Data Science Capstone - Orthopedic Patient Predictions

Raven Houck

4/7/2019

Introduction

This report is part of the "HarvardX: PH125.9x Data Science: Capstone" course. The main goal of this project is to use all acquired skills that have been learned throughout the Data Science Professional Certificate program and to apply them to a real world problem. The main goal of this project is to use machine learning techniques to predict the class of orthopedic patients - either normal or abnormal. Each patient is represented in the dataset by six biomechanical attributes derives from the shape and orientation of the pelvis and lumbar spine. These variables are as follows: pelvic incidence, pelvic tilt, lumbar lordosis angle, sacral slope, pelvic radius, and the grade of spondylolisthesis. These variables will be used to classify the patients as belonging to one of two categories - normal or abnormal. The given task will be accomplished using a curated dataset from the UCI Machine Learning Repository. The following three statistical approaches will be used: k-nearest neighbors (kNN), logistic regression, and random forests. The method with the best prediction performance will be determined based on the highest calculated overall accuracy.

Methods and Analysis

Three machine learning algorithms are used to predict the class of orthopedic patients in this dataset.

Method 1: k-nearest Neighbors (kNN)

To start, k-nearest neighbors (kNN) will be used, since it is easiest to adapt to multiple dimensions. The main theory behind kNN is to define the distance between all observations based on the features. Then, for any point we want to estimate, we look for the k nearest points and take an average of 0s and 1s associated with those points.

Method 2: Logistic Regression

Next, logistic regression will be used as a machine learning algorithm on this dataset. Logistic regression is a sound regression approach that can be extended to categorical data. Logistic regression is a specific case of a set of generalized linear models. In general, this method is used when analyzing a dataset that had one or more independent variables that determine a specific outcome.

Method 3: Random Forests

Random Forests is a very popular machine learning approach that is able to address the shortcomings of decision trees using a clever idea. The main goal of this approach is to

improve prediction performacne and reduce instability by averaging multiple decision trees - a forest of trees that are conducted with randomness.

Train and Test Set Creation

In order to develop machine learning algorithms to predict the class of orthopedic patients, the dataset must first be downloaded and split into a train and test set.

```
#Dataset: [Biomechanical Features of Orthopedic Patients]
#(https://www.kaggle.com/uciml/biomechanical-features-of-orthopedic-patients)

setwd("/Users/Raven/Documents/R/Projects/Orthopedic_Patients")
ortho_data <- read.csv("column_2C_weka.csv")

#Create a train and test set - the test set will be 25% of the ortho_data
dataset
set.seed(1)
ind <- createDataPartition(ortho_data$class, times=1, p=0.25, list=FALSE)
train <- ortho_data[-ind, ]
test <- ortho_data[ind, ]
```

Data Exploration

Before diving in to the machine learning algorithms, let's take a look at the dataset to get familiar with it. We will make sure the train and test sets were created properly, and familiarize ourselves with the variables in this dataset.

First, let's take a look at the dimensions of the train and test set.

```
dim(train)

## [1] 232   7

dim(test)

## [1] 78  7

dim(ortho_data)

## [1] 310   7
```

Here, we see that we successfully created a test set that is 25% of the complete dataset. Next, let's explore the variable governing the dataset.

```
head(ortho_data)

##   pelvic_incidence pelvic_tilt.numeric lumbar_lordosis_angle sacral_slope
## 1         63.02782           22.552586              39.60912     40.47523
## 2         39.05695           10.060991              25.01538     28.99596
## 3         68.83202           22.218482              50.09219     46.61354
## 4         69.29701           24.652878              44.31124     44.64413
## 5         49.71286            9.652075              28.31741     40.06078
```

```
## 6         40.25020           13.921907               25.12495     26.32829
##   pelvic_radius degree_spondylolisthesis    class
## 1      98.67292                -0.254400 Abnormal
## 2     114.40543                 4.564259 Abnormal
## 3     105.98514                -3.530317 Abnormal
## 4     101.86850                11.211523 Abnormal
## 5     108.16872                 7.918501 Abnormal
## 6     130.32787                 2.230652 Abnormal
```

This shows that there are 6 variables we should explore when predicting the class of orthopedic patients. These variables include pelvic_incidence, pelvic_tilt.numeric, lumbar_lordosis_angle, sacral_slope, pelvic_radius, and degree_spondylolisthesis. Additionally, we will classify patients as belonging to one of two categories: normal or abnormal.

Let's see how many patients are considered abnormal in this dataset.

```
ortho_data %>% filter(class=="Abnormal") %>% tally
```

```
##     n
## 1 210
```

Thus, we see that 210 out of a total of 310 patients in this dataset are considered abnormal.

