Models (Cholesterol with mean imputation)

Satwik Behera

2023-05-02

Data Preprocessing

```
# Suppress Warnings
options(warn=-1)



# Installing, loading packages
pacman::p_load("tidyverse", "phia", "MASS", "car", "pROC", "caret", "scales",
               "lattice", "randomForest", "rpart", "rpart.plot", "e1071", "reshape2")

# Loading and reading the dataset
df <- read.csv("heart.csv")
head(df)
```

```
##   Age Sex ChestPainType RestingBP Cholesterol FastingBS RestingECG MaxHR
## 1  40   M           ATA       140         289         0     Normal   172
## 2  49   F           NAP       160         180         0     Normal   156
## 3  37   M           ATA       130         283         0         ST    98
## 4  48   F           ASY       138         214         0     Normal   108
## 5  54   M           NAP       150         195         0     Normal   122
## 6  39   M           NAP       120         339         0     Normal   170
##   ExerciseAngina Oldpeak ST_Slope HeartDisease
## 1              N     0.0       Up            0
## 2              N     1.0     Flat            1
## 3              N     0.0       Up            0
## 4              Y     1.5     Flat            1
## 5              N     0.0       Up            0
## 6              N     0.0       Up            0
```

```
# Converting all the relevant columns to factors
cols <- c("Sex", "ChestPainType", "RestingECG", "ExerciseAngina",
          "ST_Slope", "HeartDisease", "FastingBS")

df <- df %>%
  mutate_at(cols, factor)

set.seed(123)

# remove the row with RestingBP = 0
df <- df %>%
  filter(RestingBP != 0)
```

```r
# define the columns to be scaled
cols_to_scale <- c("Age", "RestingBP", "Oldpeak", "MaxHR")

# standardize the other columns using scale function
df[, cols_to_scale] <- as.data.frame(scale(df[, cols_to_scale]))

# Mean Imputation in Cholesterol Columns\
m_Chol <- mean(df$Cholesterol, na.rm = T)
df$Cholesterol <- replace(df$Cholesterol, 0, m_Chol)

# Train-Test Split
smp_size <- floor(0.80 * nrow(df))
train_ind <- sample(seq_len(nrow(df)), size = smp_size)
train <- df[train_ind, ]
test <- df[-train_ind, ]
```

Logistic Regression Models

```r
model1 <- glm(HeartDisease ~ ., data=train, family="binomial") ##
model2 <- glm(HeartDisease ~ (Sex + ChestPainType + FastingBS + Cholesterol +
                                ExerciseAngina + Oldpeak +ST_Slope),
              data=train, family="binomial") # Significant from model1

model3 <- glm(HeartDisease ~ (Sex + ChestPainType + FastingBS + Cholesterol +
                                ExerciseAngina + Oldpeak +ST_Slope)^2,
              data=train, family="binomial") ###

model4 <- step(model3, trace=0) #####

# model5 <- glm(HeartDisease ~ .^2, data=train, family="binomial")

# model6 <- step(model5, trace=0)
```

Model Summaries

```r
summary(model4)
```

```
##
## Call:
## glm(formula = HeartDisease ~ Sex + ChestPainType + FastingBS +
##     Cholesterol + ExerciseAngina + Oldpeak + ST_Slope + ChestPainType:FastingBS +
##     FastingBS:Cholesterol + FastingBS:ST_Slope + Cholesterol:Oldpeak +
##     ExerciseAngina:ST_Slope + Oldpeak:ST_Slope, family = "binomial",
##     data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.69013  -0.35112   0.02103   0.41460   2.84142
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept)                  -3.4340693  1.6309911  -2.106  0.03525 *
## SexM                          1.7957183  0.3263996   5.502 3.76e-08 ***
## ChestPainTypeATA             -1.7059005  0.4047273  -4.215 2.50e-05 ***
## ChestPainTypeNAP             -1.5069549  0.3156094  -4.775 1.80e-06 ***
## ChestPainTypeTA              -1.3170183  0.5577736  -2.361  0.01822 *
## FastingBS1                    7.2858350  2.5984364   2.804  0.00505 **
## Cholesterol                  -0.0003403  0.0012937  -0.263  0.79251
## ExerciseAnginaY               3.6314607  1.6431644   2.210  0.02710 *
## Oldpeak                       0.7266038  0.5732497   1.268  0.20497
## ST_SlopeFlat                  3.6283428  1.6125535   2.250  0.02445 *
## ST_SlopeUp                    1.3753486  1.6190943   0.849  0.39563
## ChestPainTypeATA:FastingBS1  -0.4385347  1.4440984  -0.304  0.76138
## ChestPainTypeNAP:FastingBS1  -2.8861337  1.1914860  -2.422  0.01542 *
## ChestPainTypeTA:FastingBS1   -1.4164543  1.5283628  -0.927  0.35404
## FastingBS1:Cholesterol       -0.0154947  0.0048366  -3.204  0.00136 **
## FastingBS1:ST_SlopeFlat      -0.4592890  2.0295974  -0.226  0.82097
## FastingBS1:ST_SlopeUp        -3.2181928  2.1414217  -1.503  0.13288
## Cholesterol:Oldpeak           0.0020057  0.0013241   1.515  0.12985
## ExerciseAnginaY:ST_SlopeFlat -2.8530335  1.6737114  -1.705  0.08827 .
## ExerciseAnginaY:ST_SlopeUp   -2.4373536  1.7060028  -1.429  0.15309
## Oldpeak:ST_SlopeFlat         -1.1467853  0.5781918  -1.983  0.04732 *
## Oldpeak:ST_SlopeUp           -0.1810794  0.6105921  -0.297  0.76680
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1008.47  on 732  degrees of freedom
## Residual deviance:  446.88  on 711  degrees of freedom
## AIC: 490.88
##
## Number of Fisher Scoring iterations: 8
```

```
"ROC curve and AUC"
```

```
## [1] "ROC curve and AUC"
```

```
p_mod1<-predict(model1, newdata=test, type = "response")
p_mod3<-predict(model3, newdata=test, type = "response")
p_mod4<-predict(model4, newdata=test, type = "response")

roc_mod1=roc(response=test$HeartDisease, predictor= factor(p_mod1, ordered = TRUE))
```

