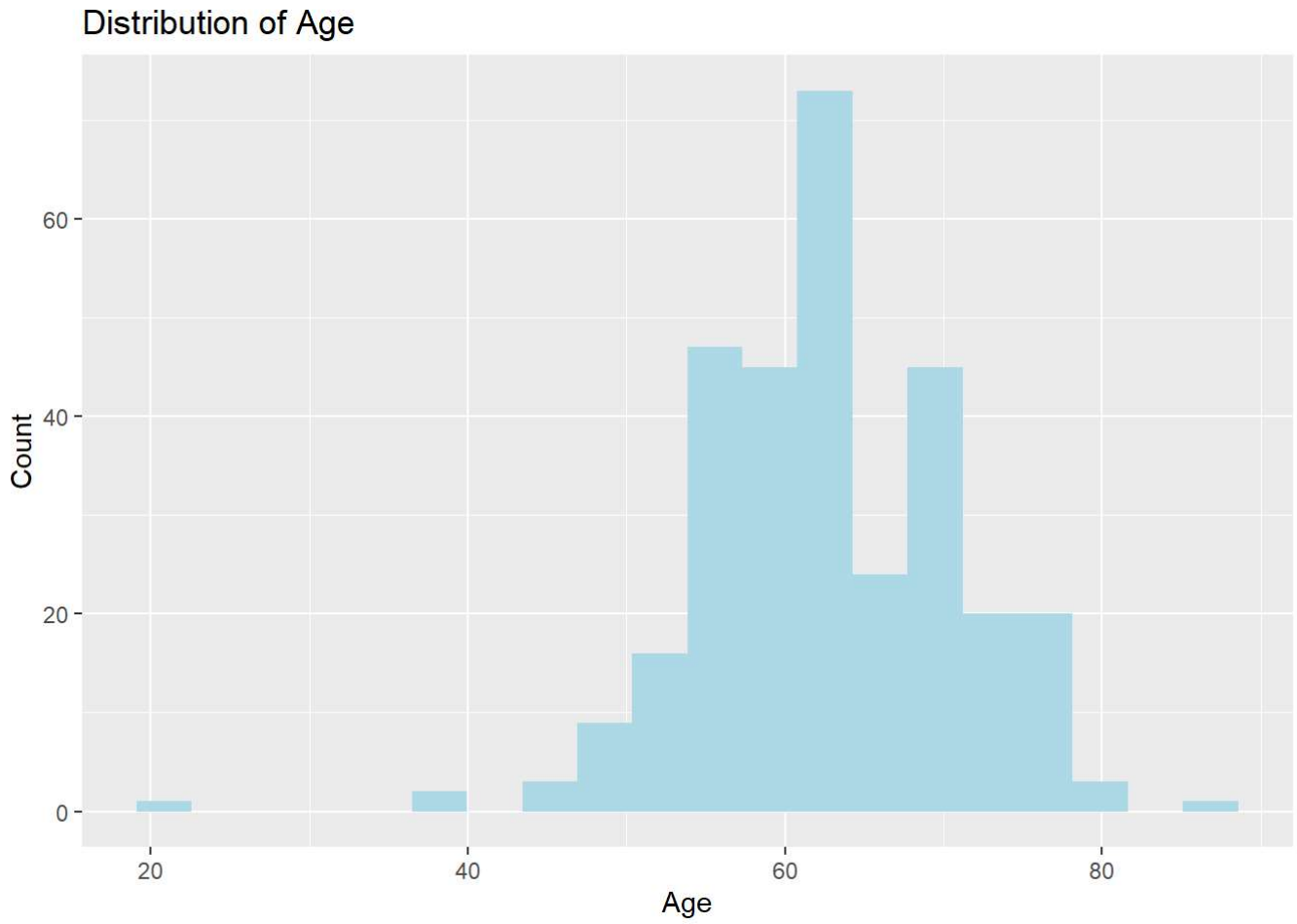
Machine learning Project-Kanimozhi Subramanian

2023-04-11

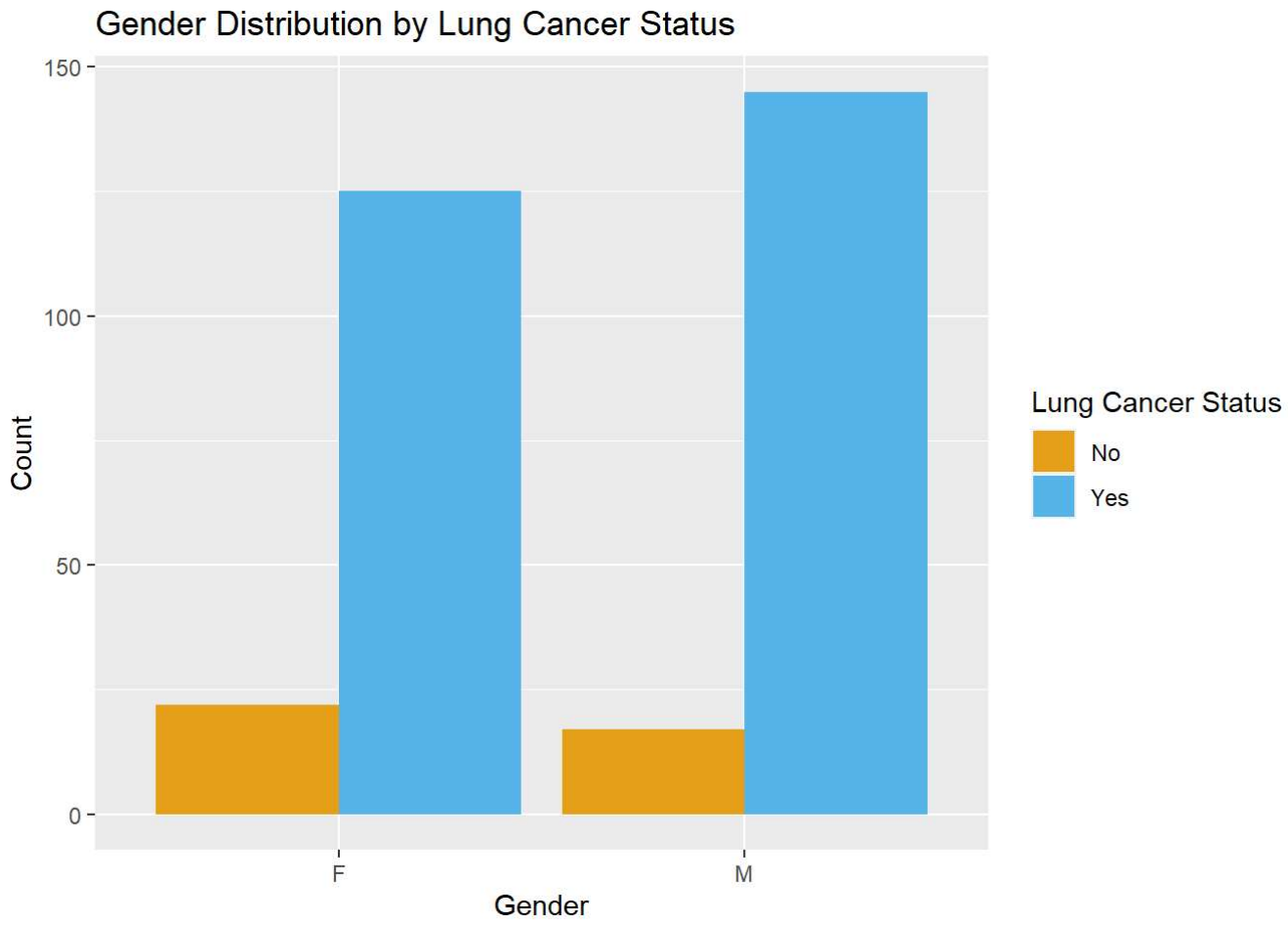
Distribution of Age

```
library(ggplot2)
lungcancer <- read.csv("C:/Users/RENJITH/Downloads/lung_cancer.csv")
ggplot(data = lungcancer, aes(x = AGE)) +
  geom_histogram(fill = "lightblue", bins = 20) +
  labs(title = "Distribution of Age", x = "Age", y = "Count")
```



