pre_run

Ziqi Zhou

3/12/2020

Introduction

The red wine is one of the most popular alcoholic beverages in the world. Most of our team members enjoy drinking red wine and have a great interest in factors that can affect the quality of the red wine. We want to figure out what determined the quality of the wine. Based on this motivation, we choose this dataset "Red Wine Quality".

This dataset has 12 columns and 1599 rows. And after omitting the NA data, we still got 1599 rows which means there is no missing value in this dataset.

Variable introduction 1. fixed acidity : most acids involved with wine or fixed or nonvolatile (do not evaporate readily) 2.volatile acidity : the amount of acetic acid in wine, which at too high of levels can lead to an unpleasant, vinegar taste 3.citric acid : found in small quantities, citric acid can add 'freshness' and flavor to wines 4.residual sugar : the amount of sugar remaining after fermentation stops, it's rare to find wines with less than 1 gram/liter and wines with greater than 45 grams/liter are considered sweet 5.chlorides : the amount of salt in the wine 6.free sulfur dioxide : the free form of SO2 exists in equilibrium between molecular SO2 (as a dissolved gas) and bisulfite ion; it prevents microbial growth and the oxidation of wine 7.total sulfur dioxide : amount of free and bound forms of S02; in low concentrations, SO2 is mostly undetectable in wine, but at free SO2 concentrations over 50 ppm, SO2 becomes evident in the nose and taste of wine 8.density : the density of water is close to that of water depending on the percent alcohol and sugar content 9.pH : describes how acidic or basic a wine is on a scale from 0 (very acidic) to 14 (very basic); most wines are between 3-4 on the pH scale 10.sulphates : a wine additive which can contribute to sulfur dioxide gas (S02) levels, wich acts as an antimicrobial and antioxidant 11.alcohol : the percent alcohol content of the wine 12.quality (response): output variable (based on sensory data, score between 0 and 10)

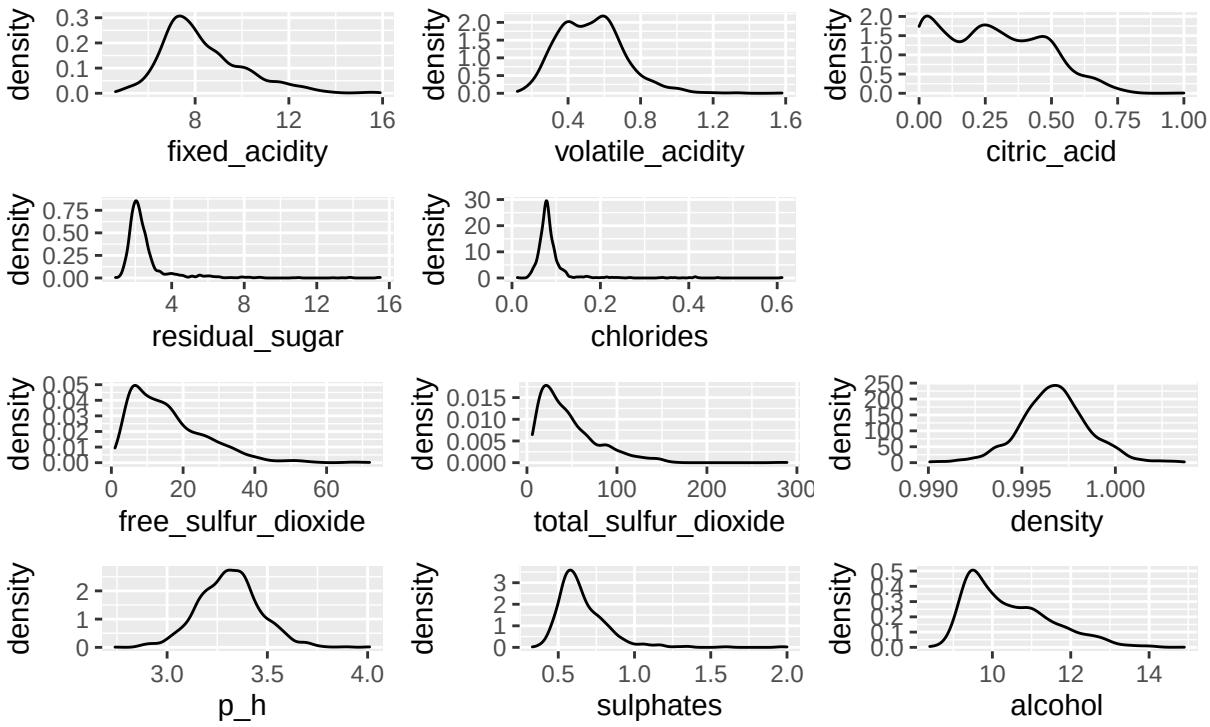
We would like to know what factors influence the quality of the wine and try to build a model to predict the quality of wine given the specific factors.

I used the code below to prepare and clean the data.

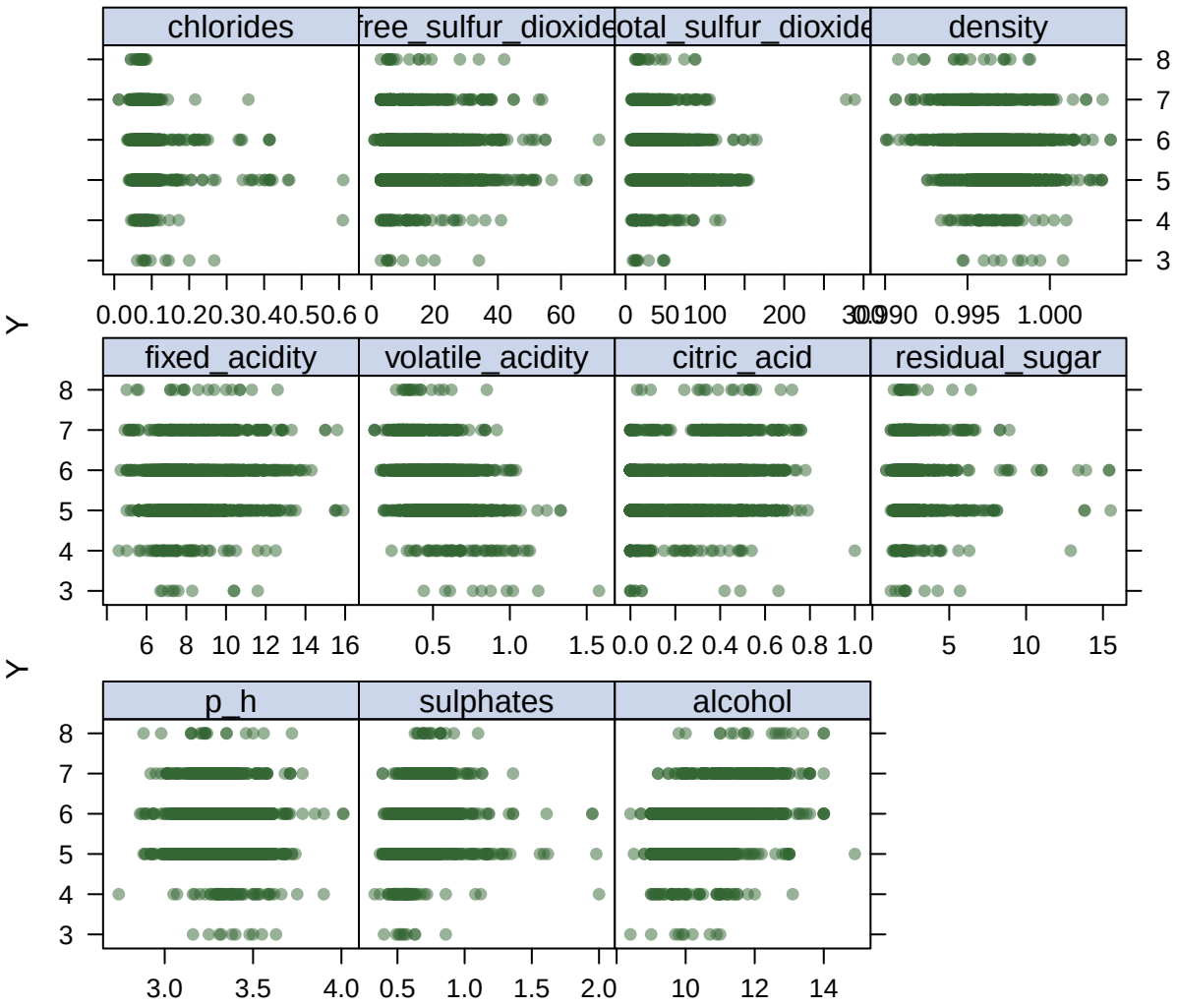
```
# Separate data into training data and test data
set.seed(11)
trRows = createDataPartition(wine$quality, p = .75, list = F)

#all data
x = model.matrix(quality ~ .,wine)[,-1]
y = wine$quality

# Training data
x1 = model.matrix(quality ~ .,wine)[trRows,-1]
y1 = wine$quality[trRows]

# test data
```

```
x2 = model.matrix(quality ~ .,wine)[-trRows,-1]
y2 = wine$quality[-trRows]
```

