ADS 503 Final Project

April Chia

2025-06-05

Packages used

```r
library(caret)
library(tidyverse)
library(ggplot2)
library(reshape2)
library(corrplot)
library(Hmisc)
library(mlbench)
library(e1071)
library(randomForest)
library(gt)
library(pls)
library(elasticnet)
library(pROC)
```

Import dataset

```r
cancer_data <- read.csv("breast-cancer.csv")
head(cancer_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
## 1               0.2654         0.4601                 0.11890
## 2               0.1860         0.2750                 0.08902
## 3               0.2430         0.3613                 0.08758
## 4               0.2575         0.6638                 0.17300
## 5               0.1625         0.2364                 0.07678
## 6               0.1741         0.3985                 0.12440
```

EDA

```r
summary(cancer_data)
```

```
##        id             diagnosis          radius_mean      texture_mean
##  Min.   :     8670   Length:569         Min.   : 6.981   Min.   : 9.71
##  1st Qu.:   869218   Class :character   1st Qu.:11.700   1st Qu.:16.17
##  Median :   906024   Mode  :character   Median :13.370   Median :18.84
##  Mean   : 30371831                      Mean   :14.127   Mean   :19.29
##  3rd Qu.:  8813129                      3rd Qu.:15.780   3rd Qu.:21.80
##  Max.   :911320502                      Max.   :28.110   Max.   :39.28
##  perimeter_mean     area_mean      smoothness_mean   compactness_mean
##  Min.   : 43.79   Min.   : 143.5   Min.   :0.05263   Min.   :0.01938
##  1st Qu.: 75.17   1st Qu.: 420.3   1st Qu.:0.08637   1st Qu.:0.06492
##  Median : 86.24   Median : 551.1   Median :0.09587   Median :0.09263
##  Mean   : 91.97   Mean   : 654.9   Mean   :0.09636   Mean   :0.10434
##  3rd Qu.:104.10   3rd Qu.: 782.7   3rd Qu.:0.10530   3rd Qu.:0.13040
##  Max.   :188.50   Max.   :2501.0   Max.   :0.16340   Max.   :0.34540
##  concavity_mean    concave.points_mean symmetry_mean    fractal_dimension_mean
##  Min.   :0.00000   Min.   :0.00000     Min.   :0.1060   Min.   :0.04996
##  1st Qu.:0.02956   1st Qu.:0.02031     1st Qu.:0.1619   1st Qu.:0.05770
##  Median :0.06154   Median :0.03350     Median :0.1792   Median :0.06154
##  Mean   :0.08880   Mean   :0.04892     Mean   :0.1812   Mean   :0.06280
##  3rd Qu.:0.13070   3rd Qu.:0.07400     3rd Qu.:0.1957   3rd Qu.:0.06612
##  Max.   :0.42680   Max.   :0.20120     Max.   :0.3040   Max.   :0.09744
##    radius_se        texture_se      perimeter_se       area_se
##  Min.   :0.1115   Min.   :0.3602   Min.   : 0.757   Min.   :  6.802
##  1st Qu.:0.2324   1st Qu.:0.8339   1st Qu.: 1.606   1st Qu.: 17.850
##  Median :0.3242   Median :1.1080   Median : 2.287   Median : 24.530
##  Mean   :0.4052   Mean   :1.2169   Mean   : 2.866   Mean   : 40.337
##  3rd Qu.:0.4789   3rd Qu.:1.4740   3rd Qu.: 3.357   3rd Qu.: 45.190
##  Max.   :2.8730   Max.   :4.8850   Max.   :21.980   Max.   :542.200
##  smoothness_se      compactness_se      concavity_se     concave.points_se
##  Min.   :0.001713   Min.   :0.002252   Min.   :0.00000   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
```

```r
# Data types
str(cancer_data)
```

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

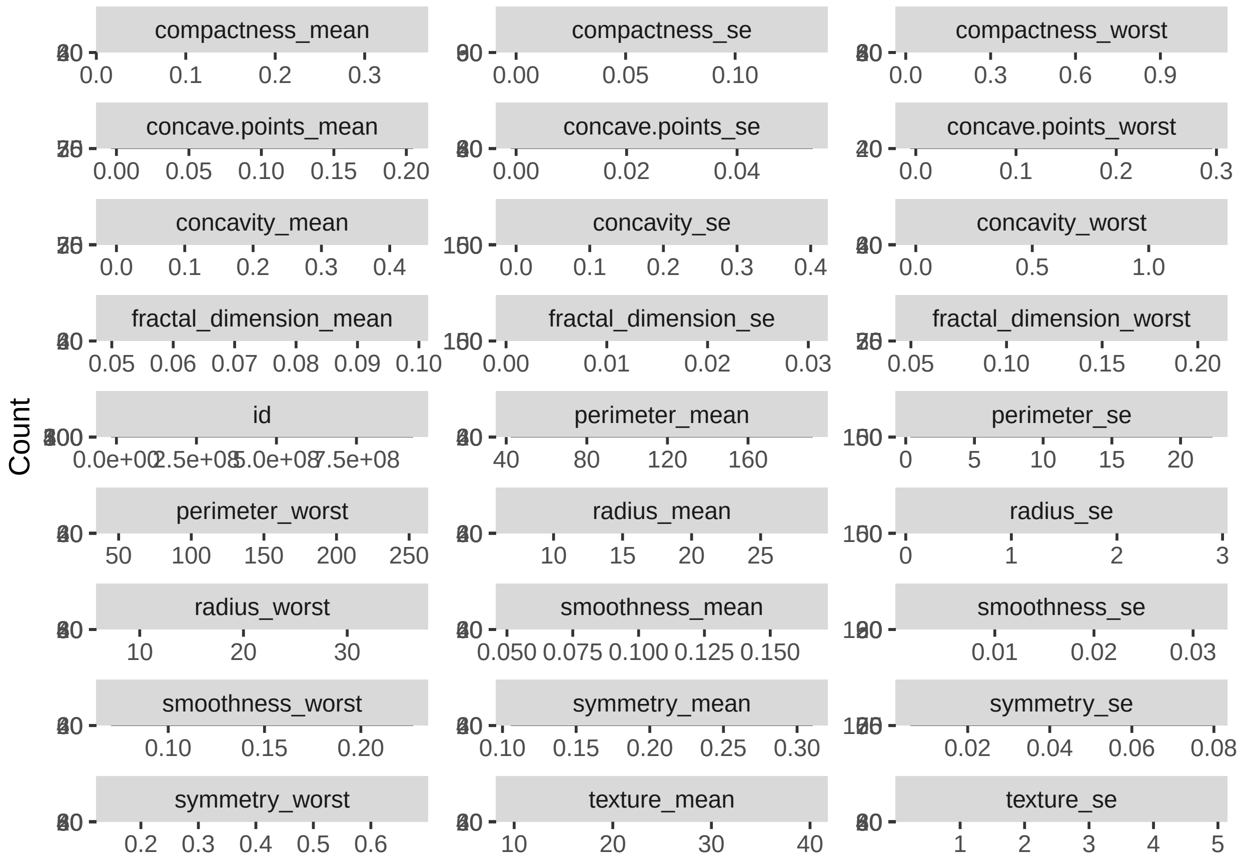
```
# Missing values
sum(is.na(cancer_data))
```

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

```
# Duplicates
sum(duplicated(cancer_data))
```

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

```
# Distribution of predictors
cancer_data |>
pivot_longer(-diagnosis, names_to = 'feature', values_to = 'value') |>
ggplot(aes(x = value)) +
geom_histogram(bins = 30) +
facet_wrap(~ feature, scales = "free", ncol = 3) +
labs(title = 'Glass Data Features', x = "values", y = "Count")
```



```r
# Distribution of diagnosis classes
table(cancer_data$diagnosis)
```

```
##
##   B   M
## 357 212
```

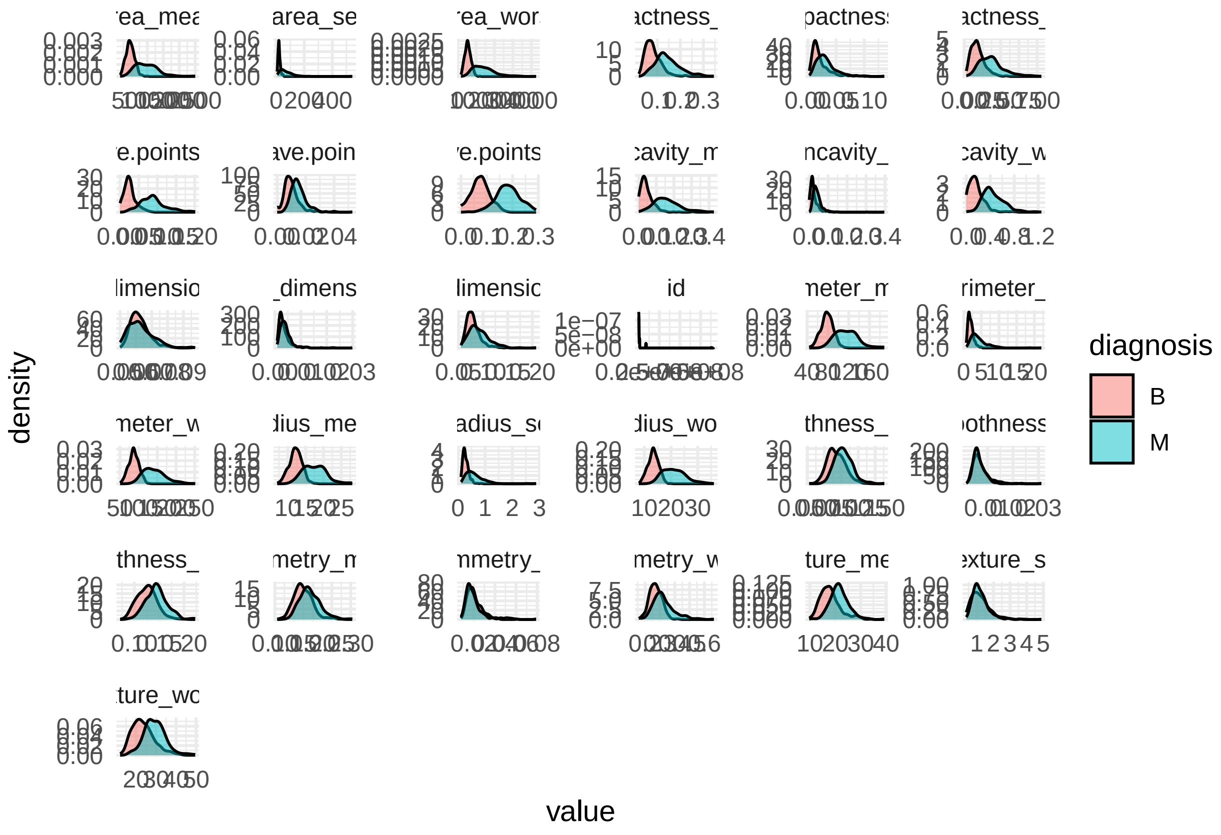
```r
prop.table(table(cancer_data$diagnosis))
```

```
##
##         B         M
## 0.6274165 0.3725835
```

```r
# Relationship between predictors and response
predictor_data <- cancer_data[, names(cancer_data) != "diagnosis"]

# Convert to long format
df_long <- data.frame(
  diagnosis = rep(cancer_data$diagnosis, times = ncol(predictor_data)),
  feature = rep(names(predictor_data), each = nrow(cancer_data)),
  value = as.vector(as.matrix(predictor_data))
)

ggplot(df_long, aes(x = value, fill = diagnosis)) +
  geom_density(alpha = 0.5) +
  facet_wrap(~ feature, scales = "free") +
  theme_minimal()
```

