Breast Cancer Diagnosis using Classification

Sean Cairns

2023-11-16

## Introduction

## Data Exploration

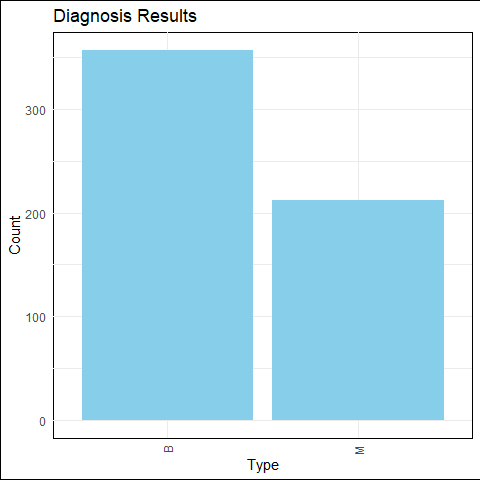
## [1] "Data Dimensions: Rows = 569 , Columns = 33"

## Shuffling the data

# shuffle the data  
set.seed(100)  
shuffle\_index <- sample(1:nrow(data))  
data <- data[shuffle\_index, ]  
head(data)

## id diagnosis radius\_mean texture\_mean perimeter\_mean area\_mean  
## 503 91505 B 12.540 16.32 81.25 476.3  
## 358 901028 B 13.870 16.21 88.52 593.7  
## 470 911366 B 11.620 18.18 76.38 408.8  
## 516 916221 B 11.340 18.61 72.76 391.2  
## 98 862261 B 9.787 19.94 62.11 294.5  
## 7 844359 M 18.250 19.98 119.60 1040.0  
## smoothness\_mean compactness\_mean concavity\_mean concave.points\_mean  
## 503 0.11580 0.10850 0.059280 0.032790  
## 358 0.08743 0.05492 0.015020 0.020880  
## 470 0.11750 0.14830 0.102000 0.055640  
## 516 0.10490 0.08499 0.043020 0.025940  
## 98 0.10240 0.05301 0.006829 0.007937  
## 7 0.09463 0.10900 0.112700 0.074000  
## symmetry\_mean fractal\_dimension\_mean radius\_se texture\_se perimeter\_se  
## 503 0.1943 0.06612 0.2577 1.0950 1.566  
## 358 0.1424 0.05883 0.2543 1.3630 1.737  
## 470 0.1957 0.07255 0.4101 1.7400 3.027  
## 516 0.1927 0.06211 0.2430 1.0100 1.491  
## 98 0.1350 0.06890 0.3350 2.0430 2.132  
## 7 0.1794 0.05742 0.4467 0.7732 3.180  
## area\_se smoothness\_se compactness\_se concavity\_se concave.points\_se  
## 503 18.49 0.009702 0.015670 0.025750 0.011610  
## 358 20.74 0.005638 0.007939 0.005254 0.006042  
## 470 27.85 0.014590 0.032060 0.049610 0.018410  
## 516 18.19 0.008577 0.016410 0.020990 0.011070  
## 98 20.05 0.011130 0.014630 0.005308 0.005250  
## 7 53.91 0.004314 0.013820 0.022540 0.010390  
## symmetry\_se fractal\_dimension\_se radius\_worst texture\_worst perimeter\_worst  
## 503 0.02801 0.002480 13.57 21.40 86.67  
## 358 0.01544 0.002087 15.11 25.58 96.74  
## 470 0.01807 0.005217 13.36 25.40 88.14  
## 516 0.02434 0.001217 12.47 23.03 79.15  
## 98 0.01801 0.005667 10.92 26.29 68.81  
## 7 0.01369 0.002179 22.88 27.66 153.20  
## area\_worst smoothness\_worst compactness\_worst concavity\_worst  
## 503 552.0 0.1580 0.17510 0.18890  
## 358 694.4 0.1153 0.10080 0.05285  
## 470 528.1 0.1780 0.28780 0.31860  
## 516 478.6 0.1483 0.15740 0.16240  
## 98 366.1 0.1316 0.09473 0.02049  
## 7 1606.0 0.1442 0.25760 0.37840  
## concave.points\_worst symmetry\_worst fractal\_dimension\_worst X  
## 503 0.08411 0.3155 0.07538 NA  
## 358 0.05556 0.2362 0.07113 NA  
## 470 0.14160 0.2660 0.09270 NA  
## 516 0.08542 0.3060 0.06783 NA  
## 98 0.02381 0.1934 0.08988 NA  
## 7 0.19320 0.3063 0.08368 NA

