Wisconsin Diagnostic Breast Cancer - Machine Learning Models

Hieu Nguyen - hieunguyenthanh@gmail.com

10/4/2021

Contents

Introduction **2**

Goals . . . 2

Dataset . . . 2

Key steps . . . 3

WDBC dataset analysis **4**

Installing and loading required libraries . . . 4

WDBC Dataset download . . . 4

Data exploration and visualisation . . . 5

Approach to Machine Learning modelling . . . 14

Data Preparation . . . 14

Running Machine Learning Models . . . 15

Results . . . 18

Conclusion **19**

Summary . . . 19

Future work . . . 19

References **20**

Introduction

The aim of this project is to use Machine Learning models to predict whether or not a patient has breast cancer based on list of features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

Goals

Prediction accuracy will be used as key metric to measure performance of difference Machine Learning models.

Our objective is to analyze dataset and find best Machine Learning models that return highest prediction accuracy.

Dataset

We will use Wisconsin Diagnostic Breast Cancer (WDBC) Data Set in this project. Detail information of this dataset can be found on this page https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

Dataset information:

- Number of instances: 569
- Number of attributes: 32 (ID, diagnosis, 30 real-valued input features)
- Attribute information:
    1. ID number
    2. Diagnosis (M = malignant, B = benign)
    3. 3-32: Ten real-valued features are computed for each cell nucleus:
        a) radius (mean of distances from center to points on the perimeter)
        b) texture (standard deviation of gray-scale values)
        c) perimeter
        d) area
        e) smoothness (local variation in radius lengths)
        f) compactness (perimeter^2 / area - 1.0)
        g) concavity (severity of concave portions of the contour)
        h) concave points (number of concave portions of the contour)
        i) symmetry
        j) fractal dimension (“coastline approximation” - 1)

The mean, standard error, and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features.
For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.
All feature values are recoded with four significant digits.

- Missing attribute values: none
- Class distribution: 357 benign, 212 malignant

Key steps

The key steps are the following:

1. Installation of required packages and loading of libraries
2. Downloading and formatting the dataset for further processing
3. Analyze to understand the dataset and get insights of its features to be used in different Machine Learning models
4. Partitioning of dataset into training and testing data
5. Training different models on the training dataset, and testing them on the testing dataset
6. Evaluation of different Mechine Learning models through computation of prediction accuracy
7. Reporting of results

WDBC dataset analysis

Installing and loading required libraries

Loading required package libraries to use its functions in our WDBC data analysis

```
### Check and install libraries if not exist
if(!require(tidyverse)) install.packages("tidyverse",
                                         repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret",
                                     repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table",
                                          repos = "http://cran.us.r-project.org")
if(!require(matrixStats)) install.packages("matrixStats",
                                           repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2",
                                       repos = "http://cran.us.r-project.org")
if(!require(reshape2)) install.packages("reshape2",
                                        repos = "http://cran.us.r-project.org")
if(!require(psych)) install.packages("psych",
                                     repos = "http://cran.us.r-project.org")

### Load libraries
library(tidyverse)
library(caret)
library(data.table)
library(matrixStats)
library(ggplot2)
library(reshape2)
library(psych)
```

WDBC Dataset download

WDBC dataset can be download from Machine Learning Repository, UCI: http://archive.ics.uci.edu/ml/index.php. Dataset is in form of comma-separated so we can use read.csv() function to read.

We will use features as described in dataset to name the columns and converting **diagnostic** values from character **"B"/"M"** to integer **0/1** respectively so that we can do some mathematical processing.

```
dl <- tempfile()
url <- "https://archive.ics.uci.edu/ml/machine-learning-databases"
file <- "breast-cancer-wisconsin/wdbc.data"
download.file(paste(url, file, sep = "/"),dl)

data_raw <- read.csv(dl, header = FALSE)

features_list <- c("radius", "texture", "perimeter", "area", "smoothness",
                   "compactness", "concavity", "concave_points", "symmetry",
                   "fractal_dimension")

colnames(data_raw) <- c("ID", "diagnostic",
                    paste(features_list,"mean", sep = "_"),
                    paste(features_list,"se", sep = "_"),
```

```
                    paste(features_list, "worst", sep = "_"))

# Remove ID column and store data in wdbc
wdbc <- data_raw[-1]

# Convert diagnostic value from B/M to 0/1
wdbc[which(wdbc[,1] == "B"),1] <- 0
wdbc[which(wdbc[,1] == "M"),1] <- 1
wdbc[,1] <- strtoi(wdbc[,1])  # convert character to integer
wdbc <- as.matrix(wdbc) # convert to matrix class

rm(dl, features_list, url, file)
```

Data exploration and visualisation

Structure of dataset by using str() function:

```
str(data_raw)
```

```
## 'data.frame':    569 obs. of  32 variables:
##  $ ID                     : int  842302 842517 84300903 84348301 84358402 843786 844359 84458202 844
##  $ diagnostic             : 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 ...
```

