CYO - Fertility Report

Yemi Akinwale

11/8/2020

I. INTRODUCTION

Machine learning is building of models or algorithms to make predictions using data. One or more features or predictors are processed from a series of observations to predict an outcome.

The outcome can be divided into categorical and continuous. A categorical outcome is referred to as classification, while a continuous outcome is prediction.

In this project, the outcome is classification. The data set is semen sample of 100 volunteers, analyzed according to WHO 2010 criteria. Sperm concentrations are related to socio-demographic data, environmental factors, health status, and life habits. All these factors affect semen and are considered in determining the fertility of the semen to either remain normal or altered. There are nine (9) of these factors considered in these observations.

Executive summary

The goal of this project is to make classification of the semen fertility to either be Normal or Altered. All the predictors will be used in determining the classification. We are to make classification for fertility diagnosis Three different algorithms (linear, non-linear and advanced non-linear) are used. The purpose is to study how each algorithm performed in training, predicting and accuracy.

II. METHODS/ANALYSIS

The following three (3) algorithms were used having considered 1.) The size of the data set 2.) The numbers of predictors (bias) and 3.)The classification outcome. These algorithms have features to work better with this set of conditions.

1. Linear Discriminant Analysis (lda): This is a simple linear algorithm. It has high bias and low variance.
2. Kernel Nearest Neighbors (knn): Knn is a non linear algorithm with low bias and high variance. It leverage on average of similar predictors considered as neighbors
3. Random Forest (rf): Random forest is complex non linear method. It improve performance and reduces instability by averaging multiple trees (samples) constructed with randomness

Overall accuracy as it is the best metric for classification problems is used to rate the algorithms performance.

Process and techniques of the analysis

1. Install all necessary packages

Packages are automatically installed with if(!require) code.

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse

## -- Attaching packages ------------------------------------------ tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.1
## v tidyr   1.1.1     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts --------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: data.table

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose
```

```
library(tidyverse)
library(caret)
library(data.table)
library(readr)
```

2. Download and load dataset

The data is obtained from UCI machine learning repository. The data is clean with complete inputs for all attributes.

The data is downloaded using the readr package, saved as fertility and the columns renamed appropriately.

2.1 The URL to download the data

```
url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/00244/fertility_Diagnosis.txt"
```

2.2 Load the data and name as fertility

```
fertility <- read_csv(url, col_names = FALSE)
```

```
## Parsed with column specification:
## cols(
##   X1 = col_double(),
##   X2 = col_double(),
##   X3 = col_double(),
##   X4 = col_double(),
##   X5 = col_double(),
##   X6 = col_double(),
##   X7 = col_double(),
##   X8 = col_double(),
##   X9 = col_double(),
##   X10 = col_character()
## )
```

2.3 set the column names

```
colnames(fertility) <- c("Season", "Age", "Diseases", "Accident", "Surgical", "Fevers", "Alcohol", "Smol
```

3. Attributes Information and names:

The followings are definition of attributes used in the dataset.

1.Season - Season in which the analysis was performed. 1) winter, 2) spring, 3) Summer, 4) fall. (-1, -0.33, 0.33, 1)

2.Age - Age at the time of analysis. 18-36 (0, 1)

3.Diseases - Childish diseases (ie , chicken pox, measles, mumps, polio) 1) yes, 2) no. (0, 1)

4.Accident - Accident or serious trauma 1) yes, 2) no. (0, 1)

5.Surgical - Surgical intervention 1) yes, 2) no. (0, 1)

6.Fevers - High fevers in the last year 1) less than three months ago, 2) more than three months ago, 3) no. (-1, 0, 1)

7.Alcohol - Frequency of alcohol consumption 1) several times a day, 2) every day, 3) several times a week, 4) once a week, 5) hardly ever or never (0, 1)

8.Smoking - Smoking habit 1) never, 2) occasional 3) daily. (-1, 0, 1)

9.Sitting - Number of hours spent sitting per day - 16 (0, 1)

10.Diagnosis - This is the Output. The Diagnosis can either be normal (N) or altered (O)

Each attributes has different classes which are represented numerically. For example, Age, is the age of the volunteer as at the time of the analysis. The age bracket used is 18 to 36. This was further represented as 0 to 1 with age 18 as 0.5 and 36 as 1. Similarly, winter as season is represented as -1.

The regression for this categorical outcome can be presented thus:

Diagnosis(y) = Season + Age + Diseases + Accident + Surgical + Fevers + Alcohol + Smoking + Sitting

4. Summarize data set

4.1 Dimensions of data set

The dimension of the data is 100 rows by 10 columns.

```r
dim(fertility)
```

```
## [1] 100  10
```

4.2 Types of attributes

This shows the class of each attribute in the data set.

```r
sapply(fertility, class)
```

```
##      Season         Age    Diseases    Accident    Surgical      Fevers
##   "numeric"   "numeric"   "numeric"   "numeric"   "numeric"   "numeric"
##     Alcohol     Smoking     Sitting   Diagnosis
##   "numeric"   "numeric"   "numeric" "character"
```

4.3 Glimpse at the data

A glimpse of the data set shows first few rows and columns.

