UNIVERSITÀ DEGLI STUDI DI PADOVA
Dipartimento di Matematica

"Breast Cancer Prediction"
An Analysis of Classification Models

Mohammad Matin Parvanian

July 2023

Contents

1 INTRODUCTION

Breast cancer is a significant health concern that affects many individuals globally. Early detection and accurate prediction of breast cancer can greatly improve patient outcomes and survival rates. Statistical learning techniques have shown promise in predicting and diagnosing breast cancer using various clinical and imaging features.

In this paper, I aim to conduct a systematic analysis of several classification models for breast cancer prediction, including, logistic regression, LDA, QDA, Naive Bayes,and KNN, can be used to build predictive models.I evaluate the performance of these models on the available dataset, using various performance metrics and feature selection techniques. Additionally, I provide insights into the most informative features of breast cancer prediction.

2 PREPRATION OF THE DATASET

2.1 Data Collection

Breast cancer is the most common cancer amongst women in the world(https://www.kaggle.com/datasets/utkarshx27/breast-cancer-wisconsin-diagnostic-dataset). It accounts for 25% of all cancer cases, and affected over 2.1 Million people in 2015 alone. It starts when cells in the breast begin to grow out of control. These cells usually form tumors that can be seen via X-ray or felt as lumps in the breast area. The key challenges against it's detection is how to classify tumors into malignant (cancerous) or benign(non cancerous). This dataset is consist of 569 females. Features were computationally extracted from digital images of fine needle aspirate biopsy slides. Features correspond to properties of cell nuclei, such as size, shape and regularity. The mean, standard error, and worst value of each of 10 nuclear parameters is reported for a total of 30 features.

Let's import the necessary libraries.

```
options(warn = -1)
library(ggplot2, logical.return = FALSE , warn.conflicts = FALSE)
library(reshape2, logical.return = FALSE , warn.conflicts = FALSE)
library(MASS, logical.return = FALSE , warn.conflicts = FALSE)
library(e1071, logical.return = FALSE , warn.conflicts = FALSE)
library(class, logical.return = FALSE , warn.conflicts = FALSE)
library(gridExtra, logical.return = FALSE , warn.conflicts = FALSE)
library(dplyr, logical.return = FALSE , warn.conflicts = FALSE)
library(glmnet, logical.return = FALSE , warn.conflicts = FALSE)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-7
```

```
library(pROC, logical.return = FALSE , warn.conflicts = FALSE)
```

```
## Type 'citation("pROC")' for a citation.
```

```r
library(car, logical.return = FALSE , warn.conflicts = FALSE)
```

```
## Loading required package: carData
```

Let's import the dataset. You can see that we have 32 variables with 30 continuous variables and 1 output which indicates if the cancer is malignant or benign.

```r
setwd("E:\\2023-2024A\\Statistical Learning\\data")
Cancer = read.csv("breast-cancer.csv")
attach(Cancer)
str(Cancer)
```

```
## '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 ...
```

```r
dim(Cancer)
```

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

Some useful information about features:

- radius. Nucleus radius (mean of distances from center to points on perimeter).
- texture. Nucleus texture (standard deviation of grayscale values).
- perimeter. Nucleus perimeter.
- area. Nucleus area.
- smoothness. Nucleus smoothness (local variation in radius lengths).
- compactness. Nucleus compactness (perimeter^2/area - 1).
- concavity, Nucleus concavity (severity of concave portions of the contour).
- concave_pts. Number of concave portions of the nucleus contour.
- symmetry. Nucleus symmetry.
- fractal_dim. Nucleus fractal dimension ("coastline approximation" -1).

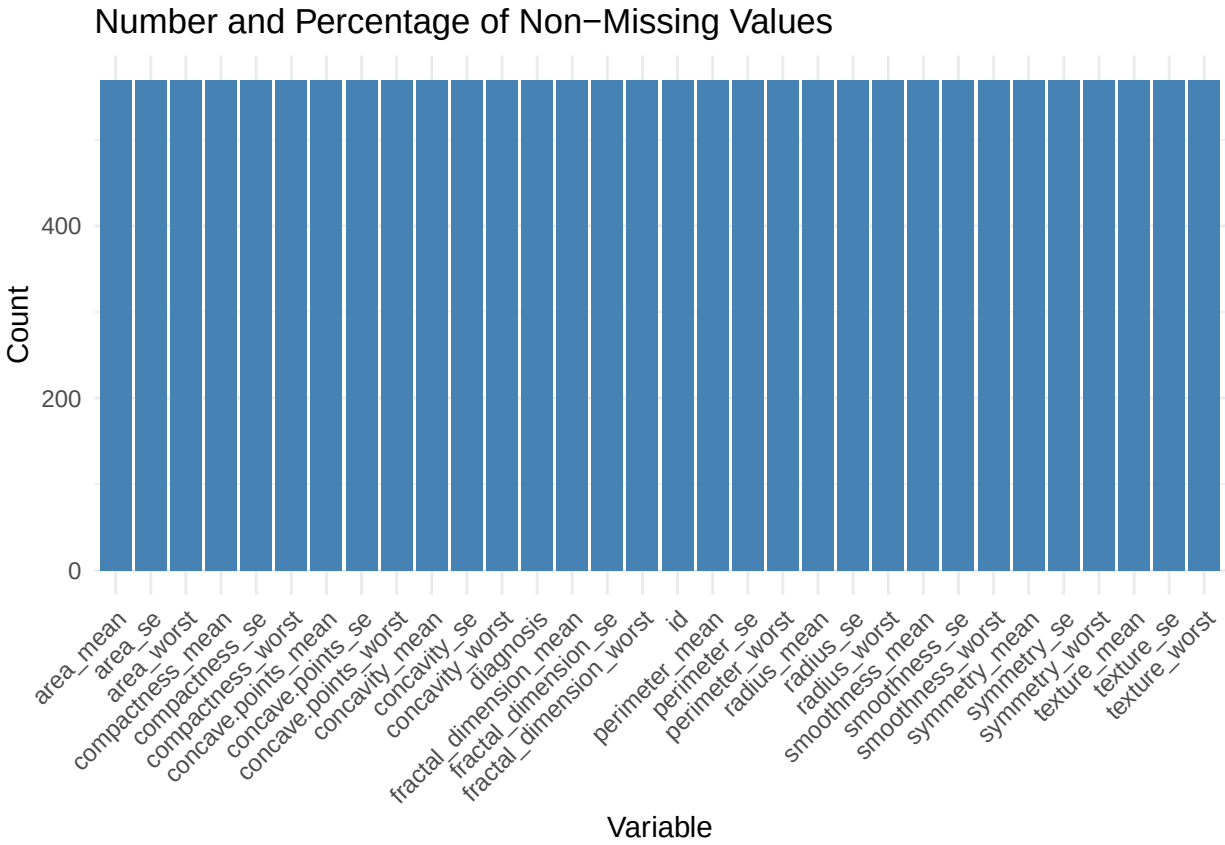
2.2 Preprocessing

There is no missing values in the dataset.

```
sum(is.na(Cancer))
```

```
## [1] 0
```

```
missing_counts = data.frame(
  Variable = names(Cancer),
  Count = colSums(!is.na(Cancer)),
  Percentage = colMeans(!is.na(Cancer)) * 100
)

ggplot(missing_counts, aes(x = Variable, y = Count)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(title = "Number and Percentage of Non-Missing Values",
       x = "Variable",
       y = "Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



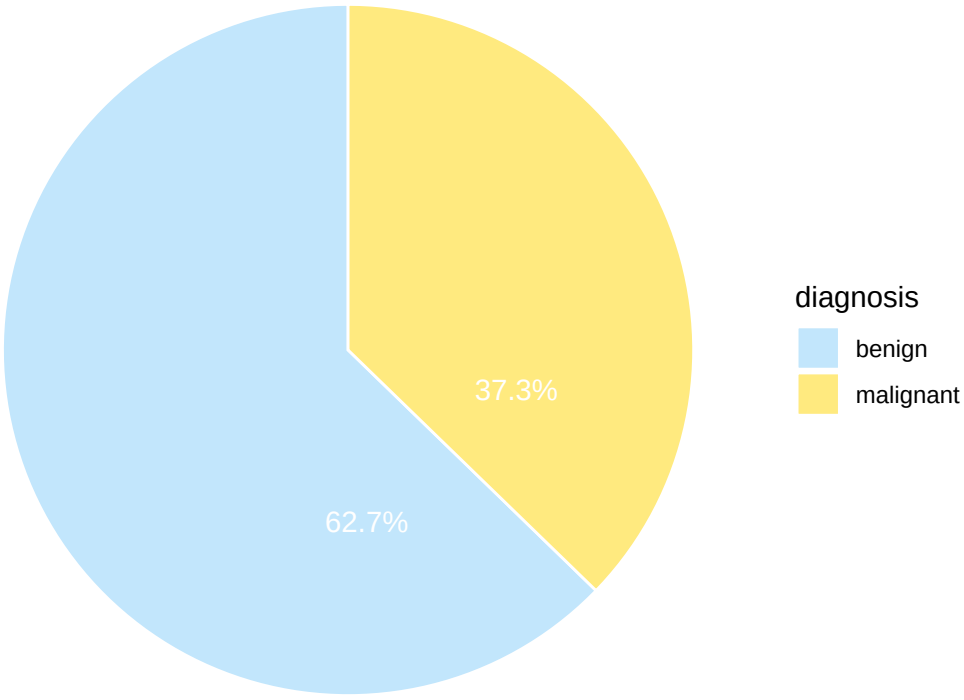
We can remove the column id since it is not necessary to be in the models and results.

```
Cancer = Cancer[-1]
```

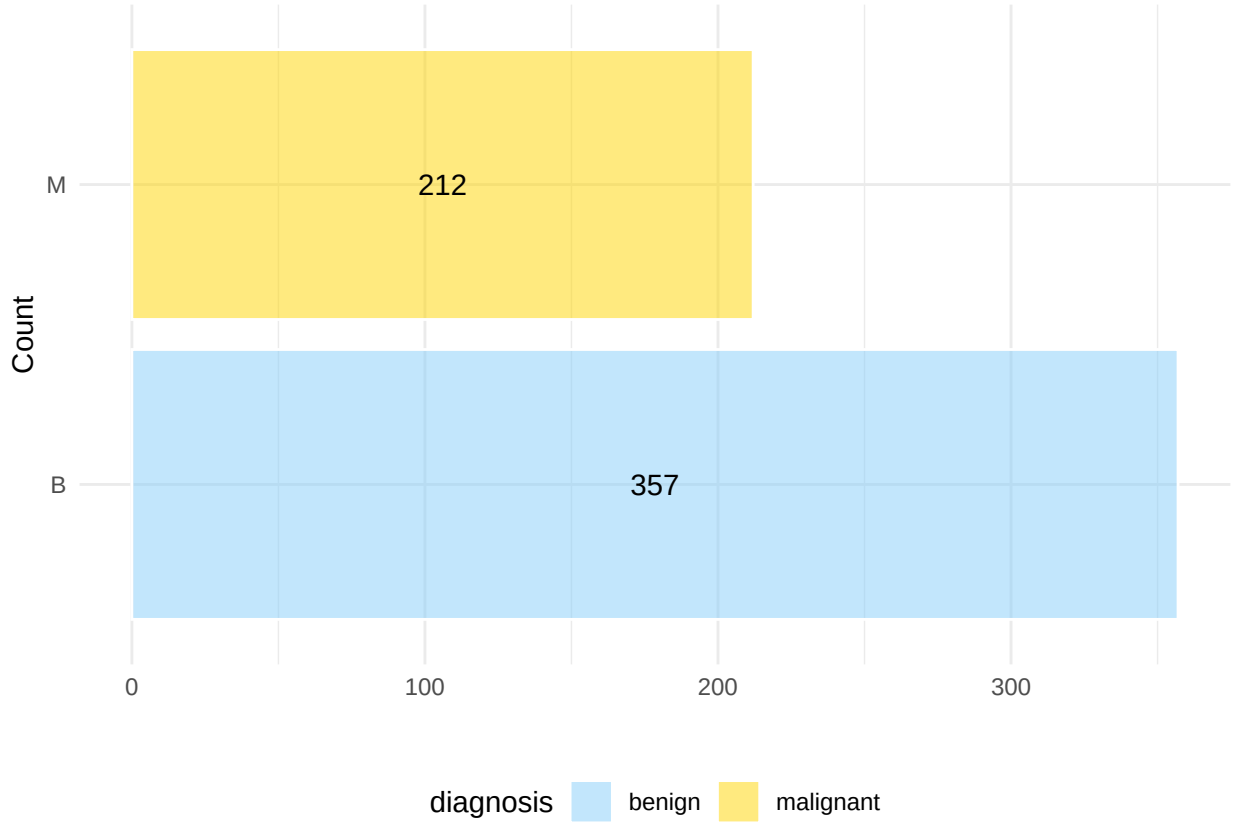
2.3 Exploratory and Data Analysis

Let's check the response variable first. Our response variable is diagnosis. As you can see the response variable is not balanced completely but since the difference among 2 classes is not too much it is not going to make any major problem.

```
Cancer %>%
  group_by(diagnosis) %>%
  summarise(n = n()) %>%
  mutate(Percentage = round(n/sum(n)*100, 1)) %>%
  ggplot(aes(x="", y=n, fill = factor(diagnosis))) +
  geom_bar(width = 1, color = "white", alpha = 0.5, stat = "identity") +
  coord_polar("y", start=0) +
  labs(fill = "diagnosis", x = "", y = "") +
  theme_void() +
  geom_text(aes(y = n/1.3, label = paste0(Percentage, "%")), color = "white", size = 4) +
  scale_fill_manual(values = c('lightskyblue', 'gold'), labels = c("benign", "malignant"))
```

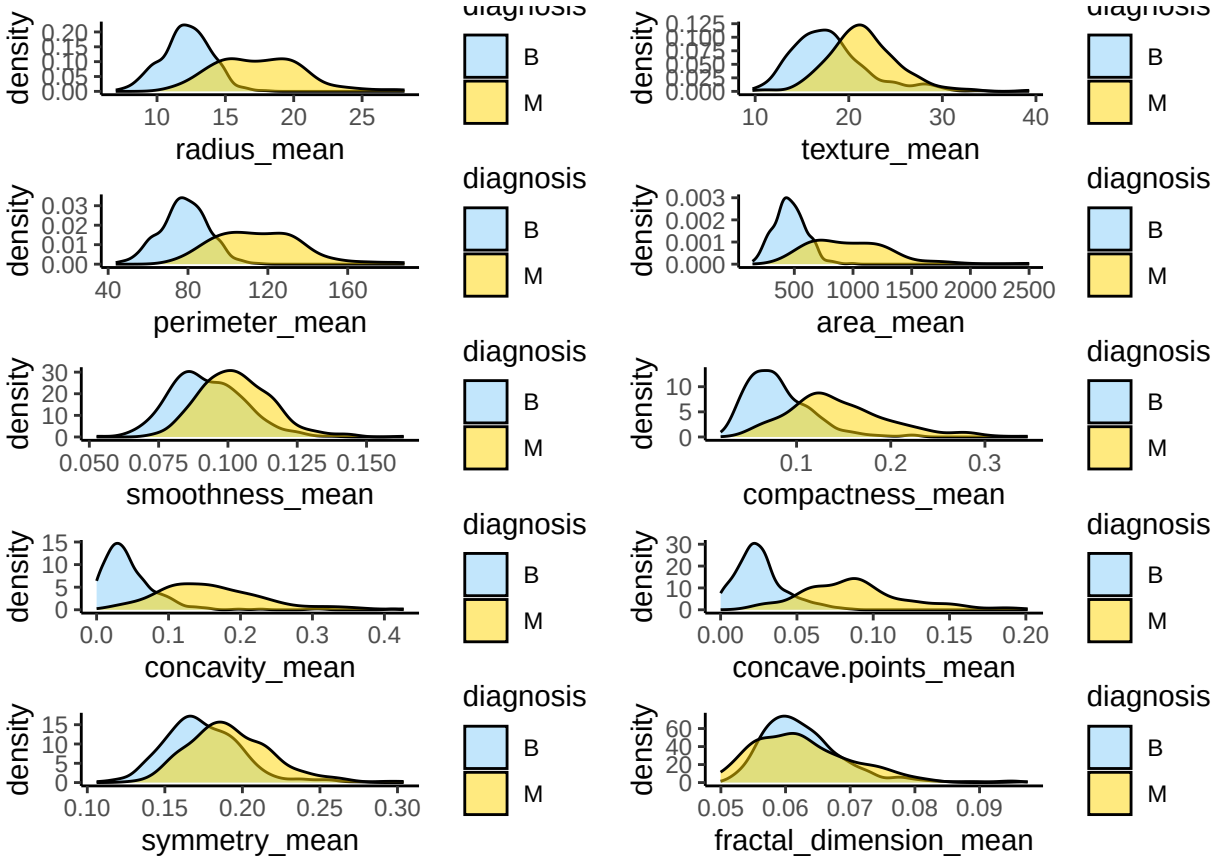