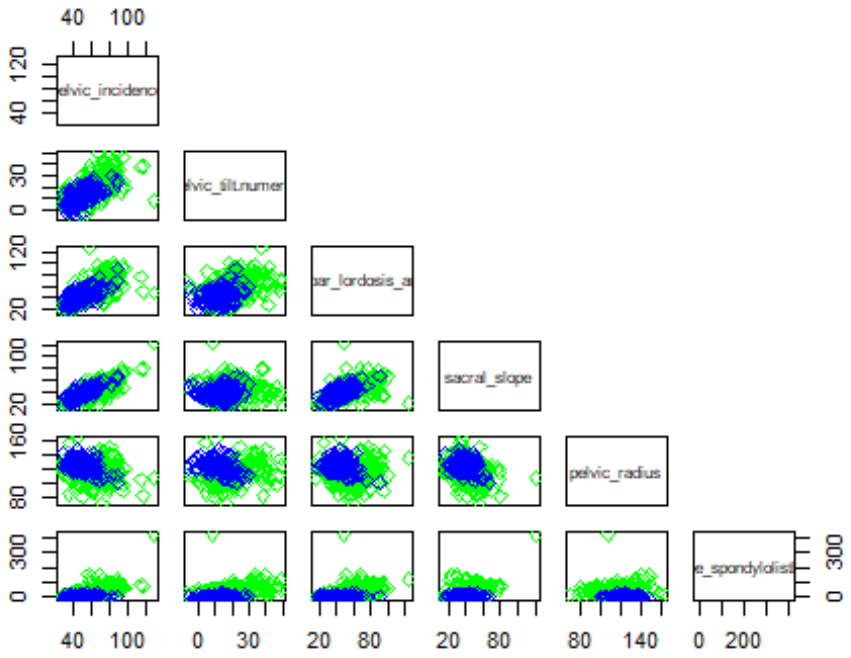
Next, let's start to examine the relationship between these variables and the class of the patient. The following plot shows the relationship between pelvic incidence, pelvic tilt, and the class of the patient.

```
ortho_data %>% ggplot(aes(pelvic_incidence, pelvic_tilt.numeric,
color=class)) + geom_point()
```

This scatterplot already shows a trend - patients with lower pelvic incidence and pelvic tilt tend to be normal, whereas patients with higher pelvic incidence and pelvic tilt tend to be abnormal.

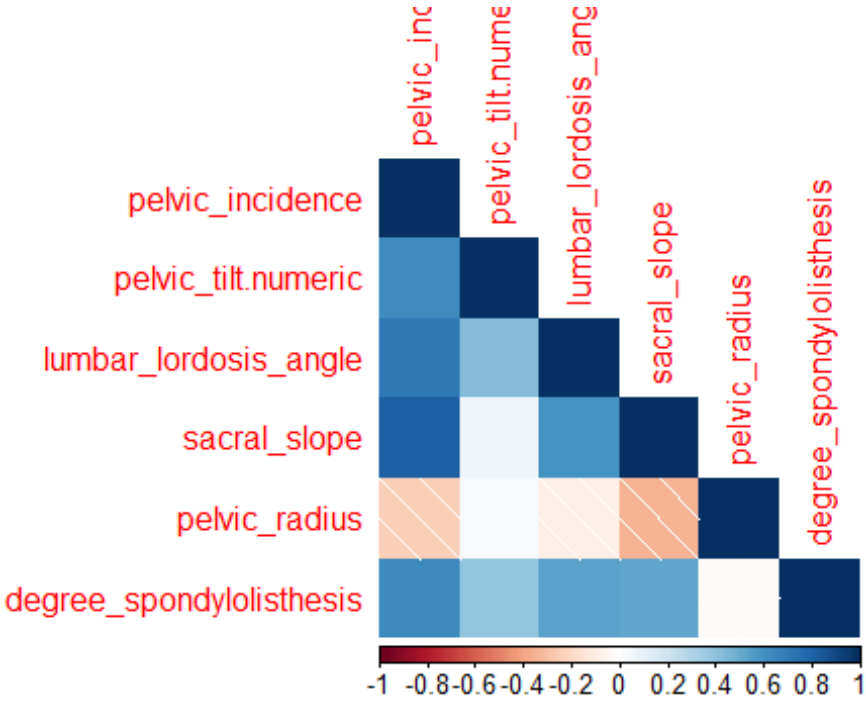
We have already began to see a trend regarding two of the variables in this dataset. The following plot will display the trends between each of the six variables that we are focused on.

```
pairs(ortho_data[,1:6], col=c("green","blue")[ortho_data$class], pch=5,
upper.panel=NULL)
```



