MLPH Final Project - Lung Cancer Prediction Models

Farah Beche and Han Oo

2024-05-13

```r
#Import Dataset
raw_data <- read.csv("/Users/farahbeche/Downloads/lungcancerdataset.csv")
# Explore dataset
head(raw_data)
```

```
##   GENDER AGE SMOKING YELLOW_FINGERS ANXIETY PEER_PRESSURE CHRONIC.DISEASE
## 1      M  69       1              2       2             1               1
## 2      M  74       2              1       1             1               2
## 3      F  59       1              1       1             2               1
## 4      M  63       2              2       2             1               1
## 5      F  63       1              2       1             1               1
## 6      F  75       1              2       1             1               2
##   FATIGUE ALLERGY WHEEZING ALCOHOL.CONSUMING COUGHING SHORTNESS.OF.BREATH
## 1       2       1        2                 2        2                   2
## 2       2       2        1                 1        1                   2
## 3       2       1        2                 1        2                   2
## 4       1       1        1                 2        1                   1
## 5       1       1        2                 1        2                   2
## 6       2       2        2                 1        2                   2
##   SWALLOWING.DIFFICULTY CHEST.PAIN LUNG_CANCER
## 1                     2          2         YES
## 2                     2          2         YES
## 3                     1          2          NO
## 4                     2          2          NO
## 5                     1          1          NO
## 6                     1          1         YES
```

```r
summary(raw_data)
```

```
##     GENDER               AGE           SMOKING      YELLOW_FINGERS
##  Length:309         Min.   :21.00   Min.   :1.000   Min.   :1.00
##  Class :character   1st Qu.:57.00   1st Qu.:1.000   1st Qu.:1.00
##  Mode  :character   Median :62.00   Median :2.000   Median :2.00
##                     Mean   :62.67   Mean   :1.563   Mean   :1.57
##                     3rd Qu.:69.00   3rd Qu.:2.000   3rd Qu.:2.00
##                     Max.   :87.00   Max.   :2.000   Max.   :2.00
##     ANXIETY      PEER_PRESSURE   CHRONIC.DISEASE    FATIGUE
##  Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000
##  1st Qu.:1.000   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:1.000
##  Median :1.000   Median :2.000   Median :2.000   Median :2.000
##  Mean   :1.498   Mean   :1.502   Mean   :1.505   Mean   :1.673
##  3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:2.000
##  Max.   :2.000   Max.   :2.000   Max.   :2.000   Max.   :2.000
##     ALLERGY         WHEEZING     ALCOHOL.CONSUMING    COUGHING
##  Min.   :1.000   Min.   :1.000   Min.   :1.000     Min.   :1.000
##  1st Qu.:1.000   1st Qu.:1.000   1st Qu.:1.000     1st Qu.:1.000
##  Median :2.000   Median :2.000   Median :2.000     Median :2.000
##  Mean   :1.557   Mean   :1.557   Mean   :1.557     Mean   :1.579
##  3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:2.000     3rd Qu.:2.000
##  Max.   :2.000   Max.   :2.000   Max.   :2.000     Max.   :2.000
##  SHORTNESS.OF.BREATH SWALLOWING.DIFFICULTY   CHEST.PAIN    LUNG_CANCER
##  Min.   :1.000       Min.   :1.000         Min.   :1.000   Length:309
##  1st Qu.:1.000       1st Qu.:1.000         1st Qu.:1.000   Class :character
##  Median :2.000       Median :1.000         Median :2.000   Mode  :character
##  Mean   :1.641       Mean   :1.469         Mean   :1.557
##  3rd Qu.:2.000       3rd Qu.:2.000         3rd Qu.:2.000
##  Max.   :2.000       Max.   :2.000         Max.   :2.000
```

Observations:

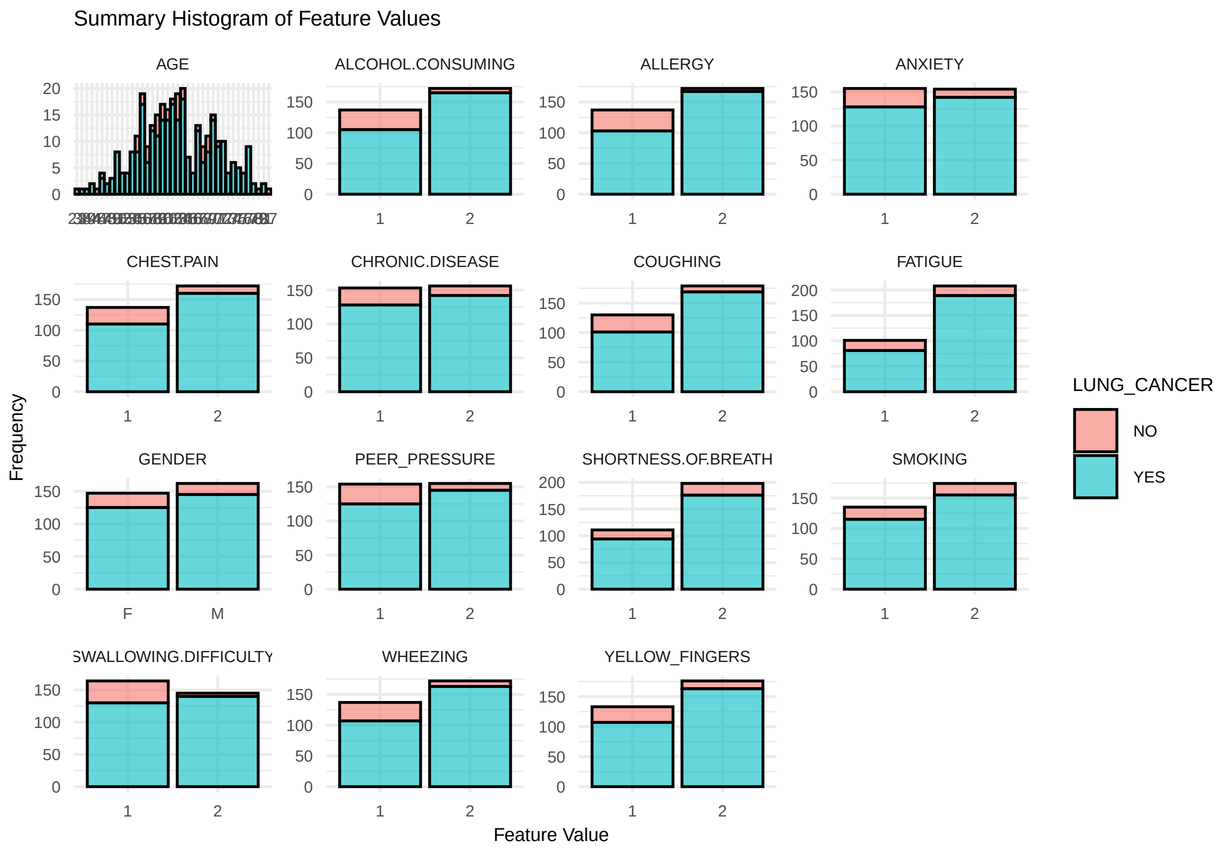
- Predictors are all binary data, except for the patient's age.
- Response variable is 'LUNG_CANCER'
- The binary predictors use different ways to represent the data
    - LUNG_CANCER is represented by YES, NO
    - GENDER is represented by M(male), F(female)
    - SMOKING, YELLOW_FINGERS, ANXIETY, etc. are represented by 1(NO), 2(YES)

```
# Explore dataset
# Check for missing values
missing_values <- colSums(is.na(raw_data))
missing_values
```

```
##                GENDER                   AGE               SMOKING
##                     0                     0                     0
##        YELLOW_FINGERS               ANXIETY         PEER_PRESSURE
##                     0                     0                     0
##       CHRONIC.DISEASE               FATIGUE               ALLERGY
##                     0                     0                     0
##              WHEEZING     ALCOHOL.CONSUMING              COUGHING
##                     0                     0                     0
##   SHORTNESS.OF.BREATH SWALLOWING.DIFFICULTY            CHEST.PAIN
##                     0                     0                     0
##           LUNG_CANCER
##                     0
```