diagnosis\_plot <- ggplot(data, aes(x = diagnosis)) +  
 geom\_bar(fill = "skyblue") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust = 1),  
 plot.background = element\_rect(fill = "white"),  
 panel.background = element\_rect(fill = "white")) +  
 labs(title = "Diagnosis Results",  
 x = "Type",  
 y = "Count")  
ggsave("./Images/Diagnosis.png", plot = diagnosis\_plot, width = 8, height = 6, dpi = 300)  
  
diagnosis\_plot

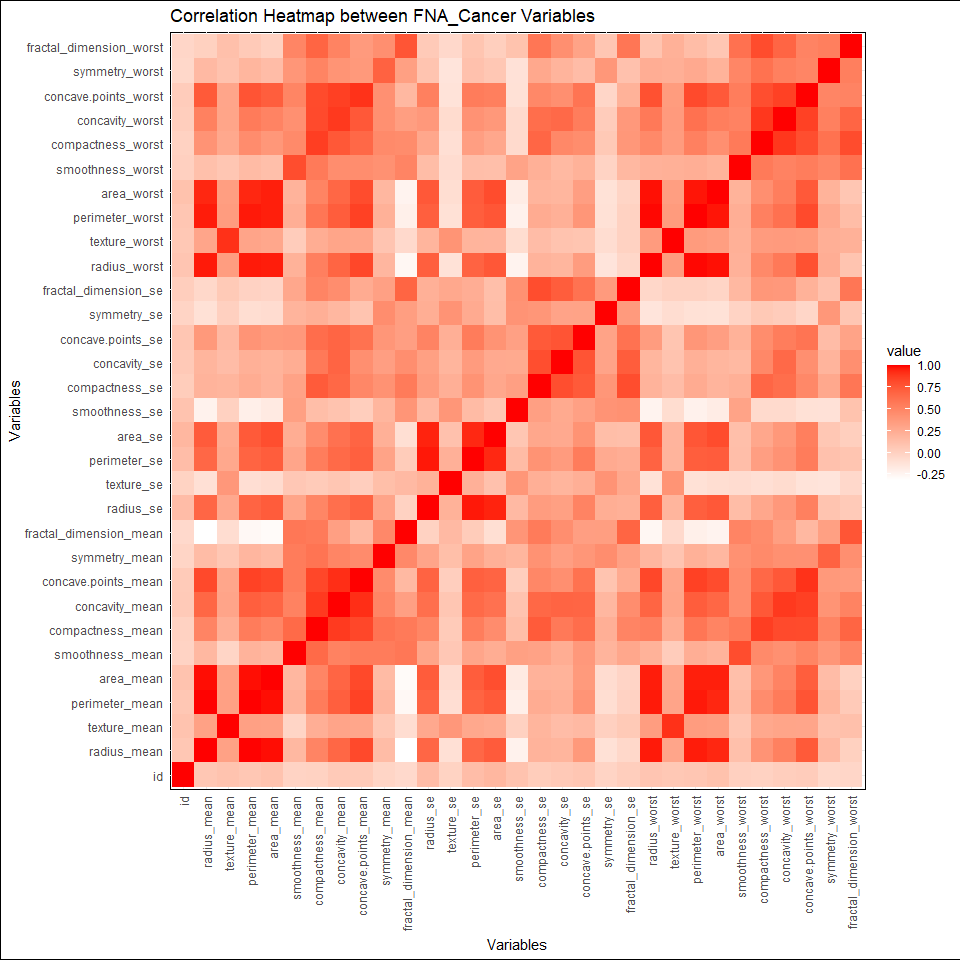


## Cleaning the data

# Drop unwanted variable(s) and null values  
data <- data %>% select(-c(X)) %>%  
na.omit()  
head(data)

## id diagnosis radius\_mean texture\_mean perimeter\_mean area\_mean  
## 503 91505 B 12.540 16.32 81.25 476.3  
## 358 901028 B 13.870 16.21 88.52 593.7  
## 470 911366 B 11.620 18.18 76.38 408.8  
## 516 916221 B 11.340 18.61 72.76 391.2  
## 98 862261 B 9.787 19.94 62.11 294.5  
## 7 844359 M 18.250 19.98 119.60 1040.0  
## smoothness\_mean compactness\_mean concavity\_mean concave.points\_mean  
## 503 0.11580 0.10850 0.059280 0.032790  
## 358 0.08743 0.05492 0.015020 0.020880  
## 470 0.11750 0.14830 0.102000 0.055640  
## 516 0.10490 0.08499 0.043020 0.025940  
## 98 0.10240 0.05301 0.006829 0.007937  
## 7 0.09463 0.10900 0.112700 0.074000  
## symmetry\_mean fractal\_dimension\_mean radius\_se texture\_se perimeter\_se  
## 503 0.1943 0.06612 0.2577 1.0950 1.566  
## 358 0.1424 0.05883 0.2543 1.3630 1.737  
## 470 0.1957 0.07255 0.4101 1.7400 3.027  
## 516 0.1927 0.06211 0.2430 1.0100 1.491  
## 98 0.1350 0.06890 0.3350 2.0430 2.132  
## 7 0.1794 0.05742 0.4467 0.7732 3.180  
## area\_se smoothness\_se compactness\_se concavity\_se concave.points\_se  
## 503 18.49 0.009702 0.015670 0.025750 0.011610  
## 358 20.74 0.005638 0.007939 0.005254 0.006042  
## 470 27.85 0.014590 0.032060 0.049610 0.018410  
## 516 18.19 0.008577 0.016410 0.020990 0.011070  
## 98 20.05 0.011130 0.014630 0.005308 0.005250  
## 7 53.91 0.004314 0.013820 0.022540 0.010390  
## symmetry\_se