Final

Team 10

5/2/2021

Introduction

Describe your data set. Provide proper motivation for your work.

What questions are you trying to answer? How did you prepare and clean the data?

This is the introduction of the dataset and the aims of this projects:

Exploratory analysis

In total 855 wines were classified as "Good" quality and 744 as "Poor" quality. The average values for the 11 features for wines of good and poor quality was shown in Table 1. Fixed acidity, volatile acidity, citric acid, chlorides, free sulfur dioxide, total sulfur dioxide, density, sulphates and alcohol were significantly associated with the wine quality (P-values for t-tests < 0.05), which suggests important predictors.

We also built the density plots to explore the distribution of the 11 continuous variables over "Poor" and "Good" quality of wine (Figure 1). The plots showed that wine with good and poor quality did not differ for PH and residual sugar, while different types of wine differs in other variables, which was consistent with the t-test results.

```
# Read in data
wine <- read.csv("./winequality-red.csv", stringsAsFactors = FALSE) %>%
  janitor::clean_names() %>%
  na.omit() %>%
  mutate(qual = case_when(quality > 5 ~ "good", quality <= 5 ~ "poor")) %>%
  mutate(qual = as.factor(qual)) %>%
  dplyr::select(-quality)

theme1 <- transparentTheme(trans = .4)
trellis.par.set(theme1)
featurePlot(x = wine[, 1:11],
            y = wine$qual,
            scales = list(x = list(relation = "free"),
                          y = list(relation = "free")),
            plot = "density", pch = "|",
            auto.key = list(columns = 2))
```

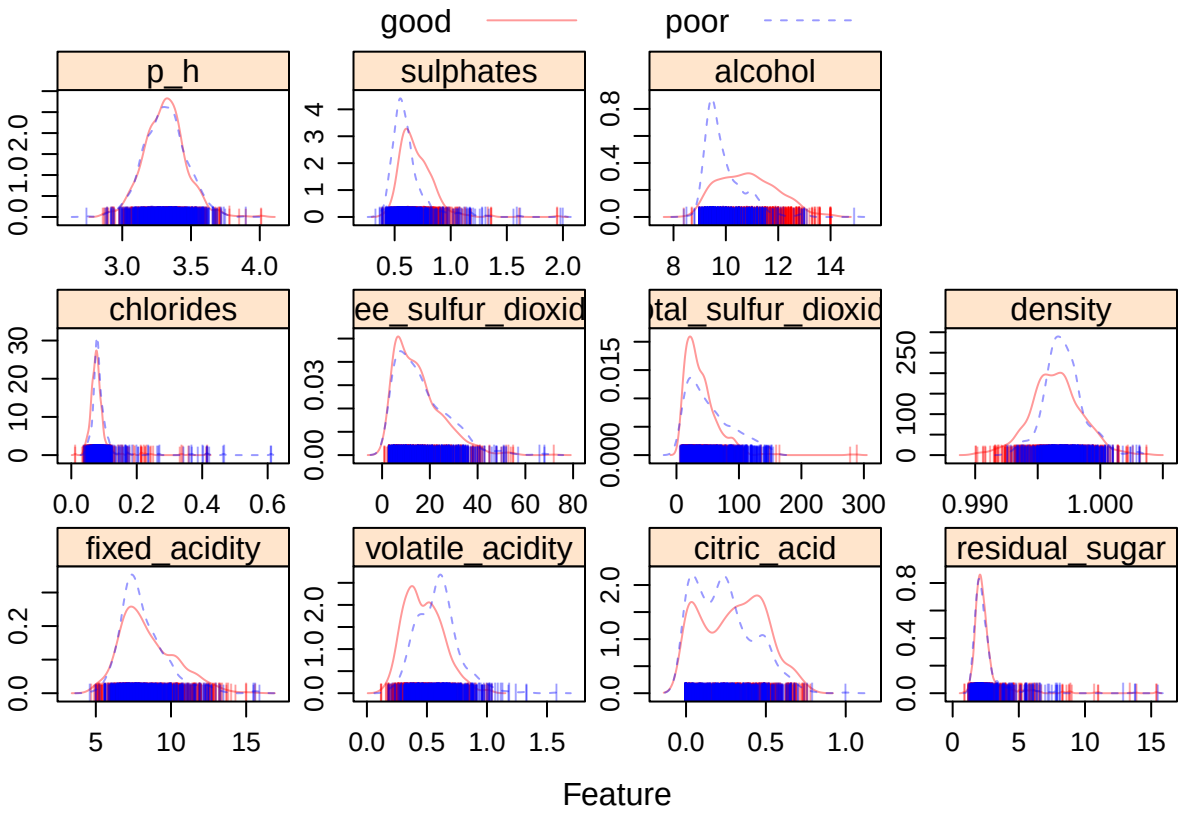


Figure 1. Descriptive plots between wine quality and predictive features.

Table 1. Basic characteristics of wines over good and poor quality.

```
# Create a variable list which we want in Table 1
listVars <- c("fixed_acidity", "volatile_acidity","citric_acid", "residual_sugar", "chlorides", "free_su

tab1 <- CreateTableOne(vars = listVars, strata = 'qual', data = wine)

tab1
```

```
##                                  Stratified by qual
##                                   good          poor          p      test
##   n                                 855           744
##   fixed_acidity (mean (SD))        8.47 (1.86)   8.14 (1.57)  <0.001
##   volatile_acidity (mean (SD))     0.47 (0.16)   0.59 (0.18)  <0.001
##   citric_acid (mean (SD))          0.30 (0.20)   0.24 (0.18)  <0.001
##   residual_sugar (mean (SD))       2.54 (1.42)   2.54 (1.39)   0.931
##   chlorides (mean (SD))            0.08 (0.04)   0.09 (0.06)  <0.001
##   free_sulfur_dioxide (mean (SD)) 15.27 (10.04) 16.57 (10.89)  0.014
##   total_sulfur_dioxide (mean (SD)) 39.35 (27.25) 54.65 (36.72) <0.001
##   density (mean (SD))              1.00 (0.00)   1.00 (0.00)  <0.001
##   p_h (mean (SD))                  3.31 (0.15)   3.31 (0.15)   0.896
##   sulphates (mean (SD))            0.69 (0.16)   0.62 (0.18)  <0.001
##   alcohol (mean (SD))             10.86 (1.11)   9.93 (0.76)  <0.001
```

```r
# Table 1
table1(~ fixed_acidity + volatile_acidity + citric_acid + residual_sugar + chlorides+free_sulfur_dioxid
```

```
## [1] "<table class=\"Rtable1\">\n<thead>\n<tr>\n<th class='rowlabel firstrow lastrow'></th>\n<th class
```

Model Building

We randomly selected 70% of the observations as the training data and the rest as the test data. All the 11 predictors were included into analysis. We performed linear methods, non-linear methods and the tree method to predict the classification of wine quality. For linear methods, we trained (penalized) logistic regression model and linear discriminant analysis (LDA). The assumptions for logistic regression includes observations being independent of each other and the linearity of independent variables and log odds. LDA assumes normally distributed features, but LDA was robust for classification. For nonlinear models, we performed generalized additive model (GAM), multivariate adaptive regression splines (MARS), KNN model and quadratic discriminant analysis (QDA). For tree models, we conducted classification tree, boosting and random forest model. We calculated the ROC and accuracy for model selection, and also investigated the variable importance. 10-fold cross-validation (CV) were used for all model buildings.

```r
### Data Partition
set.seed(1)
indexTrain <- createDataPartition(y = wine$qual, p = 0.7, list = FALSE)
trainData <- wine[indexTrain, ]
testData <- wine[-indexTrain, ]
```

Linear models The multiple logistic regression showed that among the 11 predictors, volatile acidity, citric acid, free sulfur dioxide, total sulfur dioxide, sulphates and alcohol were significantly associated with wine quality (P-values < 0.05), explaining 25.1% of the total variance in wine quality. When applying this model to the test data, the accuracy is 0.75 (95%CI: 0.71-0.79) and the ROC is 0.818, which suggests relatively good fit for the data. When performing the penalized logistic regression, we found that when maximizing the ROC, the best tuning parameter was alpha=1 and lambda=0.00086, the accuracy was 0.75 (95%CI: 0.71-0.79) and the ROC was also 0.818. Since lambda was close to zero and the ROC was the same as the full logistic regression model, the penalization was relatively small, which suggested that the full logistic regression model was simple enough for classification. However, since logistic regression requires there to be little or no multicollinearity among the independent variables, the model may be disturbed by collinearity between the 11 predictors, if there was any.

```r
### Logistic regression
# Using caret
ctrl <- trainControl(method = "cv", number = 10,
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)
set.seed(1)
model.glm <- train(x = trainData %>% dplyr::select(-qual),
                   y = trainData$qual,
                   method = "glm",
                   metric = "ROC",
                   trControl = ctrl)

# Checking the significance of predictors
summary(model.glm)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2944  -0.8157  -0.3045   0.8380   3.3331
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -12.336077  94.275480  -0.131  0.89589
## fixed_acidity         -0.128817   0.115195  -1.118  0.26346
## volatile_acidity       3.766894   0.605226   6.224 4.85e-10 ***
## citric_acid            1.971704   0.693754   2.842  0.00448 **
## residual_sugar        -0.018813   0.067682  -0.278  0.78104
## chlorides              2.207284   1.825163   1.209  0.22652
## free_sulfur_dioxide   -0.022291   0.010162  -2.194  0.02826 *
## total_sulfur_dioxide   0.017924   0.003574   5.015 5.30e-07 ***
## density               19.301566  96.123157   0.201  0.84085
## p_h                    0.573935   0.842937   0.681  0.49595
## sulphates             -2.260253   0.513122  -4.405 1.06e-05 ***
## alcohol               -0.921104   0.126440  -7.285 3.22e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1547.2  on 1119  degrees of freedom
## Residual deviance: 1158.3  on 1108  degrees of freedom
## AIC: 1182.3
##
## Number of Fisher Scoring iterations: 4
```

```
# Building confusion matrix
test.pred.prob <- predict(model.glm, newdata = testData,
                          type = "prob")
test.pred <- rep("good", length(test.pred.prob$good))
test.pred[test.pred.prob$good < 0.5] <- "poor"

confusionMatrix(data = as.factor(test.pred),
                reference = testData$qual,
                positive = "good")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction good poor
##       good  194   58
##       poor   62  165
##
##                Accuracy : 0.7495
##                  95% CI : (0.7082, 0.7877)
##     No Information Rate : 0.5344
##     P-Value [Acc > NIR] : <2e-16
```

