Stats project proposal

Sruthi Srikantan

2023-11-12

1200 0's and 811 1's

```
##
##    0    1
## 1200  811
```

Split data into train/test set

```
set.seed(123457)

train.prop <- 0.80
strats <- water_1$Potability
rr <- split(1:length(strats), strats)
idx <- sort(as.numeric(unlist(sapply(rr,
        function(x) sample(x, length(x)*train.prop)))))
water_1.train <- water_1[idx, ]
water_1.test <- water_1[-idx, ]

#see whether the proportions of the two levels of the response are the same for train, test, and entire
summary(water_1.train$Potability)/nrow(water_1.train)
```

```
##         0         1
## 0.5970149 0.4029851
```

```
summary(water_1.test$Potability)/nrow(water_1.test)
```

```
##         0         1
## 0.5955335 0.4044665
```

```
summary(water_1$Potability)/nrow(water_1)
```

```
##         0         1
## 0.5967181 0.4032819
```

Fit full, null, and reduced models

```
full.logit <- glm(Potability ~ Chloramines+Sulfate+Organic_carbon+Trihalomethanes, data = water_1.train
                  family = binomial(link = "logit"))
summary(full.logit)
```

```
##
## Call:
## glm(formula = Potability ~ Chloramines + Sulfate + Organic_carbon +
##     Trihalomethanes, family = binomial(link = "logit"), data = water_1.train)
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      0.0562508  0.5597546   0.100    0.920
## Chloramines      0.0232666  0.0316821   0.734    0.463
## Sulfate         -0.0014659  0.0012330  -1.189    0.234
## Organic_carbon  -0.0107948  0.0151829  -0.711    0.477
## Trihalomethanes  0.0004174  0.0031521   0.132    0.895
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2168.2  on 1607  degrees of freedom
## Residual deviance: 2165.7  on 1603  degrees of freedom
## AIC: 2175.7
##
## Number of Fisher Scoring iterations: 4
```

```
#null model
null.logit <- glm(Potability ~ 1, data = water_1.train,
                  family = binomial(link = "logit"))
summary(null.logit)
```

```
##
## Call:
## glm(formula = Potability ~ 1, family = binomial(link = "logit"),
##     data = water_1.train)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.39304    0.05084  -7.731 1.07e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2168.2  on 1607  degrees of freedom
## Residual deviance: 2168.2  on 1607  degrees of freedom
## AIC: 2170.2
##
## Number of Fisher Scoring iterations: 4
```

```
red.logit<-glm(Potability ~ Chloramines+Trihalomethanes, data = water_1.train,
                  family = binomial(link = "logit"))
summary(red.logit)
```

```
##
## Call:
## glm(formula = Potability ~ Chloramines + Trihalomethanes, family = binomial(link = "logit"),
##     data = water_1.train)
```

```
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -0.5987051  0.3098081  -1.933   0.0533 .
## Chloramines      0.0234506  0.0316753   0.740   0.4591
## Trihalomethanes  0.0005803  0.0031470   0.184   0.8537
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2168.2  on 1607  degrees of freedom
## Residual deviance: 2167.7  on 1605  degrees of freedom
## AIC: 2173.7
##
## Number of Fisher Scoring iterations: 4
```

```
reduction_full <- red.logit$deviance - full.logit$deviance
reduction_full # since the reduction of 1.98 deviance is low, this suggests that chloramines + Trihalom
```

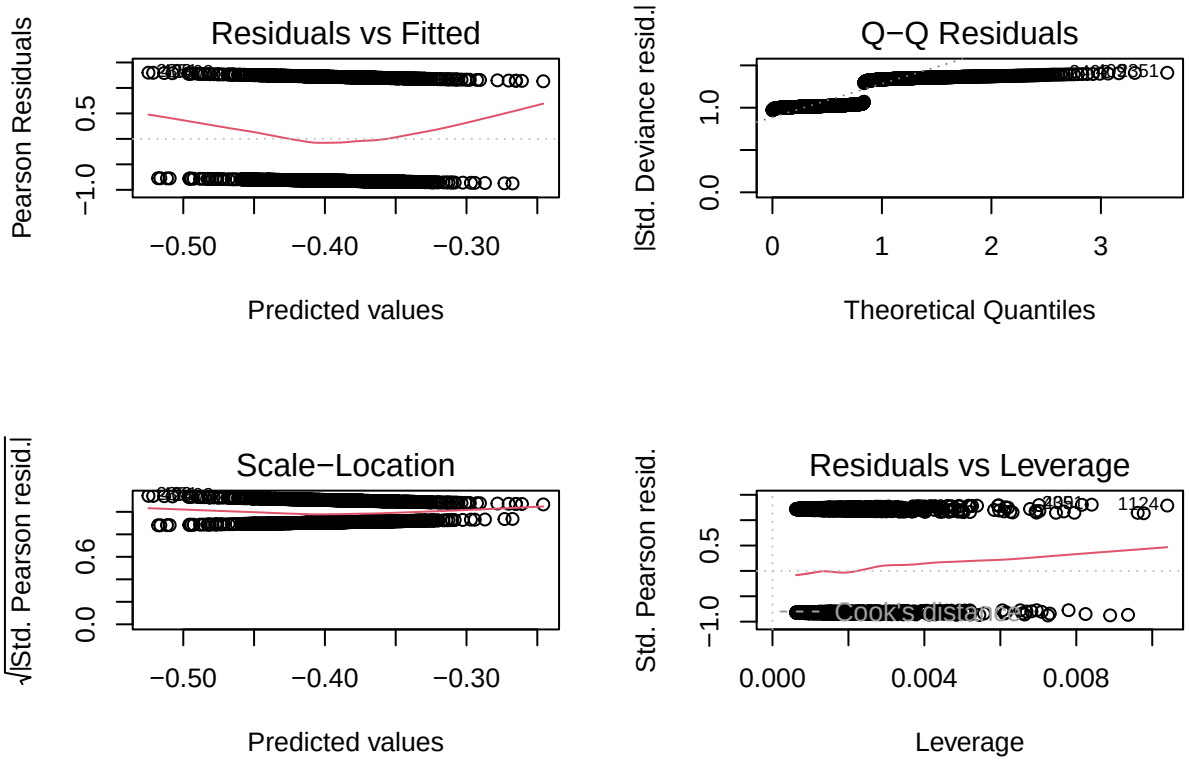
```
## [1] 1.980933
```

Checking for outliers

```
library(MASS)
extpts_red <- which(abs(studres(red.logit)) > 3)
extpts_red # no outliers for studentized residuals
```

```
## named integer(0)
```