- There are no missing values in any of the variables.

```r
# Explore dataset
# Summary histogram of feature values for each predictor variable
summary_histogram <- raw_data %>%
  gather(key = "Variable", value = "Value", -LUNG_CANCER) %>%
  ggplot(aes(x = Value, fill = as.factor(LUNG_CANCER))) +
  geom_bar(stat = "count", color = "black", alpha = 0.6) +
  facet_wrap(~ Variable, scales = "free") +
  labs(title = "Summary Histogram of Feature Values",
       x = "Feature Value",
       y = "Frequency",
       fill = "LUNG_CANCER") +
  theme_minimal() +
  theme(text = element_text(size = 7))

print(summary_histogram)
```



```r
#Data Cleaning
# Encoding GENDER, assigning 0 to "M" and 1 to "F"
# Encoding LUNG_CANCER, assigning 0 to "NO" and 1 to "YES"
# Create a new dataset 'data' with encoded variables
raw_data$GENDER <- as.numeric(ifelse(raw_data$GENDER == "M", 0, 1))
raw_data$LUNG_CANCER <- as.numeric(ifelse(raw_data$LUNG_CANCER == "NO", 0, 1))

data <- raw_data
head(data)
```

```
##   GENDER AGE SMOKING YELLOW_FINGERS ANXIETY PEER_PRESSURE CHRONIC.DISEASE
## 1      0  69       1              2       2             1               1
## 2      0  74       2              1       1             1               2
## 3      1  59       1              1       1             2               1
## 4      0  63       2              2       2             1               1
## 5      1  63       1              2       1             1               1
## 6      1  75       1              2       1             1               2
##   FATIGUE ALLERGY WHEEZING ALCOHOL.CONSUMING COUGHING SHORTNESS.OF.BREATH
## 1       2       1        2                 2        2                   2
## 2       2       2        1                 1        1                   2
## 3       2       1        2                 1        2                   2
## 4       1       1        1                 2        1                   1
## 5       1       1        2                 1        2                   2
## 6       2       2        2                 1        2                   2
##   SWALLOWING.DIFFICULTY CHEST.PAIN LUNG_CANCER
## 1                     2          2           1
## 2                     2          2           1
## 3                     1          2           0
## 4                     2          2           0
## 5                     1          1           0
## 6                     1          1           1
```

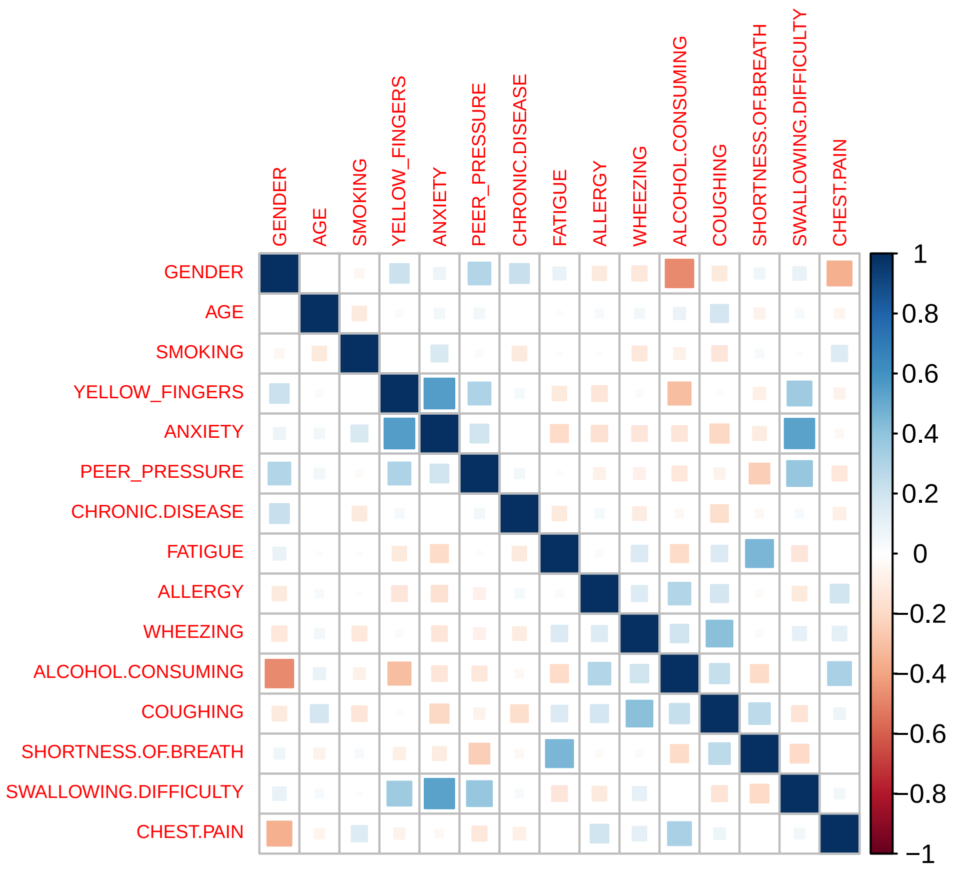
- GENDER and LUNG_CANCER were encoded to 0 and 1.

```r
# Calculate the totals
total_healthy <- sum(data$LUNG_CANCER == 0)
total_lung_cancer <- sum(data$LUNG_CANCER == 1)

# Create a dataframe
totals_df <- data.frame(
  Category = c("Healthy Patients", "Lung Cancer Patients"),
  Total = c(total_healthy, total_lung_cancer)
)

print(totals_df)
```

