Appendix (Modeling)

0. Data cleaning and take logs

```
# Load dataset
data = read.csv("winequality-red.csv")
data = na.omit(data)

# Take log
data$fixed.acidity = log(data$fixed.acidity)
data$residual.sugar = log(data$residual.sugar)
data$free.sulfur.dioxide = log(data$free.sulfur.dioxide)
data$total.sulfur.dioxide = log(data$total.sulfur.dioxide)

# Check NAs
data = na.omit(data)

# Split the data set into training and testing
train = sample(1:nrow(data), size=0.7*nrow(data))
test = -train
data_train = data[train, ]
data_test = data[test, ]
```

1. Predicting quality

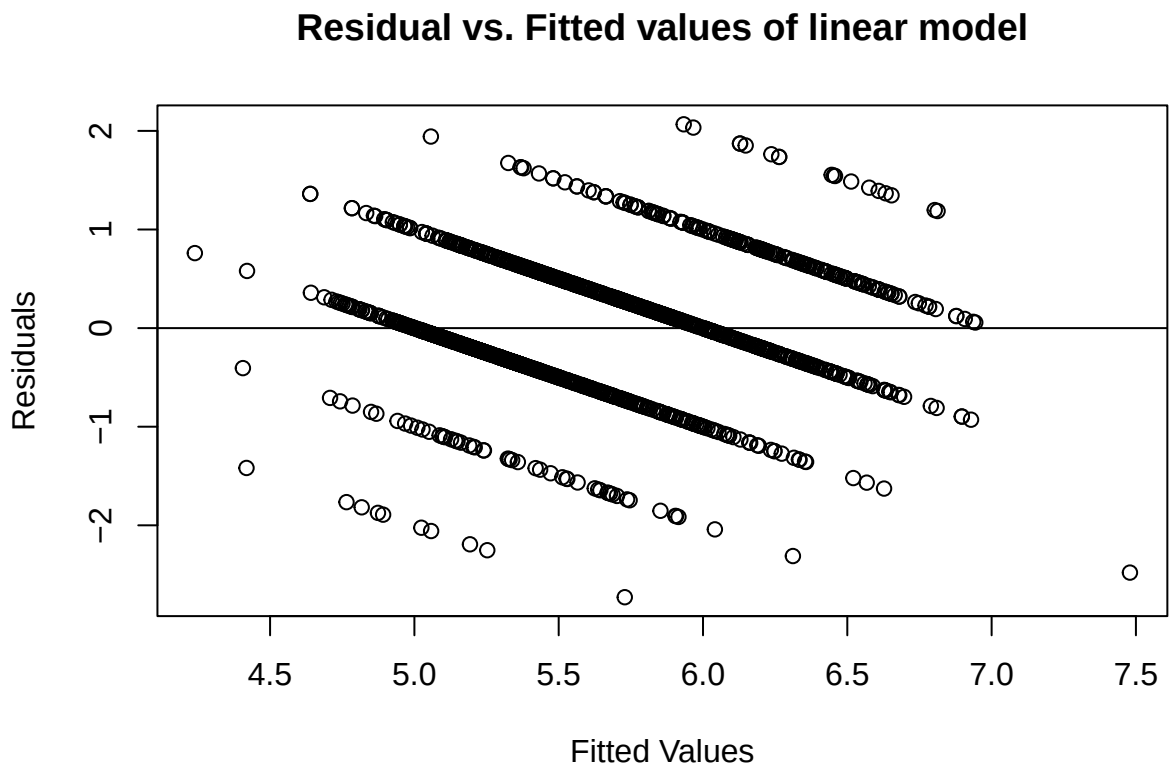
1.1. Linear Regression

```
# Perform Linear Regression
linear <- lm(quality~., data=data)
summary(linear)
```

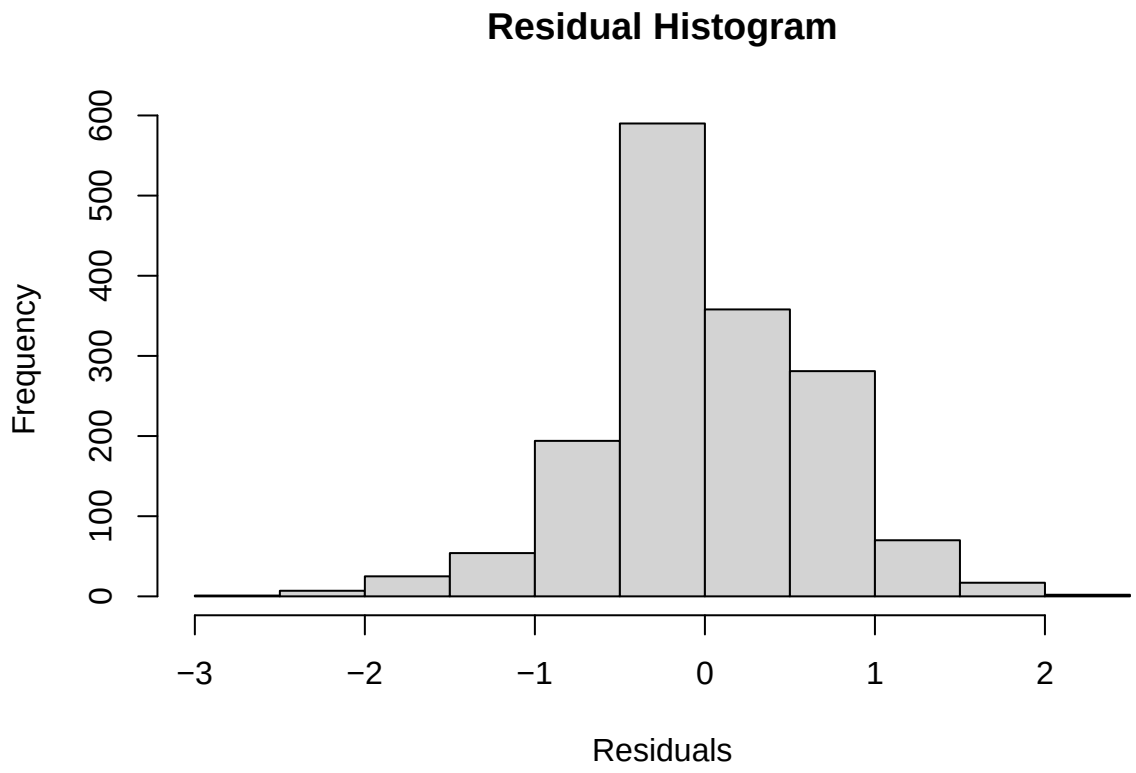
```
##
## Call:
## lm(formula = quality ~ ., data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.72895 -0.35917 -0.06067  0.44884  2.06679
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)           31.29781   22.28976   1.404 0.160475
```

```
## fixed.acidity          0.46253    0.23219   1.992 0.046540 *
## volatile.acidity      -1.12564    0.11978  -9.397  < 2e-16 ***
## citric.acid           -0.27120    0.14490  -1.872 0.061439 .
## residual.sugar         0.06921    0.06339   1.092 0.275103
## chlorides             -1.68676    0.41611  -4.054 5.29e-05 ***
## free.sulfur.dioxide    0.10798    0.04009   2.693 0.007148 **
## total.sulfur.dioxide  -0.15476    0.04080  -3.793 0.000155 ***
## density              -28.31356   22.98804  -1.232 0.218257
## pH                    -0.23638    0.19850  -1.191 0.233913
## sulphates              0.92653    0.11531   8.035 1.80e-15 ***
## alcohol                0.26870    0.02728   9.850  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6486 on 1587 degrees of freedom
## Multiple R-squared:  0.3594, Adjusted R-squared:  0.355
## F-statistic: 80.95 on 11 and 1587 DF,  p-value: < 2.2e-16
```

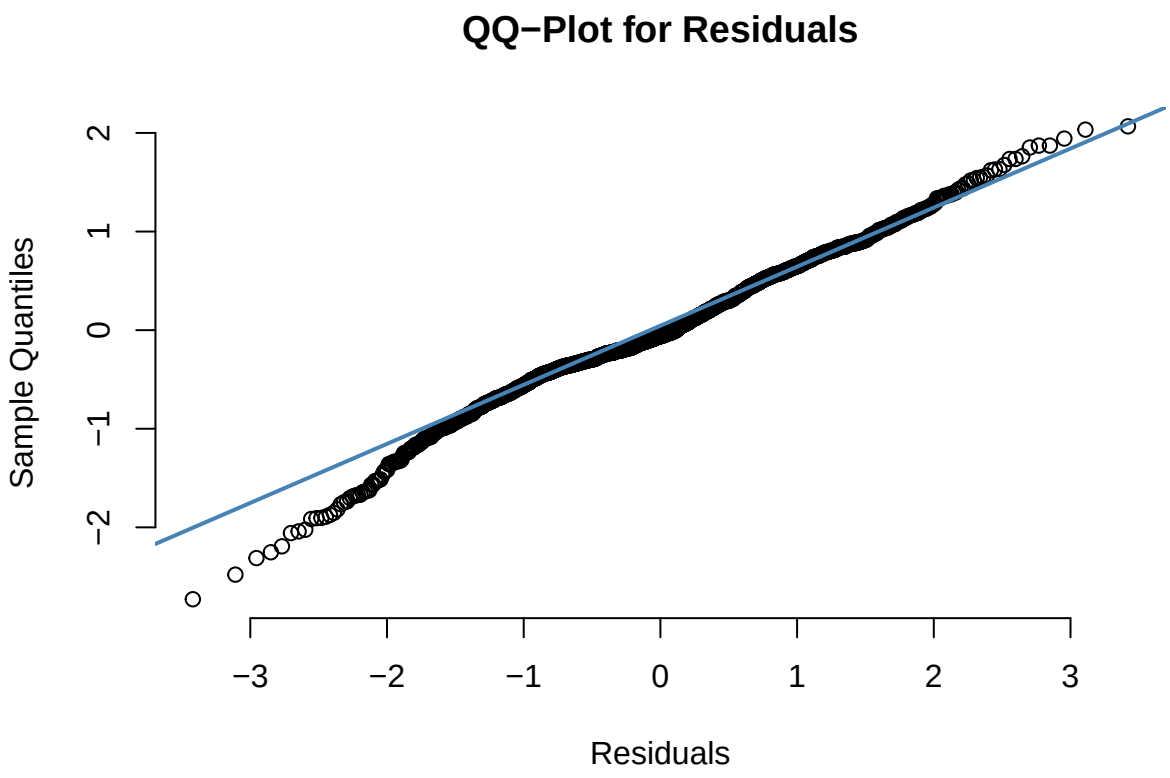
```r
# Check assumptions:
# 1. Linearity/functional form
plot(linear$fitted.values, linear$residuals,
     main = "Residual vs. Fitted values of linear model",
     xlab = "Fitted Values", ylab = "Residuals")
abline(0,0)
```



```r
# 2. Normality
hist(linear$residuals, xlab = "Residuals", main="Residual Histogram")
```




```r
qqnorm(linear$residuals, frame=FALSE, xlab = "Residuals", main="QQ-Plot for Residuals")
qqline(linear$residuals, col="steelblue", lwd=2)
```




```
shapiro.test(linear$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  linear$residuals
## W = 0.9899, p-value = 4.433e-09
```

```
ks.test(linear$residuals, "pnorm")
```

```
## Warning in ks.test.default(linear$residuals, "pnorm"): ties should not be
## present for the Kolmogorov-Smirnov test
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  linear$residuals
## D = 0.1347, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
# 3. Homoscedasticity: exists
bptest(linear)
```

```
##
```

```
##  studentized Breusch-Pagan test
##
## data:  linear
## BP = 77.178, df = 11, p-value = 5.169e-12
```

```
# 4. Uncorrelated error
dwtest(formula=linear, alternative="two.sided")
```

```
##
##  Durbin-Watson test
##
## data:  linear
## DW = 1.7499, p-value = 4.031e-07
## alternative hypothesis: true autocorrelation is not 0
```

```
# Check mlticollinearity
vif(linear)
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##             8.144082             1.747602             3.026655
##       residual.sugar            chlorides  free.sulfur.dioxide
##             1.950421             1.457004             2.867376
## total.sulfur.dioxide              density                   pH
##             3.147184             7.150754             3.567879
##            sulphates              alcohol
##             1.451209             3.210131
```

