Breast Cancer Analysis

ERICK@

2024-04

Breast Cancer Analysis

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. ##SUMMARY Summary: **Early diagnosis of cancer is critical for its successful treatment**. Thus, there is a high demand for accurate and cheap diagnostic methods. In this project we explored the applicability of decision tree machine learning techniques (**CART, Random Forests, and Boosted Trees,Naive Bayes**) for breast cancer diagnosis using digitized images of tissue samples. The data was obtained from UC Irvine Machine Learning Repository ("Breast Cancer Wisconsin data set" created by William H. Wolberg, W. Nick Street, and Olvi L. Mangasarian). The most accurate traditional method for diagnostic is a rather invasive technique, called breast biopsy, where a small piece of breast tissue is surgically removed, and then the tissue sample has to be examined by specialist. However, a much less invasive technique can be used, where the samples can be obtained by a minimally invasive fine needle aspirate method. The sample obtained by this method can be easily digitized and used for computationally based diagnostic. Using machine learning methods for diagnostic can significantly increase processing speed and on a big scale can make the diagnostic significantly cheaper.

Here we studied the applicability of **Random Forests** and **Boosted Trees methods and Naive Bayes for cancer prediction**. We used CART method for comparison as well. The CART model achieved an estimated accuracy of about 91%. Random Forests 94% and Boosted Trees models achieved an estimated accuracy of about 97% on this dataset.

Data Cleaning and Loading

First the necessary libraries are loaded in R environment. ggplot library is used to make plots, corrplot is used to make corelation plots, caret is used to make data processing and machine learning

```
library(tidyverse)
library(caret)
library(e1071)#SVM, NAIVEBAYES MODELS
library(randomForest)#RANDOMFOREST MODEL
library(gridExtra, )
library(pROC)
library(corrplot)
library(janitor)
```

Data loading

The data is taken from UCI Repository and downloaded and saved into the localmachine

```
df <- read_csv("C:/Users/langa/OneDrive/Desktop/Dataset/BreastCancersData.csv")
```

Seeing the structure and the summary of the data

```
glimpse(df)
```

```
## Rows: 568
## Columns: 33
## $ id                      <dbl> 842302, 842517, 84300903, 84348301, 84358402, ~
## $ diagnosis               <chr> "M", "M", "M", "M", "M", "M", "M", "M", "M", "~
## $ radius_mean             <dbl> 17.990, 20.570, 19.690, 11.420, 20.290, 12.450~
## $ texture_mean            <dbl> 10.38, 17.77, 21.25, 20.38, 14.34, 15.70, 19.9~
## $ perimeter_mean          <dbl> 122.80, 132.90, 130.00, 77.58, 135.10, 82.57, ~
## $ area_mean               <dbl> 1001.0, 1326.0, 1203.0, 386.1, 1297.0, 477.1, ~
## $ smoothness_mean         <dbl> 0.11840, 0.08474, 0.10960, 0.14250, 0.10030, 0~
## $ compactness_mean        <dbl> 0.27760, 0.07864, 0.15990, 0.28390, 0.13280, 0~
## $ concavity_mean          <dbl> 0.30010, 0.08690, 0.19740, 0.24140, 0.19800, 0~
## $ `concave points_mean`   <dbl> 0.14710, 0.07017, 0.12790, 0.10520, 0.10430, 0~
## $ symmetry_mean           <dbl> 0.2419, 0.1812, 0.2069, 0.2597, 0.1809, 0.2087~
## $ fractal_dimension_mean  <dbl> 0.07871, 0.05667, 0.05999, 0.09744, 0.05883, 0~
## $ radius_se               <dbl> 1.0950, 0.5435, 0.7456, 0.4956, 0.7572, 0.3345~
## $ texture_se              <dbl> 0.9053, 0.7339, 0.7869, 1.1560, 0.7813, 0.8902~
## $ perimeter_se            <dbl> 8.589, 3.398, 4.585, 3.445, 5.438, 2.217, 3.18~
## $ area_se                 <dbl> 153.40, 74.08, 94.03, 27.23, 94.44, 27.19, 53.~
## $ smoothness_se           <dbl> 0.006399, 0.005225, 0.006150, 0.009110, 0.0114~
## $ compactness_se          <dbl> 0.049040, 0.013080, 0.040060, 0.074580, 0.0246~
## $ concavity_se            <dbl> 0.05373, 0.01860, 0.03832, 0.05661, 0.05688, 0~
## $ `concave points_se`     <dbl> 0.015870, 0.013400, 0.020580, 0.018670, 0.0188~
## $ symmetry_se             <dbl> 0.03003, 0.01389, 0.02250, 0.05963, 0.01756, 0~
## $ fractal_dimension_se    <dbl> 0.006193, 0.003532, 0.004571, 0.009208, 0.0051~
## $ radius_worst            <dbl> 25.38, 24.99, 23.57, 14.91, 22.54, 15.47, 22.8~
## $ texture_worst           <dbl> 17.33, 23.41, 25.53, 26.50, 16.67, 23.75, 27.6~
## $ perimeter_worst         <dbl> 184.60, 158.80, 152.50, 98.87, 152.20, 103.40,~
## $ area_worst              <dbl> 2019.0, 1956.0, 1709.0, 567.7, 1575.0, 741.6, ~
## $ smoothness_worst        <dbl> 0.1622, 0.1238, 0.1444, 0.2098, 0.1374, 0.1791~
## $ compactness_worst       <dbl> 0.6656, 0.1866, 0.4245, 0.8663, 0.2050, 0.5249~
## $ concavity_worst         <dbl> 0.71190, 0.24160, 0.45040, 0.68690, 0.40000, 0~
## $ `concave points_worst`  <dbl> 0.26540, 0.18600, 0.24300, 0.25750, 0.16250, 0~
## $ symmetry_worst          <dbl> 0.4601, 0.2750, 0.3613, 0.6638, 0.2364, 0.3985~
## $ fractal_dimension_worst <dbl> 0.11890, 0.08902, 0.08758, 0.17300, 0.07678, 0~
## $ ...33                   <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
```

```
df$diagnosis <- as_factor(df$diagnosis)
levels(df$diagnosis)
```

```
## [1] "M" "B"
```

```
view(df)
sum(duplicated(df))#no duplicates
```

