

Data Science: Capstone - Create Your Own Project (Prediction of Breast Cancer)

KLC

May 1, 2020

Objective

The aim of this project is to build a machine learning algorithm to predict whether a breast mass cell is “benign” or “malignant”. Breast lumps that are benign are mostly non-cancerous and not life threatening. They do not spread outside of the breast. Malignant lumps are cancerous however. Such kind of prediction algorithm could help medical practitioners to detect and diagnose breast cancer.

We will train a few machine learning models and measure their performance with their prediction sensitivity and F1 score. Our aim is to choose a model which yields the highest sensitivity (i.e. low false-negative) and F1 score.

This report will first explore the dataset, then analyse several models, compare their performance and conclude the result.

Dataset

Breast Cancer Wisconsin (Diagnostic) DataSet obtained from Kaggle (<https://www.kaggle.com/lbronchal/breast-cancer-dataset-analysis>) is used for this project. Such data, collected in 1993 by the University of Wisconsin, contains 569 samples of measurements on cells in suspicious lumps in a women’s breast. 20% of the data will be used for testing, while the remaining will be used for training the machine learning algorithm.

****Description of data:**** The dataset contains 569 observations with 33 variables including 30 “features” as listed below. “Features” are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass, which describe characteristics of the cell nuclei present in the image.

Attribute Information:

- 1) ID number
- 2) Diagnosis (M = malignant, B = benign) 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 ($\text{perimeter}^2 / \text{area} - 1.0$)
- 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. All feature values are recorded with 4 significant digits.

```
#####
# Load dataset
#####

data <- read.csv("https://raw.githubusercontent.com/happycheers/BreastCancer/master/data.csv")
```

Data exploration

First, let’s grab an overview of the dataset.

```
#Structure of the dataset
str(data)
```

```
## 'data.frame': 569 obs. of 33 variables:
## $ id : int 842302 842517 84300903 84348301 84358402 843786 844359 84458202 844...
## $ diagnosis : Factor w/ 2 levels "B","M": 2 2 2 2 2 2 2 2 2 ...
## $ radius_mean : num 18 20.6 19.7 11.4 20.3 ...
## $ texture_mean : num 10.4 17.8 21.2 20.4 14.3 ...
## $ perimeter_mean : num 122.8 132.9 130 77.6 135.1 ...
## $ area_mean : num 1001 1326 1203 386 1297 ...
## $ smoothness_mean : num 0.1184 0.0847 0.1096 0.1425 0.1003 ...
## $ compactness_mean : num 0.2776 0.0786 0.1599 0.2839 0.1328 ...
## $ concavity_mean : num 0.3001 0.0869 0.1974 0.2414 0.198 ...
## $ concave.points_mean : num 0.1471 0.0702 0.1279 0.1052 0.1043 ...
## $ symmetry_mean : num 0.242 0.181 0.207 0.26 0.181 ...
## $ fractal_dimension_mean : num 0.0787 0.0567 0.06 0.0974 0.0588 ...
## $ radius_se : num 1.095 0.543 0.746 0.496 0.757 ...
## $ texture_se : num 0.905 0.734 0.787 1.156 0.781 ...
## $ perimeter_se : num 8.59 3.4 4.58 3.44 5.44 ...
## $ area_se : num 153.4 74.1 94 27.2 94.4 ...
## $ smoothness_se : num 0.0064 0.00522 0.00615 0.00911 0.01149 ...
## $ compactness_se : num 0.049 0.0131 0.0401 0.0746 0.0246 ...
## $ concavity_se : num 0.0537 0.0186 0.0383 0.0566 0.0569 ...
## $ concave.points_se : num 0.0159 0.0134 0.0206 0.0187 0.0188 ...
## $ symmetry_se : num 0.03 0.0139 0.0225 0.0596 0.0176 ...
## $ fractal_dimension_se : num 0.00619 0.00353 0.00457 0.00921 0.00511 ...
## $ radius_worst : num 25.4 25 23.6 14.9 22.5 ...
## $ texture_worst : num 17.3 23.4 25.5 26.5 16.7 ...
## $ perimeter_worst : num 184.6 158.8 152.5 98.9 152.2 ...
## $ area_worst : num 2019 1956 1709 568 1575 ...
## $ smoothness_worst : num 0.162 0.124 0.144 0.21 0.137 ...
## $ compactness_worst : num 0.666 0.187 0.424 0.866 0.205 ...
## $ concavity_worst : num 0.712 0.242 0.45 0.687 0.4 ...
```

```
## $ concave.points_worst : num 0.265 0.186 0.243 0.258 0.163 ...
## $ symmetry_worst : num 0.46 0.275 0.361 0.664 0.236 ...
## $ fractal_dimension_worst: num 0.1189 0.089 0.0876 0.173 0.0768 ...
## $ X : logi NA NA NA NA NA NA ...
```