fractal\_dimension\_se radius\_worst texture\_worst perimeter\_worst  
## 503 0.02801 0.002480 13.57 21.40 86.67  
## 358 0.01544 0.002087 15.11 25.58 96.74  
## 470 0.01807 0.005217 13.36 25.40 88.14  
## 516 0.02434 0.001217 12.47 23.03 79.15  
## 98 0.01801 0.005667 10.92 26.29 68.81  
## 7 0.01369 0.002179 22.88 27.66 153.20  
## area\_worst smoothness\_worst compactness\_worst concavity\_worst  
## 503 552.0 0.1580 0.17510 0.18890  
## 358 694.4 0.1153 0.10080 0.05285  
## 470 528.1 0.1780 0.28780 0.31860  
## 516 478.6 0.1483 0.15740 0.16240  
## 98 366.1 0.1316 0.09473 0.02049  
## 7 1606.0 0.1442 0.25760 0.37840  
## concave.points\_worst symmetry\_worst fractal\_dimension\_worst  
## 503 0.08411 0.3155 0.07538  
## 358 0.05556 0.2362 0.07113  
## 470 0.14160 0.2660 0.09270  
## 516 0.08542 0.3060 0.06783  
## 98 0.02381 0.1934 0.08988  
## 7 0.19320 0.3063 0.08368

heatmap\_data <- data[, -which(names(data) == 'diagnosis')]  
cor\_mat <- melt(cor(heatmap\_data))  
  
cor\_heatmap <- ggplot(cor\_mat, aes(Var1, Var2, fill = value)) +  
 geom\_tile() +   
 scale\_fill\_gradient(low='white', high = "red") +   
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust = 1),  
 plot.background = element\_rect(fill = "white"),  
 panel.background = element\_rect(fill = "white")) +   
 labs(title = "Correlation Heatmap between FNA\_Cancer Variables",  
 x = "Variables",  
 y = "Variables")  
ggsave("cor\_heatmap.png", plot = cor\_heatmap, width = 8, height = 6, dpi = 300)  
  
cor\_heatmap



## Creating the train and test set

target <- data$diagnosis  
train\_indices <- createDataPartition(target, p = 0.8, list = FALSE)  
  
# Split data into training and test sets  
train\_data <- data[train\_indices, ]  
test\_data <- data[-train\_indices, ]  
  
  
# Print the dimensions of the train and test data  
train\_dimensions <- dim(train\_data)  
test\_dimensions <- dim(test\_data)  
  
  
print(paste("Train Data Dimensions: Rows =", train\_dimensions[1], ", Columns =", train\_dimensions[2]))