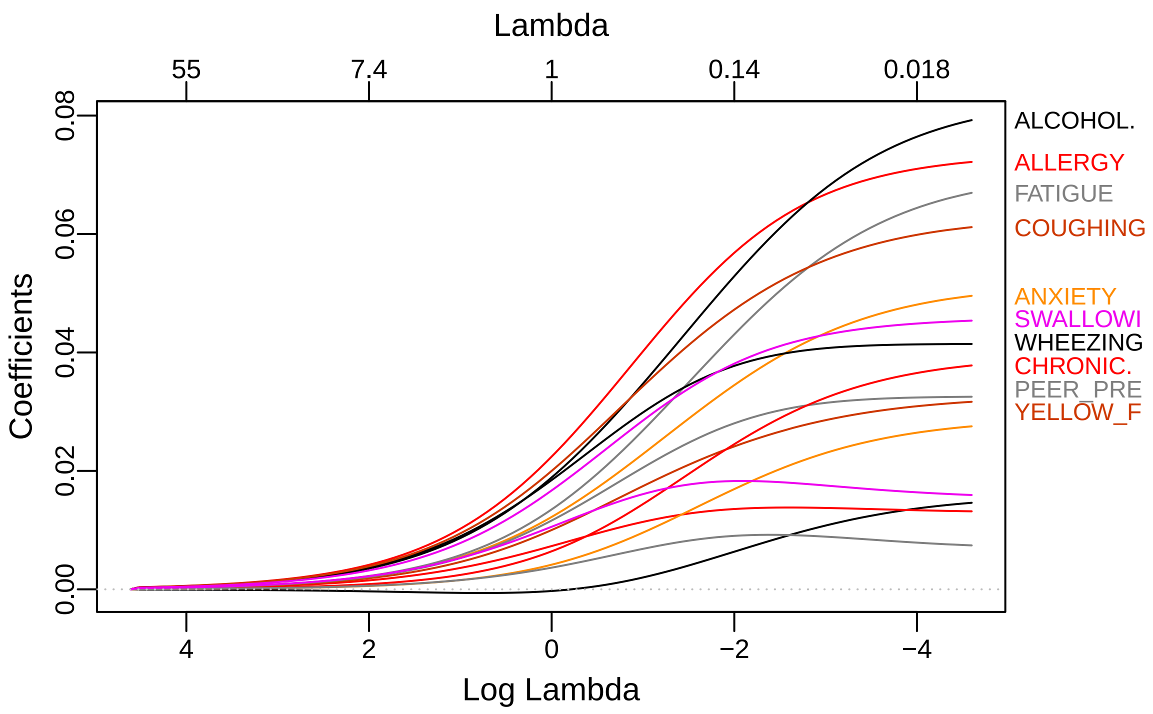
```
##               Category Total
## 1     Healthy Patients    39
## 2 Lung Cancer Patients   270
```

```r
#Split data into test and train
set.seed(123)

train_index <- sample(1:nrow(data), 0.8 * nrow(data))
train_data <- data[train_index, ]
test_data <- data[-train_index, ]
val_data <- data[-train_index, ] #validation set
```

```r
# Compute the correlation matrix
corr_df <- train_data %>% select(-LUNG_CANCER)
cor_matrix <- cor(corr_df)
# Plot correlation matrix
corrplot(cor_matrix, method = "square", tl.cex = 0.6)
```

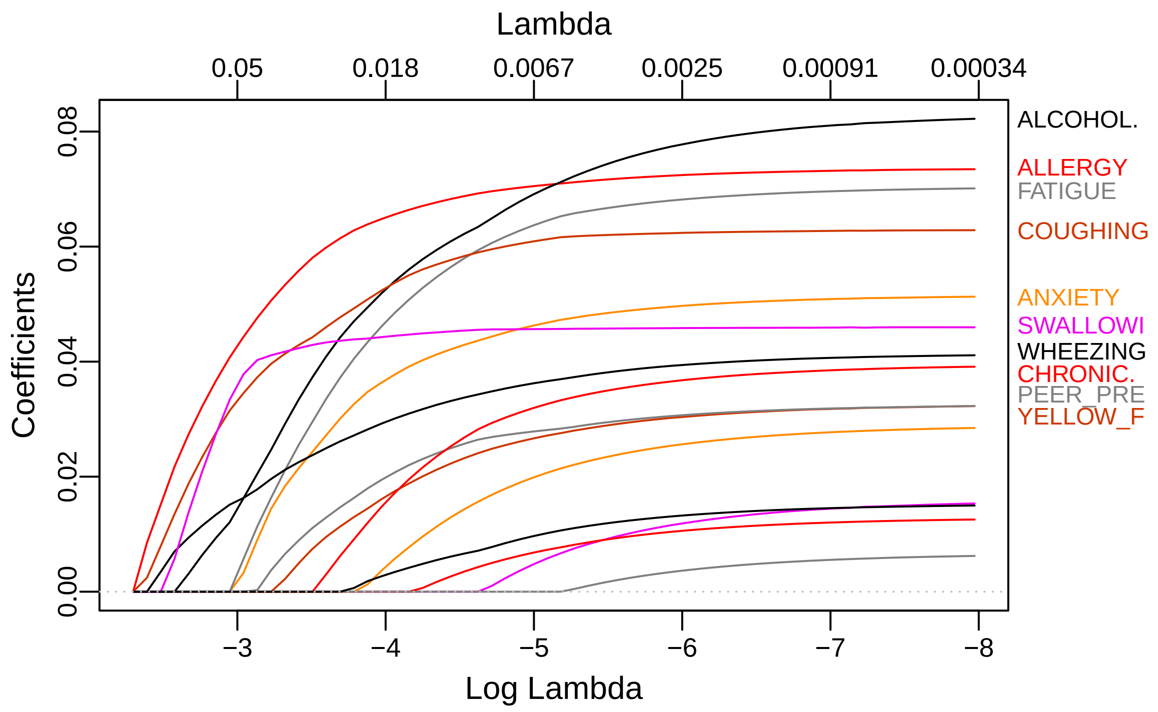


```r
# Ridge solution path
x_tr <- as.matrix(train_data[, -16])
y_tr <- train_data[, 16, drop = T]
x_te <- as.matrix(test_data[, -16])
y_te <- test_data[, 16, drop = T]
std_fit <- preProcess(x_tr, method = c("center", "scale"))
x_tr_std <- predict(std_fit, x_tr)
x_te_std <- predict(std_fit, x_te)
fit_ridge <- glmnet(x_tr_std, y_tr, alpha = 0)
plot_glmnet(fit_ridge)
```



```r
# Ridge solution path
#Standardization
set.seed(0)
cv_fit_ridge <- cv.glmnet(x_tr_std, y_tr, alpha = 0, nfolds=5)
r_tr_pred <- predict(cv_fit_ridge, newx = x_tr_std)
r_te_pred <- predict(cv_fit_ridge, newx = x_te_std)
r_tr_error <- mean((r_tr_pred - y_tr)^2)
r_te_error <- mean((r_te_pred - y_te)^2)
best_coefR <- as.vector(coef(cv_fit_ridge))
best_coefR <- c(coef(cv_fit_ridge)[1], best_coefR)
best_coefR <- best_coefR[-1]
variables <- c("GENDER", "SMOKING", "AGE", "YELLOW_FINGERS", "ANXIETY", "PEER_PRESSURE", "CHRONIC.DISEA
coef_names <- c("(Intercept)", variables)
ridge_df <- data.frame(coef_names, best_coefR, Method = "Ridge Regression", Test_Error = r_te_error) %>
```

