DATA624 - Project 2

Glen Dale Davis & Tora Mullings

2023-12-17

Packages:

```
library(tidyverse)
library(httr)
library(readxl)
library(DataExplorer)
library(psych)
library(knitr)
library(snakecase)
library(RColorBrewer)
library(VIM)
library(ggcorrplot)
library(caret)
library(randomForest)
library(cowplot)
library(car)
library(MASS)
select <- dplyr::select
library(earth)
library(rminer)
library(writexl)
```

```
cur_theme <- theme_set(theme_classic())
palette <- brewer.pal(n = 12, name = "Paired")
greys <- brewer.pal(n = 9, name = "Greys")
```

Introduction:

New regulations require ABC Beverage to understand our manufacturing process and the predictive factors. We need to be able to report to leadership our predictive model of `PH`.

We load the historical dataset provided, as well as the evaluation dataset we will later make predictions on.

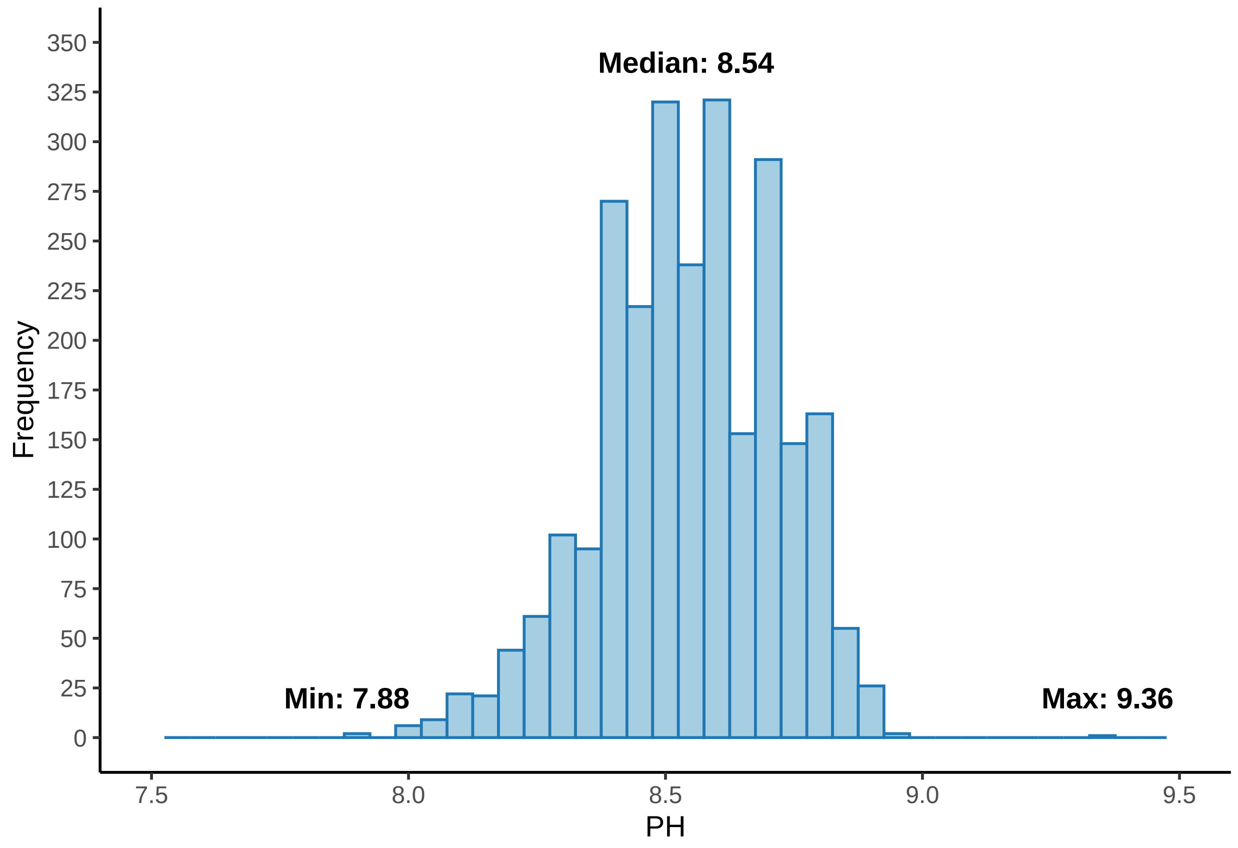
```
my_url1 <- "https://github.com/geedoubledee/data624_project2/raw/main/StudentData.xlsx"
temp <- tempfile(fileext = ".xlsx")
req <- GET(my_url1, authenticate(Sys.getenv("GITHUB_PAT"), ""),
           write_disk(path = temp))
main_df <- readxl::read_excel(temp)
colnames(main_df) <- to_screaming_snake_case(colnames(main_df))
```

```
my_url2 <- "https://github.com/geedoubledee/data624_project2/raw/main/StudentEvaluation.xlsx"
temp <- tempfile(fileext = ".xlsx")
req <- GET(my_url2, authenticate(Sys.getenv("GITHUB_PAT"), ""),
           write_disk(path = temp))
eval_df <- readxl::read_excel(temp)
colnames(eval_df) <- to_screaming_snake_case(colnames(eval_df))
```

Exploratory Data Analysis:

We take a look at the distribution for the response variable and a summary of it.

```
annotations <- data.frame(x = c(min(main_df$PH, na.rm = TRUE),
                                round(median(main_df$PH, na.rm = TRUE), 2),
                                max(main_df$PH, na.rm = TRUE)),
                          y = c(20, 340, 20),
                          label = c("Min:", "Median:", "Max:"))
p0 <- main_df |>
    ggplot(aes(x = PH)) +
    geom_histogram(binwidth = 0.05, color = palette[2], fill = palette[1]) +
    geom_text(data = annotations,
              aes(x = x, y = y, label = paste(label, x)),
              size = 4, fontface = "bold") +
    labs(y = "Frequency") +
    scale_x_continuous(limits = c(7.5, 9.5), breaks = seq(7.5, 9.5, 0.5)) +
    scale_y_continuous(limits = c(0, 350), breaks = seq(0, 350, 25))
p0
```



```r
summary(main_df$PH)
```