First 6 rows and header

```
head(data)
```

```
##      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
## 6   843786      M      12.45      15.70      82.57      477.1
##      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
## 6      0.12780      0.17000      0.1578      0.08089
##      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
## 6      0.2087      0.07613      0.3345      0.8902      2.217
##      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
## 6   27.19      0.007510      0.03345      0.03672      0.01137
##      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
## 6   0.02165      0.005082      15.47      23.75      103.40
##      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
## 6    741.6      0.1791      0.5249      0.5355
##      concave.points_worst symmetry_worst fractal_dimension_worst X
## 1      0.2654      0.4601      0.11890 NA
## 2      0.1860      0.2750      0.08902 NA
## 3      0.2430      0.3613      0.08758 NA
```

```
## 4          0.2575          0.6638          0.17300 NA
## 5          0.1625          0.2364          0.07678 NA
## 6          0.1741          0.3985          0.12440 NA
```

```
# Summary of statistics
summary(data)
```

```
##          id          diagnosis radius_mean texture_mean
## Min.      : 8670      B:357      Min.      : 6.981      Min.      : 9.71
## 1st Qu.: 869218      M:212      1st Qu.:11.700      1st Qu.:16.17
## Median : 906024                      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.      :0.00000      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. :0.00000 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
## X
## Mode:logical
## NA's:569
##
##
##
##
```

```
# Summarize number of diagnosis ("B" and "M") in the dataset
data %>% group_by(diagnosis) %>% summarize(n())
```

```
## # A tibble: 2 x 2
##   diagnosis `n()`
##   <fct>      <int>
## 1 B          357
## 2 M          212
```

We note from the above that the diagnosis is slightly unbalanced. We may have to adjust the data when using some models so that they can work properly.

Data cleaning

```
#####
# Data cleaning
#####

# Remove columns 1 and 33 as irrelevant
data <- data[,-33]
data <- data[,-1]

# Check if there are missing values
map_int(data, function(.x) sum(is.na(.x)))
```