Gender Distribution plot by Lung Cancer status

```
library(ggplot2)
Lungcancerdata<-read.csv("C:/Users/RENJITH/Downloads/lung_cancer.csv")
ggplot(data = Lungcancerdata, aes(x = GENDER, fill = LUNG_CANCER)) +
  geom_bar(position = "dodge") +
  labs(title = "Gender Distribution by Lung Cancer Status", x = "Gender", y = "Count") +
  scale_fill_manual(values = c("#E69F18", "#56B4E9"), name = "Lung Cancer Status",
                    labels = c("No", "Yes"))
```



Correlation plot

```
# Load required package
library(ggplot2)
library(psych)
```

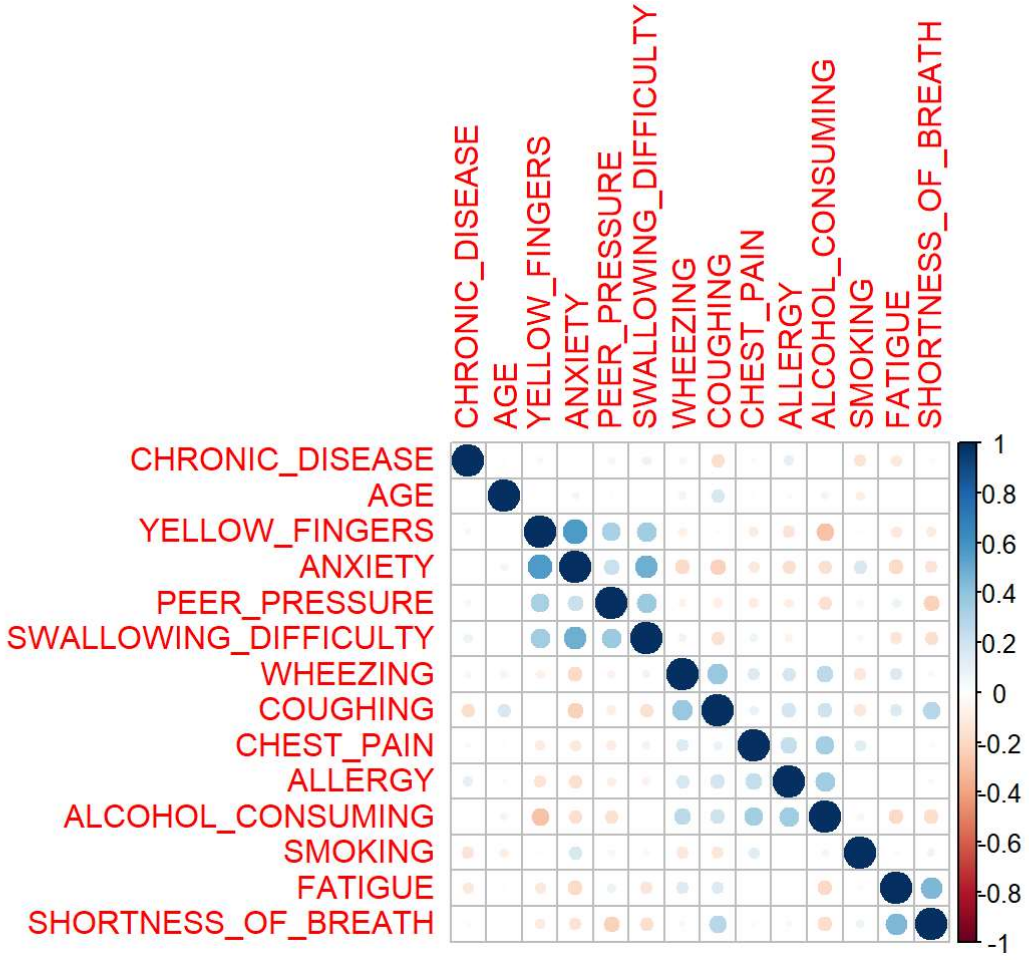
```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
# Load data
df <- read.csv("C:/Users/RENJITH/Downloads/lung_cancer.csv")

# Create correlation matrix
corr_matrix <- cor(df[,2:15])
corrplot(cor(df[ ,2:15]), method = "circle", order = "hclust")
```