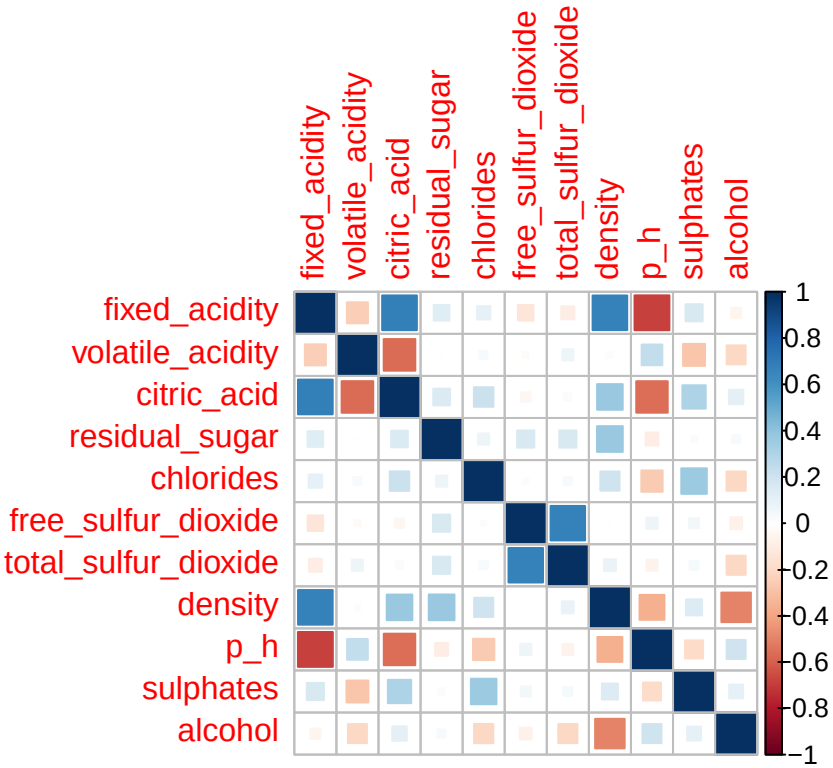
Exploratory Data Analysis




```
#Create scatter plots of responses and predictors
theme1 <- trellis.par.get()
theme1$plot.symbol$col <- rgb(.2, .4, .2, .5)
theme1$plot.symbol$pch <- 16
theme1$plot.line$col <- rgb(.8, .1, .1, 1)
theme1$plot.line$lwd <- 2
theme1$strip.background$col <- rgb(.0, .2, .6, .2)
trellis.par.set(theme1)
featurePlot(x, y, plot = "scatter", labels = c("","Y"),
            type = c("p"), layout = c(4, 2))
```




```
corrplot(cor(x1), method = "square", type = "full")
```



description positive correlation negative correlation pH and fixed acidity alcohol and density citric acid and fixed acidity citric acid and volatile acidity

Models

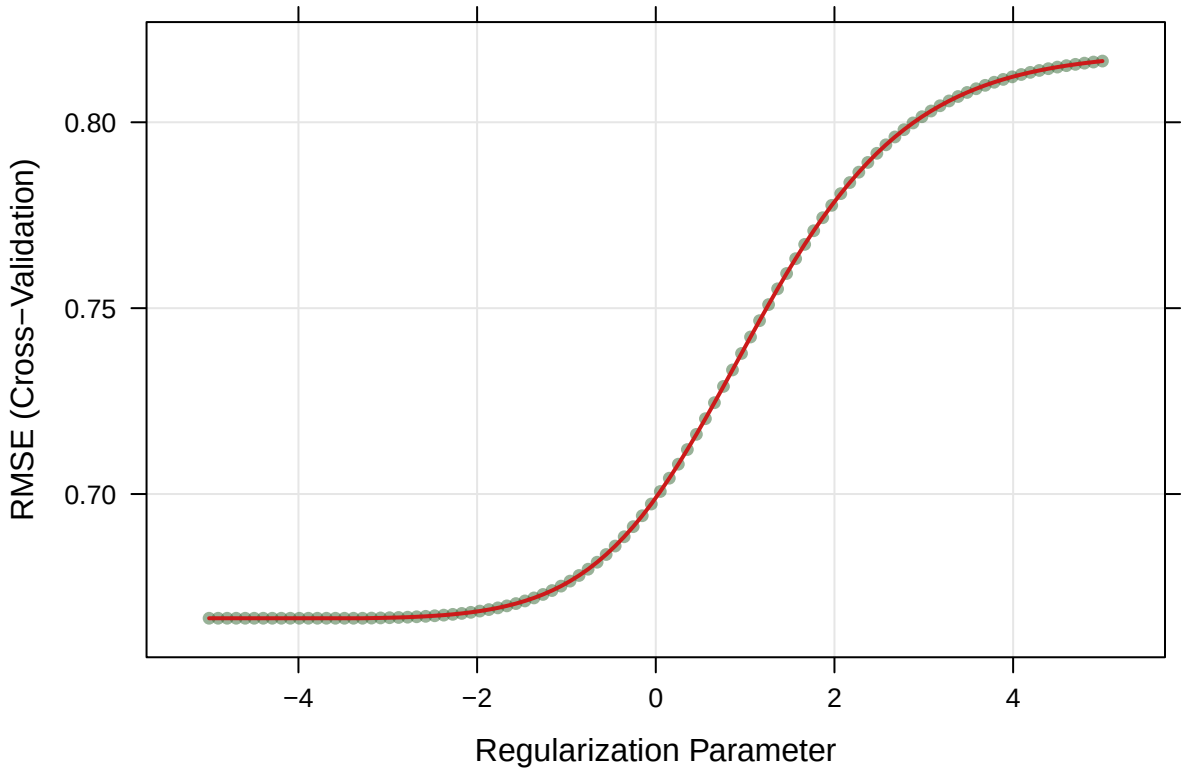
But we cannot use test error to choose model. However we could report the test error of the final model

```
#linear fit R^2 = 0.3458
set.seed(11)
lm.fit <-train(x1, y1,method = "lm",
               trControl = ctrl1)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.63366 -0.38173 -0.05442  0.45126  2.09579
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -1.836e+01  2.536e+01  -0.724  0.46936
## fixed_acidity        -2.658e-02  3.134e-02  -0.848  0.39665
## volatile_acidity     -1.163e+00  1.453e-01  -8.004 2.85e-15 ***
## citric_acid          -2.286e-01  1.780e-01  -1.284  0.19928
```

```
## residual_sugar       -4.174e-03  1.714e-02  -0.244  0.80762
## chlorides            -2.232e+00  4.894e-01  -4.561 5.62e-06 ***
## free_sulfur_dioxide   5.725e-03  2.593e-03   2.208  0.02743 *
## total_sulfur_dioxide -4.096e-03  8.567e-04  -4.781 1.96e-06 ***
## density               2.373e+01  2.590e+01   0.917  0.35958
## p_h                  -6.572e-01  2.284e-01  -2.877  0.00409 **
## sulphates             9.009e-01  1.371e-01   6.570 7.52e-11 ***
## alcohol               2.988e-01  3.118e-02   9.581  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6633 on 1188 degrees of freedom
## Multiple R-squared:  0.3518, Adjusted R-squared:  0.3458
## F-statistic: 58.62 on 11 and 1188 DF,  p-value: < 2.2e-16
```

```
#Ridge fit
set.seed(11)
ridge.fit <-train(x1, y1,method = "glmnet",
                  tuneGrid =expand.grid(alpha = 0,
                                        lambda =exp(seq(-5, 5, length=100))),
                  trControl = ctrl1)

plot(ridge.fit, xTrans =function(x)log(x))
```




```
ridge.fit$bestTune
```

```
##    alpha     lambda
## 17     0 0.03391702
```

```r
coef(ridge.fit$finalModel,ridge.fit$bestTune$lambda)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                  1
## (Intercept)           9.0114668597
## fixed_acidity         0.0009016876
## volatile_acidity     -1.0898151502
## citric_acid          -0.1422535882
## residual_sugar        0.0061130661
## chlorides            -2.0975893668
## free_sulfur_dioxide   0.0046283547
## total_sulfur_dioxide -0.0037296680
## density              -4.4319113330
## p_h                  -0.4225965078
## sulphates             0.9001022715
## alcohol               0.2607045994
```

```r
#test error
predict_y_ridge = predict(ridge.fit, s = ridge.fit$bestTune, newdata = x2)
mse_ridge = mean((y2-predict_y_ridge)^2)
```

