DATA 624: Project 2

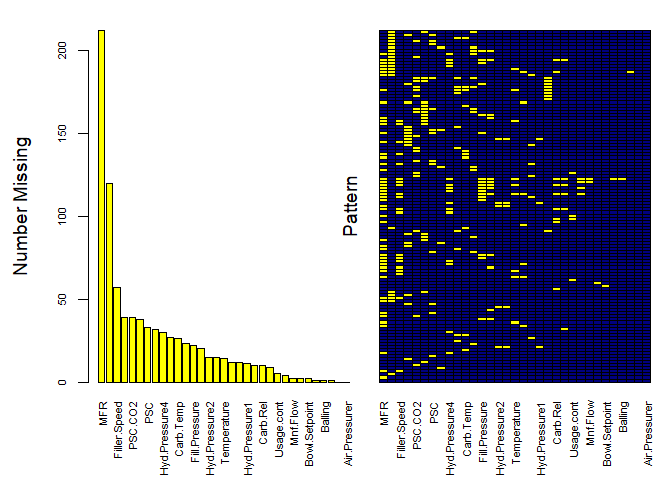
Author: Wei Zhou

PROBLEM STATEMENT

Due to new regulations by ABC Beverage, the company leadership requires that the production team have a better understanding of the manufacturing process, the predictive factors and their relationship to the pH of the beverages. Therefore, this project was an effort to find the optimal predictive variables related to the pH of the beverages and evaluate the accuracy of the same with rigorous statistical testing.

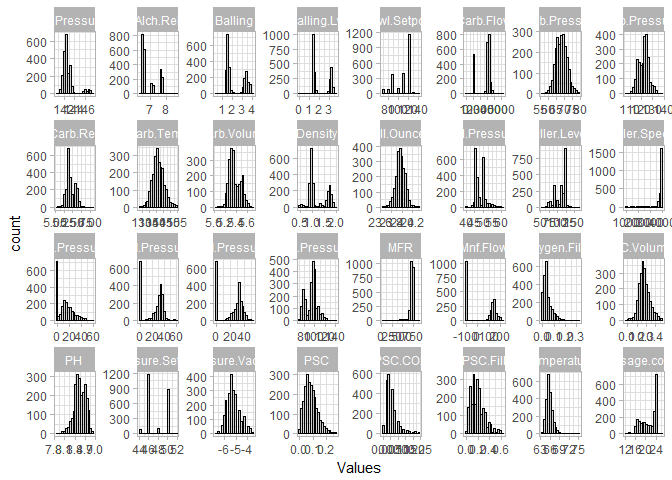
## Missing Values

As shown in the above summary, there are missing values across the variables -- the frequency and pattern of these missing values are presented below:



The variable MFR has over 200 missing values, and the variable Brand.Code is missing 120 values. Due to these high proportions of missingness, observations missing these variables are dropped.

Numeric variables are plotted below to determine the best method of imputation:

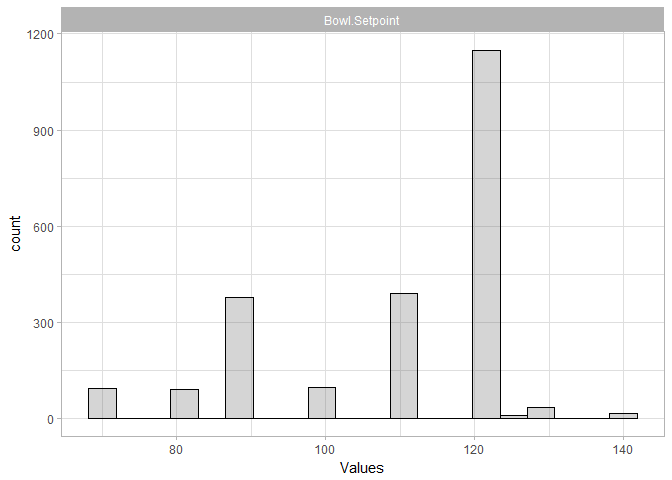


Many of the variables have normal or somewhat-normal distributions, and appear to be continuous variables. Many of the variables show varying levels of skewness, and a few of the skewed distributions follow an almost chi-squared or log-normal distribution. These are predictors (PSC.Distribution and PSC.C02.Distribution) where transformations should be considered.

