An R-based Machine Learning Protocol To Predict Diagnostic Biomarkers in Breast Cancer Using RNA-sequencing data

Pallabi Shaw, Kirthana Viswanathan and Manjari Kiran*

2026-02-20

Loading Required Libraries

```
library(dplyr)
library(DESeq2)
library(caret)
library(randomForest)
library(pROC)
library(ggplot2)
library(ggrepel)
```

Raw Counts Data Loading and Filtering

Raw tumor (TP) and normal (NT) samples were loaded. They were filtered against genes having less than 10 counts across columns. To ensure it is prepared for next steps, it was transformed into a genes x samples dataframe.

```
tumor_raw  <- read.csv("raw_counts_brca.csv", check.names = FALSE)
normal_raw <- read.csv("raw_counts_normal.csv", check.names = FALSE)

# For example: Visualizing the tumor counts file for reference
head(tumor_raw[, 1:5])
```

```
##                              shortLetterCode ENSG00000000003 ENSG00000000005
## TCGA.BH.A18H.01A.11R.A12D.07              TP             426               3
## TCGA.E2.A14P.01A.31R.A12D.07              TP            1033              21
## TCGA.AN.A04A.01A.21R.A034.07              TP            1417               9
## TCGA.AQ.A0Y5.01A.11R.A14M.07              TP             879               9
## TCGA.A8.A07I.01A.11R.A00Z.07              TP            1117              29
## TCGA.E9.A1R4.01A.21R.A14D.07              TP             711              15
##                              ENSG00000000419 ENSG00000000457
## TCGA.BH.A18H.01A.11R.A12D.07            1439            1049
## TCGA.E2.A14P.01A.31R.A12D.07            4553             932
## TCGA.AN.A04A.01A.21R.A034.07            1982            1227
## TCGA.AQ.A0Y5.01A.11R.A14M.07            2623            1727
## TCGA.A8.A07I.01A.11R.A00Z.07            1426             974
## TCGA.E9.A1R4.01A.21R.A14D.07            1943            1285
```

```
# Transpose to Genes x Samples
tumor_raw <- as.data.frame(t(tumor_raw))
normal_raw <- as.data.frame(t(normal_raw))

# Combine Tumor + Normal
counts <- cbind(tumor_raw, normal_raw)
cat(paste("Counts Dataframe Dimensions:", paste(dim(counts), collapse = " x ")))
```

```
## Counts Dataframe Dimensions: 60619 x 1224
```

```
# Count Filtering
keep <- rowSums(counts >= 10) >= 10
counts_filtered <- counts[keep, ]
cat(paste("Counts Dataframe Dimensions After Filtering:", paste(dim(counts_filtered), collapse = " x ")))
```

```
## Counts Dataframe Dimensions After Filtering: 39293 x 1224
```

Stratified Train-Test Split (70:30)

The filtered counts were split into training and testing datasets.

Note: *The shortLetterCode column in ml_data contains the condition labels for tumor (TP) and normal (NT) samples.*

```
ml_data <- as.data.frame(t(counts_filtered))

set.seed(123)
trainIndex <- createDataPartition(
  ml_data$shortLetterCode,
  p = 0.7,
  list = FALSE
)
trainData <- ml_data[trainIndex, ]
testData  <- ml_data[-trainIndex, ]

# To visualize the the 70:30 sample split
table(trainData$shortLetterCode)
```

```
##
##  NT  TP
##  80 778
```

```
table(testData$shortLetterCode)
```

```
##
##  NT  TP
##  33 333
```