## [1] "Train Data Dimensions: Rows = 456 , Columns = 32"

print(paste("Test Data Dimensions: Rows =", test\_dimensions[1], ", Columns =", test\_dimensions[2]))

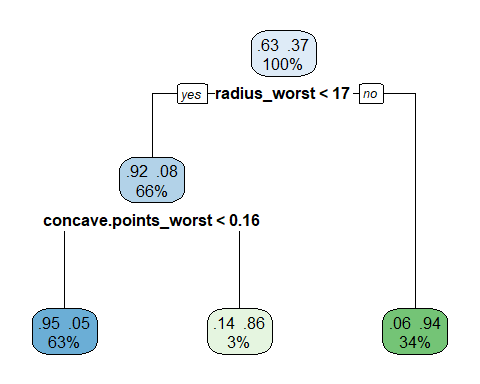
## [1] "Test Data Dimensions: Rows = 113 , Columns = 32"

# Decision Tree Model

# List of parameters  
ctrl <- trainControl(method = "cv",   
 number = 10,   
 search = "random",   
 verboseIter = FALSE,   
 classProbs = TRUE  
 )  
  
# List of hyper parameters to loop through  
hyper\_grid <- expand.grid(  
 cp = runif(100, min = 0.01, max = 0.5)  
)  
  
  
# Training model  
dt\_model <- train(  
 diagnosis ~ .,  
 data = train\_data, # Need to recombine features and target for gridsearch  
 method = "rpart",  
 trControl = ctrl,  
 tuneGrid = hyper\_grid  
   
)  
print(dt\_model)