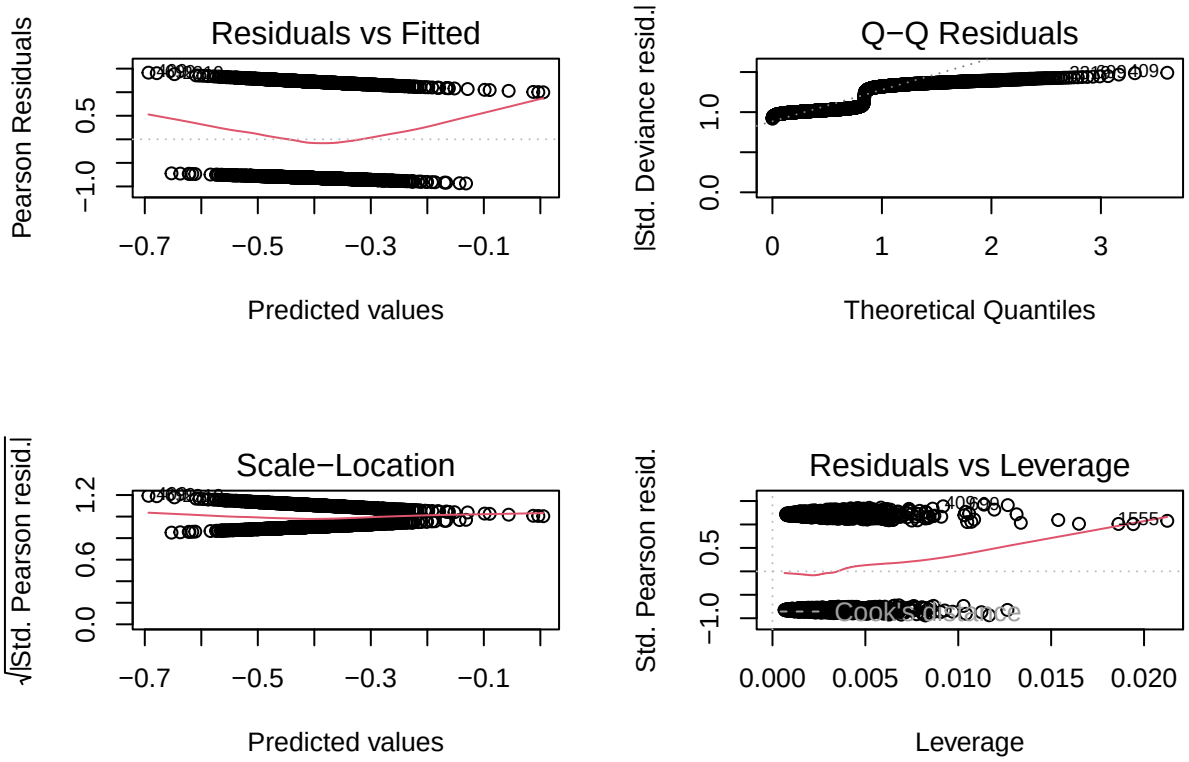
```
#Checking model fit
par(mfrow = c(2, 2))
plot(red.logit) # residuals seem to be evenly distributed around 0 and data follows normality assumptio
```




```
extpts_full <- which(abs(studres(full.logit)) > 3)# no outliers for studentized residuals
extpts_full
```

```
## named integer(0)
```

```
par(mfrow = c(2, 2))
plot(full.logit)# residuals seem to be evenly distributed around 0 and data follows normality assumptio
```



Assess test data accuracy

```
library(pROC)
```

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

##
## Attaching package: 'pROC'

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

```
#ROC curve reduced model
pred.red <- predict(red.logit, newdata = water_1.test, type="response")
pred.full <- predict(full.logit, newdata = water_1.test, type="response")

roc.red <- roc(water_1.test$Potability ~ pred.red, print.auc=T, algorithm=2)
```

