it is popularly used in the medical sector for diagnosis of medical conditions based on laboratory measurements.

3.3.1 Methodology

- The “rpart” package are used in this section.
- Set seed to 1 for all models to ensure consistent reproducibility of results .
- The data will be split training and testing set (~ 80/20 split). Testing set will be used to for final experiment and compared with the training set.
- The method parameter is set to “rpart” in the train function
- Using classification model the “cv” K-fold cross-validation
- The preprocess function simplifies the dataset for the model
- Using tuneGrid argument to tune the “cp” (Complexity Parameter) value to optimize the model as this has an effect on the performance of the model.
- Validating model with Confusion matrix.The postive class is set to “Abnormal” and negative class set to “Normal”.

```

set.seed(1)
tr <- trainControl(method="cv", number = 10)
rpart.model <- train(class~.,
                     data=Training,
                     method="rpart",
                     preProcess=c("center", "scale"),
                     tuneGrid=expand.grid(.cp=c(0.002,0.005,0.01,0.015,0.02,0.03
)),
                     trControl=tr)

```

As seen, the cp value has an effect on the performance of the model. The (Figure 9) shows. However, the model had the highest accuracy value of 83.6% when the cp value was 0.03.

Analyzing the decision tree diagram in (Figure 10), the variable “degree_spondylolisthes” is the root node, which is the starting point of the tree and the leaves on the tree include the variables “pelvic_radius”, “sacral_slope”, “pelvic_tilt.numeric”.

The first split tests whether the variable “degree_spondylolisthes” is ≥ -0.43 . If yes, the model says to predicts Abnormal, and if no, the model moves on to the next split. Then, the second split checks whether or not the variable “pelvic_radius” is < 0.53 . If no, the model says to predict Normal, but if yes, the model moves on to the next split. The third split checks whether or not the variable “sacral_slope” is < -0.21 . If no, the model says to predict Normal and if yes, the model moves to the next split. The fourth split checks whether or not the variable “pelvic_tilt.numeric” is ≥ -0.77 . If yes, then the model says to predict Abnormal and if no, the model says to predict Normal.

```

print(rpart.model)

## CART
##
## 248 samples
##   6 predictor
##   2 classes: 'Abnormal', 'Normal'
##
## Pre-processing: centered (6), scaled (6)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 224, 223, 223, 223, 223, 223, ...
## Resampling results across tuning parameters:
##
##   cp      Accuracy   Kappa
##   0.002   0.8313333   0.6054464
##   0.005   0.8313333   0.6054464
##   0.010   0.8313333   0.6054464
##   0.015   0.8233333   0.6138606
##   0.020   0.8233333   0.6138606
##   0.030   0.8363333   0.5990035
##

```

```
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.03.
```

```
#Checking for complexity parameter accuracy
plot(rpart.model)
```

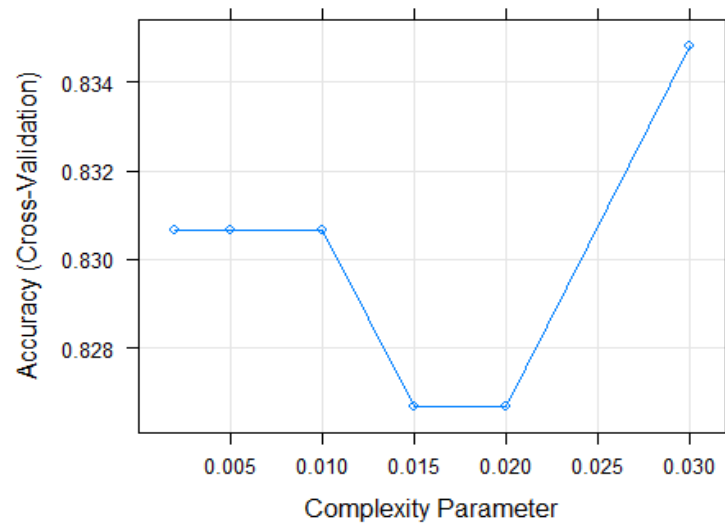


Figure 9: Accuracy(Cross-Validation) vs Complexity Parameter

```
#Visualizing the results
fancyRpartPlot(rpart.model$finalModel)
```

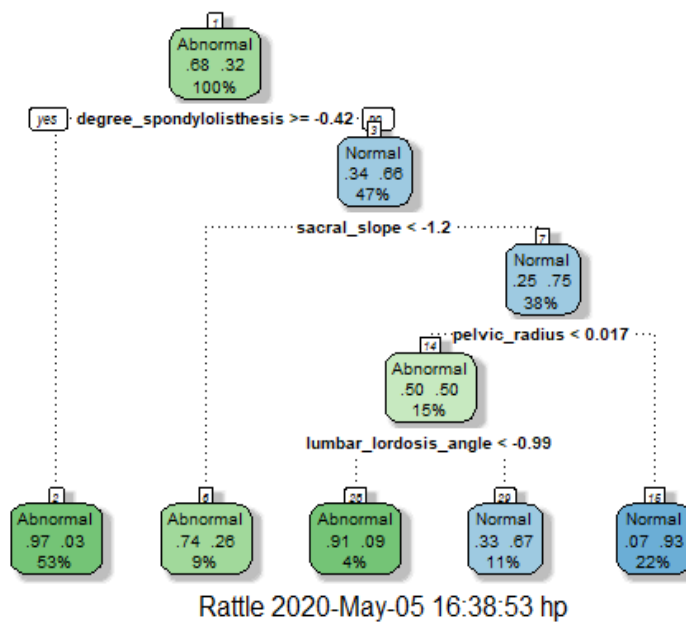


Figure 10: Decison Tree

3.3.2 Validating CART Model

3.3.3 Observations

Training - Accuracy : 0.8992 with 95% CI : (0.9852, 1) The model correctly predicted 161 as Abnormal and incorrectly predicted 7 to be Normal. Also, correctly predicted 62 patients to be Normal and incorrectly predicted 18 to be Abnormal which were actually Normal

