Project Wine

Yongfan Zhao

2024-02-15

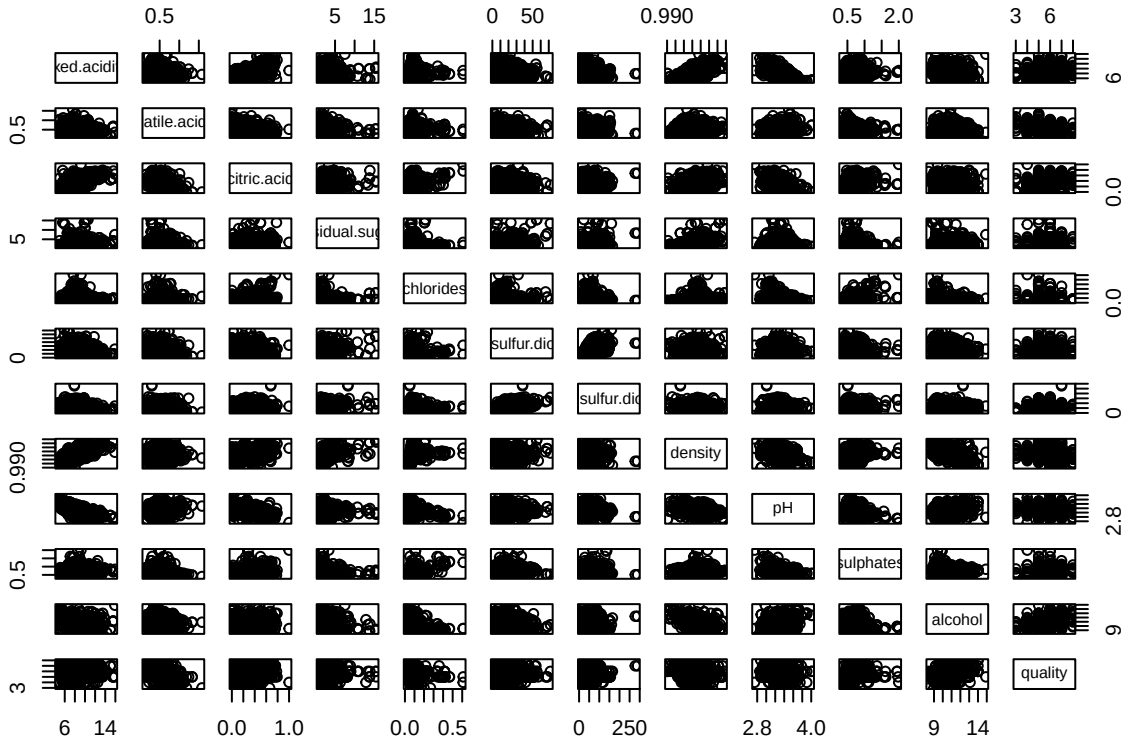
```
rm(list = ls())
Red_Wine = read.csv("winequality-red.csv")
View(Red_Wine)
Red_Wine = na.omit(Red_Wine)
# use summary function to understand the Data
# range, median, mean
summary(Red_Wine)
```

```
##  fixed.acidity   volatile.acidity  citric.acid    residual.sugar
##  Min.   : 4.60   Min.   :0.1200   Min.   :0.000   Min.   : 0.900
##  1st Qu.: 7.10   1st Qu.:0.3900   1st Qu.:0.090   1st Qu.: 1.900
##  Median : 7.90   Median :0.5200   Median :0.260   Median : 2.200
##  Mean   : 8.32   Mean   :0.5278   Mean   :0.271   Mean   : 2.539
##  3rd Qu.: 9.20   3rd Qu.:0.6400   3rd Qu.:0.420   3rd Qu.: 2.600
##  Max.   :15.90   Max.   :1.5800   Max.   :1.000   Max.   :15.500
##    chlorides       free.sulfur.dioxide total.sulfur.dioxide    density
##  Min.   :0.01200   Min.   : 1.00       Min.   :  6.00       Min.   :0.9901
##  1st Qu.:0.07000   1st Qu.: 7.00       1st Qu.: 22.00       1st Qu.:0.9956
##  Median :0.07900   Median :14.00       Median : 38.00       Median :0.9968
##  Mean   :0.08747   Mean   :15.87       Mean   : 46.47       Mean   :0.9967
##  3rd Qu.:0.09000   3rd Qu.:21.00       3rd Qu.: 62.00       3rd Qu.:0.9978
##  Max.   :0.61100   Max.   :72.00       Max.   :289.00       Max.   :1.0037
##        pH          sulphates         alcohol         quality
##  Min.   :2.740   Min.   :0.3300   Min.   : 8.40   Min.   :3.000
##  1st Qu.:3.210   1st Qu.:0.5500   1st Qu.: 9.50   1st Qu.:5.000
##  Median :3.310   Median :0.6200   Median :10.20   Median :6.000
##  Mean   :3.311   Mean   :0.6581   Mean   :10.42   Mean   :5.636
##  3rd Qu.:3.400   3rd Qu.:0.7300   3rd Qu.:11.10   3rd Qu.:6.000
##  Max.   :4.010   Max.   :2.0000   Max.   :14.90   Max.   :8.000
```

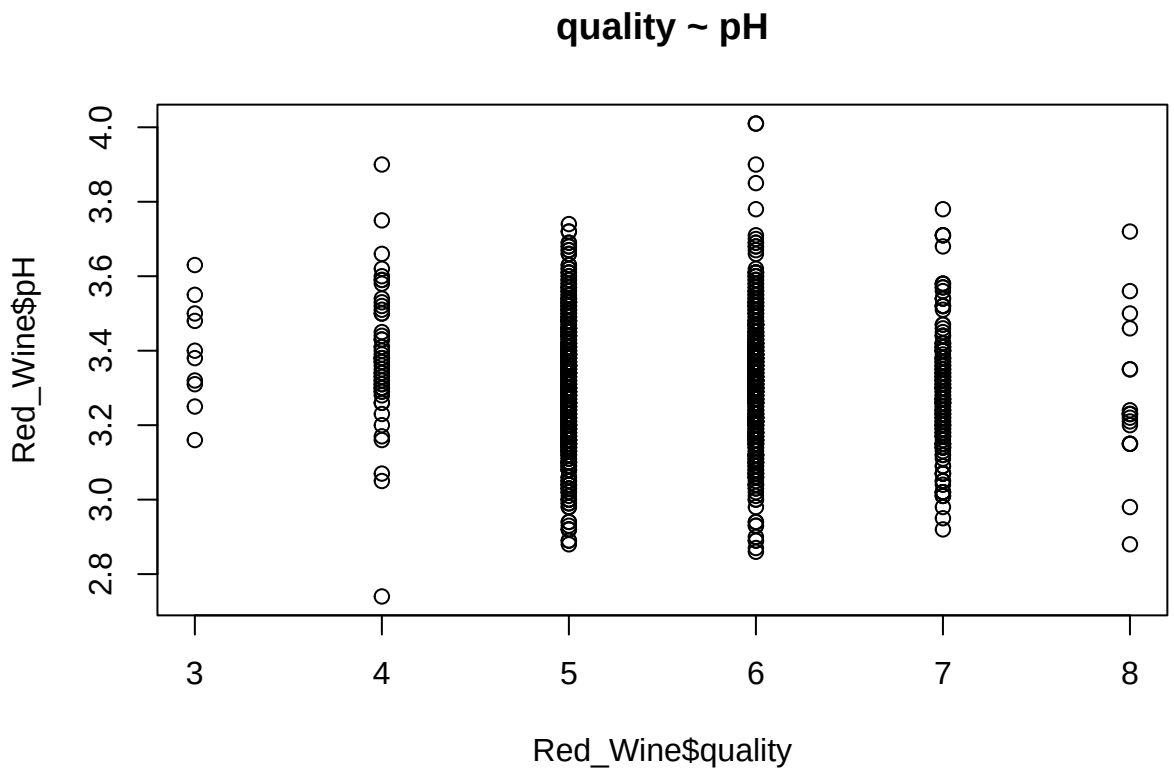
```
 # sd for the each predictor
sd = sapply(Red_Wine,sd)
sd
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##          1.741096318          0.179059704          0.194801137
##       residual.sugar            chlorides  free.sulfur.dioxide
##          1.409928060          0.047065302         10.460156970
## total.sulfur.dioxide              density                   pH
##         32.895324478          0.001887334          0.154386465
##            sulphates              alcohol              quality
##          0.169506980          1.065667582          0.807569440
```