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

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

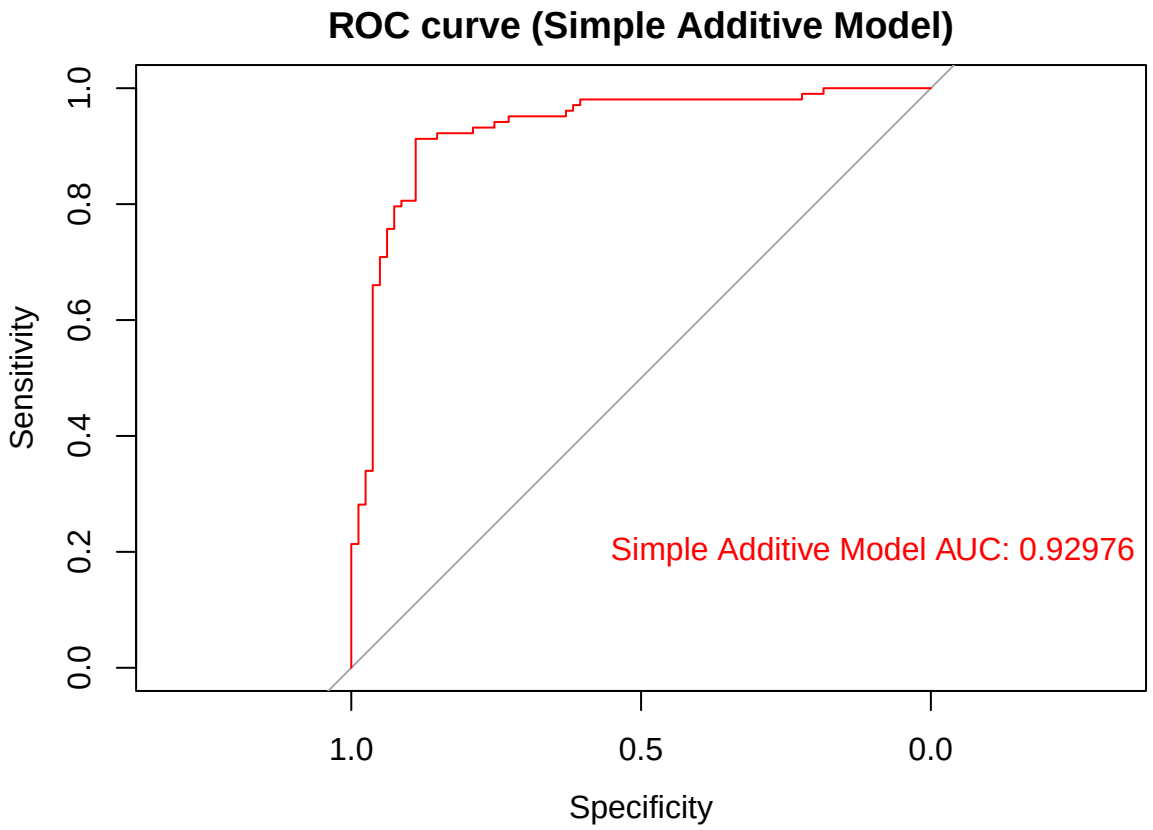
```
roc_mod3=roc(response=test$HeartDisease, predictor= factor(p_mod3, ordered = TRUE))
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

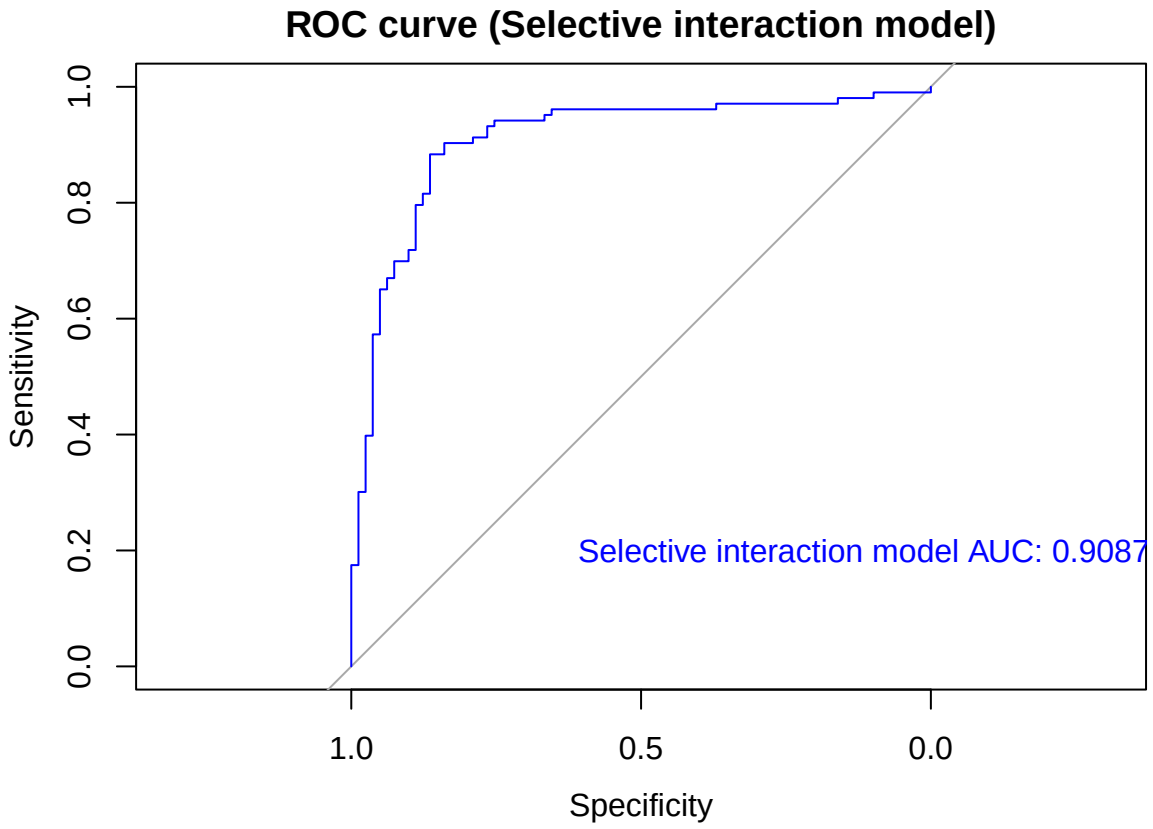
```
roc_mod4=roc(response=test$HeartDisease, predictor= factor(p_mod4, ordered = TRUE))
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