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

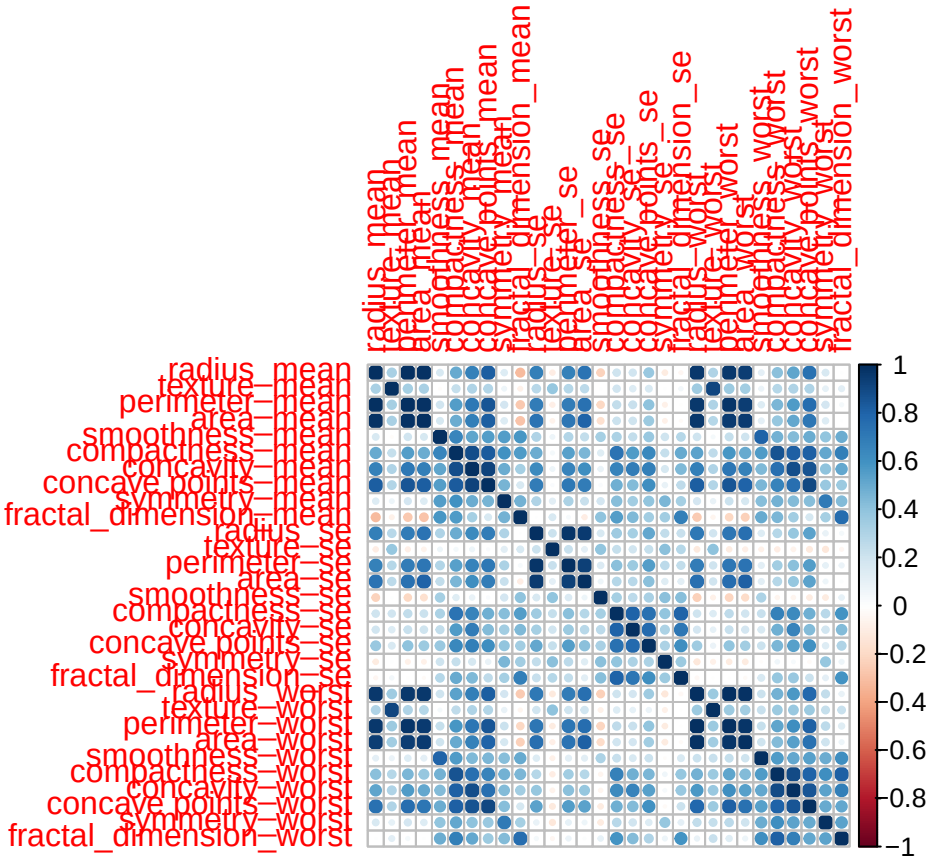
```
df$...33 <- NULL#REMOVE UNNECESARY VARIABLE
#remove ID COLUM
df$id <- NULL
```

```
## we find that there are no missing values
## we find that data is little unbalanced
df %>% tabyl(diagnosis) %>% adorn_pct_formatting()#percentage composition
```

```
##  diagnosis   n percent
##          M 212   37.3%
##          B 356   62.7%
```

```
## we then show some correlation

df %>% select(-diagnosis) %>% cor() %>% corrplot()
```