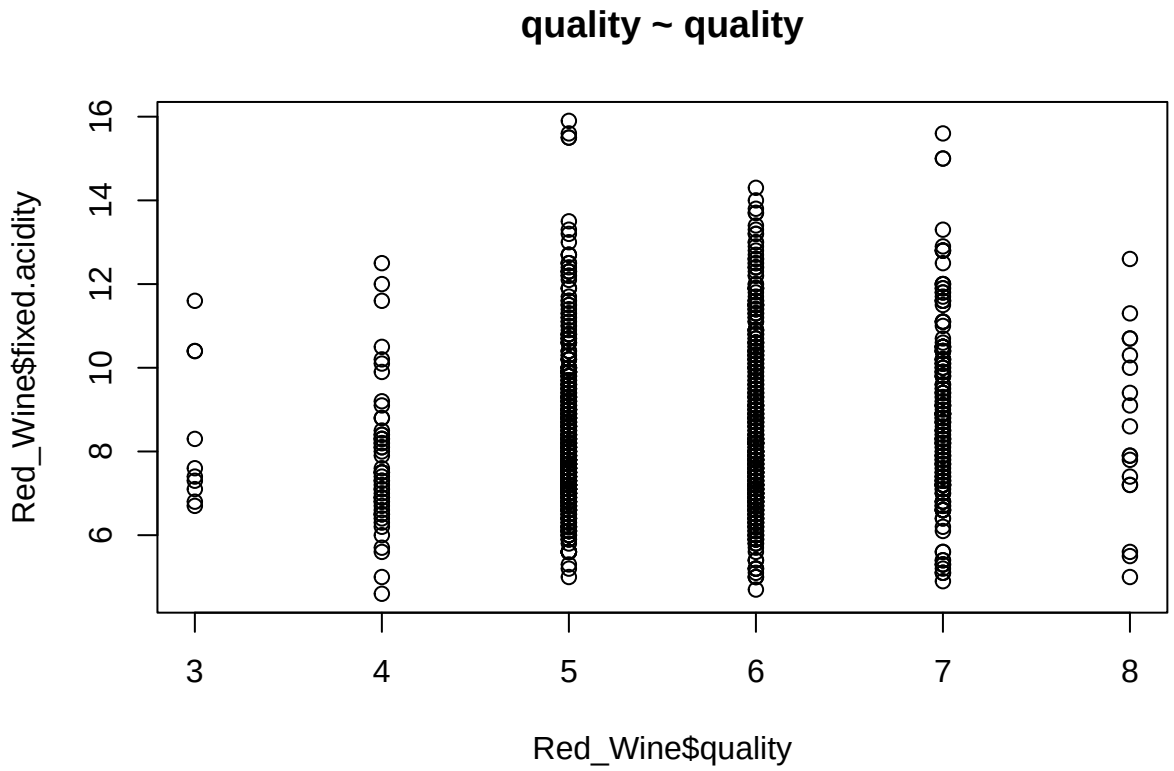
```r
# use pairs to see the relation between target and predictor
pairs(Red_Wine)
```




```r
plot(Red_Wine$quality,Red_Wine$pH,main = "quality ~ pH")
```



```r
plot(Red_Wine$quality,Red_Wine$fixed.acidity,main = "quality ~ quality")
```




```r
# correlation of each data
round(cor(Red_Wine),digits = 3)
```

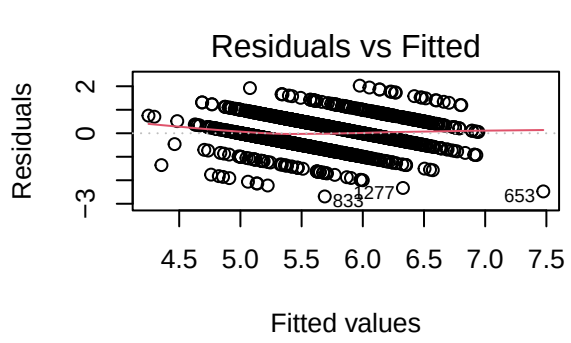
```
##                       fixed.acidity volatile.acidity citric.acid residual.sugar
## fixed.acidity                 1.000           -0.256       0.672          0.115
## volatile.acidity             -0.256            1.000      -0.552          0.002
## citric.acid                   0.672           -0.552       1.000          0.144
## residual.sugar                0.115            0.002       0.144          1.000
## chlorides                     0.094            0.061       0.204          0.056
## free.sulfur.dioxide          -0.154           -0.011      -0.061          0.187
## total.sulfur.dioxide         -0.113            0.076       0.036          0.203
## density                       0.668            0.022       0.365          0.355
## pH                           -0.683            0.235      -0.542         -0.086
## sulphates                     0.183           -0.261       0.313          0.006
## alcohol                      -0.062           -0.202       0.110          0.042
## quality                       0.124           -0.391       0.226          0.014
##                       chlorides free.sulfur.dioxide total.sulfur.dioxide density
## fixed.acidity             0.094              -0.154                -0.113   0.668
## volatile.acidity          0.061              -0.011                 0.076   0.022
## citric.acid               0.204              -0.061                 0.036   0.365
## residual.sugar            0.056               0.187                 0.203   0.355
## chlorides                 1.000               0.006                 0.047   0.201
## free.sulfur.dioxide       0.006               1.000                 0.668  -0.022
## total.sulfur.dioxide      0.047               0.668                 1.000   0.071
## density                   0.201              -0.022                 0.071   1.000
## pH                       -0.265               0.070                -0.066  -0.342
## sulphates                 0.371               0.052                 0.043   0.149
## alcohol                  -0.221              -0.069                -0.206  -0.496
```

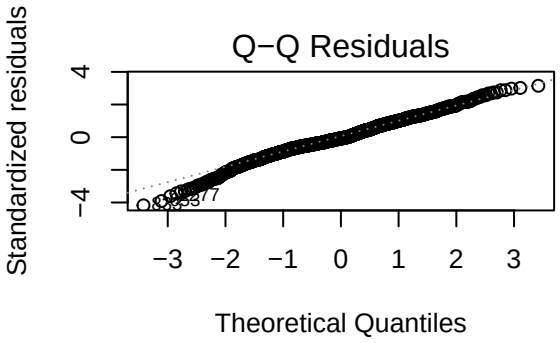
```
## quality                 -0.129              -0.051               -0.185  -0.175
##                          pH sulphates alcohol quality
## fixed.acidity        -0.683     0.183  -0.062   0.124
## volatile.acidity      0.235    -0.261  -0.202  -0.391
## citric.acid          -0.542     0.313   0.110   0.226
## residual.sugar       -0.086     0.006   0.042   0.014
## chlorides            -0.265     0.371  -0.221  -0.129
## free.sulfur.dioxide   0.070     0.052  -0.069  -0.051
## total.sulfur.dioxide -0.066     0.043  -0.206  -0.185
## density              -0.342     0.149  -0.496  -0.175
## pH                    1.000    -0.197   0.206  -0.058
## sulphates            -0.197     1.000   0.094   0.251
## alcohol               0.206     0.094   1.000   0.476
## quality              -0.058     0.251   0.476   1.000
```

