pimaindian randomforest project

Ronak

6/19/2020

This is a project where we use a machine learning classifier random forest to predict in the pima indians data set whether the person might be suffering from diabetes or not and in order to get finer results we will use repeated cross validation for resampling as well as the fact we will cross check answer with a confusion matrix

```
library(neuralnet)
```

```
## Warning: package 'neuralnet' was built under R version 4.0.2
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.0.2
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.0.2
```

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

```
## v tibble  3.0.1     v dplyr   1.0.0
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## v purrr   0.3.4
```

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

```
library(mlbench)
```

```
## Warning: package 'mlbench' was built under R version 4.0.2
```

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.0.2
```

```
library(ranger)
```

```
## Warning: package 'ranger' was built under R version 4.0.2
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.2
```

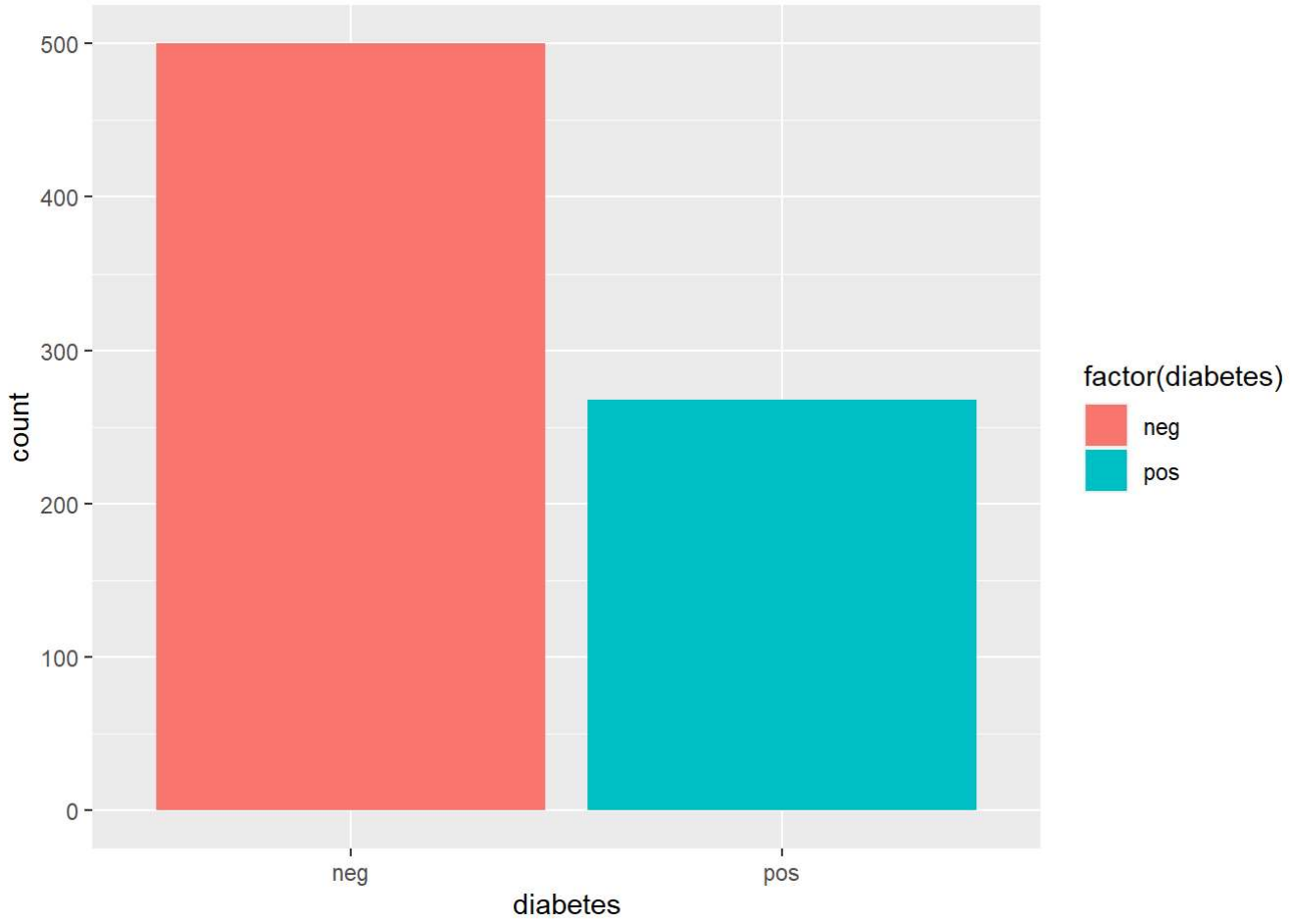
```
## Loading required package: lattice
```

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

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

First we load the required libraries

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




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

Then we assign the data and use str() function to see what variables does the data set contain as the target variable is a categorical variable we will have to create a new binary column which represents the same so we make a new binary variable and to see the distrbution of data we use gg plot function

