

Regression applied to diagnosis prediction.

Alexandre Rosa

2023-09-20

What is the main objective of this code?

This code was built to analyze a database of cancer diagnosis with different classifications and using ETL methods, clean and explore the data. Also, using a *regression model based on different variables to predict the final diagnosis*.

Database analysis steps:

- Verify and install specific libraries.
- Assign our database to a data frame.
- Explore our data and check inconsistencies as null values.
- Split data into test and training set.
- Create the model based into the training dataset and explore the results.
- Evaluate model based on specific measurement tools.
- Create the model based into the test dataset and explore the results.
- Evaluate model based on specific measurement tools.
- Explore results obtained in data analysis.

What is the result of this data analysis?

Based on this analysis I was able to understand that we have 3 variables that are able to describe specific diagnosis and forecast more accurately and fast this results.

Resulting in an accuracy of more than >90% and MSE >80%. Also, that compactness mean, radius worst and area worst are the most important variables to split benign and malignant tumors.

```
knitr::opts_chunk$set(echo = TRUE)
```

CODE

Database: <https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic>

Paper: <https://minds.wisconsin.edu/bitstream/handle/1793/59692/TR1131.pdf?sequence=1>

```
# Define the list of packages to load
pacotes <- c("tidyr", "tidyverse", "ggplot2", "readr", "caret", "dplyr",
             "randomForest", "rpart", "stats", "tinytex")

# Install and load missing packages
```

```

missing_packages <- pacotes[!pacotes %in% installed.packages()]
if (length(missing_packages) > 0) {
  install.packages(missing_packages, dependencies = TRUE)
}

# Load the required packages and show the result of installation.
sapply(pacotes, require, character = TRUE)

```

```
## Carregando pacotes exigidos: tidyr
```

```
## Warning: package 'tidyr' was built under R version 4.3.1
```

```
## Carregando pacotes exigidos: tidyverse
```

```
## Warning: package 'tidyverse' was built under R version 4.3.1
```

```
## Warning: package 'ggplot2' was built under R version 4.3.1
```

```
## Warning: package 'tibble' was built under R version 4.3.1
```

```
## Warning: package 'readr' was built under R version 4.3.1
```

```
## Warning: package 'purrr' was built under R version 4.3.1
```

```
## Warning: package 'dplyr' was built under R version 4.3.1
```

```
## Warning: package 'stringr' was built under R version 4.3.1
```

```
## Warning: package 'forcats' was built under R version 4.3.1
```

```
## Warning: package 'lubridate' was built under R version 4.3.1
```

```
## -- Attaching core tidyverse packages ----- tidyverse 2.0.0 --
```

```
## v dplyr      1.1.2      v purrr      1.0.1
```

```
## v forcats    1.0.0      v readr      2.1.4
```

```
## v ggplot2    3.4.2      v stringr    1.5.0
```

```
## v lubridate  1.9.2      v tibble     3.2.1
```

```
## -- Conflicts ----- tidyverse_conflicts() --
```

```
## x dplyr::filter() masks stats::filter()
```

```
## x dplyr::lag()    masks stats::lag()
```

```
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
## Carregando pacotes exigidos: caret
```

```
## Warning: package 'caret' was built under R version 4.3.1
```

```
## Carregando pacotes exigidos: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##   lift
##
## Carregando pacotes exigidos: randomForest
```

```
## Warning: package 'randomForest' was built under R version 4.3.1
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##   combine
##
## The following object is masked from 'package:ggplot2':
##
##   margin
##
## Carregando pacotes exigidos: rpart
## Carregando pacotes exigidos: tinytex
```

```
## Warning: package 'tinytex' was built under R version 4.3.1
```

```
##      tidyr      tidyverse      ggplot2      readr      caret      dplyr
##      TRUE        TRUE        TRUE        TRUE        TRUE        TRUE
## randomForest      rpart      stats      tinytex
##      TRUE        TRUE        TRUE        TRUE
```

```
#set working environment if it's not set to the correct..
setwd("C:/Users/user/Documents/R_Programming/Breast_cancer_data_set/")
getwd()
```

```
## [1] "C:/Users/user/Documents/R_Programming/Breast_cancer_data_set"
```

```
# Get data and explore it

#Read csv file, assign to a variable and view data.
data_h <- read.csv("breast-cancer.csv", sep = ",")

#Explore data analyzing what is inside our database
str(data_h)
```