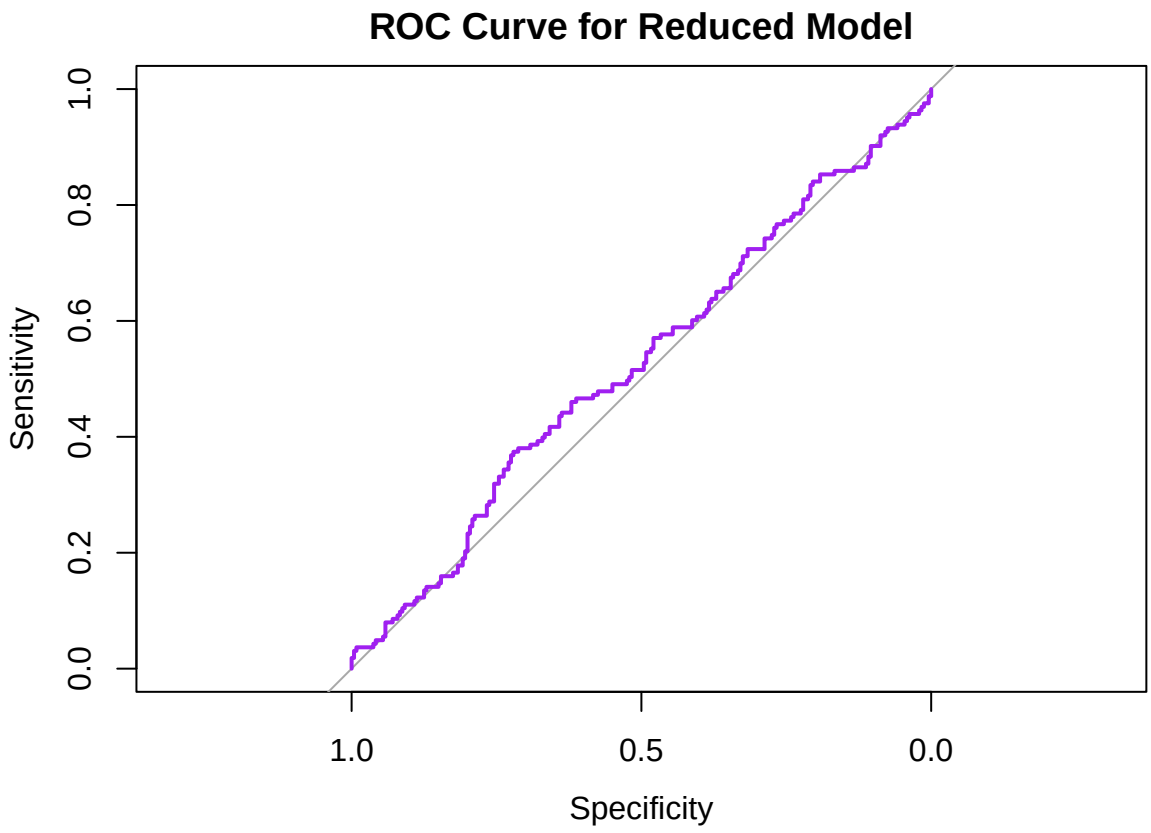
```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```r
auc(roc.red)
```

```
## Area under the curve: 0.525
```

```r
plot(roc.red, main = "ROC Curve for Reduced Model", col = "purple", lwd = 2)
```




```r
#ROC curve full model
roc.full <- roc(water_1.test$Potability ~ pred.full, print.auc=T, algorithm=2)
```

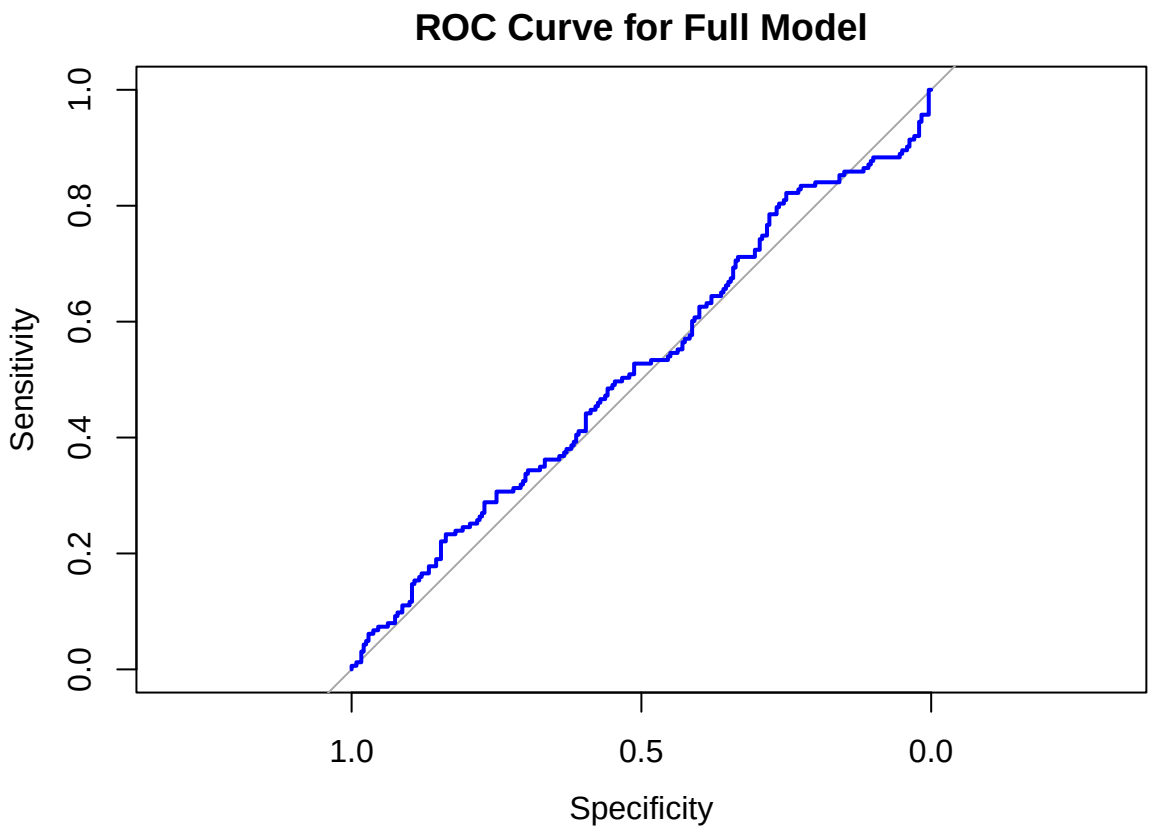
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls > cases
```

```r
auc(roc.full)
```

```
## Area under the curve: 0.5187
```

```r
plot(roc.full, main = "ROC Curve for Full Model", col = "blue", lwd = 2)
```



Confusion Matrix

```
library(caret)
```

```
## Loading required package: ggplot2
```

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

```
f <- ifelse(pred.full > 0.65,1,0)
(cm.full <- confusionMatrix(reference=as.factor(water_1.test$Potability),
            data=as.factor(f), mode="everything"))
```