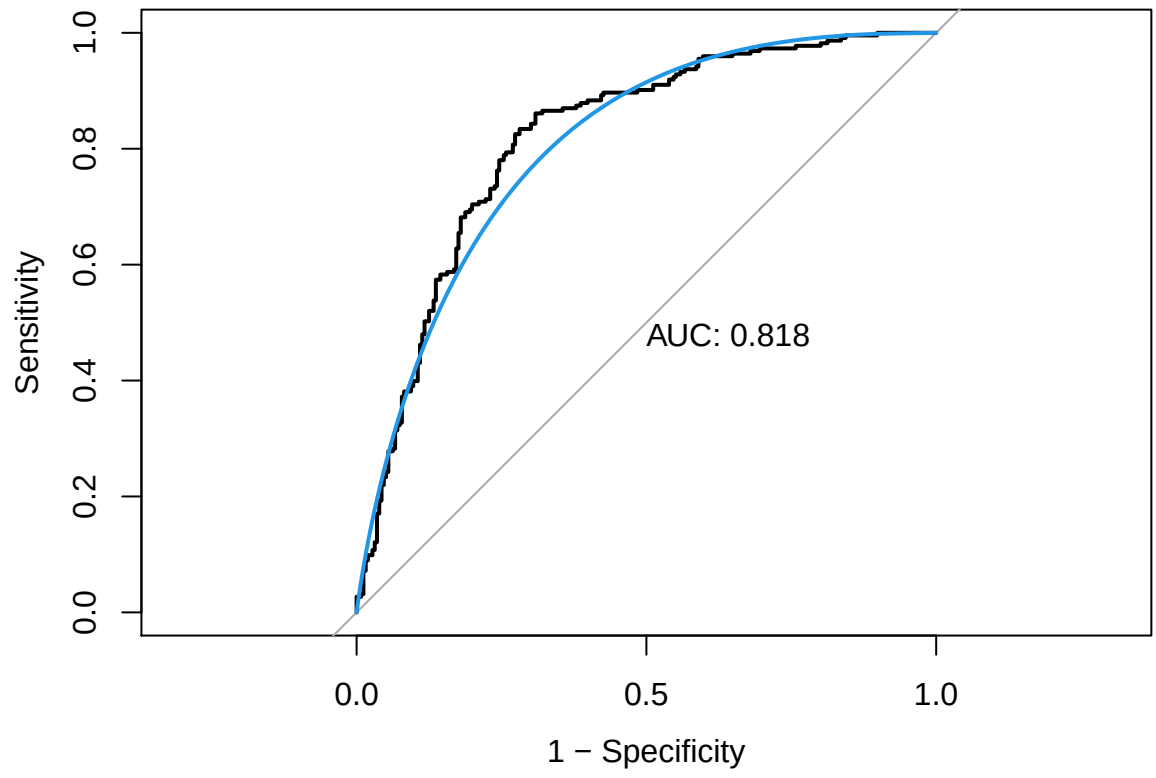
```
##
##                  Kappa : 0.4971
##
##  Mcnemar's Test P-Value : 0.7842
##
##             Sensitivity : 0.7578
##             Specificity : 0.7399
##          Pos Pred Value : 0.7698
##          Neg Pred Value : 0.7269
##              Prevalence : 0.5344
##          Detection Rate : 0.4050
##    Detection Prevalence : 0.5261
##       Balanced Accuracy : 0.7489
##
##        'Positive' Class : good
##
```

```
# Plot the test ROC
roc.glm <- roc(testData$qual, test.pred.prob$good)
```

```
## Setting levels: control = good, case = poor
```

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

```
plot(roc.glm, legacy.axes = TRUE, print.auc = TRUE)
plot(smooth(roc.glm), col = 4, add = TRUE)
```

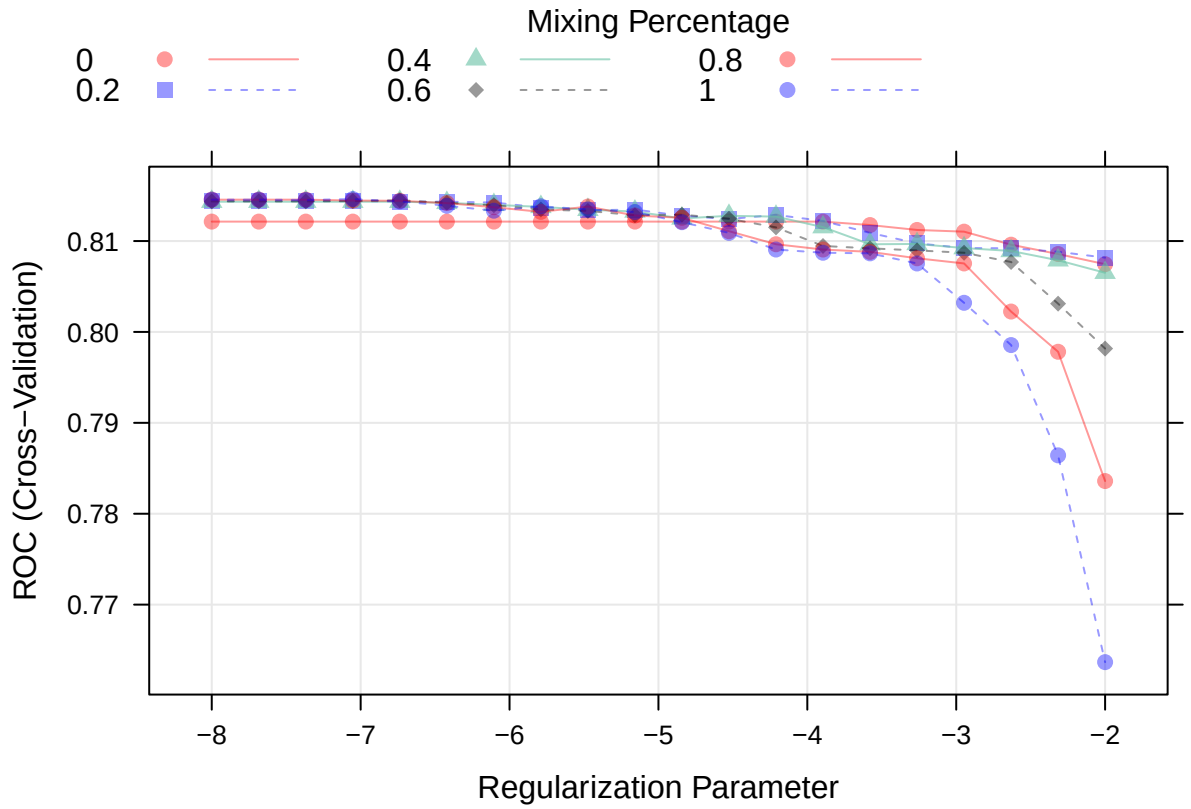


```r
## Test error and train error
error.test.glm <- mean(testData$qual != test.pred)

train.pred.prob.glm <- predict(model.glm, newdata = trainData,
                               type = "prob")
train.pred.glm <- rep("good", length(train.pred.prob.glm$good))
train.pred.glm[train.pred.prob.glm$good < 0.5] <- "poor"
error.trian.glm <- mean(trainData$qual != train.pred.glm)
```

```r
### Penalized logistic regression
glmnGrid <- expand.grid(.alpha = seq(0, 1, length = 6),
                        .lambda = exp(seq(-8, -2, length = 20)))
set.seed(1)
model.glmn <- train(x = trainData %>% dplyr::select(-qual),
                    y = trainData$qual,
                    method = "glmnet",
                    tuneGrid = glmnGrid,
                    metric = "ROC",
                    trControl = ctrl)

plot(model.glmn, xTrans = function(x) log(x))
```




```r
# select the best tune
model.glmn$bestTune
```

```
##     alpha       lambda
## 104     1 0.0008651293
```

```
# Building confusion matrix
test.pred.prob2 <- predict(model.glmn, newdata = testData,
                           type = "prob")
test.pred2 <- rep("good", length(test.pred.prob2$good))
test.pred2[test.pred.prob2$good < 0.5] <- "poor"

confusionMatrix(data = as.factor(test.pred2),
                reference = testData$qual,
                positive = "good")
```

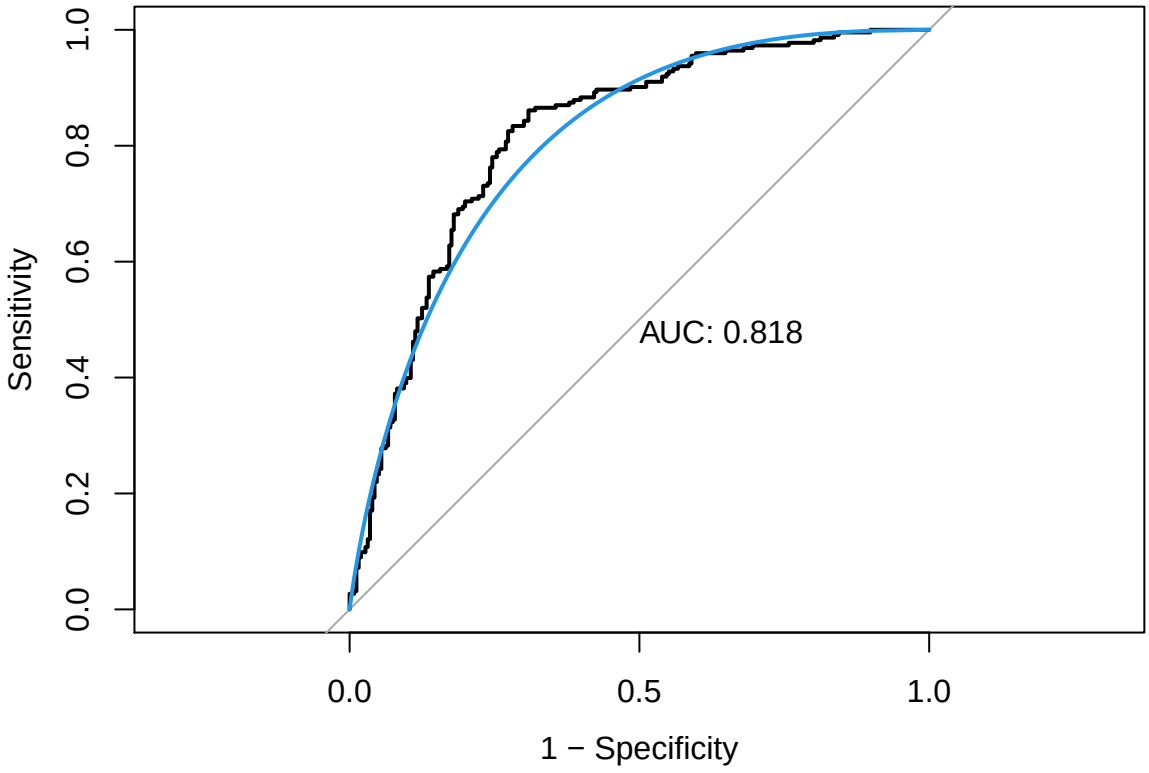
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction good poor
##       good  196   58
##       poor   60  165
##
##                Accuracy : 0.7537
##                  95% CI : (0.7125, 0.7916)
##     No Information Rate : 0.5344
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.5052
##
##  Mcnemar's Test P-Value : 0.9267
##
##             Sensitivity : 0.7656
##             Specificity : 0.7399
##          Pos Pred Value : 0.7717
##          Neg Pred Value : 0.7333
##              Prevalence : 0.5344
##          Detection Rate : 0.4092
##    Detection Prevalence : 0.5303
##       Balanced Accuracy : 0.7528
##
##        'Positive' Class : good
##
```

