final-projectR.R

gvuma

2025-04-06

```r
# --- Package Installation (run only once) ---
if(!require("readxl")) install.packages("readxl")
```

```
## Loading required package: readxl

## Warning: package 'readxl' was built under R version 4.3.3
```

```r
if(!require("corrplot")) install.packages("corrplot")
```

```
## Loading required package: corrplot

## Warning: package 'corrplot' was built under R version 4.3.3

## corrplot 0.95 loaded
```

```r
if(!require("caret")) install.packages("caret")
```

```
## Loading required package: caret

## Warning: package 'caret' was built under R version 4.3.3

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.3.3

## Loading required package: lattice
```

```r
if(!require("pROC")) install.packages("pROC")
```

```
## Loading required package: pROC

## Warning: package 'pROC' was built under R version 4.3.3

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

##
## Attaching package: 'pROC'

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

```r
if(!require("randomForest")) install.packages("randomForest")
```

```
## Loading required package: randomForest

## Warning: package 'randomForest' was built under R version 4.3.3

## randomForest 4.7-1.2

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

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

```r
if(!require("xgboost")) install.packages("xgboost")
```

```
## Loading required package: xgboost

## Warning: package 'xgboost' was built under R version 4.3.3
```

```r
if(!require("neuralnet")) install.packages("neuralnet")
```

```
## Loading required package: neuralnet

## Warning: package 'neuralnet' was built under R version 4.3.3
```

```r
if(!require("glmnet")) install.packages("glmnet")
```

```
## Loading required package: glmnet

## Warning: package 'glmnet' was built under R version 4.3.3

## Loading required package: Matrix

## Loaded glmnet 4.1-8
```

```r
# --- Load Libraries ---
library(readxl)
library(corrplot)
library(caret)
library(pROC)
library(randomForest)
library(xgboost)
library(neuralnet)
library(glmnet)
```

```r
# --- Step 1: Load the Dataset ---
framingham <- read_excel("C:/Users/gvuma/Downloads/framingham (1).xlsx")

# --- Convert Columns to Numeric ---
# List of columns that should be numeric
numeric_cols <- c("age", "cigsPerDay", "totChol", "sysBP", "diaBP", "BMI", "heartRate", "glucose")
for(col in numeric_cols){
  framingham[[col]] <- as.numeric(as.character(framingham[[col]]))
}
```

```
## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion
```

```r
# Check structure after conversion
str(framingham)
```

```
## tibble [4,239 x 16] (S3: tbl_df/tbl/data.frame)
##  $ Sex           : chr [1:4239] "male" "female" "male" "female" ...
##  $ age           : num [1:4239] 39 46 48 61 46 43 63 45 52 43 ...
##  $ education     : chr [1:4239] "4" "2" "1" "3" ...
##  $ currentSmoker : chr [1:4239] "No" "No" "Yes" "Yes" ...
##  $ cigsPerDay    : num [1:4239] 0 0 20 30 23 0 0 20 0 30 ...
##  $ BPMeds        : chr [1:4239] "0" "0" "0" "0" ...
##  $ prevalentStroke: num [1:4239] 0 0 0 0 0 0 0 0 0 0 ...
##  $ prevalentHyp  : num [1:4239] 0 0 0 1 0 1 0 0 1 1 ...
##  $ diabetes      : chr [1:4239] "No" "No" "No" "No" ...
##  $ totChol       : num [1:4239] 195 250 245 225 285 228 205 313 260 225 ...
##  $ sysBP         : num [1:4239] 106 121 128 150 130 ...
##  $ diaBP         : num [1:4239] 70 81 80 95 84 110 71 71 89 107 ...
##  $ BMI           : num [1:4239] 27 28.7 25.3 28.6 23.1 ...
##  $ heartRate     : num [1:4239] 80 95 75 65 85 77 60 79 76 93 ...
##  $ glucose       : num [1:4239] 77 76 70 103 85 99 85 78 79 88 ...
##  $ TenYearCHD    : num [1:4239] 0 0 0 1 0 0 1 0 0 0 ...
```

```r
# --- Step 2: Data Cleaning ---
# Check for missing values
print(colSums(is.na(framingham)))
```