Modelling

We are going to get a training and a testing set to use when building some models:

```r
## We are going to get a training and a testing set to use when building some models:
set.seed(1234)
library(rsample)
split <- initial_split(df, prop = 7/10)
train_data <- training(split)
test_data <- testing(split)
```

```r
## Applying learning models
fitControl <- trainControl(method="cv",
               number = 5,
               preProcOptions = list(thresh = 0.99),#threshold for pca preprocess
               classProbs = TRUE,
               summaryFunction = twoClassSummary)
```

Applying learning models

Model1: Random Forest

Building the model on the training data

```r
## random forest
model_rf <- train(diagnosis~.,
                  train_data,
                  method="ranger",
                  metric="ROC",
                  #tuneLength=10,
                  #tuneGrid = expand.grid(mtry = c(2, 3, 6)),
                  preProcess = c('center', 'scale'),
                  trControl=fitControl)
```

```r
## testing for random forets
pred_rf <- predict(model_rf, test_data)
cm_rf <- confusionMatrix(pred_rf, test_data$diagnosis,
                         positive = "M")
cm_rf
```

Testing on the testing data

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   M   B
##          M  57   4
##          B   4 106
##
##                Accuracy : 0.9532
```

```
##                  95% CI : (0.9099, 0.9796)
##     No Information Rate : 0.6433
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8981
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9344
##             Specificity : 0.9636
##          Pos Pred Value : 0.9344
##          Neg Pred Value : 0.9636
##              Prevalence : 0.3567
##          Detection Rate : 0.3333
##    Detection Prevalence : 0.3567
##       Balanced Accuracy : 0.9490
##
##        'Positive' Class : M
##
```

We find that accuracy of this model is 95%

Model2: Naive Bayes

Building and testing the model

```
# install.packages("klaR")
# model_nb <- train(diagnosis~.,
#                   train_data,
#                   method="nb",
#                   metric="ROC",
#                   #tuneLength=10,
#                   #tuneGrid = expand.grid(mtry = c(2, 3, 6)),
#                   preProcess = c('center', 'scale'),
#                   trControl=fitControl)

model_nb <- naiveBayes(diagnosis~., data=train_data)
```

```
## testing for random forets
pred_nb <- predict(model_nb, test_data)
cm_nb <- confusionMatrix(pred_nb, test_data$diagnosis,
                         positive = "M")
cm_nb
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   M   B
##          M  55   9
##          B   6 101
##
##                Accuracy : 0.9123
```

```
##                  95% CI : (0.8594, 0.9501)
##     No Information Rate : 0.6433
##     P-Value [Acc > NIR] : 3.798e-16
##
##                   Kappa : 0.8109
##
##  Mcnemar's Test P-Value : 0.6056
##
##             Sensitivity : 0.9016
##             Specificity : 0.9182
##          Pos Pred Value : 0.8594
##          Neg Pred Value : 0.9439
##              Prevalence : 0.3567
##          Detection Rate : 0.3216
##    Detection Prevalence : 0.3743
##       Balanced Accuracy : 0.9099
##
##        'Positive' Class : M
##
```

Accuracy of this model is found to be 91%

Model3: Boosted tree

```
library(gbm)
set.seed(1)
gbm_model <- train(diagnosis ~ ., train_data, method="gbm", verbose=FALSE)
gbm_model
```

```
## Stochastic Gradient Boosting
##
## 397 samples
##  30 predictor
##   2 classes: 'M', 'B'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 397, 397, 397, 397, 397, 397, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.9383208  0.8668139
##   1                  100      0.9460147  0.8840026
##   1                  150      0.9518194  0.8964748
##   2                   50      0.9458812  0.8836387
##   2                  100      0.9493015  0.8911753
##   2                  150      0.9518868  0.8967205
##   3                   50      0.9479781  0.8879575
##   3                  100      0.9537456  0.9007284
##   3                  150      0.9545374  0.9023854
##
```

```
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150, interaction.depth =
##  3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
gbm_model$finalModel
```

```
## A gradient boosted model with bernoulli loss function.
## 150 iterations were performed.
## There were 30 predictors of which 30 had non-zero influence.
```

