**Student: Seif Kungulio**

**Date: 03/09/2025**

**Subject: Project 8**

**Class: DSCI 502**

**Section: 01W**

**Instructor: Sean Yang**

**File Name: Project8\_Kungulio\_Seif.docx**

1. Load the dataset in breast\_cancer\_data.csv into R. Call the loaded data breast\_cancer\_data. Make sure that you have the directory set to the correct location for the data.

>

> ## 1. Load the dataset in breast\_cancer\_data.csv into R. Call the loaded data

> ## breast\_cancer\_data. Make sure that you have the directory set to the

> ## correct location for the data.

>

> # Load necessary libraries

> library(ggplot2) # For visualization

> library(dplyr) # For data manipulation

> library(pscl) # For McFadden's R-squared calculation

>

> # Set the working directory to the correct location for the dataset.

> setwd("C:/PROJECTS/Maryville/DSCI-502/Week8")

>

> # Load the data from breast\_cancer\_data.csv

> breast\_cancer\_data <- read.csv("breast\_cancer\_data.csv")

>

> # Display the dimensions (rows and columns) of the dataframe

> dim(breast\_cancer\_data) # Shows the number of rows and columns in the dataset.

[1] 569 32

>

A screenshot of a computer screen

AI-generated content may be incorrect.

>

> # Display column names

> colnames(breast\_cancer\_data)

[1] "id" "diagnosis" "radius\_mean"

[4] "texture\_mean" "perimeter\_mean" "area\_mean"

[7] "smoothness\_mean" "compactness\_mean" "concavity\_mean"

[10] "concave.points\_mean" "symmetry\_mean" "fractal\_dimension\_mean"

[13] "radius\_se" "texture\_se" "perimeter\_se"

[16] "area\_se" "smoothness\_se" "compactness\_se"

[19] "concavity\_se" "concave.points\_se" "symmetry\_se"

[22] "fractal\_dimension\_se" "radius\_worst" "texture\_worst"

[25] "perimeter\_worst" "area\_worst" "smoothness\_worst"

[28] "compactness\_worst" "concavity\_worst" "concave.points\_worst"

[31] "symmetry\_worst" "fractal\_dimension\_worst"

>

A computer screen shot of white text

AI-generated content may be incorrect.

>

> # Displays the structure of the breast\_cancer\_data object.

> str(breast\_cancer\_data)

'data.frame': 569 obs. of 32 variables:

$ id : int 842302 842517 84300903 84348301 84358402 843786 844359 84458202 844981 84501001 ...

$ 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 ...

>

A screenshot of a computer

AI-generated content may be incorrect.

1. Define a user defined function BoxplotPredictorOnTarget with two arguments, the target and one predictor to plot the box plot of predictor based on different category of the target. Then use this user defined function to generate the box plot:

>

> ## 2. Define a user defined function BoxplotPredictorOnTarget with two

> ## arguments, the target and one predictor to plot the box plot of predictor

> ## based on different category of the target. Then use this user defined

> ## function to generate the box plot:

>

> # Convert Diagnosis to a factor

> breast\_cancer\_data$diagnosis <- factor(breast\_cancer\_data$diagnosis,

+ levels = c("B", "M"),

+ labels = c("Benign", "Malignant"))

>

> # Define the user-defined function for boxplot

> BoxplotPredictorOnTarget <- function(target, predictor) {

+ ggplot(breast\_cancer\_data, aes(x = .data[[target]],

+ y = .data[[predictor]],

+ fill = .data[[target]])) +

+ geom\_boxplot() + theme\_test() +

+ labs(title = paste("Boxplot of", predictor, "by", target),

+ x = target, y = predictor)

+ }

>

A computer screen shot of a program code

AI-generated content may be incorrect.