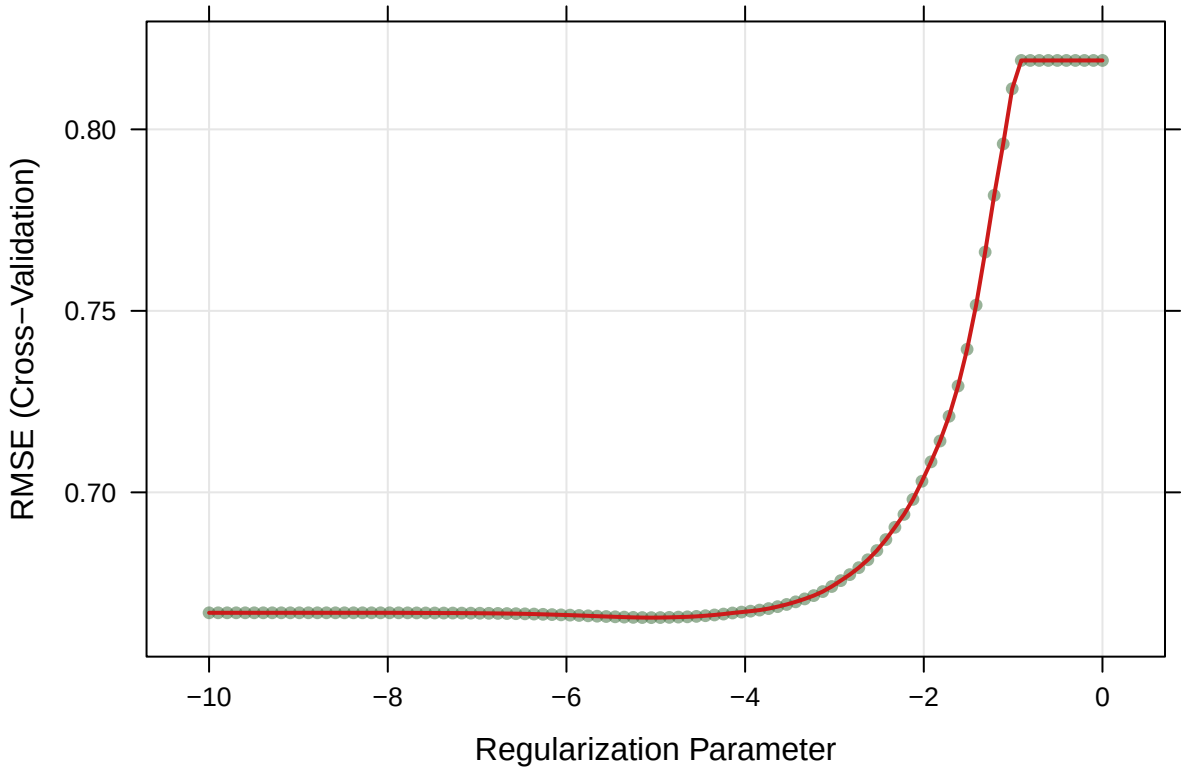
```r
set.seed(11)
# Lasso
lasso.fit = train(x1, y1,method = "glmnet",
                  tuneGrid =expand.grid(alpha = 1,
                                        lambda =exp(seq(-10, 0, length=100))),
                  trControl = ctrl1)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```r
summary(lasso.fit)
```

```
##             Length Class      Mode
## a0           77    -none-     numeric
## beta        847    dgCMatrix  S4
## df           77    -none-     numeric
## dim           2    -none-     numeric
## lambda       77    -none-     numeric
## dev.ratio    77    -none-     numeric
## nulldev       1    -none-     numeric
## npasses       1    -none-     numeric
## jerr          1    -none-     numeric
## offset        1    -none-     logical
## call          5    -none-     call
## nobs          1    -none-     numeric
## lambdaOpt     1    -none-     numeric
## xNames       11    -none-     character
## problemType   1    -none-     character
## tuneValue     2    data.frame list
## obsLevels     1    -none-     logical
## param         0    -none-     list
```

```
plot(lasso.fit, xTrans =function(x)log(x))
```




```
lasso.fit$bestTune
```