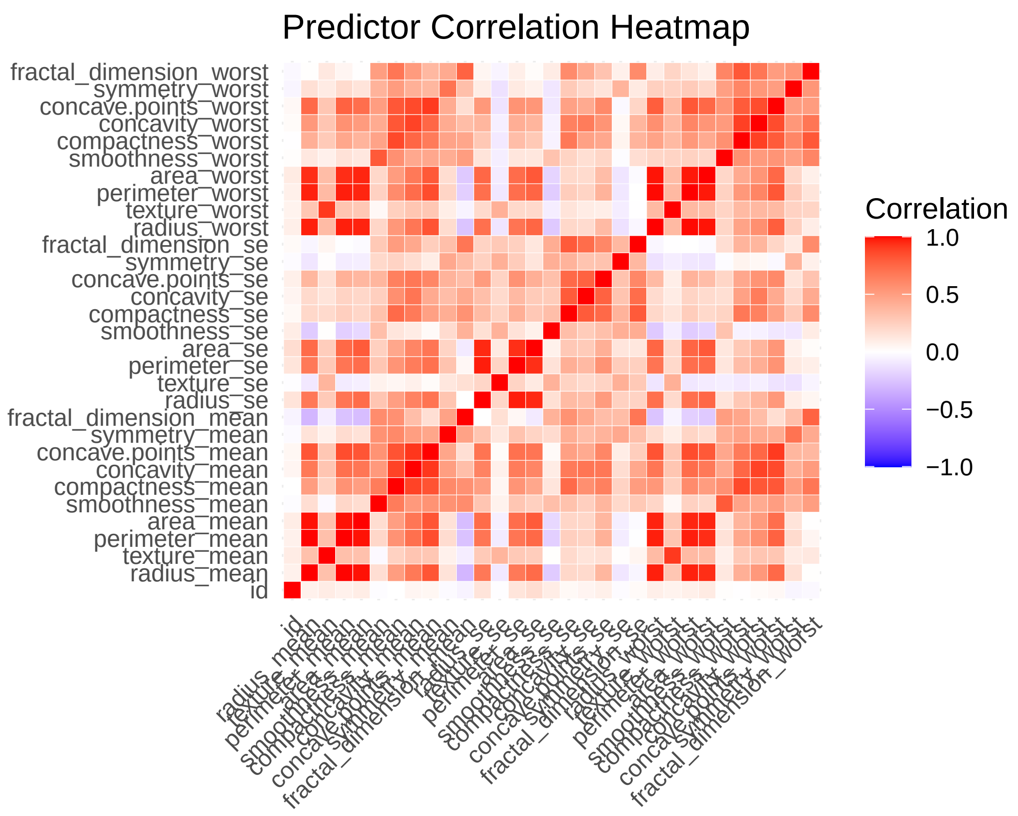


```r
# Predictors w/ near zero variance
degenerate <- nearZeroVar(predictor_data)
print(degenerate)
```

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

```r
# Correlation between predictors
cor_matrix <- cor(predictor_data)
cor_long <- melt(cor_matrix)

ggplot(cor_long, aes(Var1, Var2, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1, 1), space = "Lab",
                       name = "Correlation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1)) +
  coord_fixed() +
  labs(title = "Predictor Correlation Heatmap", x = "", y = "")
```



```r
# Skewness
apply(cancer_data[, -2], 2, skewness)
```

```
##                      id             radius_mean            texture_mean
##               6.4396595               0.9374168               0.6470241
##          perimeter_mean               area_mean         smoothness_mean
##               0.9854334               1.6370654               0.4539207
##        compactness_mean          concavity_mean     concave.points_mean
##               1.1838556               1.3938008               1.1650124
##           symmetry_mean  fractal_dimension_mean               radius_se
##               0.7217877               1.2976191               3.0723468
##              texture_se            perimeter_se                 area_se
##               1.6377733               3.4254803               5.4185001
##           smoothness_se          compactness_se            concavity_se
##               2.3022616               1.8922032               5.0835502
##       concave.points_se             symmetry_se    fractal_dimension_se
##               1.4370701               2.1835728               3.9033041
##            radius_worst           texture_worst         perimeter_worst
##               1.0973059               0.4956970               1.1222227
##              area_worst        smoothness_worst       compactness_worst
##               1.8495814               0.4132383               1.4657948
##         concavity_worst    concave.points_worst          symmetry_worst
##               1.1441794               0.4900213               1.4263764
## fractal_dimension_worst
##               1.6538237
```

Pre-processing

```
# Remove uneccessary columns
df <- cancer_data[, -which(names(cancer_data) == "id")]
head(df)
```

```
##   diagnosis radius_mean texture_mean perimeter_mean area_mean smoothness_mean
## 1         M       17.99        10.38         122.80    1001.0         0.11840
## 2         M       20.57        17.77         132.90    1326.0         0.08474
## 3         M       19.69        21.25         130.00    1203.0         0.10960
## 4         M       11.42        20.38          77.58     386.1         0.14250
## 5         M       20.29        14.34         135.10    1297.0         0.10030
## 6         M       12.45        15.70          82.57     477.1         0.12780
##   compactness_mean concavity_mean concave.points_mean symmetry_mean
## 1          0.27760         0.3001             0.14710        0.2419
## 2          0.07864         0.0869             0.07017        0.1812
## 3          0.15990         0.1974             0.12790        0.2069
## 4          0.28390         0.2414             0.10520        0.2597
## 5          0.13280         0.1980             0.10430        0.1809
## 6          0.17000         0.1578             0.08089        0.2087
##   fractal_dimension_mean radius_se texture_se perimeter_se area_se
## 1                0.07871    1.0950     0.9053        8.589  153.40
## 2                0.05667    0.5435     0.7339        3.398   74.08
## 3                0.05999    0.7456     0.7869        4.585   94.03
## 4                0.09744    0.4956     1.1560        3.445   27.23
## 5                0.05883    0.7572     0.7813        5.438   94.44
## 6                0.07613    0.3345     0.8902        2.217   27.19
##   smoothness_se compactness_se concavity_se concave.points_se symmetry_se
## 1      0.006399        0.04904      0.05373           0.01587     0.03003
## 2      0.005225        0.01308      0.01860           0.01340     0.01389
## 3      0.006150        0.04006      0.03832           0.02058     0.02250
## 4      0.009110        0.07458      0.05661           0.01867     0.05963
## 5      0.011490        0.02461      0.05688           0.01885     0.01756
## 6      0.007510        0.03345      0.03672           0.01137     0.02165
##   fractal_dimension_se radius_worst texture_worst perimeter_worst area_worst
## 1             0.006193        25.38         17.33          184.60     2019.0
## 2             0.003532        24.99         23.41          158.80     1956.0
## 3             0.004571        23.57         25.53          152.50     1709.0
## 4             0.009208        14.91         26.50           98.87      567.7
## 5             0.005115        22.54         16.67          152.20     1575.0
## 6             0.005082        15.47         23.75          103.40      741.6
##   smoothness_worst compactness_worst concavity_worst concave.points_worst
## 1           0.1622            0.6656          0.7119               0.2654
## 2           0.1238            0.1866          0.2416               0.1860
## 3           0.1444            0.4245          0.4504               0.2430
## 4           0.2098            0.8663          0.6869               0.2575
## 5           0.1374            0.2050          0.4000               0.1625
## 6           0.1791            0.5249          0.5355               0.1741
##   symmetry_worst fractal_dimension_worst
## 1         0.4601                 0.11890
## 2         0.2750                 0.08902
## 3         0.3613                 0.08758
## 4         0.6638                 0.17300
## 5         0.2364                 0.07678
## 6         0.3985                 0.12440
```

```r
# Convert diagnosis to factor
df$diagnosis <- factor(df$diagnosis, levels = c("B", "M"))

# BoxCox Transformation
non_bct_cols <- c("smoothness_mean", "texture_worst", "smoothness_worst", "concave.points_worst")
bct_cols <- setdiff(names(df), non_bct_cols)

params <- preProcess(df[, bct_cols], method = "BoxCox")
df_transformed <- predict(params, df[, bct_cols])
df[, bct_cols] <- df_transformed
```

```r
# Confirm transformation
apply(df_transformed[, -1], 2, skewness)
```

```
##             radius_mean            texture_mean          perimeter_mean
##            -0.018084005            -0.013801528            -0.018259725
##               area_mean        compactness_mean          concavity_mean
##             0.283456808            -0.033906489             1.393800804
##     concave.points_mean           symmetry_mean  fractal_dimension_mean
##             1.165012377             0.001737667             0.150646585
##               radius_se              texture_se            perimeter_se
##             0.027176088             0.029036809             0.069227942
##                 area_se           smoothness_se          compactness_se
##             0.115303422            -0.024011982            -0.004019758
##            concavity_se       concave.points_se             symmetry_se
##             5.083550174             1.437070137             0.054910585
##    fractal_dimension_se            radius_worst         perimeter_worst
##             0.012191507             0.026399596             0.061225231
##              area_worst       compactness_worst         concavity_worst
##             0.067682043            -0.220675829             1.144179410
##          symmetry_worst fractal_dimension_worst
##            -0.056548989             0.047053460
```

Data splitting

```r
set.seed(123)
trainIndex <- createDataPartition(df_transformed$diagnosis, p = 0.8, list = FALSE)

train <- df_transformed[trainIndex, ]
test <- df_transformed[-trainIndex, ]
```

Models

```r
ctrl <- trainControl(method = "cv",
                     number = 10,
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE,
                     savePredictions = TRUE)

set.seed(123)
# Logistic Regression
lr_model <- train(x = train[, -1],
                  y = train$diagnosis,
                  method = "glm",
                  preProcess = c("center", "scale"),
                  metric = "ROC",
                  trControl = ctrl)
```

```
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
lr_model
```

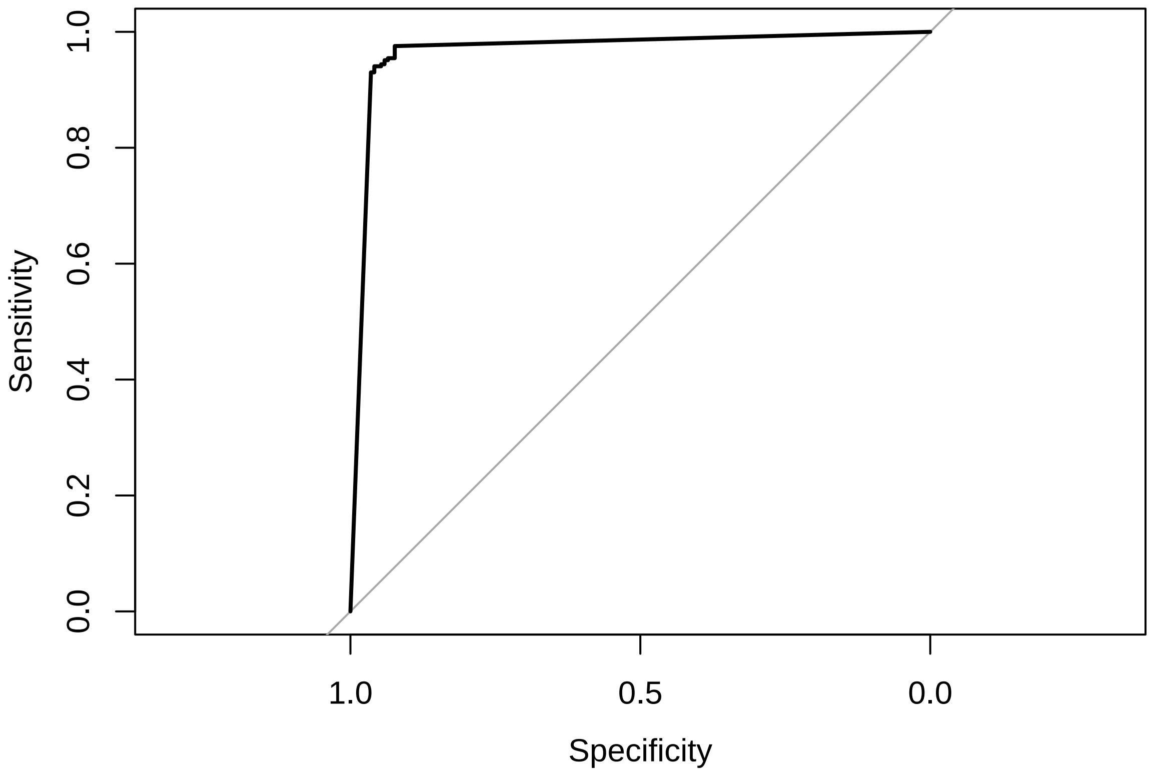
```
## Generalized Linear Model
##
## 456 samples
##  26 predictor
##   2 classes: 'B', 'M'
##
## Pre-processing: centered (26), scaled (26)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 410, 410, 411, 411, 410, 411, ...
## Resampling results:
##
##   ROC        Sens       Spec
##   0.9667959  0.9506158  0.9352941
```

```r
testResults <- data.frame(obs = test$diagnosis, LogReg = predict(lr_model, test[, -1]))
```

```r
# ROC curve for log reg
lr_roc <- roc(response = lr_model$pred$obs,
              predictor = lr_model$pred$M,
              levels = rev(levels(lr_model$pred$obs)))
```