## CART   
##   
## 456 samples  
## 31 predictor  
## 2 classes: 'B', 'M'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 410, 410, 411, 410, 410, 410, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.01310523 0.9319324 0.8548378  
## 0.01367985 0.9319324 0.8548378  
## 0.01817391 0.9341546 0.8583256  
## 0.01846586 0.9341546 0.8583256  
## 0.02479602 0.9385024 0.8678780  
## 0.03231561 0.9385024 0.8678780  
## 0.04912799 0.9275845 0.8460173  
## 0.04934460 0.9275845 0.8460173  
## 0.05117029 0.9275845 0.8460173  
## 0.05227439 0.9275845 0.8460173  
## 0.05940908 0.9209662 0.8286421  
## 0.06889630 0.9165217 0.8182612  
## 0.08106233 0.9165217 0.8179033  
## 0.08272409 0.9165217 0.8179033  
## 0.08390305 0.9165217 0.8179033  
## 0.08592358 0.9165217 0.8179033  
## 0.08747613 0.9165217 0.8179033  
## 0.09228034 0.9165217 0.8179033  
## 0.09491572 0.9165217 0.8179033  
## 0.09572605 0.9165217 0.8179033  
## 0.09667169 0.9165217 0.8179033  
## 0.11103613 0.9165217 0.8179033  
## 0.11436596 0.9165217 0.8179033  
## 0.11995900 0.9165217 0.8179033  
## 0.12206473 0.9165217 0.8179033  
## 0.12292336 0.9165217 0.8179033  
## 0.13643325 0.9165217 0.8179033  
## 0.14368346 0.9165217 0.8179033  
## 0.14703735 0.9165217 0.8179033  
## 0.15439010 0.9165217 0.8179033  
## 0.15614461 0.9165217 0.8179033  
## 0.15812570 0.9165217 0.8179033  
## 0.15823798 0.9165217 0.8179033  
## 0.16901749 0.9165217 0.8179033  
## 0.17618342 0.9165217 0.8179033  
## 0.17890031 0.9165217 0.8179033  
## 0.18391546 0.9165217 0.8179033  
## 0.18860955 0.9165217 0.8179033  
## 0.18949527 0.9165217 0.8179033  
## 0.19006731 0.9165217 0.8179033  
## 0.19087778 0.9165217 0.8179033  
## 0.19135097 0.9165217 0.8179033  
## 0.19471793 0.9165217 0.8179033  
## 0.19476622 0.9165217 0.8179033  
## 0.19954949 0.9165217 0.8179033  
## 0.20818667 0.9165217 0.8179033  
## 0.20992788 0.9165217 0.8179033  
## 0.22341114 0.9165217 0.8179033  
## 0.23632957 0.9165217 0.8179033  
## 0.24034235 0.9165217 0.8179033  
## 0.24294552 0.9165217 0.8179033  
## 0.24407474 0.9165217 0.8179033  
## 0.25102515 0.9165217 0.8179033  
## 0.26140499 0.9165217 0.8179033  
## 0.26267552 0.9165217 0.8179033  
## 0.28447362 0.9165217 0.8179033  
## 0.29063477 0.9165217 0.8179033  
## 0.29489364 0.9165217 0.8179033  
## 0.30061015 0.9165217 0.8179033  
## 0.30268092 0.9165217 0.8179033  
## 0.31160610 0.9165217 0.8179033  
## 0.31742775 0.9165217 0.8179033  
## 0.32773512 0.9165217 0.8179033  
## 0.32985154 0.9165217 0.8179033  
## 0.34435493 0.9165217 0.8179033  
## 0.34561151 0.9165217 0.8179033  
## 0.34779631 0.9165217 0.8179033  
## 0.35748612 0.9165217 0.8179033  
## 0.36099295 0.9165217 0.8179033  
## 0.37606047 0.9165217 0.8179033  
## 0.37689799 0.9165217 0.8179033  
## 0.38541293 0.9165217 0.8179033  
## 0.38752028 0.9165217 0.8179033  
## 0.38945826 0.9165217 0.8179033  
## 0.38996477 0.9165217 0.8179033  
## 0.39521041 0.9165217 0.8179033  
## 0.39525630 0.9165217 0.8179033  
## 0.39598864 0.9165217 0.8179033  
## 0.40155283 0.9165217 0.8179033  
## 0.41276681 0.9165217 0.8179033  
## 0.41663454 0.9165217 0.8179033  
## 0.42100120 0.9165217 0.8179033  
## 0.42132384 0.9165217 0.8179033  
## 0.43030424 0.9165217 0.8179033  
## 0.43258437 0.9165217 0.8179033  
## 0.43624771 0.9165217 0.8179033  
## 0.43721993 0.9165217 0.8179033  
## 0.44686022 0.9165217 0.8179033  
## 0.44809517 0.9165217 0.8179033  
## 0.44811502 0.9165217 0.8179033  
## 0.45290439 0.9165217 0.8179033  
## 0.46613122 0.9165217 0.8179033  
## 0.46656733 0.9165217 0.8179033  
## 0.47160407 0.9165217 0.8179033  
## 0.48342486 0.9165217 0.8179033  
## 0.48912681 0.9165217 0.8179033  
## 0.49077533 0.9165217 0.8179033  
## 0.49092449 0.9165217 0.8179033  
## 0.49140050 0.9165217 0.8179033  
## 0.49336280 0.9165217 0.8179033  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.03231561.

# Getting best model  
best\_model <- dt\_model$finalModel  
# Plotting model  
rpart.plot(best\_model, extra = 105)

 ## Showing Feature Importance

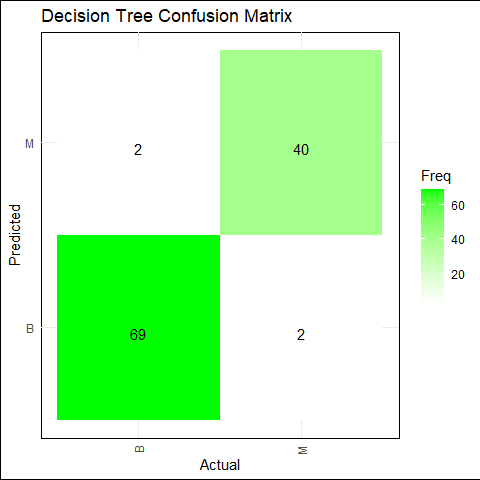
## rpart variable importance  
##   
## only 20 most important variables shown (out of 31)  
##   
## Overall  
## concave.points\_worst 100.000  
## perimeter\_worst 98.901  
## concave.points\_mean 94.986  
## radius\_worst 92.346  
## area\_worst 91.471  
## concavity\_mean 8.735  
## compactness\_worst 7.621  
## texture\_worst 0.000  
## texture\_se 0.000  
## smoothness\_se 0.000  
## radius\_mean 0.000  
## concavity\_worst 0.000  
## concavity\_se 0.000  
## area\_se 0.000  
## fractal\_dimension\_worst 0.000  
## fractal\_dimension\_mean 0.000  
## smoothness\_mean 0.000  
## fractal\_dimension\_se 0.000  
## radius\_se 0.000  
## texture\_mean 0.000