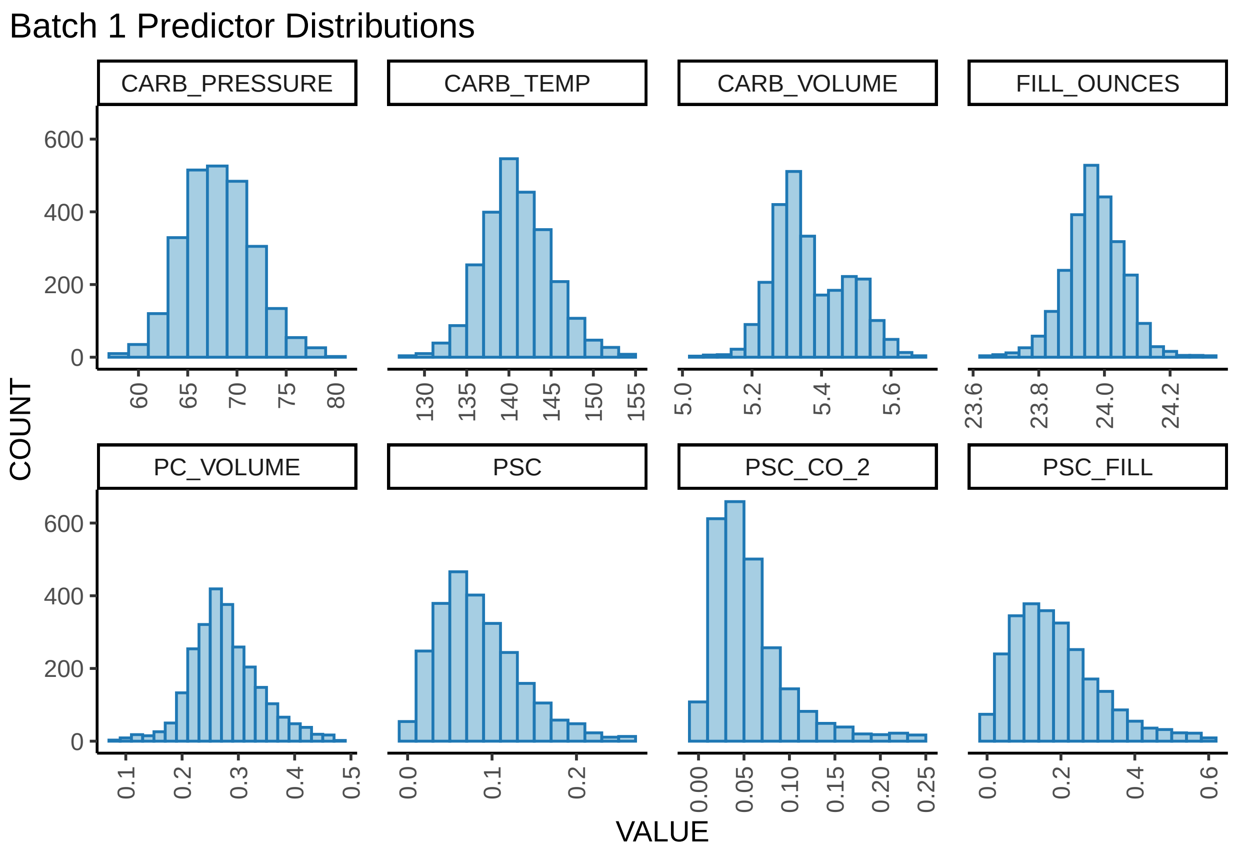
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##   7.880   8.440   8.540   8.546   8.680   9.360       4
```

The median PH value is 8.54 and ranges between 7.88 and 9.36. 50 percent of observations have values between 8.44 and 8.68 though. There are 4 observations with missing PH values. This is a small enough percentage of our total observations to justify simple list-wise deletion. We lose little by removing these observations, and we would gain little by imputing them.

```r
main_df <- main_df |>
    filter(!is.na(PH))
```

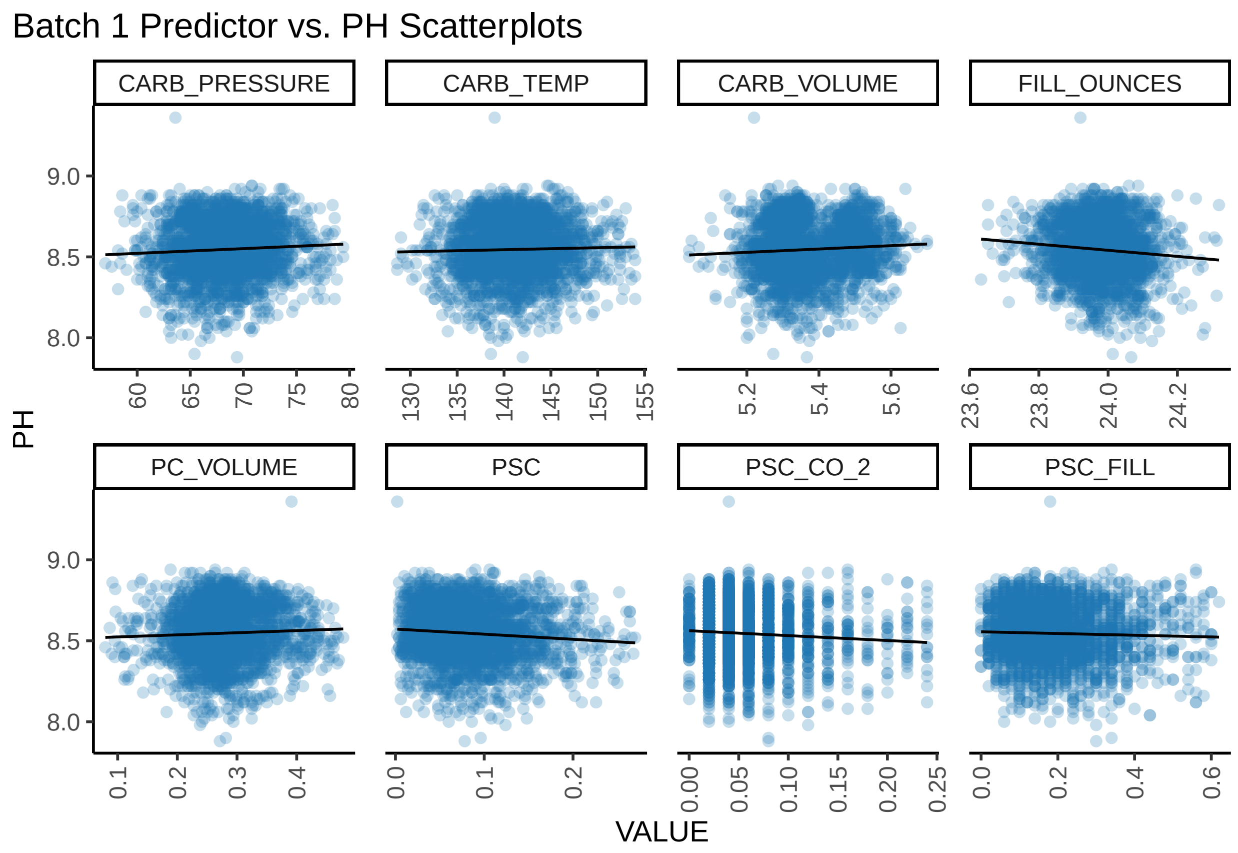
We take a look at histograms for the numeric predictor variables, as well as scatterplots of each numeric predictor and the response, in batches since there are so many of them.