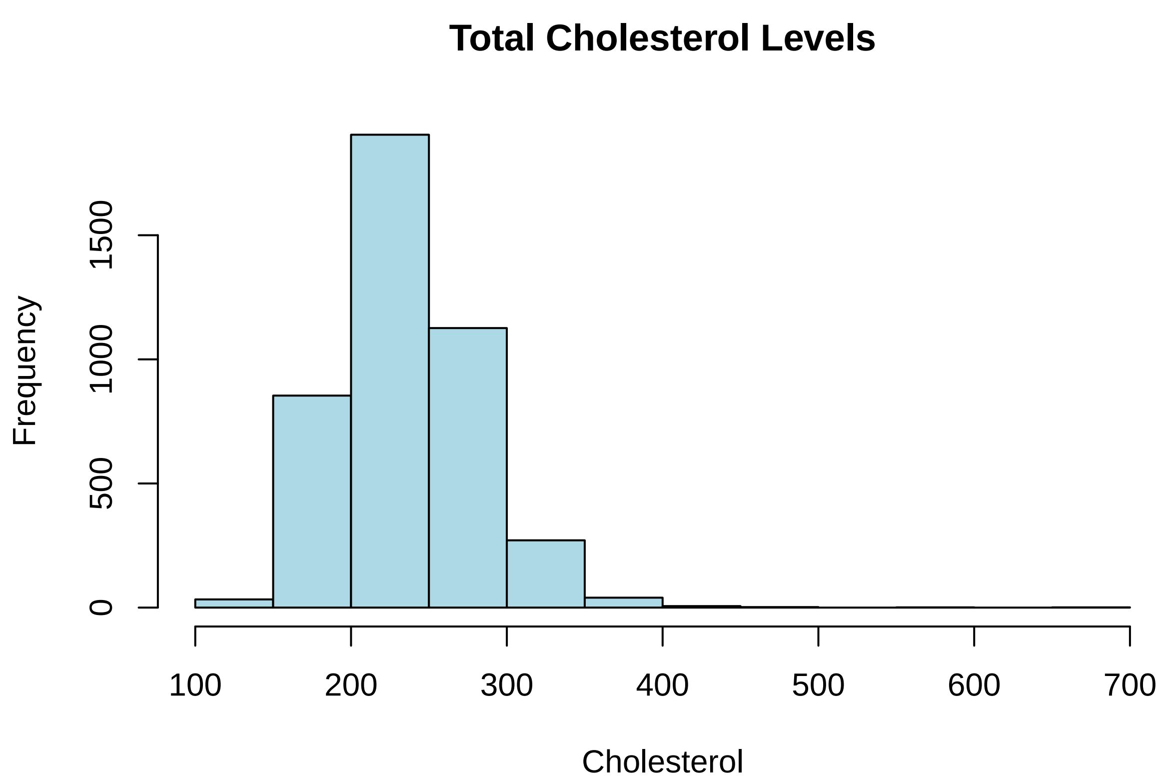
```
##             Sex             age       education   currentSmoker      cigsPerDay
##               0               0               0               0              29
##          BPMeds prevalentStroke    prevalentHyp        diabetes         totChol
##               0               0               0               0              50
##           sysBP           diaBP             BMI       heartRate         glucose
##               0               0              19               0             388
##      TenYearCHD
##               0
```

```r
# Impute missing values for numerical variables with their mean (if any)
for(col in numeric_cols) {
  if(any(is.na(framingham[[col]]))) {
    framingham[[col]][is.na(framingham[[col]])] <- mean(framingham[[col]], na.rm = TRUE)
  }
}

# Define a function for mode imputation for categorical/binary variables
mode_impute <- function(x) {
  unique_x <- unique(x[!is.na(x)])
  unique_x[which.max(tabulate(match(x, unique_x)))]
}

# Impute missing values for binary variables (if any)
if(any(is.na(framingham$currentSmoker))) {
  framingham$currentSmoker[is.na(framingham$currentSmoker)] <- mode_impute(framingham$currentSmoker)
}
if(any(is.na(framingham$diabetes))) {
  framingham$diabetes[is.na(framingham$diabetes)] <- mode_impute(framingham$diabetes)
}

# Convert appropriate columns to factors
framingham$Sex <- as.factor(framingham$Sex)
framingham$currentSmoker <- as.factor(framingham$currentSmoker)
framingham$TenYearCHD <- as.factor(framingham$TenYearCHD)
framingham$diabetes <- as.factor(framingham$diabetes)

# --- Step 3: Exploratory Data Analysis (EDA) ---
# Summary statistics
summary(framingham)
```