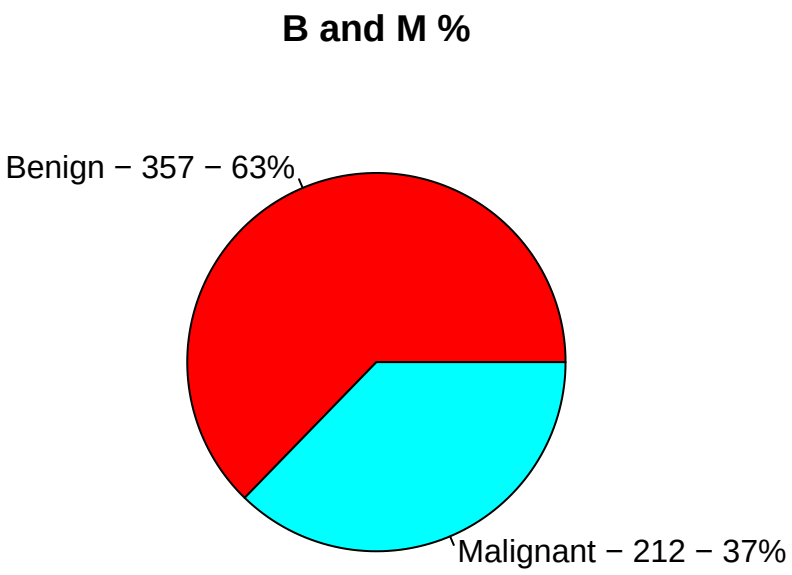
We observe that dataset has 569 observation and 32 variables which are ID of each observation, diagnostic outputs and 30 features of each observation.

Number of Benign and Malignant in **diagnostic** output and its ratio:

```
data_raw %>% group_by(diagnostic) %>%
  summarize(count = n(), count_percent = round(count/nrow(data_raw)*100)) %>%
  knitr::kable()
```

diagnostic	count	count_percent
B	357	63
M	212	37

```
# Plot PIE chart for % of B and M
sl <- c(sum(data_raw$diagnostic == "B"), sum(data_raw$diagnostic == "M"))
lbs <- c("Benign", "Malignant")
sl_pct <- round(sl / sum(sl) * 100)
lbs <- paste(lbs, sl, sep = " - ")
lbs <- paste(lbs, sl_pct, sep = " - ")
lbs <- paste(lbs, "%", sep ="")
pie(sl, labels = lbs, col = rainbow(length(sl)), main = "B and M %")
```



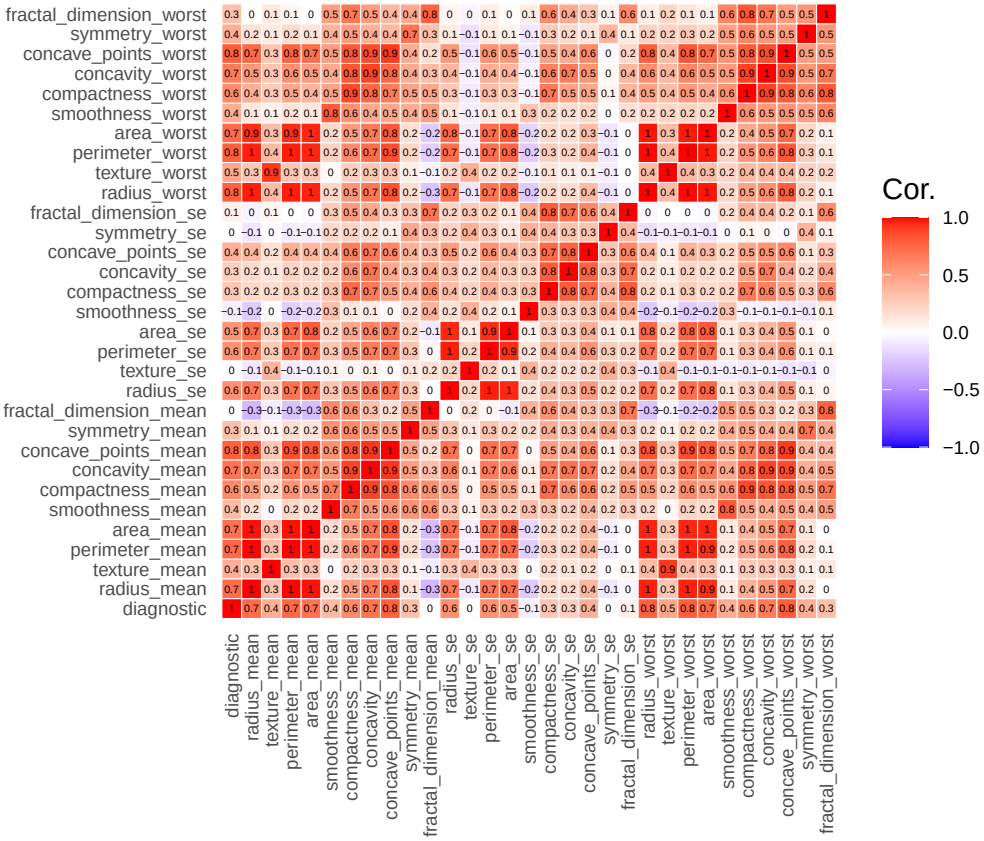
There are 357 benign and 212 malignant outputs in **diagnostic**, ratio is 63% and 37% respectively. We should pay attention to this ratio to ensure keep same ratio in training and testing data when we split dataset to train Machine Learning models and test its predictions.