```
# Plot the test ROC
roc.glmn <- roc(testData$qual, test.pred.prob2$good)
```

```
## Setting levels: control = good, case = poor
```

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

```
plot(roc.glm, legacy.axes = TRUE, print.auc = TRUE)
plot(smooth(roc.glm), col = 4, add = TRUE)
```




```
## Test error and train error
error.test.glmn <- mean(testData$qual != test.pred2)

train.pred.prob.glmn <- predict(model.glmn, newdata = trainData,
                            type = "prob")
train.pred.glmn <- rep("good", length(train.pred.prob.glmn$good))
train.pred.glmn[train.pred.prob.glmn$good < 0.5] <- "poor"
error.trian.glmn <- mean(trainData$qual != train.pred.glmn)
```

Nonlinear models In the GAM model, only the degree of freedom for volatile acidity was equal to 1, suggesting linear association, while smoothing spline was applied for all other 10 variables. The results showed that alcohol, citric acid, residual sugar, sulphates, fixed acidity, volatile acidity, chlorides and total sulfur dioxide were significant predictors (P-values < 0.05). In total, these variables explained 39.1% of the total variance in wine quality. The confusion matrix using the test data showed that the accuracy for GAM was 0.76 (95%CI: 0.72-0.80) and the ROC was 0.829. The MARS model showed that when maximizing the ROC, we included 5 terms out of 11 predictors, with nprune equal to 5 and degree of 2. In total, these predictors and hinge functions explained 32.2% of the total variance. According to the MARS output, the 3 most important predictors were total sulfur dioxide, alcohol and sulphates. When applying the MARS model to the test data, the accuracy is 0.76 (95%CI: 0.72, 0.80) and the ROC is 0.823. We also performed the KNN model for classification. When k was equal to 22, the ROC was maximized. The accuracy for KNN model was 0.63 (95%CI: 0.59-0.68) and the ROC was 0.672.

The advantage of GAM and MARS is that both two models are nonparametric models and able to deal with highly complex nonlinear relationship. Specifically, MARS model can include potential interaction effects into the model. However, because of the model complexity, time-consuming computation and the

high propensity of overfitting are the limitations for the two models. As for the KNN model, when k was large, the prediction may not be accurate.

```
### GAM
set.seed(1)
model.gam <- train(x = trainData %>% dplyr::select(-qual),
                   y = trainData$qual,
                    method = "gam",
                    metric = "ROC",
                    trControl = ctrl)


