DATA 624: Project 2

Group 2

Group Members:
Vinicio Haro
Sang Yoon (Andy) Hwang
Julian McEachern
Jeremy O'Brien
Bethany Poulin

10 December 2019

Contents

1 Executive Summary

pH is a central component to the manufacturing of a commercial beverages, it is an indicator of both the manufacturing process's health and the ultimate flavor appeal of the final product. In fact pH plays a role in multiple facets of beverage appeal. The flavor, mouth-feel and aesthetic experience of a given product is distinctly tied to pH relative to other beverage qualities that brands use distinguish themselves from other liquid refreshments.

Because of the importance PH plays in the design of a products drinking experience, pH is a key performance indicator in the beverage manufacturing process. pH is tested for and tracked diligently, as the final pH is dependent on and vulnerable to even slight changes in production methods.

Having monitored and recorded these production variables, as well as the final pH, we have the opportunity to improve production outcomes by closely controlling pH in our beverages with predictive modeling. Modeling provides the potential to catch and correct variations in process which negative impact our taget pH.

Each group member worked individually to experiment with preprocessing while exploring a distinct set of model methods. Upon review, our team created a singular preprocessing protocol and data set, based on our most successful methods and evaluated our most performant models built over these data.

2 Data Exploration

Data Preperation was the most discussed and influential part of our modeling process. It was clear from early on that in order to build a useful model with such a narrow range of pH values, we needed to make justified decisions on how to groom the data. Our decisions would likely be as or more unfluential than the model we picked.

The beverage dataset includes 2,571 cases, 32 predictor variables, and a single response variable. One of these predictor variables (`BrandCode`) is categorical with four levels - A through D; for the purpose of our analysis we interpreted these to represent four distinct beverage brands.

Missing Values

While we found missing observations in both response and predictor variables, in our assessment the extent of NAs did not suggest a systemic issue in measurement or recording that imputing values could not remedy.

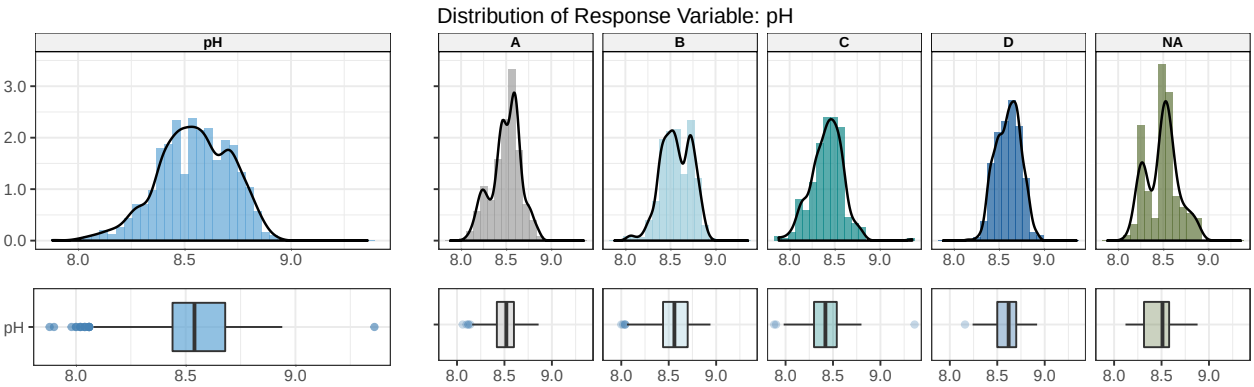
- The response variable (PH) is missing a total of four observations (< 1%)
- Most (30) predictor variables are missing at least one observation, but only eleven are missing more than 1% of total cases and only three are missing more than 2% of total cases.

Table 2.1: Variables with Highest Frequency of Missing Values

	MFR	BrandCode	FillerSpeed	PCVolume	PSCCO2	FillOunces	PSC	CarbPressure1	HydPressure4	CarbPressure	CarbTemp
n	212.0	120.0	57.0	39.0	39.0	38.0	33.0	32.0	30.0	27.0	26
%	8.2	4.7	2.2	1.5	1.5	1.5	1.3	1.2	1.2	1.1	1

Response Variable

Our target variable is pH. pH is an inversely logarithmically scaled measure of hydrogen ions in solutions and reflects how acidic or basic a water-based solution is and as such is a continuous variable. While ph in general is centered around a neutral value of 7, pH ranges from highly acidic 0 and to highly alkaline at 14 our sample pH distribution is approximately normal and centered around 8.546 (i.e. slightly basic), with some negative skew and outliers .



