Data 624 Project 2

Zach Herold, Anthony Pagan, Betsy Rosalen

5/10/2020

Project Description

The data science team at ABC Beverage has been asked to provide an analysis of our manufacturing process, the predictive factors, and a predictive model of PH in order to comply with new regulations. This report details the steps taken in our analysis, including the assumptions made, the methodology used, the models tested, the model selected and the selection process, and the findings and conclusions reached from our analysis.

Data Description

We were given a dataset that consisted of 31 numerical predictor variables detailing a wide range of production processes, 1 categorical variable `Brand.Code`, and our target variable, `PH`. Summary statistics for these variables can be seen in the two tables below.

Table 1: Summary statistics for numerical variables

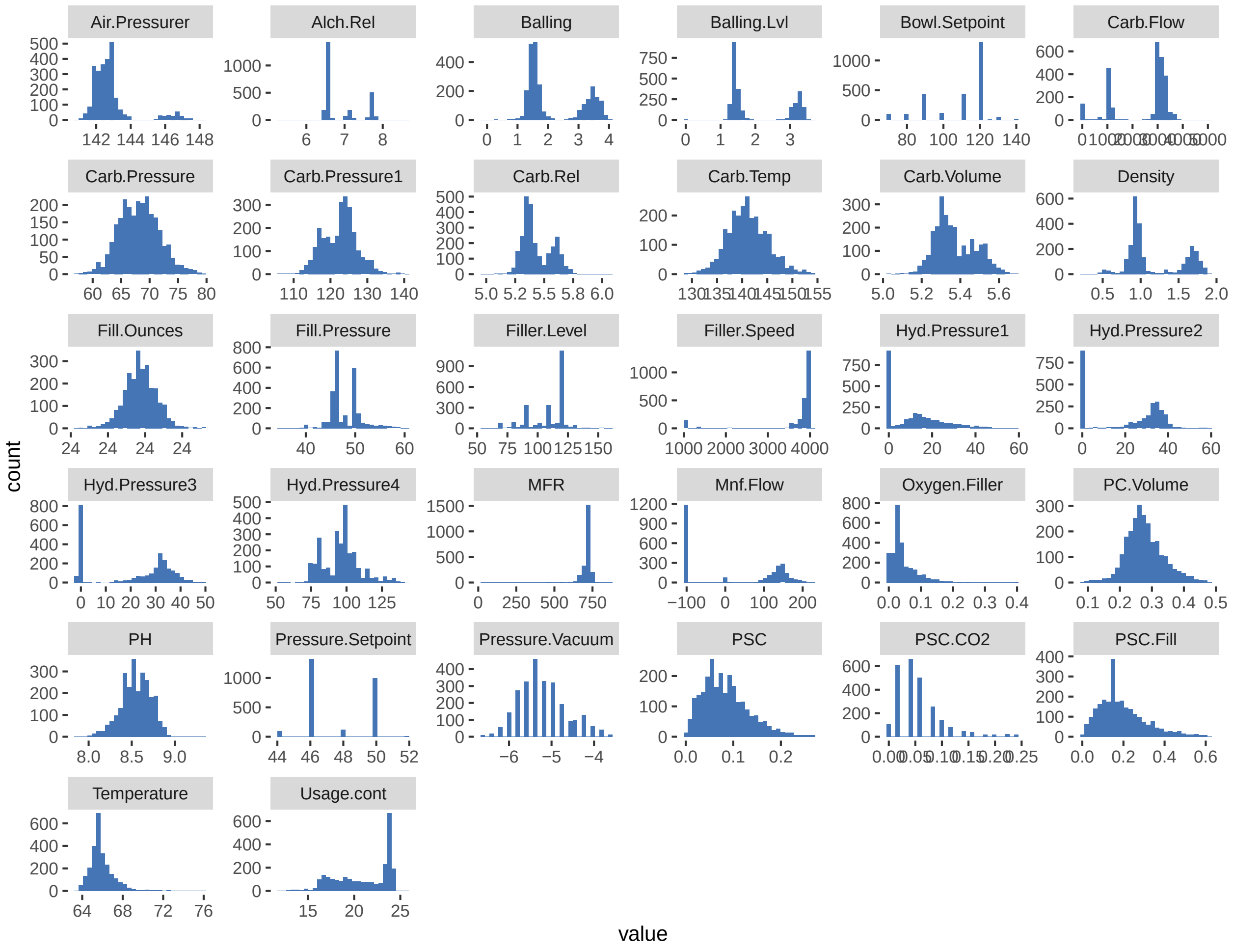
	n	mean	sd	min	median	max	range	skew	kurtosis	se
PH	2567	8.55	0.17	7.88	8.54	9.36	1.48	-0.29	0.06	0.00
Carb.Volume	2561	5.37	0.11	5.04	5.35	5.70	0.66	0.39	-0.47	0.00
Fill.Ounces	2533	23.97	0.09	23.63	23.97	24.32	0.69	-0.02	0.86	0.00
PC.Volume	2532	0.28	0.06	0.08	0.27	0.48	0.40	0.34	0.67	0.00
Carb.Pressure	2544	68.19	3.54	57.00	68.20	79.40	22.40	0.18	-0.01	0.07
Carb.Temp	2545	141.09	4.04	128.60	140.80	154.00	25.40	0.25	0.24	0.08
PSC	2538	0.08	0.05	0.00	0.08	0.27	0.27	0.85	0.65	0.00
PSC.Fill	2548	0.20	0.12	0.00	0.18	0.62	0.62	0.93	0.77	0.00
PSC.CO2	2532	0.06	0.04	0.00	0.04	0.24	0.24	1.73	3.73	0.00
Mnf.Flow	2569	24.57	119.48	-100.20	65.20	229.40	329.60	0.00	-1.87	2.36
Carb.Pressure1	2539	122.59	4.74	105.60	123.20	140.20	34.60	0.05	0.14	0.09
Fill.Pressure	2549	47.92	3.18	34.60	46.40	60.40	25.80	0.55	1.41	0.06
Hyd.Pressure1	2560	12.44	12.43	-0.80	11.40	58.00	58.80	0.78	-0.14	0.25
Hyd.Pressure2	2556	20.96	16.39	0.00	28.60	59.40	59.40	-0.30	-1.56	0.32
Hyd.Pressure3	2556	20.46	15.98	-1.20	27.60	50.00	51.20	-0.32	-1.57	0.32
Hyd.Pressure4	2541	96.29	13.12	52.00	96.00	142.00	90.00	0.55	0.63	0.26
Filler.Level	2551	109.25	15.70	55.80	118.40	161.20	105.40	-0.85	0.05	0.31
Filler.Speed	2514	3687.20	770.82	998.00	3982.00	4030.00	3032.00	-2.87	6.71	15.37
Temperature	2557	65.97	1.38	63.60	65.60	76.20	12.60	2.39	10.16	0.03
Usage.cont	2566	20.99	2.98	12.08	21.79	25.90	13.82	-0.54	-1.02	0.06
Carb.Flow	2569	2468.35	1073.70	26.00	3028.00	5104.00	5078.00	-0.99	-0.58	21.18
Density	2570	1.17	0.38	0.24	0.98	1.92	1.68	0.53	-1.20	0.01
MFR	2359	704.05	73.90	31.40	724.00	868.60	837.20	-5.09	30.46	1.52
Balling	2570	2.20	0.93	-0.17	1.65	4.01	4.18	0.59	-1.39	0.02
Pressure.Vacuum	2571	-5.22	0.57	-6.60	-5.40	-3.60	3.00	0.53	-0.03	0.01
Oxygen.Filler	2559	0.05	0.05	0.00	0.03	0.40	0.40	2.66	11.09	0.00
Bowl.Setpoint	2569	109.33	15.30	70.00	120.00	140.00	70.00	-0.97	-0.06	0.30
Pressure.Setpoint	2559	47.62	2.04	44.00	46.00	52.00	8.00	0.20	-1.60	0.04
Air.Pressurer	2571	142.83	1.21	140.80	142.60	148.20	7.40	2.25	4.73	0.02
Alch.Rel	2562	6.90	0.51	5.28	6.56	8.62	3.34	0.88	-0.85	0.01
Carb.Rel	2561	5.44	0.13	4.96	5.40	6.06	1.10	0.50	-0.29	0.00
Balling.Lvl	2570	2.05	0.87	0.00	1.48	3.66	3.66	0.59	-1.49	0.02

