Project Two: Understanding Predictive Factors for ABC Beverage

Salma Elshahawy, John K. Hancock, and Farhana Zahir

5/23/2021

Contents

OVERVIEW

The data science team of Salma Elshahawy, John K. Hancock, and Farhana Zahir have prepared the following technical report to address the issue of understanding ABC's manufacturing process and its predictive factors. This report is the predictive value of the PH.

The report consists of the following:

PART 1: THE DATASETS

PART 2: DATA PREPARATION

PART 3: EXPERIMENTATION

PART 4: EVALUATE MODELS

PART 5: USE THE BEST MODEL TO FORECAST PH

PART 6: CONCLUSIONS

PART 1: THE DATASETS

In this section, we did the following:

```
Import the Datasets
Evaluate the Dataset
Devise a Data Preparation Strategy
```

Import the Data

The excel files, StudentData.xlsx and StudentEvaluation.xlsx, are hosted on the team's Github page. Here, they're downloaded and read into the dataframes, beverage_training_data and beverage_test_data.

```
temp_train_file <- tempfile(pattern="StudentData", fileext = ".xlsx")
temp_eval_file <- tempfile(pattern="StudentEvaluation", fileext = ".xlsx")

student_train <-  "https://github.com/JohnKHancock/CUNY_DATA624_Project2/blob/main/raw/StudentData.xlsx
```

```
student_eval <-   "https://github.com/JohnKHancock/CUNY_DATA624_Project2/blob/main/raw/StudentEvaluatio



student_data <- GET(student_train,
                authenticate(Sys.getenv("GITHUB_PAT"), ""),
                write_disk(path = temp_train_file))

student_eval <-  GET(student_eval,
                authenticate(Sys.getenv("GITHUB_PAT"), ""),
                write_disk(path = temp_eval_file))
```

Evaluate the Dataset

After importing the Beverage Training dataset, we see that there are 2,571 observations consisiting of 32 predictor variables and one dependent variable, PH. We also see that "Brand Code" is a factor variable that will need to be handled as well as several observations with a number of NAs.

For the Beverage Testing dataset, we see 267 observations, the 32 predictors, and the dependent variable PH which is all NAs. This is the data that we will have to predict. Same as the training set, We also see that "Brand Code" is a character variable that will need to be handled as well as several observations with a number of NAs.

Beverage Training Data

```
dim(beverage_training_data)
```

```
## [1] 2571   33
```

```
typeof(beverage_training_data$`Brand Code`)
```

```
## [1] "character"
```

Beverage Testing Data

Devise a Data Preparation Strategy

After analyzing the data, we devised the following processes in order to prepare the data for analysis

A. Isolate predictors from the dependent variable

B. Correct the Predictor Names

C. Create a data frame of numeric values only

D. Identify and Impute Missing Data

E. Identify and Address Outliers

F. Check for and remove correlated predictors

G. Identify Near Zero Variance Predictors

H. Impute missing values and Create dummy variables for Brand.Code

I. Impute missing data for Dependent Variable PH

PART 2: DATA PREPARATION

A. Isolate predictors from the dependent variables

For the training set, remove the predictor variable, PH and store it into the variable, y_train.

```
predictors <- subset(beverage_training_data, select = -c(PH))
predictors_evaluate <- subset(beverage_test_data, select = -c(PH))
y_train <- as.data.frame(beverage_training_data$PH)
colnames(y_train) <- c("PH")
```

B. Correct the Predictor Names

Correct the space in the predictor names using the make.names function. The space in the names may be problematic. This was applied to both datasets.

```
colnames(predictors)
```

```
##  [1] "Brand Code"        "Carb Volume"       "Fill Ounces"
##  [4] "PC Volume"         "Carb Pressure"     "Carb Temp"
##  [7] "PSC"               "PSC Fill"          "PSC CO2"
## [10] "Mnf Flow"          "Carb Pressure1"    "Fill Pressure"
## [13] "Hyd Pressure1"     "Hyd Pressure2"     "Hyd Pressure3"
## [16] "Hyd Pressure4"     "Filler Level"      "Filler Speed"
## [19] "Temperature"       "Usage cont"        "Carb Flow"
## [22] "Density"           "MFR"               "Balling"
## [25] "Pressure Vacuum"   "Oxygen Filler"     "Bowl Setpoint"
## [28] "Pressure Setpoint" "Air Pressurer"     "Alch Rel"
## [31] "Carb Rel"          "Balling Lvl"
```

```
colnames(predictors)<- make.names(colnames(predictors))
colnames(predictors)
```

```
##  [1] "Brand.Code"        "Carb.Volume"       "Fill.Ounces"
##  [4] "PC.Volume"         "Carb.Pressure"     "Carb.Temp"
##  [7] "PSC"               "PSC.Fill"          "PSC.CO2"
## [10] "Mnf.Flow"          "Carb.Pressure1"    "Fill.Pressure"
## [13] "Hyd.Pressure1"     "Hyd.Pressure2"     "Hyd.Pressure3"
## [16] "Hyd.Pressure4"     "Filler.Level"      "Filler.Speed"
## [19] "Temperature"       "Usage.cont"        "Carb.Flow"
## [22] "Density"           "MFR"               "Balling"
## [25] "Pressure.Vacuum"   "Oxygen.Filler"     "Bowl.Setpoint"
## [28] "Pressure.Setpoint" "Air.Pressurer"     "Alch.Rel"
## [31] "Carb.Rel"          "Balling.Lvl"
```

```
colnames(predictors_evaluate)<- make.names(colnames(predictors_evaluate))
colnames(predictors_evaluate)
```

```
##  [1] "Brand.Code"        "Carb.Volume"       "Fill.Ounces"
##  [4] "PC.Volume"         "Carb.Pressure"     "Carb.Temp"
##  [7] "PSC"               "PSC.Fill"          "PSC.CO2"
## [10] "Mnf.Flow"          "Carb.Pressure1"    "Fill.Pressure"
## [13] "Hyd.Pressure1"     "Hyd.Pressure2"     "Hyd.Pressure3"
## [16] "Hyd.Pressure4"     "Filler.Level"      "Filler.Speed"
## [19] "Temperature"       "Usage.cont"        "Carb.Flow"
## [22] "Density"           "MFR"               "Balling"
## [25] "Pressure.Vacuum"   "Oxygen.Filler"     "Bowl.Setpoint"
## [28] "Pressure.Setpoint" "Air.Pressurer"     "Alch.Rel"
## [31] "Carb.Rel"          "Balling.Lvl"
```

C. Create a data frame of numeric values only

We saw earlier that Brand.Code is a categorical value. Because of that we subset the dataframe to remove it. We will handle this variable later.

```
num_predictors <- subset (predictors, select = -Brand.Code)
num_predictors <- as.data.frame(num_predictors)
```

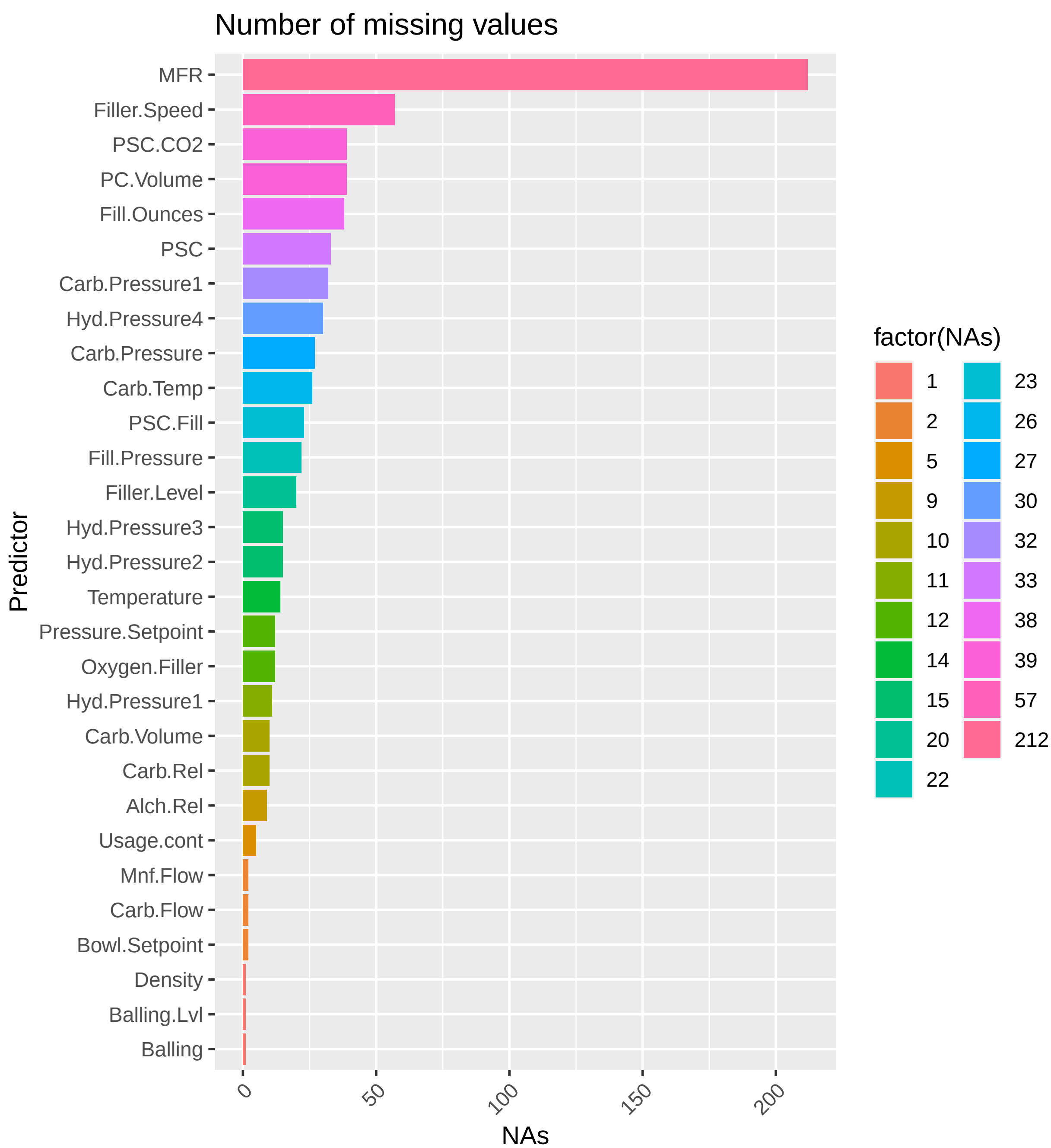
D. Identify and Impute Missing Data

The predictor MFR has the most missing values at 212. I used knn imputation to handle missing values. After the knn imputation, there are still missing values for Brand.Code which will be handled in a later section.

Training Data

Predictors	NAs
MFR	212
Filler.Speed	57
PC.Volume	39
PSC.CO2	39
Fill.Ounces	38
PSC	33

```
missingData  %>%
  ggplot() +
    geom_bar(aes(x=reorder(Predictors,NAs), y=NAs, fill=factor(NAs)), stat = 'identity', ) +
    labs(x='Predictor', y="NAs", title='Number of missing values') +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) + coord_flip()
```



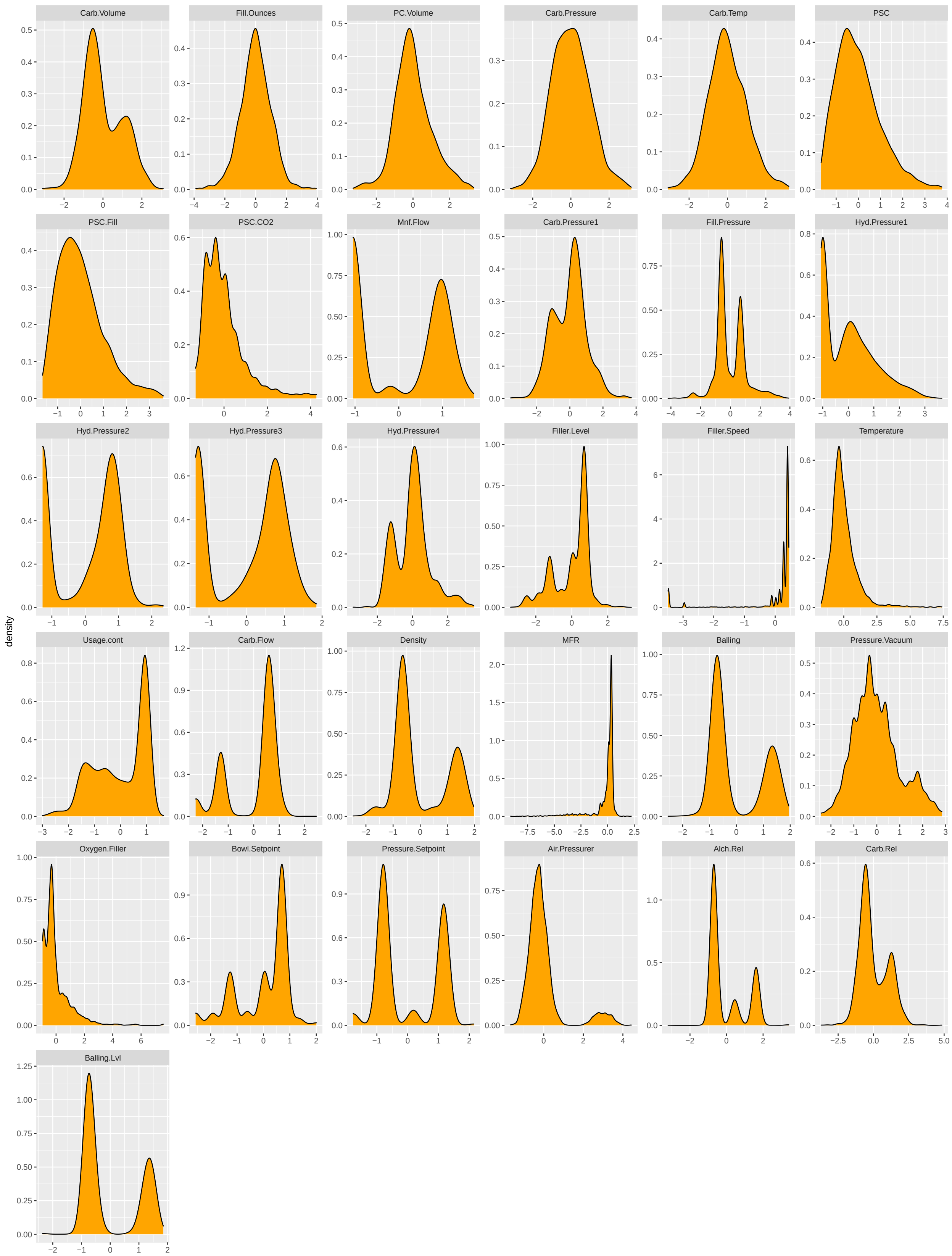

```r
missingData <- as.data.frame(colSums(is.na(predictors_imputed)))
colnames(missingData) <- c("NAs")
missingData <- cbind(Predictors = rownames(missingData), missingData)
rownames(missingData) <- 1:nrow(missingData)
missingData <- missingData[missingData$NAs != 0,] %>%
                arrange(desc(NAs))
head(missingData)
```

```
## [1] Predictors NAs
## <0 rows> (or 0-length row.names)
```