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

```r
plot(lr_roc, legaces.axes = TRUE)
```



```r
# Confusion Matrix
confusionMatrix(testResults$LogReg, testResults$obs, positive = "M")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 68  1
##          M  3 41
##
##                Accuracy : 0.9646
##                  95% CI : (0.9118, 0.9903)
##     No Information Rate : 0.6283
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9249
##
##  Mcnemar's Test P-Value : 0.6171
##
```

```
##             Sensitivity : 0.9762
##             Specificity : 0.9577
##          Pos Pred Value : 0.9318
##          Neg Pred Value : 0.9855
##              Prevalence : 0.3717
##          Detection Rate : 0.3628
##    Detection Prevalence : 0.3894
##       Balanced Accuracy : 0.9670
##
##        'Positive' Class : M
##
```

```
# Penalized Logistic Regression
plrGrid <- expand.grid(alpha = c(0, .1, .2, .4, .6, .8, 1),
                       lambda = seq(.01, .2, length = 10))

set.seed(123)
plr_model <- train(x = train[, -1],
                   y = train$diagnosis,
                   method = "glmnet",
                   tuneGrid = plrGrid,
                   preProcess = c("center", "scale"),
                   metric = "ROC",
                   trControl = ctrl)
plr_model
```

```
## glmnet
##
## 456 samples
##  26 predictor
##   2 classes: 'B', 'M'
##
## Pre-processing: centered (26), scaled (26)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 410, 410, 411, 411, 410, 411, ...
## Resampling results across tuning parameters:
##
##   alpha  lambda      ROC        Sens       Spec
##   0.0    0.01000000  0.9947117  0.9791872  0.9294118
##   0.0    0.03111111  0.9947117  0.9791872  0.9294118
##   0.0    0.05222222  0.9936902  0.9791872  0.9235294
##   0.0    0.07333333  0.9926615  0.9756158  0.9117647
##   0.0    0.09444444  0.9918357  0.9756158  0.9058824
##   0.0    0.11555556  0.9910099  0.9756158  0.9058824
##   0.0    0.13666667  0.9901985  0.9720443  0.8882353
##   0.0    0.15777778  0.9889670  0.9720443  0.8823529
##   0.0    0.17888889  0.9885540  0.9720443  0.8823529
##   0.0    0.20000000  0.9883512  0.9720443  0.8823529
##   0.1    0.01000000  0.9949218  0.9896552  0.9411765
##   0.1    0.03111111  0.9938931  0.9860837  0.9235294
##   0.1    0.05222222  0.9918502  0.9791872  0.9176471
##   0.1    0.07333333  0.9910316  0.9757389  0.9117647
##   0.1    0.09444444  0.9895972  0.9757389  0.9058824
##   0.1    0.11555556  0.9891843  0.9756158  0.8941176
##   0.1    0.13666667  0.9885613  0.9756158  0.8823529
##   0.1    0.15777778  0.9885613  0.9756158  0.8764706
##   0.1    0.17888889  0.9885613  0.9756158  0.8764706
##   0.1    0.20000000  0.9885613  0.9756158  0.8705882
##   0.2    0.01000000  0.9951246  0.9860837  0.9411765
##   0.2    0.03111111  0.9924660  0.9826355  0.9235294
##   0.2    0.05222222  0.9912417  0.9791872  0.9176471
##   0.2    0.07333333  0.9896045  0.9791872  0.9117647
##   0.2    0.09444444  0.9894016  0.9791872  0.8941176
##   0.2    0.11555556  0.9891843  0.9756158  0.8882353
##   0.2    0.13666667  0.9887786  0.9756158  0.8882353
##   0.2    0.15777778  0.9889815  0.9756158  0.8823529
##   0.2    0.17888889  0.9889815  0.9791872  0.8764706
##   0.2    0.20000000  0.9887641  0.9791872  0.8647059
##   0.4    0.01000000  0.9941032  0.9789409  0.9411765
##   0.4    0.03111111  0.9912489  0.9860837  0.9235294
##   0.4    0.05222222  0.9900174  0.9826355  0.9117647
##   0.4    0.07333333  0.9896045  0.9791872  0.8882353
##   0.4    0.09444444  0.9898073  0.9826355  0.8882353
##   0.4    0.11555556  0.9891843  0.9826355  0.8823529
##   0.4    0.13666667  0.9889670  0.9896552  0.8764706
##   0.4    0.15777778  0.9889597  0.9896552  0.8411765
##   0.4    0.17888889  0.9885468  0.9931034  0.8352941
##   0.4    0.20000000  0.9875254  0.9931034  0.8294118
##   0.6    0.01000000  0.9932918  0.9825123  0.9411765
##   0.6    0.03111111  0.9910461  0.9860837  0.9117647
##   0.6    0.05222222  0.9904231  0.9895320  0.9117647
##   0.6    0.07333333  0.9904086  0.9895320  0.8941176
##   0.6    0.09444444  0.9895900  0.9895320  0.8764706
##   0.6    0.11555556  0.9887496  0.9965517  0.8588235
##   0.6    0.13666667  0.9889525  0.9965517  0.8352941
##   0.6    0.15777778  0.9885323  0.9965517  0.8117647
##   0.6    0.17888889  0.9881266  0.9965517  0.8000000
##   0.6    0.20000000  0.9875109  0.9965517  0.7941176
##   0.8    0.01000000  0.9926760  0.9825123  0.9411765
##   0.8    0.03111111  0.9904303  0.9860837  0.9117647
##   0.8    0.05222222  0.9908287  0.9895320  0.9058824
##   0.8    0.07333333  0.9895755  0.9895320  0.8705882
##   0.8    0.09444444  0.9889380  0.9929803  0.8588235
##   0.8    0.11555556  0.9885323  0.9894089  0.8235294
##   0.8    0.13666667  0.9875036  0.9965517  0.8117647
##   0.8    0.15777778  0.9873008  0.9965517  0.8000000
##   0.8    0.17888889  0.9873080  0.9965517  0.7823529
##   0.8    0.20000000  0.9871124  1.0000000  0.7705882
##   1.0    0.01000000  0.9924804  0.9825123  0.9352941
##   1.0    0.03111111  0.9906259  0.9895320  0.9235294
##   1.0    0.05222222  0.9902057  0.9895320  0.8941176
##   1.0    0.07333333  0.9889525  0.9894089  0.8588235
##   1.0    0.09444444  0.9862793  0.9858374  0.8352941
##   1.0    0.11555556  0.9856781  0.9929803  0.8235294
##   1.0    0.13666667  0.9836352  0.9929803  0.8000000
##   1.0    0.15777778  0.9830194  0.9929803  0.7705882
##   1.0    0.17888889  0.9830194  0.9964286  0.7235294
##   1.0    0.20000000  0.9830194  1.0000000  0.6823529
##
```

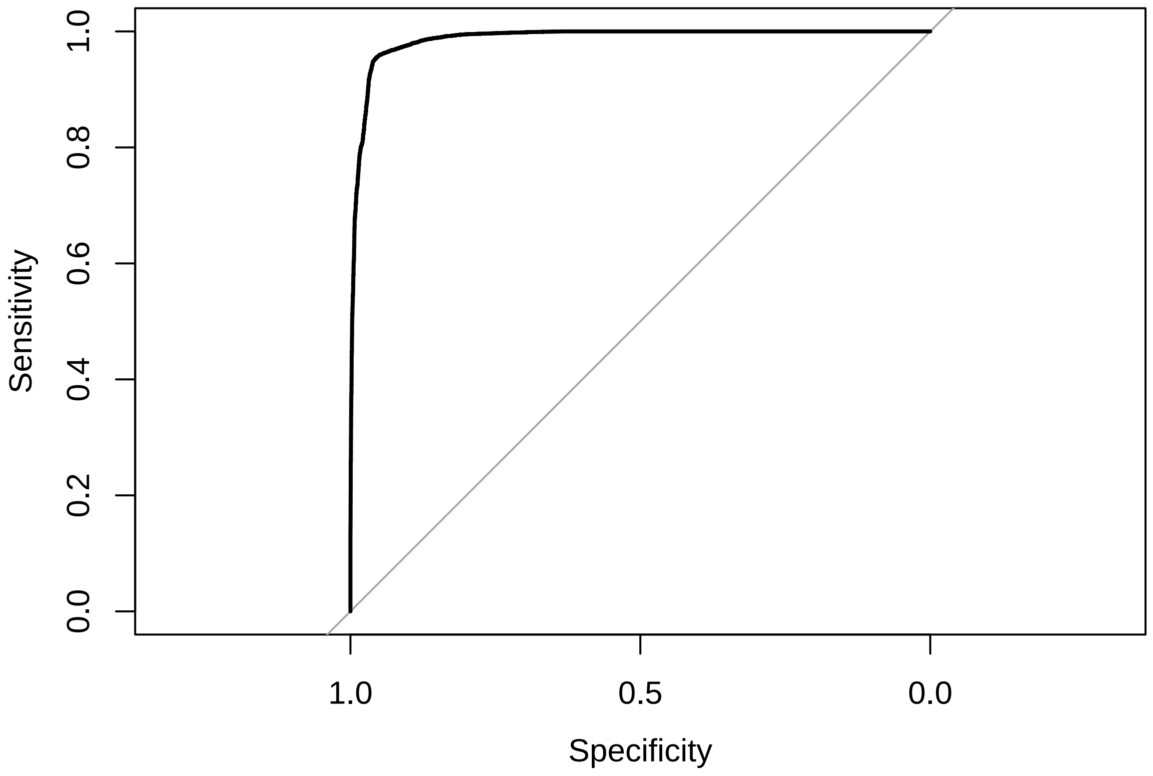
```
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 0.2 and lambda = 0.01.
```

```
testResults$PLR <- predict(plr_model, test[, -1])
```

```
# ROC curve for penalized log reg
plr_roc <- roc(response = plr_model$pred$obs,
               predictor = plr_model$pred$M,
               levels = rev(levels(plr_model$pred$obs)))
```

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

```
plot(plr_roc, legaces.axes = TRUE)
```



```
# Confusion Matrix
confusionMatrix(testResults$PLR, testResults$obs, positive = "M")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 70  1
##          M  1 41
##
##                Accuracy : 0.9823
##                  95% CI : (0.9375, 0.9978)
##     No Information Rate : 0.6283
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9621
##
```

```
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9762
##             Specificity : 0.9859
##          Pos Pred Value : 0.9762
##          Neg Pred Value : 0.9859
##              Prevalence : 0.3717
##          Detection Rate : 0.3628
##    Detection Prevalence : 0.3717
##       Balanced Accuracy : 0.9811
##
##        'Positive' Class : M
##
```

```
# LDA
set.seed(123)
lda_model <- train(x = train[, -1],
                   y = train$diagnosis,
                   method = "lda",
                   preProcess = c("center", "scale"),
                   metric = "ROC",
                   trControl = ctrl)
lda_model
```

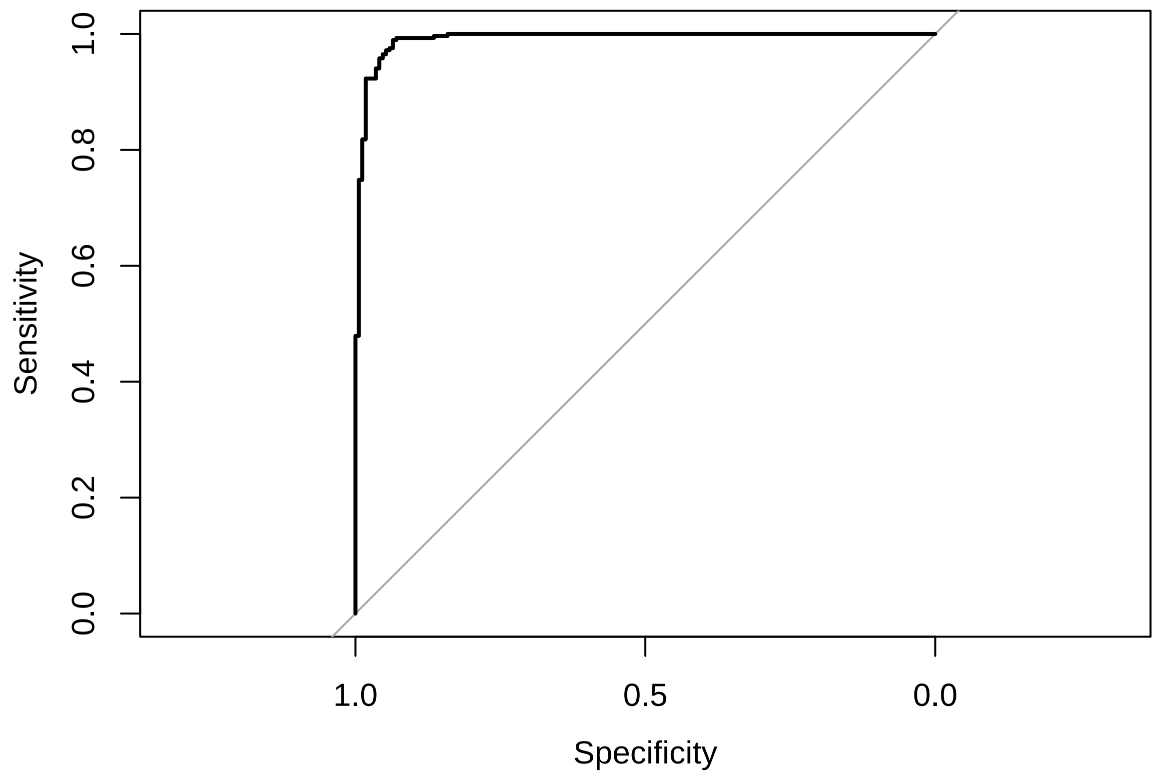
```
## Linear Discriminant Analysis
##
## 456 samples
##  26 predictor
##   2 classes: 'B', 'M'
##
## Pre-processing: centered (26), scaled (26)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 410, 410, 411, 411, 410, 411, ...
## Resampling results:
##
##   ROC        Sens       Spec
##   0.9930745  0.9929803  0.8941176
```

```
testResults$LDA <- predict(lda_model, test[, -1])
```

```
# ROC curve for LDA
lda_roc <- roc(response = lda_model$pred$obs,
              predictor = lda_model$pred$M,
              levels = rev(levels(lda_model$pred$obs)))
```