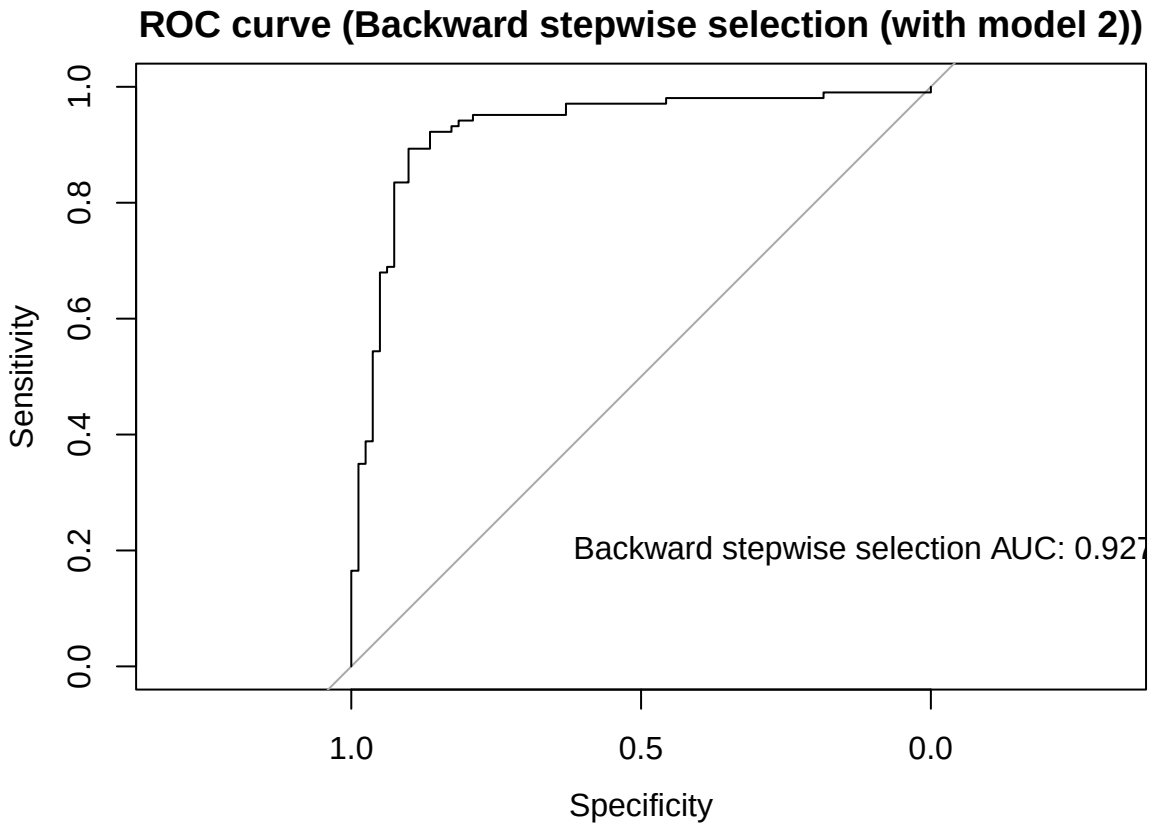
```
plot(roc_mod1, col="red", lwd=1, main="ROC curve (Simple Additive Model)")
text(0.1, 0.2, paste("Simple Additive Model AUC:", round(auc(roc_mod1), 5)), col = "red")
```




```
plot(roc_mod3, col="blue", lwd=1, main="ROC curve (Selective interaction model)")
text(0.1, 0.2, paste("Selective interaction model AUC:", round(auc(roc_mod3), 5)), col = "blue")
```

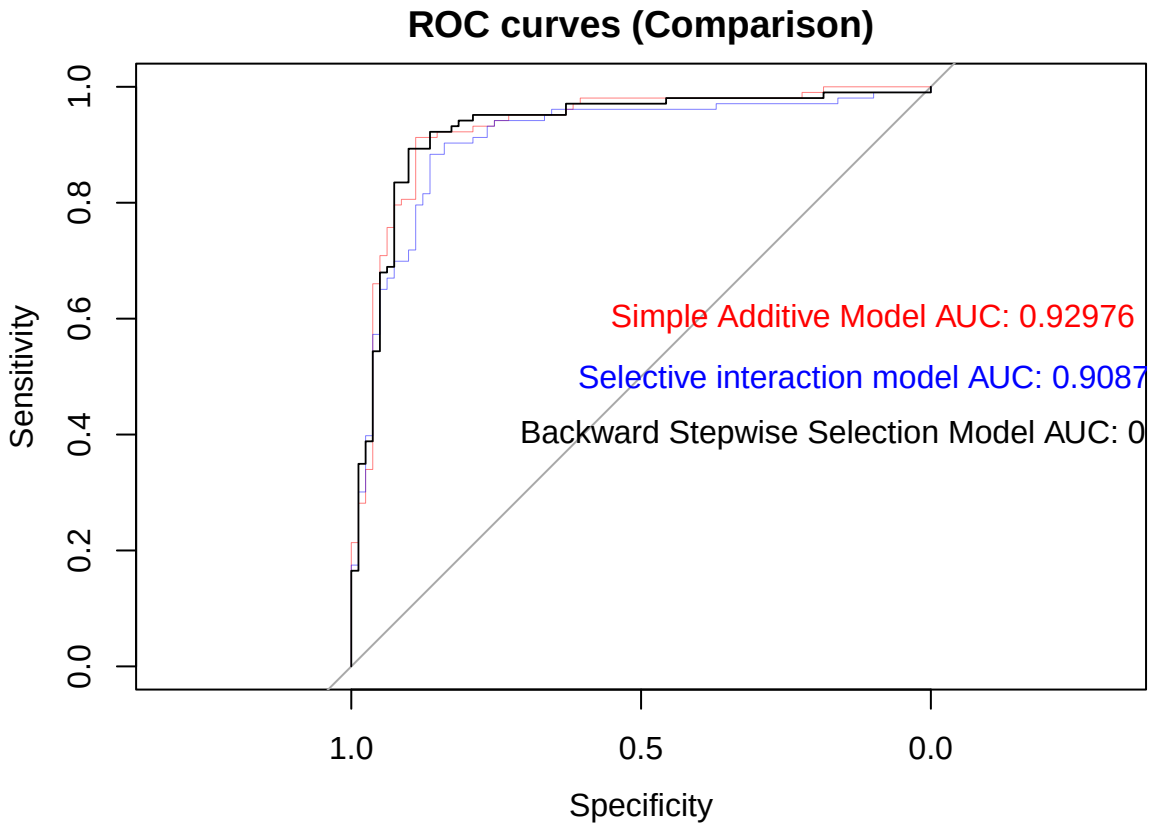



```
plot(roc_mod4, col="black", lwd=1, main="ROC curve (Backward stepwise selection (with model 2))")
text(0.1, 0.2, paste("Backward stepwise selection AUC:", round(auc(roc_mod4), 5)), col = "black")
```



Plotting ROC Curves

```
plot(roc_mod1, col=alpha("red", 0.5), lwd=0.5, main="ROC curves (Comparison)")
lines(roc_mod3, col=alpha("blue", 0.5), lwd=0.5)
lines(roc_mod4, col="black", lwd=0.9)
text(0.1, 0.6, paste("Simple Additive Model AUC:", round(auc(roc_mod1), 5)), col = "red")
text(0.1, 0.5, paste("Selective interaction model AUC:", round(auc(roc_mod3), 5)), col = "blue")
text(0.1, 0.4, paste("Backward Stepwise Selection Model AUC:", round(auc(roc_mod4), 5)), col = "black")
```



Confusion Matrix

```
predicted <- predict(model4, test, type="response")
predicted <- ifelse(predicted < 0.5, "0", "1")
prop.table(table(test$HeartDisease, predicted))
```

```
##    predicted
##                0          1
##   0 0.36413043 0.07608696
##   1 0.04347826 0.51630435
```

