IDV Learner Project

Luke Douglas

Introduction

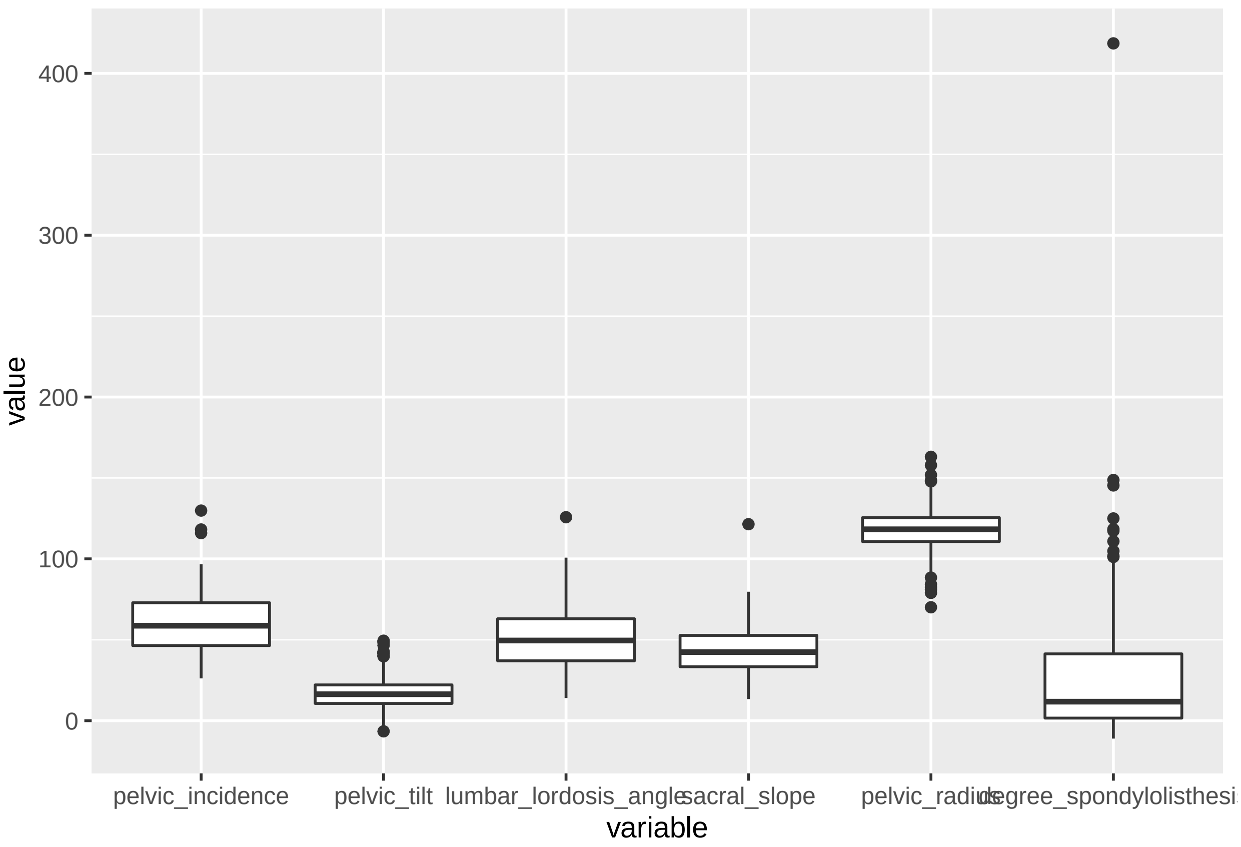
Looking at the suggested data sets linked in the IDV Learners 'Project Overview', the 'Biomechanical Features of Orthopedic Patients' data set (https://www.kaggle.com/uciml/biomechanical-features-of-orthopedic-patients) in the linked Kaggle 'curated list' seemed interesting. Given the sometimes high costs of medical care and the high level of importance of good treatment for patients, was there a space to help reduce diagnostic effort by providers while also decreasing the chance of error? Classification problems like this are perfect for machine learning.

The 'Biomechanical Features of Orthopedic Patients' data set exists in two versions, one classifying "patients as belonging to one out of three categories: Normal (100 patients), Disk Hernia (60 patients) or Spondylolisthesis (150 patients)" (Kaggle) while the other has only two categories, where "Disk Hernia and Spondylolisthesis were merged into a single category labelled as 'abnormal' ". (Kaggle) The 3 category data set was selected for this project as it seemed more interesting to work with.

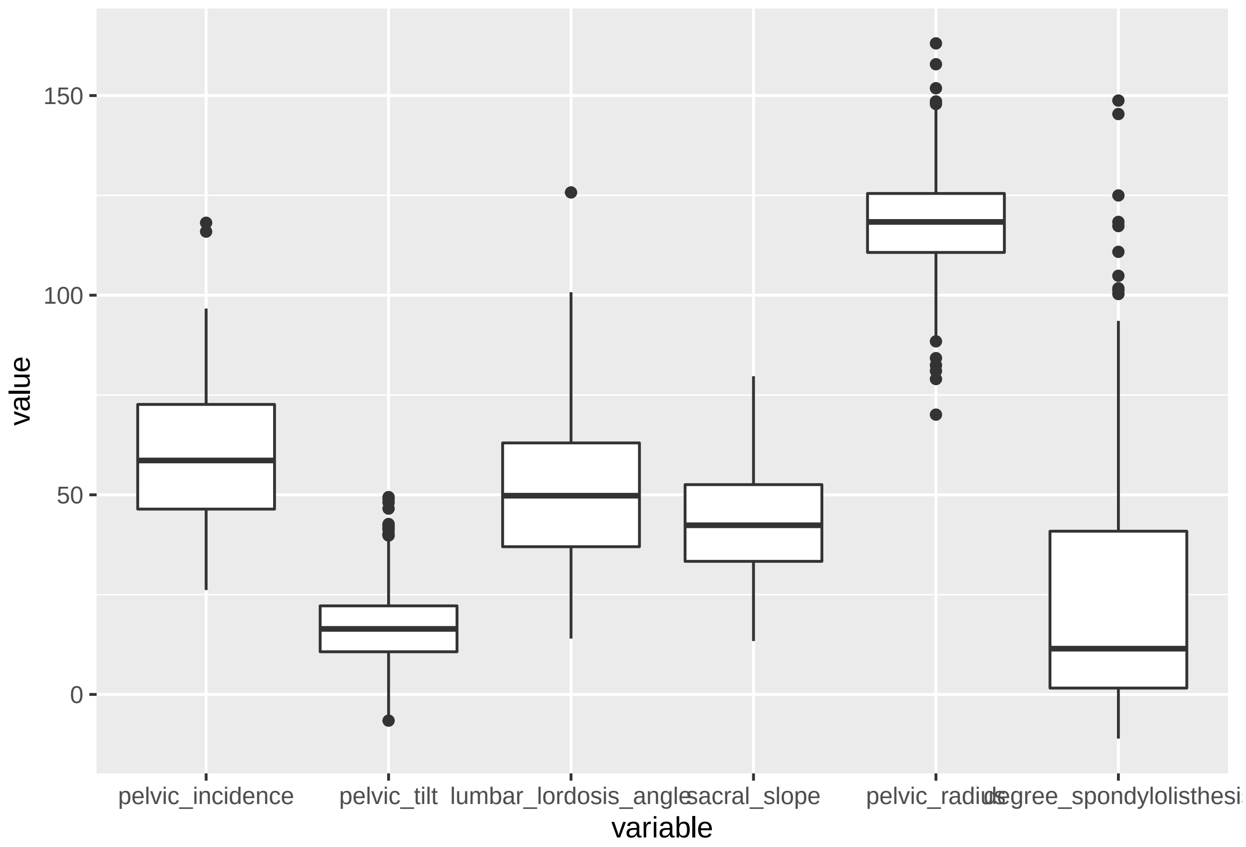
Four classification models were built on a training data set: 1) using the average, 2) a multinomial logistic regression, 3) a k-nearest neighbors model, and 4) a random forest. Testing these models against a test set showed similar accuracy between the multinomial logistic regression, the k-nearest neighbors, and the random forest. Given the context of medical diagnosis, how this accuracy is achieved is also important and a detailed examination of the confusion matricies of each model shows the random forest to be the most appropriate classifier for orthopedic patient diagnostics.

Methods

The first step is to input the data. Minimal structural cleaning was needed other than setting the dependent variable as a factor rather than a character string. Looking at a boxplot of each variable highlights that there is a potential error 'degree_spondylolisthesis' where the value is greater than 400. If these are degrees, 400 represents over a full rotation - which seems impossible. Perhaps someone misplaced a decimal?



Filtering out this one row and building a boxplot again shows data that looks much better.



Next we create training and test sets; setting aside 10% of the data into a test set for use in the 'Results' section.

```
# create training and test sets
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = raw_clean$class, times = 1, p = 0.1, list = FALSE)
train_set <- raw_clean[-test_index,]
test_set <- raw_clean[test_index,]
```

Looking at the training set, one sees there are 278 observations of 6 independent variables (the biomechanical features) and the one dependent variable of 'class', the patient diagnosis. Quick summary stats show a large amount of variation within each feature, which bodes well for being able to use them to determine patient diagnoses.

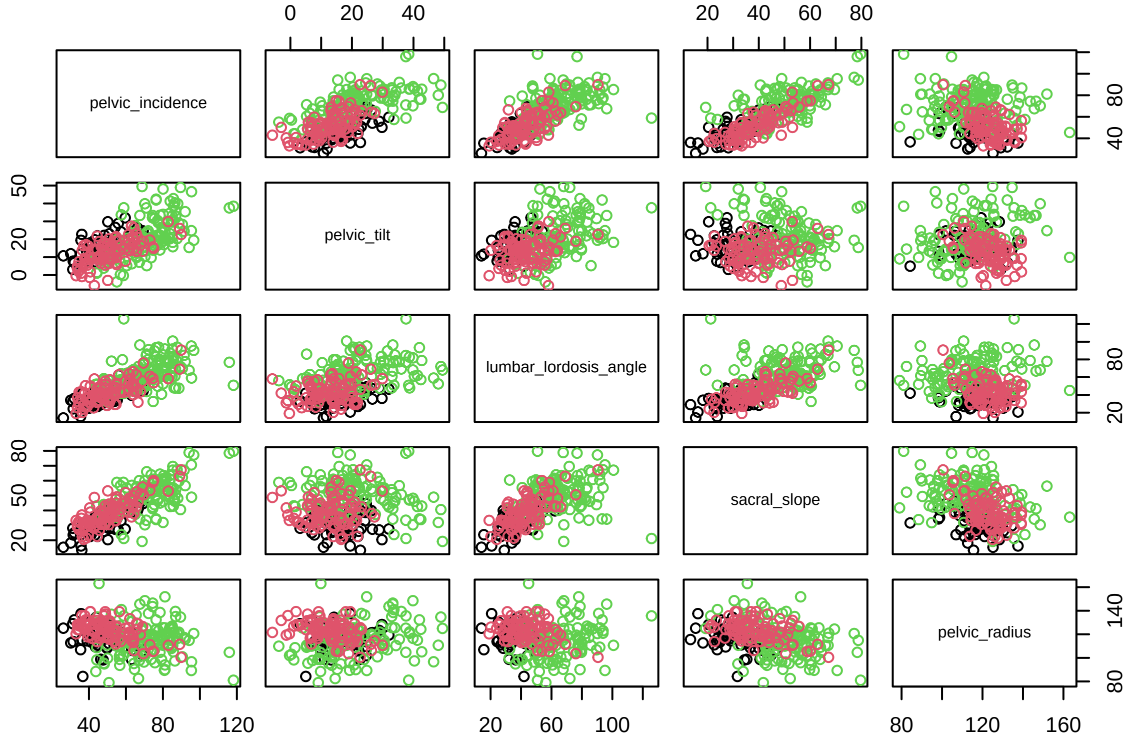
```
# display some summary statistics of the training set
str(train_set)
```

```
## 'data.frame':    278 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                   : Factor w/ 3 levels "Hernia","Normal",..: 1 1 1 1 1 1 1 1 1 1 ...
```

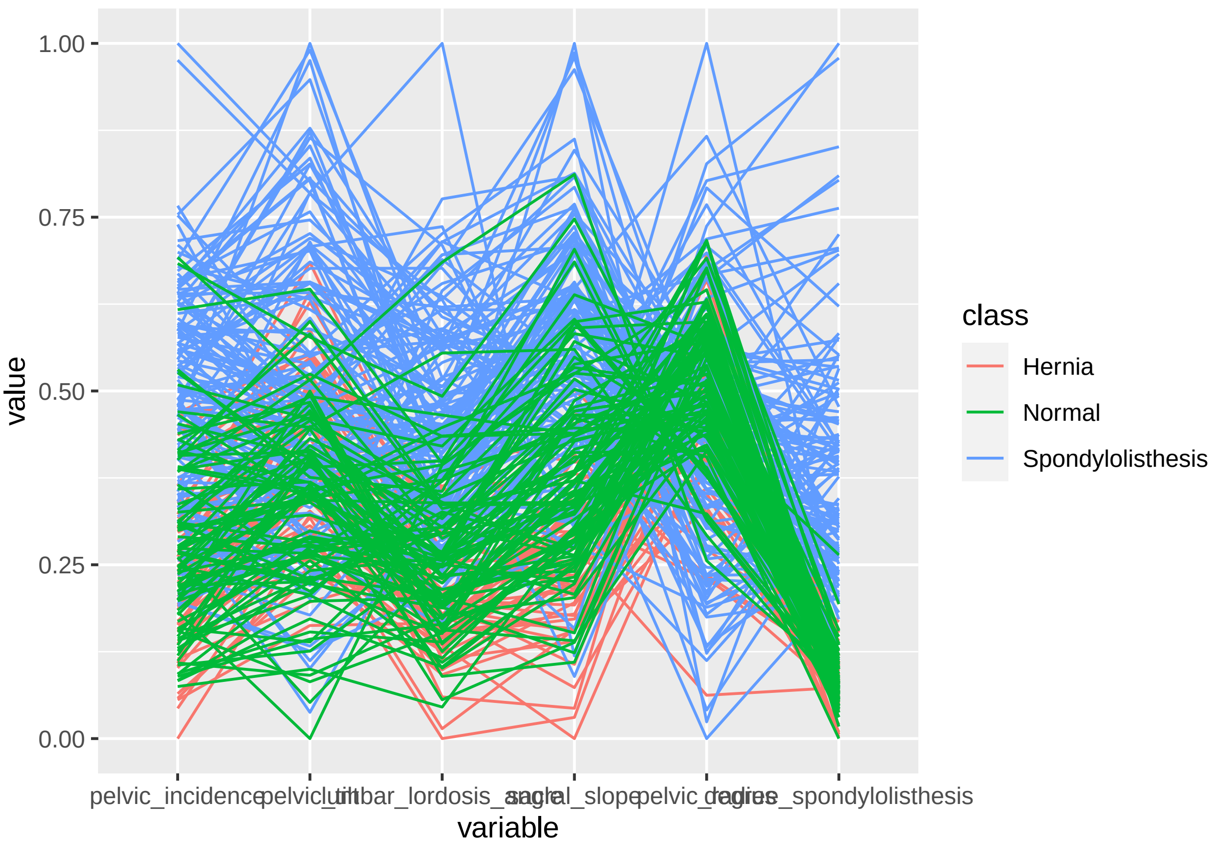
```
summary(train_set)
```

```
##  pelvic_incidence  pelvic_tilt      lumbar_lordosis_angle  sacral_slope
##  Min.   : 26.15   Min.   :-5.846    Min.   : 14.00        Min.   :13.37
##  1st Qu.: 47.40   1st Qu.:10.684    1st Qu.: 37.00        1st Qu.:33.43
##  Median : 58.56   Median :16.485    Median : 49.56        Median :42.40
##  Mean   : 60.62   Mean   :17.717    Mean   : 52.16        Mean   :42.90
##  3rd Qu.: 73.47   3rd Qu.:22.382    3rd Qu.: 63.01        3rd Qu.:52.48
##  Max.   :118.14   Max.   :49.432    Max.   :125.74        Max.   :79.70
##  pelvic_radius   degree_spondylolisthesis                class
##  Min.   : 79.0   Min.   :-11.058          Hernia           : 54
##  1st Qu.:110.9   1st Qu.:  1.485          Normal           : 90
##  Median :118.0   Median : 11.337          Spondylolisthesis:134
##  Mean   :117.8   Mean   : 25.180
##  3rd Qu.:125.4   3rd Qu.: 41.287
##  Max.   :163.1   Max.   :148.754
```

Visualizing the data can also give a sense for how successful a machine learning model may be. Looking at a quick scatter plot matrix of the independent variables that is colored by dependent variable shows some separation by class - a good sign!



With a parallel coordinates plot, there also seems to be some interesting grouping by class - individuals with Spondylolisthesis look to generally have higher values across all features while it looks like those with hernias tend to have lower amounts of lumbar lordoisis angle and sacral slope.



Setting baselines: Using the mode and multinominal logistic regression

For the effort of training machine learning models to be worthwhile, they should at least beat simple prediction methods in terms of accuracy. Here two simple approaches are used to provide this 'baseline': 1) the most frequent class in the data set (i.e. the 'mode') and 2) a multinominal logistic regression. Since these are both statistical approaches rather than machine learning ones, we can check the validity of each on the training set.

Mode

Looking at the number of observations of each class in the training set, Spondylolisthesis is the mode. Validating this 'model' by using it predict class on our training set shows it is not very accurate even on the training data, so hopes are not high it will do well on the test data...

```
# use the mode dependent variable in the training set as the predictor in the test set
train_freq <- train_set %>% count(class) %>% arrange(desc(n)) #find the mode
train_mu <- train_set %>% mutate(mu = train_freq[1,1])
confusionMatrix(train_mu$mu, train_set$class)$overall["Accuracy"]
```

```
##  Accuracy
## 0.4820144
```

Regression

Maybe a regression can improve on using the mode? Since the dependent variable 'class' is a un-ordered categorical variable (i.e. there isn't a ranked order to the three classes), multinomial logistic regression is the best option for predicting predict class. Validating this model against the training set shows a much higher accuracy - excellent!

```
# build a multinominal logistic regression model using the training set
# to predict the test set
train_multinom <- multinom(class ~ . , data = train_set)
```

```
## # weights:  24 (14 variable)
## initial  value 305.414216
## iter  10 value 154.832864
## iter  20 value 80.527559
## iter  30 value 78.736688
## iter  40 value 78.702559
## iter  50 value 78.686675
## final  value 78.686201
## converged
```

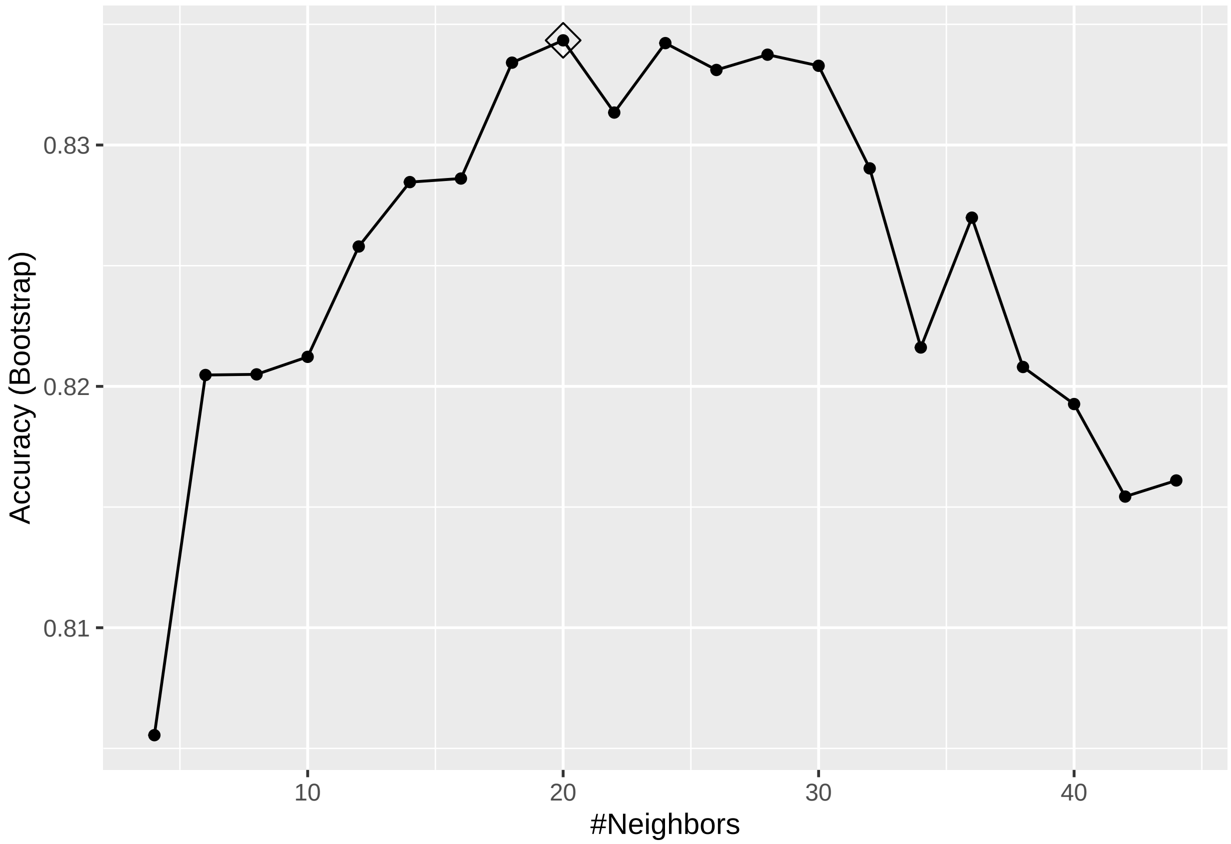
```
confusionMatrix(predict(train_multinom, train_set), train_set$class)$overall["Accuracy"]
```

```
##  Accuracy
## 0.8776978
```

k-nearest neighbors (kNN) algorithm

Since this is a classification problem, k-nearest neighbors (kNN) is one potential algorithm that might work. We can use 'train' from the caret package with the method 'knn' to do this. Passing the algorithm a large-ish set of potential ks (from 4 to 44, using only the 'evens' to avoid a tie since there's an odd number of classes) lets the package use crossvalidation and select the best k across bootstrapped samples. Plotting this set of ks shows the best tune for the data (given a seed of 1 just before running the algorithm).

```
# fit a knn model to the training set
set.seed(1, sample.kind="Rounding")
train_knn <- train(class ~ ., method = "knn",
                   data = train_set,
                   tuneGrid = data.frame(k = seq(4, 44, 2)))
ggplot(train_knn, highlight = TRUE)
```



```
# display the model
train_knn
```

```
## k-Nearest Neighbors
##
## 278 samples
##   6 predictor
##   3 classes: 'Hernia', 'Normal', 'Spondylolisthesis'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 278, 278, 278, 278, 278, 278, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    4  0.8055487  0.6866598
##    6  0.8204710  0.7106017
##    8  0.8204976  0.7094964
##   10  0.8212238  0.7102629
##   12  0.8257957  0.7168386
##   14  0.8284630  0.7216703
##   16  0.8286117  0.7217337
##   18  0.8334120  0.7293186
##   20  0.8343369  0.7307768
##   22  0.8313449  0.7257376
##   24  0.8342224  0.7304608
##   26  0.8331099  0.7284161
##   28  0.8337432  0.7296579
##   30  0.8332829  0.7282954
##   32  0.8290330  0.7210330
##   34  0.8216114  0.7087428
##   36  0.8269909  0.7174271
##   38  0.8208032  0.7071841
##   40  0.8192671  0.7044535
##   42  0.8154333  0.6979118
##   44  0.8161016  0.6987121
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 20.
```

This gives a maximum accuracy of:

```
# display the maximum accuracy
max(train_knn$results$Accuracy)
```

```
## [1] 0.8343369
```

Random forest

Another classification algorithm to try is a random forest - here we again use train() from the caret package to train a model on the training set.

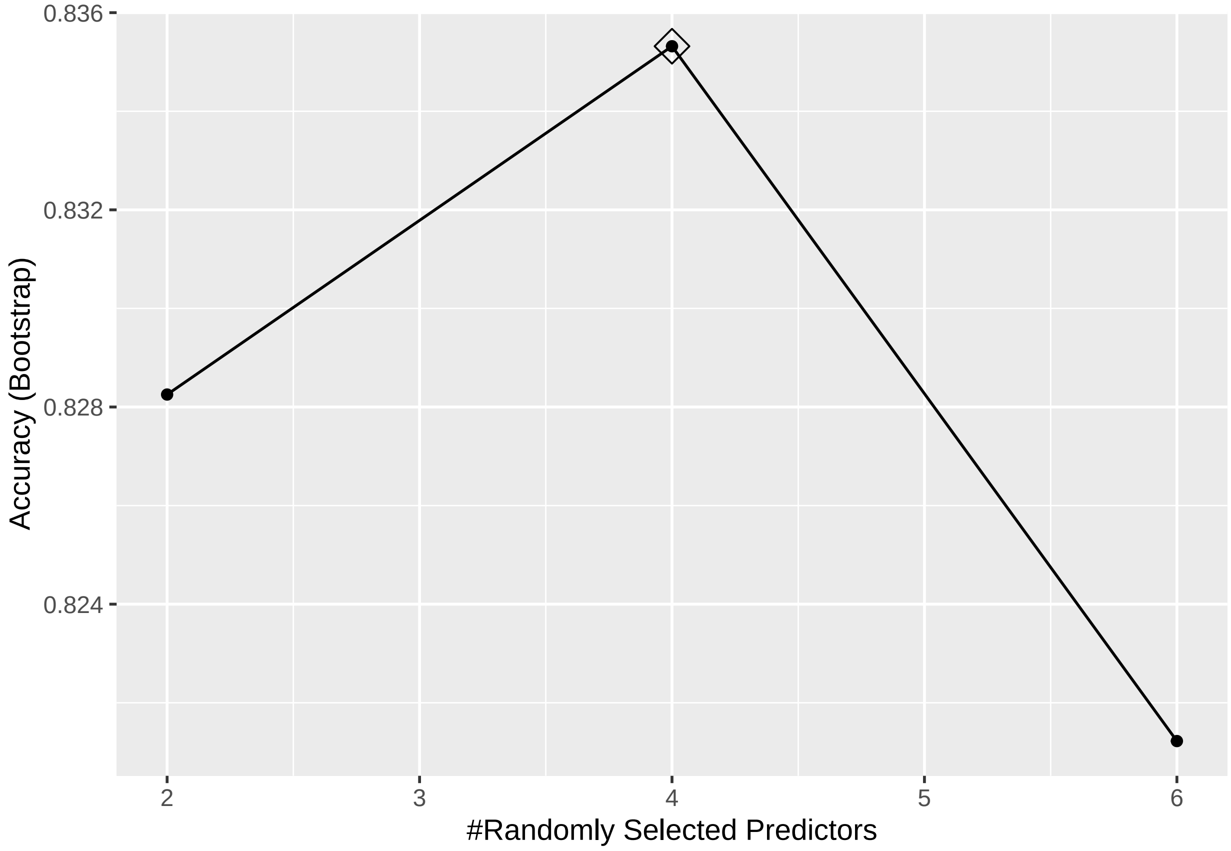
Given the small number of variables, there isn't much space to tune the random forest with different numbers of randomly sampled variables for each tree (i.e. 'mtry'). Graphing the accuracy of the model versus the number of randomly selected predictors shows which 'mtry' gives the maximum model accuracy.

```
# fit a random forest model to the training set
set.seed(1, sample.kind="Rounding")
train_rf <- train(class ~., method = "rf", data = train_set)

train_rf
```

```
## Random Forest
##
## 278 samples
##   6 predictor
##   3 classes: 'Hernia', 'Normal', 'Spondylolisthesis'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 278, 278, 278, 278, 278, 278, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.8282529  0.7208022
##   4     0.8353187  0.7331540
##   6     0.8212233  0.7108859
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 4.
```

```
ggplot(train_rf, highlight = TRUE)
```



```
max(train_rf$results$Accuracy)
```

```
## [1] 0.8353187
```

Given the small number of variables, there isn't much space to tune the random forest with different numbers of randomly sampled variables for each tree. Graphing the accuracy of the model versus the number of randomly selected predictors shows the maximum accuracy is achieved with an mtry of four.

Results

Now that we've built our models (mode / regression / knn / random forest), let's see how they do. Applying each model against the test set shows they differ in accuracy.

```
# use the mode in the training set to predict the test set and add the results to the table
test_mu <- test_set %>% mutate(mu = train_freq[1,1])
cm_mode <- confusionMatrix(test_mu$mu, test_set$class)
results_table[2,2] <-  cm_mode$overall["Accuracy"]

# test the regression / "Nominal Logistic Regression" and add the results to the table
cm_regression <- confusionMatrix(predict(train_multinom, test_set), test_set$class)
results_table[2,3] <- cm_regression$overall["Accuracy"]
```

```
# test knn and add the results to the table
cm_knn <- confusionMatrix(predict(train_knn, test_set), test_set$class)
results_table[2,4] <- cm_knn$overall["Accuracy"]

# test random forest and add the results to the table
cm_rf <- confusionMatrix(predict(train_rf, test_set), test_set$class)
results_table[2,5] <- cm_rf$overall["Accuracy"]

results_table
```

```
## # A tibble: 2 x 5
##   set    mode regression   knn random_forest
##   <chr> <dbl>      <dbl> <dbl>         <dbl>
## 1 train 0.482      0.878 0.834         0.835
## 2 test  0.484      0.806 0.806         0.806
```

With this data, the accuracy of the models looks the same! But that's not the end of the story, as accuracy can hide important nuance, especially since these models are supposed to be used in a medical context. Setting aside the mode-based model since it's clearly inferior, let's look at the tables for the other three confusion matrices to make sure this accuracy is actually helpful in a medical context.

```
# look at the confusion matrix tables for the three 'high-performing' models
cm_regression$table
```

```
##                      Reference
## Prediction            Hernia Normal Spondylolisthesis
##   Hernia                   3      1                 1
##   Normal                   3      8                 0
##   Spondylolisthesis        0      1                14
```

```
cm_knn$table
```

```
##                      Reference
## Prediction            Hernia Normal Spondylolisthesis
##   Hernia                   3      1                 0
##   Normal                   3      8                 1
##   Spondylolisthesis        0      1                14
```

```
cm_rf$table
```

```
##                      Reference
## Prediction            Hernia Normal Spondylolisthesis
##   Hernia                   4      2                 0
##   Normal                   2      7                 1
##   Spondylolisthesis        0      1                14
```

Looking at this by diagnosis:

- Hernia: The random forest is best, correctly diagnosing four hernias and with only two false positives (seeing a hernia where there is none). Knn has three correct and one false positive. The regression also has three correct and one false positive. Worryingly, the regression also completely misdiagnoses a case of spondylolisthesis as a hernia - the only instance with a complete confusion between two non-normal conditions.

- Normal: The regression and kNN both have eight correct 'normal' diagnoses for 'normal' patients, while the random forest has 7. But the regression would also let three people with hearnias walk away thinking they're 'normal' and knn would do similarly for four people (three hernias and one spondylolisthesis - while the random forest would give three mistaken diagnoses of normal (two hernias and one spondylolisthesis).
- Spondylolisthesis: All three models correctly identifying 14 cases of spondylolisthesis. All three models have one 'false positive' of seeing spondylolisthesis where there is none.

Conclusion

Given the above investigation, it seems like the random forest is the best model to use to support providers in assessing orthopedic patients. It is both more accurate overall, more conservative (in that it lets fewer 'Normals' through), and has zero cases of complete misdiagnosis of non-normal conditions (e.g. confusing a hernia with spondylolisthesis).

To continue investigating 'Biomechanical Features of Orthopedic Patients', a good next step would be to increase the pool of data - either gathering more observations or adding additional independent variables. Doing either could help create greater differences in model performance and would enable deep investigations into different biomechanical features of hernias and spondylolisthesis.