Full Logistic Regression

```
library(pROC)
```

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

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

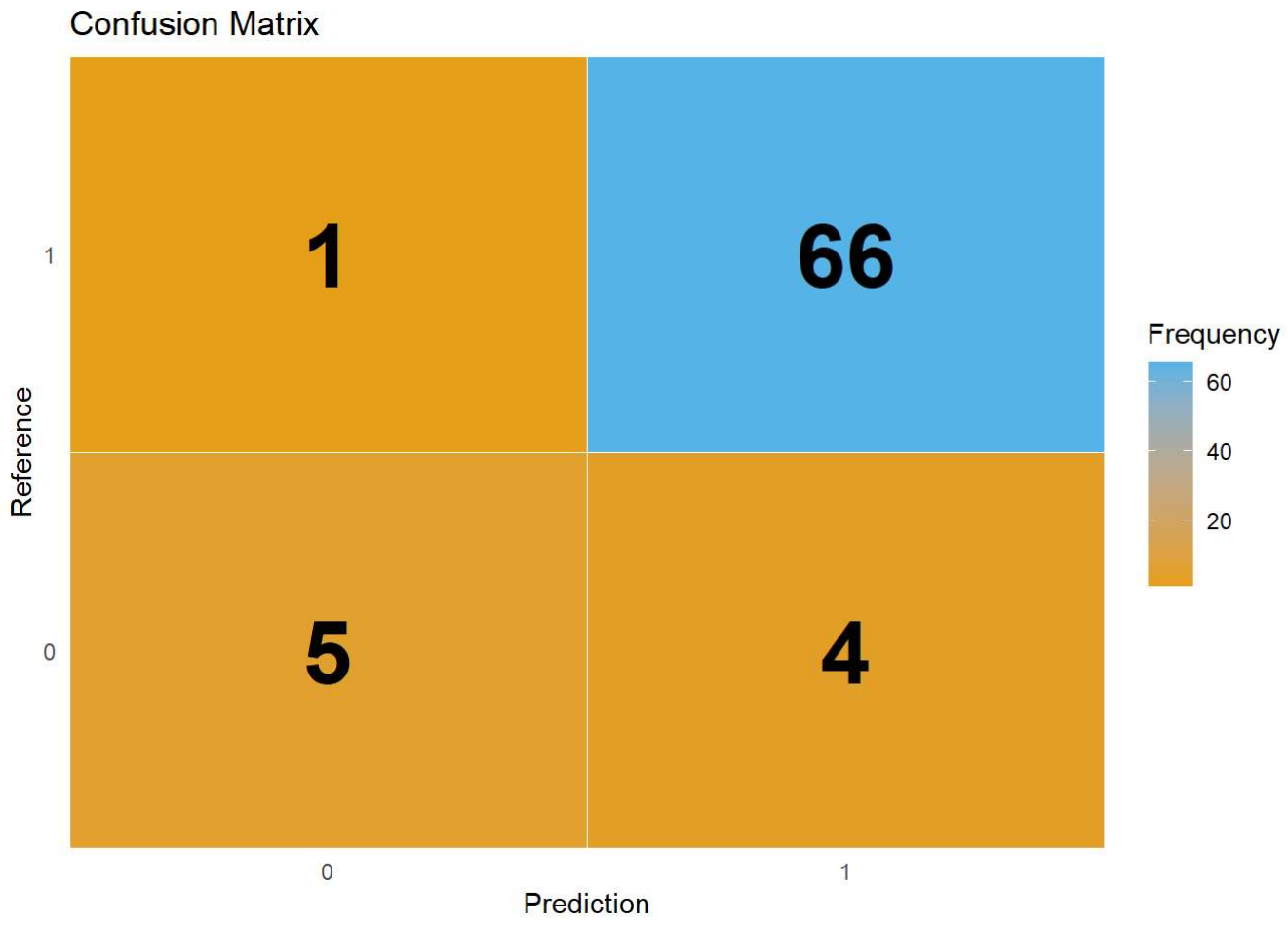
```
library(magrittr)
library(caret)
```

```
## Loading required package: lattice
```

```
library(tinytex)

#Load the dataset
Lungcancerdata<-read.csv("C:/Users/RENJITH/Downloads/lung_cancer.csv")

#Check for missing values
Lungcancerdata <- na.omit(Lungcancerdata)

# Splitting the dataset into train and test data
set.seed(123)
training.samples <- Lungcancerdata$LUNG_CANCER %>% createDataPartition(p = 0.75, list = FALSE)
train.data <- Lungcancerdata[training.samples, ]
test.data <- Lungcancerdata[-training.samples, ]

#Splitting the Input predictors and output for train data
x <- model.matrix(LUNG_CANCER~., train.data)[,-1]
y <- ifelse(train.data$LUNG_CANCER == "YES", 1, 0)
train.data$LUNG_CANCER<- ifelse(train.data$LUNG_CANCER == "YES", 1, 0)

# Build full logistic model
full.model <- glm(LUNG_CANCER ~., data = train.data, family = binomial)
summary(full.model)
```

```
##
## Call:
## glm(formula = LUNG_CANCER ~ ., family = binomial, data = train.data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.59203   0.00717   0.04693   0.13912   2.43719
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -34.57689    7.25609  -4.765 1.89e-06 ***
## GENDERM                -0.96921    0.87858  -1.103  0.26996
## AGE                     0.03801    0.03619   1.050  0.29353
## SMOKING                 2.57253    0.92863   2.770  0.00560 **
## YELLOW_FINGERS          1.65049    0.88892   1.857  0.06335 .
## ANXIETY                 0.89984    0.96248   0.935  0.34983
## PEER_PRESSURE           1.33518    0.83059   1.608  0.10794
## CHRONIC_DISEASE         3.63427    1.11163   3.269  0.00108 **
## FATIGUE                 3.14336    0.95884   3.278  0.00104 **
## ALLERGY                 1.07228    0.91578   1.171  0.24164
## WHEEZING                0.77089    1.04607   0.737  0.46116
## ALCOHOL_CONSUMING       1.79722    1.02077   1.761  0.07830 .
## COUGHING                4.21401    1.44857   2.909  0.00362 **
## SHORTNESS_OF_BREATH    -1.23642    0.90239  -1.370  0.17064
## SWALLOWING_DIFFICULTY   3.78045    1.46289   2.584  0.00976 **
## CHEST_PAIN              1.18806    0.88395   1.344  0.17893
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 178.950  on 232  degrees of freedom
## Residual deviance:  64.449  on 217  degrees of freedom
## AIC: 96.449
##
## Number of Fisher Scoring iterations: 8
```

```
# Make predictions
probabilities <- full.model %>% predict(test.data, type = "response")

# Model accuracy
predicted.classes <- ifelse(probabilities > 0.5, "YES", "NO")

# Change the predicted classes and observed classes into levels of 0 and 1.
observed.classes <- test.data$LUNG_CANCER
observed.classes<-ifelse(observed.classes == "YES", 1, 0)
predicted.classes<-ifelse(predicted.classes == "YES", 1, 0)


observed.classes<-factor(observed.classes,levels = c(0,1))
predicted.classes<-factor(predicted.classes,levels=c(0,1))

# Model Accuracy
cm<-confusionMatrix(predicted.classes,observed.classes)

# create the confusion matrix plot
library(ggplot2)
cm_plot <- ggplot(data = as.data.frame(cm$table),
                  aes(x = Prediction, y = Reference, fill = as.numeric(Freq))) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "#E69F18", high = "#56B4E9") +
  theme_minimal() +
  labs(title = "Confusion Matrix", x = "Prediction", y = "Reference") +
  geom_text(aes(label = Freq), size = 12, fontface = "bold") +
  scale_x_discrete(expand = c(0, 0.1)) +
  scale_y_discrete(expand = c(0, 0.1))+
  guides(fill = guide_colorbar(title = "Frequency"))
cm_plot
```