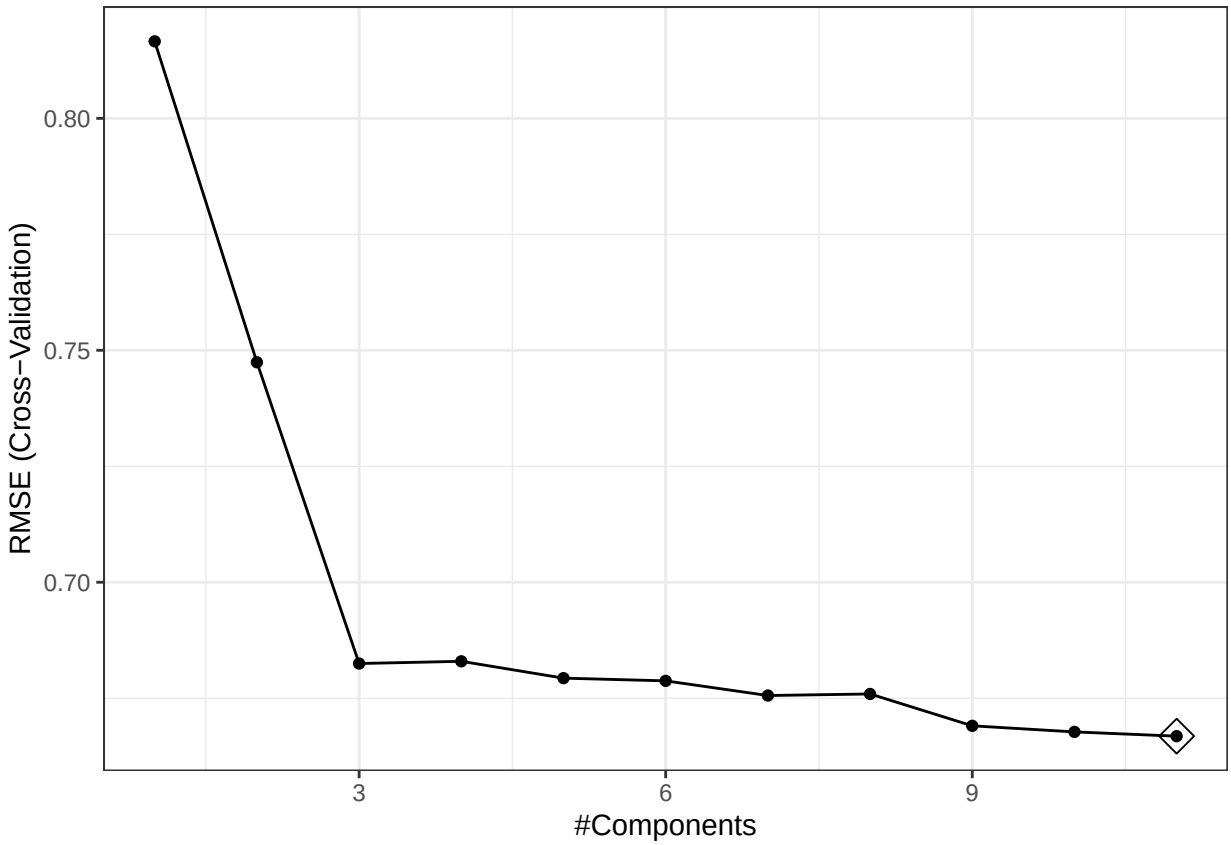
```
##    alpha      lambda
## 50     1 0.006406097
```

```
coef(lasso.fit$finalModel,lasso.fit$bestTune$lambda)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                1
## (Intercept)           4.346836094
## fixed_acidity         .
## volatile_acidity     -1.100042570
## citric_acid          -0.113691704
## residual_sugar        .
## chlorides            -1.984816342
## free_sulfur_dioxide   0.003753622
## total_sulfur_dioxide -0.003415656
## density               .
## p_h                  -0.373742873
## sulphates             0.875405364
## alcohol               0.271411577
```

```
#test error
predict_y_lasso = predict(lasso.fit, s = lasso.fit$bestTune, newdata = x2)
test_error_lasso = mean((y2-predict_y_lasso)^2)
```

```
#pcr
set.seed(11)
pcr.fit <-train(x1, y1,
                method = "pcr",
                tuneGrid  =data.frame(ncomp = 1:11),
                trControl = ctrl1,
                preProc =c("center", "scale"))

ggplot(pcr.fit, highlight = TRUE) +theme_bw()
```




```
# so the ncomp = 11
#pcr.fit$finalModel$coefficients
```

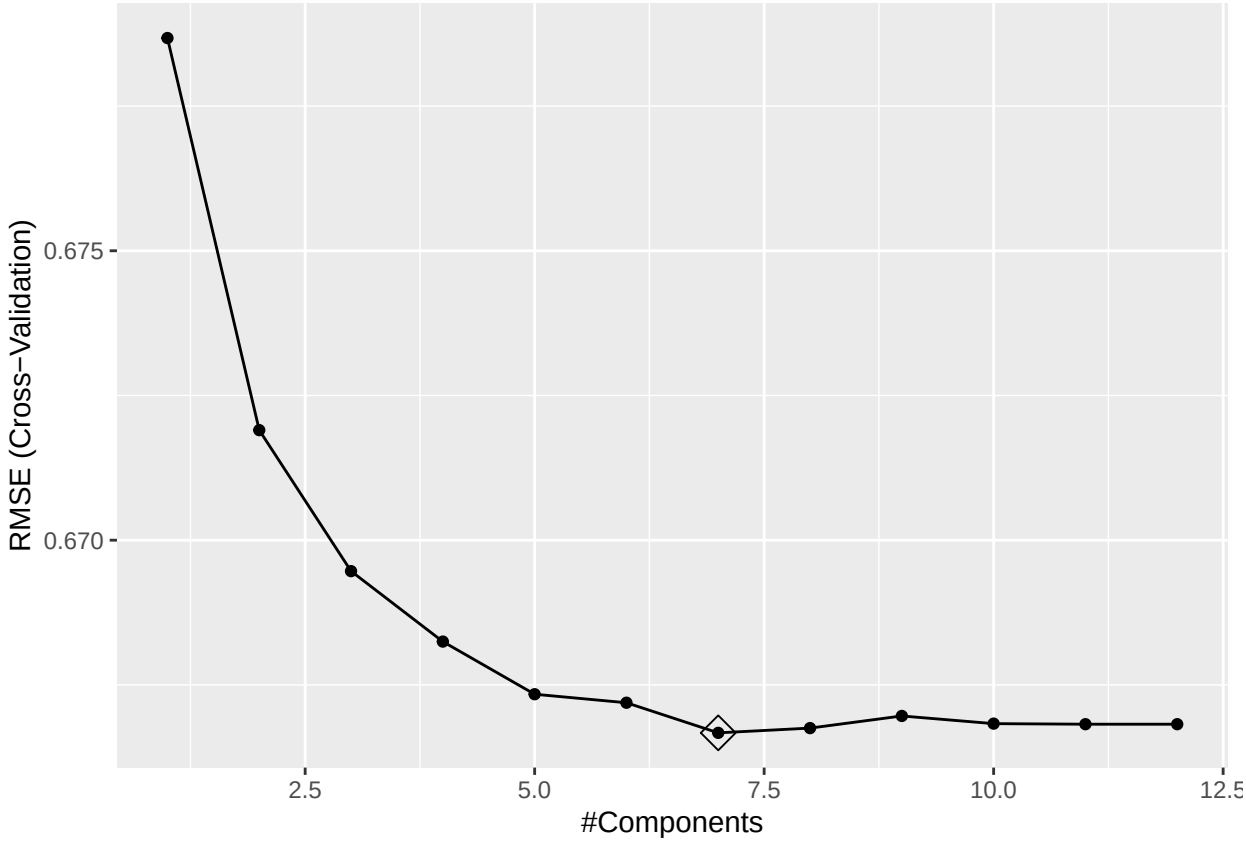
, , 11 comps

.outcome

fixed_acidity -0.046151690 volatile_acidity -0.207362623 citric_acid -0.044630750 residual_sugar -0.006087958 chlorides -0.105520584 free_sulfur_dioxide 0.059433323 total_sulfur_dioxide -0.134563961 density 0.045446567 p_h -0.101488031 sulphates 0.151034045 alcohol 0.324604503

```
#pls
set.seed(11)
pls.fit <-train(x1, y1,method = "pls",
                tuneGrid =data.frame(ncomp = 1:12),
```

```
                trControl = ctrl1,preProc =c("center", "scale"))

predy2.pls2 <-predict(pls.fit, newdata = x2)


ggplot(pls.fit, highlight = TRUE)
```



Treat the response quality as continous variable or classified variable How to treat quality as classified variable? factor the response

```
#fit GAM  R^2 = 0.415
set.seed(11)
gam.fit <- train(x1, y1,
                 method = "gam",
                 tuneGrid = data.frame(method = "GCV.Cp", select = FALSE),
                 trControl = ctrl1)
summary(gam.fit)
```