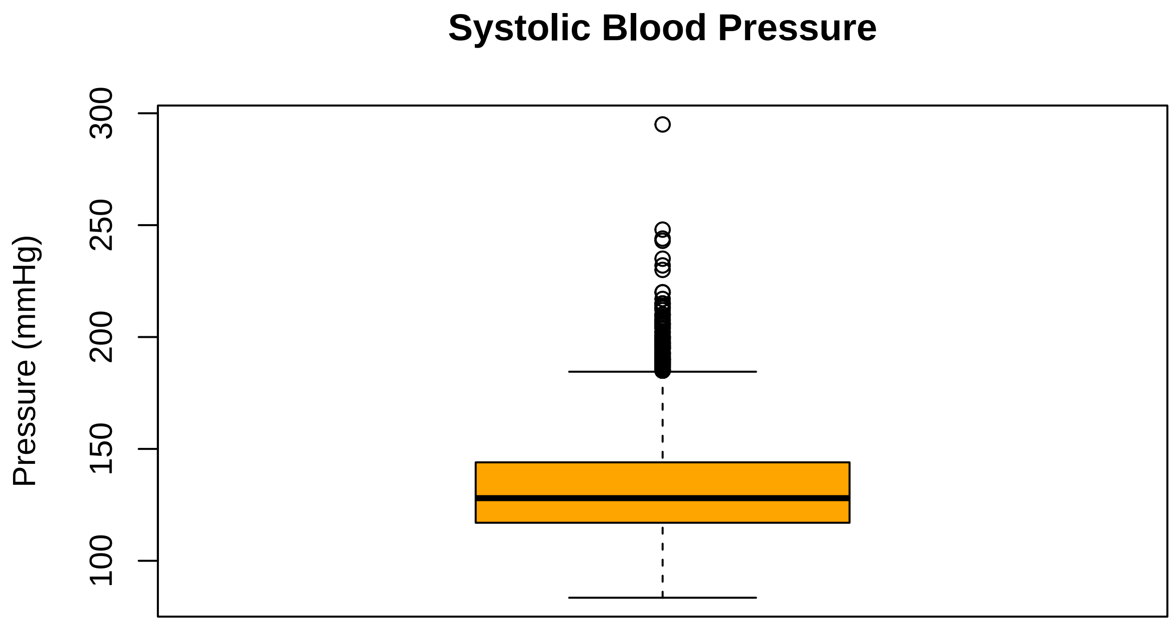
```
##      Sex            age         education         currentSmoker
##  female:2420   Min.   :32.00   Length:4239        No :2145
##  male  :1819   1st Qu.:42.00   Class :character   Yes:2094
##                Median :49.00   Mode  :character
##                Mean   :49.58
##                3rd Qu.:56.00
##                Max.   :70.00
##    cigsPerDay        BPMeds          prevalentStroke     prevalentHyp
##  Min.   : 0.000   Length:4239        Min.   :0.000000   Min.   :0.0000
##  1st Qu.: 0.000   Class :character   1st Qu.:0.000000   1st Qu.:0.0000
##  Median : 0.000   Mode  :character   Median :0.000000   Median :0.0000
##  Mean   : 9.004                      Mean   :0.005898   Mean   :0.3105
##  3rd Qu.:20.000                      3rd Qu.:0.000000   3rd Qu.:1.0000
##  Max.   :70.000                      Max.   :1.000000   Max.   :1.0000
##  diabetes      totChol          sysBP           diaBP            BMI
##  No :4130   Min.   :107.0   Min.   : 83.5   Min.   : 48.0   Min.   :15.54
##  Yes: 109   1st Qu.:206.0   1st Qu.:117.0   1st Qu.: 75.0   1st Qu.:23.07
##             Median :234.0   Median :128.0   Median : 82.0   Median :25.41
##             Mean   :236.7   Mean   :132.3   Mean   : 82.9   Mean   :25.80
##             3rd Qu.:262.0   3rd Qu.:144.0   3rd Qu.: 90.0   3rd Qu.:28.02
##             Max.   :696.0   Max.   :295.0   Max.   :142.5   Max.   :56.80
##    heartRate         glucose       TenYearCHD
```

```
##  Min.   : 44.00   Min.   : 40.00   0:3596
##  1st Qu.: 68.00   1st Qu.: 72.00   1: 643
##  Median : 75.00   Median : 80.00
##  Mean   : 75.88   Mean   : 81.96
##  3rd Qu.: 83.00   3rd Qu.: 85.00
##  Max.   :143.00   Max.   :394.00
```

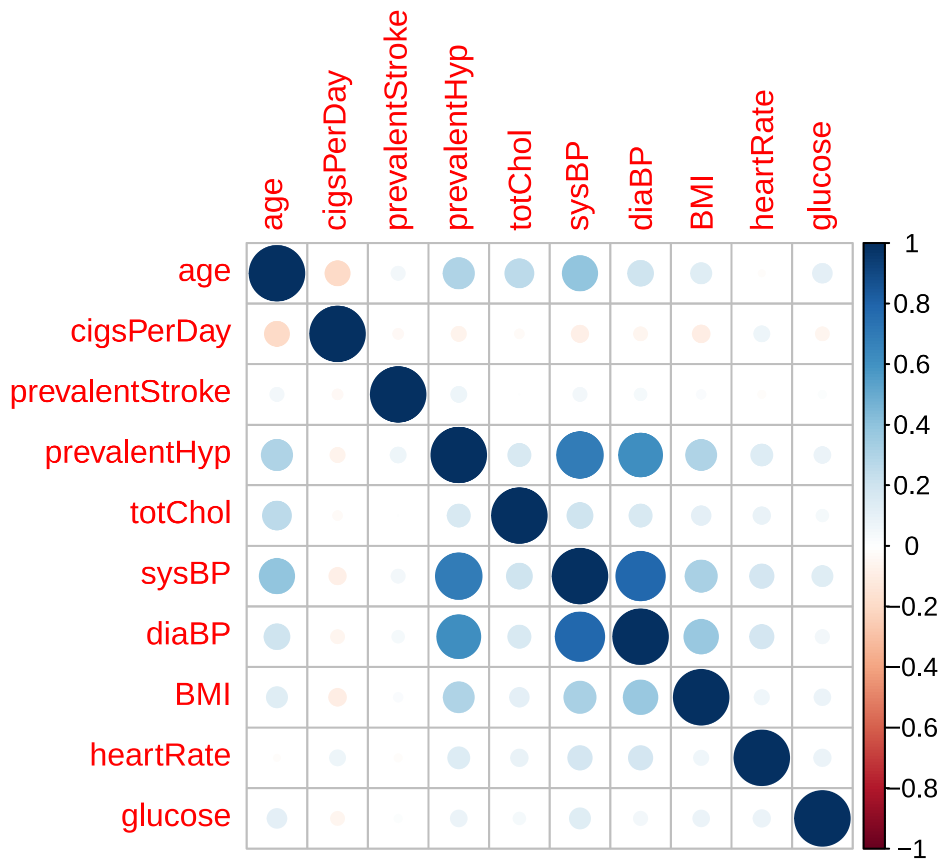
```r
# Histogram for total cholesterol
hist(framingham$totChol, main = "Total Cholesterol Levels",
     xlab = "Cholesterol", col = "lightblue")
```




```r
# Boxplot for systolic blood pressure
boxplot(framingham$sysBP, main = "Systolic Blood Pressure",
        ylab = "Pressure (mmHg)", col = "orange")
```




```
# Correlation matrix for numerical variables
numeric_vars <- framingham[, sapply(framingham, is.numeric)]
corrplot(cor(numeric_vars, use = "complete.obs"), method = "circle")
```




```r
# Class distribution of TenYearCHD
print(table(framingham$TenYearCHD))
```

```
##
##    0    1
## 3596  643
```

```r
# --- Step 4: Logistic Regression ---
# Fit the logistic regression model
model <- glm(TenYearCHD ~ age + totChol + sysBP + diaBP + BMI + glucose + currentSmoker + diabetes + Se
             data = framingham, family = binomial)

# Summary of the model
summary(model)
```

```
##
## Call:
## glm(formula = TenYearCHD ~ age + totChol + sysBP + diaBP + BMI +
##     glucose + currentSmoker + diabetes + Sex, family = binomial,
##     data = framingham)
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -8.797115   0.541460 -16.247  < 2e-16 ***
## age               0.061505   0.006067  10.137  < 2e-16 ***
```

```
## totChol           0.001949   0.001019   1.912  0.05585 .
## sysBP             0.017291   0.003244   5.330 9.81e-08 ***
## diaBP            -0.001416   0.005912  -0.239  0.81072
## BMI               0.006968   0.011655   0.598  0.54992
## glucose           0.006396   0.002128   3.006  0.00265 **
## currentSmokerYes  0.383265   0.096965   3.953 7.73e-05 ***
## diabetesYes       0.200011   0.293012   0.683  0.49486
## Sexmale           0.592760   0.095654   6.197 5.76e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3608.4  on 4238  degrees of freedom
## Residual deviance: 3232.5  on 4229  degrees of freedom
## AIC: 3252.5
##
## Number of Fisher Scoring iterations: 5
```

```r
# Calculate and print odds ratios
print(exp(coef(model)))
```

```
##      (Intercept)              age          totChol            sysBP
##     0.0001511686     1.0634355744     1.0019510770     1.0174415067
##            diaBP              BMI          glucose currentSmokerYes
##     0.9985851879     1.0069925699     1.0064163081     1.4670670208
##      diabetesYes          Sexmale
##     1.2214163620     1.8089734796
```

```r
# Add predicted probabilities to the dataset
framingham$predicted_risk <- predict(model, framingham, type = "response")

# --- Step 5: Model Evaluation ---
# Create a binary classification (cutoff = 0.5)
framingham$predicted_class <- ifelse(framingham$predicted_risk > 0.5, 1, 0)

# Confusion matrix
confusionMatrix(as.factor(framingham$predicted_class), framingham$TenYearCHD)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3574  605
##          1   22   38
##
##                Accuracy : 0.8521
##                  95% CI : (0.841, 0.8626)
##     No Information Rate : 0.8483
##     P-Value [Acc > NIR] : 0.2543
##
##                   Kappa : 0.0844
##
```