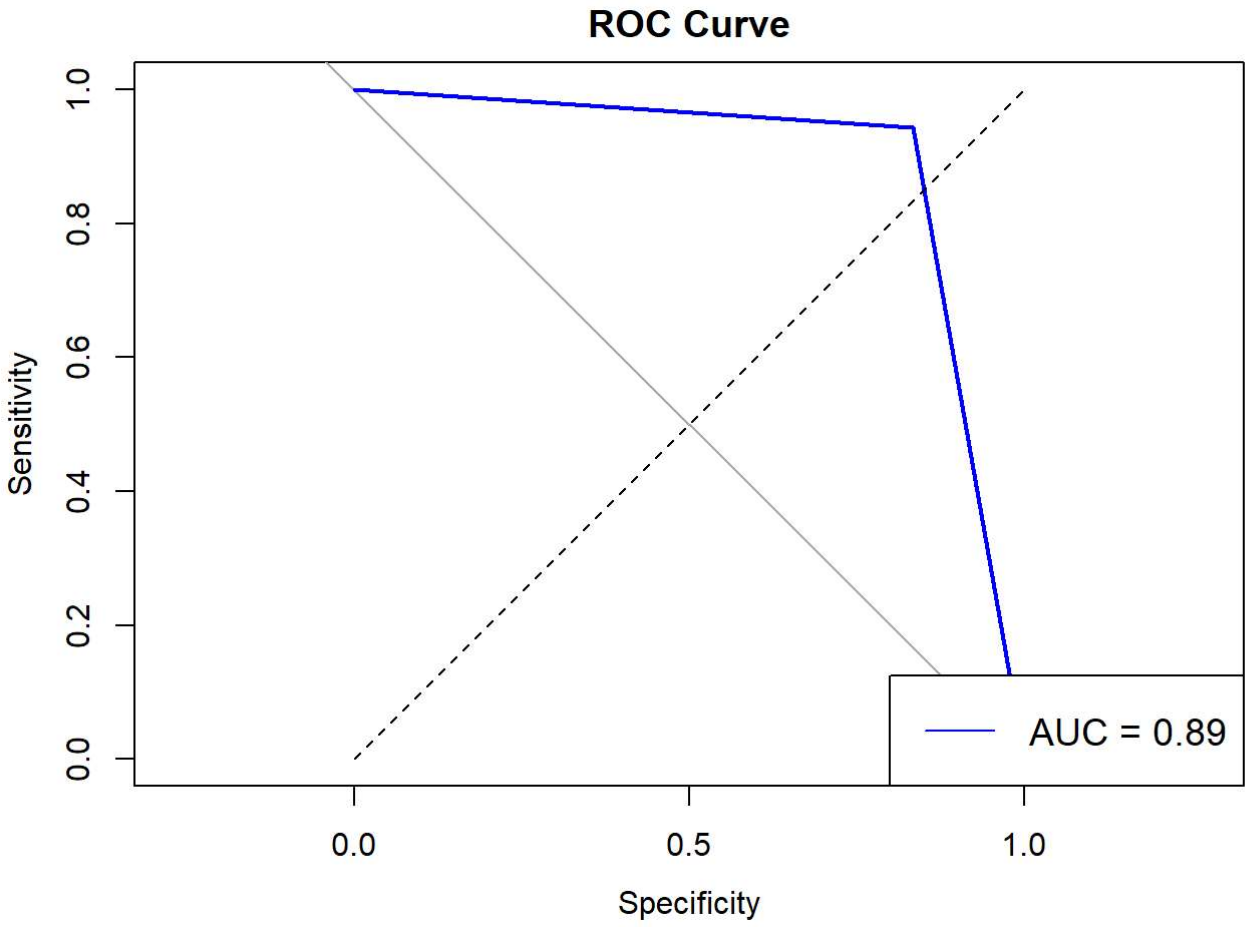



```
# Change the predicted classes and observed classes to numeric
predicted.classes <- as.numeric(predicted.classes)
observed.classes <- as.numeric(observed.classes)

# Calculate the ROC curve using predicted probabilities
roc_curve <- roc(predicted.classes, observed.classes)
```

```
## Setting levels: control = 1, case = 2
```

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

```
# Plot the ROC curve
plot(roc_curve, col = "blue", main = "ROC Curve",xlim=c(0,1))
lines(x = c(0, 1), y = c(0, 1), lty = 2)
legend("bottomright", legend = paste0("AUC = ", round(auc(roc_curve), 2)), col = "blue", lty =
1, cex = 1.2)
```