```r
non_numeric <- c("BRAND_CODE")
all_numeric <- colnames(main_df |> select(-all_of(c("PH", non_numeric))))
n = 8
all_numeric_chunks <- split(all_numeric, ceiling(seq_along(all_numeric)/n))
remove <- c("PH", non_numeric)
pivot_df <- main_df |>
    select(-all_of(remove)) |>
    pivot_longer(cols = all_of(all_numeric), names_to = "PREDICTOR",
                 values_to = "VALUE")
p1a <- pivot_df |>
    filter(PREDICTOR %in% all_numeric_chunks[[1]]) |>
    ggplot(aes(x = VALUE)) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "CARB_PRESSURE"),
                   binwidth = 2) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "CARB_TEMP"),
                   binwidth = 2) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "CARB_VOLUME"),
                   binwidth = 0.04) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "FILL_OUNCES"),
                   binwidth = 0.04) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "PC_VOLUME"),
                   binwidth = 0.02) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "PSC"),
                   binwidth = 0.02) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "PSC_CO_2"),
                   binwidth = 0.02) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "PSC_FILL"),
                   binwidth = 0.04) +
    facet_wrap(vars(PREDICTOR), ncol = 4, scales = "free_x") +
    labs(y = "COUNT",
         title = "Batch 1 Predictor Distributions") +
    theme(panel.spacing.x = unit(4, "mm"),
          axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5),
          plot.title.position = "plot")
p1a
```



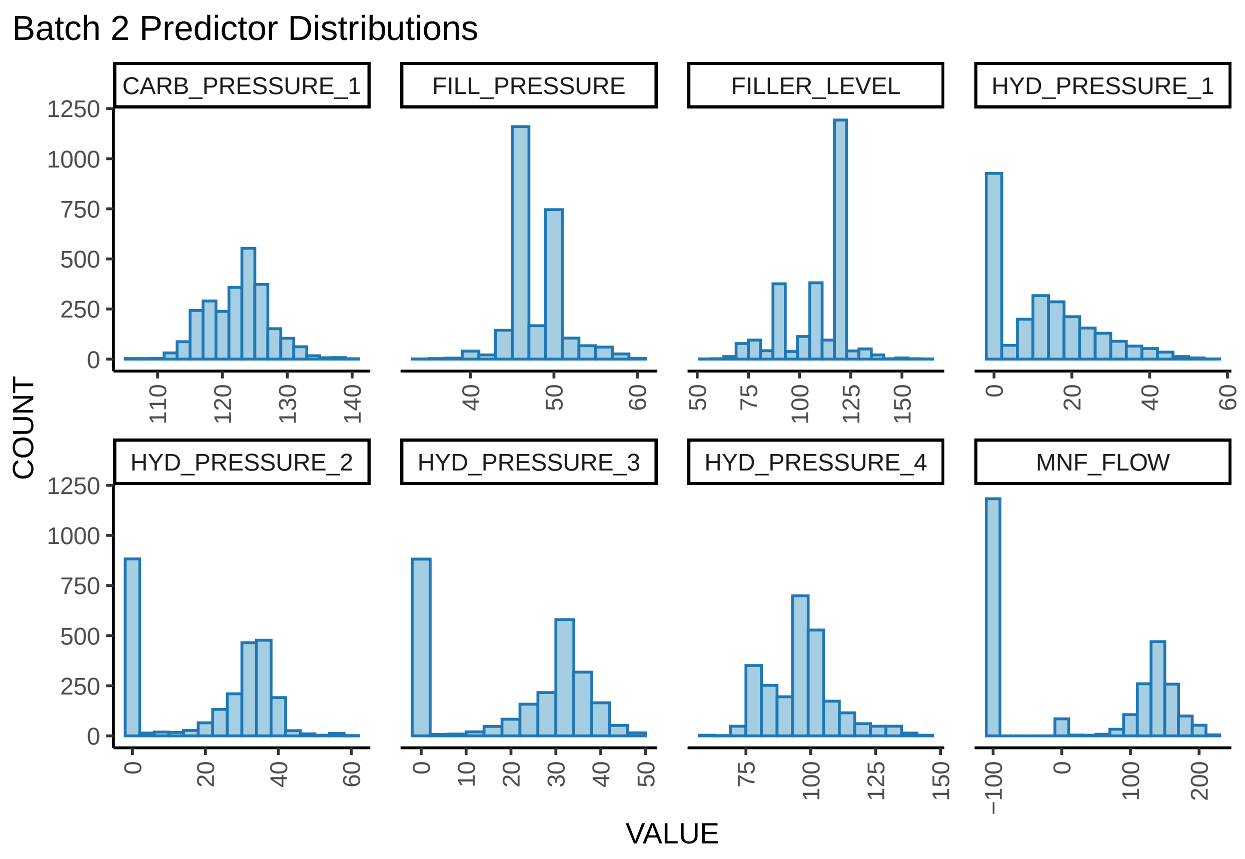
In the first batch of numeric predictors, we see that `PSC`, `PSC_CO_2`, and `PSC_FILL` are all right-skewed, and the distribution for `CARB_VOLUME` is multimodal. The distributions for the rest of the variables are nearly normal.