```
## Warning in confusionMatrix.default(reference =
## as.factor(water_1.test$Potability), : Levels are not in the same order for
## reference and data. Refactoring data to match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 240 163
##          1   0   0
##
##                Accuracy : 0.5955
##                  95% CI : (0.5458, 0.6438)
```

```
##     No Information Rate : 0.5955
##     P-Value [Acc > NIR] : 0.5215
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.5955
##          Neg Pred Value :    NaN
##               Precision : 0.5955
##                  Recall : 1.0000
##                      F1 : 0.7465
##              Prevalence : 0.5955
##          Detection Rate : 0.5955
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 0
##
```

```
b <- ifelse(pred.red > 0.65,1,0)
(cm.red <- confusionMatrix(reference=as.factor(water_1.test$Potability),
            data=as.factor(b), mode="everything"))
```

```
## Warning in confusionMatrix.default(reference =
## as.factor(water_1.test$Potability), : Levels are not in the same order for
## reference and data. Refactoring data to match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 240 163
##          1   0   0
##
##                Accuracy : 0.5955
##                  95% CI : (0.5458, 0.6438)
##     No Information Rate : 0.5955
##     P-Value [Acc > NIR] : 0.5215
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.5955
##          Neg Pred Value :    NaN
##               Precision : 0.5955
##                  Recall : 1.0000
##                      F1 : 0.7465
```

```
##              Prevalence : 0.5955
##          Detection Rate : 0.5955
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 0
##
```

Check for Multicolinearity

```
library(car)
```

```
## Loading required package: carData
```

```
vif(full.logit)# no multicollineaity, values not between 5 and 10
```

```
##     Chloramines         Sulfate  Organic_carbon Trihalomethanes
##        1.000435        1.002441        1.001509        1.001598
```

```
vif(red.logit)# no multicollineaity, values not between 5 and 10
```

```
##     Chloramines Trihalomethanes
##        1.000209        1.000209
```

Quadratic Model

```
quadratic_model_full <- glm(Potability ~ Chloramines+Sulfate+Organic_carbon+Trihalomethanes + I(Trihalo
summary(quadratic_model_full)
```

```
##
## Call:
## glm(formula = Potability ~ Chloramines + Sulfate + Organic_carbon +
##     Trihalomethanes + I(Trihalomethanes^2), family = binomial,
##     data = water_1.train)
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)           1.886e-01  7.895e-01   0.239    0.811
## Chloramines           2.318e-02  3.169e-02   0.731    0.464
## Sulfate              -1.471e-03  1.233e-03  -1.193    0.233
## Organic_carbon       -1.089e-02  1.519e-02  -0.717    0.474
## Trihalomethanes      -3.736e-03  1.776e-02  -0.210    0.833
## I(Trihalomethanes^2)  3.152e-05  1.327e-04   0.238    0.812
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2168.2  on 1607  degrees of freedom
## Residual deviance: 2165.6  on 1602  degrees of freedom
## AIC: 2177.6
##
## Number of Fisher Scoring iterations: 4
```

```
quadratic_model_red <- glm(Potability ~ Chloramines+ Trihalomethanes + I(Trihalomethanes^2), family = bi
summary(quadratic_model_red)
```

```
##
## Call:
## glm(formula = Potability ~ Chloramines + Trihalomethanes + I(Trihalomethanes^2),
##     family = binomial, data = water_1.train)
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -4.921e-01  6.253e-01  -0.787    0.431
## Chloramines           2.338e-02  3.168e-02   0.738    0.460
## Trihalomethanes      -2.847e-03  1.775e-02  -0.160    0.873
## I(Trihalomethanes^2)  2.601e-05  1.326e-04   0.196    0.844
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2168.2  on 1607  degrees of freedom
## Residual deviance: 2167.6  on 1604  degrees of freedom
## AIC: 2175.6
##
## Number of Fisher Scoring iterations: 4
```

```
anova(quadratic_model_full, full.logit)
```

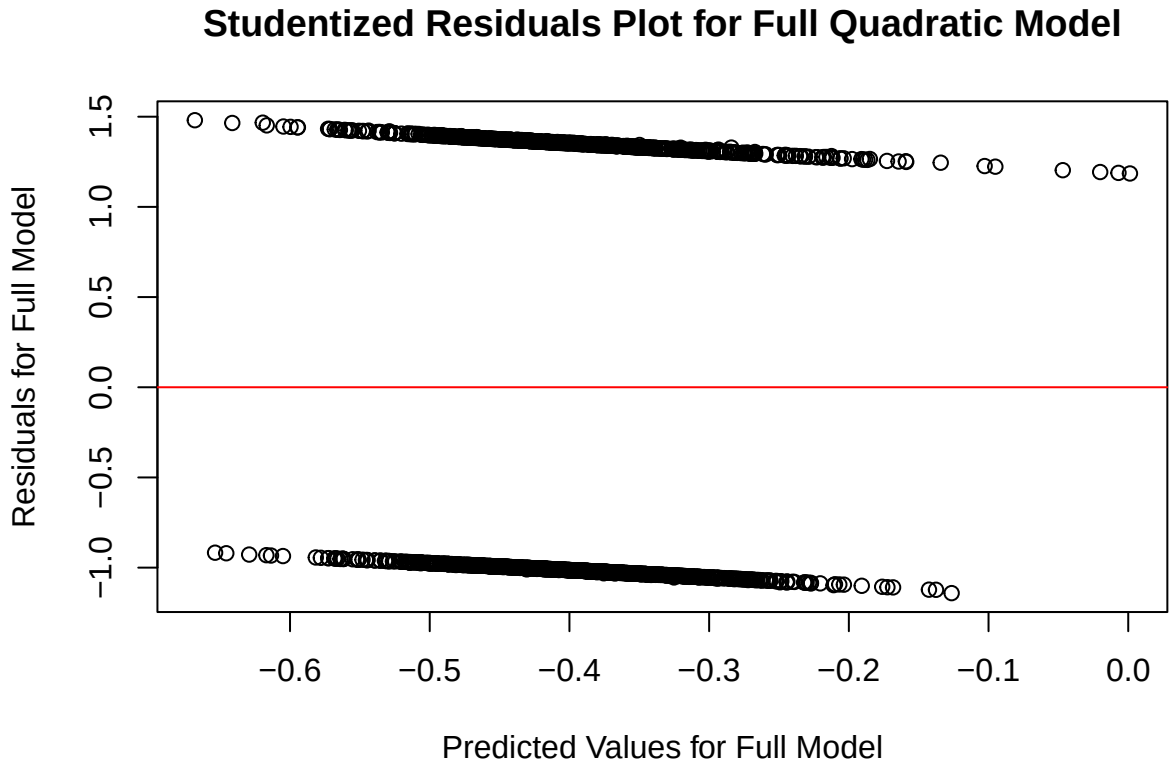
```
## Analysis of Deviance Table
##
## Model 1: Potability ~ Chloramines + Sulfate + Organic_carbon + Trihalomethanes +
##     I(Trihalomethanes^2)
## Model 2: Potability ~ Chloramines + Sulfate + Organic_carbon + Trihalomethanes
##   Resid. Df Resid. Dev Df  Deviance
## 1      1602     2165.6
## 2      1603     2165.7 -1 -0.056341
```

