Mid-Term Project
Wisconsin Breast Cancer Dataset

Nihel Charfi

MSCS 6520 Business Analytics

Spring 2019

3/3/2019

❖ Abstract:

This report is about the analysis of ***Breast Cancer Wisconsin (Diagnostic)*** *Dataset*, obtained from Kaggle website. The purpose of this project is to build various KNN models, analyze the results and select an effective model.

<https://www.kaggle.com/uciml/breast-cancer-wisconsin-data/home >

❖ About the dataset:

This breast cancer database was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg. Dataset contains about 569 instances and 33 features. Each instance has one of the two classes: 2 or 4 (2 for benign, 4 for malignant). This is a class unbalanced dataset for (62.74%) of Benign class and (37.25%) of malignant.

```
> str(bc_data)
Classes 'tbl_df', 'tbl' and 'data.frame':       569 obs. of  33 variables:
 $ id                     : int  842302 842517 84300903 84348301 84358402 84378
6 844359 84458202 844981 84501001 ...
 $ diagnosis              : Factor w/ 2 levels "B","M": 2 2 2 2 2 2 2 2 2 2 ...
 $ 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 ...
 $ X                      : logi  NA NA NA NA NA NA ...
>
```

❖ Handling missing values:

Checking the missing values is an important step in machine learning. In general, all data sources include errors and missing values. Data cleaning addresses these anomalies. Missing values can be imputed either by dropping the rows or replacing them by a default value.
In Wisconsin Breast Cancer dataset, the last column is not right because it not a part of the data. However, this dataset does not have any missing value:

```
> bc_data$X  = NULL;
> #Check again for the total missing values in each column
> colSums(is.na(bc_data))
                     id               diagnosis             radius_mean
                      0                       0                       0
            texture_mean          perimeter_mean               area_mean
                      0                       0                       0
         smoothness_mean        compactness_mean         concavity_mean
                      0                       0                       0
    concave.points_mean          symmetry_mean fractal_dimension_mean
                      0                       0                       0
              radius_se             texture_se           perimeter_se
                      0                       0                       0
                area_se          smoothness_se         compactness_se
                      0                       0                       0
           concavity_se     concave.points_se            symmetry_se
                      0                       0                       0
  fractal_dimension_se          radius_worst          texture_worst
                      0                       0                       0
        perimeter_worst            area_worst       smoothness_worst
                      0                       0                       0
      compactness_worst        concavity_worst  concave.points_worst
                      0                       0                       0
        symmetry_worst fractal_dimension_worst
                      0                       0
```

❖ Data analysis:

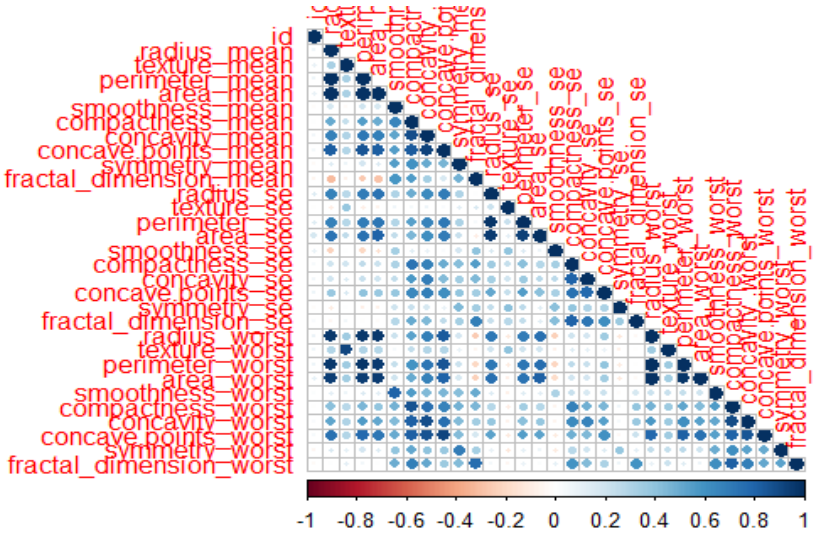
- ***Correlation Analysis:***

This step is to reduce the data dimension and decrease the number of predictors by finding the correlation between the variables. Correlation defines the mutual relationship or association between the variables. It helps finding the redundancies among the features. Some classifiers assume feature independence. For classifier like kNN, adding more redundant features will artificially inflate their importance. Using the Pearson correlation method, the coefficients are calculated for breast cancer dataset. Then filter out the columns that have correlation value greater than 0.9.

```
> #Find thecorrelation between variables
> cor_1<-cor(bc_data_1[-1],method="pearson")
> corrplot(cor_1,method="number",type="lower")
> corrplot(cor_1,type="lower")
```

