# Predicting Instances of Coronary Disease Using Various Machine Learning Techniques

Justin Carter

Department of Statistics, University of Minnesota

STAT 4052, Statistical Machine Learning II

Dr. Algeri

May 3, 2024

## Introduction

The goal of this analysis is to predict whether or not a person has a coronary disease or myocardial infarction using features that include information about their current health. The features used in this analysis are whether a patient has high cholesterol or blood pressure, if they have had their cholesterol checked in the last five years, if they have had a stroke, if they have diabetes, if they have smoked more than 100 cigarettes in their life, their age, sex, level of education and BMI (body mass index). Another goal of this analysis is to predict which of the features have the most impact on coronary disease and myocardial infarction. According to the analysis we conducted, the most important variables in predicting coronary disease are whether a patient has had a stroke, if they have high blood pressure, BMI, and their age. In this analysis, we will use three different machine learning methods and two ways of dealing with missing data and report and discuss the results.

## Methods

The methods used to predict coronary disease and myocardial infarction include Logistic Regression, K-Nearest Neighbors (KNN) and Random Forest (RF). To evaluate accuracy of the models used, we employ k-folds cross-validation to estimate error rate, sensitivity, and specificity. Because this is an imbalance class problem (many more instances of absent coronary disease than present), we pay special attention to sensitivity as this metric gives us a good understanding of the accuracy in predicting the positive class (presence of coronary disease). It is important that we generate a model that can accurately predict both classes, as overall error rate will be strongly influenced by the accuracy of our negative response class (absence of coronary disease). Additionally, we evaluate ROC curves for each of our models to see the relationship of specificity and sensitivity at different thresholds.

In the original dataset, we have missing values for our coronary disease, stroke, and diabetes. Initially, we remove the observations containing any missing values. We then create our Logistic Regression, KNN, and RF models and evaluate their accuracy. After this analysis, we return to the original dataset with missing values, and impute them using iterative regression. We use a multinomial model to predict the missing values according to the other features, and update them in each iteration. We choose to run 20 iterations of this method, as through some exploratory analysis, the missing values have converged at this number of iterations. More iterations require greater runtime as well, so we choose a lower number while ensuring convergence. Using the new dataset, we perform the same analysis using the same methods described before, and compare the differences in models.

*Logistic regression models:* Logistic regression will perform well when there is linearity between the predictor and response variables. To estimate the accuracy of a logistic model, we split the data into

10 folds, and use stepwise selection to choose the best model for each training set. To estimate the accuracy, we average the error rate, specificity, and sensitivity for each of our folds. To generate a ROC curve, we train the model selected by stepwise selection on the entire dataset on a randomly selected set of 80% of our observations, and use the remaining 20% as a validation set.

*KNN models:* KNN offers a flexible, non-linear, non-parametric model that performs well when the assumption of linearity is not upheld. To estimate accuracy of a KNN model, we choose 6 values of k (3, 5, 10, 15, 20, 25), and use 10 folds to assess the error rate, specificity, and sensitivity for each value of k. We again use the average over all 10 folds for each value of k. We select a value of k based on our results, and use this to create a ROC curve by training a model on the previously split data (80% training, 20% validation).

*RF models:* RF provides us with another non-linear, non-parametric method that will decrease variance when compared to other tree based methods. We tune our mtry parameter (number of variables considered at each split), and train a RF model using this parameter. We use Out-of-Bag (OOB) estimates of error rate, sensitivity, and specificity. We train another model with the same parameters as our chosen model using our same testing and validation sets described before to create a ROC curve.

## Results

*Logistic Regression, no data imputation:* We use backwards stepwise selection to obtain an optimal model according to AIC (Akaike Information Criterion). The model selected, trained on the entire non-imputed dataset is:

$$\textit{Coronary Disease} \sim \textit{High Blood Pressure} + \textit{High Cholesterol} + \textit{BMI} + \textit{Smoker} + \textit{Stroke} + \textit{Diabetes} + \textit{Sex} + \textit{Age}$$

The elimination of two variables, cholesterol check (<5 years), and education were omitted from the model as their elimination led to a reduction in AIC. The model summary is shown below in table 1:

| | Estimate | Std. Error | z value | Pr(>\|z\|) | |
|---|---|---|---|---|---|
| (Intercept) | -6.457578 | 0.366739 | -17.608 | < 2e-16 | *** |
| HighBP1 | 0.879710 | 0.123238 | 7.138 | 9.45e-13 | *** |
| HighChol1 | 0.518220 | 0.111458 | 4.649 | 3.33e-06 | *** |
| BMI | 0.023213 | 0.008249 | 2.814 | 0.004892 | ** |
| Smoker1 | 0.447431 | 0.103348 | 4.329 | 1.50e-05 | *** |
| Stroke1 | 1.584045 | 0.152023 | 10.420 | < 2e-16 | *** |
| Diabetes1 | 0.509098 | 0.275176 | 1.850 | 0.064302 | . |
| Diabetes2 | 0.412492 | 0.120268 | 3.430 | 0.000604 | *** |
| Sex1 | 0.322366 | 0.103394 | 3.118 | 0.001822 | ** |
| Age | 0.223605 | 0.022729 | 9.838 | < 2e-16 | *** |

***Table 1***

Lower p-values for a coefficient indicate stronger probability that the coefficient does not equal 0 (no effect on the response variable). Therefore, using this knowledge, we conclude that having a stroke, age, and high blood pressure have a strong impact on whether a person has coronary disease according to our logistic regression model.

Using 10-fold cross-validation, we estimate the error rate = 0.08783406, sensitivity = 0.07504374, and specificity = 0.9915444. These values indicate that although overall error is relatively low, the error for the positive class (presence of coronary disease) is very high, while the error for the negative class (absence of coronary disease) is low.

*KNN, no data imputation:* We use 10-fold cross validation to select the optimal parameter k (number of nearest neighbors with which we classify new observations) and obtain an estimate for error rate, sensitivity and specificity. Error rate, sensitivity, and specificity estimates for each value of k are provided in the tables below.

| | Error Rate | Sensitivity | Specificity |
|---|---|---|---|
| k=3 | 0.09643211 | 0.05290610 | 0.9841115 |
| k=5 | 0.09029698 | 0.02277278 | 0.9937054 |
| k=10 | 0.08713723 | 0.00365570 | 0.9988553 |
| k=15 | 0.08660999 | 0.00000000 | 0.9998081 |
| k=20 | 0.08643455 | 0.00000000 | 1.0000000 |
| k=25 | 0.08643455 | 0.00000000 | 1.0000000 |

***Table 2***

Typically, we will choose the optimal value of k by selecting the model that has the lowest error rate estimate. However, in the case of our data, the imbalance in classes causes sensitivity to be very low. We want to maximize sensitivity as this indicates our models ability to correctly predict observations of the positive class. As we see in the table, sensitivity is reduced greatly from k=3 to k=5, and reaches 0 as we increase k. Though the models with sensitivity = 0 have the lowest error rates, they are useless models as they predict every observation as the negative class. Therefore, we select the model with k=3, even though it has the highest error rate of the tested k parameters, because it maximizes sensitivity which is the important class in our predictive model.

The model selected has a higher error rate due to high flexibility. However, due to the influence of many negative class observations, when we increase k, our model tends to predict negative class outcomes for new observations. Therefore, KNN is affected by class imbalance and may not be an optimal solution for our predictive model.