model.gam$finalModel
```

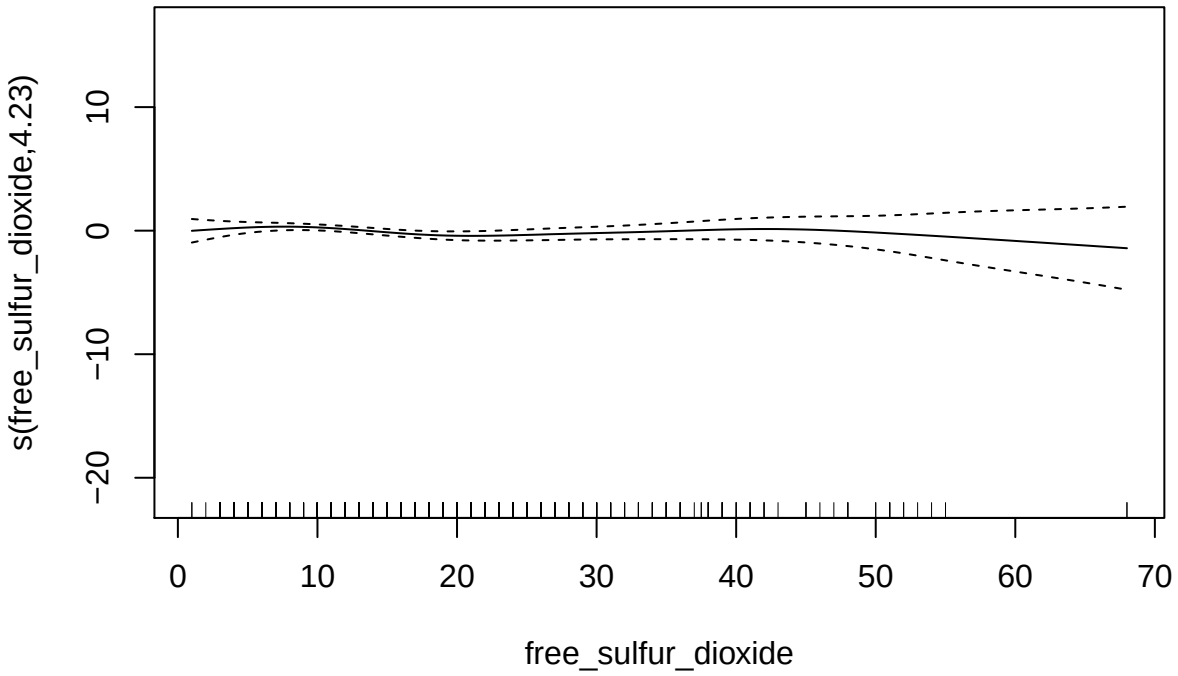
```
##
## Family: binomial
## Link function: logit
##
## Formula:
## .outcome ~ s(free_sulfur_dioxide) + s(alcohol) + s(citric_acid) +
##     s(residual_sugar) + s(p_h) + s(sulphates) + s(fixed_acidity) +
##     s(volatile_acidity) + s(chlorides) + s(total_sulfur_dioxide) +
##     s(density)
##
## Estimated degrees of freedom:
## 4.232 6.021 1.804 7.709 0.478 3.774 3.492
## 1.000 5.916 6.421 7.333  total = 49.18
##
## UBRE score: -0.02749885
```

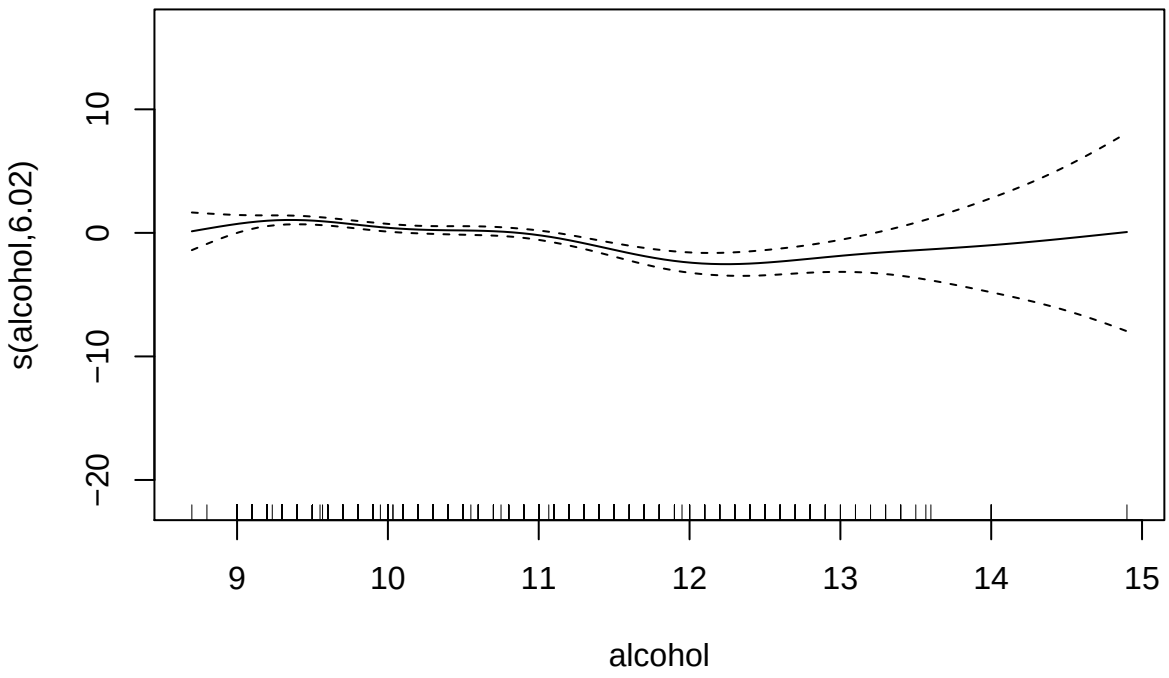
```
summary(model.gam)
```

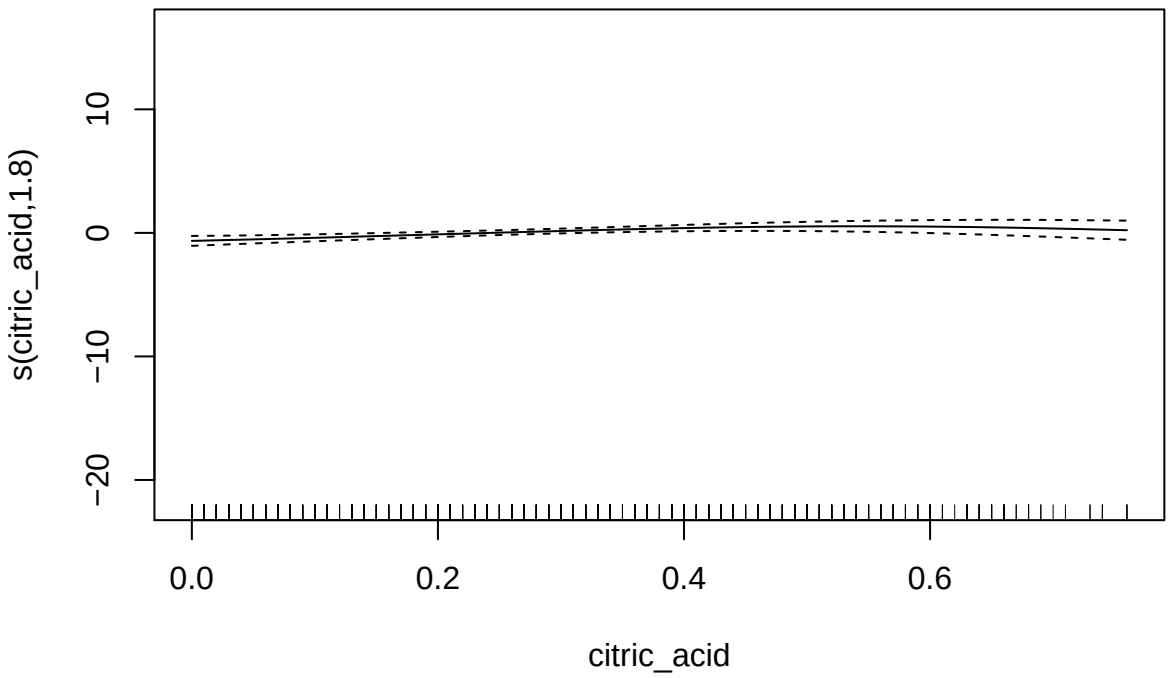
```
##
## Family: binomial
## Link function: logit
##
## Formula:
## .outcome ~ s(free_sulfur_dioxide) + s(alcohol) + s(citric_acid) +
##     s(residual_sugar) + s(p_h) + s(sulphates) + s(fixed_acidity) +
##     s(volatile_acidity) + s(chlorides) + s(total_sulfur_dioxide) +
##     s(density)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.26628    0.09157  -2.908  0.00364 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                            edf Ref.df Chi.sq  p-value
## s(free_sulfur_dioxide)  4.2319      9  8.024 0.077961 .
## s(alcohol)              6.0214      9 47.408  < 2e-16 ***
## s(citric_acid)          1.8039      9 10.876 0.000814 ***
```

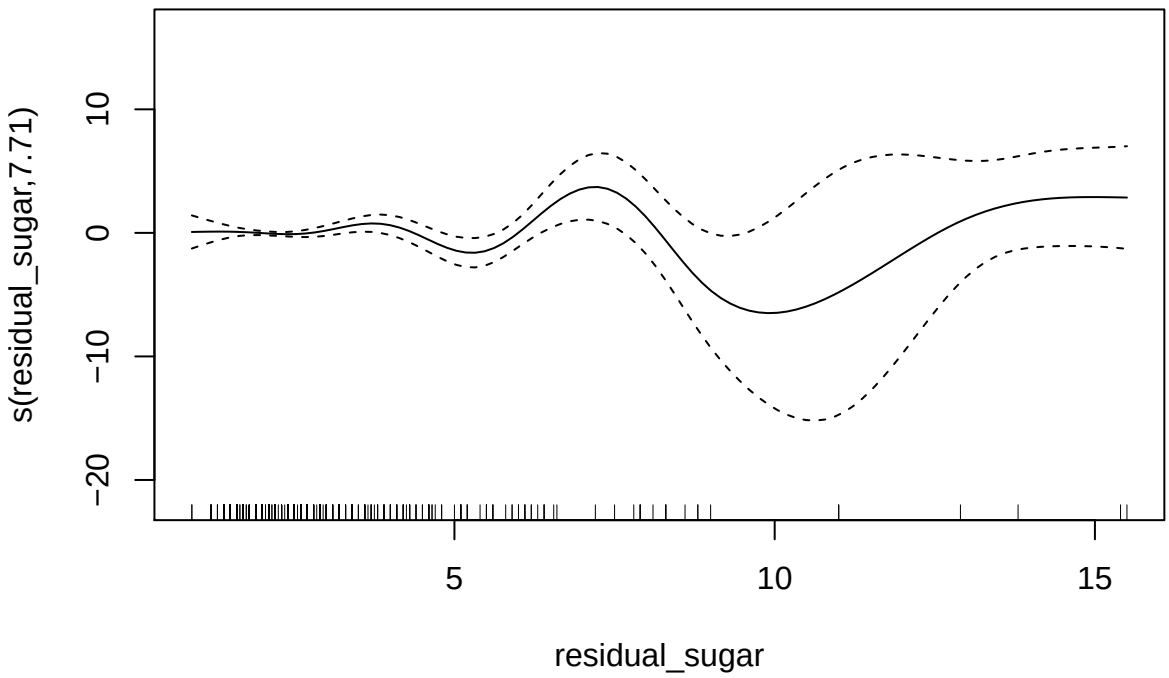
```
