Machine Learning Report

Red Wine Quality Dataset Analysis

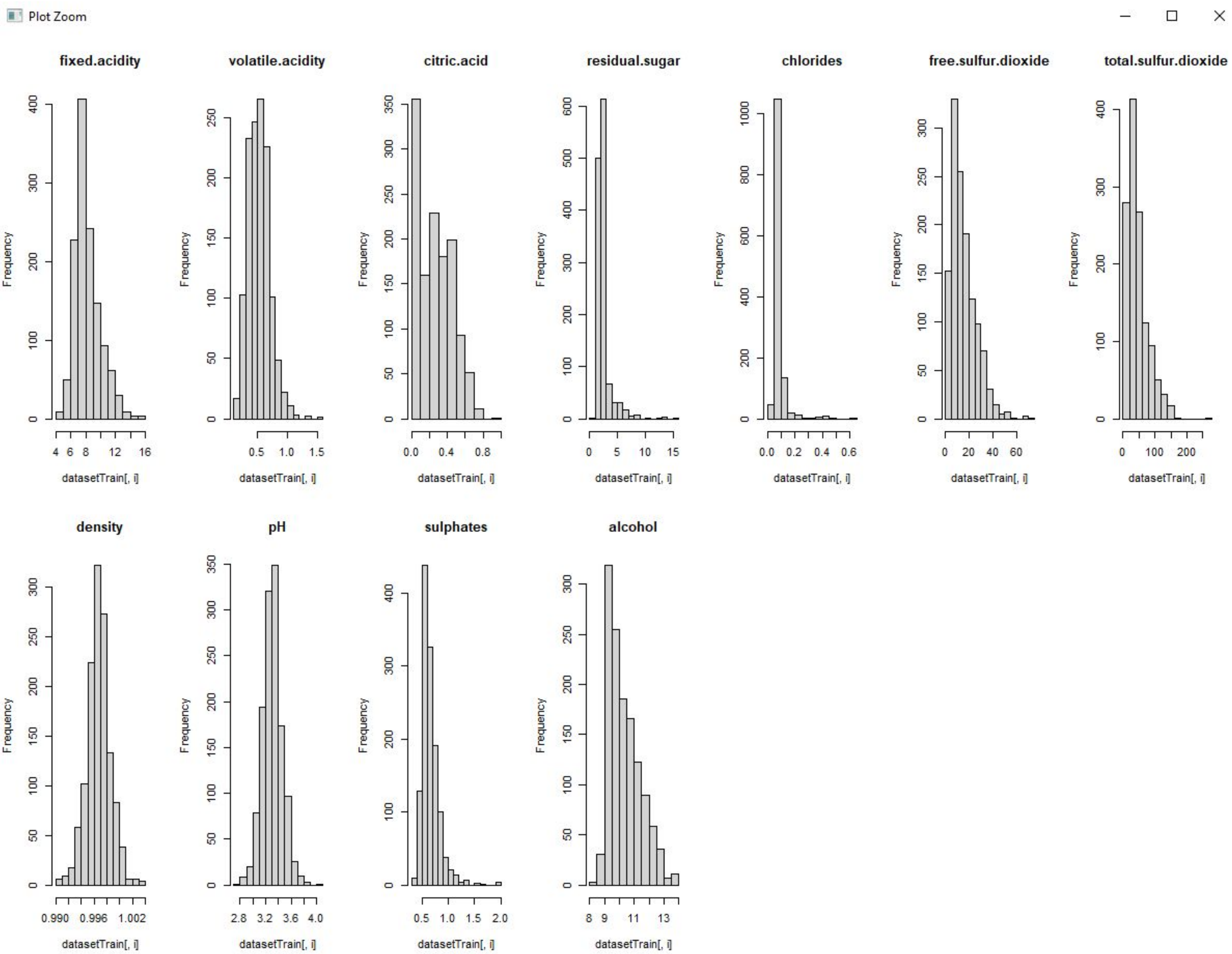
by Mihai Calin
CEN4 S1.A

Red Wine Quality Dataset
Description and Initial Observations

The Red Wine Quality Dataset provides us with a selection of red wines and their certain characteristics, which values vary from wine to wine, and each wine has a certain quality.

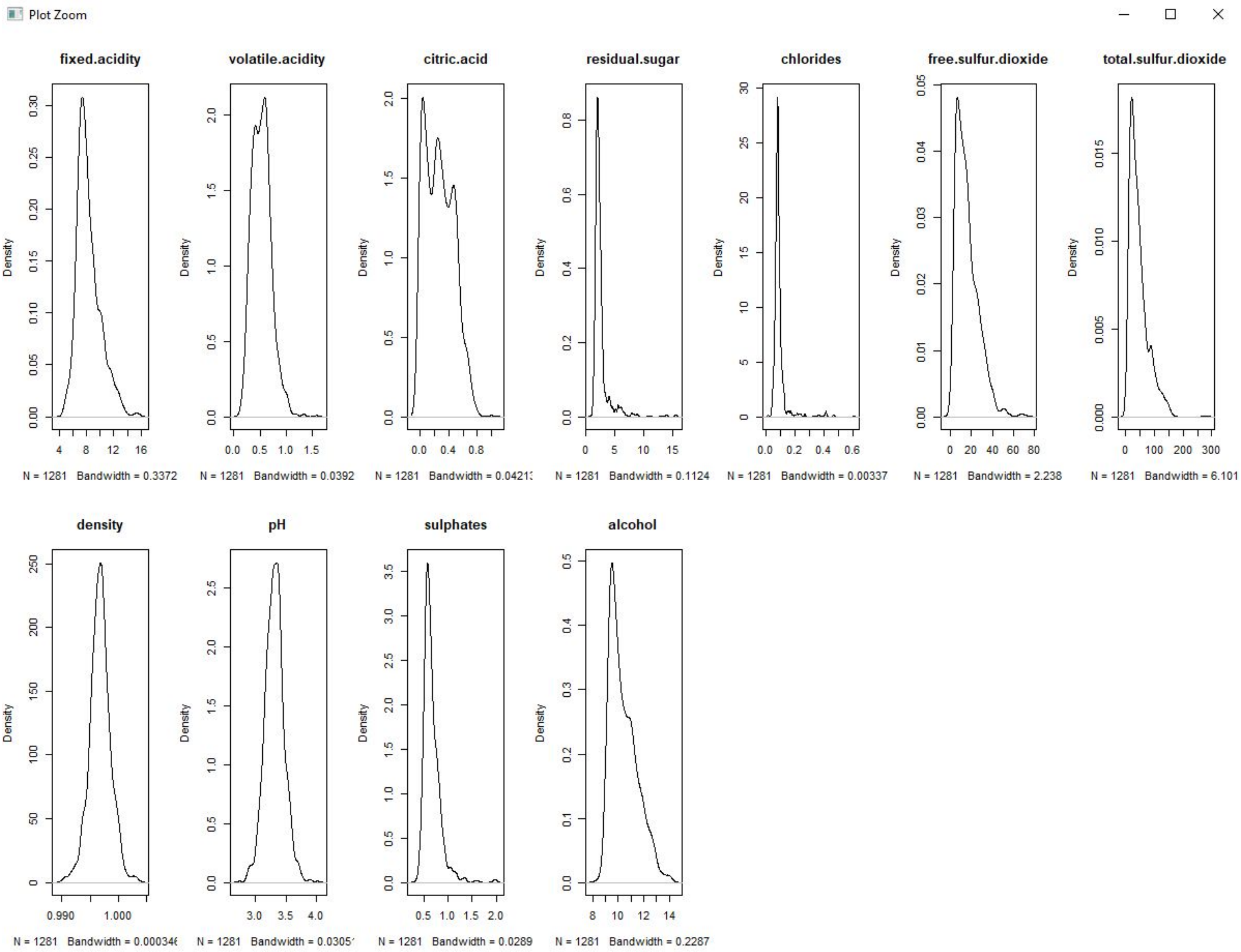
Based on the information available on the dataset, and using machine learning algorithms I'll try to train the data so that it'll predict the quality of the red wine with the least margin of error possible.

When it comes to the dataset, it's a reasonable large dataset, in which we have 12 variables (that make for the characteristics of our dear wines, ex: fixed acidity, volatile acidity, pH, residual sugar, quality etc).



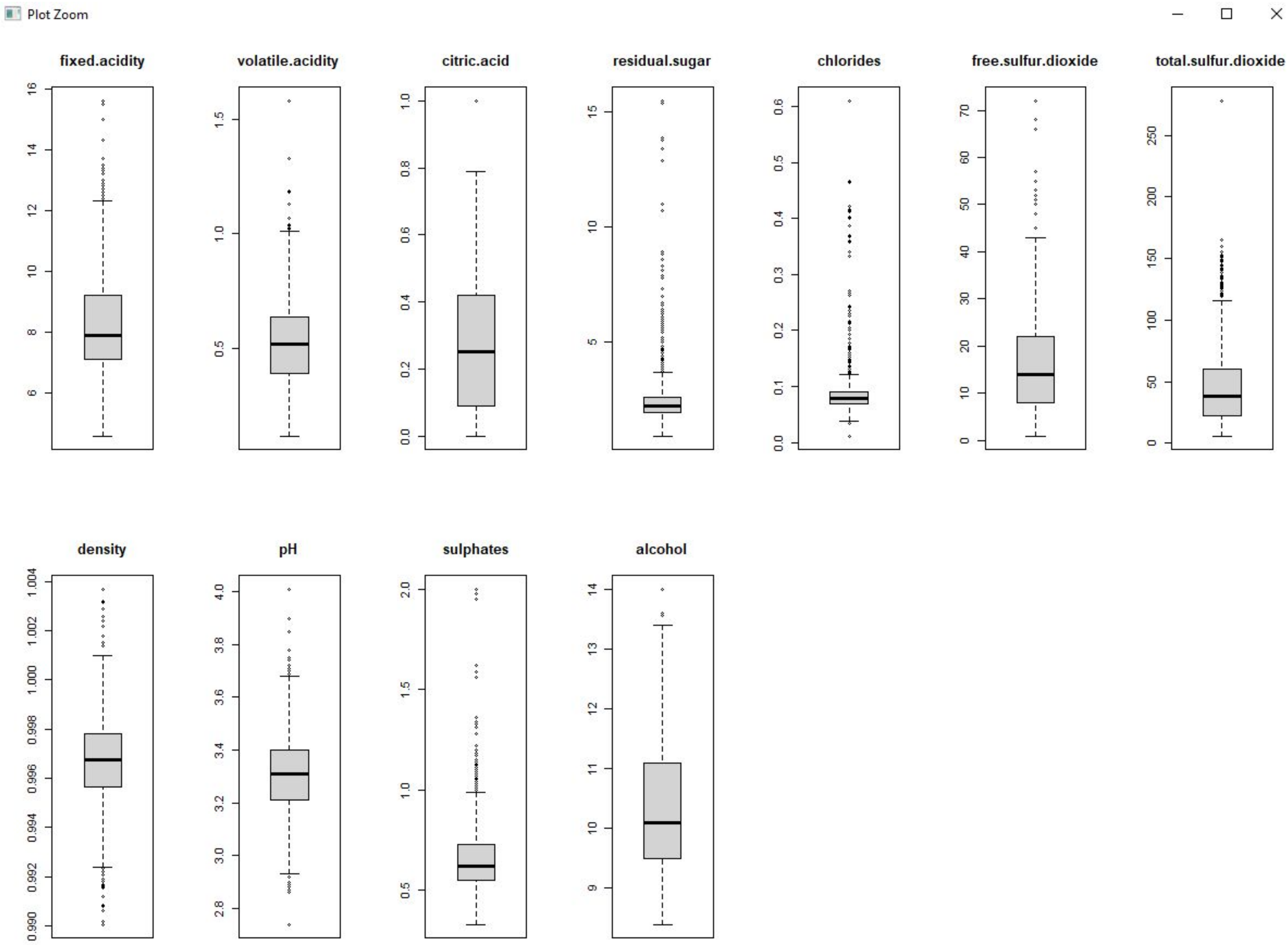
A short summary on the data:

- citric acid - seems to be somewhat uniformly distributed
- residual sugar - has a min - 0.9, and a max - 15, which is a massive difference
- chlorides - same as residual sugar, min - 0.012, max - 0.611
- free sulfur dioxide, total sulfur dioxide - same explanation as above



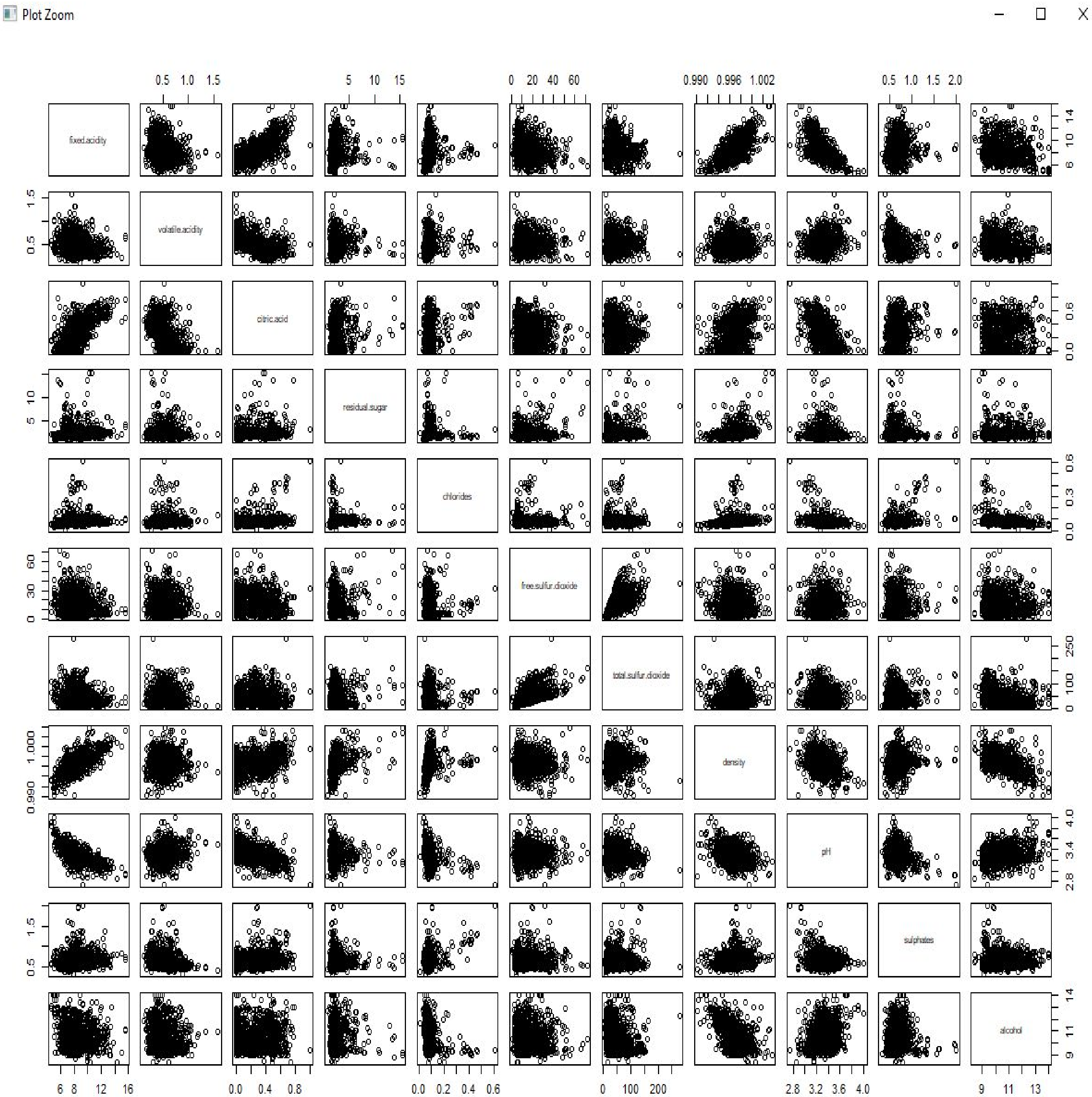
A short discussion on the results regarding the outliers :

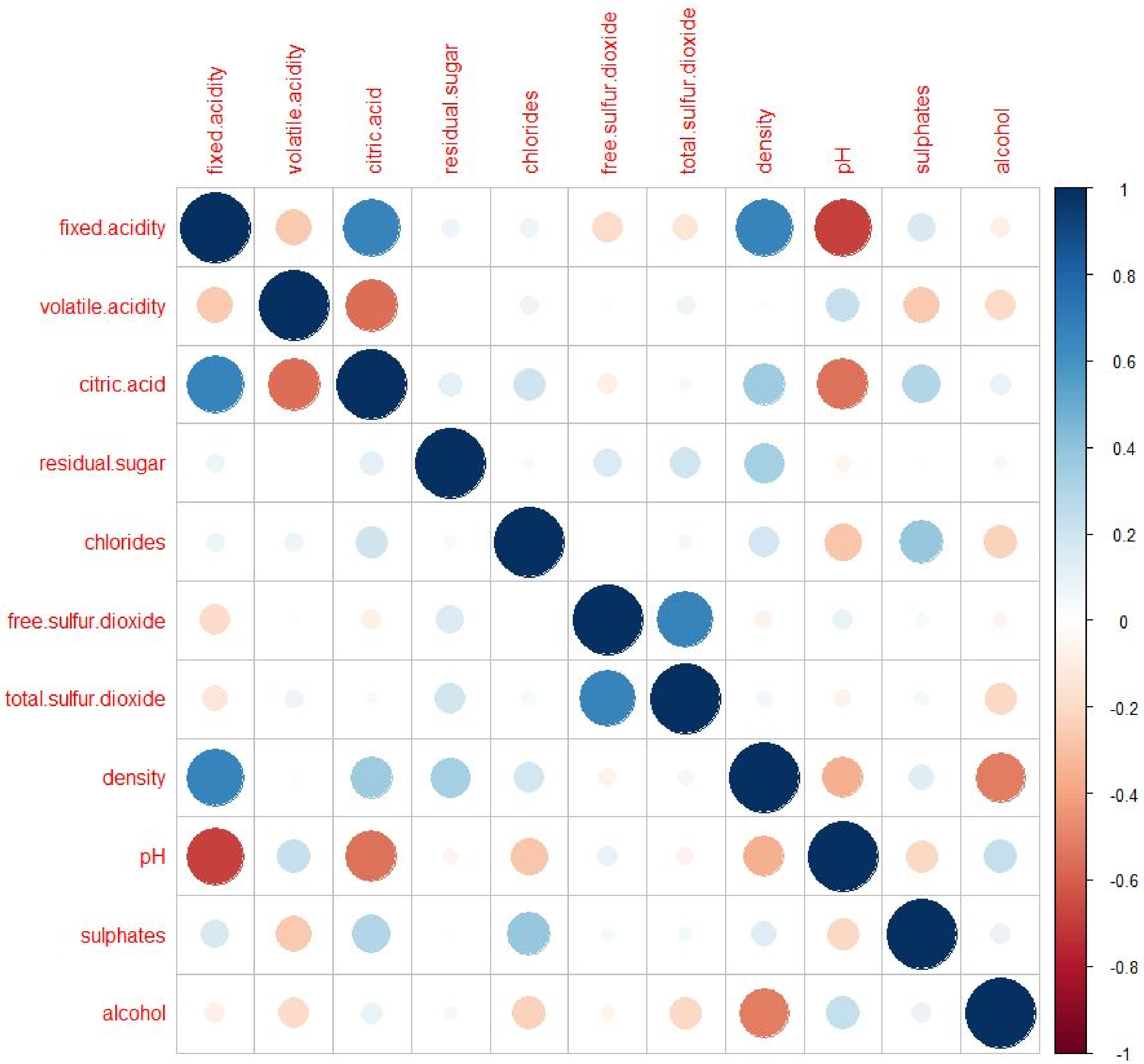
- residual sugar: extreme outliers on the higher values
- chlorides: same as above
- sulphates: same as above, but not as extreme
- total sulphur dioxide: there is a concentration of outliers on the higher values



Since the excess of outliers is a known factor that impedes proper prediction I'll be trying to eliminate or at least reduce them, but for now we'll go ahead and see the initial results, so we can determine what impact they have on our predictions.

Now let’s see the correlations:

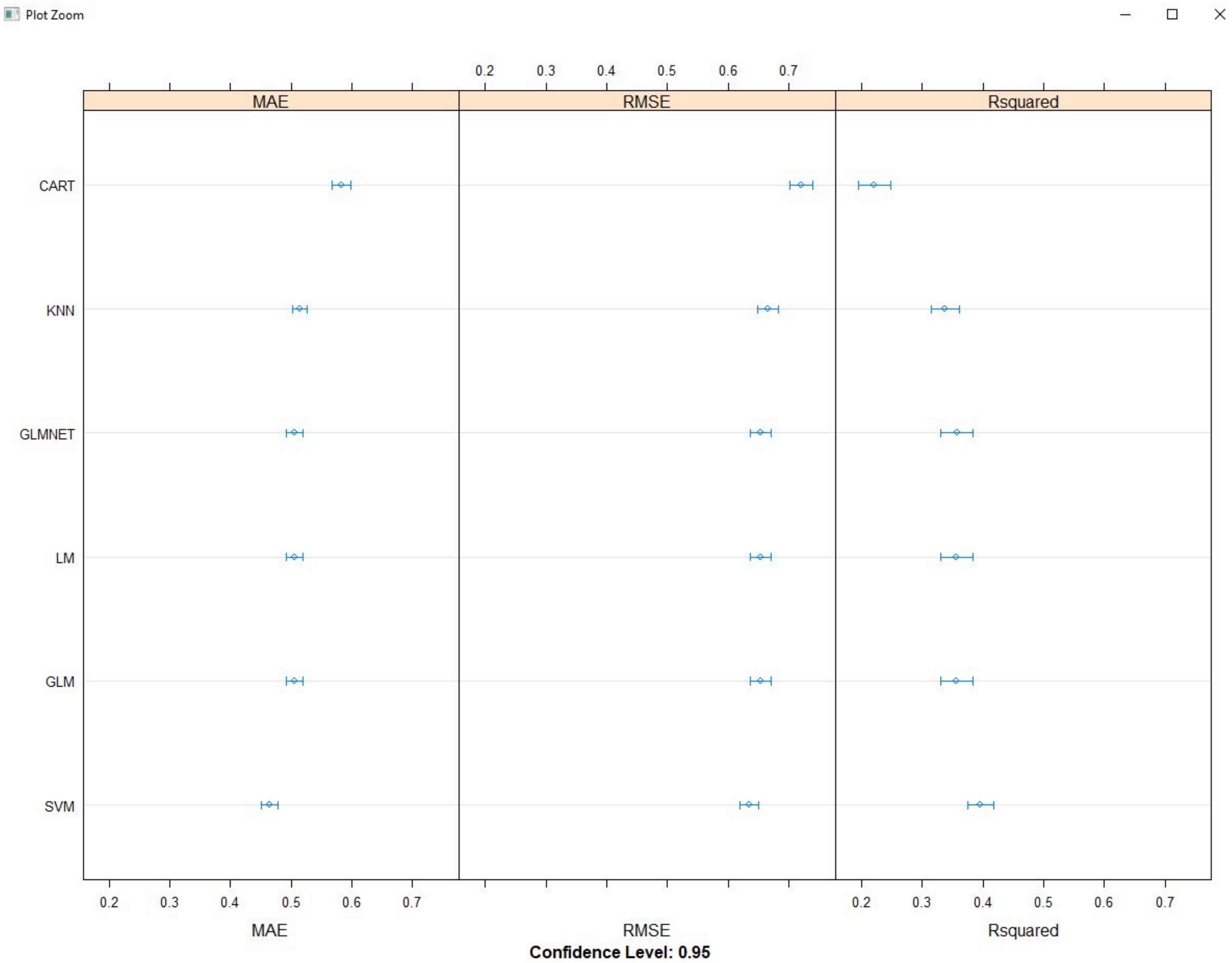




And the results are quite pleasant, because we don't have high correlations between variables, even though the fixed acidity and the free sulphur dioxide have a relatively higher correlation then the other variables.