*RF, no data imputation:* We begin by selecting an optimal value for mtry by testing different values and selecting the one that minimizes OOB error rate. The results are shown in the figure below:

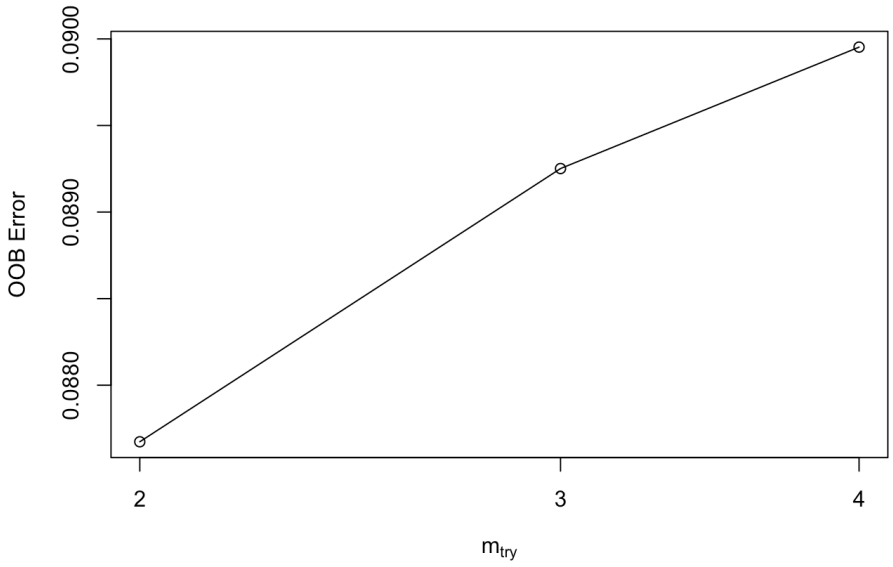



***Figure 1***

We select mtry = 2 for our model, according to the figure. We then train the model on the full non-imputed dataset and obtain the following plot of variable importance shown in the figure below:

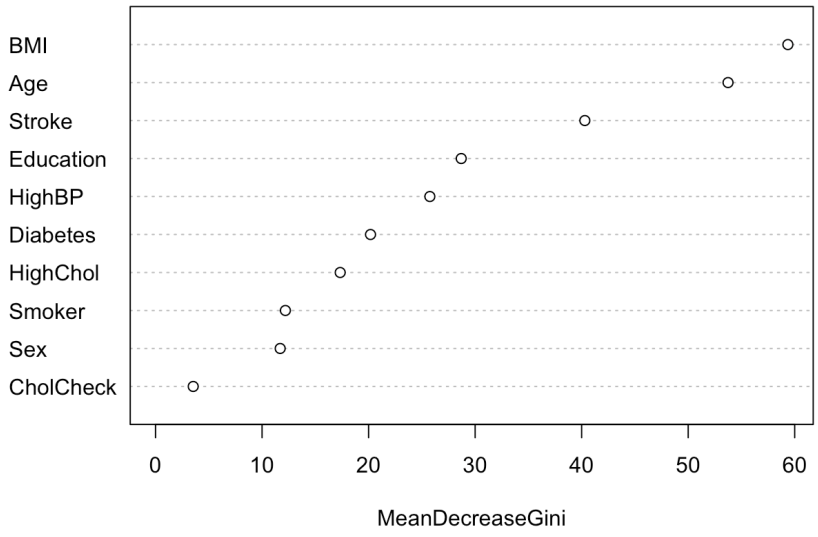



***Figure 2***

This figure shows the average decrease in the gini index (measure of node purity, lower values are more pure). The variables, BMI, Age, and Stroke decrease the gini index the most, indicating that these variables are the most important in our predictive model. Stroke and age appear again as important variables similar to the logistic model described earlier.

We obtain estimated error rate = 0.087 sensitivity = 0.0061 and specificity = 0.9988. As we can see from these metrics, predicting the positive class remains difficult. Sensitivity is very low in the RF model, while overall error rate and specificity are high. Again, this is likely due to the imbalance of classes. At each split for each tree, there are many more instances of the negative class, influencing the prediction of individual trees and eventually the ensemble classifier as a whole.

*Model comparisons:* Error rates for the models are similar between RF and Logistic Regression. The KNN model chosen has a greater error rate than both of these models. Based solely on error rate, we would choose the RF model because it has the lowest overall error rate. However, we also want to examine the sensitivity of the models as our important class includes observations that have coronary disease. The lowest sensitivity is our RF model, and it only has three true positive predictions. Therefore, though this model shows the lowest error rate, we do not want to use it for future predictions, as it is not effective for predicting observations with coronary disease. The next best model in terms of sensitivity is the KNN model, reaching around 5%. This model offers greater predictive accuracy for the positive class, though it has a greater error rate overall and is less effective for predicting the negative class. Our logistic regression model offers the highest sensitivity at around 7.5%. It also has the second best error rate, indicating high performance at predicting the negative class. For these reasons of balancing maximal sensitivity and minimizing error rate, we choose our logistic model to predict future observations. We

generate ROC curves for each model below to observe the relationship between sensitivity and specificity for the various classification thresholds.