```
Cancer %>%
  group_by(diagnosis) %>%
  summarise(n = n()) %>%
  mutate(number = n) %>%
  ggplot(aes(x = factor(diagnosis), y = n, fill = factor(diagnosis))) +
  geom_bar(stat = "identity", color = "white", alpha = 0.5) +
  labs(x = "Count", y = "", fill = "diagnosis") +
  theme_minimal() +
  theme(legend.position = "bottom") +
  geom_text(aes(label = paste0(number)), position = position_stack(vjust = 0.5),
            color = "black", size = 4) +
  scale_fill_manual(values=c('lightskyblue', 'gold'),labels = c("benign", "malignant"))+
  coord_flip()
```



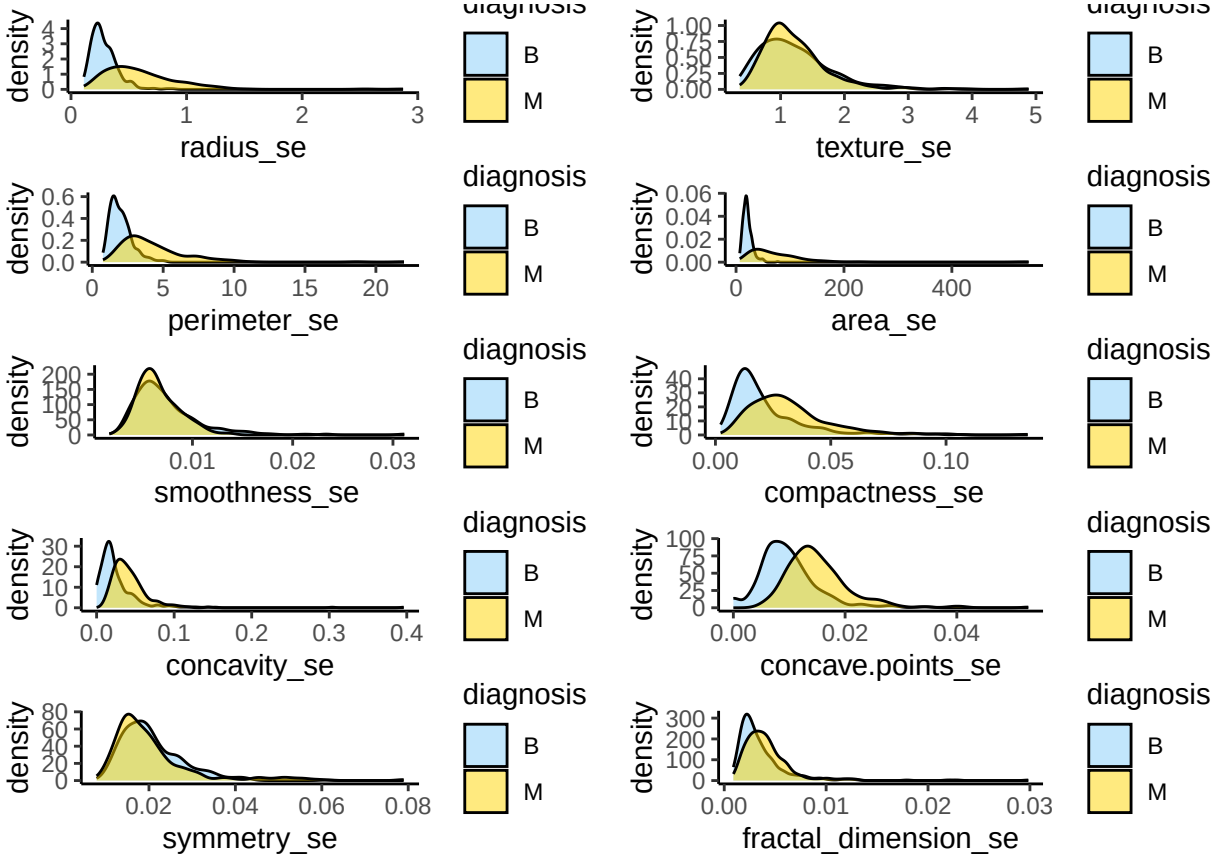
Now, let's check the plot of other features. From the plot below, you can see that for all the available features the mean of variables for malignant observes is greater than benign. so we can say that these variables are effective in being malignant or benign.

```
variables = names(Cancer)[2:11]
plots = list()
for (i in 1:length(variables)) {
  variable = variables[i]

  p = ggplot(Cancer, aes(x = .data[[variable]], fill = factor(diagnosis))) +
    geom_density(alpha = 0.5) +
    scale_fill_manual(values = c("lightskyblue", "gold"), name = "diagnosis") +
    labs(x = variable) +
    theme_classic()

  plots[[i]] = p
}
grid.arrange(grobs = plots, ncol = 2)
```



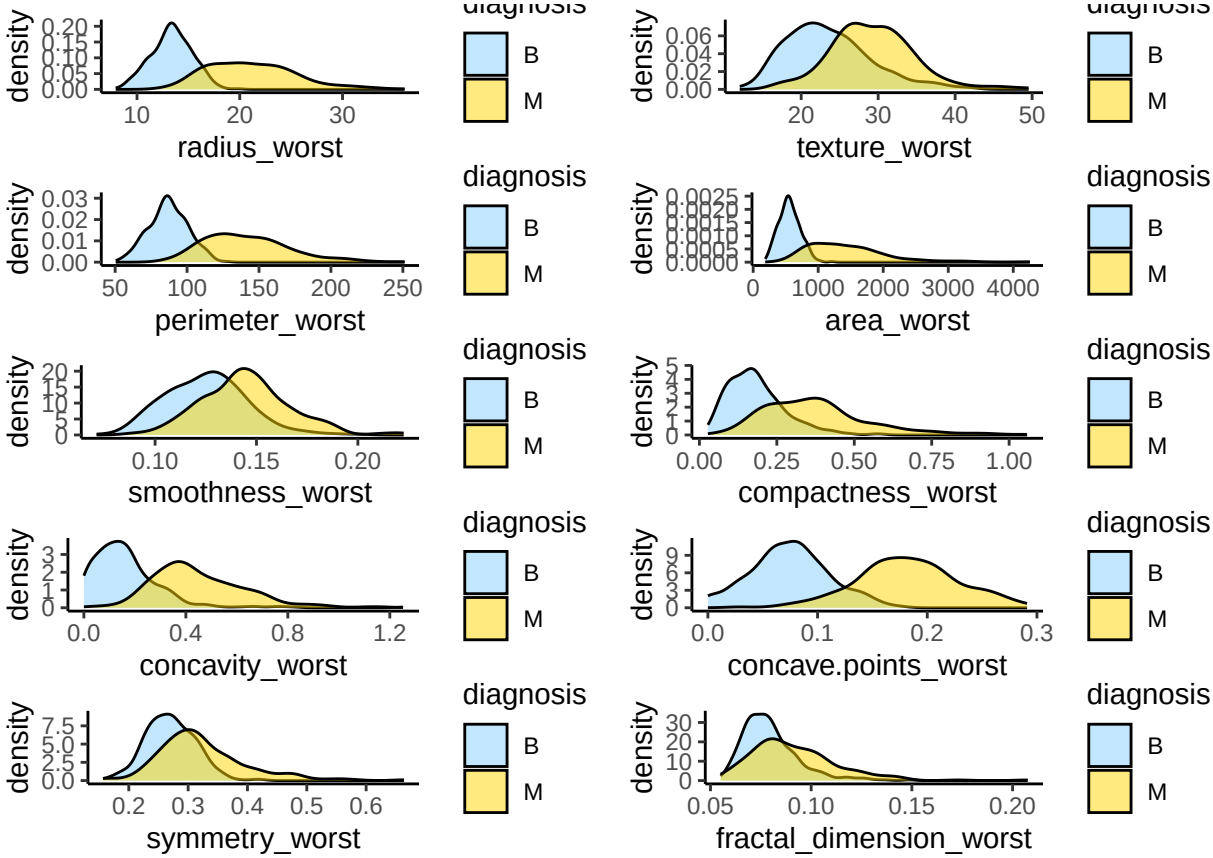
From the plot below, you can see that for most of the available features the standard error of variables for malignant observes is greater than benign except smoothness_se and symmetry_se where for smoothness_se the number of malignant cases is greater than benign, and for symmetry_se we can see that the standard error of symmetry of benign is greater than malignant cases. so we can say that these variables are effective in being malignant or benign.

```
variables = names(Cancer)[12:21]
plots = list()
for (i in 1:length(variables)) {
  variable <- variables[i]

  p = ggplot(Cancer, aes(x = .data[[variable]], fill = factor(diagnosis))) +
    geom_density(alpha = 0.5) +
    scale_fill_manual(values = c("lightskyblue", "gold"), name = "diagnosis") +
    labs(x = variable) +
    theme_classic()

  plots[[i]] = p
}
grid.arrange(grobs = plots, ncol = 2)
```



From the plot below, you can see that for all the available features the worst value of variables for malignant observes is greater than benign.

```
variables = names(Cancer)[22:31]
plots = list()
for (i in 1:length(variables)) {
  variable <- variables[i]

  p = ggplot(Cancer, aes(x = .data[[variable]], fill = factor(diagnosis))) +
    geom_density(alpha = 0.5) +
    scale_fill_manual(values = c("lightskyblue", "gold"), name = "diagnosis") +
    labs(x = variable) +
    theme_classic()

  plots[[i]] = p
}
grid.arrange(grobs = plots, ncol = 2)
```



In In the next part, you can see the boxplot of all variables with respect to the response variable. in order to have a good view I normalized all features to have the same scale. Moreover, linearly scaling the features is useful for some models especially for KNN since KNN is very sensitive to scales of features.

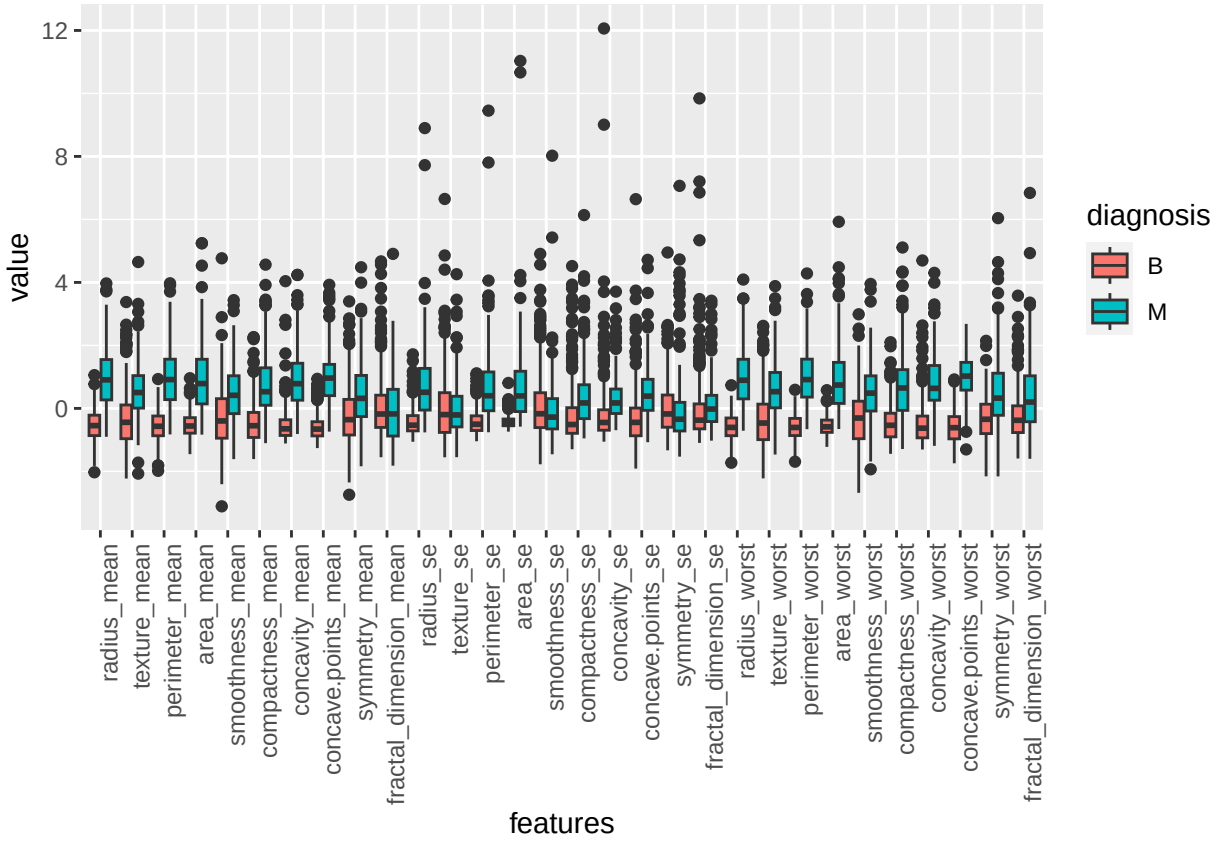
```
Cancer_normalized = Cancer
Cancer_normalized[names(Cancer)[-1]] = scale(Cancer_normalized[names(Cancer)[-1]])
head(Cancer_normalized)
```

```
##   diagnosis radius_mean texture_mean perimeter_mean  area_mean smoothness_mean
## 1         M   1.0960995   -2.0715123      1.2688173  0.9835095       1.5670875
## 2         M   1.8282120   -0.3533215      1.6844726  1.9070303      -0.8262354
## 3         M   1.5784992    0.4557859      1.5651260  1.5575132       0.9413821
## 4         M  -0.7682333    0.2535091     -0.5921661 -0.7637917       3.2806668
## 5         M   1.7487579   -1.1508038      1.7750113  1.8246238       0.2801253
## 6         M  -0.4759559   -0.8346009     -0.3868077 -0.5052059       2.2354545
##   compactness_mean concavity_mean concave.points_mean symmetry_mean
## 1        3.2806281     2.65054179           2.5302489   2.215565542
## 2       -0.4866435    -0.02382489           0.5476623   0.001391139
## 3        1.0519999     1.36227979           2.0354398   0.938858720
## 4        3.3999174     1.91421287           1.4504311   2.864862154
## 5        0.5388663     1.36980615           1.4272370  -0.009552062
## 6        1.2432416     0.86554001           0.8239307   1.004517928
##   fractal_dimension_mean  radius_se texture_se perimeter_se    area_se
## 1              2.2537638  2.4875451 -0.5647681    2.8305403  2.4853907
## 2             -0.8678888  0.4988157 -0.8754733    0.2630955  0.7417493
## 3             -0.3976580  1.2275958 -0.7793976    0.8501802  1.1802975
```