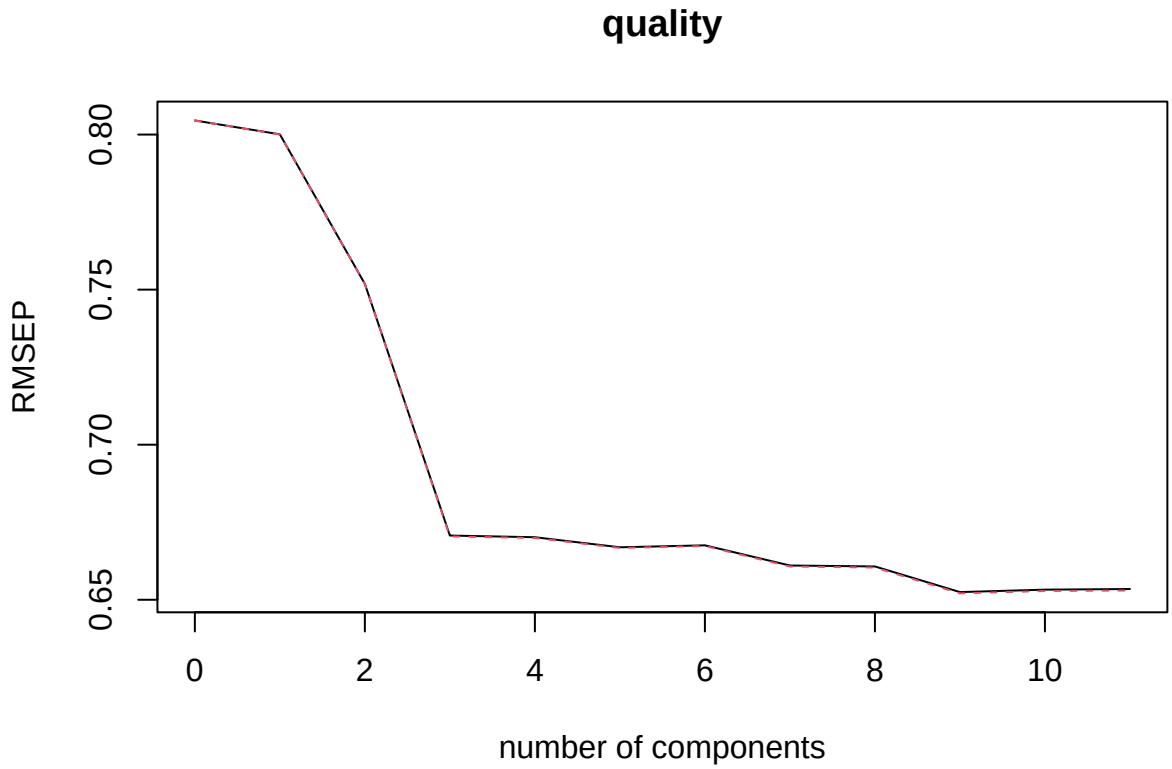
1.2. Principal Component Regression

```
# Data preprocessing
xtrain = model.matrix(quality~., data_train)[,-12]
ytrain = data_train$quality
xtest = model.matrix(quality~., data_test)[,-12]
ytest = data_test$quality

# Perform PCR
pcr_model <- pcr(quality~., data=data_train, scale=TRUE, validation="CV")
summary(pcr_model)
```

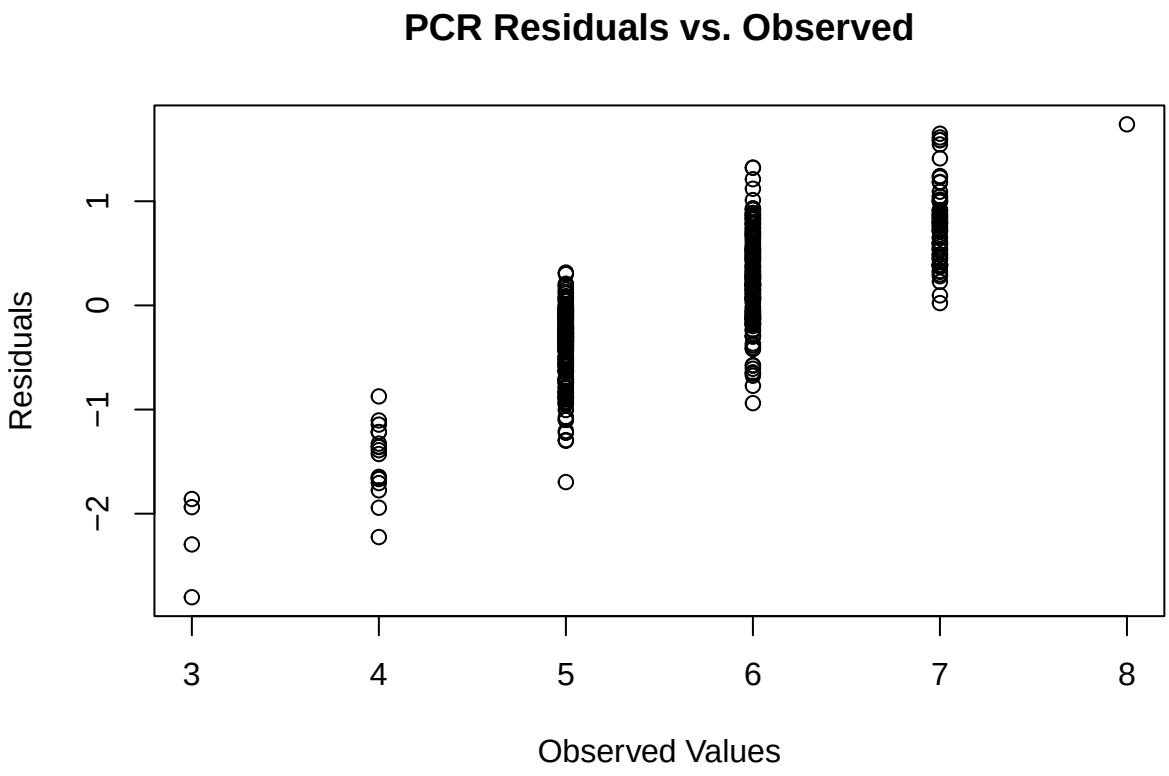
```
## Data:    X dimension: 1119 11
##  Y dimension: 1119 1
## Fit method: svdpc
## Number of components considered: 11
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.8045   0.8001   0.7519   0.6707   0.6702   0.6669   0.6676
## adjCV       0.8045   0.8000   0.7518   0.6704   0.6699   0.6667   0.6673
##        7 comps  8 comps  9 comps  10 comps  11 comps
```

```
## CV       0.6611   0.6607   0.6525    0.6533    0.6535
## adjCV    0.6607   0.6604   0.6521    0.6528    0.6530
##
## TRAINING: % variance explained
##          1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X         28.146    46.30    60.43    71.33    80.54    86.67    92.05    95.71
## quality    1.339    13.08    31.03    31.20    32.01    32.02    33.61    33.81
##          9 comps  10 comps  11 comps
## X          98.05     99.50    100.00
## quality    35.68     35.68     35.75
```

```r
validationplot(pcr_model) # ncomp = 9 may be the best choice
```




```r
pcr_pred = predict(pcr_model, data_test, ncomp=9)

# Plot Residuals
residuals_pcr <- ytest - pcr_pred
plot(ytest, residuals_pcr, main = "PCR Residuals vs. Observed",
     xlab = "Observed Values", ylab = "Residuals")
```




```
mean(abs(pcr_pred - ytest)) # MAE of PCR model
```

