American International University - Bangladesh

Introduction to Data Science [A]

Final-Term Project Report

Submitted to -

Tohedul Islam Sir

Dataset – Breast Cancer

Submitted by –

FARUQUE, SAKIF AL

20-42308-1

BSc. in CSE

Dataset Description –

Breast cancer is the most common cancer among women in the world. It accounts for 25% of all cancer cases and affected more than 2.1 million people in 2015 alone. It begins when breast cells begin to grow out of control. These cells usually form tumors that can be seen on X-rays or felt as lumps in the breast.

The key challenge against its detection is how to classify tumors into malignant (cancerous) or benign(non-cancerous). Here, 'diagnosis' is the target feature/attribute whose value is to be defined as malignant or benign and the rest of the features will be used for classifying that target attribute.

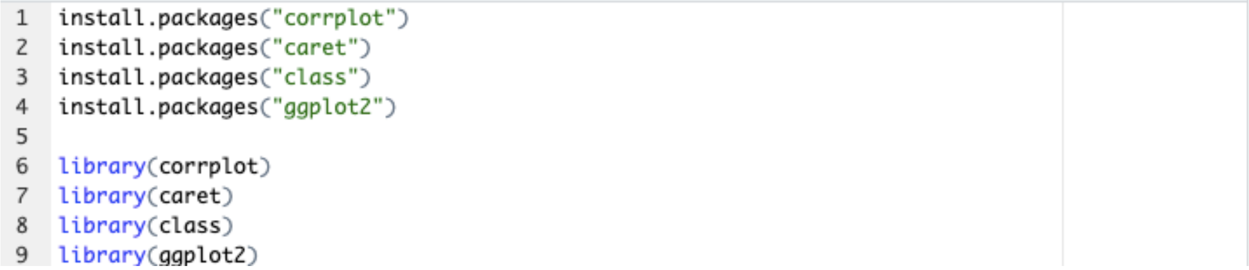
For classification, the KNN algorithm is used. After building the model, accuracy is checked with the dividing and 10-fold cross-validation approaches. Finally, the confusion matrix will be built and precision & recall values will be calculated.

List of feature description is below,

Attribute	Description	Expected Value
Id	Unique ID	Serial values
Diagnosis	Target	M - Malignant B - Benign
radius_mean	Radius of Lobes	Decimal
texture_mean	Mean of Surface Texture	Decimal
perimeter_mean	Outer Perimeter of Lobes	Decimal
area_mean	Mean Area of Lobes	Decimal
smoothness_mean	Mean of Smoothness Levels	Decimal
compactness_mean	Mean of Compactness	Decimal
concavity_mean	Mean of Concavity	Decimal
concave points_mean	Mean of Cocave Points	Decimal
symmetry_mean	Mean of Symmetry	Decimal
fractal_dimension_mean	Mean of Fractal Dimension	Decimal
radius_se	SE of Radius	Decimal
texture_se	SE of Texture	Decimal
perimeter_se	Perimeter of SE	Decimal
area_se	Are of SE	Decimal
smoothness_se	SE of Smoothness	Decimal

compactness_se	SE of compactness	Decimal
concavity_se	SEE of concavity	Decimal
concave points_se	SE of concave points	Decimal
symmetry_se	SE of symmetry	Decimal
fractal_dimension_se	SE of Fractal Dimension	Decimal
radius_worst	Worst Radius	Decimal
texture_worst	Worst Texture	Decimal
perimeter_worst	Worst Permimeter	Decimal
area_worst	Worst Area	Decimal
smoothness_worst	Worst Smoothness	Decimal
compactness_worst	Worse Compactness	Decimal
concavity_worst	Worst Concavity	Decimal
concave points_worst	Worst Concave Points	Decimal
symmetry_worst	Worst Symmetry	Decimal
fractal_dimension_worst	Worst Fractal Dimension	Decimal

The libraries which are used for this project,



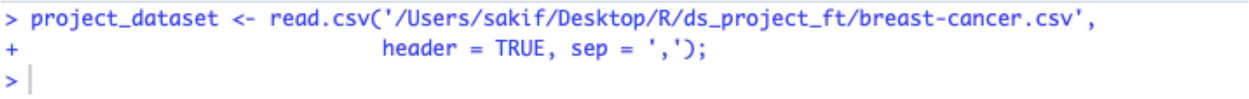
```r
install.packages("corrplot")
install.packages("caret")
install.packages("class")
install.packages("ggplot2")

library(corrplot)
library(caret)
library(class)
library(ggplot2)
```

Figure 1: Libraries & Modules

Reading Dataset –

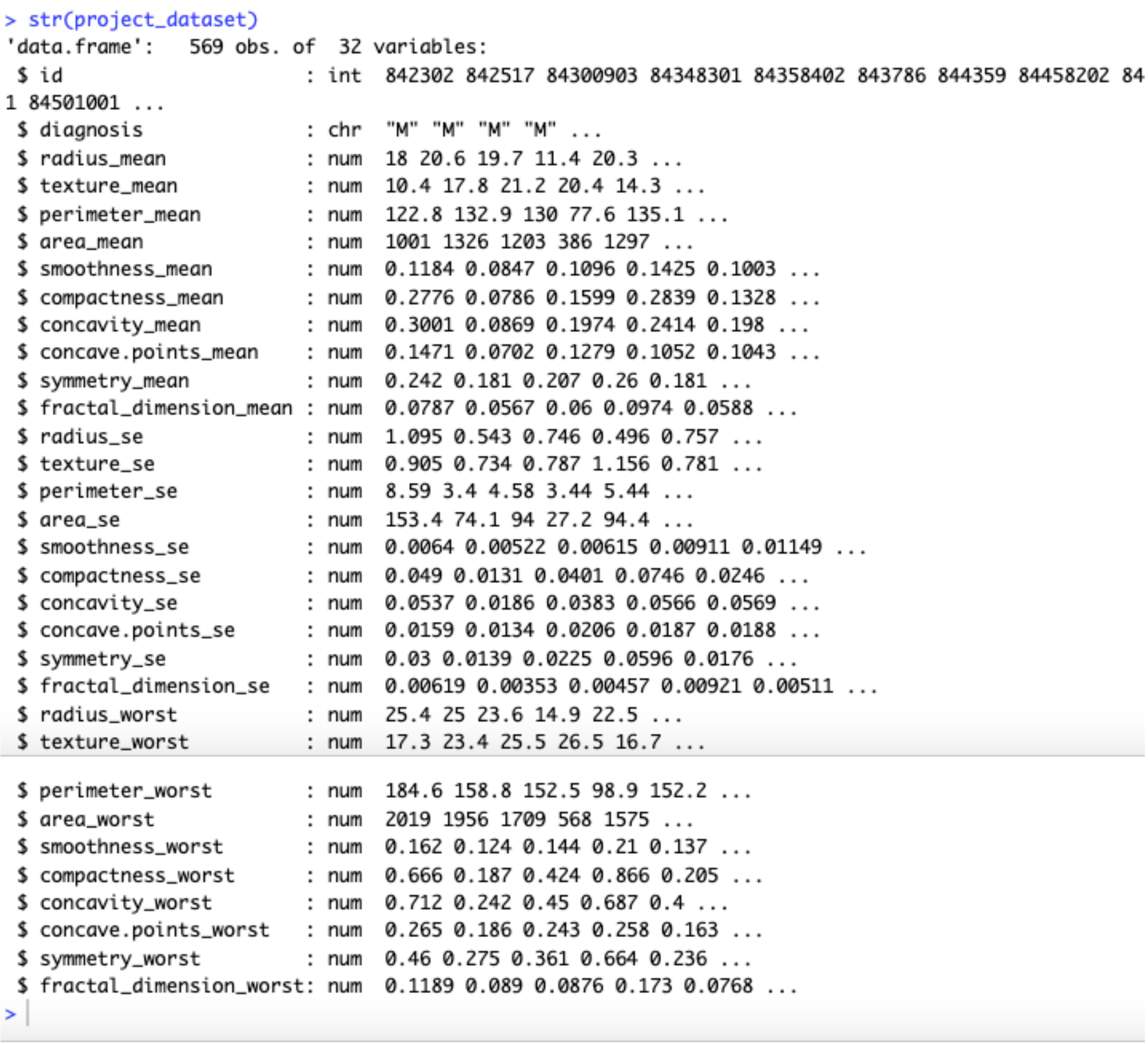
The dataset for this project named ‘project_dataset’ is read from the file ‘breast_cancer.csv’ using the function below,



```
> project_dataset <- read.csv('/Users/sakif/Desktop/R/ds_project_ft/breast-cancer.csv',
+                             header = TRUE, sep = ',');
> 
```