```r
sel <- c("PH", all_numeric_chunks[[1]])
p1b <- main_df |>
    select(all_of(sel)) |>
    pivot_longer(cols = all_of(all_numeric_chunks[[1]]), names_to = "PREDICTOR",
                 values_to = "VALUE") |>
    ggplot(aes(x = VALUE, y = PH)) +
    geom_point(color = palette[2], fill = palette[1], alpha = 0.25) +
    geom_smooth(method = "lm", color = "black", linewidth = 0.5, se = FALSE) +
    facet_wrap(~PREDICTOR, ncol = 4, scales = "free_x") +
    labs(title = "Batch 1 Predictor vs. PH Scatterplots") +
    theme(panel.spacing.x = unit(4, "mm"),
          axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5),
          plot.title.position = "plot")
p1b
```



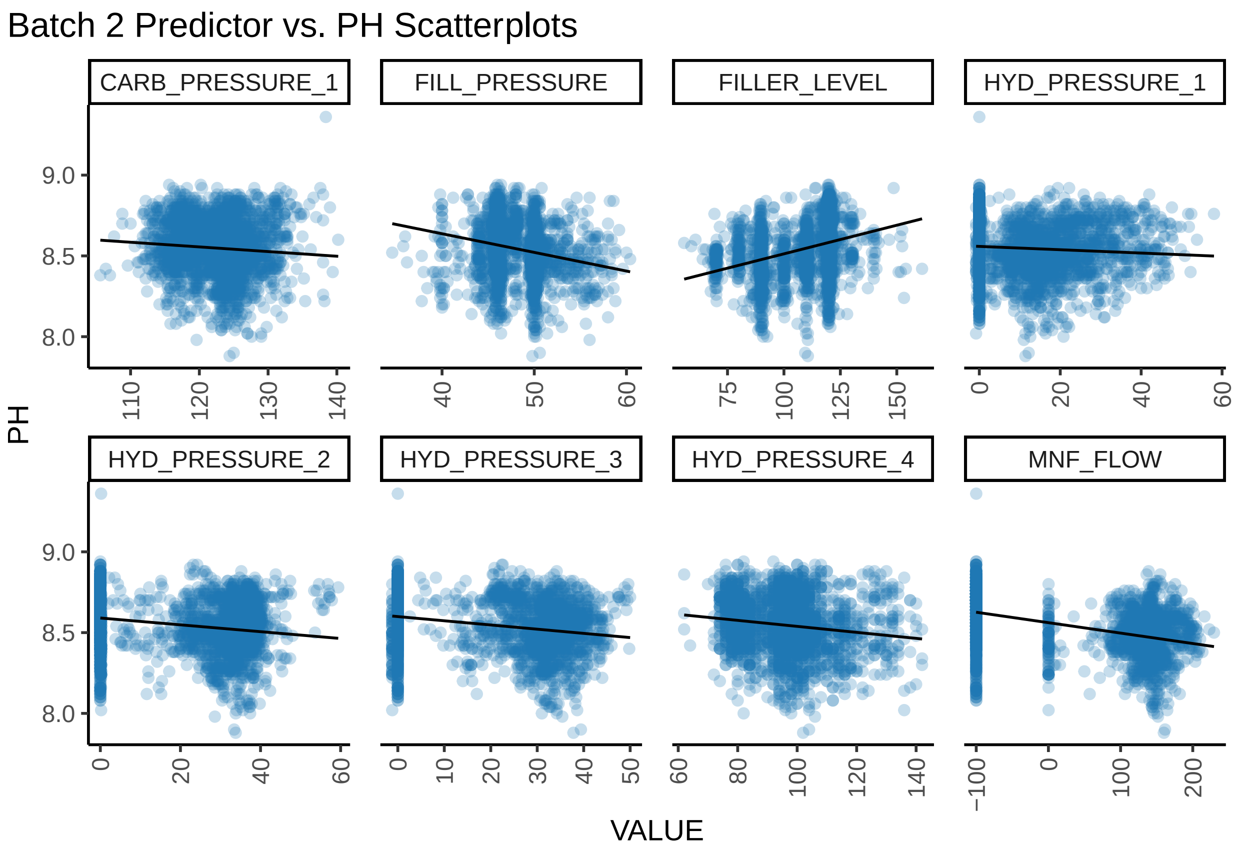
There are no linear relationships discernable from these scatterplots. There may be two clusters in `CARB_VOLUME`.

```r
p2a <- pivot_df |>
    filter(PREDICTOR %in% all_numeric_chunks[[2]]) |>
    ggplot(aes(x = VALUE)) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "CARB_PRESSURE_1"),
                   binwidth = 2) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "FILL_PRESSURE"),
                   binwidth = 2) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "FILLER_LEVEL"),
                   binwidth = 6) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "HYD_PRESSURE_1"),
                   binwidth = 4) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "HYD_PRESSURE_2"),
                   binwidth = 4) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "HYD_PRESSURE_3"),
                   binwidth = 4) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "HYD_PRESSURE_4"),
                   binwidth = 6) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "MNF_FLOW"),
                   binwidth = 20) +
    facet_wrap(vars(PREDICTOR), ncol = 4, scales = "free_x") +
    labs(y = "COUNT",
         title = "Batch 2 Predictor Distributions") +
    theme(panel.spacing.x = unit(4, "mm"),
          axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5),
          plot.title.position = "plot")
p2a
```



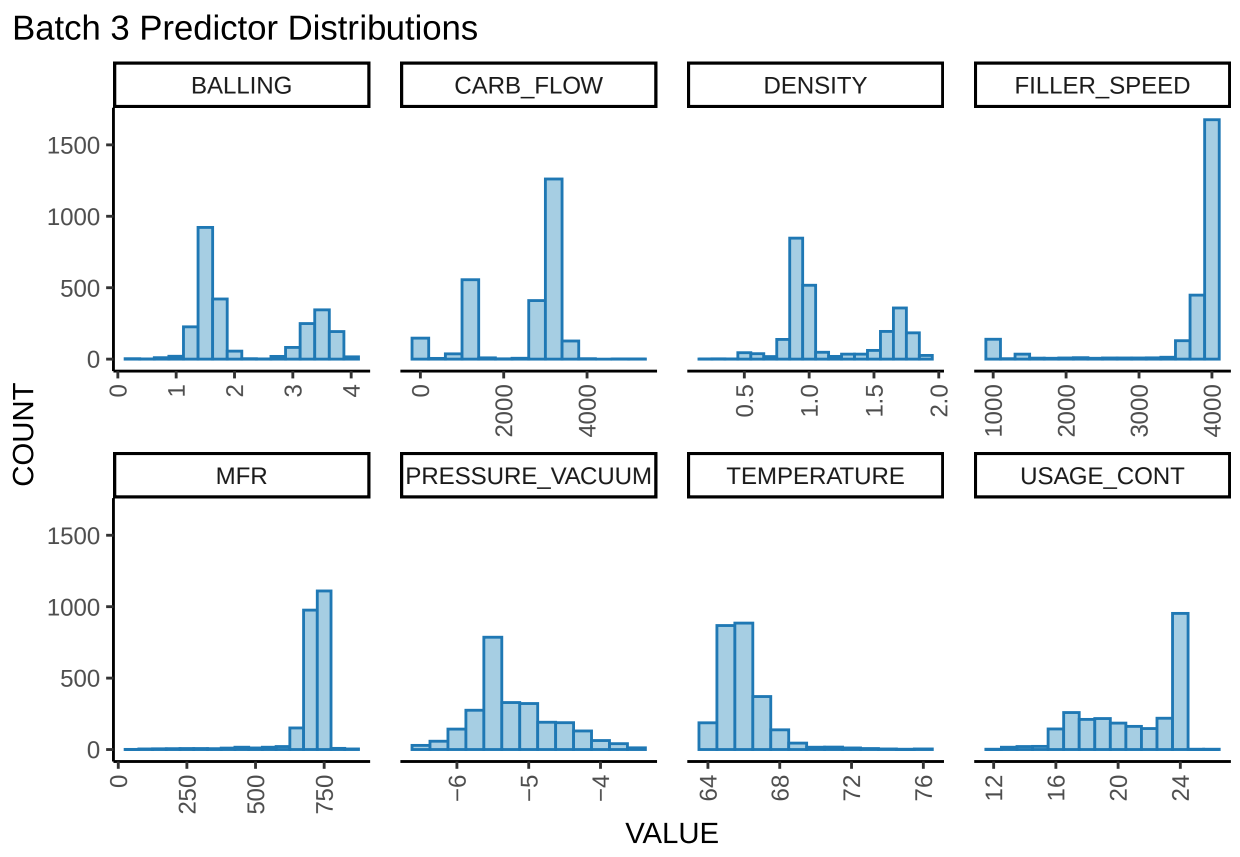
In the second batch of numeric predictors, we see that `HYD_PRESSURE_1`, `HYD_PRESSURE_2`, and `HYD_PRESSURE_3` are heavy with zero value observations, skewing their distributions. Most observations for `MNF_FLOW` are around -100, and its distribution might be degenerate. We'll check for degeneracy for this variable and any others shortly. `FILL_PRESSURE` and `FILLER_LEVEL` are multimodal. `HYD_PRESSURE_4` is right-skewed. `CARB_PRESSURE_1` has the only nearly normal distribution here.