## s(residual_sugar)       7.7091      9 17.691 0.015860 *
## s(p_h)                  0.4784      9  0.783 0.164933
## s(sulphates)            3.7737      9 56.546  < 2e-16 ***
## s(fixed_acidity)        3.4917      9 13.818 0.000877 ***
## s(volatile_acidity)     1.0000      9 26.986  < 2e-16 ***
## s(chlorides)            5.9163      9 14.507 0.013552 *
## s(total_sulfur_dioxide) 6.4206      9 29.741 1.22e-05 ***
## s(density)              7.3330      9 12.897 0.064167 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.391   Deviance explained =   36%
## UBRE = -0.027499  Scale est. = 1         n = 1120
```

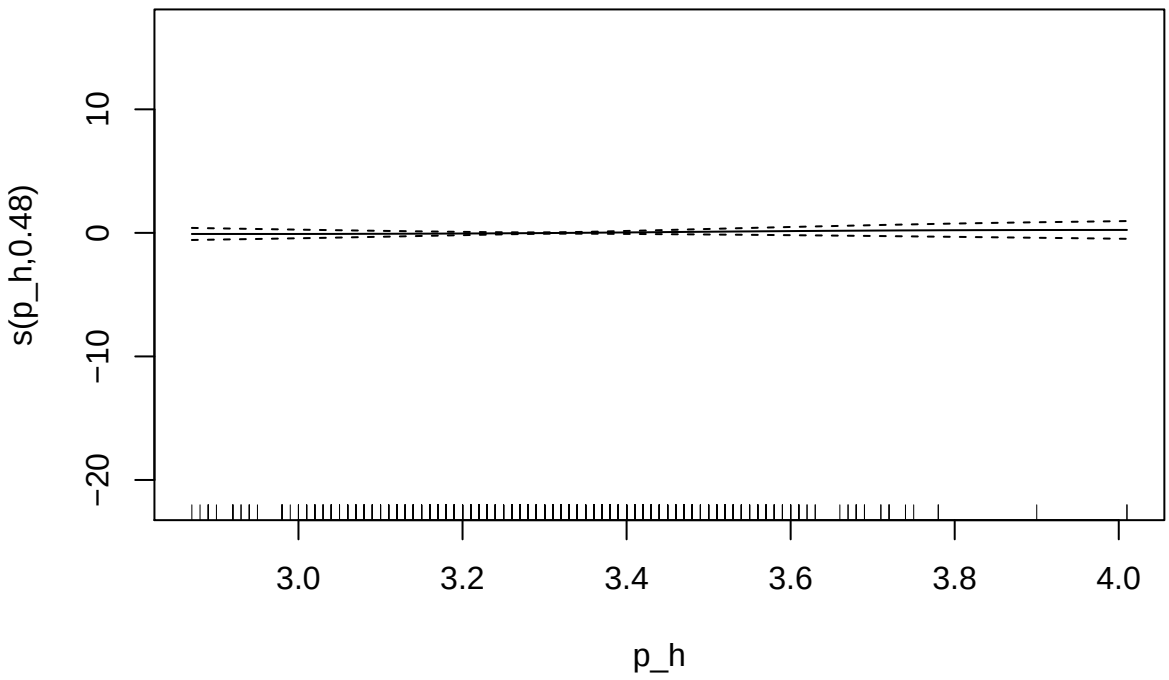
```
plot(model.gam$finalModel)
```

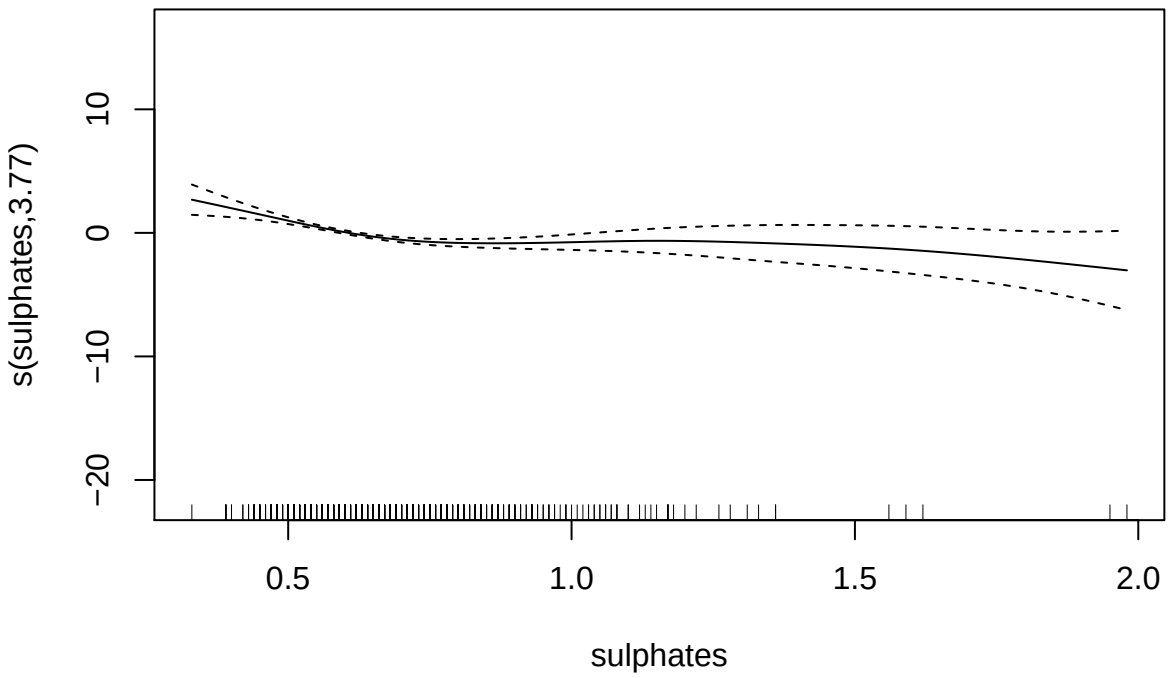


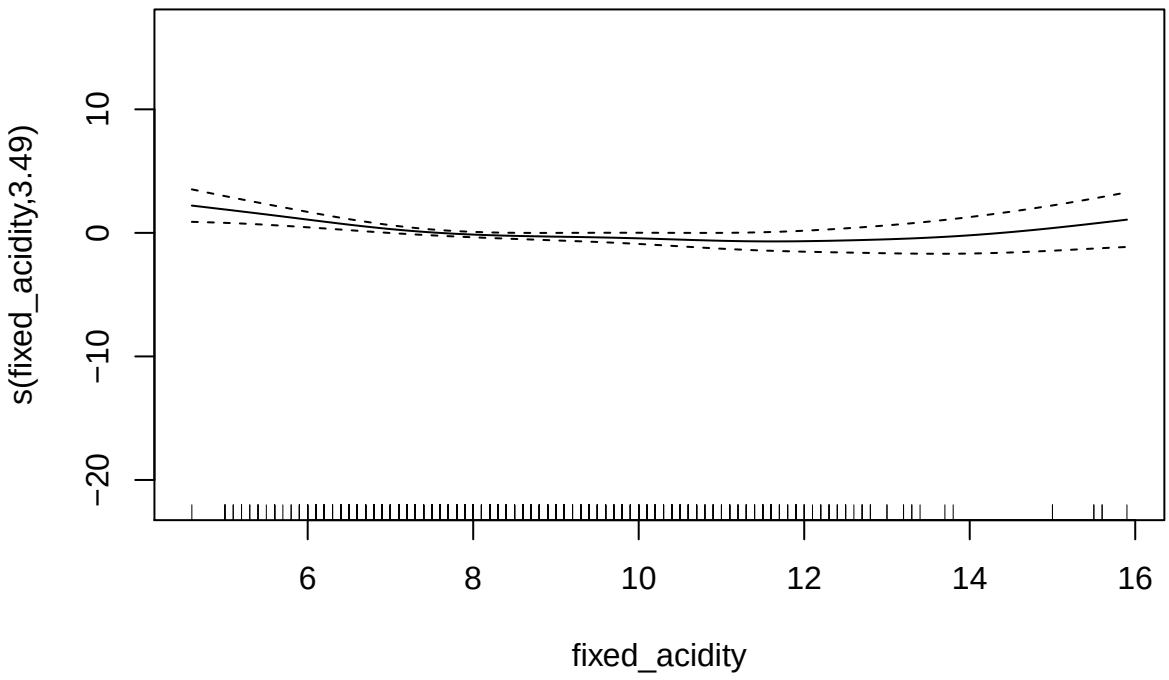

s(alcohol,6.02)
10
0
−10
−20
9
10
11
12
13
14
15
alcohol

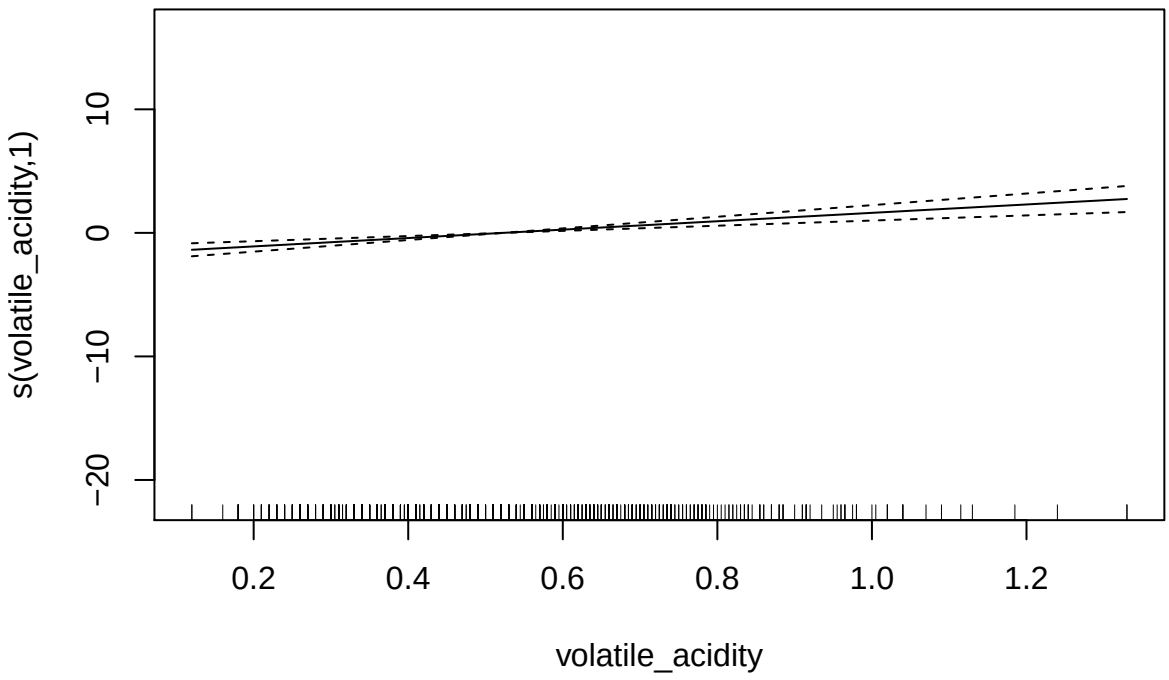

s(citric_acid,1.8)
10
0
−10
−20
0.0
0.2
0.4
0.6
citric_acid

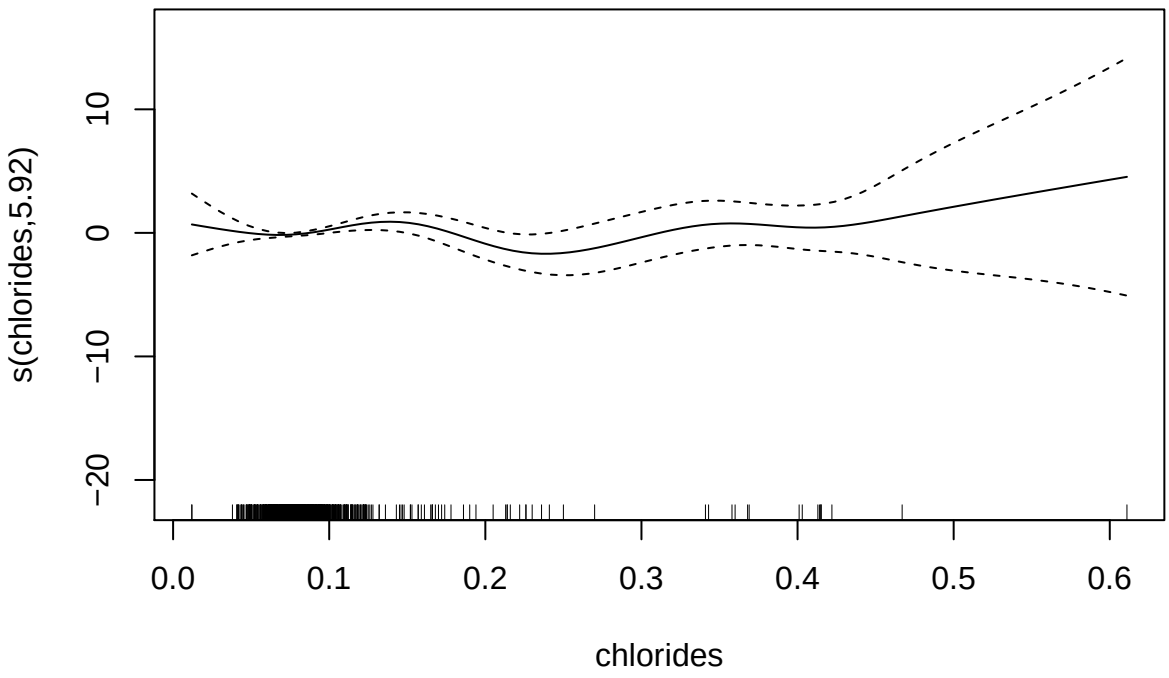

s(residual_sugar,7.71)
10
0
−10
−20
5
10
15
residual_sugar

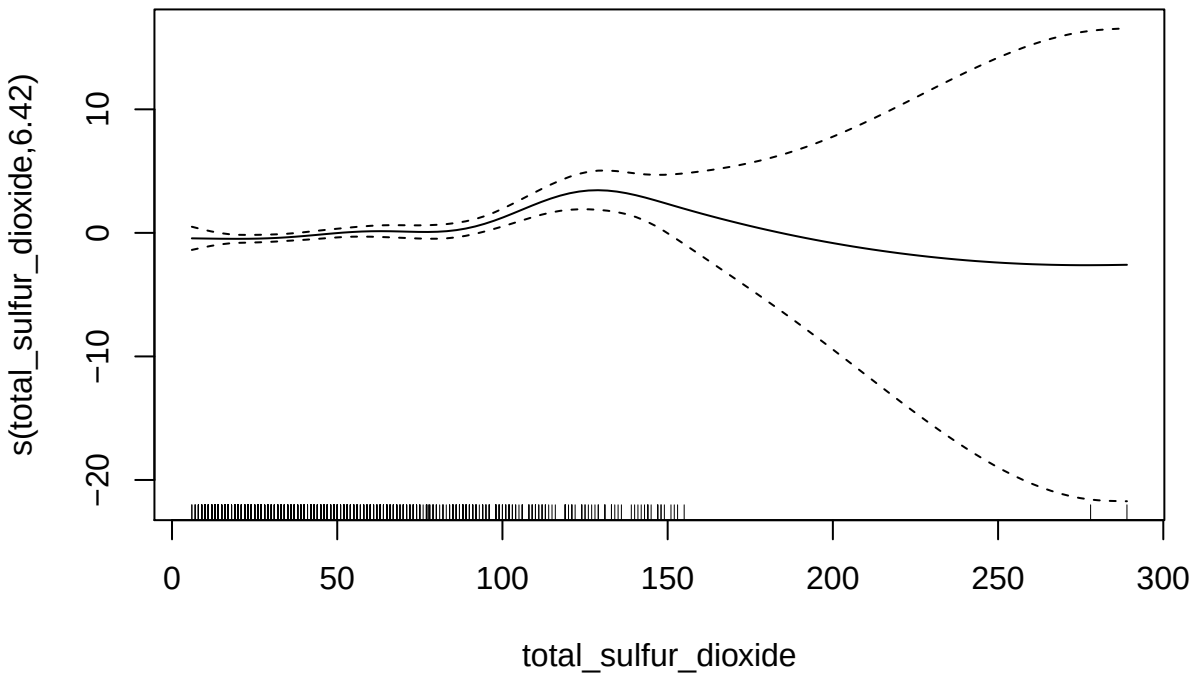

s(p_h,0.48)
10
0
−10
−20
3.0
3.2
3.4
3.6
3.8
4.0
p_h

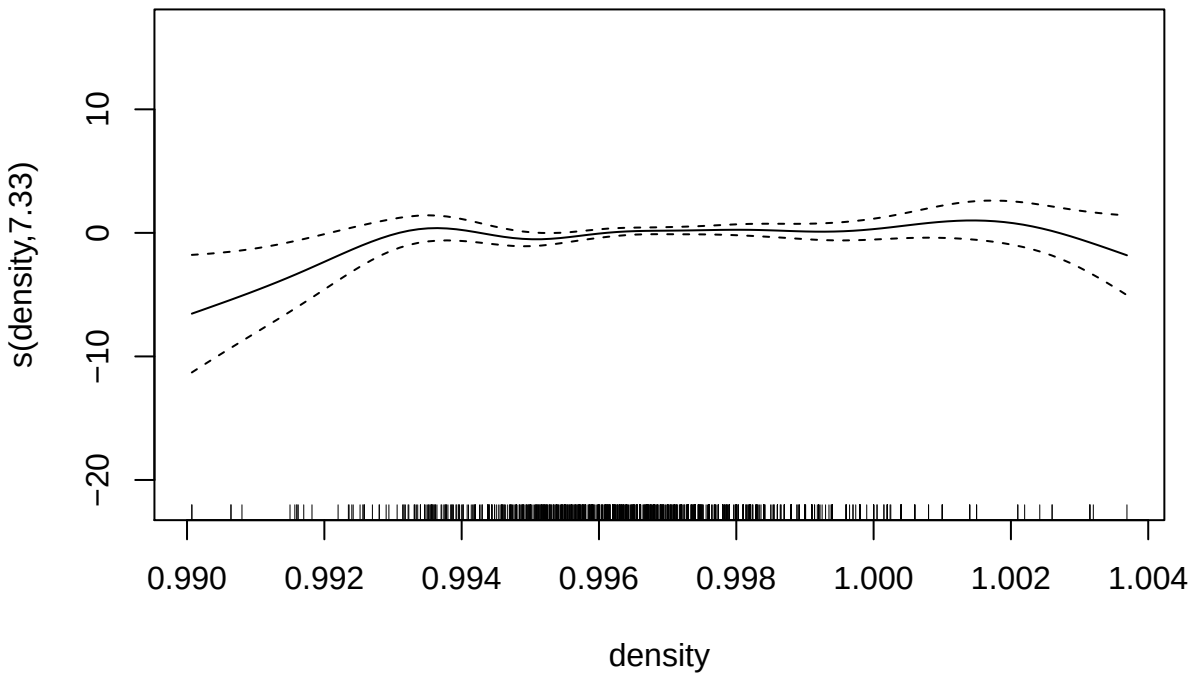

s(sulphates,3.77)
10
0
−10
−20
0.5
1.0
1.5
2.0
sulphates


s(fixed_acidity,3.49)
10
0
−10
−20
6
8
10
12
14
16
fixed_acidity


s(volatile_acidity,1)
10
0
−10
−20
0.2
0.4
0.6
0.8
1.0
1.2
volatile_acidity


s(chlorides,5.92)
10
0
−10
−20
0.0
0.1
0.2
0.3
0.4
0.5
0.6
chlorides


s(total_sulfur_dioxide,6.42)
10
0
−10
−20
0
50
100
150
200
250
300
total_sulfur_dioxide