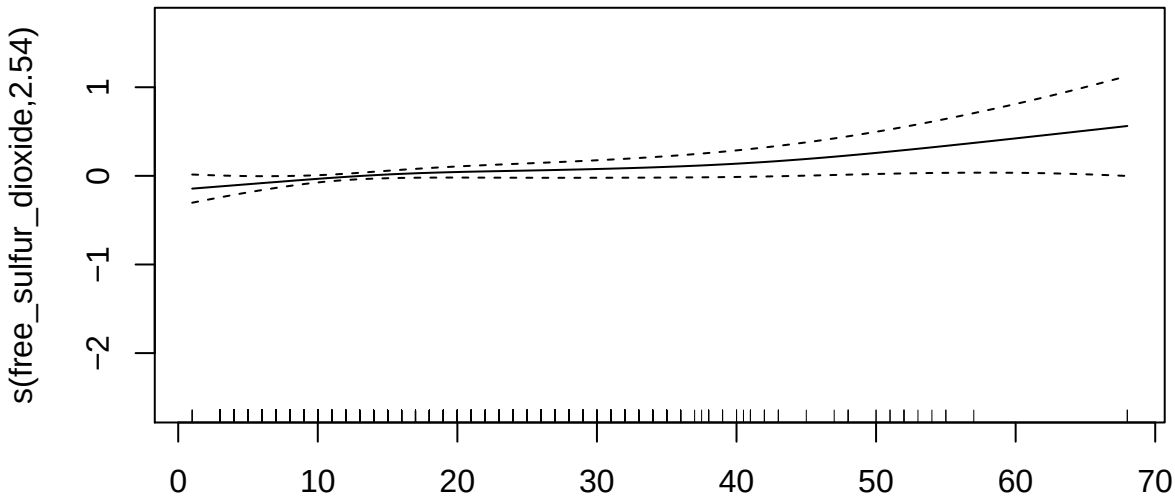
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ s(free_sulfur_dioxide) + s(alcohol) + s(citric_acid) +
##     s(residual_sugar) + s(p_h) + s(fixed_acidity) + s(sulphates) +
##     s(volatile_acidity) + s(chlorides) + s(total_sulfur_dioxide) +
```

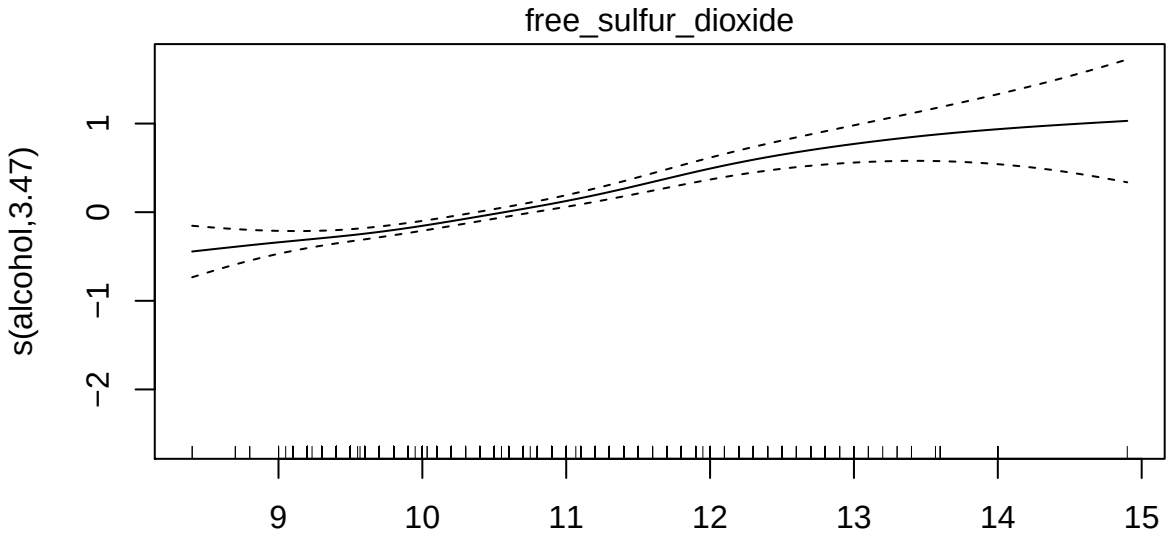
```
##     s(density)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.6342     0.0181   311.2   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                            edf Ref.df      F  p-value
## s(free_sulfur_dioxide)  2.543  3.253  2.301 0.074201 .
## s(alcohol)              3.472  4.361 16.911 4.96e-14 ***
## s(citric_acid)          3.037  3.832  1.371 0.261404
## s(residual_sugar)       6.622  7.700  1.142 0.325172
## s(p_h)                  1.000  1.000  9.941 0.001657 **
## s(fixed_acidity)        5.777  6.883  3.750 0.000685 ***
## s(sulphates)            3.739  4.648 23.298  < 2e-16 ***
## s(volatile_acidity)     2.247  2.910 16.978 1.40e-10 ***
## s(chlorides)            6.650  7.683  3.323 0.001635 **
## s(total_sulfur_dioxide) 7.371  8.230  4.851 4.95e-06 ***
## s(density)              1.614  2.043  0.568 0.566382
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.415   Deviance explained = 43.7%
## GCV = 0.40868  Scale est. = 0.39333   n = 1200
```

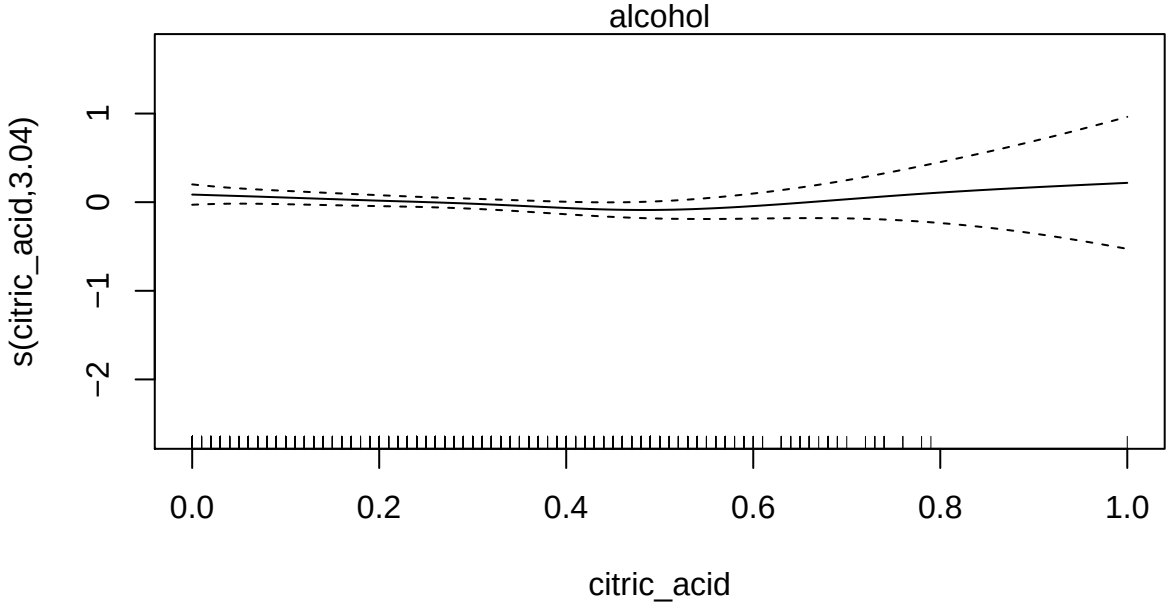
```
gam.fit$finalModel
```

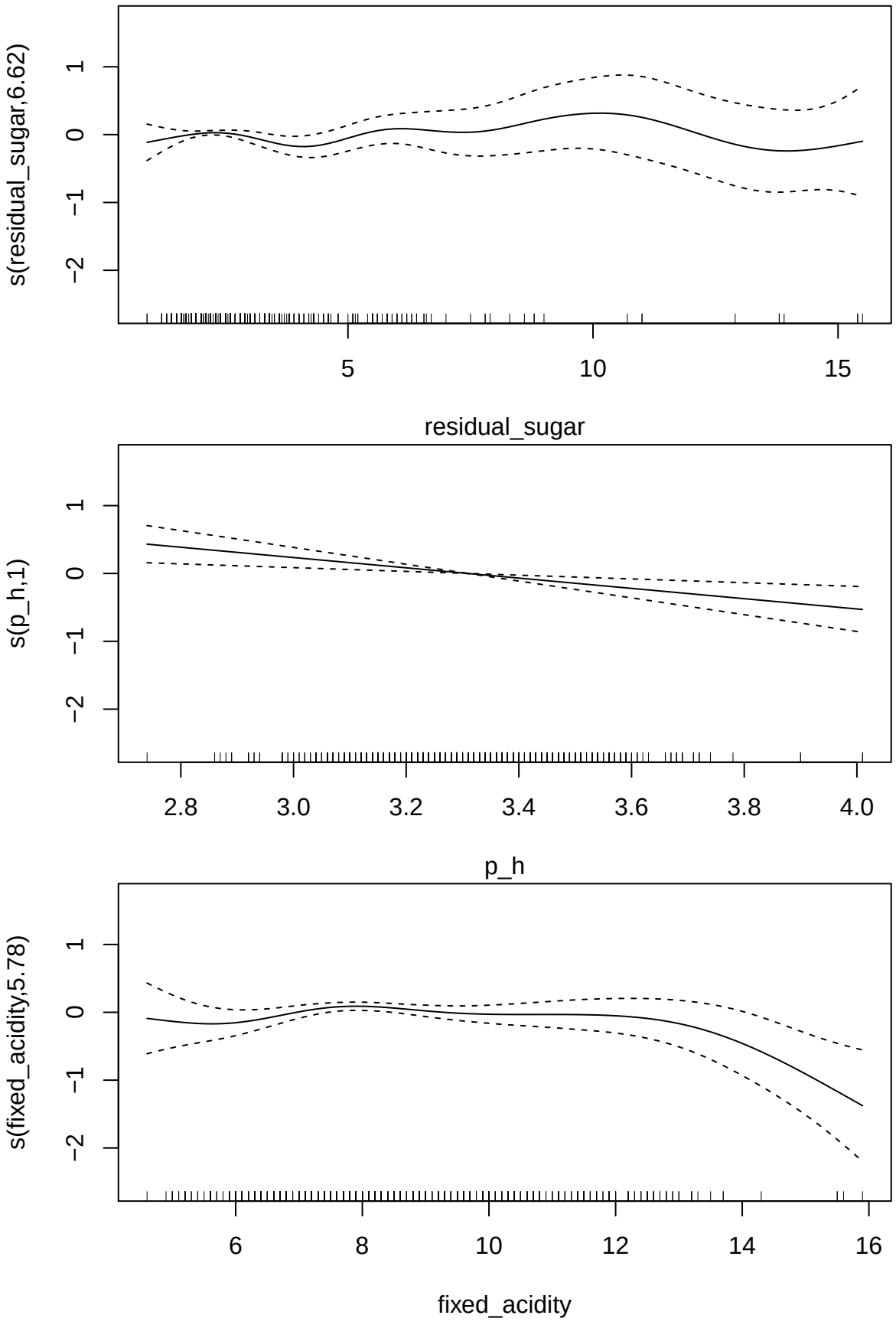
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ s(free_sulfur_dioxide) + s(alcohol) + s(citric_acid) +
##     s(residual_sugar) + s(p_h) + s(fixed_acidity) + s(sulphates) +
##     s(volatile_acidity) + s(chlorides) + s(total_sulfur_dioxide) +
##     s(density)
##
## Estimated degrees of freedom:
## 2.54 3.47 3.04 6.62 1.00 5.78 3.74
## 2.25 6.65 7.37 1.61  total = 45.07
##
## GCV score: 0.40868
```

