PROJECT 2: PREDICTING PH

DATA 624 - Predictive Analytics

Group 2

Group Members:

Juliann McEachern

10 December 2019

Contents

Introduction

This project is designed to evaluate production data from a beverage manufacturing company. Our assignment is to predict PH, a Key Performance Indicator (KPI), with a high degree of accuracy through predictive modeling. After thorough examination, we approached this task by splitting the provided data into training and test sets. We evaluated several models on this split and found that **what-ever-worked-best** method yielded the best results.

Each group member worked individually to create their own solution. We built our final submission by collaboratively evaluating and combining each others' approaches. Our introduction should further outline individual responsibilities. For example, **so-and-so** was responsible for **xyz task**.

For replication and grading purposes, we made our code avaliable in the appendix section. This code, along with the provided data, score-set results, and individual contributions, can also be accessed through our group github repository:

- Pretend I'm a working link to R Source Code
- Pretend I'm a working link to Provided Data
- Pretend I'm a working link to Excel Results
- Pretend I'm a working link to Individual Work

1 Data Exploration

The beverage manufacturing production dataset contained 33 columns/variables and 2,571 rows/cases. In our initial review, we found that the response variable, PH, had four missing observations.

We also identified that 94% of the predictor variables had missing data points. Despite this high occurance, the NA values in the majority of these predictors accounted for less than 1% of the total observations. Only eleven variables were missing more than 1% of data.

Table 1.1: Variables with Highest Frequency of NA 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

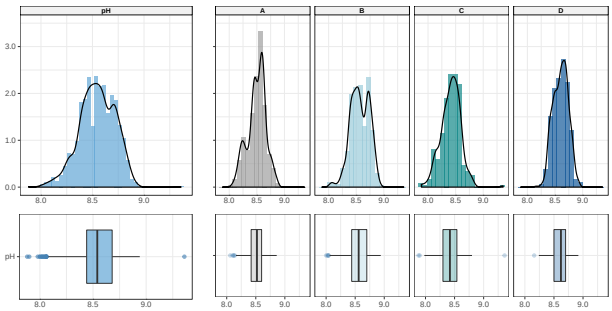
Response Variable

Understanding the influence pH has on our predictors is key to building an accurate predictive model. pH is a measure of acidity/alkalinity that must conform in a critical range. The value of pH ranges from 0 to 14, where 0 is acidic, 7 is neutral, and 14 is basic.

Figure 1.1 shows that our response distribution follows a somewhat normal pattern and is centered around 8.5. The histogram for pH is bimodal in the aggregate, but varies by brand. The boxplot view allows us to better visualize the effect outliers have on the skewness within our target variable.

Brand A has a negatively skewed, multimodal distribution, which could be suggestive of several distinct underlying re-