Figure 2: Reading the dataset

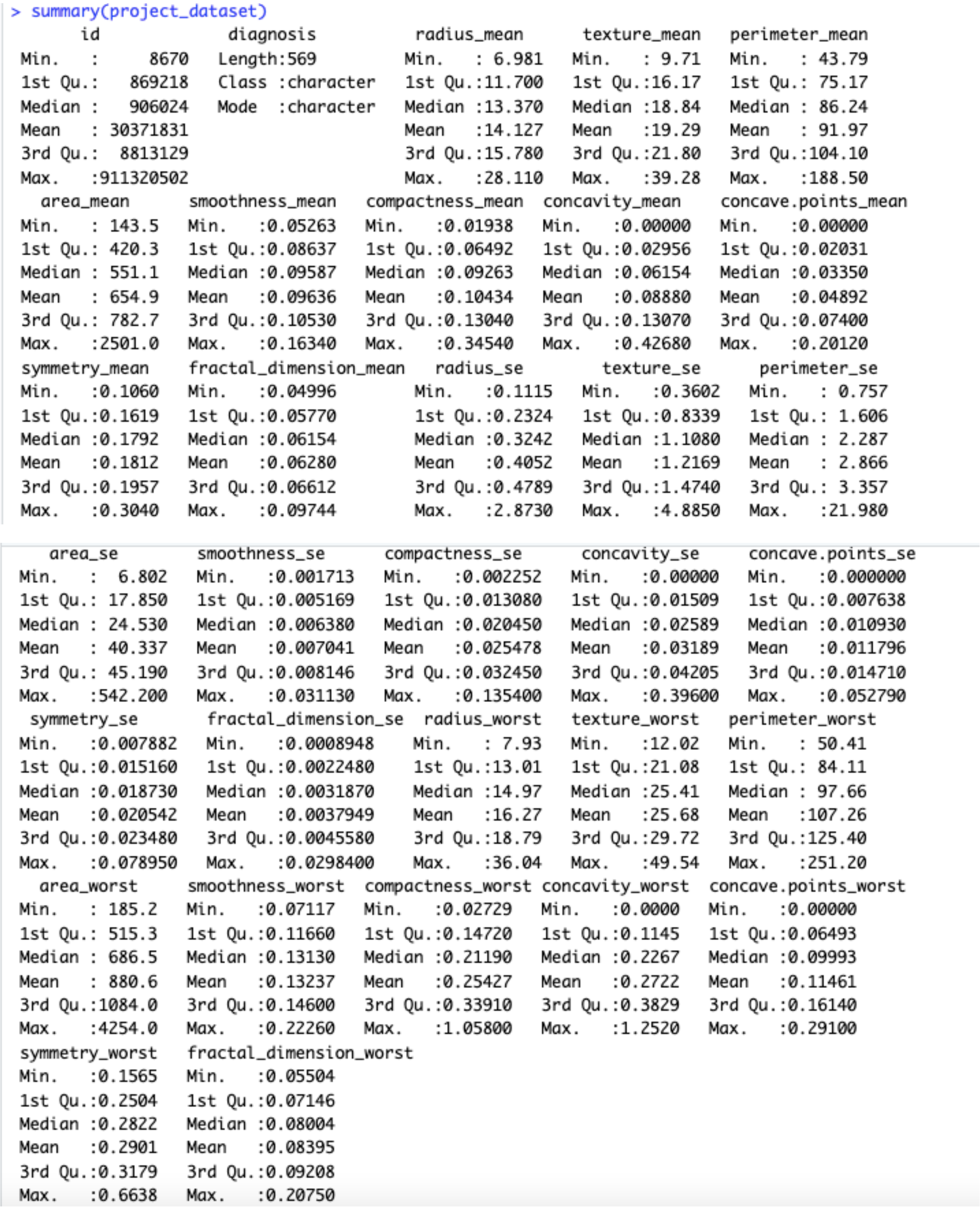
The dataset has 569 of instances and 32 attributes. Below, the details about every feature is shown,



```
> str(project_dataset)
'data.frame':	569 obs. of  32 variables:
 $ id                     : int  842302 842517 84300903 84348301 84358402 843786 844359 84458202 84
1 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 ...
> 
```

Figure 3: Details of the dataset

As, the ‘id’ attribute is used for indexing the instances. The summary of the ‘id’ attribute should not be considered. On the other hand, ‘diagnosis’ attribute is categorical. That’s why, only ‘mode value’ of this attribute is considerable. Rest of the features are summarized with their ‘min’, ‘max’, ‘1st quadrate’, ‘3rd quadrate’, ‘mean’, ‘median’ values. The summary of each feature is in below,



```
> summary(project_dataset)
       id              diagnosis          radius_mean      texture_mean    perimeter_mean  
 Min.   :     8670   Length:569         Min.   : 6.981   Min.   : 9.71   Min.   : 43.79  
 1st Qu.:   869218   Class :character   1st Qu.:11.700   1st Qu.:16.17   1st Qu.: 75.17  
 Median :   906024   Mode  :character   Median :13.370   Median :18.84   Median : 86.24  
 Mean   : 30371831                      Mean   :14.127   Mean   :19.29   Mean   : 91.97  
 3rd Qu.:  8813129                      3rd Qu.:15.780   3rd Qu.:21.80   3rd Qu.:104.10  
 Max.   :911320502                      Max.   :28.110   Max.   :39.28   Max.   :188.50  
   area_mean      smoothness_mean   compactness_mean  concavity_mean    concave.points_mean
 Min.   : 143.5   Min.   :0.05263   Min.   :0.01938   Min.   :0.00000   Min.   :0.00000    
 1st Qu.: 420.3   1st Qu.:0.08637   1st Qu.:0.06492   1st Qu.:0.02956   1st Qu.:0.02031    
 Median : 551.1   Median :0.09587   Median :0.09263   Median :0.06154   Median :0.03350    
 Mean   : 654.9   Mean   :0.09636   Mean   :0.10434   Mean   :0.08880   Mean   :0.04892    
 3rd Qu.: 782.7   3rd Qu.:0.10530   3rd Qu.:0.13040   3rd Qu.:0.13070   3rd Qu.:0.07400    
 Max.   :2501.0   Max.   :0.16340   Max.   :0.34540   Max.   :0.42680   Max.   :0.20120    
 symmetry_mean    fractal_dimension_mean   radius_se        texture_se      perimeter_se   
 Min.   :0.1060   Min.   :0.04996          Min.   :0.1115   Min.   :0.3602   Min.   : 0.757  
 1st Qu.:0.1619   1st Qu.:0.05770          1st Qu.:0.2324   1st Qu.:0.8339   1st Qu.: 1.606  
 Median :0.1792   Median :0.06154          Median :0.3242   Median :1.1080   Median : 2.287  
 Mean   :0.1812   Mean   :0.06280          Mean   :0.4052   Mean   :1.2169   Mean   : 2.866  
 3rd Qu.:0.1957   3rd Qu.:0.06612          3rd Qu.:0.4789   3rd Qu.:1.4740   3rd Qu.: 3.357  
 Max.   :0.3040   Max.   :0.09744          Max.   :2.8730   Max.   :4.8850   Max.   :21.980  
```