Calculate the correlation among variables in dataset and visualize on heatmap:

```
wdbc_cor_melted <- melt(cor(wdbc))

ggheatmap <- ggplot(data = wdbc_cor_melted, aes(Var2, Var1, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0,
                       limit = c(-1,1), space = "Lab", name="Cor.") + theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust =1, vjust = 0.5, size = 7),
        axis.text.y = element_text(size = 7), legend.text = element_text(size = 7)) +
  coord_fixed()

ggheatmap + geom_text(aes(Var2, Var1, label=round(value,1)), color="black", size=1.3) +
  theme(axis.title.x = element_blank(), axis.title.y = element_blank(),
        panel.grid.major = element_blank(), panel.border = element_blank(),
        panel.background = element_blank(), axis.ticks = element_blank())
```



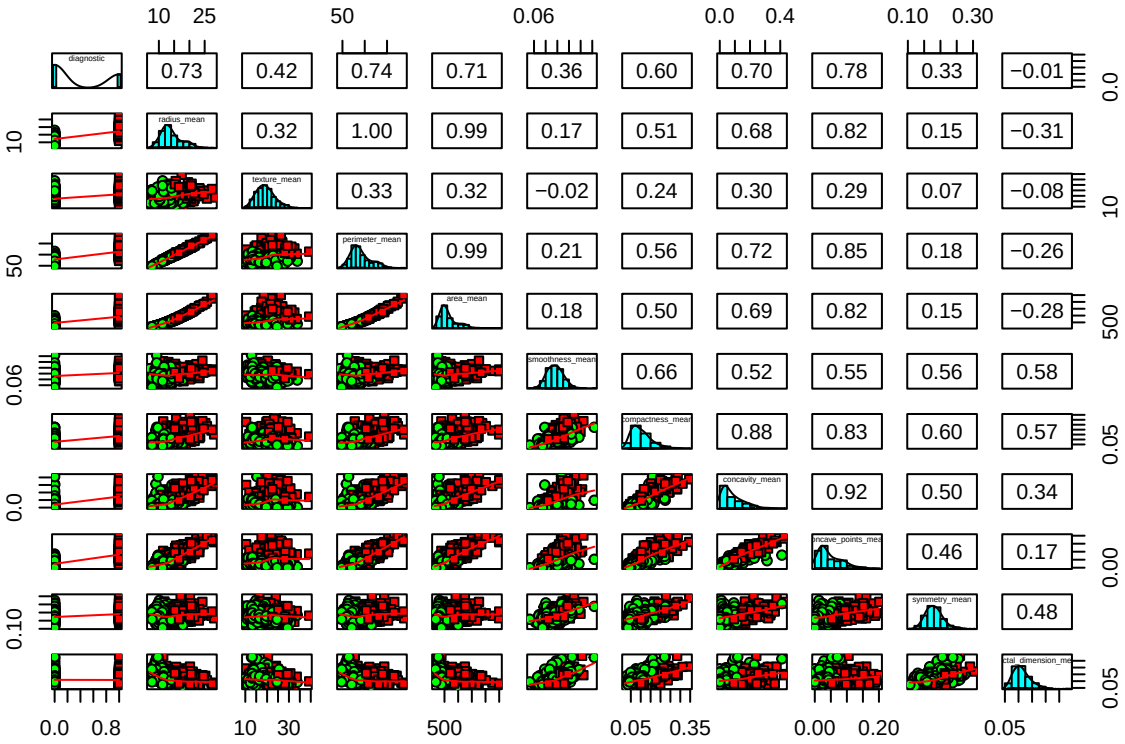
We observe that some variables are highly correlated to each others, like:

- perimeter_ and area_ are highly correlated with radius_
- all _worst features except symmetry_worst and fractal_dimension_worst are highly correlated to _mean features respectively
- concavity_mean is highly correlated with concave_points_mean

We make pairs-plot among the variables to see more detail of correlation:

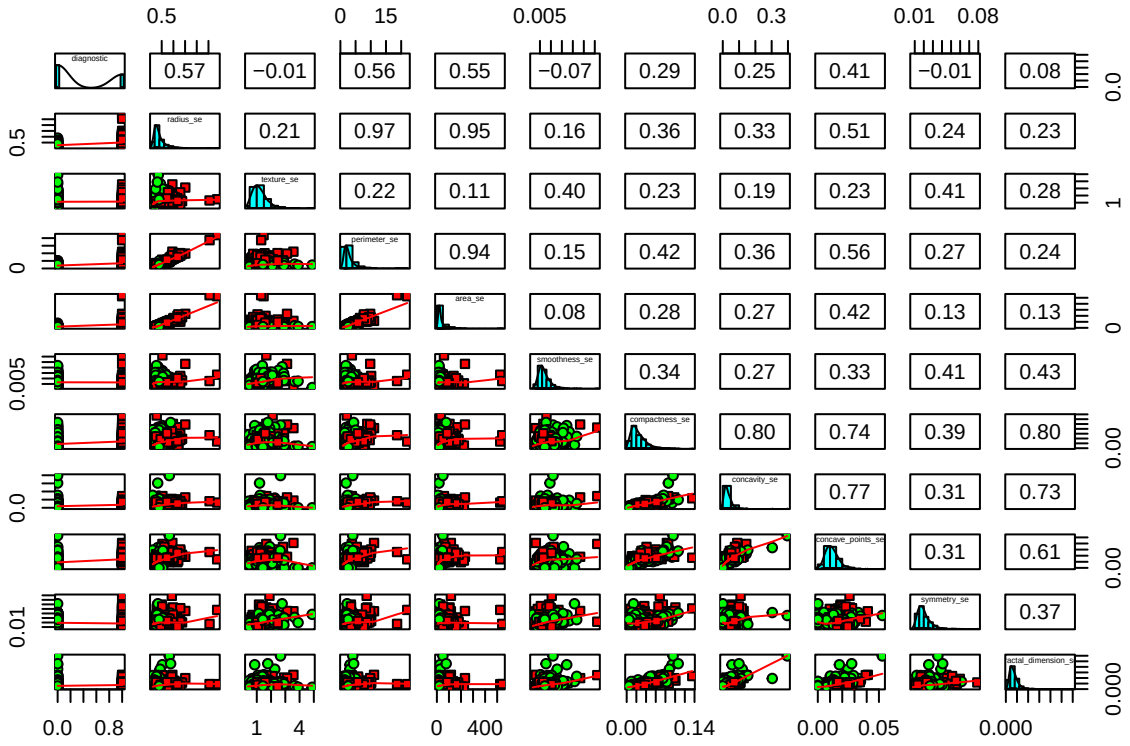
Pairs plot of Diagnostic and all **mean** features

```
pairs.panels(wdbc[,c(1:11)], pch = 21 + as.numeric(wdbc[,1]),
             bg = c("green", "red")[wdbc[,1] + 1], ellipses = FALSE,
             cex.cor = 1.2, cex.labels = 0.4)
```