Although this plot enables us to see some relationships between the variables, we want to look further into this. Next, the corrplot package is used to display correlations between variables. Note that positive correlations are displayed in blue, while negative correlations are displayed in red. Additionally, color intensity is proportional to the correlation coefficients.

```
#Install the necessary package
library(corrplot)

## Warning: package 'corrplot' was built under R version 3.5.3

## corrplot 0.84 loaded

M <- cor(ortho_data[,1:6])
corrplot(M, method="shade", type="lower")
```



Here, we see that the majority of these variables are positively correlated, meaning that one increases as the other increases. However, we see that pelvic radius in paticular has a slightly negative correlation coefficient.

Results

Algorithm 1 - k-nearest neighbors (kNN)

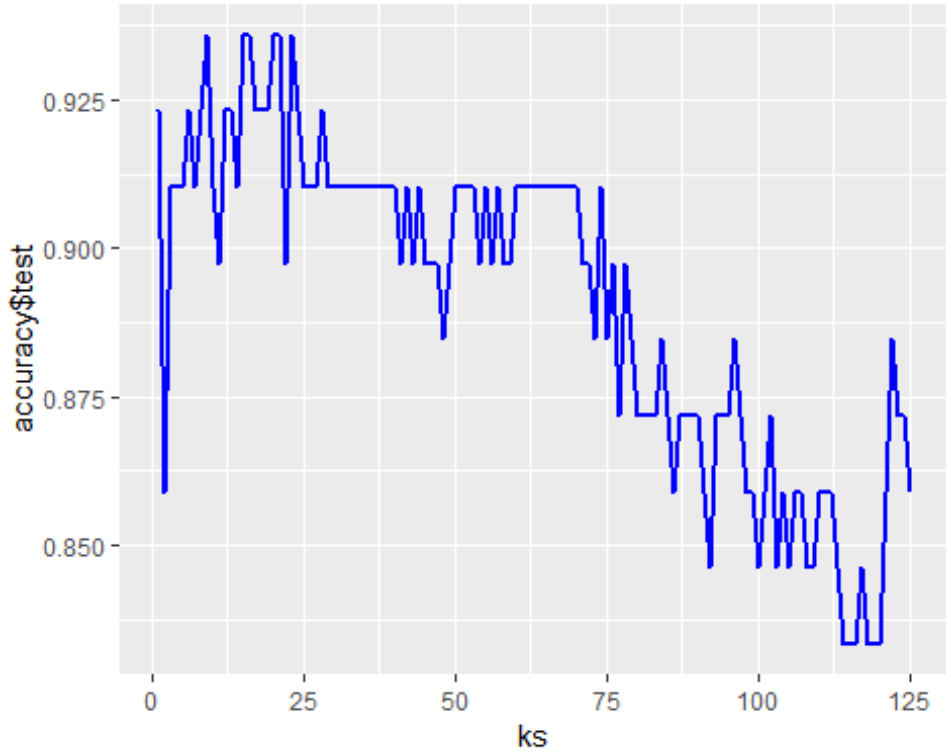
Let's begin by creating the kNN machine learning algorithm and determining it's accuracy.

```
#K-nearest neighbors
#Note that at this stage, we set k=5 for simplicity
knn_fit <- knn3(class~., data=train, k=5)
y_hat_knn <- predict(knn_fit, test, type="class")
confusionMatrix(data=y_hat_knn, reference=test$class)$overall["Accuracy"]

##  Accuracy
## 0.9102564
```

Using k set to be 5, the knn model achieves an accuracy of 91%. We can improve the accuracy of this model by determing the value of k that maximizes accuracy.

```
#Let's determine a k that maximizes accuracy
library(purrr)

#set.seed is used to ensure reproducibility
set.seed(2019)
```

```r
ks <- seq(1,125,1)

#Create the accuracy function
accuracy <- map_df(ks, function(k){
  fit <- knn3(class ~ ., data = train, k = k)

  y_hat <- predict(fit, test, type = "class")
  cm_train <- confusionMatrix(data = y_hat, reference = test$class)
  test_error <- cm_train$overall["Accuracy"]

  tibble(test = test_error)
})

#Display these results in a plot
ggplot() + geom_line(aes(x = ks, y = accuracy$test), color="blue", size=1)
```



Clearly, one can see that the maximum accuracy is achieved at a k value somewhere around 5 to 25. However, in order to determine the precise value of k that maximizes accuracy, and the resulting overall accuracy, the following code is needed.

```r
ks[which.max(accuracy$test)]
```

```
## [1] 9
```

```r
max(accuracy$test)
```

```
## [1] 0.9358974
```

```
#We see that k=9 produces the maxiumum accuracy of 93.6%
```

Thus, we see that k=9 maximizes the accuracy achieved by the kNN approach. This accuracy is calculated as 93.6%.