Now to run the algorithms and begin the train of the data, I've split the dataset into 2 other datasets of different ration: one for training (80% of the data), and one for testing (20% of the data).

The result of the first run of the algorithms:



And the results seem promising, the margin of error is pretty small (RMSE), similarly with the median absolute error(MAE), which is good because smaller the two parameters are, then the precision of the predictions is going to be higher.

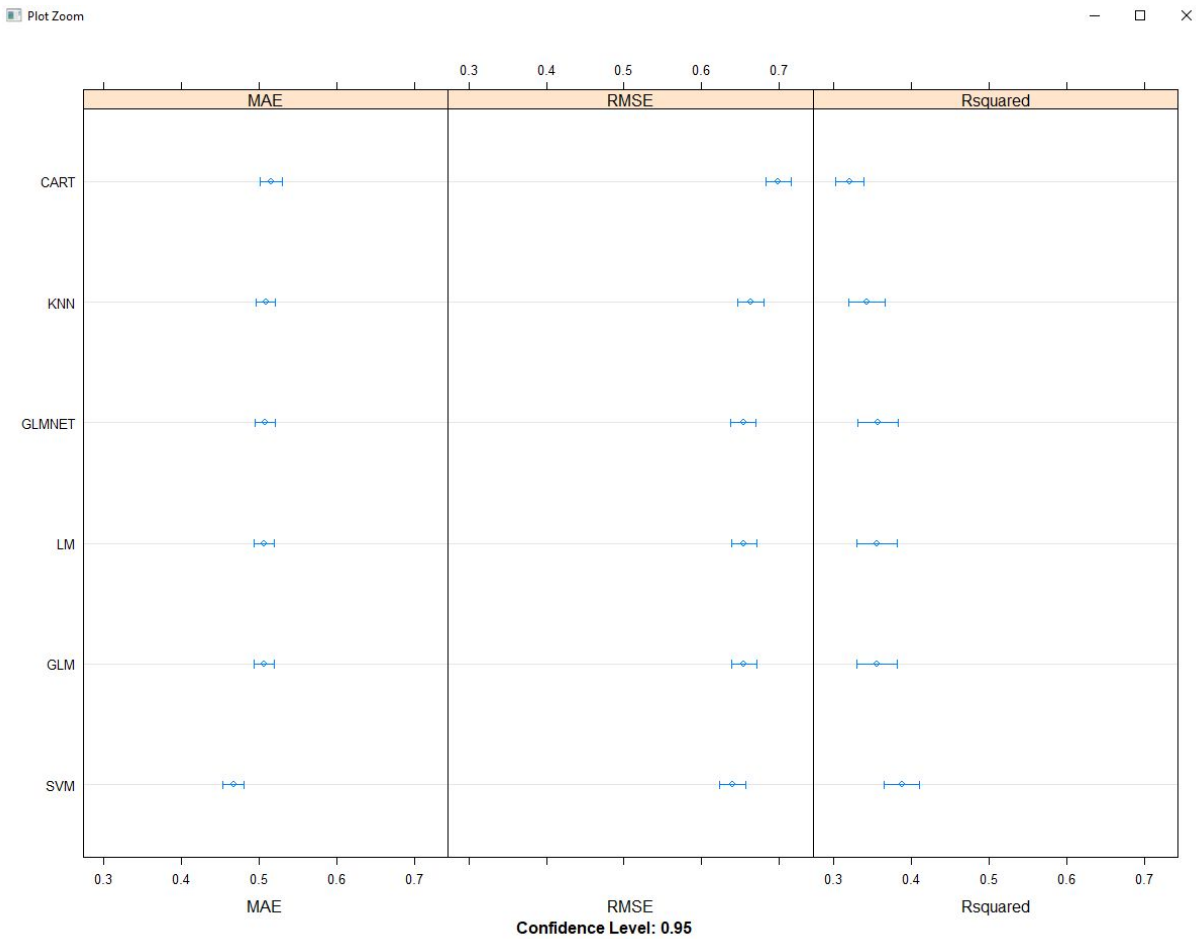
Rgarding the Rsquared, here the result is not so good since it's pretty close to 0, and the closer the value of Rsquared is to 1, the better the result.

Now I’ve tried to search for the variables that are highly correlated and eliminate them from the dataset, to see how that is going to affect the result.

I’ve slowly reduced the correlation factor for the cutoff, and only at 60% I found 2 variables that much correlation.

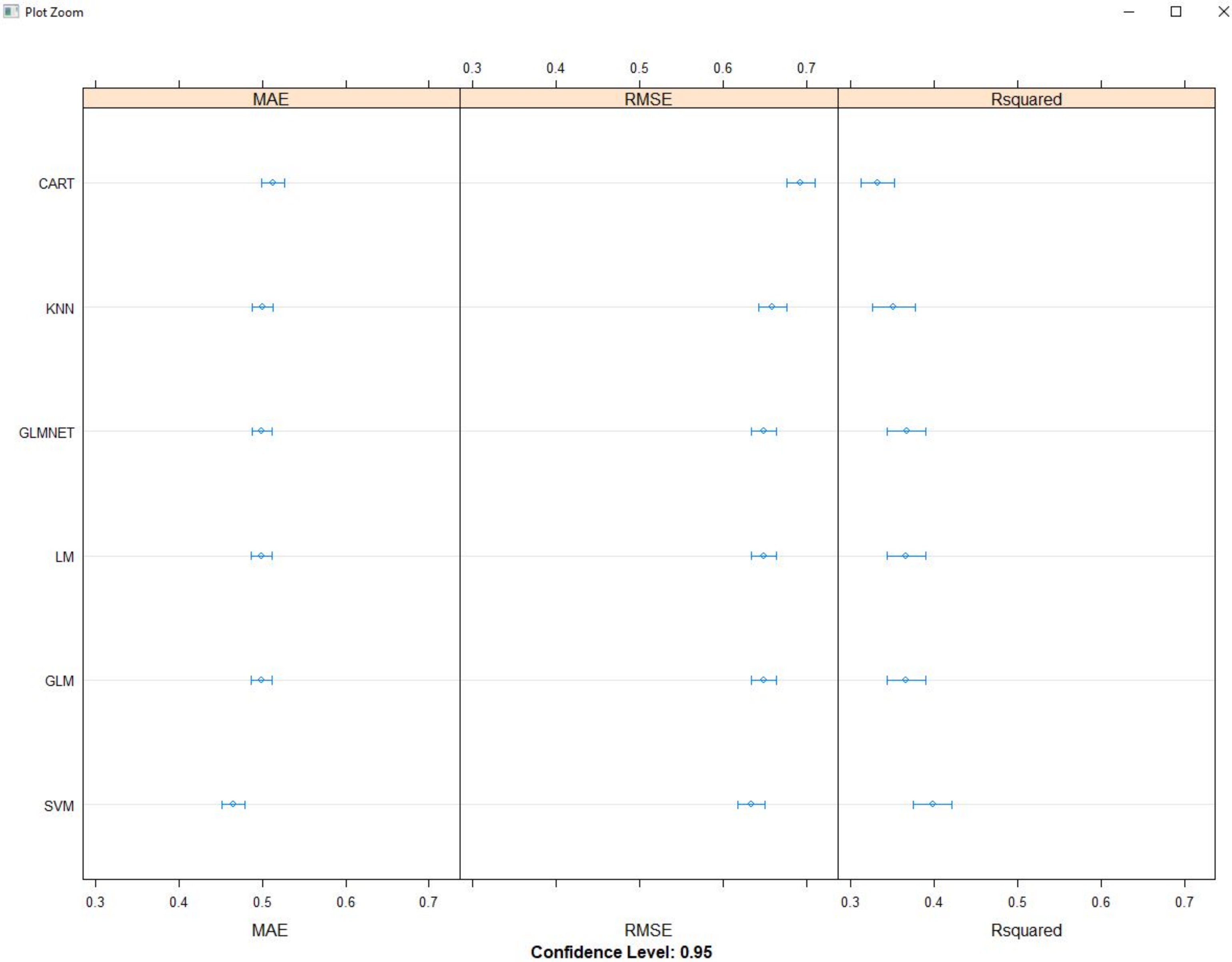
```
> # find attributes that are highly corrected
> set.seed(7)
> cutoff <- 0.60
> correlations <- cor(datasetTrain[,1:10])
> highlyCorrelated <- findCorrelation(correlations, cutoff=cutoff)
> for (value in highlyCorrelated) {
+   print(names(datasetTrain)[value])
+ }
[1] "fixed.acidity"
[1] "free.sulfur.dioxide"
> # create a new dataset without highly corrected features
> dataset_features <- datasetTrain[,-highlyCorrelated]
> dim(dataset_features)
[1] 1361    9
>
```

Now, the results of the algorithms on the new dataset that doesn’t include the highly correlated variables:



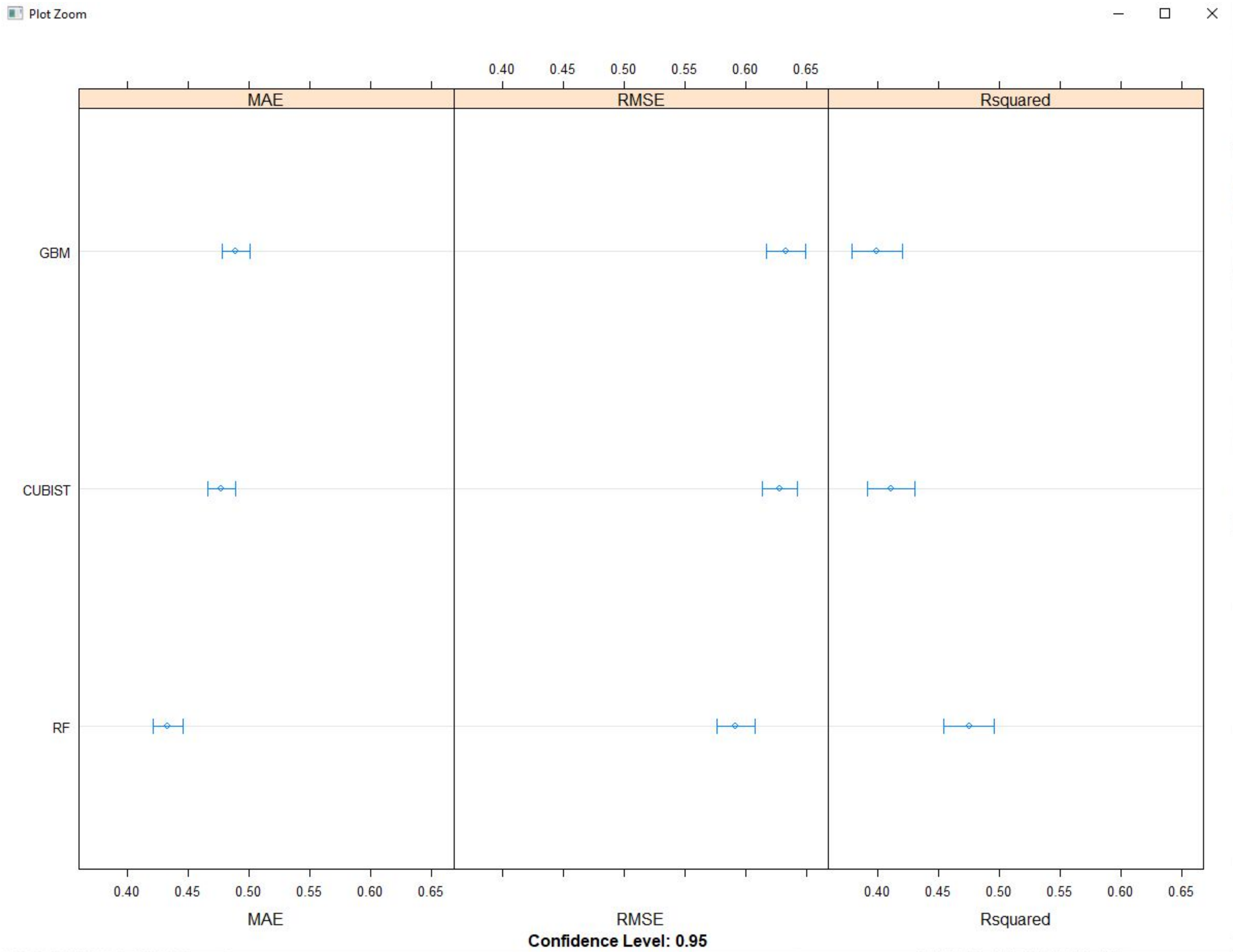
And the changes are insignificant, so we’ll proceed with the initial dataset.

Now, let's try to run the algorithms with the Box-Cox Transform:

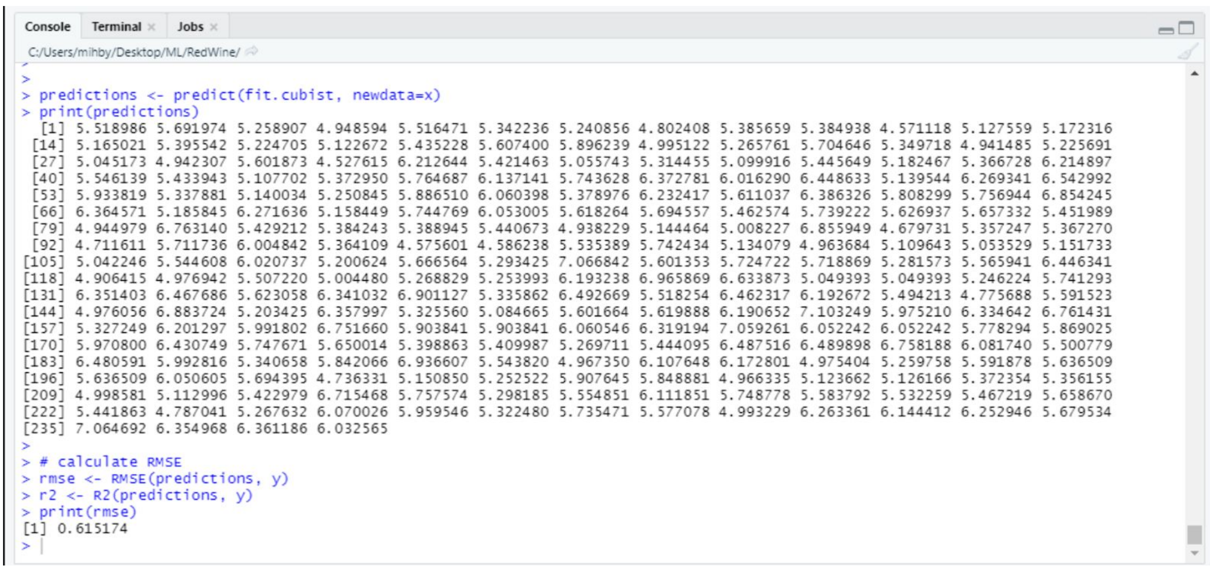


And, again the changes proved insignificant.

Now, let's try to run other algorithms, to see if it'll have any significant changes:

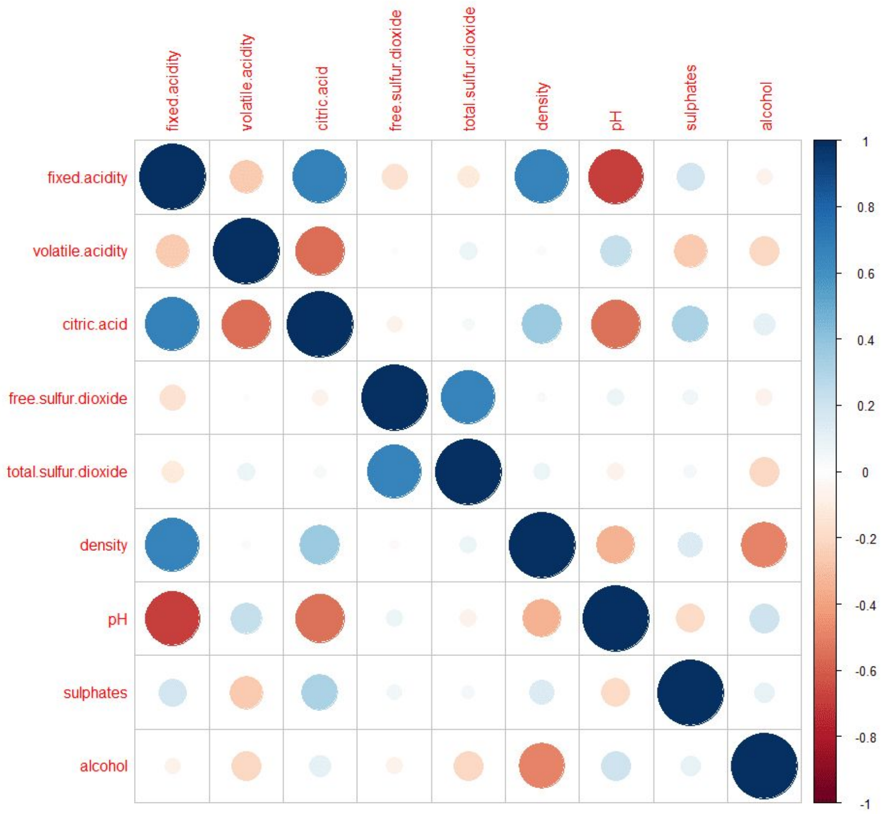


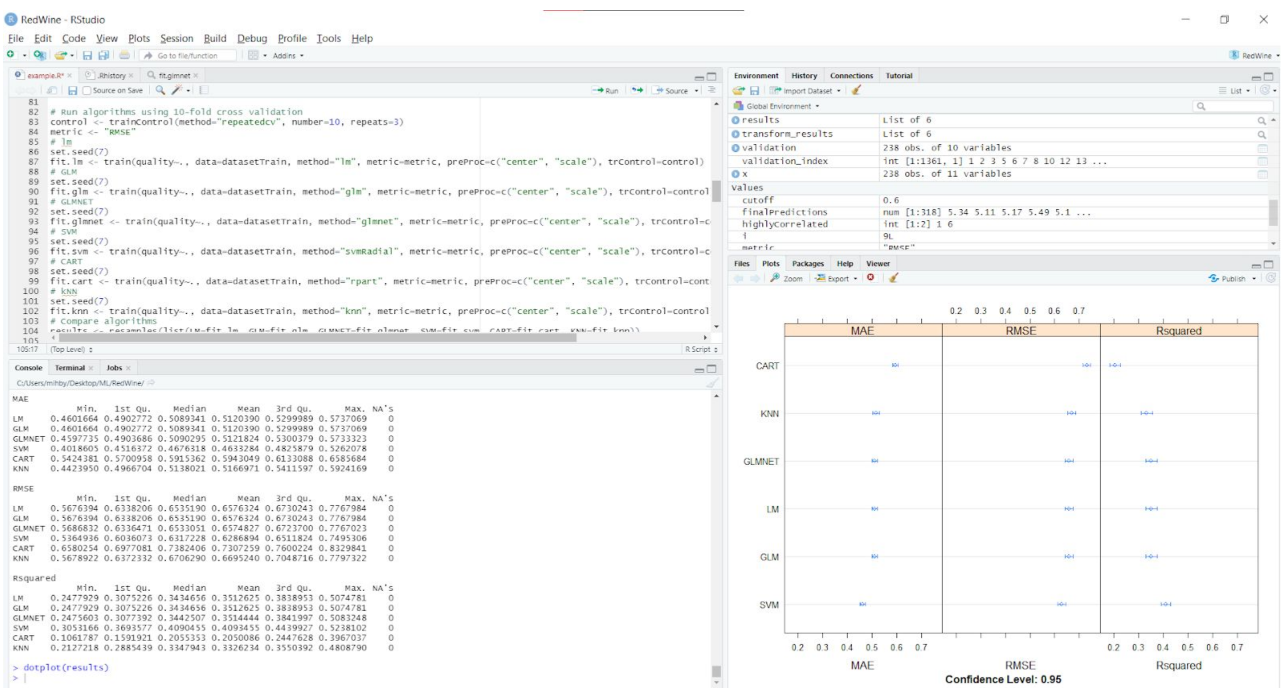
And now we have a better result, so I'll be using the this algorithm to run on new data (the remaining dataset that I've saved for testing) and see how far off are the predictions:



```
Console  Terminal  Jobs
C:/Users/mihby/Desktop/ML/RedWine/
>
> predictions <- predict(fit.cubist, newdata=x)
> print(predictions)
  [1] 5.518986 5.691974 5.258907 4.948594 5.516471 5.342236 5.240856 4.802408 5.385659 5.384938 4.571118 5.127559 5.172316
 [14] 5.165021 5.395542 5.224705 5.122672 5.435228 5.607400 5.896239 4.995122 5.265761 5.704646 5.349718 4.941485 5.225691
 [27] 5.045173 4.942307 5.601873 4.527615 6.212644 5.421463 5.055743 5.314455 5.099916 5.445649 5.182467 5.366728 6.214897
 [40] 5.546139 5.433943 5.107702 5.372950 5.764687 6.137141 5.743628 6.372781 6.016290 6.448633 5.139544 6.269341 6.542992
 [53] 5.933819 5.337881 5.140034 5.250845 5.886510 6.060398 5.378976 6.232417 5.611037 6.386326 5.808299 5.756944 6.854245
 [66] 6.364571 5.185845 6.271636 5.158449 5.744769 6.053005 5.618264 5.694557 5.462574 5.739222 5.626937 5.657332 5.451989
 [79] 4.944979 6.763140 5.429212 5.384243 5.388945 5.440673 4.938229 5.144464 5.008227 6.855949 4.679731 5.357247 5.367270
 [92] 4.711611 5.711736 6.004842 5.364109 4.575601 4.586238 5.535389 5.742434 5.134079 4.963684 5.109643 5.053529 5.151733
[105] 5.042246 5.544608 6.020737 5.200624 5.666564 5.293425 7.066842 5.601353 5.724722 5.718869 5.281573 5.565941 6.446341
[118] 4.906415 4.976942 5.507220 5.004480 5.268829 5.253993 6.193238 6.965869 6.633873 5.049393 5.049393 5.246224 5.741293
[131] 6.351403 6.467686 5.623058 6.341032 6.901127 5.335862 6.492669 5.518254 6.462317 6.192672 5.494213 4.775688 5.591523
[144] 4.976056 6.883724 5.203425 6.357997 5.325560 5.084665 5.601664 5.619888 6.190652 7.103249 5.975210 6.334642 6.761431
[157] 5.327249 6.201297 5.991802 6.751660 5.903841 5.903841 6.060546 6.319194 7.059261 6.052242 6.052242 5.778294 5.869025
[170] 5.970800 6.430749 5.747671 5.650014 5.398863 5.409987 5.269711 5.444095 6.487516 6.489898 6.758188 6.081740 5.500779
[183] 6.480591 5.992816 5.340658 5.842066 6.936607 5.543820 4.967350 6.107648 6.172801 4.975404 5.259758 5.591878 5.636509
[196] 5.636509 6.050605 5.694395 4.736331 5.150850 5.252522 5.907645 5.848881 4.966335 5.123662 5.126166 5.372354 5.356155
[209] 4.998581 5.112996 5.422979 6.715468 5.757574 5.298185 5.554851 6.111851 5.748778 5.583792 5.532259 5.467219 5.658670
[222] 5.441863 4.787041 5.267632 6.070026 5.959546 5.322480 5.735471 5.577078 4.993229 6.263361 6.144412 6.252946 5.679534
[235] 7.064692 6.354968 6.361186 6.032565
>
> # calculate RMSE
> rmse <- RMSE(predictions, y)
> r2 <- R2(predictions, y)
> print(rmse)
[1] 0.615174
>
```

To improve the results I've tried to eliminate the outliers, towards this goal I've removed the variables with the most outliers (chlorides and residual sugar) and compare the results:





And again, the changes are insignificant, we I'll proceed with the initial dataset.