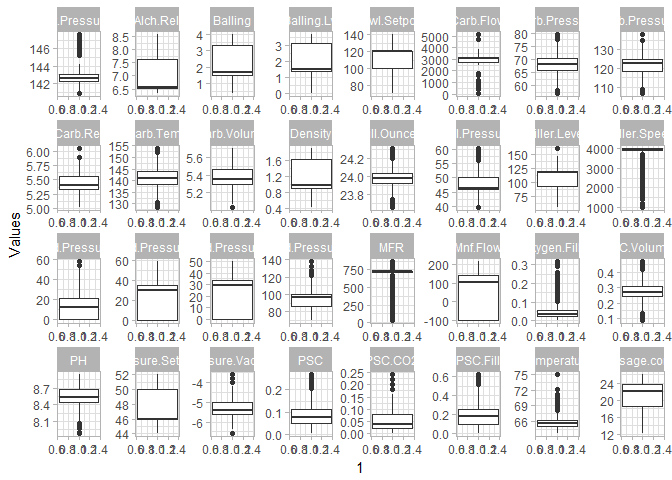
Another interesting pattern are the number of 0 values in the Hyd.Pressure(1-3) variables. The Hyd.Pressure does not follow this same pattern, so depending on the relationship between these variables and the target and/or other variables, they may be removed outright.

Some of the predictors appear to be discrete, rather than continuous distributions (Pressure.Setpoint, Alch.Rel), however, many of these variables are just constrained to a few values, but are still continuous. Out of all the variables, Bowl.Setpoint does seem to be a discrete distribution. A table of all values in the variable is below:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **70** | **80** | **90** | **100** | **110** | **120** | **122** | **126** | **130** | **134** | **140** |
| 93 | 89 | 376 | 95 | 390 | 1148 | 1 | 8 | 33 | 1 | 16 |

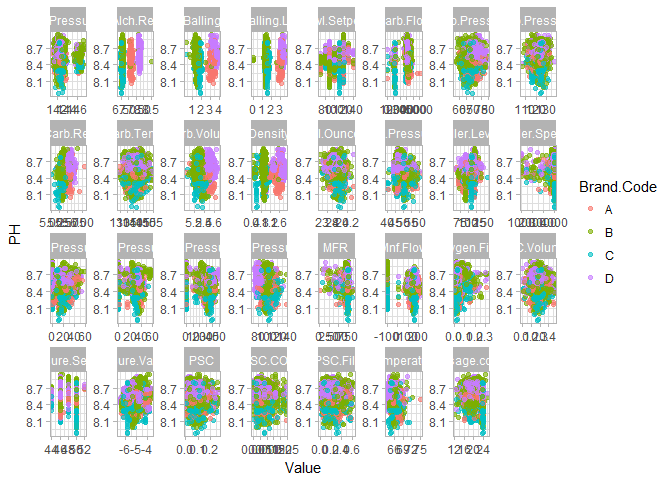


Boxplots provide an alternate view that allows the spread of the data to be viewed a little more clearly, as well as the presence of outliers and their quantities:



A number of these variables are so highly skewed, such as Filler.Speed, Oxygen.Filler, MFR, and Air.Pressurer, that many of the observations for that predictor are recognized as outliers. This suggests that imputing with the mean will not be accurate; an alternate method should be investigated.

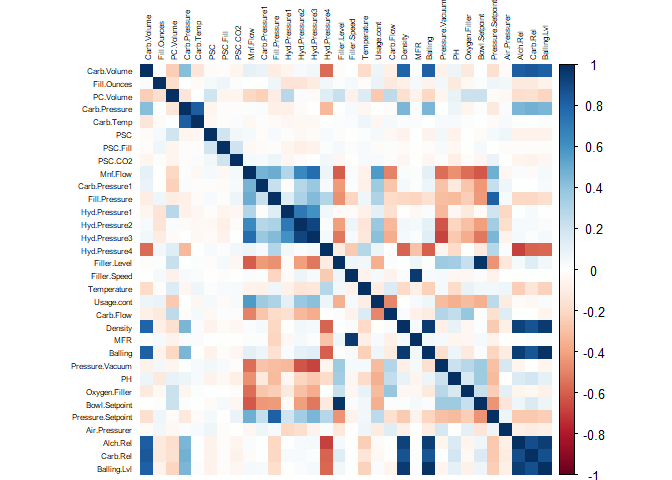
Before moving on to correlations between predictors, the relationship between the target variable PH and the other predictors is visualized using scatterplots.