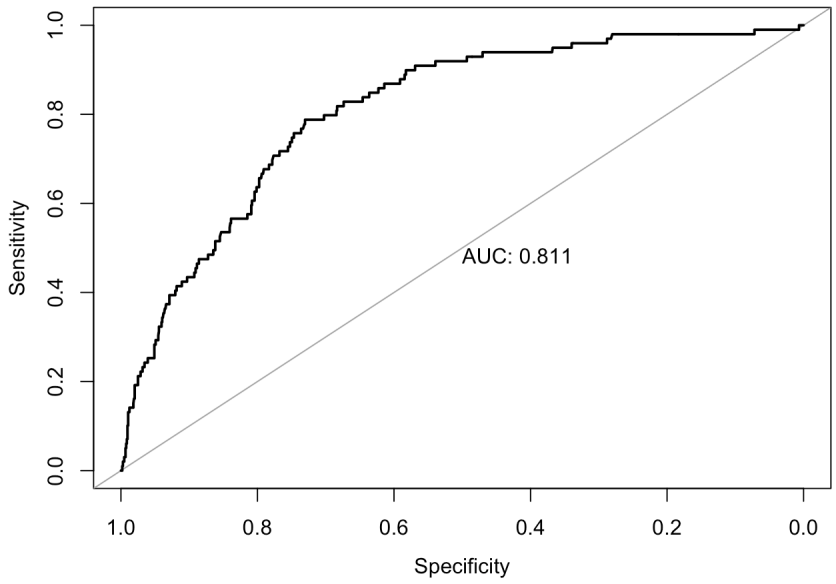



***Figure 3, Logistic Regression ROC curve***

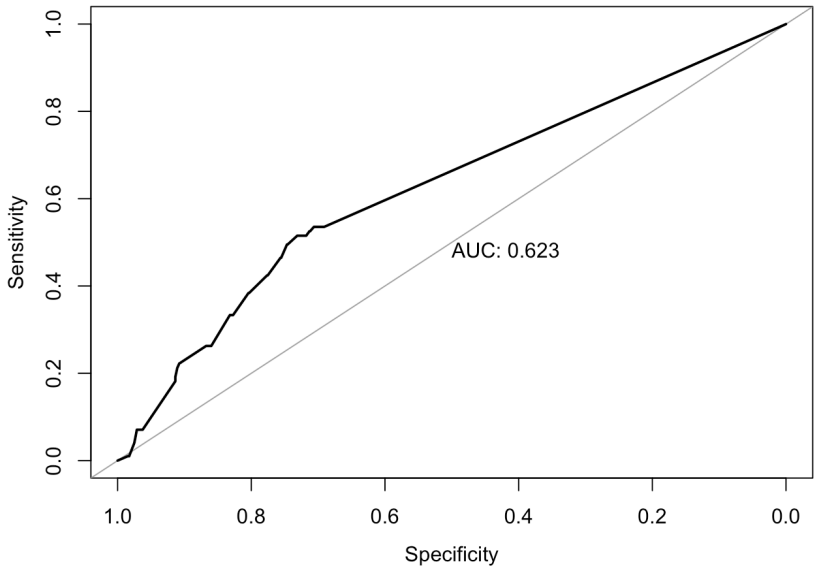



***Figure 4, KNN ROC curve***

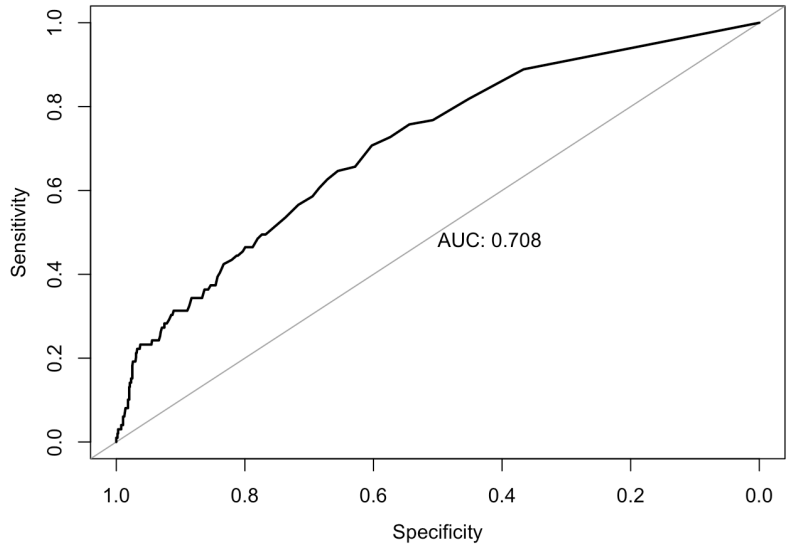



***Figure 5, RF ROC Curve***

Note that plotting sensitivity against specificity from 1 to 0 is equivalent to plotting true positive rate against false positive rate from 0 to 1. A perfect model (0% error rate) will have an AUC (area under the curve) of 1, and a ROC plot that has a point at (1,1). Therefore, we choose the model with the highest AUC, and look for the plot that has a point closest to (1,1) (0% error rate, sensitivity = 1, specificity = 1). As we can see from the ROC curves, logistic regression offers the greatest area under the curve (AUC) indicating greater maximization of sensitivity and specificity for various classification thresholds. Additionally, it has the point that is closest to (1,1), indicating that this model has a classification threshold that maximizes both sensitivity and specificity greater than the other models. The next strongest model is our RF model, and last is the KNN model. From this comparison of AUC and ROC plots, we again strengthen the case for choosing logistic regression to predict future observations. From our analysis, it seems as though there is a linear relationship between predictor and response variables that is not modeled by the non-linear methods of KNN and RF. Additionally, KNN and RF are affected by the class imbalance issue more greatly than our logistic regression model.

*Error rates, sensitivity, and specificity after iterative regression imputation:* After the iterative regression imputation, we perform the same analysis performed above for each method, and report the important metrics. For logistic regression, we have error rate = 0.08957143 sensitivity = 0.0753866 and specificity = 0.9929196. This indicates that we have a slightly higher error rate, but higher sensitivity and specificity indicating a higher proportion of the positive class is present in the data we have imputed. The AUC = 0.805 which is similar to the non-imputed data value. Because we see an increase in sensitivity and specificity and the error rate is similar, we conclude that the model has improved. For KNN, we again choose k=3 to maximize sensitivity. With this model, we have error rate = 0.10028571 sensitivity =

0.051396373 and specificity = 0.9828968. We see an increase in error and a decrease in specificity and sensitivity compared to our non-imputed model. However, the AUC = 0.676 indicates a stronger balance between sensitivity and specificity overall across different classification thresholds. Still, we prefer the KNN model created with the non-imputed data for this case. For RF, we again try various values of mtry and choose 2 as our optimal value. We have error rate = 0.0899 sensitivity = 0.0063 and specificity = 0.999. Similar to logistic regression, we see an improvement in sensitivity and specificity, and an increase in error rate. The AUC = 0.774 also improves, indicating greater performance across various classification thresholds. Again, we prefer the data imputed model because an increase in sensitivity and specificity suggests greater predictive ability despite the slight increase in error rate.

Overall, while the differences can be viewed as minimal between the two methods for dealing with missing values in the data, we prefer the iterative regression imputation because it increases sensitivity and specificity for our RF and logistic regression models. The increase in error rate is likely due to the greater proportion of positive cases in the imputed data. As we know, the positive cases have lower predictive performance, driving lower error rate despite increases in our other metrics.

## Discussion

Based on the results of our analysis, we choose the logistic regression model for predicting future outcomes for patients. This model was the most easily interpreted, and had the highest sensitivity and specificity between models. Though the RF model had slightly lower error rate, it had significantly lower sensitivity and struggled to generate accurate predictions for the positive class. The KNN model was the weakest, demonstrating the worst error rate and AUC for the ROC curve. The logistic regression and RF models helped us understand the most important variables through coefficient p-values and variable importance respectively. We identify age and presence of a previous stroke as the strongest indicators of coronary heart disease, as these both had a high mean decrease in the gini index in our RF model and a very low p-value in our logistic regression model.

Predicting the presence of coronary disease remains a difficult problem for a number of reasons, we will highlight a few. One reason it is difficult to predict is that there are many more observations of patients without coronary disease, making the data for coronary disease more limited than for individuals without coronary disease. Additionally, the data only describes people that currently have coronary disease. Across all of the features, we may have two patients, one with and one without coronary disease with very similar metrics. However, just because one does not currently have coronary disease does not mean they will not develop it in the future. This makes it hard to distinguish between classes, because even if someone possesses concerning health metrics, they may not be indicated as someone having

coronary disease, even if they will develop it in the near future. In other words, there is no clear class distinction between the variables, which makes predicting the positive class much more difficult.

One suggestion for future research is containing a larger database as more observations generally increase model predictive accuracy overall. Additionally, data regarding whether someone will develop coronary disease in the future may be helpful to create a more accurate predictive model; this could be done by using health records of people in the past rather than more current records. More machine learning methods could also be tested to determine which would be most useful in this case of predictive modeling.