Testing - Accuracy : 0.7903 with 95% CI : (0.8548, 0.9337) The model correctly predicted 33 as Abnormal and incorrectly predicted 9 to be Normal. Also, correctly predicted 16 patients to be Normal and incorrectly predicted 4 to be Abnormal which were actually Normal

95% CI : (0.6682, 0.8834) which overlaps with 95% CI : (0.8548, 0.9337)

So there is no significant difference between the accuracy of the training and testing sets. The training set performs better than the test set which is normal in most cases.

```
# ensure reproducibility of results by setting the seed to a known value
set.seed(1)

#Predicting the training and the test
trainResultsRpart <- predict(rpart.model ,Training)
testResultsRpart <- predict(rpart.model ,Testing)

#Confusion matrix (Setting positive class to Abnormal)
confusionMatrix(trainResultsRpart,Training$class,positive ="Abnormal",mode="everything")

## Confusion Matrix and Statistics
##
##              Reference
## Prediction Abnormal Normal
##   Abnormal      161     18
##   Normal         7      62
##
##              Accuracy : 0.8992
##              95% CI : (0.8548, 0.9337)
##   No Information Rate : 0.6774
##   P-Value [Acc > NIR] : <2e-16
##
##              Kappa : 0.7607
##
##   Mcnemar's Test P-Value : 0.0455
##
##              Sensitivity : 0.9583
##              Specificity : 0.7750
##   Pos Pred Value : 0.8994
##   Neg Pred Value : 0.8986
##              Precision : 0.8994
```

```

##              Recall : 0.9583
##              F1 : 0.9280
##              Prevalence : 0.6774
##              Detection Rate : 0.6492
##              Detection Prevalence : 0.7218
##              Balanced Accuracy : 0.8667
##
##              'Positive' Class : Abnormal
##

confusionMatrix(testResultsRpart, Testing$class, positive = "Abnormal", mode="everything")

## Confusion Matrix and Statistics
##
##              Reference
## Prediction Abnormal Normal
##   Abnormal      33      4
##   Normal        9     16
##
##              Accuracy : 0.7903
##              95% CI : (0.6682, 0.8834)
##              No Information Rate : 0.6774
##              P-Value [Acc > NIR] : 0.03518
##
##              Kappa : 0.5497
##
##              Mcnemar's Test P-Value : 0.26726
##
##              Sensitivity : 0.7857
##              Specificity : 0.8000
##              Pos Pred Value : 0.8919
##              Neg Pred Value : 0.6400
##              Precision : 0.8919
##              Recall : 0.7857
##              F1 : 0.8354
##              Prevalence : 0.6774
##              Detection Rate : 0.5323
##              Detection Prevalence : 0.5968
##              Balanced Accuracy : 0.7929
##
##              'Positive' Class : Abnormal
##

```

3.4 KNN(K Nearest Neighbour)

This is a lazy algorithm used for both classification and regression. This algorithm uses 'feature similarity' to predict the new data point values. This algorithm was used in this project because despite the simplicity, this algorithm can outperform other classifiers and it is used in a variety of applications such as economic forecasting and genetics etc.

3.4.1 Methodology

- The “caret” package is used.
- Set seed to 1 for all models to ensure consistent reproducibility of results.
- The data will be split training and testing set (~ 80/20 split). Testing set will be used to for final experiment and compared with the training set.
- The method parameter is set to “knn” in the train function
- Using the same classification model the “cv” K-fold cross-validation
- The preprocess function simplifies the dataset for the model
- Using tuneGrid argument to tune the “k” value to optimize the model as this has an effect on the performance of the model.
- Validating model with Confusion matrix.

```
# ensure reproducibility of results by setting the seed to a known value
set.seed(1)
tr <- trainControl(method="cv", number = 10)
knn.model <- train(class~.,
                    data=Training,
                    method="knn",
                    preprocess=c("center", "scale"),
                    tuneGrid=expand.grid(.k=1:10), #testing k value from 1 to 10
                    trControl=tr)
```

From the (Figure 11) shows the various tested values of k from 1 to 10. As seen, the value of “k” which is 2, has the highest model optimization of 83% accuracy.

```
print(knn.model)

## k-Nearest Neighbors
##
## 248 samples
## 6 predictor
## 2 classes: 'Abnormal', 'Normal'
##
## Pre-processing: centered (6), scaled (6)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 224, 223, 223, 223, 223, 223, ...
## Resampling results across tuning parameters:
##
##  k   Accuracy   Kappa
##  1  0.8225000  0.6094478
##  2  0.7945000  0.5478649
##  3  0.8148333  0.5785610
##  4  0.7906667  0.5278181
##  5  0.8268333  0.5978207
##  6  0.8388333  0.6290178
##  7  0.8270000  0.5995538
##  8  0.8148333  0.5600392
##  9  0.8108333  0.6045437
```

```
## 10 0.8273333 0.6045437
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 2.

plot(knn.model)
```