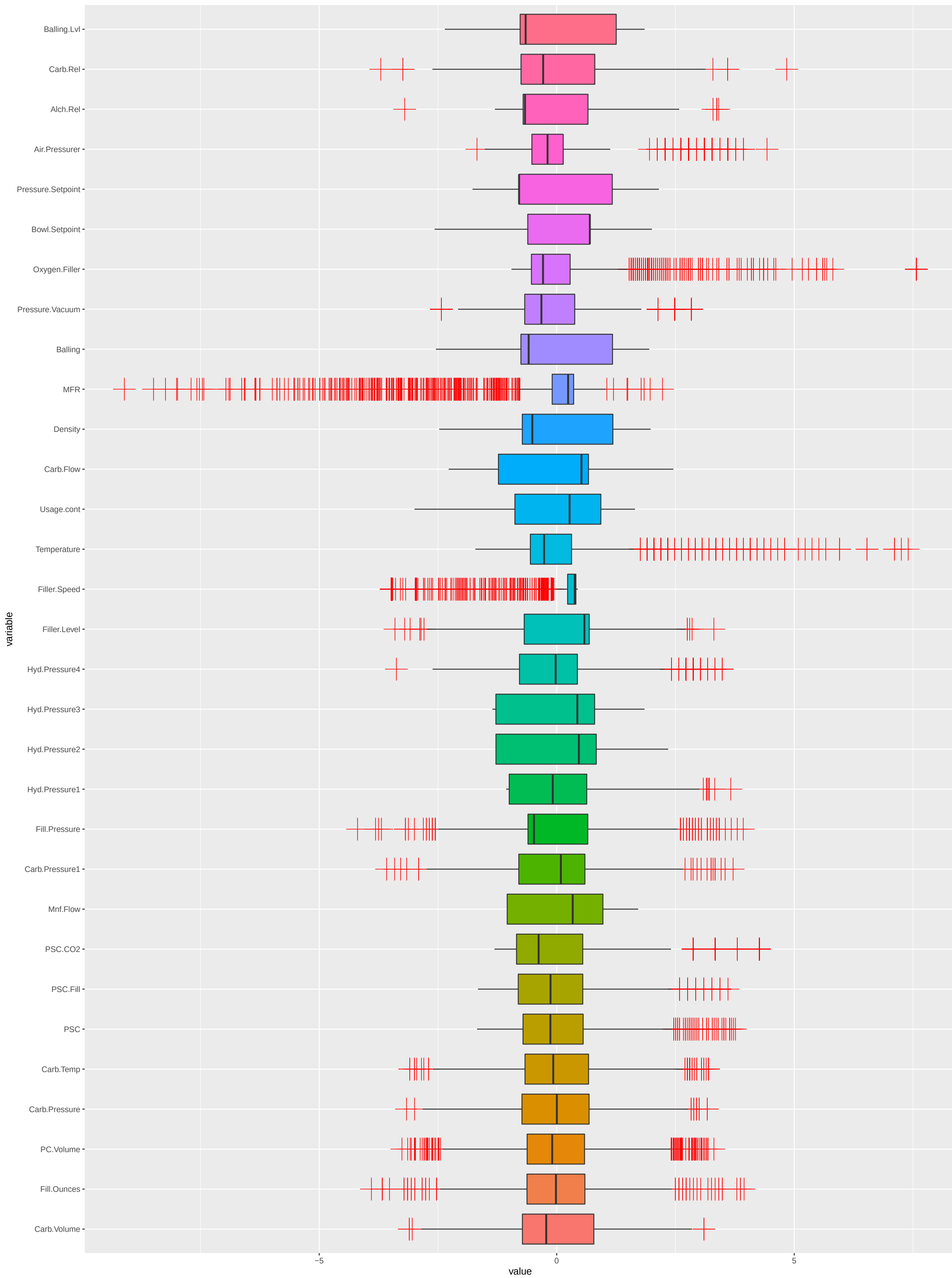
E. Identify Skewness and Outliers

Next we looked at the distributions of the numeric variables. There are only four predictors that are normally distributed. The box plots show a high number of outliers in the data. To correct for this, the pre processing step of center and scale was used. We centered and scaled these distributions.

```
par(mfrow = c(3, 3))
datasub = melt(predictors_imputed)
suppressWarnings(ggplot(datasub, aes(x= value)) +
                   geom_density(fill='orange') + facet_wrap(~variable, scales = 'free') )
```


Carb.Volume
Fill.Ounces
PC.Volume
Carb.Pressure
Carb.Temp
PSC
PSC.Fill
PSC.CO2
Mnf.Flow
Carb.Pressure1
Fill.Pressure
Hyd.Pressure1
Hyd.Pressure2
Hyd.Pressure3
Hyd.Pressure4
Filler.Level
Filler.Speed
Temperature
Usage.cont
Carb.Flow
Density
MFR
Balling
Pressure.Vacuum
Oxygen.Filler
Bowl.Setpoint
Pressure.Setpoint
Air.Pressurer
Alch.Rel
Carb.Rel
Balling.Lvl
density
value

```r
ggplot(data = datasub , aes(x=variable, y=value)) +
  geom_boxplot(outlier.colour="red", outlier.shape=3, outlier.size=8,aes(fill=variable)) +
  coord_flip() + theme(legend.position = "none")
```



Balling.Lvl
Carb.Rel
Alch.Rel
Air.Pressurer
Pressure.Setpoint
Bowl.Setpoint
Oxygen.Filler
Pressure.Vacuum
Balling
MFR
Density
Carb.Flow
Usage.cont
Temperature
Filler.Speed
Filler.Level
Hyd.Pressure4
Hyd.Pressure3
Hyd.Pressure2
Hyd.Pressure1
Fill.Pressure
Carb.Pressure1
Mnf.Flow
PSC.CO2
PSC.Fill
PSC
Carb.Temp
Carb.Pressure
PC.Volume
Fill.Ounces
Carb.Volume
variable
−5
0
5
value

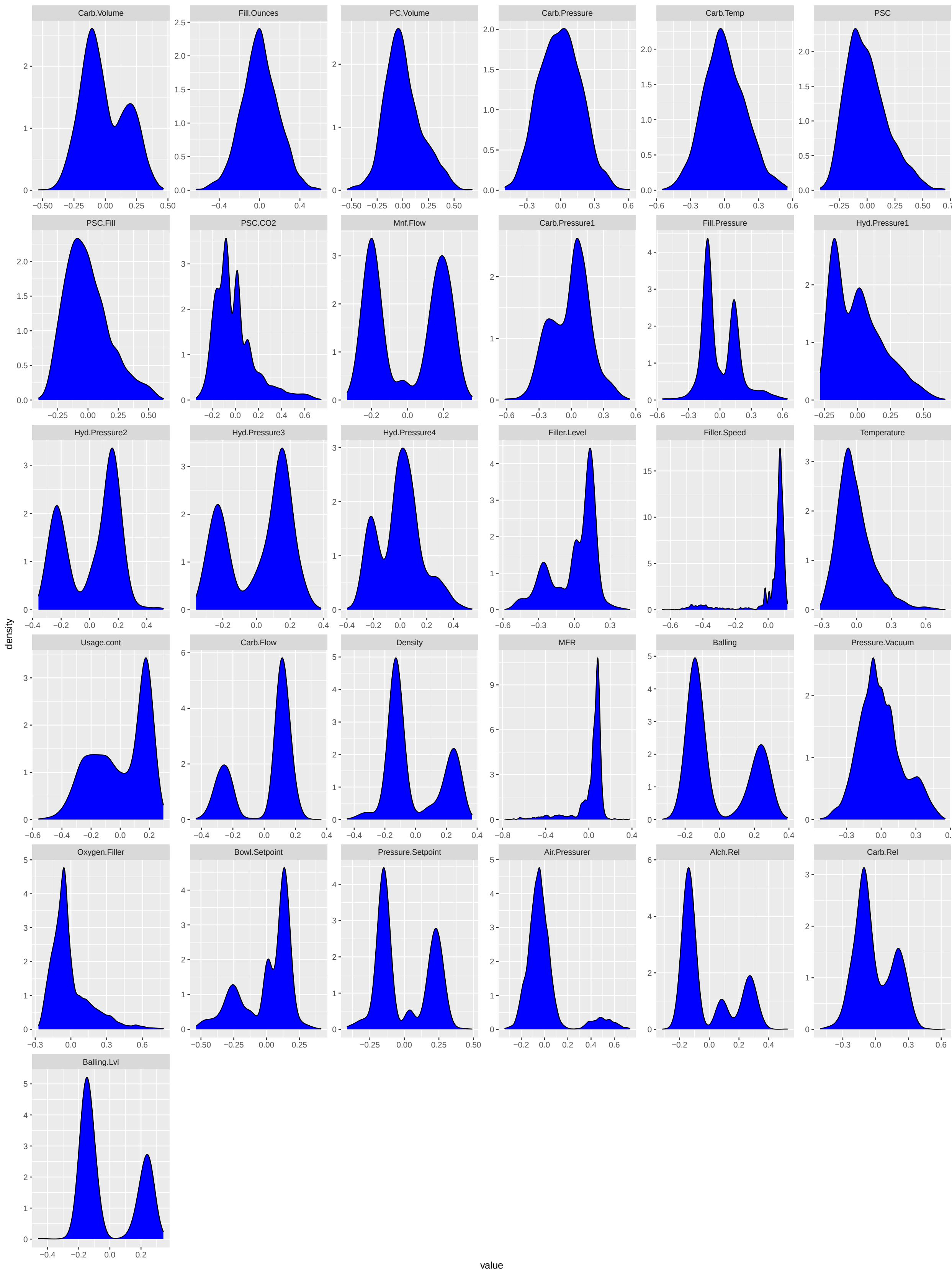
```r
preprocessing <- preProcess(as.data.frame(predictors_imputed), method = c("center", "scale"))
preprocessing
```

```
## Created from 2571 samples and 31 variables
##
## Pre-processing:
##   - centered (31)
##   - ignored (0)
##   - scaled (31)
```

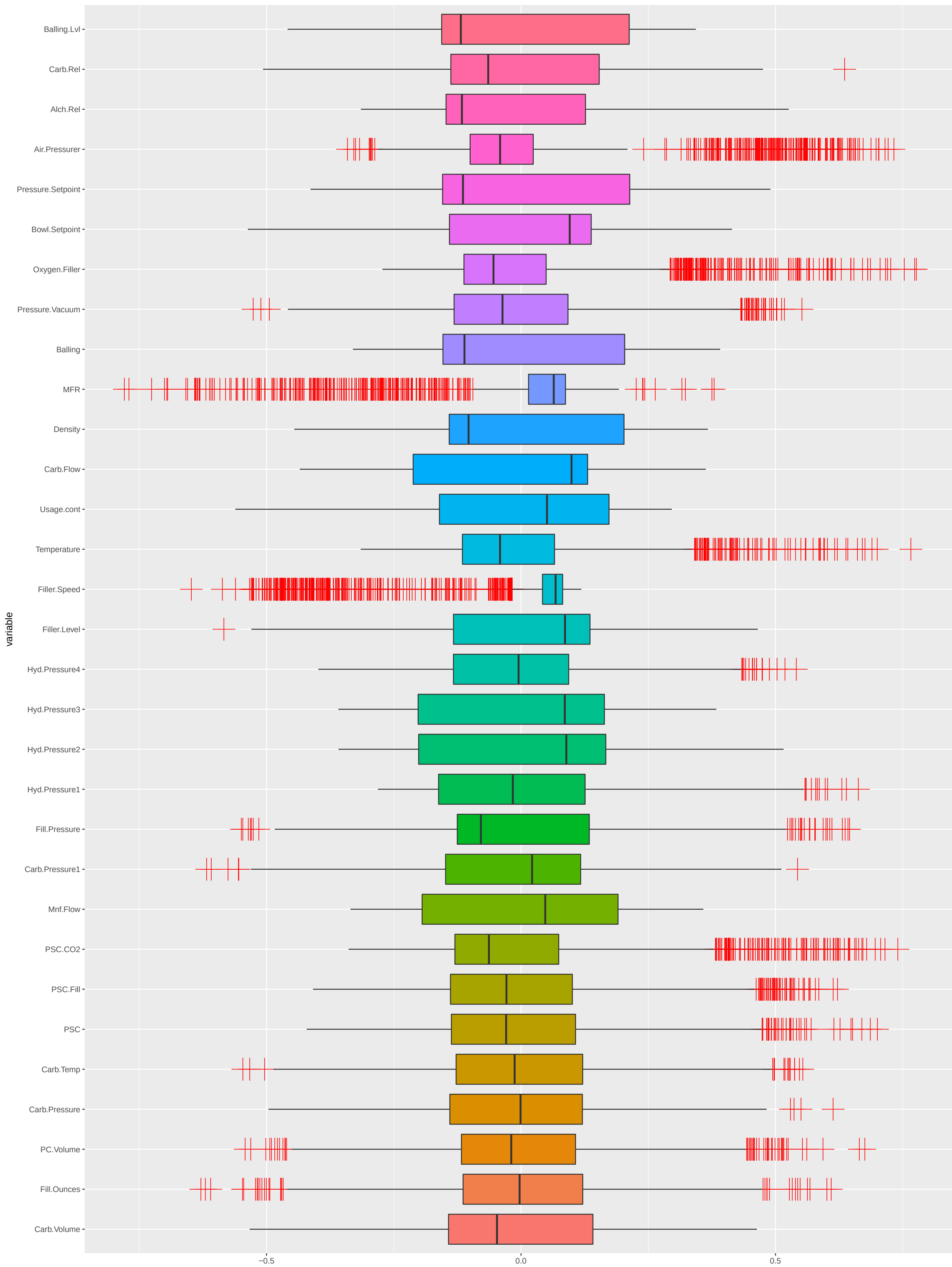
```r
num_predictors_01 <- predict(preprocessing, predictors_imputed)
```

```r
num_predictors_02 <- spatialSign(num_predictors_01)
num_predictors_02 <- as.data.frame(num_predictors_02)
```

```r
par(mfrow = c(3, 3))
datasub = melt(num_predictors_02)
suppressWarnings(ggplot(datasub, aes(x= value)) +
                   geom_density(fill='blue') + facet_wrap(~variable, scales = 'free') )
```


Carb.Volume
Fill.Ounces
PC.Volume
Carb.Pressure
Carb.Temp
PSC
PSC.Fill
PSC.CO2
Mnf.Flow
Carb.Pressure1
Fill.Pressure
Hyd.Pressure1
Hyd.Pressure2
Hyd.Pressure3
Hyd.Pressure4
Filler.Level
Filler.Speed
Temperature
Usage.cont
Carb.Flow
Density
MFR
Balling
Pressure.Vacuum
Oxygen.Filler
Bowl.Setpoint
Pressure.Setpoint
Air.Pressurer
Alch.Rel
Carb.Rel
Balling.Lvl
density
value

```
ggplot(data = datasub , aes(x=variable, y=value)) +
  geom_boxplot(outlier.colour="red", outlier.shape=3, outlier.size=8,aes(fill=variable)) +
  coord_flip() + theme(legend.position = "none")
```


Balling.Lvl
Carb.Rel
Alch.Rel
Air.Pressurer
Pressure.Setpoint
Bowl.Setpoint
Oxygen.Filler
Pressure.Vacuum
Balling
MFR
Density
Carb.Flow
Usage.cont
Temperature
Filler.Speed
Filler.Level
Hyd.Pressure4
Hyd.Pressure3
Hyd.Pressure2
Hyd.Pressure1
Fill.Pressure
Carb.Pressure1
Mnf.Flow
PSC.CO2
PSC.Fill
PSC
Carb.Temp
Carb.Pressure
PC.Volume
Fill.Ounces
Carb.Volume
variable
−0.5
0.0
0.5
value

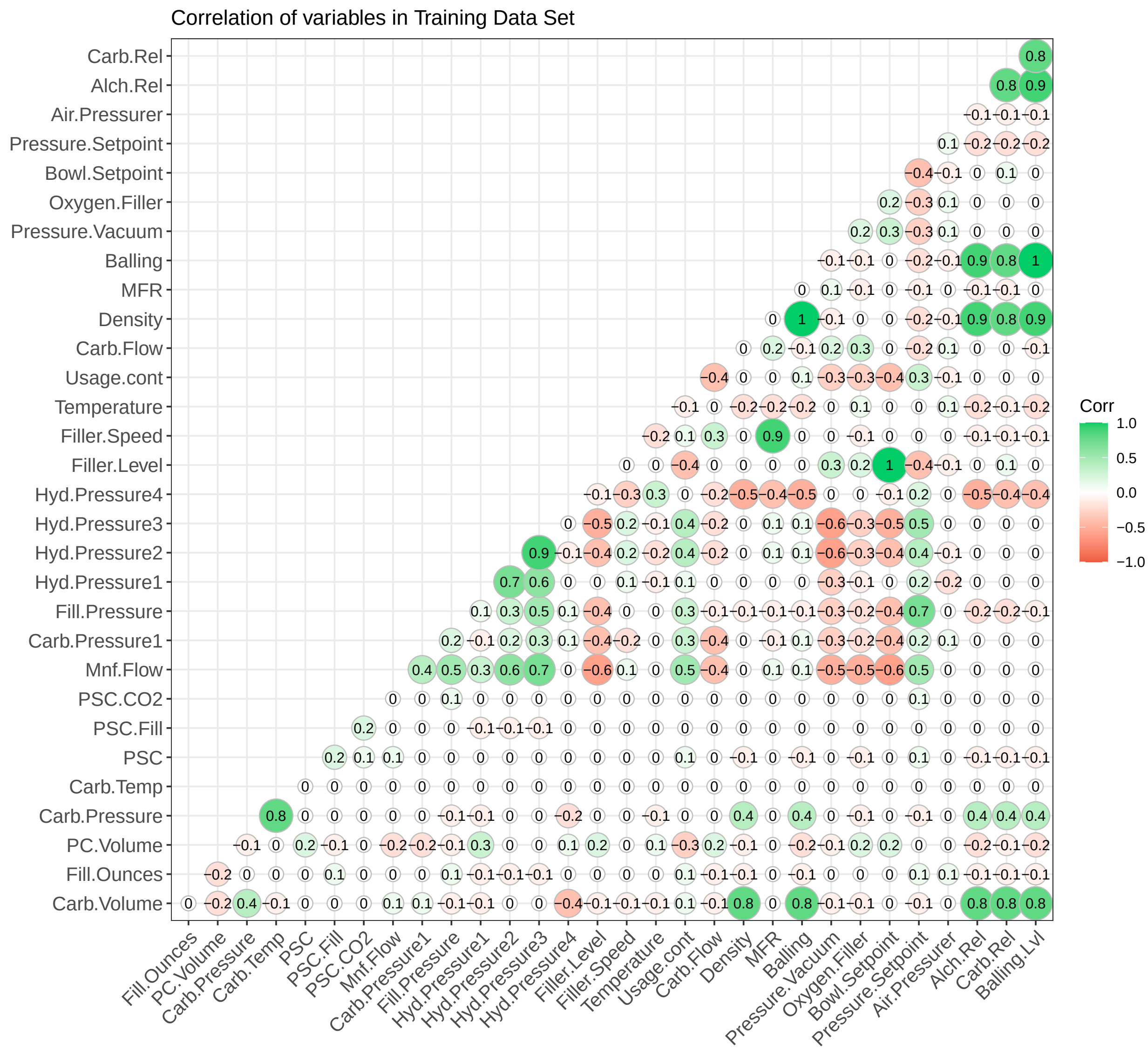
F. Check for and remove correlated predictors