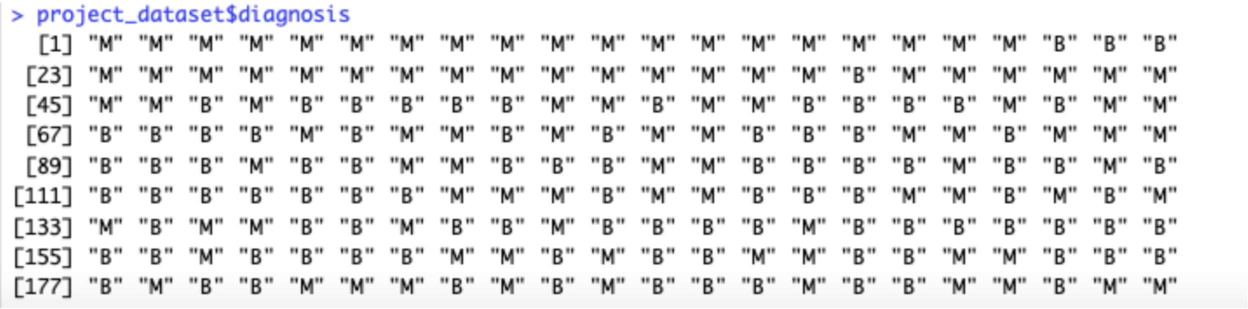
```
    area_se        smoothness_se      compactness_se      concavity_se     concave.points_se 
 Min.   :  6.802   Min.   :0.001713   Min.   :0.002252   Min.   :0.00000   Min.   :0.000000  
 1st Qu.: 17.850   1st Qu.:0.005169   1st Qu.:0.013080   1st Qu.:0.01509   1st Qu.:0.007638  
 Median : 24.530   Median :0.006380   Median :0.020450   Median :0.02589   Median :0.010930  
 Mean   : 40.337   Mean   :0.007041   Mean   :0.025478   Mean   :0.03189   Mean   :0.011796  
 3rd Qu.: 45.190   3rd Qu.:0.008146   3rd Qu.:0.032450   3rd Qu.:0.04205   3rd Qu.:0.014710  
 Max.   :542.200   Max.   :0.031130   Max.   :0.135400   Max.   :0.39600   Max.   :0.052790  
  symmetry_se        fractal_dimension_se  radius_worst    texture_worst   perimeter_worst 
 Min.   :0.007882   Min.   :0.0008948    Min.   : 7.93   Min.   :12.02   Min.   : 50.41  
 1st Qu.:0.015160   1st Qu.:0.0022480    1st Qu.:13.01   1st Qu.:21.08   1st Qu.: 84.11  
 Median :0.018730   Median :0.0031870    Median :14.97   Median :25.41   Median : 97.66  
 Mean   :0.020542   Mean   :0.0037949    Mean   :16.27   Mean   :25.68   Mean   :107.26  
 3rd Qu.:0.023480   3rd Qu.:0.0045580    3rd Qu.:18.79   3rd Qu.:29.72   3rd Qu.:125.40  
 Max.   :0.078950   Max.   :0.0298400    Max.   :36.04   Max.   :49.54   Max.   :251.20  
   area_worst     smoothness_worst  compactness_worst concavity_worst  concave.points_worst
 Min.   : 185.2   Min.   :0.07117   Min.   :0.02729   Min.   :0.0000   Min.   :0.00000     
 1st Qu.: 515.3   1st Qu.:0.11660   1st Qu.:0.14720   1st Qu.:0.1145   1st Qu.:0.06493     
 Median : 686.5   Median :0.13130   Median :0.21190   Median :0.2267   Median :0.09993     
 Mean   : 880.6   Mean   :0.13237   Mean   :0.25427   Mean   :0.2722   Mean   :0.11461     
 3rd Qu.:1084.0   3rd Qu.:0.14600   3rd Qu.:0.33910   3rd Qu.:0.3829   3rd Qu.:0.16140     
 Max.   :4254.0   Max.   :0.22260   Max.   :1.05800   Max.   :1.2520   Max.   :0.29100     
 symmetry_worst   fractal_dimension_worst
 Min.   :0.1565   Min.   :0.05504        
 1st Qu.:0.2504   1st Qu.:0.07146        
 Median :0.2822   Median :0.08004        
 Mean   :0.2901   Mean   :0.08395        
 3rd Qu.:0.3179   3rd Qu.:0.09208        
 Max.   :0.6638   Max.   :0.20750        
```

Figure 4: Summary of the dataset

Data Transformation –

As we know, for KNN classification algorithm every feature of the data should be in numerical type. Though, all the features have the numerical data except the ‘diagnosis’ attribute. This ‘diagnosis’ attribute has the value of ‘M’ or ‘B’. Now, this value should have the form of numerical to be operated. That’s why, ‘M’ is transformed into ‘1’ which means ‘cancerous’ and ‘B’ is transform into ‘0’ which means ‘non-cancerous’. In order to do this, below steps are followed,

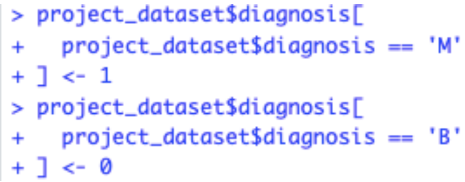
Step-1: Observing the ‘diagnosis’ attribute first,



```
> project_dataset$diagnosis
  [1] "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "B" "B" "B"
 [23] "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "B" "M" "M" "M" "M" "M" "M"
 [45] "M" "M" "B" "M" "B" "B" "B" "B" "B" "M" "M" "B" "M" "M" "B" "B" "B" "B" "M" "B" "M" "M"
 [67] "B" "B" "B" "B" "M" "B" "M" "M" "B" "M" "B" "M" "M" "B" "B" "B" "M" "M" "B" "M" "M" "M"
 [89] "B" "B" "B" "M" "B" "B" "M" "M" "B" "B" "B" "M" "M" "B" "B" "B" "B" "M" "B" "B" "M" "B"
[111] "B" "B" "B" "B" "B" "B" "B" "M" "M" "M" "B" "M" "M" "B" "B" "B" "M" "M" "B" "M" "B" "M"
[133] "M" "B" "M" "M" "B" "B" "M" "B" "B" "M" "B" "B" "B" "B" "M" "B" "B" "B" "B" "B" "B" "B"
[155] "B" "B" "M" "B" "B" "B" "B" "M" "M" "B" "M" "B" "B" "M" "M" "B" "B" "M" "M" "B" "B" "B"
[177] "B" "M" "B" "B" "M" "M" "M" "B" "M" "B" "M" "B" "B" "B" "M" "B" "B" "M" "M" "B" "M" "M"
```

Figure 5: Raw value of 'diagnosis' attribute

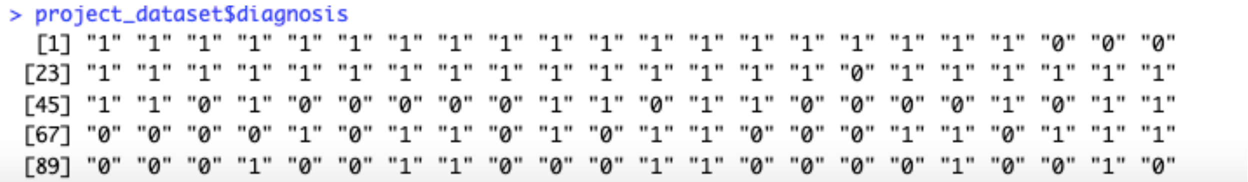
Step-2: Transforming the data,



```
> project_dataset$diagnosis[
+   project_dataset$diagnosis == 'M'
+ ] <- 1
> project_dataset$diagnosis[
+   project_dataset$diagnosis == 'B'
+ ] <- 0
```

Figure 6: Code of processing 'diagnosis' attribute

Step-3: Again, observing for ensuring the transformation,



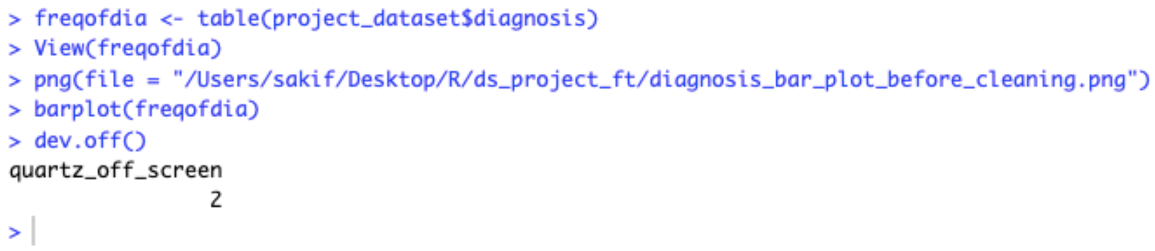
```
> project_dataset$diagnosis
  [1] "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "0" "0" "0"
 [23] "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "0" "1" "1" "1" "1" "1" "1"
 [45] "1" "1" "0" "1" "0" "0" "0" "0" "0" "1" "1" "0" "1" "1" "0" "0" "0" "0" "1" "0" "1" "1"
 [67] "0" "0" "0" "0" "1" "0" "1" "1" "0" "1" "0" "1" "1" "0" "0" "0" "1" "1" "0" "1" "1" "1"
 [89] "0" "0" "0" "1" "0" "0" "1" "1" "0" "0" "0" "1" "1" "0" "0" "0" "0" "1" "0" "0" "1" "0"
```

Figure 7: Transformed 'diagnosis' attribute

Though these data are in factor type but ready for the classification algorithm.