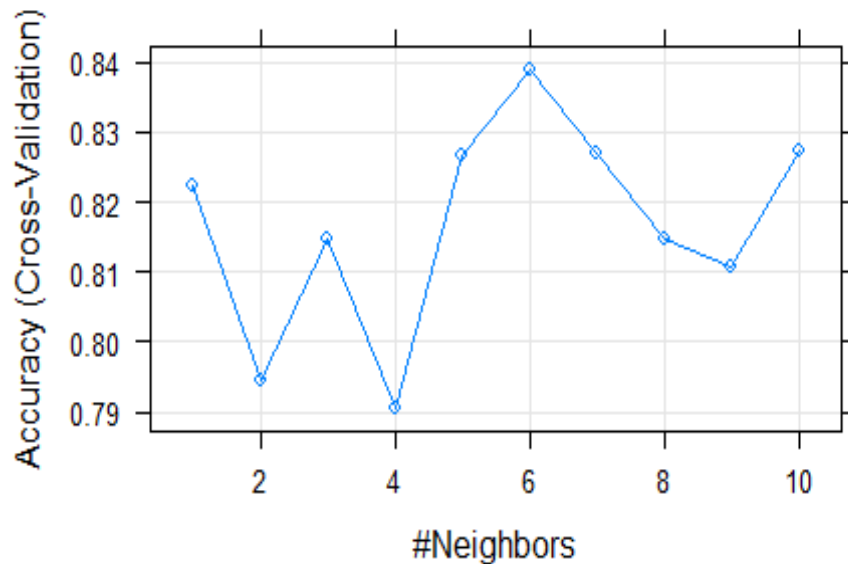


Figure 11: Accuracy (Cross-Validation) vs Neighbours

3.4.2 Validating KNN Model

3.4.3 Observations

Training - Accuracy : 0.8629 95% CI : (0.8137, 0.9032) The model correctly predicted 152 as Abnormal and incorrectly predicted 16 to be Normal. Also, correctly predicted 62 patients to be Normal and incorrectly predicted 18 to be Abnormal which were actually Normal

Testing - Accuracy : 0.7419 with 95% CI : (0.615, 0.8447) The model correctly predicted 33 as Abnormal and incorrectly predicted 9 to be Normal. Also, correctly predicted 13 patients to be Normal and incorrectly predicted 7 to be Abnormal which were actually Normal

95% CI : (0.615, 0.8447) which overlaps with 95% CI : (0.8137, 0.9032)

There is no significant difference between the accuracy of the training and testing sets.

```
set.seed(1)

#Predicting the training and the test
trainResultsKnn <- predict(knn.model , Training)
testResultsKnn <- predict(knn.model , Testing)
```

```

#Confusion matrix
confusionMatrix(trainResultsKnn, Training$class, positive = "Abnormal", mode="everything")

## Confusion Matrix and Statistics
##
##              Reference
## Prediction Abnormal Normal
##   Abnormal      152      18
##   Normal        16      63
##
##              Accuracy : 0.8629
##              95% CI : (0.8137, 0.9032)
##   No Information Rate : 0.6774
##   P-Value [Acc > NIR] : 1.515e-11
##
##              Kappa : 0.6842
##
##  Mcnemar's Test P-Value : 0.8638
##
##              Sensitivity : 0.9048
##              Specificity : 0.7750
##              Pos Pred Value : 0.8941
##              Neg Pred Value : 0.7949
##              Precision : 0.8941
##              Recall : 0.9048
##              F1 : 0.8994
##              Prevalence : 0.6774
##              Detection Rate : 0.6129
##              Detection Prevalence : 0.6855
##              Balanced Accuracy : 0.8399
##
##              'Positive' Class : Abnormal
##

confusionMatrix(testResultsKnn, Testing$class, positive = "Abnormal", mode="everything")

## Confusion Matrix and Statistics
##
##              Reference
## Prediction Abnormal Normal
##   Abnormal      33      7
##   Normal        9     13
##
##              Accuracy : 0.7419
##              95% CI : (0.615, 0.8447)
##   No Information Rate : 0.6774
##   P-Value [Acc > NIR] : 0.1712
##

```



```
##                Kappa : 0.4246
##
## Mcnemar's Test P-Value : 0.8026
##
##                Sensitivity : 0.7857
##                Specificity : 0.6500
##                Pos Pred Value : 0.8250
##                Neg Pred Value : 0.5909
##                Precision : 0.8250
##                Recall : 0.7857
##                F1 : 0.8049
##                Prevalence : 0.6774
##                Detection Rate : 0.5323
##                Detection Prevalence : 0.6452
##                Balanced Accuracy : 0.7179
##
##                'Positive' Class : Abnormal
##
```

3.5 SVM (Support Vector Machine)

This algorithm is said to be a promising classifying approach as it is used in the medical sector for detecting diseases such as diabetes by using simple variables. In this project, this algorithm will be used to achieve the most suitable hyperplane which splits the classes in the dataset.