Data Preparation and Running DESeq2 analysis

DESeq2 analysis was run only on the training dataset to obtain differentially expressed genes (DEGs). This is done to prevent data leakage to the testing dataset. DEGs were visualized using a volcano plot.

```
# Preparing Training data for DESeq2
trainData <- as.data.frame(t(trainData)) # Getting a genes x samples
train_group <- as.vector(unlist(trainData[1, ]))# Storing the counts
train_counts <- trainData[-1, ] # Storing the labels separately (TP or NT)
train_counts <- as.data.frame(
  apply(train_counts, 2, function(x) as.numeric(trimws(x))),
  row.names = rownames(train_counts)) # converting to numeric without changing rownames
train_counts[is.na(train_counts)] <- 0 # Converting NA counts to 0
train_counts <- as.matrix(train_counts) # DESeq2 requires a matrix as input for counts

col_data <- data.frame(
  row.names = colnames(train_counts),
  condition = factor(train_group))
# Creating a DESeq2 object
dds_train <- DESeqDataSetFromMatrix(
  countData = train_counts,
  colData = col_data,
  design = ~ condition
)

# Preparing the Testing dataset similarly
testData <- as.data.frame(t(testData))
test_group <- as.vector(unlist(testData[1, ]))
test_counts <- testData[-1, ]
test_counts <- as.data.frame(
  apply(test_counts, 2, function(x) as.numeric(trimws(x))),
  row.names = rownames(test_counts))
test_counts[is.na(test_counts)] <- 0
test_counts <- as.matrix(test_counts)

col_data <- data.frame(
  row.names = colnames(test_counts),
  condition = factor(test_group))
dds_test <- DESeqDataSetFromMatrix(
  countData = test_counts,
  colData = col_data,
  design = ~ condition
)
# Running DESeq2 on the Training dataset only
dds_train <- DESeq(dds_train)
res_train <- results(dds_train)
res_df_train <- as.data.frame(res_train)
```

```
# Volcano Plot
# Recreate clean dataframe from DESeq2 results
res_df_train <- as.data.frame(res_train)

# Remove NA padj
res_df_train <- res_df_train[!is.na(res_df_train$padj), ]

# Convert columns explicitly
res_df_train$padj <- as.numeric(res_df_train$padj)
res_df_train$log2FoldChange <- as.numeric(res_df_train$log2FoldChange)

# Add gene ID column
res_df_train$gene_id <- rownames(res_df_train)

res_df_train$threshold <- ifelse(
  res_df_train$padj < 0.05 & abs(res_df_train$log2FoldChange) > 1,
  "Significant",
  "Non-Significant"
)

res_df_train$threshold <- factor(
  res_df_train$threshold,
  levels = c("Non-Significant", "Significant")
)
# Order by padj
res_ordered <- res_df_train[order(res_df_train$padj), ]

# Keep only significant
res_sig <- res_ordered[res_ordered$threshold == "Significant", ]

# Taking the top 10 genes
top_genes <- head(res_sig, 10)

ggplot(res_df_train,
  aes(x = log2FoldChange,
          y = -log10(padj),
          color = threshold)) +
  geom_point(alpha = 0.5) +
  geom_text_repel(data = top_genes,
                  aes(label = gene_id),
                  size = 2,
                  box.padding = 0.4) +
  theme_minimal() +
  scale_color_manual(values = c("grey", "red")) +
  labs(title = "Volcano Plot (Training Data)",
       x = "Log2 Fold Change",
       y = "-Log10 Adjusted p-value",
       color = "Differential Expression")
```

Feature Selection and Data Normalization

The top 200 differentially expressed genes (DEGs), which are our features, were selected based on a threshold of **p-value < 0.05** and **log2 fold change > 1**. **Variance stabilizing transformation (VST)** normalization was applied to both the training and testing datasets. To evaluate the ability of these selected DEGs to distinguish between tumor and normal samples, **principal component analysis (PCA)** was performed, and the resulting clusters were visualized.

```
# Selecting Top 200 Genes
res_sig <- res_df_train %>%
  filter(padj < 0.05 & abs(log2FoldChange) > 1) %>%
  arrange(padj)
top_genes <- rownames(res_sig)[1:200]
cat("Number of significant DEGs:", nrow(res_sig), "\n")
```

```
## Number of significant DEGs: 9087
```