Table 2: Summary of categorical variable, Brand.Code

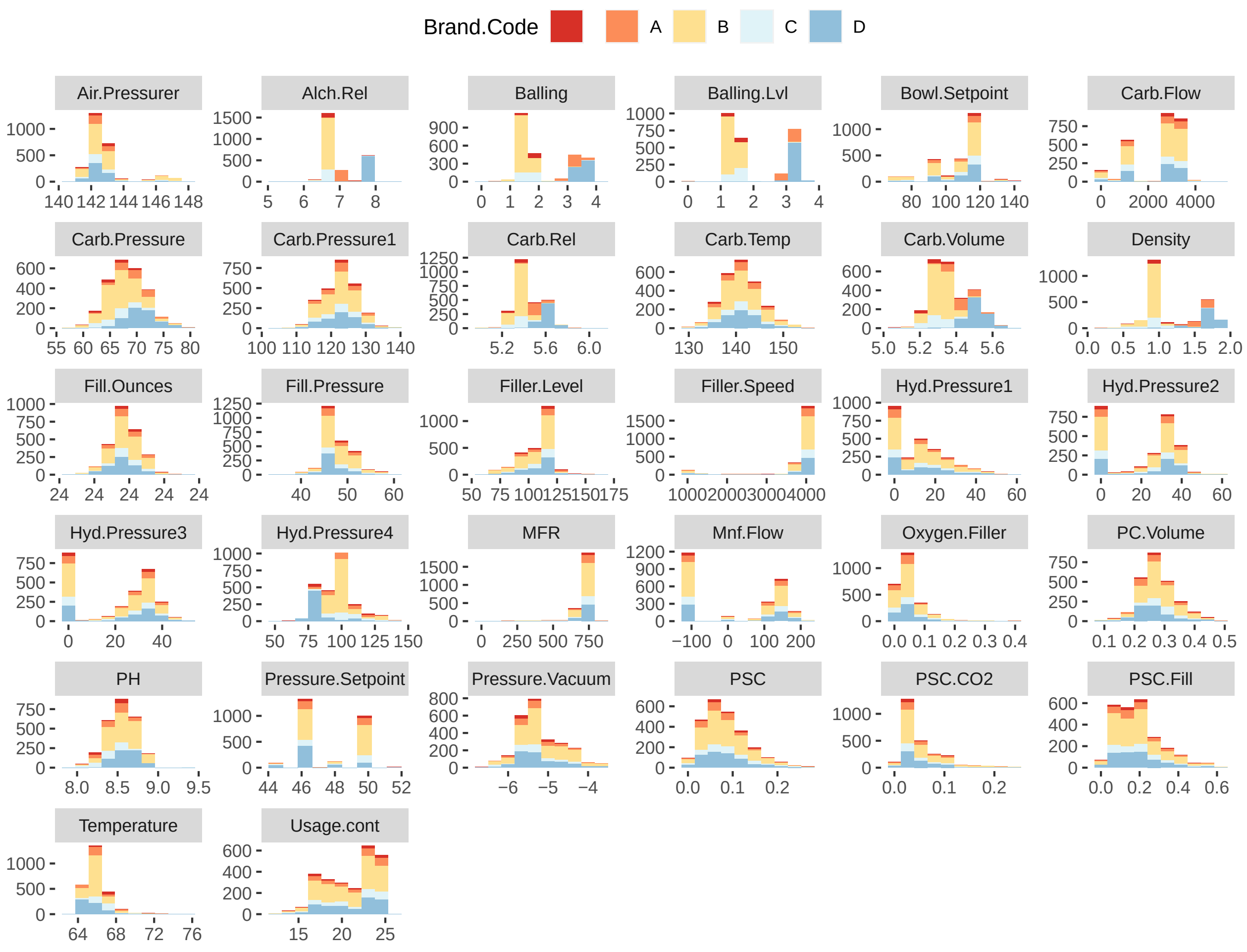
Brand.Code
: 120
A: 293
B:1239
C: 304
D: 615

Distributions

Our predictors have a wide range of distributions with some normal, some skewed, some bi-modal, and some with high zero inflation. Standardization and normalization were used for model building, the specifics of which will be described for each model in the “Models” section of the report below. Our target, `PH`, has a mostly normal distribution.

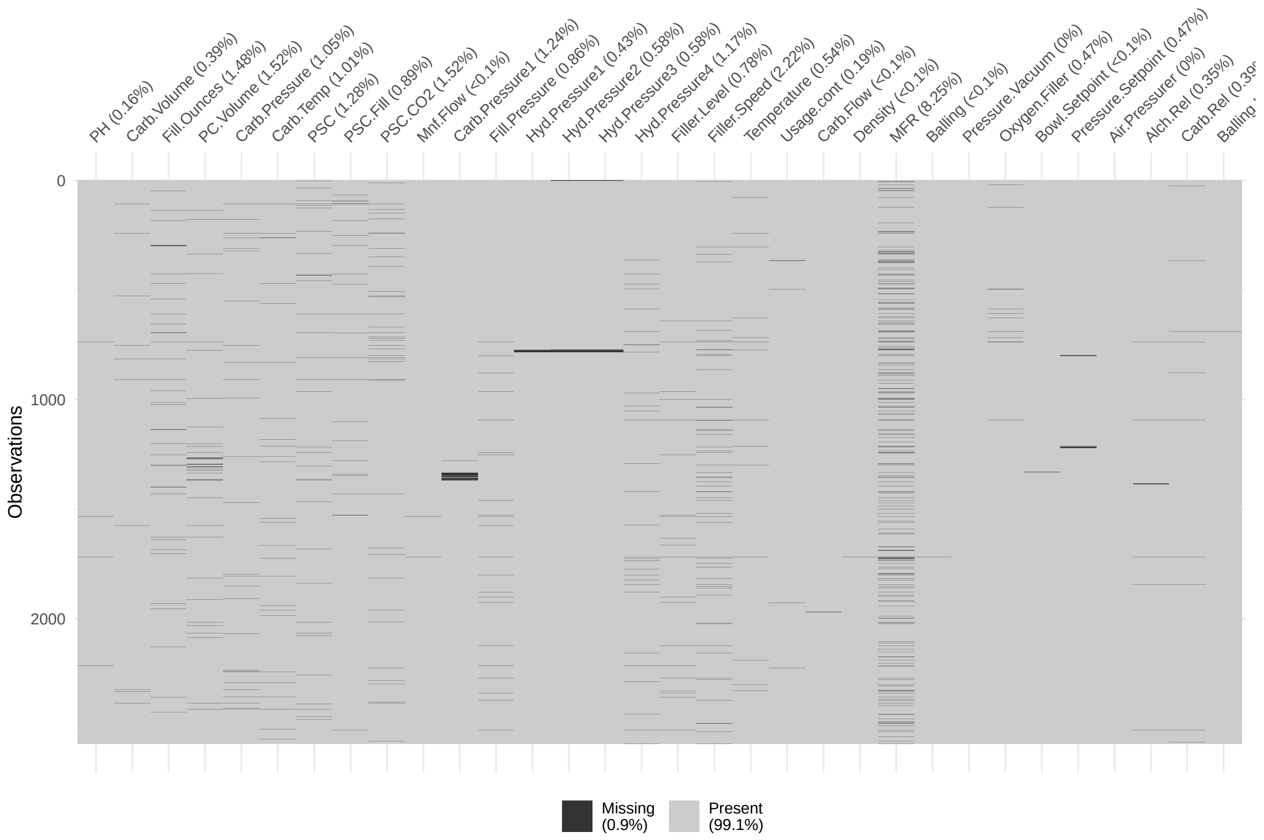


Since we only have one categorical variable, `Brand.Code`, which is related to brand and marketing rather than manufacturing process, we plotted it against each predictor and the target to see if there were any noticeable patterns that may be relevant to our model. As you can see in the plots below the brand code is evenly distributed among most predictors and most importantly evenly distributed in our target variable, `PH`, so it does not appear to have any predictive value and so was removed from our dataset for model training.

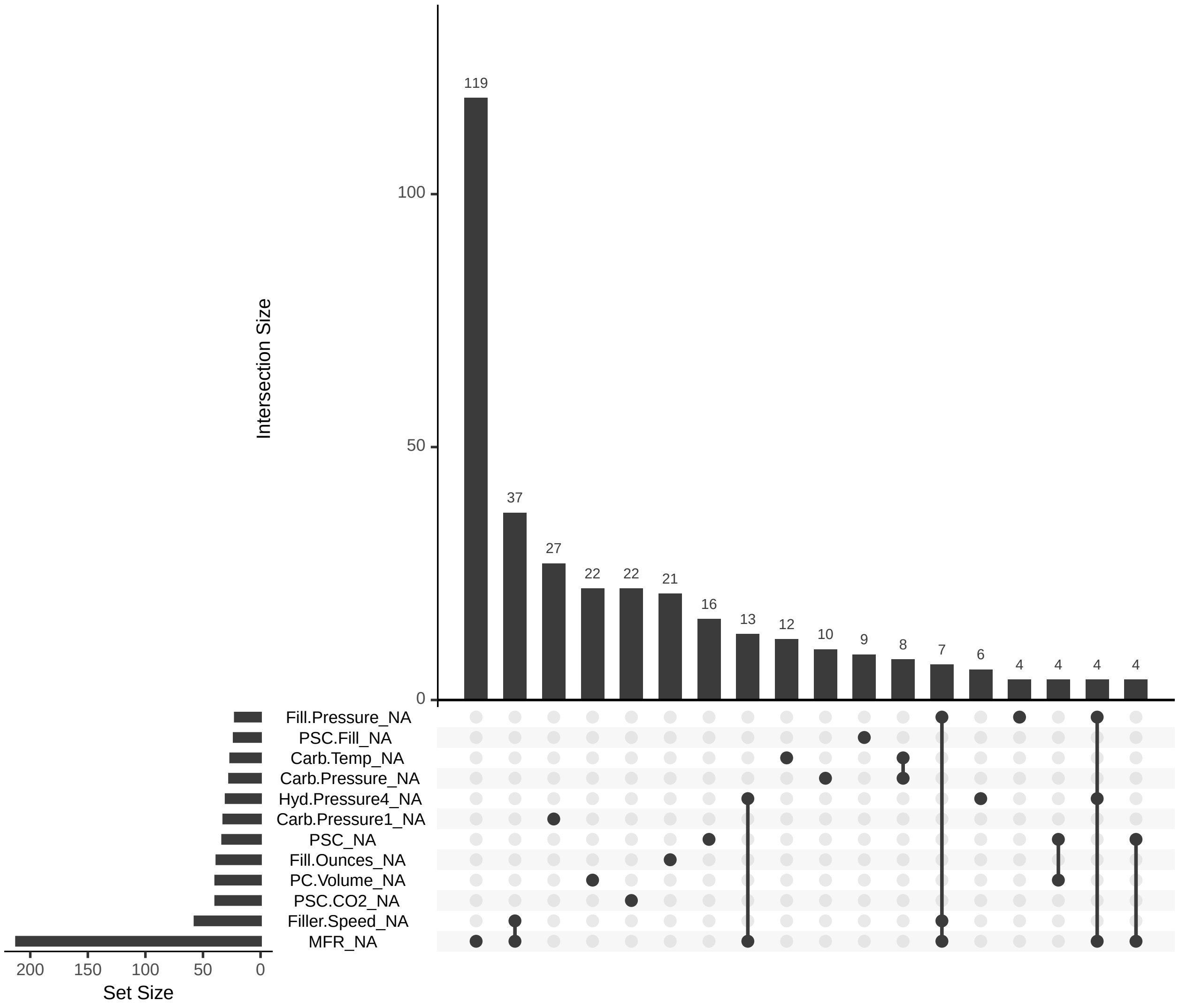


Missing Values

There was some missing data in our predictors most noticeably in `MFR`, which had 8.25% missing values as can be seen in the plot below. There doesn't seem to be any pattern in the missingness however, so it is unlikely that it has any predictive value.