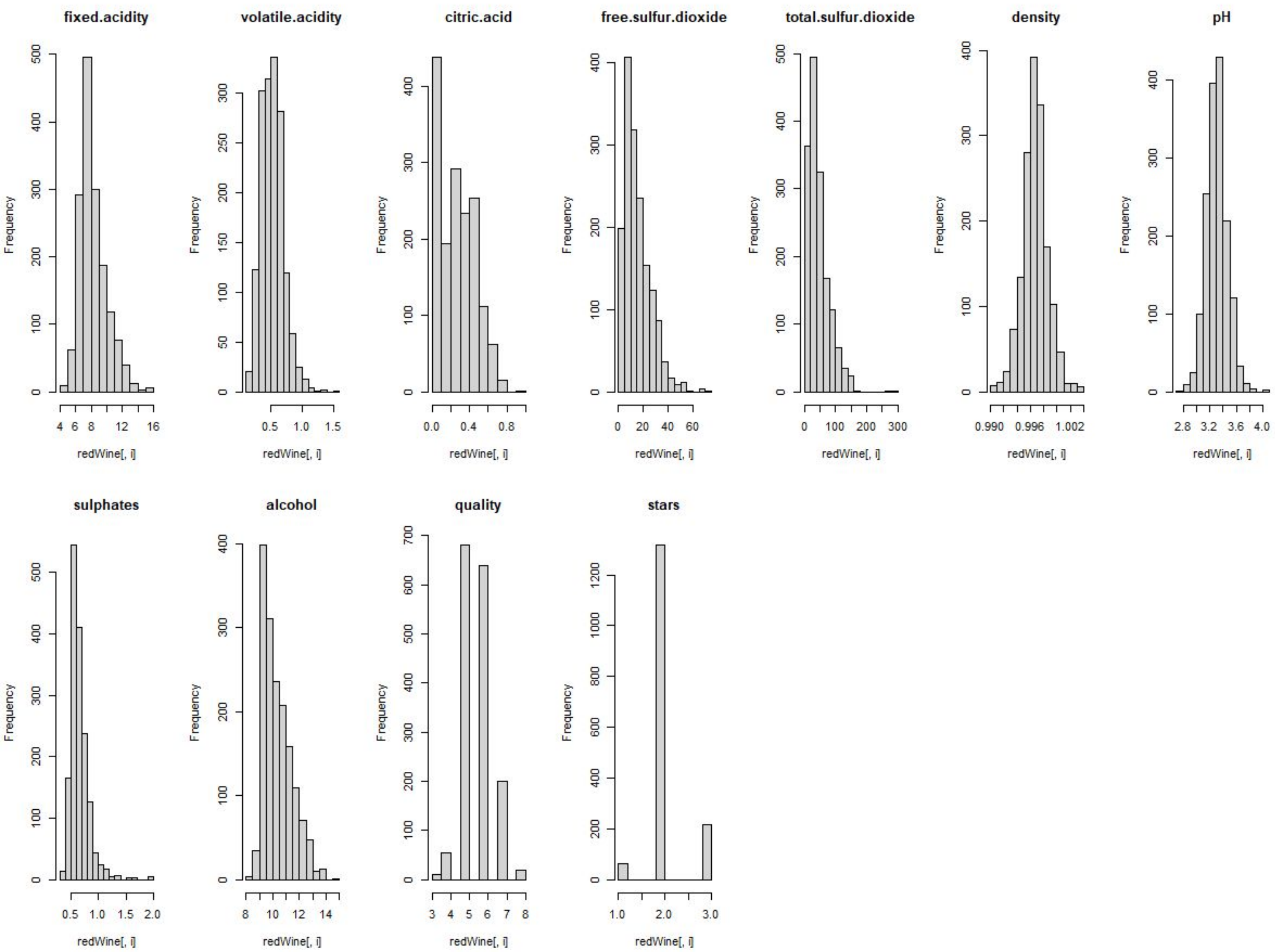
Hoping that I’ll improve the results I’ve changed the split ration of the two datasets (85/15) leaving more data for the training dataset.

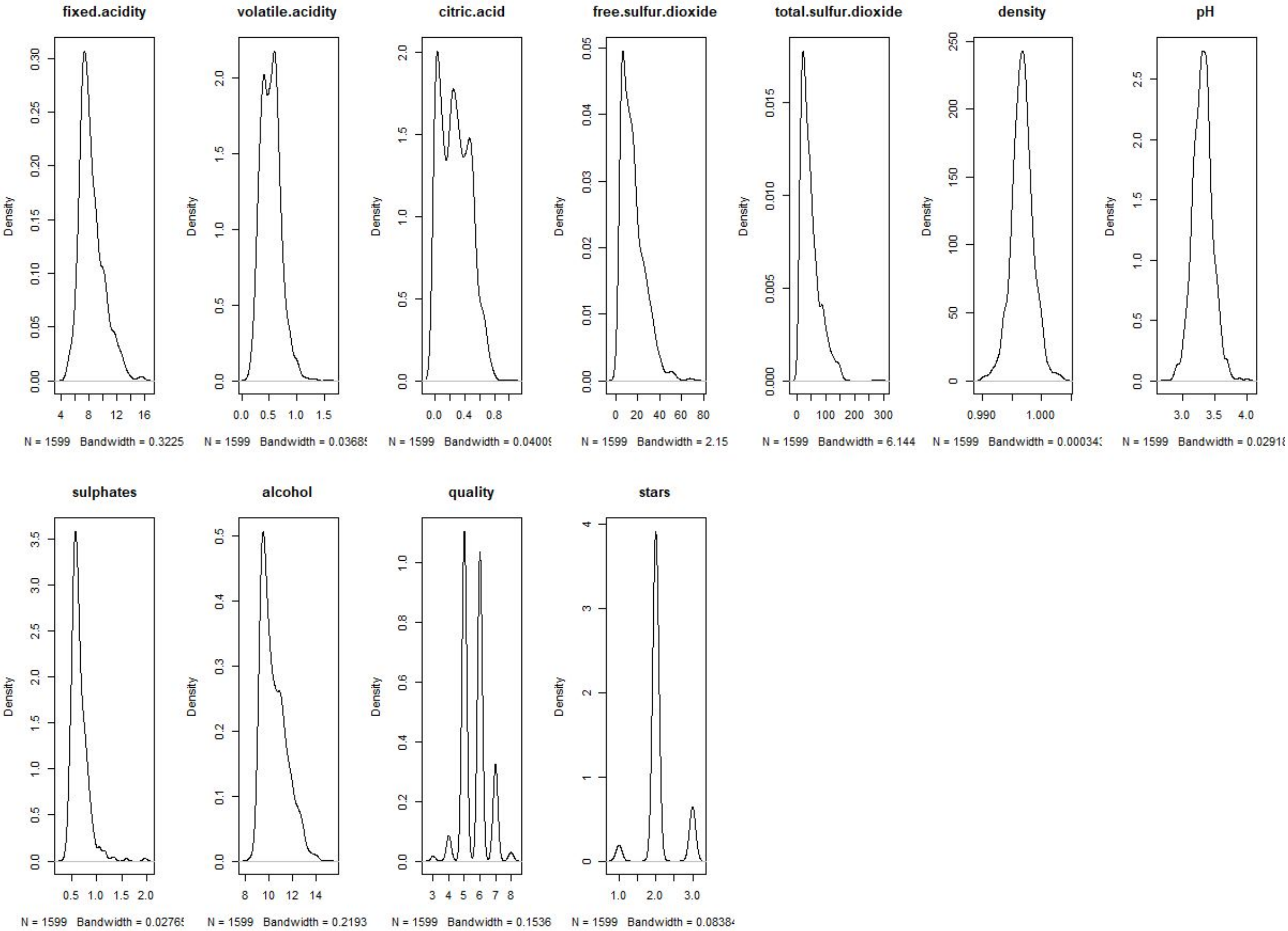
And I’ve changed the output variable, creating a new system of rating the quality of the red wines, the new variable is names “Stars” and has 3 possible values:

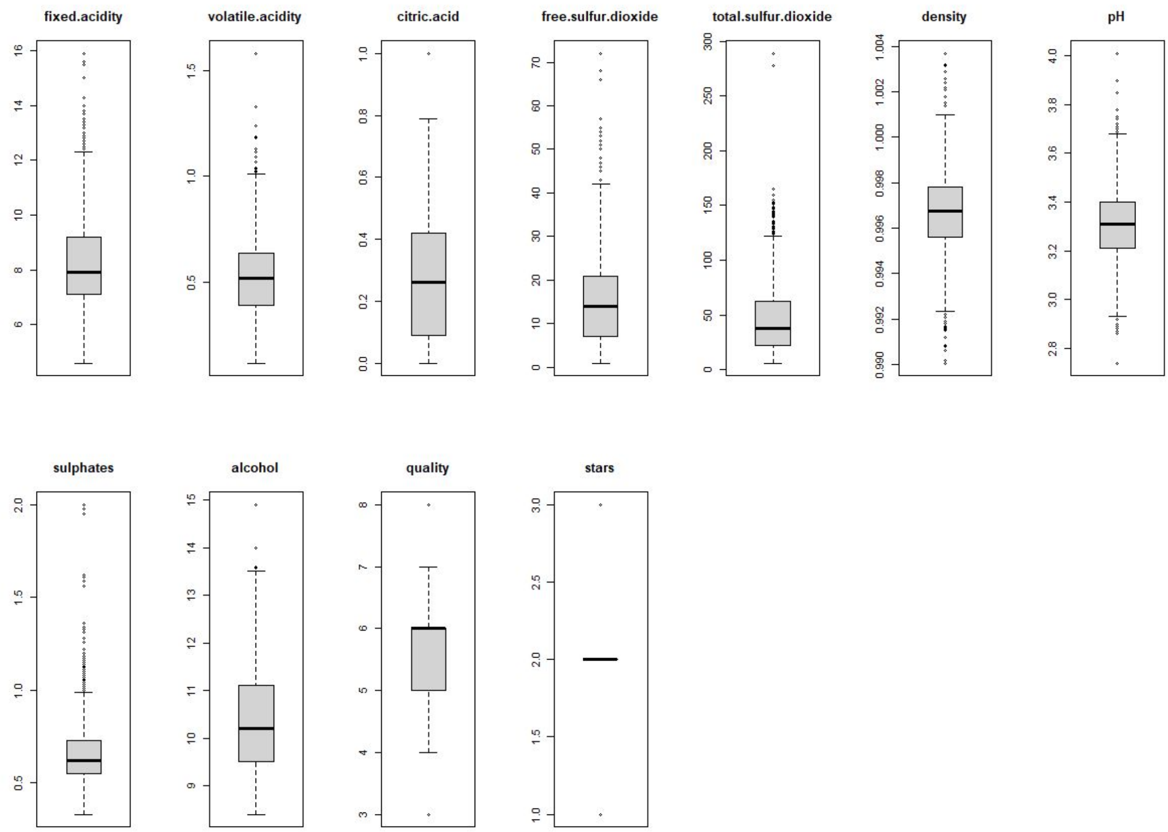
- 1 is for the wines that have a quality below 5 (poor quality)
- 2 is for the wines that have a quality between 5 and 7 (good quality)
- 3 is for the wines that have a quality over 7 (great quality)

