Project 1

Kiet Nguyen

2023-02-19

INTRODUCTION OF THE PROJECT

- This project investigates the Breast Cancer data frame at Wisconsin.
- The data frame has 32 columns, with 31 attributes of a tumor and 1 diagnosis of Malignant **(M)** or Benign **(B)**.
- The goal of this project is to:

1. Build a **Classification Model** to classifying if a tumor is Malignant or Benign, based on their attribute.
2. Determine which attribute play the most important roles in diagnosing cancer.
3. Investigate **TWO** most important attributes and their correlation to each other and the diagnosis.

- We will be using **80-20 Train-Test Split**, along with **Polynomial Kernels** training method

LOADING NECESSARY LIBRARY

The four packages we need for this projects are **tidyverse**, **dplyr**, **plotly** and **caret**. The classification model is created by mainly using the **caret** package.

```
library(tidyverse) #used for 2D plot
library(dplyr)
library(caret) # Classification and Regression Training
library(plotly) #used for 3D plot
```

LOADING AND CLEANING DATA

The first and foremost step is loading the data, We are using the **Breast Cancer Wisconsin (Diagnostic) Data Set** provided publicly on Kaggle, by UC Irvine. In this data set, we are only investigating on attributes that contributes to the diagnosis of tumors. Finally, we are cleaning if any data points are null or not.

```
# Loading data and shows their attribute.
cancer <- read.csv("data.csv")

# exclude columns id and X since they are not attributes of determining cancer
cancer <- select(cancer, -c(id, X))

# Check if data has any null values in any columns we are searching for.
sum(is.na(cancer))
```

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

```r
# No null Value, we are good to process

# Summary on the data set
str(cancer)
```

```
## 'data.frame':    569 obs. of  31 variables:
##  $ diagnosis              : chr  "M" "M" "M" "M" ...
##  $ radius_mean            : num  18 20.6 19.7 11.4 20.3 ...
##  $ texture_mean           : num  10.4 17.8 21.2 20.4 14.3 ...
##  $ perimeter_mean         : num  122.8 132.9 130 77.6 135.1 ...
##  $ area_mean              : num  1001 1326 1203 386 1297 ...
##  $ smoothness_mean        : num  0.1184 0.0847 0.1096 0.1425 0.1003 ...
##  $ compactness_mean       : num  0.2776 0.0786 0.1599 0.2839 0.1328 ...
##  $ concavity_mean         : num  0.3001 0.0869 0.1974 0.2414 0.198 ...
##  $ concave.points_mean    : num  0.1471 0.0702 0.1279 0.1052 0.1043 ...
##  $ symmetry_mean          : num  0.242 0.181 0.207 0.26 0.181 ...
##  $ fractal_dimension_mean : num  0.0787 0.0567 0.06 0.0974 0.0588 ...
##  $ radius_se              : num  1.095 0.543 0.746 0.496 0.757 ...
##  $ texture_se             : num  0.905 0.734 0.787 1.156 0.781 ...
##  $ perimeter_se           : num  8.59 3.4 4.58 3.44 5.44 ...
##  $ area_se                : num  153.4 74.1 94 27.2 94.4 ...
##  $ smoothness_se          : num  0.0064 0.00522 0.00615 0.00911 0.01149 ...
##  $ compactness_se         : num  0.049 0.0131 0.0401 0.0746 0.0246 ...
##  $ concavity_se           : num  0.0537 0.0186 0.0383 0.0566 0.0569 ...
##  $ concave.points_se      : num  0.0159 0.0134 0.0206 0.0187 0.0188 ...
##  $ symmetry_se            : num  0.03 0.0139 0.0225 0.0596 0.0176 ...
##  $ fractal_dimension_se   : num  0.00619 0.00353 0.00457 0.00921 0.00511 ...
##  $ radius_worst           : num  25.4 25 23.6 14.9 22.5 ...
##  $ texture_worst          : num  17.3 23.4 25.5 26.5 16.7 ...
##  $ perimeter_worst        : num  184.6 158.8 152.5 98.9 152.2 ...
##  $ area_worst             : num  2019 1956 1709 568 1575 ...
##  $ smoothness_worst       : num  0.162 0.124 0.144 0.21 0.137 ...
##  $ compactness_worst      : num  0.666 0.187 0.424 0.866 0.205 ...
##  $ concavity_worst        : num  0.712 0.242 0.45 0.687 0.4 ...
##  $ concave.points_worst   : num  0.265 0.186 0.243 0.258 0.163 ...
##  $ symmetry_worst         : num  0.46 0.275 0.361 0.664 0.236 ...
##  $ fractal_dimension_worst: num  0.1189 0.089 0.0876 0.173 0.0768 ...
```