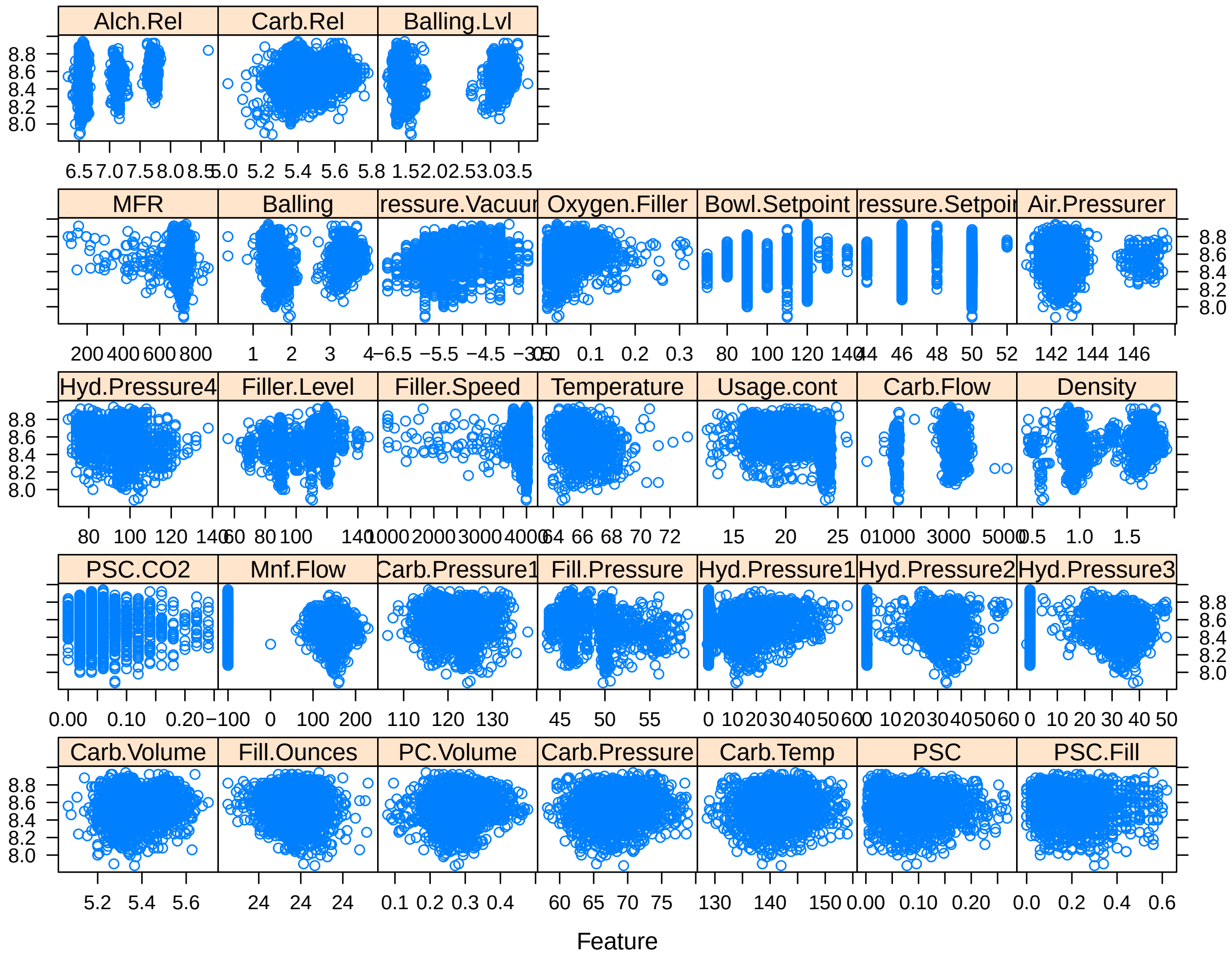


The plot below confirms that there is no apparent pattern in the missing values. Incomplete cases comprise only about 17% of the substantial dataset, and since there did not seem to be any patterns to the missingness, the decision was made to remove them completely from our dataset leaving us with the remaining 2129 complete cases.



Relationships Between Variables

The plots below were used to assess if there were any clear linear relationships between the predictors and the target, `PH`. Few, if any, relationships are immediately apparent.



A correlation plot shows some strong correlations between predictors. The `findCorrelation` function from the `caret` library recommends removing the `MFR`, `Hyd.Pressure2`, `Carb.Rel`, `Air.Pressurer`, `Carb.Flow`, `Hyd.Pressure4`, and `Filler.Level` variables at a 0.85 correlation cutoff. Upping the cutoff to 0.9 only removes one variable, `Carb.Flow`, from that list.

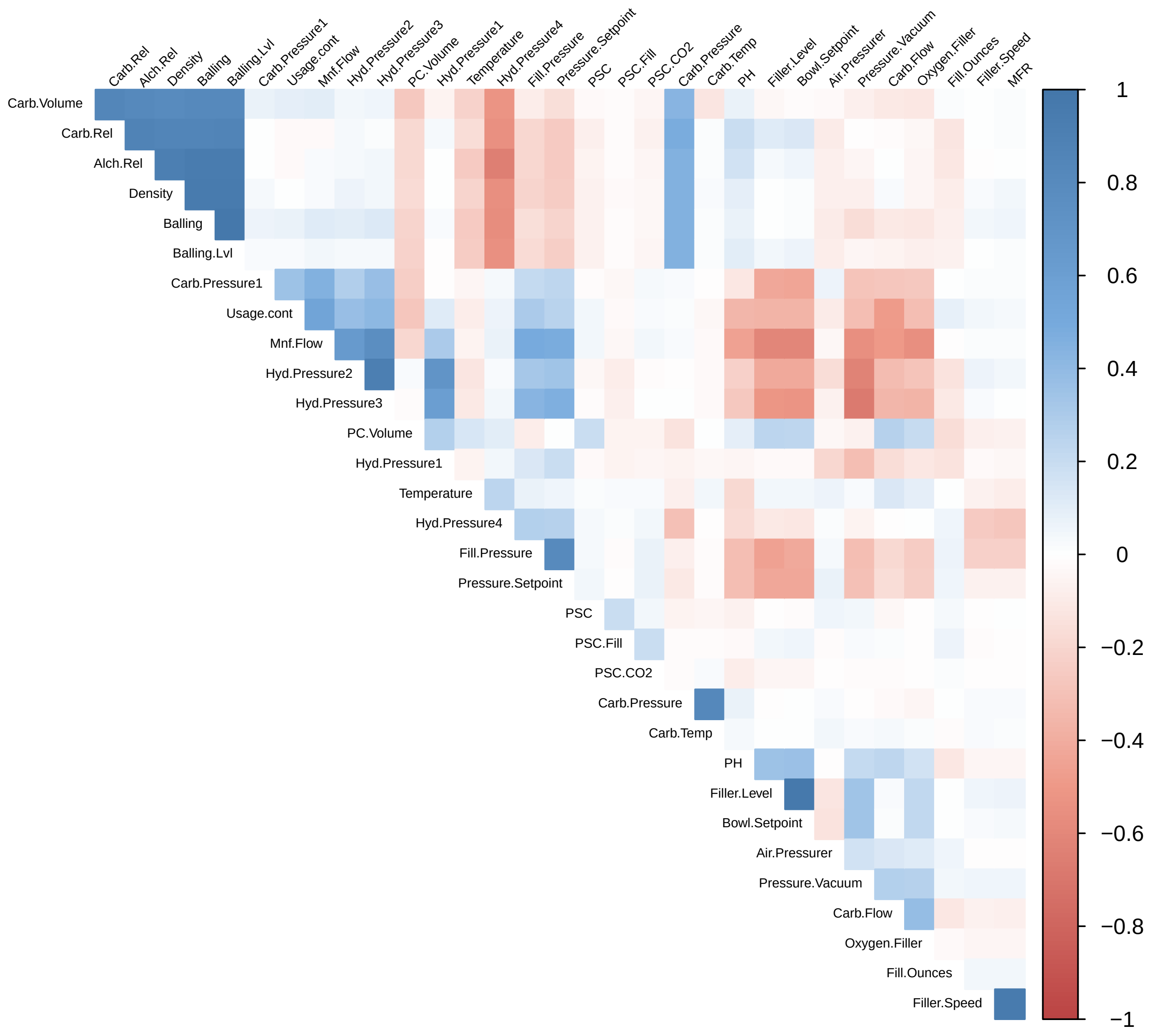


Table 3: Variables recommended for removal by caret::findCorrelation function at the 0.85 cutoff

x
MFR
Hyd.Pressure2
Carb.Rel
Air.Pressurer
Carb.Flow
Hyd.Pressure4
Filler.Level

Before we removed any variables we decided to do our own analysis of the highly correlated variables to see if we came up with the same conclusions as the `findCorrelation` function. First we found all pairs of variables that had a 0.85 correlation or more, which was 13 pairs. Then we found the frequency of each variable that was a member of one or more of these pairs. The results can been seen in the two tables below.