Handling Missing Values –

Now, let’s check the recently transformed attribute(‘diagnosis’) for missing values. For doing this, the bar plot has used. In this plot, if any value exists except ‘M’ & ‘B’, another bar will be sketched in it. The plot is in the below,



```
> freqofdia <- table(project_dataset$diagnosis)
> View(freqofdia)
> png(file = "/Users/sakif/Desktop/R/ds_project_ft/diagnosis_bar_plot_before_cleaning.png")
> barplot(freqofdia)
> dev.off()
quartz_off_screen
                2
> 
```

Figure 8: Code of bar plot

Through this bar plot, it is concluded that, there is no missing value in the ‘diagnosis’ attribute.

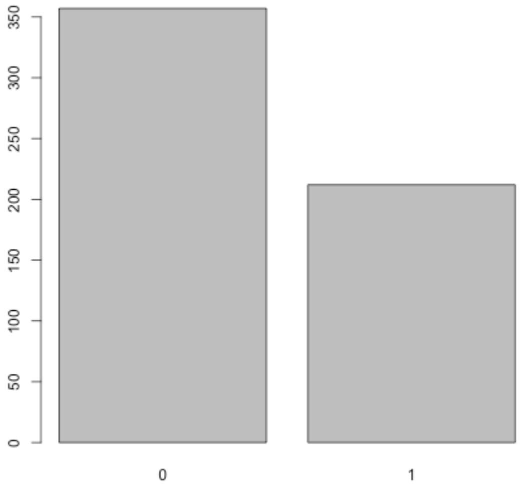
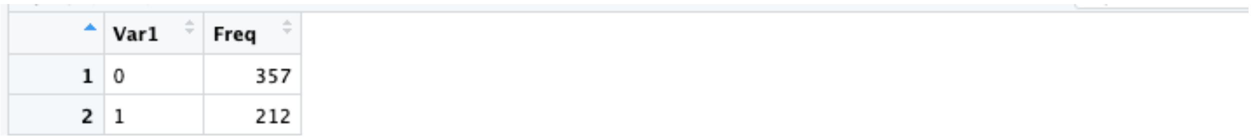


Figure 9: Applied bar plot in 'diagnosis' attribute

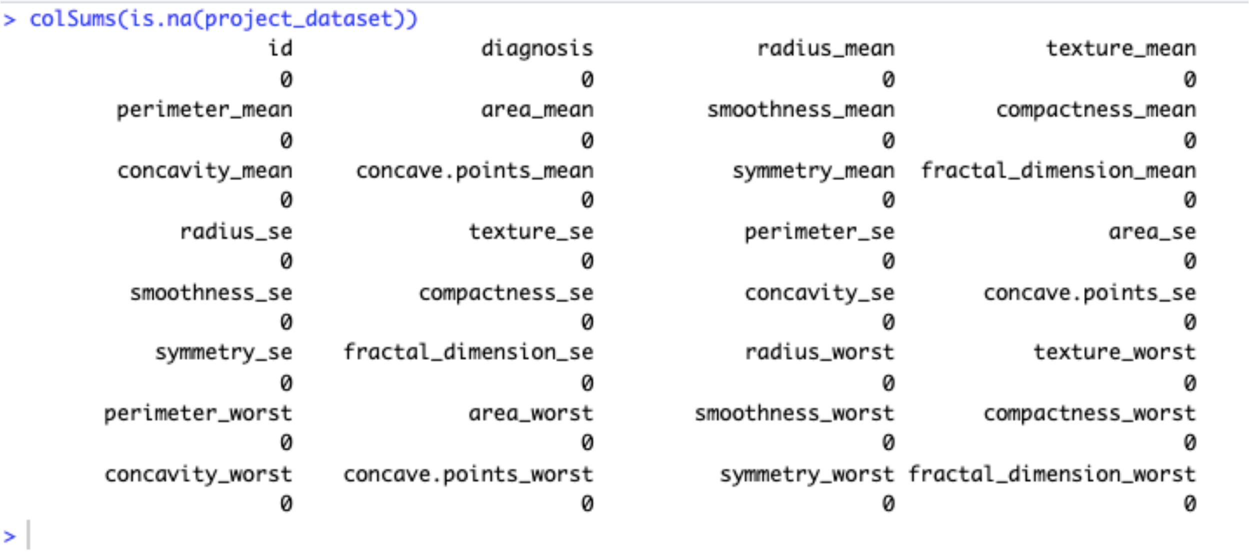
This frequency table emphasizes on, there is no values in the ‘diagnosis’ attribute except ‘0’ and ‘1’.



	Var1	Freq
1	0	357
2	1	212

Figure 10: Frequency table for 'diagnosis' attribute

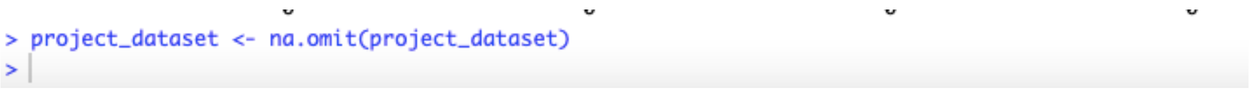
After this operation, the whole dataset has been checked for the missing values. Fortunately, there is no missing value at all.



```
> colSums(is.na(project_dataset))
                     id               diagnosis             radius_mean            texture_mean
                      0                       0                       0                       0
         perimeter_mean               area_mean         smoothness_mean        compactness_mean
                      0                       0                       0                       0
         concavity_mean     concave.points_mean           symmetry_mean  fractal_dimension_mean
                      0                       0                       0                       0
              radius_se              texture_se            perimeter_se                 area_se
                      0                       0                       0                       0
          smoothness_se          compactness_se            concavity_se       concave.points_se
                      0                       0                       0                       0
            symmetry_se    fractal_dimension_se            radius_worst           texture_worst
                      0                       0                       0                       0
        perimeter_worst              area_worst        smoothness_worst       compactness_worst
                      0                       0                       0                       0
        concavity_worst    concave.points_worst          symmetry_worst fractal_dimension_worst
                      0                       0                       0                       0
> 
```

Figure 11: Code of checking missing values

For the extra level of satisfaction, the process of missing value elimination was performed in the whole dataset.



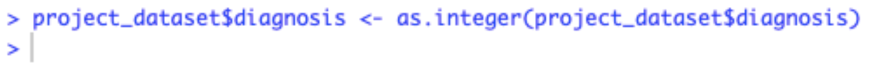
```
> project_dataset <- na.omit(project_dataset)
> 
```

Figure 12: Code of omitting missing values

Correlation & Feature Selection –

Correlation is one of the best strategy to select the appropriate features related to the target attribute and remove the attributes whose values are considered as duplicated or unnecessary for the classification or other model building processes. Here, the Pearson correlation coefficient approach has been used for defining the correlation among the whole dataset.

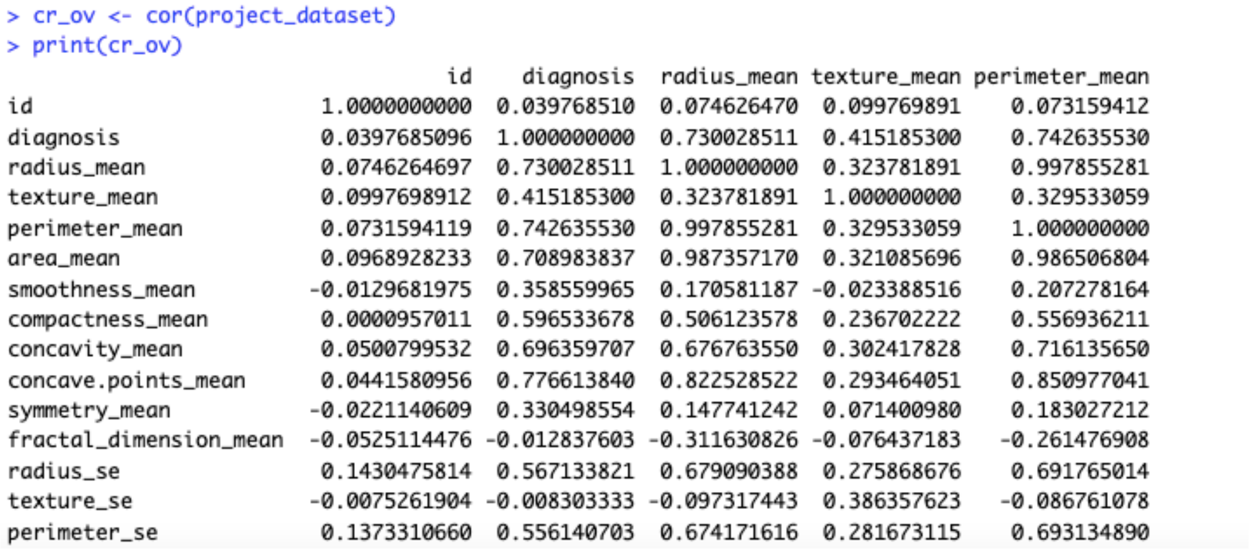
As we know, correlation can only be established among the numerical data and from the before process, the ‘diagnosis’ attribute was transformed into ‘factor’ datatype. So, this feature is converted into ‘integer’ first.



```
> project_dataset$diagnosis <- as.integer(project_dataset$diagnosis)
> 
```