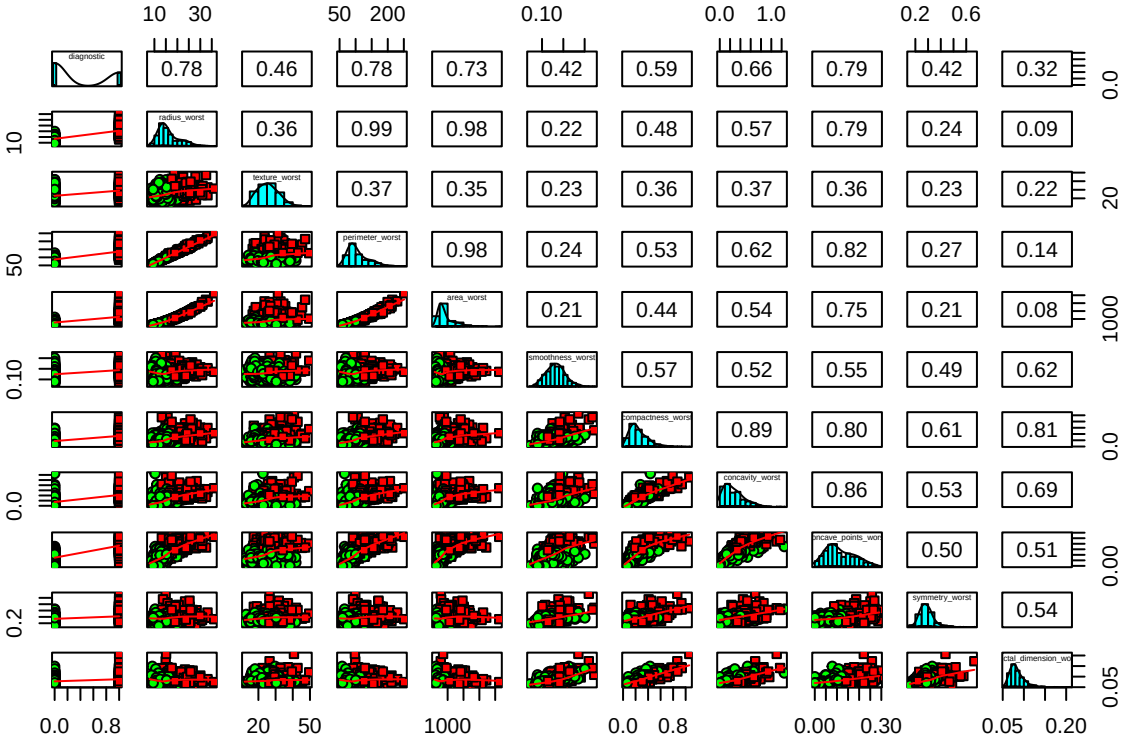
Pairs Plot of Diagnostic and all **se** features

```
pairs.panels(wdbc[,c(1, 12:21)], pch = 21 + as.numeric(wdbc[,1]),
             bg = c("green", "red")[wdbc[,1] + 1], ellipses = FALSE,
             cex.cor = 1.2, cex.labels = 0.4)
```