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

```
plot(lda_roc, legaces.axes = TRUE)
```



```r
# Confusion Matrix
confusionMatrix(testResults$LDA, testResults$obs, positive = "M")
```

```
## 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
```

```
##
```

```r
# PLSDA
plsGrid <- expand.grid(ncomp = 1:20)

set.seed(123)
pls_model <- train(x = train[, -1],
                   y = train$diagnosis,
                   method = "pls",
                   tuneGrid = plsGrid,
                   preProcess = c("center", "scale"),
                   metric = "ROC",
                   trControl = ctrl)

pls_model
```

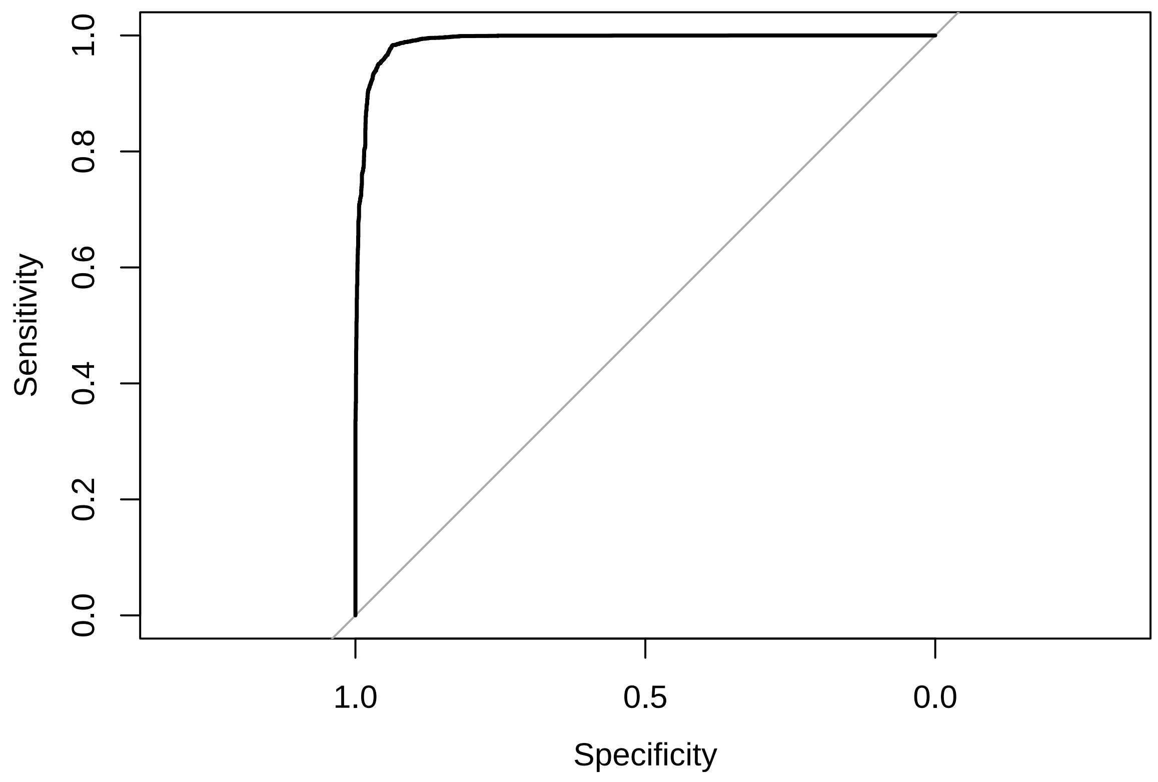
```
## Partial Least Squares
##
## 456 samples
##  26 predictor
##   2 classes: 'B', 'M'
##
## Pre-processing: centered (26), scaled (26)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 410, 410, 411, 411, 410, 411, ...
## Resampling results across tuning parameters:
##
##   ncomp  ROC        Sens       Spec
##    1     0.9797015  0.9685961  0.8588235
##    2     0.9887569  0.9757389  0.9000000
##    3     0.9880107  0.9929803  0.9176471
##    4     0.9939076  0.9965517  0.8941176
##    5     0.9939003  0.9929803  0.9058824
##    6     0.9918574  0.9965517  0.9058824
##    7     0.9922704  0.9894089  0.9058824
##    8     0.9932701  0.9929803  0.8941176
##    9     0.9928644  1.0000000  0.8941176
##   10     0.9930817  1.0000000  0.9000000
##   11     0.9916618  0.9964286  0.8882353
##   12     0.9924732  0.9928571  0.8941176
##   13     0.9904158  0.9894089  0.8941176
##   14     0.9922704  0.9928571  0.9058824
##   15     0.9916691  0.9894089  0.9000000
##   16     0.9924804  0.9929803  0.9058824
##   17     0.9941104  0.9929803  0.8941176
##   18     0.9943060  0.9929803  0.8941176
##   19     0.9945161  0.9929803  0.8941176
##   20     0.9932918  0.9929803  0.8882353
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was ncomp = 19.
```

```r
testResults$PLS <- predict(pls_model, test[, -1])
```

```r
# ROC curve for PLS
pls_roc <- roc(response = pls_model$pred$obs,
               predictor = pls_model$pred$M,
               levels = rev(levels(pls_model$pred$obs)))
```

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

```r
plot(pls_roc, legaces.axes = TRUE)
```



```r
# Confusion Matrix
confusionMatrix(testResults$PLS, testResults$obs, positive = "M")
```

```
## 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
##
```

```
# MDA
mdaGrid <- expand.grid(subclasses = 1:3)

set.seed(123)
mda_model <- train(x = train[, -1],
                   y = train$diagnosis,
                   method = "mda",
                   tuneGrid = mdaGrid,
                   preProcess = c("center", "scale"),
                   metric = "ROC",
                   trControl = ctrl)
mda_model
```

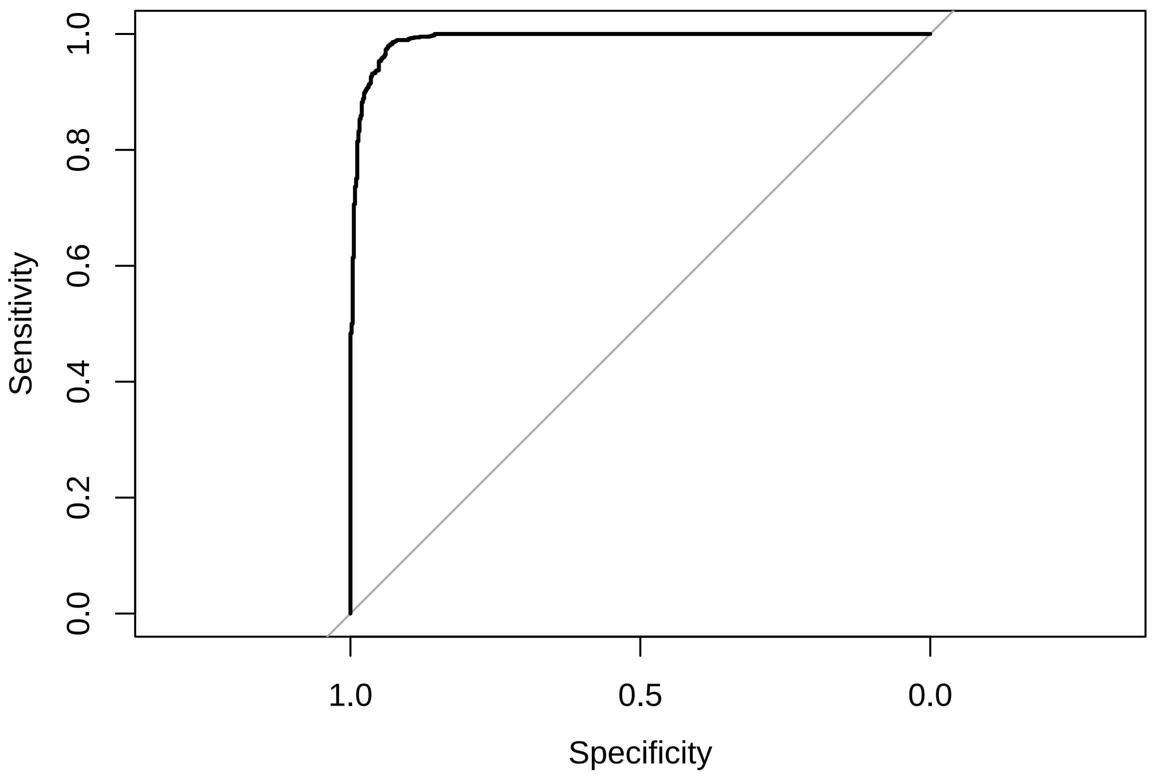
```
## Mixture Discriminant Analysis
##
## 456 samples
##  26 predictor
##   2 classes: 'B', 'M'
##
## Pre-processing: centered (26), scaled (26)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 410, 410, 411, 411, 410, 411, ...
## Resampling results across tuning parameters:
##
##   subclasses  ROC        Sens       Spec
##   1           0.9930745  0.9929803  0.8941176
##   2           0.9895972  0.9929803  0.9000000
##   3           0.9949073  0.9894089  0.9058824
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was subclasses = 3.
```

```
testResults$MDA <- predict(mda_model, test[, -1])
```

```
# ROC curve for MDA
mda_roc <- roc(response = mda_model$pred$obs,
               predictor = mda_model$pred$M,
               levels = rev(levels(mda_model$pred$obs)))
```

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

```
plot(mda_roc, legaces.axes = TRUE)
```



```r
# Confusion Matrix
confusionMatrix(testResults$MDA, testResults$obs, positive = "M")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 70  1
##          M  1 41
##
##                Accuracy : 0.9823
##                  95% CI : (0.9375, 0.9978)
##     No Information Rate : 0.6283
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9621
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9762
##             Specificity : 0.9859
##          Pos Pred Value : 0.9762
##          Neg Pred Value : 0.9859
##              Prevalence : 0.3717
##          Detection Rate : 0.3628
##    Detection Prevalence : 0.3717
##       Balanced Accuracy : 0.9811
##
##        'Positive' Class : M
```

```
##
```

```r
# Decision Trees
set.seed(123)
rpart_model <- train(x = train[, -1],
                     y = train$diagnosis,
                     method = "rpart",
                     tuneLength = 30,
                     preProcess = c("center", "scale"),
                     metric = "ROC",
                     trControl = ctrl)
rpart_model
```

```
## CART
##
## 456 samples
##  26 predictor
##   2 classes: 'B', 'M'
##
## Pre-processing: centered (26), scaled (26)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 410, 410, 411, 411, 410, 411, ...
## Resampling results across tuning parameters:
##
##   cp          ROC        Sens       Spec
##   0.00000000  0.9395791  0.9301724  0.8882353
##   0.02778905  0.9162743  0.9334975  0.8764706
##   0.05557809  0.9166872  0.9369458  0.8941176
##   0.08336714  0.9143763  0.9405172  0.8882353
##   0.11115619  0.9143763  0.9405172  0.8882353
##   0.13894523  0.9143763  0.9405172  0.8882353
##   0.16673428  0.9143763  0.9405172  0.8882353
##   0.19452333  0.9143763  0.9405172  0.8882353
##   0.22231237  0.9143763  0.9405172  0.8882353
##   0.25010142  0.9143763  0.9405172  0.8882353
##   0.27789047  0.9143763  0.9405172  0.8882353
##   0.30567951  0.9143763  0.9405172  0.8882353
##   0.33346856  0.9143763  0.9405172  0.8882353
##   0.36125761  0.9143763  0.9405172  0.8882353
##   0.38904665  0.9143763  0.9405172  0.8882353
##   0.41683570  0.9143763  0.9405172  0.8882353
##   0.44462475  0.9143763  0.9405172  0.8882353
##   0.47241379  0.9143763  0.9405172  0.8882353
##   0.50020284  0.9143763  0.9405172  0.8882353
##   0.52799189  0.9143763  0.9405172  0.8882353
##   0.55578093  0.9143763  0.9405172  0.8882353
##   0.58356998  0.9143763  0.9405172  0.8882353
##   0.61135903  0.9143763  0.9405172  0.8882353
##   0.63914807  0.9143763  0.9405172  0.8882353
##   0.66693712  0.9143763  0.9405172  0.8882353
##   0.69472617  0.9143763  0.9405172  0.8882353
##   0.72251521  0.9143763  0.9405172  0.8882353
##   0.75030426  0.9143763  0.9405172  0.8882353
##   0.77809331  0.9143763  0.9405172  0.8882353
##   0.80588235  0.6529774  0.9647783  0.3411765
```