MODIFYING THE DATA SET

Since the **diagnosis** stores the value as **M** and **B**, we are changing these values into **1** and **0** respectively.

```r
diagnosis = recode_factor(cancer$diagnosis, 'M' = '1', 'B' = '0')
cancer$diagnosis = diagnosis

# Data frame after being modified
head(cancer)
```

```
##   diagnosis radius_mean texture_mean perimeter_mean area_mean smoothness_mean
## 1         1       17.99        10.38         122.80    1001.0         0.11840
## 2         1       20.57        17.77         132.90    1326.0         0.08474
## 3         1       19.69        21.25         130.00    1203.0         0.10960
```

```
## 4         1       11.42        20.38          77.58     386.1         0.14250
## 5         1       20.29        14.34         135.10    1297.0         0.10030
## 6         1       12.45        15.70          82.57     477.1         0.12780
##   compactness_mean concavity_mean concave.points_mean symmetry_mean
## 1          0.27760         0.3001             0.14710        0.2419
## 2          0.07864         0.0869             0.07017        0.1812
## 3          0.15990         0.1974             0.12790        0.2069
## 4          0.28390         0.2414             0.10520        0.2597
## 5          0.13280         0.1980             0.10430        0.1809
## 6          0.17000         0.1578             0.08089        0.2087
##   fractal_dimension_mean radius_se texture_se perimeter_se area_se
## 1                0.07871    1.0950     0.9053        8.589  153.40
## 2                0.05667    0.5435     0.7339        3.398   74.08
## 3                0.05999    0.7456     0.7869        4.585   94.03
## 4                0.09744    0.4956     1.1560        3.445   27.23
## 5                0.05883    0.7572     0.7813        5.438   94.44
## 6                0.07613    0.3345     0.8902        2.217   27.19
##   smoothness_se compactness_se concavity_se concave.points_se symmetry_se
## 1      0.006399        0.04904      0.05373           0.01587     0.03003
## 2      0.005225        0.01308      0.01860           0.01340     0.01389
## 3      0.006150        0.04006      0.03832           0.02058     0.02250
## 4      0.009110        0.07458      0.05661           0.01867     0.05963
## 5      0.011490        0.02461      0.05688           0.01885     0.01756
## 6      0.007510        0.03345      0.03672           0.01137     0.02165
##   fractal_dimension_se radius_worst texture_worst perimeter_worst area_worst
## 1             0.006193        25.38         17.33          184.60     2019.0
## 2             0.003532        24.99         23.41          158.80     1956.0
## 3             0.004571        23.57         25.53          152.50     1709.0
## 4             0.009208        14.91         26.50           98.87      567.7
## 5             0.005115        22.54         16.67          152.20     1575.0
## 6             0.005082        15.47         23.75          103.40      741.6
##   smoothness_worst compactness_worst concavity_worst concave.points_worst
## 1           0.1622            0.6656          0.7119               0.2654
## 2           0.1238            0.1866          0.2416               0.1860
## 3           0.1444            0.4245          0.4504               0.2430
## 4           0.2098            0.8663          0.6869               0.2575
## 5           0.1374            0.2050          0.4000               0.1625
## 6           0.1791            0.5249          0.5355               0.1741
##   symmetry_worst fractal_dimension_worst
## 1         0.4601                 0.11890
## 2         0.2750                 0.08902
## 3         0.3613                 0.08758
## 4         0.6638                 0.17300
## 5         0.2364                 0.07678
## 6         0.3985                 0.12440
```