```
## 4              4.9066020  0.3260865 -0.1103120    0.2863415 -0.2881246
## 5             -0.5619555  1.2694258 -0.7895490    1.2720701  1.1893103
## 6              1.8883435 -0.2548461 -0.5921406   -0.3210217 -0.2890039
##   smoothness_se compactness_se concavity_se concave.points_se symmetry_se
## 1    -0.2138135     1.31570389    0.7233897        0.66023900   1.1477468
## 2    -0.6048187    -0.69231710   -0.4403926        0.25993335  -0.8047423
## 3    -0.2967439     0.81425704    0.2128891        1.42357487   0.2368272
## 4     0.6890953     2.74186785    0.8187979        1.11402678   4.7285198
## 5     1.4817634    -0.04847723    0.8277425        1.14319885  -0.3607748
## 6     0.1562093     0.44515196    0.1598845       -0.06906279   0.1340009
##   fractal_dimension_se radius_worst texture_worst perimeter_worst area_worst
## 1           0.90628565    1.8850310   -1.35809849       2.3015755  1.9994782
## 2          -0.09935632    1.8043398   -0.36887865       1.5337764  1.8888270
## 3           0.29330133    1.5105411   -0.02395331       1.3462906  1.4550043
## 4           2.04571087   -0.2812170    0.13386631      -0.2497196 -0.5495377
## 5           0.49888916    1.2974336   -1.46548091       1.3373627  1.2196511
## 6           0.48641784   -0.1653528   -0.31356043      -0.1149083 -0.2441054
##   smoothness_worst compactness_worst concavity_worst concave.points_worst
## 1        1.3065367         2.6143647       2.1076718            2.2940576
## 2       -0.3752817        -0.4300658      -0.1466200            1.0861286
## 3        0.5269438         1.0819801       0.8542223            1.9532817
## 4        3.3912907         3.8899747       1.9878392            2.1738732
## 5        0.2203623        -0.3131190       0.6126397            0.7286181
## 6        2.0467119         1.7201029       1.2621327            0.9050914
##   symmetry_worst fractal_dimension_worst
## 1      2.7482041               1.9353117
## 2     -0.2436753               0.2809428
## 3      1.1512420               0.2012142
## 4      6.0407261               4.9306719
## 5     -0.8675896              -0.3967505
## 6      1.7525273               2.2398308
```

From the plot below, you can see that the results from previous plots are still extractable from this plot. for most of the variables, the value of malignant cases is greater than benign. there exist some outliers but I don't take any action for them since I fitted my models with outliers and without these outliers and the results are better when I don't remove them or use some strategies like IQR.

```
data_melted = melt(Cancer_normalized, id.vars = "diagnosis",
                   variable.name = "features", value.name = "value")

# Create the boxplot
ggplot(data_melted, aes(x = features, y = value, fill = diagnosis)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(x = "features", y = "value") +
  guides(fill = guide_legend(title = "diagnosis"))
```



Now, since our response variable is not numeric I need to change it to a numeric variable. You can see that the diagnosis variable classes are changed to 0 and 1.

```
Cancer$diagnosis[Cancer$diagnosis=='M']=1
Cancer$diagnosis[Cancer$diagnosis=='B']=0
attach(Cancer)
```

```
## The following objects are masked from Cancer (pos = 3):
##
##     area_mean, area_se, area_worst, compactness_mean, compactness_se,
##     compactness_worst, concave.points_mean, concave.points_se,
##     concave.points_worst, concavity_mean, concavity_se,
##     concavity_worst, diagnosis, fractal_dimension_mean,
##     fractal_dimension_se, fractal_dimension_worst, perimeter_mean,
##     perimeter_se, perimeter_worst, radius_mean, radius_se,
##     radius_worst, smoothness_mean, smoothness_se, smoothness_worst,
##     symmetry_mean, symmetry_se, symmetry_worst, texture_mean,
##     texture_se, texture_worst
```

```
table(diagnosis)
```

```
## diagnosis
##   0   1
## 357 212
```

3 MODELS

In this section, we will fit our models to the breast cancer dataset. As mentioned earlier, we will consider five different algorithms: Logistic Regression, K-Nearest Neighbors (KNN), Naive Bayes, Linear Discriminant Analysis (LDA), and Quadratic Discriminant Analysis (QDA). For each model, we will explore different scenarios by applying feature selection methods. Our goal is to identify the best model that is most suitable for our dataset.

Before going through the details of our models there exist some important points that we need to discuss.

Firstly, as we are dealing with a classification problem, each model is accompanied by a confusion matrix. The confusion matrix consists of four different values. Firstly, we have the true positive, which represents the number of instances correctly predicted as having cancer. Secondly, we have the false positive, which indicates the number of instances predicted as having cancer but actually being non cancerous. The third value is the false negative, which signifies the instances that have cancer but are predicted as non cancerous. Lastly, we have the true negative, which represents the number of instances correctly predicted as non cancerous.

Secondly, while it is important to prioritize higher true positives, higher true negatives, and lower false positives, the most crucial aspect for us is minimizing the false negative rate. You may wonder why. In the context of medical diagnosis, a false negative implies that the model fails to identify individuals who actually have cancer. The consequences of false negatives can be severe, as it could result in delayed or missed treatment, leading to potential health risks and complications for individuals requiring medical attention. Missing cases of cancer can have long-term health implications.

Regarding false positives, while they can lead to unnecessary follow-up tests or treatments, they are generally less severe compared to false negatives. In the case of cancer, a false positive may result in additional medical evaluations or interventions, but it typically does not pose the same immediate health risks as missing a true positive case. However, false positives can still cause anxiety, inconvenience, and potential economic costs associated with unnecessary medical procedures or treatments.

Considering the potential health risks and consequences associated with missing cases of cancer, minimizing false negatives (FN) is typically of higher importance in this situation. The primary goal is to ensure that individuals with cancer are correctly identified and receive the necessary care and management.

All in all, the relative importance of false negatives and false positives may vary based on specific circumstances, such as the prevalence of cancer in the population, the availability of follow-up confirmatory tests, the cost of those tests, and the potential impact of false positives on individuals' well-being. It is important to assess the specific context and consider the trade-offs between false negatives and false positives to determine the optimal approach for diabetes classification.

3.1 Logistic Regression

Let's fit our Logistic Regression model with all of our independent variables. We considered 80% of our data as the training dataset, and the remaining data as the test dataset. I considered 4 different thresholds 0.3, 0.4, 0.5, 0.6 but the result is the same for all the available metrics. In order to prevent overlapping I defined a function that contains information about the metrics and confusion matrix. The logistic regression has obtained good results.

```
Cancer$diagnosis = as.numeric(as.character(Cancer$diagnosis))
set.seed(123)
test_index = sample(nrow(Cancer), 0.2 * nrow(Cancer))
train = Cancer[-test_index, ]
test = Cancer[test_index, ]
```

```r
model = glm(diagnosis ~ ., data = train, family = "binomial")


calculate_metrics = function(predictions, actual) {
  confusion_matrix = table(predictions, actual)
  true_positive = confusion_matrix[2, 2]
  false_positive = confusion_matrix[1, 2]
  false_negative = confusion_matrix[2, 1]
  true_negative = confusion_matrix[1, 1]

  recall = true_positive / (true_positive + false_negative)
  precision = true_positive / (true_positive + false_positive)
  f1_score = 2 * precision * recall / (precision + recall)
  accuracy = (true_positive + true_negative) / sum(confusion_matrix)

  return(data.frame(
    Metric = c("Recall", "Precision", "F1 Score", "Accuracy"),
    Value = c(recall, precision, f1_score, accuracy)
  ))
}

thresholds = c(0.3, 0.4, 0.5, 0.6)
results_train = data.frame()
results_test = data.frame()

for (threshold in thresholds) {

  pred_class_train = ifelse(predict(model, newdata = train, type = "response") >
                              threshold, 1, 0)
  results_train_threshold = calculate_metrics(pred_class_train, train$diagnosis)
  results_train_threshold$Set = "Train"
  results_train_threshold$Threshold = threshold
  results_train = rbind(results_train, results_train_threshold)


  pred_class_test = ifelse(predict(model, newdata = test, type = "response") >
                             threshold, 1, 0)
  results_test_threshold = calculate_metrics(pred_class_test, test$diagnosis)
  results_test_threshold$Set = "Test"
  results_test_threshold$Threshold = threshold
  results_test = rbind(results_test, results_test_threshold)
}


print(results_train)
```

```
##       Metric Value   Set Threshold
## 1     Recall     1 Train       0.3
## 2  Precision     1 Train       0.3
## 3   F1 Score     1 Train       0.3
## 4   Accuracy     1 Train       0.3
## 5     Recall     1 Train       0.4
## 6  Precision     1 Train       0.4
```

```
## 7   F1 Score     1 Train       0.4
## 8   Accuracy     1 Train       0.4
## 9     Recall     1 Train       0.5
## 10 Precision     1 Train       0.5
## 11  F1 Score     1 Train       0.5
## 12  Accuracy     1 Train       0.5
## 13    Recall     1 Train       0.6
## 14 Precision     1 Train       0.6
## 15  F1 Score     1 Train       0.6
## 16  Accuracy     1 Train       0.6
```

```
print(results_test)
```

```
##       Metric     Value  Set Threshold
## 1     Recall 0.8125000 Test       0.3
## 2  Precision 0.7878788 Test       0.3
## 3   F1 Score 0.8000000 Test       0.3
## 4   Accuracy 0.8849558 Test       0.3
## 5     Recall 0.8125000 Test       0.4
## 6  Precision 0.7878788 Test       0.4
## 7   F1 Score 0.8000000 Test       0.4
## 8   Accuracy 0.8849558 Test       0.4
## 9     Recall 0.8125000 Test       0.5
## 10 Precision 0.7878788 Test       0.5
## 11  F1 Score 0.8000000 Test       0.5
## 12  Accuracy 0.8849558 Test       0.5
## 13    Recall 0.8125000 Test       0.6
## 14 Precision 0.7878788 Test       0.6
## 15  F1 Score 0.8000000 Test       0.6
## 16  Accuracy 0.8849558 Test       0.6
```

you can see the confusion matrix of the logistic regression. As you can see among all the cases in the test set we have 6 cases recognized as false negatives which are the ones that had malignant breast cancer but the model was unable to predict them correctly.

```
table(pred_class_test, test$diagnosis)
```

```
##
## pred_class_test  0  1
##               0 74  7
##               1  6 26
```

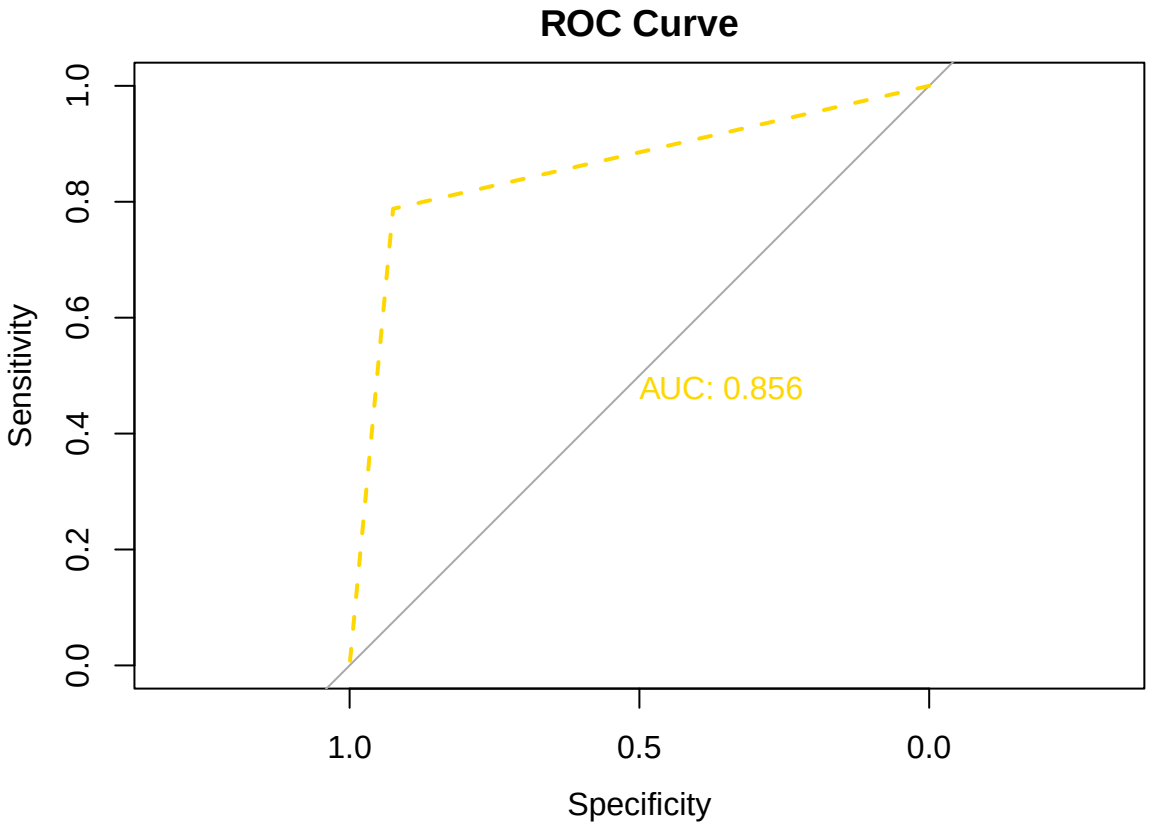
Below, you can see the ROC curve related to logistic regression which has a good value of AUC. The AUC values indicate the overall performance of each model in terms of their ability to discriminate between positive and negative instances. A higher AUC suggests better predictive performance and a greater ability to correctly classify instances. Therefore, based on the AUC values, the logistic regression model appears to have the best performance among the three models in terms of its ability to distinguish between the classes.

```
roc_obj = roc(test$diagnosis,pred_class_test)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(roc_obj, main = "ROC Curve", print.auc = TRUE,lty = 2 , col = 'gold')
```



3.2 Backward Feature selection

In this section, I applied backward feature selection in order to find the best features which can make better predictions.As you can see the backward feature selection has choosed fractal_dimension_mean, texture_se, symmetry_worst, radius_se, compactness_mean, smoothness_se, perimeter_worst, smoothness_worst, concave.points_mean, and texture_mean as preferred features.