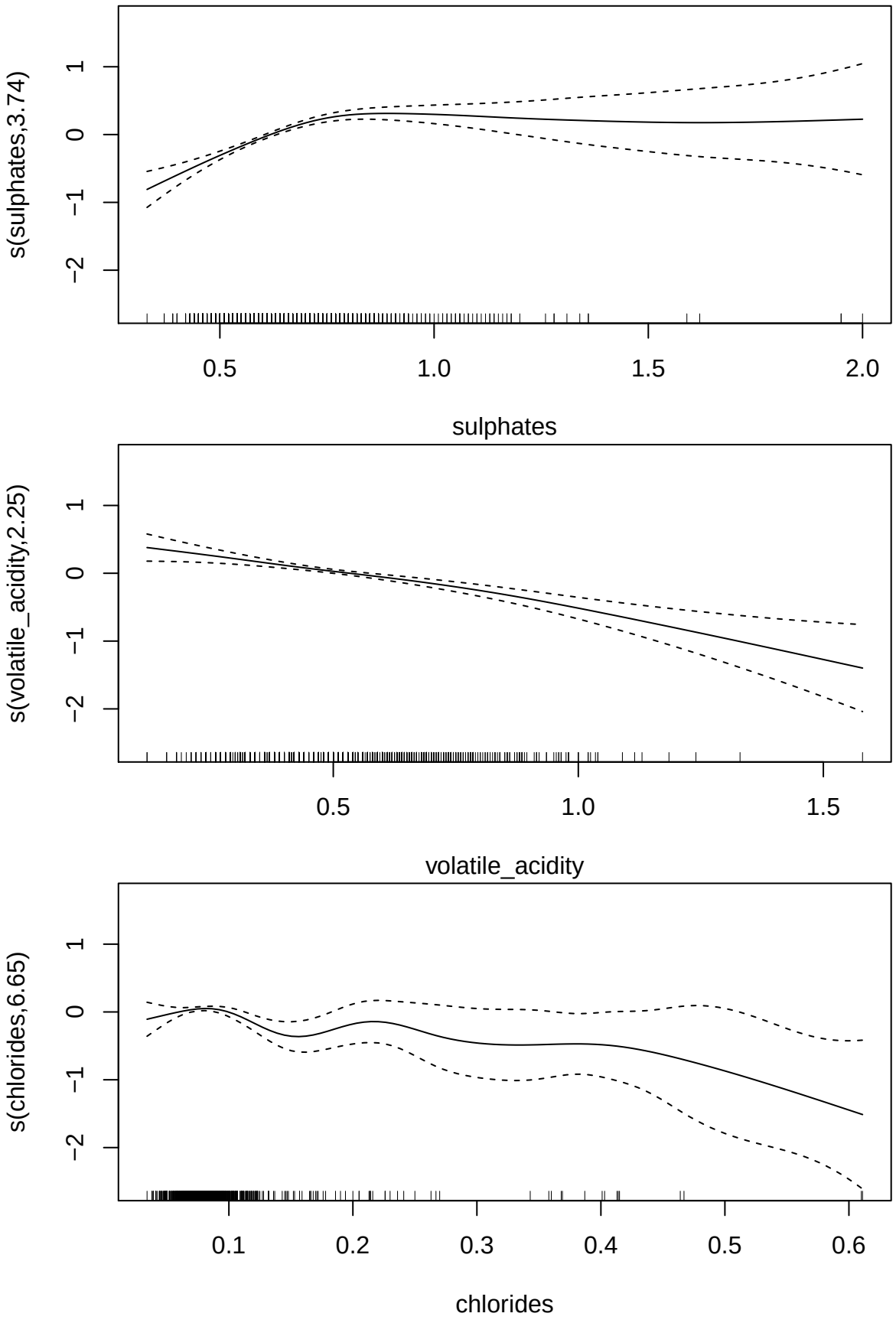
```
plot(gam.fit$finalModel)
```

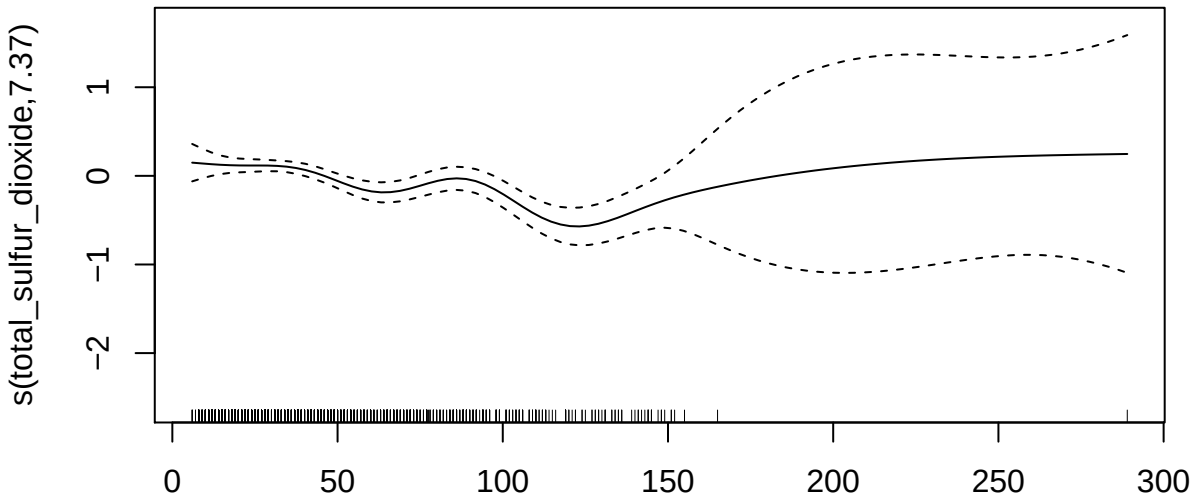

s(free_sulfur_dioxide,2.54)
1
0
−1
−2
0
10
20
30
40
50
60
70

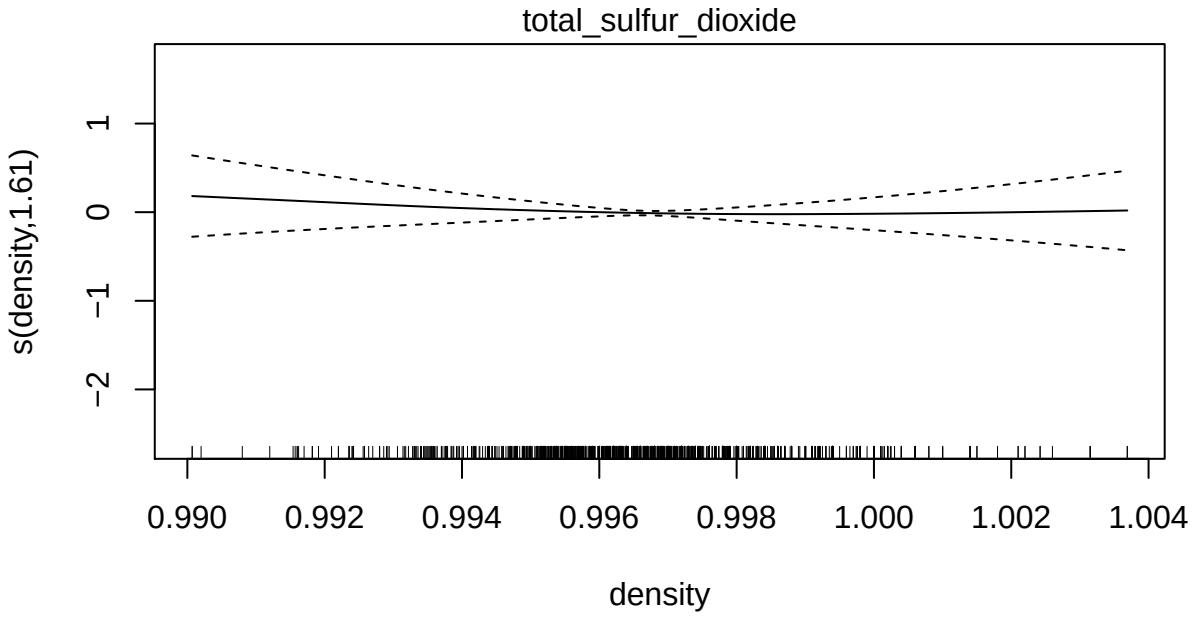

free_sulfur_dioxide
s(alcohol,3.47)
1
0
−1
−2
9
10
11
12
13
14
15


alcohol
s(citric_acid,3.04)
1
0
−1
−2
0.0
0.2
0.4
0.6
0.8
1.0
citric_acid


s(residual_sugar,6.62)
1
0
−1
−2
5
10
15
residual_sugar
s(p_h,1)
1
0
−1
−2
2.8
3.0
3.2
3.4
3.6
3.8
4.0
p_h
s(fixed_acidity,5.78)
1
0
−1
−2
6
8
10
12
14
16
fixed_acidity


s(sulphates,3.74)
1
0
−1
−2
0.5
1.0
1.5
2.0
sulphates
s(volatile_acidity,2.25)
1
0
−1
−2
0.5
1.0
1.5
volatile_acidity
s(chlorides,6.65)
1
0
−1
−2
0.1
0.2
0.3
0.4
0.5
0.6
chlorides

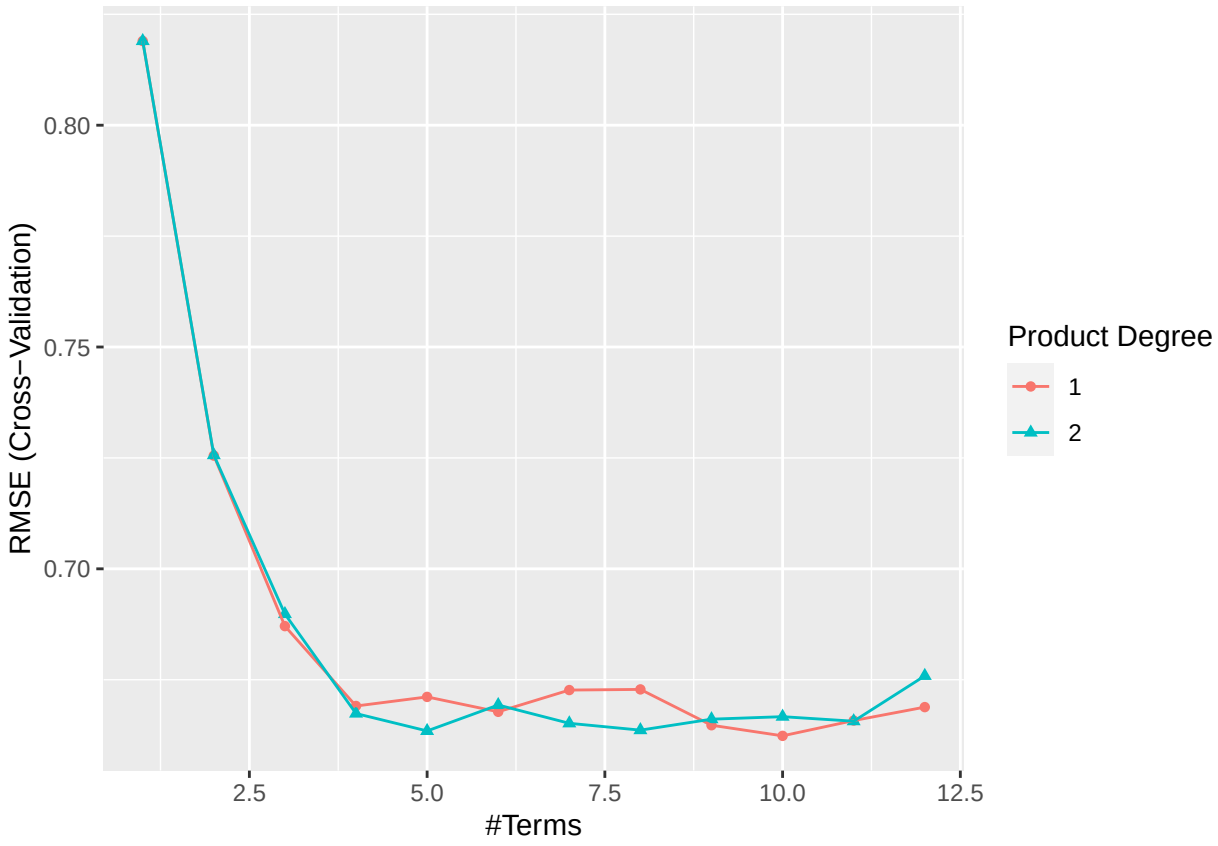





```
# MARS R^2 = 0.4271106
mars_grid <- expand.grid(degree = 1:2,
                         nprune = 1:12)

set.seed(11)
mars.fit <- train(x1, y1,
                  method = "earth",
                  tuneGrid = mars_grid,
                  trControl = ctrl1)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```
ggplot(mars.fit)
```




```
summary(mars.fit)
```

```
## Call: earth(x=matrix[1200,11], y=c(5,5,5,5,5,5,7...), keepxy=TRUE, degree=1,
##             nprune=10)
##
##                              coefficients
## (Intercept)                     5.7870745
## h(fixed_acidity-13)            -0.5234070