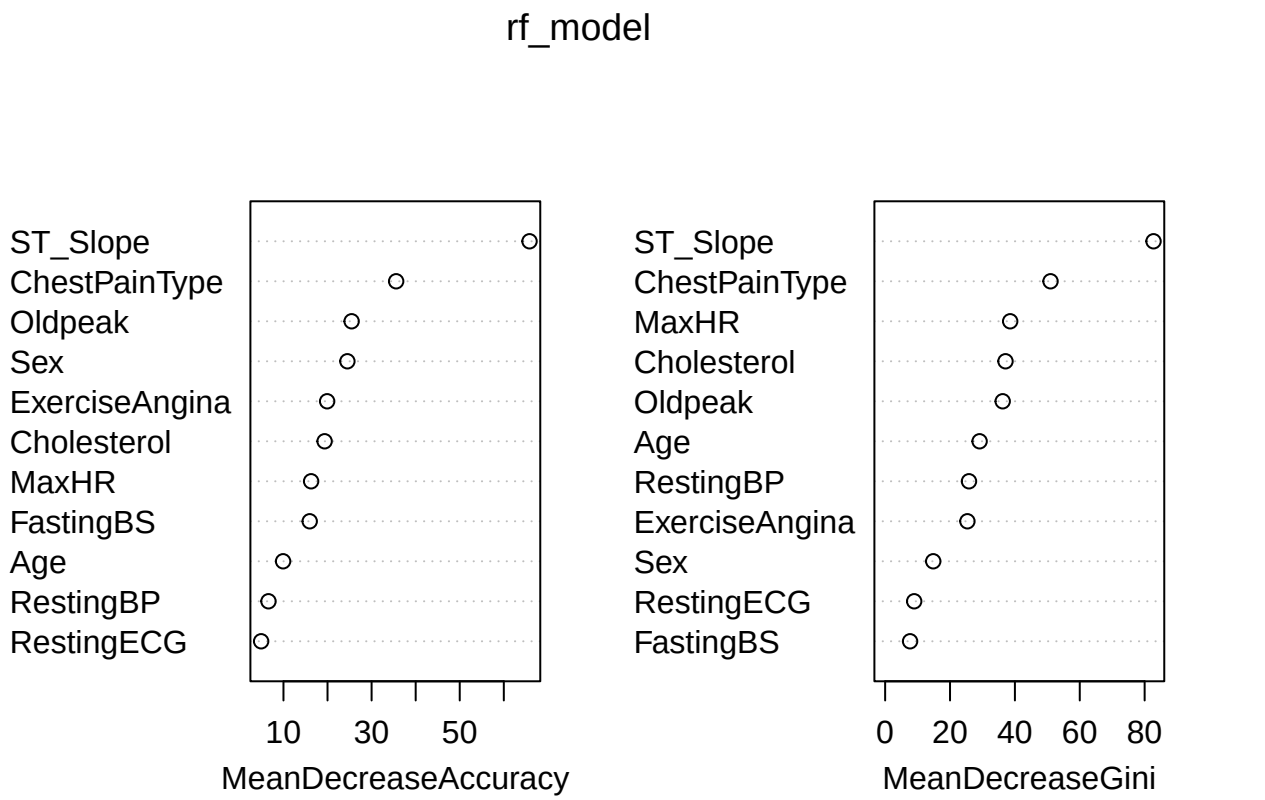
Random Forest

```
# Build the random forest model
rf_model <- randomForest(HeartDisease ~ ., data = train,
                         importance = TRUE, ntree = 500, mtry = sqrt(ncol(train)))
# Make predictions on the test set
rf_predictions <- predict(rf_model, newdata = test)
# Evaluate the model performance
confusionMatrix(rf_predictions, test$HeartDisease)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 68  8
```

```
##          1 13 95
##
##                Accuracy : 0.8859
##                  95% CI : (0.8308, 0.9279)
##     No Information Rate : 0.5598
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7669
##
##  Mcnemar's Test P-Value : 0.3827
##
##             Sensitivity : 0.8395
##             Specificity : 0.9223
##          Pos Pred Value : 0.8947
##          Neg Pred Value : 0.8796
##              Prevalence : 0.4402
##          Detection Rate : 0.3696
##    Detection Prevalence : 0.4130
##       Balanced Accuracy : 0.8809
##
##        'Positive' Class : 0
##
```

```
varImpPlot(rf_model)
```

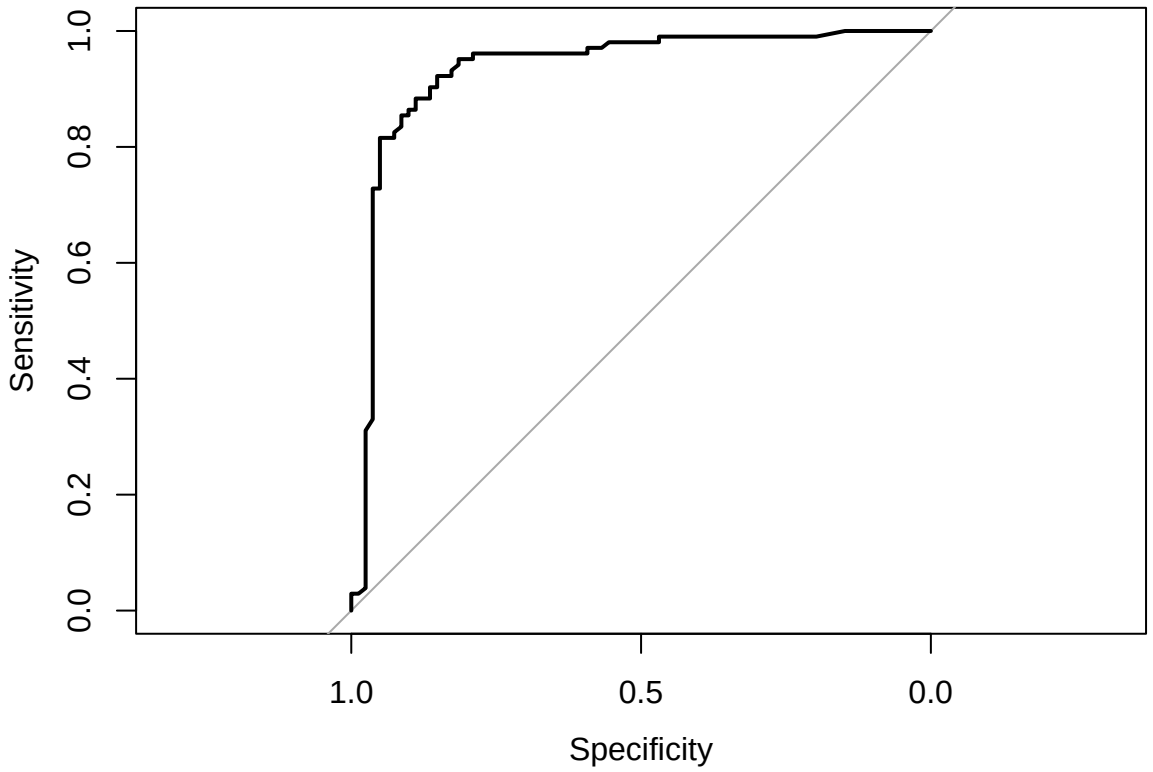



```
rf_roc <- roc(test$HeartDisease, predict(rf_model, newdata = test, type = "prob")[,2])
```