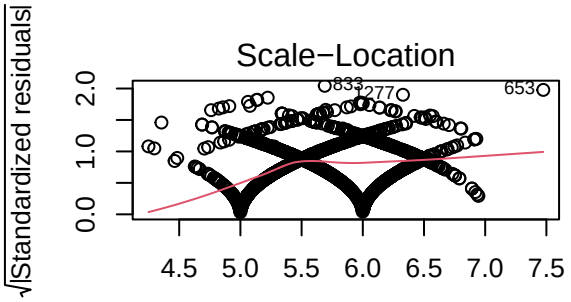
```
lm.fit = lm(Red_Wine$quality ~ .,data = Red_Wine)
summary(lm.fit)
```

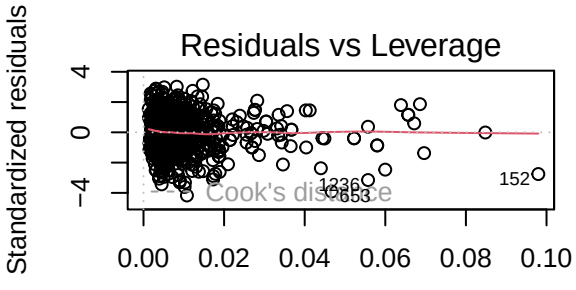
```
##
## Call:
## lm(formula = Red_Wine$quality ~ ., data = Red_Wine)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.68911 -0.36652 -0.04699  0.45202  2.02498
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           2.197e+01  2.119e+01   1.036   0.3002
## fixed.acidity         2.499e-02  2.595e-02   0.963   0.3357
## volatile.acidity     -1.084e+00  1.211e-01  -8.948  < 2e-16 ***
## citric.acid          -1.826e-01  1.472e-01  -1.240   0.2150
## residual.sugar        1.633e-02  1.500e-02   1.089   0.2765
## chlorides            -1.874e+00  4.193e-01  -4.470 8.37e-06 ***
## free.sulfur.dioxide   4.361e-03  2.171e-03   2.009   0.0447 *
## total.sulfur.dioxide -3.265e-03  7.287e-04  -4.480 8.00e-06 ***
## density              -1.788e+01  2.163e+01  -0.827   0.4086
## pH                   -4.137e-01  1.916e-01  -2.159   0.0310 *
## sulphates             9.163e-01  1.143e-01   8.014 2.13e-15 ***
## alcohol               2.762e-01  2.648e-02  10.429  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.648 on 1587 degrees of freedom
## Multiple R-squared:  0.3606, Adjusted R-squared:  0.3561
## F-statistic: 81.35 on 11 and 1587 DF,  p-value: < 2.2e-16
```

```
par(mfrow = c(2,2))
plot(lm.fit)
```










```
####################
set.seed(66)
dim(Red_Wine)
```

```
## [1] 1599   12
```

```
attach(Red_Wine)
train = sample(nrow(Red_Wine),nrow(Red_Wine)/2,replace = FALSE)
lm.fit1 = lm(quality ~ alcohol + volatile.acidity +sulphates+total.sulfur.dioxide + chlorides,data = Red
summary(lm.fit1)
```

```
##
## Call:
## lm(formula = quality ~ alcohol + volatile.acidity + sulphates +
##     total.sulfur.dioxide + chlorides, data = Red_Wine[train,
##     ])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.25853 -0.40285 -0.04007  0.42492  2.00348
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           3.3312044  0.2909791  11.448  < 2e-16 ***
## alcohol               0.2556504  0.0238567  10.716  < 2e-16 ***
## volatile.acidity     -1.3134004  0.1378472  -9.528  < 2e-16 ***
## sulphates             0.9728511  0.1617876   6.013 2.78e-09 ***
## total.sulfur.dioxide -0.0025558  0.0007723  -3.309 0.000978 ***
## chlorides            -2.1740042  0.5345435  -4.067 5.24e-05 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6612 on 793 degrees of freedom
## Multiple R-squared:  0.356,  Adjusted R-squared:  0.3519
## F-statistic: 87.67 on 5 and 793 DF,  p-value: < 2.2e-16
```

```
Wine_test = Red_Wine[-train,"quality"]
lm.pred = predict(lm.fit1,Red_Wine[-train,])
lm_mse = mean((lm.pred -Wine_test)^2)
```

```
# linear regression with interaction
lm.fit2 = lm(quality ~ I(alcohol^2) + I(volatile.acidity^2) +log(sulphates)+log(total.sulfur.dioxide) +
summary(lm.fit2)
```

```
##
## Call:
## lm(formula = quality ~ I(alcohol^2) + I(volatile.acidity^2) +
##     log(sulphates) + log(total.sulfur.dioxide) + log(chlorides),
##     data = Red_Wine[train, ])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.33255 -0.41172 -0.04979  0.44163  2.06029
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 4.702351   0.247212  19.022  < 2e-16 ***
## I(alcohol^2)                0.011465   0.001122  10.215  < 2e-16 ***
## I(volatile.acidity^2)      -0.998813   0.109468  -9.124  < 2e-16 ***
## log(sulphates)              0.818717   0.117289   6.980 6.23e-12 ***
## log(total.sulfur.dioxide) -0.107535   0.034411  -3.125 0.001843 **
## log(chlorides)            -0.293377   0.078544  -3.735 0.000201 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6589 on 793 degrees of freedom
## Multiple R-squared:  0.3605, Adjusted R-squared:  0.3564
## F-statistic: 89.39 on 5 and 793 DF,  p-value: < 2.2e-16
```

```
lm.pred2 = predict(lm.fit2,Red_Wine[-train,])
lm1_mse = mean((lm.pred2 -Wine_test)^2)
```