We identified five variables that are highly correlated with other variables at above .9. Highly correlated variables lead to Multicollinearity which reduces the precision of the estimate coefficients and weakens the statistical power of regression models.

```
tooHigh <- findCorrelation(cor(num_predictors_02, use="na.or.complete"), cutoff = .9, names = TRUE)
tooHigh
```

```
## [1] "Balling"       "Hyd.Pressure3" "Balling.Lvl"   "Alch.Rel"
## [5] "Bowl.Setpoint"
```

```
corr <- round(cor(num_predictors_02, use="na.or.complete"), 1)

ggcorrplot(corr,
           type="lower",
           lab=TRUE,
           lab_size=3,
           method="circle",
           colors=c("tomato2", "white", "springgreen3"),
           title="Correlation of variables in Training Data Set",
           ggtheme=theme_bw)
```




```
num_predictors_02[ ,c(tooHigh)] <- list(NULL)
colnames(num_predictors_02)
```

```
##  [1] "Carb.Volume"       "Fill.Ounces"       "PC.Volume"
##  [4] "Carb.Pressure"     "Carb.Temp"         "PSC"
##  [7] "PSC.Fill"          "PSC.CO2"           "Mnf.Flow"
## [10] "Carb.Pressure1"    "Fill.Pressure"     "Hyd.Pressure1"
## [13] "Hyd.Pressure2"     "Hyd.Pressure4"     "Filler.Level"
## [16] "Filler.Speed"      "Temperature"       "Usage.cont"
## [19] "Carb.Flow"         "Density"           "MFR"
## [22] "Pressure.Vacuum"   "Oxygen.Filler"     "Pressure.Setpoint"
## [25] "Air.Pressurer"     "Carb.Rel"
```

G. Identify Near Zero Variance Predictors

Remove the zero variance predictor. There are no near zero variance predictors

```
caret::nearZeroVar(num_predictors_02, names = TRUE)
```

```
## character(0)
```

H. Impute missing values and Create dummy variables for Brand.Code

Earlier, we saw that there are 120 missing values for Brand.Code, a factor variable. The imputation strategy here is to impute with the most frequent value, "B". After imputation, Brand.Code was converted to dummy variables. The converted Brand.Code predictor is joined to the num_predictors_02.

```
BrandCodeNAs <- predictors$Brand.Code[is.na(predictors$Brand.Code ==TRUE)]
length(BrandCodeNAs)
```

```
## [1] 120
```

```
predictors$Brand.Code <- as.factor(predictors$Brand.Code)
levels(predictors$Brand.Code )
```

```
## [1] "A" "B" "C" "D"
```

```
table(predictors$Brand.Code)
```

```
##
##    A    B    C    D
##  293 1239  304  615
```

```
predictors$Brand.Code[is.na(predictors$Brand.Code)] = "B"
```

```
predictors$Brand.Code[is.na(predictors$Brand.Code)]
```

```
## factor(0)
## Levels: A B C D
```

```
mod<- dummyVars(~Brand.Code,
          data=predictors,
          levelsOnly = FALSE)
mod
```

```
## Dummy Variable Object
##
## Formula: ~Brand.Code
## 1 variables, 1 factors
## Variables and levels will be separated by '.'
## A less than full rank encoding is used
```

Brand.Code.A	Brand.Code.B	Brand.Code.C	Brand.Code.D
0	1	0	0
1	0	0	0
0	1	0	0
1	0	0	0
1	0	0	0
1	0	0	0

```
eval.data <- cbind(dummies, num_predictors_02)
```

I. Impute missing data for Dependent Variable PH

The final step is to impute missing values for the dependent variable, PH, with the median for PH.

```
y_train[is.na(y_train$PH),] <- median(y_train$PH,na.rm=TRUE)
```

```
processed.train <- cbind(y_train, eval.data)
```

```
missingData <- as.data.frame(colSums(is.na(processed.train)))
colnames(missingData) <- c("NAs")
missingData <- cbind(Predictors = rownames(missingData), missingData)
rownames(missingData) <- 1:nrow(missingData)
missingData <- missingData[missingData$NAs != 0,] %>%
                arrange(desc(NAs))
head(missingData)
```

```
## [1] Predictors NAs
## <0 rows> (or 0-length row.names)
```

PART 3: EXPERIMENTATION

Split the Time Series

Before we begin with the experimentation, We split the training data into train and test sets

```
evaluation.split <- initial_split(processed.train, prop = 0.7, strata = "PH")
train <- training(evaluation.split)
test <- testing(evaluation.split)
```

Modeling

We examined 12 models. We looked at Linear Models, Non Linear Regression Models, and Tree Based Models. For all of the models, MNF.Flow was the most important predictor with the exception of the bag tree model. Other consistently important predictors include predictor, Brand C and D. Residuals for each model appear random with no discernable patterns. In Part 4, we evaluated the metrics from each model.

Linear Models

```
set.seed(100)
x_train <- train[, 2:29]
y_train <- as.data.frame(train$PH)
colnames(y_train) <- c("PH")

x_test <- test[, 2:29]
y_test <- as.data.frame(test$PH)
colnames(y_test) <- c("PH")
ctrl <- trainControl(method = "cv", number = 10)
```

Basic linear model

```
lmFit1 <- train(x_train, y_train$PH,
                method = "lm",
                trControl = ctrl)
```

```r
summary(lmFit1)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.52446 -0.07795  0.01077  0.08818  0.80884
##
## Coefficients: (1 not defined because of singularities)
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        8.625784   0.014677 587.725  < 2e-16 ***
## Brand.Code.A      -0.093499   0.014458  -6.467 1.29e-10 ***
## Brand.Code.B      -0.089526   0.021626  -4.140 3.64e-05 ***
## Brand.Code.C      -0.211304   0.023570  -8.965  < 2e-16 ***
## Brand.Code.D             NA         NA      NA       NA
## Carb.Volume       -0.023764   0.051673  -0.460  0.64566
## Fill.Ounces       -0.057694   0.018727  -3.081  0.00210 **
## PC.Volume         -0.035070   0.022935  -1.529  0.12641
## Carb.Pressure     -0.002142   0.076030  -0.028  0.97753
## Carb.Temp          0.016619   0.069297   0.240  0.81049
## PSC               -0.015499   0.018835  -0.823  0.41067
## PSC.Fill          -0.017197   0.018394  -0.935  0.34993
## PSC.CO2           -0.034417   0.018308  -1.880  0.06029 .
## Mnf.Flow          -0.385179   0.033263 -11.580  < 2e-16 ***
## Carb.Pressure1     0.164129   0.021344   7.690 2.43e-14 ***
## Fill.Pressure      0.058071   0.029784   1.950  0.05137 .
## Hyd.Pressure1      0.026422   0.029476   0.896  0.37017
## Hyd.Pressure2      0.069207   0.037740   1.834  0.06685 .
## Hyd.Pressure4      0.019572   0.028098   0.697  0.48617
## Filler.Level       0.167006   0.027056   6.173 8.31e-10 ***
## Filler.Speed       0.032854   0.048096   0.683  0.49463
## Temperature       -0.123123   0.022888  -5.379 8.46e-08 ***
## Usage.cont        -0.123677   0.022099  -5.597 2.53e-08 ***
## Carb.Flow          0.049147   0.025001   1.966  0.04948 *
## Density           -0.141041   0.048587  -2.903  0.00374 **
## MFR               -0.018238   0.045936  -0.397  0.69138
## Pressure.Vacuum   -0.025752   0.023775  -1.083  0.27889
## Oxygen.Filler     -0.047076   0.024027  -1.959  0.05024 .
## Pressure.Setpoint -0.042546   0.027955  -1.522  0.12821
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1374 on 1773 degrees of freedom
## Multiple R-squared:  0.3903, Adjusted R-squared:  0.3811
## F-statistic: 42.04 on 27 and 1773 DF,  p-value: < 2.2e-16
```

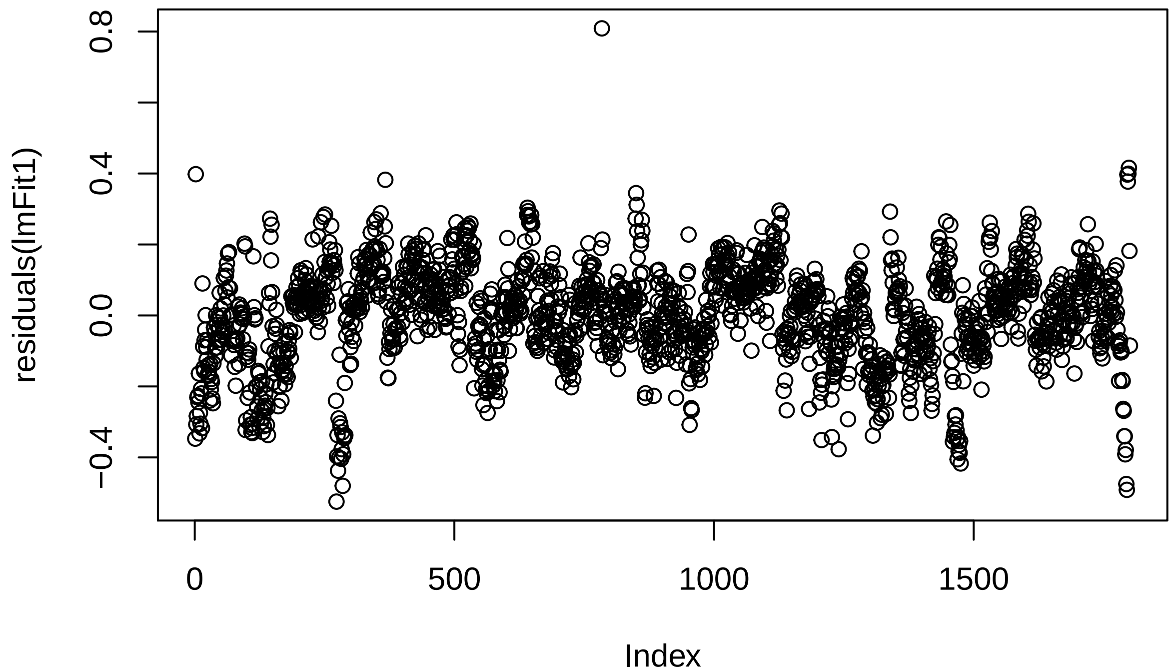
```r
lmFit1$results
```

```
##   intercept      RMSE Rsquared       MAE      RMSESD RsquaredSD     MAESD
## 1      TRUE 0.1389641 0.367775 0.1078644 0.005401945 0.03499629 0.0029008
```

```r
varImp(lmFit1)
```

```
## lm variable importance
##
##   only 20 most important variables shown (out of 27)
##
##                   Overall
## Mnf.Flow          100.000
## Brand.Code.C       77.364
## Carb.Pressure1     66.325
## Brand.Code.A       55.741
## Filler.Level       53.191
## Usage.cont         48.204
## Temperature        46.326
## Brand.Code.B       35.594
## Fill.Ounces        26.426
## Density            24.886
## Carb.Flow          16.773
## Oxygen.Filler      16.717
## Fill.Pressure      16.635
## PSC.CO2            16.030
## Hyd.Pressure2      15.631
## PC.Volume          12.994
## Pressure.Setpoint  12.931
## Pressure.Vacuum     9.133
## PSC.Fill            7.850
## Hyd.Pressure1       7.516
```

```
plot(residuals(lmFit1) )
```



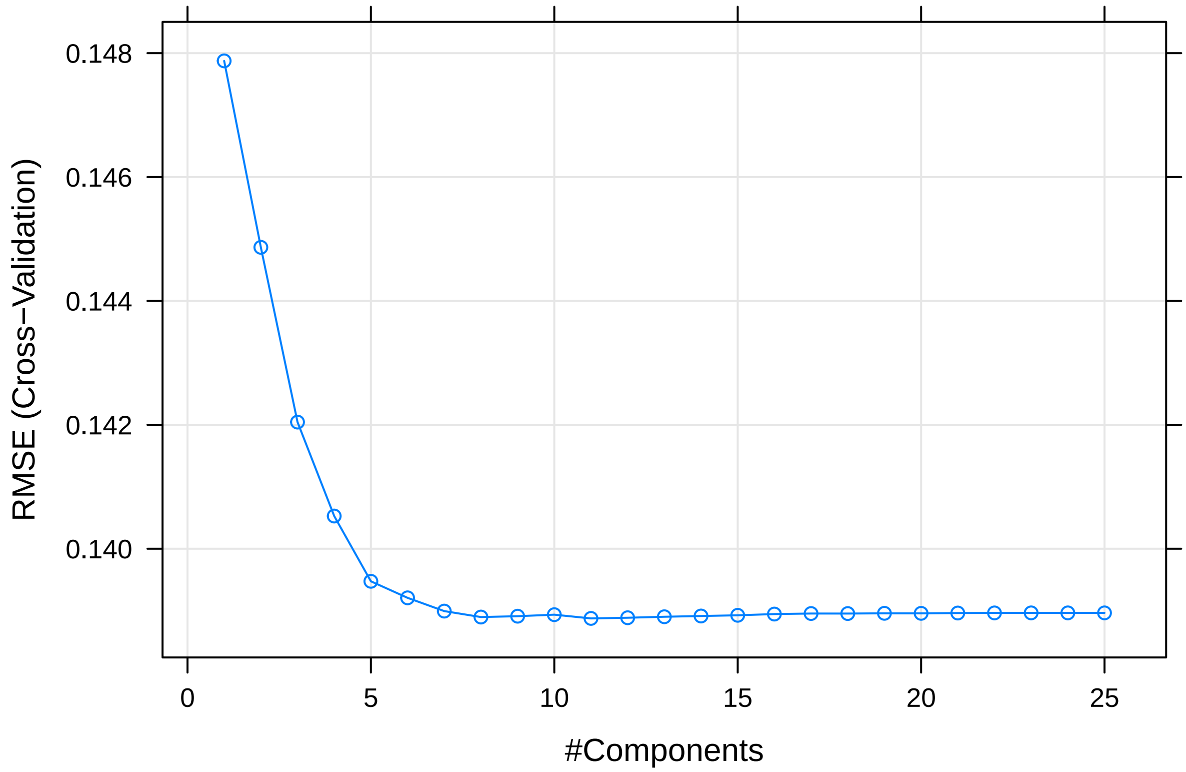
Partial Least Squares or PLS

```r
set.seed(100)
plsFit1 <- train(x_train, y_train$PH,
  method = "pls",
  tuneLength = 25,
  trControl = ctrl)
```

```r
summary(plsFit1)
```

```
## Data:    X dimension: 1801 28
##  Y dimension: 1801 1
## Fit method: oscorespls
## Number of components considered: 11
## TRAINING: % variance explained
##           1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X           12.60    31.39     46.1    54.68    59.16    65.46    69.68
## .outcome    29.05    32.01     35.4    37.43    38.32    38.57    38.79
##           8 comps  9 comps  10 comps  11 comps
## X           72.18    74.28     76.18     78.91
## .outcome    38.94    38.99     39.02     39.03
```

```r
plot(plsFit1)
```




```r
plsFit1$bestTune
```

```
##    ncomp
## 11    11
```

```r
train_set_results <- plsFit1$results %>%
  filter(ncomp==8)

train_set_results
```