## Appendix: R-codes:

```r
library(caTools)

heart_disease_full <- read.table("/Users/justincarter/Downloads/heart_disease6.txt")

heart_disease_full$HeartDiseaseorAttack <- as.factor(heart_disease_full$HeartDiseaseorAttack)
heart_disease_full$HighBP <- as.factor(heart_disease_full$HighBP)
heart_disease_full$HighChol <- as.factor(heart_disease_full$HighChol)
heart_disease_full$CholCheck <- as.factor(heart_disease_full$CholCheck)
heart_disease_full$Smoker <- as.factor(heart_disease_full$Smoker)
heart_disease_full$Stroke<- as.factor(heart_disease_full$Stroke)
heart_disease_full$Diabetes <- as.factor(heart_disease_full$Diabetes)
heart_disease_full$Sex<- as.factor(heart_disease_full$Sex)
heart_disease_full$Education <- as.factor(heart_disease_full$Education)

heart_disease <- heart_disease_full %>% drop_na(HeartDiseaseorAttack, Stroke, Diabetes)
```

```r
set.seed(2024)

kfolds_logistic <- function(data, nfolds){
  fold <- createFolds(1:nrow(data),k=nfolds,list=FALSE)
  er_estimates <- rep(0, nfolds)
  sens_estimates <- rep(0, nfolds)
  spec_estimates <- rep(0, nfolds)
  for(i in 1:nfolds){
    train <- data[fold != i,]
    test <-  data[fold == i,]
    logistic_full_mod <- glm(HeartDiseaseorAttack ~., data = train, family = "binomial")
    log_mod <- step(logistic_full_mod,scope=list(lower=~1,upper=~.), trace = FALSE)
    predictions <- rep(0, length(test[,1]))
    probs <- predict(log_mod, newdata = test[,2:11], type = "response")
    predictions[probs > 0.5] <- 1
    predictions <- as.factor(predictions)
    er_estimates[i] <- sum(predictions != test[,1]) / length(test[,1])
    sens_estimates[i] <- sum(predictions == 1 & test[,1] == 1) /
                         (sum(predictions == 1 & test[,1] == 1) +
                         sum(predictions == 0 & test[,1] == 1))
    spec_estimates[i] <- sum(predictions == 0 & test[,1] == 0) /
                         (sum(predictions == 0 & test[,1] == 0) +
                         sum(predictions == 1 & test[,1] == 0))

  }
  print(mean(er_estimates))
  print(mean(sens_estimates))
  print(mean(spec_estimates))
}

kfolds_logistic(heart_disease, 10)
```

```
## [1] 0.08783406
## [1] 0.07504374
## [1] 0.9915444
```

```r
logistic_full_mod <- glm(HeartDiseaseorAttack ~., data = heart_disease, family = "binomial")
log_mod1 <- step(logistic_full_mod,scope=list(lower=~1,upper=~.), trace = FALSE)

summary(log_mod1)
```

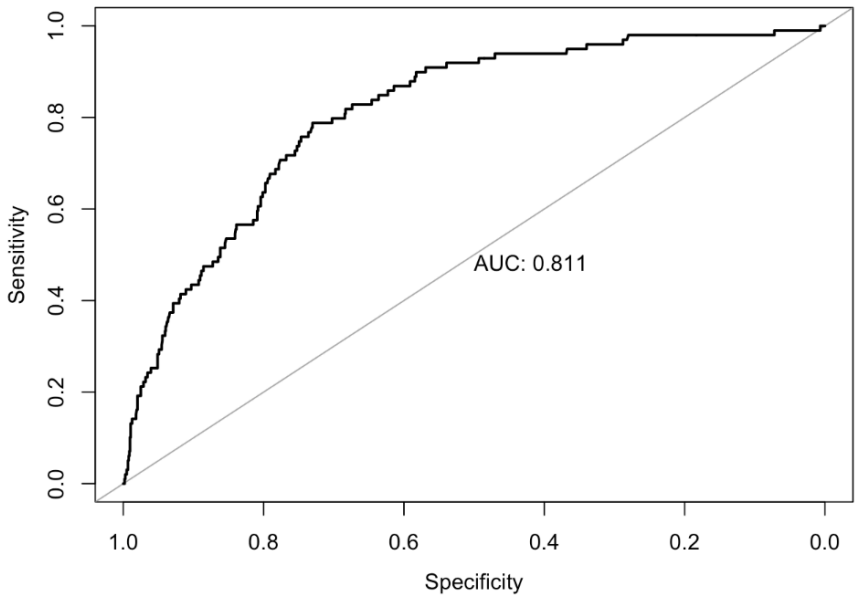
```
##
## Call:
## glm(formula = HeartDiseaseorAttack ~ HighBP + HighChol + BMI +
##     Smoker + Stroke + Diabetes + Sex + Age, family = "binomial",
##     data = heart_disease)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.457578   0.366739 -17.608  < 2e-16 ***
## HighBP1      0.879710   0.123238   7.138 9.45e-13 ***
## HighChol1    0.518220   0.111458   4.649 3.33e-06 ***
## BMI          0.023213   0.008249   2.814 0.004892 **
## Smoker1      0.447431   0.103348   4.329 1.50e-05 ***
## Stroke1      1.584045   0.152023  10.420  < 2e-16 ***
## Diabetes1    0.509098   0.275176   1.850 0.064302 .
## Diabetes2    0.412492   0.120268   3.430 0.000604 ***
## Sex1         0.322366   0.103394   3.118 0.001822 **
## Age          0.223605   0.022729   9.838  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3356.1  on 5702  degrees of freedom
## Residual deviance: 2731.6  on 5693  degrees of freedom
## AIC: 2751.6
##
## Number of Fisher Scoring iterations: 6
```

```
data_split_1 <- sample.split(heart_disease$HeartDiseaseorAttack, SplitRatio = 0.8 )

train_1 <- filter(heart_disease, data_split_1 == TRUE)
test_1 <- filter(heart_disease, data_split_1 == FALSE)
train_1 <-  as.data.frame(train_1)
test_1 <- as.data.frame(test_1)

log_ROC_mod1 <- glm(HeartDiseaseorAttack ~ HighBP + HighChol + Smoker + Diabetes +
                      Stroke + Sex + BMI + Age, data = train_1, family = "binomial")
log_ROC <- roc(test_1$HeartDiseaseorAttack, predict(log_ROC_mod1, newdata = test_1[,2:11],
                                                    type = "response"))
```