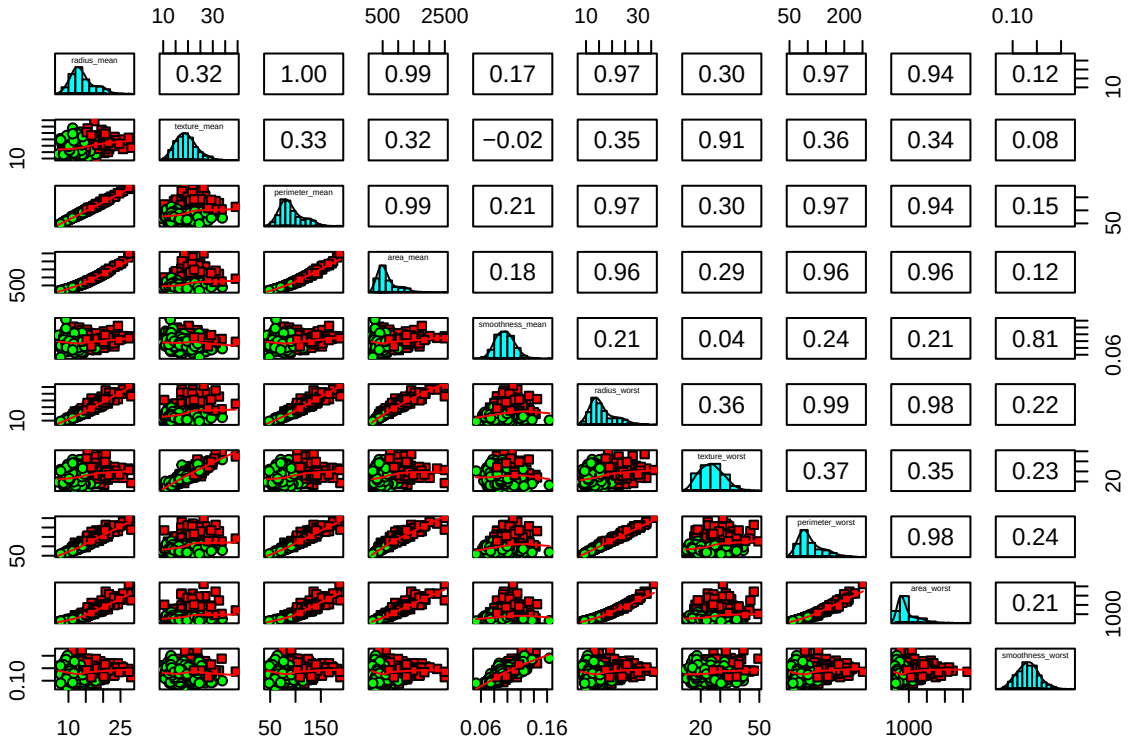
Pairs Plot of Diagnostic and all **worst** features

```
pairs.panels(wdbc[,c(1, 22:31)], pch = 21 + as.numeric(wdbc[,1]),
             bg = c("green", "red")[wdbc[,1] + 1], ellipses = FALSE,
             cex.cor = 1.2, cex.labels = 0.4)
```



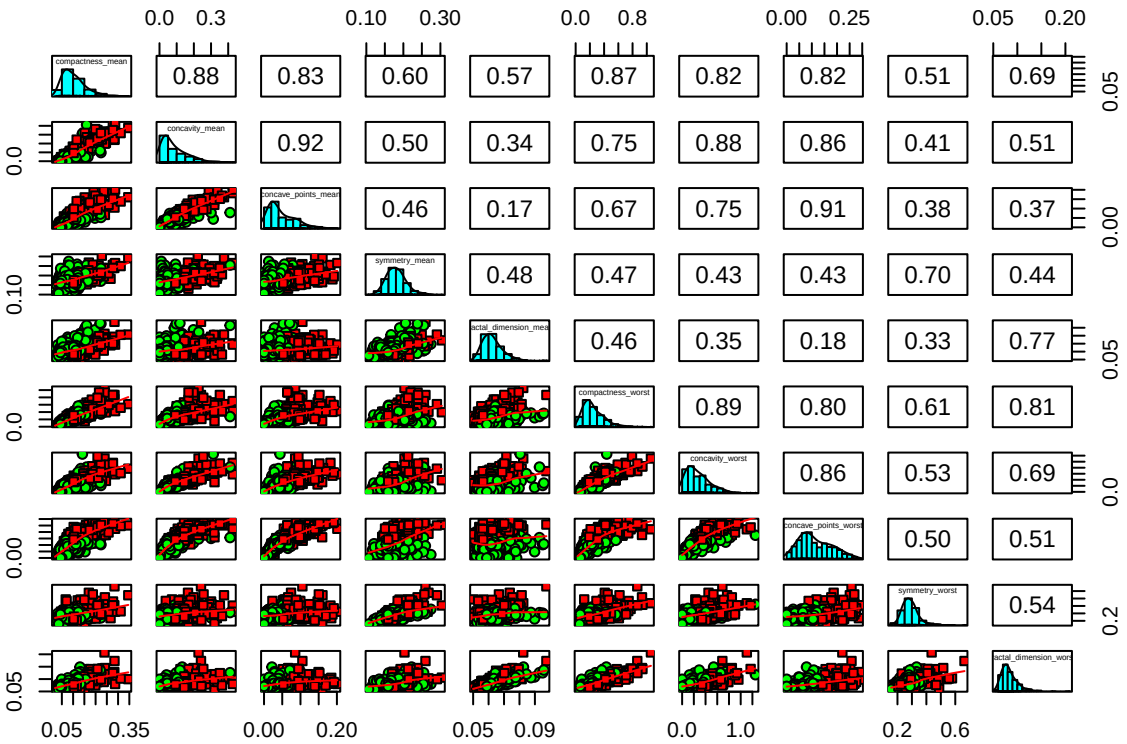
Pairs Plot of fist 5 **mean** and **worst** features

```
pairs.panels(wdbc[,c(2:6, 22:26)], pch = 21 + as.numeric(wdbc[,1]),
             bg = c("green", "red")[wdbc[,1] + 1], ellipses = FALSE,
             cex.cor = 1.2, cex.labels = 0.4)
```



Pairs Plot of last 5 **mean** and **worst** features

```
pairs.panels(wdbc[,c(7:11, 27:31)], pch = 21 + as.numeric(wdbc[,1]),
             bg = c("green", "red")[wdbc[,1] + 1], ellipses = FALSE,
             cex.cor = 1.2, cex.labels = 0.4)
```



Highly correlated among variables will lead to multicollinearity which may have negative affect on Machine Learning models prediction results. Removing highly correlated variables is one of solutions to deal with multicollinearity.