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

```
head(data_h,5)
```

```
##      id diagnosis radius_mean texture_mean perimeter_mean area_mean
## 1    842302      M      17.99      10.38      122.80      1001.0
## 2    842517      M      20.57      17.77      132.90      1326.0
## 3  84300903      M      19.69      21.25      130.00      1203.0
## 4  84348301      M      11.42      20.38       77.58       386.1
## 5  84358402      M      20.29      14.34      135.10      1297.0
##      smoothness_mean compactness_mean concavity_mean concave.points_mean
## 1      0.11840      0.27760      0.3001      0.14710
## 2      0.08474      0.07864      0.0869      0.07017
## 3      0.10960      0.15990      0.1974      0.12790
## 4      0.14250      0.28390      0.2414      0.10520
## 5      0.10030      0.13280      0.1980      0.10430
##      symmetry_mean fractal_dimension_mean radius_se texture_se perimeter_se
## 1      0.2419      0.07871      1.0950      0.9053      8.589
## 2      0.1812      0.05667      0.5435      0.7339      3.398
## 3      0.2069      0.05999      0.7456      0.7869      4.585
## 4      0.2597      0.09744      0.4956      1.1560      3.445
## 5      0.1809      0.05883      0.7572      0.7813      5.438
##      area_se smoothness_se compactness_se concavity_se concave.points_se
## 1    153.40      0.006399      0.04904      0.05373      0.01587
## 2     74.08      0.005225      0.01308      0.01860      0.01340
```

```
## 3  94.03      0.006150      0.04006      0.03832      0.02058
## 4  27.23      0.009110      0.07458      0.05661      0.01867
## 5  94.44      0.011490      0.02461      0.05688      0.01885
##  symmetry_se fractal_dimension_se radius_worst texture_worst perimeter_worst
## 1    0.03003              0.006193      25.38      17.33      184.60
## 2    0.01389              0.003532      24.99      23.41      158.80
## 3    0.02250              0.004571      23.57      25.53      152.50
## 4    0.05963              0.009208      14.91      26.50      98.87
## 5    0.01756              0.005115      22.54      16.67      152.20
##  area_worst smoothness_worst compactness_worst concavity_worst
## 1    2019.0          0.1622          0.6656          0.7119
## 2    1956.0          0.1238          0.1866          0.2416
## 3    1709.0          0.1444          0.4245          0.4504
## 4     567.7          0.2098          0.8663          0.6869
## 5    1575.0          0.1374          0.2050          0.4000
##  concave.points_worst symmetry_worst fractal_dimension_worst
## 1              0.2654          0.4601              0.11890
## 2              0.1860          0.2750              0.08902
## 3              0.2430          0.3613              0.08758
## 4              0.2575          0.6638              0.17300
## 5              0.1625          0.2364              0.07678
```

```
dim(data_h)
```

```
## [1] 569 32
```

```
summary(data_h)
```

```
##      id      diagnosis      radius_mean      texture_mean
## Min.   :    8670 Length:569      Min.   : 6.981      Min.   : 9.71
## 1st Qu.:  869218 Class :character 1st Qu.:11.700      1st Qu.:16.17
## Median :  906024 Mode  :character  Median :13.370      Median :18.84
## Mean   : 30371831      Mean   :14.127      Mean   :19.29
## 3rd Qu.:  8813129      3rd Qu.:15.780      3rd Qu.:21.80
## Max.   :911320502      Max.   :28.110      Max.   :39.28
## perimeter_mean area_mean smoothness_mean compactness_mean
## Min.   : 43.79      Min.   : 143.5      Min.   :0.05263      Min.   :0.01938
## 1st Qu.: 75.17      1st Qu.: 420.3      1st Qu.:0.08637      1st Qu.:0.06492
## Median : 86.24      Median : 551.1      Median :0.09587      Median :0.09263
## Mean   : 91.97      Mean   : 654.9      Mean   :0.09636      Mean   :0.10434
## 3rd Qu.:104.10      3rd Qu.: 782.7      3rd Qu.:0.10530      3rd Qu.:0.13040
## Max.   :188.50      Max.   :2501.0      Max.   :0.16340      Max.   :0.34540
## concavity_mean concave.points_mean symmetry_mean fractal_dimension_mean
## Min.   :0.00000      Min.   :0.00000      Min.   :0.1060      Min.   :0.04996
## 1st Qu.:0.02956      1st Qu.:0.02031      1st Qu.:0.1619      1st Qu.:0.05770
## Median :0.06154      Median :0.03350      Median :0.1792      Median :0.06154
## Mean   :0.08880      Mean   :0.04892      Mean   :0.1812      Mean   :0.06280
## 3rd Qu.:0.13070      3rd Qu.:0.07400      3rd Qu.:0.1957      3rd Qu.:0.06612
## Max.   :0.42680      Max.   :0.20120      Max.   :0.3040      Max.   :0.09744
## radius_se texture_se perimeter_se area_se
## Min.   :0.1115      Min.   :0.3602      Min.   : 0.757      Min.   : 6.802
## 1st Qu.:0.2324      1st Qu.:0.8339      1st Qu.: 1.606      1st Qu.:17.850
## Median :0.3242      Median :1.1080      Median : 2.287      Median :24.530
```

```
## Mean :0.4052 Mean :1.2169 Mean : 2.866 Mean : 40.337
## 3rd Qu.:0.4789 3rd Qu.:1.4740 3rd Qu.: 3.357 3rd Qu.: 45.190
## Max. :2.8730 Max. :4.8850 Max. :21.980 Max. :542.200
## smoothness_se compactness_se concavity_se concave.points_se
## Min. :0.001713 Min. :0.002252 Min. :0.000000 Min. :0.000000
## 1st Qu.:0.005169 1st Qu.:0.013080 1st Qu.:0.01509 1st Qu.:0.007638
## Median :0.006380 Median :0.020450 Median :0.02589 Median :0.010930
## Mean :0.007041 Mean :0.025478 Mean :0.03189 Mean :0.011796
## 3rd Qu.:0.008146 3rd Qu.:0.032450 3rd Qu.:0.04205 3rd Qu.:0.014710
## Max. :0.031130 Max. :0.135400 Max. :0.39600 Max. :0.052790
## symmetry_se fractal_dimension_se radius_worst texture_worst
## Min. :0.007882 Min. :0.0008948 Min. : 7.93 Min. :12.02
## 1st Qu.:0.015160 1st Qu.:0.0022480 1st Qu.:13.01 1st Qu.:21.08
## Median :0.018730 Median :0.0031870 Median :14.97 Median :25.41
## Mean :0.020542 Mean :0.0037949 Mean :16.27 Mean :25.68
## 3rd Qu.:0.023480 3rd Qu.:0.0045580 3rd Qu.:18.79 3rd Qu.:29.72
## Max. :0.078950 Max. :0.0298400 Max. :36.04 Max. :49.54
## perimeter_worst area_worst smoothness_worst compactness_worst
## Min. : 50.41 Min. : 185.2 Min. :0.07117 Min. :0.02729
## 1st Qu.: 84.11 1st Qu.: 515.3 1st Qu.:0.11660 1st Qu.:0.14720
## Median : 97.66 Median : 686.5 Median :0.13130 Median :0.21190
## Mean :107.26 Mean : 880.6 Mean :0.13237 Mean :0.25427
## 3rd Qu.:125.40 3rd Qu.:1084.0 3rd Qu.:0.14600 3rd Qu.:0.33910
## Max. :251.20 Max. :4254.0 Max. :0.22260 Max. :1.05800
## concavity_worst concave.points_worst symmetry_worst fractal_dimension_worst
## Min. :0.0000 Min. :0.000000 Min. :0.1565 Min. :0.05504
## 1st Qu.:0.1145 1st Qu.:0.06493 1st Qu.:0.2504 1st Qu.:0.07146
## Median :0.2267 Median :0.09993 Median :0.2822 Median :0.08004
## Mean :0.2722 Mean :0.11461 Mean :0.2901 Mean :0.08395
## 3rd Qu.:0.3829 3rd Qu.:0.16140 3rd Qu.:0.3179 3rd Qu.:0.09208
## Max. :1.2520 Max. :0.29100 Max. :0.6638 Max. :0.20750
```

```
glimpse(data_h)
```

```
## Rows: 569
## Columns: 32
## $ id <int> 842302, 842517, 84300903, 84348301, 84358402, ~
## $ diagnosis <chr> "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~
```

```
#Check if we have any null values in the data frame.
data_h[is.na(data_h) == TRUE]
```

```
## character(0)
```

```
#Calculate the number of missing values in each column, in case we have
colSums(is.na(data_h))
```

```
##          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
```

```
#At this time we need to check if both classification categories
# have a significant representation in our dataset.
# Count the frequency of each class in the 'Category' column
class_counts <- table(data_h$diagnosis)
# Calculate the percentage of each class
class_percentages <- prop.table(class_counts) * 100
```

```

# Create a data frame with counts and percentages
class_summary <- data.frame(Class = names(class_counts),
                             Count = as.numeric(class_counts),
                             Percentage = class_percentages)

# Print the summary table
print(class_summary)