Fig. 1.1: Distribution of Response Variable: pH

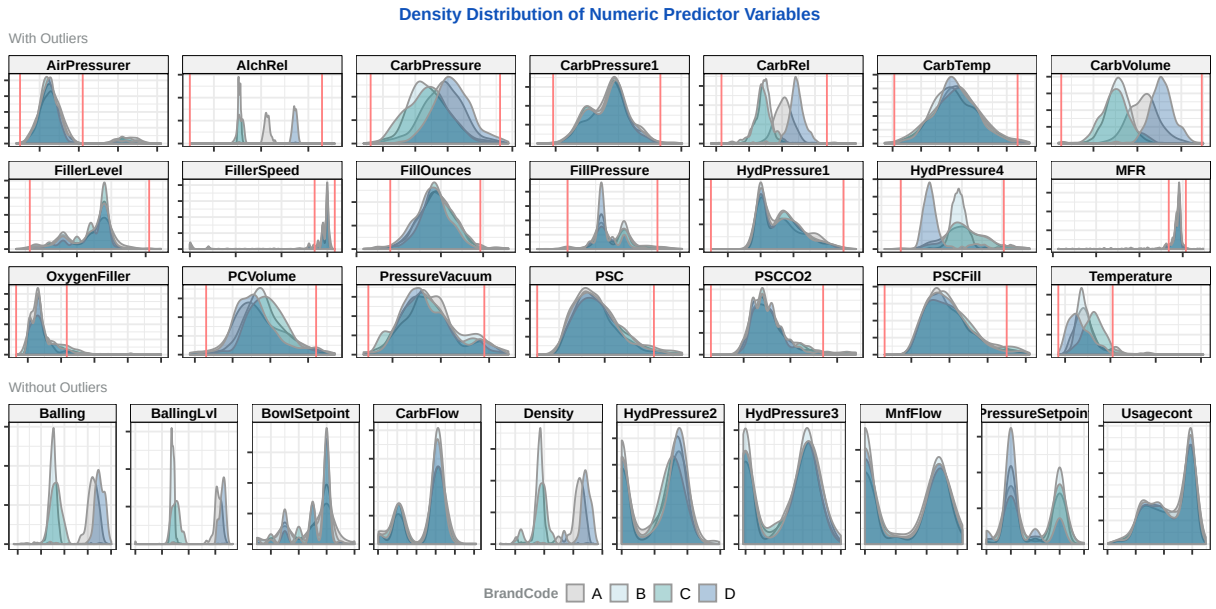


sponse patterns or a higher degree of variation in pH response for this brand. The density plot and histogram for Brand B show two bimodal peaks with a slight positive skew. These peaks indicate that this brand has two distinct response values that occur more frequently. The distribution for Brand C and D are both more normal, with a slight negative skew. Brand D has the highest median pH value and Brand C has the lowest. Brand C also appears to have the largest spread of pH values.

Predictor Variables

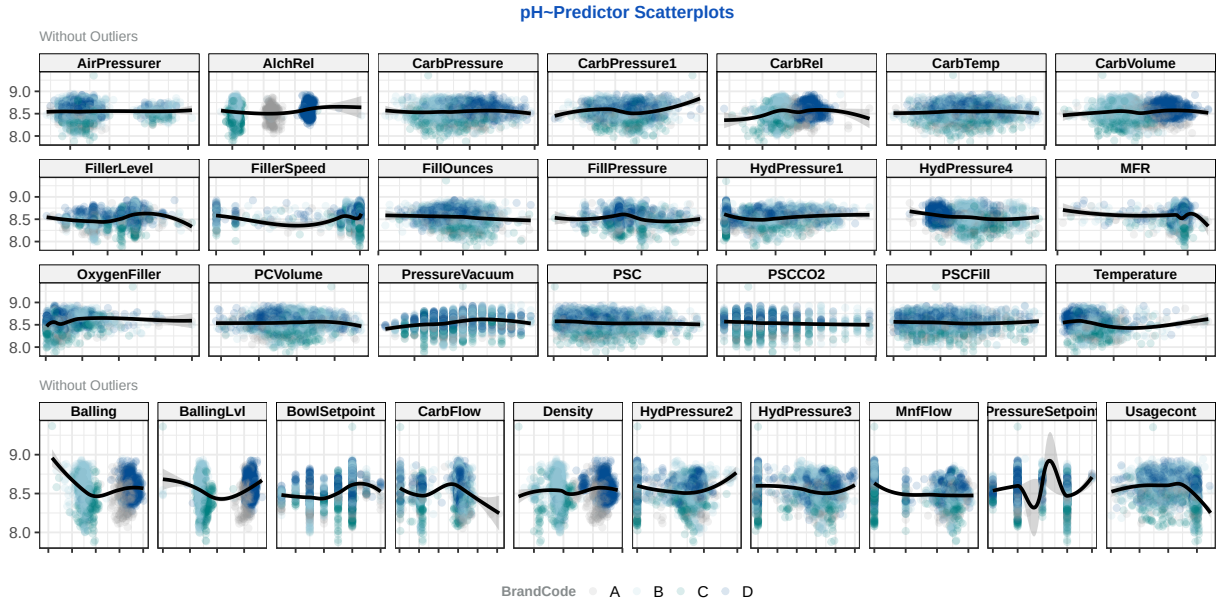
We examined the density of our variables to visualize the distribution of the predictors. Many of these variables contain outliers and present with a skewed distribution. The outliers fall outside the red-line boundaries, and highlight which predictors have heavier tails.

The density plots also contain an overlay of the only categorical indicator, `BrandCode`. This view shows us that some variables, including `AlchRel`, `CarbRel`, `CarbVolume`, `HydPressure4`, and `Tempature`, are strongly influenced by brand type.

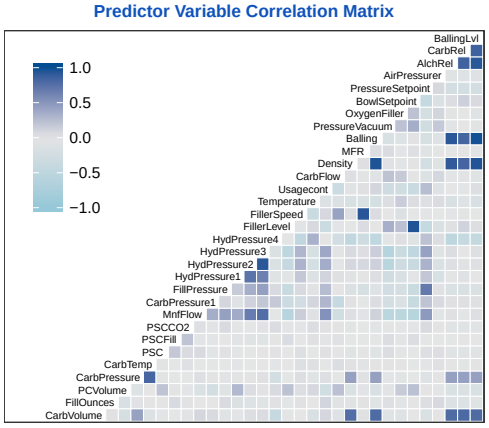


We also looked at the relationship of our predictors against the response variable below. There are a few predictors that have a weak, linear association with our response variable. However, most of the indicators show no strong patterns. Given these trends, we do not expect linear modeling to provide optimal predictions for pH.