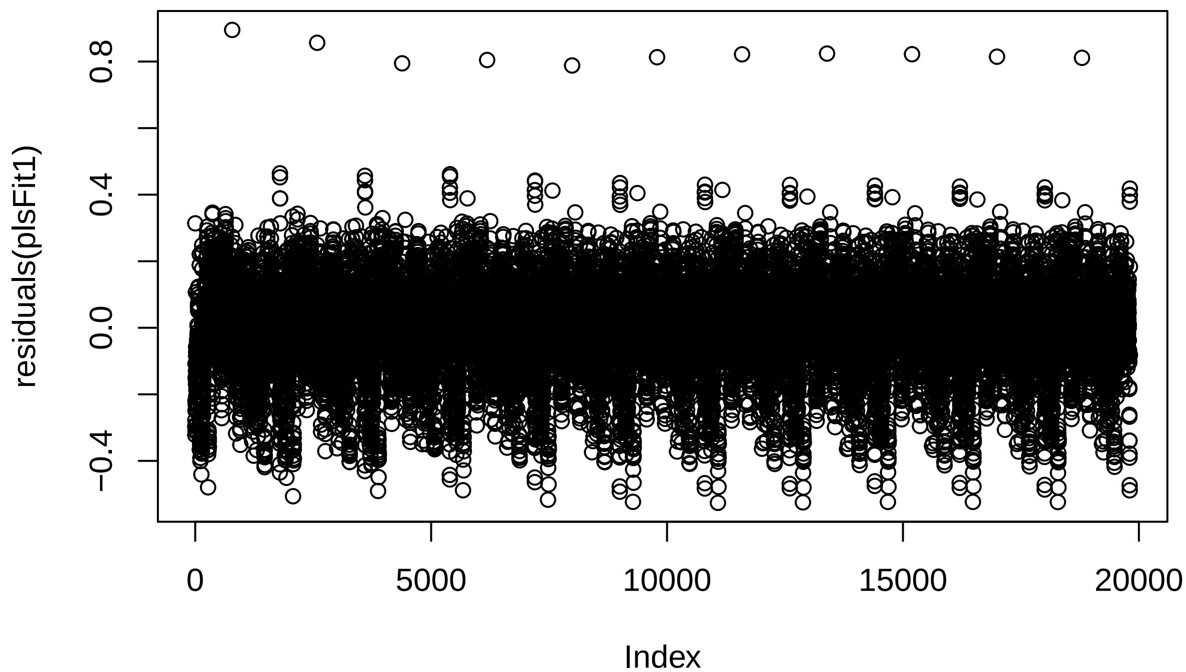
```
##   ncomp      RMSE  Rsquared       MAE      RMSESD RsquaredSD       MAESD
## 1     8 0.1388971 0.3681363 0.1080925 0.005459839 0.03566505 0.002818444
```

```r
varImp(plsFit1)
```

```
## pls variable importance
##
##   only 20 most important variables shown (out of 28)
##
##                   Overall
## Mnf.Flow           100.00
## Brand.Code.C        89.97
## Brand.Code.D        75.97
## Filler.Level        70.96
## Usage.cont          66.51
## Pressure.Setpoint   55.65
## Brand.Code.B        46.84
## Fill.Pressure       45.43
```

```
## Hyd.Pressure2       42.37
## Pressure.Vacuum     41.01
## Temperature         34.75
## Carb.Flow           32.73
## Oxygen.Filler       32.71
## Brand.Code.A        29.47
## Carb.Pressure1      25.61
## Hyd.Pressure4       23.67
## Fill.Ounces         21.70
## PSC                 20.20
## Hyd.Pressure1       17.79
## PSC.CO2             15.00
```

```r
plot(residuals(plsFit1) )
```



Ridge Regression

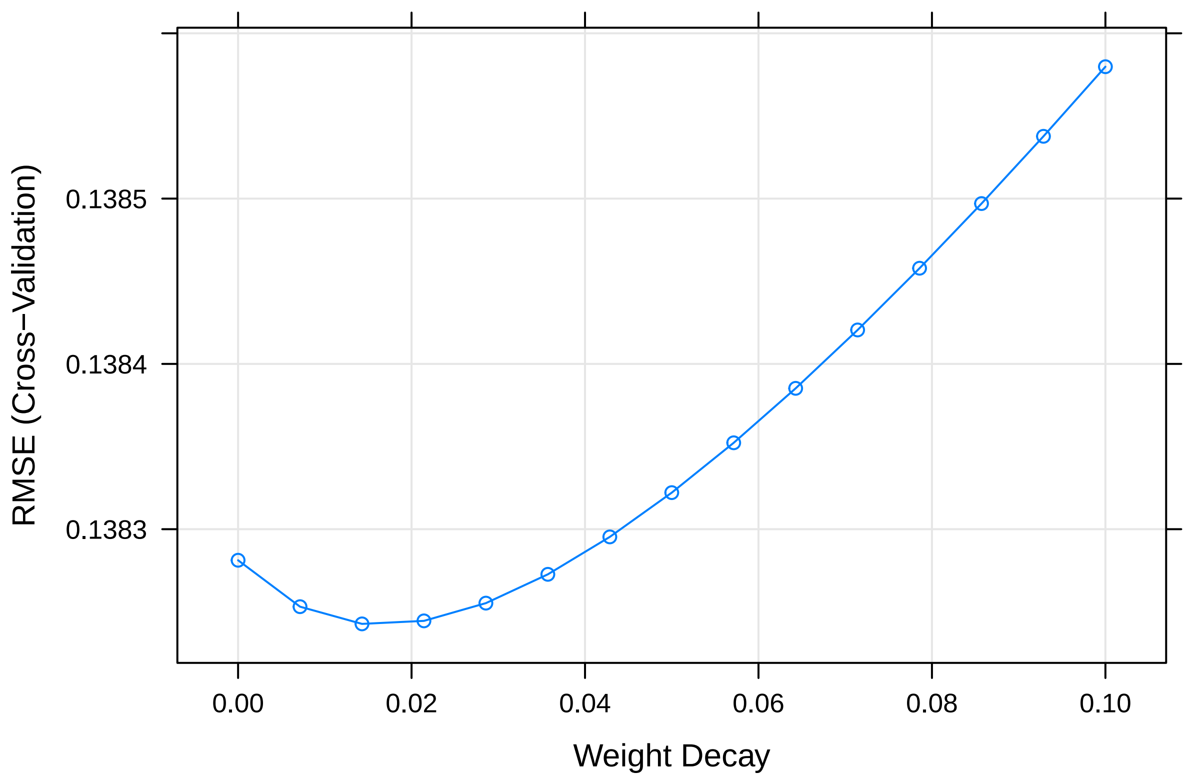
```r
ridgeGrid <- data.frame(.lambda = seq(0, .1, length = 15))

ridgeRegFit <- train(x_train, y_train$PH,
                     method = "ridge",
                     tuneGrid = ridgeGrid,
                     trControl = ctrl)
```

```r
summary(ridgeRegFit)
```

```
##             Length Class      Mode
## call          4    -none-     call
```

```
## actions       29    -none-     list
## allset        28    -none-     numeric
## beta.pure    812    -none-     numeric
## vn            28    -none-     character
## mu             1    -none-     numeric
## normx         28    -none-     numeric
## meanx         28    -none-     numeric
## lambda         1    -none-     numeric
## L1norm        29    -none-     numeric
## penalty       29    -none-     numeric
## df            29    -none-     numeric
## Cp            29    -none-     numeric
## sigma2         1    -none-     numeric
## xNames        28    -none-     character
## problemType    1    -none-     character
## tuneValue      1    data.frame list
## obsLevels      1    -none-     logical
## param          0    -none-     list
```

```
plot(ridgeRegFit)
```




```
ridgeRegFit$bestTune
```

```
##       lambda
## 3 0.01428571
```

```
train_set_results <- ridgeRegFit$results
```

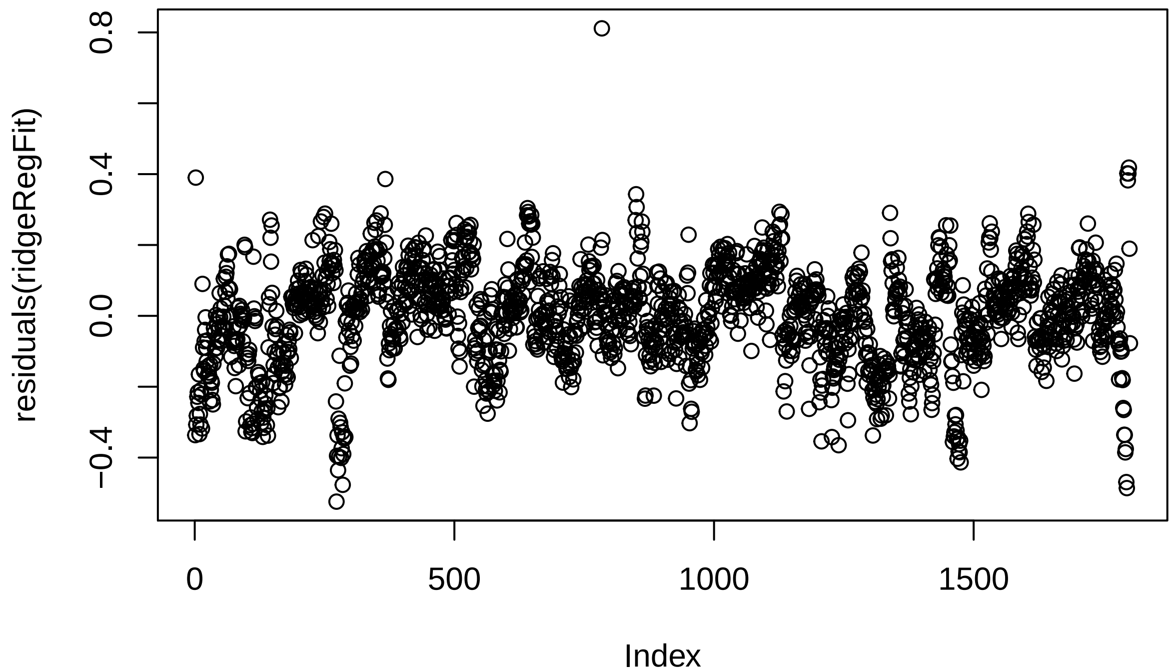
```r
train_set_results[row.names(train_set_results) == 3, ]
```

```
##       lambda      RMSE  Rsquared      MAE      RMSESD RsquaredSD       MAESD
## 3 0.01428571 0.1382427 0.3756954 0.107517 0.007556739 0.05095327 0.005721879
```

```r
varImp(ridgeRegFit)
```

```
## loess r-squared variable importance
##
##   only 20 most important variables shown (out of 28)
##
##                  Overall
## Mnf.Flow          100.000
## Filler.Level       74.270
## Usage.cont         70.762
## Pressure.Setpoint  51.601
## Fill.Pressure      43.998
## Hyd.Pressure1      40.263
## Oxygen.Filler      37.237
## Brand.Code.C       35.356
## Hyd.Pressure2      31.516
## Pressure.Vacuum    29.163
## Carb.Flow          26.963
## Carb.Pressure1     20.468
## Temperature        20.435
## Density            18.446
## Brand.Code.D       14.420
## Hyd.Pressure4      10.339
## MFR                 7.233
## Carb.Volume         7.002
## Fill.Ounces         6.348
## PSC                 5.047
```

```r
plot(residuals(ridgeRegFit) )
```



Non Linear Regression

KNN

```
knnModel <- train(x = x_train, y = y_train$PH,
                  method = "knn",
                  tuneLength = 25,
                  trControl = ctrl)

knnModel
```

```
## k-Nearest Neighbors
##
## 1801 samples
##   28 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1621, 1621, 1620, 1621, 1620, 1619, ...
## Resampling results across tuning parameters:
##
##   k   RMSE       Rsquared   MAE
##    5  0.1340149  0.4198952  0.1018072
##    7  0.1319761  0.4332301  0.1016147
##    9  0.1318947  0.4335663  0.1016890
##   11  0.1317862  0.4339747  0.1022382
##   13  0.1325901  0.4266643  0.1029896
```

```
##   15  0.1327474  0.4259332  0.1033898
##   17  0.1331351  0.4234831  0.1037501
##   19  0.1335267  0.4208473  0.1044374
##   21  0.1337902  0.4187518  0.1047829
##   23  0.1344088  0.4143112  0.1053090
##   25  0.1347482  0.4120417  0.1056039
##   27  0.1354875  0.4051696  0.1061722
##   29  0.1356462  0.4038117  0.1063172
##   31  0.1361376  0.3994086  0.1067051
##   33  0.1369304  0.3922715  0.1075400
##   35  0.1374899  0.3867060  0.1078387
##   37  0.1379881  0.3824862  0.1082361
##   39  0.1383345  0.3797083  0.1084839
##   41  0.1387659  0.3755893  0.1088894
##   43  0.1389945  0.3735262  0.1091350
##   45  0.1393386  0.3705752  0.1095241
##   47  0.1395985  0.3680951  0.1096171
##   49  0.1397903  0.3660948  0.1097383
##   51  0.1401446  0.3627572  0.1099668
##   53  0.1403417  0.3610692  0.1101443
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 11.
```

```
knnPred <- predict(knnModel, newdata = x_test)

knn_res <- postResample(pred = knnPred, obs = y_test$PH)
knn_res
```

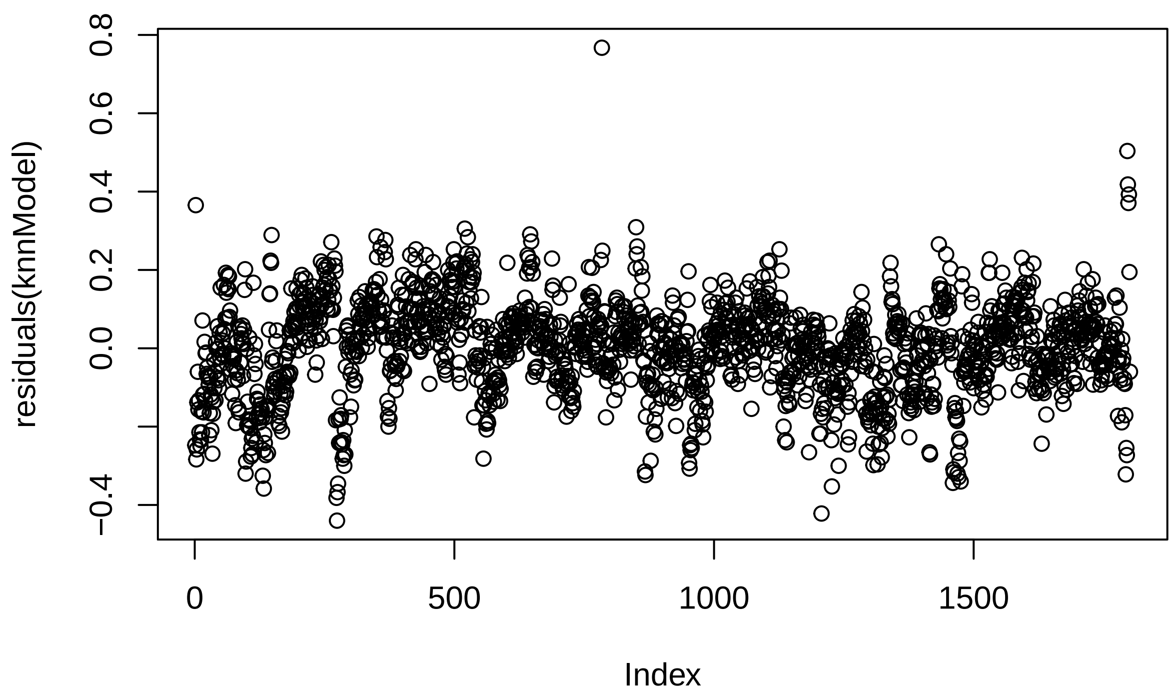
```
##      RMSE  Rsquared       MAE
## 0.1278973 0.4150442 0.1003442
```

```
varImp(knnModel)
```

```
## loess r-squared variable importance
##
##   only 20 most important variables shown (out of 28)
##
##                  Overall
## Mnf.Flow         100.000
## Filler.Level      74.270
## Usage.cont        70.762
## Pressure.Setpoint  51.601
## Fill.Pressure     43.998
## Hyd.Pressure1     40.263
## Oxygen.Filler     37.237
## Brand.Code.C      35.356
## Hyd.Pressure2     31.516
## Pressure.Vacuum   29.163
## Carb.Flow         26.963
## Carb.Pressure1    20.468
## Temperature       20.435
## Density           18.446
## Brand.Code.D      14.420
## Hyd.Pressure4     10.339
```

```
## MFR                    7.233
## Carb.Volume            7.002
## Fill.Ounces            6.348
## PSC                    5.047
```

```r
plot(residuals(knnModel))
```