We will remove these variables:

- perimeter_ and area_
- concave_points_mean
- all _worst features except symmetry_worst and fractal_dimension_worst

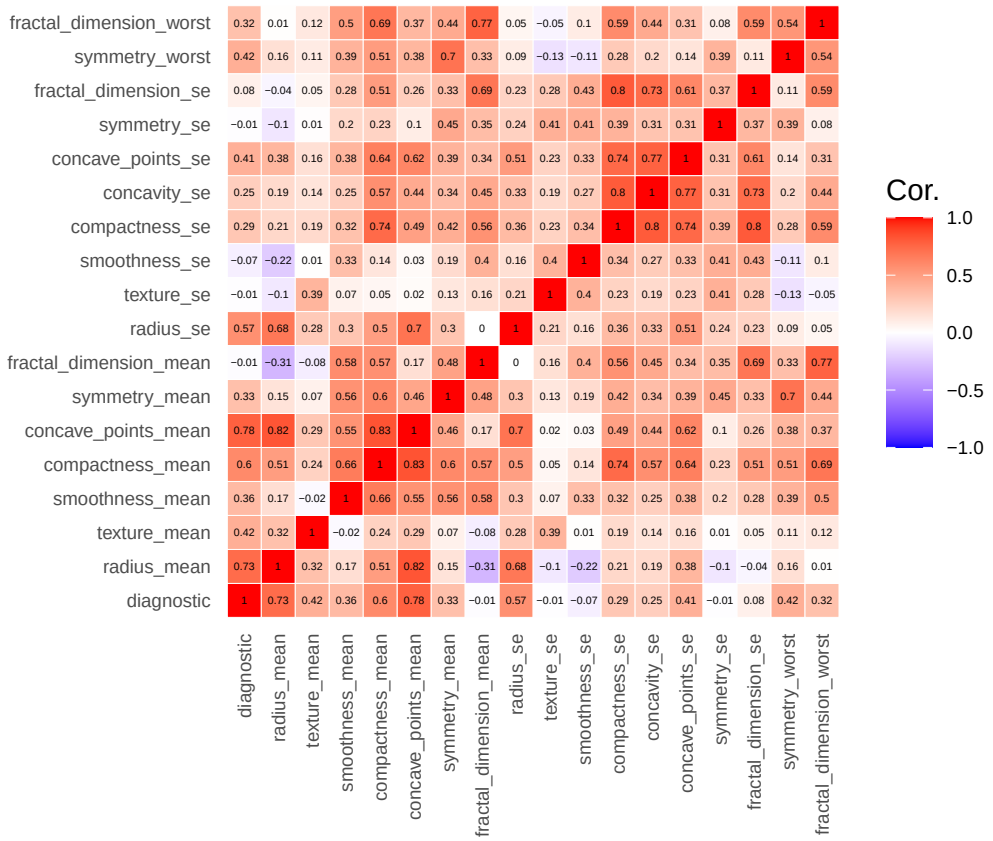
and visualize correlation matrix on heatmap

```
wdbc_cor_dropped_melted <- melt(cor(wdbc[,c(-4,-5,-8,-14,-15,-22:-29)]))

ggheatmap_dropped <- ggplot(data = wdbc_cor_dropped_melted,
                            aes(Var2, Var1, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0,
```

```
                      limit= c(-1,1), space= "Lab", name= "Cor.") + theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5, size = 7),
        axis.text.y = element_text(size = 7),legend.text = element_text(size = 7)) +
  coord_fixed()

ggheatmap_dropped + geom_text(aes(Var2, Var1, label = round(value,2)),
                              color = "black", size = 1.3) +
  theme(axis.title.x = element_blank(), axis.title.y = element_blank(),
        panel.grid.major = element_blank(), panel.border = element_blank(),
        panel.background = element_blank(), axis.ticks = element_blank())
```



As shown on heatmap, there are no highly correlation among variables

Approach to Machine Learning modelling

Now we have two datasets:
* Original **wdbc** dataset with some highly correlated variables which lead to multicollinearity issues.
* **wdbc** dropped dataset: we removed some variables that highly correlated with others.

We will apply some Machine Learning models (glm, lda, qda, loess, knn, rf, and ensemble of these 6 models) on the two dataset and compare its prediction accuracy.