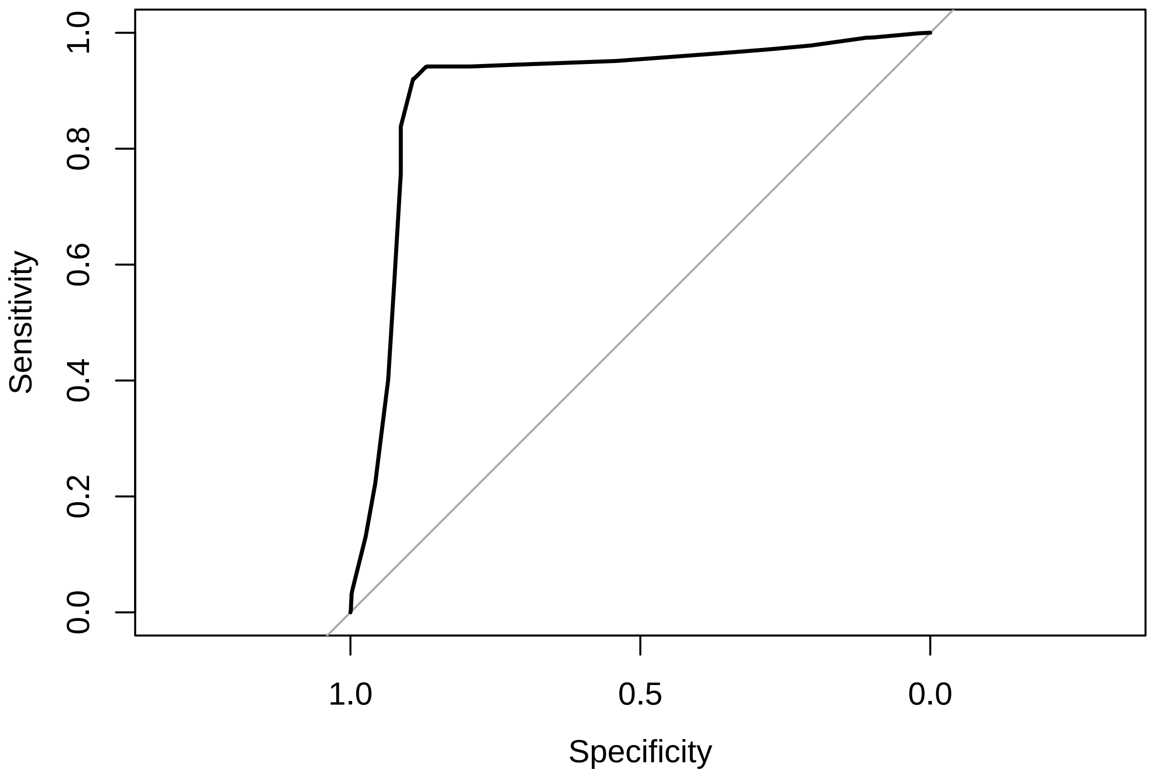
```
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.
```

```
testResults$DecisionTree <- predict(rpart_model, test[, -1])
```

```
# ROC curve for decision trees
rpart_roc <- roc(response = rpart_model$pred$obs,
              predictor = rpart_model$pred$M,
              levels = rev(levels(rpart_model$pred$obs)))
```

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

```
plot(rpart_roc, legaces.axes = TRUE)
```



```
# Confusion Matrix
confusionMatrix(testResults$DecisionTree, testResults$obs, positive = "M")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 64  5
##          M  7 37
##
##                Accuracy : 0.8938
##                  95% CI : (0.8218, 0.9439)
##     No Information Rate : 0.6283
##     P-Value [Acc > NIR] : 1.762e-10
##
##                   Kappa : 0.7748
```

```
##
##  Mcnemar's Test P-Value : 0.7728
##
##             Sensitivity : 0.8810
##             Specificity : 0.9014
##          Pos Pred Value : 0.8409
##          Neg Pred Value : 0.9275
##              Prevalence : 0.3717
##          Detection Rate : 0.3274
##    Detection Prevalence : 0.3894
##       Balanced Accuracy : 0.8912
##
##        'Positive' Class : M
##
```

```
# Random Forest
mtryValues <- seq(1, 10, 1)
rfGrid <- data.frame(mtry = mtryValues)

set.seed(123)
rf_model <- train(x = train[, -1],
                  y = train$diagnosis,
                  method = "rf",
                  ntree = 1000,
                  preProcess = c("center", "scale"),
                  tuneGrid = rfGrid,
                  metric = "ROC",
                  trControl = ctrl)
rf_model
```

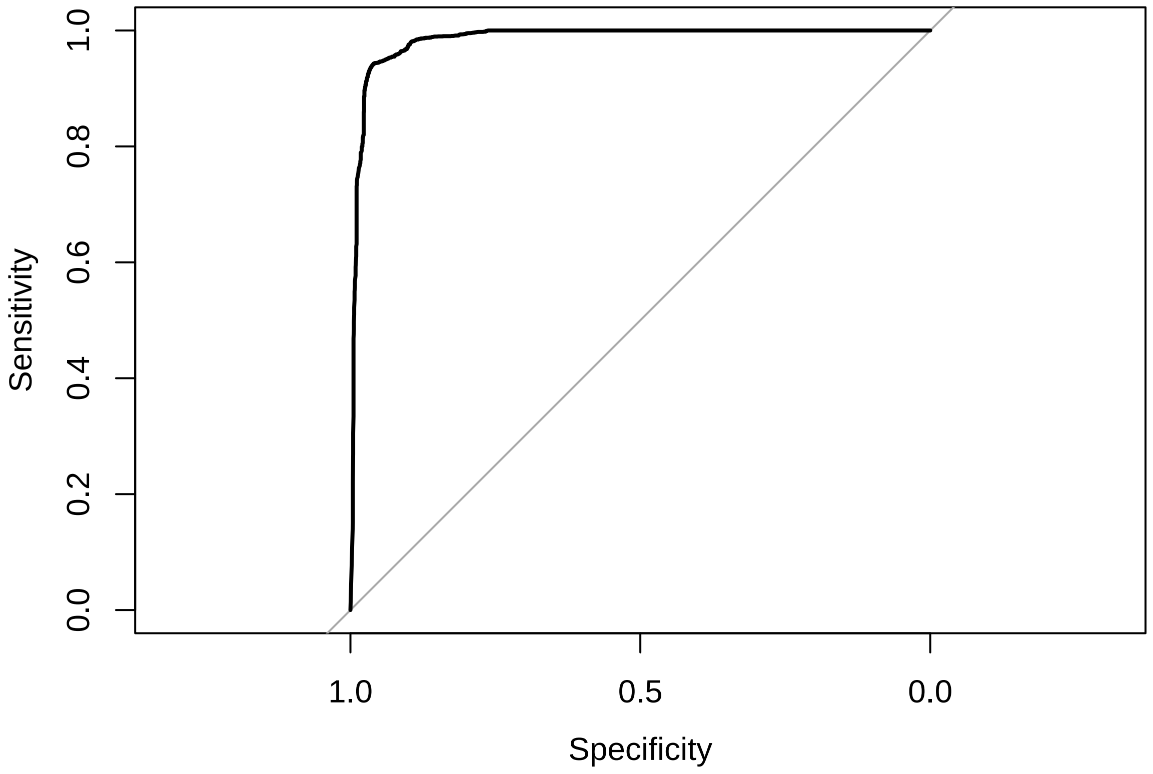
```
## Random Forest
##
## 456 samples
##  26 predictor
##   2 classes: 'B', 'M'
##
## Pre-processing: centered (26), scaled (26)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 410, 410, 411, 411, 410, 411, ...
## Resampling results across tuning parameters:
##
##   mtry  ROC        Sens       Spec
##    1    0.9881339  0.9580049  0.9235294
##    2    0.9869168  0.9545567  0.9235294
##    3    0.9886410  0.9580049  0.9235294
##    4    0.9867068  0.9509852  0.9294118
##    5    0.9872139  0.9545567  0.9176471
##    6    0.9872102  0.9545567  0.9294118
##    7    0.9869168  0.9545567  0.9294118
##    8    0.9873153  0.9580049  0.9235294
##    9    0.9874167  0.9545567  0.9235294
##   10    0.9872139  0.9511084  0.9176471
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 3.
```

```r
testResults$RF <- predict(rf_model, test[, -1])
```

```r
# ROC curve for random forest
rf_roc <- roc(response = rf_model$pred$obs,
              predictor = rf_model$pred$M,
              levels = rev(levels(rf_model$pred$obs)))
```

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

```r
plot(rf_roc, legaces.axes = TRUE)
```



```r
# Confusion Matrix
confusionMatrix(testResults$RF, testResults$obs, positive = "M")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 68  3
##          M  3 39
##
##                Accuracy : 0.9469
##                  95% CI : (0.888, 0.9803)
##     No Information Rate : 0.6283
##     P-Value [Acc > NIR] : 1.866e-15
##
##                   Kappa : 0.8863
##
##  Mcnemar's Test P-Value : 1
##
```

```
##             Sensitivity : 0.9286
##             Specificity : 0.9577
##          Pos Pred Value : 0.9286
##          Neg Pred Value : 0.9577
##              Prevalence : 0.3717
##          Detection Rate : 0.3451
##    Detection Prevalence : 0.3717
##       Balanced Accuracy : 0.9432
##
##        'Positive' Class : M
##
```

```
# Boosted Trees
gbmGrid <-expand.grid(interaction.depth = c(1, 3, 5, 7, 9),
                      n.trees = (1:20)*100,
                      shrinkage = c(.01, .1),
                      n.minobsinnode = 5)

set.seed(123)
gbm_model <- train(x = train[, -1],
                   y = train$diagnosis,
                   method = "gbm",
                   preProcess = c("center", "scale"),
                   tuneGrid = gbmGrid,
                   verbose = FALSE,
                   metric = "ROC",
                   trControl = ctrl)
gbm_model
```