```
#Performance on testing set:

pred5 <- predict(gbm_model, test_data)
confusionMatrix(pred5, test_data$diagnosis, positive="M")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   M   B
##          M  58   3
##          B   3 107
##
##                Accuracy : 0.9649
##                  95% CI : (0.9252, 0.987)
##     No Information Rate : 0.6433
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9235
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9508
##             Specificity : 0.9727
##          Pos Pred Value : 0.9508
##          Neg Pred Value : 0.9727
##              Prevalence : 0.3567
##          Detection Rate : 0.3392
##    Detection Prevalence : 0.3567
##       Balanced Accuracy : 0.9618
##
##        'Positive' Class : M
##
```

Accuracy was found to be 96%

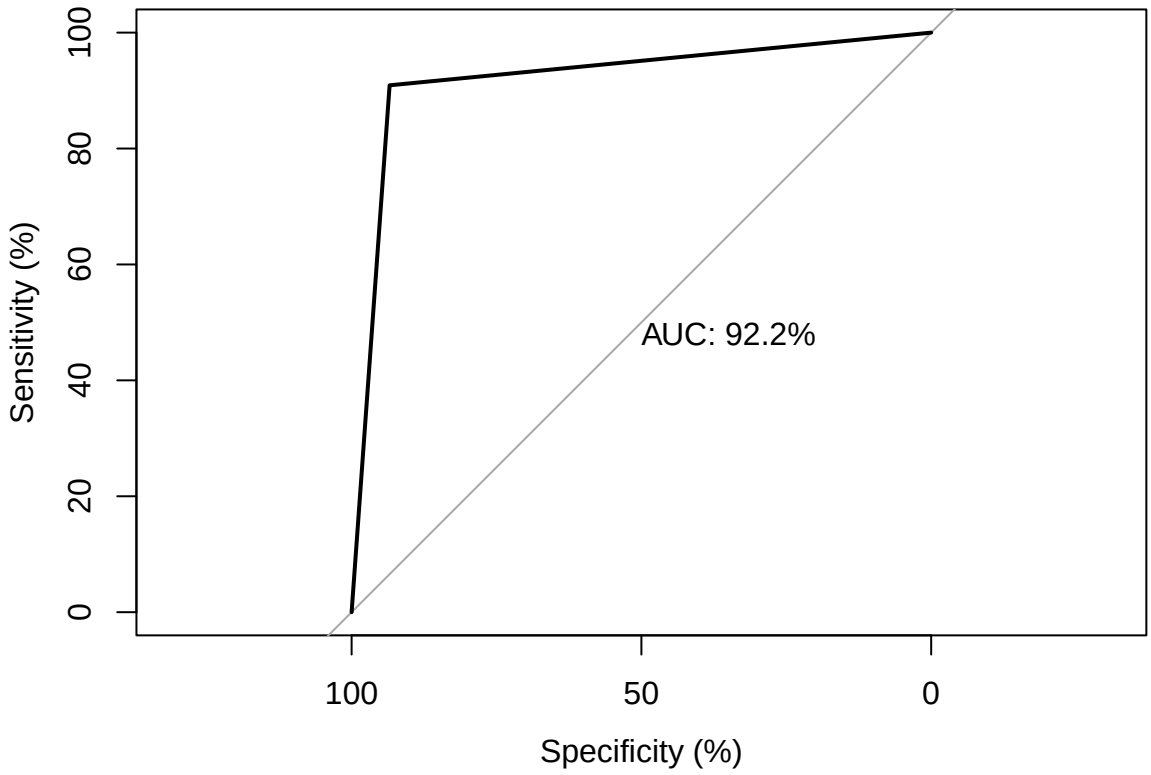
Logistic Regression

```
log_model <- glm(diagnosis~., data = train_data, family = 'binomial')
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
pred_log <- predict(log_model,test_data, type = 'response')
```

```
roc(test_data$diagnosis, pred_log, percent=TRUE, plot=TRUE,
    print.auc=TRUE)
```




```
##
## Call:
## roc.default(response = test_data$diagnosis, predictor = pred_log,     percent = TRUE, plot = TRUE, pr
##
## Data: pred_log in 61 controls (test_data$diagnosis M) < 110 cases (test_data$diagnosis B).
## Area under the curve: 92.18%
```

Accuracy was found to be 92%

Accuracy Measure

Boosted Tree: 96% Random Forest : 95%, Logistic Regression: 92%, Naive Bayes : 91% .

Boosted tree method has given the best accuracy among the four