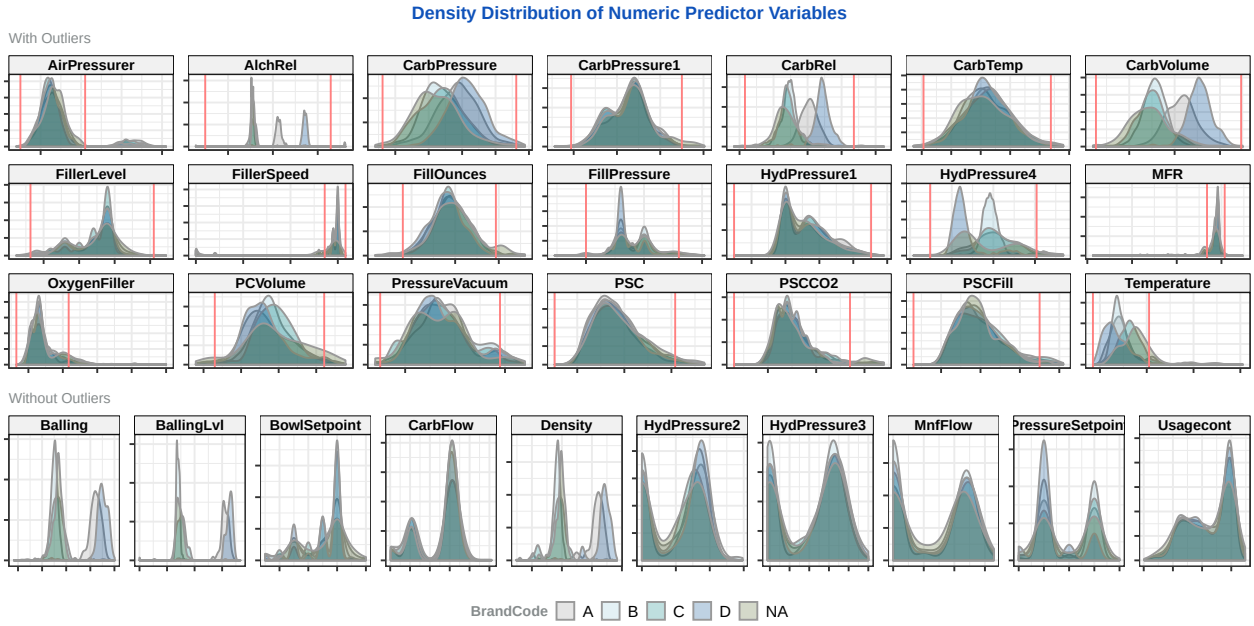
When pH is evaluated by Brand Code we see some slight variation in the distribution:

- A (293 observations) appears to be multimodal and have the most outliers, with a mean slightly lower than the aggregate (8.495)
- B (1293 observations) appears to be bimodal with a number of outliers, as well as a mean nearest the aggregate (8.562)
- C (304 observations) appears to be bimodal and is the most acid (8.419)
- D (615 observations) is the most normal distribution and also has the highest alkalinity (8.603)
- NA (120 observations) most is the most variant grouping, is bimodal and has a similar to mean A & C (8.49)

Predictor Variables

Based on our exploration of ph relative to Brand Code, we decided to compare predictor variable distributions using the same segmentation and included outlier boundary markers (in red) showing. Individual samples falling outside of these boundaries would typically be considered outliers of the sample set as a whole.

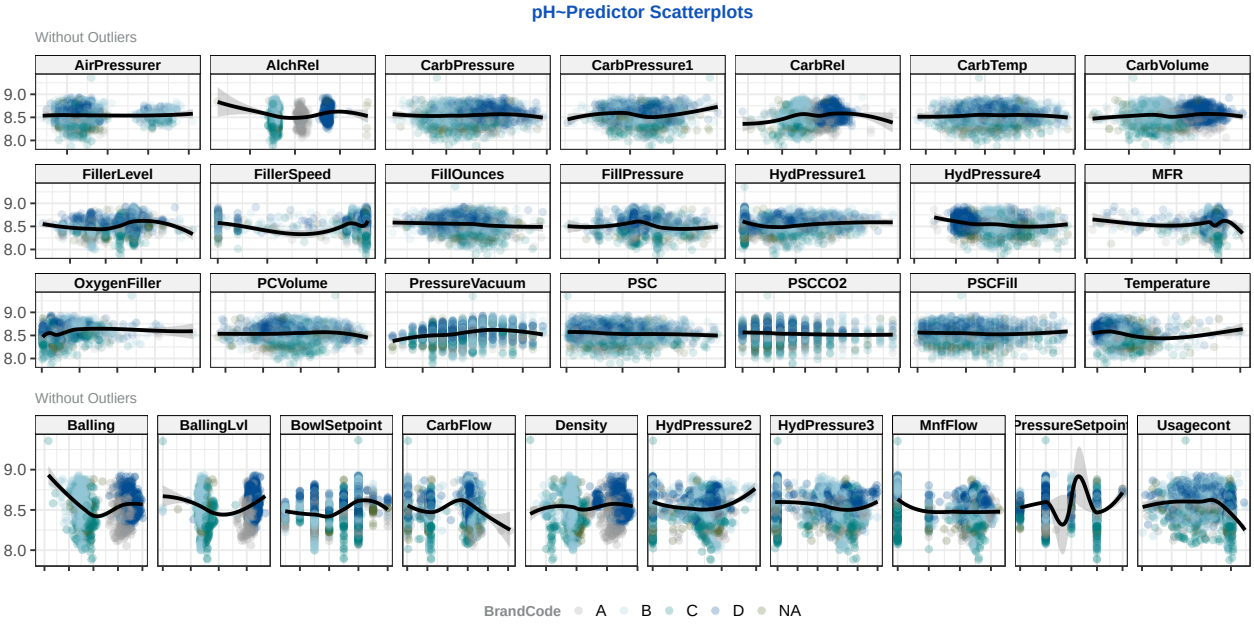
When viewed in light of the `BrandCode` overlay there is evidence that some of our variables, such as `AlchRel`, `CarbRel`, `CarbVolume`, `HydPressure4`, and `Tempature`, are strongly influenced by brand. We may need to consider this in our preprocessing and modeling proceedures.