```
anova(quadratic_model_red, red.logit)
```

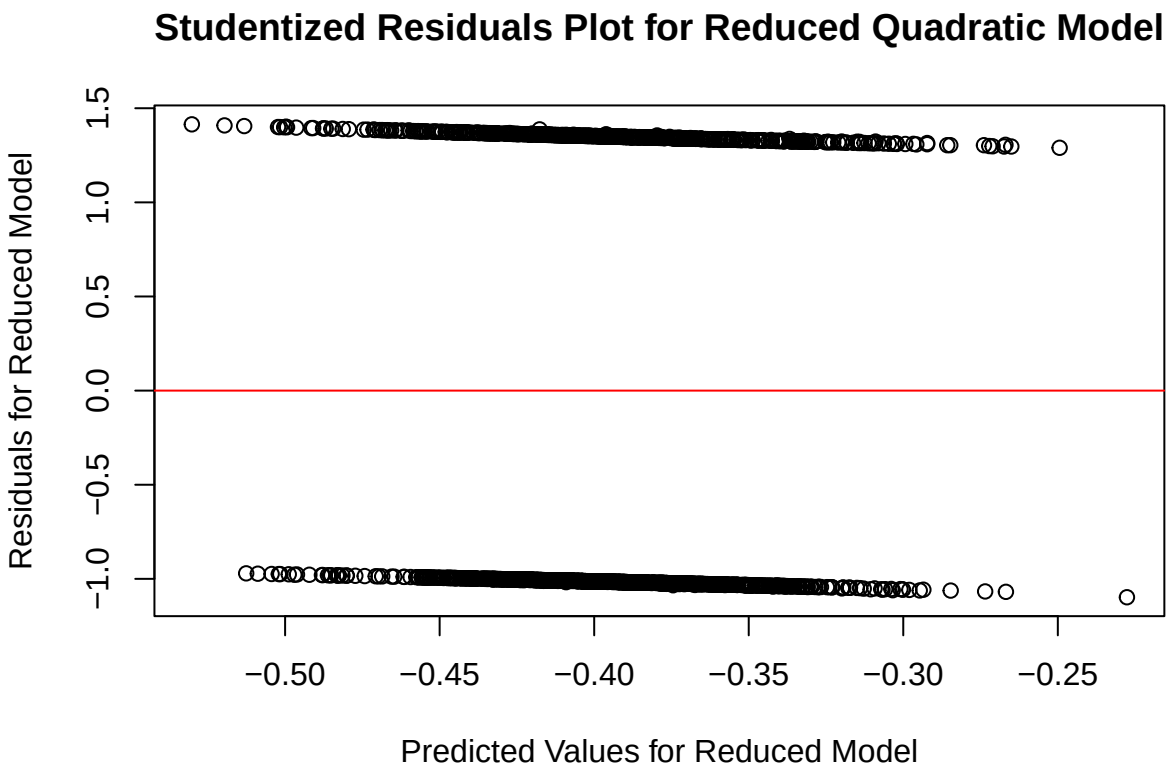
```
## Analysis of Deviance Table
##
## Model 1: Potability ~ Chloramines + Trihalomethanes + I(Trihalomethanes^2)
## Model 2: Potability ~ Chloramines + Trihalomethanes
##   Resid. Df Resid. Dev Df  Deviance
## 1      1604     2167.6
## 2      1605     2167.7 -1 -0.038438
```

```
#finding predictions and residuals of quadratic model
predicted_full <- predict(quadratic_model_full)
residuals_full <- rstudent(quadratic_model_full)
predicted_red <- predict(quadratic_model_red)
residuals_red<- rstudent(quadratic_model_red)
```

```r
# Create a residual plot
plot(predicted_full, residuals_full, xlab = "Predicted Values for Full Model", ylab = "Residuals for Ful
     main = "Studentized Residuals Plot for Full Quadratic Model")
abline(h = 0, col = "red") # residual plot shows no quadratic relationship between variables
```




```r
plot(predicted_red, residuals_red, xlab = "Predicted Values for Reduced Model", ylab = "Residuals for R
     main = "Studentized Residuals Plot for Reduced Quadratic Model")
abline(h = 0, col = "red")# residual plot shows no quadratic relationship between variables
```



Classification tree

```
library(rpart)
fit.allp <- rpart(Potability ~., method = "class", data = water_1.train,
                  control = rpart.control(minsplit = 1, cp = 0.001))
printcp(fit.allp)
```