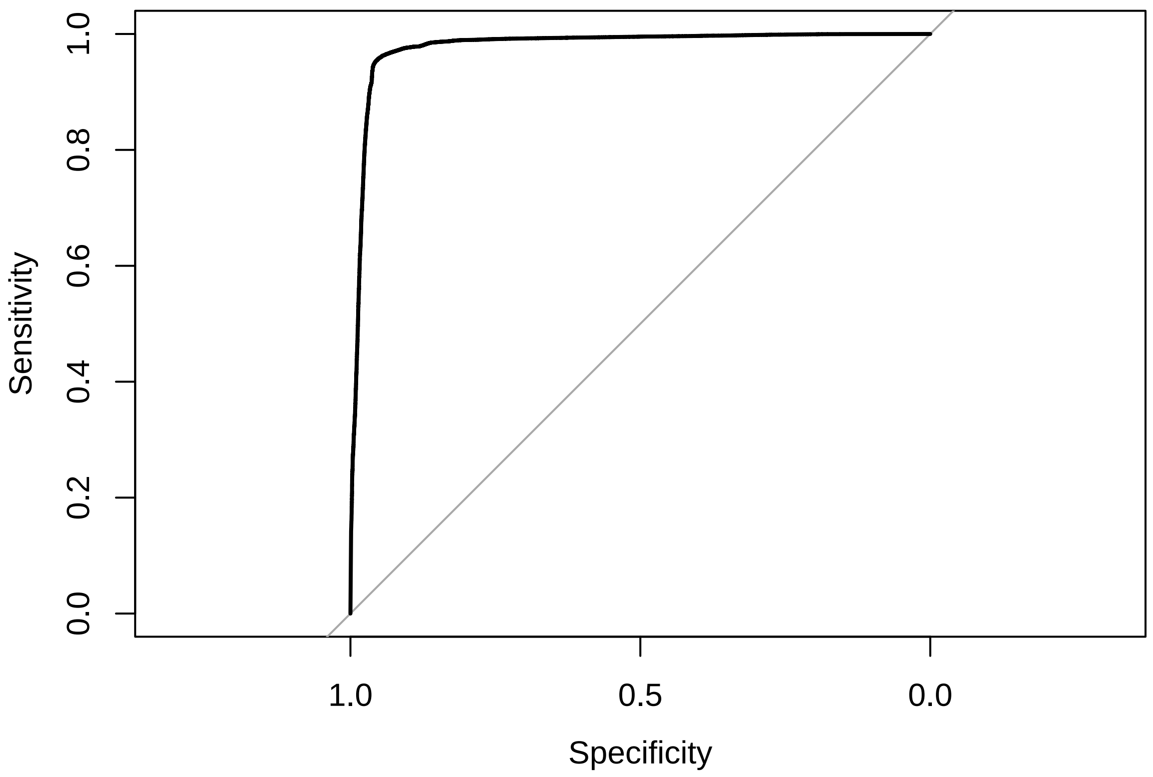
```
## Stochastic Gradient Boosting
##
## 456 samples
##  26 predictor
##   2 classes: 'B', 'M'
##
## Pre-processing: centered (26), scaled (26)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 410, 410, 411, 411, 410, 411, ...
## Resampling results across tuning parameters:
##
##   shrinkage  interaction.depth  n.trees  ROC        Sens       Spec
##   0.01       1                   100     0.9782599  0.9545567  0.8529412
##   0.01       1                   200     0.9838670  0.9509852  0.8705882
##   0.01       1                   300     0.9859461  0.9509852  0.8823529
##   0.01       1                   400     0.9873660  0.9651478  0.9000000
##   0.01       1                   500     0.9877644  0.9651478  0.9176471
##   0.01       1                   600     0.9889815  0.9615764  0.9176471
##   0.01       1                   700     0.9891915  0.9651478  0.9176471
##   0.01       1                   800     0.9898073  0.9615764  0.9176471
##   0.01       1                   900     0.9904158  0.9615764  0.9176471
##   0.01       1                  1000     0.9906114  0.9615764  0.9235294
##   0.01       1                  1100     0.9904086  0.9615764  0.9294118
##   0.01       1                  1200     0.9908143  0.9650246  0.9235294
##   0.01       1                  1300     0.9906042  0.9650246  0.9294118
##   0.01       1                  1400     0.9906042  0.9650246  0.9352941
##   0.01       1                    1500     0.9901912  0.9650246  0.9352941
##   0.01       1                    1600     0.9901912  0.9650246  0.9352941
##   0.01       1                    1700     0.9908070  0.9684729  0.9352941
##   0.01       1                    1800     0.9906042  0.9684729  0.9352941
##   0.01       1                    1900     0.9910171  0.9684729  0.9352941
##   0.01       1                    2000     0.9914155  0.9684729  0.9352941
##   0.01       3                     100     0.9869023  0.9650246  0.8882353
##   0.01       3                     200     0.9889597  0.9615764  0.9117647
##   0.01       3                     300     0.9899957  0.9650246  0.9235294
##   0.01       3                     400     0.9910171  0.9614532  0.9294118
##   0.01       3                     500     0.9907998  0.9578818  0.9294118
##   0.01       3                     600     0.9910026  0.9578818  0.9352941
##   0.01       3                     700     0.9914155  0.9578818  0.9411765
##   0.01       3                     800     0.9916256  0.9614532  0.9411765
##   0.01       3                     900     0.9920313  0.9650246  0.9411765
##   0.01       3                    1000     0.9922414  0.9650246  0.9411765
##   0.01       3                    1100     0.9926543  0.9650246  0.9411765
##   0.01       3                    1200     0.9928499  0.9650246  0.9411765
##   0.01       3                    1300     0.9932556  0.9650246  0.9411765
##   0.01       3                    1400     0.9932483  0.9684729  0.9411765
##   0.01       3                    1500     0.9930382  0.9684729  0.9411765
##   0.01       3                    1600     0.9928354  0.9684729  0.9411765
##   0.01       3                    1700     0.9930455  0.9684729  0.9411765
##   0.01       3                    1800     0.9932483  0.9684729  0.9352941
##   0.01       3                    1900     0.9932483  0.9684729  0.9352941
##   0.01       3                    2000     0.9932483  0.9684729  0.9352941
##   0.01       5                     100     0.9840771  0.9721675  0.9058824
##   0.01       5                     200     0.9883512  0.9650246  0.9235294
##   0.01       5                     300     0.9895827  0.9614532  0.9294118
##   0.01       5                     400     0.9908070  0.9650246  0.9294118
##   0.01       5                     500     0.9912127  0.9684729  0.9411765
##   0.01       5                     600     0.9922414  0.9614532  0.9352941
##   0.01       5                     700     0.9922414  0.9650246  0.9352941
##   0.01       5                     800     0.9926471  0.9650246  0.9411765
##   0.01       5                     900     0.9928499  0.9650246  0.9411765
##   0.01       5                    1000     0.9926471  0.9650246  0.9411765
##   0.01       5                    1100     0.9926471  0.9650246  0.9411765
##   0.01       5                    1200     0.9928499  0.9615764  0.9352941
##   0.01       5                    1300     0.9928499  0.9615764  0.9352941
##   0.01       5                    1400     0.9930455  0.9615764  0.9352941
##   0.01       5                    1500     0.9932483  0.9615764  0.9352941
##   0.01       5                    1600     0.9932483  0.9684729  0.9352941
##   0.01       5                    1700     0.9934512  0.9684729  0.9352941
##   0.01       5                    1800     0.9930455  0.9684729  0.9352941
##   0.01       5                    1900     0.9932483  0.9684729  0.9352941
##   0.01       5                    2000     0.9934512  0.9684729  0.9352941
##   0.01       7                     100     0.9859751  0.9685961  0.9058824
##   0.01       7                     200     0.9895465  0.9685961  0.9235294
##   0.01       7                     300     0.9915966  0.9685961  0.9352941
##   0.01       7                     400     0.9915966  0.9614532  0.9294118
##   0.01       7                     500     0.9922124  0.9614532  0.9294118
##   0.01       7                     600     0.9920023  0.9614532  0.9352941
##   0.01       7                     700     0.9926253  0.9614532  0.9411765
##   0.01       7                     800     0.9924225  0.9650246  0.9235294
##   0.01       7                         900     0.9926326  0.9615764  0.9294118
##   0.01       7                        1000     0.9926326  0.9615764  0.9294118
##   0.01       7                        1100     0.9926326  0.9615764  0.9294118
##   0.01       7                        1200     0.9930382  0.9615764  0.9352941
##   0.01       7                        1300     0.9930382  0.9615764  0.9411765
##   0.01       7                        1400     0.9934512  0.9615764  0.9411765
##   0.01       7                        1500     0.9936540  0.9615764  0.9411765
##   0.01       7                        1600     0.9936540  0.9650246  0.9411765
##   0.01       7                        1700     0.9934512  0.9650246  0.9411765
##   0.01       7                        1800     0.9934512  0.9650246  0.9411765
##   0.01       7                        1900     0.9934512  0.9684729  0.9411765
##   0.01       7                        2000     0.9932483  0.9684729  0.9411765
##   0.01       9                         100     0.9881121  0.9685961  0.9000000
##   0.01       9                         200     0.9889597  0.9651478  0.9235294
##   0.01       9                         300     0.9916111  0.9650246  0.9294118
##   0.01       9                         400     0.9916039  0.9650246  0.9294118
##   0.01       9                         500     0.9913938  0.9650246  0.9294118
##   0.01       9                         600     0.9915966  0.9614532  0.9294118
##   0.01       9                         700     0.9922124  0.9580049  0.9294118
##   0.01       9                         800     0.9917995  0.9580049  0.9294118
##   0.01       9                         900     0.9922052  0.9580049  0.9294118
##   0.01       9                        1000     0.9921979  0.9615764  0.9352941
##   0.01       9                        1100     0.9921979  0.9615764  0.9352941
##   0.01       9                        1200     0.9930310  0.9615764  0.9352941
##   0.01       9                        1300     0.9928209  0.9615764  0.9294118
##   0.01       9                        1400     0.9926181  0.9615764  0.9352941
##   0.01       9                        1500     0.9928209  0.9615764  0.9352941
##   0.01       9                        1600     0.9930238  0.9615764  0.9352941
##   0.01       9                        1700     0.9928209  0.9650246  0.9352941
##   0.01       9                        1800     0.9928209  0.9650246  0.9294118
##   0.01       9                        1900     0.9930310  0.9650246  0.9352941
##   0.01       9                        2000     0.9924152  0.9684729  0.9352941
##   0.10       1                         100     0.9895827  0.9615764  0.9352941
##   0.10       1                         200     0.9895755  0.9615764  0.9411765
##   0.10       1                         300     0.9920385  0.9719212  0.9411765
##   0.10       1                         400     0.9930527  0.9719212  0.9411765
##   0.10       1                         500     0.9932628  0.9719212  0.9352941
##   0.10       1                         600     0.9924442  0.9753695  0.9294118
##   0.10       1                         700     0.9926543  0.9753695  0.9294118
##   0.10       1                         800     0.9922341  0.9753695  0.9294118
##   0.10       1                         900     0.9918285  0.9753695  0.9294118
##   0.10       1                        1000     0.9922341  0.9753695  0.9294118
##   0.10       1                        1100     0.9924370  0.9753695  0.9294118
##   0.10       1                        1200     0.9922414  0.9753695  0.9294118
##   0.10       1                        1300     0.9924442  0.9753695  0.9294118
##   0.10       1                        1400     0.9920241  0.9753695  0.9294118
##   0.10       1                        1500     0.9924442  0.9753695  0.9352941
##   0.10       1                        1600     0.9920313  0.9753695  0.9411765
##   0.10       1                        1700     0.9916329  0.9753695  0.9352941
##   0.10       1                        1800     0.9918357  0.9753695  0.9352941
##   0.10       1                        1900     0.9922414  0.9753695  0.9352941
##   0.10       1                        2000     0.9920313  0.9753695  0.9352941
##   0.10       3                         100     0.9924152  0.9719212  0.9352941
##   0.10       3                         200     0.9924225  0.9720443  0.9352941
##    0.10        3                     300    0.9930382  0.9754926  0.9294118
##    0.10        3                     400    0.9934512  0.9754926  0.9352941
##    0.10        3                     500    0.9932338  0.9719212  0.9294118
##    0.10        3                     600    0.9930310  0.9754926  0.9352941
##    0.10        3                     700    0.9928282  0.9719212  0.9352941
##    0.10        3                     800    0.9926326  0.9754926  0.9411765
##    0.10        3                     900    0.9936540  0.9719212  0.9411765
##    0.10        3                    1000    0.9936540  0.9719212  0.9411765
##    0.10        3                    1100    0.9932411  0.9684729  0.9352941
##    0.10        3                    1200    0.9930382  0.9684729  0.9352941
##    0.10        3                    1300    0.9926326  0.9684729  0.9352941
##    0.10        3                    1400    0.9926326  0.9684729  0.9352941
##    0.10        3                    1500    0.9926326  0.9684729  0.9352941
##    0.10        3                    1600    0.9926326  0.9684729  0.9352941
##    0.10        3                    1700    0.9924225  0.9684729  0.9352941
##    0.10        3                    1800    0.9922124  0.9684729  0.9352941
##    0.10        3                    1900    0.9903868  0.9684729  0.9411765
##    0.10        3                    2000    0.9908070  0.9684729  0.9352941
##    0.10        5                     100    0.9901478  0.9614532  0.9411765
##    0.10        5                     200    0.9916039  0.9719212  0.9235294
##    0.10        5                     300    0.9914010  0.9684729  0.9235294
##    0.10        5                     400    0.9924152  0.9684729  0.9294118
##    0.10        5                     500    0.9926108  0.9684729  0.9294118
##    0.10        5                     600    0.9924080  0.9684729  0.9294118
##    0.10        5                     700    0.9928209  0.9684729  0.9294118
##    0.10        5                     800    0.9928137  0.9684729  0.9294118
##    0.10        5                     900    0.9928137  0.9684729  0.9352941
##    0.10        5                    1000    0.9928137  0.9684729  0.9411765
##    0.10        5                    1100    0.9928137  0.9719212  0.9352941
##    0.10        5                    1200    0.9929151  0.9719212  0.9294118
##    0.10        5                    1300    0.9908867  0.9719212  0.9352941
##    0.10        5                    1400    0.9908867  0.9719212  0.9352941
##    0.10        5                    1500    0.9912924  0.9719212  0.9411765
##    0.10        5                    1600    0.9889597  0.9719212  0.9411765
##    0.10        5                    1700    0.9915966  0.9684729  0.9411765
##    0.10        5                    1800    0.9915025  0.9684729  0.9470588
##    0.10        5                    1900    0.9915025  0.9684729  0.9411765
##    0.10        5                    2000    0.9917270  0.9684729  0.9411765
##    0.10        7                     100    0.9909591  0.9580049  0.9235294
##    0.10        7                     200    0.9917488  0.9615764  0.9235294
##    0.10        7                     300    0.9921689  0.9719212  0.9235294
##    0.10        7                     400    0.9923790  0.9719212  0.9235294
##    0.10        7                     500    0.9926036  0.9719212  0.9176471
##    0.10        7                     600    0.9928137  0.9719212  0.9176471
##    0.10        7                     700    0.9922052  0.9719212  0.9176471
##    0.10        7                     800    0.9930310  0.9719212  0.9176471
##    0.10        7                     900    0.9931324  0.9719212  0.9235294
##    0.10        7                    1000    0.9909954  0.9719212  0.9235294
##    0.10        7                    1100    0.9889742  0.9719212  0.9235294
##    0.10        7                    1200    0.9889815  0.9719212  0.9235294
##    0.10        7                    1300    0.9893871  0.9719212  0.9235294
##    0.10        7                    1400    0.9916256  0.9719212  0.9235294
##    0.10        7                    1500    0.9915314  0.9754926  0.9235294
##    0.10        7                    1600    0.9917415  0.9720443  0.9294118
##   0.10       7                    1700      0.9907346  0.9685961  0.9352941
##   0.10       7                    1800      0.9930672  0.9651478  0.9470588
##   0.10       7                    1900      0.9926615  0.9651478  0.9470588
##   0.10       7                    2000      0.9924587  0.9616995  0.9470588
##   0.10       9                     100      0.9901550  0.9650246  0.9235294
##   0.10       9                     200      0.9920023  0.9650246  0.9411765
##   0.10       9                     300      0.9928282  0.9650246  0.9411765
##   0.10       9                     400      0.9932411  0.9685961  0.9470588
##   0.10       9                     500      0.9930455  0.9720443  0.9411765
##   0.10       9                     600      0.9930600  0.9720443  0.9529412
##   0.10       9                     700      0.9930672  0.9720443  0.9411765
##   0.10       9                     800      0.9923573  0.9720443  0.9352941
##   0.10       9                     900      0.9909374  0.9720443  0.9352941
##   0.10       9                    1000      0.9892133  0.9720443  0.9352941
##   0.10       9                    1100      0.9890032  0.9720443  0.9411765
##   0.10       9                    1200      0.9894089  0.9685961  0.9411765
##   0.10       9                    1300      0.9890104  0.9687192  0.9411765
##   0.10       9                    1400      0.9885975  0.9651478  0.9470588
##   0.10       9                    1500      0.9906259  0.9651478  0.9529412
##   0.10       9                    1600      0.9911475  0.9651478  0.9588235
##   0.10       9                    1700      0.9904303  0.9651478  0.9588235
##   0.10       9                    1800      0.9908432  0.9616995  0.9647059
##   0.10       9                    1900      0.9910533  0.9616995  0.9647059
##   0.10       9                    2000      0.9906476  0.9581281  0.9647059
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 5
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 900, interaction.depth =
##  3, shrinkage = 0.1 and n.minobsinnode = 5.
```

```
testResults$BoostedTree <- predict(gbm_model, test[, -1])
```

```
# ROC curve for boosted trees
gbm_roc <- roc(response = gbm_model$pred$obs,
               predictor = gbm_model$pred$M,
               levels = rev(levels(gbm_model$pred$obs)))
```