Table 4: Highly Correlated Variable Pairs

Var1	Var2	Correlation
Balling	Balling.Lvl	0.99
Filler.Level	Bowl.Setpoint	0.98
Density	Balling.Lvl	0.96
Density	Balling	0.95
Filler.Speed	MFR	0.95
Alch.Rel	Balling.Lvl	0.94
Balling	Alch.Rel	0.94
Hyd.Pressure2	Hyd.Pressure3	0.92
Density	Alch.Rel	0.92
Alch.Rel	Carb.Rel	0.88
Carb.Rel	Balling.Lvl	0.87
Balling	Carb.Rel	0.85
Density	Carb.Rel	0.85

Table 5: Frequency of Variables Involved in Highly Correlated Pairs

vars	Freq
Alch.Rel	4
Balling	4
Balling.Lvl	4
Carb.Rel	4
Density	4
Bowl.Setpoint	1
Filler.Level	1
Filler.Speed	1
Hyd.Pressure2	1
Hyd.Pressure3	1
MFR	1

What we found is that there were exactly 5 variables that were most frequently associated with highly correlated pairs. Each of these variables, `Alch.Rel`, `Balling`, `Balling.Lvl`, `Carb.Rel`, `Density` was involved in 4 pairs. None of the other variables were involved in more than one pair. By removing `Balling.Lvl`, we could eliminate 4 of our highly correlated pairs, by removing `Density` we got rid of 3 more, and with `Balling` 2 more. so we were left with only the following 4 pairs of highly correlated variables.

Table 6: Highly Correlated Variable Pairs

Var1	Var2	Correlation
Filler.Level	Bowl.Setpoint	0.98
Filler.Speed	MFR	0.95
Hyd.Pressure2	Hyd.Pressure3	0.92
Alch.Rel	Carb.Rel	0.88

Each of the variables in these pairs only appears once so we needed to get rid of one variable from each pair in order to eliminate all pairs of variables with a 0.85 correlation or more. So we decide to remove the 4 with the lowest correlation to `PH` without removing two from the same pair. This eliminated `Filler.Speed`, `Alch.Rel`, `Hyd.Pressure2` and `Filler.Level`.

So in the end we still removed 7 variables but not all the same ones recommended by the `findCorrelation` function. Only 2 of the variables recommended by the function matched our list, `Hyd.Pressure2` and `Filler.Level`.

Models

Next we partitioned our dataset into training and validation subsets by randomly selecting 70% for training and leaving the remaining 30% set aside for testing.

We then tuned a full range of model types including: Linear Regression, Ridge Regression, Lasso, Random Forest, Tree Bag, CTree, Classification and Regression Tree (CART), Multivariate Adaptive Regression Splines (MARS), K-Nearest Neighbors (KNN) and Support Vector Machine (SVM) using repeated cross-validation on all models. The RMSE, R^2, and MAE statistics for each of these models are presented in the table below, ordered by the lowest RMSE to highest and thus best predictive performance to worst.

Table 7: MODELS

	RMSE	Rsquared	MAE
Random Forest	0.120	0.559	0.088
Tree Bag	0.134	0.455	0.102
SVM	0.136	0.444	0.097
MARS	0.137	0.413	0.104
KNN	0.142	0.369	0.106
Lasso	0.145	0.349	0.111
Ridge Regression	0.145	0.348	0.111
Linear Regression	0.145	0.348	0.111
CTree	0.151	0.292	0.116
CART	0.159	0.205	0.125

Random Forest Model

The random forest model was selected for further tuning based on the lowest RMSE and MAE statistics. Although it also had the highest R^2 value that statistic should only be used to compare performance between variously tuned models of the same type, not between models of different types, so it's relevance is not significant in this case.

Top Ten Variables in the Initial Random Forest Model by Importance Score

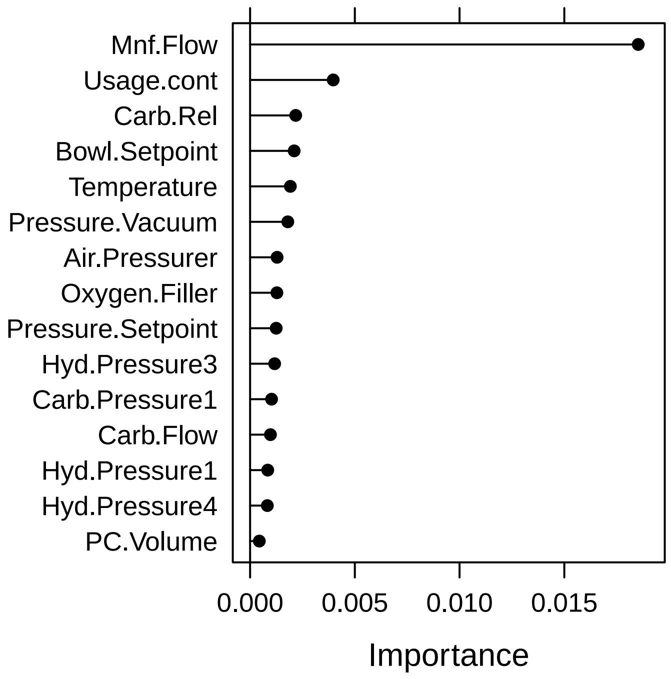
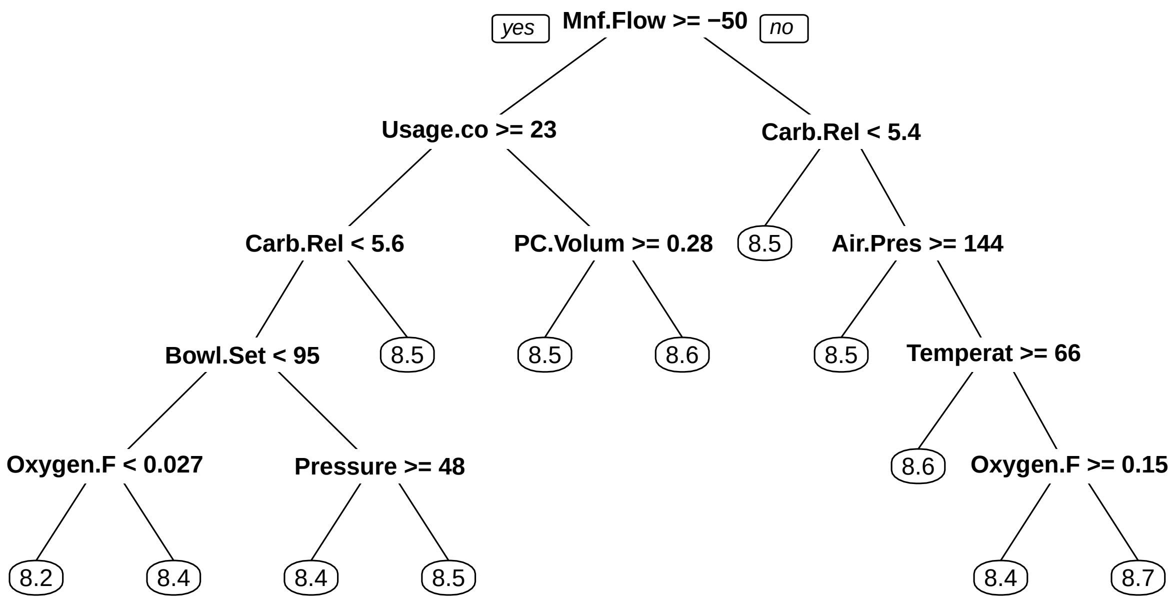


Table 8: Variable Importance Scores

	Overall
Mnf.Flow	0.01853
Usage.cont	0.00397
Carb.Rel	0.00218
Bowl.Setpoint	0.00210
Temperature	0.00192
Pressure.Vacuum	0.00180
Air.Pressurer	0.00129
Oxygen.Filler	0.00128
Pressure.Setpoint	0.00125
Hyd.Pressure3	0.00117

For comparison we also plotted a tree diagram which gave us similar results with the top three predictors also taking the top 3 nodes in the tree.



Fine Tuning the Random Forest Model

Since we had removed 7 predictors before tuning our models we decided to try re-tuning the best performing model, the random forest model, using the full set of predictors. This resulted in a small improvement in performance on the validation set as measured by the RMSE and MAE as well as the improved R^2 value as shown in the table below.

Table 9: Accuracy Measures for Random Forest Model using Full Set of Predictors

	RMSE	Rsquared	MAE
Random Forest Model All Predictors	0.11068	0.60894	0.07569

Top 10 Variables in the Random Forest Model using all Predictors by Importance Score

Table 10: Variable Importance Scores

	%IncMSE	IncNodePurity
Mnf.Flow	0.01220	6.7313
Usage.cont	0.00579	4.5364
Bowl.Setpoint	0.00538	2.7553
Temperature	0.00259	2.6779
Carb.Rel	0.00380	2.4227
Filler.Level	0.00314	2.3245
Balling.Lvl	0.00311	2.1875
Oxygen.Filler	0.00262	2.1504
Alch.Rel	0.00382	2.0364
Carb.Pressure1	0.00126	1.8438

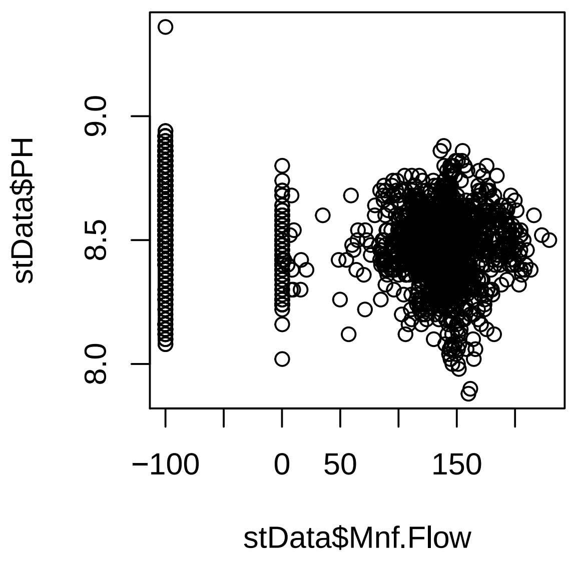
Three more linear models were tested using the top predictors from our random forest model, however, none of them resulted in any improvement in performance.