Neural Network

```r
nnetGrid <- expand.grid(.decay = c(0, .01, 1),
                        .size = c(1:10),
                        .bag = FALSE)
set.seed(100)
nnetTune <- train(x = x_train,
                  y = y_train$PH,
                  method = "avNNet",
                  tuneGrid = nnetGrid,
                  trControl = ctrl,
                  linout = FALSE,  trace = FALSE,
                  MaxNWts = 5* (ncol(x_train) + 1) + 5 + 1,
                  maxit = 250)
```

```r
nnetTune
```

```
## Model Averaged Neural Network
##
## 1801 samples
##   28 predictor
```

```
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1620, 1621, 1621, 1620, 1621, 1620, ...
## Resampling results across tuning parameters:
##
##   decay  size  RMSE      Rsquared    MAE
##   0.00    1    7.547784         NaN  7.545766
##   0.00    2    7.547784         NaN  7.545766
##   0.00    3    7.547784         NaN  7.545766
##   0.00    4    7.547784         NaN  7.545766
##   0.00    5    7.547784         NaN  7.545766
##   0.00    6         NaN         NaN       NaN
##   0.00    7         NaN         NaN       NaN
##   0.00    8         NaN         NaN       NaN
##   0.00    9         NaN         NaN       NaN
##   0.00   10         NaN         NaN       NaN
##   0.01    1    7.547789  0.04386820  7.545771
##   0.01    2    7.547788  0.05012390  7.545770
##   0.01    3    7.547787  0.04638795  7.545769
##   0.01    4    7.547787  0.04833517  7.545769
##   0.01    5    7.547786  0.05088307  7.545769
##   0.01    6         NaN         NaN       NaN
##   0.01    7         NaN         NaN       NaN
##   0.01    8         NaN         NaN       NaN
##   0.01    9         NaN         NaN       NaN
##   0.01   10         NaN         NaN       NaN
##   1.00    1    7.548144  0.04194292  7.546127
##   1.00    2    7.548061  0.04345842  7.546044
##   1.00    3    7.548018  0.04196447  7.546000
##   1.00    4    7.547992  0.04233421  7.545974
##   1.00    5    7.547974  0.04274944  7.545957
##   1.00    6         NaN         NaN       NaN
##   1.00    7         NaN         NaN       NaN
##   1.00    8         NaN         NaN       NaN
##   1.00    9         NaN         NaN       NaN
##   1.00   10         NaN         NaN       NaN
##
## Tuning parameter 'bag' was held constant at a value of FALSE
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were size = 1, decay = 0 and bag = FALSE.
```

```r
summary(nnetTune)
```

```
##             Length Class      Mode
## model        5     -none-     list
## repeats      1     -none-     numeric
## bag          1     -none-     logical
## seeds        5     -none-     numeric
## names       28     -none-     character
## terms        3     terms      call
## coefnames   28     -none-     character
## xlevels      0     -none-     list
## xNames      28     -none-     character
## problemType  1     -none-     character
```

```
## tuneValue    3     data.frame list
## obsLevels    1     -none-     logical
## param        4     -none-     list
```

```
nnetTune$bestTune
```

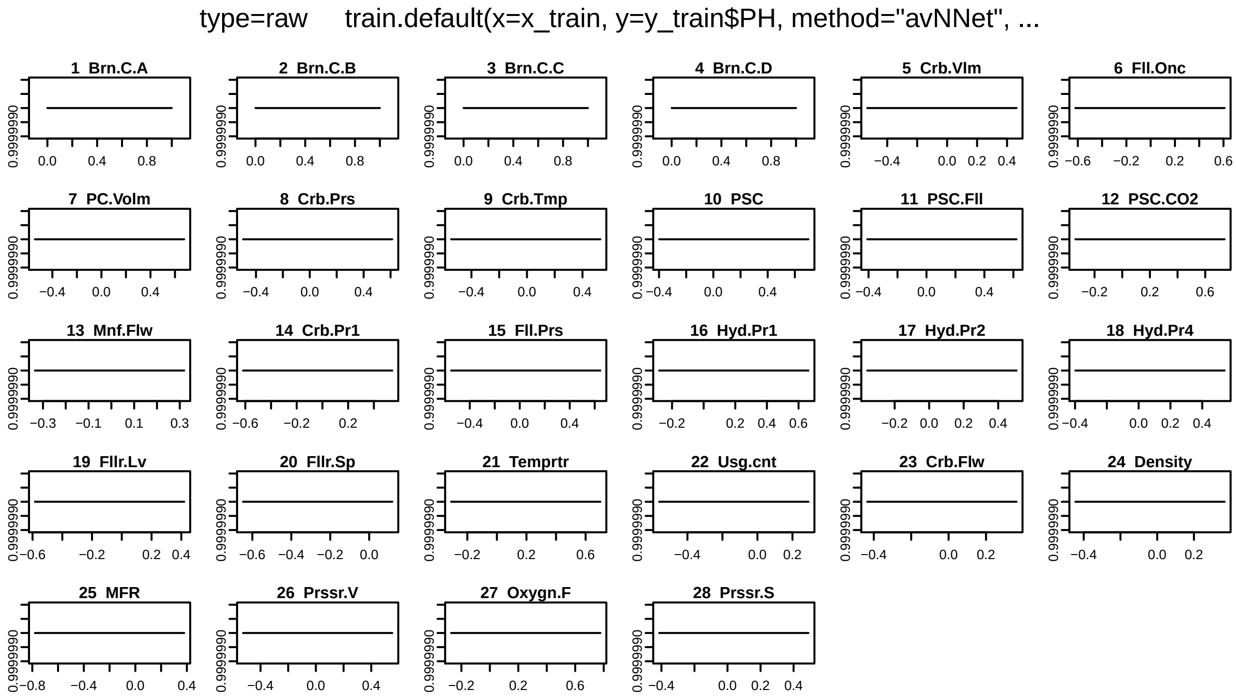
```
##   size decay   bag
## 1    1     0 FALSE
```

```
nnetPred <- predict(nnetTune, newdata=x_test)
NNET <- postResample(pred = nnetPred, obs = y_test$PH)
NNET
```

```
##     RMSE Rsquared      MAE
## 7.547201       NA 7.545351
```

```
plotmo(nnetTune)
```

```
##  plotmo grid:    Brand.Code.A Brand.Code.B Brand.Code.C Brand.Code.D
##                             0            1            0            0
##  Carb.Volume Fill.Ounces   PC.Volume Carb.Pressure    Carb.Temp         PSC
##  -0.04645293 0.008689866 -0.01710397 -0.0007182825 -0.008149797 -0.02915067
##     PSC.Fill     PSC.CO2    Mnf.Flow Carb.Pressure1 Fill.Pressure Hyd.Pressure1
##  -0.02981358 -0.06377821 -0.02013707     0.02018178   -0.07738622   -0.01698975
##  Hyd.Pressure2 Hyd.Pressure4 Filler.Level Filler.Speed Temperature Usage.cont
##     0.07755498  -0.004587089   0.09206058     0.067918 -0.03273905 0.05305285
##   Carb.Flow    Density        MFR Pressure.Vacuum Oxygen.Filler
##  0.09796908 -0.1026698 0.06451337     0.004038687   -0.05295248
##  Pressure.Setpoint
##         -0.1154038
```




```r
varImp(nnetTune)
```

```
## loess r-squared variable importance
##
##   only 20 most important variables shown (out of 28)
##
##                    Overall
## Mnf.Flow           100.000
## Filler.Level        74.270
## Usage.cont          70.762
## Pressure.Setpoint   51.601
## Fill.Pressure       43.998
## Hyd.Pressure1       40.263
## Oxygen.Filler       37.237
## Brand.Code.C        35.356
## Hyd.Pressure2       31.516
## Pressure.Vacuum     29.163
## Carb.Flow           26.963
## Carb.Pressure1      20.468
## Temperature         20.435
## Density             18.446
## Brand.Code.D        14.420
## Hyd.Pressure4       10.339
## MFR                  7.233
## Carb.Volume          7.002
## Fill.Ounces          6.348
```

```
## PSC                  5.047
```

Multivariate Adaptive Regression Splines (MARS)

```r
set.seed(100)
marsGrid <- expand.grid(.degree = 1:2, .nprune = 2:38)
marsTuned <- train(x = x_train, y =  y_train$PH,
                   method = "earth",
                   tuneGrid = marsGrid,
                   trControl = ctrl)
marsTuned
```

```
## Multivariate Adaptive Regression Spline
##
## 1801 samples
##   28 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1620, 1621, 1621, 1620, 1621, 1620, ...
## Resampling results across tuning parameters:
##
##   degree  nprune  RMSE       Rsquared   MAE
##   1        2      0.1542164  0.2217362  0.1195072
##   1        3      0.1468508  0.2953406  0.1138499
##   1        4      0.1452570  0.3104815  0.1121958
##   1        5      0.1437347  0.3243984  0.1109075
##   1        6      0.1404495  0.3545138  0.1087082
##   1        7      0.1394742  0.3633495  0.1073645
##   1        8      0.1384858  0.3720735  0.1066987
##   1        9      0.1372680  0.3829202  0.1053837
##   1       10      0.1358953  0.3951145  0.1048610
##   1       11      0.1357252  0.3966879  0.1045986
##   1       12      0.1359052  0.3951784  0.1049076
##   1       13      0.1360120  0.3946591  0.1054601
##   1       14      0.1357370  0.3971147  0.1051532
##   1       15      0.1360705  0.3945715  0.1053267
##   1       16      0.1357233  0.3976718  0.1046216
##   1       17      0.1357158  0.3981053  0.1046602
##   1       18      0.1354330  0.4004641  0.1044503
##   1       19      0.1354592  0.4003640  0.1043838
##   1       20      0.1354196  0.4006061  0.1043569
##   1       21      0.1356451  0.3988864  0.1045525
##   1       22      0.1354537  0.4004854  0.1044934
##   1       23      0.1352699  0.4019418  0.1043376
##   1       24      0.1353009  0.4017355  0.1042640
##   1       25      0.1353180  0.4016334  0.1042644
##   1       26      0.1353157  0.4016763  0.1041587
##   1       27      0.1352625  0.4021464  0.1042006
##   1       28      0.1352128  0.4025286  0.1041860
##   1       29      0.1352128  0.4025286  0.1041860
##   1       30      0.1352128  0.4025286  0.1041860
##   1       31      0.1353116  0.4016722  0.1041457
##   1       32      0.1353144  0.4016416  0.1041427
##   1       33      0.1352295  0.4023365  0.1041075
```

```
##   1       34      0.1351542  0.4029966  0.1040646
##   1       35      0.1351542  0.4029966  0.1040646
##   1       36      0.1351542  0.4029966  0.1040646
##   1       37      0.1351542  0.4029966  0.1040646
##   1       38      0.1351542  0.4029966  0.1040646
##   2        2      0.1537405  0.2265201  0.1188948
##   2        3      0.1463416  0.3011021  0.1134635
##   2        4      0.1461410  0.3022045  0.1139416
##   2        5      0.1431990  0.3316124  0.1114897
##   2        6      0.1416153  0.3452882  0.1102524
##   2        7      0.1397388  0.3625854  0.1087208
##   2        8      0.1389192  0.3696850  0.1082247
##   2        9      0.1436782  0.3525740  0.1079955
##   2       10      0.1423040  0.3648018  0.1067096
##   2       11      0.1418526  0.3720871  0.1062987
##   2       12      0.1402129  0.3859035  0.1048715
##   2       13      0.1397072  0.3902822  0.1040967
##   2       14      0.1387731  0.3969865  0.1033965
##   2       15      0.1399163  0.3883268  0.1030990
##   2       16      0.1394903  0.3944617  0.1023772
##   2       17      0.1390311  0.3987787  0.1019587
##   2       18      0.1393932  0.3955354  0.1025003
##   2       19      0.1393256  0.3944866  0.1024584
##   2       20      0.1393912  0.3943803  0.1021602
##   2       21      0.1390628  0.3976023  0.1023312
##   2       22      0.1392143  0.3962562  0.1020363
##   2       23      0.1394411  0.3950862  0.1019348
##   2       24      0.1392690  0.3967150  0.1017345
##   2       25      0.1388123  0.3992114  0.1013344
##   2       26      0.1381927  0.4033762  0.1009827
##   2       27      0.1380066  0.4040897  0.1009660
##   2       28      0.1381255  0.4040401  0.1011905
##   2       29      0.1383748  0.4033621  0.1013298
##   2       30      0.1384980  0.4036653  0.1013044
##   2       31      0.1382166  0.4041369  0.1012274
##   2       32      0.1381347  0.4050201  0.1010860
##   2       33      0.1381347  0.4050201  0.1010860
##   2       34      0.1381347  0.4050201  0.1010860
##   2       35      0.1381347  0.4050201  0.1010860
##   2       36      0.1381347  0.4050201  0.1010860
##   2       37      0.1381347  0.4050201  0.1010860
##   2       38      0.1381347  0.4050201  0.1010860
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were nprune = 34 and degree = 1.
```

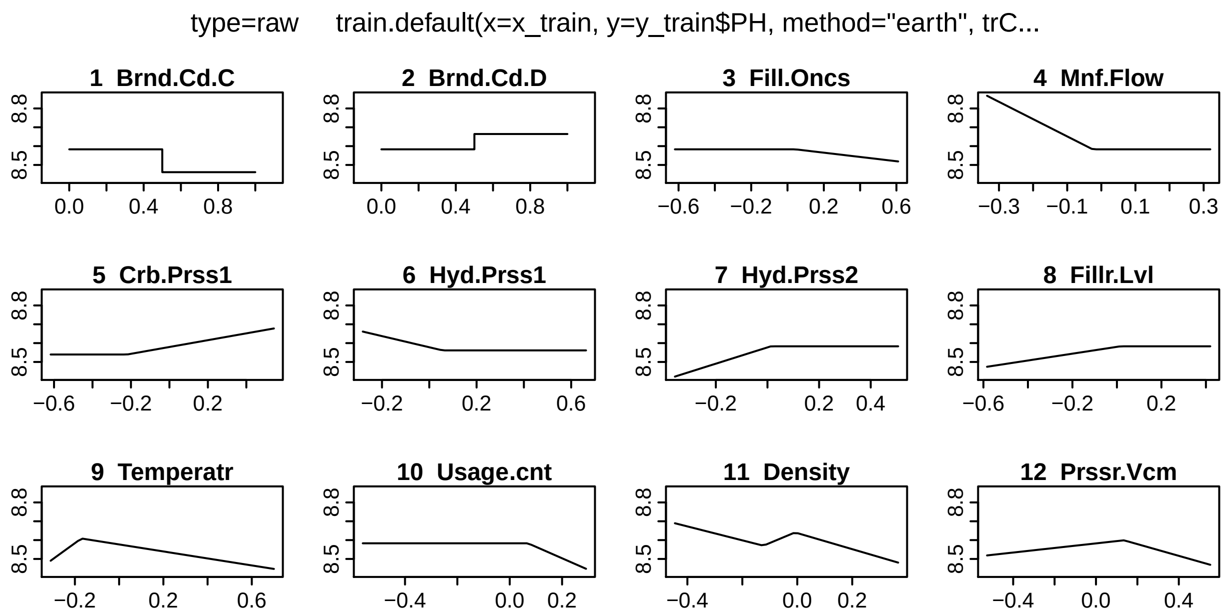
```r
marsPred <- predict(marsTuned, newdata=x_test)
MARS <- postResample(pred = marsPred, obs = y_test$PH)
MARS
```

```
##       RMSE   Rsquared        MAE
## 0.12797018 0.41432533 0.09927371
```

```r
plotmo(marsTuned)
```

```
##  plotmo grid:    Brand.Code.A Brand.Code.B Brand.Code.C Brand.Code.D
```

```
##                              0             1             0             0
##   Carb.Volume Fill.Ounces   PC.Volume Carb.Pressure    Carb.Temp          PSC
##   -0.04645293 0.008689866 -0.01710397 -0.0007182825 -0.008149797 -0.02915067
##      PSC.Fill     PSC.CO2    Mnf.Flow Carb.Pressure1 Fill.Pressure Hyd.Pressure1
##   -0.02981358 -0.06377821 -0.02013707     0.02018178   -0.07738622   -0.01698975
##   Hyd.Pressure2 Hyd.Pressure4 Filler.Level Filler.Speed Temperature Usage.cont
##      0.07755498  -0.004587089   0.09206058     0.067918 -0.03273905 0.05305285
##    Carb.Flow    Density        MFR Pressure.Vacuum Oxygen.Filler
##   0.09796908 -0.1026698 0.06451337     0.004038687   -0.05295248
##   Pressure.Setpoint
##          -0.1154038
```

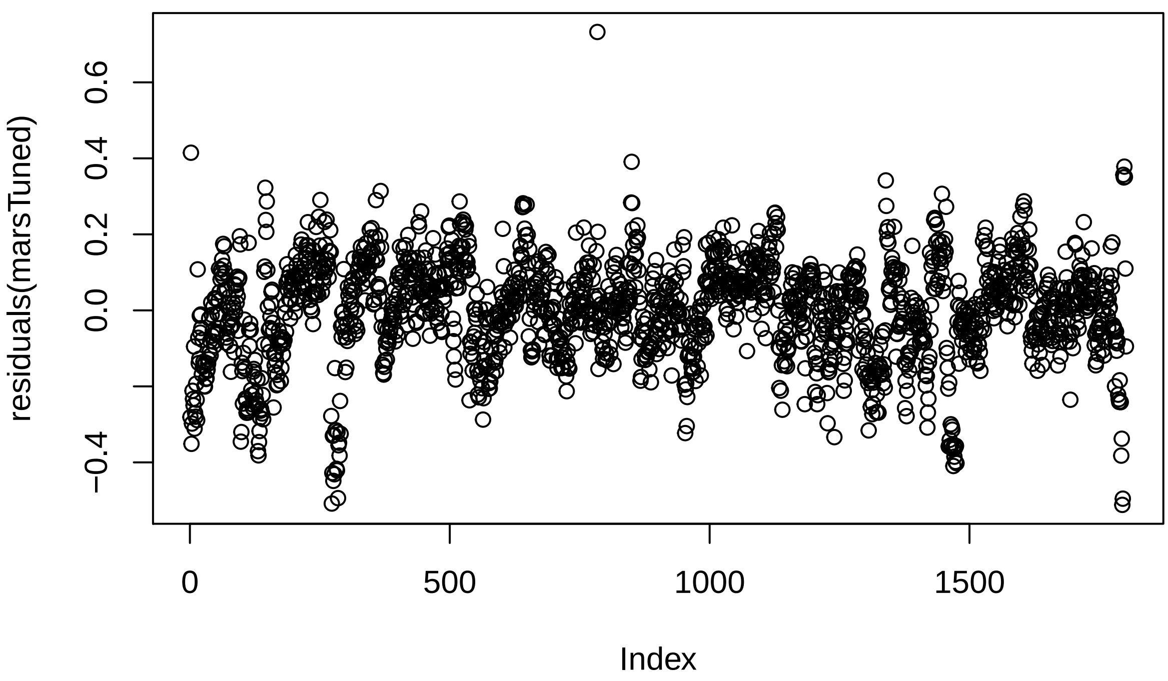



```r
varImp(marsTuned)
```

```
## earth variable importance
##
##                 Overall
## Mnf.Flow        100.000
## Brand.Code.C     65.411
## Brand.Code.D     46.362
## Usage.cont       42.769
## Carb.Pressure1   37.998
## Temperature      31.778
## Filler.Level     27.531
## Pressure.Vacuum  27.531
## Density          19.088
## Hyd.Pressure2    11.645
```

```
## Hyd.Pressure1     9.455
## Fill.Ounces       0.000
```

```r
plot(residuals(marsTuned))
```