Predictor - Response Variable Relationships

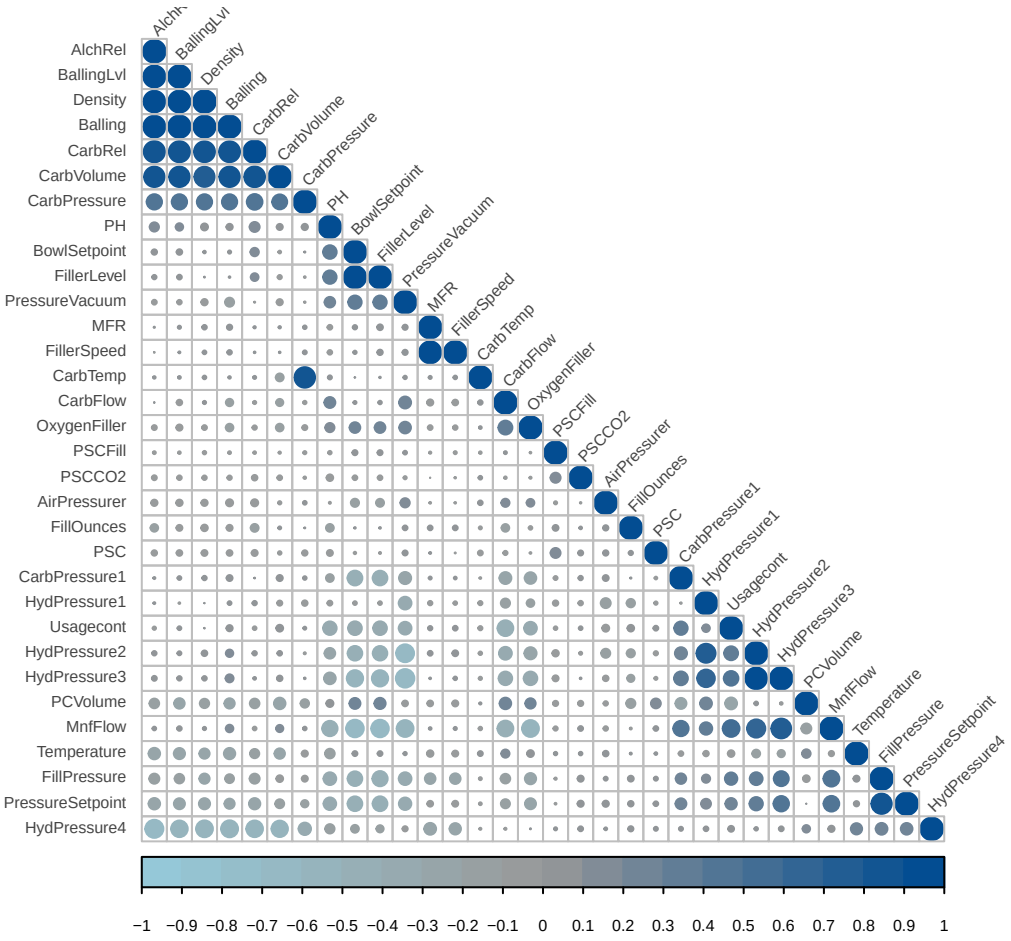
Plotting each predictor variable reinforced our suspicions about brand relationships between predictors and our response variable pH. `AlchRel` shows distinct groupings by `BrandCode` as do `Carb` related and `Balling` related variables. Some of the other variables, `PSCO2`, `BowlSetpoint`, `MinFlow`, and `PressureSetup` show unique dicretized features likely related to system processes.

Without question, these plots solidify the need to explicitly account for brand in our preprocessing including both imputation and train-test splits to be sure that we capture the true influence brand has over the pH. Also as no predictor variable shows a pronounced monotonic linear relationship with pH, non-linear approaches to modeling will likely provide better models.



Correlations Between Predictors

Collinearity measures between numeric predictors indicate that several of these variables are correlated, with correlation values exceeding ± 0.7. This is somethng we had to consider when deciding between final models as both support vector machines with radial kernels as well as MARS models are vulnerable to collinearity. We left the correlated varibles in the models to begin with and took them into consideration when comparing model test metrics to assess our path forward.



3 Data Preparation

In our exploration, we discovered missing data, extreme outliers, and multicollinearity all challenges we needed to address both in our processing of the data and model building. We were strategic about when and how to impute, split, scale and transform when preparing our models. Our goal was to build the most predictive and stable models. To ensure that we were all working with the same data and processing, the team established a seed (58677) which we used for all randomized processes.

Train/Test Splits

After extensive deliberation, we decided to use train-test-split over repeated cross validations. Spliting was done prior to preprocessing the data to stave off the negative effects of training leakage on our evaluation model.

We divided the data into 80% training and 20 testing observation using our team seed. However given our intuition that brand code is influential on pH, we used a method often applied to target variables in classification models. We prioritzed the brands before splits to ensure that all brands and unknowns would be fully represented in both the training and testing sets.