```
# Building confusion matrix
test.pred.prob3 <- predict(model.gam, newdata = testData,
                           type = "prob")
test.pred3 <- rep("good", length(test.pred.prob3$good))
test.pred3[test.pred.prob3$good < 0.5] <- "poor"

confusionMatrix(data = as.factor(test.pred3),
                reference = testData$qual,
                positive = "good")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction good poor
##       good  200   59
##       poor   56  164
##
##                Accuracy : 0.7599
##                  95% CI : (0.7191, 0.7975)
##     No Information Rate : 0.5344
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.5171
##
##  Mcnemar's Test P-Value : 0.8521
##
```

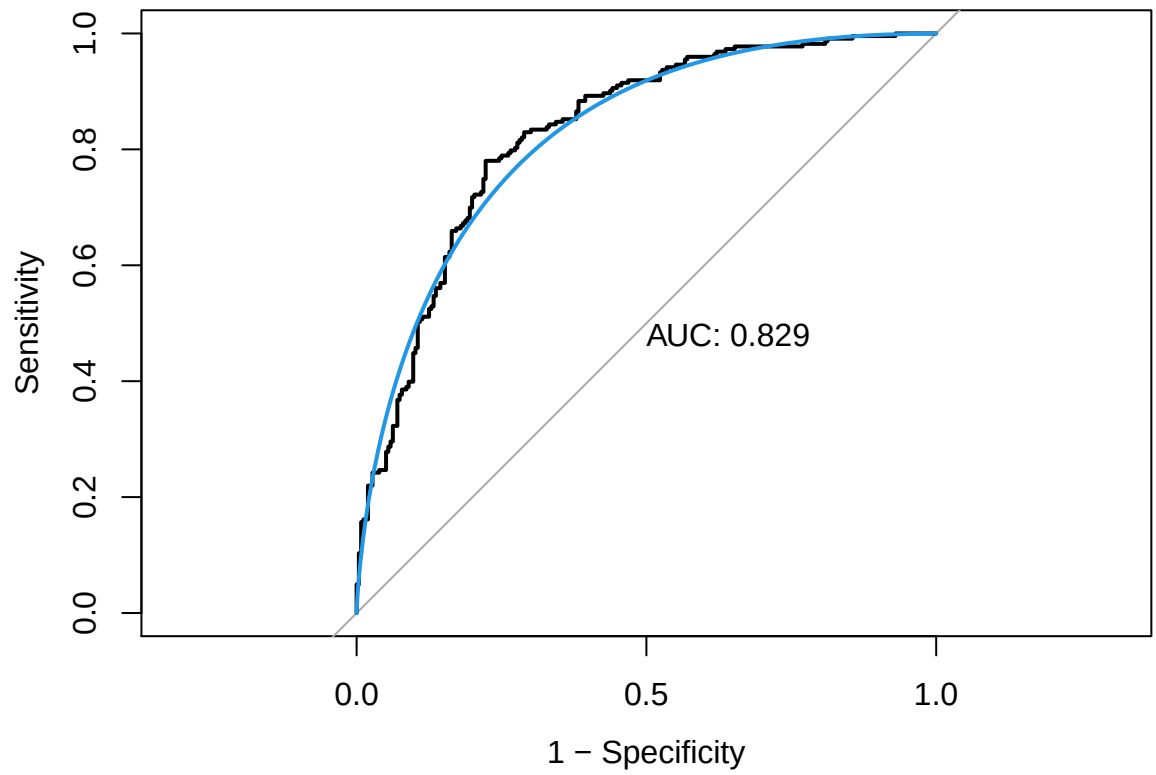
```
##             Sensitivity : 0.7812
##             Specificity : 0.7354
##          Pos Pred Value : 0.7722
##          Neg Pred Value : 0.7455
##              Prevalence : 0.5344
##          Detection Rate : 0.4175
##    Detection Prevalence : 0.5407
##       Balanced Accuracy : 0.7583
##
##        'Positive' Class : good
##
```

```r
# Plot the test ROC
roc.gam <- roc(testData$qual, test.pred.prob3$good)
```

```
## Setting levels: control = good, case = poor
```

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

```r
plot(roc.gam, legacy.axes = TRUE, print.auc = TRUE)
plot(smooth(roc.gam), col = 4, add = TRUE)
```




```r
## Test error and train error
error.test.gam <- mean(testData$qual != test.pred3)
```

```r
train.pred.prob.gam <- predict(model.gam, newdata = trainData,
                           type = "prob")
train.pred.gam <- rep("good", length(train.pred.prob.gam$good))
train.pred.gam[train.pred.prob.gam$good < 0.5] <- "poor"
error.trian.gam <- mean(trainData$qual != train.pred.gam)
```

```r
### MARS
set.seed(1)
model.mars <- train(x = trainData %>% dplyr::select(-qual),
                    y = trainData$qual,
                    method = "earth",
                    tuneGrid = expand.grid(degree = 1:3,
                                           nprune = 2:23),
                    metric = "ROC",
                    trControl = ctrl)
```

```
## Loading required package: earth
```

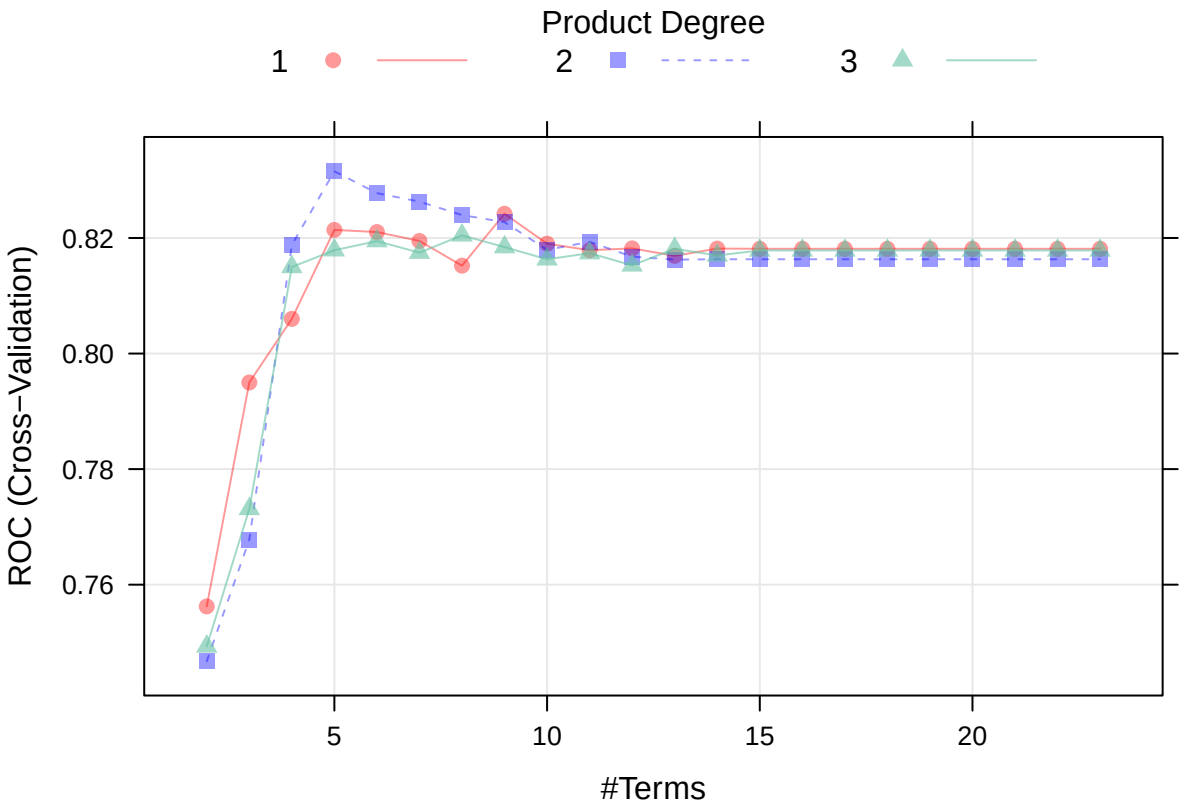
```
## Loading required package: Formula
```

