PH Production Report

Jeff Shamp, Misha Kollontai, John Kellogg

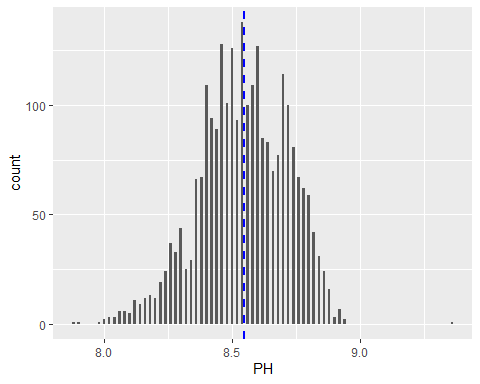
2021-05-18

## Intro

New regulations in the pH level of beverages produced at the Cunyville plant required executive leadership to ask the data team to conduct a study analyzing the numerous factors within our manufacturing process. As the data team, we endeavored to model the pH output within our plant based on all available predictive factors. The goal was to identify the predictors that are most indicative of the final pH value of our product as well as providing a model that sufficiently and accurately predicts pH based on given values of predictors. The report will provide the required data to efficiently steer manufacturing processes towards a desired pH level with (ideally) the minimum required changes to settings within the overall system.

## Data Exploration

The available data, provided to us, for our analysis included the output pH values along with 32 predictor variables in a set of 2,571 observations. The predictor variables included one categorical variable for the Brand associated with each observation. Each other predictor is a numerical setting within our manufacturing process including various pressure measures, temperature settings, fill levels, etc. The output variable PH is fairly normally distributed around a mean of 8.55 as can be seen in the graph below.



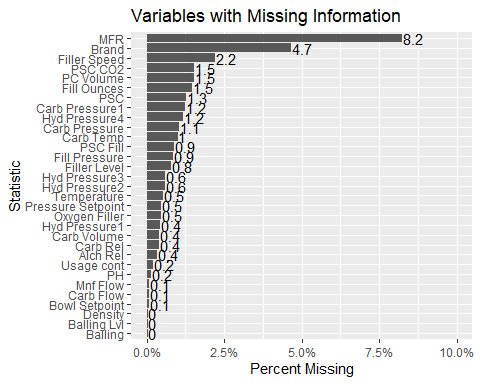
#### Data Risks

In a significant portion of the available data, there is a measurement missing for at least one predictor of a given observation (1,483 of 2,571). In the 32 predictors, we identified a large amount of missing data as well as a few variables with seemingly ‘erroneous’ data.

#### Outliers/Errors

Before any models could be generated from this dataset, the large number of missing data requires attention. Below, we see that three predictors were missing data in more than 2% of observations and eleven predictors missing in at least 1% of observations. The missing data will heavily skew our predictions.

Example of missing data \* Hydraulic Pressures 1, 2, & 3 \* Large sections of the data has no value listed