Eliminate all high correlated features:

```
> highly_correlated<-findCorrelation(cor_1,cutoff=0.9)
> print(highly_correlated)
 [1]  8  9 24 22  4 25  2 14 15  3
> exclude<-colnames(cor_1)[highly_correlated]
>
```



Correlation between different features

After excluding 10 variables, I ended up with 21 predictors + the target variable "diagnosis" as the final result:

```
> bc_data_new <- as.tibble(bc_data_new)
> head(bc_data_new)
# A tibble: 6 x 22
  diagnosis     id area_mean smoothness_mean compactness_mean symmetry_mean
  <fct>      <dbl>     <dbl>           <dbl>            <dbl>         <dbl>
1 M          8.42e5     1001           0.118            0.278         0.242
2 M          8.43e5     1326           0.0847           0.0786        0.181
3 M          8.43e7     1203           0.110            0.160         0.207
4 M          8.43e7      386.          0.142            0.284         0.260
5 M          8.44e7     1297           0.100            0.133         0.181
6 M          8.44e5      477.          0.128            0.17          0.209
# ... with 16 more variables: fractal_dimension_mean <dbl>, radius_se <dbl>,
#   texture_se <dbl>, smoothness_se <dbl>, compactness_se <dbl>, concavity_se <dbl>,
#   concave.points_se <dbl>, symmetry_se <dbl>, fractal_dimension_se <dbl>,
#   texture_worst <dbl>, smoothness_worst <dbl>, compactness_worst <dbl>,
#   concavity_worst <dbl>, concave.points_worst <dbl>, symmetry_worst <dbl>,
#   fractal_dimension_worst <dbl>
```

```
> str(bc_data_new)
Classes 'tbl_df', 'tbl' and 'data.frame':       569 obs. of  22 variables:
 $ diagnosis              : Factor w/ 2 levels "B","M": 2 2 2 2 2 2 2 2 2 2 ...
 $ id                     : num  842302 842517 84300903 84348301 84358402 ...
 $ 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 ...
 $ 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 ...
 $ 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 ...
 $ texture_worst          : num  17.3 23.4 25.5 26.5 16.7 ...
 $ 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 ...
```