This view helps us further visualize the effect `BrandCode` has on our predictor and pH values. For example, `AlchRel` shows distinct `BrandCode` groupings. Other variables, such as `PSCO2`, `BowlSetpoint`, `MinFlow`, and `PressureSetup` show unique features likely related to system processes.



Lastly, we examined collinearity measures between our numeric predictors and found that several of these variables were heavily related, with correlation values exceeding ± 0.7.



2 Data Preparation

In our exploration, we detected missing data, extreme outliers, and multicollinearity. We kept these factors in mind and applied strategic transformations when preparing our models to evaluate their performance with and without normalization changes.

Train/Test Splits:

We divided the production dataset using an 80/20 split to create a train and test set. All models incorporated k-folds cross-validation set at 10 folds to protect against overfitting the data. We set up unique model tuning grids to find the optimal parameters for each regression type to ensure the highest accuracy within our predictions.

Data Imputation:

We applied a Multiple Imputation by Chained Equations (MICE) algorthim to predict the missing data using sequential regression.

This method filled in all incomplete cases, including `BrandCode`, our one unordered categorical variable.

Pre-Processing:

Due to the strong non-normality exhibited in the data, we tested our models using three different approaches: (1) No pre-processing techniques, (2) centering and Scaling, and (3) removing zero (and near-zero) variance and box-cox, centering, and scaling transformations.

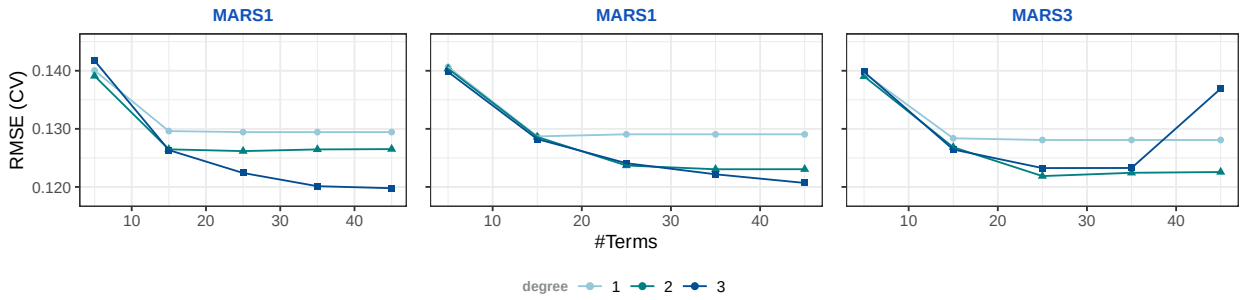
3 Modeling

We compared the effectiveness of the Multivariate Adaptive Regression Splines (MARS) to the Elastic Net (eNet) model.

Multivariate Adaptive Regression Splines

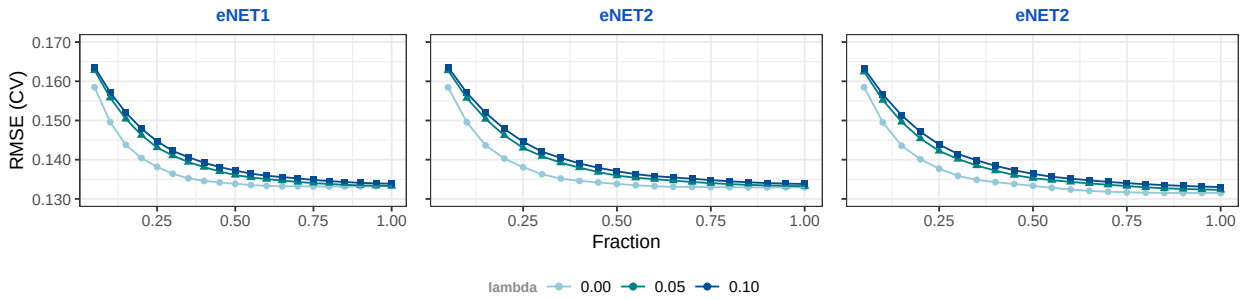
MARS modeling was selected to assess the non-linear features in our data. This method uses a weighted sum to models non-linearities and interactions between variables. The model assesses cut-points between features that create the smallest error and prunes insignificant points to improve model accuracy.