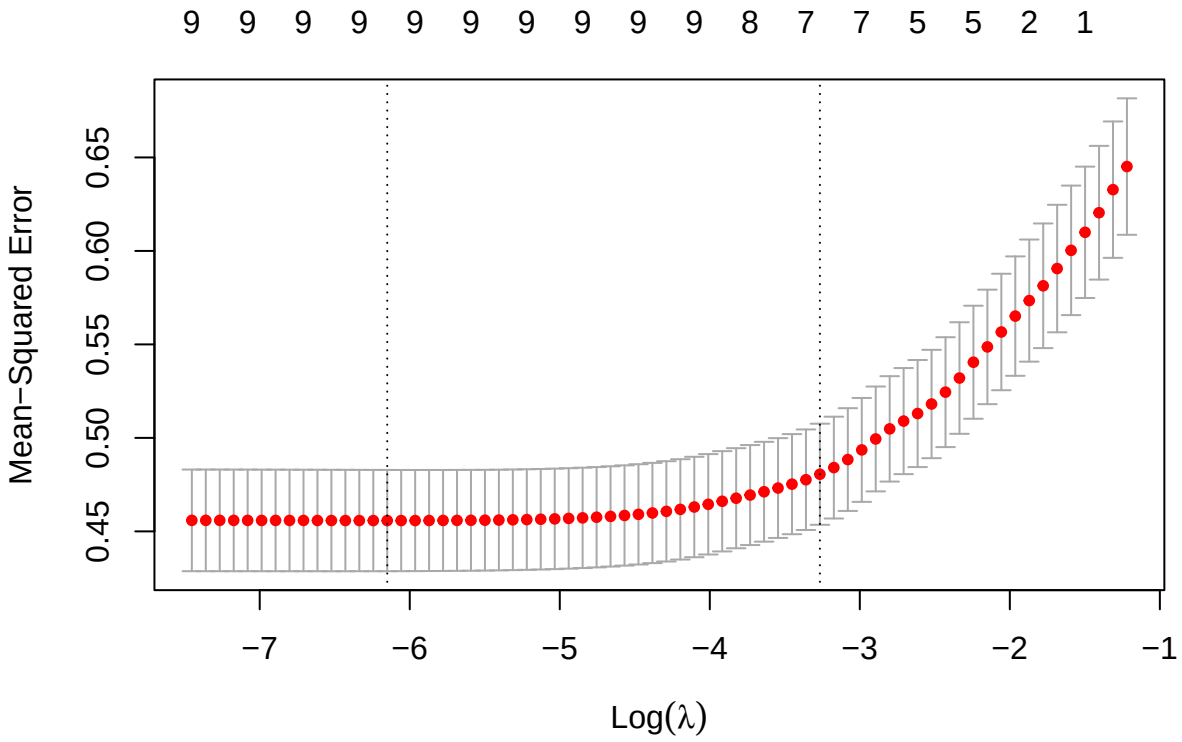
```
## [1] 0.5025666
```

```
mean((pcr_pred - ytest)^2) # MSE of PCR model
```

```
## [1] 0.42845
```

1.3. Lasso

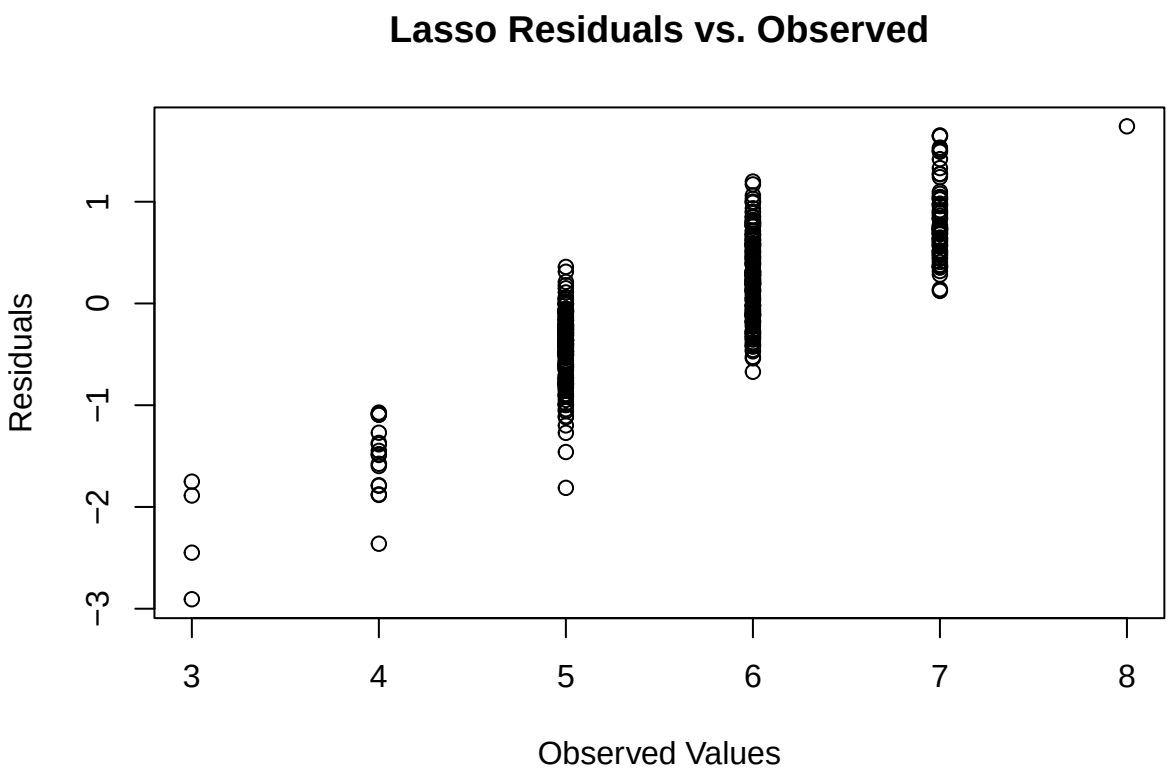
```
cv_out <- cv.glmnet(xtrain, ytrain, alpha=1)
lasso_lambda <- cv_out$lambda.min
plot(cv_out)
```




```
lasso_model <- glmnet(xtrain, ytrain, alpha=1, lambda=lasso_lambda)
coef(lasso_model)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                s0
## (Intercept)          186.4576300
## (Intercept)            .
## fixed.acidity          1.3941526
## volatile.acidity      -0.9193499
## citric.acid            .
## residual.sugar         0.3561764
## chlorides             -2.2021823
## free.sulfur.dioxide    0.1501710
## total.sulfur.dioxide  -0.2149384
## density              -186.4419841
## pH                     0.5787580
## sulphates              1.4406617
```

```
lasso_pred = predict(lasso_model, s=lasso_lambda, newx=xtest)

# Plot Residuals
residuals_lasso <- ytest - lasso_pred
plot(ytest, residuals_lasso, main = "Lasso Residuals vs. Observed",
     xlab = "Observed Values", ylab = "Residuals")
```




```
mean(abs(lasso_pred - ytest)) # MAE of lasso model
```

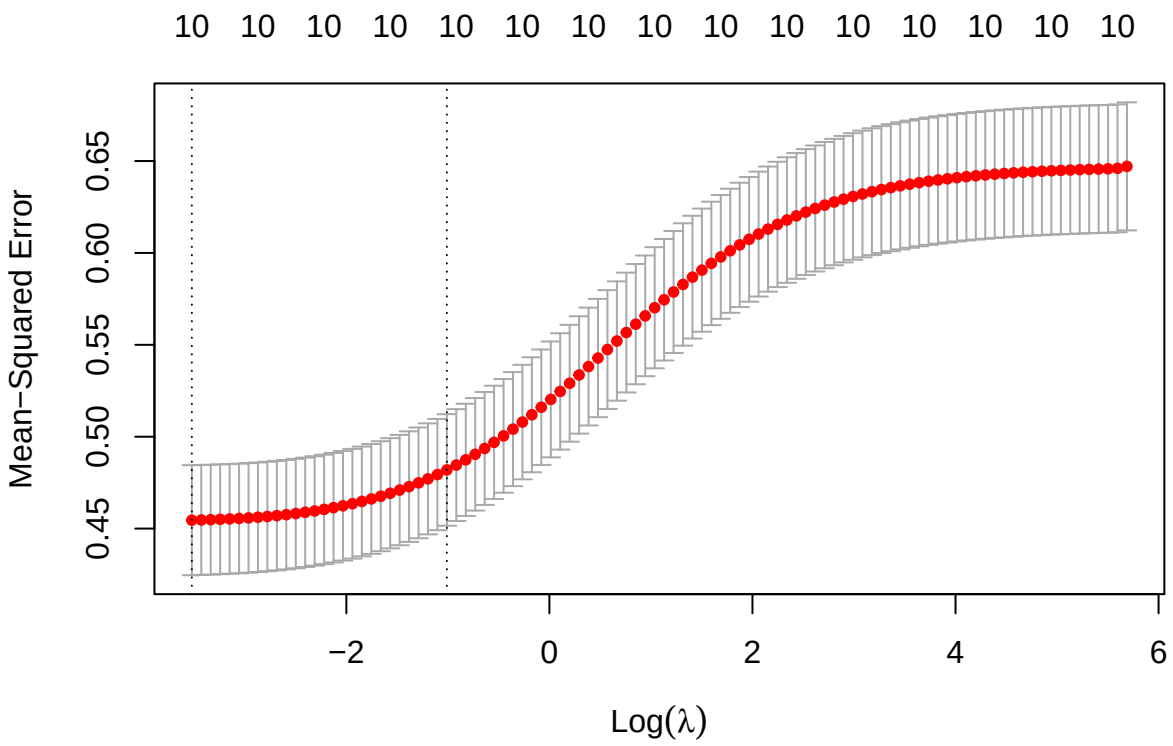
```
## [1] 0.5252256
```

```
mean((lasso_pred - ytest)^2) # MSE of lasso model
```

```
## [1] 0.4516902
```