Data Imputation

Missing values for all numerical values were imputed using the `caret` package, and bagging algorithm so that the same range of imputed values could be applied to the test and validation sets without confounding our training data.

We decided not to impute the Brand Code. Instead, missing values were labelled NA prior to creating dummy variables so that four new variables, `brandA`, `brandB`, `brandC` and `brandD` were created. NA values would be reflected by all zeroes. This was done so that each of the observations with a known brand would be more accurately described by the other variables in our regressions relative to pH, without error introduced by the imputed brand values.

Our imputation method was saved and applied to both the test and evaluation data sets so that an identical observation in the the test or evaluation set would be imputed as it was in training.

4 Models

During preliminary exploration we assessed the effectiveness of more than ten different linear and non-linear regression models, settling on four models that exhibited the most favorable Root Mean Squared Error. We tuned those models, and then chose the best performing model of that set to use in our final analyses.

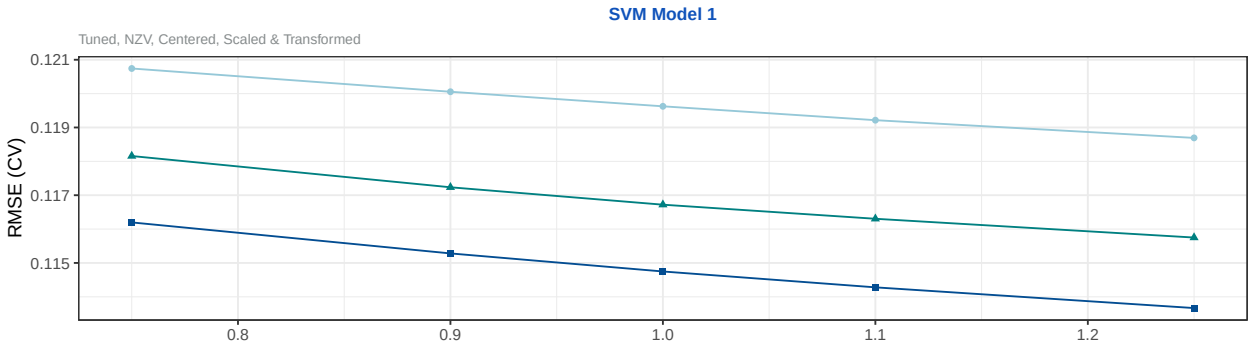
Support Vector Machines (SVM) Regression

Support vector machine (SVM) regression with a radial bias function kernel is a promising choice for predicting beverage pH because it excels when working with data which may not be linearly separable, is immune to the effect of outliers (which we had) and excels with high variable to observation ratios.

ur SVM model received the following preprocessing

- Removal of near zero variance variables
- Centering
- Scaling
- Box-Cox transformations of skewed variables

Although less efficient to train than many other models, with grid tuning the radial bias kernel SVM provided a robust final model after grid tuning, using a radial kernel. The final model's parameters were $\sigma = 0.1$ and $cost = 1.25$ returning a $RMSE = 0.09377$



Cubist Tree Regression

Because cubist models work in two different capacities, to predict a continuous outcome variable, they work exceedingly well on non-linear data which we have and can describe node values well due to their hyper localized regressions and regional concensus.

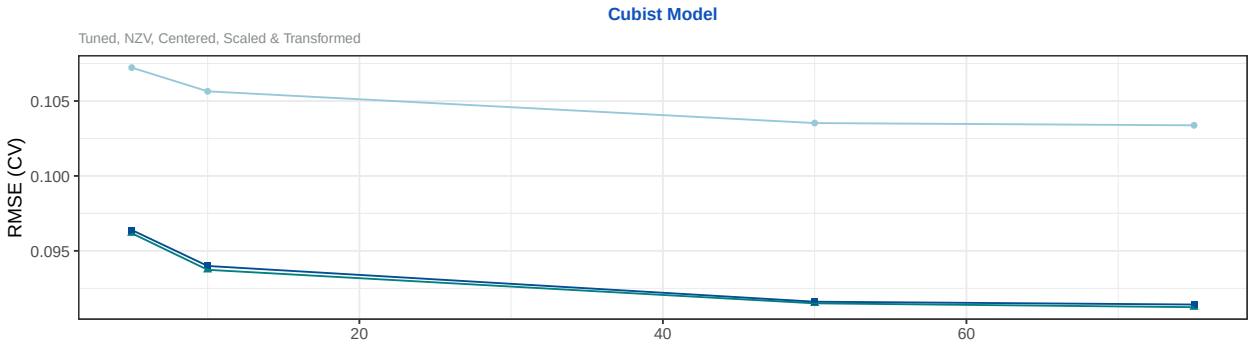
Because they are trees, cubist models are very forgiving of many of the same things that other tree-based models are, and thus need very little pre-processing. However, because the final nodes are linear regressions, normal data is preferable.

Accordingly our cubist model recieved the following preprocessing :

- Removal of near zero variance variables
- Centering
- Scaling
- Box-Cox transformations of skewed variables

It may or may not have been necessary to center and scale, but given the small differences in pH we are trying to predict and the potential for magnitude of variables to influence our node regressions, this was done as a precaution and presents few if any risks

Our final training model had an RMSE of 0.002812 with 75 committees oand 100 rules.