## Predict testing Data Based off of Training Model

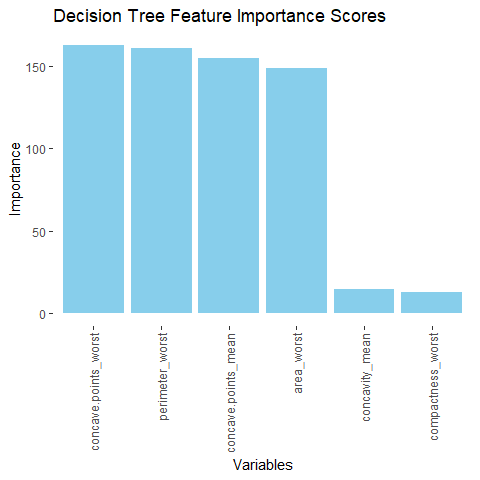
# Predicting new data to see performance  
predictions <- predict(best\_model, newdata = test\_data, type= 'class')  
  
# Reformatting data to fit confusion matrix  
actual\_labels <- factor(test\_data$diagnosis, levels = c("B", "M"))  
predicted\_labels <- factor(predictions, levels = c("B", "M"))  
  
# Creating confusion matrix  
conf\_mat <- confusionMatrix(data = predicted\_labels, reference = actual\_labels)  
conf\_mat

## 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 : 0.9718   
## Specificity : 0.9524   
## Pos Pred Value : 0.9718   
## Neg Pred Value : 0.9524   
## Prevalence : 0.6283   
## Detection Rate : 0.6106   
## Detection Prevalence : 0.6283   
## Balanced Accuracy : 0.9621   
##   
## 'Positive' Class : B   
##

cm = as.data.frame(conf\_mat$table)  
  
conf\_plot <- ggplot(data = cm, aes(x = Reference, y = Prediction, fill = Freq)) +  
 geom\_tile() +  
 geom\_text(aes(label = Freq), vjust = 1) +  
 labs(x = "Actual", y = "Predicted", title = "Decision Tree Confusion Matrix") +  
 scale\_fill\_gradient(low = "white", high = "green") + # Adjust colors as needed  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust = 1),  
 plot.background = element\_rect(fill = "white"),  
 panel.background = element\_rect(fill = "white"))  
   
ggsave("./Images/DecisionTree\_Confusion\_Matrix.png", plot = conf\_plot, width = 8, height = 6, dpi = 300)  
conf\_plot



importance <- as.data.frame(varImp(best\_model))  
  
importance\_scores <- importance$Overall  
importance\_labels <- rownames(importance)  
importance\_data <- data.frame(importance\_scores, importance\_labels)  
  
importance\_plot <- ggplot(data = head(importance\_data, 6), aes(x = reorder(importance\_labels, -importance\_scores), y = importance\_scores)) +   
 labs(x = "Variables", y = "Importance", title = "Decision Tree Feature Importance Scores") +  
 geom\_bar(stat = "identity", fill = "skyblue") +  
 theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust = 1),  
 plot.background = element\_rect(fill = "white"),  
 panel.background = element\_rect(fill = "white"))  
   
ggsave("./Images/DecisionTree\_Importance\_Plot.png", plot = importance\_plot, width = 8, height = 6, dpi = 300)  
importance\_plot

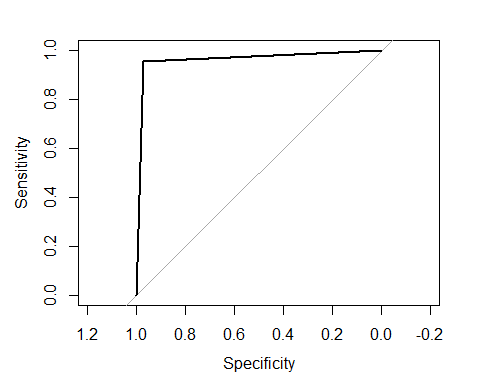


# Assuming 'predictions' contains class labels from the decision tree  
# Convert predicted labels to binary probabilities  
predicted\_probabilities <- ifelse(predictions == "M", 1, 0) # Assuming 'M' represents positive class  
  