```
## Loading required package: plotmo
```

```
## Loading required package: plotrix
```

```
## Loading required package: TeachingDemos
```

```r
plot(model.mars)
```



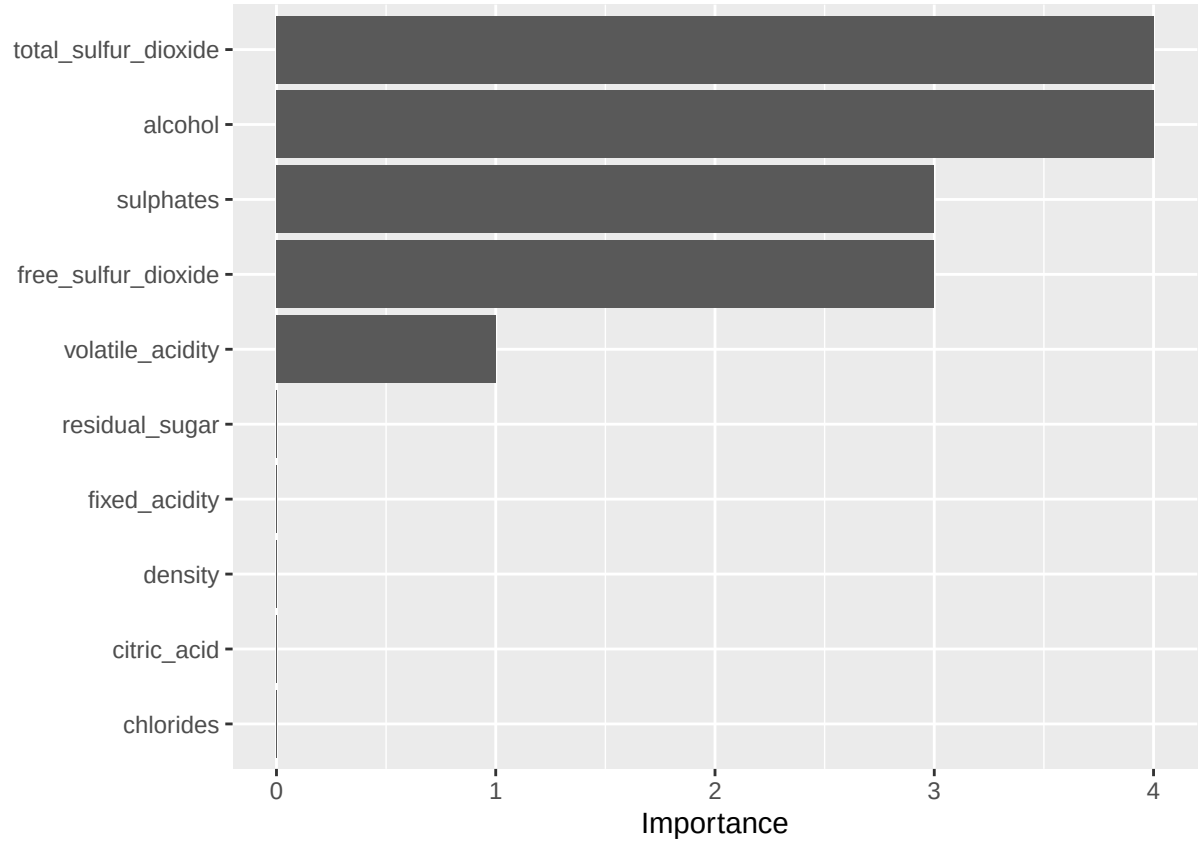

```
model.mars$bestTune
```

```
##    nprune degree
## 26      5      2
```

```
model.mars$finalModel
```

```
## GLM (family binomial, link logit):
##  nulldev   df       dev   df   devratio     AIC iters converged
##  1547.21 1119   1118.07 1115      0.277    1128     5         1
##
## Earth selected 5 of 22 terms, and 5 of 11 predictors (nprune=5)
## Termination condition: Reached nk 23
## Importance: total_sulfur_dioxide, alcohol, free_sulfur_dioxide, sulphates, ...
## Number of terms at each degree of interaction: 1 2 2
## Earth GCV 0.1721529    RSS 189.0425    GRSq 0.3092675    RSq 0.3215578
```

```
vip(model.mars$finalModel)
```




```
# Building confusion matrix
test.pred.prob4 <- predict(model.mars, newdata = testData,
                           type = "prob")
test.pred4 <- rep("good", length(test.pred.prob4$good))
test.pred4[test.pred.prob4$good < 0.5] <- "poor"

confusionMatrix(data = as.factor(test.pred4),
                reference = testData$qual,
                positive = "good")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction good poor
##       good  201   63
##       poor   55  160
##
##                Accuracy : 0.7537
##                  95% CI : (0.7125, 0.7916)
##     No Information Rate : 0.5344
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.5038
##
##  Mcnemar's Test P-Value : 0.5193
##
```

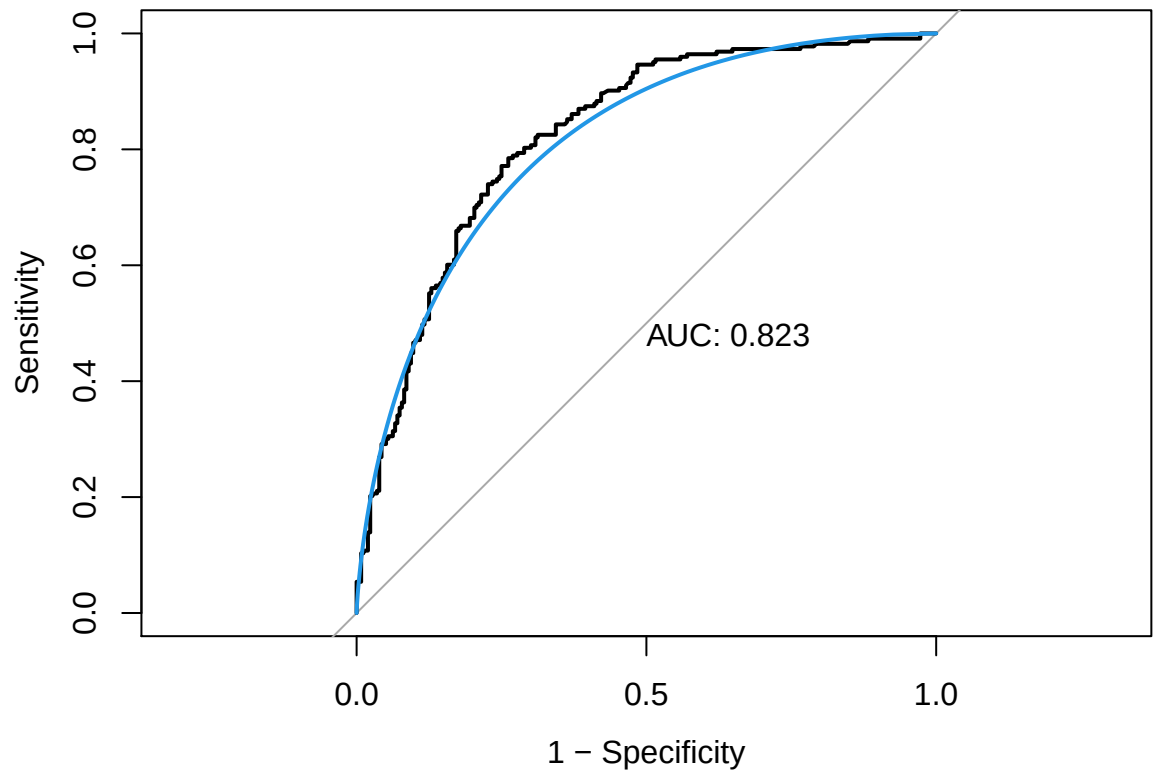
```
##             Sensitivity : 0.7852
##             Specificity : 0.7175
##          Pos Pred Value : 0.7614
##          Neg Pred Value : 0.7442
##              Prevalence : 0.5344
##          Detection Rate : 0.4196
##    Detection Prevalence : 0.5511
##       Balanced Accuracy : 0.7513
##
##        'Positive' Class : good
##
```

```
# Plot the test ROC
roc.mars <- roc(testData$qual, test.pred.prob4$good)
```

```
## Setting levels: control = good, case = poor
```

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

```
plot(roc.mars, legacy.axes = TRUE, print.auc = TRUE)
plot(smooth(roc.mars), col = 4, add = TRUE)
```



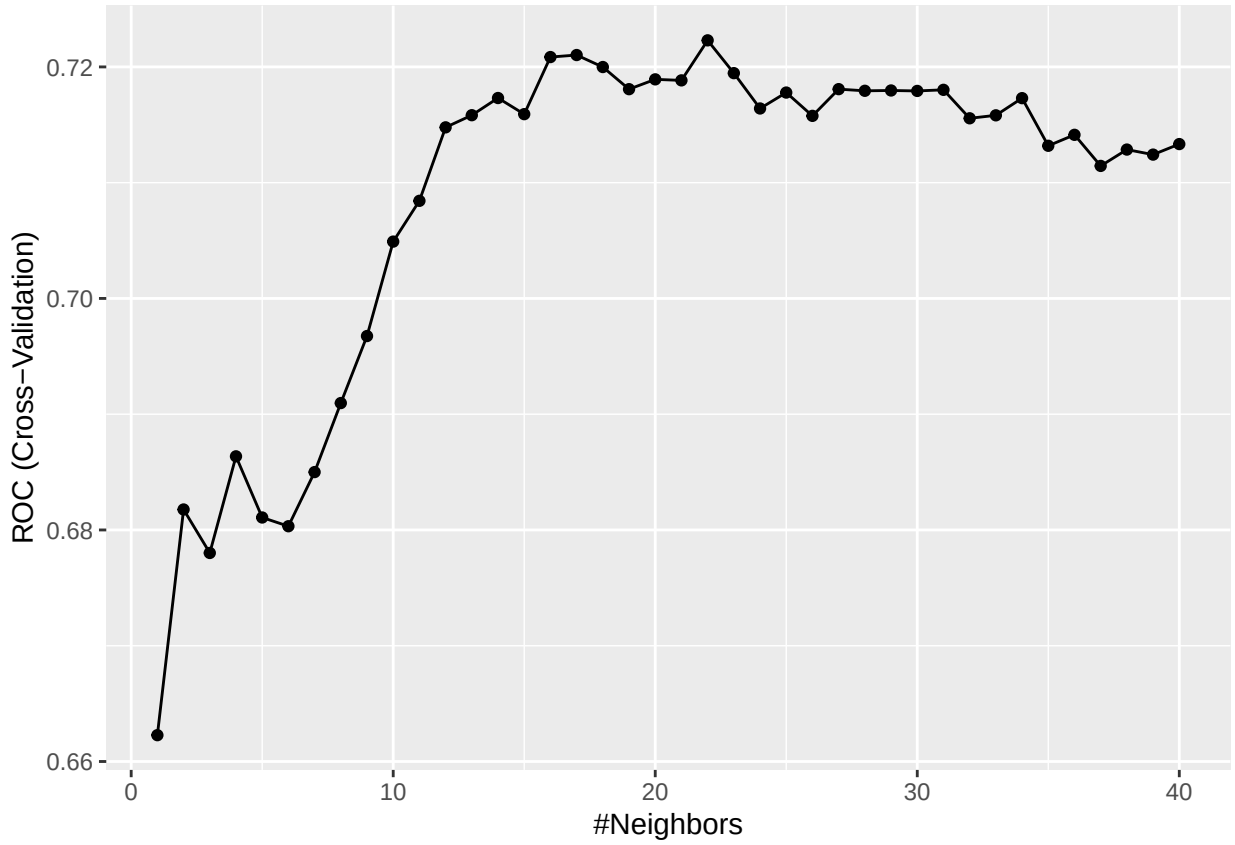

```
## Test error and train error
error.test.mars <- mean(testData$qual != test.pred4)
```

```
train.pred.prob.mars <- predict(model.mars, newdata = trainData,
                                type = "prob")
train.pred.mars <- rep("good", length(train.pred.prob.mars$good))
train.pred.mars[train.pred.prob.mars$good < 0.5] <- "poor"
error.trian.mars <- mean(trainData$qual != train.pred.mars)
```