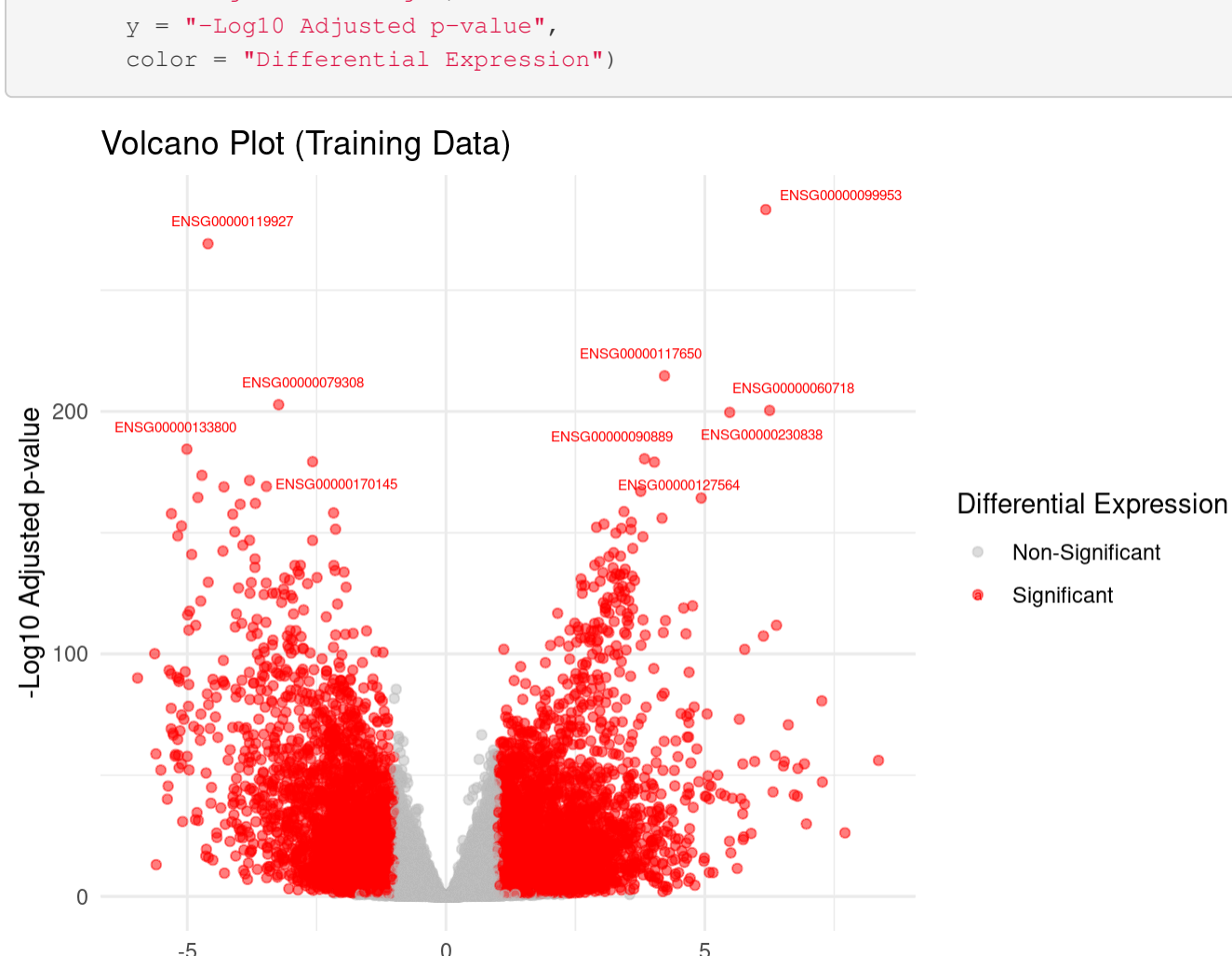
```
# Normalizing the Training dataset
# Using Variance Stabilizing Transformation on Raw counts (training data)
vsd_train <- vst(dds_train, blind=FALSE)
train_expr <- assay(vsd_train)[top_genes, ]
train_ml <- as.data.frame(t(train_expr))
train_ml$shortLetterCode <- train_group
train_ml <- train_ml %>% select(shortLetterCode, everything())

# Normalizing the Testing dataset
vsd_test <- vst(dds_test, blind = FALSE)
test_expr <- assay(vsd_test)[top_genes, ]
test_ml <- as.data.frame(t(test_expr))
test_ml$shortLetterCode <- test_group
test_ml <- test_ml %>% select(shortLetterCode, everything())

# PCA plot to visualize distinct clusters for Tumor and Normal
## PCA Plot (Training Data)
pca_data <- plotPCA(vsd_train,
                    intgroup="condition",
                    returnData=TRUE)

percentVar <- round(100 * attr(pca_data, "percentVar"))

ggplot(pca_data, aes(PC1, PC2, color=condition)) +
  geom_point(size=3) +
  xlab(paste0("PC1: ", percentVar[1], "%")) +
  ylab(paste0("PC2: ", percentVar[2], "%")) +
  theme_minimal() +
  labs(title="PCA Plot (Training Data)")
```



The PCA plot showed clear separation between tumor and normal samples, supporting effective DEG-based feature selection, though some overlap of normal samples within the tumor cluster may reflect data imbalance.