Figure 13: Code of converting 'diagnosis' attribute factor to integer

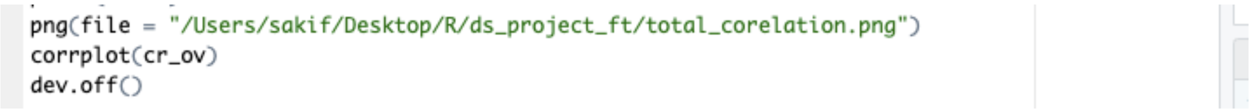
Now the correlation defining operation can be performed through the ‘cor’ function and the calculation is in below,



```
> cr_ov <- cor(project_dataset)
> print(cr_ov)
                                 id    diagnosis  radius_mean texture_mean perimeter_mean
id                     1.0000000000  0.039768510  0.074626470  0.099769891    0.073159412
diagnosis              0.0397685096  1.000000000  0.730028511  0.415185300    0.742635530
radius_mean            0.0746264697  0.730028511  1.000000000  0.323781891    0.997855281
texture_mean           0.0997698912  0.415185300  0.323781891  1.000000000    0.329533059
perimeter_mean         0.0731594119  0.742635530  0.997855281  0.329533059    1.000000000
area_mean              0.0968928233  0.708983837  0.987357170  0.321085696    0.986506804
smoothness_mean       -0.0129681975  0.358559965  0.170581187 -0.023388516    0.207278164
compactness_mean       0.0000957011  0.596533678  0.506123578  0.236702222    0.556936211
concavity_mean         0.0500799532  0.696359707  0.676763550  0.302417828    0.716135650
concave.points_mean    0.0441580956  0.776613840  0.822528522  0.293464051    0.850977041
symmetry_mean         -0.0221140609  0.330498554  0.147741242  0.071400980    0.183027212
fractal_dimension_mean -0.0525114476 -0.012837603 -0.311630826 -0.076437183   -0.261476908
radius_se              0.1430475814  0.567133821  0.679090388  0.275868676    0.691765014
texture_se            -0.0075261904 -0.008303333 -0.097317443  0.386357623   -0.086761078
perimeter_se           0.1373310660  0.556140703  0.674171616  0.281673115    0.693134890
```

Figure 14: View of correlation coefficients among the attributes

As we can see the above calculation is a little bit messy. So, in convenience to understand the correlation, the ‘corrplot’ function has been used to have a correlation plot diagram. In the plot diagram, common points represent the correlation between two attributes and the color intensities represent the measurement of the correlation coefficient. Below is the diagram,



```
png(file = "/Users/sakif/Desktop/R/ds_project_ft/total_corelation.png")
corrplot(cr_ov)
dev.off()
```

Figure 15: Plotting the correlation plot among the dataset

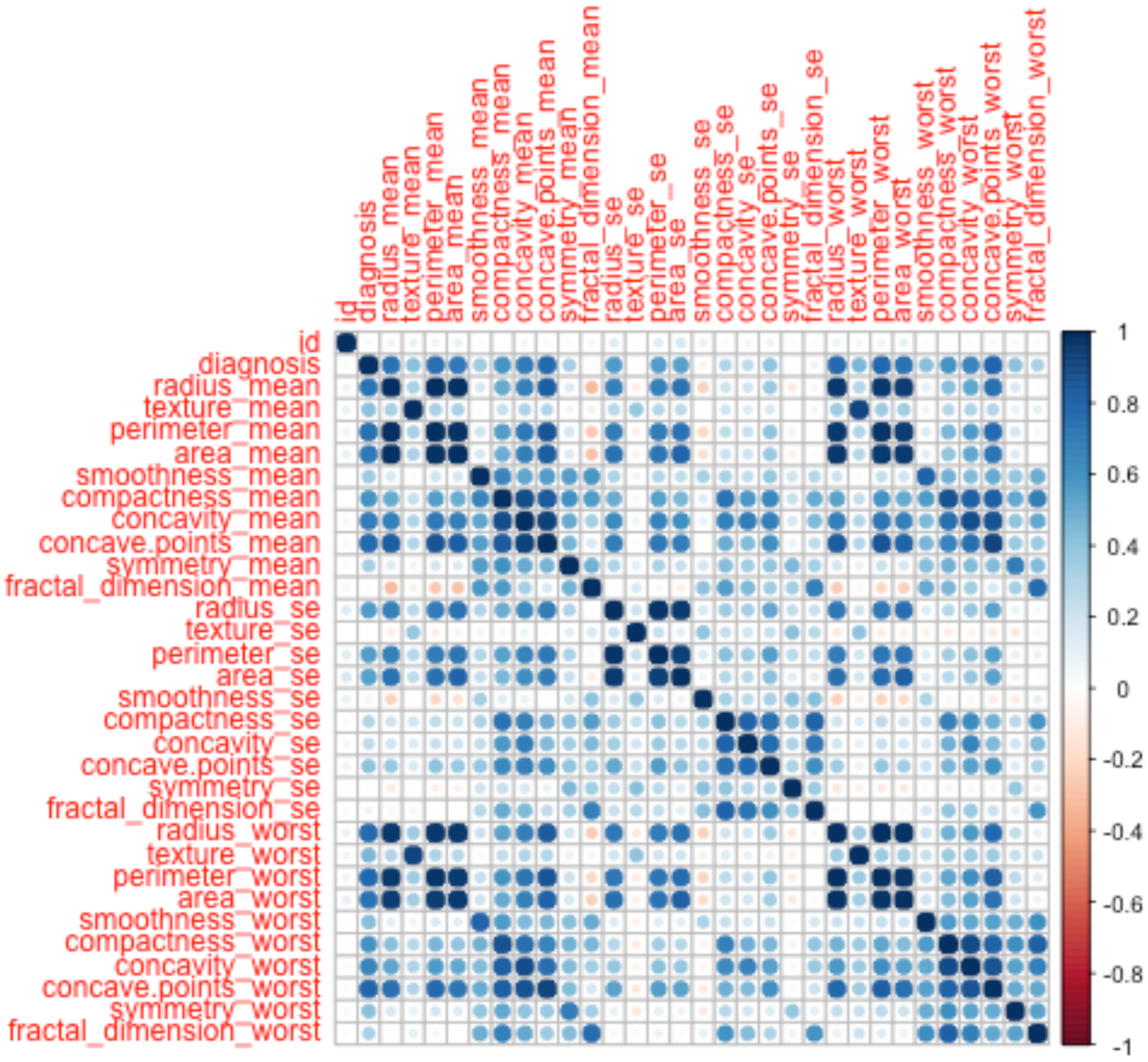


Figure 16: Correlation diagram of whole dataset

From the correlation diagram there are some observations such as,

Observation-1: the ‘id’ attribute is not correlated with any of the features as expected because it is only useful for indexing.

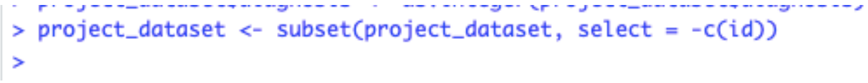
Observation-2: The features ‘fractal_dimension_mean’, ‘texture_se’, ‘smoothness_se’, ‘symmetry_se’, ‘fractal_dimension_se’ are not correlated with target attribute ‘diagnosis’.