BUILDING MODEL

We are building our model by distributing our data set into 80% training set, 20% testing set, and the training set for cross validation (CV). Of course, they will be randomly distributed to make the model least biased as possible.

```r
#set random seed number to get reproducible model
set.seed(13)

# Building subsets
TrainingIndex <- createDataPartition(cancer$diagnosis, p = 0.8, list = FALSE)
trainingSet <- cancer[TrainingIndex,] #80% random data points
testingSet <- cancer[-TrainingIndex,] #the rest 20%

#Check if the TrainingIndex is randomly chosen
head(TrainingIndex, 10) # looks good
```

```
##       Resample1
##  [1,]         1
##  [2,]         2
##  [3,]         3
##  [4,]         5
##  [5,]         6
##  [6,]         7
##  [7,]         8
##  [8,]         9
##  [9,]        10
## [10,]        11
```

BUILDING THE MODEL USING SUPPORT VECTOR MACHINE (SVM)

In this step, we are utilizing the support vector machine algorithm with polynomial kernel. SVM helps to create a support vector classifier that separates the **B** and **M** values, while the polynomial kernel helps to alleviate the overlapping data points that can not be easily separated by a linear kernel.

```r
# Our training model
Model <- train(diagnosis ~ ., data = trainingSet,
               method = "svmPoly",
               preProcess = c("scale","center"),
               trControl = trainControl(method = "none"),
               tuneGrid = data.frame(degree= 1, scale = 1, C = 1))

# Our CV model
cvModel <- train(diagnosis ~ ., data = trainingSet,
               method = "svmPoly",
               preProcess = c("scale","center"),
               trControl = trainControl(method = "none", number = 25), # 25 iterations
               tuneGrid = data.frame(degree= 1, scale = 1, C = 1))
```

APPLY MODEL FOR PREDICTION

We are applying **predict** function using the trained model to predict output on our subsets. In this case, we are using the **leave-one-out** method. This step helps to reduce overfitting on our model.

```r
Model.training <- predict(Model,trainingSet) # Apply model to predict trainingSet
Model.testing <- predict(Model, testingSet) # Apply model to predict testingSet
Model.cv <- predict(cvModel, trainingSet) # Cross Validation
```

MODEL PERFORMANCE- Confusion Matrix and Statistics

In this step, we are checking if our training model is reliable or not, by using the confusion matrix. This will provides information such as accuracy rate, which, however, does not need to be exact for a classification model to be trustworthy.

```
# Print out confusion matrix of three models we are investigating on
trainingConfusion <- confusionMatrix(Model.training, trainingSet$diagnosis)
testConfusion <- confusionMatrix(Model.testing, testingSet$diagnosis)
cvConfusion <- confusionMatrix(Model.cv, trainingSet$diagnosis)
```

Statistic on the training data set using training model:

```
print(trainingConfusion)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   0
##          1 164   0
##          0   6 286
##
##                Accuracy : 0.9868
##                  95% CI : (0.9716, 0.9952)
##     No Information Rate : 0.6272
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9717
##
##  Mcnemar's Test P-Value : 0.04123
##
##             Sensitivity : 0.9647
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9795
##              Prevalence : 0.3728
##          Detection Rate : 0.3596
##    Detection Prevalence : 0.3596
##       Balanced Accuracy : 0.9824
##
##        'Positive' Class : 1
##
```

Statistic on the test data set using training model:

```
print(testConfusion)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  0
##          1 39  1
```

```
##          0  3 70
##
##                Accuracy : 0.9646
##                  95% CI : (0.9118, 0.9903)
##     No Information Rate : 0.6283
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9235
##
##  Mcnemar's Test P-Value : 0.6171
##
##             Sensitivity : 0.9286
##             Specificity : 0.9859
##          Pos Pred Value : 0.9750
##          Neg Pred Value : 0.9589
##              Prevalence : 0.3717
##          Detection Rate : 0.3451
##    Detection Prevalence : 0.3540
##       Balanced Accuracy : 0.9572
##
##        'Positive' Class : 1
##
```

Cross-Validation on the training model

```
print(cvConfusion)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   0
##          1 164   0
##          0   6 286
##
##                Accuracy : 0.9868
##                  95% CI : (0.9716, 0.9952)
##     No Information Rate : 0.6272
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9717
##
##  Mcnemar's Test P-Value : 0.04123
##
##             Sensitivity : 0.9647
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9795
##              Prevalence : 0.3728
##          Detection Rate : 0.3596
##    Detection Prevalence : 0.3596
##       Balanced Accuracy : 0.9824
##
##        'Positive' Class : 1
##
```