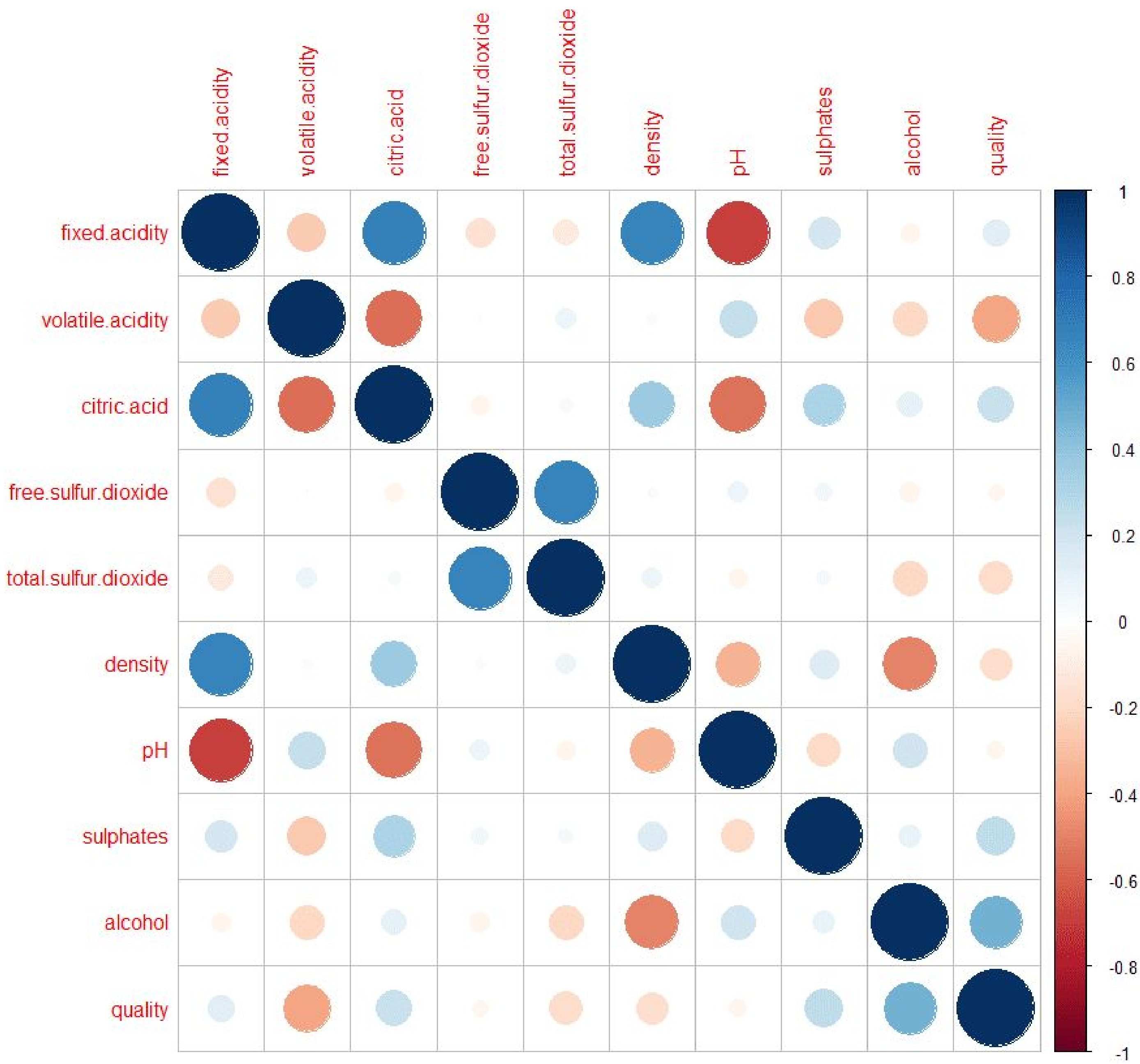
And with this addition we now have the “Quality” as an input variable that should provide valuable information.

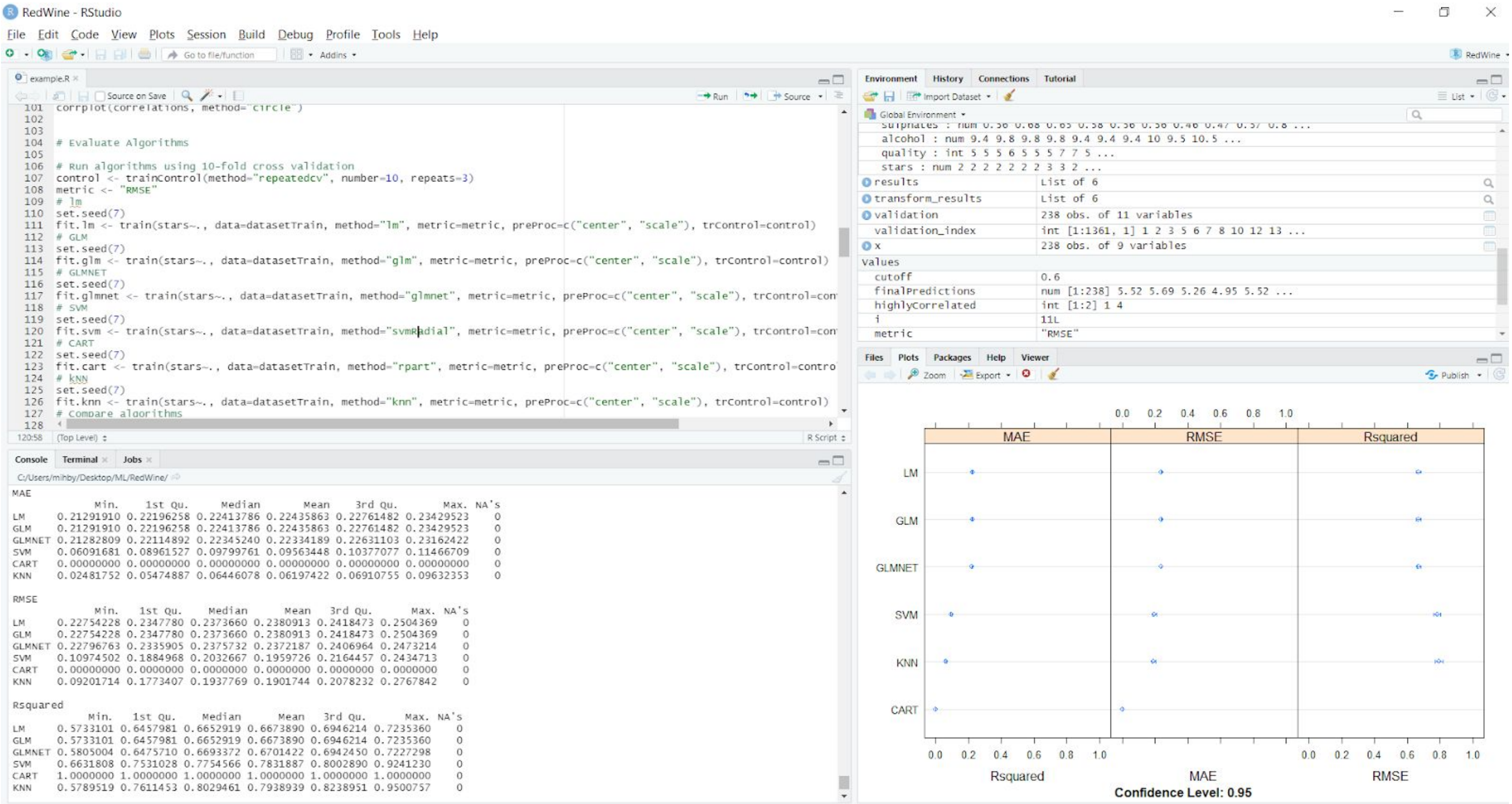
And now I’ll repeat the process and evaluate the changes.

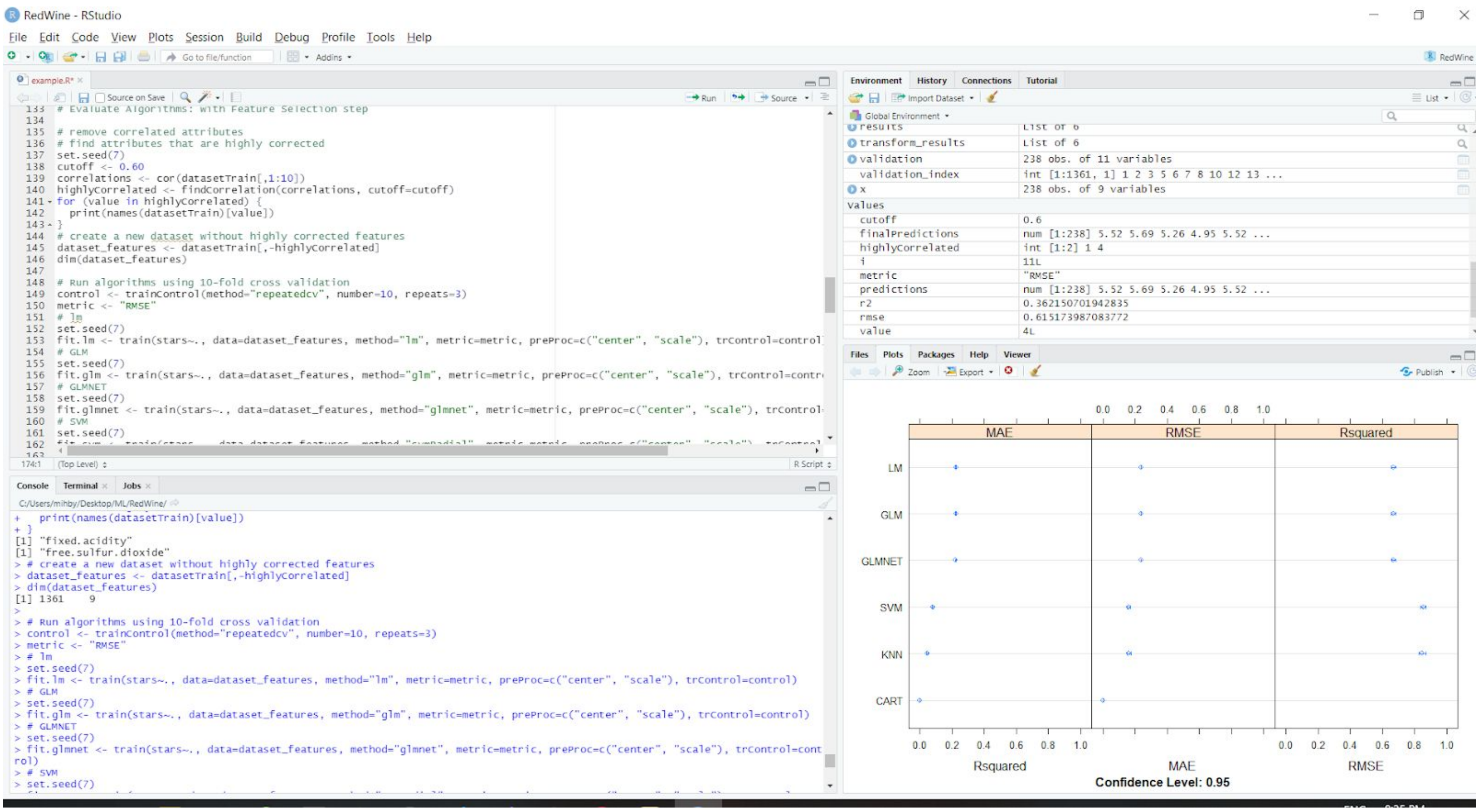


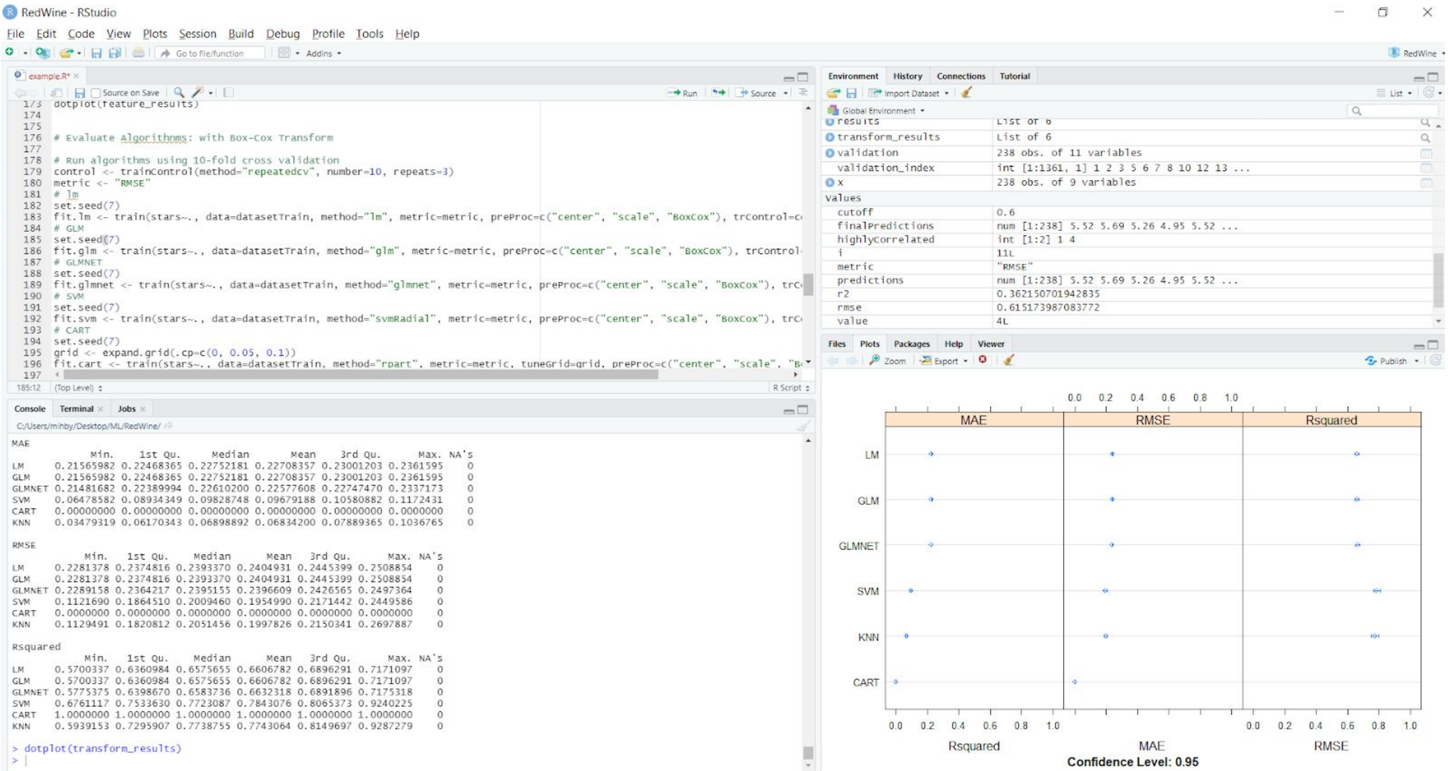

fixed.acidity
volatile.acidity
citric.acid
free.sulfur.dioxide
total.sulfur.dioxide
density
pH
sulphates
alcohol
quality
stars
Density
N = 1599 Bandwidth = 0.3225
N = 1599 Bandwidth = 0.03685
N = 1599 Bandwidth = 2.15
N = 1599 Bandwidth = 6.144
N = 1599 Bandwidth = 0.2193
N = 1599 Bandwidth = 0.1536

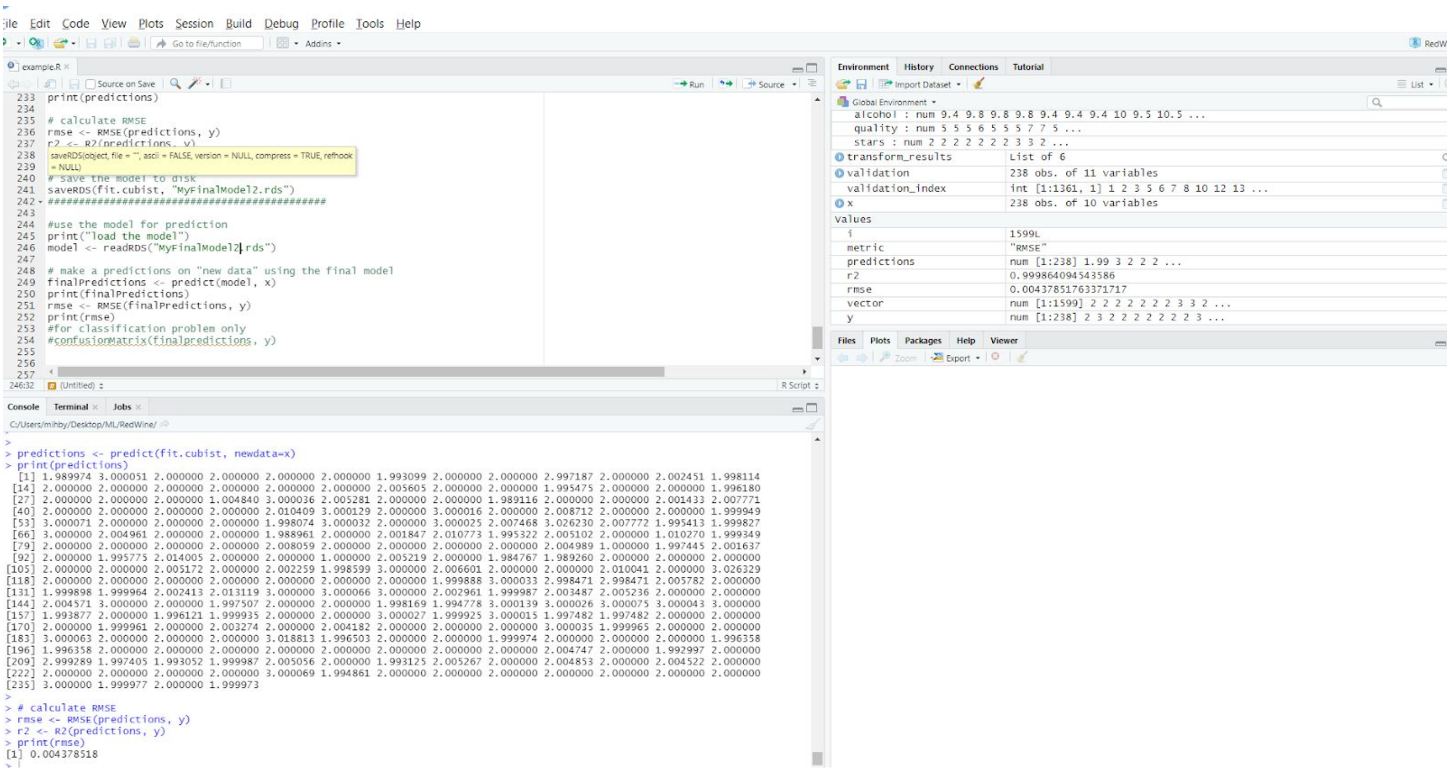

fixed.acidity
volatile.acidity
citric.acid
free.sulfur.dioxide
total.sulfur.dioxide
density
pH
sulphates
alcohol
quality
stars


fixed.acidity
volatile.acidity
citric.acid
free.sulfur.dioxide
total.sulfur.dioxide
density
pH
sulphates
alcohol
quality
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1


RedWine - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
example.R
corrplot(correlations, method="circle")
# Evaluate Algorithms
# Run algorithms using 10-fold cross validation
control <- trainControl(method="repeatedcv", number=10, repeats=3)
metric <- "RMSE"
# lm
set.seed(7)
fit.lm <- train(stars~., data=datasetTrain, method="lm", metric=metric, preProc=c("center", "scale"), trControl=control)
# GLM
set.seed(7)
fit.glm <- train(stars~., data=datasetTrain, method="glm", metric=metric, preProc=c("center", "scale"), trControl=control)
# GLMNET
set.seed(7)
fit.glmnet <- train(stars~., data=datasetTrain, method="glmnet", metric=metric, preProc=c("center", "scale"), trControl=con
# SVM
set.seed(7)
fit.svm <- train(stars~., data=datasetTrain, method="svmRadial", metric=metric, preProc=c("center", "scale"), trControl=con
# CART
set.seed(7)
fit.cart <- train(stars~., data=datasetTrain, method="rpart", metric=metric, preProc=c("center", "scale"), trControl=contro
# kNN
set.seed(7)
fit.knn <- train(stars~., data=datasetTrain, method="knn", metric=metric, preProc=c("center", "scale"), trControl=control)
# Compare algorithms
Console Terminal Jobs
C:/Users/mihby/Desktop/ML/RedWine/
MAE
Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
LM 0.21291910 0.22196258 0.22413786 0.22435863 0.22761482 0.23429523 0
GLM 0.21291910 0.22196258 0.22413786 0.22435863 0.22761482 0.23429523 0
GLMNET 0.21282809 0.22114892 0.22345240 0.22334189 0.22631103 0.23162422 0
SVM 0.06091681 0.08961527 0.09799761 0.09563448 0.10377077 0.11466709 0
CART 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0
KNN 0.02481752 0.05474887 0.06446078 0.06197422 0.06910755 0.09632353 0
RMSE
Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
LM 0.22754228 0.2347780 0.2373660 0.2380913 0.2418473 0.2504369 0
GLM 0.22754228 0.2347780 0.2373660 0.2380913 0.2418473 0.2504369 0
GLMNET 0.22796763 0.2335905 0.2375732 0.2372187 0.2406964 0.2473214 0
SVM 0.10974502 0.1884968 0.2032667 0.1959726 0.2164457 0.2434713 0
CART 0.00000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0
KNN 0.09201714 0.1773407 0.1937769 0.1901744 0.2078232 0.2767842 0
Rsquared
Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
LM 0.5733101 0.6457981 0.6652919 0.6673890 0.6946214 0.7235360 0
GLM 0.5733101 0.6457981 0.6652919 0.6673890 0.6946214 0.7235360 0
GLMNET 0.5805004 0.6475710 0.6693372 0.6701422 0.6942450 0.7227298 0
SVM 0.6631808 0.7531028 0.7754566 0.7831887 0.8002890 0.9241230 0
CART 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 0
KNN 0.5789519 0.7611453 0.8029461 0.7938939 0.8238951 0.9500757 0
Environment History Connections Tutorial
alcohol : num 9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
quality : int 5 5 5 6 5 5 5 7 7 5 ...
stars : num 2 2 2 2 2 2 2 3 3 2 ...
results List of 6
transform_results List of 6
validation 238 obs. of 11 variables
validation_index int [1:1361, 1] 1 2 3 5 6 7 8 10 12 13 ...
x 238 obs. of 9 variables
Values
cutoff 0.6
finalPredictions num [1:238] 5.52 5.69 5.26 4.95 5.52 ...
highlyCorrelated int [1:2] 1 4
i 11L
metric "RMSE"
Files Plots Packages Help Viewer
MAE
RMSE
Rsquared
LM
GLM
GLMNET
SVM
KNN
CART
Confidence Level: 0.95