```

```

##   Class Count Percentage.Var1 Percentage.Freq
## 1    B   357             B      62.74165
## 2    M   212             M      37.25835

```

```

# Prepare data to use and split into test and train data frame.
# Replace data from categorical value to numerical and change the format.
data_h2 <- data_h
data_h2$diagnosis <- ifelse(data_h2$diagnosis == "M",1,
                            ifelse(data_h2$diagnosis == "B",2,
                                    data_h2$diagnosis))
data_h2$diagnosis <- as.numeric(data_h2$diagnosis)

# Split Data into Training and Testing in R based on 70% training
# and 30% for test.
sample_size = floor(0.7*nrow(data_h2))

# Set a random dataset.
set.seed(777)

# randomly split data in test and train using random split created
# in the last step.
train = sample(seq_len(nrow(data_h2)),size = sample_size)
# split data into train dataset
data_h_train = data_h2[train ,]
dim(data_h_train)

```

```
## [1] 398 32
```

```

# split data into test dataset
data_h_test = data_h2[-train,]
dim(data_h_test)

```

```
## [1] 171 32
```

```

# Create variables to prepare, apply linear regression model
# and assign it to a variable.
# Create Y variable
nomes <- "diagnosis"
y_variable <- nomes

# Create X variable extracting column names from dataset using a variable.
nomes <- data_h[,3:32]
x_variable <- names(nomes)

```



```
#Create complete formula with all column names.
formula_str <- paste(y_variable,"~",paste(x_variable, collapse = " + "))
```

```
# Print the formula
cat("Regression Formula:", formula_str, "\n")
```

```
## Regression Formula: diagnosis ~ radius_mean + texture_mean + perimeter_mean + area_mean + smoothness
```

```
#Create a prediction model using variable created to assign column names and show a summary.
model <- lm(formula_str,data= data_h_train)
model %>% summary()
```

```
##
## Call:
## lm(formula = formula_str, data = data_h_train)
##
## Residuals:
```