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

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

```
plot(rf_roc)
```

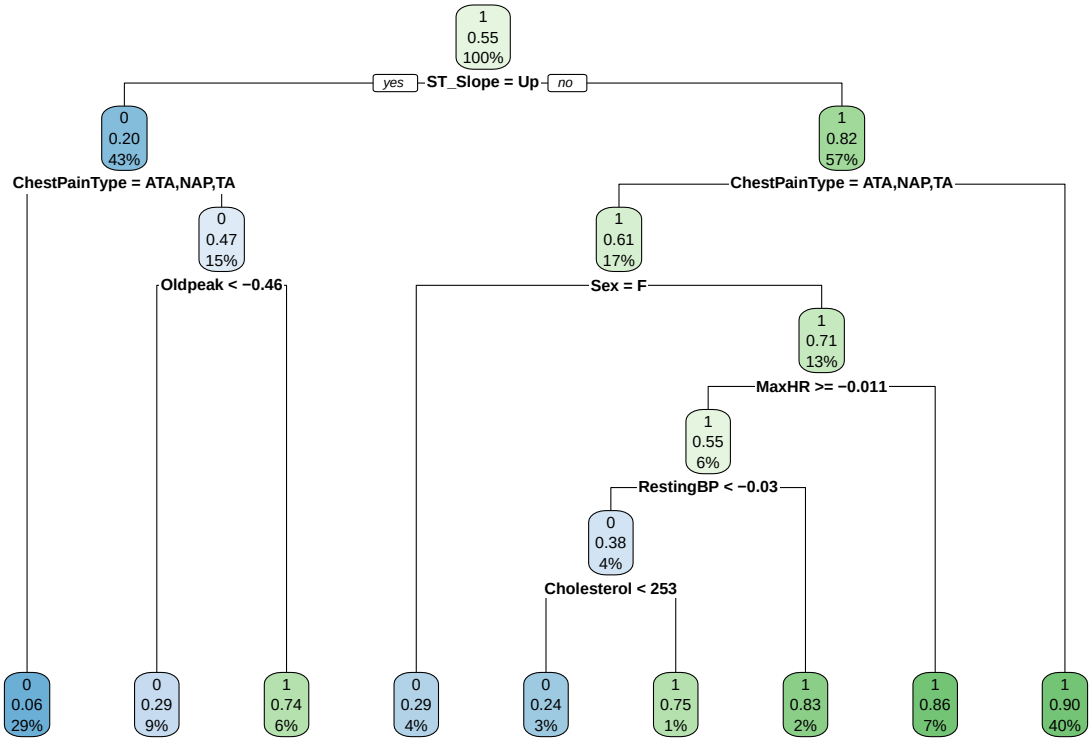



```
auc(rf_roc)
```

```
## Area under the curve: 0.933
```

Decision Tree

```
# Build the decision tree model
dt_model <- rpart(HeartDisease ~ ., data = train, method = "class")
# Visualize the decision tree
rpart.plot(dt_model)
```




```
# Make predictions on the test set
dt_pred <- predict(dt_model, test, type = "class")
# Evaluate the model
confusionMatrix(dt_pred, test$HeartDisease)
```

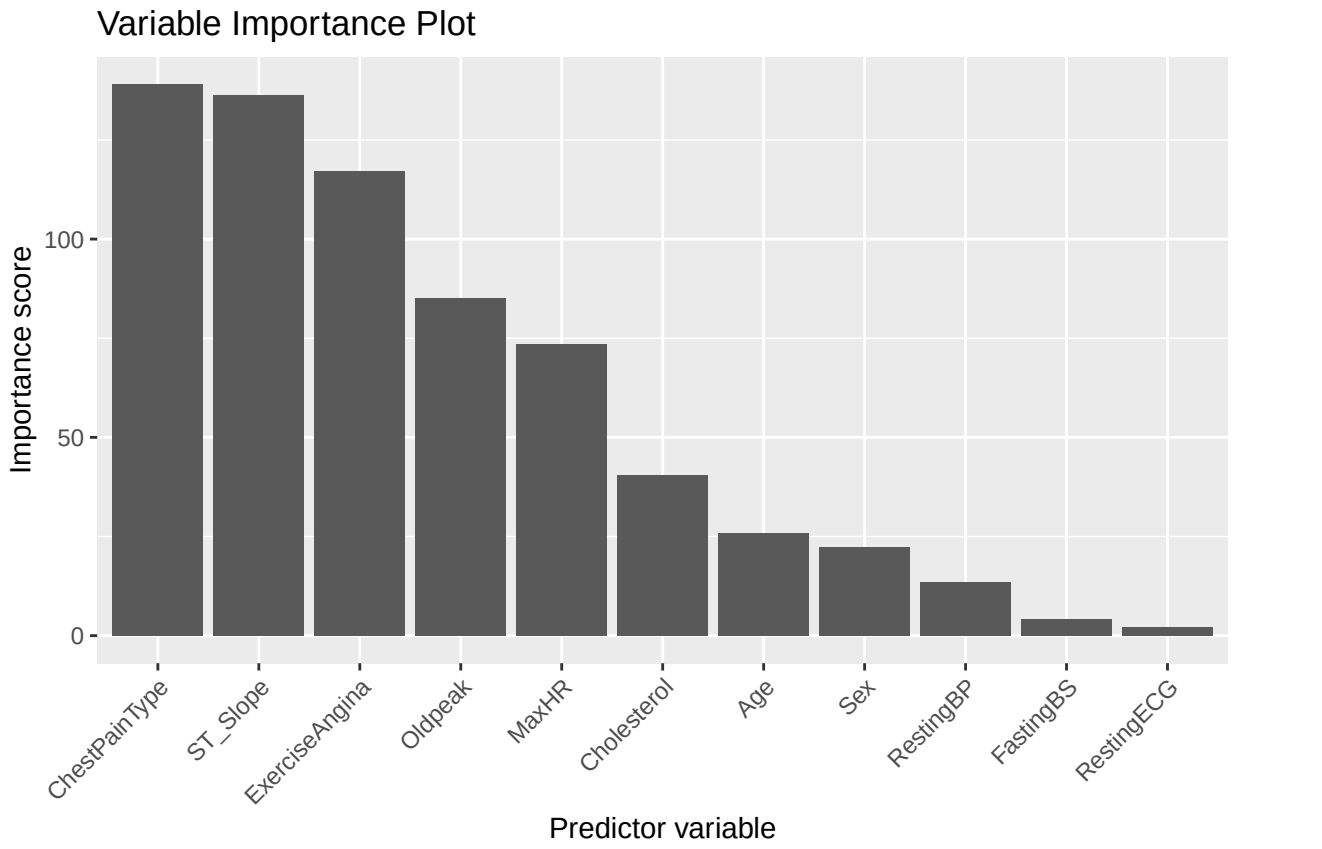
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 71 16
##          1 10 87
##
##                Accuracy : 0.8587
##                  95% CI : (0.7998, 0.9056)
##     No Information Rate : 0.5598
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7155
##
##  Mcnemar's Test P-Value : 0.3268
##
##             Sensitivity : 0.8765
##             Specificity : 0.8447
##          Pos Pred Value : 0.8161
##          Neg Pred Value : 0.8969
##              Prevalence : 0.4402
##          Detection Rate : 0.3859
##    Detection Prevalence : 0.4728
##       Balanced Accuracy : 0.8606
##
```