```
backward_model = step(model, direction = "backward")
```

```
## Start:  AIC=62
## diagnosis ~ radius_mean + texture_mean + perimeter_mean + area_mean +
##     smoothness_mean + compactness_mean + concavity_mean + concave.points_mean +
##     symmetry_mean + fractal_dimension_mean + radius_se + texture_se +
##     perimeter_se + area_se + smoothness_se + compactness_se +
##     concavity_se + concave.points_se + symmetry_se + fractal_dimension_se +
##     radius_worst + texture_worst + perimeter_worst + area_worst +
##     smoothness_worst + compactness_worst + concavity_worst +
##     concave.points_worst + symmetry_worst + fractal_dimension_worst
##
```

```
##                           Df   Deviance AIC
## - symmetry_mean            1 1.5243e-07  60
## - concavity_se             1 1.5282e-07  60
## - symmetry_se              1 1.5284e-07  60
## - area_mean                1 1.5291e-07  60
## - compactness_se           1 1.5446e-07  60
## - texture_worst            1 1.5584e-07  60
## - fractal_dimension_se     1 1.5597e-07  60
## - texture_se               1 1.5667e-07  60
## - fractal_dimension_worst  1 1.5681e-07  60
## - compactness_worst        1 1.5704e-07  60
## - symmetry_worst           1 1.5751e-07  60
## - area_se                  1 1.5753e-07  60
## - concave.points_worst     1 1.5823e-07  60
## - concavity_mean           1 1.5850e-07  60
## - perimeter_mean           1 1.5939e-07  60
## - area_worst               1 1.6031e-07  60
## - radius_mean              1 1.6048e-07  60
## - smoothness_mean          1 1.6085e-07  60
## - concave.points_se        1 1.6662e-07  60
## - texture_mean             1 1.6758e-07  60
## - fractal_dimension_mean   1 1.7589e-07  60
## - compactness_mean         1 1.7641e-07  60
## - concavity_worst          1 1.7848e-07  60
## - perimeter_worst          1 1.8001e-07  60
## - perimeter_se             1 1.8823e-07  60
## - smoothness_worst         1 2.0176e-07  60
## - radius_worst             1 2.0894e-07  60
## - smoothness_se            1 2.1019e-07  60
## - concave.points_mean      1 2.1261e-07  60
## - radius_se                1 2.8075e-07  60
## <none>                       1.5166e-07  62
##
## Step:  AIC=60
## diagnosis ~ radius_mean + texture_mean + perimeter_mean + area_mean +
##     smoothness_mean + compactness_mean + concavity_mean + concave.points_mean +
##     fractal_dimension_mean + radius_se + texture_se + perimeter_se +
##     area_se + smoothness_se + compactness_se + concavity_se +
##     concave.points_se + symmetry_se + fractal_dimension_se +
##     radius_worst + texture_worst + perimeter_worst + area_worst +
##     smoothness_worst + compactness_worst + concavity_worst +
##     concave.points_worst + symmetry_worst + fractal_dimension_worst
##
##                           Df   Deviance AIC
## - symmetry_se              1 1.5320e-07  58
## - concavity_se             1 1.5379e-07  58
## - area_mean                1 1.5418e-07  58
## - compactness_se           1 1.5572e-07  58
## - texture_worst            1 1.5582e-07  58
## - fractal_dimension_worst  1 1.5707e-07  58
## - concave.points_worst     1 1.5735e-07  58
## - fractal_dimension_se     1 1.5757e-07  58
## - compactness_worst        1 1.5794e-07  58
## - concavity_mean           1 1.5861e-07  58
```

```
## - texture_se               1 1.5886e-07  58
## - area_se                  1 1.5940e-07  58
## - smoothness_mean          1 1.6153e-07  58
## - radius_mean              1 1.6325e-07  58
## - perimeter_mean           1 1.6473e-07  58
## - area_worst               1 1.6824e-07  58
## - texture_mean             1 1.6976e-07  58
## - concave.points_se        1 1.7098e-07  58
## - symmetry_worst           1 1.7374e-07  58
## - concavity_worst          1 1.7663e-07  58
## - compactness_mean         1 1.7675e-07  58
## - fractal_dimension_mean   1 1.7756e-07  58
## - perimeter_worst          1 1.8545e-07  58
## - perimeter_se             1 1.9037e-07  58
## - radius_worst             1 2.2161e-07  58
## - concave.points_mean      1 2.2229e-07  58
## - smoothness_worst         1 2.2923e-07  58
## - smoothness_se            1 2.3249e-07  58
## - radius_se                1 2.8946e-07  58
## <none>                       1.5243e-07  60
##
## Step:  AIC=58
## diagnosis ~ radius_mean + texture_mean + perimeter_mean + area_mean +
##     smoothness_mean + compactness_mean + concavity_mean + concave.points_mean +
##     fractal_dimension_mean + radius_se + texture_se + perimeter_se +
##     area_se + smoothness_se + compactness_se + concavity_se +
##     concave.points_se + fractal_dimension_se + radius_worst +
##     texture_worst + perimeter_worst + area_worst + smoothness_worst +
##     compactness_worst + concavity_worst + concave.points_worst +
##     symmetry_worst + fractal_dimension_worst
##
##                           Df   Deviance AIC
## - concavity_se             1 1.5450e-07  56
## - texture_worst            1 1.5645e-07  56
## - compactness_worst        1 1.5859e-07  56
## - compactness_se           1 1.5902e-07  56
## - area_se                  1 1.5928e-07  56
## - concavity_mean           1 1.6017e-07  56
## - concave.points_worst     1 1.6045e-07  56
## - fractal_dimension_se     1 1.6112e-07  56
## - area_mean                1 1.6268e-07  56
## - smoothness_mean          1 1.6422e-07  56
## - fractal_dimension_worst  1 1.6539e-07  56
## - perimeter_mean           1 1.6551e-07  56
## - texture_se               1 1.6841e-07  56
## - area_worst               1 1.6885e-07  56
## - texture_mean             1 1.7289e-07  56
## - radius_mean              1 1.7554e-07  56
## - concavity_worst          1 1.7819e-07  56
## - fractal_dimension_mean   1 1.7887e-07  56
## - compactness_mean         1 1.8065e-07  56
## - concave.points_se        1 1.8667e-07  56
## - perimeter_se             1 1.9567e-07  56
## - perimeter_worst          1 2.0359e-07  56
```

```
## - radius_worst             1 2.2131e-07  56
## - symmetry_worst           1 2.2486e-07  56
## - concave.points_mean      1 2.2601e-07  56
## - smoothness_worst         1 2.4757e-07  56
## - radius_se                1 2.8504e-07  56
## - smoothness_se            1 3.0567e-07  56
## <none>                       1.5320e-07  58
##
## Step:  AIC=56
## diagnosis ~ radius_mean + texture_mean + perimeter_mean + area_mean +
##     smoothness_mean + compactness_mean + concavity_mean + concave.points_mean +
##     fractal_dimension_mean + radius_se + texture_se + perimeter_se +
##     area_se + smoothness_se + compactness_se + concave.points_se +
##     fractal_dimension_se + radius_worst + texture_worst + perimeter_worst +
##     area_worst + smoothness_worst + compactness_worst + concavity_worst +
##     concave.points_worst + symmetry_worst + fractal_dimension_worst
##
##                           Df   Deviance AIC
## - area_mean                1 1.5651e-07  54
## - texture_worst            1 1.5857e-07  54
## - area_se                  1 1.6047e-07  54
## - concave.points_worst     1 1.6052e-07  54
## - concavity_mean           1 1.6091e-07  54
## - compactness_se           1 1.6109e-07  54
## - fractal_dimension_se     1 1.6385e-07  54
## - fractal_dimension_worst  1 1.6609e-07  54
## - perimeter_mean           1 1.6614e-07  54
## - smoothness_mean          1 1.6725e-07  54
## - compactness_worst        1 1.6770e-07  54
## - area_worst               1 1.6810e-07  54
## - texture_se               1 1.6953e-07  54
## - texture_mean             1 1.7387e-07  54
## - radius_mean              1 1.7586e-07  54
## - fractal_dimension_mean   1 1.8191e-07  54
## - compactness_mean         1 1.8651e-07  54
## - concave.points_se        1 1.8777e-07  54
## - perimeter_se             1 2.0258e-07  54
## - perimeter_worst          1 2.1357e-07  54
## - concavity_worst          1 2.3040e-07  54
## - symmetry_worst           1 2.3275e-07  54
## - radius_worst             1 2.3929e-07  54
## - concave.points_mean      1 2.5651e-07  54
## - smoothness_worst         1 2.6997e-07  54
## - smoothness_se            1 3.1464e-07  54
## - radius_se                1 3.1735e-07  54
## <none>                       1.5450e-07  56
##
## Step:  AIC=54
## diagnosis ~ radius_mean + texture_mean + perimeter_mean + smoothness_mean +
##     compactness_mean + concavity_mean + concave.points_mean +
##     fractal_dimension_mean + radius_se + texture_se + perimeter_se +
##     area_se + smoothness_se + compactness_se + concave.points_se +
##     fractal_dimension_se + radius_worst + texture_worst + perimeter_worst +
##     area_worst + smoothness_worst + compactness_worst + concavity_worst +
```

```
##     concave.points_worst + symmetry_worst + fractal_dimension_worst
##
##                           Df   Deviance AIC
## - texture_worst            1 1.5813e-07  52
## - area_se                  1 1.6021e-07  52
## - compactness_se           1 1.6147e-07  52
## - concavity_mean           1 1.6343e-07  52
## - fractal_dimension_se     1 1.6458e-07  52
## - fractal_dimension_worst  1 1.6819e-07  52
## - concave.points_worst     1 1.6820e-07  52
## - smoothness_mean          1 1.6835e-07  52
## - texture_se               1 1.7122e-07  52
## - compactness_worst        1 1.7247e-07  52
## - area_worst               1 1.7430e-07  52
## - texture_mean             1 1.7435e-07  52
## - perimeter_mean           1 1.7460e-07  52
## - radius_mean              1 1.7756e-07  52
## - fractal_dimension_mean   1 1.8662e-07  52
## - compactness_mean         1 1.8803e-07  52
## - concave.points_se        1 1.9201e-07  52
## - perimeter_worst          1 2.1845e-07  52
## - perimeter_se             1 2.2117e-07  52
## - concavity_worst          1 2.3487e-07  52
## - radius_worst             1 2.5602e-07  52
## - concave.points_mean      1 2.5640e-07  52
## - symmetry_worst           1 2.6145e-07  52
## - smoothness_worst         1 3.0528e-07  52
## - radius_se                1 3.2958e-07  52
## - smoothness_se            1 3.4591e-07  52
## <none>                       1.5651e-07  54
##
## Step:  AIC=52
## diagnosis ~ radius_mean + texture_mean + perimeter_mean + smoothness_mean +
##     compactness_mean + concavity_mean + concave.points_mean +
##     fractal_dimension_mean + radius_se + texture_se + perimeter_se +
##     area_se + smoothness_se + compactness_se + concave.points_se +
##     fractal_dimension_se + radius_worst + perimeter_worst + area_worst +
##     smoothness_worst + compactness_worst + concavity_worst +
##     concave.points_worst + symmetry_worst + fractal_dimension_worst
##
##                           Df   Deviance AIC
## - area_se                  1 1.6446e-07  50
## - fractal_dimension_se     1 1.6590e-07  50
## - concavity_mean           1 1.6728e-07  50
## - concave.points_worst     1 1.6828e-07  50
## - compactness_se           1 1.7104e-07  50
## - fractal_dimension_worst  1 1.7313e-07  50
## - compactness_worst        1 1.7400e-07  50
## - perimeter_mean           1 1.7842e-07  50
## - radius_mean              1 1.8110e-07  50
## - area_worst               1 1.8147e-07  50
## - smoothness_mean          1 1.8619e-07  50
## - fractal_dimension_mean   1 1.9017e-07  50
## - compactness_mean         1 1.9178e-07  50
```

```
## - concave.points_se        1 2.0119e-07  50
## - perimeter_worst          1 2.1639e-07  50
## - perimeter_se             1 2.3539e-07  50
## - concavity_worst          1 2.4240e-07  50
## - radius_worst             1 2.6358e-07  50
## - smoothness_worst         1 2.9118e-07  50
## - symmetry_worst           1 3.0899e-07  50
## - texture_mean             1 3.1894e-07  50
## - radius_se                1 3.3204e-07  50
## - texture_se               1 3.5391e-07  50
## - concave.points_mean      1 4.6658e-07  50
## - smoothness_se            1 6.0706e-07  50
## <none>                       1.5813e-07  52
##
## Step:  AIC=50
## diagnosis ~ radius_mean + texture_mean + perimeter_mean + smoothness_mean +
##     compactness_mean + concavity_mean + concave.points_mean +
##     fractal_dimension_mean + radius_se + texture_se + perimeter_se +
##     smoothness_se + compactness_se + concave.points_se + fractal_dimension_se +
##     radius_worst + perimeter_worst + area_worst + smoothness_worst +
##     compactness_worst + concavity_worst + concave.points_worst +
##     symmetry_worst + fractal_dimension_worst
##
##                           Df   Deviance AIC
## - concavity_mean           1 1.6830e-07  48
## - compactness_se           1 1.7094e-07  48
## - fractal_dimension_se     1 1.7392e-07  48
## - compactness_worst        1 1.7432e-07  48
## - concave.points_worst     1 1.7461e-07  48
## - fractal_dimension_worst  1 1.7738e-07  48
## - perimeter_mean           1 1.8152e-07  48
## - smoothness_mean          1 1.8308e-07  48
## - radius_mean              1 1.8356e-07  48
## - area_worst               1 1.9643e-07  48
## - fractal_dimension_mean   1 1.9994e-07  48
## - concave.points_se        1 2.0701e-07  48
## - compactness_mean         1 2.2537e-07  48
## - perimeter_worst          1 2.3065e-07  48
## - concavity_worst          1 2.4703e-07  48
## - radius_worst             1 2.6037e-07  48
## - perimeter_se             1 2.8250e-07  48
## - smoothness_worst         1 2.9135e-07  48
## - texture_mean             1 3.2042e-07  48
## - symmetry_worst           1 3.2379e-07  48
## - texture_se               1 3.5914e-07  48
## - radius_se                1 3.6474e-07  48
## - concave.points_mean      1 4.4602e-07  48
## - smoothness_se            1 6.0506e-07  48
## <none>                       1.6446e-07  50
##
## Step:  AIC=48
## diagnosis ~ radius_mean + texture_mean + perimeter_mean + smoothness_mean +
##     compactness_mean + concave.points_mean + fractal_dimension_mean +
##     radius_se + texture_se + perimeter_se + smoothness_se + compactness_se +
```

```
##     concave.points_se + fractal_dimension_se + radius_worst +
##     perimeter_worst + area_worst + smoothness_worst + compactness_worst +
##     concavity_worst + concave.points_worst + symmetry_worst +
##     fractal_dimension_worst
##
##                           Df   Deviance AIC
## - compactness_se           1 1.7320e-07  46
## - fractal_dimension_se     1 1.7526e-07  46
## - compactness_worst        1 1.7565e-07  46
## - fractal_dimension_worst  1 1.7803e-07  46
## - concave.points_worst     1 1.7840e-07  46
## - smoothness_mean          1 1.8379e-07  46
## - perimeter_mean           1 1.9541e-07  46
## - radius_mean              1 1.9857e-07  46
## - fractal_dimension_mean   1 1.9947e-07  46
## - area_worst               1 2.0855e-07  46
## - concave.points_se        1 2.1269e-07  46
## - compactness_mean         1 2.3293e-07  46
## - perimeter_worst          1 2.3329e-07  46
## - radius_worst             1 2.6200e-07  46
## - perimeter_se             1 2.8213e-07  46
## - concavity_worst          1 3.2074e-07  46
## - symmetry_worst           1 3.2337e-07  46
## - smoothness_worst         1 3.2722e-07  46
## - texture_mean             1 3.4214e-07  46
## - radius_se                1 3.6816e-07  46
## - texture_se               1 3.8369e-07  46
## - smoothness_se            1 9.1839e-07  46
## - concave.points_mean      1 1.0135e-06  46
## <none>                       1.6830e-07  48
##
## Step:  AIC=46
## diagnosis ~ radius_mean + texture_mean + perimeter_mean + smoothness_mean +
##     compactness_mean + concave.points_mean + fractal_dimension_mean +
##     radius_se + texture_se + perimeter_se + smoothness_se + concave.points_se +
##     fractal_dimension_se + radius_worst + perimeter_worst + area_worst +
##     smoothness_worst + compactness_worst + concavity_worst +
##     concave.points_worst + symmetry_worst + fractal_dimension_worst
##
##                           Df   Deviance AIC
## - concave.points_worst     1 1.8503e-07  44
## - smoothness_mean          1 1.9149e-07  44
## - compactness_worst        1 2.0006e-07  44
## - fractal_dimension_mean   1 2.0323e-07  44
## - perimeter_mean           1 2.0820e-07  44
## - radius_mean              1 2.0822e-07  44
## - fractal_dimension_se     1 2.1177e-07  44
## - concave.points_se        1 2.2491e-07  44
## - area_worst               1 2.2611e-07  44
## - fractal_dimension_worst  1 2.3040e-07  44
## - compactness_mean         1 2.4763e-07  44
## - perimeter_worst          1 2.6749e-07  44
## - radius_worst             1 2.9191e-07  44
## - perimeter_se             1 2.9532e-07  44
```