Observation-3: There are some independent attributes which are highly correlated with other independent attributes, such as

Independent attributes	Highly correlated independent attributes
Radius_mean	perimeter_mean, area_mean, radius_worst, perimeter_worst, area_worst
Texture_mean	texture_worst
Compactness_mean	concavity_worst, concave.points_worst

Now, these observations will be used in omitting the unnecessary features and the rest of the features will be counted as the selected features. To select the feature, operations should be performed in steps regarding the observation,

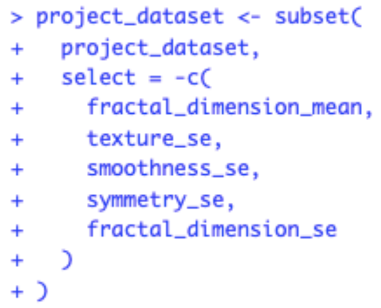
Handling observation-1: The 'id' attribute is omitted.



```
> project_dataset <- subset(project_dataset, select = -c(id))
>
```

Figure 17: Eliminating the 'id' attribute

Handling observation-2: The observed attributes which are not correlated should be omitted through below code,



```
> project_dataset <- subset(
+   project_dataset,
+   select = -c(
+     fractal_dimension_mean,
+     texture_se,
+     smoothness_se,
+     symmetry_se,
+     fractal_dimension_se
+   )
+ )
```

Figure 18: Eliminating the attributes which have no correlation with target attribute

Let’s check the correlation plot after omitting,

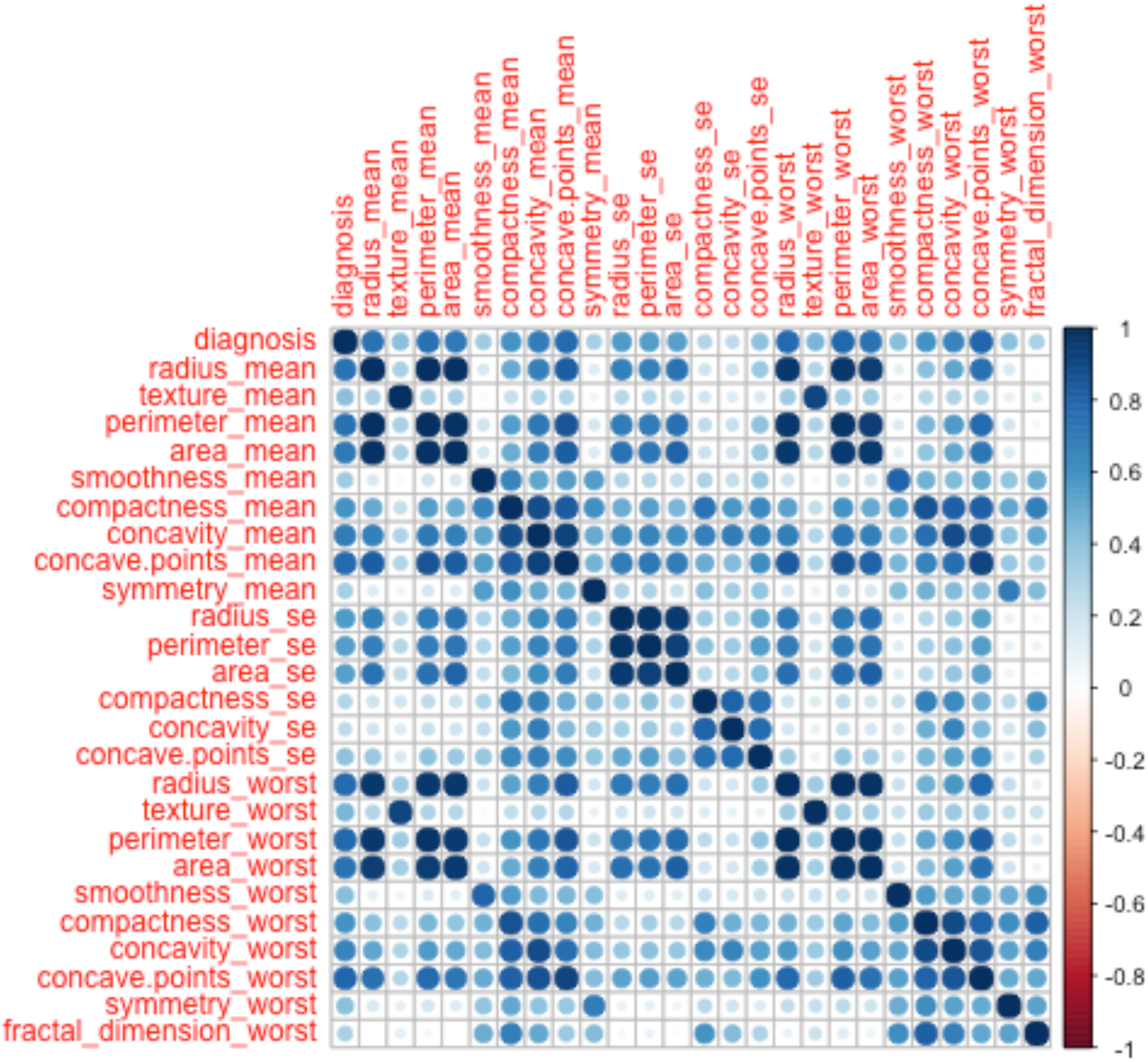
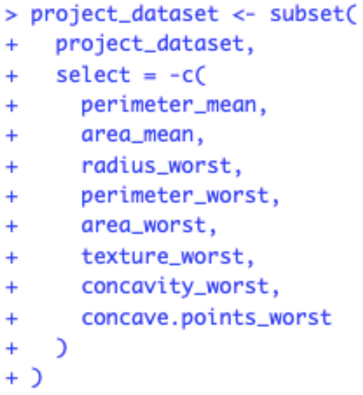


Figure 19: Correlation plot after handling 2nd observation

Handling observation-3: After handling the 2nd observation, the correlation plot says there are some independent features which are highly correlated with other independent features. Now, these highly correlated features should be omitted through below code,



```
> project_dataset <- subset(
+   project_dataset,
+   select = -c(
+     perimeter_mean,
+     area_mean,
+     radius_worst,
+     perimeter_worst,
+     area_worst,
+     texture_worst,
+     concavity_worst,
+     concave.points_worst
+   )
+ )
```

Figure 20: Eliminating the highly correlated independent attributes

Now, let's check the final correlation plot,

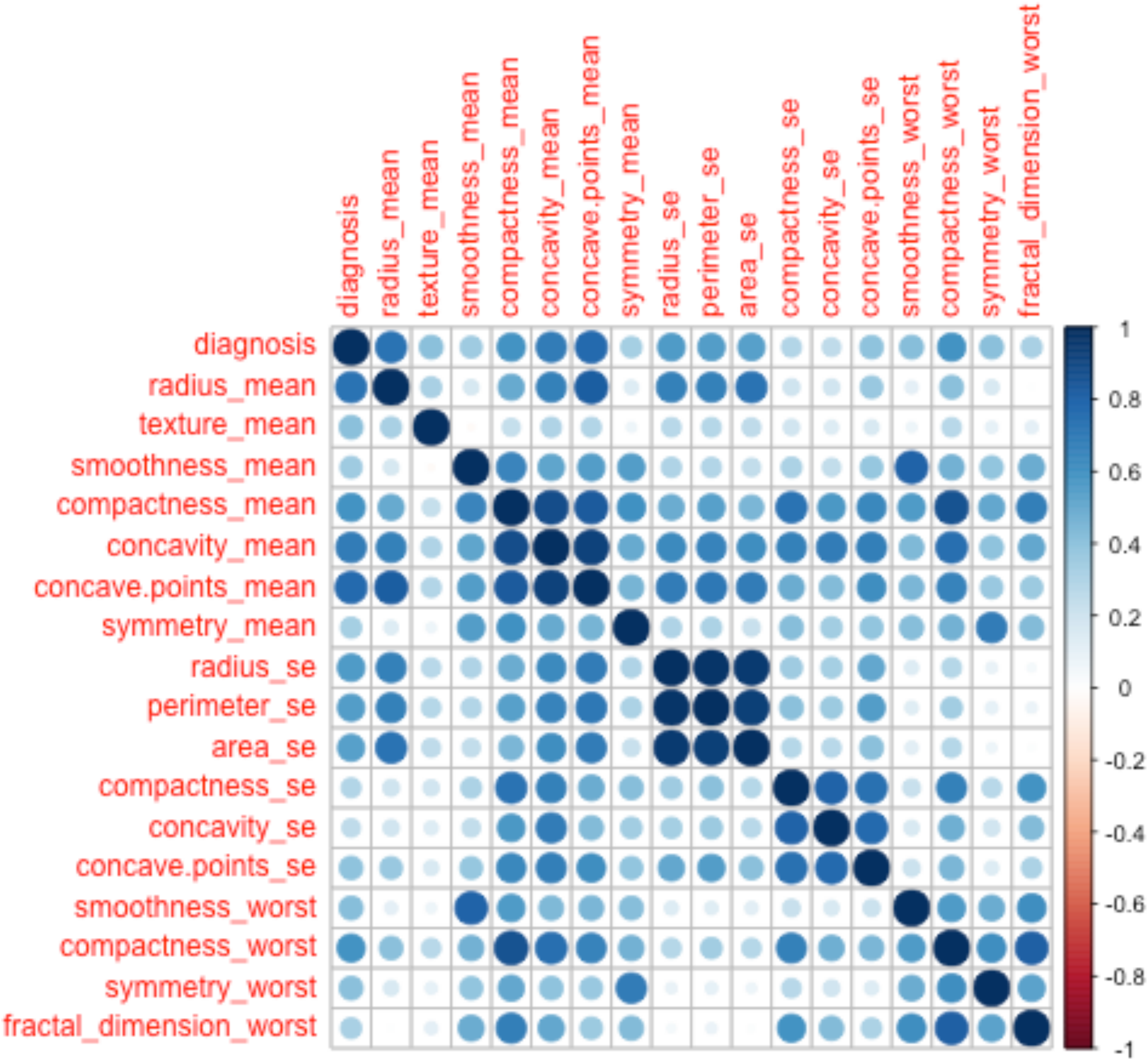
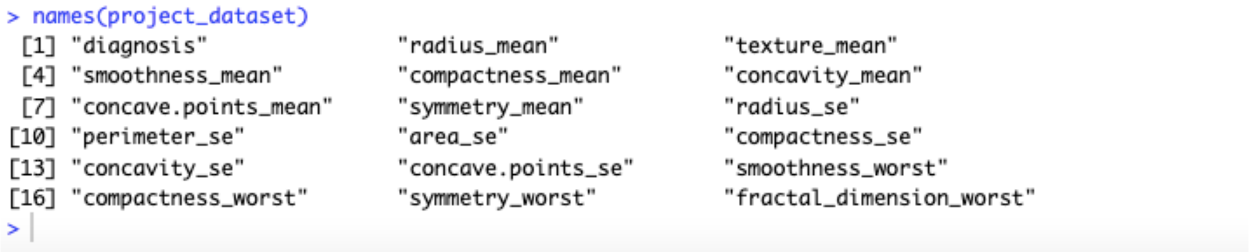