Building the Random Forest Model

The data were split into training and testing sets with **x (features)** and **y (labels)** variables. A **5-fold cross-validation** was applied to the training set to prevent overfitting, and the model was trained using 200 trees.

```
# x_train and x_test contain the features or counts data (input variable)
# y_train and y_test contain the sample labels (output variable)
y_train <- as.factor(train_ml$shortLetterCode)
y_train <- factor(y_train, levels = c("NT", "TP"))
x_train <- train_ml  %>% dplyr::select(-shortLetterCode)
x_train <- as.matrix(x_train)
cat("x-train =", paste(dim(x_train), collapse = " x "),
    " | y-train =", length(y_train), "\n")
```

```
## x-train = 858 x 200  | y-train = 858
```

```
# Test split
## Only selecting the DEGs from the x_test set
y_test <- as.factor(test_ml$shortLetterCode)
y_test  <- factor(y_test,  levels = c("NT", "TP"))
x_test <- test_ml  %>% dplyr::select(-shortLetterCode)
x_test <- as.matrix(x_test)
cat("x-test =", paste(dim(x_test), collapse = " x "),
    " | y-test =", length(y_test), "\n")
```

```
## x-test = 366 x 200  | y-test = 366
```

```
## Ensuring identical feature order between x_train and x_test
r <- identical(colnames(x_train), colnames(x_test))
# Output should be true

# Building the Random Forest Model
set.seed(123)

# Train Control for 5-Fold CV
ctrl <- trainControl(
  method = "cv",
  number = 5,
  classProbs = TRUE,
  summaryFunction = twoClassSummary,
  savePredictions = "final"
)

# Train Random Forest with 5-Fold CV
rf_model <- train(
  x = x_train,
  y = y_train,
  method = "rf",
  metric = "ROC",
  trControl = ctrl,
  ntree = 200,
  importance = TRUE
)

print(rf_model)
```

```
## Random Forest
##
## 858 samples
## 200 predictors
##   2 classes: 'NT', 'TP'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 687, 686, 686, 686, 687
## Resampling results across tuning parameters:
##
##   mtry  ROC        Sens    Spec
##     2   0.9987565  0.9000  0.9948553
##   101   0.9995983  0.9375  0.9961456
##   200   0.9996384  0.9375  0.9935732
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 200.
```

Evaluating Model Performance: Testing Data

The trained model was evaluated on the testing data, and the **Confusion Matrix**, **Receiver Operating Characteristic (ROC) curve** and **Area Under the Curve (AUC)** were computed.

```
# Predicting on Test Data
pred_class <- predict(rf_model, x_test)

# Evaluation of Model Performance

## Confusion matrix
confusionMatrix(pred_class, y_test)
```