	Min	1Q	Median	3Q	Max
	-0.79611	-0.12315	0.02975	0.16023	0.66050

```
##
## Coefficients:
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	4.487e+00	5.449e-01	8.235	3.18e-15 ***
radius_mean	2.907e-02	2.185e-01	0.133	0.8942
texture_mean	-8.854e-03	1.028e-02	-0.861	0.3897
perimeter_mean	-5.827e-03	3.229e-02	-0.180	0.8569
area_mean	1.966e-04	6.460e-04	0.304	0.7611
smoothness_mean	-1.234e+00	2.449e+00	-0.504	0.6146
compactness_mean	5.376e+00	1.727e+00	3.113	0.0020 **
concavity_mean	-1.604e+00	1.323e+00	-1.212	0.2261
concave.points_mean	-3.529e+00	2.495e+00	-1.414	0.1581
symmetry_mean	-1.023e-01	9.441e-01	-0.108	0.9137
fractal_dimension_mean	3.940e-01	7.312e+00	0.054	0.9571
radius_se	-4.579e-01	4.003e-01	-1.144	0.2533
texture_se	1.808e-02	4.530e-02	0.399	0.6901
perimeter_se	1.074e-02	5.322e-02	0.202	0.8402
area_se	7.931e-04	1.710e-03	0.464	0.6431
smoothness_se	-1.238e+01	8.140e+00	-1.520	0.1293
compactness_se	2.231e+00	3.376e+00	0.661	0.5091
concavity_se	2.474e+00	2.467e+00	1.003	0.3166
concave.points_se	-1.209e+01	7.188e+00	-1.682	0.0934 .
symmetry_se	-1.761e+00	3.216e+00	-0.547	0.5844
fractal_dimension_se	-1.172e+01	1.711e+01	-0.685	0.4935
radius_worst	-1.706e-01	7.474e-02	-2.282	0.0231 *
texture_worst	-6.200e-03	8.756e-03	-0.708	0.4794
perimeter_worst	3.900e-03	8.288e-03	0.470	0.6383
area_worst	8.862e-04	4.012e-04	2.209	0.0278 *
smoothness_worst	-9.053e-01	1.708e+00	-0.530	0.5965
compactness_worst	-8.258e-01	5.533e-01	-1.492	0.1365
concavity_worst	-3.420e-02	3.798e-01	-0.090	0.9283
concave.points_worst	-1.306e-01	1.163e+00	-0.112	0.9107
symmetry_worst	-7.517e-01	6.221e-01	-1.208	0.2277
fractal_dimension_worst	-2.941e+00	3.407e+00	-0.863	0.3886

```

## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2405 on 367 degrees of freedom
## Multiple R-squared:  0.7741, Adjusted R-squared:  0.7556
## F-statistic: 41.92 on 30 and 367 DF,  p-value: < 2.2e-16

#create a summary variable to create a descending order in next chapter.
model_summary <- summary(model)
# Sort the summary by the "Pr(>|t|)" column in ascending order
sorted_summary <- model_summary$coefficients[order
                                                    (model_summary$coefficients[, "Pr(>|t|)"]), ]

#print summary with ascending order
print(sorted_summary)

##              Estimate   Std. Error   t value   Pr(>|t|)
## (Intercept)      4.487144e+00  5.448548e-01  8.23548580 3.180260e-15
## compactness_mean  5.376430e+00  1.727171e+00  3.11285356 1.998239e-03
## radius_worst     -1.705603e-01  7.474406e-02 -2.28192372 2.306571e-02
## area_worst        8.861691e-04  4.012065e-04  2.20876047 2.780867e-02
## concave.points_se -1.208976e+01  7.187790e+00 -1.68198550 9.342227e-02
## smoothness_se     -1.237728e+01  8.140480e+00 -1.52046121 1.292562e-01
## compactness_worst -8.257530e-01  5.533477e-01 -1.49228591 1.364833e-01
## concave.points_mean -3.529248e+00  2.495169e+00 -1.41443236 1.580828e-01
## concavity_mean    -1.604164e+00  1.323138e+00 -1.21239365 2.261416e-01
## symmetry_worst     -7.517420e-01  6.220686e-01 -1.20845522 2.276503e-01
## radius_se         -4.579219e-01  4.002577e-01 -1.14406761 2.533412e-01
## concavity_se       2.473911e+00  2.466671e+00  1.00293505 3.165530e-01
## fractal_dimension_worst -2.940685e+00  3.406932e+00 -0.86314739 3.886201e-01
## texture_mean      -8.854253e-03  1.028214e-02 -0.86112975 3.897288e-01
## texture_worst     -6.199702e-03  8.756449e-03 -0.70801551 4.793850e-01
## fractal_dimension_se -1.172355e+01  1.710518e+01 -0.68538039 4.935366e-01
## compactness_se     2.231074e+00  3.376122e+00  0.66083928 5.091300e-01
## symmetry_se       -1.760612e+00  3.216317e+00 -0.54740007 5.844366e-01
## smoothness_worst  -9.052717e-01  1.708221e+00 -0.52995009 5.964670e-01
## smoothness_mean   -1.234362e+00  2.449297e+00 -0.50396574 6.145878e-01
## perimeter_worst    3.899458e-03  8.287981e-03  0.47049546 6.382806e-01
## area_se           7.930998e-04  1.710219e-03  0.46374156 6.431078e-01
## texture_se        1.807930e-02  4.530108e-02  0.39909204 6.900575e-01
## area_mean         1.965733e-04  6.459928e-04  0.30429647 7.610745e-01
## perimeter_se      1.073742e-02  5.321590e-02  0.20177085 8.402077e-01
## perimeter_mean    -5.826833e-03  3.229432e-02 -0.18042902 8.569154e-01
## radius_mean       2.907039e-02  2.185154e-01  0.13303589 8.942379e-01
## concave.points_worst -1.306217e-01  1.163407e+00 -0.11227509 9.106667e-01
## symmetry_mean     -1.023261e-01  9.440888e-01 -0.10838610 9.137487e-01
## concavity_worst   -3.420335e-02  3.797860e-01 -0.09005953 9.282890e-01
## fractal_dimension_mean 3.939741e-01  7.311516e+00  0.05388405 9.570569e-01