## h(0.84-volatile_acidity)        0.6671854
## h(volatile_acidity-0.84)       -2.2825458
## h(chlorides-0.076)             -2.1019573
## h(13-free_sulfur_dioxide)      -0.0242403
## h(130-total_sulfur_dioxide)     0.0044658
## h(3.68-p_h)                     0.6352655
## h(0.75-sulphates)              -2.2140074
## h(12.4-alcohol)                -0.2935155
##
## Selected 10 of 21 terms, and 8 of 11 predictors
## Termination condition: Reached nk 23
## Importance: alcohol, sulphates, volatile_acidity, total_sulfur_dioxide, ...
## Number of terms at each degree of interaction: 1 9 (additive model)
## GCV 0.4199783    RSS 488.1412    GRSq 0.3760728    RSq 0.3946656
```

```
mars.fit$bestTune
```

```
##    nprune degree
```

```
## 10     10      1
```

```
coef(mars.fit$finalModel)
```

```
##                 (Intercept)             h(12.4-alcohol)
##                5.787074493               -0.293515451
##    h(volatile_acidity-0.84)    h(0.84-volatile_acidity)
##               -2.282545844                0.667185395
##          h(0.75-sulphates) h(130-total_sulfur_dioxide)
##               -2.214007395                0.004465848
##         h(chlorides-0.076)                h(3.68-p_h)
##               -2.101957322                0.635265464
##        h(fixed_acidity-13)   h(13-free_sulfur_dioxide)
##               -0.523407009               -0.024240344
```