3.5.1 Methodology

- The “caret” package is used.
- Set seed to 1 for all models to ensure consistent reproducibility of results .
- The data will be split training and testing set (~ 80/20 split). Testing set will be used to for final experiment and compared with the training set.
- The method parameter is set to “svmLinear” in the train function
- Using the same classification model the “cv” K-fold cross-validation
- The preprocess function simplifies the dataset for the model.
- Using tuneGrid argument to tune the “C” value to optimize the model as this has an effect on the performance of the model.
- Validating model with Confusion matrix. The positive class is set to “Abnormal” and negative class set to “Normal”.

```
tr <- trainControl(method="cv", number = 10)
svm.model <- train(class~.,
  data=Training,
  method="svmLinear",
  preprocess=c("center", "scale"),
  tuneGrid=expand.grid(C=c(1:20)),
  trControl=tr)
```

As seen from the (Figure 12) as the cost increases, the model accuracy increases. The model accuracy is the highest accuracy at 87.5% with a kappa value of 72.1% when the Cost of the model is set to 9.

```
print(svm.model)

## Support Vector Machines with Linear Kernel
##
## 248 samples
## 6 predictor
## 2 classes: 'Abnormal', 'Normal'
##
## Pre-processing: centered (6), scaled (6)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 223, 223, 223, 223, 224, 223, ...
## Resampling results across tuning parameters:
##
##  C    Accuracy    Kappa
##  1  0.8626667  0.6855761
##  2  0.8626667  0.6855761
##  3  0.8626667  0.6855761
##  4  0.8666667  0.6934184
##  5  0.8666667  0.6934184
##  6  0.8666667  0.6934184
##  7  0.8666667  0.6934184
##  8  0.8666667  0.6934184
##  9  0.8666667  0.6934184
## 10  0.8666667  0.6934184
## 11  0.8666667  0.6934184
## 12  0.8666667  0.6934184
## 13  0.8666667  0.6934184
## 14  0.8666667  0.6934184
## 15  0.8666667  0.6934184
## 16  0.8666667  0.6934184
## 17  0.8666667  0.6934184
## 18  0.8666667  0.6934184
## 19  0.8666667  0.6934184
## 20  0.8666667  0.6934184
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 9.

plot(svm.model)
```

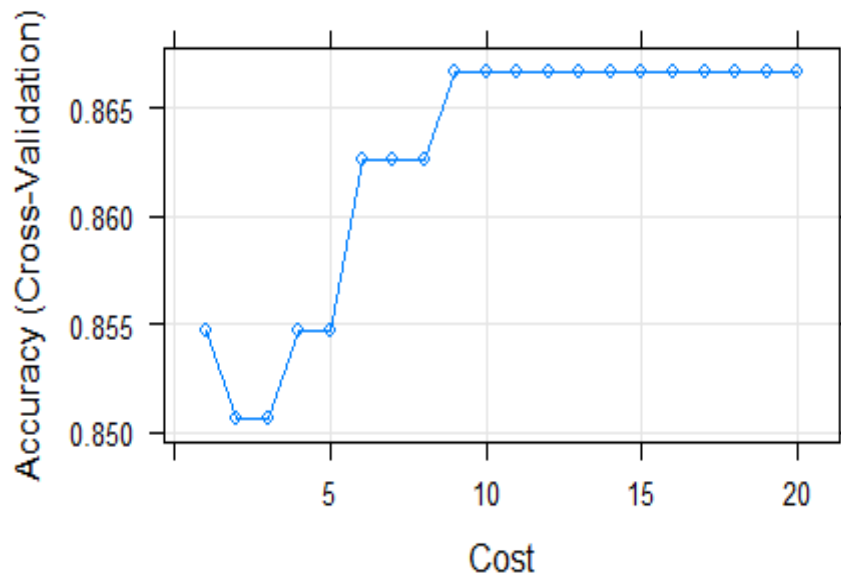


Figure 12: Accuracy (Cross-Validation) vs Cost

3.5.2 Validating SVM Model

3.5.3 Observations

Training - Accuracy : 0.8669 with 95% CI : (0.8182, 0.9066) The model correctly predicted 151 as Abnormal and incorrectly predicted 17 to be Normal. Also, correctly predicted 64 patients to be Normal and incorrectly predicted 16 to be Abnormal which were actually Normal

Testing - Accuracy : 0.7903 with 95% CI : (0.6682, 0.8834) The model correctly predicted 33 as Abnormal and incorrectly predicted 9 to be Normal. Also, correctly predicted 16 patients to be Normal and incorrectly predicted 4 to be Abnormal which were actually Normal

95% CI : (0.6682, 0.8834) which overlaps with 95% CI : (0.8182, 0.9066)