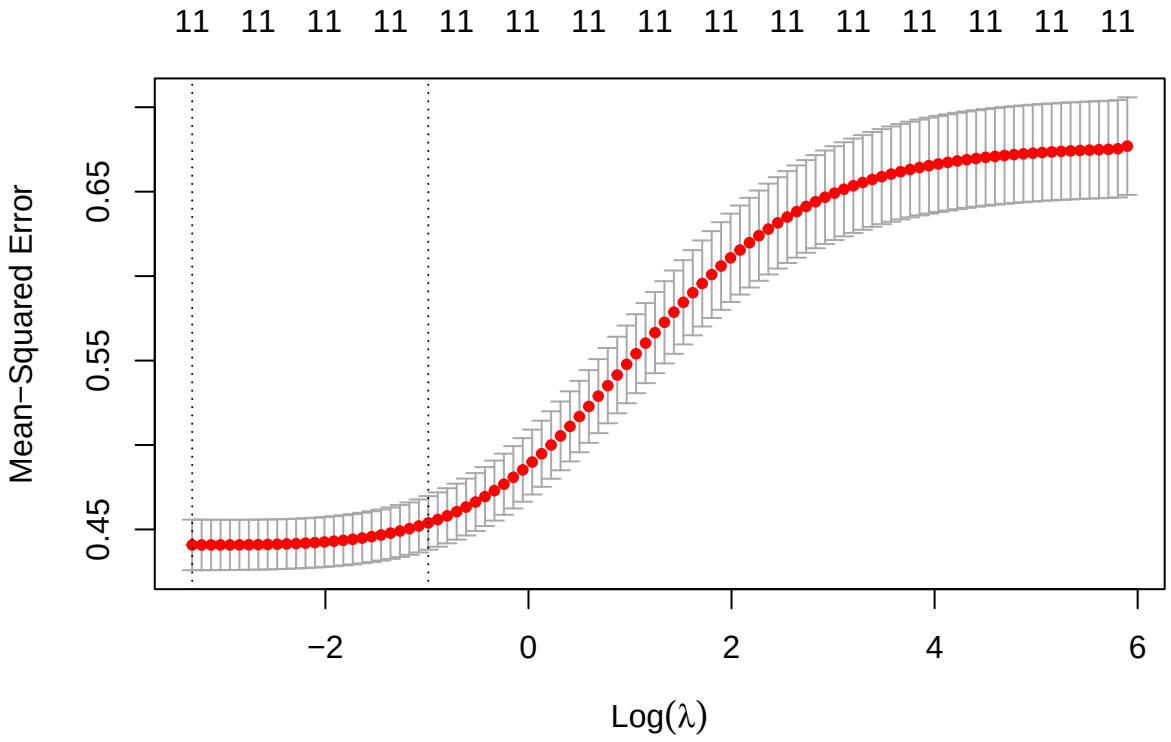
Ridge Regression

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```
x = model.matrix(quality~.,Red_Wine)[,-1]
y = Red_Wine$quality
grid = 10 ^ seq(10,-2,length = 100)
ridge.mod = glmnet(x[train,],y[train],alpha = 0,lambda = grid,thresh = 1e-12)
# find the best lamba
cv.out = cv.glmnet(x[train,],y[train],alpha = 0)
plot(cv.out)
```



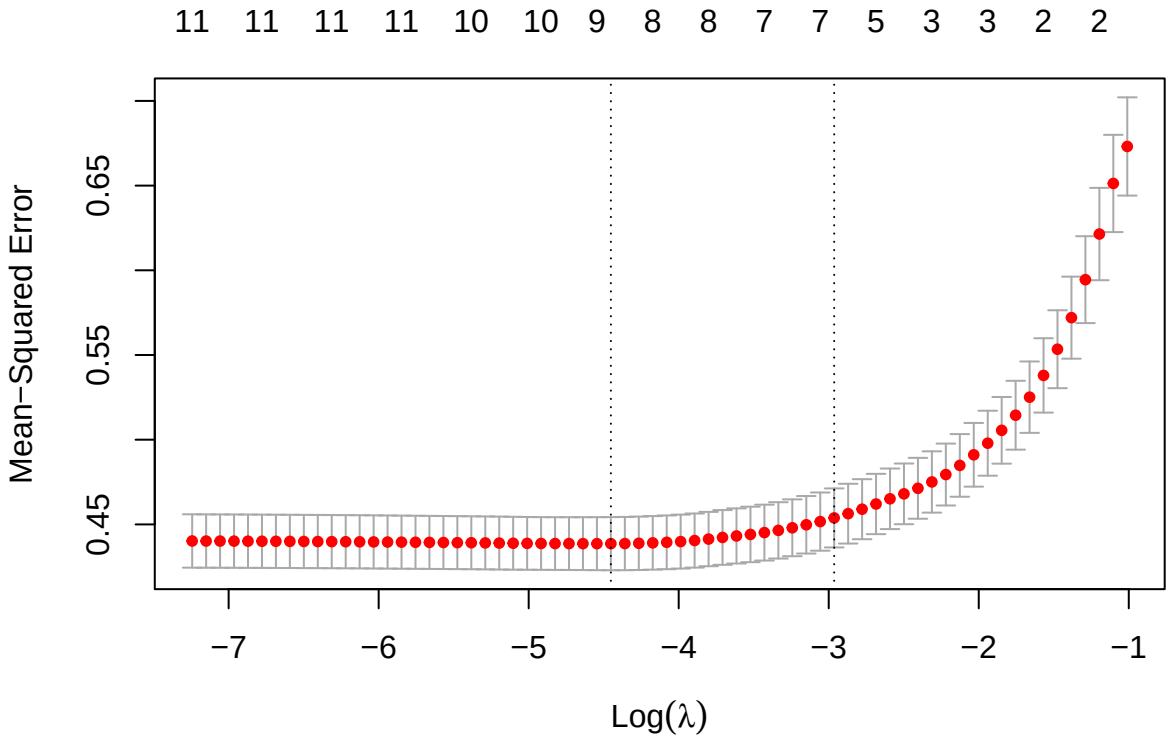

```
bestlam = cv.out$lambda.min
bestlam
```

```
## [1] 0.03642799
```

```
ridge.pred = predict(ridge.mod,s = bestlam,x = x[train,],y = y[train],newx = x[-train,],exact = T)
ridge_mse = mean((ridge.pred-Wine_test)^2)
```

lasso Regression

```
lasso.mod = glmnet(x[train,],y[train],alpha = 1,lambda = grid,thresh = 1e-12)
# find the best lamba
cv.out.lasso = cv.glmnet(x[train,],y[train],alpha = 1)
plot(cv.out.lasso)
```




```
bestlam_lasso = cv.out.lasso$lambda.min
bestlam_lasso
```

```
## [1] 0.01165429
```

```
lasso.pred = predict(lasso.mod,s = bestlam,x = x[train,],y = y[train],newx = x[-train,],exact = T)
lasso_mse = mean((lasso.pred-Wine_test)^2)
```

```
lasso.coef = predict(lasso.mod,type = "coefficients",s = bestlam)[-1,]
# get non-zero coefficients
lasso.coef[lasso.coef!=0]
```

```
##        fixed.acidity     volatile.acidity           chlorides
##          0.004429025         -1.196743499        -1.193591994
## total.sulfur.dioxide                   pH           sulphates
##         -0.001526670         -0.134384945         0.629998435
##              alcohol
##          0.251486602
```

for the non-zero coef, there are 7 coef,compare those three regression. R ^ 2 :Ridge < LM.FIT1 < LASSO < LM.FIT

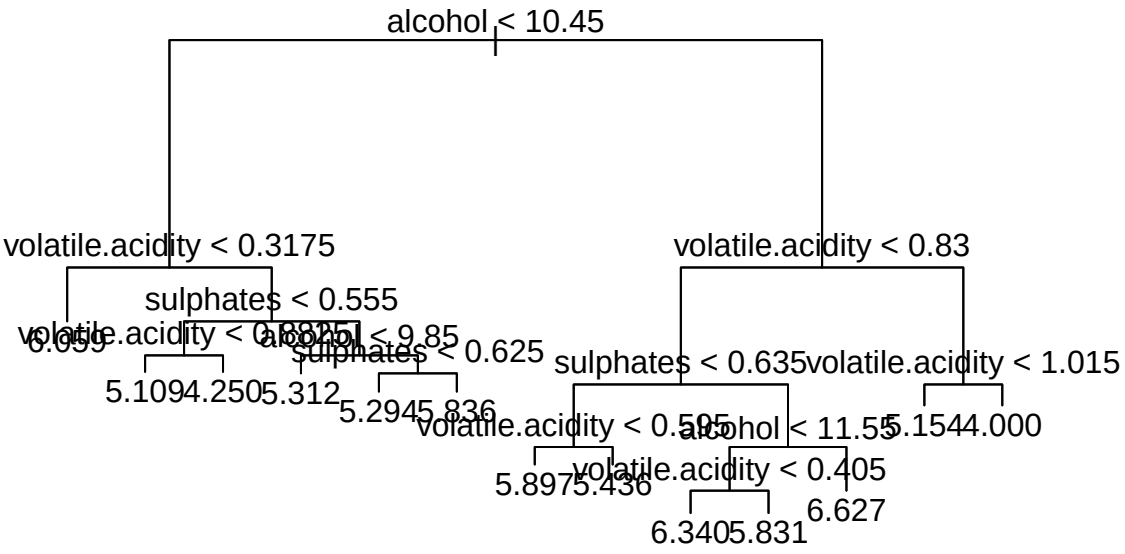
```
# TREES
library(tree)
tree.wine = tree(quality ~.,data = Red_Wine[train,])
summary(tree.wine)
```

```
##
## Regression tree:
## tree(formula = quality ~ ., data = Red_Wine[train, ])
## Variables actually used in tree construction:
## [1] "alcohol"          "volatile.acidity" "sulphates"
## Number of terminal nodes:  13
```

```
## Residual mean deviance:  0.4044 = 317.8 / 786
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -2.3120 -0.3122 -0.1094  0.0000  0.3733  1.9410
```