# Create a prediction using training data and evaluate the model created
#based on training dataset.

#add our prediction to a data frame.
data_h_train$train_predicted_value <- stats::predict(model,data_h_train)

```

```
# Evaluate our predicted model based on different metrics as MAE, MSE and RMSE.
```

```
# Calculate the Mean Absolute Error (MAE)
```

```
mae <- mean(abs(data_h_train$train_predicted_value - data_h_train$diagnosis))
```

```
# Calculate the Mean Squared Error (MSE)
```

```
mse <- mean((data_h_train$train_predicted_value - data_h_train$diagnosis)^2)
```

```
# Calculate the Root Mean Squared Error (RMSE)
```

```
rmse <- sqrt(mse)
```

```
# MSE is calculated by taking the average of the squared differences between  
#the predicted values and the actual value
```

```
#A lower MSE indicates a better fit of the model to the data.
```

```
cat("Mean Squared Error (MSE):", mse, "\n")
```

```
## Mean Squared Error (MSE): 0.05332976
```

```
#MAE is calculated by taking the average of the absolute differences  
#between the predicted values and the actual values.
```

```
#A lower MAE indicates a better fit of the model to the data.
```

```
cat("Mean Absolute Error (MAE):", mae, "\n")
```

```
## Mean Absolute Error (MAE): 0.1782873
```

```
# RMSE is calculated by taking the square root of the MSE.
```

```
cat("Root Mean Squared Error (RMSE):", rmse, "\n")
```

```
## Root Mean Squared Error (RMSE): 0.2309324
```

```
#Create an evaluation based on confusion matrix, sensitivity and specificity.  
#Convert data to numeric, than we can use it.
```

```
data_h_train$train_predicted_value <- ifelse(data_h_train$  
                                             train_predicted_value>1.4, 2, 1)
```

```
#Create a confusion matrix
```

```
confusion_matrix_train <- caret::confusionMatrix(factor(data_h_train$  
                                                         train_predicted_value), factor(data_h_train$d
```

```
print(confusion_matrix_train)
```

```
## Confusion Matrix and Statistics
```

```
##
```

```
##           Reference
```

```
## Prediction  1    2
```

```
##           1 128   1
```

```
##           2  24 245
```

```
##
```

```
##           Accuracy : 0.9372
```

```
##           95% CI : (0.9087, 0.9589)
```

```
## No Information Rate : 0.6181
```

```
## P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##           Kappa : 0.863
##
## Mcnemar's Test P-Value : 1.083e-05
##
##           Sensitivity : 0.8421
##           Specificity : 0.9959
##           Pos Pred Value : 0.9922
##           Neg Pred Value : 0.9108
##           Prevalence : 0.3819
##           Detection Rate : 0.3216
##           Detection Prevalence : 0.3241
##           Balanced Accuracy : 0.9190
##
##           'Positive' Class : 1
##
```

```
# Retrieve sensitivity and specificity from the confusion matrix object
sensitivity <- confusion_matrix_train$byClass["Sensitivity"]
specificity <- confusion_matrix_train$byClass["Specificity"]

# Print sensitivity and specificity
#Sensitivity: is also called the true positive rate and is exactly
#equal to recall.
cat("Sensitivity:", sensitivity, "\n")
```

```
## Sensitivity: 0.8421053
```

```
#Specificity: is also called the true negative rate and is equal to TN/(TN+FP)
cat("Specificity:", specificity, "\n")
```

```
## Specificity: 0.995935
```

```
# Now we need to calculate prediction based on test data and evaluate it.

#create a prediction using test set, based in model that was based
#on training model
data_h_test$test_predicted_value <- stats::predict(model,data_h_test)
```

```
# Calculate the Mean Absolute Error (MAE)
mae <- mean(abs(data_h_test$test_predicted_value - data_h_test$diagnosis))

# Calculate the Mean Squared Error (MSE)
mse <- mean((data_h_test$test_predicted_value - data_h_test$diagnosis)^2)

# Calculate the Root Mean Squared Error (RMSE)
rmse <- sqrt(mse)

cat("Mean Squared Error (MSE):", mse, "\n")
```

```
## Mean Squared Error (MSE): 0.05978929
```

```
cat("Mean Absolute Error (MAE):", mae, "\n")
```

```
## Mean Absolute Error (MAE): 0.2030555
```

```
cat("Root Mean Squared Error (RMSE):", rmse, "\n")
```

```
## Root Mean Squared Error (RMSE): 0.2445185
```

```
#Convert data to numeric, than we can use it.
```

```
data_h_test$test_predicted_value <- ifelse(data_h_test$  
                                           test_predicted_value>1.4, 2, 1)
```

```
#Create a confusion matrix
```

```
confusion_matrix_test <- caret::confusionMatrix(factor(data_h_test$  
                                                         test_predicted_value),  
                                                  factor(data_h_test$diagnosis))
```