So there is no significant difference between the accuracy of the training and testing sets. The training set performs better than the test set which is normal in most cases.

```
set.seed(1)
```

```
#Predicting the training and the test
```

```
trainResultsSVM <- predict(svm.model ,Training)
```

```
testResultsSVM <- predict(svm.model ,Testing)
```

```
#Confusion matrix
```

```

confusionMatrix(trainResultsSVM, Training$class, positive = "Abnormal", mode = "everything")

## Confusion Matrix and Statistics
##
##              Reference
## Prediction Abnormal Normal
##   Abnormal      151      16
##   Normal        17      64
##
##              Accuracy : 0.8669
##              95% CI : (0.8182, 0.9066)
##   No Information Rate : 0.6774
##   P-Value [Acc > NIR] : 4.95e-12
##
##              Kappa : 0.6965
##
##  Mcnemar's Test P-Value : 1
##
##              Sensitivity : 0.8988
##              Specificity : 0.8000
##              Pos Pred Value : 0.9042
##              Neg Pred Value : 0.7901
##              Precision : 0.9042
##              Recall : 0.8988
##              F1 : 0.9015
##              Prevalence : 0.6774
##              Detection Rate : 0.6089
##              Detection Prevalence : 0.6734
##              Balanced Accuracy : 0.8494
##
##              'Positive' Class : Abnormal
##

confusionMatrix(testResultsSVM, Testing$class, positive = "Abnormal", mode = "everything")

## Confusion Matrix and Statistics
##
##              Reference
## Prediction Abnormal Normal
##   Abnormal      33      4
##   Normal        9      16
##
##              Accuracy : 0.7903
##              95% CI : (0.6682, 0.8834)
##   No Information Rate : 0.6774
##   P-Value [Acc > NIR] : 0.03518
##
##              Kappa : 0.5497

```

```
##
## McNemar's Test P-Value : 0.26726
##
##          Sensitivity : 0.7857
##          Specificity : 0.8000
##          Pos Pred Value : 0.8919
##          Neg Pred Value : 0.6400
##          Precision : 0.8919
##          Recall : 0.7857
##          F1 : 0.8354
##          Prevalence : 0.6774
##          Detection Rate : 0.5323
##          Detection Prevalence : 0.5968
##          Balanced Accuracy : 0.7929
##
##          'Positive' Class : Abnormal
##
```

4 Conclusion

4.1 Evaluating Model Performance and Analysis

The performance analysis is based on the Accuracy, Kappa and Recall. As seen from Training data result on the data frame below, all the models performed well, especially the Random forest which performed excellently well with the seen data. Also when tested with the unseen data, from the Testing data result, the SVM and the CART model are almost similar. However, the RandomForest has an edge over both in terms performances with reference to Accuracy of 80.60, Kappa of 61.90, but share the same Recall of 78.6% with CART model.

The result further shows that the RandomForest and CART model have the same Recall of 78.6% .The Recall judges the models by how few false negatives are predicted. Furthermore, It would be more of a critical problem if the medical test model predicted diagnosis of patients to be Normal (False Negative) when these patients were actually Abnormal rather than if the patients were diagnosed to be Abnormal when they were actually Normal (False Positive) because if thats the case, the medical personnel or administrator could further recommend or carry out various tests to verify the result.

Concisely, it is more appropriate to seek medical check if the system predicts Abnormal when you are actually Normal than to walk around with the thought of being healthy while the patient is Abnormal. This is point the Recall shows its importance in evaluating this medical data, as it is better to have fewer False Negatives predicted and this is correctly predicted 78.6% of the time.

Also, the accuracy simply shows how accurate the models predicts the patients status to either be Abnormal or Normal with reference to the original data. It is expected that the seen data (Training data) to perform better in terms of accuracy from the unseen data. However, as seen from the accuracy from the Training data results and Testing data results, all models still perform good with the unseen data, this simply means there is no sign of over fitting.

The Kappa basically measures the agreement between two raters (Abnormal or Normal), in the context of classification, observed (or actual) class and predicted class as it measures the degree of agreement between the two evaluators. The Random Forest has a Kappa value of 61.90 (substantial agreement). This shows that the classification predictions of the Random Forest model is substantial reliable. In addition, the reliability of the model to predict Abnormal patients correctly and Normal patients correct is very high when compared to other models. kNN seems to have the lowest Kappa of 42.46 (fair agreement) which is fairly reliable. This model would not be advised to use in this scenario.

```
Testingdata <- read.csv("Testing_Models.csv")
Trainingdata <- read.csv("Training_models.csv")
```

```
(Testingdata)
```

```
##  Model_Testing Accuracy Kappa Recall
## 1  RandomForest      80.60 61.90  78.60
## 2           CART      79.03 54.97  78.60
## 3           KNN      74.19 42.46  78.57
## 4           SVM      79.03 54.97  78.57
```

```
(Trainingdata)
```

```
##  Model_Training Accuracy  Kappa Recall
## 1  RandomForest    100.00 100.00 100.00
## 2           CART     89.92  76.07  95.83
## 3           KNN     86.29  68.42  89.94
## 4           SVM     86.69  69.65  89.88
```

4.1 Importance of Attributes on models

For the RandomForest model, the feature “degree_spondylolisthesis” is seen to be classified as the most important followed by the “pelvic radius” for making predictions. However, the “lumbar_lordosis_angle” is seen to be the least important feature in this model. The “degree_spondylolisthesis” and “pelvic_radius” contributed to most of the performance for this model.

```
varImp(RF.model)
```

```
## rf variable importance
##
##                               Overall
## degree_spondylolisthesis 100.000
```

```
## pelvic_radius          25.936
## pelvic_incidence       5.323
## pelvic_tilt.numeric    4.052
## lumbar_lordosis_angle  1.367
## sacral_slope           0.000
```