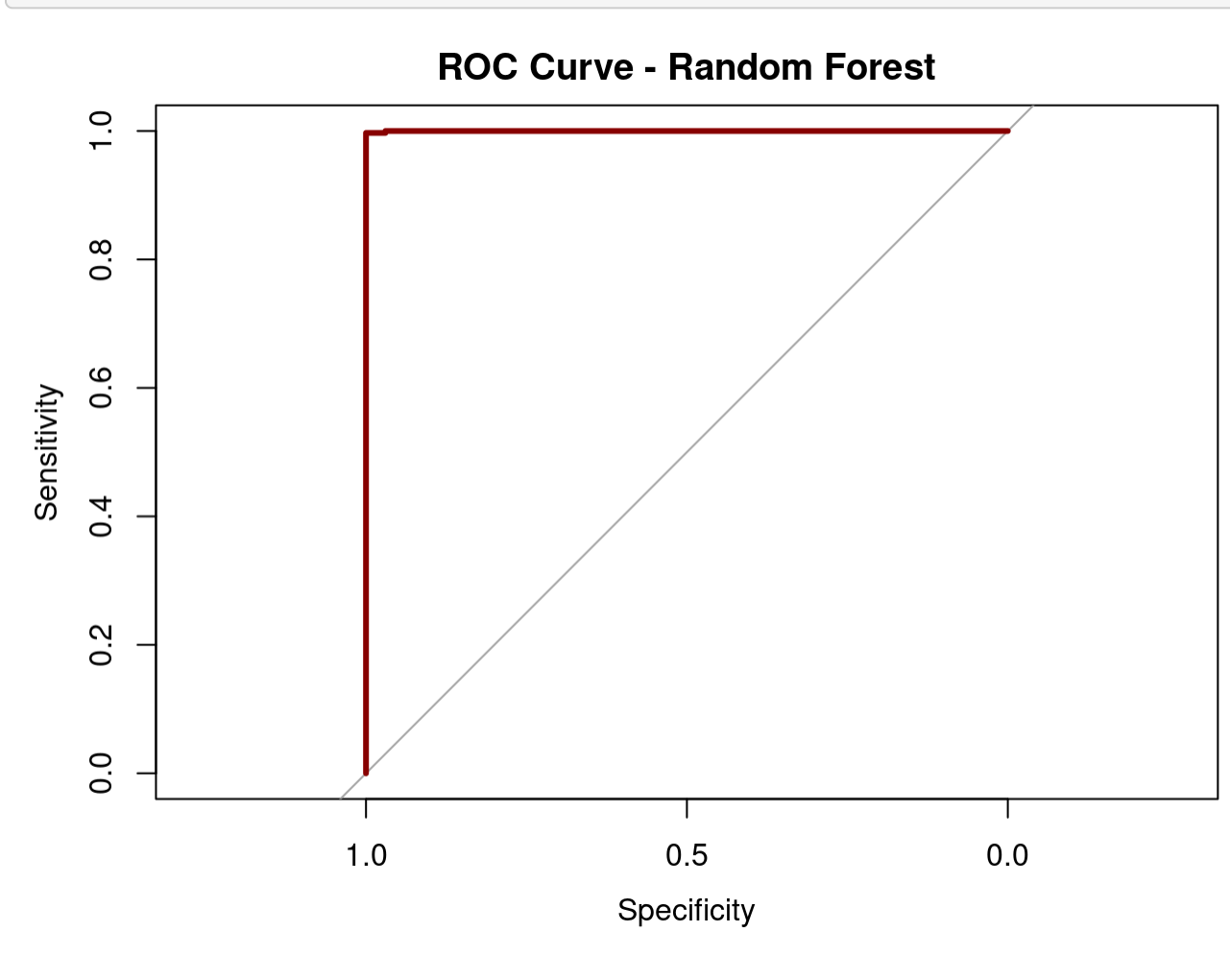
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  NT  TP
##         NT  32   0
##         TP   1 333
##
##                Accuracy : 0.9973
##                  95% CI : (0.9849, 0.9999)
##     No Information Rate : 0.9098
##     P-Value [Acc > NIR] : 3.563e-14
##
##                   Kappa : 0.9831
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.96970
##             Specificity : 1.00000
##          Pos Pred Value : 1.00000
##          Neg Pred Value : 0.99701
##              Prevalence : 0.09016
##          Detection Rate : 0.08743
##    Detection Prevalence : 0.08743
##       Balanced Accuracy : 0.98485
##
##        'Positive' Class : NT
##
```

```
## ROC Curve + AUC
pred_prob <- predict(rf_model, x_test, type = "prob")[,2]
roc_obj <- roc(y_test, pred_prob)

auc(roc_obj)
```

```
## Area under the curve: 0.9999
```

```
plot(roc_obj,
     col = "darkred",
     lwd = 3,
     main = "ROC Curve - Random Forest")
```



Evaluating Model Performance: Independent dataset

An independent dataset comprising 18 normal and 15 breast tumor samples was used to evaluate the performance of the model trained on the top 200 DEGs.