```
## - smoothness_worst         1 3.3108e-07  44
## - concavity_worst          1 3.3472e-07  44
## - texture_mean             1 3.4663e-07  44
## - symmetry_worst           1 3.5415e-07  44
## - texture_se               1 3.8426e-07  44
## - radius_se                1 3.9979e-07  44
## - concave.points_mean      1 9.8833e-07  44
## - smoothness_se            1 1.0443e-06  44
## <none>                       1.7320e-07  46
##
## Step:  AIC=44
## diagnosis ~ radius_mean + texture_mean + perimeter_mean + smoothness_mean +
##     compactness_mean + concave.points_mean + fractal_dimension_mean +
##     radius_se + texture_se + perimeter_se + smoothness_se + concave.points_se +
##     fractal_dimension_se + radius_worst + perimeter_worst + area_worst +
##     smoothness_worst + compactness_worst + concavity_worst +
##     symmetry_worst + fractal_dimension_worst
##
##                           Df   Deviance AIC
## - smoothness_mean          1 1.9683e-07  42
## - fractal_dimension_se     1 2.1450e-07  42
## - perimeter_mean           1 2.2422e-07  42
## - radius_mean              1 2.2537e-07  42
## - compactness_worst        1 2.2585e-07  42
## - area_worst               1 2.3210e-07  42
## - concave.points_se        1 2.3267e-07  42
## - fractal_dimension_worst  1 2.3451e-07  42
## - fractal_dimension_mean   1 2.4153e-07  42
## - compactness_mean         1 2.5614e-07  42
## - perimeter_worst          1 3.0195e-07  42
## - smoothness_worst         1 3.3799e-07  42
## - concavity_worst          1 3.5174e-07  42
## - radius_worst             1 3.5175e-07  42
## - symmetry_worst           1 3.5600e-07  42
## - perimeter_se             1 3.6380e-07  42
## - texture_se               1 4.2239e-07  42
## - texture_mean             1 4.3452e-07  42
## - radius_se                1 5.6472e-07  42
## - smoothness_se            1 1.1503e-06  42
## - concave.points_mean      1 1.1816e-06  42
## <none>                       1.8503e-07  44
##
## Step:  AIC=42
## diagnosis ~ radius_mean + texture_mean + perimeter_mean + compactness_mean +
##     concave.points_mean + fractal_dimension_mean + radius_se +
##     texture_se + perimeter_se + smoothness_se + concave.points_se +
##     fractal_dimension_se + radius_worst + perimeter_worst + area_worst +
##     smoothness_worst + compactness_worst + concavity_worst +
##     symmetry_worst + fractal_dimension_worst
##
##                           Df   Deviance AIC
## - radius_mean              1 2.2305e-07  40
## - perimeter_mean           1 2.2591e-07  40
## - compactness_worst        1 2.4008e-07  40
```

```
## - area_worst               1 2.4612e-07  40
## - fractal_dimension_se     1 2.6050e-07  40
## - fractal_dimension_worst  1 2.7949e-07  40
## - concave.points_se        1 3.0364e-07  40
## - perimeter_worst          1 3.1165e-07  40
## - fractal_dimension_mean   1 3.2239e-07  40
## - compactness_mean         1 3.3408e-07  40
## - smoothness_worst         1 3.4596e-07  40
## - radius_worst             1 3.6877e-07  40
## - concavity_worst          1 3.9276e-07  40
## - symmetry_worst           1 4.0263e-07  40
## - perimeter_se             1 4.2300e-07  40
## - texture_mean             1 5.4201e-07  40
## - radius_se                1 5.7962e-07  40
## - texture_se               1 6.9781e-07  40
## - smoothness_se            1 1.1908e-06  40
## - concave.points_mean      1 1.2000e-06  40
## <none>                       1.9683e-07  42
##
## Step:  AIC=40
## diagnosis ~ texture_mean + perimeter_mean + compactness_mean +
##     concave.points_mean + fractal_dimension_mean + radius_se +
##     texture_se + perimeter_se + smoothness_se + concave.points_se +
##     fractal_dimension_se + radius_worst + perimeter_worst + area_worst +
##     smoothness_worst + compactness_worst + concavity_worst +
##     symmetry_worst + fractal_dimension_worst
##
##                           Df   Deviance AIC
## - perimeter_mean           1 2.3046e-07  38
## - compactness_worst        1 2.6521e-07  38
## - area_worst               1 2.7033e-07  38
## - fractal_dimension_se     1 2.7497e-07  38
## - fractal_dimension_worst  1 2.8735e-07  38
## - concave.points_se        1 3.1716e-07  38
## - perimeter_worst          1 3.3595e-07  38
## - concavity_worst          1 3.9498e-07  38
## - smoothness_worst         1 4.0203e-07  38
## - radius_worst             1 4.2086e-07  38
## - perimeter_se             1 4.2680e-07  38
## - fractal_dimension_mean   1 4.6074e-07  38
## - radius_se                1 5.8948e-07  38
## - symmetry_worst           1 6.1186e-07  38
## - texture_mean             1 6.2813e-07  38
## - texture_se               1 8.8267e-07  38
## - smoothness_se            1 1.2227e-06  38
## - concave.points_mean      1 1.4313e-06  38
## - compactness_mean         1 2.0819e-06  38
## <none>                       2.2305e-07  40
##
## Step:  AIC=38
## diagnosis ~ texture_mean + compactness_mean + concave.points_mean +
##     fractal_dimension_mean + radius_se + texture_se + perimeter_se +
##     smoothness_se + concave.points_se + fractal_dimension_se +
##     radius_worst + perimeter_worst + area_worst + smoothness_worst +
```

```
##     compactness_worst + concavity_worst + symmetry_worst + fractal_dimension_worst
##
##                           Df Deviance    AIC
## - compactness_worst        1  0.00000 36.000
## - area_worst               1  0.00000 36.000
## - fractal_dimension_se     1  0.00000 36.000
## - fractal_dimension_worst  1  0.00000 36.000
## - concave.points_se        1  0.00000 36.000
## - perimeter_worst          1  0.00000 36.000
## - perimeter_se             1  0.00000 36.000
## - radius_worst             1  0.00000 36.000
## - fractal_dimension_mean   1  0.00000 36.000
## - concavity_worst          1  0.00000 36.000
## - symmetry_worst           1  0.00000 36.000
## - smoothness_worst         1  0.00000 36.000
## - texture_mean             1  0.00000 36.000
## - radius_se                1  0.00000 36.000
## - texture_se               1  0.00000 36.000
## - smoothness_se            1  0.00000 36.000
## - compactness_mean         1  0.00001 36.000
## - concave.points_mean      1  1.49861 37.499
## <none>                        0.00000 38.000
##
## Step:  AIC=36
## diagnosis ~ texture_mean + compactness_mean + concave.points_mean +
##     fractal_dimension_mean + radius_se + texture_se + perimeter_se +
##     smoothness_se + concave.points_se + fractal_dimension_se +
##     radius_worst + perimeter_worst + area_worst + smoothness_worst +
##     concavity_worst + symmetry_worst + fractal_dimension_worst
##
##                           Df Deviance    AIC
## - area_worst               1    0.000 34.000
## - fractal_dimension_worst  1    0.000 34.000
## - perimeter_worst          1    0.000 34.000
## - fractal_dimension_se     1    0.000 34.000
## - concave.points_se        1    0.000 34.000
## - perimeter_se             1    0.000 34.000
## - radius_worst             1    0.000 34.000
## - concavity_worst          1    0.000 34.000
## - fractal_dimension_mean   1    0.000 34.000
## - symmetry_worst           1    0.000 34.000
## - radius_se                1    0.000 34.000
## - smoothness_worst         1    0.000 34.000
## - texture_mean             1    0.000 34.000
## - texture_se               1    0.000 34.000
## - smoothness_se            1    0.000 34.000
## <none>                          0.000 36.000
## - concave.points_mean      1   17.825 51.825
## - compactness_mean         1   20.697 54.697
##
## Step:  AIC=34
## diagnosis ~ texture_mean + compactness_mean + concave.points_mean +
##     fractal_dimension_mean + radius_se + texture_se + perimeter_se +
##     smoothness_se + concave.points_se + fractal_dimension_se +
```

```
##     radius_worst + perimeter_worst + smoothness_worst + concavity_worst +
##     symmetry_worst + fractal_dimension_worst
##
##                           Df Deviance    AIC
## - fractal_dimension_worst  1   0.0000 32.000
## - fractal_dimension_se     1   0.0000 32.000
## - concave.points_se        1   0.0000 32.000
## - concavity_worst          1   0.0000 32.000
## - perimeter_se             1   0.0000 32.000
## - radius_worst             1   0.0000 32.000
## - fractal_dimension_mean   1   0.0000 32.000
## - smoothness_worst         1   0.0000 32.000
## - texture_mean             1   0.0000 32.000
## - radius_se                1   0.0000 32.000
## - perimeter_worst          1   0.0000 32.000
## - smoothness_se            1   0.0000 32.000
## - symmetry_worst           1   0.0000 32.000
## - texture_se               1   0.0001 32.000
## <none>                         0.0000 34.000
## - concave.points_mean      1  21.6373 53.637
## - compactness_mean         1  26.5184 58.518
##
## Step:  AIC=32
## diagnosis ~ texture_mean + compactness_mean + concave.points_mean +
##     fractal_dimension_mean + radius_se + texture_se + perimeter_se +
##     smoothness_se + concave.points_se + fractal_dimension_se +
##     radius_worst + perimeter_worst + smoothness_worst + concavity_worst +
##     symmetry_worst
##
##                          Df Deviance    AIC
## - fractal_dimension_se    1    0.000 30.000
## - concave.points_se       1    0.000 30.000
## - concavity_worst         1    0.000 30.000
## - perimeter_se            1    0.000 30.000
## - radius_worst            1    0.000 30.000
## - smoothness_worst        1    0.000 30.000
## - radius_se               1    0.000 30.000
## - perimeter_worst         1    0.000 30.000
## <none>                         0.000 32.000
## - texture_se              1   12.433 42.433
## - fractal_dimension_mean  1   14.851 44.851
## - smoothness_se           1   17.290 47.290
## - texture_mean            1   17.943 47.943
## - symmetry_worst          1   18.302 48.302
## - concave.points_mean     1   21.640 51.640
## - compactness_mean        1   30.084 60.084
##
## Step:  AIC=30
## diagnosis ~ texture_mean + compactness_mean + concave.points_mean +
##     fractal_dimension_mean + radius_se + texture_se + perimeter_se +
##     smoothness_se + concave.points_se + radius_worst + perimeter_worst +
##     smoothness_worst + concavity_worst + symmetry_worst
##
##                          Df Deviance    AIC
```

```
## - concave.points_se       1    0.000 28.000
## - concavity_worst         1    0.000 28.000
## - perimeter_se            1    0.000 28.000
## - radius_worst            1    0.000 28.000
## <none>                         0.000 30.000
## - texture_se              1   13.659 41.659
## - fractal_dimension_mean  1   15.147 43.147
## - perimeter_worst         1   16.990 44.990
## - radius_se               1   18.694 46.694
## - smoothness_worst        1   18.825 46.825
## - texture_mean            1   20.079 48.079
## - symmetry_worst          1   20.735 48.735
## - smoothness_se           1   21.310 49.310
## - concave.points_mean     1   27.291 55.291
## - compactness_mean        1   37.568 65.568
##
## Step:  AIC=28
## diagnosis ~ texture_mean + compactness_mean + concave.points_mean +
##     fractal_dimension_mean + radius_se + texture_se + perimeter_se +
##     smoothness_se + radius_worst + perimeter_worst + smoothness_worst +
##     concavity_worst + symmetry_worst
##
##                          Df Deviance    AIC
## - concavity_worst         1    0.000 26.000
## - perimeter_se            1    0.000 26.000
## - radius_worst            1    0.000 26.000
## <none>                         0.000 28.000
## - perimeter_worst         1   17.111 43.111
## - fractal_dimension_mean  1   17.310 43.310
## - texture_se              1   17.486 43.486
## - smoothness_worst        1   18.915 44.915
## - radius_se               1   19.001 45.001
## - texture_mean            1   20.677 46.677
## - smoothness_se           1   21.541 47.541
## - symmetry_worst          1   21.903 47.903
## - compactness_mean        1   37.578 63.578
## - concave.points_mean     1   37.654 63.654
##
## Step:  AIC=26
## diagnosis ~ texture_mean + compactness_mean + concave.points_mean +
##     fractal_dimension_mean + radius_se + texture_se + perimeter_se +
##     smoothness_se + radius_worst + perimeter_worst + smoothness_worst +
##     symmetry_worst
##
##                          Df Deviance    AIC
## - radius_worst            1    0.000 24.000
## - perimeter_se            1    0.000 24.000
## <none>                         0.000 26.000
## - perimeter_worst         1   23.639 47.639
## - radius_se               1   25.608 49.608
## - fractal_dimension_mean  1   27.781 51.781
## - texture_se              1   29.165 53.165
## - symmetry_worst          1   34.172 58.172
## - smoothness_se           1   35.575 59.575
```

```
## - compactness_mean        1   39.545 63.545
## - smoothness_worst        1   40.424 64.424
## - texture_mean            1   45.799 69.799
## - concave.points_mean     1   47.078 71.078
##
## Step:  AIC=24
## diagnosis ~ texture_mean + compactness_mean + concave.points_mean +
##     fractal_dimension_mean + radius_se + texture_se + perimeter_se +
##     smoothness_se + perimeter_worst + smoothness_worst + symmetry_worst
##
##                          Df Deviance    AIC
## - perimeter_se            1    0.000 22.000
## <none>                         0.000 24.000
## - radius_se               1   27.156 49.156
## - fractal_dimension_mean  1   27.879 49.879
## - texture_se              1   29.553 51.553
## - symmetry_worst          1   34.622 56.622
## - smoothness_se           1   38.232 60.232
## - compactness_mean        1   39.568 61.568
## - perimeter_worst         1   45.241 67.241
## - concave.points_mean     1   47.872 69.872
## - smoothness_worst        1   48.155 70.155
## - texture_mean            1   51.289 73.289
##
## Step:  AIC=22
## diagnosis ~ texture_mean + compactness_mean + concave.points_mean +
##     fractal_dimension_mean + radius_se + texture_se + smoothness_se +
##     perimeter_worst + smoothness_worst + symmetry_worst
##
##                          Df Deviance    AIC
## <none>                         0.000 22.000
## - fractal_dimension_mean  1   27.998 47.998
## - texture_se              1   29.884 49.884
## - symmetry_worst          1   34.642 54.642
## - radius_se               1   35.812 55.812
## - compactness_mean        1   40.116 60.116
## - smoothness_se           1   42.139 62.139
## - perimeter_worst         1   45.872 65.872
## - smoothness_worst        1   49.880 69.880
## - concave.points_mean     1   50.337 70.337
## - texture_mean            1   52.583 72.583
```

```
backward_pred_class_train = ifelse(predict(backward_model,
                                    newdata = train, type = "response") > 0.5, 1, 0)
backward_results_train = calculate_metrics(backward_pred_class_train, train$diagnosis)
backward_results_train$Set = "Train"


backward_pred_class_test = ifelse(predict(backward_model,
                                   newdata = test, type = "response") > 0.5, 1, 0)
backward_results_test = calculate_metrics(backward_pred_class_test, test$diagnosis)
backward_results_test$Set = "Test"
```

```r
print(backward_results_train)
```

```
##      Metric Value   Set
## 1    Recall     1 Train
## 2 Precision     1 Train
## 3  F1 Score     1 Train
## 4  Accuracy     1 Train
```

```r
print(backward_results_test)
```

```
##      Metric     Value  Set
## 1    Recall 0.8620690 Test
## 2 Precision 0.7575758 Test
## 3  F1 Score 0.8064516 Test
## 4  Accuracy 0.8938053 Test
```

In this model the number of false negatives are 4 which is lower than logistic regression with all features.