```
# 13 nodes,
```

```
#plot the tree
plot(tree.wine)
text(tree.wine,pretty = 0)
```



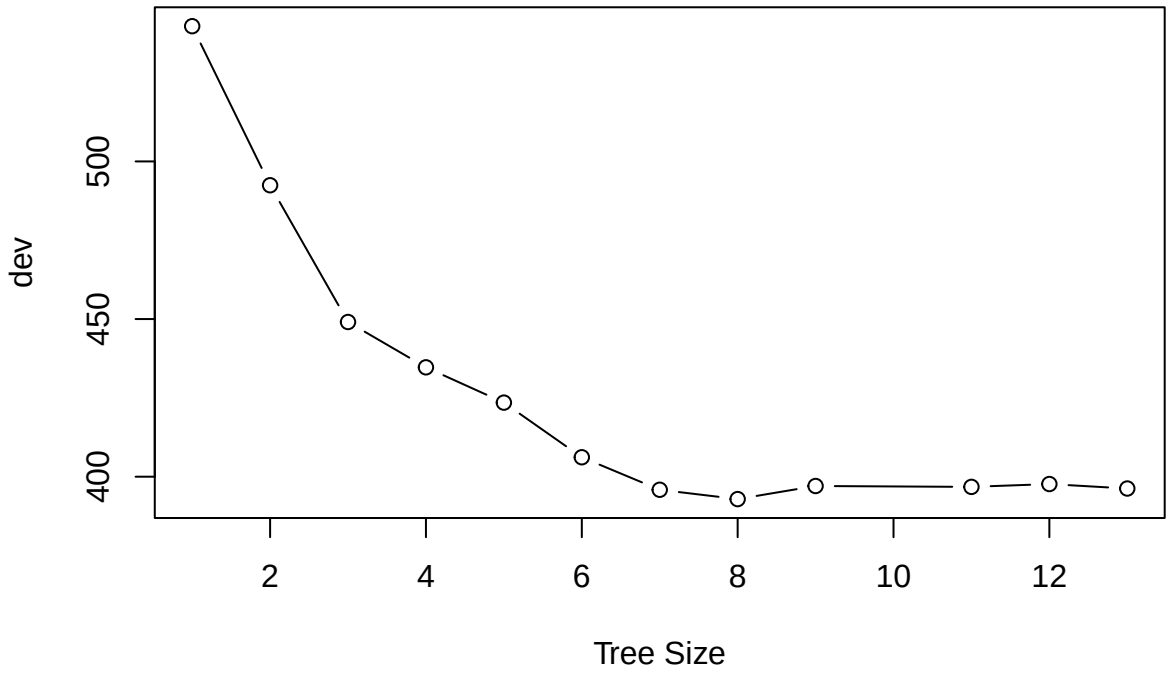

```
tree.pred = predict(tree.wine,Red_Wine[-train,])
tree_mse = mean((tree.pred - Wine_test)^2)
```

prune tree

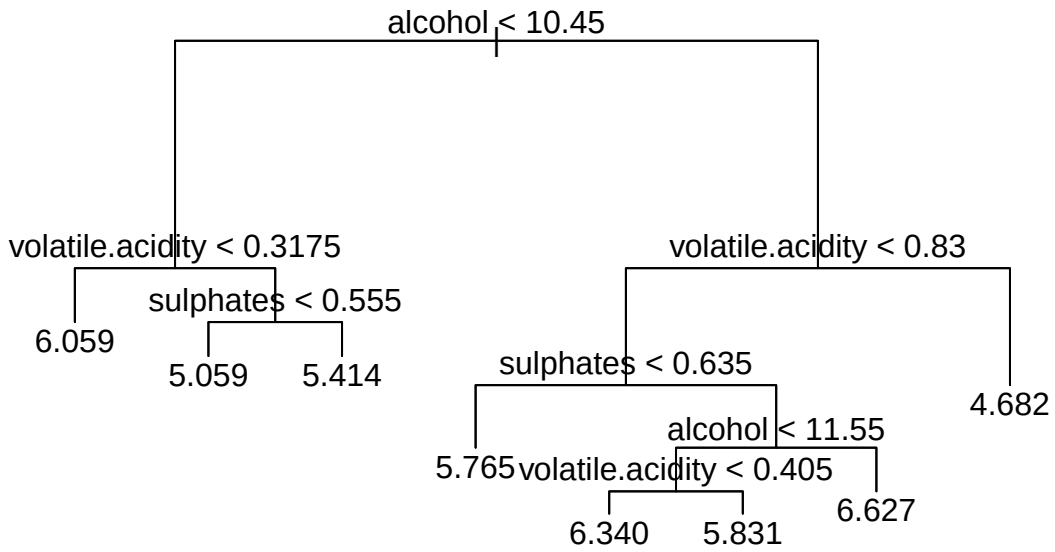
```
cv.out.tree = cv.tree(tree.wine,FUN = prune.tree)
cv.out.tree
```

```
## $size
##  [1] 13 12 11  9  8  7  6  5  4  3  2  1
##
## $dev
##  [1] 396.2448 397.6669 396.7684 397.0445 392.8938 395.8512 406.1593 423.4958
##  [9] 434.6871 449.0610 492.4407 542.8909
##
## $k
##  [1]      -Inf  5.560662  5.911772  6.129975  7.080420  7.559492 11.499052
##  [8] 14.759779 18.230322 21.167095 39.558796 76.913333
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```r
plot(cv.out.tree$size,cv.out.tree$dev,type = "b",xlab = "Tree Size",ylab = "dev")
```




```r
wine.prune = prune.tree(tree.wine,best = 8)
plot(wine.prune)
text(wine.prune,pretty = 0)
```




```r
prune.pred = predict(wine.prune,Red_Wine[-train])
prune_mse = mean((prune.pred - Wine_test)^2)
```

```
## Warning in prune.pred - Wine_test: longer object length is not a multiple of
## shorter object length
```

```r
# prune not work will in this case
```

Bagging

```r
library(randomForest)
```

```
## randomForest 4.7-1.1
```

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

```r
bagging.fit = randomForest(quality~.,data = Red_Wine,subset = train,mtry = 11,importance = TRUE)
bagging.fit
```

```
##
## Call:
##  randomForest(formula = quality ~ ., data = Red_Wine, mtry = 11,      importance = TRUE, subset = tra
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 11
##
##           Mean of squared residuals: 0.4006737
##                     % Var explained: 40.53
```

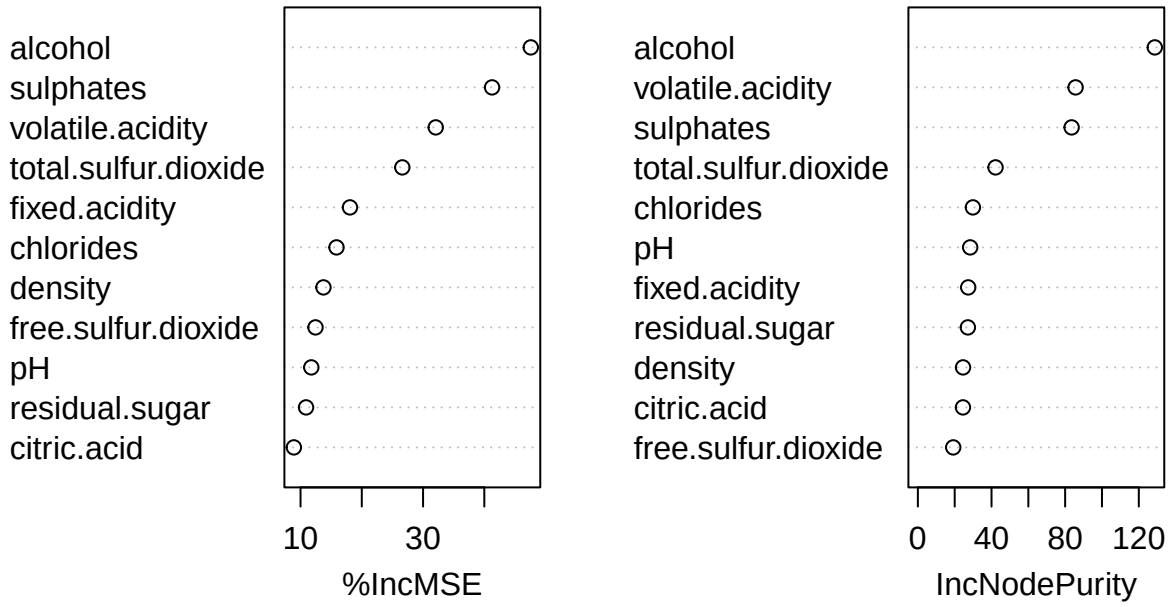
```r
bagging.pred = predict(bagging.fit,Red_Wine[-train,])
bagging_mse = mean((bagging.pred-Wine_test)^2)
```