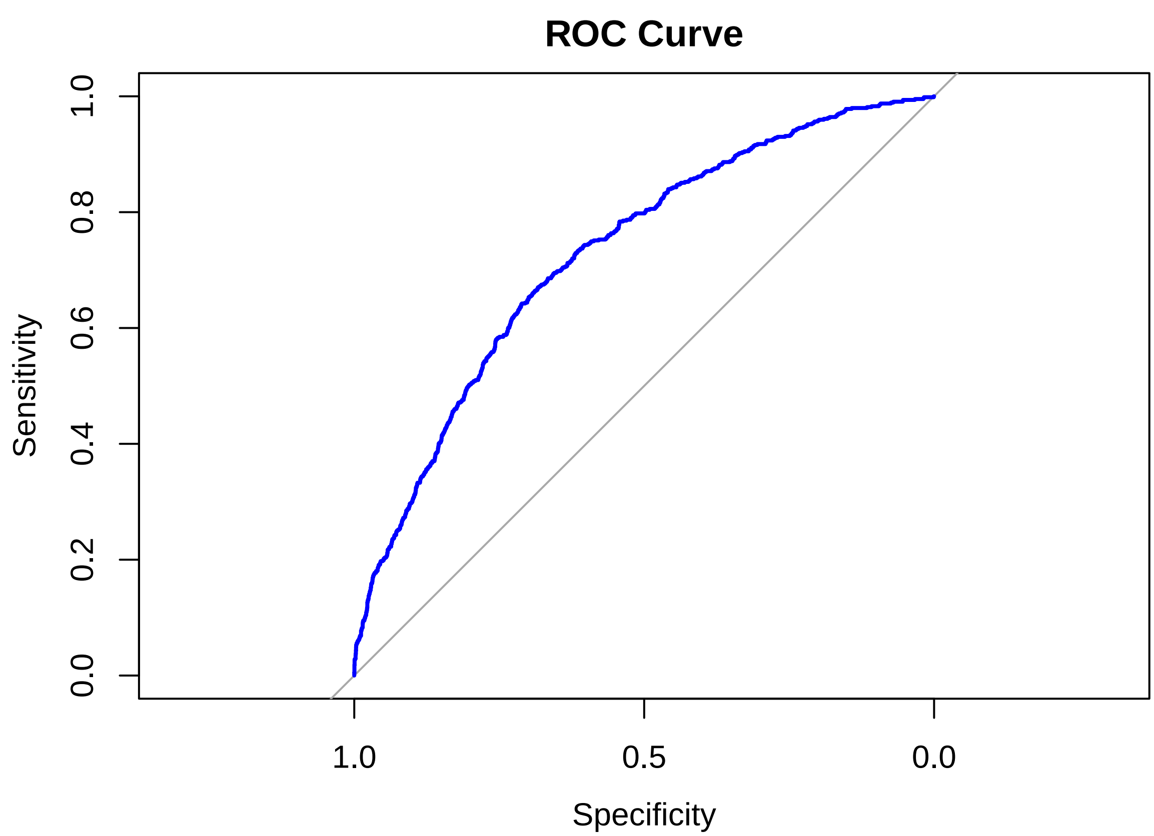
```
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9939
##             Specificity : 0.0591
##          Pos Pred Value : 0.8552
##          Neg Pred Value : 0.6333
##              Prevalence : 0.8483
##          Detection Rate : 0.8431
##    Detection Prevalence : 0.9858
##       Balanced Accuracy : 0.5265
##
##        'Positive' Class : 0
##
```

```r
# ROC Curve and AUC
roc_curve <- roc(framingham$TenYearCHD, framingham$predicted_risk)
```