# Compute ROC curve  
roc\_curve <- roc(actual\_labels, predicted\_probabilities)

## Setting levels: control = B, case = M

## Setting direction: controls < cases

# Plot ROC curve using ggplot  
plot(roc\_curve)



## Logistic Model

# Converting the factor levels to a binary format for logistic regression model  
# 1 = 'M' (malignant) 0 = 'B' (benign)  
train\_data$diagnosis <- as.factor(ifelse(train\_data$diagnosis == "M", 1, 0))  
test\_data$diagnosis <- as.factor(ifelse(test\_data$diagnosis == "M", 1, 0))  
  
# Train a logistic regression model using the glm function with binomial family parameter  
logistic\_model <- glm(diagnosis ~ ., data = train\_data, family = "binomial")

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

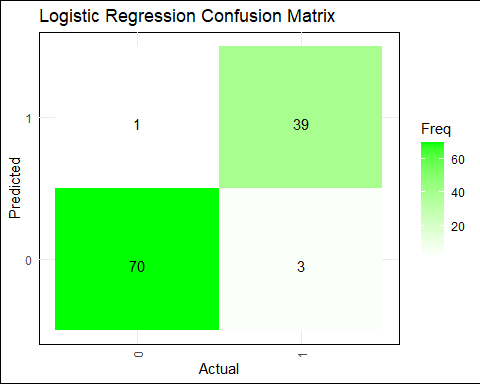
# Summary of the model to view coefficients and statistics  
summary(logistic\_model)

##   
## Call:  
## glm(formula = diagnosis ~ ., family = "binomial", data = train\_data)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)  
## (Intercept) -7.193e+02 5.502e+05 -0.001 0.999  
## id -2.065e-07 1.467e-04 -0.001 0.999  
## radius\_mean -7.078e+02 1.043e+05 -0.007 0.995  
## texture\_mean 3.477e+01 4.010e+03 0.009 0.993  
## perimeter\_mean 4.376e+01 1.014e+04 0.004 0.997  
## area\_mean 3.044e+00 6.817e+02 0.004 0.996  
## smoothness\_mean 7.850e+02 1.711e+06 0.000 1.000  
## compactness\_mean -1.079e+04 1.646e+06 -0.007 0.995  
## concavity\_mean 1.730e+03 1.240e+06 0.001 0.999  
## concave.points\_mean 1.385e+04 3.415e+06 0.004 0.997  
## symmetry\_mean -1.591e+03 2.882e+05 -0.006 0.996  
## fractal\_dimension\_mean 6.662e+03 4.121e+06 0.002 0.999  
## radius\_se -2.197e+02 2.473e+05 -0.001 0.999  
## texture\_se 1.137e+02 2.744e+04 0.004 0.997  
## perimeter\_se -5.250e+01 2.618e+04 -0.002 0.998  
## area\_se 8.471e+00 1.979e+03 0.004 0.997  
## smoothness\_se -4.507e+04 1.016e+07 -0.004 0.996  
## compactness\_se 1.623e+04 4.287e+06 0.004 0.997  
## concavity\_se -6.819e+03 1.972e+06 -0.003 0.997  
## concave.points\_se 2.792e+04 3.329e+06 0.008 0.993  
## symmetry\_se -6.491e+03 2.991e+06 -0.002 0.998  
## fractal\_dimension\_se -1.449e+05 1.722e+07 -0.008 0.993  
## radius\_worst 1.826e+02 6.658e+04 0.003 0.998  
## texture\_worst -1.199e+01 3.237e+03 -0.004 0.997  
## perimeter\_worst 7.539e+00 3.693e+03 0.002 0.998  
## area\_worst -9.043e-01 8.199e+02 -0.001 0.999  
## smoothness\_worst 2.204e+03 1.245e+06 0.002 0.999  
## compactness\_worst -1.028e+03 8.421e+05 -0.001 0.999  
## concavity\_worst 9.492e+02 1.570e+05 0.006 0.995  
## concave.points\_worst -1.332e+03 1.175e+06 -0.001 0.999  
## symmetry\_worst 2.243e+03 3.741e+05 0.006 0.995  
## fractal\_dimension\_worst 1.167e+04 3.226e+06 0.004 0.997  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 6.0231e+02 on 455 degrees of freedom  
## Residual deviance: 1.0595e-06 on 424 degrees of freedom  
## AIC: 64  
##   
## Number of Fisher Scoring iterations: 25