```
##
## Classification tree:
## rpart(formula = Potability ~ ., data = water_1.train, method = "class",
##     control = rpart.control(minsplit = 1, cp = 0.001))
##
## Variables actually used in tree construction:
## [1] Chloramines     Conductivity    Hardness        Organic_carbon
## [5] ph              Solids          Sulfate         Trihalomethanes
## [9] Turbidity
##
## Root node error: 648/1608 = 0.40299
##
## n= 1608
##
##           CP nsplit rel error  xerror     xstd
## 1  0.0462963      0  1.000000 1.00000 0.030353
## 2  0.0339506      1  0.953704 0.97531 0.030225
## 3  0.0246914      3  0.885802 0.92438 0.029919
## 4  0.0216049      7  0.783951 0.91358 0.029846
```

```
## 5  0.0108025      8  0.762346 0.86111 0.029457
## 6  0.0092593     11  0.729938 0.85494 0.029407
## 7  0.0077160     13  0.711420 0.85802 0.029433
## 8  0.0069444     21  0.641975 0.85494 0.029407
## 9  0.0061728     28  0.591049 0.84105 0.029292
## 10 0.0046296     30  0.578704 0.84722 0.029344
## 11 0.0038580     46  0.503086 0.89506 0.029716
## 12 0.0030864     58  0.450617 0.89506 0.029716
## 13 0.0025720     99  0.313272 0.90741 0.029804
## 14 0.0023148    102  0.305556 0.91512 0.029857
## 15 0.0020576    121  0.259259 0.91975 0.029888
## 16 0.0015432    124  0.253086 0.91975 0.029888
## 17 0.0010000    254  0.046296 0.93364 0.029979
```

```
(rootnode_err <- sum(water_1.train$Potability=='present')/nrow(water_1.train))
```

```
## [1] 0
```

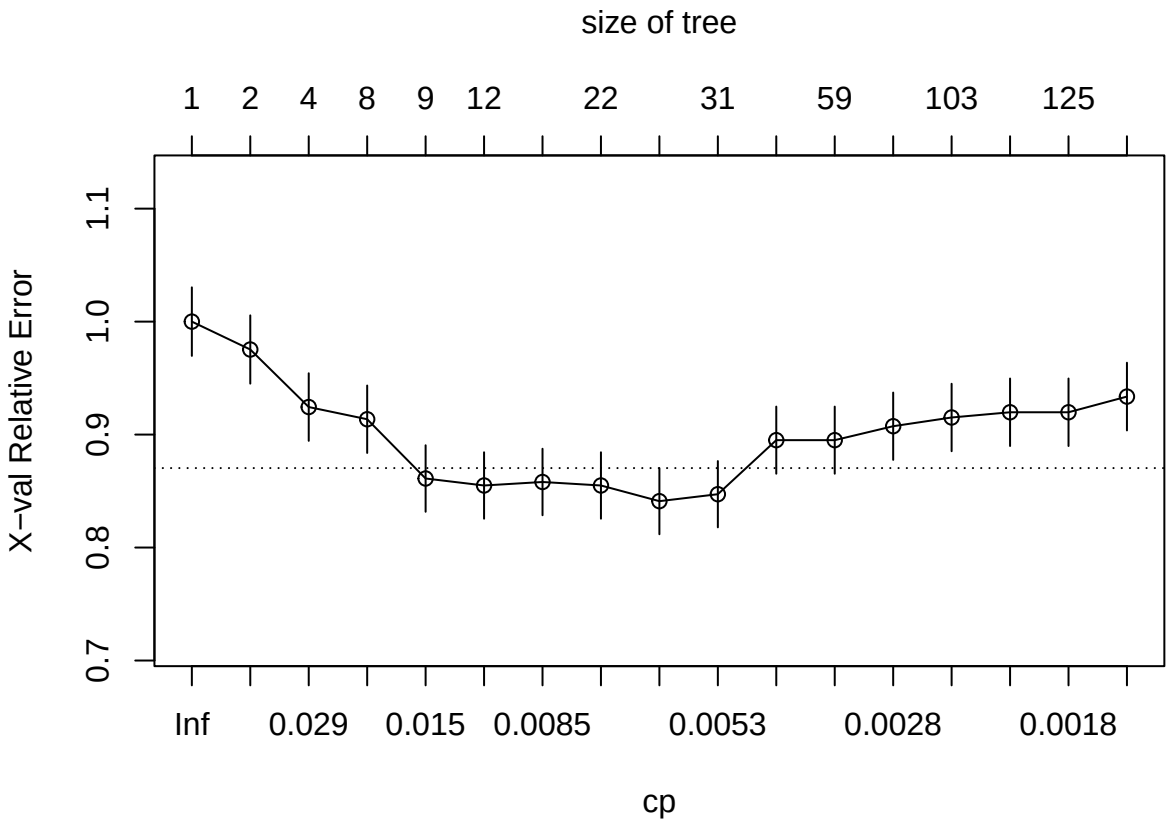
```
(cp= fit.allp$cptable[which.min(fit.allp$cptable[, "xerror"]), "CP"])
```