Support Vector Machines (SVM)

```r
set.seed(100)
svmLTuned <- train(x = x_train, y =  y_train$PH,
                   method = "svmLinear",
                   tuneLength = 25,
                   trControl = trainControl(method = "cv"))
svmLTuned
```

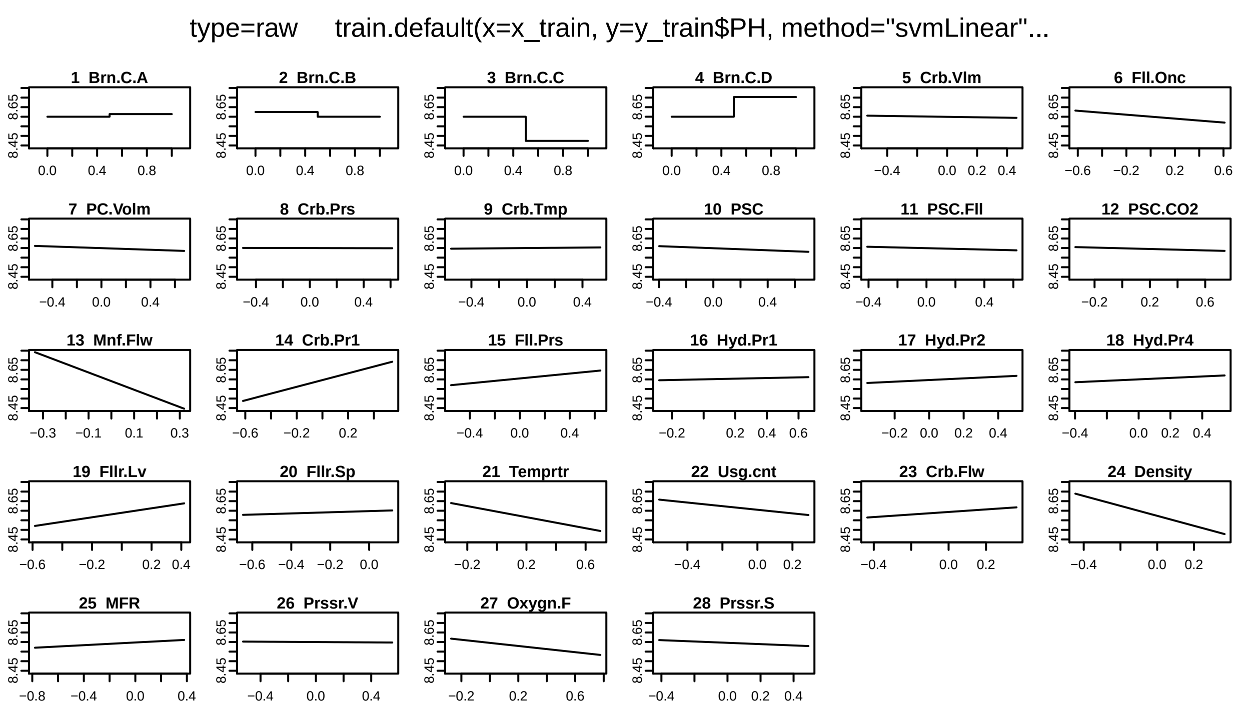
```
## Support Vector Machines with Linear Kernel
##
## 1801 samples
##   28 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1620, 1621, 1621, 1620, 1621, 1620, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.1409549  0.3563769  0.1070378
##
## Tuning parameter 'C' was held constant at a value of 1
```

```r
svmLPred <- predict(svmLTuned, newdata=x_test)
svmL<- postResample(pred = svmLPred, obs = y_test$PH)
svmL
```

```
##      RMSE  Rsquared       MAE
## 0.1310617 0.3907003 0.1002711
```

```r
plotmo(svmLTuned)
```

```
##  plotmo grid:    Brand.Code.A Brand.Code.B Brand.Code.C Brand.Code.D
##                             0            1            0            0
##  Carb.Volume Fill.Ounces   PC.Volume Carb.Pressure    Carb.Temp         PSC
##  -0.04645293 0.008689866 -0.01710397 -0.0007182825 -0.008149797 -0.02915067
##     PSC.Fill     PSC.CO2    Mnf.Flow Carb.Pressure1 Fill.Pressure Hyd.Pressure1
##  -0.02981358 -0.06377821 -0.02013707      0.02018178   -0.07738622   -0.01698975
##  Hyd.Pressure2 Hyd.Pressure4 Filler.Level Filler.Speed Temperature Usage.cont
##     0.07755498   -0.004587089   0.09206058     0.067918 -0.03273905 0.05305285
##   Carb.Flow    Density        MFR Pressure.Vacuum Oxygen.Filler
##  0.09796908 -0.1026698 0.06451337     0.004038687   -0.05295248
##  Pressure.Setpoint
##         -0.1154038
```




```r
varImp(svmLTuned)
```

```
## loess r-squared variable importance
##
##   only 20 most important variables shown (out of 28)
##
```

```
##                   Overall
## Mnf.Flow          100.000
## Filler.Level       74.270
## Usage.cont         70.762
## Pressure.Setpoint  51.601
## Fill.Pressure      43.998
## Hyd.Pressure1      40.263
## Oxygen.Filler      37.237
## Brand.Code.C       35.356
## Hyd.Pressure2      31.516
## Pressure.Vacuum    29.163
## Carb.Flow          26.963
## Carb.Pressure1     20.468
## Temperature        20.435
## Density            18.446
## Brand.Code.D       14.420
## Hyd.Pressure4      10.339
## MFR                 7.233
## Carb.Volume         7.002
## Fill.Ounces         6.348
## PSC                 5.047
```

Tree Based Models

```r
resamples <- resamples( list(CondInfTree =ctreeModel,
                             BaggedTree = baggedTreeModel,
                             BoostedTree = gbmModel,
                             Cubist=cubistModel) )
summary(resamples)
```

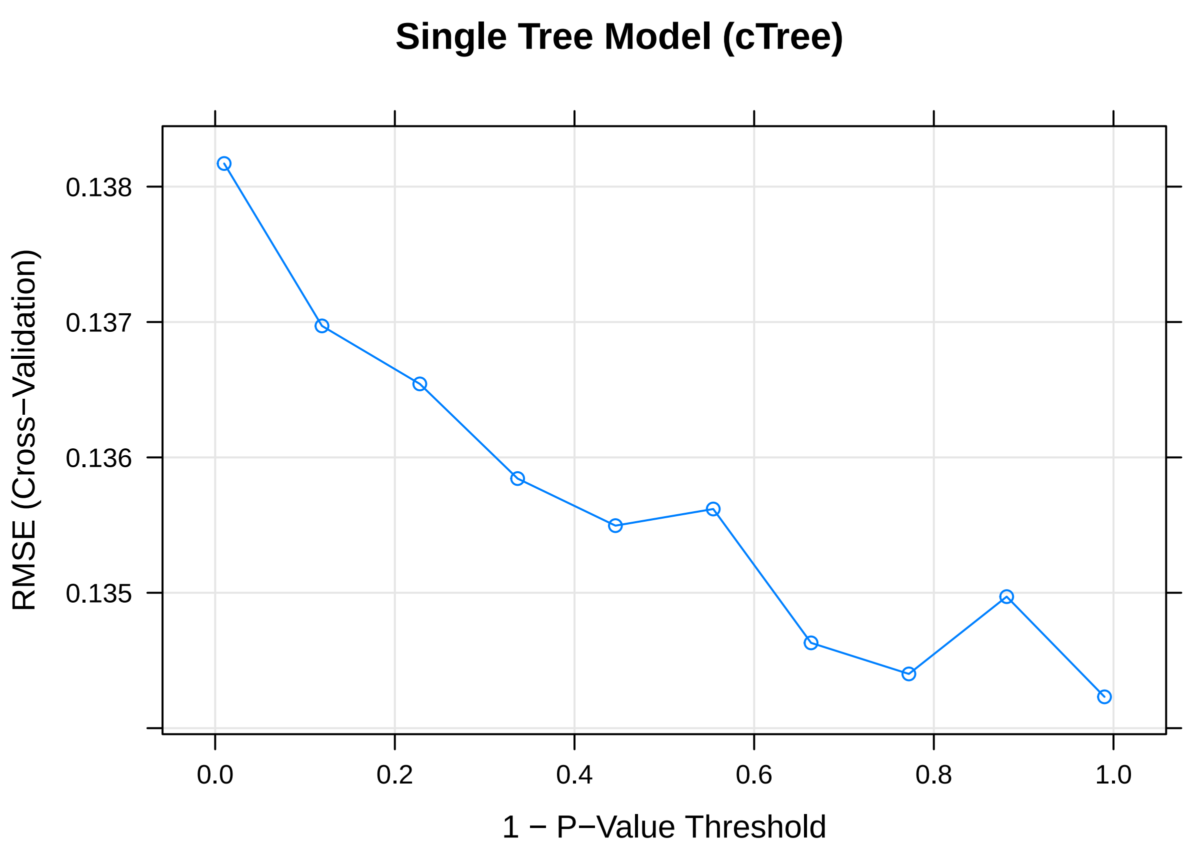
```
##
## Call:
## summary.resamples(object = resamples)
##
## Models: CondInfTree, BaggedTree, BoostedTree, Cubist
## Number of resamples: 10
##
## MAE
##                   Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## CondInfTree 0.09748700 0.09973819 0.10137885 0.10184894 0.10284267 0.10890526
## BaggedTree  0.07614880 0.08173620 0.08436665 0.08381540 0.08723185 0.08885333
## BoostedTree 0.08382721 0.09004830 0.09050145 0.09128126 0.09315838 0.09837226
## Cubist      0.07497506 0.08143732 0.08259427 0.08221446 0.08347221 0.08683017
##             NA's
## CondInfTree    0
## BaggedTree     0
## BoostedTree    0
## Cubist         0
##
## RMSE
##                   Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## CondInfTree 0.12261617 0.1325279 0.1343978 0.1342313 0.1385185 0.1410012    0
## BaggedTree  0.10121752 0.1130539 0.1157395 0.1149936 0.1208552 0.1225774    0
## BoostedTree 0.10985549 0.1186117 0.1201996 0.1208558 0.1219393 0.1300881    0
```

```
## Cubist      0.09954642 0.1092555 0.1136065 0.1124479 0.1163646 0.1217142    0
##
## Rsquared
##                   Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## CondInfTree 0.3599060 0.3929146 0.4062546 0.4163824 0.4292919 0.4990760    0
## BaggedTree  0.4885028 0.5339024 0.5511945 0.5680971 0.5953822 0.6908128    0
## BoostedTree 0.4738912 0.4935708 0.5189947 0.5231860 0.5378904 0.6324343    0
## Cubist      0.5299562 0.5528523 0.5717225 0.5845657 0.6079971 0.6898532    0
```

Single Tree Models - cTree

```
convert_top_20_to_df <- function(df){
          df1 <- as.data.frame(df)
          df1['Predictors']  <- rownames(df)
          colnames(df1) <- c("Overall", "Predictors")
          rownames(df1) <- 1:nrow(df1)

          return (df1)
}
```

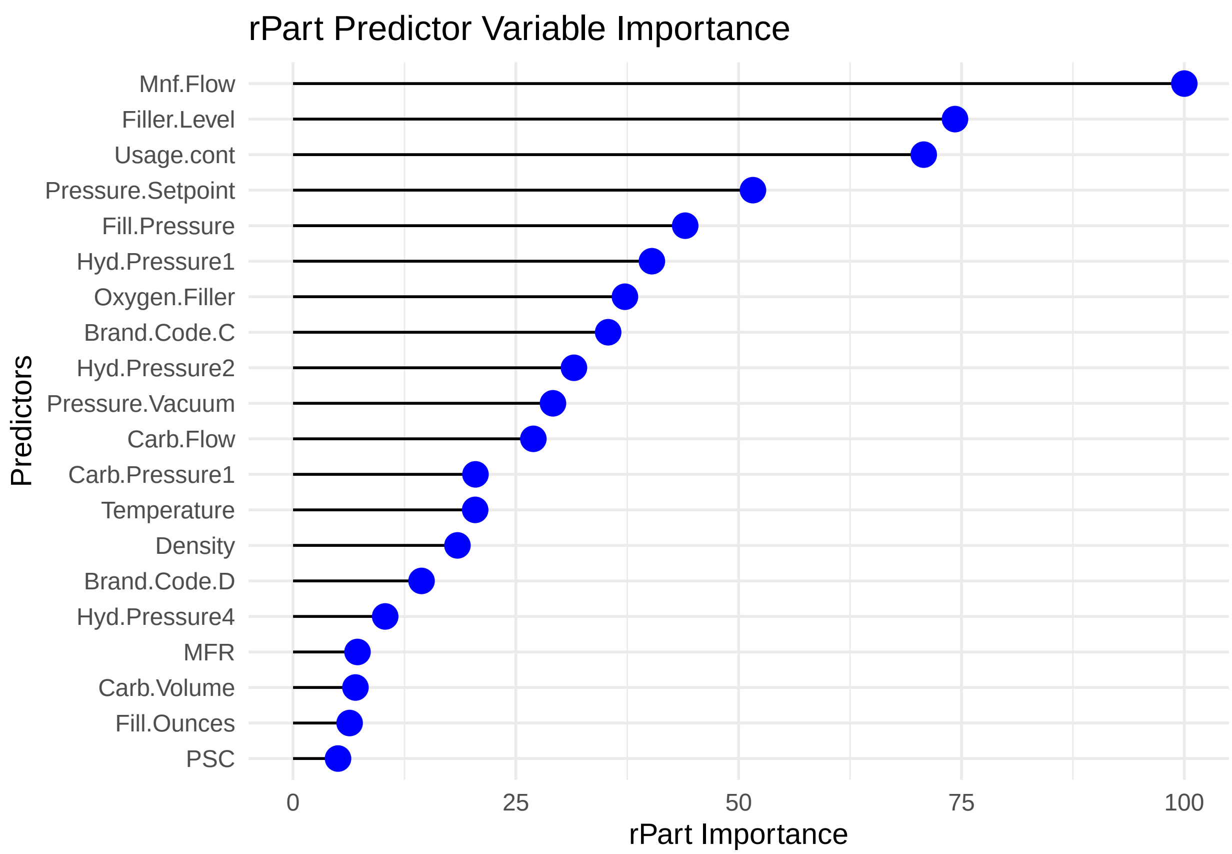
```
plot(ctreeModel, main = "Single Tree Model (cTree)")
```




```
ctree_20 <- varImp(ctreeModel)
ctree_20 <- ctree_20$importance %>%
  arrange(desc(Overall))
ctree_20 <-   head(ctree_20,20)
ctree_20
```

```
##                         Overall
## Mnf.Flow             100.000000
## Filler.Level          74.270079
## Usage.cont            70.761544
## Pressure.Setpoint     51.601015
## Fill.Pressure         43.998446
## Hyd.Pressure1         40.262801
## Oxygen.Filler         37.237249
## Brand.Code.C          35.355850
## Hyd.Pressure2         31.516464
## Pressure.Vacuum       29.163224
## Carb.Flow             26.962829
## Carb.Pressure1        20.467789
## Temperature           20.434779
## Density               18.446136
## Brand.Code.D          14.419748
## Hyd.Pressure4         10.338789
## MFR                    7.232773
## Carb.Volume            7.001843
## Fill.Ounces            6.348329
## PSC                    5.046529
```

```
ctree_20_df<- convert_top_20_to_df(ctree_20)

ctree_20_df %>%
            arrange(Overall)%>%
            mutate(name = factor(Predictors, levels=c(Predictors))) %>%
            ggplot(aes(x=name, y=Overall)) +
            geom_segment(aes(xend = Predictors, yend = 0)) +
            geom_point(size = 4, color = "blue") +
            theme_minimal() +
            coord_flip() +
            labs(title="rPart Predictor Variable Importance",
               y="rPart Importance", x="Predictors") +
            scale_y_continuous()
```




```
cTreePred <- predict(ctreeModel, newdata=x_test)
cTreePred <- postResample(pred = cTreePred, obs = y_test$PH)
cTreePred
```

```
##       RMSE   Rsquared        MAE
## 0.12554977 0.43777524 0.09836585
```