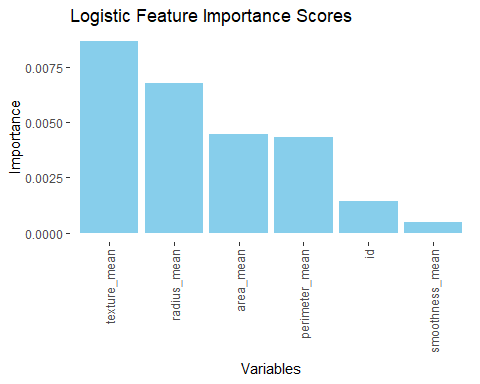
## Logistical Confusion Matrix

# Make predictions on the test data  
predictions <- predict(logistic\_model, newdata = test\_data, type = "response")  
  
# Convert predicted probabilities to binary predictions based on a 0.5 cutoff  
binary\_predictions <- ifelse(predictions > 0.5, "1", "0")  
  
# Create a confusion matrix to evaluate the model  
confusion\_matrix <- confusionMatrix(as.factor(binary\_predictions), test\_data$diagnosis)

## PLOTTING CONFUSION MATRIX  
cm = as.data.frame(confusion\_matrix$table)  
  
conf\_plot <- ggplot(data = cm, aes(x = Reference, y = Prediction, fill = Freq)) +  
 geom\_tile() +  
 geom\_text(aes(label = Freq), vjust = 1) +  
 labs(x = "Actual", y = "Predicted", title = "Logistic Regression Confusion Matrix") +  
 scale\_fill\_gradient(low = "white", high = "green") + # Adjust colors as needed  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust = 1),  
 plot.background = element\_rect(fill = "white"),  
 panel.background = element\_rect(fill = "white"))  
ggsave("./Images/Logistic\_ConfusionMatrix\_Plot.png", plot = importance\_plot, width = 8, height = 6, dpi = 300)  
  
print(conf\_plot)



importance <- as.data.frame(varImp(logistic\_model))  
  
importance\_scores <- importance$Overall  
importance\_labels <- rownames(importance)  
importance\_data <- data.frame(importance\_scores, importance\_labels)  
  
importance\_plot <- ggplot(data = head(importance\_data, 6), aes(x = reorder(importance\_labels, -importance\_scores), y = importance\_scores)) +   
 labs(x = "Variables", y = "Importance", title = "Logistic Feature Importance Scores") +  
 geom\_bar(stat = "identity", fill = "skyblue") +  
 theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust = 1),  
 plot.background = element\_rect(fill = "white"),  
 panel.background = element\_rect(fill = "white"))  
ggsave("./Images/logistic\_Importance\_Plot.png", plot = importance\_plot, width = 8, height = 6, dpi = 300)  
  
importance\_plot



# Defining classes  
binary\_predictions <- ifelse(predictions > 0.5, "M", "B")  
# Coping feature diagnosis  
test\_data$diagnosis\_binary <- test\_data$diagnosis  
  
# Creating ROC curve from test and prediction datasets  
roc\_curve <- roc(test\_data$diagnosis\_binary, as.numeric(binary\_predictions == "M"))

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

## Setting direction: controls < cases

# or use plot(roc\_curve) to plot curve  
  
# creating dataframes to be used in ggplot  
roc\_df <- data.frame(  
 FPR = roc\_curve$specificities,  
 TPR = roc\_curve$sensitivities  
)  
  
ggplot(roc\_df, aes(x = FPR, y = TPR)) +  
 geom\_line(color = "blue") +  
 geom\_abline(slope = 1, intercept = 0, linetype = "dashed") +   
 labs(title = "Receiver Operating Characteristic (ROC) Curve", x = "False Positive Rate (1 - Specificity)", y = "True Positive Rate (Sensitivity)") +  
 theme\_minimal()