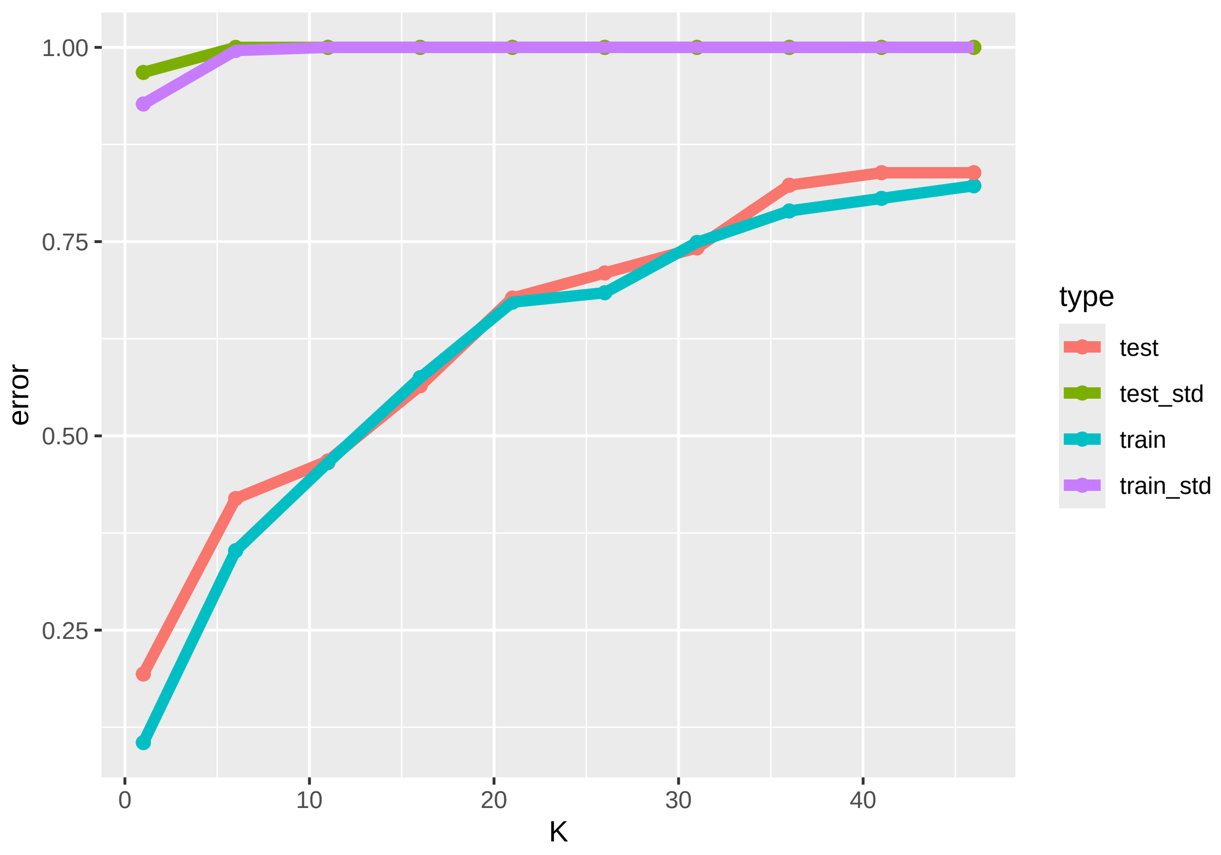
```
# Lasso Solution Path
fit_lasso <- glmnet(x_tr_std, y_tr)
plot_glmnet(fit_lasso)
```



```
# Lasso Solution Path
cv_fit_lasso <- cv.glmnet(x_tr, y_tr, alpha = 1, nfolds = 5)
l_tr_pred <- predict(cv_fit_lasso, newx = x_tr)
l_te_pred <- predict(cv_fit_lasso, newx = x_te)
l_tr_error <- mean((l_tr_pred - y_tr)^2)
l_te_error <- mean((l_te_pred - y_te)^2)
best_coefL <- as.vector(coef(cv_fit_lasso))
variables <- c("GENDER", "SMOKING", "YELLOW_FINGERS", "ANXIETY", "PEER_PRESSURE", "CHRONIC.DISEASE", "F
coef_names <- c("(Intercept)", variables)
lasso_df <- data.frame(coef_names, best_coefL, Method = "Lasso", Test_Error = l_te_error) %>% pivot_wid

rl_errors_df <- data.frame(
  Model = c("Ridge", "Lasso"),
  Testing_Error = c(r_te_error, l_te_error)
)
print(rl_errors_df)
```

```
##   Model Testing_Error
## 1 Ridge    0.09986412
## 2 Lasso    0.10405987
```

```r
#KNN
# Standardizing vairables for optimization
fit_std <- preProcess(train_data, method = "scale")
train_std <- predict(fit_std, newdata = train_data)
test_std <- predict(fit_std, newdata = test_data)

# Sequence of k values for KNN algorithm
k_seq <- seq(from = 1, to = 50, by = 5)
train_error_seq_std <- test_error_seq_std <- NULL

# Fitting KNN model on std predictor variables
for (k in k_seq) {
  fit_knn_std <- knnreg(LUNG_CANCER ~ YELLOW_FINGERS + ANXIETY + PEER_PRESSURE + CHRONIC.DISEASE + FATI

  pred_knn_train <- predict(fit_knn_std, newdata = train_std, type = "class")
  train_error_seq_std <- c(train_error_seq_std, mean(pred_knn_train != train_data$LUNG_CANCER))

  pred_knn_test <- predict(fit_knn_std, newdata = test_std, type = "class")
  test_error_seq_std <- c(test_error_seq_std, mean(pred_knn_test != test_data$LUNG_CANCER))
}

# Fitting KNN on unstd predictor variables
train_error_seq <- test_error_seq <- NULL

for (k in k_seq) {
  fit_knn <- knnreg(LUNG_CANCER ~ YELLOW_FINGERS + ANXIETY + PEER_PRESSURE + CHRONIC.DISEASE + FATIGUE

  pred_knn_train <- predict(fit_knn, newdata = train_data, type = "class")
  train_error_seq <- c(train_error_seq, mean(pred_knn_train != train_data$LUNG_CANCER))

  pred_knn_test <- predict(fit_knn, newdata = test_data, type = "class")
  test_error_seq <- c(test_error_seq, mean(pred_knn_test != test_data$LUNG_CANCER))
}

# Combining results
knn_std <- data.frame(K = k_seq, error = c(train_error_seq_std, test_error_seq_std),
                         type = rep(c("train_std", "test_std"), each = length(k_seq)))

knn_unstd <- data.frame(K = k_seq, error = c(train_error_seq, test_error_seq),
                     type = rep(c("train", "test"), each = length(k_seq)))

# Combine standardized and unstandardized results
knn_combined <- rbind(knn_std, knn_unstd)

ggplot(knn_combined, aes(x = K, y = error, color = type)) +
  geom_point(size = 2) +
  geom_line(size = 2)
```