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

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

```r
plot(roc_curve, main = "ROC Curve", col = "blue")
```



```r
print(auc(roc_curve))
```

```
## Area under the curve: 0.727
```

```r
# --- Step 6: Logistic Regression with Regularization (Ridge) ---
x <- model.matrix(TenYearCHD ~ age + totChol + sysBP + diaBP + BMI + glucose + currentSmoker + diabetes
y <- framingham$TenYearCHD

# Convert the factor variable TenYearCHD to numeric (0 or 1)
y <- as.numeric(framingham$TenYearCHD) - 1  # Convert factor levels to 0 and 1

# Fit the ridge regression model
ridge_model <- cv.glmnet(x, y, alpha = 0)  # L2 Regularization (Ridge)

# Summary of the ridge model
print(ridge_model)
```

```
##
## Call:  cv.glmnet(x = x, y = y, alpha = 0)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure       SE Nonzero
## min  0.017    92  0.1178 0.004716       9
## 1se  1.116    47  0.1224 0.004820       9
```

```r
# Predict using the ridge model
framingham$ridge_predicted_risk <- predict(ridge_model, s = "lambda.min", newx = x, type = "response")

# --- Step 7: Random Forest Model ---
rf_model <- randomForest(TenYearCHD ~ age + totChol + sysBP + diaBP + BMI + glucose + currentSmoker + di
                         data = framingham, ntree = 500)

# Predict using Random Forest
framingham$rf_predicted_risk <- predict(rf_model, framingham, type = "response")

# --- Step 8: XGBoost Model ---
# Convert categorical variables to numeric (one-hot encoding or factor conversion)
train_data <- model.matrix(~ age + totChol + sysBP + diaBP + BMI + glucose + currentSmoker + diabetes +

# Convert the target variable (TenYearCHD) to numeric (0/1)
train_label <- as.numeric(framingham$TenYearCHD) - 1  # Convert factor levels to 0 and 1

# Train the XGBoost model
xgb_model <- xgboost(data = train_data, label = train_label, nrounds = 100, objective = "binary:logisti
```

```
## [1]  train-logloss:0.551835
## [2]  train-logloss:0.473143
## [3]  train-logloss:0.424738
## [4]  train-logloss:0.392471
## [5]  train-logloss:0.370043
## [6]  train-logloss:0.352434
```