Algorithm 2 - Logistic Regression

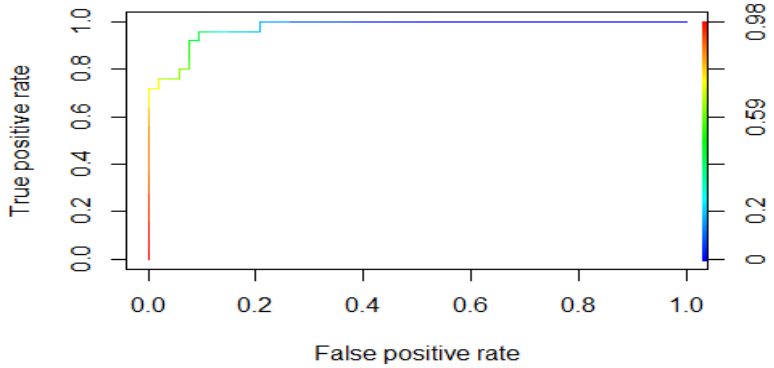
Next, a machine learning algorithm using logistic regression will be created and used to predict the classification of orthopedic patients.

```
#The logistic regression approach
#Begin creating the algorithm
glm <- glm(class~., data=train, family="binomial")

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

pred_glm <- predict(glm, newdata=test, type="response")
pred <- prediction (pred_glm,test$class)

#Determine the accuracy
accuracy <- as.numeric(performance(pred,"auc")@y.values)
accuracy

## [1] 0.9758491
```

Here, we see that the logistic regression approach produces a very strong accuracy of 97.6%.

We can further analyze this approach by creating an ROC curve. An ROC curve is a performance measurement for classification problems at various threshold settings. In this case, the ROC is a probility curve that represents how well the model can distinguish between classes.

```
#Let's look at a ROC plot for logistic regression
roc.plot <- performance(pred,"tpr","fpr")
plot(roc.plot, colorize=TRUE)
```

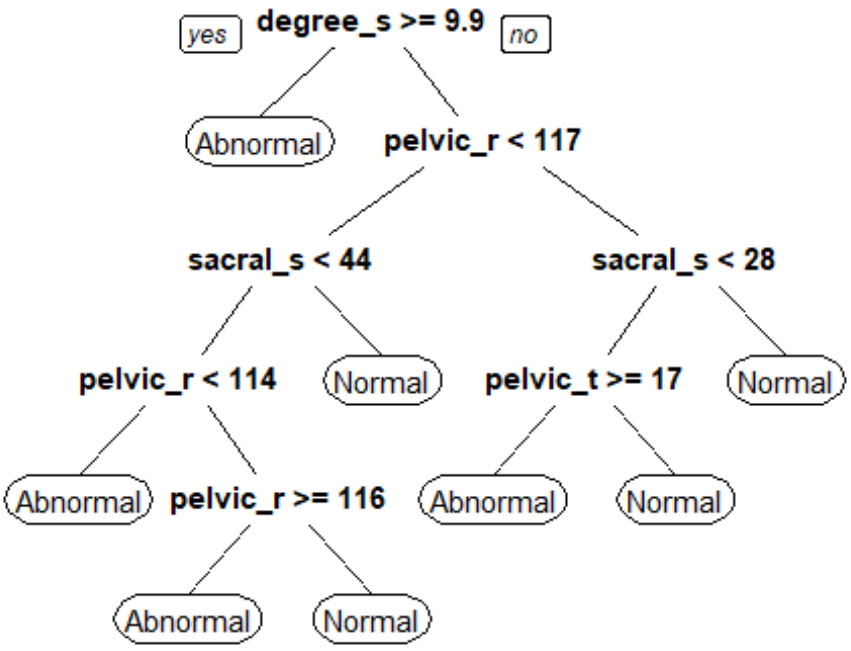


This plot displays a very low occurence of false positives. Therefore, we are able to state that this method has strong ability to distinguish between the two classes - normal and abnormal. Thus, logistic regression has a very high accuracy, and strong capability to distinguish between classes.