Data Preparation

Scaling data to standard distribution:

```
wdbc_scaled <- sweep(wdbc[,-1], 2, colMeans(wdbc[,-1]))
wdbc_scaled <- sweep(wdbc_scaled, 2, colSds(wdbc[,-1]), FUN = "/")
```

Creating training and testing data from two dataset:

```
set.seed(1, sample.kind = "Rounding")
test_index <- createDataPartition(y = as.factor(wdbc[,1]),
                                  times = 1, p = 0.25, list = FALSE)

train_x <- wdbc_scaled[-test_index,]
train_x_dropped <- train_x[,c(-3,-4,-7,-13,-14,-21:-28)]
train_y <- as.factor(wdbc[-test_index, 1])

test_x <- wdbc_scaled[test_index,]
test_x_dropped <- test_x[,c(-3,-4,-7,-13,-14,-21:-28)]
test_y <- as.factor(wdbc[test_index, 1])
```

Training and testing data percentage of original dataset:

```
data.frame(total = nrow(data_raw),
           train = nrow(train_x),
           train_percent = round(nrow(train_x)/nrow(data_raw)*100),
           test = nrow(test_x),
           test_percent = round(nrow(test_x)/nrow(data_raw)*100),
           row.names = "Number of observation") %>%
  knitr::kable()
```

	total	train	train_percent	test	test_percent
Number of observation	569	426	75	143	25

Cross check the ratio of B and M in training and testing data to ensure similar with original ratio data:

```
data.frame(
  diagnostic = c("B", "M"),
  total = c(sum(data_raw$diagnostic == "B"), sum(data_raw$diagnostic == "M")),
  total_percent = c(round(sum(data_raw$diagnostic == "B")/nrow(data_raw)*100),
                    round(sum(data_raw$diagnostic == "M")/nrow(data_raw)*100)),
```

```
  train_y = c(sum(train_y == 0), sum(train_y == 1)),
  train_y_per = c(round(sum(train_y == 0)/length(train_y)*100),
                  round(sum(train_y == 1)/length(train_y)*100)),
  test_y = c(sum(test_y == 0), sum(test_y == 1)),
  test_y_per = c(round(sum(test_y == 0)/length(test_y)*100),
                 round(sum(test_y == 1)/length(test_y)*100))) %>%
  knitr::kable()
```

diagnostic	total	total_percent	train_y	train_y_per	test_y	test_y_per
B	357	63	267	63	90	63
M	212	37	159	37	53	37

We can see that ratio of B and M in training/ testing dataset are same as ratio in original dataset.

Running Machine Learning Models

We will run different Machine Learning models on two dataset to compare its prediction accuracy results.

Logistic regression model

```
set.seed(1, sample.kind = "Rounding")
train_glm <- train(train_x, train_y, method = "glm")
glm_preds <- predict(train_glm, test_x)

set.seed(1, sample.kind = "Rounding")
train_glm_dropped <- train(train_x_dropped, train_y, method = "glm")
glm_preds_dropped <- predict(train_glm_dropped, test_x_dropped)

accuracy_table <- data.frame(
  glm = c(mean(glm_preds == test_y), mean(glm_preds_dropped == test_y)),
  row.names = c("Full Features","Features-Dropped"))
```

LDA model

```
set.seed(1, sample.kind = "Rounding")
train_lda <- train(train_x, train_y, method = "lda")
lda_preds <- predict(train_lda, test_x)

set.seed(1, sample.kind = "Rounding")
train_lda_dropped <- train(train_x_dropped, train_y, method = "lda")
lda_preds_dropped <- predict(train_lda_dropped, test_x_dropped)

accuracy_table <- accuracy_table %>% mutate(lda = c(mean(lda_preds == test_y),
                                                    mean(lda_preds_dropped == test_y)))
```

QDA model

```
set.seed(1, sample.kind = "Rounding")
train_qda <- train(train_x, train_y, method = "qda")
qda_preds <- predict(train_qda, test_x)
```

```
set.seed(1, sample.kind = "Rounding")
train_qda_dropped <- train(train_x_dropped, train_y, method = "qda")
qda_preds_dropped <- predict(train_qda_dropped, test_x_dropped)

accuracy_table <- accuracy_table %>% mutate(qda = c(mean(qda_preds == test_y),
                                                    mean(qda_preds_dropped == test_y)))
```

Loess model

```
set.seed(1, sample.kind = "Rounding")
train_loess <- train(train_x, train_y, method = "gamLoess")
loess_preds <- predict(train_loess, test_x)

set.seed(1, sample.kind = "Rounding")
train_loess_dropped <- train(train_x_dropped, train_y, method = "gamLoess")
loess_preds_dropped <- predict(train_loess_dropped, test_x_dropped)

accuracy_table <- accuracy_table %>% mutate(loess = c(mean(loess_preds == test_y),
                                                      mean(loess_preds_dropped == test_y)))
```

K-nearest neighbors model

```
set.seed(1, sample.kind = "Rounding")
train_knn <- train(train_x, train_y,
                   method = "knn",
                   tuneGrid = data.frame(k = seq(3, 21, 2)))
knn_preds <- predict(train_knn, test_x)

set.seed(1, sample.kind = "Rounding")
train_knn_dropped <- train(train_x_dropped, train_y,
                           method = "knn",
                           tuneGrid = data.frame(k = seq(3, 21, 2)))
knn_preds_dropped <- predict(train_knn_dropped, test_x_dropped)

accuracy_table <- accuracy_table %>% mutate(knn = c(mean(knn_preds == test_y),
                                                    mean(knn_preds_dropped == test_y)))
```