```r
# random forest
min_mean = Inf
for (i in 1:11){
rf.fit = randomForest(quality~.,data = Red_Wine,subset = train,mtry = i,importance = TRUE)
bagging.pred = predict(bagging.fit,Red_Wine[-train,])
rf_MSE = mean((bagging.pred-Wine_test)^2)
if(rf_MSE< min_mean){
  min_mean = rf_MSE
  best_mtry = i
}
}
cat("Minimum mean:", min_mean, "with mtry =", best_mtry, "\n")
```

```
## Minimum mean: 0.3342029 with mtry = 1
```

```r
rf_mse = min_mean
```

```r
varImpPlot(rf.fit)
```




```
importance(rf.fit)
```

```
##                       %IncMSE IncNodePurity
## fixed.acidity        18.071975      27.34916
## volatile.acidity     32.069219      85.67413
## citric.acid           8.912563      24.49292
## residual.sugar       10.902992      27.15757
## chlorides            15.872484      29.89000
## free.sulfur.dioxide  12.440197      19.21082
## total.sulfur.dioxide 26.613665      42.18373
## density              13.741017      24.53474
## pH                   11.760270      28.40956
## sulphates            41.264372      83.50861
## alcohol              47.583639     128.68512
```

boosting

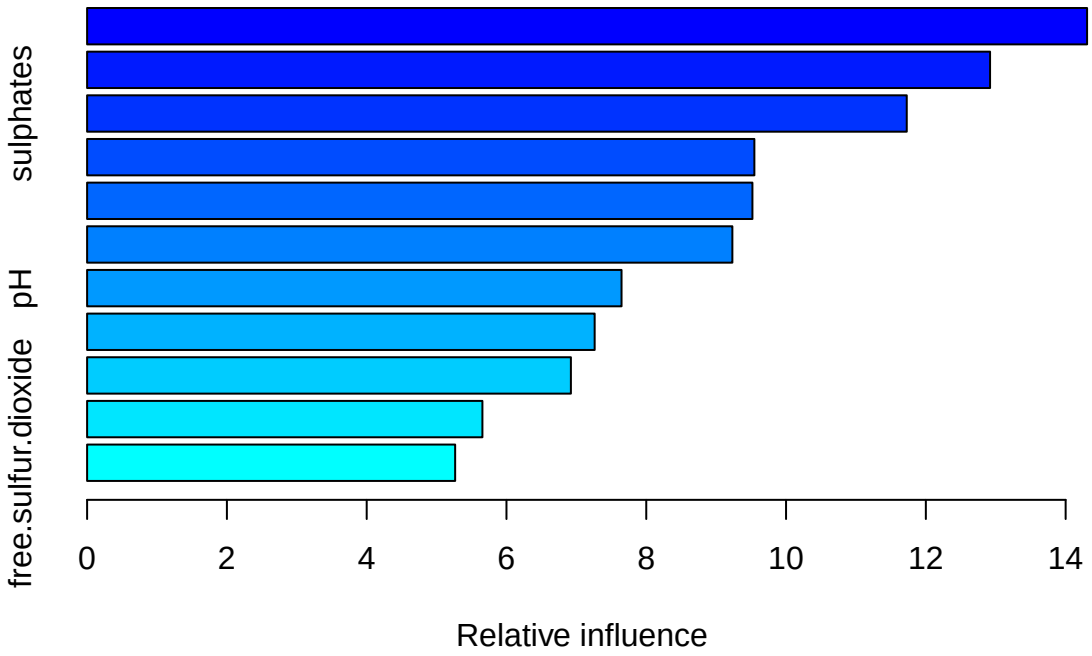
```
library(gbm)
```

```
## Loaded gbm 2.1.9
```

```
## This version of gbm is no longer under development. Consider transitioning to gbm3, https://github.co
```

```
boost.fit = gbm(quality~.,data = Red_Wine[train,],distribution = "gaussian",n.trees = 5000,interaction.
```

```
summary(boost.fit)
```



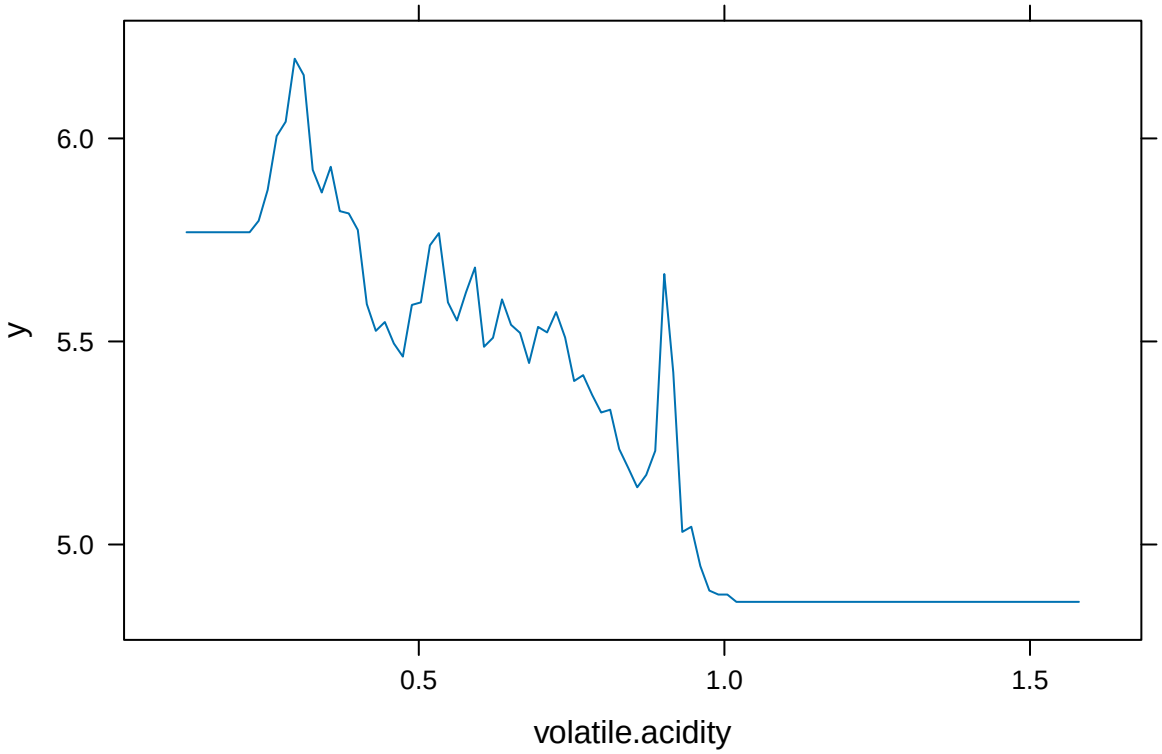

```
##                                      var   rel.inf
## volatile.acidity         volatile.acidity 14.306817
## alcohol                           alcohol 12.918694
## sulphates                       sulphates 11.726890
## total.sulfur.dioxide total.sulfur.dioxide  9.546487
## chlorides                       chlorides  9.518603
## density                           density  9.231772
## pH                                     pH  7.646158
## citric.acid                   citric.acid  7.261388
## fixed.acidity               fixed.acidity  6.921577
## residual.sugar             residual.sugar  5.655800
## free.sulfur.dioxide   free.sulfur.dioxide  5.265815
```