Our RMSE Cross Validation plots show us that pre-processing transformations did not have improve the MARS model. The model performed best on our training data when no transformations were applied.



Elastic Net

The Elastic Net model was used as it can handle a large number of predictor variables. It combines ridge and lasso regression techniques to reduce the size of coefficients. Our RMSE Cross Validation plots show us that the best tune for both eNET were also similiar, with eNET3 performing the best. The third version of this model incorporated all pre-processing methods.



4 Regression Analysis

Accuracy

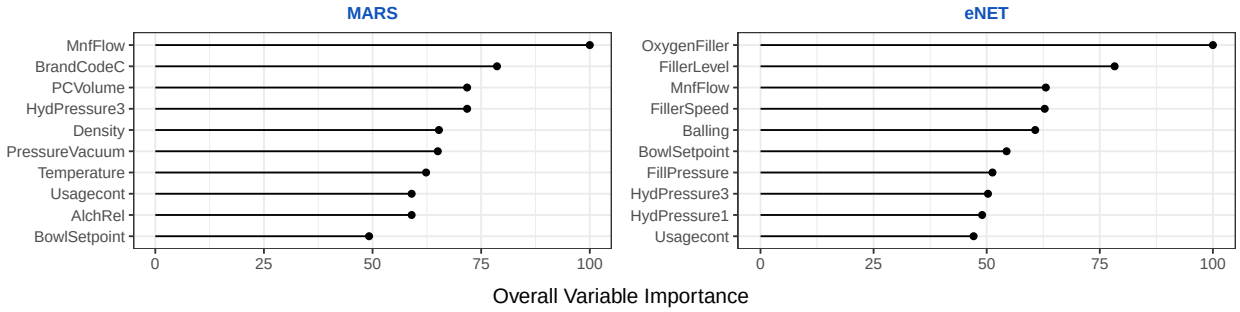
The MARS model performed the best with the lowest accuracy scores. The MARS model fit the data the best with the lowest variation between predicted and observed data for both the training and the test set.

Table 4.1: Accuracy Measures

	MARS_Train	MARS_Test	eNET_Train	eNET_Test
RMSE	**0.11978**	**0.12432**	0.13150	0.13506
Rsquared	**0.52474**	**0.49261**	0.41769	0.39995
MAE	**0.08950**	**0.09487**	0.10261	0.10643
MAPE	**0.01157**	**0.01114**	0.01280	0.01249

Variable Importance

Variable importance varied greatly between the two models. The predictors, `MnfFlow`, `HydPressure3`, `Useagecont`, and `BowelSetpoint`, both ranked in the top 10 important variables for MARS and eNET model.



5 Conclusion

This section should contain final thoughts and save/discuss our student evaluation predictions.