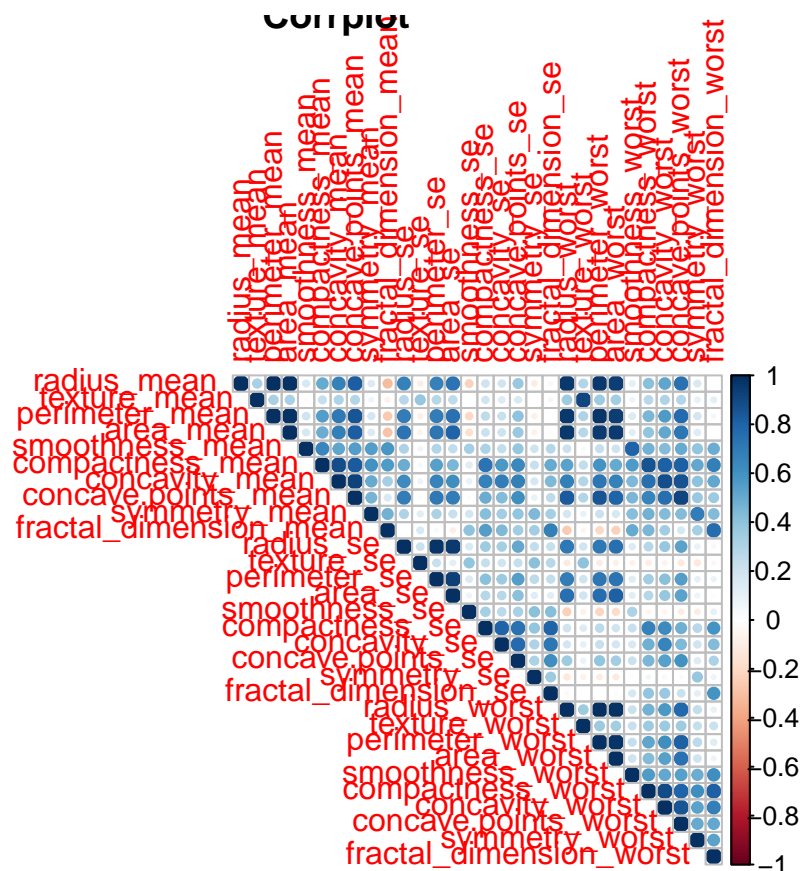
```
##           diagnosis           radius_mean           texture_mean
##           0                0                0
##   perimeter_mean           area_mean           smoothness_mean
##           0                0                0
##   compactness_mean           concavity_mean           concave.points_mean
##           0                0                0
##           symmetry_mean           fractal_dimension_mean           radius_se
##           0                0                0
```

```
##          texture_se      perimeter_se      area_se
##          0              0              0
##      smoothness_se      compactness_se      concavity_se
##          0              0              0
##      concave.points_se      symmetry_se      fractal_dimension_se
##          0              0              0
##      radius_worst      texture_worst      perimeter_worst
##          0              0              0
##      area_worst      smoothness_worst      compactness_worst
##          0              0              0
##      concavity_worst      concave.points_worst      symmetry_worst
##          0              0              0
## fractal_dimension_worst
##          0
```

We now have removed the 1st column "id" and the 33rd column "X" as they appear irrelevant to our prediction. We have also checked that there are no missing values in the dataset.

Some models such as naive bayes do not work well with highly-correlated variables as they assume the predictor variables are independent with each other. Therefore, we will check if the variables of the dataset are highly correlated and will remove them if their correlation coefficients are higher than 0.9 or lower than -0.9.

```
# Plot the correlation among variables
corrplot(cor(data[,2:31]) , main=" Corrplot" , method = "circle" , type = "upper")
```



```

# Identify variables with correlation coefficient higher than 0.9 or lower than -0.9
to_drop_col <- findCorrelation(cor(data[,2:31]), cutoff=0.9)

# Adjust the result by one column shift
to_drop_col <- to_drop_col + 1

# Remove highly correlated variables
new_data <- data[,-to_drop_col]

# Cross-check if highly correlated variables have been removed
findCorrelation(cor(new_data[,2:21]), cutoff=0.9)

```

```
## integer(0)
```

Now, we are going to divide the dataset into training (80%) and testing (20%) datasets.

```

#####
# Create training and testing sets
#####

# Divide the data set into training (80%) and testing (20%) sets
set.seed(1234, sample.kind="Rounding")

```

```
## Warning in set.seed(1234, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```

index <- createDataPartition(new_data$diagnosis, times=1, p=0.8, list = FALSE)
train <- new_data[index, ]
test <- new_data[-index, ]

```

Data Analysis - Modelling Approach

In the following, we will train a naive bayes model, logistic regression model, k-nearest neighbor model and random forest model.

```

# Cross validation with 10 folds
tc <- trainControl(method="cv", number = 10, classProbs=TRUE, summaryFunction = twoClassSummary)

```

Naive Bayes Model

```

#####
# Naive bayes model
#####

# Train a naive bayes model
naiveb_model <- train(diagnosis~.,
                      train,
                      method="nb",

```

```

metric = "ROC",
preProcess=c('center','scale'),
trControl=tc)

# Predict testing set
naiveb_pred <- predict(naiveb_model, test)

# summarize results (set positive as "M" so that the sensitivity is correct)
naiveb_result <- confusionMatrix(naiveb_pred, test$diagnosis, positive = "M")
naiveb_result