```
rows<-createDataPartition(df$binary,times=1,p=.7,list=F)

train<-df[rows,]
test<-df[-rows,]
dim(train)
```

```
## [1] 538  10
```

```
dim(test)
```

```
## [1] 230  10
```

```
names(test)
```

```
##  [1] "pregnant" "glucose"  "pressure" "triceps"  "insulin"  "mass"
##  [7] "pedigree" "age"      "diabetes" "binary"
```

After this we create the data partition for the training and the test set of our model and then we check the dimensions of the dataset

```
train<-train[,-9]
test<-test[,-9]
names(test)
```

```
## [1] "pregnant" "glucose"  "pressure" "triceps"  "insulin"  "mass"     "pedigree"
## [8] "age"      "binary"
```

```
names(train)
```

```
## [1] "pregnant" "glucose"  "pressure" "triceps"  "insulin"  "mass"     "pedigree"
## [8] "age"      "binary"
```

```
str(train)
```

```
## 'data.frame':    538 obs. of  9 variables:
##  $ pregnant: num  6 1 8 5 3 10 2 8 4 10 ...
##  $ glucose : num  148 85 183 116 78 115 197 125 110 139 ...
##  $ pressure: num  72 66 64 74 50 0 70 96 92 80 ...
##  $ triceps : num  35 29 0 0 32 0 45 0 0 0 ...
##  $ insulin : num  0 0 0 0 88 0 543 0 0 0 ...
##  $ mass    : num  33.6 26.6 23.3 25.6 31 35.3 30.5 0 37.6 27.1 ...
##  $ pedigree: num  0.627 0.351 0.672 0.201 0.248 ...
##  $ age     : num  50 31 32 30 26 29 53 54 30 57 ...
##  $ binary  : num  1 0 1 0 1 0 1 1 0 0 ...
```

Here we basically removed the extra variable “diabetes” from the training and test as they would have have given a wrong accuracy value as the model will show a higher prediction accuracy than the expected output model so we remove those 2 variables and clean our data

```
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        0.7444238  0.4088323
##   2     extratrees  0.7434944  0.3857527
##   5     gini        0.7342007  0.3807476
##   5     extratrees  0.7453532  0.4064550
##   8     gini        0.7286245  0.3778130
##   8     extratrees  0.7397770  0.3967689
##
## 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.
```

so i created a model train where used the method random forest for classification and we train the data on the training set we just created and we use repreated cv resampling method to get more finer results we do this twice because we don't want a very long run time and the model to be fairly accurate

-repeatedcv is resampling method which does repeated cross validation for better accuracy

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

```
##   [1] 0 1 1 0 0 1 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0
##  [38] 0 1 0 0 0 1 1 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 1 0 0
##  [75] 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 1 1 0 1 0 1 1 0 0 1 0 1 0 0
## [112] 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 1 1 0
## [149] 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1
## [186] 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 1 1 0 0 0 0 1 1 1
## [223] 0 0 0 0 0 1 0 0
## Levels: 0 1
```

```
confusionMatrix(pred_test,as.factor(test$binary))#datatype of test var should be same as the way
you defined it in the model
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 131  37
##          1  17  45
##
##                Accuracy : 0.7652
##                  95% CI : (0.705, 0.8184)
##     No Information Rate : 0.6435
##     P-Value [Acc > NIR] : 4.768e-05
##
##                   Kappa : 0.4589
##
##  Mcnemar's Test P-Value : 0.009722
##
##             Sensitivity : 0.8851
##             Specificity : 0.5488
##          Pos Pred Value : 0.7798
##          Neg Pred Value : 0.7258
##              Prevalence : 0.6435
##          Detection Rate : 0.5696
##    Detection Prevalence : 0.7304
##       Balanced Accuracy : 0.7170
##
##        'Positive' Class : 0
##
```

After this we predicted the value of test and train and then we ran a confusion matrix of predict test of a model that was built on our training data and it give an accuracy of 78.7% percent which is very high accuracy on the basis of this model and we can also clearly see that the value was predicted correct for 181 people while it predicted it incorrectly for 49 people

Summary: In this machine learning project we took the data and added a new binary variable cleaned it and partitioned it; Then we utilized random forest classifier to make use of the data and predict whether the given population of pima-indians data set is suffering from diabetes or not we employed repeated cross validation for refined accuracy of the model and a confusion matrix to to cross check our answers accuracy (hope you liked this project)