```r
table(backward_pred_class_test, test$diagnosis)
```

```
##
## backward_pred_class_test  0  1
##                        0 76  8
##                        1  4 25
```

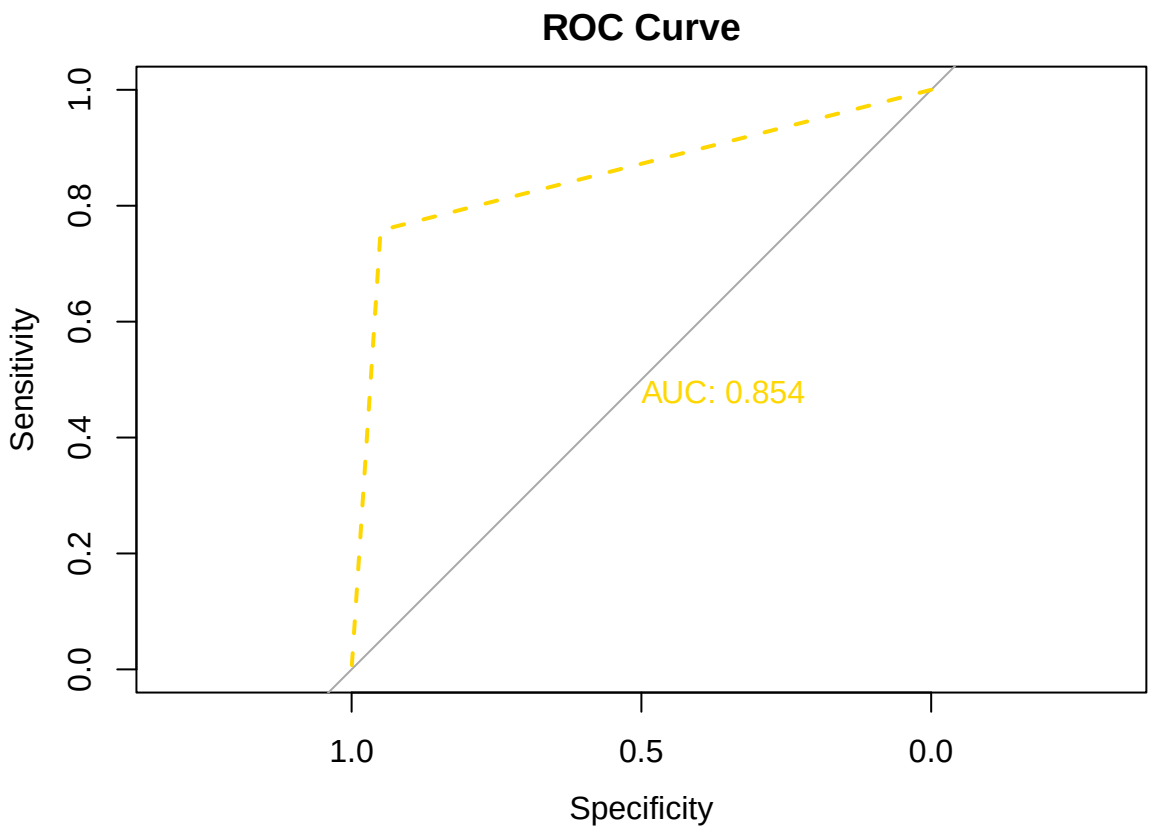
You can see the ROC curve of this model.

```r
roc_obj = roc(test$diagnosis, backward_pred_class_test)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
plot(roc_obj, main = "ROC Curve", print.auc = TRUE,lty = 2 , col = 'gold')
```



3.3 Ridge regression with cross validation

In this part, I used logistic regression for the regularization in order to prevent over-fitting which means increasing the error of training and reducing the generalization error. As you can see this model performed such an incredible model where the value of false negatives is zero so the value of recall is 1.Moreover, the model has a good value of accuracy, precision and F1-score. So the model has a highly good performance.

```
x_train = model.matrix(diagnosis ~ ., data = train)[,-1]
y_train = train$diagnosis

x_test = model.matrix(diagnosis ~ ., data = test)[,-1]
y_test = test$diagnosis

cv_model = cv.glmnet(x_train, y_train, family = "binomial", alpha = 0)

best_lambda = cv_model$lambda.min


ridge_model = glmnet(x_train, y_train, family = "binomial", alpha = 0,
                     lambda = best_lambda)



ridge_pred_class_train = ifelse(predict(ridge_model, newx = x_train, type = "response")
```

```
                                > 0.5, 1, 0)
ridge_results_train = calculate_metrics(ridge_pred_class_train, y_train)
ridge_results_train$Set = "Train"


ridge_pred_class_test = ifelse(predict(ridge_model, newx = x_test, type = "response")
                               > 0.5, 1, 0)
ridge_results_test = calculate_metrics(ridge_pred_class_test, y_test)
ridge_results_test$Set = "Test"

print(ridge_results_train)
```

```
##      Metric     Value   Set
## 1    Recall 0.9884393 Train
## 2 Precision 0.9553073 Train
## 3  F1 Score 0.9715909 Train
## 4  Accuracy 0.9780702 Train
```

```
print(ridge_results_test)
```

```
##      Metric     Value  Set
## 1    Recall 1.0000000 Test
## 2 Precision 0.8181818 Test
## 3  F1 Score 0.9000000 Test
## 4  Accuracy 0.9469027 Test
```

Below, you can see the best value of hyperparameter lambda and the confusion matrix of the model. As you can see the model has significantly improved in comparison to the two previous models.

```
best_lambda
```

```
## [1] 0.03985099
```

```
table(ridge_pred_class_test, y_test)
```

```
##                       y_test
## ridge_pred_class_test  0  1
##                     0 80  6
##                     1  0 27
```

Now let's see which features have been chosen by this model.

```
coef(ridge_model)
```

```
## 31 x 1 sparse Matrix of class "dgCMatrix"
##                                      s0
## (Intercept)             -1.690969e+01
## radius_mean              8.794815e-02
## texture_mean             7.411104e-02
## perimeter_mean           1.251538e-02
```

```
## area_mean                8.304500e-04
## smoothness_mean          8.931397e+00
## compactness_mean         1.453137e+00
## concavity_mean           4.003537e+00
## concave.points_mean      9.463537e+00
## symmetry_mean            3.142319e+00
## fractal_dimension_mean  -1.982128e+01
## radius_se                1.161758e+00
## texture_se               3.686029e-02
## perimeter_se             1.210508e-01
## area_se                  5.463153e-03
## smoothness_se           -4.294362e+01
## compactness_se          -7.952538e+00
## concavity_se             1.779338e-01
## concave.points_se        1.616191e+01
## symmetry_se             -9.367319e+00
## fractal_dimension_se    -9.084056e+01
## radius_worst             8.467271e-02
## texture_worst            7.495861e-02
## perimeter_worst          1.138526e-02
## area_worst               6.398143e-04
## smoothness_worst         1.442301e+01
## compactness_worst        9.947711e-01
## concavity_worst          1.689403e+00
## concave.points_worst     6.696873e+00
## symmetry_worst           4.819597e+00
## fractal_dimension_worst  7.619654e+00
```

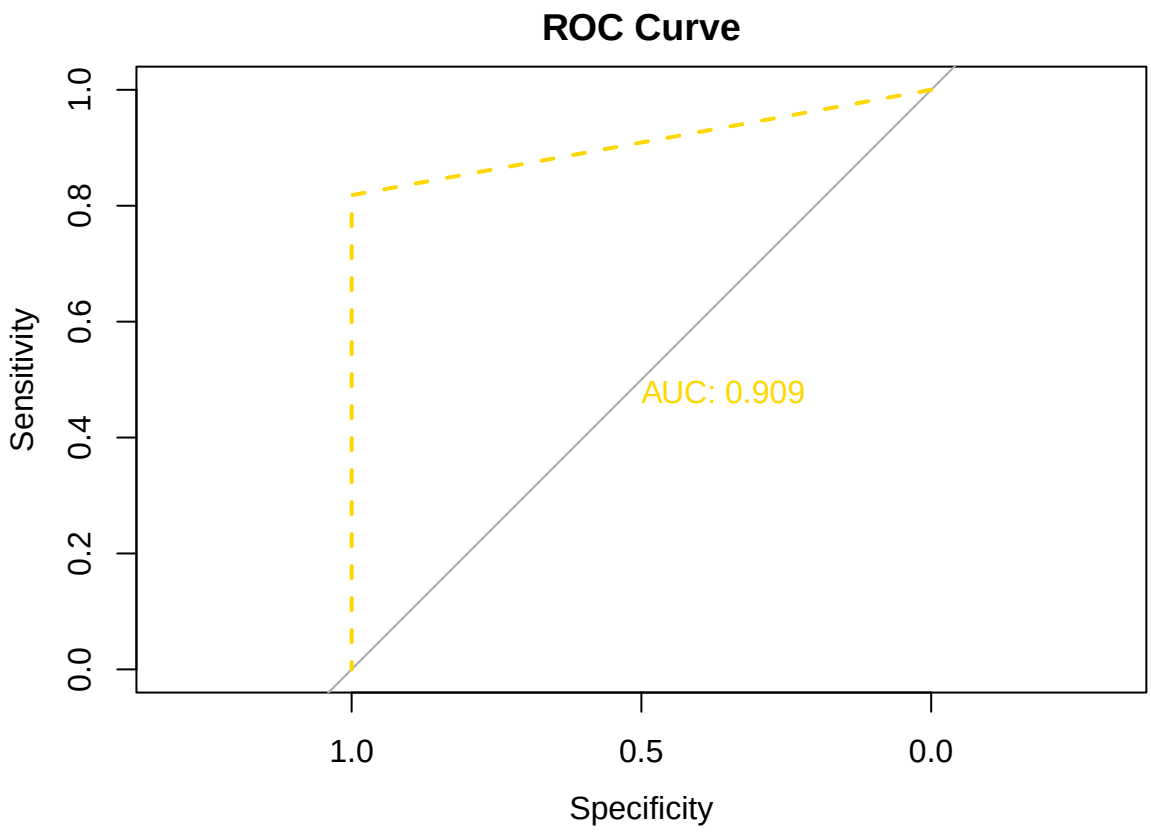
The ROC curve for the ridge is shown below.

```
roc_obj = roc(test$diagnosis,ridge_pred_class_test)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(roc_obj, main = "ROC Curve", print.auc = TRUE,lty = 2 , col = 'gold')
```



3.4 LASSO regression with cross validation

The lasso regression model is fitted below with cross validation in order to find the best hyperparameter.

```
cv_model = cv.glmnet(x_train, y_train, family = "binomial", alpha = 1)
best_lambda = cv_model$lambda.min

lasso_model = glmnet(x_train, y_train, family = "binomial", alpha = 1,
                     lambda = best_lambda)
lasso_pred_class_train = ifelse(predict(ridge_model, newx = x_train, type = "response")
                                > 0.5, 1, 0)
lasso_results_train = calculate_metrics(ridge_pred_class_train, y_train)
lasso_results_train$Set = "Train"


lasso_pred_class_test = ifelse(predict(lasso_model, newx = x_test,
                                       type = "response")> 0.5, 1, 0)

lasso_results_test = calculate_metrics(lasso_pred_class_test, y_test)
lasso_results_test$Set = "Test"


print(lasso_results_train)
```

```
##      Metric     Value   Set
## 1    Recall 0.9884393 Train
## 2 Precision 0.9553073 Train
## 3  F1 Score 0.9715909 Train
## 4  Accuracy 0.9780702 Train
```

```
print(lasso_results_test)
```

```
##      Metric     Value  Set
## 1    Recall 0.9642857 Test
## 2 Precision 0.8181818 Test
## 3  F1 Score 0.8852459 Test
## 4  Accuracy 0.9380531 Test
```

I have found the best hyperparameter and you can see the confusion matrix of the model. This model has performed very well and could obtain good metrics values.

```
best_lambda
```

```
## [1] 0.002176772
```

```
table(lasso_pred_class_test, y_test)
```

```
##                       y_test
## lasso_pred_class_test  0  1
##                     0 79  6
##                     1  1 27
```

Now let's see which coefficients has been chosen by this model. The chosen features are concave.points_mean, radius_se, smoothness_se, compactness_se, fractal_dimension_se, texture_worst, area_worst, smoothness_worst, concavity_worst, concave.points_worst, and symmetry_worst.

```
coef(lasso_model)
```

```
## 31 x 1 sparse Matrix of class "dgCMatrix"
##                                       s0
## (Intercept)             -2.506201e+01
## radius_mean              .
## texture_mean             .
## perimeter_mean           .
## area_mean                .
## smoothness_mean          .
## compactness_mean         .
## concavity_mean           .
## concave.points_mean      2.883777e+01
## symmetry_mean            .
## fractal_dimension_mean   .
## radius_se                9.195815e+00
## texture_se               .
## perimeter_se             .
## area_se                  .
```

```
## smoothness_se           -2.650703e+01
## compactness_se          -3.556757e+01
## concavity_se             .
## concave.points_se        .
## symmetry_se              .
## fractal_dimension_se    -2.094315e+02
## radius_worst             .
## texture_worst            2.344093e-01
## perimeter_worst          .
## area_worst               7.117673e-03
## smoothness_worst         2.452529e+01
## compactness_worst        .
## concavity_worst          6.845394e+00
## concave.points_worst     1.988291e+01
## symmetry_worst           6.774327e+00
## fractal_dimension_worst  .
```

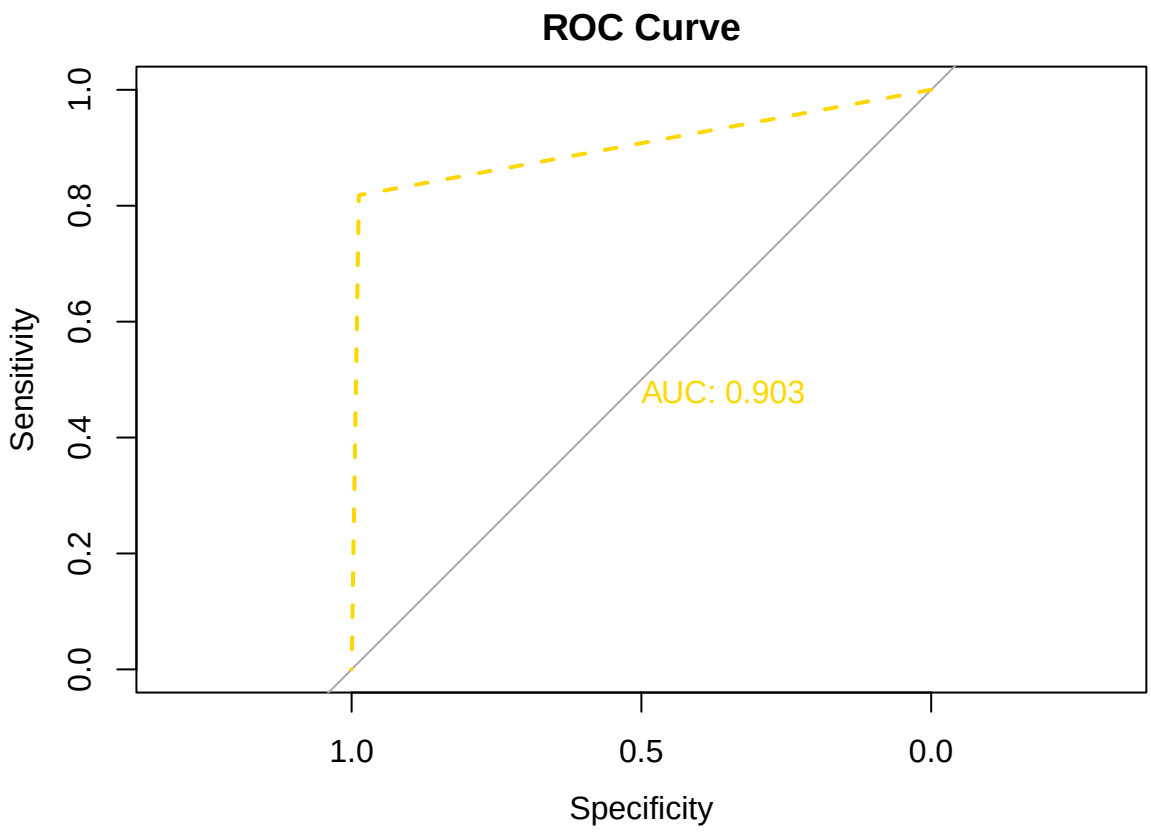
The ROC curve of this model is shown below.