```
# Reading and preparing independent dataset
indep_data <- read.csv("GSE71651.csv", header = T)
indep_data <- as.data.frame(indep_data)
indep_data$class <- as.character(indep_data$class)
head(indep_data[,1:5])
```

```
##           ID   class ENSG00000000003 ENSG00000000005 ENSG00000000419
## 1 SRR2148235   Tumor        8.175975        2.639261        8.916178
## 2 SRR2148236   Tumor        9.749349        6.410159        9.210448
## 3 SRR2148237   Tumor        6.704148        3.852458        8.876623
## 4 SRR2148238  Normal        9.212099        6.027375        8.585745
## 5 SRR2148239  Normal       10.297293        1.873562       10.229853
## 6 SRR2148240  Normal       10.257146        5.919515        9.326000
```

```
indep_data <- indep_data %>%
  mutate(class = case_when(
    class == " Tumor"  ~ "TP",
    class == " Normal" ~ "NT",
    TRUE ~ class
  ))
indep_data %>%
  group_by(class) %>%
  summarise(n = n())
```

```
## # A tibble: 2 × 2
##   class     n
##   <chr> <int>
## 1 NT       18
## 2 TP       15
```

```
# Preparing as independent dataset as the test data
indep_data <- as.data.frame(indep_data)
rownames(indep_data) <- indep_data$ID
indep_data <- indep_data %>% select(-c(ID))

cat(paste("Independent data counts dimensions:", paste(dim(indep_data), collapse = " x ")))
```

```
## Independent data counts dimensions: 33 x 58736
```

```
y_indep <- as.factor(indep_data$class)
y_indep <- factor(y_indep,  levels = c("NT", "TP"))

x_indep <- indep_data %>% dplyr::select(-class)
x_indep <- x_indep[, top_genes]

cat("x-indep =", paste(dim(x_indep), collapse = " x "),
    " | y-indep =", length(y_indep), "\n")
```

```
## x-indep = 33 x 200  | y-indep = 33
```

```
x_indep <- as.matrix(x_indep)

r <- identical(colnames(x_train), colnames(x_indep))