Figure 21: Correlation plot after final elimination

Though there are some highly correlated features which are present, the number of those features is few. Again, if these features get omitted that will make impact on the model accuracy.

After the elimination of features regarding the correlation, the selected features are in below,

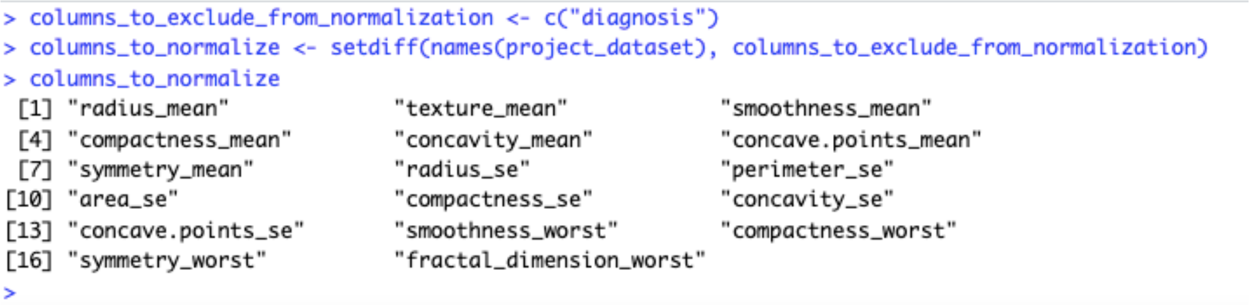


```
> names(project_dataset)
 [1] "diagnosis"               "radius_mean"              "texture_mean"
 [4] "smoothness_mean"         "compactness_mean"         "concavity_mean"
 [7] "concave.points_mean"     "symmetry_mean"            "radius_se"
[10] "perimeter_se"            "area_se"                  "compactness_se"
[13] "concavity_se"            "concave.points_se"        "smoothness_worst"
[16] "compactness_worst"       "symmetry_worst"           "fractal_dimension_worst"
> 
```

Figure 22: Final selected features for building the model

Normalization –

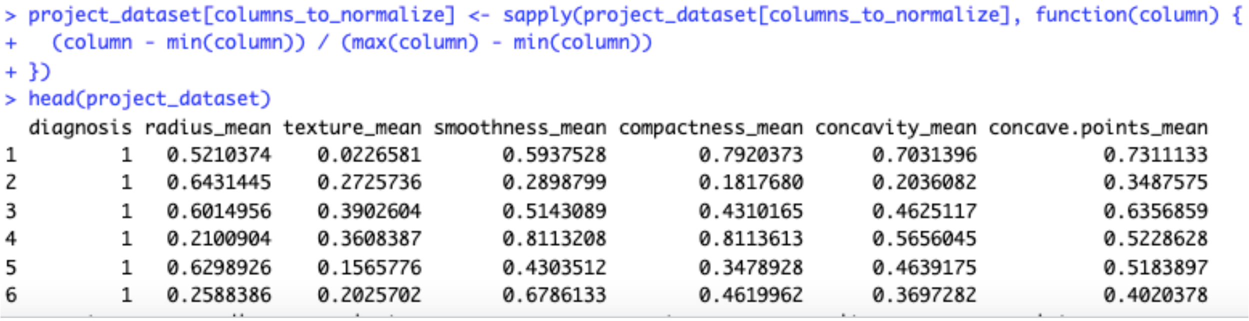
For the normalization, the min-max approach is applied to scale the values of those attribute in range of 0 to 1 here. As, the value of the 'diagnosis' attribute is categorical, normalization should not be performed to this attribute. That's why the features which have to be normalized are differentiated in below,



```
> columns_to_exclude_from_normalization <- c("diagnosis")
> columns_to_normalize <- setdiff(names(project_dataset), columns_to_exclude_from_normalization)
> columns_to_normalize
 [1] "radius_mean"            "texture_mean"            "smoothness_mean"
 [4] "compactness_mean"       "concavity_mean"          "concave.points_mean"
 [7] "symmetry_mean"          "radius_se"               "perimeter_se"
[10] "area_se"                "compactness_se"          "concavity_se"
[13] "concave.points_se"      "smoothness_worst"        "compactness_worst"
[16] "symmetry_worst"         "fractal_dimension_worst"
>
```

Figure 23: Select features to normalize

Now the min-max functionality is applied to the selected features for normalization and some data to show in below,



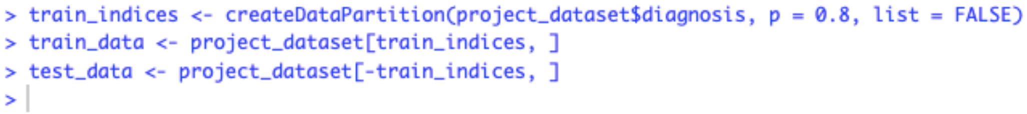
```
> project_dataset[columns_to_normalize] <- sapply(project_dataset[columns_to_normalize], function(column) {
+   (column - min(column)) / (max(column) - min(column))
+ })
> head(project_dataset)
  diagnosis radius_mean texture_mean smoothness_mean compactness_mean concavity_mean concave.points_mean
1         1   0.5210374    0.0226581       0.5937528        0.7920373      0.7031396           0.7311133
2         1   0.6431445    0.2725736       0.2898799        0.1817680      0.2036082           0.3487575
3         1   0.6014956    0.3902604       0.5143089        0.4310165      0.4625117           0.6356859
4         1   0.2100904    0.3608387       0.8113208        0.8113613      0.5656045           0.5228628
5         1   0.6298926    0.1565776       0.4303512        0.3478928      0.4639175           0.5183897
6         1   0.2588386    0.2025702       0.6786133        0.4619962      0.3697282           0.4020378
```

Figure 24: Applying the normalization and view of some normalized data

Model Building –

At this part of the project, the whole dataset is ready for developing the model. In order to do it, some steps must be followed,

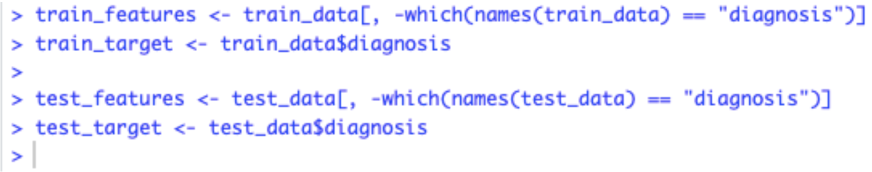
Step-1: Splitting the data instances into train data and test data. Train data will get the 80% of the instances and the test data will get 20% of the instances. Using 'createDataPartition' function from the module 'caret' the data has been divided through this process,



```
> train_indices <- createDataPartition(project_dataset$diagnosis, p = 0.8, list = FALSE)
> train_data <- project_dataset[train_indices, ]
> test_data <- project_dataset[-train_indices, ]
> 
```

Figure 25: Code of data splitting into 8:2 ratio.