Random forest model

```
set.seed(1, sample.kind = "Rounding")
train_rf <- train(train_x, train_y,
                  method = "rf",
                  tuneGrid = data.frame(mtry = c(3, 5, 7, 9)),
                  importance = TRUE)
rf_preds <- predict(train_rf, test_x)

set.seed(1, sample.kind = "Rounding")
train_rf_dropped <- train(train_x_dropped, train_y,
                          method = "rf",
                          tuneGrid = data.frame(mtry = c(3, 5, 7, 9)),
                          importance = TRUE)
rf_preds_dropped <- predict(train_rf_dropped, test_x_dropped)
```

```
accuracy_table <- accuracy_table %>% mutate(rf = c(mean(rf_preds == test_y),
                                                   mean(rf_preds_dropped == test_y)))
```

Ensemble Model: combination of six models (glm, lda, qda, loess, knn, rf) to predict output

```
ensemble <- cbind(glm = glm_preds == 0, lda = lda_preds == 0,
                  qda = qda_preds == 0, loess = loess_preds == 0,
                  knn = knn_preds == 0, rf = rf_preds == 0)

ensemble_preds <- ifelse(rowMeans(ensemble) > 0.5, 0, 1)

ensemble_dropped <- cbind(glm_dropped = glm_preds_dropped == 0,
                  lda_dropped = lda_preds_dropped == 0,
                  qda_dropped = qda_preds_dropped == 0,
                  loess_dropped = loess_preds_dropped == 0,
                  knn_dropped = knn_preds_dropped == 0,
                  rf_dropped = rf_preds_dropped == 0)

ensemble_preds_dropped <- ifelse(rowMeans(ensemble_dropped) > 0.5, 0, 1)

accuracy_table <- accuracy_table %>% mutate(ensemble = c(mean(ensemble_preds == test_y),
                                                   mean(ensemble_preds_dropped == test_y)))
```

Ensemble Mixed Model: combination highest prediction accuracy of six models (glm, lda, qda, loess, knn, rf) to predict output

```
# Create Ensemble Mixed model by:
#   - selecting highest prediction accuracy between two datasets, then
#   - combining results from 6 models (glm, lda, qda, loess, knn, rf)
#     to predict output
if (accuracy_table[1,1] > accuracy_table[2,1]) {
  glm_mixed <- glm_preds == 0} else {glm_mixed <- glm_preds_dropped == 0}

if (accuracy_table[1,2] > accuracy_table[2,2]) {
  lda_mixed <- lda_preds == 0} else {lda_mixed <- lda_preds_dropped == 0}

if (accuracy_table[1,3] > accuracy_table[2,3]) {
  qda_mixed <- qda_preds == 0} else {qda_mixed <- qda_preds_dropped == 0}

if (accuracy_table[1,4] > accuracy_table[2,4]) {
  loess_mixed <- loess_preds == 0} else {loess_mixed <- loess_preds_dropped == 0}

if (accuracy_table[1,5] > accuracy_table[2,5]) {
  knn_mixed <- knn_preds == 0} else {knn_mixed <- knn_preds_dropped == 0}

if (accuracy_table[1,6] > accuracy_table[2,6]) {
  rf_mixed <- rf_preds == 0} else {rf_mixed <- rf_preds_dropped == 0}

ensemble_mixed <- cbind(glm_mixed = glm_mixed, lda_mixed = lda_mixed,
                        qda_mixed = qda_mixed, loess_mixed = loess_mixed,
                        knn_mixed = knn_mixed, rf_mixed = rf_mixed)
```

```
ensemble_preds_mixed <- ifelse(rowMeans(ensemble_mixed) > 0.5, 0, 1)

accuracy_ensemble_mixed <- mean(ensemble_preds_mixed == test_y)
```

Results

Prediction accuracy of selected models applied on two dataset as shown in table below:

```
round(accuracy_table*100,3) %>% knitr::kable()
```

	glm	lda	qda	loess	knn	rf	ensemble
Full Features	94.406	95.804	97.203	93.706	98.601	98.601	98.601
Features-Dropped	95.105	95.105	97.203	95.804	96.503	97.902	98.601

Prediction Accuracy of Ensemble Mixed model is: **99.301**

We observe that:

- Models glm, loess: prediction accuracy of dataset without highly correlated variables (Features_Dropped) is higher than results on original dataset.
- Models lda, knn, rf: prediction accuracy of original dataset (Full Features) is higher than results of features-dropped dataset.
- Models qda, ensemble: both dataset have same prediction accuracy.
- Highest prediction accuracy is **99.301** achieved from ensemble-mixed models apply on both dataset.

Conclusion

Summary

We have successfully applying different Machine Learning models on WDBC dataset and can archive prediction accuracy upto **99.301**.

We also observe that, by removing highly correlated variables in dataset and build ensemble-mixed model, we can archive highest prediction accuracy.

Future work

1. Do further research to understand about affect of multicollinearity on each Machine Learning algorithms so that we can optimize our models and ensemble models to improve prediction accuracy.

2. Re-run these Machine Learning models again on different dataset with more observations or different ratio of training/testing on current dataset to check if Ensemble Mixed model is really can return highest prediction accuracy.

3. Research and apply other Machine Learning/ Deep learning algorithms on this dataset to improve prediction accuracy.

References

1. https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

2. https://rafalab.github.io/dsbook/introduction-to-machine-learning.html