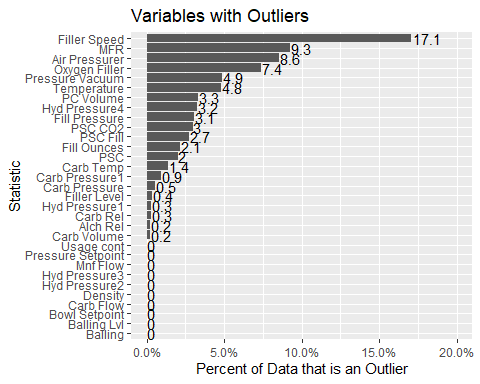


#### Erroneous Data

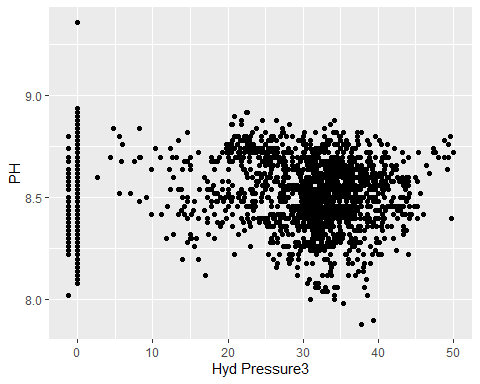
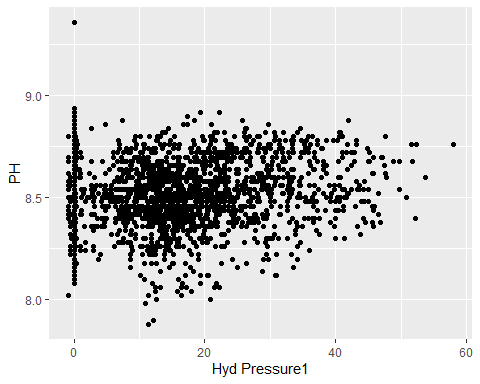
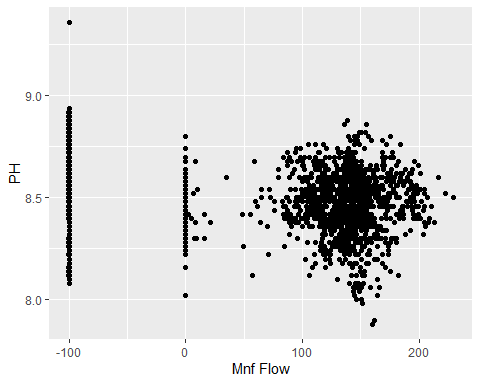
Before imputing the “missing” data, we decided to look over some of the variables and determine if any of the information appeared erroneous. We specifically looked for data points which entail where the value wasn’t missing, but simply did not make sense (i.e. a negative number for a variable where that is impossible).

Example of erroneous/missing data \* Variable ‘Mnf Flow’ (Minimum Night Flow) \* A large subset of the data showed a flow value of -100 even though no other measurements were negative. \* Hydraulic Pressures 1, 2, & 3 \* Negative values existed where negative pressure is not possible.

To start we looked over how many outliers were present in the predictors.



Though a significant number of predictors appears to contain a lot of outliers, not many of it appears to be “erroneous”. A slight dig into each predictor yielded three that contained data points that appeared to have either been recorded in error or incorrectly entered: Mnf Flow, Hyd Pressure1 and Hyd Pressure3. For each of these 3 variables, negative values don’t makes sense since neither flow rate nor hydraulic pressure are likely to be negative by design. Each negative entry for Mnf Flow is at -100 as opposed to a spread of values and the negative hydraulic pressure values are all slightly below 0.



#### Risk Resolution:

To prevent such erroneous data from skewing our results, converted the negative hydraulic pressures and ‘Mnf Flow’ readings with NA’s to treat them as “missing” alongside the other set of missing data in other variables. Missing data was imputed using K-nearest neighbor because it takes an average of the values to either side of the missing value to create the missing one and ensure a complete dataset for our modeling.

## Data Preparation

To construct the data for modeling, we first split the data into training/testing sets along an 80/20 percentage split and utilized the tidymodels package within R to create what is known as a recipe. The recipe outlines the following steps in order to pre-process a given dataset.

Steps of the Recipe 1. Specifies as the predicted variable within the dataset 2. Centers all numeric predictors, normalizing them around 0 3. Performs a Box Cox transformation on numeric predictors 4. Runs a K-Nearest Neighbor imputation on all missing data (NAs) 5. Converts all nominal predictors (Brand) into dummy variables 6. Removes observations for which the predicted variable (PH) is missing

## Model Creation

With the data ready for modeling, several models were generated. We ran so many model types to ensure we could choose the best method to produce the most accurate results. The models created and evaluated against the training data are as follows:

* Linear Regression
* Random Forest (Random Forest engine)
* Random Forest (ranger engine)
* k-Nearest Neighbors
* Cubist

## Model Selection

The models were fit to the training dataset and then each was evaluated against the test set for accuracy metrics. In the below table, we summarize 3 performance metrics for each of the 5 evaluated models: Root Mean Square Error (RMSE), R-Squared (RSQ) and Mean Absolute Error (MAE).