```

```

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##           B 67  3
##           M  4 39
##
##           Accuracy : 0.9381
##           95% CI : (0.8765, 0.9747)
##           No Information Rate : 0.6283
##           P-Value [Acc > NIR] : 1.718e-14
##
##           Kappa : 0.868
##
##           Mcnemar's Test P-Value : 1
##
##           Sensitivity : 0.9286
##           Specificity : 0.9437
##           Pos Pred Value : 0.9070
##           Neg Pred Value : 0.9571
##           Prevalence : 0.3717
##           Detection Rate : 0.3451
##           Detection Prevalence : 0.3805
##           Balanced Accuracy : 0.9361
##
##           'Positive' Class : M
##

```

Logistic Regression Model

```

#####
# Logistic regression model
#####

# Train a logistic regression model
glm_model <- train(diagnosis~.,
  train,
  method="glm",
  metric = "ROC",
  preProcess=c('center','scale'),

```



```

trControl=tc)

# Predict testing set
glm_pred <- predict(glm_model, test)

# summarize results (set positive as "M" so that the sensitivity is correct)
glm_result <- confusionMatrix(glm_pred, test$diagnosis, positive = "M")
glm_result

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B   M
##           B 69   1
##           M   2 41
##
##           Accuracy : 0.9735
##           95% CI : (0.9244, 0.9945)
##           No Information Rate : 0.6283
##           P-Value [Acc > NIR] : <2e-16
##
##           Kappa : 0.9434
##
##           Mcnemar's Test P-Value : 1
##
##           Sensitivity : 0.9762
##           Specificity : 0.9718
##           Pos Pred Value : 0.9535
##           Neg Pred Value : 0.9857
##           Prevalence : 0.3717
##           Detection Rate : 0.3628
##           Detection Prevalence : 0.3805
##           Balanced Accuracy : 0.9740
##
##           'Positive' Class : M
##

```

K-nearest Neighbor Model

```

#####
# K-nearest neighbor model
#####

# Train a KNN model
knn_model <- train(diagnosis~.,
  train,
  method="knn",
  metric = "ROC",
  preProcess=c('center', 'scale'),
  tuneLength=10,
  trControl=tc)

```

```

# Predict testing set
knn_pred <- predict(knn_model, test)

# summarize results (set positive as "M" so that the sensitivity is correct)
knn_result <- confusionMatrix(knn_pred, test$diagnosis, positive = "M")
knn_result

```

```

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##           B 71  3
##           M  0 39
##
##               Accuracy : 0.9735
##               95% CI : (0.9244, 0.9945)
##       No Information Rate : 0.6283
##       P-Value [Acc > NIR] : <2e-16
##
##               Kappa : 0.9423
##
##  Mcnemar's Test P-Value : 0.2482
##
##               Sensitivity : 0.9286
##               Specificity : 1.0000
##       Pos Pred Value : 1.0000
##       Neg Pred Value : 0.9595
##       Prevalence : 0.3717
##       Detection Rate : 0.3451
##       Detection Prevalence : 0.3451
##       Balanced Accuracy : 0.9643
##
##       'Positive' Class : M
##

```

Random Forest Model

```

#####
# Random forest model
#####

# Train a random forest model
rf_model <- train(diagnosis~.,
                  train,
                  method="rf",
                  metric = "ROC",
                  preProcess=c('center','scale'),
                  trControl=tc)

# Predict testing set
rf_pred <- predict(rf_model, test)

```

```
# summarize results (set positive as "M" so that the sensitivity is correct)
rf_result <- confusionMatrix(rf_pred, test$diagnosis, positive = "M")
rf_result
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B   M
##           B 67  3
##           M  4 39
##
##           Accuracy : 0.9381
##           95% CI : (0.8765, 0.9747)
##           No Information Rate : 0.6283
##           P-Value [Acc > NIR] : 1.718e-14
##
##           Kappa : 0.868
##
##           Mcnemar's Test P-Value : 1
##
##           Sensitivity : 0.9286
##           Specificity : 0.9437
##           Pos Pred Value : 0.9070
##           Neg Pred Value : 0.9571
##           Prevalence : 0.3717
##           Detection Rate : 0.3451
##           Detection Prevalence : 0.3805
##           Balanced Accuracy : 0.9361
##
##           'Positive' Class : M
##
```