Multivariate Adaptive Regression Splines (MARS) Regression

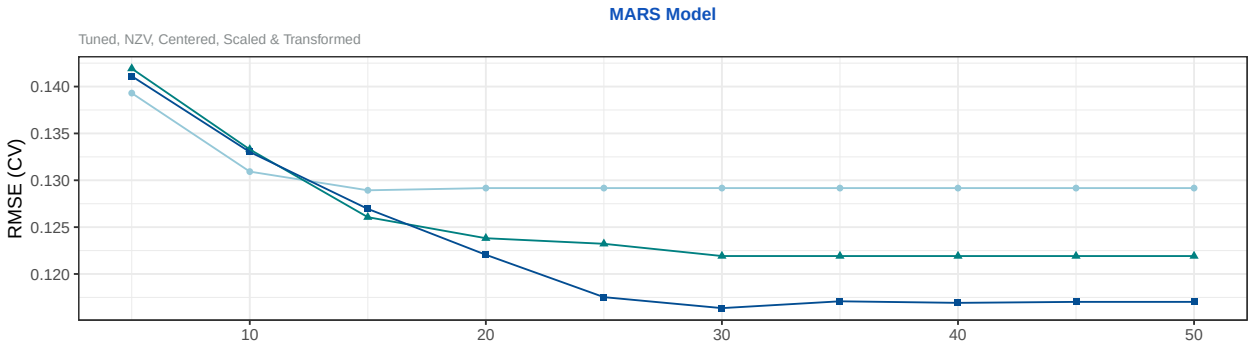
Multvariate Adaptive Regression Splines was considered because of its flexibility with regard to linear and non-linear variables. MARS also has the ability to accurately model additive processes and intertactions in a single models and it is likely to perform

well on unseen data.

Our MARS model recieved the following preprocessing:

- Removal of near zero variance variables
- Centering
- Scaling
- Box-Cox transformations of skewed variables

The final model was pruned to 30 and had a degree of interaction of 3 and returned an RMSE of 0.1070.

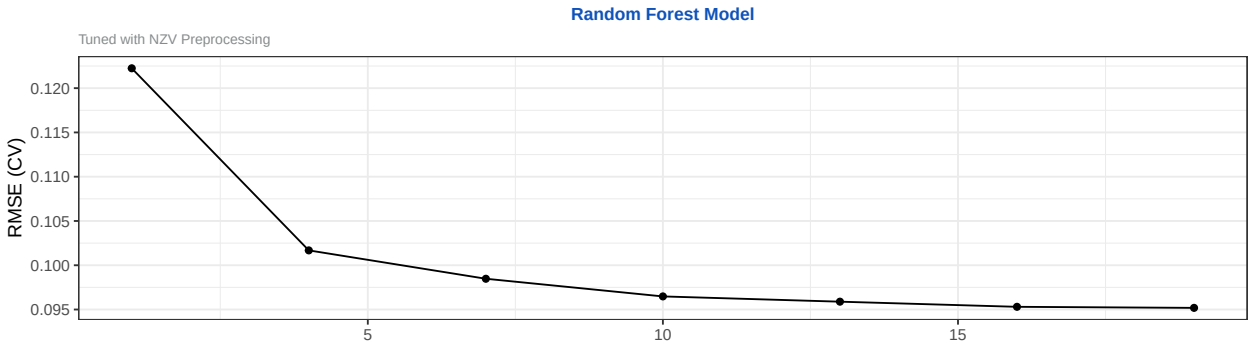


Random Forest Regression

The random forest model was a prime choice because, properly grown, it compensates well for overfitting, is completely oblivious to non-linearity, and compensates well for varying scales in variables while still training relatively quickly.

Little preprocessing was necessary. We simply removed near zero variance variables.

Our random forest model was optimized to 500 trees, mtry sampled at 19 returning a training RMSE of 0.0379.



Model Performance

Table 4.1: Training and Test Metrics

Training	MAPE	RMSE	Rsquared	MAE
MARS	0.0097	0.1069	0.6100	0.0823
Random Forest	0.0032	0.0379	0.9654	0.0272
SVM	0.0074	0.0938	0.7109	0.0629
Cubist	0.0028	0.0335	0.9648	0.0240