Ridge Regression

```
#Ridge Regression

library(glmnet)
```

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

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

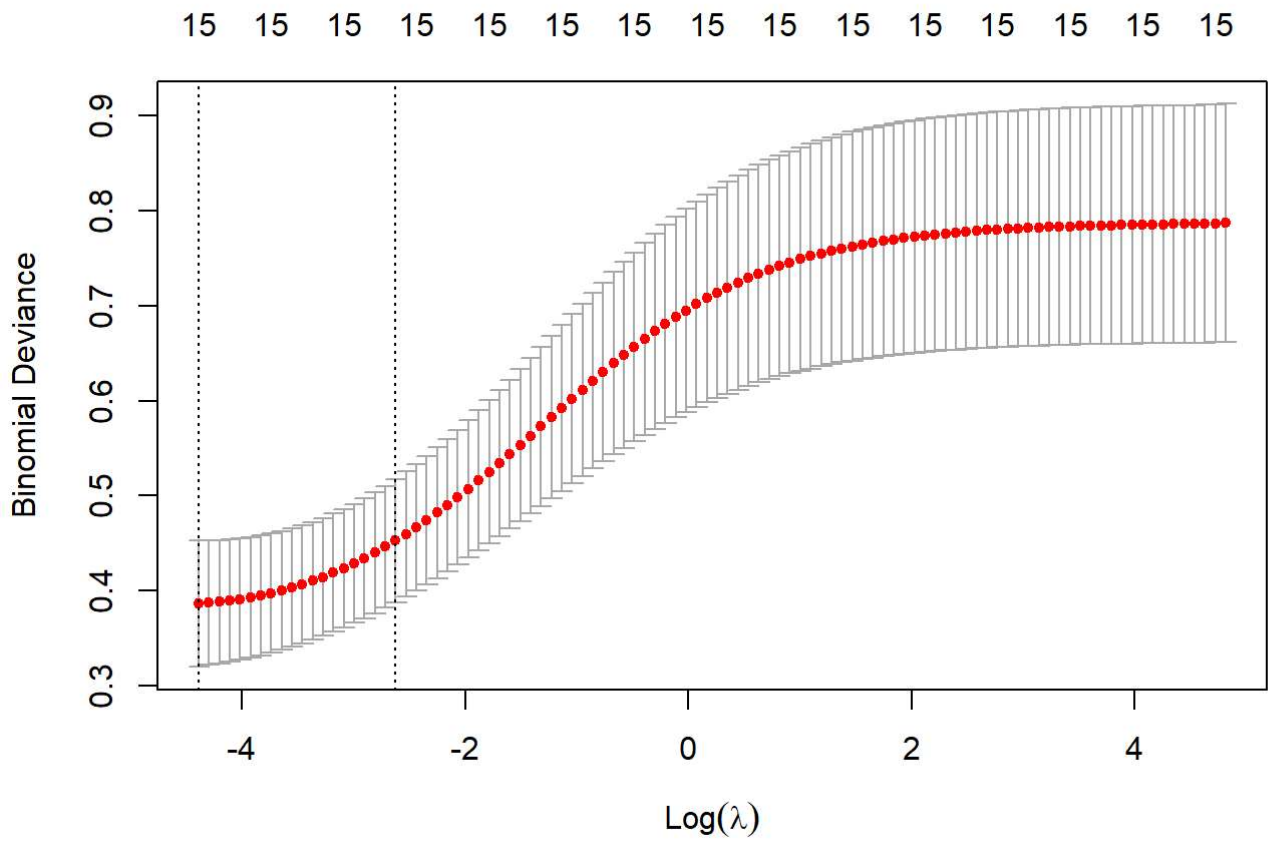
```
library(mlbench)
library(glmnet)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(caret)

#Splitting the dataset into train data and test data
training.samples <- Lungcancerdata$LUNG_CANCER %>%createDataPartition(p = 0.75, list = FALSE)
train.data <- Lungcancerdata[training.samples, ]
test.data <- Lungcancerdata[-training.samples, ]

# Splitting the dataset to input predictors and output variable
x <- model.matrix(LUNG_CANCER~., train.data)[,-1]

# Convert the outcome (class) to a numerical variable
y <- ifelse(train.data$LUNG_CANCER == "YES", 1, 0)

#Build Ridge Regression with cross validation
set.seed(1234)
cv.ridge <- cv.glmnet(x, y, alpha = 0, family = "binomial", lambda = NULL)
plot(cv.ridge)
```



```
#Fit the model with optimal lambda value
cv.ridge$lambda.min
```

```
## [1] 0.01237494
```

```
model <- glmnet(x, y, alpha = 0, family = "binomial",lambda = cv.ridge$lambda.min)

# Display regression coefficients
coef(model)
```

