Predict Diabetes Using Random Forest

Mahin Anwar

```
#Load Libraries
library(neuralnet)
```

```
## Warning: package 'neuralnet' was built under R version 4.0.3
```

```
library(tidyverse)
```

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

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.2     v dplyr   1.0.2
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## Warning: package 'dplyr' was built under R version 4.0.3
```

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

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.3
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

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

```
library(mlbench)
```

```
## Warning: package 'mlbench' was built under R version 4.0.3
```

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.0.3
```

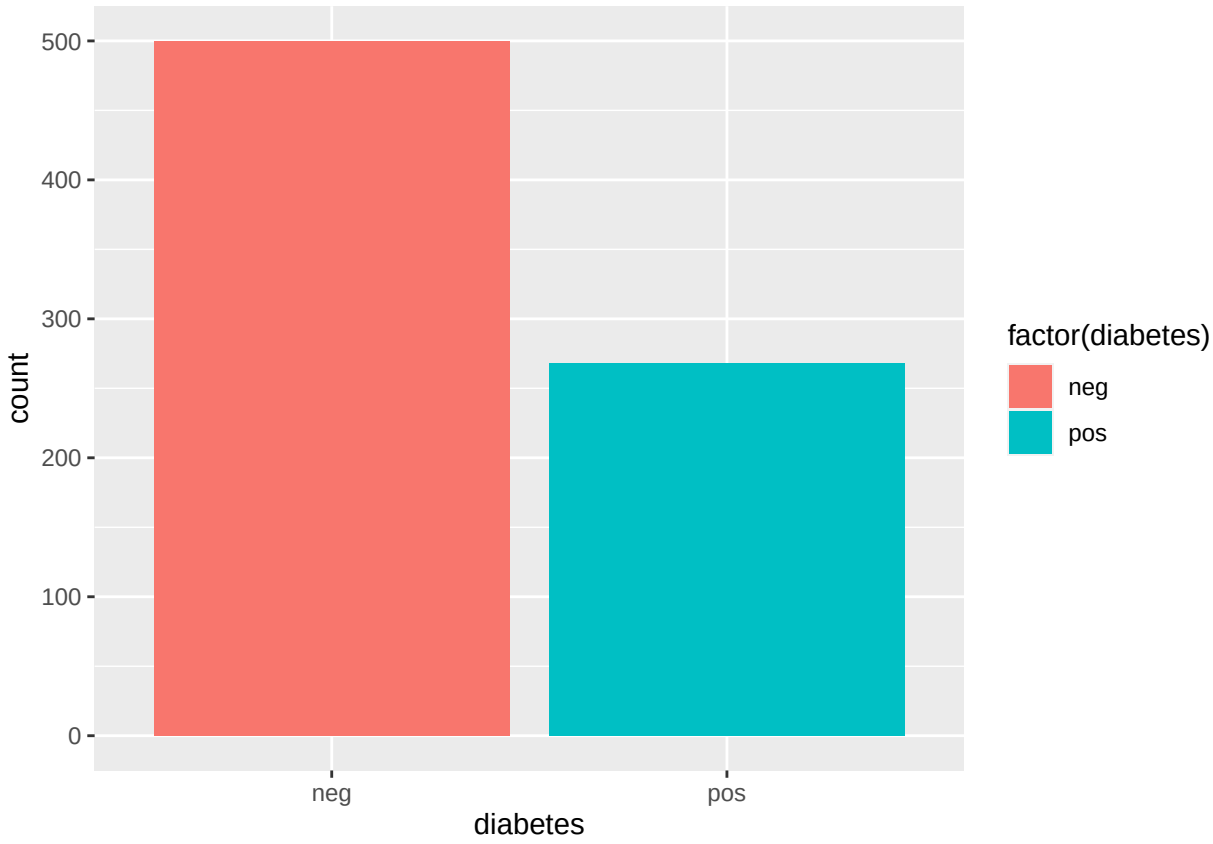
```
#Load Dataset
data("PimaIndiansDiabetes")
df <- PimaIndiansDiabetes
```

```
#Exploratory Analysis
str(df)
```

```
## 'data.frame':    768 obs. of  9 variables:
##  $ pregnant: num  6 1 8 1 0 5 3 10 2 8 ...
##  $ glucose : num  148 85 183 89 137 116 78 115 197 125 ...
```

```
##  $ pressure: num  72 66 64 66 40 74 50 0 70 96 ...
##  $ triceps : num  35 29 0 23 35 0 32 0 45 0 ...
##  $ insulin : num  0 0 0 94 168 0 88 0 543 0 ...
##  $ mass    : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
##  $ pedigree: num  0.627 0.351 0.672 0.167 2.288 ...
##  $ age     : num  50 31 32 21 33 30 26 29 53 54 ...
##  $ diabetes: Factor w/ 2 levels "neg","pos": 2 1 2 1 2 1 2 1 2 2 ...
```

```
g <- ggplot(df, aes(diabetes, fill = factor(diabetes)))
g + geom_bar()
```




```
df$binary <- ifelse(df$diabetes == 'neg', 0,1)
str(df)
```

```
## 'data.frame':    768 obs. of  10 variables:
##  $ pregnant: num  6 1 8 1 0 5 3 10 2 8 ...
##  $ glucose : num  148 85 183 89 137 116 78 115 197 125 ...
##  $ pressure: num  72 66 64 66 40 74 50 0 70 96 ...
##  $ triceps : num  35 29 0 23 35 0 32 0 45 0 ...
##  $ insulin : num  0 0 0 94 168 0 88 0 543 0 ...
##  $ mass    : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
##  $ pedigree: num  0.627 0.351 0.672 0.167 2.288 ...
##  $ age     : num  50 31 32 21 33 30 26 29 53 54 ...
##  $ diabetes: Factor w/ 2 levels "neg","pos": 2 1 2 1 2 1 2 1 2 2 ...
##  $ binary  : num  1 0 1 0 1 0 1 0 1 1 ...
```

```
#Data Partition
```

```
rows <- createDataPartition(df$binary, times = 1, p=0.7, list = FALSE)
train <- df[rows,]
test <- df[-rows,]
train <- train[,-9]
test <- test[,-9]

dim(train)
```

```
## [1] 538   9
```

```
dim(test)
```

```
## [1] 230   9
```

```
#Creating Model

model <- train(as.factor(binary) ~ . ,
               data = train,
               method = 'ranger',
               trControl = trainControl(method = 'repeatedcv', number = 2,
                                        repeats = 2)
               )
model
```

```