```r
glimpse(fertility)
```

```
## Rows: 100
## Columns: 10
## $ Season    <dbl> -0.33, -0.33, -0.33, -0.33, -0.33, -0.33, -0.33, -0.33, 1...
## $ Age       <dbl> 0.69, 0.94, 0.50, 0.75, 0.67, 0.67, 0.67, 1.00, 0.64, 0.6...
## $ Diseases  <dbl> 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ Accident  <dbl> 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, ...
## $ Surgical  <dbl> 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, ...
## $ Fevers    <dbl> 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0...
## $ Alcohol   <dbl> 0.8, 0.8, 1.0, 1.0, 0.8, 0.8, 0.8, 0.6, 0.8, 1.0, 0.8, 0....
## $ Smoking   <dbl> 0, 1, -1, -1, -1, 0, -1, -1, -1, -1, 0, 0, 1, -1, -1, 1, ...
## $ Sitting   <dbl> 0.88, 0.31, 0.50, 0.38, 0.50, 0.50, 0.44, 0.38, 0.25, 0.2...
## $ Diagnosis <chr> "N", "O", "N", "N", "O", "N", "N", "N", "N", "N", "N", "N...
```

4.4 Number of instances in each attribute

This shows the distinctive number of each variable involved in the data set e.g Season has 4 – winter, spring, summer and fall.

```r
sapply(fertility, n_distinct)
```

```
##    Season       Age  Diseases  Accident  Surgical    Fevers   Alcohol   Smoking 
##         4        18         2         2         2         3         5         3 
##   Sitting Diagnosis 
##        14         2
```

4.5 Statistical summary of the attributes

The summary shows the statistical details like the mean, median of all the variables.

```r
summary(fertility)
```

```
##      Season              Age            Diseases       Accident       Surgical   
##  Min.   :-1.0000   Min.   :0.500   Min.   :0.00   Min.   :0.00   Min.   :0.00  
##  1st Qu.:-1.0000   1st Qu.:0.560   1st Qu.:1.00   1st Qu.:0.00   1st Qu.:0.00  
##  Median :-0.3300   Median :0.670   Median :1.00   Median :0.00   Median :1.00  
##  Mean   :-0.0789   Mean   :0.669   Mean   :0.87   Mean   :0.44   Mean   :0.51  
##  3rd Qu.: 1.0000   3rd Qu.:0.750   3rd Qu.:1.00   3rd Qu.:1.00   3rd Qu.:1.00  
##  Max.   : 1.0000   Max.   :1.000   Max.   :1.00   Max.   :1.00   Max.   :1.00  
##      Fevers          Alcohol          Smoking          Sitting      
##  Min.   :-1.00   Min.   :0.200   Min.   :-1.00   Min.   :0.0600  
##  1st Qu.: 0.00   1st Qu.:0.800   1st Qu.:-1.00   1st Qu.:0.2500  
##  Median : 0.00   Median :0.800   Median :-1.00   Median :0.3800  
##  Mean   : 0.19   Mean   :0.832   Mean   :-0.35   Mean   :0.4068
##  3rd Qu.: 1.00   3rd Qu.:1.000   3rd Qu.: 0.00   3rd Qu.:0.5000
##  Max.   : 1.00   Max.   :1.000   Max.   : 1.00   Max.   :1.0000
##   Diagnosis
##  Length:100
##  Class :character
##  Mode  :character
##
##
##
```

5. Visualization and graphic explanation of data set

5.1 split predictors and output

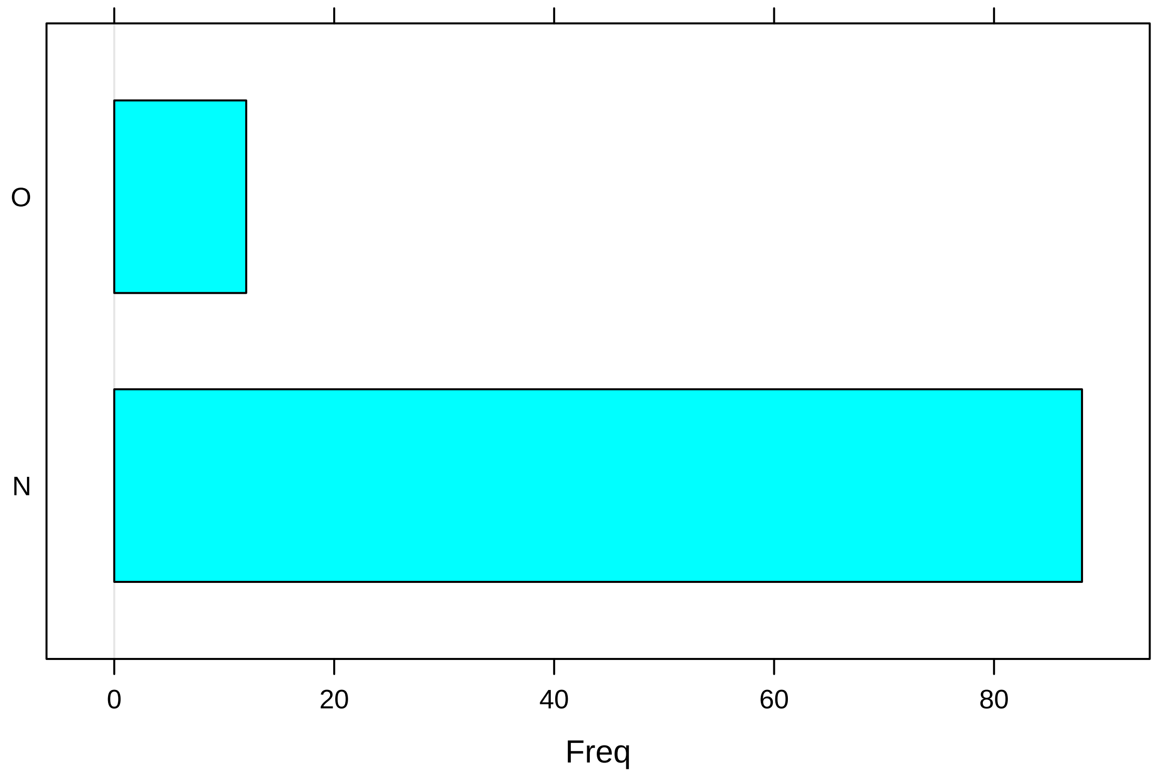
The data set – fertility is separated into predictors – x and the outcome – y