```
print(confusion_matrix_test)
```

```
## Confusion Matrix and Statistics
```

```
##
```

```
##           Reference
```

```
## Prediction  1    2
```

```
##           1  52   0
```

```
##           2   8 111
```

```
##
```

```
##           Accuracy : 0.9532
```

```
##           95% CI : (0.9099, 0.9796)
```

```
## No Information Rate : 0.6491
```

```
## P-Value [Acc > NIR] : < 2e-16
```

```
##
```

```
##           Kappa : 0.8941
```

```
##
```

```
## Mcnemar's Test P-Value : 0.01333
```

```
##
```

```
##           Sensitivity : 0.8667
```

```
##           Specificity : 1.0000
```

```
## Pos Pred Value : 1.0000
```

```
## Neg Pred Value : 0.9328
```

```
## Prevalence : 0.3509
```

```
## Detection Rate : 0.3041
```

```
## Detection Prevalence : 0.3041
```

```
## Balanced Accuracy : 0.9333
```

```
##
```

```
## 'Positive' Class : 1
```

```
##
```

```
# Retrieve sensitivity and specificity from the confusion matrix object
```

```
sensitivity <- confusion_matrix_test$byClass["Sensitivity"]
```

```
specificity <- confusion_matrix_test$byClass["Specificity"]
```

```
# Print sensitivity and specificity
```

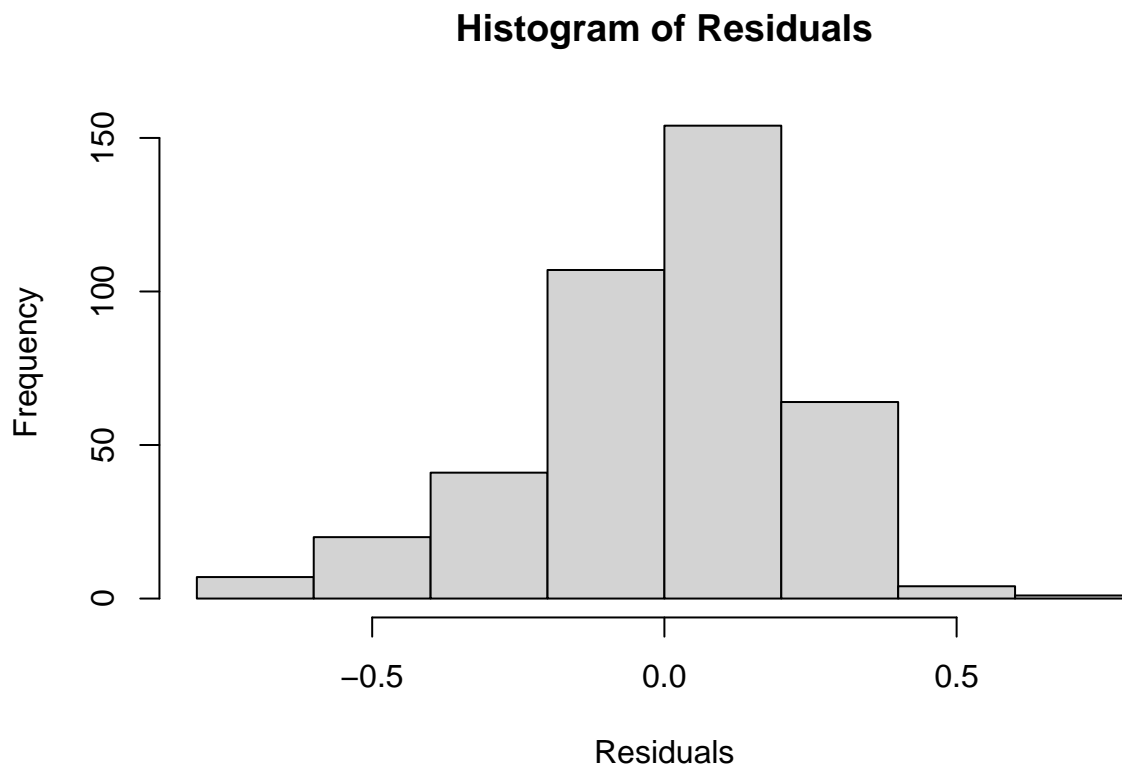
```
#Sensitivity: is also called the true positive rate and is exactly
#equal to recall.
cat("Sensitivity:", sensitivity, "\n")
```