Here we see there are no strong patterns (i.e. no defined linear pattern), but there are some relationships between PH and the other variables that may be more helpful in predicting the PH level of the soft drink. Predictors like Alch.Rel, Density, and Temperature all appear to have a stronger relationship with PH - this makes sense, as these variables have more to do with the make-up of the beverage itself, rather than the filling/bottling process.

Also interesting is the clustering of the Brand.Code within the plots. As correlations are investigated, this predictor does not have a strong relationship with the target, but combined with other predictors, may be more telling.

## Correlation

The correlation between the variables is investigated.

As the columns are organized in the data, some interesting patterns are present in the correlogram. Two areas show distinct positive correlations -- these are the predictors that have something to do with carbonation, and another area where different pressure levels correlate with each other. Another set of variables are negatively correlated with these pressure predictors, these have to do with the filling of the bottles, so this makes sense (Oxygen.Filler, Bowl.Setpoint, Pressure.Setpoint).

### Intercorrelated Predictors

Among the variables with positive correlations with each other, many of them are almost perfectly correlated with one another. We may want to consider removing some of these redundant variables, perhaps the ones that are less correlated to the target variable.

# Data Preprocessing

## Imputation

Due to the skewness of and relationship between predictors, they are imputed using k-nearest neighbors. Prior to this imputation, predictors are centered and scaled to avoid bias in predictive models, and highly-correlated predictors are removed. The preProcess function from the caret package is capable of performing all of these operations -- per the documentation for this function:

The operations are applied in this order: zero-variance filter, near-zero variance filter, correlation filter, Box-Cox/Yeo-Johnson/exponential transformation, centering, scaling, range, imputation, PCA, ICA then spatial sign.

## 

## Partitioning

With pre-processing complete, both predictor and response data are partitioned into a training and testing set.

# Model Creation

The pre-processed data is used to fit an array of models: linear models and tree based models.

## Linear Models

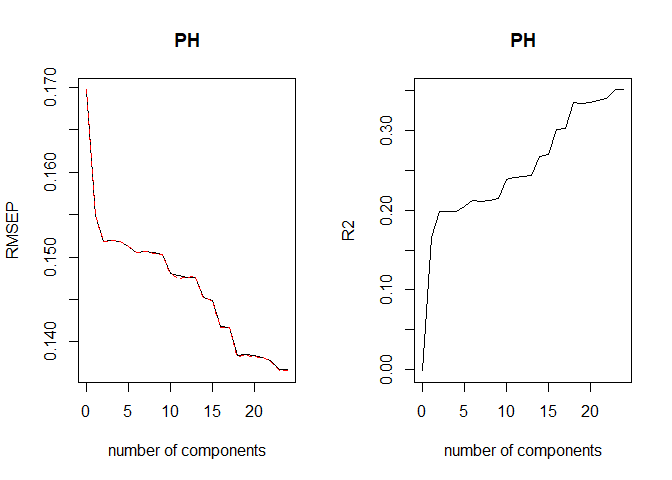
### Generalized Linear Model

Generalized linear models are extended versions of linear models which use a link function to describe the relation of the mean and the linear predictor, and a variance function. GLM can be used to model other distributions, such as binomial, Poisson, and exponential.

# fit model  
glmTune <- glm(PH ~ ., data = regressData, family = "quasipoisson")   
# get predictions & performance  
glmPred <- predict(glmTune, newdata = ph\_pred\_test\_num)  
glmPerf <- postResample(pred = glmPred, obs = ph\_resp\_test)

### Principal Components Regression Model

A principal components regression model seeks to lower the number of coefficients in a model by creating combinations of the variables that explain significant portions of the variance in the data. Each of the principal components generated are mutually orthogonal, so there is no risk of autocorrelation between predictors.

The performance of the PCR model is investigated across the number of components:

When doing PRC modeling, it is important to keep in mind that the main goal is to see a low cross validation error with a lower number of components than the number of variables in the dataset. In these results, it is clear that the number of components are too high for easy interpretation, resulting in no dimensionality reduction. Based on this, 18 principal components are used for prediction.

## Tree-Based Models

For this project we will see the constraints we encountered with our missing variables and the multiple regression. In the real world, the best model can be only as good as the variables measured by the study. We should also keep in mind that P-values can change based on the specific terms in the model. We can also appreciate that stepwise regression and best subsets regression are great tools and can get you close to the correct model. However, studies have found that they generally don't pick the correct model. With all these complications we agreed that tree-based methods are simple and useful for interpretation, but also Combining many trees can often result in dramatic improvements in prediction accuracy, at the expense of some lost interpretability. For this project we use 6 models below.

