title: “Machine Learning: Predicting Diabetes using Pima Indians Data Set” author: “Melissa” date: “June 6, 2020” output: html_document —

1. Load the libraries.

```
library(neuralnet)
library(caret)
```

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

```
## Loading required package: ggplot2
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------------------------- tidyvers
```

```
## v tibble  3.0.0     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## v purrr   0.3.3
```

```
## -- Conflicts ------------------------------------------------------------------------ tidyverse_confl
## x dplyr::compute() masks neuralnet::compute()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x purrr::lift()    masks caret::lift()
```

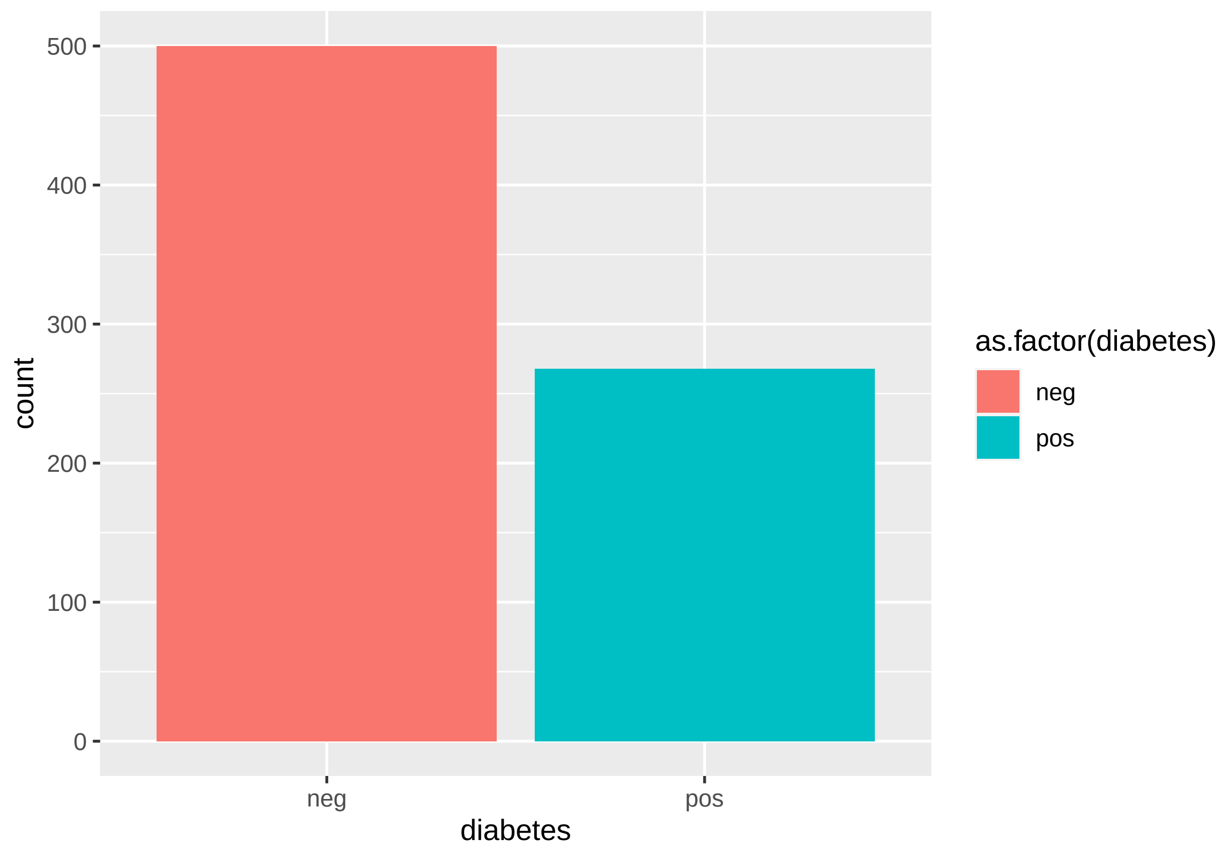
```
library(mlbench)
library(e1071)
library(ranger)
```