```
sel <- c("PH", all_numeric_chunks[[2]])
p2b <- main_df |>
    select(all_of(sel)) |>
    pivot_longer(cols = all_of(all_numeric_chunks[[2]]), names_to = "PREDICTOR",
                 values_to = "VALUE") |>
    ggplot(aes(x = VALUE, y = PH)) +
    geom_point(color = palette[2], fill = palette[1], alpha = 0.25) +
    geom_smooth(method = "lm", color = "black", linewidth = 0.5, se = FALSE) +
    facet_wrap(~PREDICTOR, ncol = 4, scales = "free_x") +
    labs(title = "Batch 2 Predictor vs. PH Scatterplots") +
    theme(panel.spacing.x = unit(4, "mm"),
          axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5),
          plot.title.position = "plot")
p2b
```



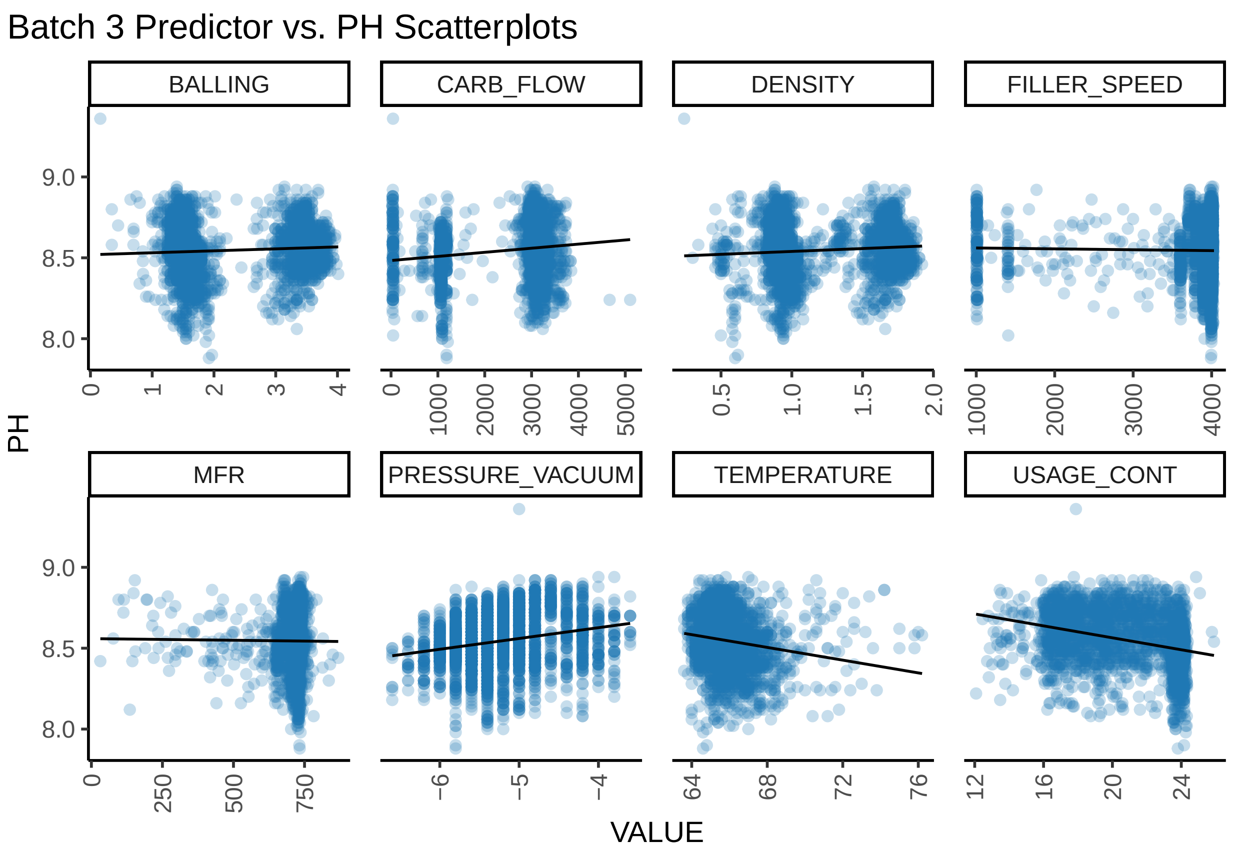
Again, linear relationships are hard to discern from these scatterplots, but there may be a negative relationship between `FILL_PRESSURE` and `PH` and a positive relationship between `FILLER_LEVEL` and `PH`. There's clustering in both variables.