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

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

```
plot(log_ROC, asp = 0, print.auc = TRUE)
```




```
set.seed(2024)
library(class)

kfolds_knn <- function(data, nfolds){
  er_estimates <- data.frame(matrix(ncol = 7, nrow = nfolds))
  sens_estimates <- data.frame(matrix(ncol = 7, nrow = nfolds))
  spec_estimates <- data.frame(matrix(ncol = 7, nrow = nfolds))
  fold <- createFolds(1:nrow(data),k=nfolds,list=FALSE)
  for(i in 1:nfolds){
    pred_3 <- knn(as.matrix(data[fold != i, 2:11]), as.matrix(data[fold == i, 2:11]),
                  cl = data$HeartDiseaseorAttack[fold !=i], k = 3)
    pred_5 <- knn(as.matrix(data[fold != i, 2:11]), as.matrix(data[fold == i, 2:11]),
                  cl = data$HeartDiseaseorAttack[fold !=i], k = 5)
    pred_10 <- knn(as.matrix(data[fold != i, 2:11]), as.matrix(data[fold == i, 2:11]),
                  cl = data$HeartDiseaseorAttack[fold !=i], k = 10)
    pred_15 <- knn(as.matrix(data[fold != i, 2:11]), as.matrix(data[fold == i, 2:11]),
                  cl = data$HeartDiseaseorAttack[fold !=i], k = 15)
    pred_20 <- knn(as.matrix(data[fold != i, 2:11]), as.matrix(data[fold == i, 2:11]),
                  cl = data$HeartDiseaseorAttack[fold !=i], k = 20)
    pred_25 <- knn(as.matrix(data[fold != i, 2:11]), as.matrix(data[fold == i, 2:11]),
                  cl = data$HeartDiseaseorAttack[fold !=i], k = 25)

    er_estimates[i,1] <- sum(pred_3 != data$HeartDiseaseorAttack[fold == i]) /
                  length(data$HeartDiseaseorAttack[fold == i])
    er_estimates[i, 2] <- sum(pred_5 != data$HeartDiseaseorAttack[fold == i]) /
                  length(data$HeartDiseaseorAttack[fold == i])
    er_estimates[i, 3] <- sum(pred_10 != data$HeartDiseaseorAttack[fold == i]) /
                  length(data$HeartDiseaseorAttack[fold == i])
    er_estimates[i, 4] <- sum(pred_15 != data$HeartDiseaseorAttack[fold == i]) /
                  length(data$HeartDiseaseorAttack[fold == i])
    er_estimates[i, 5] <- sum(pred_20 != data$HeartDiseaseorAttack[fold == i]) /
                  length(data$HeartDiseaseorAttack[fold == i])
    er_estimates[i, 6] <- sum(pred_25 != data$HeartDiseaseorAttack[fold == i]) /
                  length(data$HeartDiseaseorAttack[fold == i])
```

```
    sens_estimates[i,1] <- sum(pred_3 == 1 & data$HeartDiseaseorAttack[fold == i] == 1) /
                (sum(pred_3 == 1 & data$HeartDiseaseorAttack[fold == i] == 1) +
                sum(pred_3 == 0 & data$HeartDiseaseorAttack[fold == i] == 1))
    sens_estimates[i,2] <- sum(pred_5 == 1 & data$HeartDiseaseorAttack[fold == i] == 1) /
                (sum(pred_5 == 1 & data$HeartDiseaseorAttack[fold == i] == 1) +
                sum(pred_5 == 0 & data$HeartDiseaseorAttack[fold == i] == 1))
    sens_estimates[i,3] <- sum(pred_10 == 1 & data$HeartDiseaseorAttack[fold == i] == 1) /
                (sum(pred_10 == 1 & data$HeartDiseaseorAttack[fold == i] == 1) +
                sum(pred_10 == 0 & data$HeartDiseaseorAttack[fold == i] == 1))
    sens_estimates[i,4] <- sum(pred_15 == 1 & data$HeartDiseaseorAttack[fold == i] == 1) /
                (sum(pred_15 == 1 & data$HeartDiseaseorAttack[fold == i] == 1) +
                sum(pred_15 == 0 & data$HeartDiseaseorAttack[fold == i] == 1))
    sens_estimates[i,5] <- sum(pred_20 == 1 & data$HeartDiseaseorAttack[fold == i] == 1) /
                (sum(pred_20 == 1 & data$HeartDiseaseorAttack[fold == i] == 1) +
                sum(pred_20 == 0 & data$HeartDiseaseorAttack[fold == i] == 1))
    sens_estimates[i,6] <- sum(pred_25 == 1 & data$HeartDiseaseorAttack[fold == i] == 1) /
                (sum(pred_25 == 1 & data$HeartDiseaseorAttack[fold == i] == 1) +
                sum(pred_25 == 0 & data$HeartDiseaseorAttack[fold == i] == 1))


    spec_estimates[i,1] <- sum(pred_3 == 0 & data$HeartDiseaseorAttack[fold == i] == 0) /
                (sum(pred_3 == 0 & data$HeartDiseaseorAttack[fold == i] == 0) +
                sum(pred_3 == 1 & data$HeartDiseaseorAttack[fold == i] == 0))
    spec_estimates[i,2] <- sum(pred_5 == 0 & data$HeartDiseaseorAttack[fold == i] == 0) /
                (sum(pred_5 == 0 & data$HeartDiseaseorAttack[fold == i] == 0) +
                sum(pred_5 == 1 & data$HeartDiseaseorAttack[fold == i] == 0))
    spec_estimates[i,3] <- sum(pred_10 == 0 & data$HeartDiseaseorAttack[fold == i] == 0) /
                (sum(pred_10 == 0 & data$HeartDiseaseorAttack[fold == i] == 0) +
                sum(pred_10 == 1 & data$HeartDiseaseorAttack[fold == i] == 0))
    spec_estimates[i,4] <- sum(pred_15 == 0 & data$HeartDiseaseorAttack[fold == i] == 0) /
                (sum(pred_15 == 0 & data$HeartDiseaseorAttack[fold == i] == 0) +
                sum(pred_15 == 1 & data$HeartDiseaseorAttack[fold == i] == 0))
    spec_estimates[i,5] <- sum(pred_20 == 0 & data$HeartDiseaseorAttack[fold == i] == 0) /
                (sum(pred_20 == 0 & data$HeartDiseaseorAttack[fold == i] == 0) +
                sum(pred_20 == 1 & data$HeartDiseaseorAttack[fold == i] == 0))
    spec_estimates[i,6] <- sum(pred_25 == 0 & data$HeartDiseaseorAttack[fold == i] == 0) /
                (sum(pred_25 == 0 & data$HeartDiseaseorAttack[fold == i] == 0) +
                sum(pred_25 == 1 & data$HeartDiseaseorAttack[fold == i] == 0))
  }
  output1 <- data.frame(matrix(data = c(mean(er_estimates[,1]), mean(er_estimates[,2]),
                                        mean(er_estimates[,3]), mean(er_estimates[,4]),
                                        mean(er_estimates[,5]), mean(er_estimates[,6]),
                                        mean(sens_estimates[,1]), mean(sens_estimates[,2]),
                                        mean(sens_estimates[,3]), mean(sens_estimates[,4]),
                                        mean(sens_estimates[,5]), mean(sens_estimates[,6]),
                                        mean(spec_estimates[,1]), mean(spec_estimates[,2]),
                                        mean(spec_estimates[,3]), mean(spec_estimates[,4]),
                                        mean(spec_estimates[,5]), mean(spec_estimates[,6])),
                                        nrow=6, ncol=3))

  colnames(output1) <- c("Error Rate", "Sensitivity", "Specificity")
  rownames(output1) <- c("k=3", "k=5", "k=10", "k=15", "k=20", "k=25")
  print(output1)
}

kfolds_knn(heart_disease, 10)
```

```
##      Error Rate Sensitivity Specificity
## k=3  0.09643211  0.05290610   0.9841115
## k=5  0.09029698  0.02277278   0.9937054
```

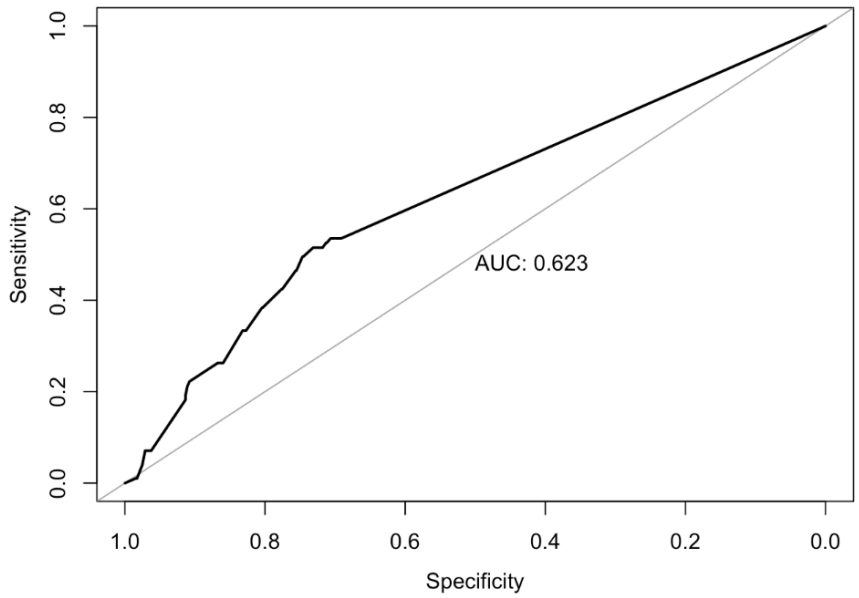
```
## k=10 0.08713723  0.00365570   0.9988553
## k=15 0.08660999  0.00000000   0.9998081
## k=20 0.08643455  0.00000000   1.0000000
## k=25 0.08643455  0.00000000   1.0000000
```

```
knn_mod_1 <- knn(train_1[,2:11], test_1[,2:11], cl = train_1$HeartDiseaseorAttack, k = 3,
                 prob = TRUE)
knn_probs <- attr(knn_mod_1, "prob")
knn_probs <- 1 - knn_probs
knn_roc <- roc(test_1$HeartDiseaseorAttack, knn_probs)
```