```
boost.pred = predict(boost.fit,newdata = Red_Wine[-train,],n.trees=5000)
mean((boost.pred-Wine_test)^2)
```

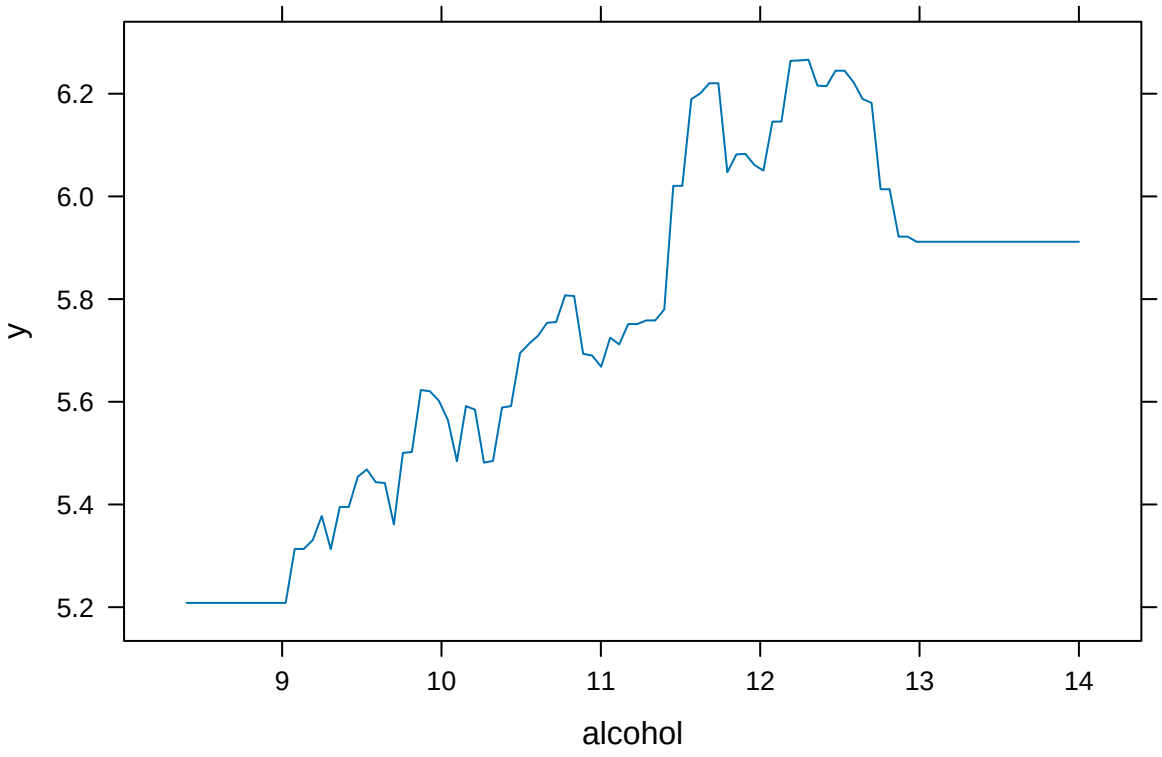
```
## [1] 0.4381224
```

```
boost.pred = predict(boost.fit,newdata = Red_Wine[-train,],n.trees=500)
boost_mse = mean((boost.pred-Wine_test)^2)
# when we use less tree get a small MSE
```

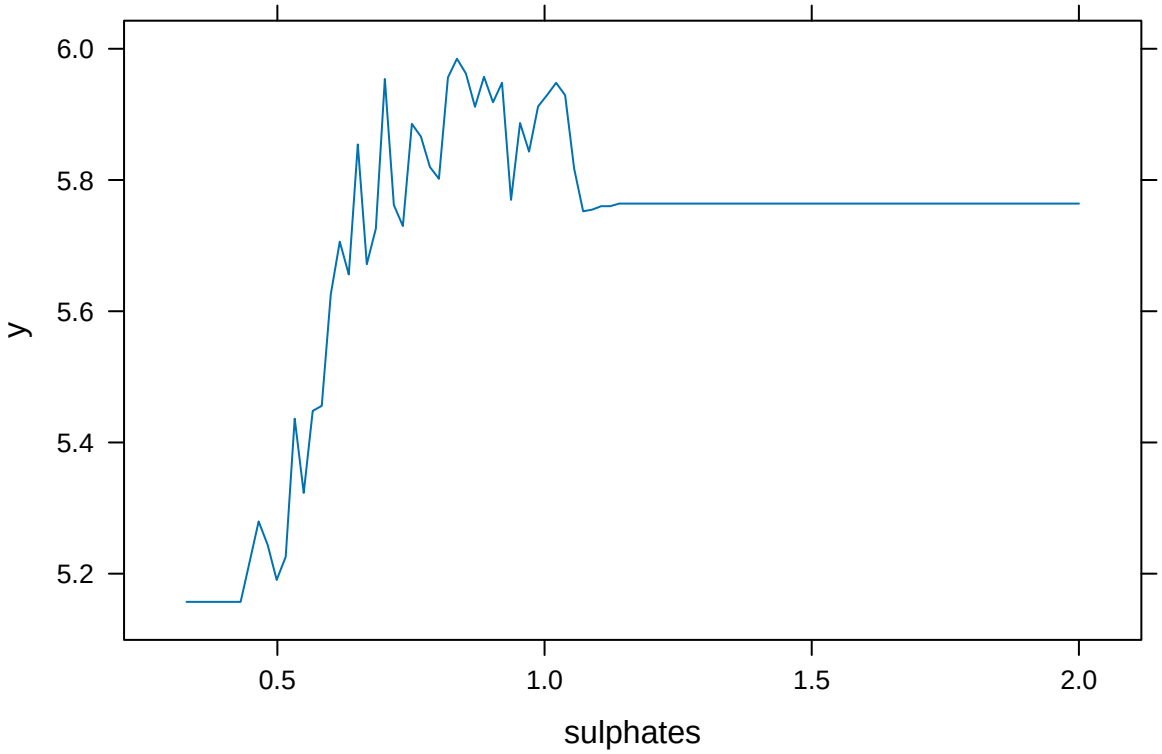
```
plot(boost.fit,i = "volatile.acidity")
```



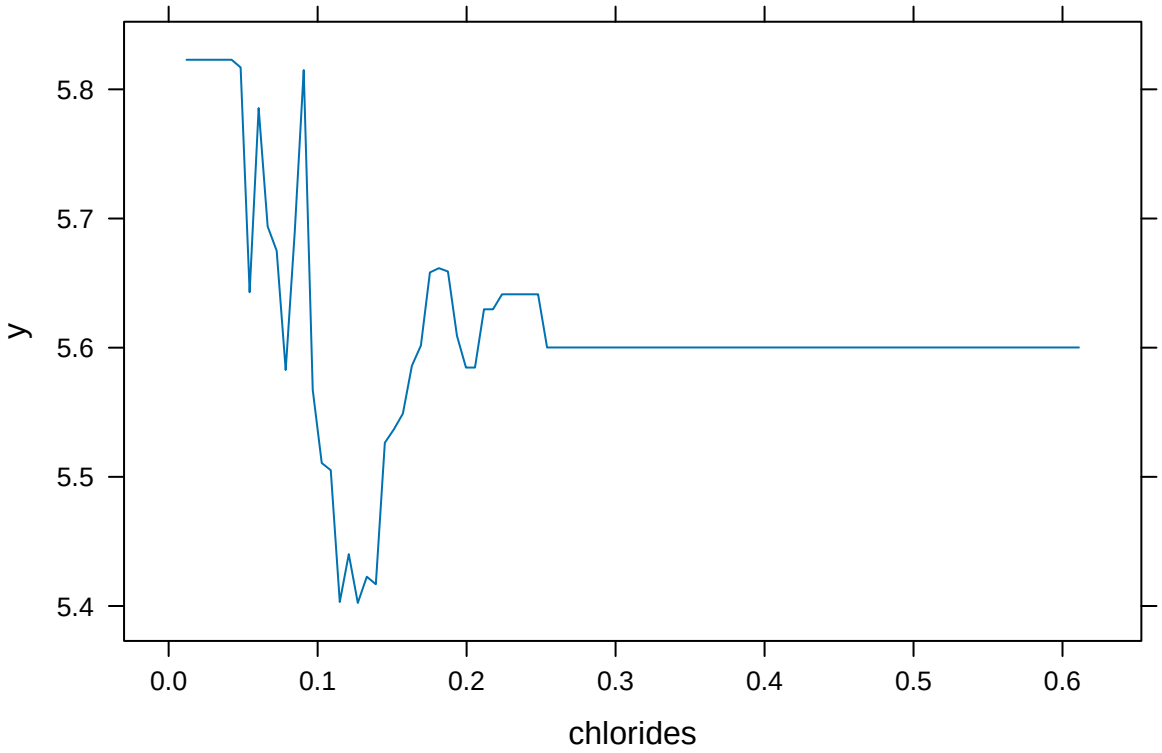