```
## [1] 0.00617284
```

```
(xerr = fit.allp$cptable[which.min(fit.allp$cptable[, "xerror"]), "xerror"])
```

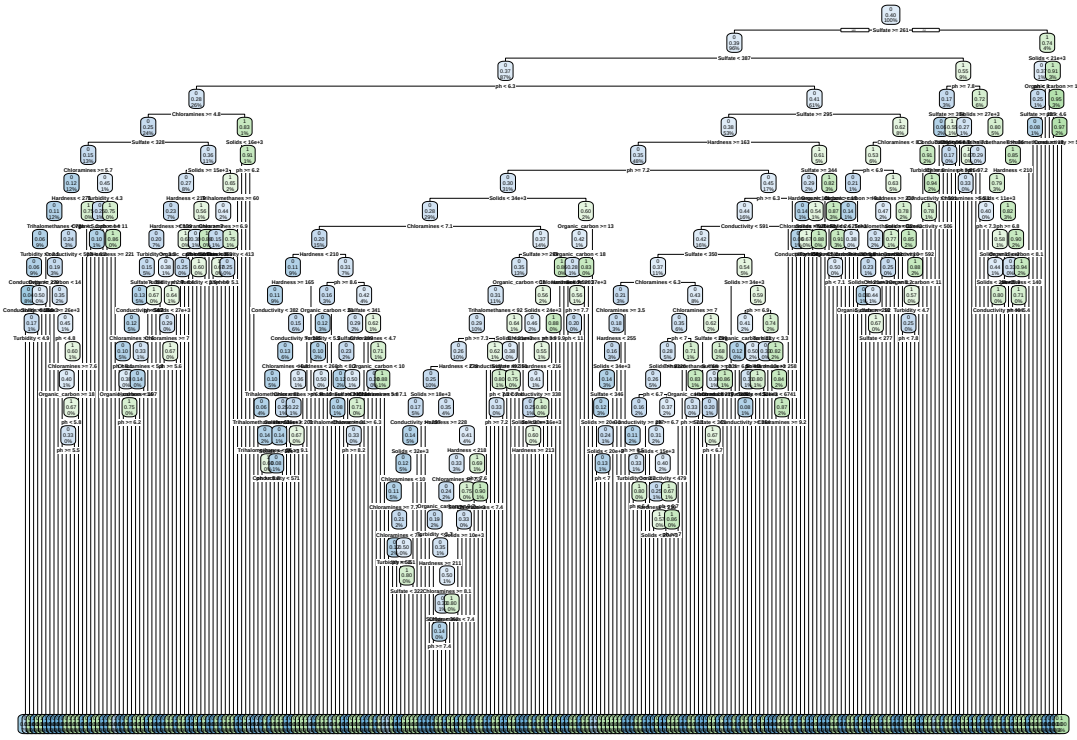
```
## [1] 0.8410494
```

```
plotcp(fit.allp)
```




```
library(rpart.plot)
rpart.plot(fit.allp, extra = "auto")
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



```
#confusion matrix
test_df <- data.frame(actual = water_1.test$Potability, pred = NA)
test_df$pred <- predict(fit.allp, newdata = water_1.test, type = "class")
(conf_matrix_base <- table(test_df$actual, test_df$pred)) #confusion matrix
```

```
##
##       0   1
##   0 151  89
##   1  87  76
```

```
#sensitivity(conf_matrix_base)
#specificity(conf_matrix_base)
(mis.rate <- conf_matrix_base[1, 2] +
   conf_matrix_base[2, 1])/sum(conf_matrix_base)
```

```
## [1] 0.4367246
```

```
#Training random forrest
library(ranger)
fit.rf.ranger <- ranger(Potability ~ ., data = water_1.train,
                        importance = 'impurity', mtry = 3)
print(fit.rf.ranger)
```

```
## Ranger result
```

```
##
## Call:
##  ranger(Potability ~ ., data = water_1.train, importance = "impurity",      mtry = 3)
##
## Type:                             Classification
## Number of trees:                  500
## Sample size:                      1608
## Number of independent variables:  9
## Mtry:                             3
## Target node size:                 1
## Variable importance mode:         impurity
## Splitrule:                        gini
## OOB prediction error:             31.59 %
```

```
library(vip)
```

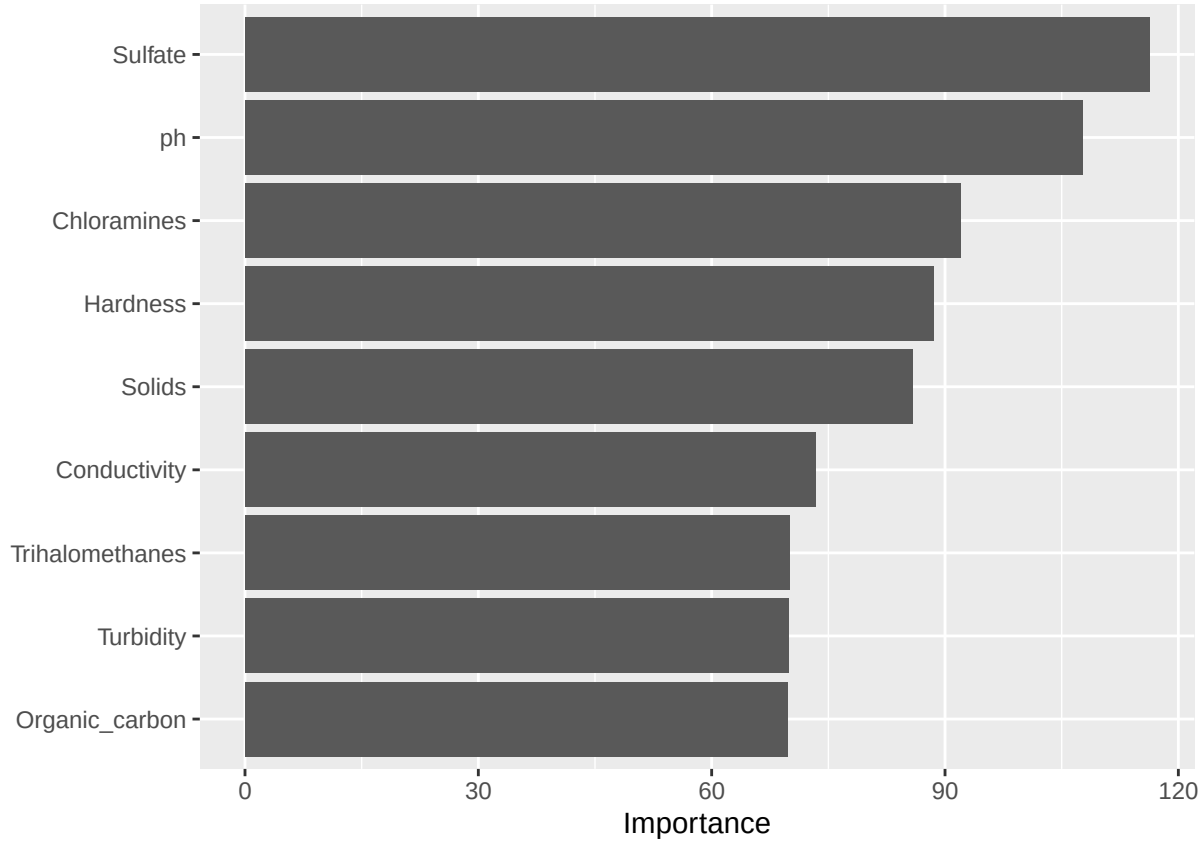
```
##
## Attaching package: 'vip'
```

```
## The following object is masked from 'package:utils':
##
##     vi
```

```
(v1 <- vi(fit.rf.ranger))
```

```
## # A tibble: 9 x 2
##   Variable        Importance
##   <chr>                <dbl>
## 1 Sulfate              116.
## 2 ph                   108.
## 3 Chloramines           92.0
## 4 Hardness              88.5
## 5 Solids                85.8
## 6 Conductivity          73.4
## 7 Trihalomethanes       70.0
## 8 Turbidity             69.9
## 9 Organic_carbon        69.8
```

```
vip(v1)
```




```
# predictions
pred <- predict(fit.rf.ranger, data = water_1.test)
test_df <- data.frame(actual =water_1.test$Potability, pred = NA)
test_df$pred <- pred$predictions
(conf_matrix_rf <- table(test_df$actual, test_df$pred)) #confusion matrix
```

```
##
##       0   1
##   0 203  37
##   1  97  66
```

```
# Sensitivity
sensitivity(conf_matrix_rf)
```

```
## [1] 0.6766667
```

```
# Specificity
specificity(conf_matrix_rf)
```

```
## [1] 0.6407767
```

```
# Missclassification error rate:
(conf_matrix_rf[1,2] + conf_matrix_rf[2,1])/sum(conf_matrix_rf)
```

```
## [1] 0.3325062
```

```
# Calculate Accuracy
accuracy <- sum(diag(conf_matrix_rf)) / sum(conf_matrix_rf)
accuracy
```

```
## [1] 0.6674938
```