```
## Sensitivity: 0.8666667
```

```
#Specificity: is also called the true negative rate and is equal to TN/(TN+FP)
cat("Specificity:", specificity, "\n")
```

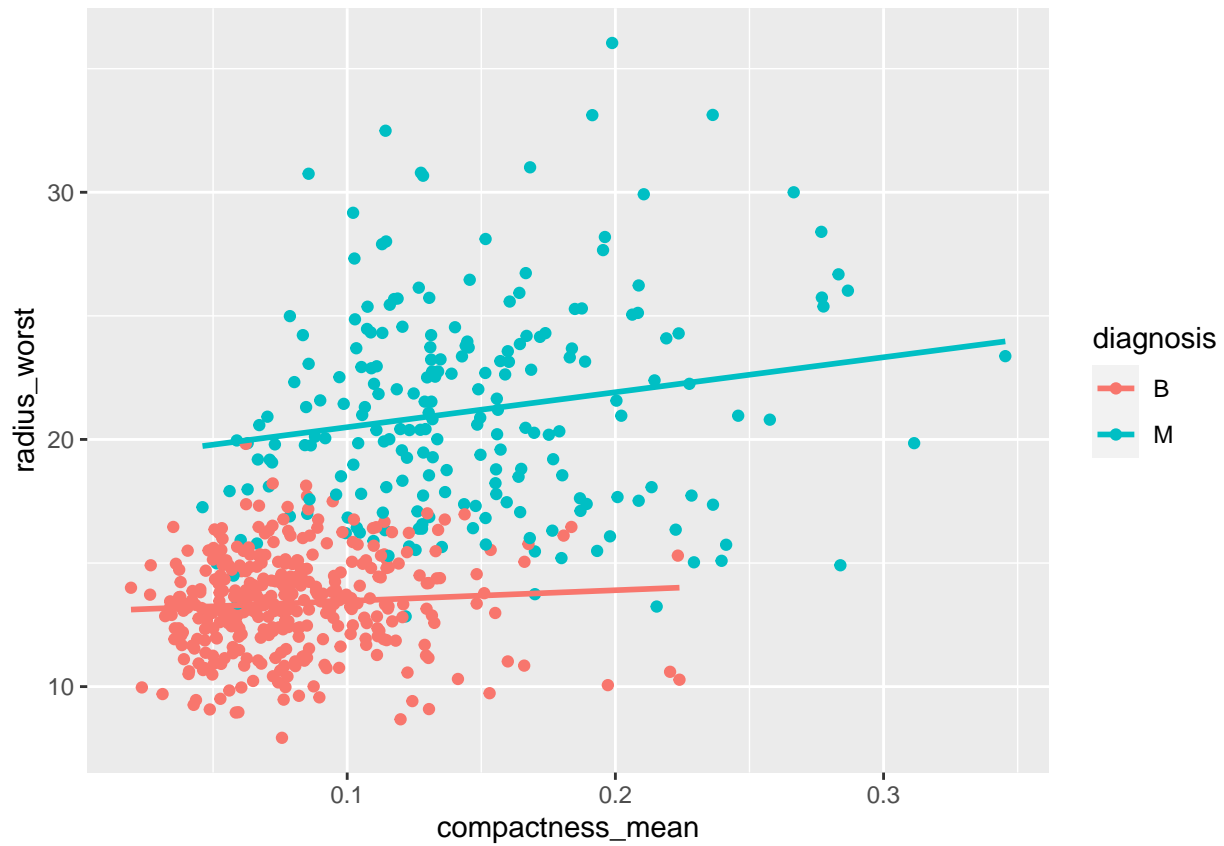
```
## Specificity: 1
```

```
# Calculate the residuals
residuals <- residuals(model)
# Create a histogram of residuals
hist(residuals, main = "Histogram of Residuals", xlab = "Residuals")
```



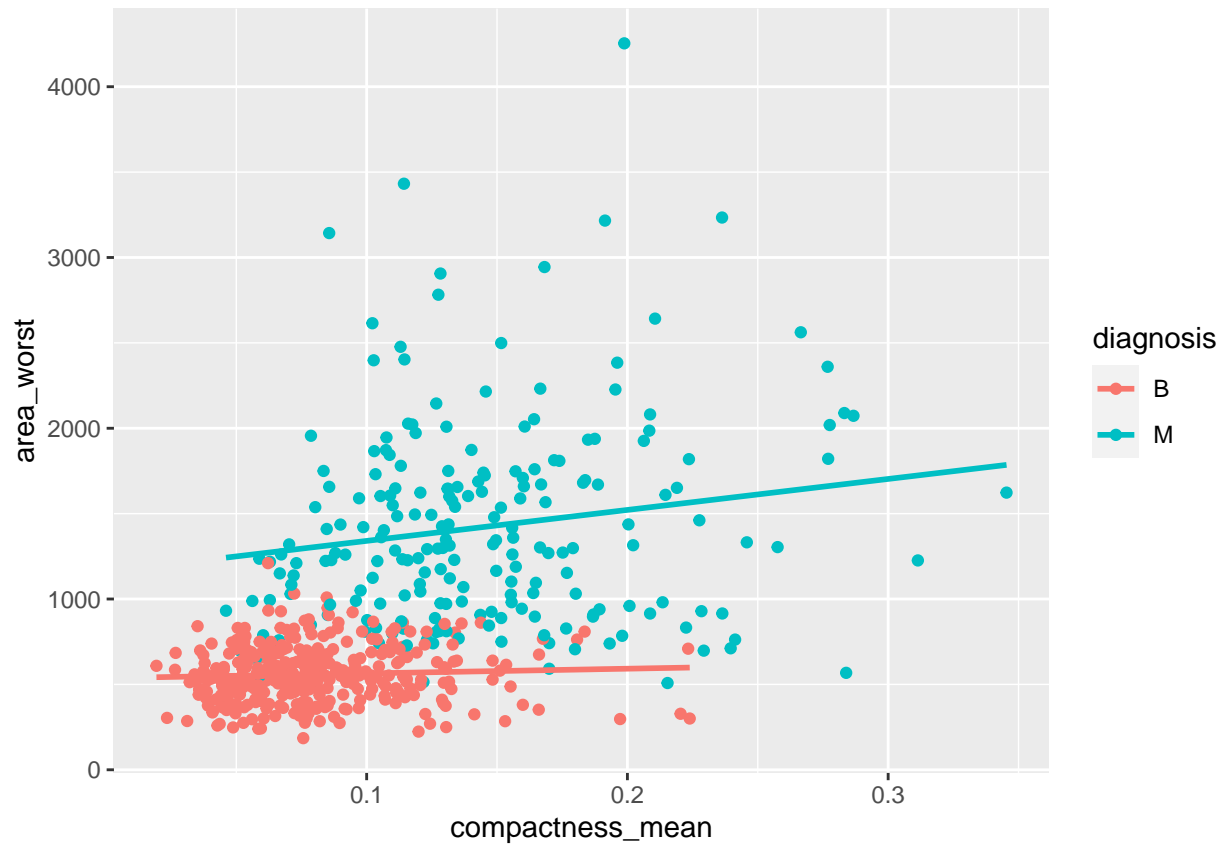
```
# Graph analysis to show two main variables that split result of
#regression analysis.
#Compactness mean and radius worst
ggplot(data_h, aes(x=compactness_mean,
                    y=radius_worst,
                    col=diagnosis))+
  geom_point()+
  geom_smooth(method='lm', se = FALSE)
```