- Based on errors, k = 7 will work best

```r
#Logistic Regression and KNN
set.seed(0)
# Fit Logistic Regression
fit_lr <- glm(LUNG_CANCER ~ YELLOW_FINGERS + ANXIETY + PEER_PRESSURE + CHRONIC.DISEASE + FATIGUE + ALLE
fit_lr_pred <- predict(fit_lr, type = "response")

# Calculate errors for Logistic Regression
lr_train_pred <- predict(fit_lr, newdata = train_data, type = "response")
lr_test_pred <- predict(fit_lr, newdata = test_data, type = "response")

train_pred_binary <- ifelse(lr_train_pred >= 0.5, 1, 0)
test_pred_binary <- ifelse(lr_test_pred >= 0.5, 1, 0)

lr_test_error <- mean(train_pred_binary != test_data$LUNG_CANCER)
lr_train_error <- mean(test_pred_binary != train_data$LUNG_CANCER)

# Fit KNN model
fit_knn <- knn3(LUNG_CANCER ~ YELLOW_FINGERS + ANXIETY + PEER_PRESSURE + CHRONIC.DISEASE + FATIGUE + AL

# Calculate errors for KNN
knn_train_pred <- predict(fit_knn, newdata = train_data)
knn_train_error <- mean(knn_train_pred != train_data$LUNG_CANCER)
knn_test_pred <- predict(fit_knn, newdata = test_data)
knn_test_error <- mean(knn_test_pred != test_data$LUNG_CANCER)

# Create dataframe for model errors
model_errors_df <- data.frame(
  Model = c("Logistic Regression", "KNN"),
  Training_Error = c(lr_train_error, knn_train_error),
  Testing_Error = c(lr_test_error, knn_test_error)
)

print(model_errors_df)
```

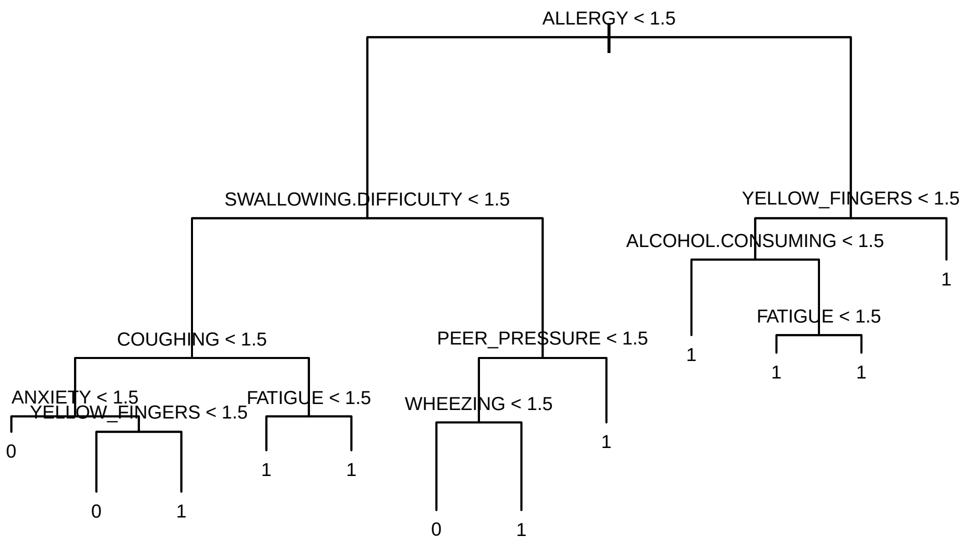
```
##                 Model Training_Error Testing_Error
## 1 Logistic Regression      0.2226721     0.2429150
## 2                 KNN      0.6923077     0.7177419
```

```r
#Logistic Regression and KNN
#Cross Validation
# Fit and LOOCV for logistic model
mod_formula_seq <- c("LUNG_CANCER ~ YELLOW_FINGERS + ANXIETY + PEER_PRESSURE + CHRONIC.DISEASE + FATIGU

loocv_lr <- sapply(mod_formula_seq, function(form){
  mod <- as.formula(form)
  fit <- glm(mod, data = train_data)
  cv.glm(train_data, fit)$delta[1]
})


# LOOCV for KNN Model with K=7
mod_formula_seq_knn <- "LUNG_CANCER ~ YELLOW_FINGERS + ANXIETY + PEER_PRESSURE + CHRONIC.DISEASE + FATIG
loocv_knn<-sapply(mod_formula_seq_knn, function(form){
  mod <- as.formula(form)
  mean(sapply(1:nrow(train_data), function(j){
    fit <- knn3(mod, data = train_data[-j, ],k=7)
    pred_lung <- predict(fit, newdata = train_data[j, ])
    (train_data$LUNG_CANCER[j] - pred_lung)^2
  }))
  }
  )

# Extract LOOCV errors
lr_loocv_error <- loocv_lr[[1]]
knn_loocv_error <- loocv_knn[[1]]

# Create dataframe for model errors
loocv_df <- data.frame(
  Model = c("Logistic Regression", "KNN"),
  LOOCV_Error = c(lr_loocv_error, knn_loocv_error)
)

print(loocv_df)
```

```
##                 Model LOOCV_Error
## 1 Logistic Regression  0.07174301
## 2                 KNN  0.44591164
```

- Compared to KNN, logistic has a relatively low misclassification rate.