```
##        'Positive' Class : 0
##
```

```
# Check variable importance
var_dt <- varImp(dt_model)
var_dt
```

```
##                   Overall
## Age             25.913749
## ChestPainType  139.010500
## Cholesterol     40.516114
## ExerciseAngina 117.076871
## FastingBS        4.163768
## MaxHR           73.604708
## Oldpeak         85.063125
## RestingBP       13.401121
## RestingECG       2.038257
## Sex             22.196519
## ST_Slope       136.274531
```

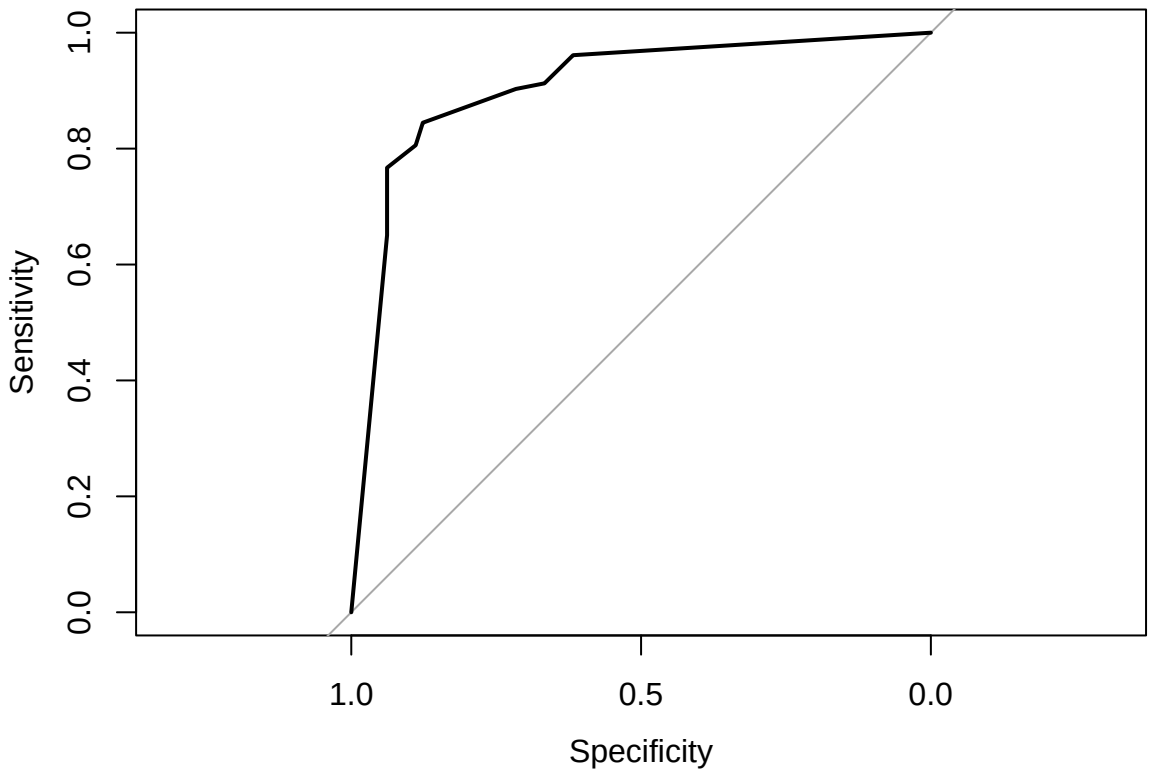
```
ggplot(var_dt, aes(x = reorder(rownames(var_dt), -Overall), y = Overall)) +
  geom_bar(stat = "identity") +
  labs(x = "Predictor variable", y = "Importance score", title = "Variable Importance Plot") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
# Generate ROC curve and calculate AUC
library(pROC)
dt_roc <- roc(test$HeartDisease, predict(dt_model, newdata = test, type = "prob")[,2])
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
plot(dt_roc)
```




```
auc(dt_roc)
```

```
## Area under the curve: 0.9057
```

Naive Bayes

```
# Build the Naive Bayes model
nb_model <- naiveBayes(HeartDisease ~ ., data = train) # Make predictions on the test set
nb_pred <- predict(nb_model, test, type = "class")
# Evaluate the model
confusionMatrix(nb_pred, test$HeartDisease)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 67 10
##          1 14 93
##
##                Accuracy : 0.8696
##                  95% CI : (0.8122, 0.9146)
##     No Information Rate : 0.5598
##     P-Value [Acc > NIR] : <2e-16
##
```

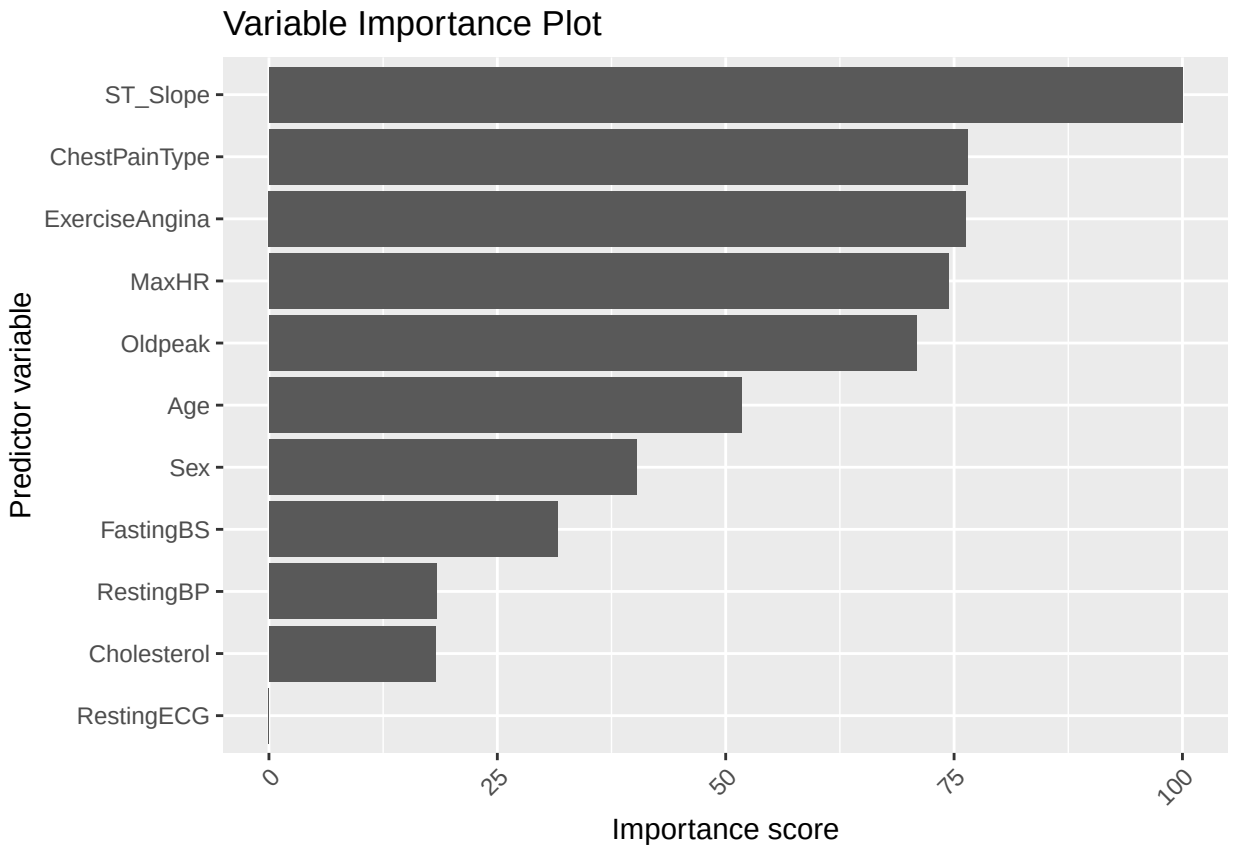
```
##                   Kappa : 0.7339
##
##  Mcnemar's Test P-Value : 0.5403
##
##             Sensitivity : 0.8272
##             Specificity : 0.9029
##          Pos Pred Value : 0.8701
##          Neg Pred Value : 0.8692
##              Prevalence : 0.4402
##          Detection Rate : 0.3641
##    Detection Prevalence : 0.4185
##       Balanced Accuracy : 0.8650
##
##        'Positive' Class : 0
##
```