```
plot(boost.fit,i = "alcohol")
```




```
plot(boost.fit,i = "sulphates")
```




```
plot(boost.fit,i = "chlorides")
```




```
library(neuralnet)
library(caret)
```

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

```
##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':
##
##     margin

## Loading required package: lattice
```

```
library(nnet)
df = read.csv("winequality-red.csv")
dim(df)
```

```
## [1] 1599   12
```

```
head(df)
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.4             0.70        0.00            1.9     0.076
## 2           7.8             0.88        0.00            2.6     0.098
## 3           7.8             0.76        0.04            2.3     0.092
## 4          11.2             0.28        0.56            1.9     0.075
## 5           7.4             0.70        0.00            1.9     0.076
## 6           7.4             0.66        0.00            1.8     0.075
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  11                   34  0.9978 3.51      0.56     9.4
## 2                  25                   67  0.9968 3.20      0.68     9.8
## 3                  15                   54  0.9970 3.26      0.65     9.8
## 4                  17                   60  0.9980 3.16      0.58     9.8
## 5                  11                   34  0.9978 3.51      0.56     9.4
## 6                  13                   40  0.9978 3.51      0.56     9.4
##   quality
## 1       5
## 2       5
## 3       5
## 4       6
## 5       5
## 6       5
```

```
var = c("volatile.acidity","alcohol","sulphates","total.sulfur.dioxide","chlorides","density","pH","citi
df = df[,c('quality',var)]
str(df)
```

```
## 'data.frame':    1599 obs. of  9 variables:
##  $ quality             : int  5 5 5 6 5 5 5 7 7 5 ...
##  $ volatile.acidity    : num  0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.5 ...
##  $ alcohol             : num  9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
##  $ sulphates           : num  0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57 0.8 ...
##  $ total.sulfur.dioxide: num  34 67 54 60 34 40 59 21 18 102 ...
##  $ chlorides           : num  0.076 0.098 0.092 0.075 0.076 0.075 0.069 0.065 0.073 0.071 ...
##  $ density             : num  0.998 0.997 0.997 0.998 0.998 ...
##  $ pH                  : num  3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.35 ...
##  $ citric.acid         : num  0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
```

```
# make change in different dataset
#record the range $ value of price
quality = df[,'quality']
max_quality = range(df['quality'])[2]
```

```
min_quality = range(df['quality'])[1]
# scale the numerical
numerical = c("quality","volatile.acidity","alcohol","sulphates","total.sulfur.dioxide","chlorides","den

norm.values = preProcess(df[,numerical],method = 'range')
df[,numerical] = predict(norm.values,df[,numerical])
head(df[,numerical])
```

```
##   quality volatile.acidity   alcohol sulphates total.sulfur.dioxide chlorides
## 1     0.4        0.3972603 0.1538462 0.1377246           0.09893993 0.1068447
## 2     0.4        0.5205479 0.2153846 0.2095808           0.21554770 0.1435726
## 3     0.4        0.4383562 0.2153846 0.1916168           0.16961131 0.1335559
## 4     0.6        0.1095890 0.2153846 0.1497006           0.19081272 0.1051753
## 5     0.4        0.3972603 0.1538462 0.1377246           0.09893993 0.1068447
## 6     0.4        0.3698630 0.1538462 0.1377246           0.12014134 0.1051753
##     density        pH citric.acid
## 1 0.5675477 0.6062992        0.00
## 2 0.4941263 0.3622047        0.00
## 3 0.5088106 0.4094488        0.04
## 4 0.5822320 0.3307087        0.56
## 5 0.5675477 0.6062992        0.00
## 6 0.5675477 0.6062992        0.00
```

```
#make change in different dataset
# convert categorical to dummies

f = as.formula(paste('quality~',
                     paste(names(df)[!names(df) %in% c("quality")],
                           collapse = '+')))
f
```

```
## quality ~ volatile.acidity + alcohol + sulphates + total.sulfur.dioxide +
##     chlorides + density + pH + citric.acid
```

```
set.seed(66)
train = sample(nrow(df),nrow(df)/2)
nn = neuralnet(f,data = df[train,],hidden = 2)
compute_mse = function(nn,df,train,quality){
  # make prediction with train
  pred.train = compute(nn,subset(df[train,],select = -c(quality)))
  pred.train.orig = pred.train$net.result *(max_quality - min_quality) + min_quality
  train.mse = mean((quality[train]-pred.train.orig)^2)
  # test
  pred.test  = compute(nn,subset(df[-train,],select = -c(quality)))
  pred.test.orig = pred.test$net.result *(max_quality - min_quality) + min_quality
  test.mse = sqrt(mean((quality[-train]-pred.test.orig)^2))
  # return
  mse = as.data.frame(rbind(train.mse,test.mse))
  return(mse)
}
mse = compute_mse(nn,df,train,quality)
mse
```

```
##                  V1
## train.mse 0.3862409
## test.mse  0.6264340
```

single layer with 5 nodes

```
nn1 = neuralnet(f,data = df[train,],hidden =4)
mse1 = compute_mse(nn1,df,train,quality)
```

{two layers, 5 nodes in each layer}

```
nn2 = neuralnet(f,data = df[train,],hidden =c(5,5))
mse2 = compute_mse(nn2,df,train,quality)
```

2layer,4 nodes in each layer

```
nn3 = neuralnet(f,data = df[train,],hidden =c(4,4))
mse3 = compute_mse(nn3,df,train,quality)
```

```
nn4 = neuralnet(f,data = df[train,],hidden =c(4,4,4))
mse4 = compute_mse(nn4,df,train,quality)
```

```
nn_mse_df = cbind(mse,mse1,mse3,mse4,mse2)
names(nn_mse_df) = c('1_layer_2nodes',"1_layer_4_nodes","2_layer_4_nodes","3_layer_4_nodes", "2_layer_5
nn_mse_df
```

```
##           1_layer_2nodes 1_layer_4_nodes 2_layer_4_nodes 3_layer_4_nodes
## train.mse      0.3862409       0.3875833       0.3586865       0.2990268
## test.mse       0.6264340       0.6195124       0.6469280       0.6746118
##           2_layer_5_nodes
## train.mse       0.3428813
## test.mse        0.8324282
```

```
regression_mse_df = as.data.frame(cbind(lm_mse,lm1_mse,ridge_mse,lasso_mse))
names(regression_mse_df) = c("lm_mse","lm1_mse","ridge_mse","lass0_mse")
regression_mse_df
```

```
##      lm_mse  lm1_mse ridge_mse lass0_mse
## 1 0.4146733 0.412079 0.4113106 0.4117974
```

```
trees_mse_df = as.data.frame(cbind(tree_mse,prune_mse,bagging_mse,rf_mse,boost_mse))
names(trees_mse_df) = c("tree_mse","prune_mse","bagging_mse","rf_mse","boost_mse")
trees_mse_df
```

```
##    tree_mse prune_mse bagging_mse    rf_mse boost_mse
## 1 0.4482091 0.8966937   0.3342029 0.3342029 0.3785012
```