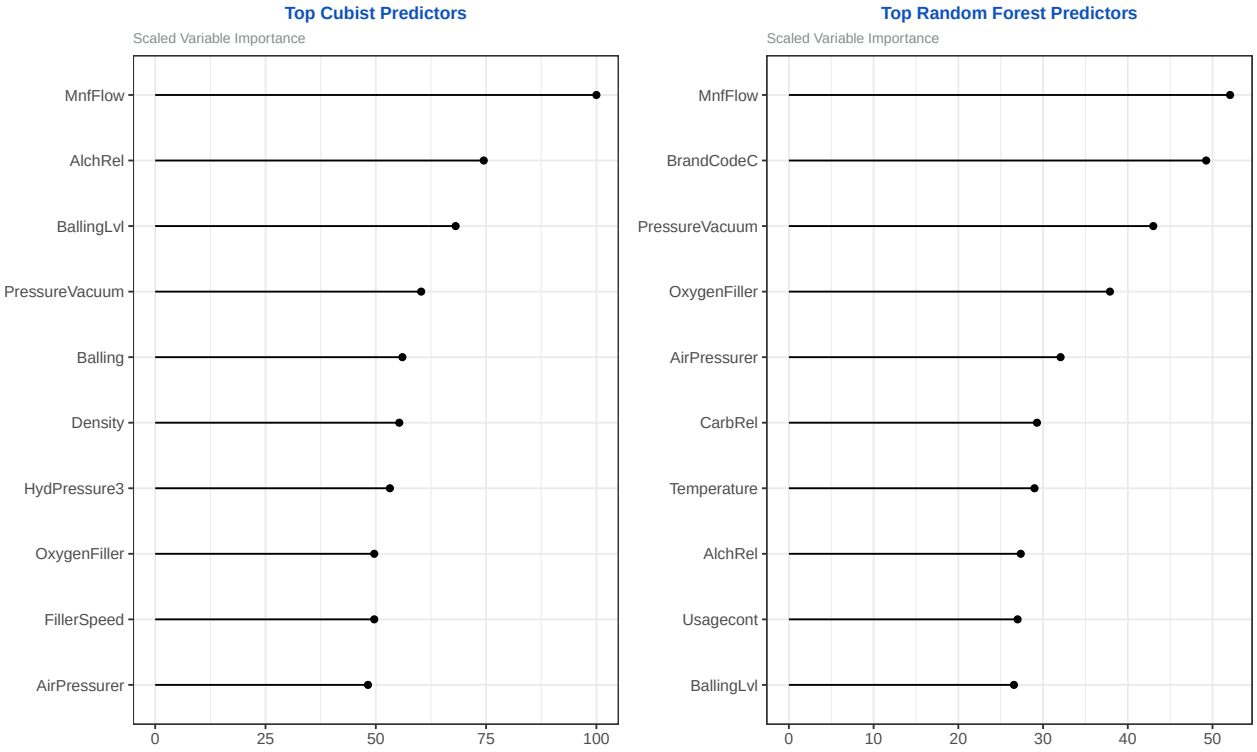
Test	MAPE	RMSE	Rsquared	MAE
MARS	0.0122	0.1401	0.3900	0.1036
Random Forest	0.0091	0.1095	0.6318	0.0772
SVM	0.0111	0.1295	0.4679	0.0944
Cubist	0.0093	0.1145	0.5873	0.0788

All four of our final models had excellent training MAPE metrics, below 1%. However, the cubist model .28% and random forest .32% models were exceptional and given their ease of deployment and ability to interpret the variable importance, it was heartening to see that their test MAPE's were both below 1% .

5 Final Model Selection

We selected random forest as the final model because of consistency in scores from training to testing. And secondarily, but non-trivially, we were somewhat concerned that the four of the top six variables of importance for the cubist model were all 60% or more correlated with each other and might becausing some localized overfitting in the regression nodes of the cubist model. Comparatively the random forest had 3 of the highly correlated variables, `AlchRel`, `CarbRel` & `BallingLvl` but they were more spread out across the importance scale.

Also interesting, the `BrandCodec` ends up being an important factor in the random forest model, supporting our hypothesis after preliminary exploration of the variables.



6 Recommendations

In looking at the top ten variables for both the cubist and the random forest model, it was interesting that most influential variables were predominatly related to gas regulation. `CarbRel`, `AirPressurer`, `OxygenFiller`, `PressureVacuum`, `BrandCodeC` `MnfFlow` were the top 6 variables in the random forest model. All but `BrandCodeC`, are related to aeration in some way. This makes sense given that where something falls on the acid-base spectrum has to do with free hydrogen ions, which are affected by presure and the presence of other naturally occuring gases such as oxygen, and introduced gases like carbon dioxide used to carbonate beverages.

Given that the gas regulation measures are all critical factors in our beverage pH predictions, if we hope to maintain tight tolerances and produce the most pleasing flavor and mouthfeel in our beverages, it would seem logical to focus on the management, maintenance and performance of systems related to these variables.

As is, our model could be used (or adapted and refined) to alert production managers when ph is shifting out of an acceptable range for a given beverage. It is also our belief that with appropriate time and data, we could build a model to operate over

live streaming data which could recommend optimal maintenance of the valves and systems regulating gas flow and pressure to optimize results and reduce out-of-range ph tests on future batches. It may also be possible to integrate a model within one or more key systems to predict when they are moving out of range and to prescribe a course of corrective action so that they beverages never leave their targeted key performance range.