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

```
plot(gbm_roc, legaces.axes = TRUE)
```



```r
# Confusion Matrix
confusionMatrix(testResults$BoostedTree, testResults$obs, positive = "M")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 71  1
##          M  0 41
##
##                Accuracy : 0.9912
##                  95% CI : (0.9517, 0.9998)
##     No Information Rate : 0.6283
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.981
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9762
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9861
##              Prevalence : 0.3717
##          Detection Rate : 0.3628
##    Detection Prevalence : 0.3628
##       Balanced Accuracy : 0.9881
##
##        'Positive' Class : M
```

```
##
```

```r
# KNN
set.seed(123)
knn_model <- train(x = train[, -1],
                   y = train$diagnosis,
                   method = "knn",
                   preProcess = c("center", "scale"),
                   tuneLength = 20,
                   metric = "ROC",
                   trControl = ctrl)
knn_model
```

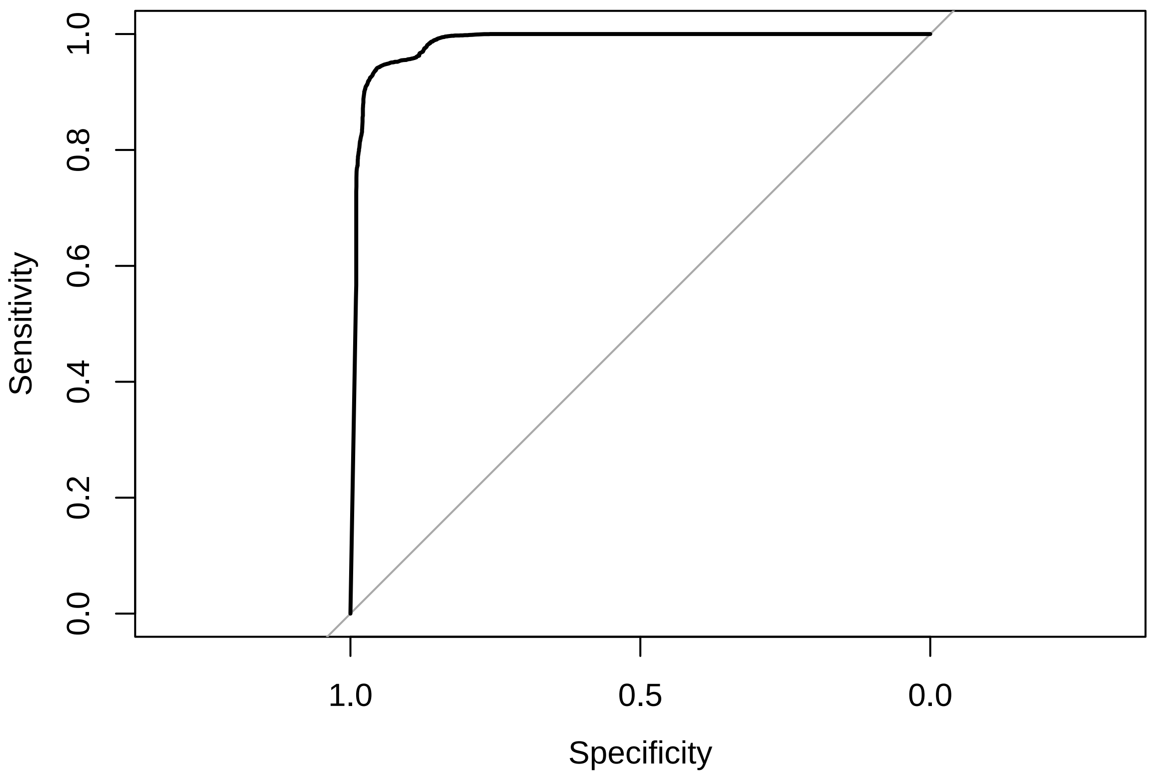
```
## k-Nearest Neighbors
##
## 456 samples
##  26 predictor
##   2 classes: 'B', 'M'
##
## Pre-processing: centered (26), scaled (26)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 410, 410, 411, 411, 410, 411, ...
## Resampling results across tuning parameters:
##
##   k   ROC        Sens       Spec
##    5  0.9867611  0.9791872  0.9058824
##    7  0.9863590  0.9720443  0.9235294
##    9  0.9854390  0.9651478  0.9058824
##   11  0.9847146  0.9687192  0.8941176
##   13  0.9840988  0.9651478  0.8882353
##   15  0.9858881  0.9685961  0.8941176
##   17  0.9854970  0.9582512  0.8941176
##   19  0.9850949  0.9616995  0.8941176
##   21  0.9852941  0.9651478  0.8882353
##   23  0.9882353  0.9650246  0.8764706
##   25  0.9872139  0.9650246  0.8705882
##   27  0.9868045  0.9546798  0.8705882
##   29  0.9872102  0.9546798  0.8647059
##   31  0.9871124  0.9581281  0.8647059
##   33  0.9868118  0.9616995  0.8647059
##   35  0.9865003  0.9616995  0.8647059
##   37  0.9857759  0.9616995  0.8647059
##   39  0.9853702  0.9651478  0.8647059
##   41  0.9875072  0.9685961  0.8647059
##   43  0.9876159  0.9685961  0.8647059
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 23.
```

```r
testResults$KNN <- predict(knn_model, test[, -1])
```

```r
# ROC curve for KNN
knn_roc <- roc(response = knn_model$pred$obs,
              predictor = knn_model$pred$M,
              levels = rev(levels(knn_model$pred$obs)))
```

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

```r
plot(knn_roc, legaces.axes = TRUE)
```



```r
# Confusion Matrix
confusionMatrix(testResults$KNN, testResults$obs, positive = "M")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 69  2
##          M  2 40
##
##                Accuracy : 0.9646
##                  95% CI : (0.9118, 0.9903)
##     No Information Rate : 0.6283
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9242
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9524
##             Specificity : 0.9718
##          Pos Pred Value : 0.9524
##          Neg Pred Value : 0.9718
##              Prevalence : 0.3717
##          Detection Rate : 0.3540
##    Detection Prevalence : 0.3717
##       Balanced Accuracy : 0.9621
##
##        'Positive' Class : M
##
```

```
# Neural Network Model
nnetGrid <- expand.grid(size = 1:2, decay = c(0, .1, .2, .3, .4, .5, 1))

set.seed(123)
nnet_model <- train(x = train[, -1],
                    y = train$diagnosis,
                    method = "nnet",
                    preProcess = c("center", "scale"),
                    tuneGrid = nnetGrid,
                    metric = "ROC",
                    linout = FALSE,
                    trace = FALSE,
                    maxit = 1000,
                    trControl = ctrl)
nnet_model
```

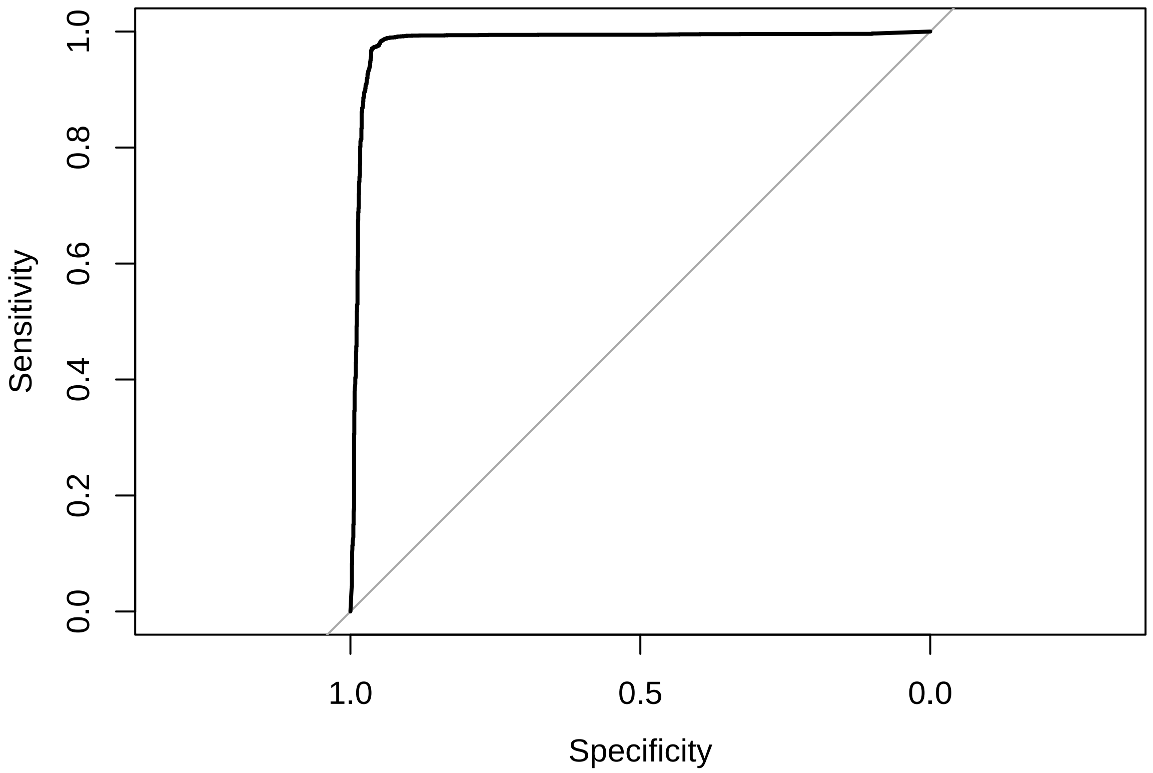
```
## Neural Network
##
## 456 samples
##  26 predictor
##   2 classes: 'B', 'M'
##
## Pre-processing: centered (26), scaled (26)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 410, 410, 411, 411, 410, 411, ...
## Resampling results across tuning parameters:
##
##   size  decay  ROC        Sens       Spec
##   1     0.0    0.9691937  0.9757389  0.9470588
##   1     0.1    0.9930310  0.9753695  0.9588235
##   1     0.2    0.9944726  0.9823892  0.9529412
##   1     0.3    0.9948855  0.9823892  0.9470588
##   1     0.4    0.9953057  0.9858374  0.9470588
##   1     0.5    0.9953057  0.9858374  0.9470588
##   1     1.0    0.9951101  0.9894089  0.9470588
##   2     0.0    0.9636048  0.9578818  0.9529412
##   2     0.1    0.9938569  0.9754926  0.9529412
##   2     0.2    0.9936468  0.9753695  0.9470588
##   2     0.3    0.9951029  0.9788177  0.9529412
##   2     0.4    0.9947044  0.9823892  0.9470588
##   2     0.5    0.9942915  0.9823892  0.9470588
##   2     1.0    0.9951101  0.9859606  0.9470588
##
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were size = 1 and decay = 0.5.
```

```
testResults$NNet <- predict(nnet_model, test[, -1])
```

```
# ROC curve for neural network
nnet_roc <- roc(response = nnet_model$pred$obs,
               predictor = nnet_model$pred$M,
               levels = rev(levels(nnet_model$pred$obs)))
```