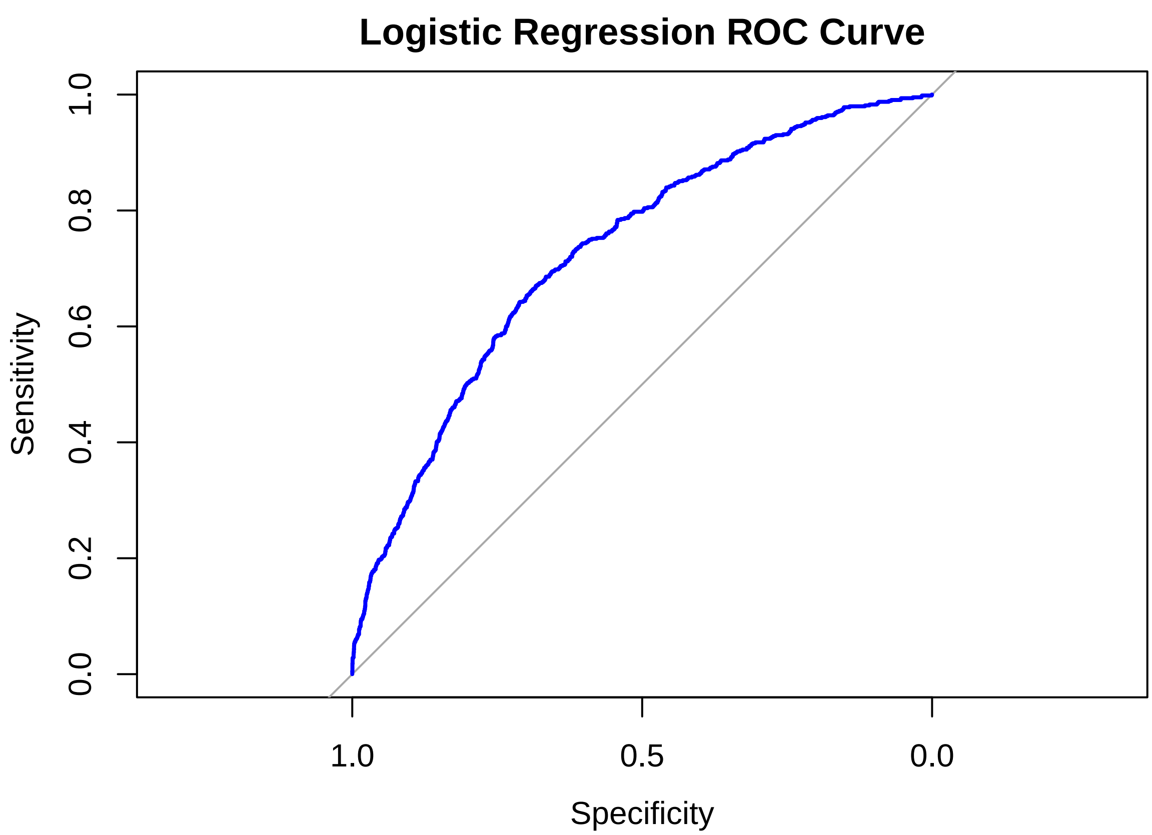
```
## [7]  train-logloss:0.339309
## [8]  train-logloss:0.329752
## [9]  train-logloss:0.321346
## [10] train-logloss:0.314769
## [11] train-logloss:0.309320
## [12] train-logloss:0.303852
## [13] train-logloss:0.298626
## [14] train-logloss:0.295827
## [15] train-logloss:0.293071
## [16] train-logloss:0.284242
## [17] train-logloss:0.281899
## [18] train-logloss:0.276878
## [19] train-logloss:0.273748
## [20] train-logloss:0.270835
## [21] train-logloss:0.266210
## [22] train-logloss:0.263529
## [23] train-logloss:0.260947
## [24] train-logloss:0.260152
## [25] train-logloss:0.258204
## [26] train-logloss:0.257113
## [27] train-logloss:0.256573
## [28] train-logloss:0.252768
## [29] train-logloss:0.249863
## [30] train-logloss:0.249122
## [31] train-logloss:0.247527
## [32] train-logloss:0.245281
## [33] train-logloss:0.240369
## [34] train-logloss:0.235007
## [35] train-logloss:0.234087
## [36] train-logloss:0.232755
## [37] train-logloss:0.229551
## [38] train-logloss:0.228901
## [39] train-logloss:0.227943
## [40] train-logloss:0.224786
## [41] train-logloss:0.222334
## [42] train-logloss:0.221869
## [43] train-logloss:0.221677
## [44] train-logloss:0.219558
## [45] train-logloss:0.217647
## [46] train-logloss:0.217040
## [47] train-logloss:0.213589
## [48] train-logloss:0.210949
## [49] train-logloss:0.209342
## [50] train-logloss:0.207745
## [51] train-logloss:0.205860
## [52] train-logloss:0.203570
## [53] train-logloss:0.199510
## [54] train-logloss:0.197728
## [55] train-logloss:0.195125
## [56] train-logloss:0.194376
## [57] train-logloss:0.191637
## [58] train-logloss:0.188408
## [59] train-logloss:0.185073
## [60] train-logloss:0.182178
```

```
## [61] train-logloss:0.179451
## [62] train-logloss:0.177468
## [63] train-logloss:0.174612
## [64] train-logloss:0.174002
## [65] train-logloss:0.172901
## [66] train-logloss:0.170891
## [67] train-logloss:0.166604
## [68] train-logloss:0.163524
## [69] train-logloss:0.159594
## [70] train-logloss:0.157515
## [71] train-logloss:0.156579
## [72] train-logloss:0.156100
## [73] train-logloss:0.155353
## [74] train-logloss:0.152680
## [75] train-logloss:0.149707
## [76] train-logloss:0.148745
## [77] train-logloss:0.145975
## [78] train-logloss:0.145589
## [79] train-logloss:0.144988
## [80] train-logloss:0.144438
## [81] train-logloss:0.143965
## [82] train-logloss:0.141862
## [83] train-logloss:0.139668
## [84] train-logloss:0.137450
## [85] train-logloss:0.136348
## [86] train-logloss:0.135037
## [87] train-logloss:0.133045
## [88] train-logloss:0.131601
## [89] train-logloss:0.129811
## [90] train-logloss:0.128034
## [91] train-logloss:0.126738
## [92] train-logloss:0.125755
## [93] train-logloss:0.124802
## [94] train-logloss:0.124480
## [95] train-logloss:0.123874
## [96] train-logloss:0.120621
## [97] train-logloss:0.118960
## [98] train-logloss:0.118703
## [99] train-logloss:0.118253
## [100]    train-logloss:0.117161
```

```r
# Predict using XGBoost
framingham$xgb_predicted_risk <- predict(xgb_model, train_data)



# --- Step 10: Model Evaluation ---
# ROC and AUC for each model

# Logistic Regression ROC
roc_curve_log_reg <- roc(framingham$TenYearCHD, framingham$predicted_risk)
```

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

```r
plot(roc_curve_log_reg, main = "Logistic Regression ROC Curve", col = "blue")
```




```r
print(auc(roc_curve_log_reg))
```

```
## Area under the curve: 0.727
```