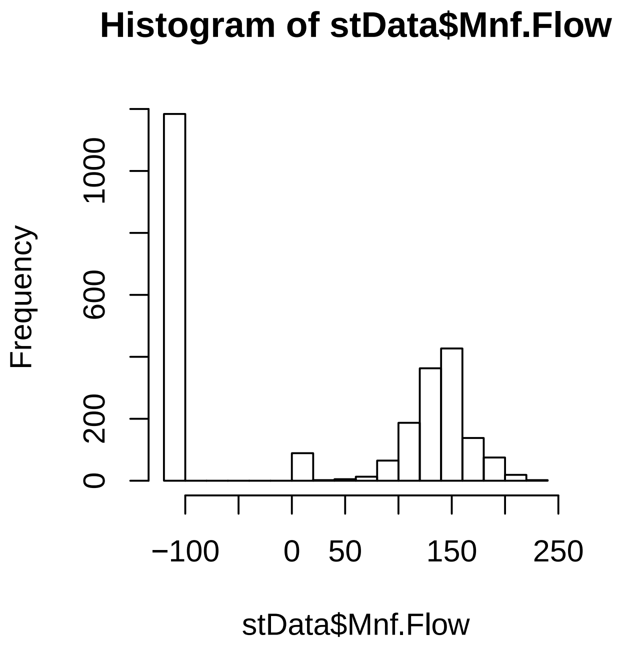
Table 11: MODELS

	RMSE	Rsquared	MAE
Linear Regression 2	0.145	0.348	0.111
Linear Regression 4	0.147	0.329	0.115
Linear Regression 3	0.147	0.328	0.115

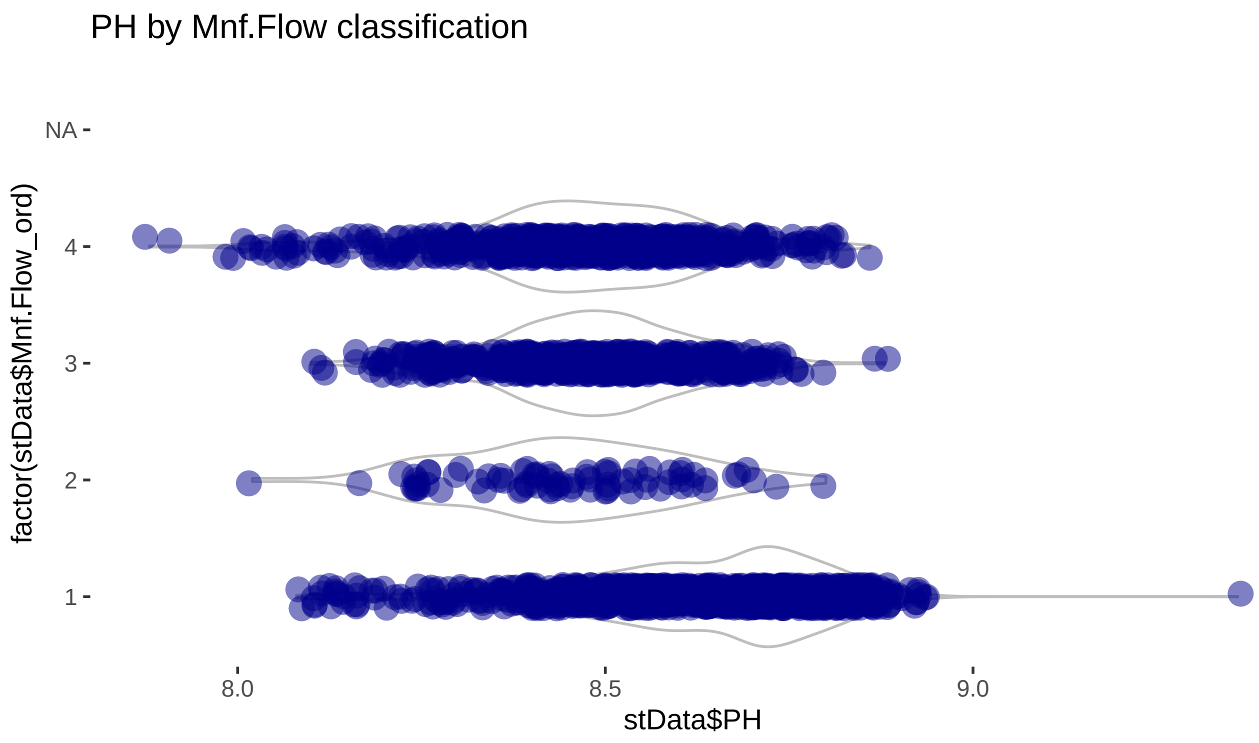
One interesting finding from this experiment was that we were able to determine that the impact of `Mnf.Flow`, `Usage.conf`, `Temperature`, `Oxygen.Filler`, and `Pressure.Setpoint` is negative due to negative coefficients and the impact of `Carb.Rel`, `Bowl.Setpoint`, `Hyd.Pressure3`, `Hyd.Pressure3` is positive due to positive coefficients. So there is a balancing act between these variables with some pulling in one direction on the pH and some in the other. Thus a change in one may necessitate a change in the others. The model coefficents can be seen in the model summary below.

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.5903 -0.0800  0.0121  0.0939  0.3733
##
## Coefficients:
##                    Estimate Std. Error t value            Pr(>|t|)
## (Intercept)        9.4968043  0.3338057   28.45 < 0.0000000000000002 ***
## Mnf.Flow          -0.0007197  0.0000614  -11.72 < 0.0000000000000002 ***
## Usage.cont        -0.0064591  0.0014898   -4.34      0.0000155215769 ***
## Carb.Rel           0.1779166  0.0305671    5.82      0.0000000071794 ***
## Bowl.Setpoint      0.0011607  0.0003121    3.72              0.00021 ***
## Temperature       -0.0243467  0.0034953   -6.97      0.0000000000049 ***
## Oxygen.Filler     -0.4484166  0.1105615   -4.06      0.0000525618345 ***
## Pressure.Setpoint -0.0064261  0.0021279   -3.02              0.00257 **
## Hyd.Pressure3      0.0022136  0.0003730    5.93      0.0000000036612 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.139 on 1481 degrees of freedom
## Multiple R-squared:  0.322,  Adjusted R-squared:  0.318
## F-statistic: 87.8 on 8 and 1481 DF,  p-value: <0.0000000000000002
```