For consistency, the same training controls are used for all tree-based models:

### Rule-Based Model

Due to their prediction being based on average, small changes in the data can drastically affect the structure (and therefore predictions) of a simple tree. An approach that may overcome this is the M5 model tree, which uses a linear regression model to predict the repose at each terminal node. A rule-based regression tree simplifies this approach by removing redundant rules resulting from splits at different parts of the tree.

library(RWeka)  
set.seed(100)  
ruleTune <- M5Rules(PH ~ ., data = mutate(ph\_pred\_train\_num, PH = ph\_resp\_train))  
rulePred <- predict(ruleTune, newdata = ph\_pred\_test\_num)  
rulePerf <- postResample(pred = rulePred, obs = ph\_resp\_test)

### Random Forest Model

While bagging offers a reduction in variance, the trees used are not completely independent of each other, since each tree (and each split of each tree), considers all original predictors. To de-correlate the trees, random forests select a bootstrapped sample of the data, then randomly select a subset of predictors to partition the trees. Once these trees have been fit, the predictions of all models in the ensemble are averaged together.

library(randomForest)  
set.seed(100)  
rfTune <- train(x = ph\_pred\_train\_num, y = ph\_resp\_train,  
 method = "rf", trControl = mdl\_ctrl, importance = TRUE)  
rfPred <- predict(rfTune, newdata = ph\_pred\_test\_num)  
rfPerf <- postResample(pred = rfPred, obs = ph\_resp\_test)

### Boosted Tree Model

Gradient boosted trees function by attempting to reduce the residuals fit by models. Initially, predictions are simply provided by the mean of the response. A regression tree is then fit based on the residuals between the actual & predicted response. This tree is used to predict the residual, and the difference between the previous residual and the predicted residual is added to the previous residual. This process is repeated for a designated number of iterations, and only the final tree is considered. This is, however, still an ensemble method, as each model fit is informed by the model before it.

library(gbm)  
set.seed(100)  
boostTune <- train(x = ph\_pred\_train\_num, y = ph\_resp\_train,  
 method = "gbm", trControl = mdl\_ctrl, verbose = FALSE)  
boostPred <- predict(boostTune, newdata = ph\_pred\_test\_num)  
boostPerf <- postResample(pred = boostPred, obs = ph\_resp\_test)

### Cubist Model

Cubist models are an extension of the rule-based model above. Terminal leaves of trees contain models based on predictors used in previous splits, with intermediate linear models at each step of the tree. Predictions made using terminal leaf's model are "smoothed" by considering predictions from models in previous nodes of the tree. The tree is reduced to a set of rules, which, for simplification, are eliminated via pruning.

library(Cubist)  
set.seed(100)  
cubistTune <- train(x = ph\_pred\_train\_num, y = ph\_resp\_train,  
 method = "cubist", trControl = mdl\_ctrl)  
cubistPred <- predict(cubistTune, newdata = ph\_pred\_test\_num)  
cubistPerf <- postResample(pred = cubistPred, obs = ph\_resp\_test)

# Model Selection & Prediction

## Model Performance Comparison

The resampled RMSE & test set prediction RMSE is shown below for each of the models created:

RMSE performance for selected models

|  |  |  |
| --- | --- | --- |
|  | Resampled | Test |
| **glm** | Inf | 6.403 |
| **pcr** | Inf | 0.1489 |
| **rule** | Inf | 0.1288 |
| **Random forest** | 0.1012 | 0.1108 |
| **boost** | 0.1154 | 0.1252 |
| **cubist** | 0.09571 | 0.1047 |

Noting that the models trained using functions other than caret::train do not return a resampled RMSE, the best model based on both resampled & test RMSE is the **Cubist model**. To avoid any bias from looking only at a single metric, the r-squared is also investigated:

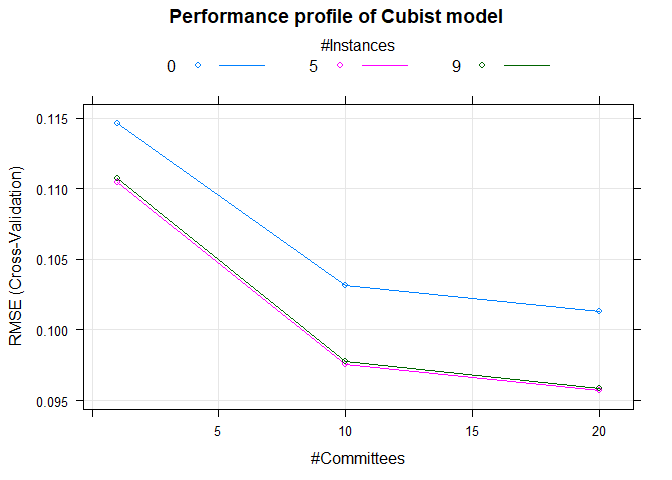
Rsquared performance for selected models

|  |  |  |
| --- | --- | --- |
|  | Resampled | Test |
| **glm** | Inf | 0.2868 |
| **pcr** | Inf | 0.2534 |
| **rule** | Inf | 0.4419 |
| **Random Forest** | 0.608 | 0.5898 |
| **boost** | 0.3736 | 0.4679 |
| **cubist** | 0.5675 | 0.6313 |

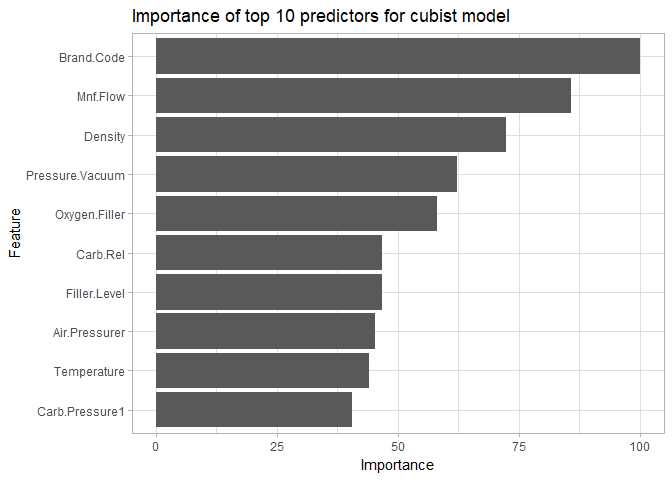
The Cubist model also performs best as judged by the r-squared metric on the test set. While the random forest model performs slightly better on resampled r-squared, it is not by a significant margin. The Cubist model will be used for prediction.

## Model Investigation

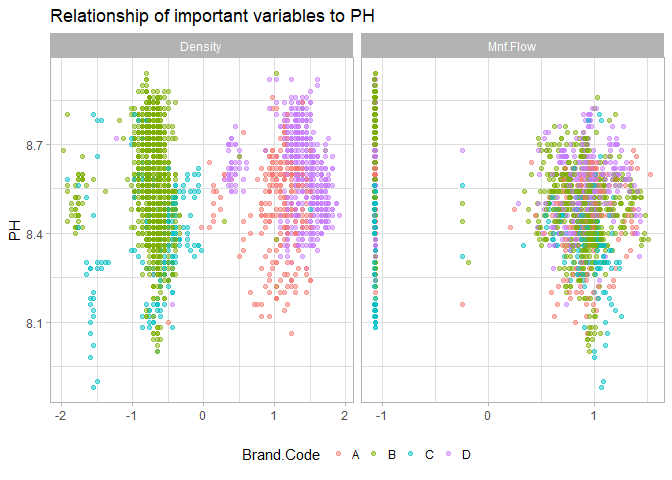
The performance profile of the selected Cubist across different tuning parameters is shown below:



For this model, the optimal RMSE of 0.0957 is obtained for the model with 5 neighbors & 20 committees. The 10 most important variables used in this final model are presented below:



The model is dominated by Brand.Code, with Mnf.Flow, Density, Pressure.Vacuum, and Oxygen.Filler registering as the next-most-important variables. The relationships between Brand.Code and the two most important numeric variables, Mnf.Flow and Density, with PH are presented below:



There is not an immediately apparent relationship between Mnf.flow and PH nor Brand.Code, but the relationship between Density and Brand.Code shows fairly clear clustering behavior. Within each cluster, there appears to be a weakly positive relationship with PH.

## Prediction of Additional Data

The Cubist model is finally used to predict additional data not part of the training or test set.

The new data is transformed in the same way as the original data:

The predicted data is included alongside this submission as *Team3\_Project2\_Fall624.csv.*