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

```
plot(nnet_roc, legaces.axes = TRUE)
```



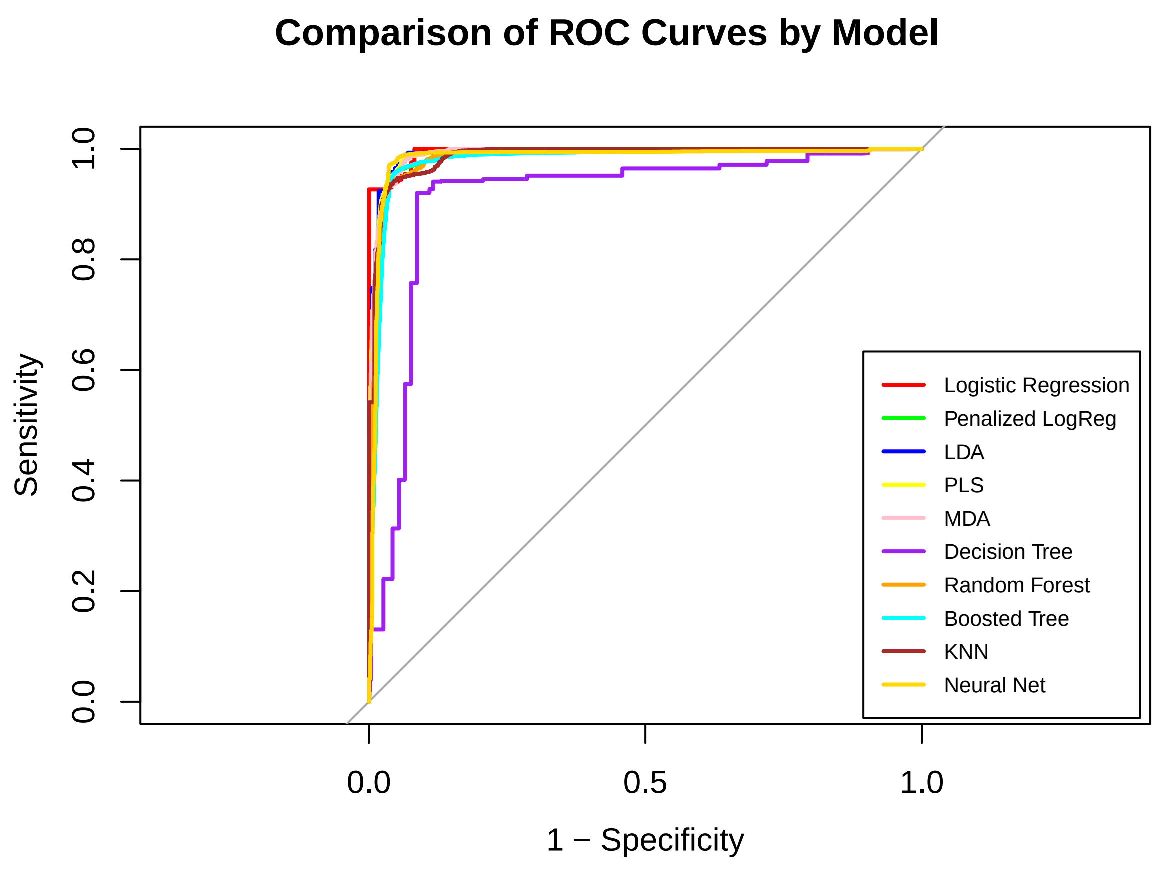
```
# Confusion Matrix
confusionMatrix(testResults$NNet, testResults$obs, positive = "M")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 69  0
##          M  2 42
##
##                Accuracy : 0.9823
##                  95% CI : (0.9375, 0.9978)
##     No Information Rate : 0.6283
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9625
##
##  Mcnemar's Test P-Value : 0.4795
##
##             Sensitivity : 1.0000
##             Specificity : 0.9718
##          Pos Pred Value : 0.9545
##          Neg Pred Value : 1.0000
##               Prevalence : 0.3717
##           Detection Rate : 0.3717
##     Detection Prevalence : 0.3894
##        Balanced Accuracy : 0.9859
##
##         'Positive' Class : M
##
```

Compare Models

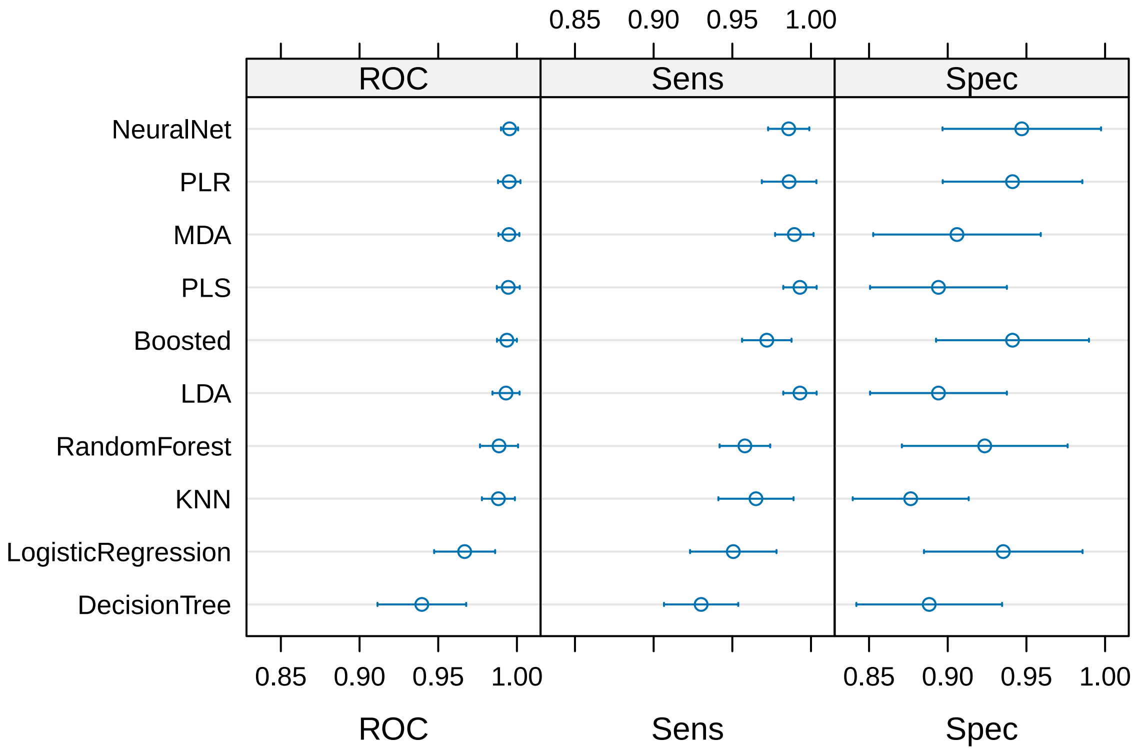
```
par(oma = c(0, 0, .75, 0))

plot(lr_roc, type = "s", col = 'red', legacy.axes = TRUE, lwd = 2)
plot(plr_roc, type = "s", add = TRUE, col = 'green', legacy.axes = TRUE, lwd = 2)
plot(lda_roc, type = "s", add = TRUE, col = 'blue', legacy.axes = TRUE, lwd = 2)
plot(pls_roc, type = "s", add = TRUE, col = 'yellow', legacy.axes = TRUE, lwd = 2)
plot(mda_roc, type = "s", add = TRUE, col = 'pink', legacy.axes = TRUE, lwd = 2)
plot(rpart_roc, type = "s", add = TRUE, col = 'purple', legacy.axes = TRUE, lwd = 2)
plot(rf_roc, type = "s", add = TRUE, col = 'orange', legacy.axes = TRUE, lwd = 2)
plot(gbm_roc, type = "s", add = TRUE, col = 'cyan', legacy.axes = TRUE, lwd = 2)
plot(knn_roc, type = "s", add = TRUE, col = 'brown', legacy.axes = TRUE, lwd = 2)
plot(nnet_roc, type = "s", add = TRUE, col = 'gold', legacy.axes = TRUE, lwd = 2)
legend("bottomright",
       legend = c("Logistic Regression", "Penalized LogReg", "LDA", "PLS",
                  "MDA", "Decision Tree", "Random Forest", "Boosted Tree",
                  "KNN", "Neural Net"),
       col = c("red", "green", "blue", "yellow", "pink",
               "purple", "orange", "cyan", "brown", "gold"),
       lwd = 2,
       cex = 0.7,
       inset = 0.01)
title(main = "Comparison of ROC Curves by Model", outer = TRUE)
```



```
# Dot plot comparing models
models <- list(LogisticRegression = lr_model, PLR = plr_model,LDA = lda_model, PLS = pls_model, MDA = mo

results_models <- resamples(models)
dotplot(results_models)
```



Confidence Level: 0.95

```r
# Table comparing performance metrics
results_table <- data.frame(
  Models = c("Logistic Regression", "Penalized LogReg", "LDA", "PLS", "MDA",
             "Decision Tree", "Random Forest", "Boosted Tree", "KNN", "Neural Net"),
  ROC_Train = c(0.9667959, 0.9951246, 0.9930745, 0.9945161, 0.9949073, 0.9395791, 0.9886410, 0.9936540,
  Sensitivity_Train = c(0.9506158, 0.9860837, 0.9929803, 0.9929803, 0.9894089, 0.9301724, 0.9580049, 0.
  Sensitivity_Test = c(0.9762, 0.9762, 0.9286, 0.9286, 0.9762, 0.8810, 0.9286, 0.9762, 0.9524, 1.0000),
  Specificity_Train = c(0.9352941, 0.9411765, 0.8941176, 0.8941176, 0.9058824, 0.8882353, 0.9235294, 0.
  Specificity_Test = c(0.9577, 0.9859, 1.0000, 1.0000, 0.9859, 0.9014, 0.9577, 1.0000, 0.9718, 0.9718)
  )

print(results_table)
```

```
##                 Models ROC_Train Sensitivity_Train Sensitivity_Test
## 1  Logistic Regression 0.9667959         0.9506158           0.9762
## 2     Penalized LogReg 0.9951246         0.9860837           0.9762
## 3                  LDA 0.9930745         0.9929803           0.9286
## 4                  PLS 0.9945161         0.9929803           0.9286
## 5                  MDA 0.9949073         0.9894089           0.9762
## 6        Decision Tree 0.9395791         0.9301724           0.8810
## 7        Random Forest 0.9886410         0.9580049           0.9286
## 8         Boosted Tree 0.9936540         0.9719212           0.9762
## 9                  KNN 0.9882353         0.9650246           0.9524
## 10          Neural Net 0.9953057         0.9858374           1.0000
##    Specificity_Train Specificity_Test
## 1          0.9352941           0.9577
## 2          0.9411765           0.9859
## 3          0.8941176           1.0000
## 4          0.8941176           1.0000
## 5          0.9058824            0.9859
## 6          0.8882353            0.9014
## 7          0.9235294            0.9577
## 8          0.9411765            1.0000
## 9          0.8764706            0.9718
## 10         0.9470588            0.9718
```