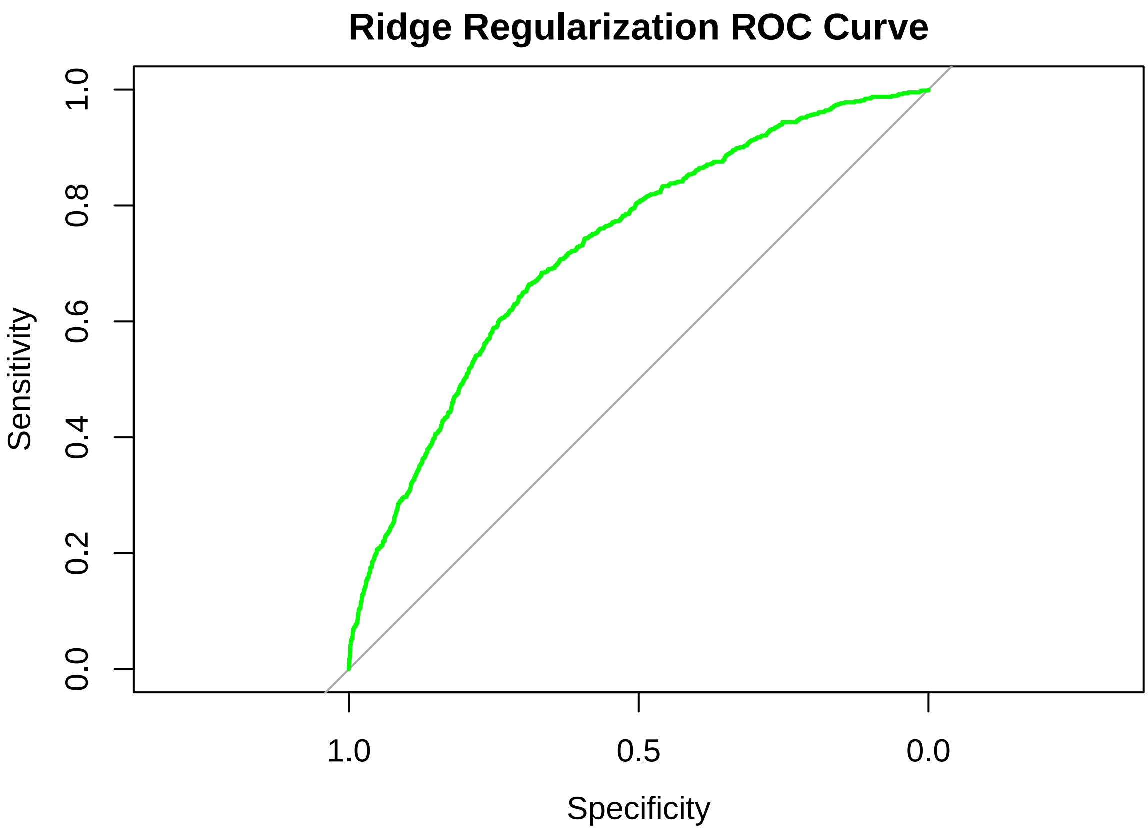
```r
# Ridge Regularization ROC
roc_curve_ridge <- roc(framingham$TenYearCHD, framingham$ridge_predicted_risk)
```

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

```
## Warning in roc.default(framingham$TenYearCHD, framingham$ridge_predicted_risk):
## Deprecated use a matrix as predictor. Unexpected results may be produced,
## please pass a numeric vector.
```

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

```r
plot(roc_curve_ridge, main = "Ridge Regularization ROC Curve", col = "green")
```



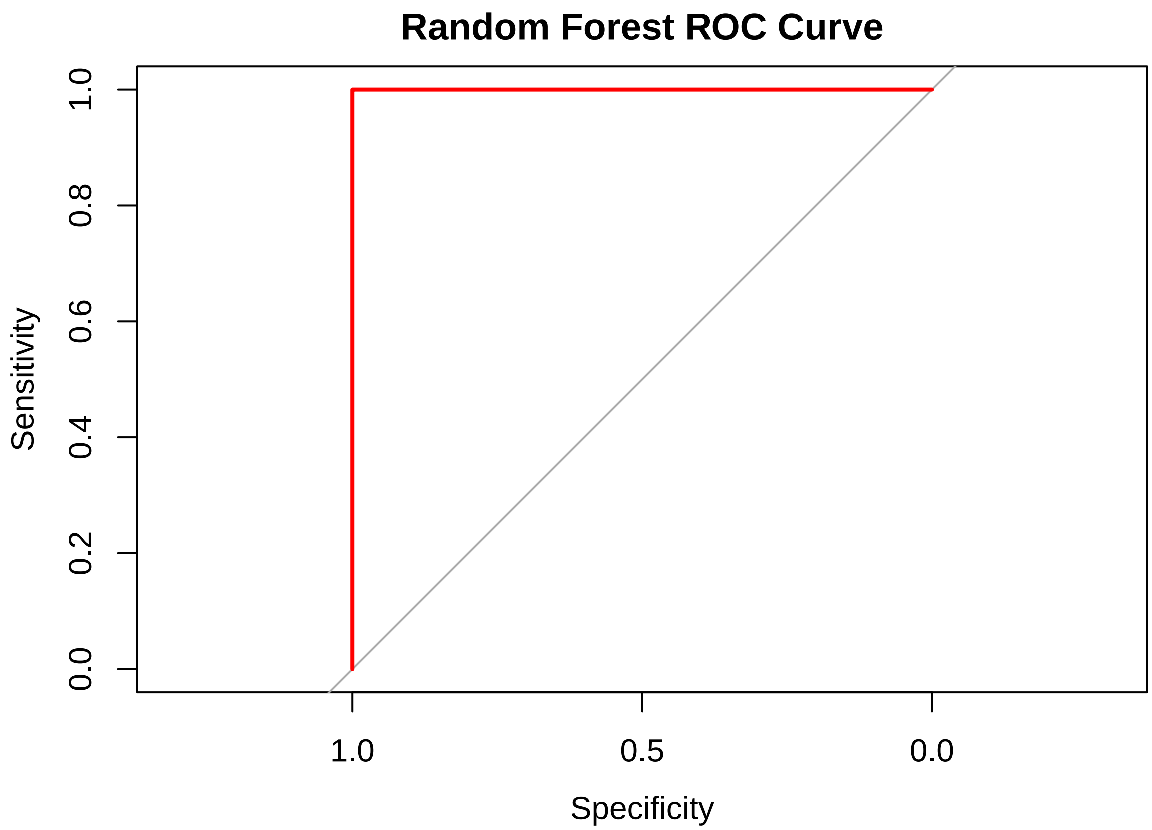

```r
print(auc(roc_curve_ridge))
```

```
## Area under the curve: 0.7255
```

```r
# Random Forest ROC
# Assuming your random forest model is trained and you need the probability predictions:
rf_pred_probs <- predict(rf_model, framingham, type = "prob")[, 2]  # Assuming '1' is the positive clas

# Now, plot the ROC for Random Forest
roc_curve_rf <- roc(framingham$TenYearCHD, rf_pred_probs)
```