1.4. Ridge

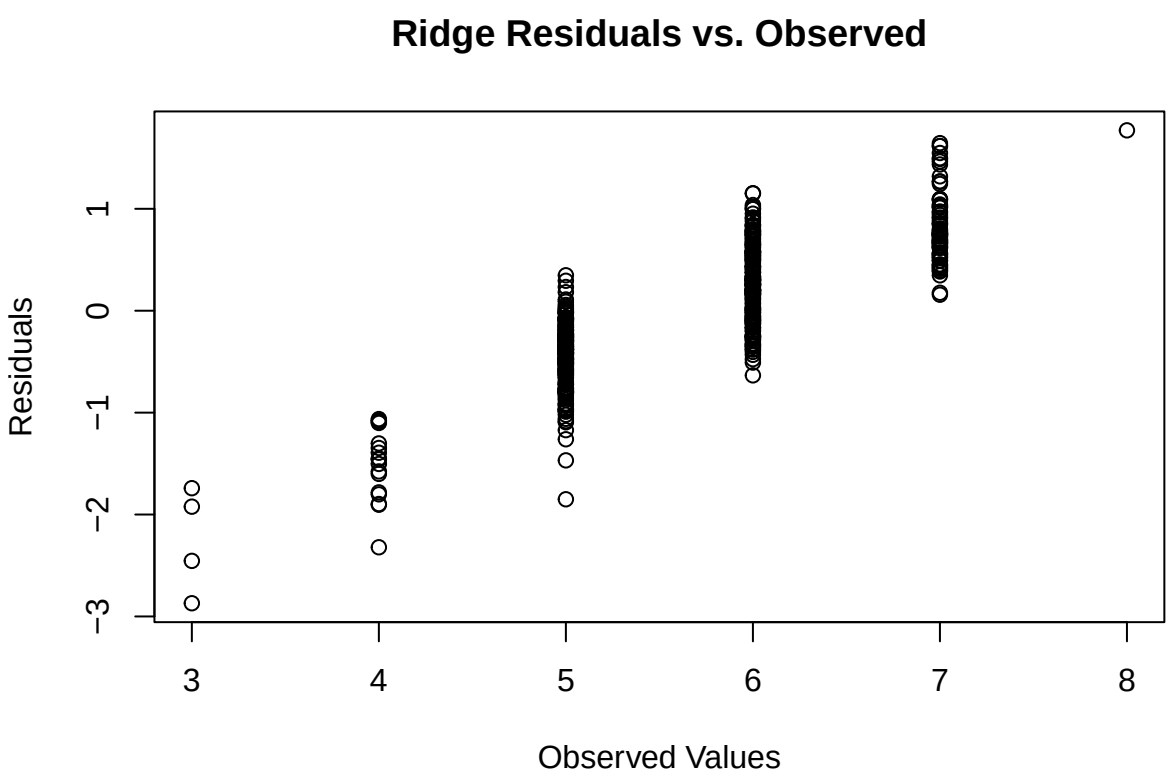
```
cv_out_2 <- cv.glmnet(xtrain, ytrain, alpha=0)
ridge_lambda <- cv_out_2$lambda.min
plot(cv_out_2)
```




```
ridge_model <- glmnet(xtrain, ytrain, alpha=0, lambda=ridge_lambda)
coef(ridge_model)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                s0
## (Intercept)          165.5168897
## (Intercept)            .
## fixed.acidity          1.1135123
## volatile.acidity      -0.9000658
## citric.acid            0.0928235
## residual.sugar         0.3188489
## chlorides             -2.3448234
## free.sulfur.dioxide    0.1472168
## total.sulfur.dioxide  -0.2193506
## density              -164.3195779
## pH                     0.4399369
## sulphates              1.3955804
```

```
ridge_pred = predict(ridge_model, s=ridge_lambda, newx=xtest)

# Plot Residuals
residuals_ridge <- ytest - ridge_pred
plot(ytest, residuals_ridge, main = "Ridge Residuals vs. Observed",
     xlab = "Observed Values", ylab = "Residuals")
```




```
mean(abs(ridge_pred - ytest)) # MAE of ridge model
```

```
## [1] 0.528796
```

```
mean((ridge_pred - ytest)^2) # MSE of ridge model
```

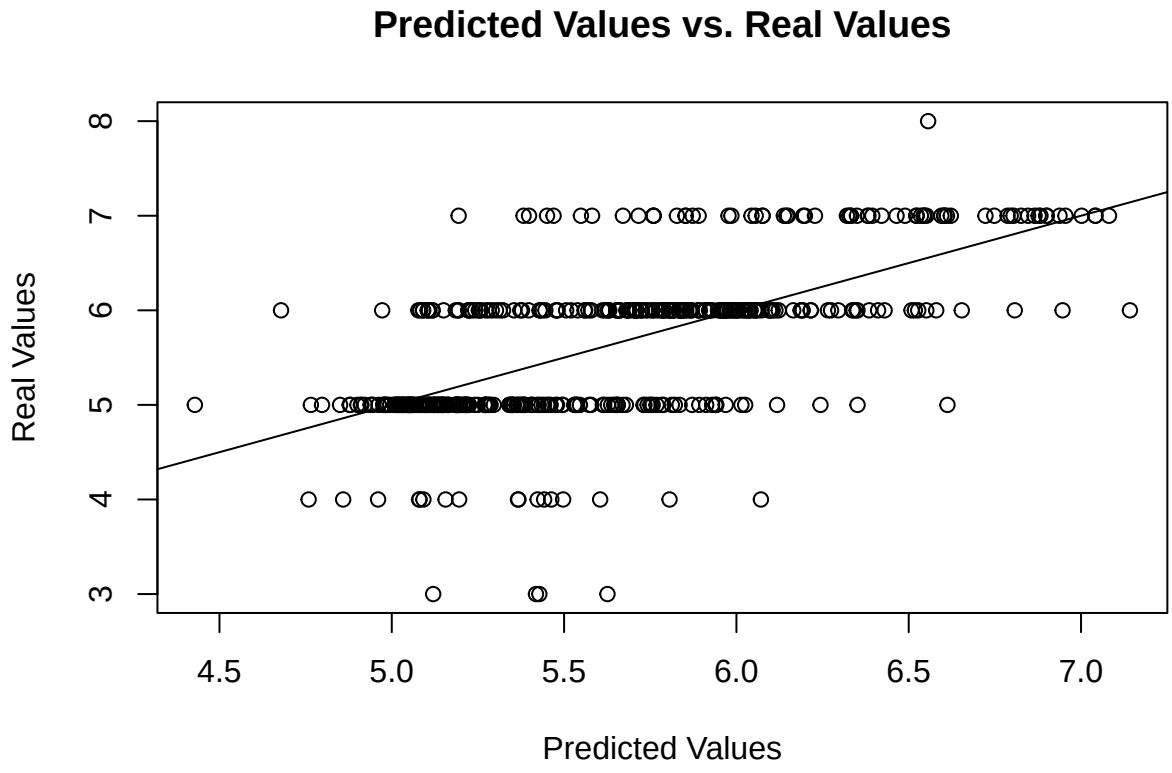
```
## [1] 0.453955
```

1.5. Random Forest

```
# Bagging (first model used all 11 variables)
rf_data = randomForest(quality~., data=data, subset=train, mtry=11, importance =TRUE)
rf_data
```

```
##
## Call:
##  randomForest(formula = quality ~ ., data = data, mtry = 11, importance = TRUE,      subset = train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 11
##
##           Mean of squared residuals: 0.335829
##                     % Var explained: 48.02
```

```r
# Make predictions
yhat_rf = predict(rf_data, newdata=data_test)
plot(yhat_rf, data_test$quality, xlab = "Predicted Values",
     ylab = "Real Values", main = "Predicted Values vs. Real Values")
abline(0,1)
```




```r
mse_rf11 = mean((yhat_rf-data_test$quality)^2) # MSE
mse_rf11
```

```
## [1] 0.3727711
```

```r
# Use mtry=5 to compare models
rf_data2 = randomForest(quality~.,data=data, subset=train, mtry=5, importance=TRUE)
rf_data2
```

```
##
## Call:
##  randomForest(formula = quality ~ ., data = data, mtry = 5, importance = TRUE,      subset = train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##           Mean of squared residuals: 0.3325311
##                     % Var explained: 48.53
```

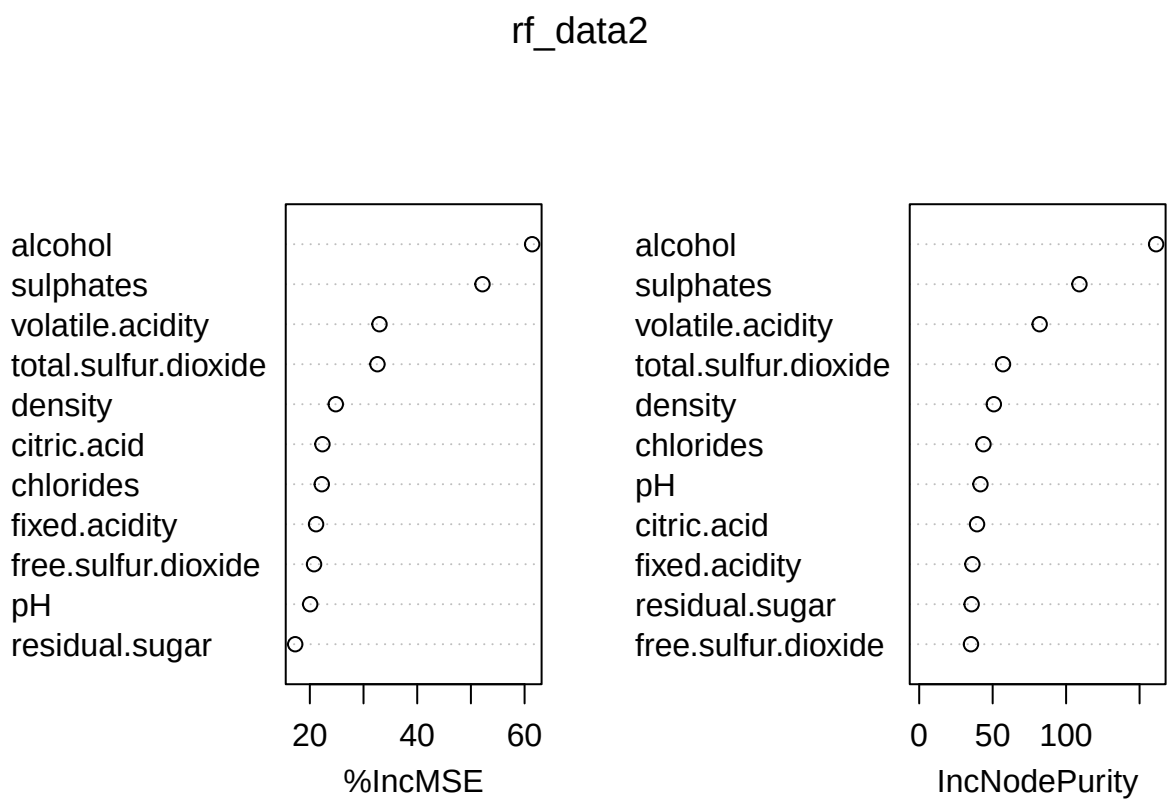
```r
yhat_rf2 = predict(rf_data2, newdata=data_test)
mse_rf5 = mean((yhat_rf2-data_test$quality)^2) # MSE
mse_rf5
```

```
## [1] 0.364782
```

```r
importance(rf_data2)
```

```
##                       %IncMSE IncNodePurity
## fixed.acidity        21.16224      36.20242
## volatile.acidity     32.97304      81.98668
## citric.acid          22.34164      39.37072
## residual.sugar       17.27260      35.67799
## chlorides            22.22601      43.74997
## free.sulfur.dioxide  20.78295      35.24376
## total.sulfur.dioxide 32.59141      57.05365
## density              24.83024      50.80715
## pH                   20.06459      41.69575
## sulphates            52.15802     109.06547
## alcohol              61.40410     161.26894
```