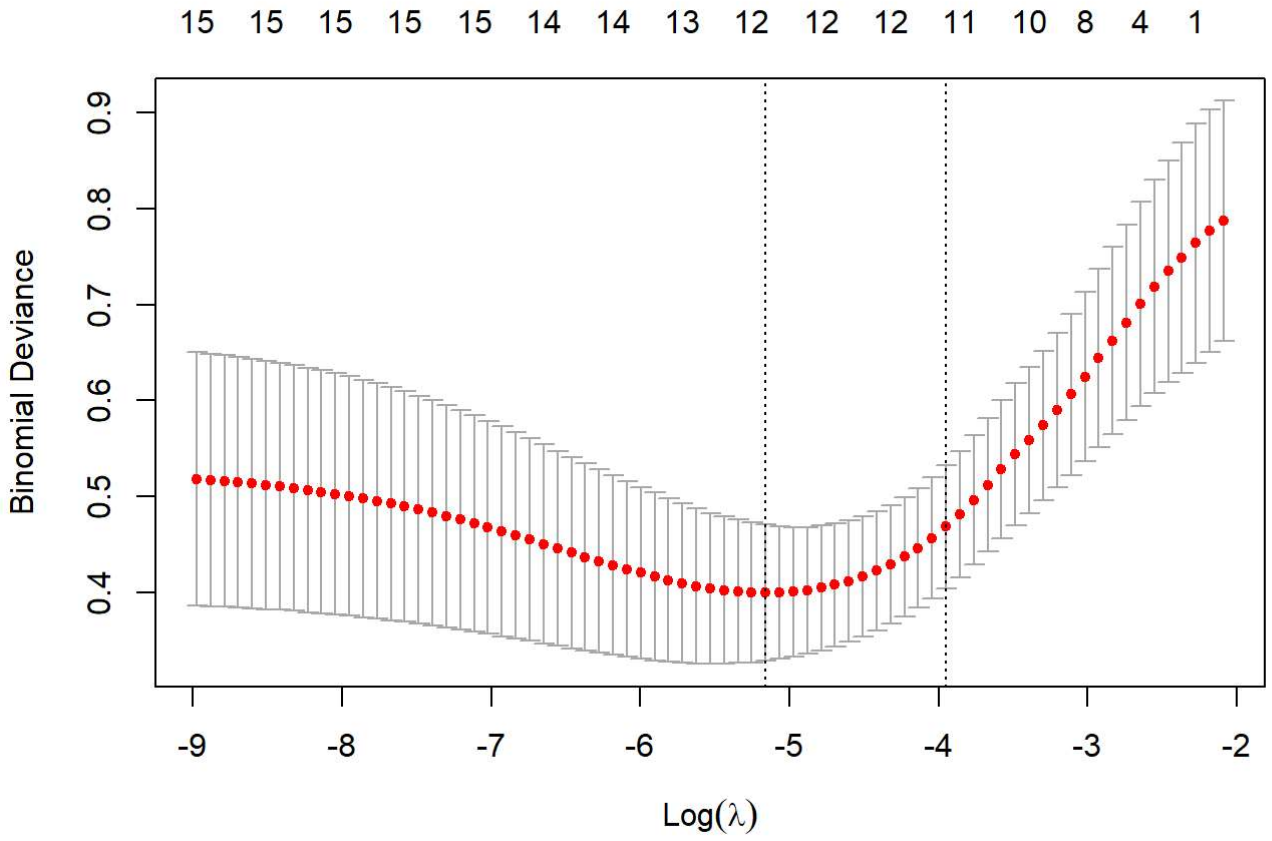
```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                                 s0
## (Intercept)           -19.55283635
## GENDERM                 0.07225213
## AGE                     0.01060799
## SMOKING                 1.15036666
## YELLOW_FINGERS          0.90480643
## ANXIETY                 0.61735109
## PEER_PRESSURE           1.31959857
## CHRONIC_DISEASE         1.41940717
## FATIGUE                 1.48634719
## ALLERGY                 1.75681969
## WHEEZING                1.15777757
## ALCOHOL_CONSUMING       1.13929246
## COUGHING                1.38411778
## SHORTNESS_OF_BREATH    -0.00464646
## SWALLOWING_DIFFICULTY   1.30562290
## CHEST_PAIN              0.72063445
```

```
x.test <- model.matrix(LUNG_CANCER ~., test.data)[,-1]
probabilities <- model %>% predict(newx = x.test)

predicted.classes <- ifelse(probabilities > 0.5, "YES", "NO")
observed.classes <- test.data$LUNG_CANCER
mean(predicted.classes == observed.classes)
```

```
## [1] 0.9078947
```

Lasso Regression

```
# Lasso regression

# Fit the Lasso Regression with cross validation
set.seed(1234)
cv.lasso <- cv.glmnet(x, y, alpha = 1, family = "binomial", lambda = NULL)
plot(cv.lasso)
```




```
# Optimal lambda value
cv.lasso$lambda.min
```

```
## [1] 0.005743936
```

```
# Fit the final model on the training data using optimal lambda
model <- glmnet(x, y, alpha = 1, family = "binomial",lambda = cv.lasso$lambda.min)

# Display regression coefficients
coef(model)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                                s0
## (Intercept)           -21.1681129
## GENDERM                 .
## AGE                     .
## SMOKING                 1.3162815
## YELLOW_FINGERS          0.9621853
## ANXIETY                 0.5432833
## PEER_PRESSURE           1.5437427
## CHRONIC_DISEASE         1.6928545
## FATIGUE                 1.6954910
## ALLERGY                 1.9793014
## WHEEZING                1.2256531
## ALCOHOL_CONSUMING       1.1520323
## COUGHING                1.6586023
## SHORTNESS_OF_BREATH     .
## SWALLOWING_DIFFICULTY   1.6653514
## CHEST_PAIN              0.6153009
```

```
# Make predictions on the test data
x.test <- model.matrix(LUNG_CANCER ~., test.data)[,-1]
probabilities <- model %>% predict(newx = x.test)

# Model accuracy
predicted.classes <- ifelse(probabilities > 0.5, "YES", "NO")
observed.classes <- test.data$LUNG_CANCER
mean(predicted.classes == observed.classes) # accuracy
```

```
## [1] 0.9078947
```