```
p3a <- pivot_df |>
    filter(PREDICTOR %in% all_numeric_chunks[[3]]) |>
    ggplot(aes(x = VALUE)) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "BALLING"),
                   binwidth = 0.25) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "CARB_FLOW"),
                   binwidth = 400) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "DENSITY"),
                   binwidth = 0.1) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "FILLER_SPEED"),
                   binwidth = 200) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "MFR"),
                   binwidth = 50) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "PRESSURE_VACUUM"),
                   binwidth = 0.25) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "TEMPERATURE"),
                   binwidth = 1) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "USAGE_CONT"),
                   binwidth = 1) +
    facet_wrap(vars(PREDICTOR), ncol = 4, scales = "free_x") +
    labs(y = "COUNT",
         title = "Batch 3 Predictor Distributions") +
    theme(panel.spacing.x = unit(4, "mm"),
          axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5),
          plot.title.position = "plot")
p3a
```



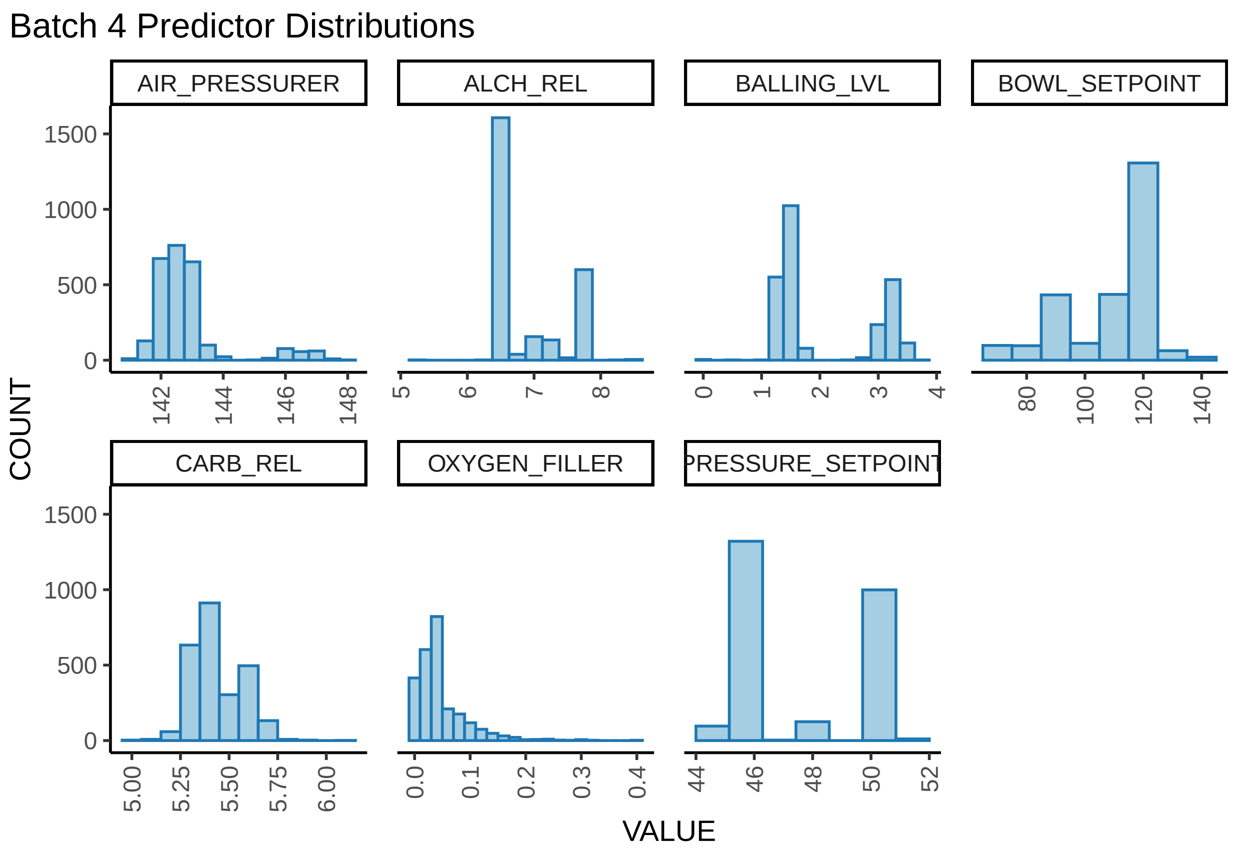
In the third batch of numeric predictors, we see multimodal distributions for `BALLING`, `CARB_FLOW`, and `DENSITY`. `FILLER_SPEED`, `MFR`, and `USAGE_CONT` are left-skewed, and `TEMPERATURE` and `PRESSURE_VACUUM` are right-skewed.