Analysing the importance of features on the CART model, The “degree_spondylolisthesis” still appears to be the most important feature and the “sacral_slope” appears to be the least important. The “pelvic_radius” drops to the third most important feature and the “pelvic_tilt.numeric” placed at the second. As seen, the “degree_spondylolisthesis” and “pelvic_radius” contributed to most of the performance for this model.

```
varImp(rpart.model)
```

```
## rpart variable importance
##
## Overall
## degree_spondylolisthesis 100.000
## pelvic_radius            45.675
## pelvic_tilt.numeric      27.353
## lumbar_lordosis_angle    26.447
## pelvic_incidence         9.951
## sacral_slope             0.000
```

For the kNN model, the “degree_spondylolisthesis” also still appears to be the most important feature for this model and the “sacral_slope” appears to still be the least important feature. Also, the “pelvic_incidence” is placed second and the “lumbar_lordosis_angle” is placed to be the third most important feature. The “degree_spondylolisthesis”, “sacral_slope”, “lumbar_lordosis_angle”, “pelvic_radius” contributed to most of the performance for this model.

```
varImp(knn.model)
```

```
## ROC curve variable importance
##
## Importance
## degree_spondylolisthesis 100.00
## pelvic_incidence         37.77
## pelvic_radius            35.75
## pelvic_tilt.numeric      23.84
## lumbar_lordosis_angle    23.13
## sacral_slope             0.00
```

Analyzing the importance of features on the SVM model, the “degree_spondylolisthesis” still appears to be the most important feature and the “sacral_slope” appears to be the least important. The “pelvic_radius” drops to the third most important feature and the “pelvic_tilt.numeric” placed at the second.

```
varImp(svm.model)
```

```
## ROC curve variable importance
##
##                               Importance
## degree_spondylolisthesis      100.00
## pelvic_incidence               37.77
## pelvic_radius                  35.75
## pelvic_tilt.numeric            23.84
## lumbar_lordosis_angle          23.13
## sacral_slope                   0.00
```

Finally, working with this medical dataset has been useful to more insight on using Machine Learning Algorithms for classification purposes. Also, after tuning the parameters of the models to attain the best results ,the analysis shows that the most suitable model to use for the patient diagnosis is the Random Forest, considering the evaluation metric values. Furthermore, from the variable importance of the models, the variables “degree_spondylolisthesis”, “pelvic_incidence”and “pelvic_radius” contributed most to the performance of the models as they appeared to always be within top 4 variables of importance for each model. On the other hand, “sacral_slope” was the last for 3 models and lumbar_lordosis_angle was the least for just 1 model.Also,the use of Artificial Neural Networks (ANNs) is growing rapidly in the world today, as this algorithm is applied in many projects on machine learning. In a future work on this project, the ANN can be useful in modelling the input and output relationships of the life expectancy dataset by learning directly from observed data.

5 Appendix

Data report overview

The dataset examined has the following dimensions:

Feature	Result
Number of observations	310
Number of variables	7

Codebook summary table

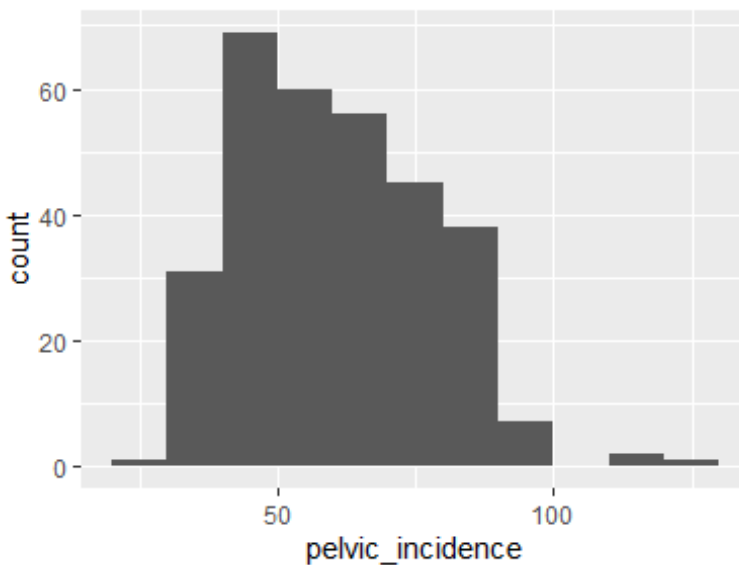
Label	Variable	Class	# unique values	Missing	Description
	pelvic_incidence	numeric	310	0.00 %	
	pelvic_tilt.numeric	numeric	310	0.00 %	
	lumbar_lordosis_angle	numeric	280	0.00 %	
	sacral_slope	numeric	281	0.00 %	

pelvic_radius	numeric	310	0.00 %
degree_spondylolisthesis	numeric	310	0.00 %
class	factor	2	0.00 %