```
x <- fertility[,1:9]
y <- fertility[,10]
```

5.2 barplot to show the distribution of the class

The barchart visualizes the two outcomes – Normal (N) and Altered (O). There are far more Normal cases than Altered.

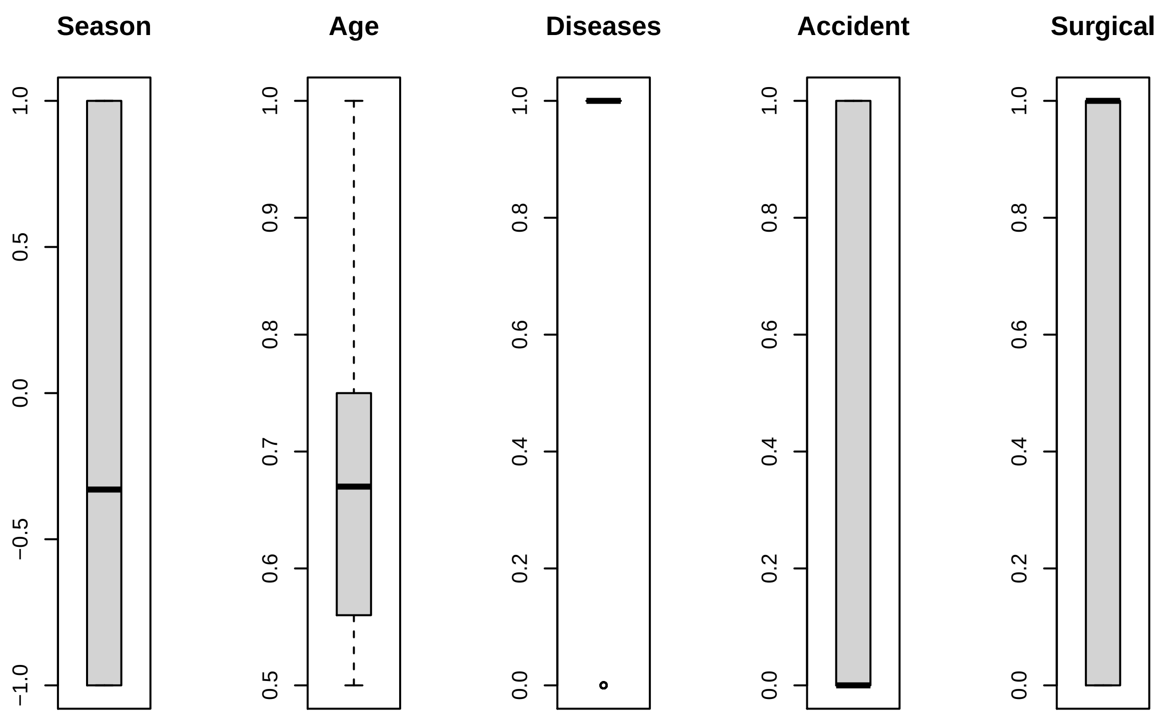
```
barchart(y)
```



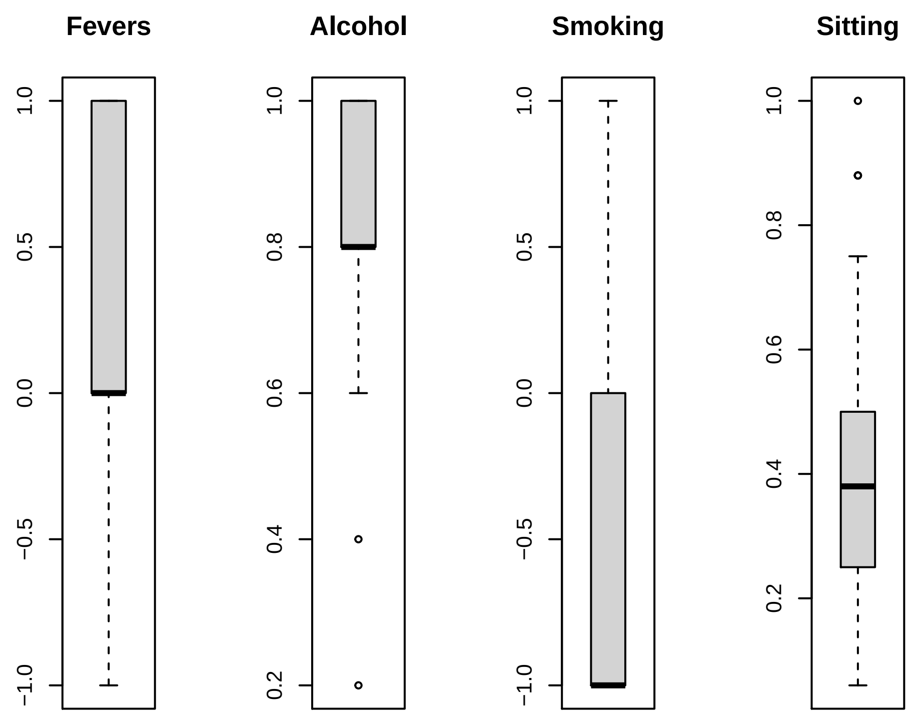
5.3 boxplot for each attribute on one image

Given that the inputs – predictors are numeric, a box plot of each predictor shows a clearer idea of the distribution of the predictors. The plot is separated into two because of space.