```
sel <- c("PH", all_numeric_chunks[[3]])
p3b <- main_df |>
    select(all_of(sel)) |>
    pivot_longer(cols = all_of(all_numeric_chunks[[3]]), names_to = "PREDICTOR",
                 values_to = "VALUE") |>
    ggplot(aes(x = VALUE, y = PH)) +
    geom_point(color = palette[2], fill = palette[1], alpha = 0.25) +
    geom_smooth(method = "lm", color = "black", linewidth = 0.5, se = FALSE) +
    facet_wrap(~PREDICTOR, ncol = 4, scales = "free_x") +
    labs(title = "Batch 3 Predictor vs. PH Scatterplots") +
    theme(panel.spacing.x = unit(4, "mm"),
          axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5),
          plot.title.position = "plot")
p3b
```



We see clustering in `BALLING`, `CARB_FLOW`, and `DENSITY`. There may be a somewhat positive relationship between `PRESSURE_VACUUM` and `PH`, as well as a somewhat negative relationship between `TEMPERATURE` and `PH`.

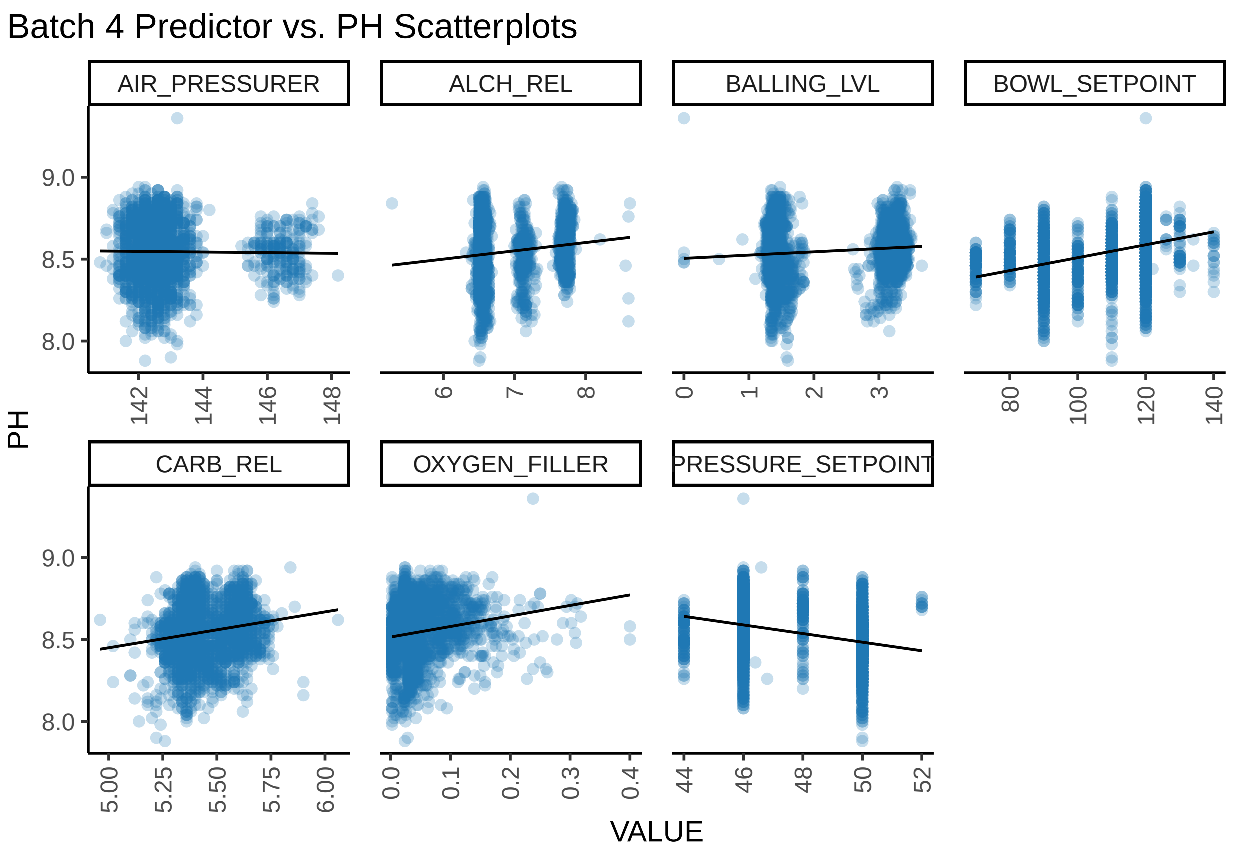
```
p4a <- pivot_df |>
    filter(PREDICTOR %in% all_numeric_chunks[[4]]) |>
    ggplot(aes(x = VALUE)) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "AIR_PRESSURER"),
                   binwidth = 0.5) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "ALCH_REL"),
                   binwidth = 0.25) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "BALLING_LVL"),
                   binwidth = 0.25) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "BOWL_SETPOINT"),
                   binwidth = 10) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "CARB_REL"),
                   binwidth = 0.1) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "OXYGEN_FILLER"),
                   binwidth = 0.02) +
    geom_histogram(color = palette[2], fill = palette[1],
                   data = subset(pivot_df, PREDICTOR == "PRESSURE_SETPOINT"),
                   bins = 8) +
    facet_wrap(vars(PREDICTOR), ncol = 4, scales = "free_x") +
    labs(y = "COUNT",
         title = "Batch 4 Predictor Distributions") +
    theme(panel.spacing.x = unit(4, "mm"),
          axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5),
          plot.title.position = "plot")
p4a
```



In the last batch of numeric predictors, we see that `AIR_PRESSURER` and `OXYGEN_FILLER` are right-skewed.

The distributions for `ALCH_REL`, `BALLING_LVL`, and `PRESSURE_SETPOINT` are multimodal. `BOWL_SETPOINT` is left-skewed. `CARB_REL` is the only variable for which the distribution is nearly normal, and that's debatable.

```r
sel <- c("PH", all_numeric_chunks[[4]])
p4b <- main_df |>
    select(all_of(sel)) |>
    pivot_longer(cols = all_of(all_numeric_chunks[[4]]), names_to = "PREDICTOR",
                 values_to = "VALUE") |>
    ggplot(aes(x = VALUE, y = PH)) +
    geom_point(color = palette[2], fill = palette[1], alpha = 0.25) +
    geom_smooth(method = "lm", color = "black", linewidth = 0.5, se = FALSE) +
    facet_wrap(~PREDICTOR, ncol = 4, scales = "free_x") +
    labs(title = "Batch 4 Predictor vs. PH Scatterplots") +
    theme(panel.spacing.x = unit(4, "mm"),
          axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5),
          plot.title.position = "plot")
p4b
```



We see clustering in `AIR_PRESSURER`, `ALCH_REL`, and `BALLING_LVL`.

Summary statistics for all numeric predictors are below.