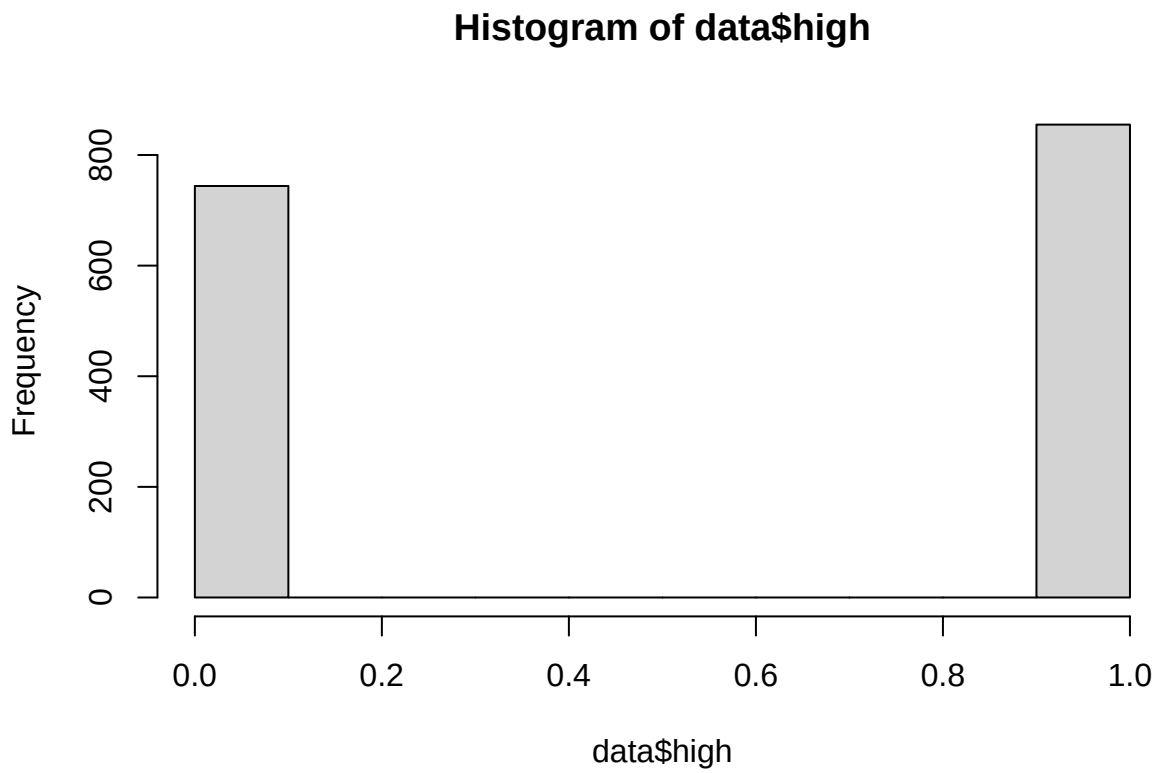
```r
varImpPlot(rf_data2)
```



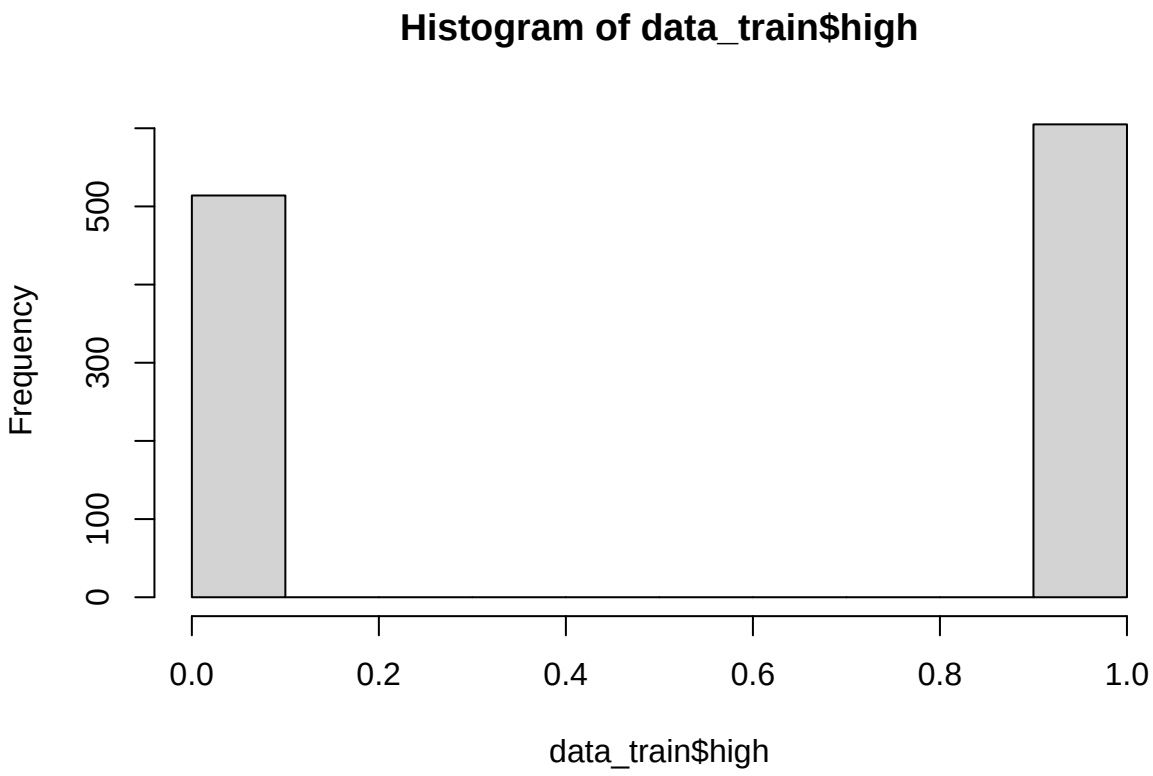
2. Classification models: Logistic, Elastic Net, & Boosting

2.0. Data processing

```
# Define high quality
high = ifelse(data$quality>=6, 1, 0)
data = data.frame(data, high)
data = data[, -12]

high = ifelse(data_train$quality>=6, 1, 0)
data_train = data.frame(data_train, high)
data_train = data_train[, -12]

high = ifelse(data_test$quality>=6, 1, 0)
data_test = data.frame(data_test, high)
data_test = data_test[, -12]

# Visualization of new variable
hist(data$high)
```




```
hist(data_train$high)
```



2.1. Logistic

```
# Logistic
logistic = glm(high~., family = "binomial", data = data)
summary(logistic)
```

```
##
## Call:
## glm(formula = high ~ ., family = "binomial", data = data)
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)           31.42720   83.96166   0.374  0.70818
## fixed.acidity          1.89432    0.87771   2.158  0.03091 *
## volatile.acidity      -3.50272    0.48375  -7.241 4.46e-13 ***
## citric.acid           -1.68260    0.54868  -3.067  0.00216 **
## residual.sugar         0.02722    0.23563   0.116  0.90802
## chlorides             -3.13063    1.56341  -2.002  0.04524 *
## free.sulfur.dioxide    0.45826    0.14865   3.083  0.00205 **
## total.sulfur.dioxide  -0.68020    0.15249  -4.461 8.17e-06 ***
## density              -43.04174   86.59136  -0.497  0.61914
## pH                     0.15974    0.74177   0.215  0.82950
## sulphates              2.73396    0.45848   5.963 2.47e-09 ***
## alcohol                0.88819    0.10789   8.233  < 2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2209.0  on 1598  degrees of freedom
## Residual deviance: 1668.5  on 1587  degrees of freedom
## AIC: 1692.5
##
## Number of Fisher Scoring iterations: 4
```

```r
# Logistic Training and Confusion Matrix
intraining <- createDataPartition(data$high, p = 0.75, list = FALSE)
training <- data[intraining,]
testing <- data[-intraining,]
train_control <- trainControl(method = "cv", number = 5)

logistic_training <- train(as.factor(high)~.,
                           data = training,
                           method = "glm",
                           family = "binomial"(link = "logit"),
                           trControl = train_control)
summary(logistic_training)
```

```
##
## Call:
## NULL
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)           43.0391   100.1214   0.430  0.66729
## fixed.acidity          1.9903     1.0374   1.919  0.05505 .
## volatile.acidity      -4.1075     0.5803  -7.078 1.46e-12 ***
## citric.acid           -1.6139     0.6337  -2.547  0.01088 *
## residual.sugar        -0.2701     0.2874  -0.940  0.34734
## chlorides             -4.1611     1.8084  -2.301  0.02139 *
## free.sulfur.dioxide    0.5022     0.1764   2.846  0.00442 **
## total.sulfur.dioxide  -0.5921     0.1810  -3.271  0.00107 **
## density              -55.2400   103.2548  -0.535  0.59266
## pH                     0.2899     0.8621   0.336  0.73670
## sulphates              2.7887     0.5355   5.207 1.91e-07 ***
## alcohol                0.8940     0.1267   7.059 1.68e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1659.2  on 1199  degrees of freedom
## Residual deviance: 1227.0  on 1188  degrees of freedom
## AIC: 1251
##
## Number of Fisher Scoring iterations: 4
```