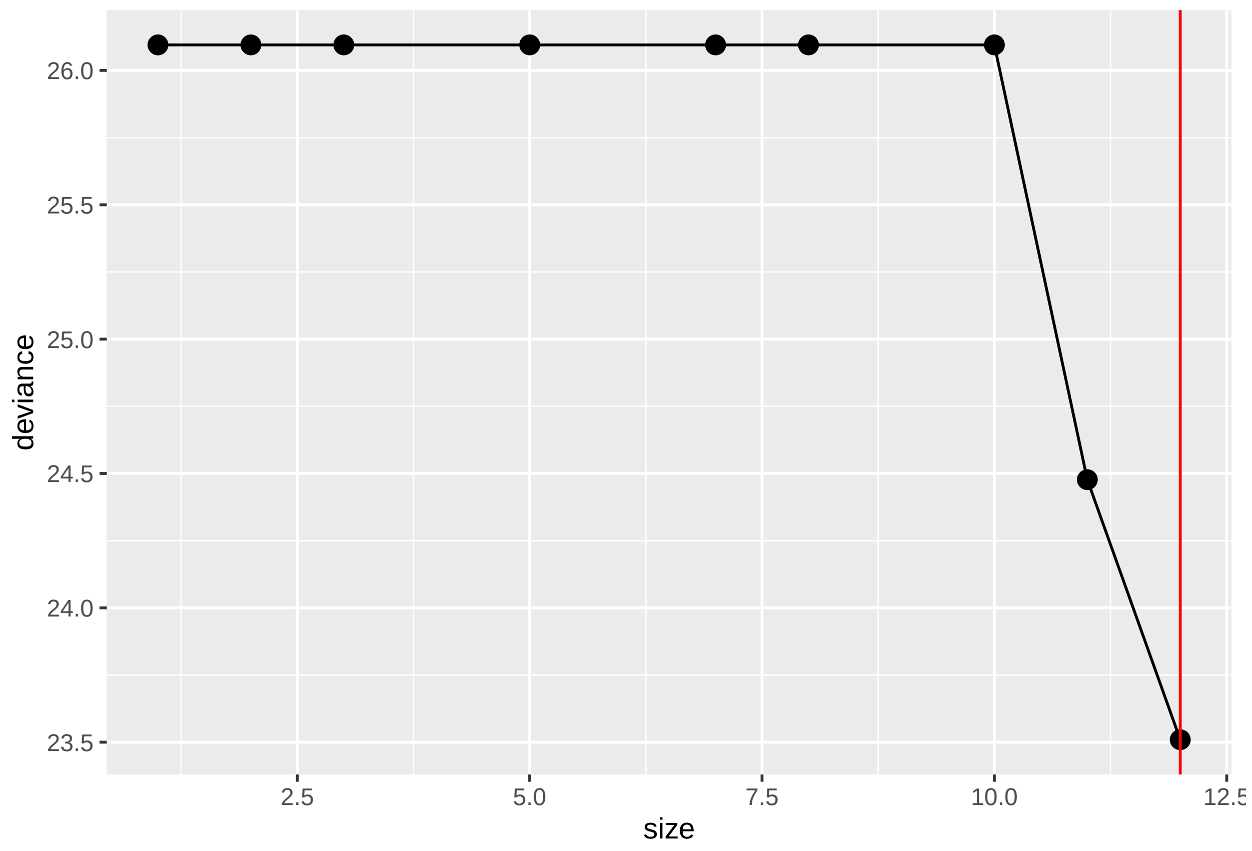
```
#Decision Tree
# Convert LUNG_CANCER to a factor variable
train_data$LUNG_CANCER <- as.factor(train_data$LUNG_CANCER)
lung_cancer_tree <- tree(LUNG_CANCER ~ YELLOW_FINGERS + ANXIETY + PEER_PRESSURE + CHRONIC.DISEASE + FATI
                         data = train_data)
plot(lung_cancer_tree)
text(lung_cancer_tree, cex= 0.6)
```



```
#Decision Tree
# Convert LUNG_CANCER to a numeric variable for error calculation
train_data$LUNG_CANCER <- as.numeric(as.character(train_data$LUNG_CANCER))
test_data$LUNG_CANCER <- as.numeric(as.character(test_data$LUNG_CANCER))
train_predictions <- predict(lung_cancer_tree, newdata = train_data)
test_predictions <- predict(lung_cancer_tree, newdata = test_data)
# Calculate errors
dt_train_error <- mean((train_predictions - train_data$LUNG_CANCER)^2)
dt_test_error <- mean((test_predictions - test_data$LUNG_CANCER)^2)
#Dataframe of errors
dt_errors_df <- data.frame(Model = c("Decision Tree"),
  Training_Error = c(dt_train_error), Testing_Error = c(dt_test_error)
)
print(dt_errors_df)
```

```
##           Model Training_Error Testing_Error
## 1 Decision Tree      0.4541269     0.4560601
```

```r
#Decision Tree
# Cross Validation
set.seed(0)
cv_lung_cancer <- cv.tree(lung_cancer_tree)
cv_lung_cancer_df <- data.frame(size = cv_lung_cancer$size, deviance = cv_lung_cancer$dev)
best_size <- cv_lung_cancer$size[which.min(cv_lung_cancer$dev)]

ggplot(cv_lung_cancer_df, aes(x = size, y = deviance)) +
  geom_point(size = 3) +
  geom_line() +
  geom_vline(xintercept = best_size, col = "red")
```



```r
cat('CV leads to the optimal tree size as ', best_size,'\n')
```

```
## CV leads to the optimal tree size as  12
```

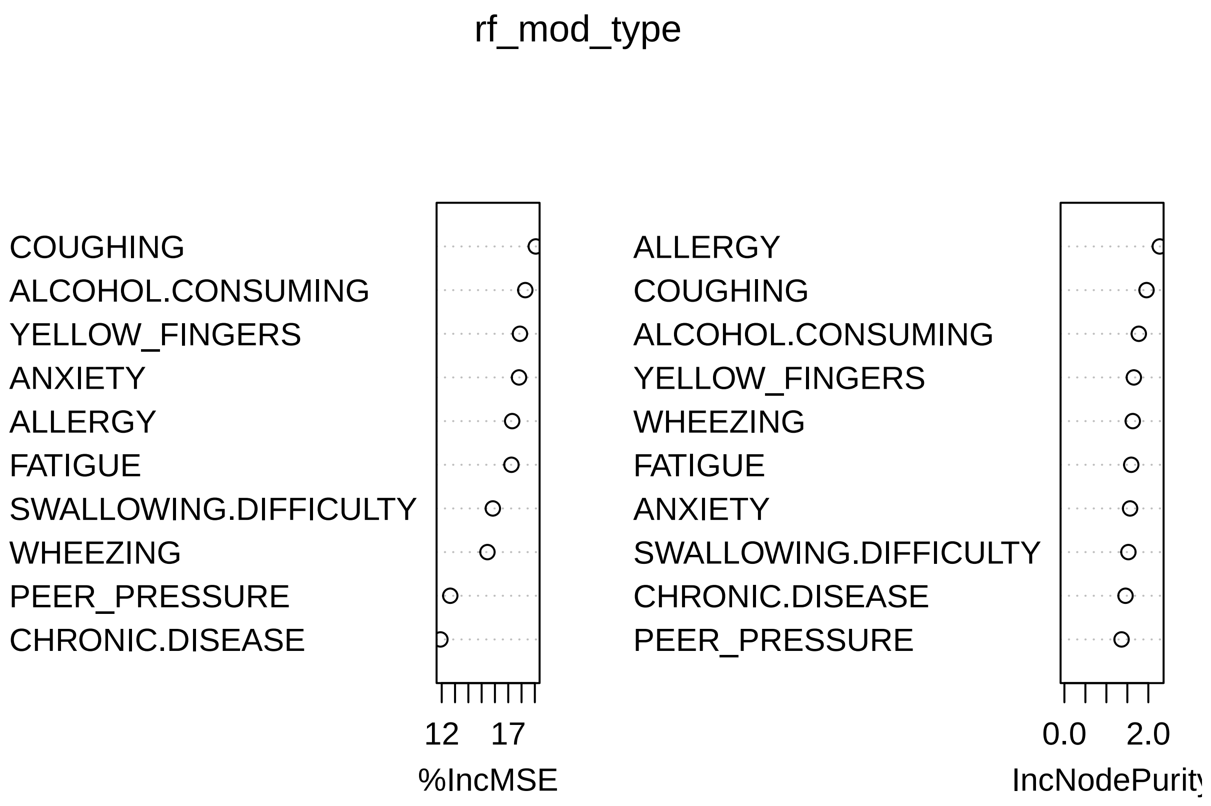
```r
# Random Forest
# Fit random forest model
set.seed(1)
rf_mod_type <- randomForest(LUNG_CANCER ~ YELLOW_FINGERS + ANXIETY + PEER_PRESSURE + CHRONIC.DISEASE + F
                            data = train_data, ntreeTry = 1000, importance = TRUE)

rf_train_pred_type <- predict(rf_mod_type, newdata = train_data, type = "response")
rf_test_pred_type <- predict(rf_mod_type, newdata = test_data, type = "response")

# Convert predicted values to factor (binary classification)
rf_train_pred_class <- as.factor(ifelse(rf_train_pred_type > 0.5, "1", "0"))
rf_test_pred_class <- as.factor(ifelse(rf_test_pred_type > 0.5, "1", "0"))

# Calculate errors
rf_train_error <- mean(rf_train_pred_class != train_data$LUNG_CANCER)
rf_test_error <- mean(rf_test_pred_class != test_data$LUNG_CANCER)

# Dataframe for errors
rf_errors_df <- data.frame(
  Model = c("Random Forrest"),
  Training_Error = c(rf_train_error),
  Testing_Error = c(rf_test_error)
)

rf_errors_df
```

```
##            Model Training_Error Testing_Error
## 1 Random Forrest     0.04048583     0.1129032
```