```r
# Check variable importance
ctrl <- trainControl(method = "cv", number = 5)

nb_model2 <- train(HeartDisease ~ ., data = train, method = "naive_bayes",
                   trControl = ctrl, tuneLength = 10, preProcess = c("center", "scale"))
var_nb <- varImp(nb_model2)
var_nb
```

```
## ROC curve variable importance
##
##                Importance
## ST_Slope           100.00
## ChestPainType       76.46
## ExerciseAngina      76.31
## MaxHR               74.43
## Oldpeak             70.84
## Age                 51.69
## Sex                 40.28
## FastingBS           31.58
## RestingBP           18.37
## Cholesterol         18.27
## RestingECG           0.00
```

```r
# Plot variable importance
ggplot(var_nb, aes(x = reorder(rownames(var_nb), -Overall), y = Overall)) +
  geom_bar(stat = "identity") +
  labs(x = "Predictor variable", y = "Importance score", title = "Variable Importance Plot") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

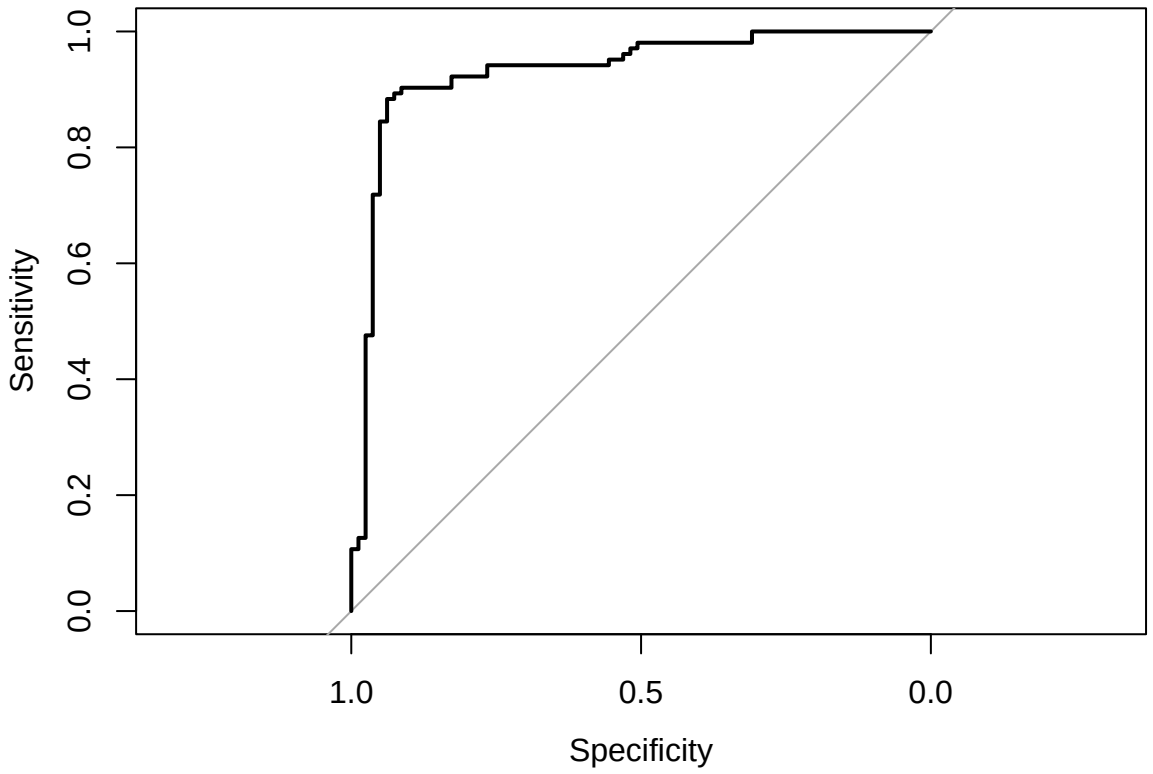



```
# Generate ROC curve and calculate AUC
nb_roc <- roc(test$HeartDisease, predict(nb_model, newdata = test, type = "raw")[,2])
```

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

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

```
plot(nb_roc)
```




```
auc(nb_roc)
```

```
## Area under the curve: 0.9323
```

kNN

```
# Preprocess the data (normalize numerical variables)
preProc <- preProcess(train[, -1], method = c("center", "scale"))
trainDataNorm <- predict(preProc, train[, -1])
testDataNorm <- predict(preProc, test[, -1])
# Train the KNN model
knn_model <- train(HeartDisease ~ ., data = trainDataNorm, method = "knn",
                   trControl = trainControl(method = "cv", number = 5),
tuneGrid = expand.grid(k = seq(1, 20, 1))) # Make predictions on the test set
knn_pred <- predict(knn_model, testDataNorm)
# Evaluate the model
confusionMatrix(knn_pred, test$HeartDisease)
```