# Running the model on independent dataset
pred_class <- predict(rf_model, x_indep)

# Evaluation of Model Performance
## Confusion matrix
confusionMatrix(pred_class, y_indep, positive = "TP")
```

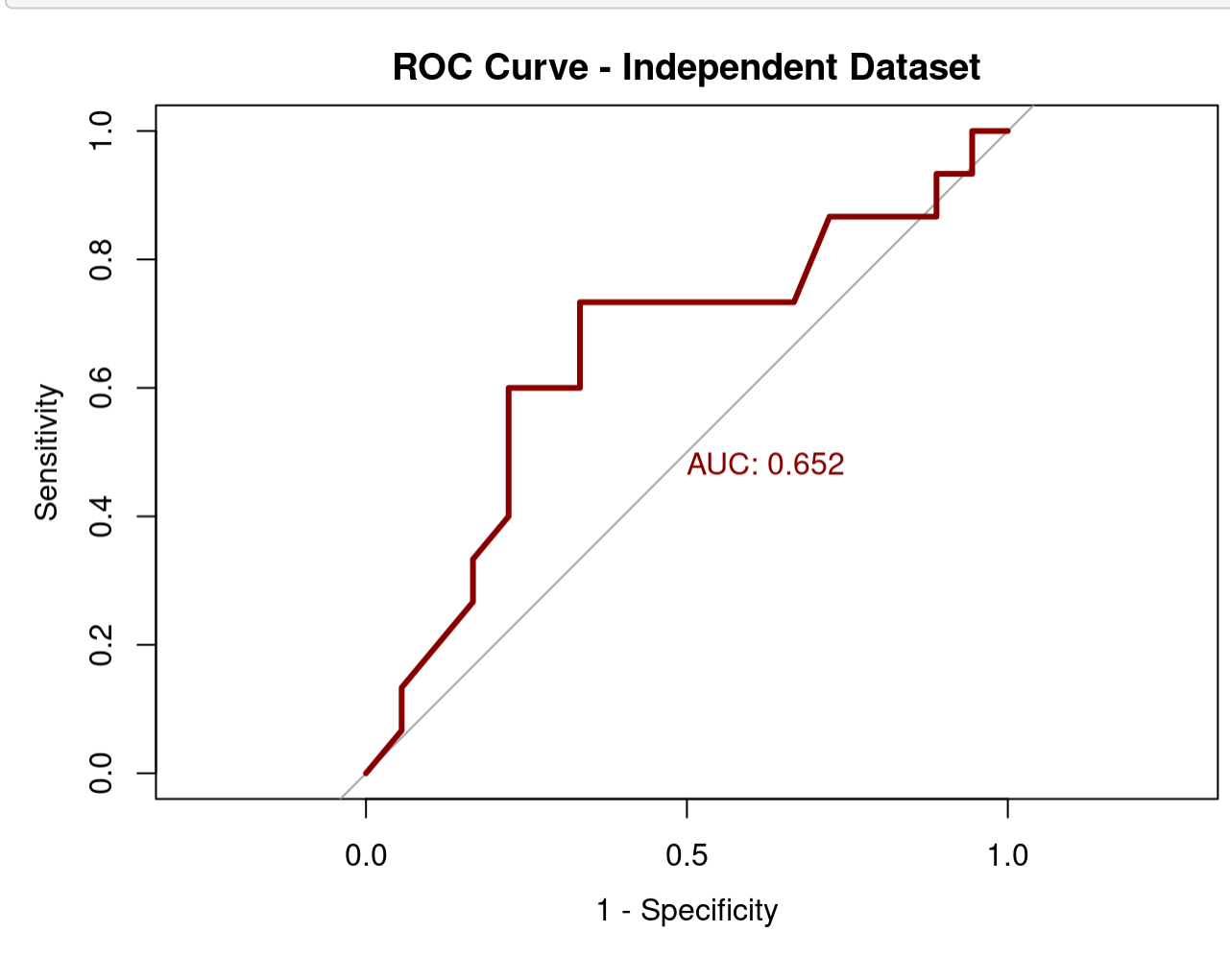
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NT TP
##         NT 12  6
##         TP  6  9
##
##                Accuracy : 0.6364
##                  95% CI : (0.4512, 0.796)
##     No Information Rate : 0.5455
##     P-Value [Acc > NIR] : 0.1916
##
##                   Kappa : 0.2667
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.6000
##             Specificity : 0.6667
##          Pos Pred Value : 0.6000
##          Neg Pred Value : 0.6667
##              Prevalence : 0.4545
##          Detection Rate : 0.2727
##    Detection Prevalence : 0.4545
##       Balanced Accuracy : 0.6333
##
##        'Positive' Class : TP
##
```

```
## ROC Curve + AUC
pred_prob <- predict(rf_model, x_indep, type = "prob")[,2]
roc_obj <- roc(y_indep, pred_prob)

auc_value <- auc(roc_obj)
cat("AUC value =", auc_value, "\n")
```

```
## AUC value = 0.6518519
```

```
plot(roc_obj,
     col="darkred",
     lwd=3,
     legacy.axes=TRUE,
     print.auc=TRUE,
     main="ROC Curve - Independent Dataset")
```