```r
par(mfrow=c(1,5))
   for(i in 1:5) {
     boxplot(x[,i], main = names(fertility)[i])
   }
```



```r
  for(i in 6:9) {
    boxplot(x[,i], main = names(fertility)[i])
}
```



6. Creation of Validation and Train sets from fertility

Using the createDataPartition function of caret package, the validation and train sets were created from the fertility data in ratio 2:8 respectively. The ratio is to enable the algorithms enough data to train and proportionate representation of the two outcomes in validation set.

```
set.seed(1, sample.kind="Rounding") # set the seed
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
test_index <- createDataPartition(y = fertility$Diagnosis, times = 1, p = 0.2, list = FALSE)
train <- fertility[-test_index,]
validation <- fertility[test_index,]
```

```
## Warning: The `i` argument of ``[`()` can't be a matrix as of tibble 3.0.0.
## Convert to a vector.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

7. Evaluate algorithms

An insight

This is a simple sampling of the outcome – classification. The accuracy which changes at every other rerun is just a guide to how better the algorithms could do.

```
y_hat <- sample(c("N", "O"), length(test_index), replace = TRUE)
mean(y_hat == validation$Diagnosis)
```

```
## [1] 0.5238095
```

The following 3 Algorithms will be used and overall accuracy will be used as metric.

7.1 Linear Discriminant Analysis (LDA) - Linear

fit the predictors in the train set

```
set.seed(1)
train.lda <- train(Diagnosis ~ ., data = train, method = "lda")
```

Predict from validation set

```
predict.lda <- predict(train.lda, validation)
```

Accuracy of classification from the validation set

```
Accuracy.lda <- mean(predict.lda == validation$Diagnosis)
Accuracy.lda
```

```
## [1] 0.8571429
```

7.2 Kernel Nearest Neighbors (kNN) - Non Linear

The fit, predict and accuracy of the knn

```
set.seed(1)
train.knn <- train(Diagnosis ~ ., data = train, method = "knn")
predict.knn <- predict(train.knn, validation)
Accuracy.knn <- mean(predict.knn == validation$Diagnosis)
Accuracy.knn
```

```
## [1] 0.8571429
```

7.3 Random Forest (rf) - Advanced

The fit, predict and accuracy of the rf

```
set.seed(1)
train.rf <- train(Diagnosis ~ ., data = train, method = "rf")
predict.rf <- predict(train.rf, validation)
Accuracy.rf <- mean(predict.rf == validation$Diagnosis)
Accuracy.rf
```

```
## [1] 0.8571429
```

III. RESULTS

8. Analysis of the Algorithms results

8.1 Analysis of each Algorithm

8.1.1 Linear Discriminant Analysis (LDA) - Linear

The train summary shows that the train set resamples the 79 data 25 repetitions. It has accuracy of 0.810277

```
print(train.lda)
```

```
## Linear Discriminant Analysis
##
## 79 samples
##  9 predictor
##  2 classes: 'N', 'O'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 79, 79, 79, 79, 79, 79, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.810277  -0.05620329
```

The Variable importance in the lda algorithm can be obtained as the coefficients of the predictors. For the analysis, Age is the most important variable.

```
train.lda$finalModel
```

```
## Call:
## lda(x, grouping = y)
##
## Prior probabilities of groups:
##         N         O
## 0.8860759 0.1139241
##
## Group means:
##       Season       Age  Diseases  Accident  Surgical    Fevers   Alcohol
## N -0.1131429 0.6630000 0.9142857 0.4857143 0.5571429 0.2142857 0.8371429
## O  0.4822222 0.7355556 0.8888889 0.2222222 0.5555556 0.2222222 0.7555556
##      Smoking   Sitting
## N -0.4142857 0.3904286
## O -0.2222222 0.3555556
##
## Coefficients of linear discriminants:
##                 LD1
## Season    0.7500631
## Age       4.6638377
## Diseases -0.0942088
## Accident -1.1774611
## Surgical -0.1717772
## Fevers    0.1579505
## Alcohol  -2.0032552
## Smoking   0.2788478
## Sitting   1.2565732
```