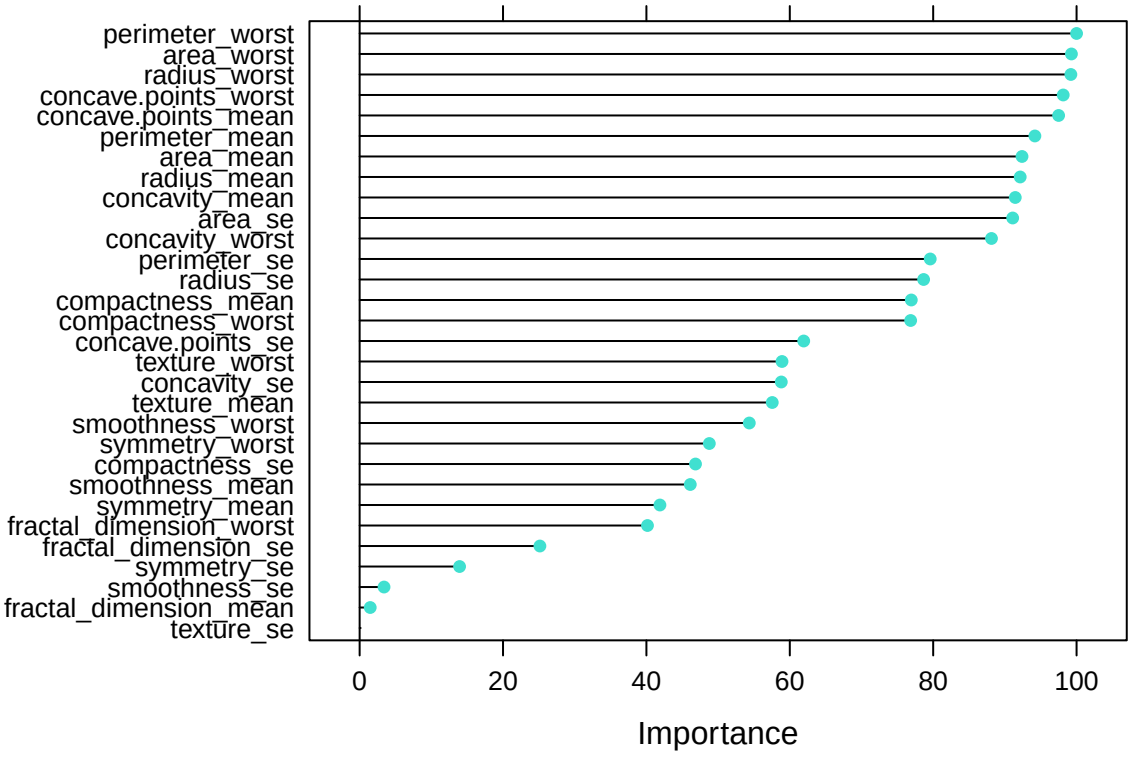
COMMENTS ON OUR CLASSIFICATION MODEL

Personally speaking, I think our model is well trained. We achieved 98% correct on training model and its cross validation. For predicting "unfed" data, we achieve 96% accuracy, which is an acceptable rate.

DETERMINING TWO IMPORTANT FEATURES FOR DIAGNOSIS

As being said in the introduction, we are trying to find two most inportant attributes that contribute to our prediction. We are using the **varImp** function, which would return a detailed comparison.

```
Importance <- varImp(Model)
plot(Importance, col = "turquoise")
```



Our model suggests that **perimeter_worst** and **area_worst** as the two most important attributes in our studying