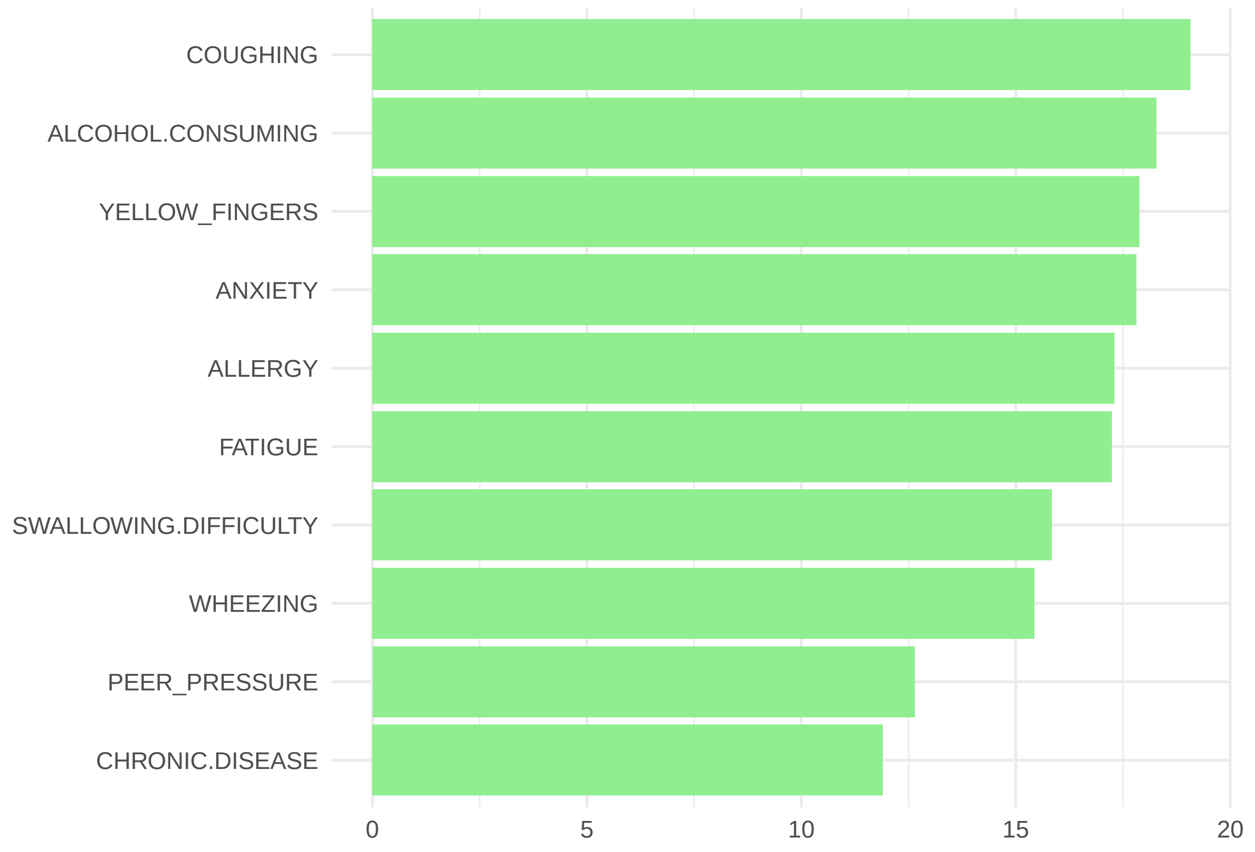
```r
# Random Forest
# Feature importance for random forest model
rf_importance <- importance(rf_mod_type)
rf_fn <- rownames(rf_importance)
rf_data <- data.frame(Feature = rf_fn, Importance = rf_importance[, 1])
rf_plot <- ggplot(rf_data, aes(x = Importance, y = reorder(Feature, Importance))) +
  geom_bar(stat = "identity", fill = "lightgreen") +
  labs(title = NULL, x = NULL, y = NULL) +
  theme_minimal()

importance(rf_mod_type)
```

```
##                        %IncMSE IncNodePurity
## YELLOW_FINGERS        17.87960      1.654784
## ANXIETY               17.80993      1.565168
## PEER_PRESSURE         12.64067      1.362142
## CHRONIC.DISEASE       11.89969      1.456343
## FATIGUE               17.24007      1.593161
## ALLERGY               17.29679      2.273151
## WHEEZING              15.43582      1.629250
## ALCOHOL.CONSUMING     18.28042      1.772893
## COUGHING              19.07052      1.957141
## SWALLOWING.DIFFICULTY 15.83987      1.525005
```

```
varImpPlot(rf_mod_type)
```