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

```r
plot(roc_curve_rf, main = "Random Forest ROC Curve", col = "red")
```



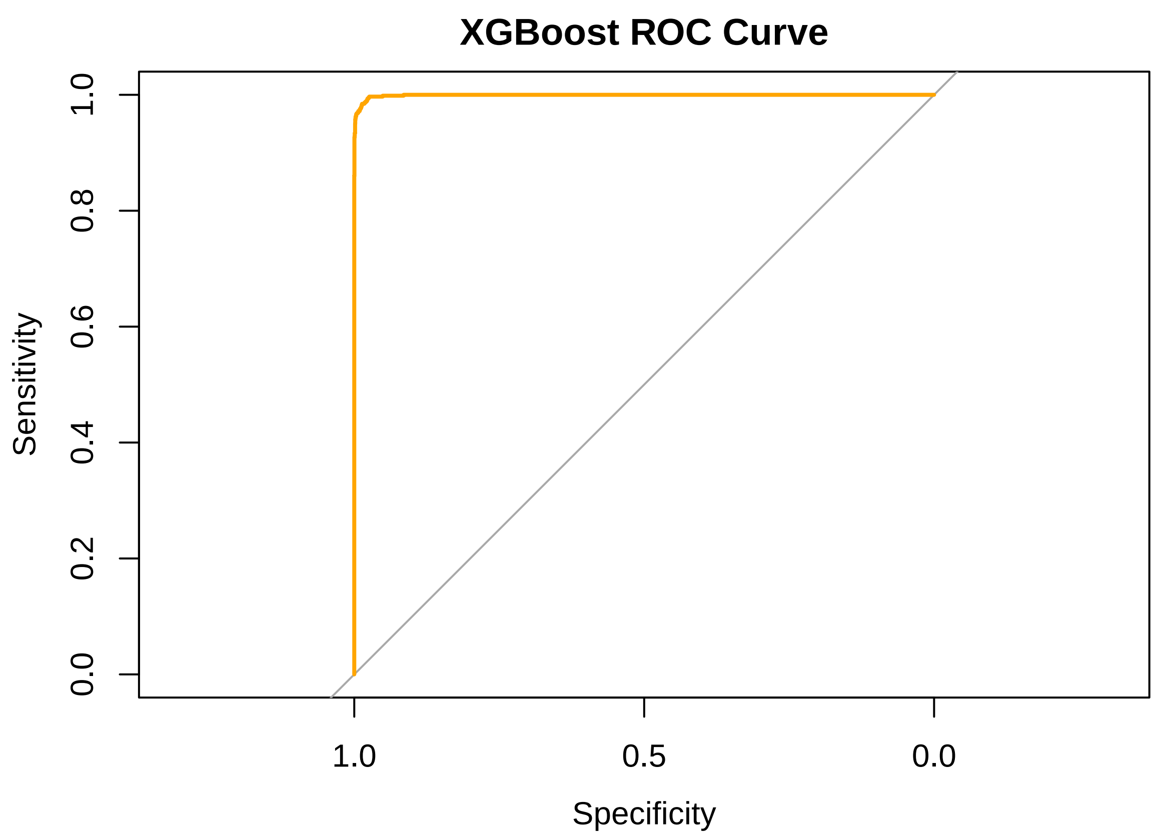

```r
print(auc(roc_curve_rf))
```

```
## Area under the curve: 1
```

```r
# XGBoost ROC
roc_curve_xgb <- roc(framingham$TenYearCHD, framingham$xgb_predicted_risk)
```

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

```r
plot(roc_curve_xgb, main = "XGBoost ROC Curve", col = "orange")
```




```
print(auc(roc_curve_xgb))
```

```
## Area under the curve: 0.9993
```

```
# --- Step 11: Save All Models ---
saveRDS(model, "logistic_model.rds")
saveRDS(ridge_model, "ridge_model.rds")
saveRDS(rf_model, "rf_model.rds")
saveRDS(xgb_model, "xgb_model.rds")
```