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

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

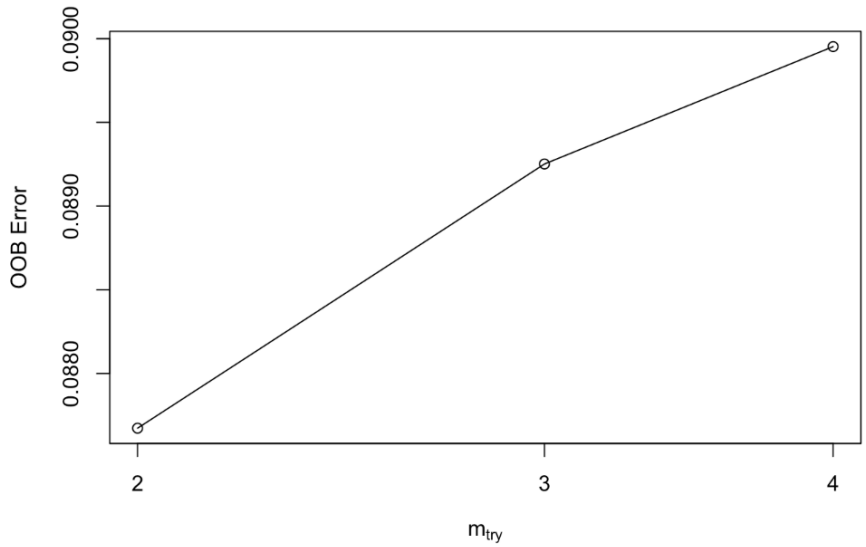
```
plot(knn_roc, asp = 0, print.auc = TRUE)
```




```
set.seed(2024)
library(randomForest)
```

```
tuned_rf_1 <- tuneRF(x = heart_disease[,2:11], y = heart_disease[,1],
                     ntreeTry = 500, stepFactor = 1.5, improve = 1e-5)
```

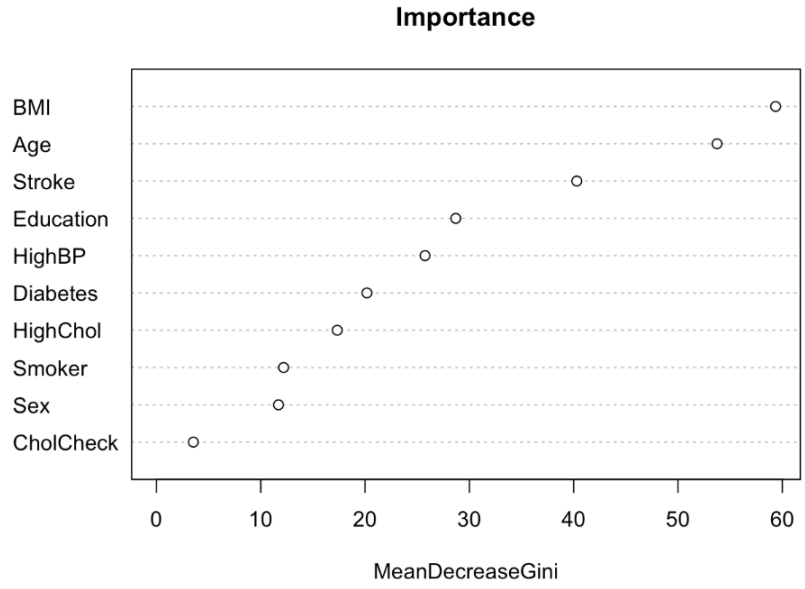
```
## mtry = 3  OOB error = 8.93%
## Searching left ...
## mtry = 2     OOB error = 8.77%
## 0.01768173 1e-05
## Searching right ...
## mtry = 4     OOB error = 9%
## -0.026 1e-05
```




```
RF_1 <- randomForest(x = heart_disease[,2:11], y = heart_disease[,1], mtry = 2, ntree = 500)
print(RF_1)
```

```
##
## Call:
##  randomForest(x = heart_disease[, 2:11], y = heart_disease[, 1],      ntree = 500, mtry = 2)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 8.7%
## Confusion matrix:
##      0 1 class.error
## 0 5204 6 0.001151631
## 1  490 3 0.993914807
```

```
varImpPlot(RF_1, main = "Importance")
```

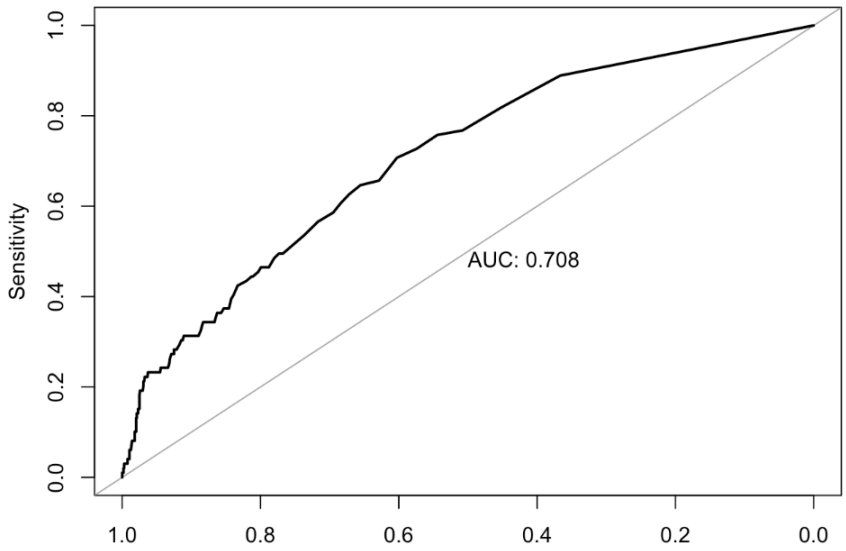



```
RF_ROC_mod_1 <- randomForest(x = train_1[,2:11], y = train_1[,1], mtry = 2, ntree = 500)
RF_probs_1 <- predict(RF_ROC_mod_1, newdata = test_1[,2:11], type = "prob")
RF_probs_1 <- RF_probs_1[, 2]
RF_ROC_1 <- roc(test_1$HeartDiseaseorAttack, RF_probs_1)
```