8.1.2 Kernel Nearest Neighbors (kNN) - Non Linear

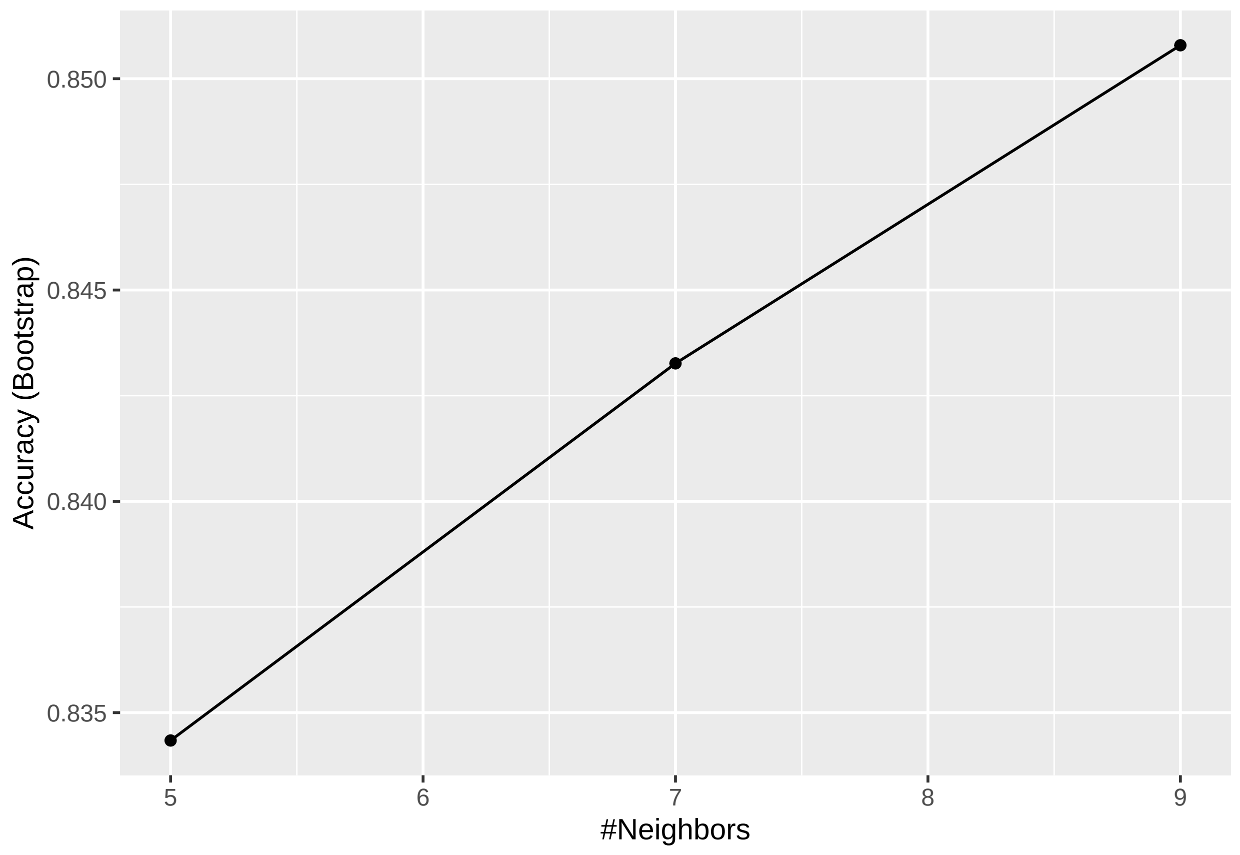
The train summary of the knn algorithms shows that the algorithm used 3 tuning parameters (k) to train and resample 25 times. The tuning parameter with the highest accuracy was selected to predict. k at 9 has the accuracy of 0.8508

```
print(train.knn)
```

```
## k-Nearest Neighbors
##
## 79 samples
##  9 predictor
##  2 classes: 'N', 'O'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 79, 79, 79, 79, 79, 79, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.8343408   0.07263221
##   7  0.8432642   0.03006773
##   9  0.8507912  -0.01123953
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```

Graphical presentation of the tuning parameters and the accuracy.

```
ggplot(train.knn)
```



8.1.3 Random Forest (rf) - Advanced

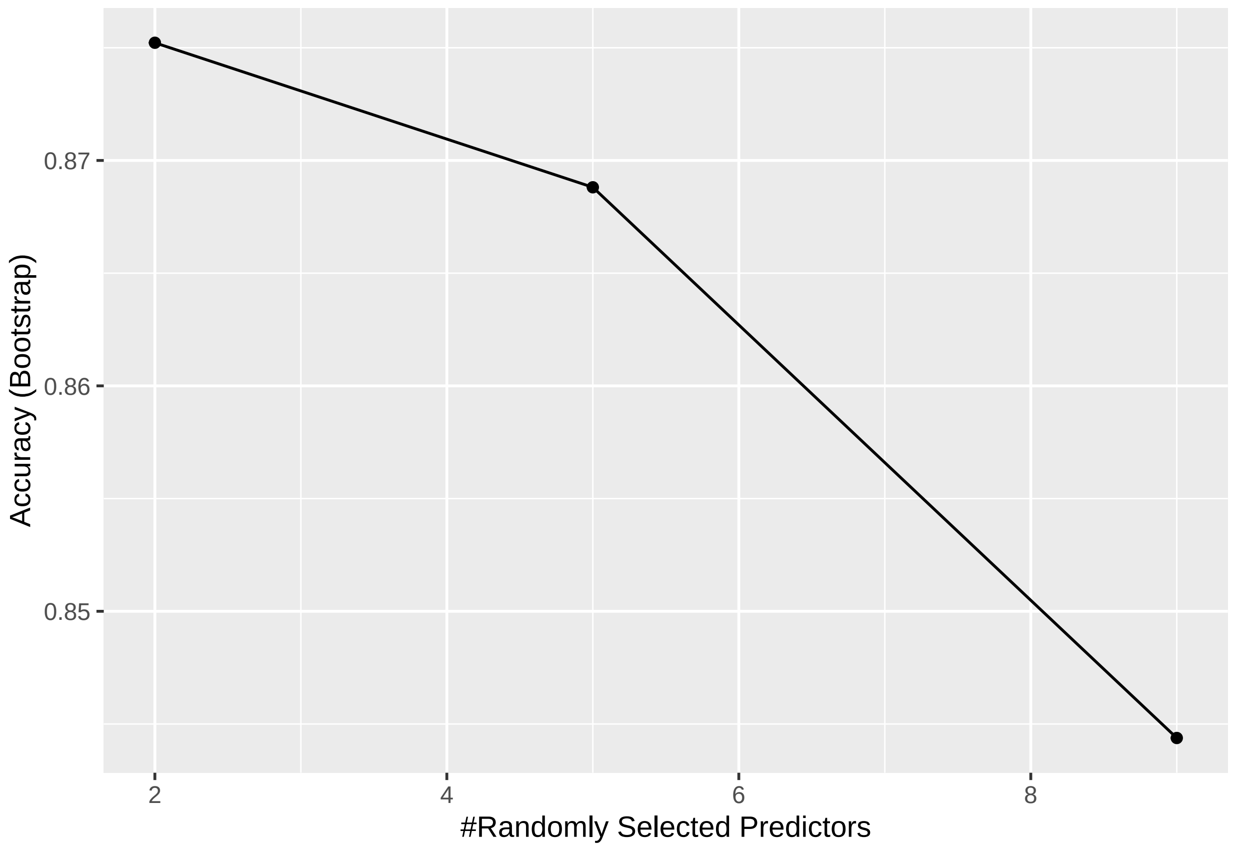
Similar to how knn use tuning parameters. Random forest tuning parameter called mtry, use the mtry 2 with the highest accuracy of 0.8752 to predict