## Random Forest
##
## 538 samples
##   8 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (2 fold, repeated 2 times)
## Summary of sample sizes: 269, 269, 269, 269
## Resampling results across tuning parameters:
##
##   mtry  splitrule   Accuracy   Kappa
##   2     gini        0.7602230  0.4415098
##   2     extratrees  0.7574349  0.4186936
##   5     gini        0.7583643  0.4402797
##   5     extratrees  0.7667286  0.4523686
##   8     gini        0.7397770  0.3985073
##   8     extratrees  0.7620818  0.4479832
##
## Tuning parameter 'min.node.size' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 5, splitrule = extratrees
##  and min.node.size = 1.
```

```
#Predict Using Train & Test Set
predict_train <- predict(model, train)
predict_test <- predict(model, test)

predict_train
```

```
##   [1] 1 0 0 1 0 1 1 0 1 1 1 0 1 0 1 1 0 0 0 0 0 0 0 1 1 1 0 0 0 1 0 0 0 1 0 0 1
##  [38] 0 0 0 0 1 0 1 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0
```

```
##  [75] 1 1 0 0 1 1 1 0 0 1 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0
## [112] 1 0 0 1 1 0 0 1 1 0 0 0 1 0 0 0 0 1 1 1 1 0 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0
## [149] 1 1 1 0 1 1 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 1 1 0 1 0 0 0 0 0 0 1 0 0 1 0 1
## [186] 0 1 0 1 1 0 0 1 0 1 0 1 0 0 1 1 0 0 1 0 1 0 1 0 1 0 1 0 0 1 0 0 1 0 1 1 1
## [223] 0 0 1 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 0 0 0 1 0 0 1 0 0 0 0
## [260] 0 0 1 1 0 1 0 0 1 0 1 1 0 1 0 1 0 1 1 0 0 0 0 1 0 1 0 1 0 0 0 1 0 1 0 0 1
## [297] 0 0 0 0 1 1 1 0 0 1 0 0 1 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 1
## [334] 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0
## [371] 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1 0 0 0 1 0 1 0
## [408] 1 0 1 0 1 0 0 0 1 1 0 1 0 0 0 0 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0
## [445] 0 0 0 0 0 0 1 1 0 0 0 0 1 0 1 1 1 0 1 1 0 0 0 0 0 1 1 0 0 0 1 0 1 0 0 0 0 0
## [482] 1 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1
## [519] 0 0 0 0 1 0 1 1 0 0 1 0 0 0 1 0 0 0 1 0
## Levels: 0 1
```

```
#Create Confusion Matrix
confusionMatrix(predict_train, as.factor(train$binary))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 357   0
##          1   0 181
##
##                Accuracy : 1
##                  95% CI : (0.9932, 1)
##     No Information Rate : 0.6636
##     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
##              Prevalence : 0.6636
##          Detection Rate : 0.6636
##    Detection Prevalence : 0.6636
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(predict_test, as.factor(test$binary))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 124  38
##          1  19  49
##
```

```
##                Accuracy : 0.7522
##                  95% CI : (0.6912, 0.8066)
##     No Information Rate : 0.6217
##     P-Value [Acc > NIR] : 1.848e-05
##
##                   Kappa : 0.4496
##
##  Mcnemar's Test P-Value : 0.01712
##
##             Sensitivity : 0.8671
##             Specificity : 0.5632
##          Pos Pred Value : 0.7654
##          Neg Pred Value : 0.7206
##              Prevalence : 0.6217
##          Detection Rate : 0.5391
##    Detection Prevalence : 0.7043
##       Balanced Accuracy : 0.7152
##
##        'Positive' Class : 0
##
```