Variable list

pelvic_incidence

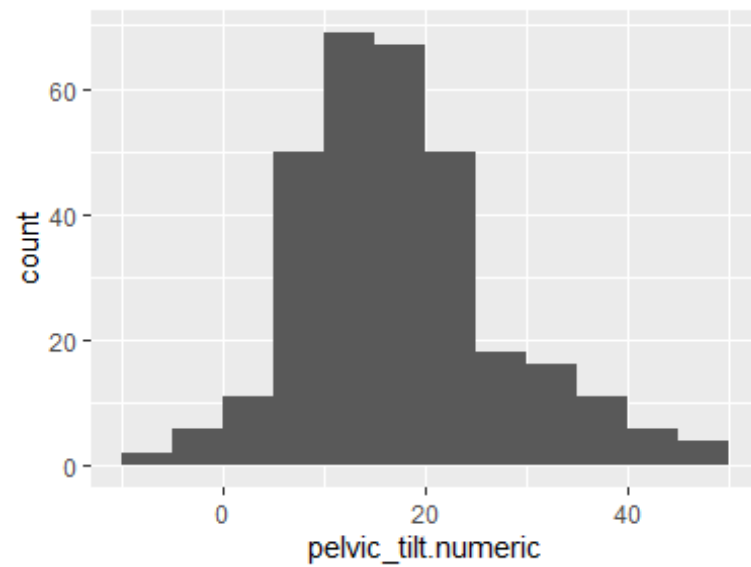
Feature	Result
Variable type	numeric
Number of missing obs.	0 (0 %)
Number of unique values	310
Median	58.69
1st and 3rd quartiles	46.43; 72.88
Min. and max.	26.15; 129.83



pelvic_tilt.numeric

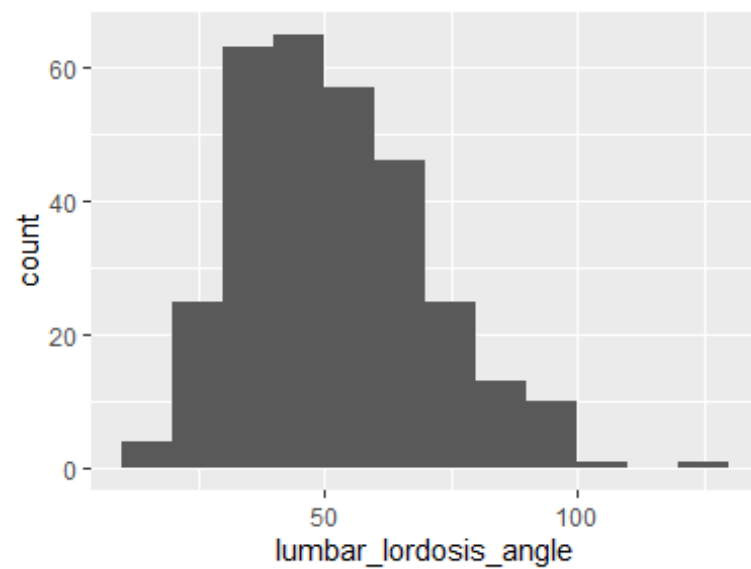
Feature	Result
Variable type	numeric
Number of missing obs.	0 (0 %)
Number of unique values	310
Median	16.36
1st and 3rd quartiles	10.67; 22.12

Min. and max. -6.55; 49.43



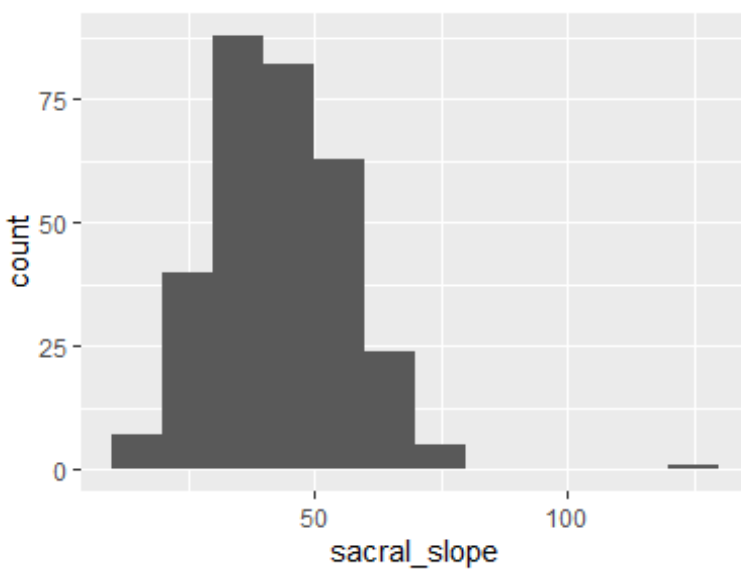
lumbar_lordosis_angle

Feature	Result
Variable type	numeric
Number of missing obs.	0 (0 %)
Number of unique values	280
Median	49.56
1st and 3rd quartiles	37; 63
Min. and max.	14; 125.74



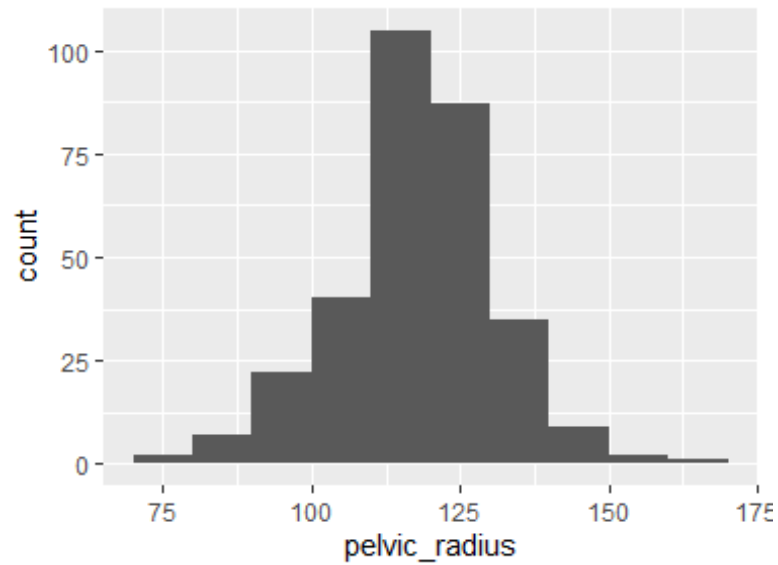
sacral_slope

Feature	Result
Variable type	numeric
Number of missing obs.	0 (0 %)
Number of unique values	281
Median	42.4
1st and 3rd quartiles	33.35; 52.7
Min. and max.	13.37; 121.43