DETERMINING perimeter_worst, area_worst AND diagnosis RELATIONSHIP

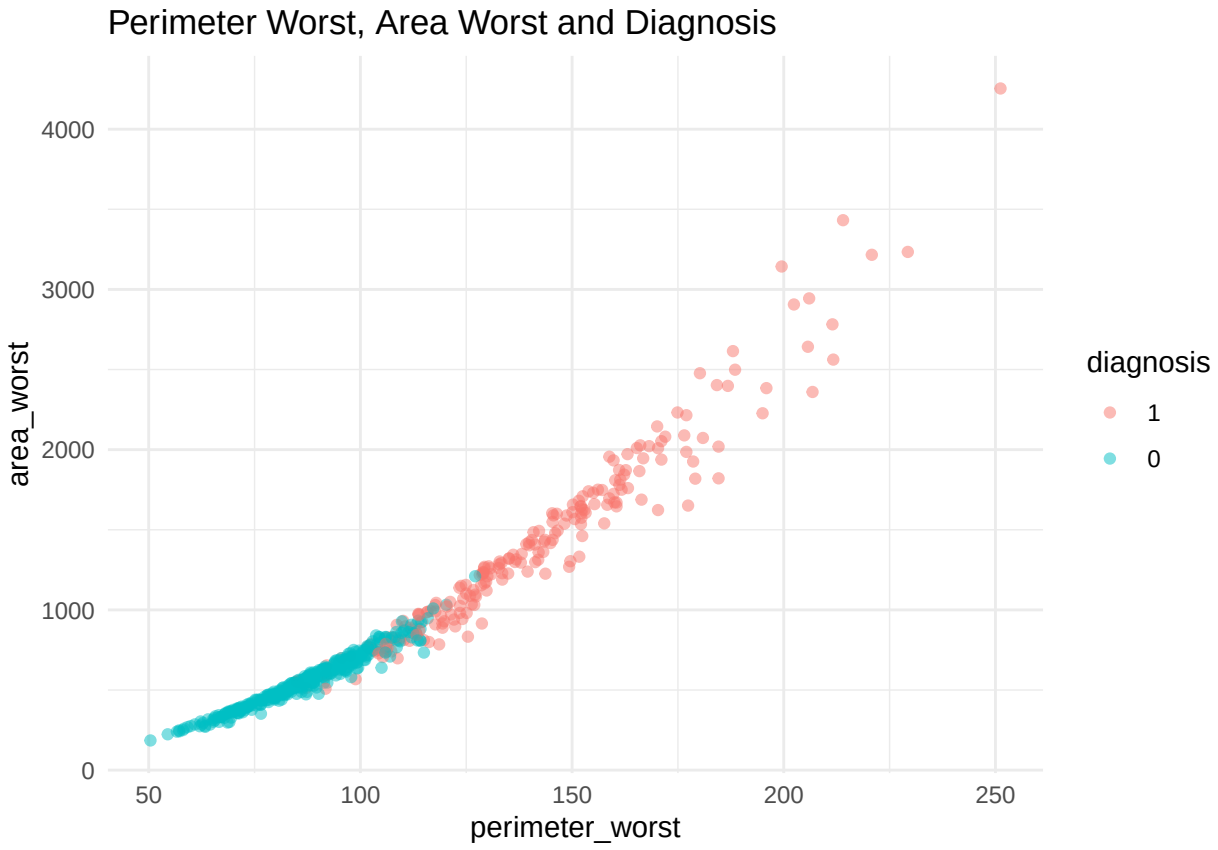
We are using **plotly** to have a better view about the distribution of tumor diagnosis base on their two important attribute.

- IN 3D view:

```
# Set up the plot
plot_ly(x = cancer$perimeter_worst, y =cancer$area_worst, z = cancer$diagnosis,
      type = "scatter3d", mode = "markers",
      color = as.factor(cancer$diagnosis), marker = list(size=2))
```

- In 2D view:

```
rad <- ggplot(cancer, aes(x= perimeter_worst, y = area_worst)) +
  geom_point(alpha = 1/2, aes(color = diagnosis)) +
  ggtitle(label = "Perimeter Worst, Area Worst and Diagnosis") +
  theme_minimal()
print(rad)
```



CONCLUSION

- It is obvious to conclude from the graph that Malignant tumor has a wider interval than Benign one. The larger values tumors are in both attribute, the more likely they are diagnosed to be Malignant
- The largest **B** tumor in both attributes, (perimeter, area) is (127.1, 1210), while the smallest **M** tumor can be as small as (85.1, 553.6).

DATA SET SOURCE

This data set is publicized online on Kaggle, and is subjected to copyrights. All rights reserved to the owners. https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data