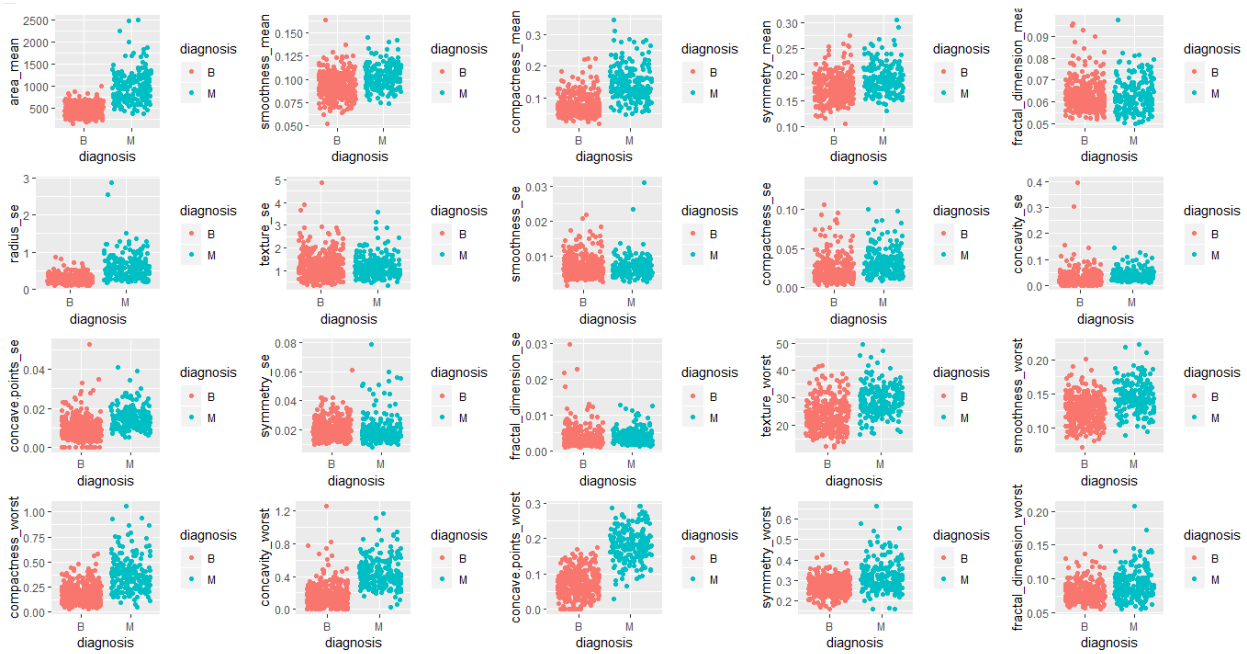
- ***Jitter plot Analysis:***

  The purpose of using jitter plot is to add a random noise to a numerical vector according to our dataset and it is used to better visualize overlapping values, such as integer covariates. Therefore, using jitter plot I' am going to identify the features that best distinguish the values of outcome variable. In order to visualize the jitter plot, I used a function loop "for" by specifying the length of the dataset column and that will plot all the features plot in one display.

```
> var_list<-names(bcdata_new)
  library("gridExtra")

  for (i in 2:length(var_list)) {
      gs <- lapply(3:22, function(i)
          ggplot(data=bcdata_new,aes_string(x=var_list[[2]], y=var_list[[i]])) +
              geom_jitter(mapping = aes(color=diagnosis)))
  }
  grid.arrange(grobs=gs, ncol=5)
```

From the plot, ***area_mean, smoothness_mean, compactness_mean, symmetry_mean, concave points_se, texture_worst, smoothness_worst, concave points_worst, symmetry_worst, fractal_dimension_worst*** are good predictors for our model as they are very correlated with the output variable. ***concavity_se, symmetry_se*** and ***fractal_dimension_se*** seem to be not good at classifying the output variable.



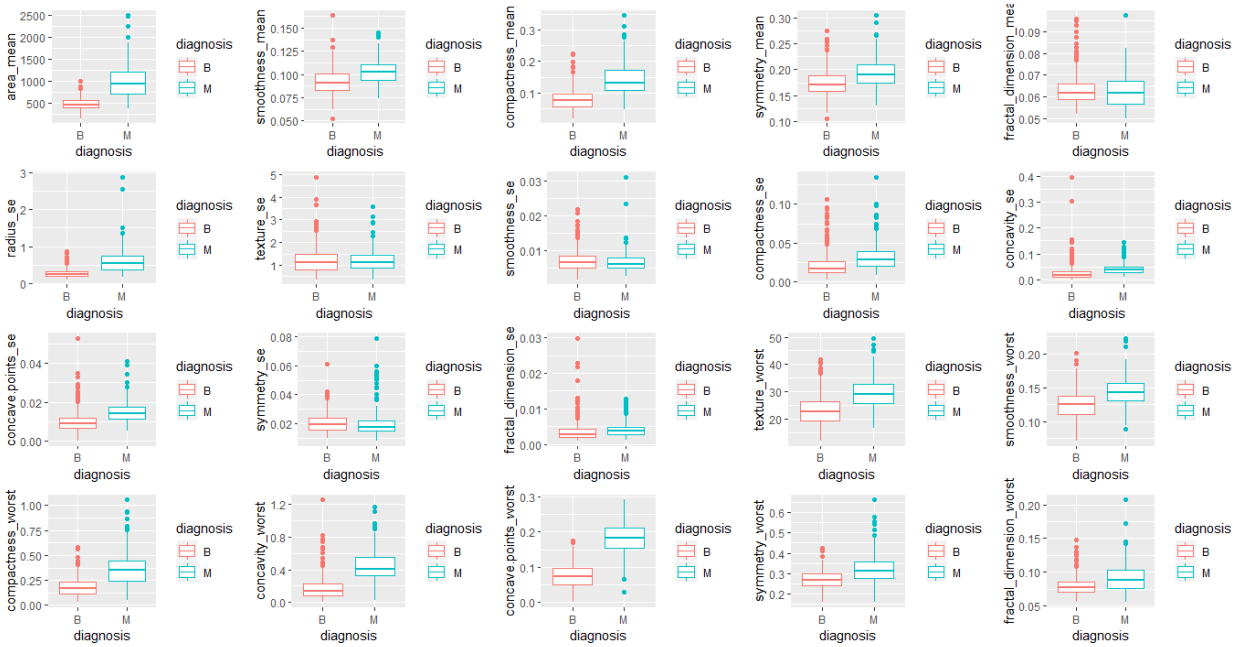
- ***Boxplot Analysis:***

```
> for (i in 2:length(var_list)) {
+   gs <- lapply(3:22, function(i)
+     ggplot(data = bcdata_new,aes_string(x=var_list[[2]], y=var_list[[i]])) +
+       geom_boxplot(mapping = aes(color=diagnosis)))
+
+ }
> grid.arrange(grobs=gs, ncol=5)
```