```
## 'geom_smooth()' using formula = 'y ~ x'
```



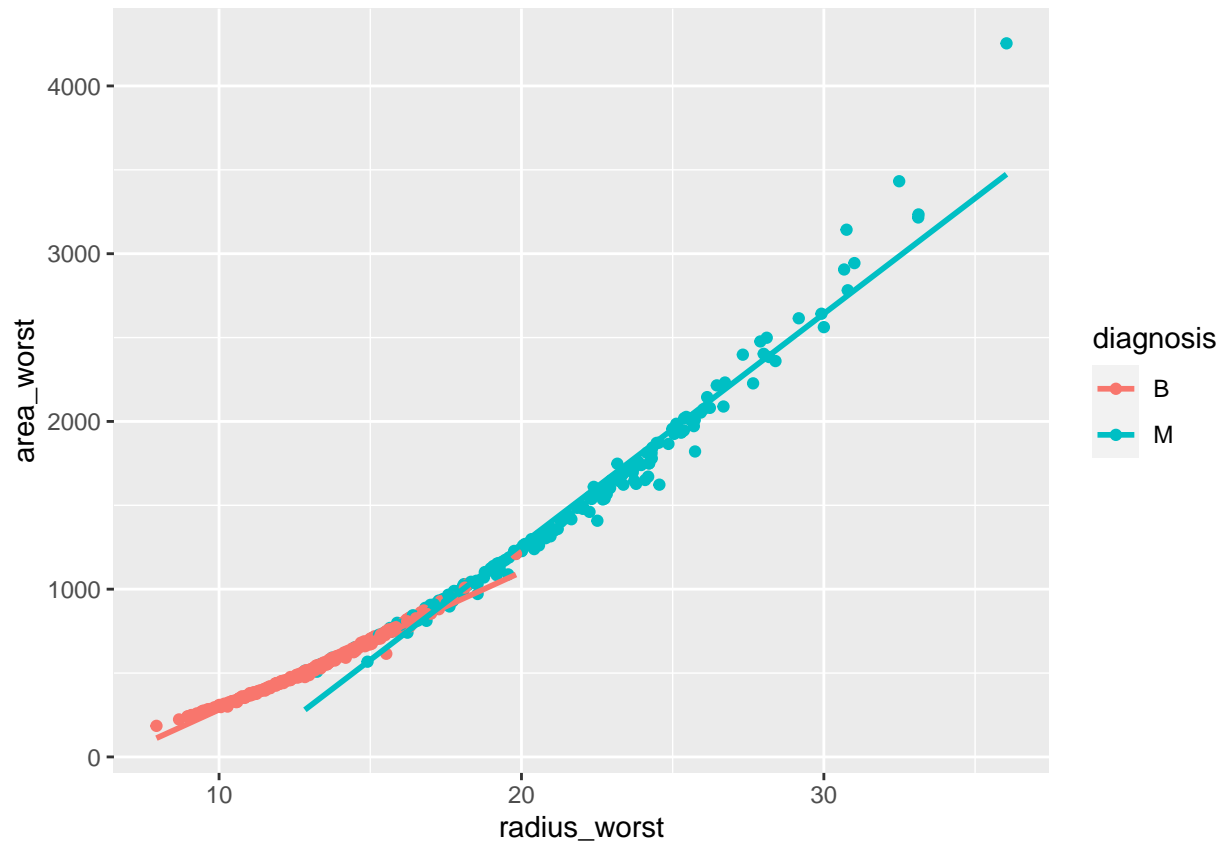
```
# Graph analysis to show two main variables that split result of  
# regression analysis.  
#Compactness mean and area worst  
ggplot(data_h, aes(x=compactness_mean,  
                    y=area_worst,  
                    col=diagnosis))+  
  geom_point()+  
  geom_smooth(method='lm', se = FALSE)
```

```
## 'geom_smooth()' using formula = 'y ~ x'
```



```
# Graph analysis to show two main variables that split result of
#regression analysis.
#Compactness mean and area worst
ggplot(data_h, aes(x=radius_worst,
                    y=area_worst,
                    col=diagnosis))+
  geom_point()+
  geom_smooth(method='lm', se = FALSE)
```

```
## 'geom_smooth()' using formula = 'y ~ x'
```

*#This final graph shows the relationship between three variables
#compactness_mean, radius_worst and area_worst.*

*#First I added a scatterplot comparing compactness_mean and radius_worst
#based in diagnosis result(left analysis in graph)*

```
multiple_graph_regression <- ggplot(data_h, aes(x = compactness_mean,
                                              y = radius_worst,
                                              color = diagnosis)) +
  geom_point(size = 3) +
```

*#Then I added a comparison between compactness_mean and area_worst based
#in diagnosis result in the same graph(right analysis in graph).*

```
  geom_point(aes(x = area_worst), size = 3) +
  labs(title = "Comparison between multiple variables:",
       x = "Compactness Mean",
       y = "Diagnosis") +
  scale_color_manual(values = c("M" = "darkmagenta", "B" = "cornflowerblue"))
```

```
# Show the combined plot
print(multiple_graph_regression)
```

Comparison between multiple variables:

