WS.R

US16120

Wed Sep 26 22:56:11 2018

library(rcdk)

## Loading required package: rcdklibs

## Loading required package: rJava

library(tidyverse)

## -- Attaching packages -------------------------------------------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.0.0 v purrr 0.2.5  
## v tibble 1.4.2 v dplyr 0.7.6  
## v tidyr 0.8.1 v stringr 1.3.1  
## v readr 1.1.1 v forcats 0.3.0

## -- Conflicts ----------------------------------------------------------------------- tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()  
## x dplyr::matches() masks rcdk::matches()

library(magrittr)

##   
## Attaching package: 'magrittr'

## The following object is masked from 'package:purrr':  
##   
## set\_names

## The following object is masked from 'package:tidyr':  
##   
## extract

library(purrr)  
library(stringr)  
library(caret)

## Loading required package: lattice

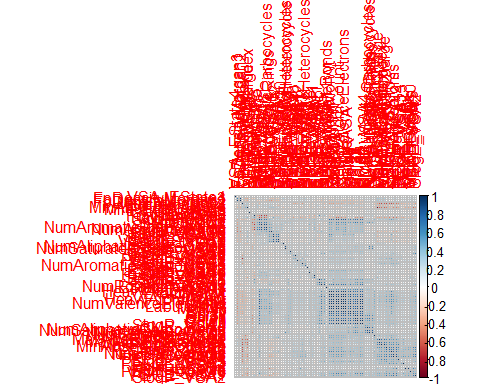
##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

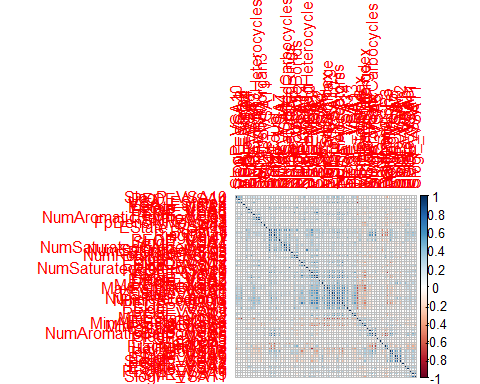
library(corrplot)

## corrplot 0.84 loaded

library(ggplot2)  
library(ggthemes)  
  
# read data  
  
## training data  
train <-  
 read.csv('cache/TR\_WS\_3158\_descrs.csv',  
 header = TRUE,  
 stringsAsFactors = FALSE) %>%  
 select(-X,-CAS,-ROMol,-SMILES,-ID) %>%  
 select(LogMolar, everything()) %>%  
 na.omit()  
  
X\_train <- train %>%  
 select(-LogMolar)  
y\_train <- train %>%  
 select(LogMolar) %>%  
 data.frame()  
  
## test data  
test <-  
 read.csv('cache/TST\_WS\_1066\_descrs.csv',  
 header = TRUE,  
 stringsAsFactors = FALSE) %>%  
 select(-X,-CAS,-ROMol,-SMILES,-ID) %>%  
 select(LogMolar, everything()) %>%  
 na.omit()  
  
X\_test <- test %>%  
 select(-LogMolar)  
y\_test <- test %>%  
 select(LogMolar) %>%  
 data.frame()  
  
# curate data  
  
## near-zero variance descriptors  
  
nzv <- nearZeroVar(X\_train, freqCut = 100/0)  
X\_train <- X\_train[ , -nzv]  
### and  
X\_test <- X\_test[ , -nzv]  
  
## highly correlated descriptors  
  
correlations <- cor(X\_train)  
corrplot::corrplot(correlations, order = 'hclust')



highCorr <- findCorrelation(correlations, cutoff = 0.85)  
X\_train <- X\_train[ , -highCorr]  
### and  
X\_test <- X\_test[ , -highCorr]  
  
correlations <- cor(X\_train)  
corrplot::corrplot(correlations, order = 'hclust')



## linear combinations  
  
comboInfo <- findLinearCombos(X\_train) # returns NULL  
# X\_train <- X\_train[ , -comboInfo$remove]  
# ### and  
# X\_test <- X\_test[ , -nzv]  
  
## center & scale descriptors  
  
preProcValues <- preProcess(X\_train, method = c("center", "scale"))  
  
X\_trainTransformed <- predict(preProcValues, X\_train)  
### and  
X\_testTransformed <- predict(preProcValues, X\_test)  
  
### PCA  
  
# pca <- preProcess(X\_trainTransformed, method = c('pca'))  
# X\_train\_pca <- predict(pca, X\_trainTransformed)  
# X\_test\_pca <- predict(pca, X\_testTransformed)  
#   
# train\_pca <- X\_train\_pca %>%  
# select(PC1, PC2) %>%  
# mutate(dataset = 'train')  
# test\_pca <- X\_test\_pca %>%  
# select(PC1, PC2) %>%  
# mutate(dataset = 'test')  
# pcaPts <- rbind(train\_pca, test\_pca)  
#   
# p <-  
# ggplot(pcaPts, aes(PC1, PC2)) +  
# geom\_point(aes(colour = factor(dataset), shape = factor(dataset))) +  
# ggthemes::theme\_tufte()  
# p  
  
# models  
  
fitControl <- trainControl(## 10-fold CV  
 method = "repeatedcv",  
 repeats = 5)  
  
set.seed(350)  
  
## multiple linear regression  
  
trainSet <- cbind(y\_train, X\_trainTransformed)  
  
mlr <- train(LogMolar ~ .,  
 data = trainSet,  
 method = 'lm',  
 trControl = fitControl)

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient  
## fit may be misleading

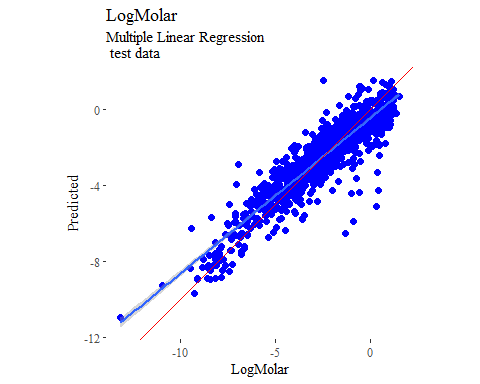
y\_predict <- predict(mlr, newdata = X\_testTransformed) %>%  
 data.frame()

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient  
## fit may be misleading

colnames(y\_predict) <- c('Predicted')  
  
data2plot <- cbind(y\_test, y\_predict)  
  
summary(lm(Predicted ~ LogMolar, data = data2plot))

##   
## Call:  
## lm(formula = Predicted ~ LogMolar, data = data2plot)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -5.0561 -0.4723 0.0592 0.5096 4.0314   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -0.43119 0.03897 -11.06 <2e-16 \*\*\*  
## LogMolar 0.81969 0.01131 72.44 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.8528 on 1061 degrees of freedom  
## Multiple R-squared: 0.8318, Adjusted R-squared: 0.8317   
## F-statistic: 5248 on 1 and 1061 DF, p-value: < 2.2e-16

p <-  
 ggplot(data2plot, aes(LogMolar, Predicted)) +  
 geom\_point(colour = "blue", size = 2) +  
 coord\_equal() +  
 # xlim(c(0, 3.5)) + ylim(c(0, 3.5)) +  
 geom\_smooth(method = 'lm') +  
 labs(title = 'LogMolar',  
 subtitle = 'Multiple Linear Regression\n test data') +  
 ggthemes::theme\_tufte()  
p <- p + geom\_abline(intercept = 0,  
 slope = 1,  
 colour = 'red')  
p



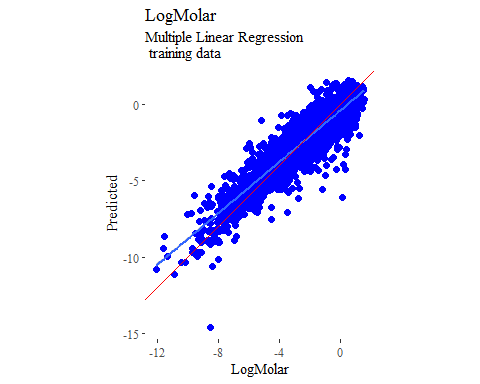
y\_predict <- predict(mlr, newdata = X\_trainTransformed) %>%  
 data.frame()

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient  
## fit may be misleading

colnames(y\_predict) <- c('Predicted')  
  
mlrPR <- postResample(pred = y\_predict, obs = X\_trainTransformed)  
rmse\_train = c(mlrPR[1])  
r2\_train = c(mlrPR[2])  
  
data2plot <- cbind(y\_train, y\_predict)  
  
summary(lm(Predicted ~ LogMolar, data = data2plot))

##   
## Call:  
## lm(formula = Predicted ~ LogMolar, data = data2plot)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -7.0398 -0.4585 0.0427 0.5139 3.7495   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -0.406677 0.022365 -18.18 <2e-16 \*\*\*  
## LogMolar 0.842372 0.006496 129.68 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.8311 on 3147 degrees of freedom  
## Multiple R-squared: 0.8424, Adjusted R-squared: 0.8423   
## F-statistic: 1.682e+04 on 1 and 3147 DF, p-value: < 2.2e-16

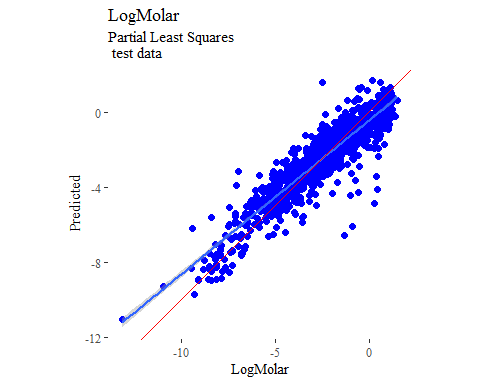
p <-  
 ggplot(data2plot, aes(LogMolar, Predicted)) +  
 geom\_point(colour = "blue", size = 2) +  
 coord\_equal() +  
 # xlim(c(0, 3.5)) + ylim(c(0, 3.5)) +  
 geom\_smooth(method='lm') +  
 labs(title = 'LogMolar',  
 subtitle = 'Multiple Linear Regression\n training data') +  
 ggthemes::theme\_tufte()  
p <- p + geom\_abline(intercept = 0,  
 slope = 1,  
 colour = 'red')  
p



## partial least squares  
  
plsModel <- train(  
 LogMolar ~ .,  
 data = trainSet,  
 method = 'pls',  
 tuneLength = 20,  
 trControl = fitControl  
)  
  
y\_predict <- predict(plsModel, newdata = X\_testTransformed) %>%  
 data.frame()  
colnames(y\_predict) <- c('Predicted')  
  
data2plot <- cbind(y\_test, y\_predict)  
  
summary(lm(Predicted ~ LogMolar, data = data2plot))

##   
## Call:  
## lm(formula = Predicted ~ LogMolar, data = data2plot)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -5.0825 -0.4828 0.0713 0.5309 4.0324   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -0.43306 0.03895 -11.12 <2e-16 \*\*\*  
## LogMolar 0.81885 0.01131 72.40 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.8524 on 1061 degrees of freedom  
## Multiple R-squared: 0.8317, Adjusted R-squared: 0.8315   
## F-statistic: 5242 on 1 and 1061 DF, p-value: < 2.2e-16

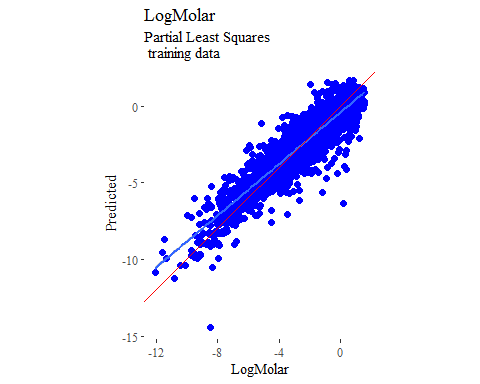
p <-  
 ggplot(data2plot, aes(LogMolar, Predicted)) +  
 geom\_point(colour = "blue", size = 2) +  
 coord\_equal() +  
 # xlim(c(0, 3.5)) + ylim(c(0, 3.5)) +  
 geom\_smooth(method = 'lm') +  
 labs(title = 'LogMolar',  
 subtitle = 'Partial Least Squares\n test data') +  
 ggthemes::theme\_tufte()  
p <- p + geom\_abline(intercept = 0,  
 slope = 1,  
 colour = 'red')  
p



y\_predict <- predict(plsModel, newdata = X\_trainTransformed) %>%  
 data.frame()  
colnames(y\_predict) <- c('Predicted')  
  
data2plot <- cbind(y\_train, y\_predict)  
  
summary(lm(Predicted ~ LogMolar, data = data2plot))

##   
## Call:  
## lm(formula = Predicted ~ LogMolar, data = data2plot)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -6.8431 -0.4604 0.0387 0.5181 3.6405   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -0.409280 0.022423 -18.25 <2e-16 \*\*\*  
## LogMolar 0.841363 0.006512 129.19 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.8332 on 3147 degrees of freedom  
## Multiple R-squared: 0.8414, Adjusted R-squared: 0.8413   
## F-statistic: 1.669e+04 on 1 and 3147 DF, p-value: < 2.2e-16

p <-  
 ggplot(data2plot, aes(LogMolar, Predicted)) +  
 geom\_point(colour = "blue", size = 2) +  
 coord\_equal() +  
 # xlim(c(0, 3.5)) + ylim(c(0, 3.5)) +  
 geom\_smooth(method='lm') +  
 labs(title = 'LogMolar',  
 subtitle = 'Partial Least Squares\n training data') +  
 ggthemes::theme\_tufte()  
p <- p + geom\_abline(intercept = 0,  
 slope = 1,  
 colour = 'red')  
p



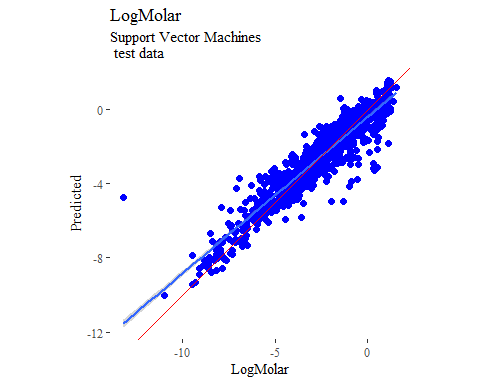
## support vector machines  
  
svmModel <- train(  
 LogMolar ~ .,  
 data = trainSet,  
 method = 'svmRadial',  
 # tuneLength = 14,  
 trControl = fitControl  
)

## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.

y\_predict <- predict(svmModel, newdata = X\_testTransformed) %>%  
 data.frame()  
colnames(y\_predict) <- c('Predicted')  
  
data2plot <- cbind(y\_test, y\_predict)  
  
summary(lm(Predicted ~ LogMolar, data = data2plot))

##   
## Call:  
## lm(formula = Predicted ~ LogMolar, data = data2plot)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3.4838 -0.3490 0.0473 0.4177 6.7868   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -0.389895 0.033448 -11.66 <2e-16 \*\*\*  
## LogMolar 0.844956 0.009711 87.01 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.732 on 1061 degrees of freedom  
## Multiple R-squared: 0.8771, Adjusted R-squared: 0.877   
## F-statistic: 7570 on 1 and 1061 DF, p-value: < 2.2e-16

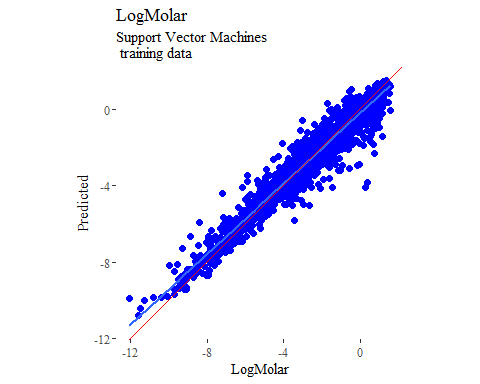
p <-  
 ggplot(data2plot, aes(LogMolar, Predicted)) +  
 geom\_point(colour = "blue", size = 2) +  
 coord\_equal() +  
 # xlim(c(0, 3.5)) + ylim(c(0, 3.5)) +  
 geom\_smooth(method = 'lm') +  
 labs(title = 'LogMolar',  
 subtitle = 'Support Vector Machines\n test data') +  
 ggthemes::theme\_tufte()  
p <- p + geom\_abline(intercept = 0,  
 slope = 1,  
 colour = 'red')  
p



y\_predict <- predict(svmModel, newdata = X\_trainTransformed) %>%  
 data.frame()  
colnames(y\_predict) <- c('Predicted')  
  
data2plot <- cbind(y\_train, y\_predict)  
  
summary(lm(Predicted ~ LogMolar, data = data2plot))

##   
## Call:  
## lm(formula = Predicted ~ LogMolar, data = data2plot)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -4.1428 -0.2348 0.0098 0.2498 2.4469   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -0.21033 0.01322 -15.91 <2e-16 \*\*\*  
## LogMolar 0.92202 0.00384 240.08 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.4914 on 3147 degrees of freedom  
## Multiple R-squared: 0.9482, Adjusted R-squared: 0.9482   
## F-statistic: 5.764e+04 on 1 and 3147 DF, p-value: < 2.2e-16

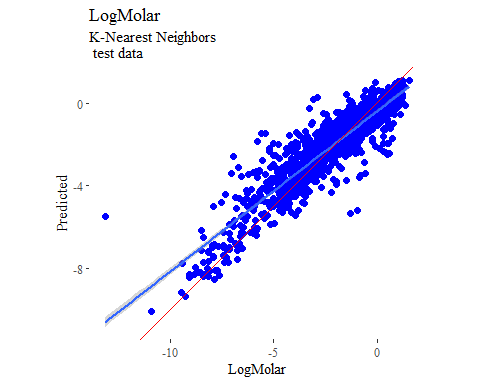
p <-  
 ggplot(data2plot, aes(LogMolar, Predicted)) +  
 geom\_point(colour = "blue", size = 2) +  
 coord\_equal() +  
 # xlim(c(0, 3.5)) + ylim(c(0, 3.5)) +  
 geom\_smooth(method='lm') +  
 labs(title = 'LogMolar',  
 subtitle = 'Support Vector Machines\n training data') +  
 ggthemes::theme\_tufte()  
p <- p + geom\_abline(intercept = 0,  
 slope = 1,  
 colour = 'red')  
p



## k-nearest neighbors  
  
knnModel <- train(  
 LogMolar ~ .,  
 data = trainSet,  
 method = 'knn',  
 tuneGrid = data.frame(.k = 1:20),  
 trControl = fitControl  
)  
  
y\_predict <- predict(knnModel, newdata = X\_testTransformed) %>%  
 data.frame()  
colnames(y\_predict) <- c('Predicted')  
  
data2plot <- cbind(y\_test, y\_predict)  
  
summary(lm(Predicted ~ LogMolar, data = data2plot))

##   
## Call:  
## lm(formula = Predicted ~ LogMolar, data = data2plot)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -4.0818 -0.4947 -0.0020 0.4792 5.1274   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -0.38818 0.03952 -9.823 <2e-16 \*\*\*  
## LogMolar 0.77541 0.01147 67.584 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.8647 on 1061 degrees of freedom  
## Multiple R-squared: 0.8115, Adjusted R-squared: 0.8113   
## F-statistic: 4568 on 1 and 1061 DF, p-value: < 2.2e-16

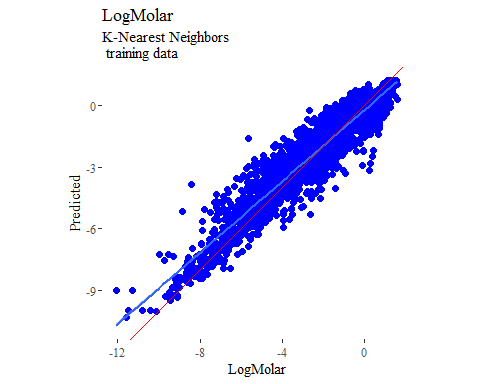
p <-  
 ggplot(data2plot, aes(LogMolar, Predicted)) +  
 geom\_point(colour = "blue", size = 2) +  
 coord\_equal() +  
 # xlim(c(0, 3.5)) + ylim(c(0, 3.5)) +  
 geom\_smooth(method = 'lm') +  
 labs(title = 'LogMolar',  
 subtitle = 'K-Nearest Neighbors\n test data') +  
 ggthemes::theme\_tufte()  
p <- p + geom\_abline(intercept = 0,  
 slope = 1,  
 colour = 'red')  
p



y\_predict <- predict(knnModel, newdata = X\_trainTransformed) %>%  
 data.frame()  
colnames(y\_predict) <- c('Predicted')  
  
data2plot <- cbind(y\_train, y\_predict)  
  
summary(lm(Predicted ~ LogMolar, data = data2plot))

##   
## Call:  
## lm(formula = Predicted ~ LogMolar, data = data2plot)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3.1436 -0.4067 -0.0041 0.3721 3.6388   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -0.211116 0.018139 -11.64 <2e-16 \*\*\*  
## LogMolar 0.867217 0.005268 164.61 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.6741 on 3147 degrees of freedom  
## Multiple R-squared: 0.8959, Adjusted R-squared: 0.8959   
## F-statistic: 2.71e+04 on 1 and 3147 DF, p-value: < 2.2e-16

p <-  
 ggplot(data2plot, aes(LogMolar, Predicted)) +  
 geom\_point(colour = "blue", size = 2) +  
 coord\_equal() +  
 # xlim(c(0, 3.5)) + ylim(c(0, 3.5)) +  
 geom\_smooth(method='lm') +  
 labs(title = 'LogMolar',  
 subtitle = 'K-Nearest Neighbors\n training data') +  
 ggthemes::theme\_tufte()  
p <- p + geom\_abline(intercept = 0,  
 slope = 1,  
 colour = 'red')  
p



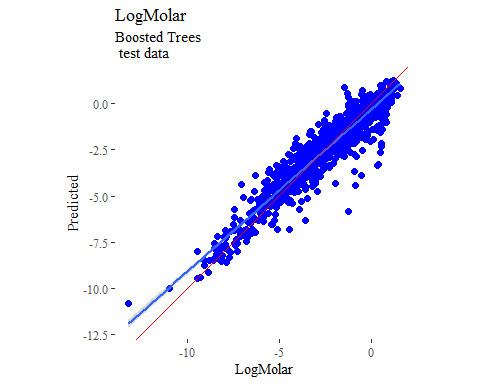
## boosted trees  
  
gbmGrid <- expand.grid(  
 .interaction.depth = seq(1, 7, by = 2),  
 .n.trees = seq(100, 1000, by = 50),  
 .shrinkage = c(0.001, 0.1),  
 .n.minobsinnode = 3  
)  
  
treeModel <- train(  
 LogMolar ~ .,  
 data = trainSet,  
 method = 'gbm',  
 tuneGrid = gbmGrid,  
 verbose = FALSE  
)

## Warning in (function (x, y, offset = NULL, misc = NULL, distribution =  
## "bernoulli", : variable 69: VSA\_EState4 has no variation.

y\_predict <- predict(treeModel, newdata = X\_testTransformed) %>%  
 data.frame()  
colnames(y\_predict) <- c('Predicted')  
  
data2plot <- cbind(y\_test, y\_predict)  
  
summary(lm(Predicted ~ LogMolar, data = data2plot))

##   
## Call:  
## lm(formula = Predicted ~ LogMolar, data = data2plot)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -4.4093 -0.3305 0.0229 0.3986 2.4858   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -0.278081 0.032391 -8.585 <2e-16 \*\*\*  
## LogMolar 0.882731 0.009405 93.862 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.7088 on 1061 degrees of freedom  
## Multiple R-squared: 0.8925, Adjusted R-squared: 0.8924   
## F-statistic: 8810 on 1 and 1061 DF, p-value: < 2.2e-16

p <-  
 ggplot(data2plot, aes(LogMolar, Predicted)) +  
 geom\_point(colour = "blue", size = 2) +  
 coord\_equal() +  
 # xlim(c(0, 3.5)) + ylim(c(0, 3.5)) +  
 geom\_smooth(method = 'lm') +  
 labs(title = 'LogMolar',  
 subtitle = 'Boosted Trees\n test data') +  
 ggthemes::theme\_tufte()  
p <- p + geom\_abline(intercept = 0,  
 slope = 1,  
 colour = 'red')  
p



y\_predict <- predict(treeModel, newdata = X\_trainTransformed) %>%  
 data.frame()  
colnames(y\_predict) <- c('Predicted')  
  
data2plot <- cbind(y\_train, y\_predict)  
  
summary(lm(Predicted ~ LogMolar, data = data2plot))

##   
## Call:  
## lm(formula = Predicted ~ LogMolar, data = data2plot)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1.31712 -0.16446 -0.00177 0.15826 1.61489   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -0.063972 0.006643 -9.63 <2e-16 \*\*\*  
## LogMolar 0.974835 0.001929 505.25 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.2469 on 3147 degrees of freedom  
## Multiple R-squared: 0.9878, Adjusted R-squared: 0.9878   
## F-statistic: 2.553e+05 on 1 and 3147 DF, p-value: < 2.2e-16

p <-  
 ggplot(data2plot, aes(LogMolar, Predicted)) +  
 geom\_point(colour = "blue", size = 2) +  
 coord\_equal() +  
 # xlim(c(0, 3.5)) + ylim(c(0, 3.5)) +  
 geom\_smooth(method='lm') +  
 labs(title = 'LogMolar',  
 subtitle = 'Boosted Trees\n training data') +  
 ggthemes::theme\_tufte()  
p <- p + geom\_abline(intercept = 0,  
 slope = 1,  
 colour = 'red')  
p