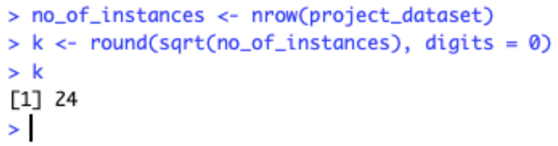
Step-2: Separating the target attribute and rest of the features instances like this,



```
> train_features <- train_data[, -which(names(train_data) == "diagnosis")]
> train_target <- train_data$diagnosis
>
> test_features <- test_data[, -which(names(test_data) == "diagnosis")]
> test_target <- test_data$diagnosis
> 
```

Figure 26: Code of separating data into train and target features

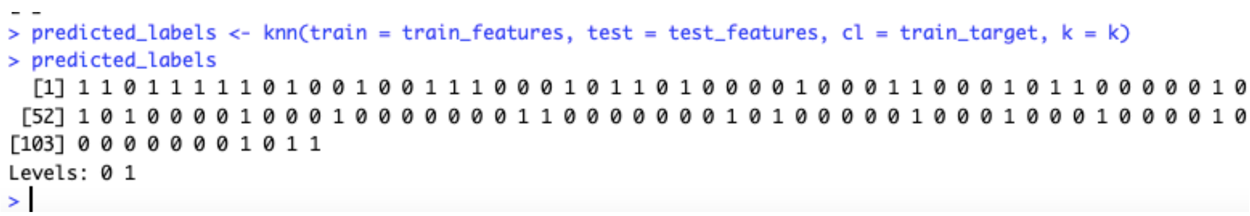
Step-3: Calculating the optimized value of K. The optimized value of K is the square root of the total instances.



```
> no_of_instances <- nrow(project_dataset)
> k <- round(sqrt(no_of_instances), digits = 0)
> k
[1] 24
> 
```

Figure 27: Calculating the optimized value of K for the KNN classification

Step-4: Developing the model through 'knn' function of 'class' module. This function returns the predicted instances of 'target' attribute against the instances of 'test_features' attributes.



```
- -
> predicted_labels <- knn(train = train_features, test = test_features, cl = train_target, k = k)
> predicted_labels
  [1] 1 1 0 1 1 1 1 1 0 1 0 0 1 0 0 1 1 1 0 0 0 1 0 1 1 0 1 0 0 0 0 1 0 0 0 1 1 0 0 0 1 0 1 1 0 0 0 0 0 1 0
 [52] 1 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 0
[103] 0 0 0 0 0 0 0 1 0 1 1
Levels: 0 1
> 
```

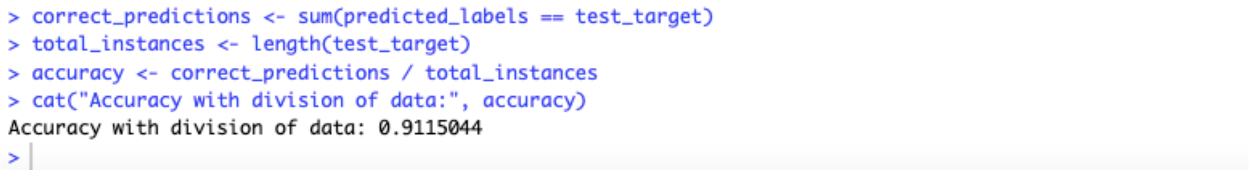
Figure 28: KNN classification is applied and the predicted labels are shown

Another approach of developing model will be followed in the Accuracy Measurement section.

Accuracy Measurement –

For calculating accuracy, two approaches are followed, first one is division and the second one is k-fold cross validation.

Approach-1: In accuracy of division, firstly the total number of correct predictions is calculated. Then the total number of correct predictions is divided by the total instances of ‘test_target’ attribute. The final result is considered as the accuracy of the model.

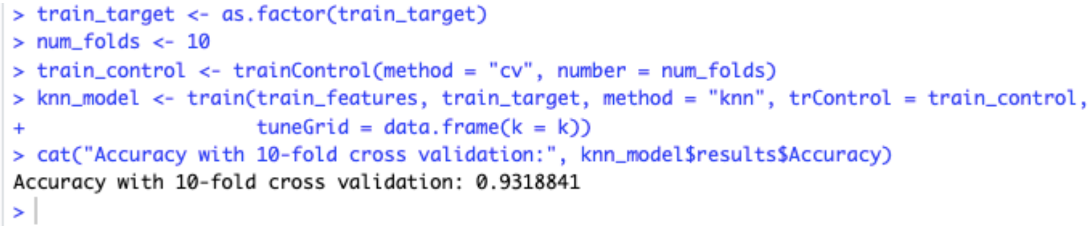


```
> correct_predictions <- sum(predicted_labels == test_target)
> total_instances <- length(test_target)
> accuracy <- correct_predictions / total_instances
> cat("Accuracy with division of data:", accuracy)
Accuracy with division of data: 0.9115044
> 
```

Figure 29: Code of calculating the accuracy in dividing method

Approach-2: In this k-fold cross validation, the k value is 10, that’s why this approach is named as 10-fold cross validation.

The ‘trainControl’ and ‘train’ functions from ‘class’ module are used for developing the model. The ‘trainControl’ function is setting the number of folds which is 10. The ‘train’ function is building the model. As the target attribute ‘diagnosis’ was converted to ‘integer’ for the correlation before, this ‘train_target’ attribute is converted to ‘factor’ again. Because the ‘train’ function only accepts the ‘factor’ type ‘train_target’. After that the property ‘$results$Accuracy’ is used to show the accuracy.

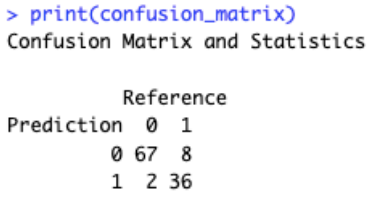


```
> train_target <- as.factor(train_target)
> num_folds <- 10
> train_control <- trainControl(method = "cv", number = num_folds)
> knn_model <- train(train_features, train_target, method = "knn", trControl = train_control,
+                    tuneGrid = data.frame(k = k))
> cat("Accuracy with 10-fold cross validation:", knn_model$results$Accuracy)
Accuracy with 10-fold cross validation: 0.9318841
> 
```

Figure 30: Code of calculating the accuracy in 10-fold cross validation method

Confusion Matrix –

For the confusion matrix is defined through the ‘confusionMatrix’ function from the ‘caret’ module. It uses two arguments ‘predicted_labels’ from the model building approach-1 and the ‘test_target’ which needs to be ‘factor’ datatype. As the target attribute ‘diagnosis’ was converted to ‘integer’ for the correlation before, this ‘test_target’ attribute is converted to ‘factor’ again. Finally, the confusion matrix is generated like this,

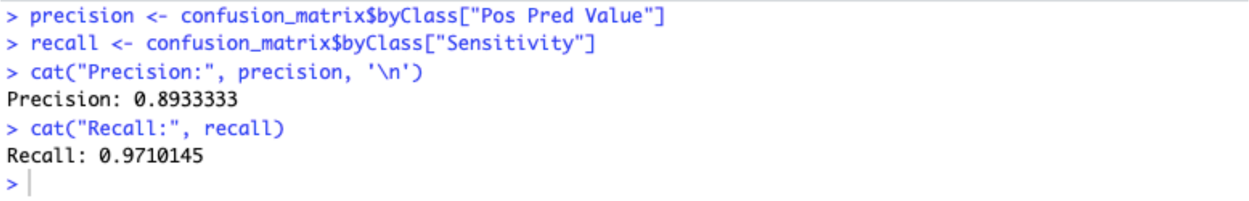


```
> print(confusion_matrix)
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 67  8
         1  2 36
```

Figure 31: Confusion matrix

Now the values of these properties ‘Pos Pred Value’ & ‘Sentivity’ which came from the object of the ‘confusionMatrix’ function represent the ‘precision’ & ‘recall’ values repectfully as like as this,



```
> precision <- confusion_matrix$byClass["Pos Pred Value"]
> recall <- confusion_matrix$byClass["Sensitivity"]
> cat("Precision:", precision, '\n')
Precision: 0.8933333
> cat("Recall:", recall)
Recall: 0.9710145
> 
```

Figure 32: The values of precision & recall