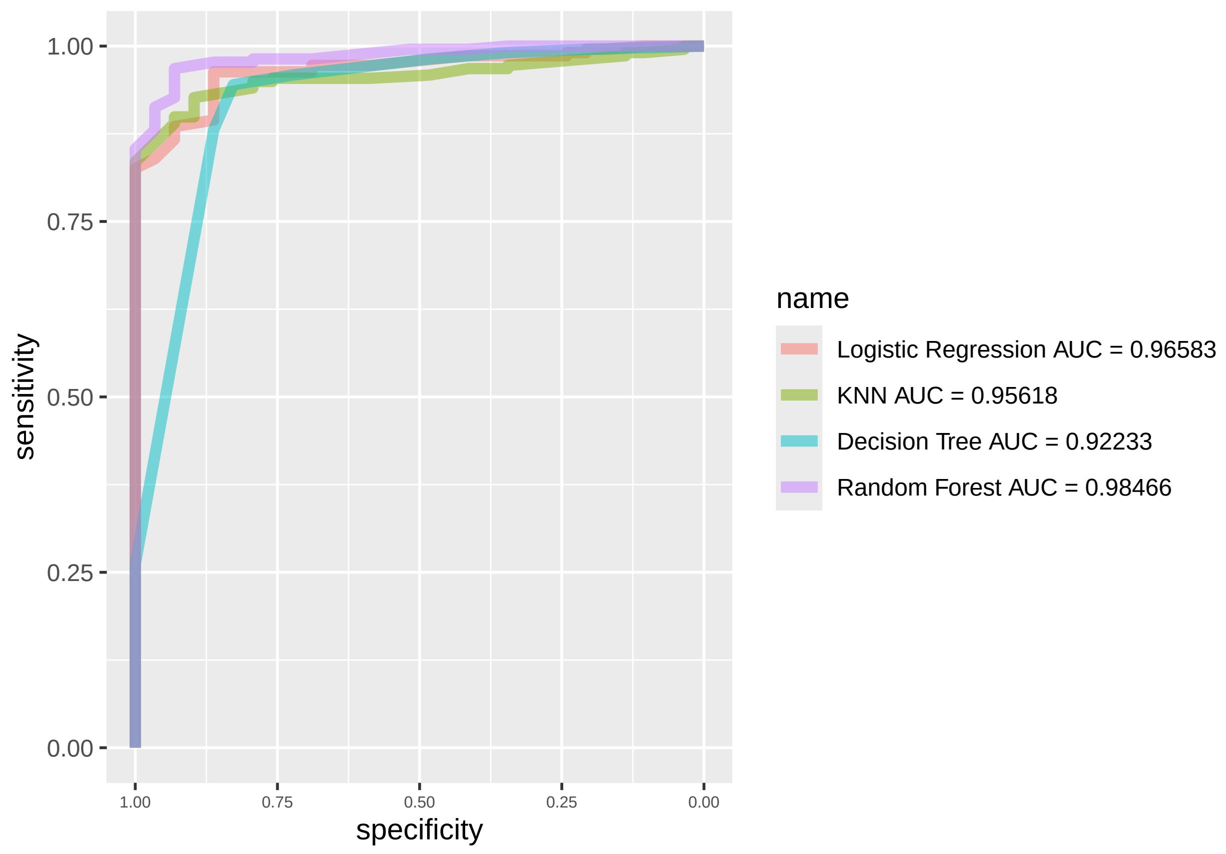


```
print(rf_plot)
```



- The most important features across random forest were COUGHING, ALCOHOL CONSUMING, and YELLOW FINGERS.

```r
#ROC/AUC PLOT
# Fit Logistic Regression
fit_lr <- glm(LUNG_CANCER ~ YELLOW_FINGERS + ANXIETY + PEER_PRESSURE + CHRONIC.DISEASE + FATIGUE + ALLE
fit_lr_pred <- predict(fit_lr, type = "response")
roc_lr <- roc(train_data$LUNG_CANCER, fit_lr_pred)
# Fit KNN model
fit_knn <- knn3(LUNG_CANCER ~ YELLOW_FINGERS + ANXIETY + PEER_PRESSURE + CHRONIC.DISEASE + FATIGUE + AL
fit_knn_pred <- predict(fit_knn, newdata = train_data, type = "prob")
roc_knn <- roc(train_data$LUNG_CANCER, fit_knn_pred[, 2])
# Fit Decision Tree model
fit_tree <- tree(LUNG_CANCER ~ YELLOW_FINGERS + ANXIETY + PEER_PRESSURE + CHRONIC.DISEASE + FATIGUE + A
fit_tree_pred <- predict(fit_tree, newdata = train_data, type = "vector")
roc_tree <- roc(train_data$LUNG_CANCER, fit_tree_pred)
# Fit Random Forest model
fit_rf <- randomForest(LUNG_CANCER ~ YELLOW_FINGERS + ANXIETY + PEER_PRESSURE + CHRONIC.DISEASE + FATIG
fit_rf_pred <- predict(fit_rf, newdata = train_data, type = "response")
roc_rf <- roc(train_data$LUNG_CANCER, fit_rf_pred)
# Plot ROC curves
rocobjs <- list("Logistic Regression" = roc_lr, "KNN" = roc_knn, "Decision Tree" = roc_tree, "Random Fo
methods_auc <- paste(c("Logistic Regression", "KNN", "Decision Tree", "Random Forest"), "AUC =", round(
ggroc(rocobjs, size = 2, alpha = 0.5) +
  scale_color_discrete(labels = methods_auc) +
  theme(axis.text.x = element_text(size = 6))
```



- Random forest had the highest AUC value at 0.984 followed by logistic regression at 0.97.

```r
#Confusion matrix for logistic regression and random forrest
#Logistic Regression
predicted_classes <- ifelse(lr_test_pred >= 0.5, "1", "0")
conf_matrix_lr <- confusionMatrix(as.factor(predicted_classes), as.factor(test_data$LUNG_CANCER), posit
print("Confusion Matrix for Logistic Regression:")
```

```
## [1] "Confusion Matrix for Logistic Regression:"
```

```r
print(conf_matrix_lr)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  6  1
##          1  4 51
##
##                Accuracy : 0.9194
##                  95% CI : (0.8217, 0.9733)
##     No Information Rate : 0.8387
##     P-Value [Acc > NIR] : 0.05172
##
##                   Kappa : 0.6608
##
##  Mcnemar's Test P-Value : 0.37109
##
##             Sensitivity : 0.9808
##             Specificity : 0.6000
##          Pos Pred Value : 0.9273
##          Neg Pred Value : 0.8571
##              Prevalence : 0.8387
##          Detection Rate : 0.8226
##    Detection Prevalence : 0.8871
##       Balanced Accuracy : 0.7904
##
##        'Positive' Class : 1
##
```

```r
#Random Forrest
predicted_classes2 <- ifelse(rf_test_pred_type >= 0.5, "1", "0")
conf_matrix_rf <- confusionMatrix(as.factor(predicted_classes2), as.factor(test_data$LUNG_CANCER), posi
print("Confusion Matrix for Random Forrest:")
```

```
## [1] "Confusion Matrix for Random Forrest:"
```

```r
print(conf_matrix_rf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  4  1
##          1  6 51
##
##                Accuracy : 0.8871
##                  95% CI : (0.7811, 0.9534)
##     No Information Rate : 0.8387
##     P-Value [Acc > NIR] : 0.1967
##
##                   Kappa : 0.4771
##
##  Mcnemar's Test P-Value : 0.1306
##
##             Sensitivity : 0.9808
##             Specificity : 0.4000
##          Pos Pred Value : 0.8947
##          Neg Pred Value : 0.8000
##              Prevalence : 0.8387
##          Detection Rate : 0.8226
##    Detection Prevalence : 0.9194
##       Balanced Accuracy : 0.6904
##
##        'Positive' Class : 1
##
```

- Both logistic regression and random forest models had high accuracy rates at 92% and 89%.