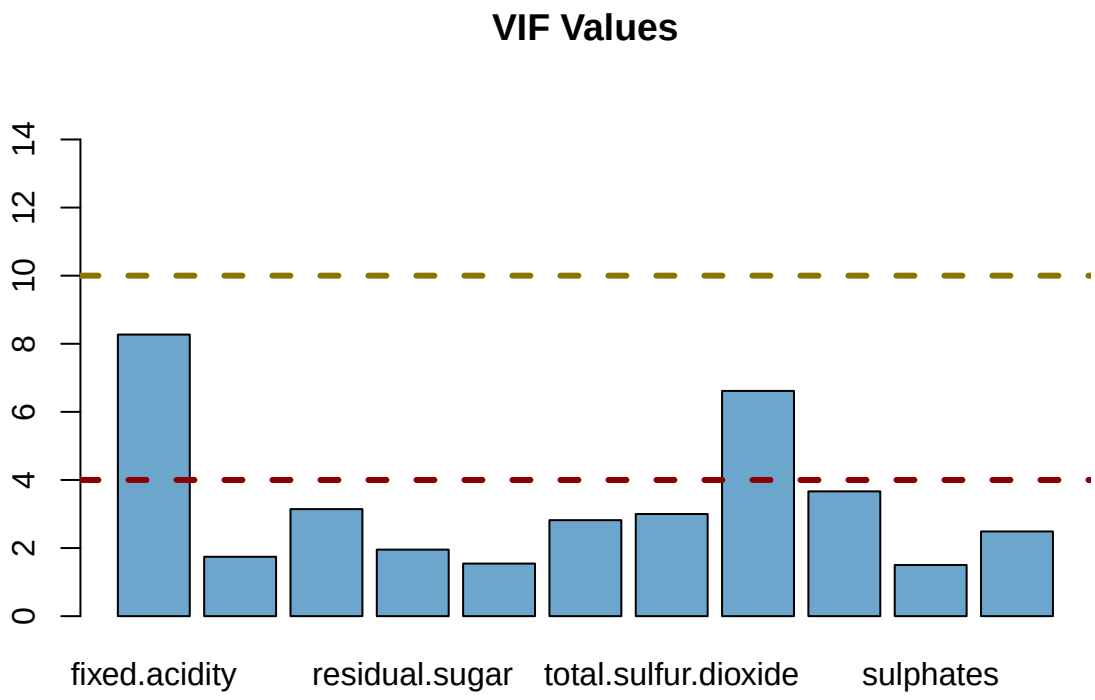
```r
pred_high <- predict(logistic_training, newdata = testing)
conf_matrix <- confusionMatrix(data = pred_high, reference = as.factor(testing$high))
conf_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 142  62
##          1  38 157
##
##                Accuracy : 0.7494
##                  95% CI : (0.7038, 0.7912)
##     No Information Rate : 0.5489
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.4998
##
##  Mcnemar's Test P-Value : 0.02145
##
##             Sensitivity : 0.7889
##             Specificity : 0.7169
##          Pos Pred Value : 0.6961
##          Neg Pred Value : 0.8051
##              Prevalence : 0.4511
##          Detection Rate : 0.3559
##    Detection Prevalence : 0.5113
##       Balanced Accuracy : 0.7529
##
##        'Positive' Class : 0
##
```

```r
# Further Analysis of Logistic
# F1-score
# Extract TP, FP, TN, FN from confusion matrix
TP <- conf_matrix[["byClass"]]["Pos Pred Value"]
FP <- conf_matrix[["byClass"]]["Neg Pred Value"]
FN <- conf_matrix[["byClass"]]["Pos Pred Value"]
TN <- conf_matrix[["byClass"]]["Neg Pred Value"]

# Calculate precision, recall, and F1 score
precision <- TP / (TP + FP)
recall <- TP / (TP + FN)
f1_score_logistic <- 2 * (precision * recall) / (precision + recall)
print(f1_score_logistic)
```

```
## Pos Pred Value
##      0.4811552
```

```r
# VIF
barplot(vif(logistic), main = "VIF Values", col = "skyblue3", ylim = c(0,15))
abline(h = 4, lwd = 3, lty = 2, col = "red4")
abline(h = 10, lwd = 3, lty = 2, col = "gold4")
```



2.2. Elastic Net

```
# Data Preprocessing
xtrain = model.matrix(high~., data_train)[,-12]
ytrain = data_train$high
xtest = model.matrix(high~., data_test)[,-12]
ytest = data_test$high
```

```
# Elastic Net: alpha = 0.2
elastic_model1 = glmnet(xtrain, ytrain, alpha=0.2)
elastic_lambda1 = cv.glmnet(xtrain, ytrain, alpha=0.2)$lambda.min
elastic_pred1 = predict(elastic_model1, s=elastic_lambda1, newx=xtest)
binary_predictions1 <- ifelse(elastic_pred1 > 0.5, 1, 0)

# Elastic Net: alpha = 0.5
elastic_model2 = glmnet(xtrain, ytrain, alpha=0.5)
elastic_lambda2 = cv.glmnet(xtrain, ytrain, alpha=0.5)$lambda.min
elastic_pred2 = predict(elastic_model2, s=elastic_lambda2, newx=xtest)
binary_predictions2 <- ifelse(elastic_pred1 > 0.5, 1, 0)

# Elastic Net: alpha = 0.8
elastic_model3 = glmnet(xtrain, ytrain, alpha=0.8)
elastic_lambda3 = cv.glmnet(xtrain, ytrain, alpha=0.8)$lambda.min
```

```r
elastic_pred3 = predict(elastic_model3, s=elastic_lambda3, newx=xtest)
binary_predictions3 <- ifelse(elastic_pred1 > 0.5, 1, 0)
```

```r
# Print results
# MAE
mean(abs(binary_predictions1 - ytest))
```

```
## [1] 0.2666667
```

```r
mean(abs(binary_predictions2 - ytest))
```

```
## [1] 0.2666667
```

```r
mean(abs(binary_predictions3 - ytest))
```

```
## [1] 0.2666667
```

```r
# MSE
mean((binary_predictions1 - ytest)^2)
```

```
## [1] 0.2666667
```

```r
mean((binary_predictions2 - ytest)^2)
```

```
## [1] 0.2666667
```

```r
mean((binary_predictions3 - ytest)^2)
```

```
## [1] 0.2666667
```

```r
# F1-score
conf_matrix <- table(binary_predictions1, ytest)
precision <- conf_matrix[2, 2] / sum(conf_matrix[, 2])
recall <- conf_matrix[2, 2] / sum(conf_matrix[2, ])
f1_score_elastic1 <- 2 * (precision * recall) / (precision + recall)
print(f1_score_elastic1)
```

```
## [1] 0.7460317
```

```r
conf_matrix <- table(binary_predictions2, ytest)
precision <- conf_matrix[2, 2] / sum(conf_matrix[, 2])
recall <- conf_matrix[2, 2] / sum(conf_matrix[2, ])
f1_score_elastic2 <- 2 * (precision * recall) / (precision + recall)
print(f1_score_elastic2)
```

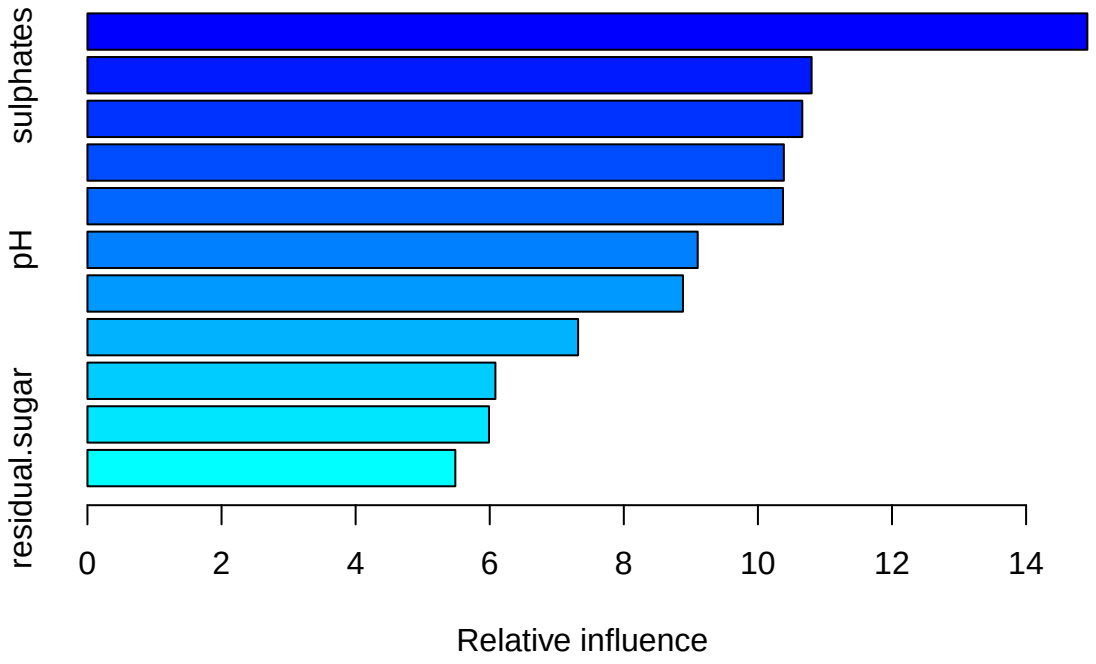
```
## [1] 0.7460317
```

```
conf_matrix <- table(binary_predictions3, ytest)
precision <- conf_matrix[2, 2] / sum(conf_matrix[, 2])
recall <- conf_matrix[2, 2] / sum(conf_matrix[2, ])
f1_score_elastic3 <- 2 * (precision * recall) / (precision + recall)
print(f1_score_elastic3)
```

```
## [1] 0.7460317
```

2.3. Boosting

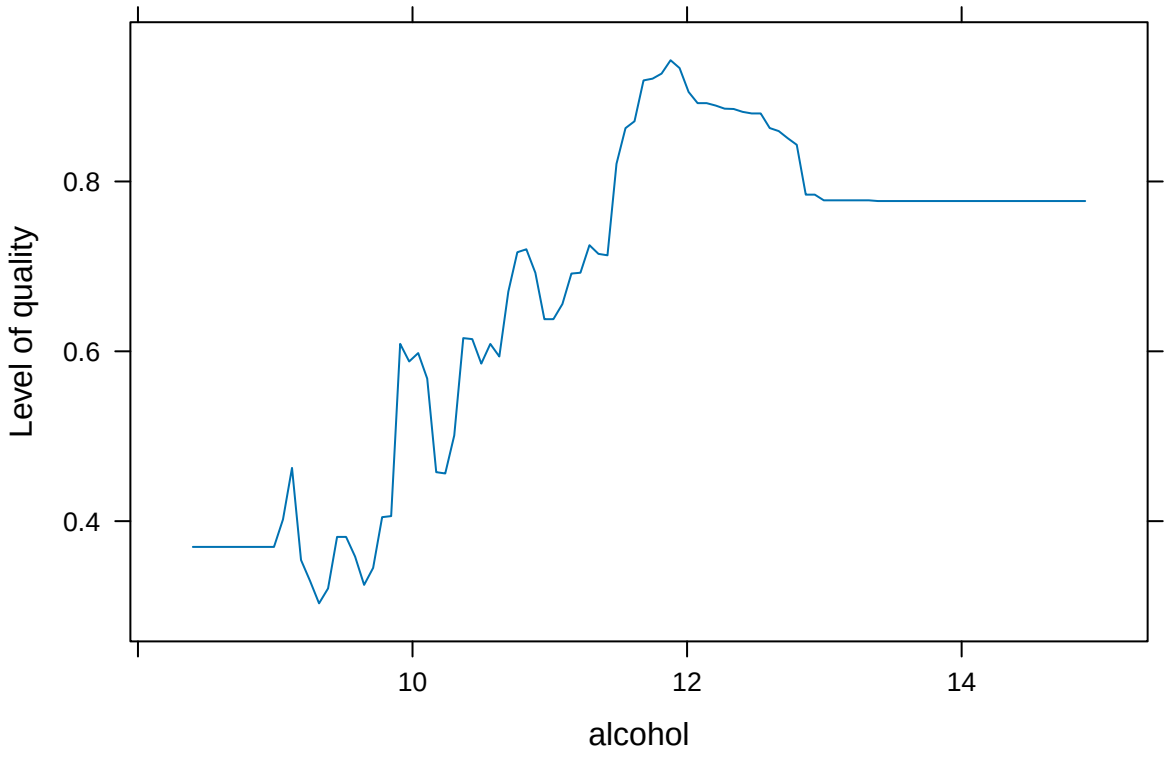
```
boost_data = gbm(high~., data=data_train, distribution="gaussian",
                 n.trees=5000, interaction.depth=4)
summary(boost_data)
```