```
print(train.rf)
```

```
## Random Forest
##
## 79 samples
##  9 predictor
##  2 classes: 'N', 'O'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 79, 79, 79, 79, 79, 79, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.8752221  0.008067088
##   5     0.8688105  0.094674672
##   9     0.8443792  0.087205370
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

This shows the graphical representation of the accuracy and mtry.

```
ggplot(train.rf)
```



The variable importance function of caret package can be used for random forest to get the importance of variables. Age is the most important predictor just as seen in lda analysis.

```
varImp(train.rf)
```

```
## rf variable importance
##
##          Overall
## Age       100.00
## Sitting    75.81
## Season     51.33
## Alcohol    42.37
## Smoking    30.90
## Accident   28.85
## Surgical   17.83
## Fevers     15.75
## Diseases    0.00
```

8.2 Comparison of Algorithms

8.2.1 Comparison of the training within the Algorithms

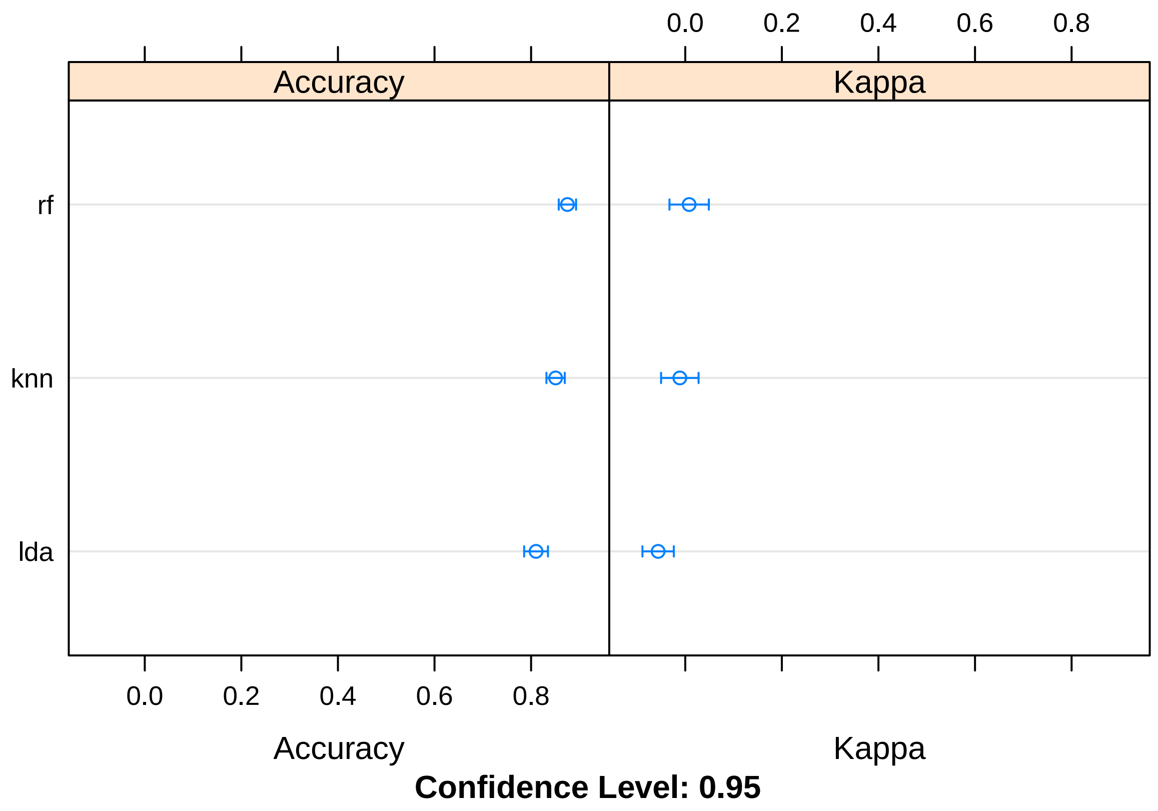
The random forest has the highest accuracy from the train summary (mean column). The plot below shows the spread and accuracy mean of the algorithms.

```r
results_train <- resamples(list(lda = train.lda, knn = train.knn, rf = train.rf))
```