Appendix

A. Code

```
# Final Grooming Training and Testing Code
library(tidyverse)
library(readxl)
library(readr)
library(MLmetrics)
library(caret)
library(fastDummies)
library(caretEnsemble)
library(caTools)
library(recipes)
require(dummies)
# SEEDING
set.seed(58677)

# DATA PREPARATION

## Import data
StudentData <- read_xlsx('/Users/bethany/Documents/P2/data/StudentData.xlsx')
StudentEvaluation <- read_xlsx('/Users/bethany/Documents/P2/data/StudentEvaluation.xlsx')

## Clean/Standardize Variable Naming Conventions for Both Sets
names(StudentData) <- gsub(" ", "", names(StudentData))
names(StudentEvaluation) <- gsub(" ", "", names(StudentEvaluation))

## Prep For Train/Test Splits
split <- StudentData %>%
  filter(complete.cases(PH)) %>%
  mutate(BrandCode=ifelse(is.na(BrandCode), NA, BrandCode))# %>%


# Subset to train test relative to even BrandCodes
set.seed(58677)
sample = sample.split(factor(split$BrandCode), SplitRatio = .8)
train = subset(split, sample == TRUE)%>%mutate(BrandCode = factor(BrandCode))
test  = subset(split, sample == FALSE)%>%mutate(BrandCode = factor(BrandCode))

## Apply Mutations to StudentEval
eval <- StudentEvaluation %>%
  mutate(BrandCode=ifelse(is.na(BrandCode), NA, BrandCode))%>%
  mutate(BrandCode = factor(BrandCode))

# PREPROCESSING

# Make Dummies

e_dummies <-dummy(eval$BrandCode)
test_dummies <-dummy(test$BrandCode)
```

```
train_dummies <-dummy(train$BrandCode)

# Build Imputation Method
ivals <- preProcess(train[,2:ncol(train)], method = "bagImpute")

# Apply Imputations
train<- predict(ivals, newdata = train[2:ncol(train)])
test <- predict(ivals, newdata = test[2:ncol(test)])
eval <- predict(ivals, newdata = eval[2:ncol(eval)])


##Bind data with Dummies

train<-cbind(train_dummies, train)
test<-cbind(test_dummies, test)
eval<-cbind(e_dummies, eval)

###Verify Imputation
#sapply(eval, function(x) sum(is.na(x)))
##sapply(train_split, function(x) sum(is.na(x)))
##sapply(test_split, function(x) sum(is.na(x)))



saveRDS(train, file = "train.rsd")
saveRDS(test, file = "test.rsd")
saveRDS(eval, file = "eval.rsd")


# MODELING
## parameters
# tl <- 5  ## tuneLength = number of parameter to be evaluated
trC <- trainControl(method = "cv", ## define resampling method
                    number = 10,
                    returnData = T,
                    savePredictions="final")

## Tune Grids
grid_mars <- expand.grid(degree=1:3,
                         nprune = seq(5, 50, by = 5))
grid_rf <- expand.grid(.mtry= seq(1, 21, by=3))
grid_svm <- expand.grid(sigma = c(.01, .015, 0.02),
                    C = c(0.75, 0.9, 1, 1.1, 1.25))
grid_cub <- expand.grid(committees = c(5, 10, 50, 75),
                        neighbors = c(1, 5, 9))

# MODELING (JUST RUNNING STANDARD - ADD preProc if desired)

## Without Additional PreProccessing
fit_mars1 <- train(PH~.,
                   data=train,
                   method = 'earth',
                   preProc = c('nzv', 'center', 'scale',"BoxCox"),
```

```
                    tuneGrid = grid_mars,
                    trControl=trC,
                    tuneLength=10,
                    metric="RMSE")
fit_rf1 <- train(PH~.,
                 data=train,
                 method="rf",
                 preProc = c('nzv'),
                 tuneGrid = grid_rf,
                 importance=T,
                 trControl=trC,
                 tuneLength=tl,
                 metric="RMSE")
fit_svm1 <- train(PH~.,
                  data=train,
                  method = "svmRadial",
                  preProc = c('nzv','center', 'scale',  "BoxCox"),
                  tuneGrid = grid_svm,
                  trControl=trC,
                  tuneLength=tl, metric="RMSE")
fit_cub1 <- train(PH~.,
                  data=train,
                  method = 'cubist',
                  preProc = c('nzv',  'center', 'scale', "BoxCox"),
                  tuneGrid = grid_cub,
                  trControl=trC,
                  tuneLength=tl,
                  metric="RMSE")

## Train Results
mars = fit_mars1$finalModel
mars_train_preds <- predict(fit_mars1, new_data = train)
mars_metrics <-postResample(mars_train_preds, obs = train$PH)
mars_train_mape <- MAPE(mars_train_preds,  train$PH)

mars_test_preds <- predict(fit_mars1,  test)
mars_test_metrics <-postResample(mars_test_preds, obs = test$PH)
mars_test_mape <- MAPE(mars_test_preds,  test$PH)
# Random Forest
rf = fit_rf1$finalModel
rf_train_preds <- predict(fit_rf1, new_data = train)
rf_metrics <-postResample(rf_train_preds, obs = train$PH)
rf_train_mape <- MAPE(rf_train_preds,  train$PH)

rf_test_preds <- predict(fit_rf1,  test)
rf_test_metrics <-postResample(rf_test_preds, obs = test$PH)
rf_test_mape <- MAPE(rf_test_preds,  test$PH)

#SVM
svm = fit_svm1$finalModel
svm_train_preds <- predict(fit_svm1, new_data = train)
svm_metrics <-postResample(svm_train_preds, obs = train$PH)
svm_train_mape <- MAPE(svm_train_preds,  train$PH)
```

```
svm_test_preds <- predict(fit_svm1,  test)
svm_test_metrics <-postResample(svm_test_preds, obs = test$PH)
svm_test_mape <- MAPE(svm_test_preds,  test$PH)

#Cubist
cube = fit_cub1$finalModel
cube_train_preds <- predict(fit_cub1, new_data = train)
cube_metrics <-postResample(cube_train_preds, obs = train$PH)
cube_train_mape <- MAPE(cube_train_preds,  train$PH)

cube_test_preds <- predict(fit_cub1,  test)
cube_test_metrics <-postResample(cube_test_preds, obs = test$PH)
cube_test_mape <- MAPE(cube_test_preds,  test$PH)


train_metrics <- rbind(mars_metrics, rf_metrics, svm_metrics, cube_metrics)%>%
  data.frame()%>%
  mutate(MAPE = c(mars_train_mape, rf_train_mape, svm_train_mape, cube_train_mape),
         Model = c('MARS', 'Random Forest', 'SVM', "Cubist"))%>%
  select(Model, MAPE, RMSE, Rsquared, MAE)

test_metrics <- rbind(mars_test_metrics, rf_test_metrics,
                      svm_test_metrics, cube_test_metrics)%>%
  data.frame()%>%
  mutate(MAPE = c(mars_test_mape, rf_test_mape, svm_test_mape, cube_test_mape),
         Model = c('MARS', 'Random Forest', 'SVM', "Cubist"))%>%
  select(Model, MAPE, RMSE, Rsquared, MAE)


## Save Models
saveRDS(fit_mars1, "fit_mars1.rsd")
saveRDS(fit_rf1, "fit_rf1.rsd")
saveRDS(fit_cub1, "fit_cub1.rsd")
saveRDS(fit_svm1, "fit_svm1.rsd")
saveRDS(train_metrics, "train_met.rsd")
saveRDS(test_metrics, "test_met.rsd")

#Script we source for RMD

library(tidyverse)
library(MLmetrics)
library(caret)
library(caretEnsemble)
library(psych)
library(stats)
library(data.table)
library(readxl)

## Import data
StudentData <- read_xlsx('data/StudentData.xlsx')
StudentEvaluation <- read_xlsx('data/StudentEvaluation.xlsx')

## Clean/Standardize Variable Naming Conventions for Both Sets
```

```
names(StudentData) <- gsub(" ", "", names(StudentData))
names(StudentEvaluation) <- gsub(" ", "", names(StudentEvaluation))

# Load saved work space
train <- readRDS(file = "data/train.rsd")
test <- readRDS(file = "data/test.rsd")
eval<- readRDS(file = "data/eval.rsd")

fit_mars1 <- readRDS("data/fit_mars1.rsd")

fit_rf1 <- readRDS("data/fit_rf1.rsd")

fit_cub1 <- readRDS("data/fit_cub1.rsd")

fit_svm1 <- readRDS("data/fit_svm1.rsd")
train_metrics <-readRDS("data/train_met.rsd")
test_metrics <-readRDS("data/test_met.rsd")

# CUSTOM FUNCTIONS
gather_if <- function(data, FUN,
                      key = "key",
                      value = "value",
                      na.rm = FALSE,
                      convert = FALSE,
                      factor_key = FALSE) {
  data %>%
  {gather(., key = key,
            value = value ,
          names(.)[sapply(., FUN = FUN)],
          na.rm = na.rm,
          convert = convert,
          factor_key = factor_key )}}

flattenCorrMatrix <- function(cormat) {
  ut <- upper.tri(cormat)
  data.table(row = rownames(cormat)[row(cormat)[ut]],
             column = rownames(cormat)[col(cormat)[ut]],
             cor  = (cormat)[ut])}

# DATA EXPLORATION


## Missing Data Analysis
MissingData <- StudentData %>% summarise_all(funs(sum(is.na(.)))) %>%
  t() %>%
  as.data.frame() %>%
  rename("n"=V1) %>%
  rownames_to_column("predictor") %>%
  arrange(desc(n)) %>%
  mutate(`%`=round((n/nrow(StudentData)*100),2))

## Outiers Analysis
outliers <-StudentData %>%
```

```
  mutate(PH=as.factor(PH))%>%
  gather_if(is.numeric, 'key', 'value') %>%
  filter(!is.na(value)) %>%
  group_by(key)  %>%
  mutate(outlier_lower = quantile(value, probs=.25, na.rm = T)-1.5 * IQR(value, na.rm = T),
         outlier_upper = 1.5 * IQR(value, na.rm = T)+quantile(value, probs = .75, na.rm = T),
         .data = outlier=ifelse(value<outlier_lower, "TRUE",
                                ifelse(value>outlier_upper, "TRUE", "FALSE")))
outlier_with <- outliers %>%
  filter(any(outlier=="TRUE"))
outlier_wo <- outliers %>%
  filter(all(outlier!="TRUE"))
outlier_freq <- outliers %>%
  select(key, outlier) %>%
  table() %>%
  as.data.frame.array() %>%
  rownames_to_column("variable") %>%
  arrange(desc(`TRUE`)) %>%
  mutate(`%`=round(`TRUE`/(`FALSE`+`TRUE`)*100,2)) %>%
  top_n(5,`%`)



#FINAL PREDICTION
eval$PH <- predict(fit_rf1, eval)
openxlsx::write.xlsx(eval, 'StudentPredictions.xlsx')
```

B. Citations

Shelton, Robert B. “PH Values Of Common Drinks.” Robert B. Shelton, DDS MAGD Dentist Longview Texas, 2019, www.sheltondentistry.com/patient-information/ph-values-common-drinks/.

Brown, Theodore L., et al. Chemistry: the Central Science. Pearson, 2018.