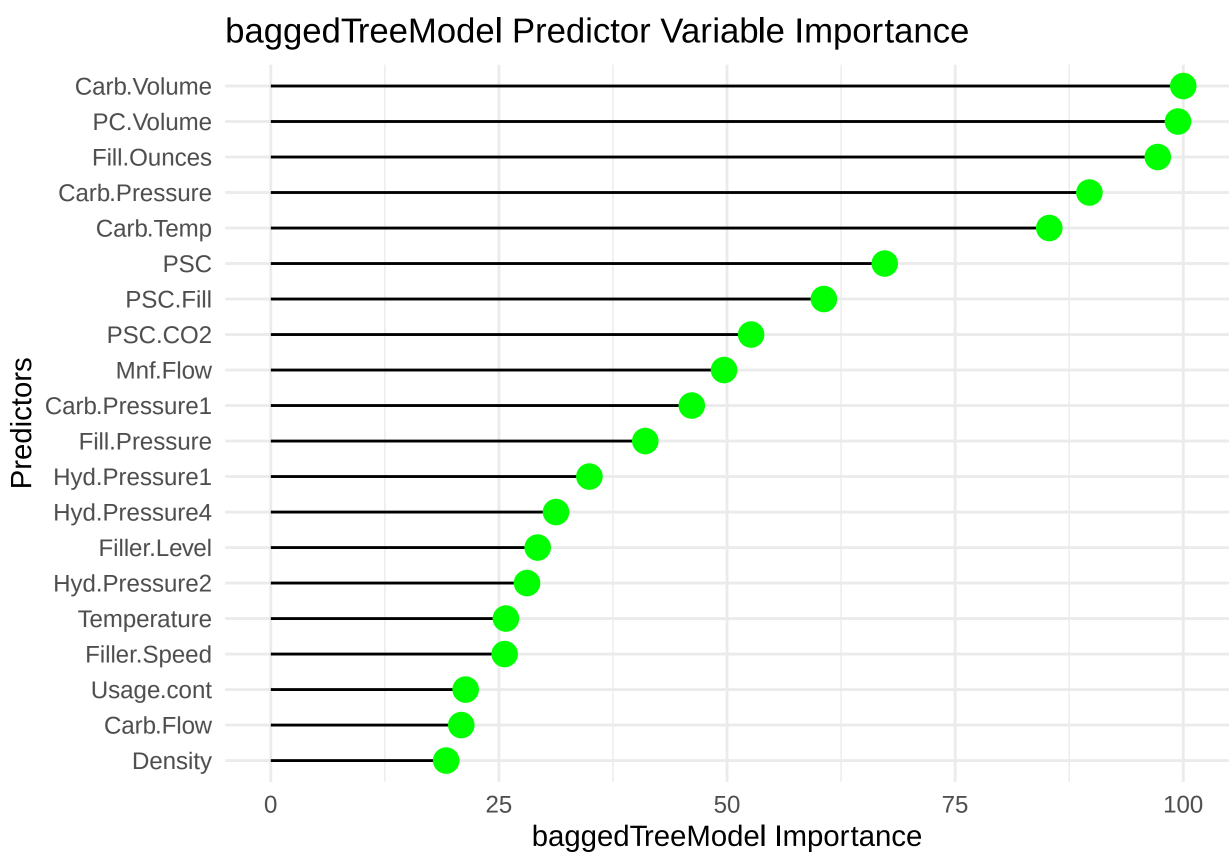
Bagged Trees - baggedTreeModel

```
baggedTreeModel_20 <- varImp(baggedTreeModel)
baggedTreeModel_20 <- baggedTreeModel_20$importance %>%
  arrange(desc(Overall))
baggedTreeModel_20 <-   head(baggedTreeModel_20,20)
baggedTreeModel_20
```

```
##                  Overall
## Carb.Volume    100.00000
## PC.Volume       99.42199
## Fill.Ounces     97.20542
## Carb.Pressure   89.71322
## Carb.Temp       85.31748
## PSC             67.28559
## PSC.Fill        60.61346
## PSC.CO2         52.64018
## Mnf.Flow        49.67340
## Carb.Pressure1  46.13642
## Fill.Pressure   41.04158
## Hyd.Pressure1   34.91706
```

```
## Hyd.Pressure4   31.25789
## Filler.Level    29.24654
## Hyd.Pressure2   28.08143
## Temperature     25.76663
## Filler.Speed    25.63116
## Usage.cont      21.35433
## Carb.Flow       20.87719
## Density         19.22116
```

```
baggedTreeModel_20_df<- convert_top_20_to_df(baggedTreeModel_20)

baggedTreeModel_20_df %>%
            arrange(Overall)%>%
            mutate(name = factor(Predictors, levels=c(Predictors))) %>%
            ggplot(aes(x=name, y=Overall)) +
            geom_segment(aes(xend = Predictors, yend = 0)) +
            geom_point(size = 4, color = "green") +
            theme_minimal() +
            coord_flip() +
            labs(title="baggedTreeModel Predictor Variable Importance",
               y="baggedTreeModel Importance", x="Predictors") +
            scale_y_continuous()
```




```
baggedTreeModelPred <- predict(baggedTreeModel, newdata=x_test)
baggedTreeModelPred <- postResample(pred = baggedTreeModelPred, obs = y_test$PH)
baggedTreeModel
```

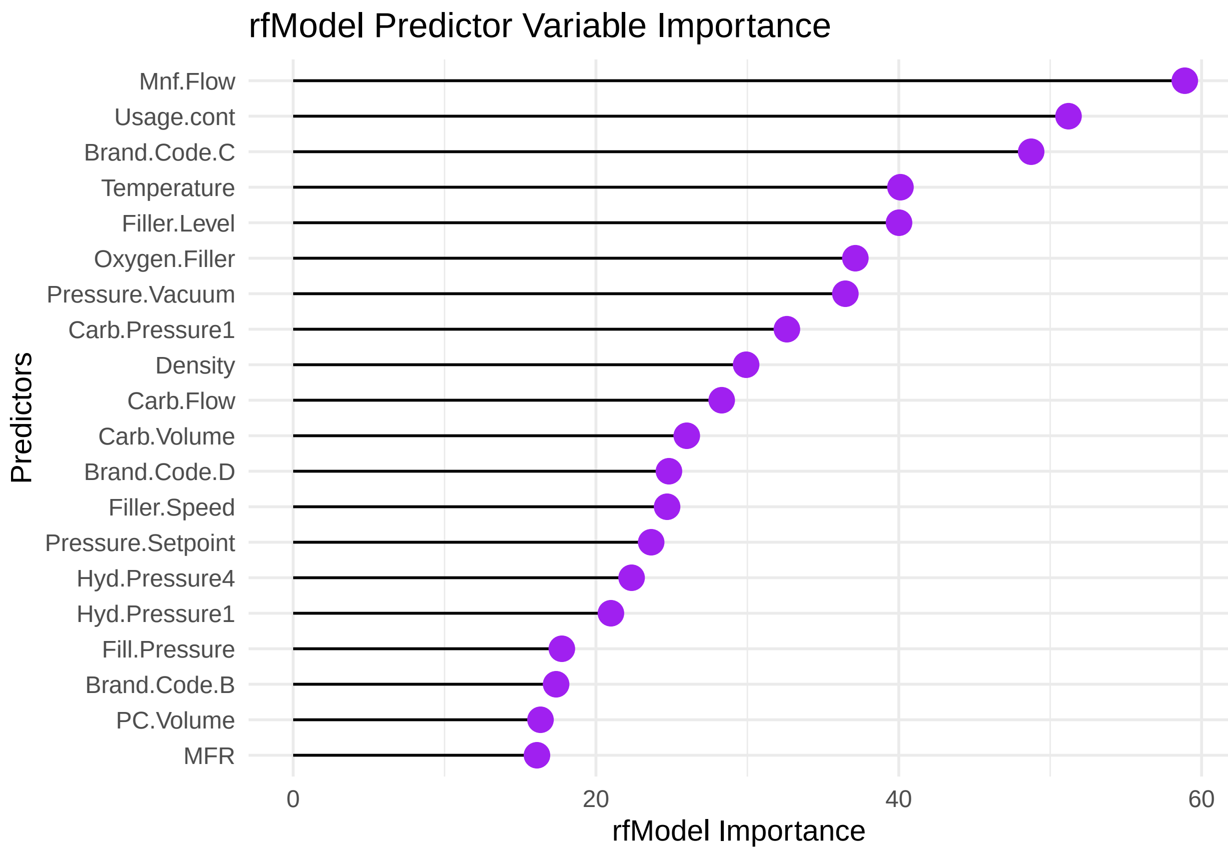
```
## Bagged CART
##
## 1801 samples
##   28 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1620, 1621, 1621, 1620, 1621, 1620, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.1149936  0.5680971  0.0838154
```

Random Forest - rfModel

```
rfModel_20 <- varImp(rfModel)
rfModel_20 <- rfModel_20 %>%
  arrange(desc(Overall))
rfModel_20 <-   head(rfModel_20,20)
rfModel_20
```

```
##                    Overall
## Mnf.Flow          58.89159
## Usage.cont        51.21166
## Brand.Code.C      48.74403
## Temperature       40.11116
## Filler.Level      40.01638
## Oxygen.Filler     37.13330
## Pressure.Vacuum   36.47272
## Carb.Pressure1    32.61052
## Density           29.91859
## Carb.Flow         28.30570
## Carb.Volume       25.99788
## Brand.Code.D      24.81922
## Filler.Speed      24.69797
## Pressure.Setpoint 23.64654
## Hyd.Pressure4     22.35474
## Hyd.Pressure1     20.98779
## Fill.Pressure     17.74656
## Brand.Code.B      17.36858
## PC.Volume         16.33673
## MFR               16.10223
```

```
rfModel_20_df<- convert_top_20_to_df(rfModel_20)

rfModel_20_df %>%
            arrange(Overall)%>%
            mutate(name = factor(Predictors, levels=c(Predictors))) %>%
            ggplot(aes(x=name, y=Overall)) +
            geom_segment(aes(xend = Predictors, yend = 0)) +
            geom_point(size = 4, color = "purple") +
            theme_minimal() +
            coord_flip() +
            labs(title="rfModel Predictor Variable Importance",
               y="rfModel Importance", x="Predictors") +
            scale_y_continuous()
```




```
rfModelPred <- predict(rfModel, newdata=x_test)
rfModelPred <- postResample(pred = rfModelPred, obs = y_test$PH)
rfModelPred
```

```
##       RMSE   Rsquared        MAE
## 0.10516632 0.62244111 0.07938496
```

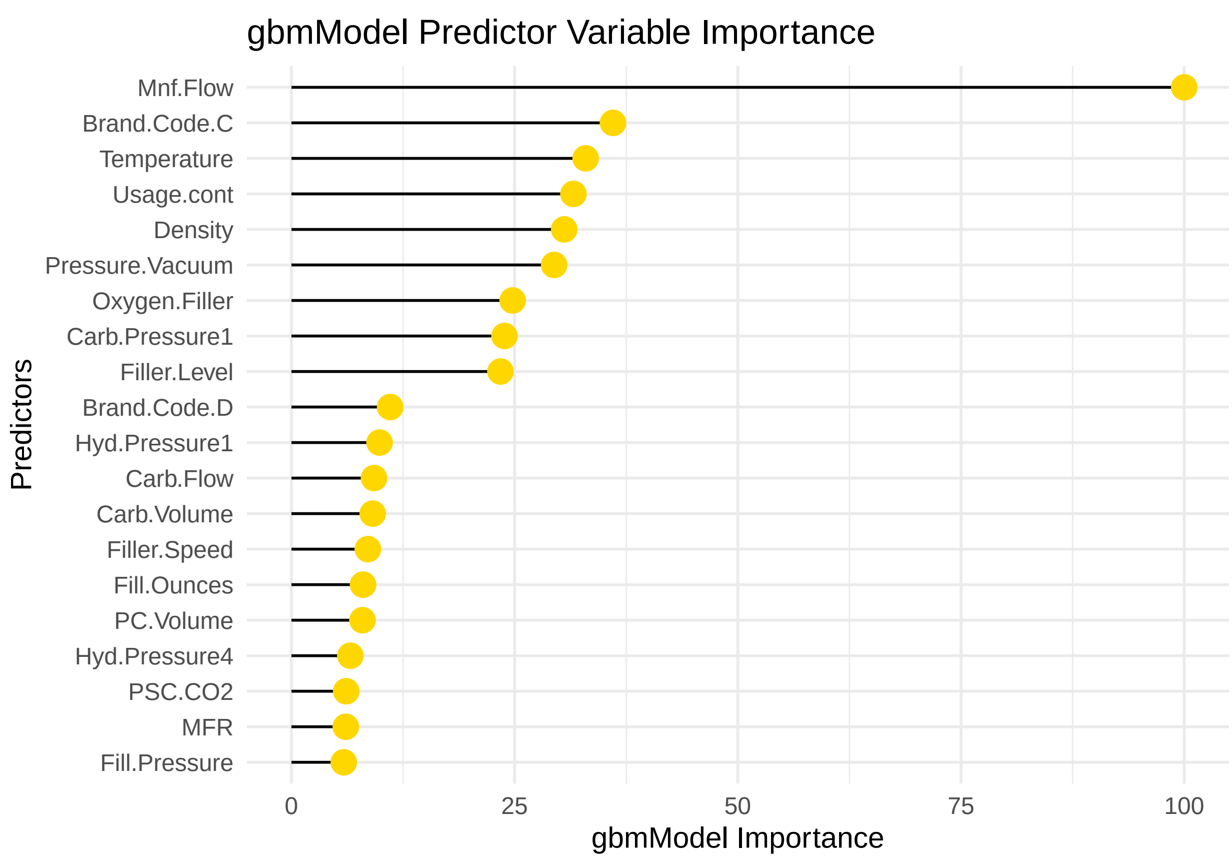
Gradient Boost Model - gbmModel

```
gbmModel_20 <- varImp(gbmModel)
gbmModel_20 <- gbmModel_20$importance %>%
  arrange(desc(Overall))
gbmModel_20 <-   head(gbmModel_20,20)
gbmModel_20
```

```
##                    Overall
## Mnf.Flow        100.000000
## Brand.Code.C     36.019765
## Temperature      32.956735
## Usage.cont       31.591014
## Density          30.556738
## Pressure.Vacuum  29.429231
## Oxygen.Filler    24.778791
## Carb.Pressure1   23.867268
## Filler.Level     23.416310
## Brand.Code.D     11.060365
```

```
## Hyd.Pressure1      9.877835
## Carb.Flow          9.255747
## Carb.Volume        9.106596
## Filler.Speed       8.565784
## Fill.Ounces        8.028047
## PC.Volume          7.962481
## Hyd.Pressure4      6.607107
## PSC.CO2            6.136717
## MFR                6.095023
## Fill.Pressure      5.866212
```

```
gbmModel_20_df<- convert_top_20_to_df(gbmModel_20)

gbmModel_20_df %>%
            arrange(Overall)%>%
            mutate(name = factor(Predictors, levels=c(Predictors))) %>%
            ggplot(aes(x=name, y=Overall)) +
            geom_segment(aes(xend = Predictors, yend = 0)) +
            geom_point(size = 4, color = "gold") +
            theme_minimal() +
            coord_flip() +
            labs(title="gbmModel Predictor Variable Importance",
               y="gbmModel Importance", x="Predictors") +
            scale_y_continuous()
```




```
gbmModelPred <- predict(gbmModel, newdata=x_test)
gbmModelPred<- postResample(pred = gbmModelPred, obs = y_test$PH)
```

```
gbmModelPred
```