SVM

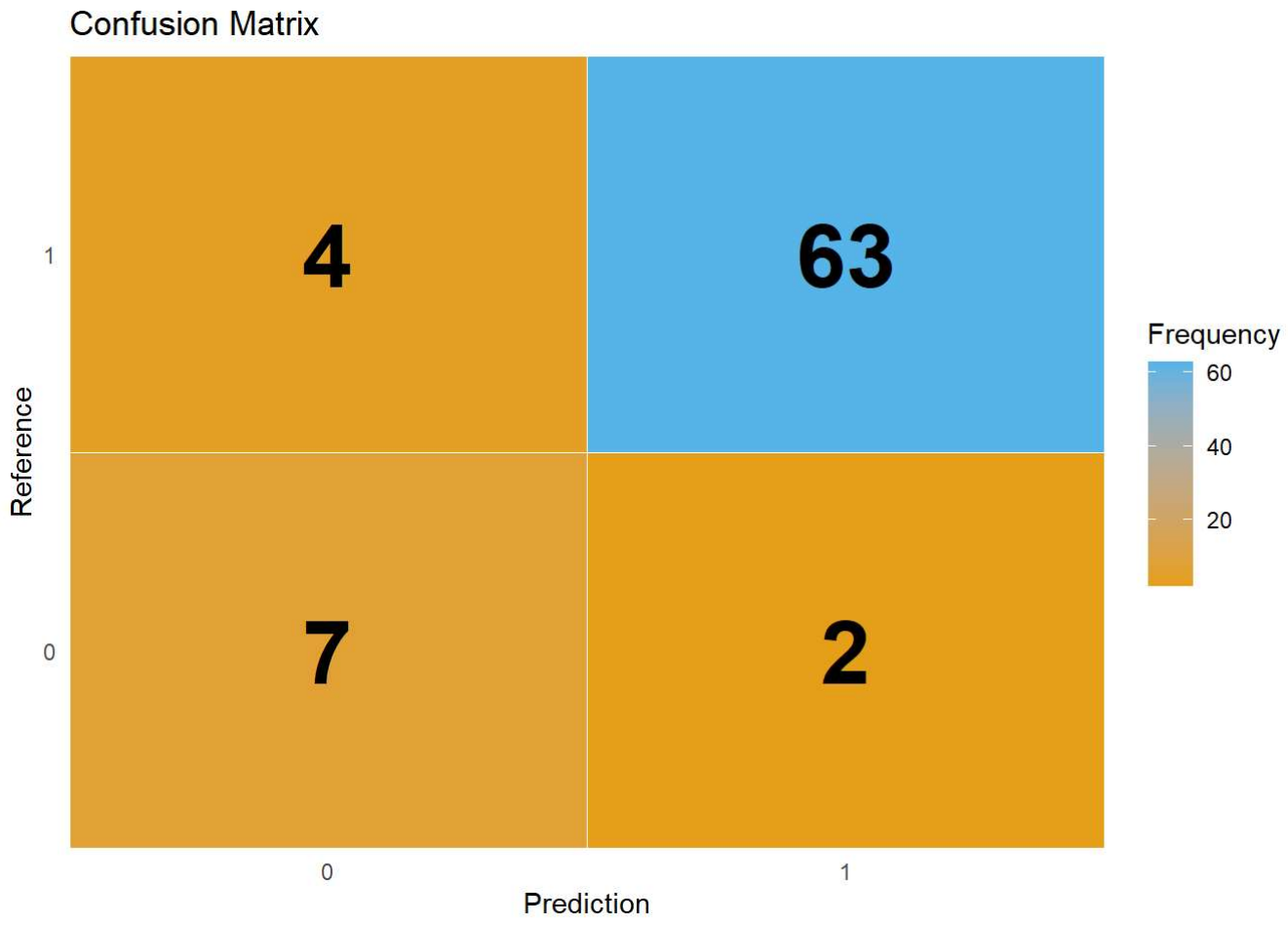
```
# Splitting the dataset
training.samples <- Lungcancerdata$LUNG_CANCER %>% createDataPartition(p = 0.75, list = FALSE)
train.data <- Lungcancerdata[training.samples, ]
test.data <- Lungcancerdata[-training.samples, ]
x.train <- train.data[,-16]

# Creating a train control object with 10-fold cross-validation
train_control <- trainControl(method = "cv", number = 10)
y.train <- ifelse(train.data$LUNG_CANCER == "YES", 1, 0)
y.train <- factor(y.train, levels=c(0,1))

# Defining the SVM model
svm_model <- train(y.train ~ ., data = cbind(x.train, y.train), method = "svmLinear", trControl
= train_control)

# Printing the tuned model's performance
print(svm_model)
```

```
## Support Vector Machines with Linear Kernel
##
## 233 samples
##  15 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 210, 209, 210, 210, 210, 209, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9181159  0.5819693
##
## Tuning parameter 'C' was held constant at a value of 1
```

```
# Splitting the test data into input predictors and output
x.test <- test.data[,-16]
y.test <- ifelse(test.data$LUNG_CANCER == "YES", 1, 0)
y.test<-factor(y.test,levels=c(0,1))

# Make the predictions
y_pred <- predict(svm_model, newdata = x.test)
y_pred <- factor(y_pred, levels=c(0,1))
y.test <- factor(y.test, levels=c(0,1))

# Calculating the accuracy of the model
# Construct the ConfusionMatrix
cm <-confusionMatrix(data = y_pred, reference = y.test)
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  7  4
##          1  2 63
##
##                Accuracy : 0.9211
##                  95% CI : (0.836, 0.9705)
##     No Information Rate : 0.8816
##     P-Value [Acc > NIR] : 0.1897
##
##                   Kappa : 0.6551
##
##  Mcnemar's Test P-Value : 0.6831
##
##             Sensitivity : 0.77778
##             Specificity : 0.94030
##          Pos Pred Value : 0.63636
##          Neg Pred Value : 0.96923
##              Prevalence : 0.11842
##          Detection Rate : 0.09211
##    Detection Prevalence : 0.14474
##       Balanced Accuracy : 0.85904
##
##        'Positive' Class : 0
##
```

```
# Confusion Matrix Plot
library(ggplot2)
cm_plot <- ggplot(data = as.data.frame(cm$table),
                  aes(x = Prediction, y = Reference, fill = as.numeric(Freq))) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "#E69F18", high = "#56B4E9") +
  theme_minimal() +
  labs(title = "Confusion Matrix", x = "Prediction", y = "Reference") +
  geom_text(aes(label = Freq), size = 12, fontface = "bold") +
  scale_x_discrete(expand = c(0, 0.1)) +
  scale_y_discrete(expand = c(0, 0.1))+
  guides(fill = guide_colorbar(title = "Frequency"))
cm_plot
```