2. Store the data frame in an object, go over the structure and do some quick exploratory analysis to see the balance of the data set.

```
data("PimaIndiansDiabetes")
df <- PimaIndiansDiabetes
str(PimaIndiansDiabetes)
```

```
## '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 ...
```

```r
ggplot(df, aes(diabetes, fill = as.factor(diabetes))) + geom_bar()
```



3. Create a binary variable for the independent/response variable.

```r
df$binary <- ifelse(df$diabetes == "neg", 0, 1)
```

4. Create a train/test split using caret's createDataPartition function.

```r
rows <- createDataPartition(df$binary, times =1,
                            p = .7, list = F)
train <- df[rows,]
test <- df[-rows,]
```

5. Create the model

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

model
```

```
## Random Forest
##
## 538 samples
##   9 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        1         1
##   2     extratrees  1         1
##   5     gini        1         1
##   5     extratrees  1         1
##   9     gini        1         1
##   9     extratrees  1         1
##
## 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 = 2, splitrule = gini
##  and min.node.size = 1.
```

6. Test the model on the Test set and build a confusion matrix.

```
pred_train <- predict(model, train)
pred_test <- predict(model, test)

pred_train
```

```
##   [1] 1 0 1 0 1 0 1 1 0 1 1 1 1 1 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 1 0
##  [38] 0 1 0 0 1 0 0 1 0 1 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1
##  [75] 1 1 0 1 1 1 0 0 1 0 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0
## [112] 0 1 1 0 0 1 0 0 1 1 1 1 0 0 1 1 0 1 0 1 1 0 0 0 0 0 1 1 0 1 0 1 1 1 0 1 1
## [149] 0 0 0 0 1 0 0 0 0 0 1 1 1 0 0 1 1 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 1 0 1 1
## [186] 0 0 1 0 0 1 0 0 1 1 0 0 1 0 0 1 1 0 0 1 0 0 0 1 0 0 1 1 0 1 0 0 0 1 0 1 1
## [223] 1 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 1 1 0 0 0 0 1 0 1
## [260] 0 0 0 0 0 0 1 1 1 0 0 1 0 1 0 1 1 1 0 1 1 0 0 0 1 1 0 1 1 0 0 0 1 0 1 0 1
## [297] 0 0 0 0 0 1 0 0 0 1 0 0 1 1 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0
## [334] 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0
## [371] 0 0 0 1 0 0 0 1 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0
## [408] 1 0 1 0 0 0 1 0 0 1 0 1 0 0 1 0 0 1 1 0 0 0 0 1 1 1 0 0 1 0 0 0 0 0 0 0 0
## [445] 1 0 0 1 0 0 0 1 1 1 0 0 0 0 1 0 0 1 0 1 1 1 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0
## [482] 1 1 0 1 1 0 0 1 0 0 1 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 1 1 1 0 0 0 0 0 1 1
## [519] 0 0 1 0 0 1 0 1 0 0 1 1 1 1 0 1 0 1 0 0
## Levels: 0 1
```

```
confusionMatrix(pred_train, as.factor(train$binary))
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction   0   1
##          0 340   0
##          1   0 198
##
##                Accuracy : 1
##                  95% CI : (0.9932, 1)
##     No Information Rate : 0.632
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.000
##             Specificity : 1.000
##          Pos Pred Value : 1.000
##          Neg Pred Value : 1.000
##              Prevalence : 0.632
##          Detection Rate : 0.632
##    Detection Prevalence : 0.632
##       Balanced Accuracy : 1.000
##
##        'Positive' Class : 0
##
```

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 160   0
##          1   0  70
##
##                Accuracy : 1
##                  95% CI : (0.9841, 1)
##     No Information Rate : 0.6957
##     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 : 0.6957
##          Detection Rate : 0.6957
##    Detection Prevalence : 0.6957
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : 0
```

##