The use of Box plots here is to check the data distribution of the variables. Data distribution of, ***radius_se, texture_se, concavity_se, symmetry_se, compactness_se, fractal_dimension_se***, ***concavity_worst*** are highly skewed.

There are far more outliers for ***fractal_dimension_mean, texture_se, smoothness_se, compactness_se, symmetry_se, fractal_dimension_se, concavity_worst***.

Box plot also shows that ***area_mean, smoothness_mean, compactness_mean, symmetry_mean, concave points_se, texture_worst, smoothness_worst, concave points_worst, symmetry_worst, fractal_dimension_worst*** can act as <u>good predictors.</u>



❖ Normalize the data:

Raw data consists of features with varying scales. For algorithms like KNN where distance between pairs of samples is measured, if distinctive features have different range of values, it can often lead to biasing results or overfitting of the model. In order to avoid this, range of the independent variables need to be standardized. This is done as a part of data pre-processing step. Many machine learning algorithms (such as SVM, K-nearest neighbors, and logistic regression) assume that data is normalized.

```
> set.seed(123)
> trainIndex <- createDataPartition(bcdata_new$diagnosis, p = 0.8, list = FALSE, times = 1)
> bcdata_Train <- bcdata_new[trainIndex, ]
> bcdata_Test <- bcdata_new[-trainIndex, ]
>
> scaler <- preProcess(bcdata_Train, method = c("center", "scale"))
> bcdata_Train <- predict(scaler, bcdata_Train)
> bcdata_Test <- predict(scaler, bcdata_Test)
```

❖ Features & Metric Selection:

From the above plots ***area_mean, smoothness_mean, compactness_mean, symmetry_mean, concave points_se, texture_worst, smoothness_worst, concave points_worst, symmetry_worst, fractal_dimension_worst*** are selected for building the first round of our model.

```
> bcdata_Train <- select(bcdata_Train,
+                        area_mean,
+                        smoothness_mean,
+                        compactness_mean,
+                        symmetry_mean,
+                        concave.points_se,
+                        texture_worst,
+                        smoothness_worst,
+                        concave.points_worst,
+                        symmetry_worst,
+                        fractal_dimension_worst,
+                        diagnosis)
```

❖ Best Model Results:

After the recursive forward feature selection, the effective model that is selected has **97.35**% accuracy. The model has 5 predictors namely **concave.points_worst + area_mean + symmetry_worst + texture_worst + compactness_mean**. Initially my view from the box plots and jitter plots was correct as their distribution let them be good predictors.

Confusion matrix of the final model is as shown below. Model could predict 40 out of 42 malignant and 70 out 71 benign classes correctly.

```
> knn_concave.points_worst_area_mean_symmetry_worst_texture_worst_compactness_mean <- train(diagnosis ~ concave.points_worst +
+                                                                                            area_mean +
+                                                                                            symmetry_worst +
+                                                                                            texture_worst +
+                                                                                           compactness_mean,
+                                                                                          data = bcdata_Train, method = "knn")
> TestPredictions_concave.points_worst_area_mean_symmetry_worst_texture_worst_compactness_mean <- predict(
+                         knn_concave.points_worst_area_mean_symmetry_worst_texture_worst_compactness_mean, bcdata_Test)
> confusionMatrix(TestPredictions_concave.points_worst_area_mean_symmetry_worst_texture_worst_compactness_mean, bcdata_Test$diagnosis)
Confusion Matrix and Statistics

          Reference
Prediction  B  M
         B 70  2
         M  1 40

               Accuracy : 0.9735
                 95% CI : (0.9244, 0.9945)
    No Information Rate : 0.6283
    P-Value [Acc > NIR] : <2e-16
```