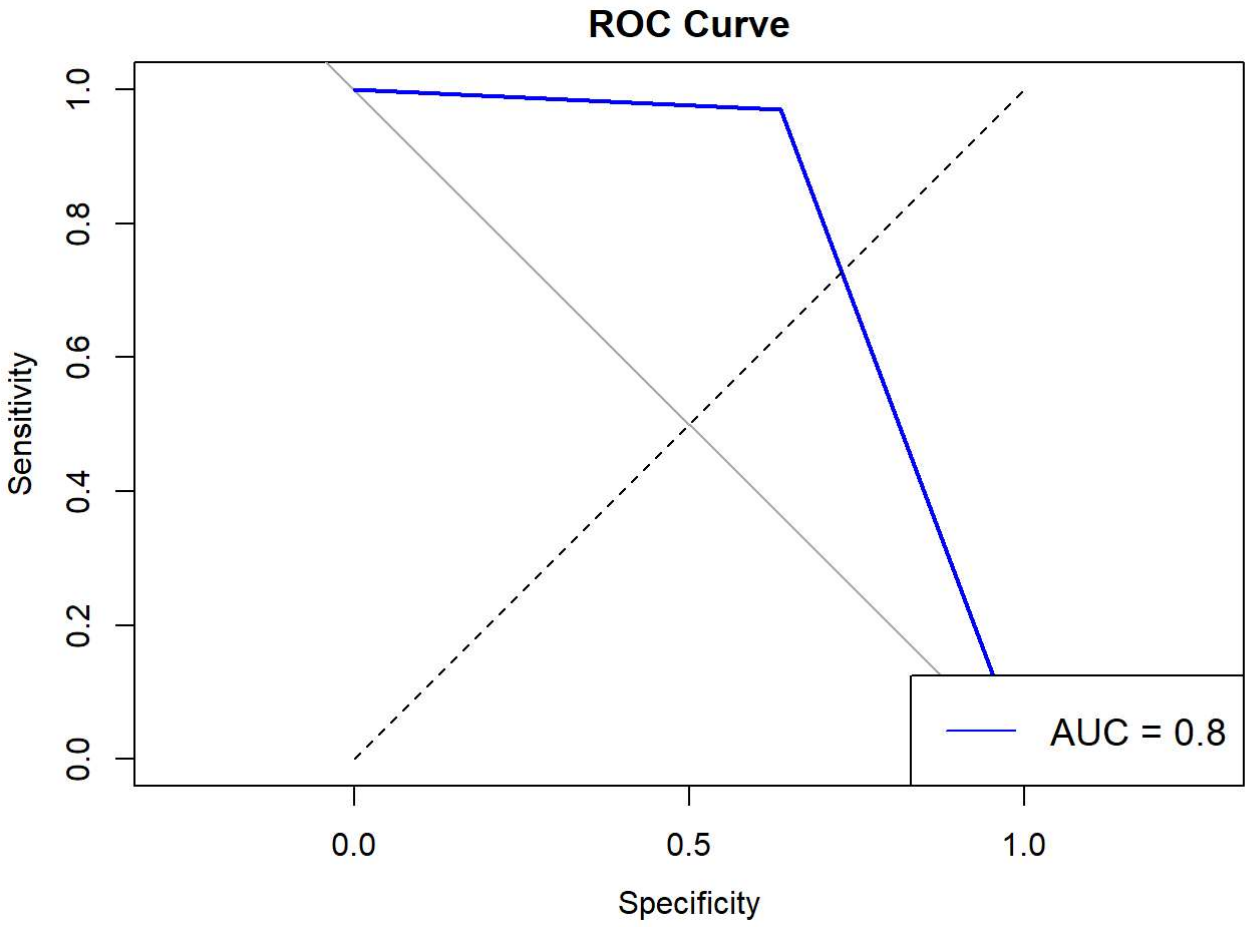



```
predicted.classes <- as.numeric(y_pred)
observed.classes <- as.numeric(y.test)

# Calculate the ROC curve using predicted probabilities
roc_curve <- roc(predicted.classes, observed.classes)
```

```
## Setting levels: control = 1, case = 2
```

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

```
# Plot the ROC curve
plot(roc_curve, col = "blue", main = "ROC Curve",xlim=c(0,1))
lines(x = c(0, 1), y = c(0, 1), lty = 2)
legend("bottomright", legend = paste0("AUC = ", round(auc(roc_curve), 2)), col = "blue", lty =
1, cex = 1.2)
```



Neural Network

```
library(neuralnet)
```

```
##
## Attaching package: 'neuralnet'
```

```
## The following object is masked from 'package:dplyr':
##
##     compute
```

```
library(caret)
library(readr)
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:psych':
##
##     alpha
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
library(e1071)

# Load the dataset
lung_data <- read_csv("C:/Users/RENJITH/Downloads/lung_cancer.csv")
```

```
## Rows: 309 Columns: 16
```

```
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (2): GENDER, LUNG_CANCER
## dbl (14): AGE, SMOKING, YELLOW_FINGERS, ANXIETY, PEER_PRESSURE, CHRONIC_DISE...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Split the dataset into training and testing sets
set.seed(123)
lung_data$LUNG_CANCER<-ifelse(lung_data$LUNG_CANCER =="YES",1,0)
lung_data$GENDER<-ifelse(lung_data$GENDER=="F",1,0)

maxs <- apply(lung_data, 2, max)
mins <- apply(lung_data, 2, min)
scaled <- as.data.frame(scale(lung_data, center = mins, scale = maxs - mins))
index <- sample(1:nrow(lung_data),round(0.75*nrow(lung_data)))

# Train-test split
train_ <- scaled[index,]
test_ <- scaled[-index,]

# Create the neural network model
n <- names(train_)
f <- as.formula(paste("LUNG_CANCER ~", paste(n[!n %in% "LUNG_CANCER"], collapse = " + ")))
nn <- neuralnet(f,data=train_,hidden=c(10,10), act.fct = "logistic", linear.output=T)

# Visual plot of the model
plot(nn)

#Make Predictions
pr.nn <- compute(nn,test_[,1:15])

# Descaling
pr.nn_ <- pr.nn$net.result*(max(lung_data$LUNG_CANCER) - min(lung_data$LUNG_CANCER)) + min(lung_
data$LUNG_CANCER)
pr.nn_ <- ifelse(pr.nn_ >= 0.5, 1, 0)


pr.nn_<-factor(pr.nn_,levels =c(0,1) )
test_$LUNG_CANCER<-factor(test_$LUNG_CANCER,levels=c(0,1))

# Confusion Matrix
cm <- confusionMatrix(pr.nn_, test_$LUNG_CANCER)
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  6  2
##          1  5 64
##
##                Accuracy : 0.9091
##                  95% CI : (0.8216, 0.9627)
##     No Information Rate : 0.8571
##     P-Value [Acc > NIR] : 0.1238
##
##                   Kappa : 0.5812
##
##  Mcnemar's Test P-Value : 0.4497
##
##             Sensitivity : 0.54545
##             Specificity : 0.96970
##          Pos Pred Value : 0.75000
##          Neg Pred Value : 0.92754
##              Prevalence : 0.14286
##          Detection Rate : 0.07792
##    Detection Prevalence : 0.10390
##       Balanced Accuracy : 0.75758
##
##        'Positive' Class : 0
##
```

```
# Confusion Matrix Plot
cm_plot <- ggplot(data = as.data.frame(cm$table),
                  aes(x = Prediction, y = Reference, fill = as.numeric(Freq))) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "#E69F18", high = "#56B4E9") +
  theme_minimal() +
  labs(title = "Confusion Matrix", x = "Prediction", y = "Reference") +
  geom_text(aes(label = Freq), size = 12, fontface = "bold") +
  scale_x_discrete(expand = c(0, 0.1)) +
  scale_y_discrete(expand = c(0, 0.1))+
  guides(fill = guide_colorbar(title = "Frequency"))
cm_plot

# Print Accuracy
print(cm$overall[1])
```

```
##  Accuracy
## 0.9090909
```