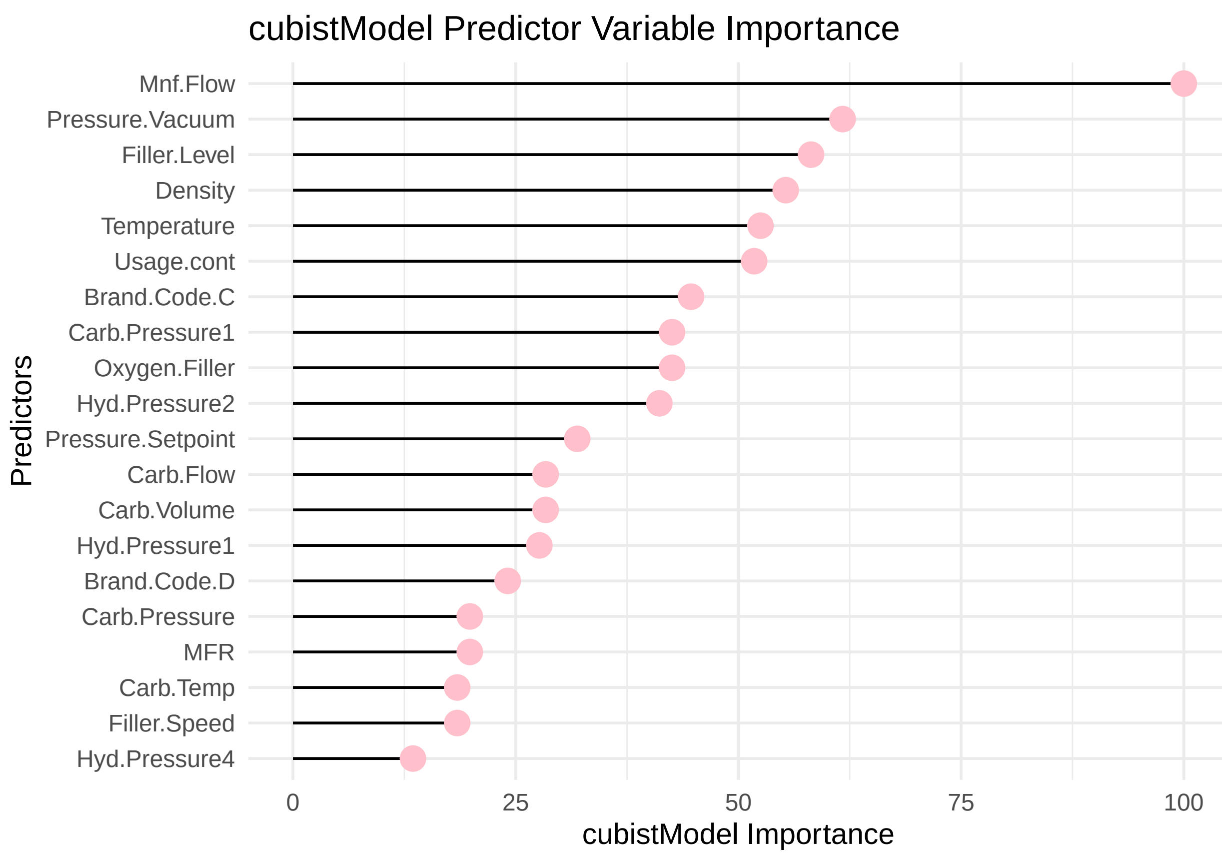
```
##      RMSE  Rsquared       MAE
## 0.1100712 0.5687185 0.0845711
```

Cubist Model - cubistModel

```
cubistModel_20 <- varImp(cubistModel)
cubistModel_20 <- cubistModel_20$importance %>%
  arrange(desc(Overall))
cubistModel_20 <-   head(cubistModel_20,20)
cubistModel_20
```

```
##                      Overall
## Mnf.Flow           100.00000
## Pressure.Vacuum     61.70213
## Filler.Level        58.15603
## Density             55.31915
## Temperature         52.48227
## Usage.cont          51.77305
## Brand.Code.C        44.68085
## Oxygen.Filler       42.55319
## Carb.Pressure1      42.55319
## Hyd.Pressure2       41.13475
## Pressure.Setpoint   31.91489
## Carb.Volume         28.36879
## Carb.Flow           28.36879
## Hyd.Pressure1       27.65957
## Brand.Code.D        24.11348
## MFR                 19.85816
## Carb.Pressure       19.85816
## Filler.Speed        18.43972
## Carb.Temp           18.43972
## Hyd.Pressure4       13.47518
```

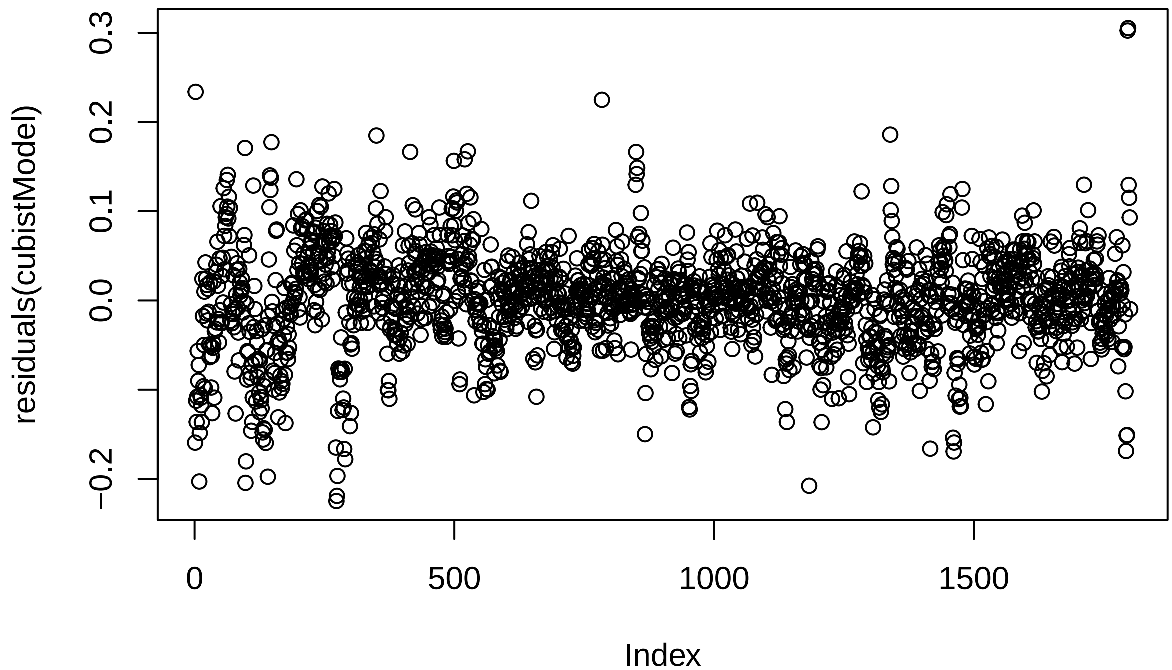
```
cubistModel_20_df<- convert_top_20_to_df(cubistModel_20)

cubistVisualMostImportant <- cubistModel_20_df %>%
                                arrange(Overall)%>%
                                mutate(name = factor(Predictors, levels=c(Predictors))) %>%
                                ggplot(aes(x=name, y=Overall)) +
                                geom_segment(aes(xend = Predictors, yend = 0)) +
                                geom_point(size = 4, color = "pink") +
                                theme_minimal() +
                                coord_flip() +
                                labs(title="cubistModel Predictor Variable Importance",
                                   y="cubistModel Importance", x="Predictors") +
                                scale_y_continuous()

cubistVisualMostImportant
```




```
cubistModelPred <- predict(cubistModel, newdata=x_test)
cubistModelPred<- postResample(pred = cubistModelPred, obs = y_test$PH)
cubistModelPred
```

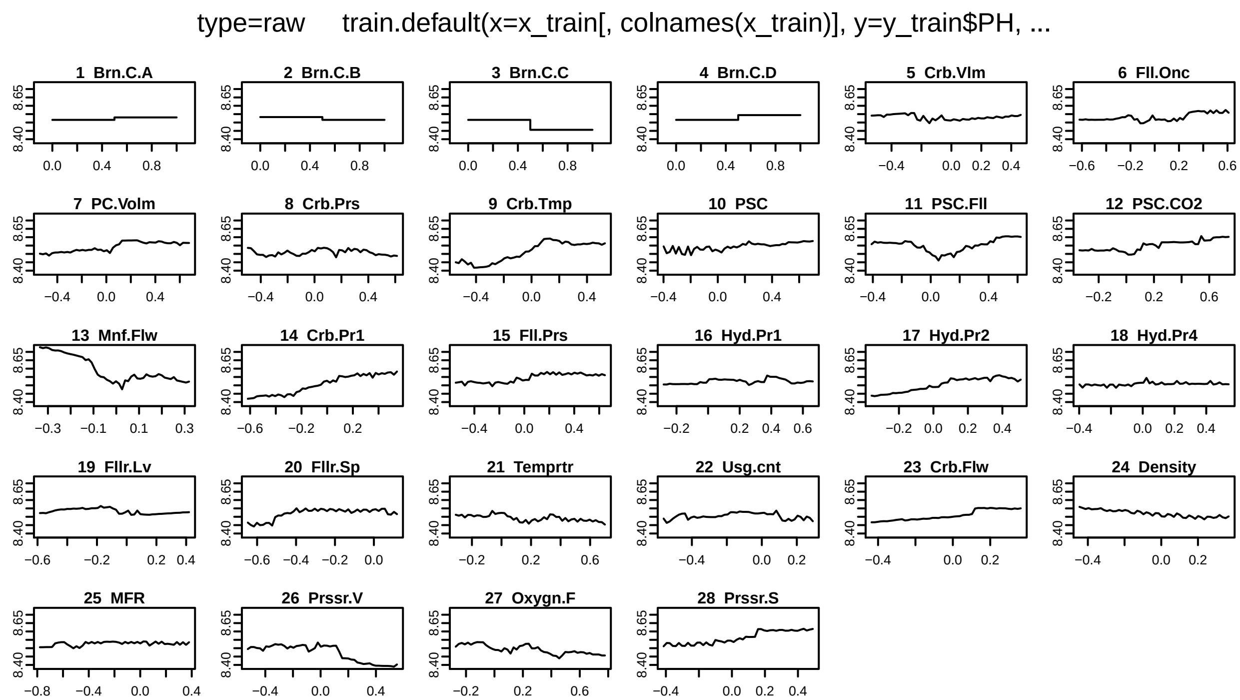
```
##       RMSE   Rsquared        MAE
## 0.10405204 0.61316724 0.07788941
```

```
plot(residuals(cubistModel))
```




```r
plotmo(cubistModel)
```

```
##  plotmo grid:    Brand.Code.A Brand.Code.B Brand.Code.C Brand.Code.D
##                             0            1            0            0
##  Carb.Volume Fill.Ounces   PC.Volume Carb.Pressure    Carb.Temp         PSC
##  -0.04645293 0.008689866 -0.01710397 -0.0007182825 -0.008149797 -0.02915067
##     PSC.Fill     PSC.CO2    Mnf.Flow Carb.Pressure1 Fill.Pressure Hyd.Pressure1
##  -0.02981358 -0.06377821 -0.02013707     0.02018178   -0.07738622   -0.01698975
##  Hyd.Pressure2 Hyd.Pressure4 Filler.Level Filler.Speed Temperature Usage.cont
##     0.07755498  -0.004587089   0.09206058     0.067918 -0.03273905 0.05305285
##   Carb.Flow    Density        MFR Pressure.Vacuum Oxygen.Filler
##  0.09796908 -0.1026698 0.06451337     0.004038687   -0.05295248
##  Pressure.Setpoint
##         -0.1154038
```



PART 4: EVALUATE MODELS

From our experimentation with 12 different models, we saw that the Cubist model had the lowest RMSE (0.10976) value as well as the lowest MAE value (0.081). It also had the highest Rsquared value (0.601).

```
knitr::kable(results_table,"markdown")
```

Model	RMSE	Rsquared	MAE
Cubist Model	0.10405204167909	0.613167241012503	0.0778894123721432
Random Forest Model	0.105166319325431	0.62244111001474	0.079384956277055
Gradient Boost Model	0.110071205384293	0.568718502340439	0.0845711016073715
baggedTree Model	0.114993612273039	0.568097114402257	0.0838154001826759
cTree Model	0.125549770014922	0.437775237075806	0.0983658534601855
Multivariate Adaptive Regression Spline	0.127970181233293	0.414325333244335	0.0992737056991776
Support Vector Machines - Linear	0.131061700205341	0.390700323584282	0.100271089395431
KNN	0.131976125996048	0.433230137065301	0.101614715138676
Ridge Regression	0.138242715968358	0.375695443714218	0.107517013724657
Partial Least Square	0.138897130894034	0.368136291804716	0.108092456297676
Linear Model	0.138964136852585	0.367775025118013	0.107864357687297
Neural Network	7.54720108092027	NA	7.54535064935065

PART 5: USE THE BEST MODEL TO FORECAST PH

We will use the Cubist model against the Student evaluation data and make predictions of the PH variable.

First, as we did with the Student train data, we have to convert the Brand.Code categorical value in the

Student evaluation data to Dummy variables.

```
mod2<- dummyVars(~Brand.Code,
          data=predictors_evaluate,
          levelsOnly = FALSE)
mod2
```

```
## Dummy Variable Object
##
## Formula: ~Brand.Code
## 1 variables, 0 factors
## Variables and levels will be separated by '.'
## A less than full rank encoding is used
```

```
dummies2 <- as.data.frame(predict(mod, predictors_evaluate))
predictors_evaluate2 <- subset(predictors_evaluate, select = -c(Brand.Code))
predictors_evaluate2 <- cbind(dummies2,predictors_evaluate)
```

```
cubistPred <- round(predict(cubistModel, newdata=predictors_evaluate2),2)
head_predictions <- head(cubistPred,10)
```

x
9.24
9.21
9.17
9.46
9.19
9.22
9.21
9.32
9.24
9.24

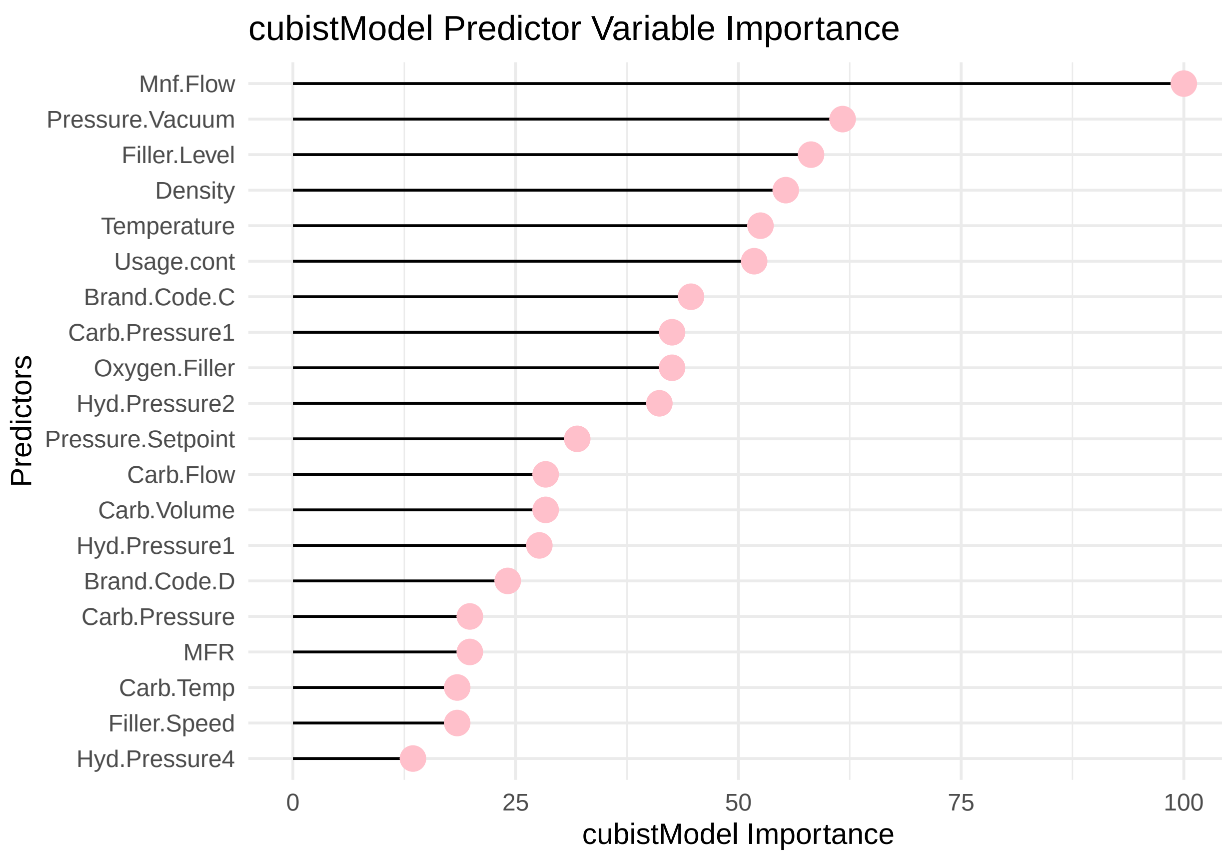
```
exported_predictions <- cbind(cubistPred,predictors_evaluate)
names(exported_predictions)[1] <- "Predicted PH"
```

PART 6: CONCLUSIONS

The data science team found that the Cubist model is the best for predicting the PH value. The most important predictors from this model are shown in the visualization below. The top five predictors are Mnf.Flow, Density, Temperature, Pressure.Vacuum, and Filler Level. Two discrete categorical factors, Brand Codes C and D, are also in the most important predictors.

We have exported the predicted PH values in the attached excel file.

```
cubistVisualMostImportant
```


cubistModel Predictor Variable Importance
Mnf.Flow
Pressure.Vacuum
Filler.Level
Density
Temperature
Usage.cont
Brand.Code.C
Carb.Pressure1
Oxygen.Filler
Hyd.Pressure2
Pressure.Setpoint
Carb.Flow
Carb.Volume
Hyd.Pressure1
Brand.Code.D
Carb.Pressure
MFR
Carb.Temp
Filler.Speed
Hyd.Pressure4
Predictors
0
25
50
75
100
cubistModel Importance