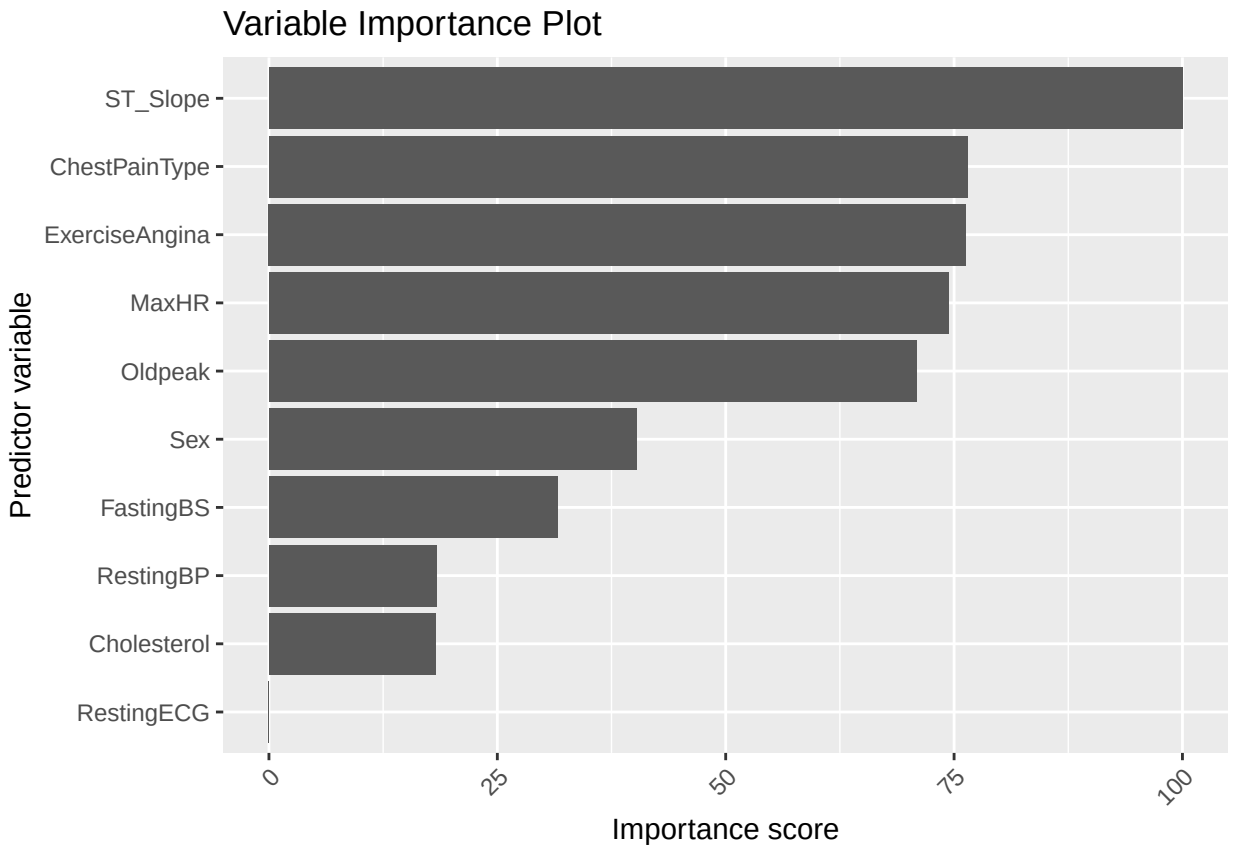
```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  0  1
##          0 70  5
##          1 11 98
##
##                Accuracy : 0.913
##                  95% CI : (0.8626, 0.9495)
##     No Information Rate : 0.5598
##     P-Value [Acc > NIR] : <2e-16
```

```
##
##                   Kappa : 0.8222
##
##  Mcnemar's Test P-Value : 0.2113
##
##             Sensitivity : 0.8642
##             Specificity : 0.9515
##          Pos Pred Value : 0.9333
##          Neg Pred Value : 0.8991
##              Prevalence : 0.4402
##          Detection Rate : 0.3804
##    Detection Prevalence : 0.4076
##       Balanced Accuracy : 0.9078
##
##        'Positive' Class : 0
##
```

```
# Check variable importance
var_knn <- varImp(knn_model)
var_knn
```

```
## ROC curve variable importance
##
##                Importance
## ST_Slope           100.00
## ChestPainType       76.46
## ExerciseAngina      76.31
## MaxHR               74.43
## Oldpeak             70.84
## Sex                 40.28
## FastingBS           31.58
## RestingBP           18.37
## Cholesterol         18.27
## RestingECG           0.00
```

```
# Create a bar plot of variable importance scores
ggplot(var_knn, aes(x = reorder(row.names(var_knn), -Overall), y = Overall)) +
  geom_bar(stat = "identity") +
  labs(x = "Predictor variable", y = "Importance score", title = "Variable Importance Plot")+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

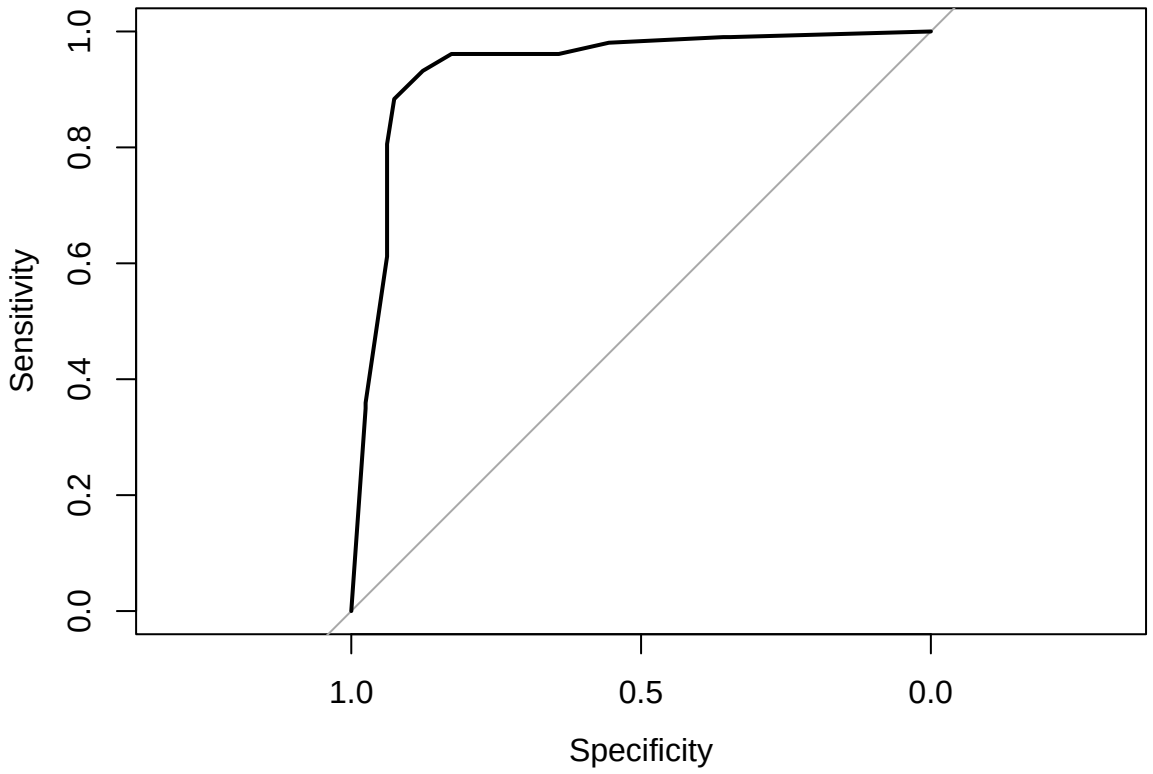



```
# Generate ROC curve and calculate AUC
knn_roc <- roc(test$HeartDisease, predict(knn_model, newdata = testDataNorm, type = "prob")[,2])
```

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

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

```
plot(knn_roc)
```




```
auc(knn_roc)
```

```
## Area under the curve: 0.9371
```