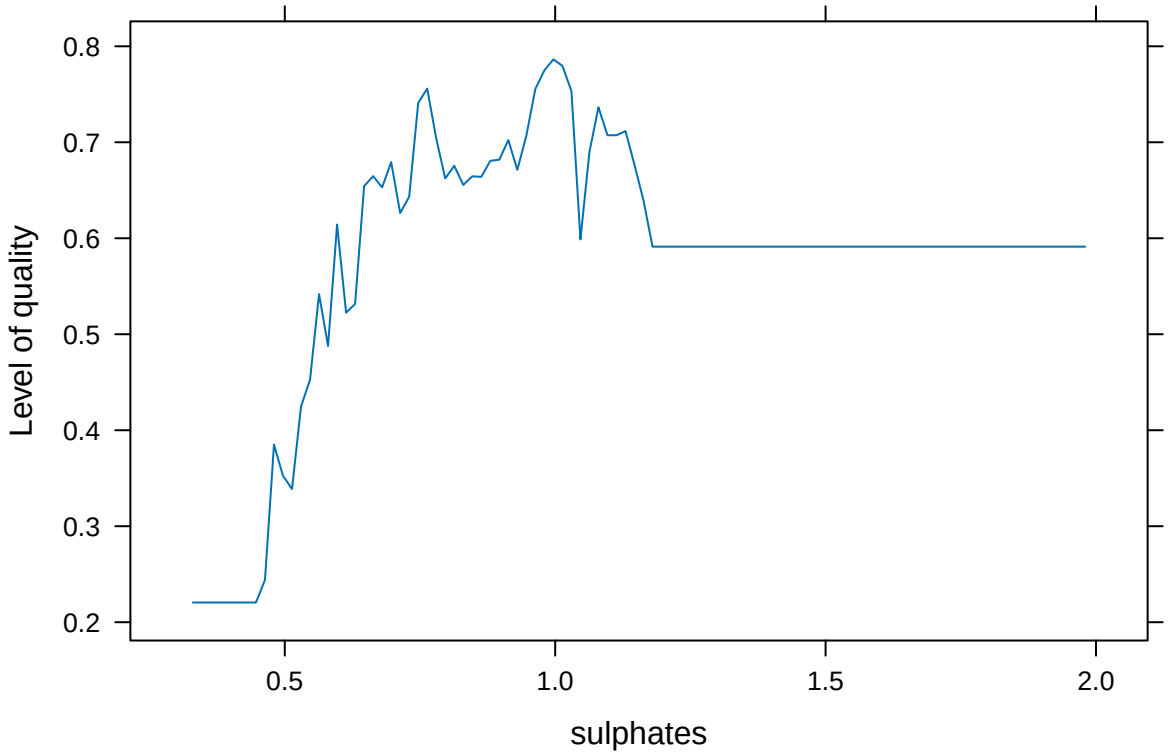

```
##                                     var   rel.inf
## alcohol                         alcohol 14.913581
## sulphates                     sulphates 10.801036
## volatile.acidity     volatile.acidity 10.661847
## chlorides                     chlorides 10.386488
## total.sulfur.dioxide total.sulfur.dioxide 10.374184
## pH                                   pH  9.101984
## density                         density  8.881995
```

```
## citric.acid                   citric.acid  7.318333
## free.sulfur.dioxide   free.sulfur.dioxide  6.084753
## fixed.acidity               fixed.acidity  5.988798
## residual.sugar           residual.sugar  5.487002
```

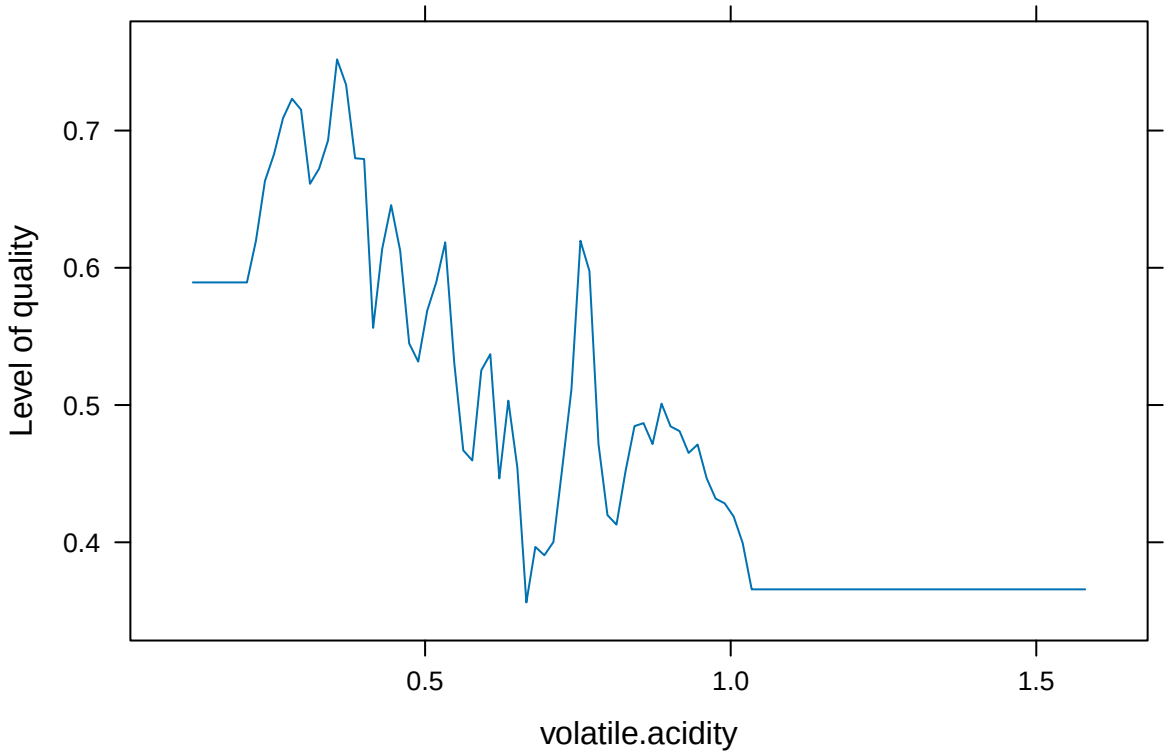
```
plot(boost_data, i="alcohol", ylab = "Level of quality")
```




```
plot(boost_data, i="sulphates", ylab = "Level of quality")
```




```
plot(boost_data, i="volatile.acidity", ylab = "Level of quality")
```



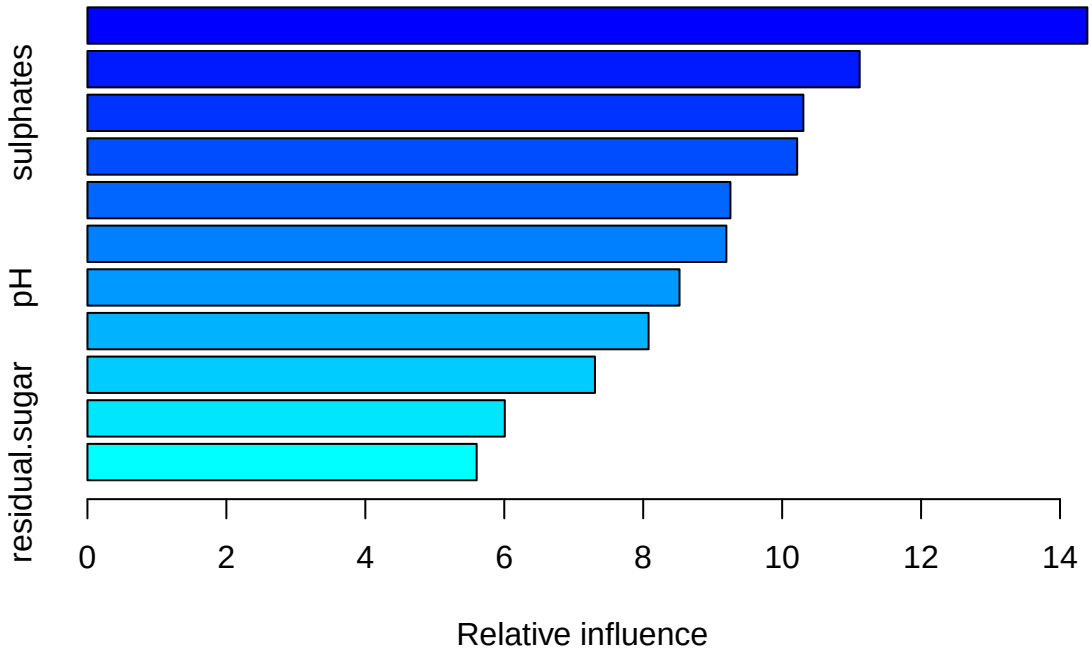

```
yhat_boost = predict(boost_data, newdata=data_test, n.trees=5000)
yhat_boost_binary <- ifelse(yhat_boost > 0.5, 1, 0)
table(yhat_boost_binary, data_test$high)
```

```
##
## yhat_boost_binary   0   1
##                 0 179  47
##                 1  51 203
```

```
# F1-score
conf_matrix <- table(yhat_boost_binary, data_test$high)
precision <- conf_matrix[2, 2] / sum(conf_matrix[, 2])
recall <- conf_matrix[2, 2] / sum(conf_matrix[2, ])
f1_score_boosting1 <- 2 * (precision * recall) / (precision + recall)
print(f1_score_boosting1)
```