* 1. area\_mean against Diagnosis

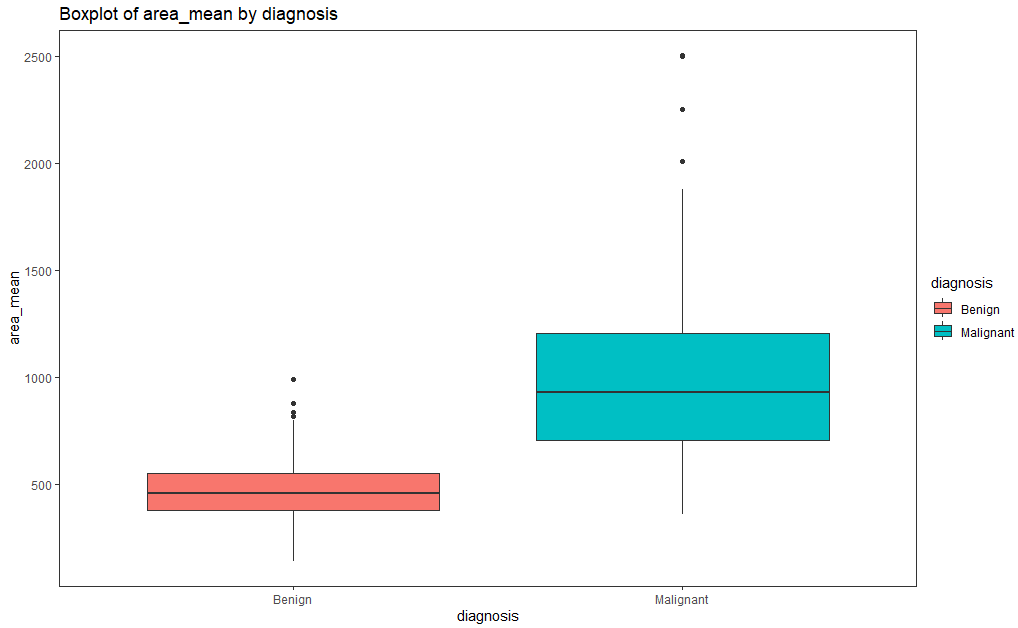
>

> ##### a) area\_mean against diagnosis

> # Boxplot of area\_mean against diagnosis

> BoxplotPredictorOnTarget("diagnosis", "area\_mean")

>



* 1. area\_se against Diagnosis

>

> ##### b) area\_se against diagnosis

> # Boxplot of area\_se against diagnosis

> BoxplotPredictorOnTarget("diagnosis", "area\_se")

>

A screenshot of a graph

AI-generated content may be incorrect.

* 1. texture\_mean against Diagnosis

>

> ##### c) texture\_mean against diagnosis

> # Boxplot of texture\_mean against diagnosis

> BoxplotPredictorOnTarget("diagnosis", "texture\_mean")

>

A diagram of a graph

AI-generated content may be incorrect.

1. Build the following logistic models to forecast the Diagnosis and recommend the best model based on McFadden/pseudo R squared to the management.
2. forecast Diagnosis using area\_mean

>

> ##### a) forecast Diagnosis using area\_mean

> model1 <- glm(diagnosis ~ area\_mean,

+ data = breast\_cancer\_data,

+ family = binomial

+ )

>

A black screen with white text

AI-generated content may be incorrect.

1. forecast the Diagnosis using area\_mean and area\_se

>

> ##### b) forecast the Diagnosis using area\_mean and area\_se

> model2 <- glm(diagnosis ~ area\_mean + area\_se,

+ data = breast\_cancer\_data,

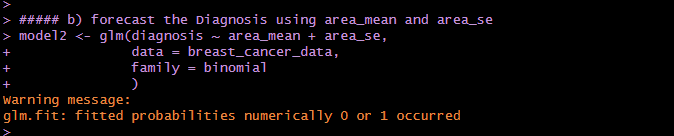
+ family = binomial

+ )

Warning message:

glm.fit: fitted probabilities numerically 0 or 1 occurred

>



1. forecast the Diagnosis using area\_mean, area\_se and texture\_mean

>

> ##### c) forecast the Diagnosis using area\_mean, area\_se and texture\_mean

> model3 <- glm(diagnosis ~ area\_mean + area\_se + texture\_mean,

+ data = breast\_cancer\_data,

+ family = binomial

+ )

Warning message:

glm.fit: fitted probabilities numerically 0 or 1 occurred

>

A black screen with white text

AI-generated content may be incorrect.

1. forecast the Diagnosis using area\_mean, area\_se, texture\_mean and concavity\_worst

>

> ##### d) forecast the Diagnosis using area\_mean, area\_se, texture\_mean and

> ##### concavity\_worst

> model4 <- glm(diagnosis ~ area\_mean + area\_se + texture\_mean + concavity\_worst,

+ data = breast\_cancer\_data,

+ family = binomial

+ )

Warning message:

glm.fit: fitted probabilities numerically 0 or 1 occurred

>

A black screen with white text

AI-generated content may be incorrect.

1. forecast the Diagnosis using area\_mean, area\_se, texture\_mean, concavity\_worst and concavity\_mean

>

> ##### e) forecast the Diagnosis using area\_mean, area\_se, texture\_mean,

> ##### concavity\_worst and concavity\_mean

> model5 <- glm(diagnosis ~ area\_mean + area\_se + texture\_mean +

+ concavity\_worst + concavity\_mean,

+ data = breast\_cancer\_data,

+ family = binomial

+ )

Warning message:

glm.fit: fitted probabilities numerically 0 or 1 occurred

>

A screen shot of a computer code

AI-generated content may be incorrect.

>

> # Function to compute McFadden's R-squared for each model

> r\_squared <- function(model) {

+ 1 - (logLik(model)[1] /

+ logLik(glm(diagnosis ~ 1, # Null model with only intercept

+ data = breast\_cancer\_data,

+ family = binomial))[1])

+ }

>

> # Store all models in a list

> models <- list(model1, model2, model3, model4, model5)

> names(models) <- c("Model 1", "Model 2", "Model 3", "Model 4", "Model 5")

>

> # Compute McFadden's R-squared for each model

> r\_squared\_values <- sapply(models, r\_squared)

>

> # Display the R-squared values, rounded to 4 decimal places

> round(r\_squared\_values, 4)

Model 1 Model 2 Model 3 Model 4 Model 5

0.5666 0.6142 0.6521 0.7833 0.7856

>

> # Recommend the best model based on the highest R-squared value

> best\_model <- names(models)[which.max(r\_squared\_values)]

> paste("The best model based on McFadden's R-squared is", best\_model)

[1] "The best model based on McFadden's R-squared is Model 5"

>

A screenshot of a computer program

AI-generated content may be incorrect.