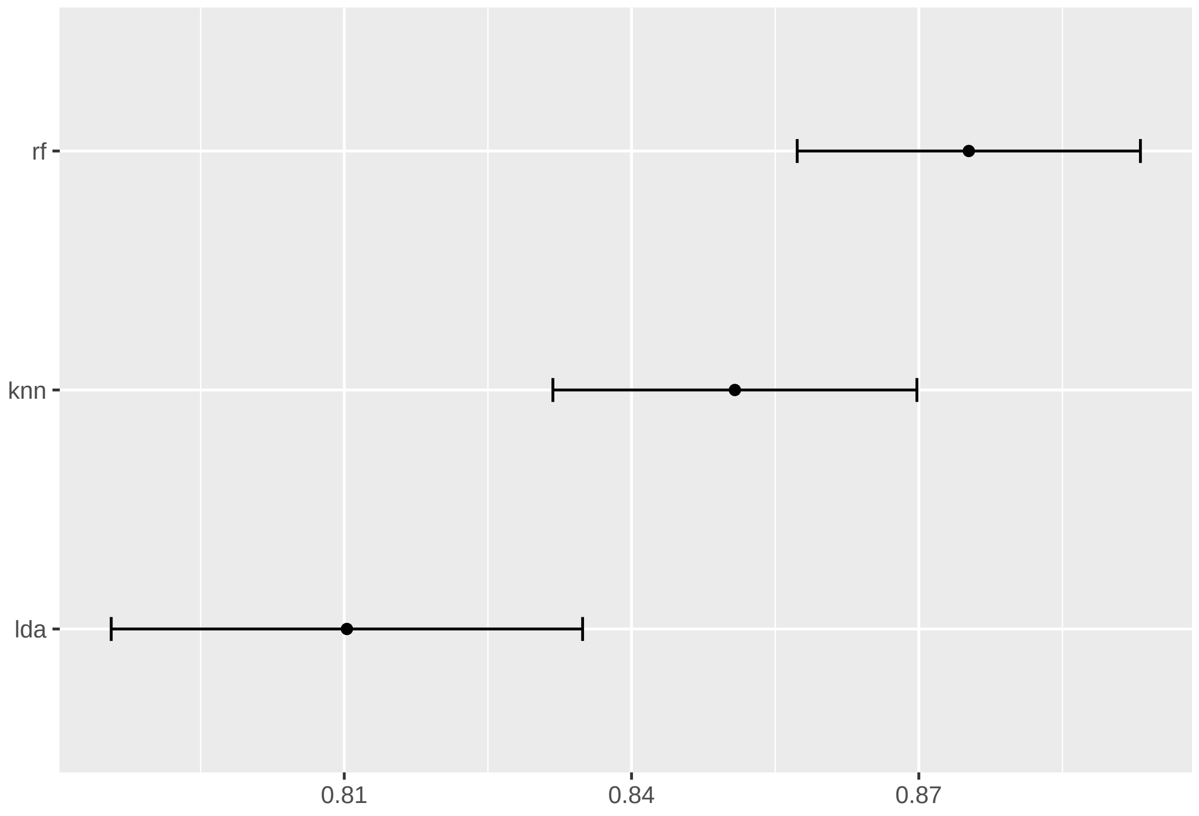
```r
summary(results_train)
```

```
##
## Call:
## summary.resamples(object = results_train)
##
## Models: lda, knn, rf
## Number of resamples: 25
##
## Accuracy
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lda 0.6521739 0.7812500 0.8214286 0.8102770 0.8484848 0.9090909    0
## knn 0.7777778 0.8148148 0.8571429 0.8507912 0.8846154 0.9354839    0
## rf  0.8000000 0.8437500 0.8846154 0.8752221 0.9032258 0.9666667    0
##
## Kappa
##            Min.     1st Qu.      Median         Mean 3rd Qu.      Max. NA's
## lda -0.19480519 -0.10526316 -0.06060606 -0.056203289       0 0.1860465    0
## knn -0.12500000 -0.05660377  0.00000000 -0.011239532       0 0.3636364    0
## rf  -0.09756098  0.00000000  0.00000000  0.008067088       0 0.4655172    0
```

```r
dotplot(results_train)
```



```r
ggplot(results_train)
```



8.2.2 Comparison of Algorithms Accuracy

All the algorithms have same overall accuracy. This is because of the small validation set.No cross validation was used because all the predictors have equal and complete inputs - no missing values.

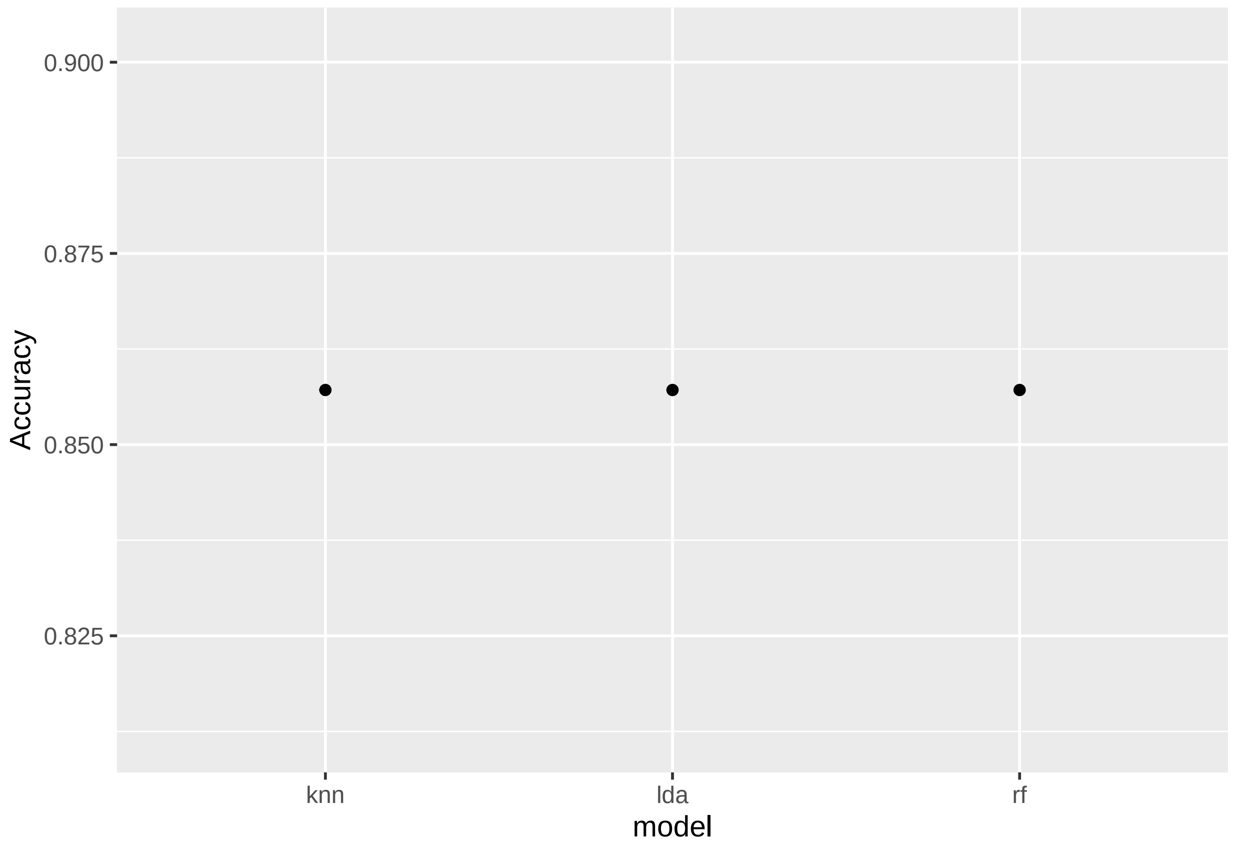
```
model <- c("lda", "knn", "rf")
```

```
Accuracy <- c(Accuracy.lda, Accuracy.knn, Accuracy.rf)
```

```
data.frame(model, Accuracy)
```

```
##   model  Accuracy
## 1   lda 0.8571429
## 2   knn 0.8571429
## 3    rf 0.8571429
```

```
qplot(model, Accuracy)
```



8.2.3 Confusion Matrix analysis

Sometimes, overall accuracy can be a deceptive measure due to unbalanced classes. A look at other metrics could help to determine how well an algorithm performed.

Like in this case, the classes are not balanced. Other metrics are analyzed below:

1. Specificity – is the proportion of actual negative outcomes that are correctly identified as such. The value is 0, which means there was no negative outcome identified as positive.
2. Sensitivity – also known as recall is the proportion of actual positive outcomes correctly identified as such. The recall value in this case is 1. This shows that all positive class "N" is correctly identified.
3. Prevalence – is the proportion the positive class in the outcome. It's like a check of the specificity accuracy. The prevalence is 0.8571 which is correct as the mean of positive outcome "N" in the validation set. Prevalence matters more in practice.
4. Balanced accuracy – is the harmonic average of specificity and sensitivity. The value is 0.5.
5. 95% CI - The confidence Interval (CI) is 64% lower and 97% upper. This is good as the 95% is within the confidence interval.

However, all the algorithms have the same values in the confusion matrix. The above analysis suffice for all the algorithms.

```r
confusionMatrix(predict.lda, factor(validation$Diagnosis))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  N  O
##          N 18  3
##          O  0  0
##
##                Accuracy : 0.8571
##                  95% CI : (0.6366, 0.9695)
##     No Information Rate : 0.8571
##     P-Value [Acc > NIR] : 0.6477
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 0.2482
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.8571
##          Neg Pred Value :    NaN
##              Prevalence : 0.8571
##          Detection Rate : 0.8571
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : N
##
```

```r
confusionMatrix(predict.knn, factor(validation$Diagnosis))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  N  O
##          N 18  3
##          O  0  0
##
##                Accuracy : 0.8571
##                  95% CI : (0.6366, 0.9695)
##     No Information Rate : 0.8571
##     P-Value [Acc > NIR] : 0.6477
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 0.2482
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.8571
##          Neg Pred Value :    NaN
##              Prevalence : 0.8571
##          Detection Rate : 0.8571
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : N
##
```

```
confusionMatrix(predict.rf, factor(validation$Diagnosis))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  N  O
##          N 18  3
##          O  0  0
##
##                Accuracy : 0.8571
##                  95% CI : (0.6366, 0.9695)
##     No Information Rate : 0.8571
##     P-Value [Acc > NIR] : 0.6477
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 0.2482
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.8571
##          Neg Pred Value :    NaN
##              Prevalence : 0.8571
##          Detection Rate : 0.8571
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : N
##
```

IV. CONCLUSION

All the algorithms used in this project performed same. They all have same overall accuracy and same as for other metrics. Although the accuracy is not that high. The algorithms performances on other metrics show that they predicted good.

The data set is small and all predictors are numeric. There is no much variation in the values of the predictors which ranges mostly between 0 and 1. The size of the validation set greatly determine the algorithms overall accuracy.

Citation

The data set was obtained from UCI machine learning site and was donated by the following: David Gil, Jose Luis Girela, Joaquin De Juan, M. Jose Gomez-Torres, and Magnus Johnsson. Predicting seminal quality

with artificial intelligence methods. Expert Systems with Applications, 39(16):12564 12573, 2012