```
roc_obj = roc(test$diagnosis,lasso_pred_class_test)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(roc_obj, main = "ROC Curve", print.auc = TRUE,lty = 2 , col = 'gold')
```



3.5 LDA

In this section, I trained the LDA model. As you can see this model could obtain good results but the value of precision if not very good since the number of false positives is higher than in other models. false positives are the one who doesn't have cancer but the model predicted them as cancerous.

```
LDA_model = lda(diagnosis ~ ., data = train)

LDA_pred_train = predict( LDA_model, newdata = train)$class
LDA_pred_test = predict( LDA_model, newdata = test)$class

LDA_train_result = calculate_metrics(LDA_pred_train, train$diagnosis)
LDA_test_result = calculate_metrics(LDA_pred_test, test$diagnosis)


print(LDA_train_result)
```

```
##      Metric     Value
## 1    Recall 0.9940120
## 2 Precision 0.9273743
## 3  F1 Score 0.9595376
## 4  Accuracy 0.9692982
```

```r
print(LDA_test_result)
```

```
##      Metric     Value
## 1    Recall 1.0000000
## 2 Precision 0.6666667
## 3  F1 Score 0.8000000
## 4  Accuracy 0.9026549
```

You can see the confusion matrix below.

```r
table(LDA_pred_test, y_test)
```

```
##              y_test
## LDA_pred_test  0  1
##             0 80 11
##             1  0 22
```

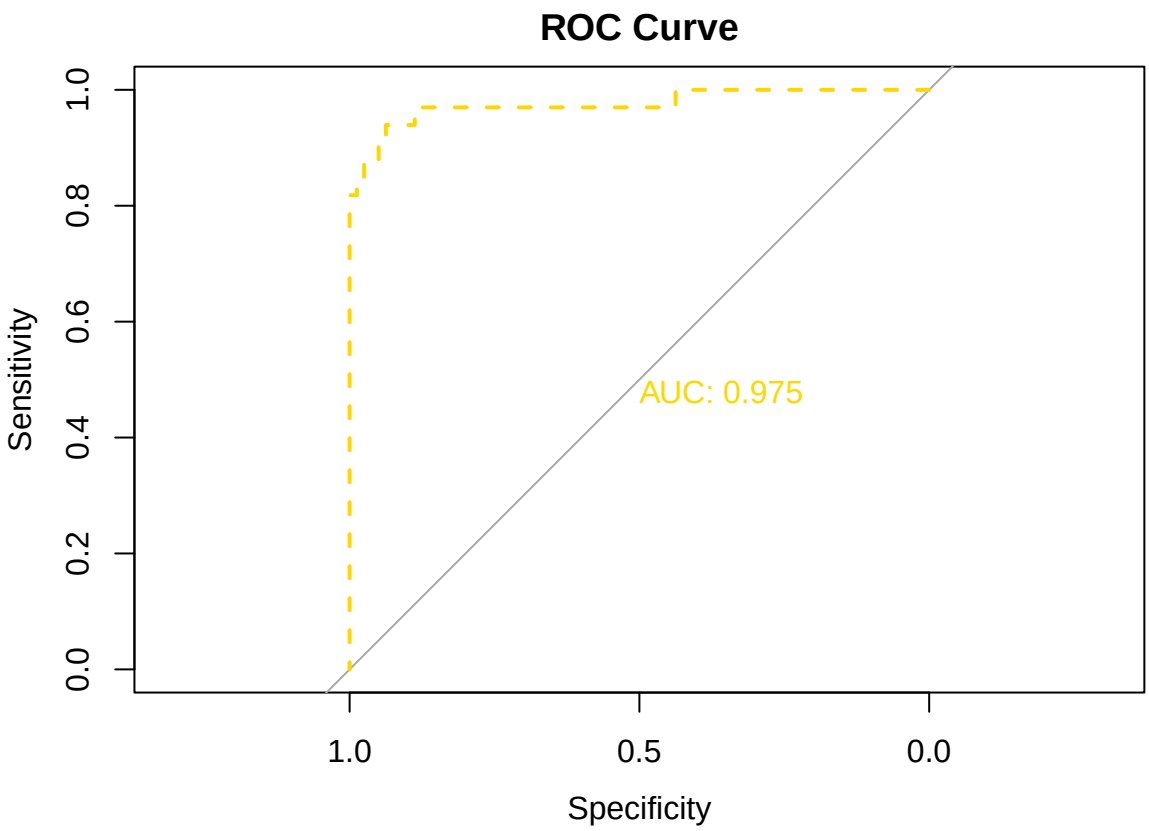
Now, let's check the ROC curve.

```r
LDA_pred_prob_test = predict(LDA_model, newdata = test)$posterior[, 1]
roc_obj = roc(test$diagnosis, LDA_pred_prob_test)
```

```
## Setting levels: control = 0, case = 1
```

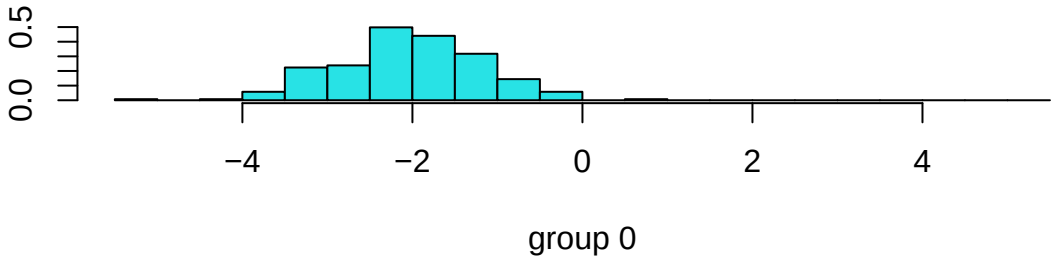
```
## Setting direction: controls > cases
```

```r
plot(roc_obj, main = "ROC Curve", print.auc = TRUE, ylim = c(0, 1), lty = 2, col = 'gold')
```

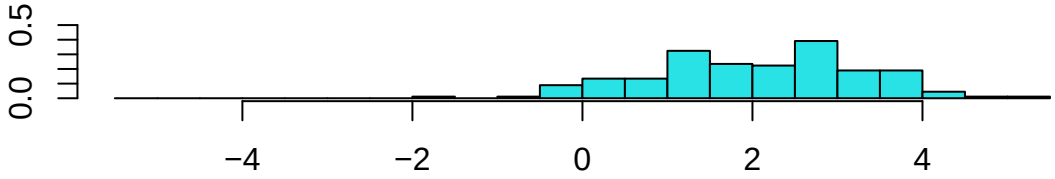


In the plots below you can see that the LDA model has preformed a good performance for two available classes.

```
plot(LDA_model)
```

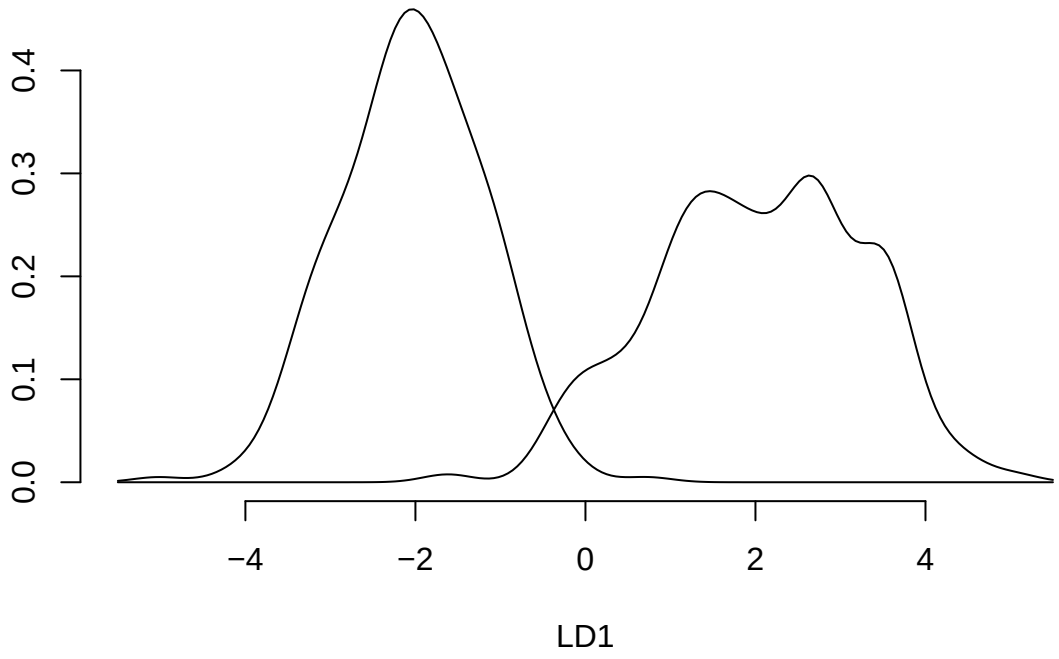


group 0



group 1

```
plot(LDA_model, type="density")
```



3.6 QDA

QDA model has preformed well but it has lower results in comparison to LDA model.

```
QDA_model = qda(diagnosis ~ ., data = train)

QDA_pred_train = predict( QDA_model, newdata = train)$class
QDA_pred_test = predict( QDA_model, newdata = test)$class

QDA_train_result = calculate_metrics(QDA_pred_train, train$diagnosis)
QDA_test_result = calculate_metrics(QDA_pred_test, test$diagnosis)


print(QDA_train_result)
```

```
##      Metric     Value
## 1    Recall 0.9715909
## 2 Precision 0.9553073
## 3  F1 Score 0.9633803
## 4  Accuracy 0.9714912
```

```
print(QDA_test_result)
```

```
##      Metric     Value
## 1    Recall 0.9354839
## 2 Precision 0.8787879
## 3  F1 Score 0.9062500
## 4  Accuracy 0.9469027
```

Below, you can see the confusion matrix and the ROC curve.

```
table(QDA_pred_test, y_test)
```

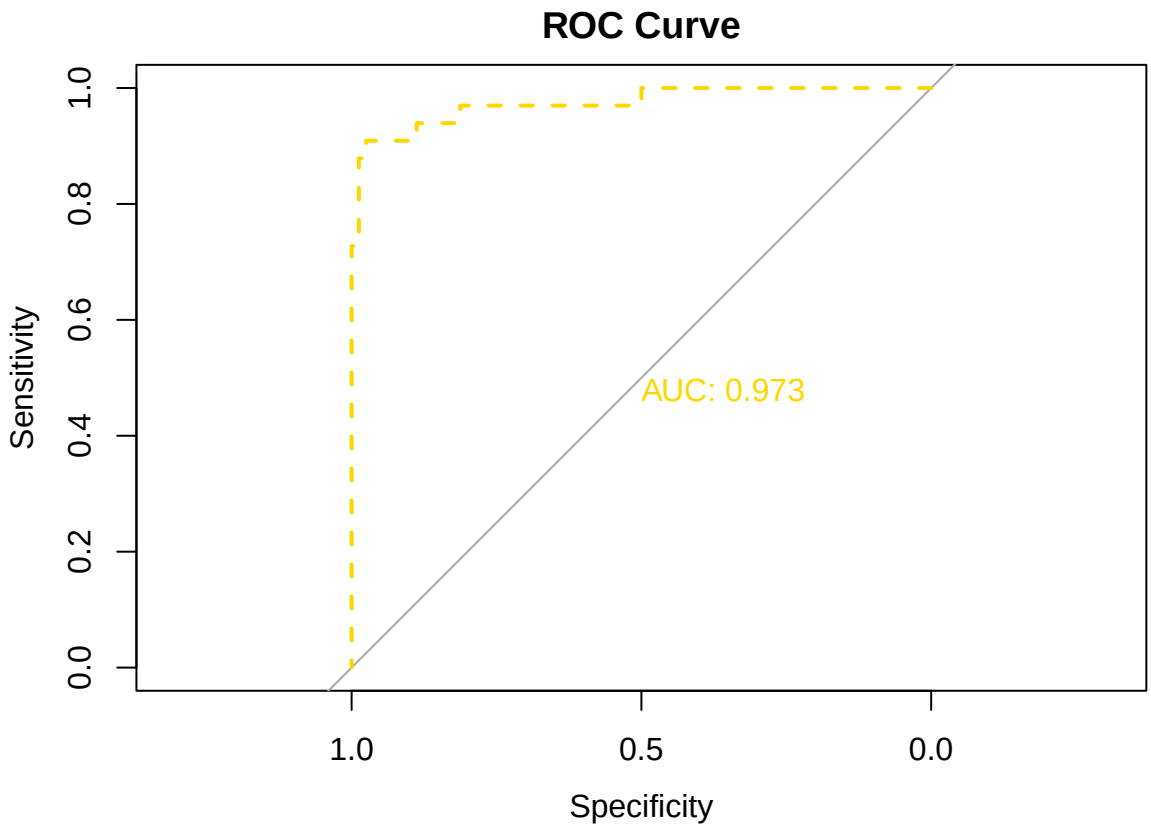
```
##              y_test
## QDA_pred_test  0  1
##             0 78  4
##             1  2 29
```

```
QDA_pred_prob_test = predict(QDA_model, newdata = test)$posterior[, 1]
roc_obj = roc(test$diagnosis, QDA_pred_prob_test)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls > cases
```

```
plot(roc_obj, main = "ROC Curve", print.auc = TRUE,lty = 2 , col = 'gold')
```



3.7 Naive Bayes

In this section, I will train a naive Bayes model. This model has performed a good job too. the value of metrics is pretty good.

```
nb_model = naiveBayes(diagnosis ~ ., data = train)

nb_pred_train = predict(nb_model, newdata = train, type = 'class')
nb_pred_test = predict(nb_model, newdata = test , type = 'class')


NB_train_result = calculate_metrics(nb_pred_train, train$diagnosis)
NB_test_result = calculate_metrics(nb_pred_test, test$diagnosis)

print(NB_train_result)
```

```
##      Metric     Value
## 1    Recall 0.9310345
## 2 Precision 0.9050279
## 3  F1 Score 0.9178470
## 4  Accuracy 0.9364035
```

```
print(NB_test_result)
```

```
##      Metric     Value
## 1    Recall 0.7941176
## 2 Precision 0.8181818
## 3  F1 Score 0.8059701
## 4  Accuracy 0.8849558
```

You can see the confusion matrix and the ROC curve of the Naive Bayes model.