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

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

```
plot(RF_ROC_1, asp = 0, print.auc = TRUE)
```



```
library(nnet)
set.seed(2024)

heart_disease_iter <- heart_disease_full
summary(heart_disease_iter)
```

```
##  HeartDiseaseorAttack HighBP   HighChol CholCheck      BMI        Smoker
##  0   :6083            0:3947   0:4025   0: 281    Min.   :14.00   0:3988
##  1   : 623            1:3053   1:2975   1:6719    1st Qu.:24.00   1:3012
##  NA's: 294                                       Median :27.00
##                                                  Mean   :28.38
##                                                  3rd Qu.:31.00
##                                                  Max.   :92.00
##   Stroke     Diabetes    Sex           Age          Education
##  0   :6182   0   :5335   0:3932   Min.   : 1.000   1:   6
##  1   : 284   1   : 138   1:3068   1st Qu.: 6.000   2: 218
##  NA's: 534   2   : 961            Median : 8.000   3: 355
##              NA's: 566            Mean   : 7.895   4:1589
##                                   3rd Qu.:10.000   5:1977
##                                   Max.   :13.000   6:2855
```

```
heart_disease_iter$HeartDiseaseorAttack[is.na(heart_disease_iter$HeartDiseaseorAttack)] <- 0
heart_disease_iter$Stroke[is.na(heart_disease_iter$Stroke)] <- 0
heart_disease_iter$Diabetes[is.na(heart_disease_iter$Diabetes)] <- 0

for(i in 1:20){
  m_HeartDiseaseorAttack <- multinom(HeartDiseaseorAttack ~.,heart_disease_iter,
                      subset=!is.na(heart_disease_full$HeartDiseaseorAttack),trace=FALSE)
  heart_disease_pred <- predict(m_HeartDiseaseorAttack,heart_disease_iter[is.na(heart_disease_full$HeartDiseaseor
Attack),])
  heart_disease_iter$HeartDiseaseorAttack[is.na(heart_disease_full$HeartDiseaseorAttack)] <- heart_disease_pred

  m_Stroke <- multinom(Stroke ~., heart_disease_iter, subset=!is.na(heart_disease_full$Stroke),trace=FALSE)
  Stroke_pred <- predict(m_Stroke,heart_disease_iter[is.na(heart_disease_full$Stroke),])
  heart_disease_iter$Stroke[is.na(heart_disease_full$Stroke)] <- Stroke_pred

  m_Diabetes <- multinom(Diabetes ~., heart_disease_iter, subset=!is.na(heart_disease_full$Diabetes),trace=FALSE)
  Diabetes_pred <- predict(m_Diabetes,heart_disease_iter[is.na(heart_disease_full$Diabetes),])
  heart_disease_iter$Diabetes[is.na(heart_disease_full$Diabetes)] <- Diabetes_pred
}
summary(heart_disease_iter)
```

```
##  HeartDiseaseorAttack HighBP   HighChol CholCheck      BMI        Smoker
##  0:6373               0:3947   0:4025   0: 281    Min.   :14.00   0:3988
##  1: 627               1:3053   1:2975   1:6719    1st Qu.:24.00   1:3012
##                                                  Median :27.00
##                                                  Mean   :28.38
##                                                  3rd Qu.:31.00
##                                                  Max.   :92.00
##  Stroke   Diabetes Sex           Age          Education
##  0:6716   0:5882   0:3932   Min.   : 1.000   1:   6
##  1: 284   1: 138   1:3068   1st Qu.: 6.000   2: 218
##           2: 980            Median : 8.000   3: 355
##                             Mean   : 7.895   4:1589
##                             3rd Qu.:10.000   5:1977
##                             Max.   :13.000   6:2855
```

```
set.seed(2024)
kfolds_logistic(heart_disease_iter, 10)
```

```
set.seed(2024)
kfolds_logistic(heart_disease_iter, 10)
```

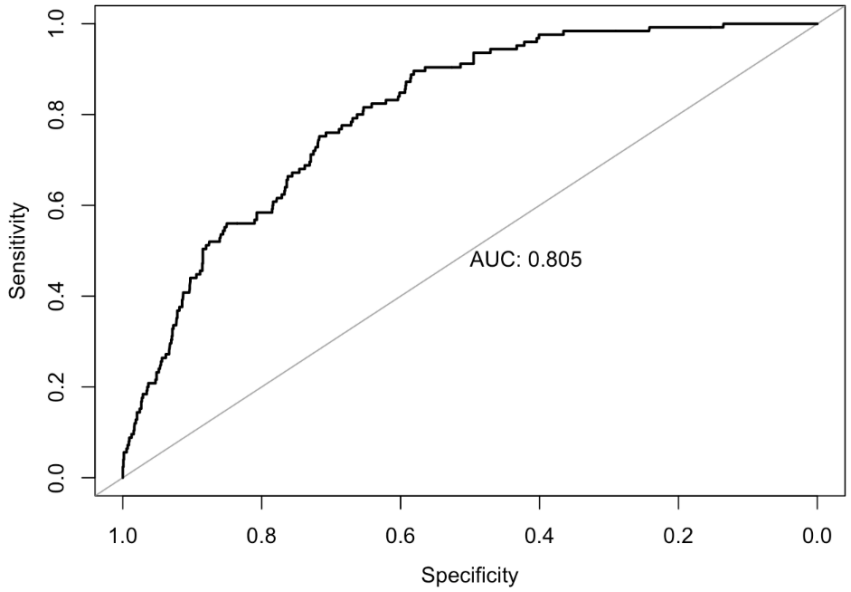
```
## [1] 0.08957143
## [1] 0.0753866
## [1] 0.9929196
```

```
logistic_full_mod_1 <- glm(HeartDiseaseorAttack ~., data = heart_disease_iter, family = "binomial")
log_mod2 <- step(logistic_full_mod,scope=list(lower=~1,upper=~.), trace = FALSE)

summary(log_mod1)
```

```
##
## Call:
## glm(formula = HeartDiseaseorAttack ~ HighBP + HighChol + BMI +
##     Smoker + Stroke + Diabetes + Sex + Age, family = "binomial",
##     data = heart_disease)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.457578   0.366739 -17.608  < 2e-16 ***
## HighBP1      0.879710   0.123238   7.138 9.45e-13 ***
## HighChol1    0.518220   0.111458   4.649 3.33e-06 ***
## BMI          0.023213   0.008249   2.814 0.004892 **
## Smoker1      0.447431   0.103348   4.329 1.50e-05 ***
## Stroke1      1.584045   0.152023  10.420  < 2e-16 ***
## Diabetes1    0.509098   0.275176   1.850 0.064302 .
## Diabetes2    0.412492   0.120268   3.430 0.000604 ***
## Sex1         0.322366   0.103394   3.118 0.001822 **
## Age          0.223605   0.022729   9.838  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3356.1  on 5702  degrees of freedom
## Residual deviance: 2731.6  on 5693  degrees of freedom
## AIC: 2751.6
##
## Number of Fisher Scoring iterations: 6
```

```
data_split_2 <- sample.split(heart_disease_iter$HeartDiseaseorAttack, SplitRatio = 0.8 )

train_2 <- filter(heart_disease_iter, data_split_2 == TRUE)
test_2 <- filter(heart_disease_iter, data_split_2 == FALSE)
train_2 <-  as.data.frame(train_2)
test_2 <- as.data.frame(test_2)

log_ROC_mod2 <- glm(HeartDiseaseorAttack ~ HighBP + HighChol + Smoker + Diabetes +
                      Stroke + Sex + BMI + Age, data = train_2, family = "binomial")
log_ROC_1 <- roc(test_2$HeartDiseaseorAttack, predict(log_ROC_mod2, newdata = test_2[,2:11],
                                                      type = "response"))
```

```
plot(log_ROC_1, asp = 0, print.auc = TRUE)
```




```
set.seed(2024)

kfolds_knn(heart_disease_iter, 10)
```

```
##       Error Rate Sensitivity Specificity
## k=3   0.10028571 0.051396373   0.9828968
## k=5   0.09485714 0.025802670   0.9915419
## k=10 0.09142857 0.003598485   0.9976606
## k=15 0.08985714 0.000000000   0.9996815
## k=20 0.08957143 0.000000000   1.0000000
## k=25 0.08957143 0.000000000   1.0000000
```

```
knn_mod_2 <- knn(train_2[,2:11], test_2[,2:11], cl = train_2$HeartDiseaseorAttack, k = 3,
                 prob = TRUE)
knn_probs_1 <- attr(knn_mod_2, "prob")
knn_probs_1 <- 1 - knn_probs_1
knn_roc_1 <- roc(test_2$HeartDiseaseorAttack, knn_probs_1)
```