- ***Rounds results:***
  - ***Round #1***

Features	Accuracies
area_mean	0.8938
smoothness_mean	0.6814
compactness_mean	0.8053
symmetry_mean	0.6195
concave.points_se	0.6726
texture_worst	0.7257
smoothness_worst	0.646
concave.points_worst	0.9204
symmetry_worst	0.6991
fractal_dimension_worst	0.6549

  - ***Round #2***

Features	KNN Accuracies
concave.points_worst + area_mean	0.9646
concave.points_worst + compactness_mean	0.9204
concave.points_worst + texture_worst	0.9027
concave.points_worst + symmetry_worst	0.9292

  - ***Round #3***

Features	KNN Accuracies
concave.points_worst + area_mean + compactness_mean	0.9381
concave.points_worst + area_mean + texture_worst	0.9381
concave.points_worst + area_mean + symmetry_worst	0.9469

I noticed most of features accuracies are going up, but the highest accuracy is going a little bit down. However, I decided to continue and see how it going to be the accuracies for the next round.

- ***Round #4***

Features	KNN Accuracies
concave.points_worst + area_mean + symmetry_worst + compactness_mean	0.9469
concave.points_worst + area_mean + symmetry_worst + texture_worst	0.9735

- ***Round #5***

Features	KNN Accuracies
concave.points_worst + area_mean + symmetry_worst + texture_worst + compactness_mean	0.9735

❖ Reflection:

- ***Were you successful at building a predictive model?***

  Based on the accuracies that I found during all the rounds and the accuracy of the best model, I think I was successful to build a predictive model of a highest accuracy 97.35%.

- ***Did you face any challenges?***

  The most challenge that I came through into my project is when I had a lot of features (33 predictors) and I had to minimize their number as much as possible so then I can select the most significant once to build my predictive model. I tried many methods like eliminate near zero variance variables and remove linear combinations, but they didn't work for me as they did remove a small number of predictors. Thus, I decided to find the high correlated variables and eliminate them which I think it was a good choice by removing 11 bad predictors.

- ***In this project, what skills did you employ? What skills do you think you can improve upon in the future? How might you go about improving those skills?***

  - In this project, I tried to follow the important steps in data mining process such as first I determined my data mining task which is breast cancer diagnosis prediction (malign or benign). Second, the step of data preprocessing by handling missing values even my dataset does not have, then the step of data exploration and visualization, by creating charts such as jitter plots and boxplots. Third, I partition the data into training and test sets then the step of feature selection of the most good predictors. Fourth, I choose KNN algorithm as the technique to build the predictive model and to perform the task. Finally, I succefuly built the model and I ended up by defining ***concave .points_worst + area_mean + symmetry_worst + texture_worst + compactness_mean*** is the best model in this dataset.

  - I think I can improve my model in the future if I choose different algorithms with KNN algorithm and then compare the results between them, and from that I choose the

model that have highest accuracy between different algorithms. For example, comparing the accuracy results between logistic regression using glmnet (family = binomial) and KNN algorithm as I noticed through the different assignments that logistic regression is more accurate and can be good classifier better that KNN to build my model.

- In addition, in forward selection step, I can use the method of backward selection without need of finding the highest correlated features in order to eliminate them because using backward method can automatically eliminate the least significant features and keeps only the most significant once and then I can build my model using these features results.

❖ References:

1. https://www.kaggle.com/uciml/breast-cancer-wisconsin-data/home
2. https://data.world/health/breast-cancer-wisconsin/workspace/file?filename=DatasetDescription.txt
3. https://cran.r-project.org/web/packages/caret/caret.pdf
4. https://www.researchgate.net/publication/311950799_Analysis_of_the_Wisconsin_Breast_Cancer_Dataset_and_Machine_Learning_for_Breast_Cancer_Detection