Finding the Top 10 Biomarkers from our dataset

```
library(biomaRt)

# Connecting to Biomart to parse for the human genes (ensembl)
mart <- useEnsembl(
  biomart = "genes",
  dataset = "hsapiens_gene_ensembl"
)

# Taking the previously obtained significant genes dataframe
# Getting gene names for the ensembl IDs

gene_annotation <- getBM(
  attributes = c("ensembl_gene_id", "hgnc_symbol"),
  filters = "ensembl_gene_id",
  values = res_sig$gene_id,
  mart = mart
)

# Merging with the DEGs table (res_sig)
res_sig_annotated <- res_sig %>%
  left_join(gene_annotation, by = c("gene_id" = "ensembl_gene_id"))

# Removing Unannotated genes
res_sig_annotated <- res_sig_annotated %>%
  filter(hgnc_symbol != "")

res_sig_annotated <- as.data.frame(res_sig_annotated)
# Ordering based on p_adj
top10_biomarkers <- res_sig_annotated %>%
  arrange(padj) %>%
  dplyr::slice(1:10) %>%
  pull(hgnc_symbol)

# Printing the top 10 biomarkers
cat(top10_biomarkers, sep = "\n")
```

```
## MMP11
## GPAM
## NEK2
## TNS1
## COL11A1
## LINC01614
## LYVE1
## KIF4A
## SIK2
## PKMYT1
```

```
sessionInfo()
```

```
## R version 4.5.2 (2025-10-31)
## Platform: x86_64-pc-linux-gnu
## Running under: Ubuntu 22.04.4 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.10.0
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0  LAPACK version 3.10.0
##
## locale:
##  [1] LC_CTYPE=en_IN.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_IN.UTF-8        LC_COLLATE=en_IN.UTF-8
##  [5] LC_MONETARY=en_IN.UTF-8    LC_MESSAGES=en_IN.UTF-8
##  [7] LC_PAPER=en_IN.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_IN.UTF-8 LC_IDENTIFICATION=C
##
## time zone: Asia/Kolkata
## tzcode source: system (glibc)
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] biomaRt_2.66.0              ggrepel_0.9.6
##  [3] pROC_1.19.0.1               randomForest_4.7-1.2
##  [5] caret_7.0-1                 lattice_0.22-5
##  [7] ggplot2_4.0.2               DESeq2_1.50.2
##  [9] SummarizedExperiment_1.40.0 Biobase_2.70.0
## [11] MatrixGenerics_1.22.0       matrixStats_1.5.0
## [13] GenomicRanges_1.62.1        Seqinfo_1.0.0
## [15] IRanges_2.44.0              S4Vectors_0.48.0
## [17] BiocGenerics_0.56.0         generics_0.1.4
## [19] dplyr_1.2.0
##
## loaded via a namespace (and not attached):
##   [1] DBI_1.2.3            httr2_1.2.2          rlang_1.1.7
##   [4] magrittr_2.0.4       otel_0.2.0           e1071_1.7-17
##   [7] compiler_4.5.2       RSQLite_2.4.6        png_0.1-8
##  [10] vctrs_0.7.1          reshape2_1.4.5       stringr_1.6.0
##  [13] crayon_1.5.3         pkgconfig_2.0.3      fastmap_1.2.0
##  [16] dbplyr_2.5.2         XVector_0.50.0       labeling_0.4.3
##  [19] utf8_1.2.6           rmarkdown_2.30       prodlim_2025.04.28
##  [22] bit_4.6.0            purrr_1.2.1          xfun_0.56
##  [25] cachem_1.1.0         jsonlite_2.0.0       progress_1.2.3
##  [28] blob_1.3.0           recipes_1.3.1        DelayedArray_0.36.0
##  [31] BiocParallel_1.44.0  prettyunits_1.2.0    parallel_4.5.2
##  [34] R6_2.6.1             bslib_0.10.0         stringi_1.8.7
##  [37] RColorBrewer_1.1-3   parallelly_1.46.1    rpart_4.1.24
##  [40] lubridate_1.9.5      jquerylib_0.1.4      Rcpp_1.1.1
##  [43] iterators_1.0.14     knitr_1.51           future.apply_1.20.1
##  [46] Matrix_1.7-4         splines_4.5.2        nnet_7.3-20
##  [49] timechange_0.4.0     tidyselect_1.2.1     rstudioapi_0.18.0
##  [52] abind_1.4-8          yaml_2.3.12          timeDate_4052.112
##  [55] codetools_0.2-19     curl_7.0.0           listenv_0.10.0
##  [58] tibble_3.3.1         plyr_1.8.9           KEGGREST_1.50.0
##  [61] withr_3.0.2          S7_0.2.1             evaluate_1.0.5
##  [64] future_1.69.0        survival_3.8-3       proxy_0.4-29
##  [67] BiocFileCache_3.0.0  xml2_1.5.2           Biostrings_2.78.0
##  [70] filelock_1.0.3       pillar_1.11.1        foreach_1.5.2
##  [73] hms_1.1.4            scales_1.4.0         globals_0.19.0
##  [76] class_7.3-23         glue_1.8.0           tools_4.5.2
##  [79] data.table_1.18.2.1  ModelMetrics_1.2.2.2 gower_1.0.2
##  [82] locfit_1.5-9.12      grid_4.5.2           AnnotationDbi_1.72.0
##  [85] ipred_0.9-15         nlme_3.1-168         cli_3.6.5
##  [88] rappdirs_0.3.4       S4Arrays_1.10.1      lava_1.8.2
##  [91] gtable_0.3.6         sass_0.4.10          digest_0.6.39
##  [94] SparseArray_1.10.8   farver_2.1.2         memoise_2.0.1
##  [97] htmltools_0.5.9      lifecycle_1.0.5      httr_1.4.8
## [100] hardhat_1.4.2        bit64_4.6.0-1        MASS_7.3-65
```