Appendix

```
library(tidyverse)
library(readxl)
library(psych)
library(mice)
library(xtable)
library(caret)
library(data.table)
library(Metrics)
```

```r
# SEEDING
set.seed(58677)

# 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 Import data
StudentData <- read_xlsx("~/GitHub/CUNY_DATA_624/Project_Two/data/StudentData.xlsx")
StudentEvaluation <- read_xlsx("~/GitHub/CUNY_DATA_624/Project_Two/data/StudentEvaluation.xlsx")

## Data Tidying
names(StudentData) <- gsub(" ", "", names(StudentData))
StudentData <- StudentData %>% mutate(BrandCode = as.factor(BrandCode))

## Summary Stats
summary_stats <- describe(StudentData)

## 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 = 0.25, na.rm = T) - 1.5 * IQR(value, na.rm = T),
 outlier_upper = 1.5 * IQR(value, na.rm = T) + quantile(value,
  probs = 0.75, na.rm = T), outlier = ifelse(value <
  outlier_lower, "TRUE", ifelse(value > outlier_upper,
  "TRUE", "FALSE")))
outlier_with <- outliers %>% filter(any(outlier == "TRUE")) %>%
 filter(!is.na(BrandCode))
outlier_wo <- outliers %>% filter(all(outlier != "TRUE")) %>%
 filter(!is.na(BrandCode))
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, `%`)
```

```r
## Correlation
cor <- StudentData %>% select(-PH, -BrandCode) %>% cor(use = "pairwise.complete.obs")
cor_freq <- findCorrelation(cor, 0.7, names = T)
cor_flat <- flattenCorrMatrix(cor) %>% arrange(row, desc(cor)) %>%
 filter(cor > 0.75 | cor < -0.75)
cor_flat_left <- cor_flat %>% slice(1:10) %>% mutate(id = row_number())
cor_flat_right <- cor_flat %>% slice(11:20) %>% mutate(id = row_number())

# DATA PREPARATION

## Imputation
init <- mice(StudentData, maxit = 0)
meth <- init$method
predM <- init$predictorMatrix
meth[c("BrandCode")] = "polyreg"
imputed <- mice(StudentData, method = meth, predictorMatrix = predM,
 m = 5, printFlag = F)
BevData <- complete(imputed)

## Train/Test Splits
trainingRows <- createDataPartition(BevData$PH, p = 0.8,
 list = FALSE)

## Split Train/Test Data
train <- BevData[trainingRows, ]
test <- BevData[-trainingRows, ]

# MODELING
tl <- 5
trC <- trainControl(method = "cv", number = 10, savePredictions = T)

## MARS
mars_grid <- expand.grid(degree = 1:3, nprune = seq(5, 50,
 by = 10))
mars_fit1 <- train(PH ~ ., data = train, method = "earth",
 tuneGrid = mars_grid, trControl = trC, tuneLength = tl)
mars_fit2 <- train(PH ~ ., data = train, method = "earth",
 preProcess = c("center", "scale"), tuneGrid = mars_grid,
 trControl = trC, tuneLength = tl)
mars_fit3 <- train(PH ~ ., data = train, method = "earth",
 preProcess = c("nzv", "zv", "BoxCox", "center", "scale"),
 tuneGrid = mars_grid, trControl = trC, tuneLength = tl)

mars_test_pred1 <- predict(mars_fit1, newdata = test)
mars_test_pred2 <- predict(mars_fit2, newdata = test)
mars_test_pred3 <- predict(mars_fit2, newdata = test)

## eNET
enet_grid <- expand.grid(lambda = c(0, 0.05, 0.1), fraction = seq(0.05,
 1, length = 20))
enet_fit1 <- train(PH ~ ., data = train, method = "enet",
 metric = "RMSE", tuneGrid = enet_grid, tuneLength = tl,
 trControl = trC)
```

```
enet_fit2 <- train(PH ~ ., data = train, method = "enet",
 preProcess = c("center", "scale"), metric = "RMSE", tuneGrid = enet_grid,
 tuneLength = tl, trControl = trC)
enet_fit3 <- train(PH ~ ., data = train, method = "enet",
 preProcess = c("nzv", "zv", "BoxCox", "center", "scale"),
 metric = "RMSE", tuneGrid = enet_grid, trControl = trC,
 tuneLength = tl)

enet_test_pred1 <- predict(enet_fit1, newdata = test)
enet_test_pred2 <- predict(enet_fit2, newdata = test)
enet_test_pred3 <- predict(enet_fit2, newdata = test)


# ACCURACY
MARS_MAPE_TRN <- Metrics::mape(mars_fit1$pred$obs, mars_fit1$pred$pred)
eNET_MAPE_TRN <- Metrics::mape(enet_fit3$pred$obs, enet_fit3$pred$pred)
MARS_MAPE_TST <- Metrics::mape(test$PH, mars_test_pred1)
eNET_MAPE_TST <- Metrics::mape(test$PH, enet_test_pred3)
MARS_PERF_TRN <- mars_fit1$results %>% as.data.frame() %>%
 filter(RMSE == min(RMSE)) %>% select(RMSE, Rsquared,
 MAE) %>% distinct() %>% mutate(Variable = "MARS_Train") %>%
 column_to_rownames("Variable") %>% t()
eNET_PERF_TRN <- enet_fit3$results %>% as.data.frame() %>%
 filter(RMSE == min(RMSE)) %>% select(RMSE, Rsquared,
 MAE) %>% distinct() %>% mutate(Variable = "eNET_Train") %>%
 column_to_rownames("Variable") %>% t()
MARS_PERF_TST <- postResample(pred = mars_test_pred1, obs = test$PH)
eNET_PERF_TST <- postResample(pred = enet_test_pred3, obs = test$PH)
bind1 <- cbind(MARS_PERF_TRN, MARS_Test = MARS_PERF_TST,
 eNET_PERF_TRN, eNET_Test = eNET_PERF_TST)
bind2 <- cbind(MARS_MAPE_TRN, MARS_MAPE_TST, eNET_MAPE_TRN,
 eNET_MAPE_TST)
row.names(bind2) <- "MAPE"
Tbl_Accuracy <- rbind(bind1, bind2)

# VARIABLE IMPORTANCE
MARS_VarImp <- varImp(mars_fit1, scale = T)
eNET_VarImp <- varImp(enet_fit3, scale = T)
```