```
## [1] 2571

## [1] 442

## [1] 0.834

## [1] 0.0123

## [1] 0.871

## [1] 0.0315

##
##    1    2    3    4
## 1184   79  637  669
```



Modeling by Brand

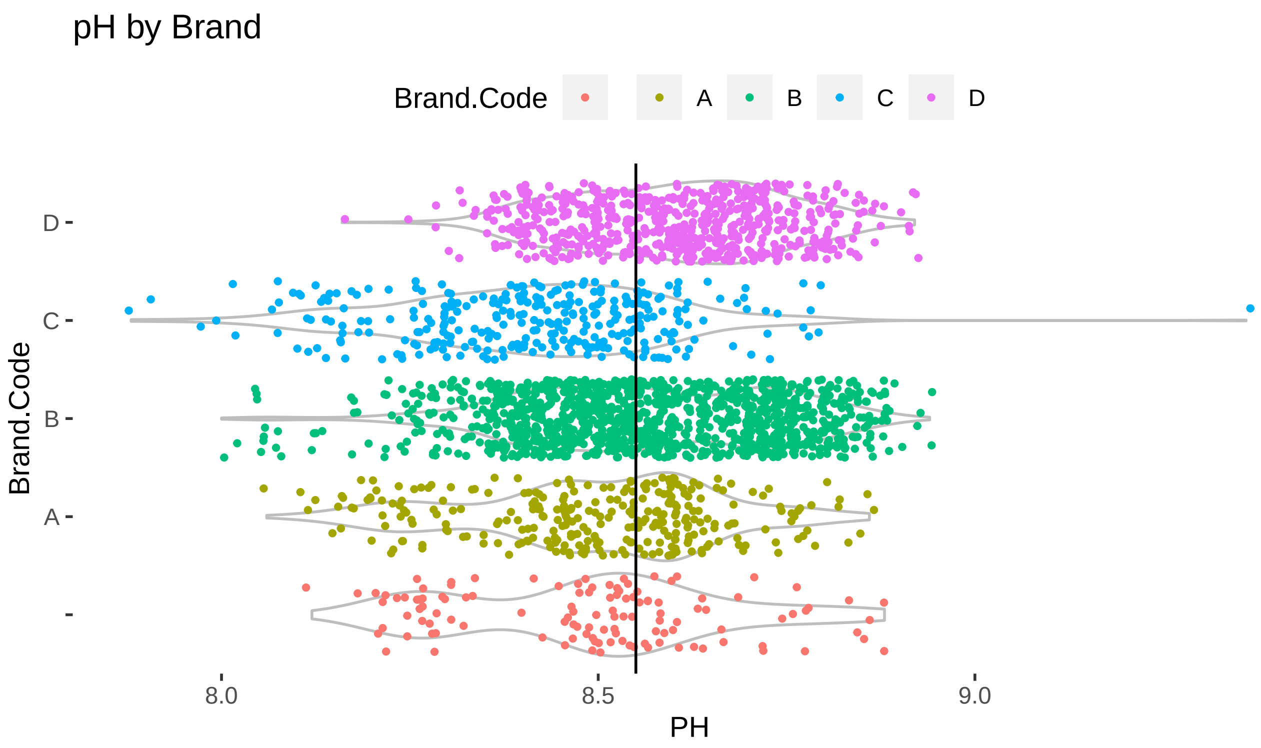
In another experiment we divided the dataset into subsets according to `Brand.Code` in order to assess what production processes are most relevant for each brand type. We imputed missing values by replacing them with the trimmed mean and then applied a random forest model to each of the four sets. Our aim was to determine if the variables found to be most important for the whole dataset carry through to the subsets.

Interestingly the random forest model performed most poorly on the brand with the highest frequency in our dataset as can be seen in the table below.

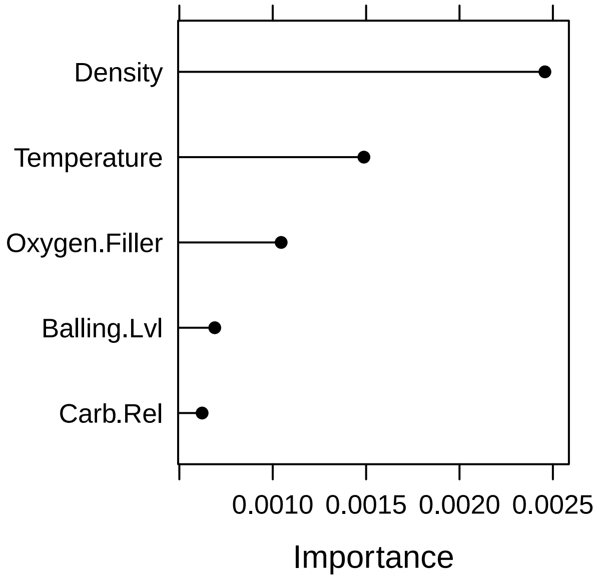
Table 12: BRANDS

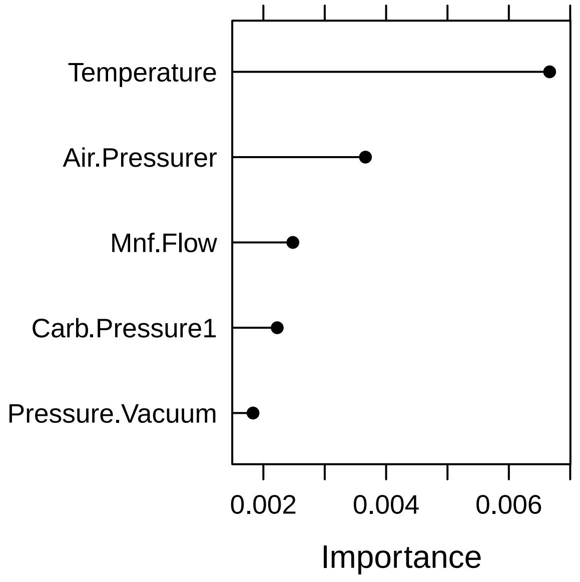
	RMSE	Rsquared	MAE	freq
C	0.166	0.166	0.133	304
A	0.173	0.123	0.140	293
D	0.178	0.053	0.139	615
B	0.211	0.023	0.168	1239

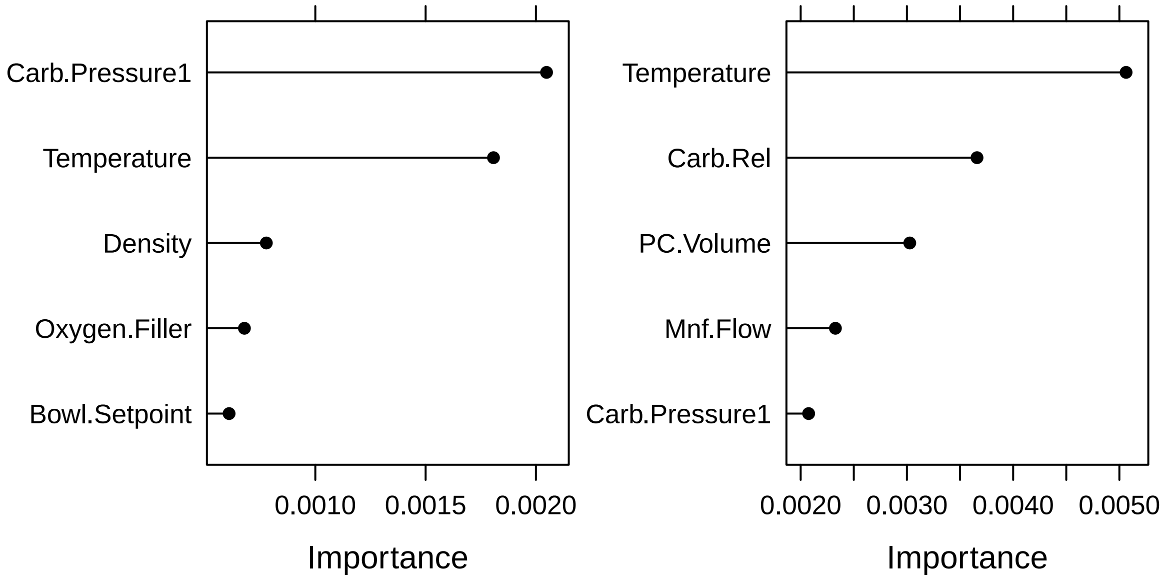
The mean pH for our datatset is 8.55, however, it is possible that pH may vary with brand profile. From this violin plot, we observe that the distribution of pH values for Brand D tends to be above mean, while that of Brand C is markedly below mean. We further investigate what factors determine the acidic signature of Brand C, with the conclusion that lower balling method levels (which promote solution alkalinity) may at least partially contribute.



We discovered that `Mnf.Flow` is no longer the most important variable at the brand level; rather, `Temperature` is, ranking in the top five for each of the four brands. By contrast `Mnf.Flow` only shows up in the top 5 list for two brands and in teh 3rd and 5th spots. These results suggest that `Mnf.Flow` may not be as robust a predictor as our other models indiciated.







Appendix

Code used in analysis

```
knitr::opts_chunk$set(echo = FALSE, message = FALSE, warning = FALSE)
options(scipen=999, digits = 2)

library(AppliedPredictiveModeling)
library(caret)
library(e1071)
library(earth)
library(faraway)
library(fpp2)
library(ggplot2)
library(gridExtra)
library(kableExtra)
require(knitr)
library(leaps)
library(lubridate)
library(MASS)
library(mlbench)
library(naniar)
library(pander)
library(pROC)
library(psych)
library(readxl)
library(reshape)
library(reshape2)
library(tidyverse)
library(tseries)
library(urca)
library(ZIM)

# Table formatting functions
```

```r
# kab_tab <- function(df, cap){
#   df %>% kable(caption=cap) %>%
#   kable_styling(bootstrap_options = c("striped", "hover", "condensed"),
#                 full_width = T)
# }
# kab_tab2 <- function(df, cap){
#   df %>% kable(caption=cap) %>%
#   kable_styling(bootstrap_options = c("striped", "hover", "condensed"),
#                 full_width = F)
# }
# For pdf output
kab_tab <- function(df, cap){
    kable(df, caption=cap, "latex", booktabs = T) %>%
        kable_styling(latex_options = c("striped", "HOLD_position", "scale_down"))
}
kab_tab2 <- function(df, cap){
    kable(df, caption=cap, "latex", booktabs = T) %>%
        kable_styling(latex_options = c("striped", "HOLD_position"))
}

#Import Data
stEval<- read.csv("https://raw.githubusercontent.com/apag101/data624Group5/master/Project2/StudentEvalua
stDatao<- read.csv("https://raw.githubusercontent.com/apag101/data624Group5/master/Project2/StudentData

# move PH to the first column of the dataframe
stData <- stDatao[,c(26,1:25,27:33)]
stEval <- stEval[,c(26,1:25,27:33)]

# https://stackoverflow.com/questions/5863097/selecting-only-numeric-columns-from-a-data-frame

num_sum <- describe(Filter(is.numeric, stData))
kab_tab(num_sum[,c(2,3,4,8,5,9,10:13)], cap="Summary statistics for numerical variables")

cat_sum <- summary(Filter(is.factor, stData))
kab_tab2(cat_sum, cap="Summary of categorical variable, Brand.Code")

stData[,-2] %>%
  gather() %>%
  ggplot(aes(value)) +
    facet_wrap(~ key, scales = "free") +
    geom_histogram(fill = '#4575b4') +
  theme(panel.background = element_blank(), legend.position="top")

stData %>%
  gather(-Brand.Code, key = "var", value = "val") %>%
  ggplot(aes(x = val, fill=Brand.Code)) +
  geom_histogram(bins=10, alpha=1) +
  facet_wrap(~ var, scales = "free") +
  scale_fill_manual("Brand.Code",
                    values = c('#d73027','#fc8d59','#fee090',
                               '#e0f3f8','#91bfdb','#4575b4')) +
  xlab("") +
  ylab("") +
  theme(panel.background = element_blank(), legend.position="top")

stData <- stData[-2]