Results

The results of each model developed above are summarized below:

```
#####
# Results
#####

# Summarize the confusion matrixes of each model
result_list <- list (naive_bayes = naiveb_result,
                    logistic_regression = glm_result,
                    KNN = knn_result,
                    random_forest = rf_result)
results <- sapply (result_list, function(x) x$byClass)

# Print the results in a table format
results %>% knitr::kable()
```

	naive_bayes	logistic_regression	KNN	random_forest
Sensitivity	0.9285714	0.9761905	0.9285714	0.9285714
Specificity	0.9436620	0.9718310	1.0000000	0.9436620
Pos Pred Value	0.9069767	0.9534884	1.0000000	0.9069767
Neg Pred Value	0.9571429	0.9857143	0.9594595	0.9571429
Precision	0.9069767	0.9534884	1.0000000	0.9069767
Recall	0.9285714	0.9761905	0.9285714	0.9285714
F1	0.9176471	0.9647059	0.9629630	0.9176471
Prevalence	0.3716814	0.3716814	0.3716814	0.3716814
Detection Rate	0.3451327	0.3628319	0.3451327	0.3451327
Detection Prevalence	0.3805310	0.3805310	0.3451327	0.3805310
Balanced Accuracy	0.9361167	0.9740107	0.9642857	0.9361167

```
# Identify the best results for each metric in confusion matrix
best_results <- apply(results, 1, which.is.max)
```

```
# Match the best results with corresponding model
report <- tibble (metric = names(best_results),
                  best_model = colnames(results)[best_results],
                  value=mapapply(function(x,y) {results[x,y]},
                                 names(best_results),
                                 best_results))
rownames(report)<-NULL
```

```
# Print the best model identified for each metric
report
```

```
## # A tibble: 11 x 3
##   metric          best_model      value
##   <chr>          <chr>          <dbl>
## 1 Sensitivity    logistic_regression 0.976
## 2 Specificity    KNN                1
## 3 Pos Pred Value KNN                1
## 4 Neg Pred Value logistic_regression 0.986
## 5 Precision      KNN                1
## 6 Recall         logistic_regression 0.976
## 7 F1            logistic_regression 0.965
## 8 Prevalence     naive_bayes        0.372
## 9 Detection Rate logistic_regression 0.363
## 10 Detection Prevalence random_forest      0.381
## 11 Balanced Accuracy logistic_regression 0.974
```

There has been discussion which metric, say accuracy, precision, recall or F1 score, we should use to select the “best model”. There is no one-size-fit-all answer. **For our case on predicting whether a breast mass cell is cancerous, undoubtedly the cost associated with false negative is high.** The consequence can be very serious for a patient where his/her cell is predicted as negative (benign) while it is actually positive (malignant). In this regard, **the sensitivity, which calculates how many of actual positives a model capture, would be more relevant in this case.** While false positives appear to cause less serious consequence in our case, it does not mean costless. A healthy person diagnosed with cancer will result in stress and high medical costs. To this end, we should consider F1 score as well, which seek a balance between sensitivity and specificity.

Based on the result table above, the **best model should be logistic regression model** which has the highest sensitivity and F1 score.

Conclusion

In this project, We have developed four machine learning models to predict classification of a breast mass cell as “benign” or “malignant”. Then, we have discussed which metric we should use to select the best model. Finally, we have selected logistic regression model as the most optimal one given its good sensitivity and F1 score. To further improve our prediction, we can in fact build more models such as neural network and support vector machine to explore if there are any better models than the one we chose.

This kind of classification prediction will have a wide use across industries, such as predicting no-shows for medical appointments, spam emails or fraudulent transactions.