XG Boost

```
#XG BOOST
library(xgboost)
library(Matrix)
# Transform the predictor matrix using dummy (or indicator or one-hot) encoding
matrix_predictors.train <-
  as.matrix(sparse.model.matrix(Potability ~., data = water_1.train))[, -1]
matrix_predictors.test <-
  as.matrix(sparse.model.matrix(Potability ~., data = water_1.test))[, -1]
train.gbm <- as.numeric(as.character(water_1.train$Potability))
dtrain <- xgb.DMatrix(data = data.matrix(matrix_predictors.train), label = water_1.train$Potability)
# Test dataset
dtest <- xgb.DMatrix(data = data.matrix(matrix_predictors.test), label = water_1.test$Potability)

# Encoding 'Kyphosis' as numeric

water_1.train$Potability <-as.numeric(as.factor(water_1.train$Potability)) - 1
water_1.test$Potability <- as.numeric(as.factor(water_1.test$Potability)) - 1


# Recreate the DMatrix objects
dtrain <- xgb.DMatrix(data = data.matrix(matrix_predictors.train), label = water_1.train$Potability)
dtest <- xgb.DMatrix(data = data.matrix(matrix_predictors.test), label = water_1.test$Potability)

# Proceed with training the model
watchlist <- list(train = dtrain, test = dtest)
param <- list(max_depth = 2, eta = 1, nthread = 2,
              objective = "binary:logistic", eval_metric = "auc")
model.xgb <- xgb.train(param, dtrain, nrounds = 2, watchlist)
```

```
## [1]  train-auc:0.557079  test-auc:0.547367
## [2]  train-auc:0.626513  test-auc:0.562820
```

```
# Making predictions on the test set
preds_test <- predict(model.xgb, dtest)

# The predictions are probabilities of the positive class.


binary_preds_test <- ifelse(preds_test > 0.5, 1, 0)
# Confusion Matrix for Test Set
conf_matrix_test <- confusionMatrix(as.factor(binary_preds_test), as.factor(water_1.test$Potability))
print(conf_matrix_test)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 212 126
##          1  28  37
##
##                Accuracy : 0.6179
##                  95% CI : (0.5685, 0.6655)
##     No Information Rate : 0.5955
##     P-Value [Acc > NIR] : 0.1944
##
##                   Kappa : 0.1221
##
##  Mcnemar's Test P-Value : 5.432e-15
##
##             Sensitivity : 0.8833
##             Specificity : 0.2270
##          Pos Pred Value : 0.6272
##          Neg Pred Value : 0.5692
##              Prevalence : 0.5955
##          Detection Rate : 0.5261
##    Detection Prevalence : 0.8387
##       Balanced Accuracy : 0.5552
##
##        'Positive' Class : 0
##
```

```
# Calculating metrics for the Test Set
accuracy_test <- conf_matrix_test$overall['Accuracy']
recall_test <- conf_matrix_test$byClass['Sensitivity']
ppv_test <- conf_matrix_test$byClass['Pos Pred Value']

# Printing metrics for the Test Set
print(paste("Test Set Accuracy is", accuracy_test))
```

```
## [1] "Test Set Accuracy is 0.617866004962779"
```

```
print(paste("Test Set  Recall is", recall_test))
```

```
## [1] "Test Set  Recall is 0.883333333333333"
```

```
print(paste("Test Set PPV is", ppv_test))
```

```
## [1] "Test Set PPV is 0.627218934911243"
```

```
# ROC and AUC for the Test Set
roc_test <- roc(water_1.test$Potability, preds_test)
```

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

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

```
auc_test <- auc(water_1.test$Potability, preds_test)
```

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

```
print(paste("AUC for Test Set is", auc_test))
```

```
## [1] "AUC for Test Set is 0.562819529652352"
```