Algorithm 3 - Random Forests

Classification Tree Before creating the Random Forests machine learning algorithm, we will create a classification tree from this dataset. This will enable us to see what the method of random forests is all about.

```
#Let's start with the Classification Tree Method
#Install the necessary packages
library(rpart)
library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 3.5.3

#Create the classification tree
rpart_1 <- rpart(class~., data=train, method="class")
prp(rpart_1)
```

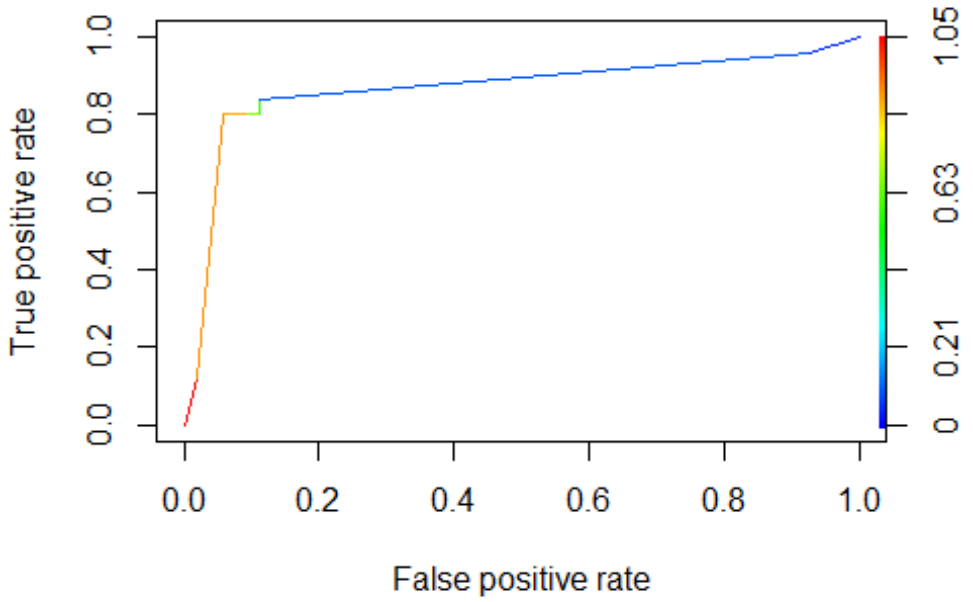


Here, we see that the classification tree is very useful in relating each of the variables and how they are correlated with the ultimate classification of the patient. Next, we must determine the accuracy to decide how successful this method will be.

```
#Determine the accuracy of this method
predict_rpart_1 <- predict(rpart_1, newdata = test, type="prob")
roc_tree <- prediction(predict_rpart_1[,2],test$class)
accuracy <- as.numeric(performance(roc_tree,"auc")@y.values)
accuracy

## [1] 0.8679245
```

The classification tree method produces an accuracy of 86.8%. This is not an awful result, but is not quite as high as the accuracies produced in kNN or logistic regression. Next, let's examine it's ability to distinguishes between the two classes.

```
#Let's consider the ROC plot for the Classification Tree Method
roc.plot_tree <- performance(roc_tree,"tpr","fpr")
plot(roc.plot_tree, colorize=TRUE)
```



Here, we see that the classification tree does not have too many false positives, but definitely has more than what was shown for logistic regression. Thus, this will not be considered the best method for predicting orthopedic patient classifications.

Random Forests Now that we saw the deficits of the classification tree method, we will impliment random forests to make up for those drawbacks. This method will improve performacne by averaging many classification trees. This estimate will be smoothed by changing the parameter that control the minimum number of data points in the nodes of the tree. Note that the randomForest and Rborist packages are needed.

```r
#Random Forests Algorithm
#Develop the algorithm
#Note that this code may take a few minutes to run
train_rf <- train(class ~ .,
                  method = "Rborist",
                  tuneGrid = data.frame(predFixed = 2, minNode = c(3,
125)),
                  data = train)

#Determine the accuracy
confusionMatrix(predict(train_rf, test), test$class)$overall["Accuracy"]
```

```
##  Accuracy
## 0.9230769
```

The random forests algorithm produces an accuracy of 91.0%. Although this is an improvement from the simple classification tree, this accuracy is still lower than that of the other methods.

Conclusion

Throughout this project, three machine learning algorithms were successfully created to predict the classification (normal or abnormal) of orthopedic patients. Although the random forests method provides clear and easy results to understand, it produced the lowest accuracy. The kNN approach was optimized by setting k=9 and achieved an accuracy of 93.6%. However, logistic regression achieved the highest accuracy of 97.6%, and proved strong ability to distinguish between two classes with a low false positive rate. Thus, logistic regression was proved to be the best machine learning algorithm for predicting the classification of orthopedic patients.