*RMSE (Root Mean Square Error)* TARGET - lowest value *RSQ (R squared)* TARGET - highest value *MAE (Mean Absolute Error)* TARGET - lowest value

## RMSE RSQ MAE  
## lm 0.1304 0.3835 0.1036  
## randomForest 0.0386 0.9622 0.0286  
## ranger 0.0412 0.9599 0.031  
## kknn 0.0768 0.7982 0.0563  
## Cubist 0.0165 0.991 0.0115

Tuning the best-performing Cubist model in terms of the Committee and Neighbors parameters did not seem to improve the performance against our test set, suggesting some over-fitting to the training data. We settled on using the original cubist model to move forward with highlighting the most important variables and final predictions.

## Variable Importance

## # A tibble: 34 x 2  
## Variable Importance  
## <chr> <dbl>  
## 1 Bowl.Setpoint 61   
## 2 Pressure.Vacuum 59   
## 3 Balling.Lvl 58.5  
## 4 Temperature 50.5  
## 5 Alch.Rel 47   
## 6 Density 43   
## 7 Usage.cont 42.5  
## 8 Balling 41.5  
## 9 Oxygen.Filler 40.5  
## 10 Carb.Rel 36   
## # ... with 24 more rows

Based on our model, it appears that the strongest predictors for predicting the output pH of our product are:

1. Bowl.Setpoint
2. Pressure.Vacuum
3. Balling.lvl
4. Temperature
5. Alch.Rel

## Prediction

With our final Cubist model, we will predict values given our test set. See attached .xlsx file.

### Code Appendix

For future use, the following code is used to generate this report.

data <- read\_excel('StudentData.xlsx') %>%  
 mutate\_if(is.character, factor) %>%  
 rename(Brand = 'Brand Code')  
#head(data)  
data %>%  
 ggplot(aes(x=PH)) +  
 geom\_histogram(binwidth = .01) +  
 geom\_vline(xintercept = mean(data$PH, na.rm = TRUE), linetype ='dashed',  
 color = 'blue', size = 1)

pct\_null <- data.frame(do.call("rbind", map(data,  
 ~ mean(is.na(.)))))  
colnames(pct\_null) <- c('PCT\_NULL')  
totalNulls <- pct\_null %>%  
 mutate(VARIABLE = rownames(.)) %>%  
 arrange(desc(PCT\_NULL)) %>%  
 filter(PCT\_NULL > 0) %>%  
 dplyr::select(VARIABLE, PCT\_NULL)  
ggplot(totalNulls, aes(x = reorder(VARIABLE, PCT\_NULL), y = PCT\_NULL,  
 label = round(PCT\_NULL\*100, 1))) +   
 geom\_text(vjust = 0.5, hjust = -0.05)+  
 geom\_bar(stat = "identity") +  
 ggtitle("Variables with Missing Information") +  
 xlab("Statistic") + ylab("Percent Missing") +   
 scale\_y\_continuous(labels=scales::percent, limits = c(0,0.1)) +   
 coord\_flip() + expand\_limits(y = 1)

totalOutliers <- data.frame(  
 sapply(data %>% select(-PH,-`Brand`),   
 function(x){length(boxplot.stats(x)$out)/nrow(data)}))  
totalOutliers$VARIABLE\_NM <- rownames(totalOutliers)  
colnames(totalOutliers) <- c('PCT\_OUTLIERS', 'VARIABLE\_NM')  
ggplot(totalOutliers,  
 aes(x = reorder(VARIABLE\_NM, PCT\_OUTLIERS), y=PCT\_OUTLIERS,  
 label = round(PCT\_OUTLIERS\*100, 1))) +   
 geom\_text(vjust = 0.5, hjust = -0.05)+ geom\_bar(stat = "identity") +  
 ggtitle("Variables with Outliers") + xlab("Statistic") +  
 ylab("Percent of Data that is an Outlier") +   
 scale\_y\_continuous(labels=scales::percent, limits = c(0,0.2)) +   
 coord\_flip()

ggplot(data, aes(x=`Mnf Flow`, y = PH)) + geom\_point()  
ggplot(data, aes(x=`Hyd Pressure1`, y = PH)) + geom\_point()  
ggplot(data, aes(x=`Hyd Pressure3`, y = PH)) + geom\_point()