# Missing Data
vis_miss(stData[-2])

gg_miss_upset(stData,
              nsets = 12,
              nintersects = 18)

summary(complete.cases(stData))

cstData <- subset(stData[-2], complete.cases(stData))

#plot checks
featurePlot(cstData[-1],cstData$PH)

#Correlation Matrix
library(corrplot)
cor.plt <- cor(cstData, use = "pairwise.complete.obs", method = "pearson")
col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))
corrplot(cor.plt, method="color", col=col(200),
         type="upper", order="hclust",
         tl.col="black", tl.srt=45, tl.cex=0.5,
         diag=FALSE
         )

kab_tab2(names(cstData)[findCorrelation(cor(cstData[-1]), cutoff = .85)],
        cap = "Variables recommended for removal by caret::findCorrelation function at the 0.85 cutoff"

# sb<-names(cstData)[findCorrelation(cor(cstData[-1]), cutoff = .85)]
# cstData<-cstData[, -which(names(cstData) %in% c(sb))]

#Displaying highly correlated pairs
cor.plt <- cor(cstData, use = "pairwise.complete.obs", method = "pearson")
cor.plt[lower.tri(cor.plt,diag=TRUE)] = NA #Prepare to drop duplicates and meaningless information
cor.plt <- as.data.frame(as.table(cor.plt)) #Turn into a 3-column table
cor.plt <- na.omit(cor.plt) #Get rid of the junk (NA's) we flagged above
cor.plt <- subset(cor.plt, abs(cor.plt$Freq)>0.85)
cor.plt <- cor.plt[order(-abs(cor.plt$Freq)),] #Sort by highest correlation (whether +ve or -ve)
rownames(cor.plt) <- c()
names(cor.plt)[3] <- "Correlation"
kab_tab2(cor.plt, cap="Highly Correlated Variable Pairs")

vars <- c(as.character(cor.plt$Var1), as.character(cor.plt$Var2))
vars <- as.data.frame(table(vars))
vars <- vars[order(-vars$Freq),]
rownames(vars) <- c()
kab_tab2(vars, cap="Frequency of Variables Involved in Highly Correlated Pairs")

cstData <- subset(cstData, select = -c(Balling.Lvl, Density, Balling))
cor.plt <- cor(cstData, use = "pairwise.complete.obs", method = "pearson")
cor.plt[lower.tri(cor.plt,diag=TRUE)] = NA #Prepare to drop duplicates and meaningless information
cor.plt <- as.data.frame(as.table(cor.plt)) #Turn into a 3-column table
cor.plt <- na.omit(cor.plt) #Get rid of the junk (NA's) we flagged above
cor.plt <- subset(cor.plt, abs(cor.plt$Freq)>0.85)
cor.plt <- cor.plt[order(-abs(cor.plt$Freq)),] #Sort by highest correlation (whether +ve or -ve)
rownames(cor.plt) <- c()
names(cor.plt)[3] <- "Correlation"
kab_tab2(cor.plt, cap="Highly Correlated Variable Pairs")

temp <- subset(cstData,
               select = c(Alch.Rel, Bowl.Setpoint, Carb.Rel,    Filler.Level,
                          Filler.Speed, Hyd.Pressure2, Hyd.Pressure3,   MFR, PH))
temp <- (cor(temp, use = "pairwise.complete.obs", method = "pearson")[1:8,9])
# sort(abs(temp))

cstData <- subset(cstData, select = -c(Filler.Speed, Alch.Rel, Hyd.Pressure2, Filler.Level))

#Partition Data
set.seed(123)
trainidx<-sample(nrow(cstData),round(0.7*nrow(cstData)),replace=F)
traindata<-cstData[trainidx,]
testdata<-cstData[-trainidx,]

require(caret)
set.seed(555)
trctrl<- trainControl(method="repeatedcv", number=3, repeats=2)

##Linear Regression
linreg <- caret::train(PH~., data=traindata, method="lm",
                trControl=trctrl)
linPred <- predict(linreg, newdata = testdata)
m1<-data.frame(postResample(pred = linPred, obs = testdata$PH)) #0.1414880 0.3775156

##Ridge Regression
ridge <- caret::train(PH~., data=traindata, method="ridge",
                trControl=trctrl)
ridgePred <- predict(ridge, newdata = testdata)
m2<-data.frame(postResample(pred = ridgePred, obs = testdata$PH)) #0.1414837 0.3775762

##Lasso Regression
lasso <- caret::train(PH~., data=traindata, method="lasso",
                trControl=trctrl)
lassoPred <- predict(lasso, newdata = testdata)
m3<-data.frame(postResample(pred = lassoPred, obs = testdata$PH)) #0.1418941 0.3762947

##RandomForest (Processed)
rforest <- caret::train(PH~., data=traindata, method="cforest",
                trControl=trctrl,
                tuneLength =2)
forPred <- predict(rforest, newdata = testdata)
m4<-data.frame(postResample(pred = forPred, obs = testdata$PH)) #0.11550844 0.59451507

##Tree Bag
```

```r
bag <- caret::train(PH~., data=traindata, method="treebag",
                trControl=trctrl,
                tuneLength =2)
bagPred <- predict(bag, newdata = testdata)
m5<-data.frame(postResample(pred = bagPred, obs = testdata$PH)) #0.12790436 0.50718903

##CTree
ctre <- caret::train(PH~., data=traindata, method="ctree2",
                trControl=trctrl,
                tuneLength =2)
ctrePred <- predict(ctre, newdata = testdata)
m6<-data.frame(postResample(pred = ctrePred, obs = testdata$PH)) #0.1519582 0.2804008

##CART
rcart<- caret::train(PH~., data=traindata, method="rpart",
                trControl=trctrl,
                tuneLength =2)
cartPred <- predict(rcart, newdata = testdata)
m7<-data.frame(postResample(pred = cartPred, obs = testdata$PH)) #0.1593219 0.2053639

##MARS
marsFit <- earth(PH~., data = traindata, degree=2, nprune=14)
marsPred <- predict(marsFit, newdata = testdata)
m8<-data.frame(postResample(pred = marsPred, obs = testdata$PH)) #0.1399919 0.3912855

##KNN
knnGrid <-  expand.grid(k = 1:20)
knnFit <- caret::train(PH~., data = traindata,
                method = "knn",
                trControl = trctrl,
                tuneGrid = knnGrid)
knnPred <- predict(knnFit, newdata = testdata)
m9<-data.frame(postResample(pred = knnPred, obs = testdata$PH)) #0.1278504 0.5088682

##SVM (Radial Kernel)
svmGrid <-  expand.grid(C = c(1,1000))
svmFit <- caret::train(PH~., data = traindata,
                 #type='eps-regression',
                 method = 'svmRadialCost',
                 trControl = trctrl,
                 tuneGrid = svmGrid)
svmPred <- predict(svmFit, newdata = testdata)
m10<-data.frame(postResample(pred = svmPred, obs = testdata$PH)) #0.13136218 0.48751241

df<-data.frame(rbind(m1[,1],m2[,1],m3[,1],m4[,1],m5[,1],m6[,1],m7[,1],
                     m8[,1],m9[,1],m10[,1]))
rownames(df)<-c("Linear Regression","Ridge Regression","Lasso","Random Forest",
                "Tree Bag","CTree","CART","MARS","KNN","SVM")
colnames(df)<-c("RMSE","Rsquared","MAE")
df <- df[order(df$RMSE),]
options(digits = 3)
kab_tab2(df, cap="MODELS")
```

```r
#Variable Importance Ranking (Random Forest)
rfImp <- varImp(rforest, scale = FALSE)
bookTheme()
plot(rfImp, top=15, scales = list(y = list(cex = 0.8)))

options(digits = 5)
rfImp2 <- rfImp$importance[order(-rfImp$importance$Overall), , drop=FALSE]
kab_tab2(head(rfImp2, 10), cap="Variable Importance Scores")

#install.packages('rpart.plot')
library(rpart.plot)
tree <- rpart(PH~., data=traindata)
prp(tree)

library(randomForest)
#Using full dataset (applying na.roughfix to missing values)
cstData_all<-subset(stData[-2])
set.seed(123)
trainidx2<-sample(nrow(cstData_all),round(0.7*nrow(cstData_all)),replace=F)
traindata2<-cstData_all[trainidx2,]
testdata2<-cstData_all[-trainidx2,]

##Additional Random Forest tuning (TUNE HERE)
#rf.model2 <- randomForest(PH~., data=traindata2, na.action=na.roughfix)
rf.model2 <- randomForest(PH~., data=traindata2, na.action=na.roughfix, importance=TRUE)
rfPred2 <- predict(rf.model2, newdata = testdata2)
m10 <- data.frame(postResample(pred = rfPred2, obs = testdata2$PH)) #0.1097018 0.6173039
m10 <- t(m10)
row.names(m10) <- c("Random Forest Model All Predictors")
kab_tab2(m10, "Accuracy Measures for Random Forest Model using Full Set of Predictors")

#with importance=TRUE, uses approach by Breiman to calculate the variable importance reported as MeanDe
#https://stackoverflow.com/questions/37888619/difference-between-varimp-caret-and-importance-randomfore
rfImp2 <- as.data.frame(importance(rf.model2, scale = FALSE))
options(digits = 5)
kab_tab2(head(rfImp2[order(-rfImp2[,2]),], 10), cap="Variable Importance Scores")

#Comparing Adj. Rsquared for OLS Linear Regression models, inputting the top-ten most important variabl
##Linear Regression using all variables (Mnf.Flow_ord instead of Mnf.Flow)
linreg2 <- caret::train(PH~., data=traindata, method="lm",  trControl=trctrl)
# summary(linreg2) #Adjusted R-squared:  0.356
linPred2 <- predict(linreg2, newdata = testdata)
m11<-data.frame(postResample(pred = linPred2, obs = testdata$PH)) #0.1414880 0.3775156