Look R-squared

MSE

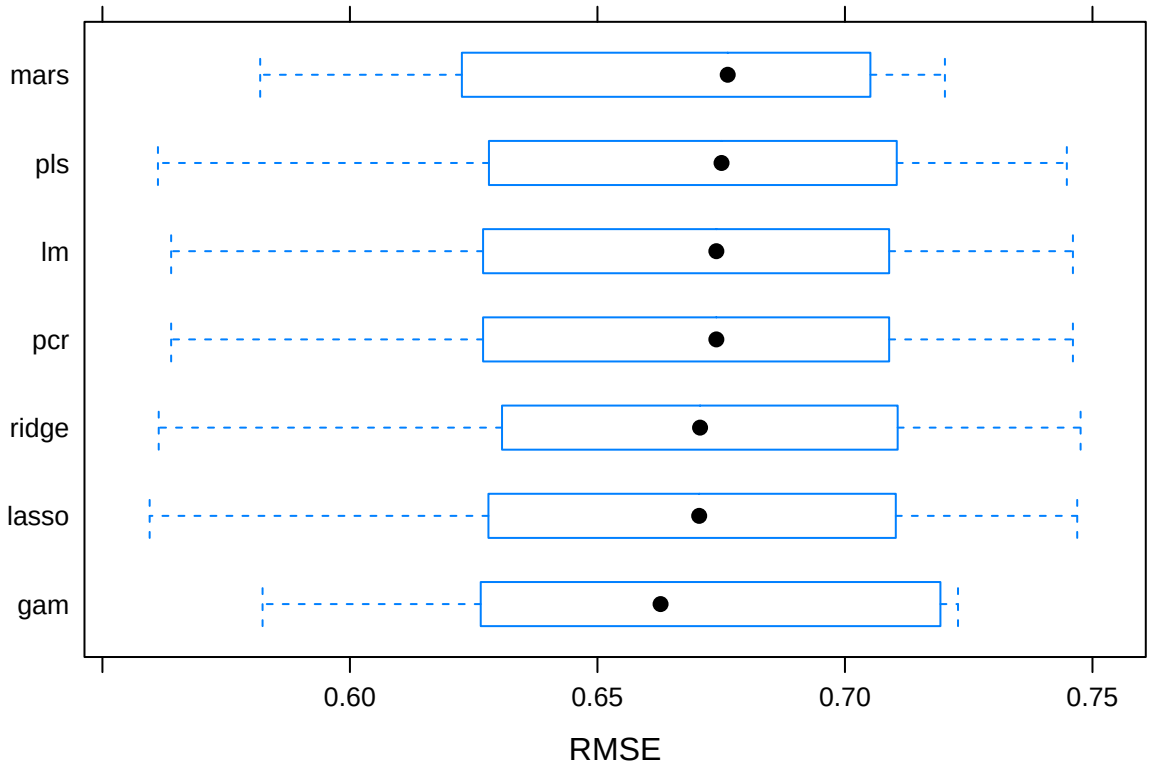
Resample

```
resamp = resamples(list(lm = lm.fit, ridge = ridge.fit, lasso = lasso.fit, mars = mars.fit, gam = gam.fi

summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: lm, ridge, lasso, mars, gam, pcr, pls
## Number of resamples: 10
##
## MAE
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lm    0.4506125 0.4795513 0.5225250 0.5155904 0.5506562 0.5767376    0
## ridge 0.4535358 0.4805494 0.5242750 0.5162212 0.5517084 0.5762671    0
## lasso 0.4543123 0.4798923 0.5228792 0.5151521 0.5507662 0.5754739    0
## mars  0.4613556 0.4737152 0.5252469 0.5105484 0.5350938 0.5579305    0
## gam   0.4660253 0.4786226 0.5201365 0.5112027 0.5336183 0.5539176    0
## pcr   0.4506125 0.4795513 0.5225250 0.5155904 0.5506562 0.5767376    0
## pls   0.4500726 0.4794441 0.5230510 0.5151372 0.5503779 0.5754380    0
##
## RMSE
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lm    0.5638848 0.6277267 0.6740203 0.6668210 0.7087648 0.7460513    0
## ridge 0.5613739 0.6309932 0.6707316 0.6665864 0.7093545 0.7476052    0
## lasso 0.5595402 0.6283752 0.6705535 0.6654834 0.7086140 0.7469167    0
## mars  0.5818724 0.6267059 0.6763165 0.6623480 0.7003578 0.7201965    0
## gam   0.5823484 0.6264522 0.6627544 0.6633735 0.7125415 0.7228416    0
## pcr   0.5638848 0.6277267 0.6740203 0.6668210 0.7087648 0.7460513    0
## pls   0.5612167 0.6285852 0.6750680 0.6666716 0.7102558 0.7448251    0
```

```
##
## Rsquared
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lm    0.2513610 0.2942118 0.3093839 0.3420745 0.4129358 0.4323412    0
## ridge 0.2453523 0.2983058 0.3097167 0.3421780 0.4073147 0.4376007    0
## lasso 0.2463093 0.3007179 0.3107483 0.3441676 0.4123104 0.4412550    0
## mars  0.2840932 0.3267065 0.3405847 0.3521095 0.3868996 0.4229414    0
## gam   0.2609263 0.3274838 0.3559511 0.3531965 0.3883876 0.4369075    0
## pcr   0.2513610 0.2942118 0.3093839 0.3420745 0.4129358 0.4323412    0
## pls   0.2487738 0.2912519 0.3112169 0.3424195 0.4112920 0.4375504    0
```

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



What predictor include

Model fitting method. assumptions. Treat the response as continous variable.

tuning parameter there is no tuning parameter

training test (MSE?) which var plays important role

limitation classification just treat response as continous we could do more flexible?