data <- data %>%  
 mutate(`Mnf Flow` = replace(`Mnf Flow`, `Mnf Flow`< 10, NA)) %>%  
 mutate(`Hyd Pressure1` = replace(`Hyd Pressure1`, `Hyd Pressure1`< 0, NA)) %>%  
 mutate(`Hyd Pressure3` = replace(`Hyd Pressure3`, `Hyd Pressure3`< 0, NA))

set.seed(312)  
data\_split <-initial\_split(data, prop=.80)  
train\_data <- training(data\_split)  
test\_data <- testing(data\_split)  
ph\_recipe <-  
 recipe(PH ~ ., data=data) %>%  
 step\_center(all\_numeric(), -all\_outcomes()) %>%  
 step\_BoxCox(all\_numeric(), -all\_outcomes()) %>%  
 step\_impute\_knn(all\_predictors()) %>%  
 step\_dummy(all\_nominal()) %>%  
 step\_naomit(all\_outcomes())  
ph\_training <- ph\_recipe %>%  
 prep() %>%  
 juice()  
ph\_testing <- ph\_recipe %>%  
 prep() %>%  
 bake(test\_data)

####Linear Regression Model##########################################  
lr\_model <- linear\_reg() %>%  
 set\_engine("lm") %>%  
 fit(PH ~ ., ph\_training)  
###Random Forest 'randomForest' Model################################  
rf\_model <- rand\_forest(trees = 100, mode = "regression") %>%  
 set\_engine("randomForest") %>%  
 fit(PH ~ ., ph\_training)  
###Random Forest 'ranger' Model######################################  
ranger\_model <- rand\_forest(trees = 100, mode = "regression") %>%  
 set\_engine("ranger") %>%  
 fit(PH ~ ., ph\_training)  
###k-Nearest Neighbors Model#########################################  
knn\_model <- nearest\_neighbor(neighbors = 10) %>%  
 set\_mode(mode = "regression") %>%  
 set\_engine("kknn") %>%  
 fit(PH ~., ph\_training)  
###Cubist Model######################################################  
cubist\_model <- cubist\_rules(committees = 10, neighbors = 2) %>%  
 set\_engine("Cubist") %>%  
 fit(PH ~., ph\_training)

model\_metrics <- function(models,test\_recipe){  
 mod\_methods <- c()  
 mod\_metrics <- NULL %>%  
 rbind(c('RMSE','RSQ','MAE'))  
 for (model in models){  
 mod\_methods <- c(mod\_methods,model$spec$engine)  
 prediction <- predict(model,test\_recipe)  
 pred\_metrics <- prediction %>%  
 bind\_cols(test\_recipe) %>%  
 metrics(truth = PH, estimate = .pred) %>%  
 pull(.estimate)  
 mod\_metrics <- rbind.data.frame(mod\_metrics,round(pred\_metrics,4))  
 }  
 mod\_metrics <- mod\_metrics %>%  
 row\_to\_names(row\_number = 1)  
 rownames(mod\_metrics) <- mod\_methods  
   
 return(mod\_metrics)  
}

models <- list(lr\_model,rf\_model,ranger\_model,knn\_model,cubist\_model)  
(model\_test <- model\_metrics(models,ph\_testing))

train\_x <- data.frame(ph\_training[,-which(names(ph\_training)=='PH')])  
train\_y <- ph\_training$PH  
cube\_model <- cubist(x = train\_x,  
 y = train\_y,  
 committees = 10,  
 neighbors = 2)  
vi(cube\_model)

eval\_data <- read\_excel('StudentEvaluation.xlsx') %>%  
 mutate\_if(is.character, factor) %>%  
 rename(Brand = 'Brand Code')  
eval\_data <- test\_data %>%  
 mutate(`Mnf Flow` = replace(`Mnf Flow`, `Mnf Flow`< 10, NA)) %>%  
 mutate(`Hyd Pressure1` = replace(`Hyd Pressure1`, `Hyd Pressure1`< 0, NA)) %>%  
 mutate(`Hyd Pressure3` = replace(`Hyd Pressure3`, `Hyd Pressure3`< 0, NA))  
final\_data <-   
 ph\_recipe %>%  
 prep(training= train\_data) %>%  
 bake(new\_data=eval\_data)

eval\_pred<- predict(cubist\_model, final\_data)

writexl::write\_xlsx(eval\_pred, "EvaluationPredcitions.")