##Linear Regression using only top 10 variables
linreg3 <- caret::train(PH ~ Mnf.Flow + Usage.cont + Carb.Rel + Bowl.Setpoint +
                          Temperature + Pressure.Vacuum  + Air.Pressurer +
                          Oxygen.Filler + Pressure.Setpoint + Hyd.Pressure3,
                        data=traindata, method="lm",  trControl=trctrl)
# summary(linreg3) # Adjusted R-squared:  0.328
linPred3 <- predict(linreg3, newdata = testdata)
m12<-data.frame(postResample(pred = linPred3, obs = testdata$PH)) #0.1414880 0.3775156
```

```r
##Linear Regression using only top 8 variables (Removing Pressure.Vacuum     + Air.Pressurer)
linreg4 <- caret::train(PH ~ Mnf.Flow + Usage.cont + Carb.Rel + Bowl.Setpoint +
                          Temperature +
                          Oxygen.Filler + Pressure.Setpoint + Hyd.Pressure3,
                        data=traindata, method="lm", trControl=trctrl)
# summary(linreg4) # Adjusted R-squared:  0.329
linPred4 <- predict(linreg4, newdata = testdata)
m13<-data.frame(postResample(pred = linPred4, obs = testdata$PH)) #0.1414880 0.3775156


df<-data.frame(rbind(m11[,1],m12[,1],m13[,1]))
rownames(df)<-c("Linear Regression 2","Linear Regression 3","Linear Regression 4")
colnames(df)<-c("RMSE","Rsquared","MAE")
df <- df[order(df$RMSE),]
options(digits = 3)
kab_tab2(df, cap="MODELS")
# df

summary(linreg4) # Adjusted R-squared:  0.329

par(mfrow=c(1,2))
#Visualing the Mnf.Flow column
plot(stData$Mnf.Flow, stData$PH)
hist(stData$Mnf.Flow)

#Investigating percent of null values in ranged bins of Mnf.Flow
nrow(stData) #2571 rows
sum(!complete.cases(stData)) #442 rows with missing data

neg.Mnf.Flow.count <- nrow(stData[stData$Mnf.Flow < 0,]) #1186 rows with negative values
neg.complete.Mnf.Flow.count <- sum(complete.cases(stData[stData$Mnf.Flow < 0,])) #989 complete cases
neg.complete.Mnf.Flow.count / neg.Mnf.Flow.count #0.8338954

nearzeros.Mnf.Flow.count <- nrow(stData[stData$Mnf.Flow >= 0 & stData$Mnf.Flow <= 1 ,]) #81 rows
nearzeros.complete.Mnf.Flow.count <-sum(complete.cases(stData[stData$Mnf.Flow >= 0 & stData$Mnf.Flow <=
nearzeros.complete.Mnf.Flow.count / nearzeros.Mnf.Flow.count #0.01234568 1% of data complete

pos.Mnf.Flow.count <- nrow(stData[stData$Mnf.Flow > 1,]) #1308 total rows with with Mnf.Flow greater th
pos.complete.Mnf.Flow.count <- sum(complete.cases(stData[stData$Mnf.Flow > 1,])) #1140 complete cases w
pos.complete.Mnf.Flow.count / pos.Mnf.Flow.count #0.8707951 87% of data complete


nearzeros.Mnf.Flow.count / nrow(stData) #.03150 3% of the total number of rows accounts for 18% of the 

#Finding the mean of the Mnf.FLow from subset of values greater than 1, used in binning
#pos.Mnf.Flow <- stData[stData$Mnf.Flow > 1,]
#mean(pos.Mnf.Flow$Mnf.Flow, na.rm = T)

#Separating the Mnf.Flow column by thresholds
stData$Mnf.Flow_ord <- cut(
  stData$Mnf.Flow,
  breaks = c(-Inf, -1, 1, 140, Inf),
  labels = c(1, 2, 3, 4),
  right  = FALSE
)
table(stData$Mnf.Flow_ord)

#Violin plot of Mnf.Flow by bins
g <-ggplot(stData, aes(x=factor(stData$Mnf.Flow_ord), y=stData$PH))
g+geom_violin(alpha=0.5, color='grey') +
  geom_jitter(alpha=0.5, size=4, aes(), position = position_jitter(width = 0.1), color='darkblue', show
  ggtitle("PH by Mnf.Flow classification") +
  coord_flip() +
  theme(panel.background = element_blank(), legend.position="top")

#Subsetting data by brand
brandA <- stDatao[stDatao$Brand.Code == 'A',]
brandB <- stDatao[stDatao$Brand.Code == 'B',]
brandC <- stDatao[stDatao$Brand.Code == 'C',]
brandD <- stDatao[stDatao$Brand.Code == 'D',]

###Add Trimmed Means to NA Value
r <- colnames(cstData_all)[ apply(cstData_all, 2, anyNA)]

cstData_all[,colnames(cstData_all) %in% r]<-data.frame(sapply(cstData_all[,colnames(cstData_all) %in% r]
      function(x) ifelse(is.na(x),
            mean(x, na.rm = TRUE, trim = .1),
            x)))

df<-data.frame()
#BrandA Training/Test Splitting
set.seed(123)
trainidxA<-sample(nrow(brandA),round(0.7*nrow(brandA)),replace=F)
traindataA<-cstData_all[trainidxA,]
testdataA<-cstData_all[-trainidxA,]

##RandomForest (on BrandA)
trctrl<- trainControl(method="repeatedcv", number=2,repeats=2)
rforestA <- caret::train(PH~., data=traindataA, method="cforest",
                trControl=trctrl, tuneLength =2, na.action=na.omit)
forPredA <- predict(rforestA, newdata = testdataA)
l1<-data.frame(postResample(pred = forPredA, obs = testdataA$PH))

#BrandB Training/Test Splitting
set.seed(123)
trainidxB<-sample(nrow(brandB),round(0.7*nrow(brandB)),replace=F)
traindataB<-cstData_all[trainidxB,]
testdataB<-cstData_all[-trainidxB,]

##RandomForest (on BrandB)
trctrl<- trainControl(method="repeatedcv", number=2,repeats=2)
rforestB <- caret::train(PH~., data=traindataB, method="cforest",
                trControl=trctrl, tuneLength =2, na.action=na.omit)
forPredB <- predict(rforestB, newdata = testdataB)
l2<-data.frame(postResample(pred = forPredB, obs = testdataB$PH))
```

```r
#BrandC Training/Test Splitting
set.seed(123)
trainidxC<-sample(nrow(brandC),round(0.7*nrow(brandC)),replace=F)
traindataC<-cstData_all[trainidxC,]
testdataC<-cstData_all[-trainidxC,]

##RandomForest (on BrandC)
trctrl<- trainControl(method="repeatedcv", number=2,repeats=2)
rforestC <- caret::train(PH~., data=traindataC, method="cforest",
                trControl=trctrl, tuneLength =2, na.action=na.omit)
forPredC <- predict(rforestC, newdata = testdataC)
l3<-data.frame(postResample(pred = forPredC, obs = testdataC$PH))

#BrandD Training/Test Splitting
set.seed(123)
trainidxD<-sample(nrow(brandD),round(0.7*nrow(brandD)),replace=F)
traindataD<-cstData_all[trainidxD,]
testdataD<-cstData_all[-trainidxD,]

##RandomForest (on BrandD)
trctrl<- trainControl(method="repeatedcv", number=2,repeats=2)
rforestD <- caret::train(PH~., data=traindataD, method="cforest",
                trControl=trctrl, tuneLength =2, na.action=na.omit)
forPredD <- predict(rforestD, newdata = testdataD)
l4<-data.frame(postResample(pred = forPredD, obs = testdataD$PH))

freq <- as.data.frame(table(stDatao$Brand.Code))
rownames(freq)<-c("", "A","B","C","D")
freq <- freq[2:5,2]

df <-data.frame(rbind(l1[,1],l2[,1],l3[,1],l4[,1]))
rownames(df)<-c("A","B","C","D")
colnames(df)<-c("RMSE","Rsquared","MAE")
df <- cbind(df, freq)
df <- df[order(df$RMSE),]
kab_tab2(df, cap="BRANDS")

#pH by Brand
ggplot(stData, aes(Brand.Code, PH)) +
  geom_violin(color = 'grey') +
  geom_jitter(aes(color = Brand.Code), size = 0.8) +
  ggtitle('pH by Brand') +
  geom_hline(yintercept =8.55) +
  coord_flip() +
  theme(panel.background = element_blank(), legend.position="top")

# par(mfrow=c(2,2)) # Doesn't work!
#Variable Importance Ranking (on Brand A)
rfImpA <- varImp(rforestA, scale = FALSE)
plot(rfImpA, top=5, scales = list(y = list(cex = 0.8)))

#Variable Importance Ranking (on Brand B)
rfImpB <- varImp(rforestB, scale = FALSE)
```

```r
plot(rfImpB, top=5, scales = list(y = list(cex = 0.8)))

#Variable Importance Ranking (on Brand C)
rfImpC <- varImp(rforestC, scale = FALSE)
plot(rfImpC, top=5, scales = list(y = list(cex = 0.8)))

#Variable Importance Ranking (on Brand D)
rfImpD <- varImp(rforestD, scale = FALSE)
plot(rfImpD, top=5, scales = list(y = list(cex = 0.8)))



eval_p<-predict(rforest,stEval, type = "raw")
# summary(eval_p)
write.csv(eval_p,"predicted_eval_values_PH.csv")
```