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

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

```
plot(knn_roc_1, asp = 0, print.auc = TRUE)
```

```
set.seed(2024)

kfolds_knn(heart_disease_iter, 10)
```

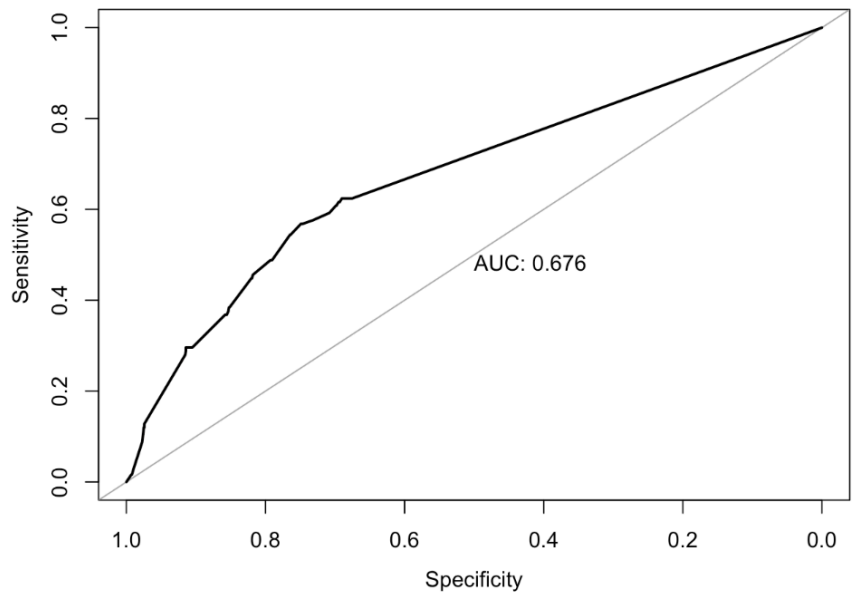
```
##      Error Rate Sensitivity Specificity
## k=3  0.10028571 0.051396373   0.9828968
## k=5  0.09485714 0.025802670   0.9915419
## k=10 0.09142857 0.003598485   0.9976606
## k=15 0.08985714 0.000000000   0.9996815
## k=20 0.08957143 0.000000000   1.0000000
## k=25 0.08957143 0.000000000   1.0000000
```

```
knn_mod_2 <- knn(train_2[,2:11], test_2[,2:11], cl = train_2$HeartDiseaseorAttack, k = 3,
                 prob = TRUE)
knn_probs_1 <- attr(knn_mod_2, "prob")
knn_probs_1 <- 1 - knn_probs_1
knn_roc_1 <- roc(test_2$HeartDiseaseorAttack, knn_probs_1)
```

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

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

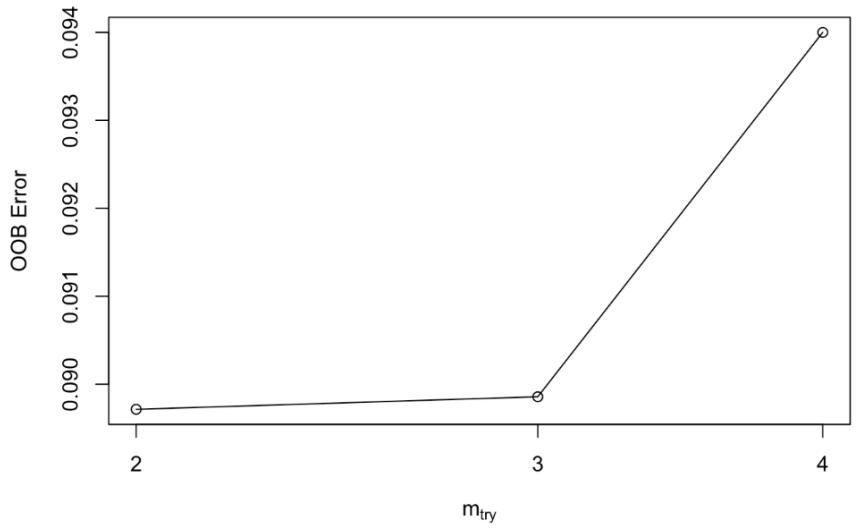
```
plot(knn_roc_1, asp = 0, print.auc = TRUE)
```




```
set.seed(2024)

tuned_rf_2 <- tuneRF(x = heart_disease_iter[,2:11], y = heart_disease_iter[,1],
                     ntreeTry = 500, stepFactor = 1.5, improve = 1e-5)
```

```
set.seed(2024)

tuned_rf_2 <- tuneRF(x = heart_disease_iter[,2:11], y = heart_disease_iter[,1],
                     ntreeTry = 500, stepFactor = 1.5, improve = 1e-5)
```

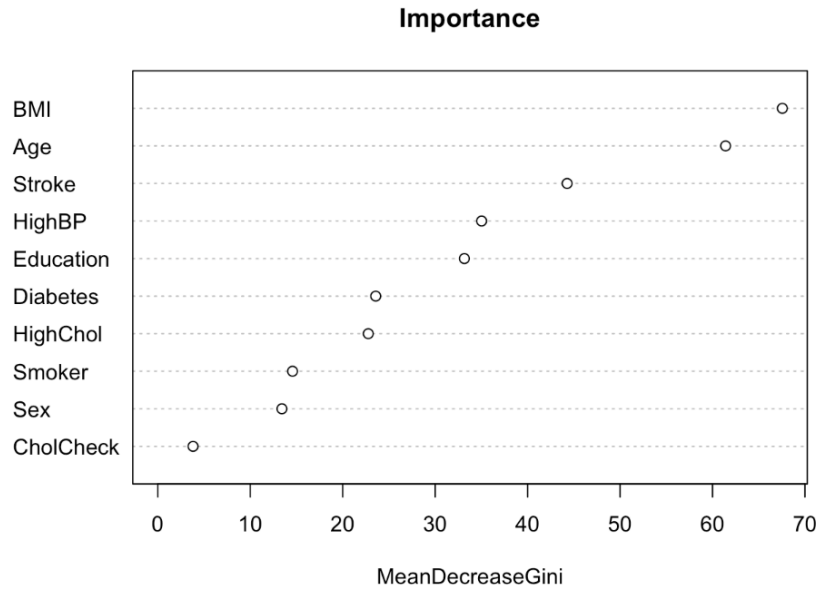
```
## mtry = 3  OOB error = 8.99%
## Searching left ...
## mtry = 2     OOB error = 8.97%
## 0.001589825 1e-05
## Searching right ...
## mtry = 4     OOB error = 9.4%
## -0.0477707 1e-05
```




```
RF_2 <- randomForest(x = heart_disease_iter[,2:11], y = heart_disease_iter[,1], mtry = 2, ntree = 500)
print(RF_2)
```

```
##
## Call:
##  randomForest(x = heart_disease_iter[, 2:11], y = heart_disease_iter[,      1], ntree = 500, mtry = 2)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 8.99%
## Confusion matrix:
##      0 1  class.error
## 0 6367 6 0.0009414718
## 1  623 4 0.9936204147
```

```
varImpPlot(RF_2, main = "Importance")
```




```
RF_ROC_mod_2 <- randomForest(x = train_2[,2:11], y = train_2[,1], mtry = 2, ntree = 500)
RF_probs_2 <- predict(RF_ROC_mod_2, newdata = test_2[,2:11], type = "prob")
RF_probs_2 <- RF_probs_2[, 2]
RF_ROC_2 <- roc(test_2$HeartDiseaseorAttack, RF_probs_2)
```

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

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

```
plot(RF_ROC_2, asp = 0, print.auc = TRUE)
```