```
### KNN
kGrid <- expand.grid(k = seq(from = 1, to = 40, by = 1))

set.seed(1)
fit.knn <- train(qual ~ .,
                 data = trainData,
                 method = "knn",
                 metric = "ROC",
                 trControl = ctrl,
                 tuneGrid = kGrid)

ggplot(fit.knn)
```




```
## The best TuneGrid
fit.knn$bestTune
```

```
##     k
## 22 22
```

```r
# Building confusion matrix
test.pred.prob7 <- predict(fit.knn, newdata = testData,
                           type = "prob")
test.pred7 <- rep("good", length(test.pred.prob7$good))
test.pred7[test.pred.prob7$good < 0.5] <- "poor"

confusionMatrix(data = as.factor(test.pred7),
                reference = testData$qual,
                positive = "good")
```

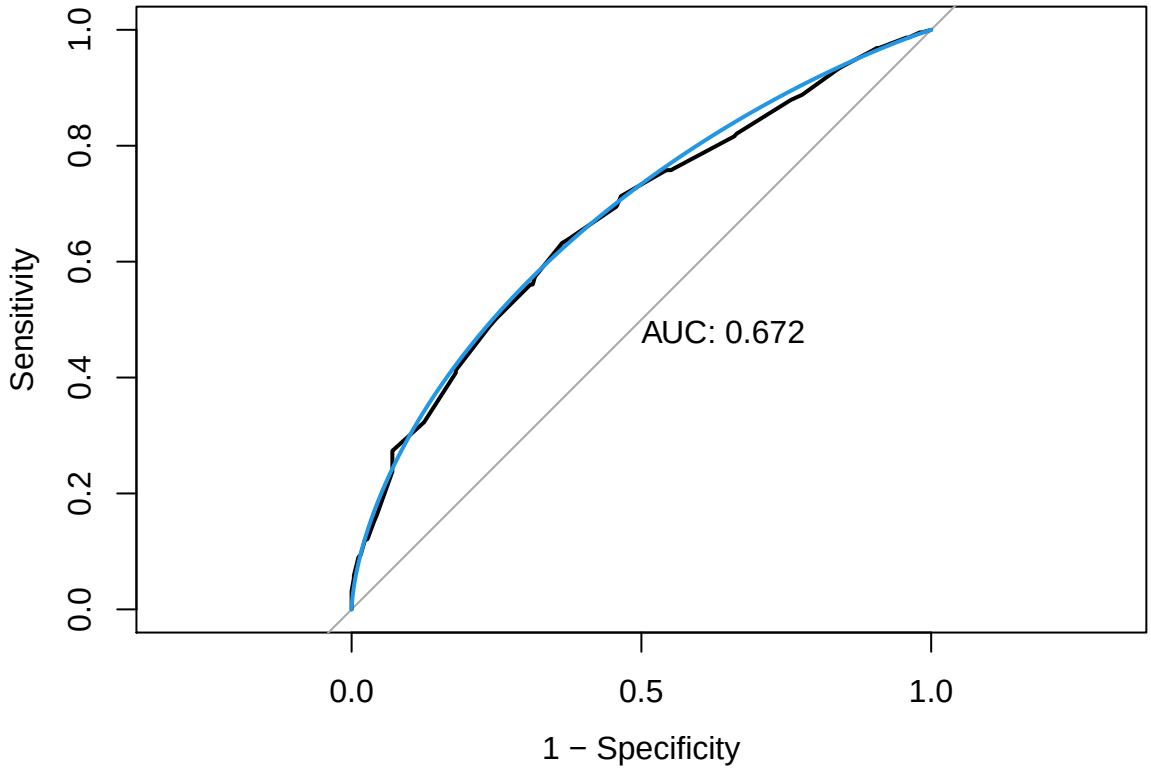
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction good poor
##       good  175   95
##       poor   81  128
##
##                Accuracy : 0.6326
##                  95% CI : (0.5876, 0.6759)
##     No Information Rate : 0.5344
##     P-Value [Acc > NIR] : 8.897e-06
##
##                   Kappa : 0.2586
##
##  Mcnemar's Test P-Value : 0.3271
##
##             Sensitivity : 0.6836
##             Specificity : 0.5740
##          Pos Pred Value : 0.6481
##          Neg Pred Value : 0.6124
##              Prevalence : 0.5344
##          Detection Rate : 0.3653
##    Detection Prevalence : 0.5637
##       Balanced Accuracy : 0.6288
##
##        'Positive' Class : good
##
```

```r
# Plot the test ROC
roc.knn <- roc(testData$qual, test.pred.prob7$good)
```

```
## Setting levels: control = good, case = poor
```

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

```r
plot(roc.knn, legacy.axes = TRUE, print.auc = TRUE)
plot(smooth(roc.knn), col = 4, add = TRUE)
```




```
## Test error and train error
error.test.knn <- mean(testData$qual != test.pred7)

train.pred.prob.knn <- predict(fit.knn, newdata = trainData,
                               type = "prob")
train.pred.knn <- rep("good", length(train.pred.prob.knn$good))
train.pred.knn[train.pred.prob.knn$good < 0.5] <- "poor"
error.trian.knn <- mean(trainData$qual != train.pred.knn)
```

Model Comparison

Training ROC

```
resamp = resamples(list(logistic = model.glm,
                        penalized_logistic = model.glmn,
                        GAM = model.gam,
                        MARS = model.mars,
                        knn = fit.knn
                        ), metric = "accuracy" )

summary(resamp)
```

```
##
## Call:
```

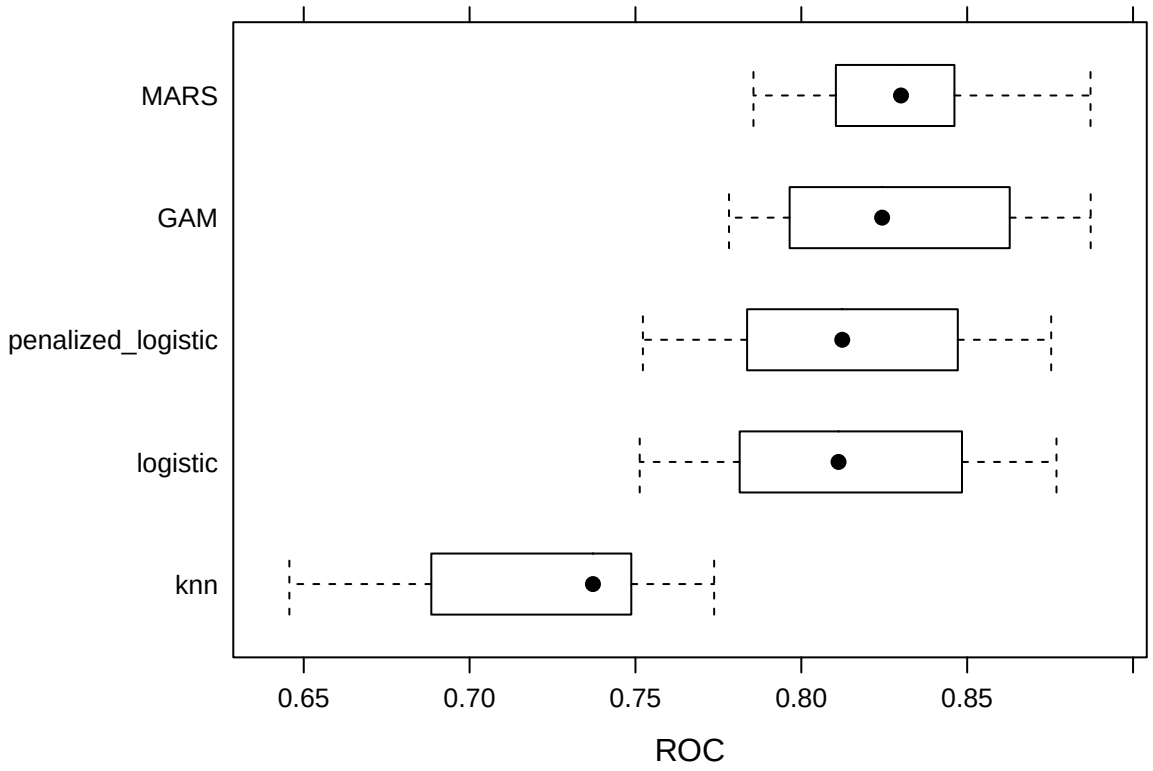
```
## summary.resamples(object = resamp)
##
## Models: logistic, penalized_logistic, GAM, MARS, knn
## Number of resamples: 10
##
## ROC
##                            Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## logistic           0.7512821 0.7829327 0.8112179 0.8141979 0.8472888 0.8769231
## penalized_logistic 0.7522436 0.7849359 0.8123397 0.8146170 0.8463454 0.8753205
## GAM                0.7782051 0.7967949 0.8243590 0.8302269 0.8598512 0.8872229
## MARS               0.7855769 0.8127653 0.8300481 0.8315484 0.8432692 0.8871795
## knn                0.6456731 0.6965946 0.7371863 0.7222954 0.7475742 0.7737179
##                    NA's
## logistic              0
## penalized_logistic    0
## GAM                   0
## MARS                  0
## knn                   0
##
## Sens
##                            Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## logistic           0.6333333 0.7083333 0.7416667 0.7498023 0.8000000 0.8813559
## penalized_logistic 0.6333333 0.7083333 0.7416667 0.7498023 0.8000000 0.8813559
## GAM                0.6666667 0.7166667 0.7750000 0.7647458 0.7958333 0.8474576
## MARS               0.7333333 0.7666667 0.7833333 0.7981638 0.8000000 0.8983051
## knn                0.5666667 0.6500000 0.7166667 0.7046328 0.7666667 0.8000000
##                    NA's
## logistic              0
## penalized_logistic    0
## GAM                   0
## MARS                  0
## knn                   0
##
## Spec
##                            Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## logistic           0.5769231 0.6826923 0.7500000 0.7312409 0.7800254 0.8461538
## penalized_logistic 0.5769231 0.6875000 0.7523585 0.7350871 0.7836538 0.8461538
## GAM                0.6153846 0.6923077 0.7115385 0.7235849 0.7464623 0.8461538
## MARS               0.6346154 0.6746190 0.7019231 0.7160015 0.7548077 0.8076923
## knn                0.5384615 0.6027758 0.6346154 0.6488026 0.6875000 0.7692308
##                    NA's
## logistic              0
## penalized_logistic    0
## GAM                   0
## MARS                  0
## knn                   0
```

```
comparison = summary(resamp)$statistics$ROC
r_square = summary(resamp)$statistics$Rsquared

knitr::kable(comparison[,1:6])
```

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
logistic	0.7512821	0.7829327	0.8112179	0.8141979	0.8472888	0.8769231
penalized_logistic	0.7522436	0.7849359	0.8123397	0.8146170	0.8463454	0.8753205
GAM	0.7782051	0.7967949	0.8243590	0.8302269	0.8598512	0.8872229
MARS	0.7855769	0.8127653	0.8300481	0.8315484	0.8432692	0.8871795
knn	0.6456731	0.6965946	0.7371863	0.7222954	0.7475742	0.7737179

```
bwplot(resamp, metric = "ROC")
```



Test and train classification error

```
Model_Name <- c("logistic regression", "penalized logistic regression", "GAM", "MARS", "KNN")
Train_Error <- c(error.trian.glm,error.trian.glmn,error.trian.gam,error.trian.mars,error.trian.knn)
Test_Error <- c(error.test.glm, error.test.glmn, error.test.gam, error.test.mars, error.test.knn)
Test_ROC = c(0.818, 0.818, 0.829, 0.823, 0.672)


df <- data.frame(Model_Name, Train_Error, Test_Error, Test_ROC)

knitr::kable(df)
```

Model_Name	Train_Error	Test_Error	Test_ROC
logistic regression	0.2553571	0.2505219	0.818
penalized logistic regression	0.2553571	0.2463466	0.818
GAM	0.2151786	0.2400835	0.829
MARS	0.2321429	0.2463466	0.823
KNN	0.2928571	0.3674322	0.672