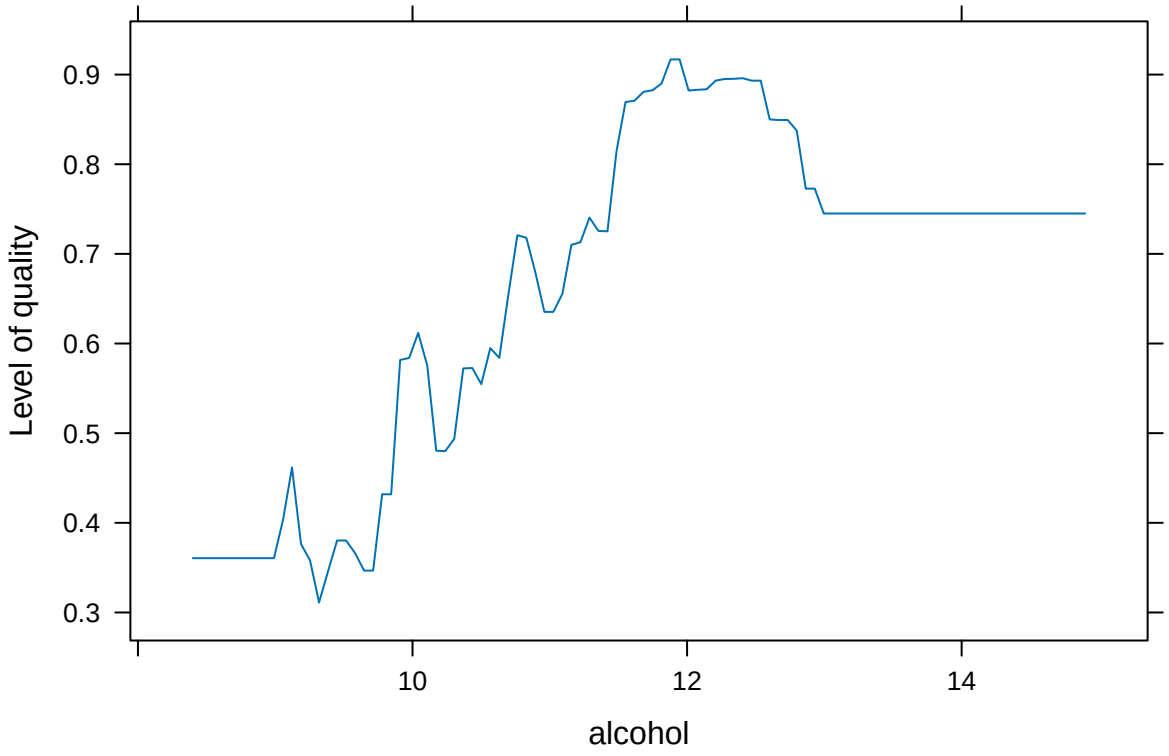
```
## [1] 0.8055556
```

```
# Fit a different learning rate
boost_data2 = gbm(high~., data=data_train, distribution="gaussian",
                  n.trees=5000, interaction.depth=4, shrinkage=0.1, verbose=F)
summary(boost_data2)
```

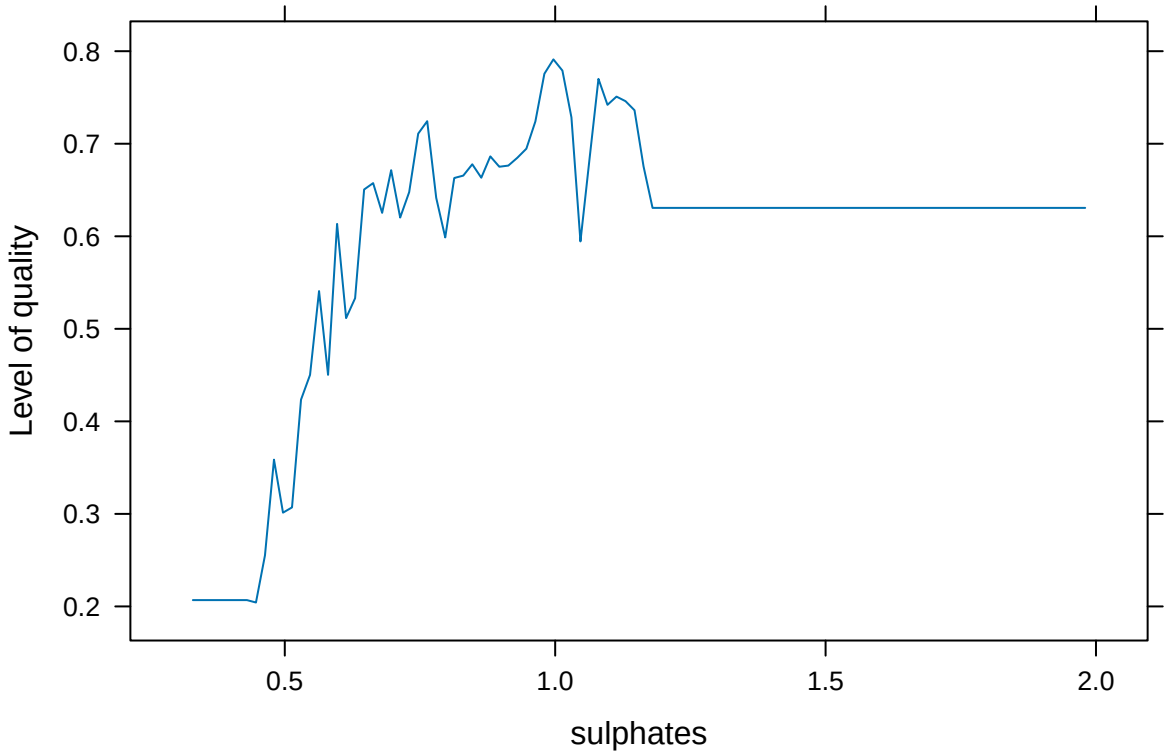



```
##                                       var   rel.inf
## alcohol                           alcohol 14.393310
## volatile.acidity         volatile.acidity 11.115902
## sulphates                       sulphates 10.305786
## total.sulfur.dioxide total.sulfur.dioxide 10.216385
## density                           density  9.255258
## chlorides                       chlorides  9.197244
## pH                                     pH  8.522558
## citric.acid                   citric.acid  8.077299
## fixed.acidity               fixed.acidity  7.305632
## free.sulfur.dioxide   free.sulfur.dioxide  6.007817
## residual.sugar             residual.sugar  5.602808
```

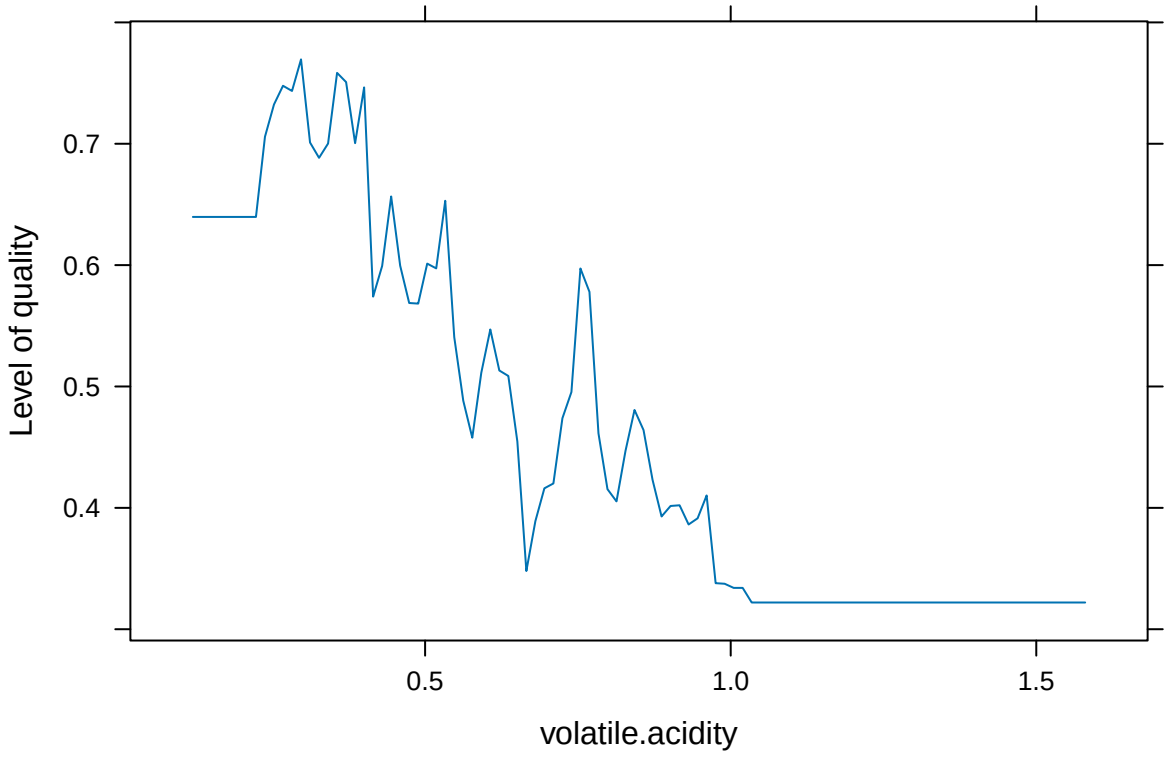
```
plot(boost_data2, i="alcohol", ylab = "Level of quality")
```




```
plot(boost_data2, i="sulphates", ylab = "Level of quality")
```




```
plot(boost_data2, i="volatile.acidity", ylab = "Level of quality")
```




```
yhat_boost2 = predict(boost_data2, newdata=data_test, n.trees=5000)
yhat_boost_binary2 <- ifelse(yhat_boost2 > 0.5, 1, 0)
table(yhat_boost_binary2, data_test$high)
```

```
##
## yhat_boost_binary2   0   1
##                  0 174  53
##                  1  56 197
```

```
# F1-score
conf_matrix <- table(yhat_boost_binary2, data_test$high)
precision <- conf_matrix[2, 2] / sum(conf_matrix[, 2])
recall <- conf_matrix[2, 2] / sum(conf_matrix[2, ])
f1_score_boosting2 <- 2 * (precision * recall) / (precision + recall)
print(f1_score_boosting2)
```

```
## [1] 0.7833002
```