```
table(nb_pred_test, test$diagnosis)
```

```
##
## nb_pred_test  0  1
##            0 73  6
##            1  7 27
```

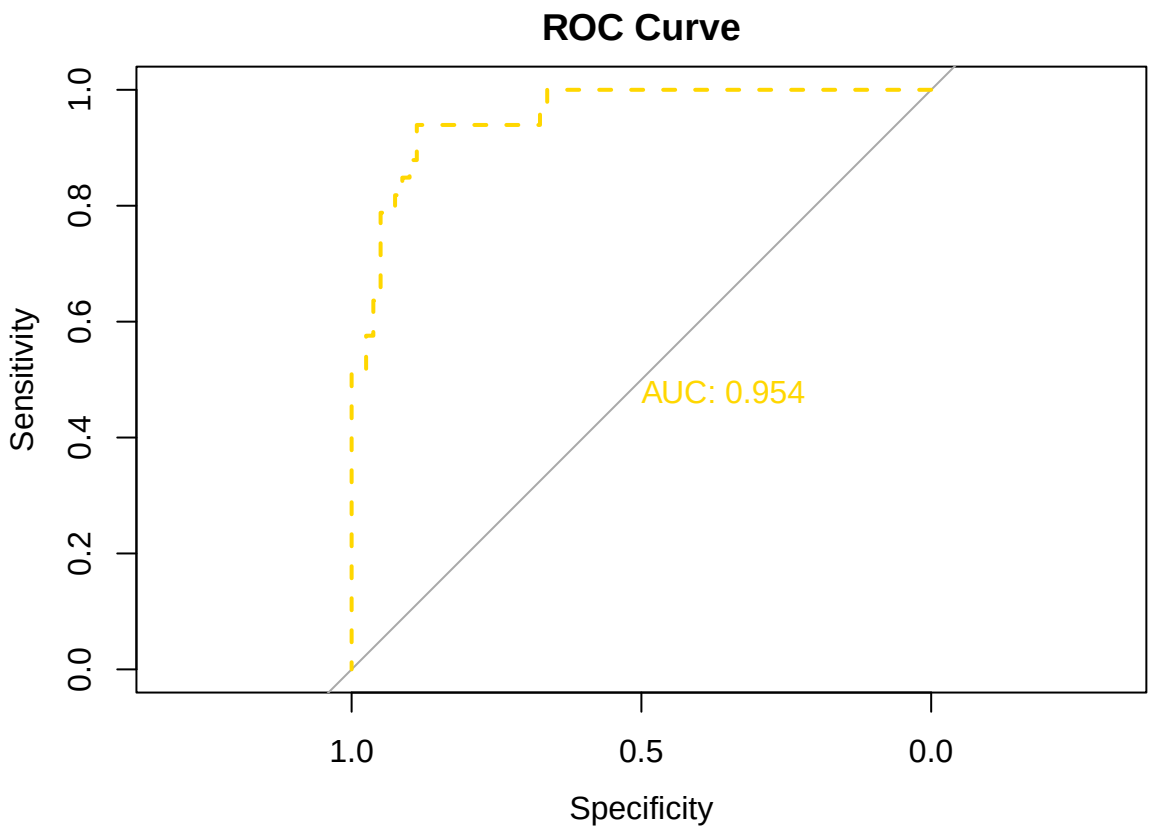
```
nb_probs_test = predict(nb_model, newdata = test, type = 'raw')
```

```
roc_obj = roc(test$diagnosis, nb_probs_test[,1])
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls > cases
```

```
plot(roc_obj, main = "ROC Curve", print.auc = TRUE,lty = 2 , col = 'gold')
```



3.8 KNN

First, Let's find the best value for K. As you can see the best value of K is 18 where the model has the highest value of metrics.

```
k_values = c(2 , 4 , 6 , 8 , 10 , 12 , 14 , 16 , 18 , 20)
KNN_results = list()

for (k in k_values) {
  KNN_model = knn(train[, -1], test[, -1], train[, 1], k)
  KNN_results[[as.character(k)]] = calculate_metrics(KNN_model, test$diagnosis)
}

for (k in k_values) {
  cat("Metrics for k =", k, ":\n")
  print(KNN_results[[as.character(k)]])
  cat("\n")
}
```

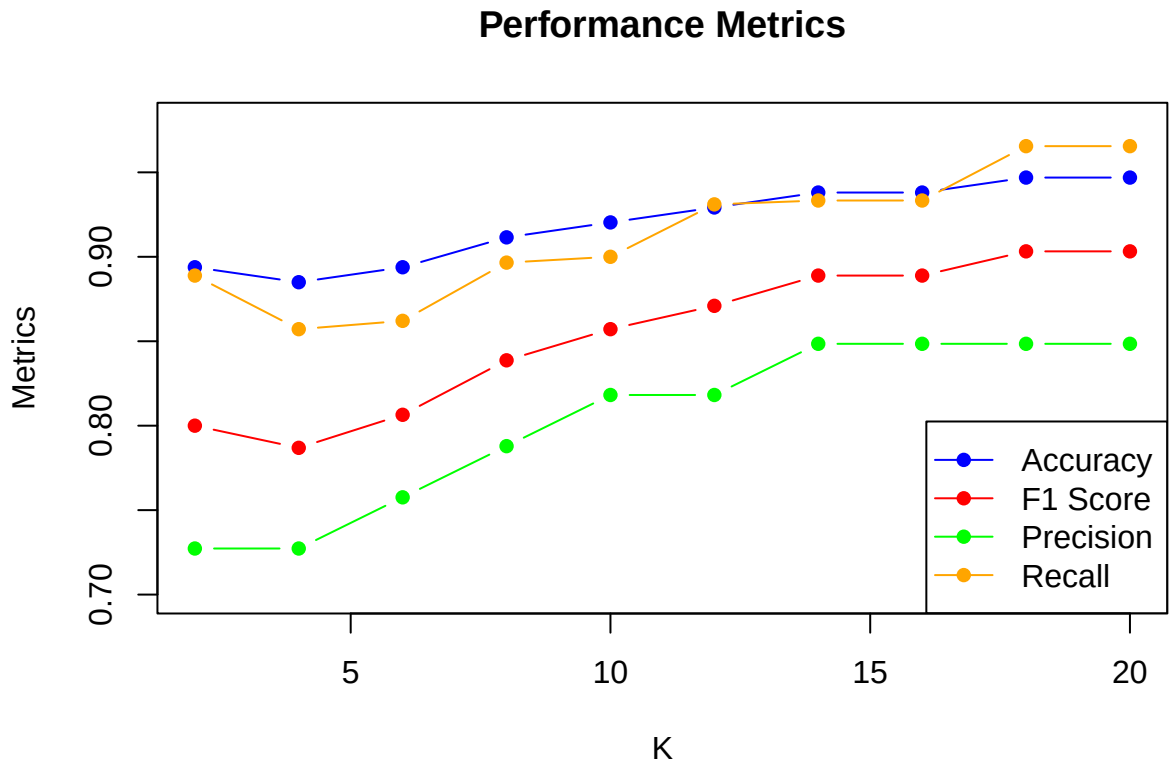
```
## Metrics for k = 2 :
##      Metric     Value
## 1    Recall 0.8888889
## 2 Precision 0.7272727
## 3  F1 Score 0.8000000
```

```
## 4  Accuracy 0.8938053
##
## Metrics for k = 4 :
##      Metric     Value
## 1    Recall 0.8571429
## 2 Precision 0.7272727
## 3  F1 Score 0.7868852
## 4  Accuracy 0.8849558
##
## Metrics for k = 6 :
##      Metric     Value
## 1    Recall 0.8620690
## 2 Precision 0.7575758
## 3  F1 Score 0.8064516
## 4  Accuracy 0.8938053
##
## Metrics for k = 8 :
##      Metric     Value
## 1    Recall 0.8965517
## 2 Precision 0.7878788
## 3  F1 Score 0.8387097
## 4  Accuracy 0.9115044
##
## Metrics for k = 10 :
##      Metric     Value
## 1    Recall 0.9000000
## 2 Precision 0.8181818
## 3  F1 Score 0.8571429
## 4  Accuracy 0.9203540
##
## Metrics for k = 12 :
##      Metric     Value
## 1    Recall 0.9310345
## 2 Precision 0.8181818
## 3  F1 Score 0.8709677
## 4  Accuracy 0.9292035
##
## Metrics for k = 14 :
##      Metric     Value
## 1    Recall 0.9333333
## 2 Precision 0.8484848
## 3  F1 Score 0.8888889
## 4  Accuracy 0.9380531
##
## Metrics for k = 16 :
##      Metric     Value
## 1    Recall 0.9333333
## 2 Precision 0.8484848
## 3  F1 Score 0.8888889
## 4  Accuracy 0.9380531
##
## Metrics for k = 18 :
##      Metric     Value
## 1    Recall 0.9655172
```

```
## 2 Precision 0.8484848
## 3  F1 Score 0.9032258
## 4  Accuracy 0.9469027
##
## Metrics for k = 20 :
##      Metric     Value
## 1    Recall 0.9655172
## 2 Precision 0.8484848
## 3  F1 Score 0.9032258
## 4  Accuracy 0.9469027
```

```
K = c(2, 4, 6, 8, 10, 12 , 14 , 16 , 18 , 20)
Accuracy = c(KNN_results$'2'[4,2],
             KNN_results$'4'[4,2],
             KNN_results$'6'[4,2],
             KNN_results$'8'[4,2],
             KNN_results$'10'[4,2],
             KNN_results$'12'[4,2],
             KNN_results$'14'[4,2],
             KNN_results$'16'[4,2],
             KNN_results$'18'[4,2],
             KNN_results$'20'[4,2])

F1_Score = c(KNN_results$'2'[3,2],
             KNN_results$'4'[3,2],
             KNN_results$'6'[3,2],
             KNN_results$'8'[3,2],
             KNN_results$'10'[3,2],
             KNN_results$'12'[3,2],
             KNN_results$'14'[3,2],
             KNN_results$'16'[3,2],
             KNN_results$'18'[3,2],
             KNN_results$'20'[3,2])

Precision = c(KNN_results$'2'[2,2],
              KNN_results$'4'[2,2],
              KNN_results$'6'[2,2],
              KNN_results$'8'[2,2],
              KNN_results$'10'[2,2],
              KNN_results$'12'[2,2],
              KNN_results$'14'[2,2],
              KNN_results$'16'[2,2],
              KNN_results$'18'[2,2],
              KNN_results$'20'[2,2])


Recall = c(KNN_results$'2'[1,2],
           KNN_results$'4'[1,2],
           KNN_results$'6'[1,2],
           KNN_results$'8'[1,2],
           KNN_results$'10'[1,2],
           KNN_results$'12'[1,2],
           KNN_results$'14'[1,2],
           KNN_results$'16'[1,2],
```

```
            KNN_results$'18'[1,2],
            KNN_results$'20'[1,2])

# Create a line plot
plot(K, Accuracy, type = "b", pch = 16, col = "blue",
     xlab = "K", ylab = "Metrics", main = "Performance Metrics" , ylim = c(0.7 , 0.98))
lines(K, F1_Score, type = "b", pch = 16, col = "red")
lines(K, Precision, type = "b", pch = 16, col = "green")
lines(K, Recall, type = "b", pch = 16, col = "orange")

# Add a legend
legend("bottomright", legend = c("Accuracy", "F1 Score", "Precision", "Recall"),
       col = c("blue", "red", "green", "orange"), lty = 1, pch = 16)
```



Below, you can see the value of metrics, the confusion matrix and the ROC curve.

```
KNN_model = knn(train[, -1], test[, -1], train[, 1], k)
KNN_probs = as.numeric(KNN_model == 1)
KNN_results$'18'
```

```
##      Metric     Value
## 1    Recall 0.9655172
## 2 Precision 0.8484848
## 3  F1 Score 0.9032258
## 4  Accuracy 0.9469027
```

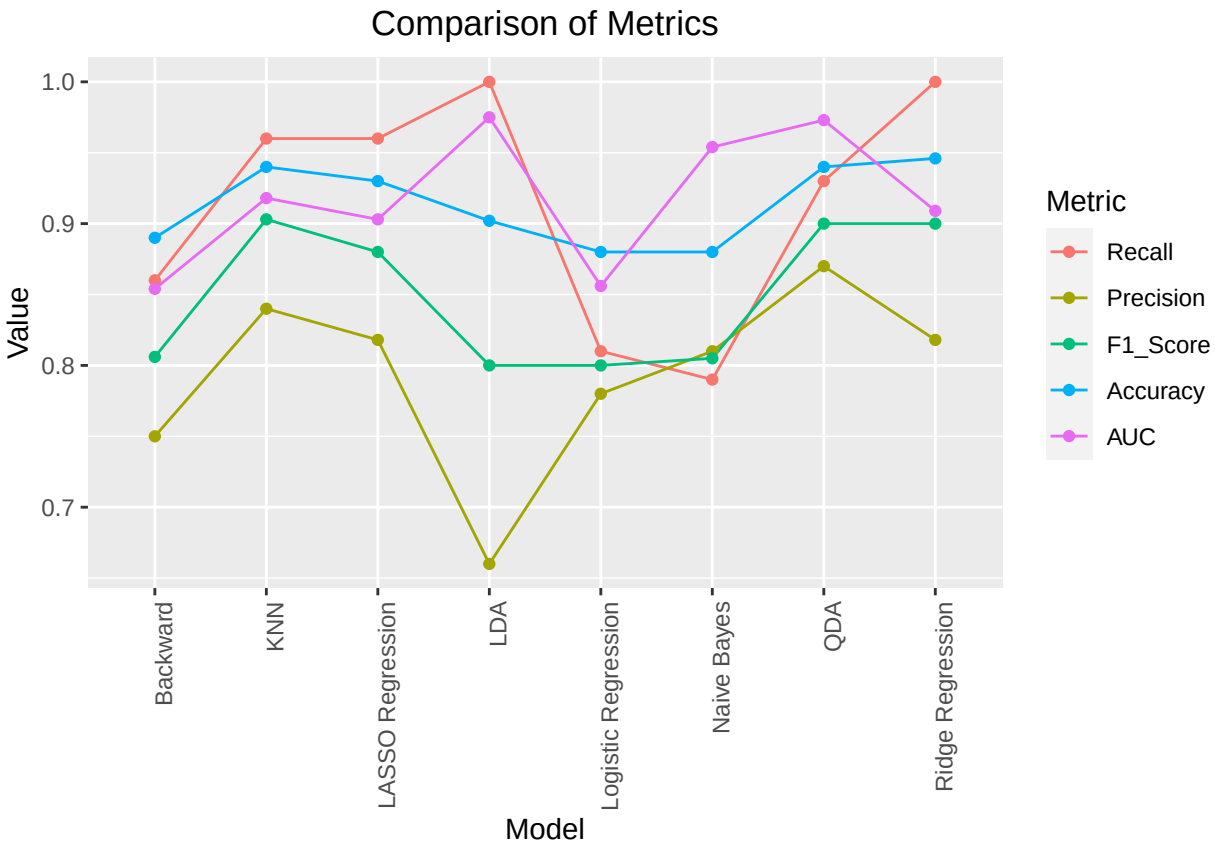
```r
table(KNN_probs , test$diagnosis)
```

```
##
## KNN_probs  0  1
##         0 79  5
##         1  1 28
```

4 Conclusion

check this grammatically Due to the plot below you can see that the Ridge regression model and the LDA model have the highest possible value for recall but the Ridge regression model has the higher value for the other metrics such as precision, F1-score, Accuracy, and AUC. So the preferred model among all the available models is the Ridge regression model.

```r
results = data.frame(
  Model = c("Naive Bayes", "Logistic Regression", "Backward", "QDA",
            "KNN", "LASSO Regression", "LDA", "Ridge Regression"),
  Recall = c(0.79, 0.81, 0.86,0.93,0.96,0.96,1,1),
  Precision = c(0.81,0.78, 0.75,0.87,0.84,0.818,0.66, 0.818),
  F1_Score = c(0.805,0.8, 0.806,0.90,0.903,0.88,0.8,0.9),
  Accuracy = c(0.88,0.88, 0.89,0.94,0.94,0.93,0.902,0.946),
  AUC = c(0.954,0.856, 0.854,0.973,0.918,0.903,0.975, 0.909)
)

# Reshape the data into long format
results_long = reshape2::melt(results, id.vars = "Model", variable.name = "Metric")

# Plot the results using a line plot
ggplot(results_long, aes(x = Model, y = value, color = Metric, group = Metric)) +
  geom_line() +
  geom_point() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(y = "Value", x = "Model", color = "Metric") +
  ggtitle("Comparison of Metrics") +
  theme(plot.title = element_text(hjust = 0.5))
```


Comparison of Metrics
1.0
0.9
0.8
0.7
Value
Backward
KNN
LASSO Regression
LDA
Logistic Regression
Naive Bayes
QDA
Ridge Regression
Model
Metric
Recall
Precision
F1_Score
Accuracy
AUC