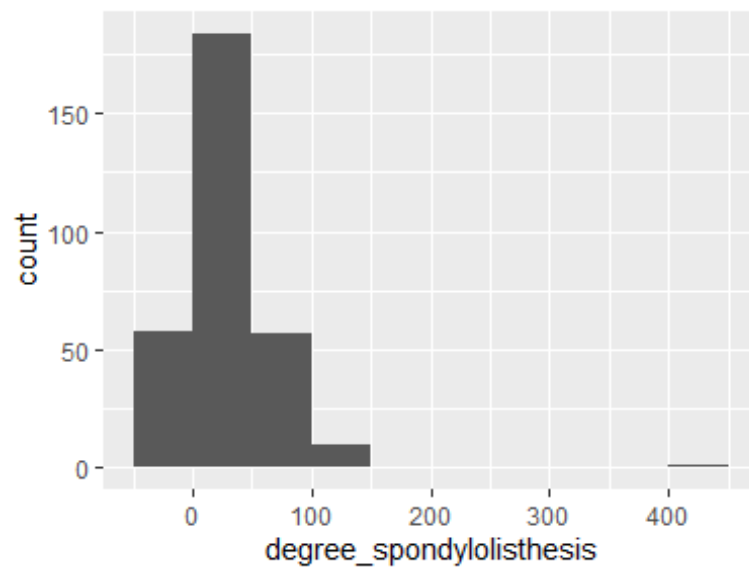
pelvic_radius

Feature	Result
Variable type	numeric
Number of missing obs.	0 (0 %)
Number of unique values	310
Median	118.27
1st and 3rd quartiles	110.71; 125.47
Min. and max.	70.08; 163.07



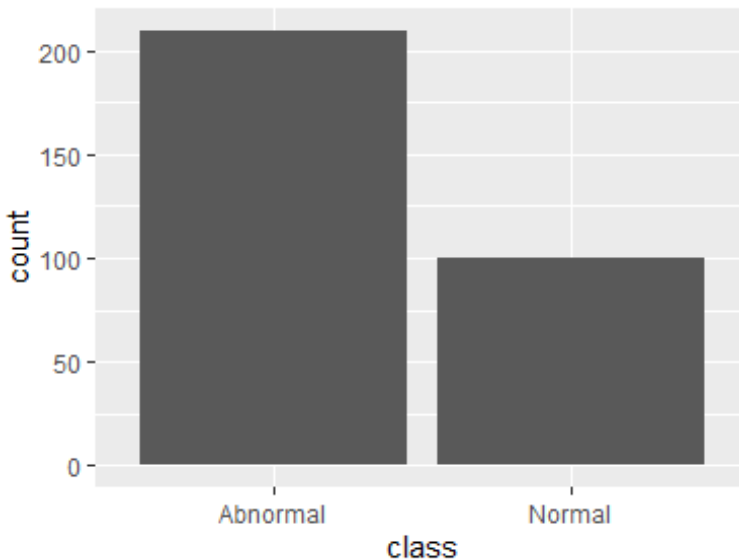
degree_spondylolisthesis

Feature	Result
Variable type	numeric
Number of missing obs.	0 (0 %)
Number of unique values	310
Median	11.77
1st and 3rd quartiles	1.6; 41.29
Min. and max.	-11.06; 418.54



class

Feature	Result
Variable type	factor
Number of missing obs.	0 (0 %)
Number of unique values	2
Mode	"Abnormal"
Reference category	Abnormal



- Observed factor levels: "Abnormal", "Normal".

Report generation information:

- Created by: Desmond Dumebi Ojei (1419100)
- Report creation time: Sat Mar 21 2020 14:01:18
- Report was run from directory: C:/Users/hp/Desktop/Data Science/Second Semester/Data Science Development
- dataMaid v1.4.0 [Pkg: 2019-12-10 from CRAN (R 3.6.3)]
- R version 3.6.1 (2019-07-05).
- Platform: x86_64-w64-mingw32/x64 (64-bit)(Windows 10 x64 (build 18363)).
- Function call: `dataMaid::makeDataReport(data = health_data, mode = c("summarize", "visualize", "check"), smartNum = FALSE, file = "codebook_health_data.Rmd", checks = list(character = "showAllFactorLevels", factor = "showAllFactorLevels", labelled = "showAllFactorLevels", haven_labelled = "showAllFactorLevels",`

```
numeric = NULL, integer = NULL, logical = NULL, Date = NULL), listChecks  
= FALSE, maxProbVals = Inf, codebook = TRUE, reportTitle = "Codebook for  
health_data")
```

```
'''
```