RedWine - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
example.R*
# Evaluate Algorithms: with Feature Selection step
# remove correlated attributes
# find attributes that are highly corrected
set.seed(7)
cutoff <- 0.60
correlations <- cor(datasetTrain[,1:10])
highlyCorrelated <- findCorrelation(correlations, cutoff=cutoff)
for (value in highlyCorrelated) {
print(names(datasetTrain)[value])
}
# create a new dataset without highly corrected features
dataset_features <- datasetTrain[,-highlyCorrelated]
dim(dataset_features)
# Run algorithms using 10-fold cross validation
control <- trainControl(method="repeatedcv", number=10, repeats=3)
metric <- "RMSE"
# lm
set.seed(7)
fit.lm <- train(stars~., data=dataset_features, method="lm", metric=metric, preProc=c("center", "scale"), trControl=control
# GLM
set.seed(7)
fit.glm <- train(stars~., data=dataset_features, method="glm", metric=metric, preProc=c("center", "scale"), trControl=contr
# GLMNET
set.seed(7)
fit.glmnet <- train(stars~., data=dataset_features, method="glmnet", metric=metric, preProc=c("center", "scale"), trControl
# SVM
set.seed(7)
Console Terminal Jobs
C:/Users/mihby/Desktop/ML/RedWine/
+ print(names(datasetTrain)[value])
+ }
[1] "fixed.acidity"
[1] "free.sulfur.dioxide"
> # create a new dataset without highly corrected features
> dataset_features <- datasetTrain[,-highlyCorrelated]
> dim(dataset_features)
[1] 1361 9
> # Run algorithms using 10-fold cross validation
> control <- trainControl(method="repeatedcv", number=10, repeats=3)
> metric <- "RMSE"
> # lm
> set.seed(7)
> fit.lm <- train(stars~., data=dataset_features, method="lm", metric=metric, preProc=c("center", "scale"), trControl=control)
> # GLM
> set.seed(7)
> fit.glm <- train(stars~., data=dataset_features, method="glm", metric=metric, preProc=c("center", "scale"), trControl=control)
> # GLMNET
> set.seed(7)
> fit.glmnet <- train(stars~., data=dataset_features, method="glmnet", metric=metric, preProc=c("center", "scale"), trControl=control)
> # SVM
> set.seed(7)
Environment History Connections Tutorial
transform_results List of 6
validation 238 obs. of 11 variables
validation_index int [1:1361, 1] 1 2 3 5 6 7 8 10 12 13 ...
x 238 obs. of 9 variables
Values
cutoff 0.6
finalPredictions num [1:238] 5.52 5.69 5.26 4.95 5.52 ...
highlyCorrelated int [1:2] 1 4
i 11L
metric "RMSE"
predictions num [1:238] 5.52 5.69 5.26 4.95 5.52 ...
r2 0.362150701942835
rmse 0.615173987083772
value 4L
Files Plots Packages Help Viewer
MAE
RMSE
Rsquared
LM
GLM
GLMNET
SVM
KNN
CART
Confidence Level: 0.95


RedWine - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
example.R*
173 dotplot(feature_results)
176 # Evaluate Algorithnms: with Box-Cox Transform
178 # Run algorithms using 10-fold cross validation
179 control <- trainControl(method="repeatedcv", number=10, repeats=3)
180 metric <- "RMSE"
181 # lm
182 set.seed(7)
183 fit.lm <- train(stars~., data=datasetTrain, method="lm", metric=metric, preProc=c("center", "scale", "BoxCox"), trControl=c
184 # GLM
185 set.seed(7)
186 fit.glm <- train(stars~., data=datasetTrain, method="glm", metric=metric, preProc=c("center", "scale", "BoxCox"), trControl
187 # GLMNET
188 set.seed(7)
189 fit.glmnet <- train(stars~., data=datasetTrain, method="glmnet", metric=metric, preProc=c("center", "scale", "BoxCox"), trC
190 # SVM
191 set.seed(7)
192 fit.svm <- train(stars~., data=datasetTrain, method="svmRadial", metric=metric, preProc=c("center", "scale", "BoxCox"), trC
193 # CART
194 set.seed(7)
195 grid <- expand.grid(.cp=c(0, 0.05, 0.1))
196 fit.cart <- train(stars~., data=datasetTrain, method="rpart", metric=metric, tuneGrid=grid, preProc=c("center", "scale", "B
Console
C:/Users/mihby/Desktop/ML/RedWine/
MAE
Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
LM 0.21565982 0.22468365 0.22752181 0.22708357 0.23001203 0.2361595 0
GLM 0.21565982 0.22468365 0.22752181 0.22708357 0.23001203 0.2361595 0
GLMNET 0.21481682 0.22389994 0.22610200 0.22577608 0.22747470 0.2337173 0
SVM 0.06478582 0.08934349 0.09828748 0.09679188 0.10580882 0.1172431 0
CART 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.0000000 0
KNN 0.03479319 0.06170343 0.06898892 0.06834200 0.07889365 0.1036765 0
RMSE
Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
LM 0.2281378 0.2374816 0.2393370 0.2404931 0.2445399 0.2508854 0
GLM 0.2281378 0.2374816 0.2393370 0.2404931 0.2445399 0.2508854 0
GLMNET 0.2289158 0.2364217 0.2395155 0.2396609 0.2426565 0.2497364 0
SVM 0.1121690 0.1864510 0.2009460 0.1954990 0.2171442 0.2449586 0
CART 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0
KNN 0.1129491 0.1820812 0.2051456 0.1997826 0.2150341 0.2697887 0
Rsquared
Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
LM 0.5700337 0.6360984 0.6575655 0.6606782 0.6896291 0.7171097 0
GLM 0.5700337 0.6360984 0.6575655 0.6606782 0.6896291 0.7171097 0
GLMNET 0.5775375 0.6398670 0.6583736 0.6632318 0.6891896 0.7175318 0
SVM 0.6761117 0.7533630 0.7723087 0.7843076 0.8065373 0.9240225 0
CART 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 0
KNN 0.5939153 0.7295907 0.7738755 0.7743064 0.8149697 0.9287279 0
> dotplot(transform_results)
MAE
RMSE
Rsquared
LM
GLM
GLMNET
SVM
KNN
CART
Confidence Level: 0.95


File Edit Code View Plots Session Build Debug Profile Tools Help
example.R
233 print(predictions)
235 # calculate RMSE
236 rmse <- RMSE(predictions, y)
237 r2 <- R2(predictions, y)
saveRDS(object, file = "", ascii = FALSE, version = NULL, compress = TRUE, refhook = NULL)
240 # save the model to disk
241 saveRDS(fit.cubist, "MyFinalModel2.rds")
242 ##
244 #use the model for prediction
245 print("load the model")
246 model <- readRDS("MyFinalModel2.rds")
248 # make a predictions on "new data" using the final model
249 finalPredictions <- predict(model, x)
250 print(finalPredictions)
251 rmse <- RMSE(finalPredictions, y)
252 print(rmse)
253 #for classification problem only
254 #confusionMatrix(finalpredictions, y)
Console
C:/Users/mihby/Desktop/ML/RedWine/
> predictions <- predict(fit.cubist, newdata=x)
> print(predictions)
[1] 1.989974 3.000051 2.000000 2.000000 2.000000 2.000000 1.993099 2.000000 2.000000 2.997187 2.000000 2.002451 1.998114
[14] 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.005605 2.000000 2.000000 1.995475 2.000000 2.000000 1.996180
[27] 2.000000 2.000000 2.000000 1.004840 3.000036 2.005281 2.000000 2.000000 1.989116 2.000000 2.000000 2.001433 2.007771
[40] 2.000000 2.000000 2.000000 2.000000 2.010409 3.000129 2.000000 3.000016 2.000000 2.008712 2.000000 2.000000 1.999949
[53] 3.000071 2.000000 2.000000 2.000000 1.998074 3.000032 2.000000 3.000025 2.007468 3.026230 2.007772 1.995413 1.999827
[66] 3.000000 2.004961 2.000000 2.000000 1.988961 2.000000 2.001847 2.010773 1.995322 2.005102 2.000000 1.010270 1.999349
[79] 2.000000 2.000000 2.000000 2.000000 2.008059 2.000000 2.000000 2.000000 2.000000 2.004989 1.000000 1.997445 2.001637
[92] 2.000000 1.995775 2.014005 2.000000 2.000000 1.000000 2.005219 2.000000 1.984767 1.989260 2.000000 2.000000 2.000000
[105] 2.000000 2.000000 2.005172 2.000000 2.002259 1.998599 3.000000 2.006601 2.000000 2.000000 2.010041 2.000000 3.026329
[118] 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 1.999888 3.000033 2.998471 2.998471 2.005782 2.000000
[131] 1.999898 1.999964 2.002413 2.013119 3.000000 3.000066 3.000000 2.002961 1.999987 2.003487 2.005236 2.000000 2.000000
[144] 2.004571 3.000000 2.000000 1.997507 2.000000 2.000000 1.998169 1.994778 3.000139 3.000026 3.000075 3.000043 3.000000
[157] 1.993877 2.000000 1.996121 1.999935 2.000000 2.000000 3.000027 1.999925 3.000015 1.997482 1.997482 2.000000 2.000000
[170] 2.000000 1.999961 2.000000 2.003274 2.000000 2.004182 2.000000 2.000000 2.000000 3.000035 1.999965 2.000000 2.000000
[183] 3.000063 2.000000 2.000000 2.000000 3.018813 1.996503 2.000000 2.000000 1.999974 2.000000 2.000000 2.000000 1.996358
[196] 1.996358 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.004747 2.000000 1.992997 2.000000
[209] 2.999289 1.997405 1.993052 1.999987 2.005056 2.000000 1.993125 2.005267 2.000000 2.004853 2.000000 2.004522 2.000000
[222] 2.000000 2.000000 2.000000 2.000000 3.000069 1.994861 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000
[235] 3.000000 1.999977 2.000000 1.999973
> # calculate RMSE
> rmse <- RMSE(predictions, y)
> r2 <- R2(predictions, y)
> print(rmse)
[1] 0.004378518

So, when I'm running the algorithms to train the data, the mean RMSE is 0.2 which is a massive improvement from 0.6, and when I'm doing the predictions on the testing dataset the RMSE is 0.004 which is so low that makes our predictions too close to being perfect, which could mean that the dataset is overtrained, and I'll need new data to test if that is the actual RMSE.

But, overall massive improvement to the results due to the addition of "Stars" as an output variable.