```r
remove <- c("n", "vars", "trimmed", "mad", "range", "se", "kurtosis")
describe <- main_df |>
    select(all_of(all_numeric)) |>
    describe() |>
    select(-all_of(remove))
knitr::kable(describe, format = "simple")
```

	mean	sd	median	min	max	skew
CARB_VOLUME	5.3703337	0.1063981	5.3466667	5.0400000	5.700	0.3904059
FILL_OUNCES	23.9749176	0.0874663	23.9733333	23.6333333	24.320	-0.0215410
PC_VOLUME	0.2772392	0.0605992	0.2713333	0.0793333	0.478	0.3468176
CARB_PRESSURE	68.1902677	3.5386086	68.2000000	57.0000000	79.400	0.1811752
CARB_TEMP	141.0922393	4.0340631	140.8000000	128.6000000	154.000	0.2427101
PSC	0.0846433	0.0492487	0.0760000	0.0020000	0.270	0.8504528
PSC_FILL	0.1952987	0.1177889	0.1800000	0.0000000	0.620	0.9352821
PSC_CO_2	0.0564399	0.0430641	0.0400000	0.0000000	0.240	1.7270393
MNF_FLOW	24.6269575	119.5013986	70.2000000	-100.2000000	229.400	0.0031327
CARB_PRESSURE_1	122.5704142	4.7272264	123.2000000	105.6000000	140.200	0.0429942
FILL_PRESSURE	47.9221656	3.1775457	46.4000000	34.6000000	60.400	0.5471107
HYD_PRESSURE_1	12.4571987	12.4330687	11.4000000	-0.8000000	58.000	0.7779346
HYD_PRESSURE_2	20.9935737	16.3784943	28.6000000	0.0000000	59.400	-0.3056277
HYD_PRESSURE_3	20.4778997	15.9714047	27.6000000	-1.2000000	50.000	-0.3210114
HYD_PRESSURE_4	96.3087830	13.0976498	96.0000000	62.0000000	142.000	0.5602427
FILLER_LEVEL	109.2523716	15.6984241	118.4000000	55.8000000	161.200	-0.8482847
FILLER_SPEED	3688.1066454	769.6282261	3982.0000000	998.0000000	4030.000	-2.8777117
TEMPERATURE	65.9648532	1.3790586	65.6000000	63.6000000	76.200	2.3920389
USAGE_CONT	20.9942155	2.9761958	21.7900000	12.0800000	25.900	-0.5351830
CARB_FLOW	2472.0530214	1070.4281545	3030.0000000	26.0000000	5104.000	-0.9916636
DENSITY	1.1744527	0.3769684	0.9800000	0.2400000	1.920	0.5311125
MFR	704.0492582	73.8983094	724.0000000	31.4000000	868.600	-5.0917729
BALLING	2.1998418	0.9295470	1.6480000	0.1600000	4.012	0.6004592
PRESSURE_VACUUM	-5.2162057	0.5703665	-5.4000000	-6.6000000	-3.600	0.5258505
OXYGEN_FILLER	0.0464281	0.0450729	0.0334000	0.0024000	0.400	2.4147972
BOWL_SETPOINT	109.3450292	15.2891482	120.0000000	70.0000000	140.000	-0.9749161
PRESSURE_SETPOINT	47.6132290	2.0387546	46.0000000	44.0000000	52.000	0.2051072
AIR_PRESSURER	142.8339696	1.2127148	142.6000000	140.8000000	148.200	2.2512354
ALCH_REL	6.8978125	0.5052561	6.5600000	5.2800000	8.620	0.8830750
CARB_REL	5.4367956	0.1287629	5.4000000	4.9600000	6.060	0.5028431
BALLING_LVL	2.0516212	0.8688888	1.4800000	0.0000000	3.660	0.5943424

Now we check for degenerate distributions.

```
nzv_predictors <- nearZeroVar(main_df, names = TRUE, saveMetrics = FALSE)
nzv_predictors
```

```
## [1] "HYD_PRESSURE_1"
```

The only near-zero-variance predictor identified is `HYD_PRESSURE_1`. We remove this predictor from the historical and evaluation datasets.

```
main_df <- main_df |>
    select(-all_of(nzv_predictors))
eval_df <- eval_df |>
    select(-all_of(nzv_predictors))
```

Next we examine the dataset's completeness.

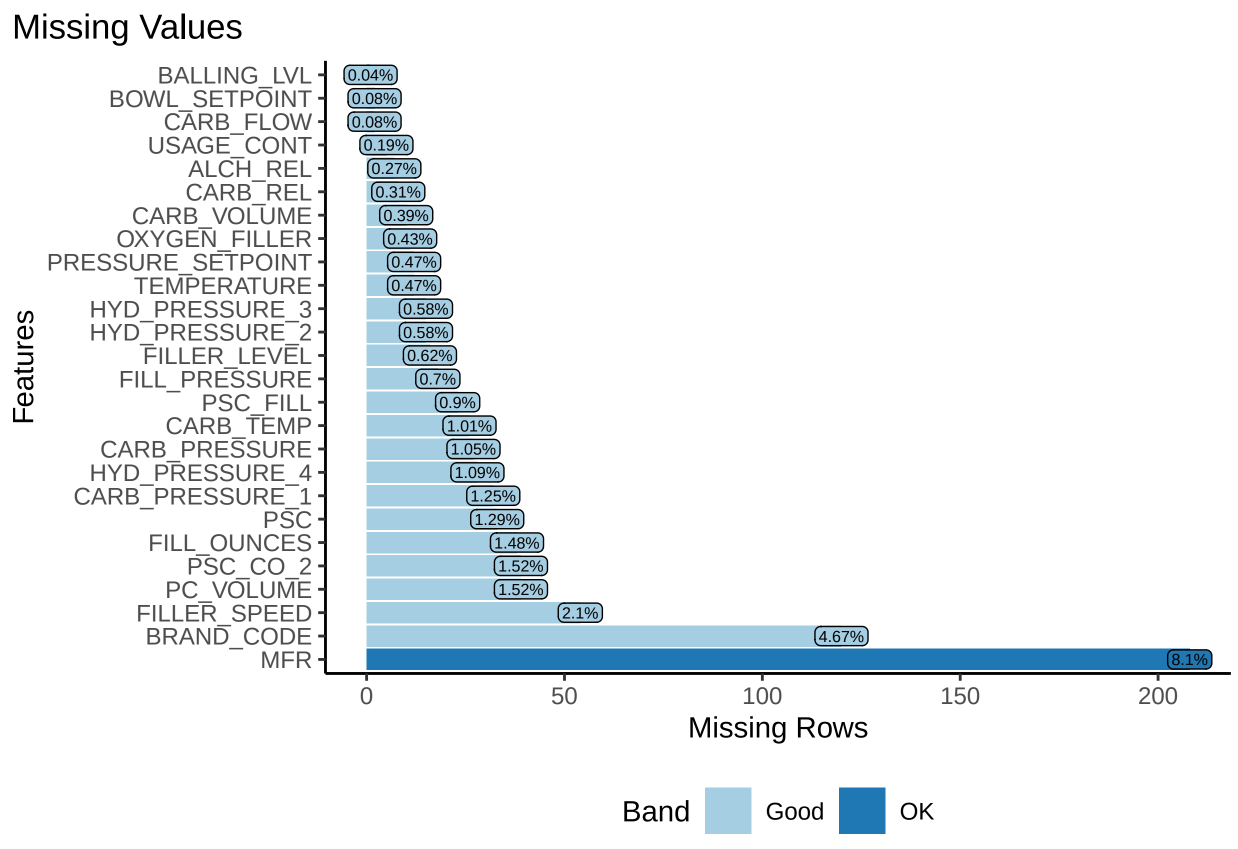
```r
remove <- c("discrete_columns", "continuous_columns", "total_observations",
            "memory_usage")
introduce <- main_df |>
    introduce() |>
    select(-all_of(remove))
knitr::kable(t(introduce), format = "simple")
```

rows	2567
columns	32
all_missing_columns	0
total_missing_values	801
complete_rows	2038

Only 2,038 out of 2,571 rows are complete, which is about 79 percent of observations. There are 844 missing values. None of our variables are completely `NA`.

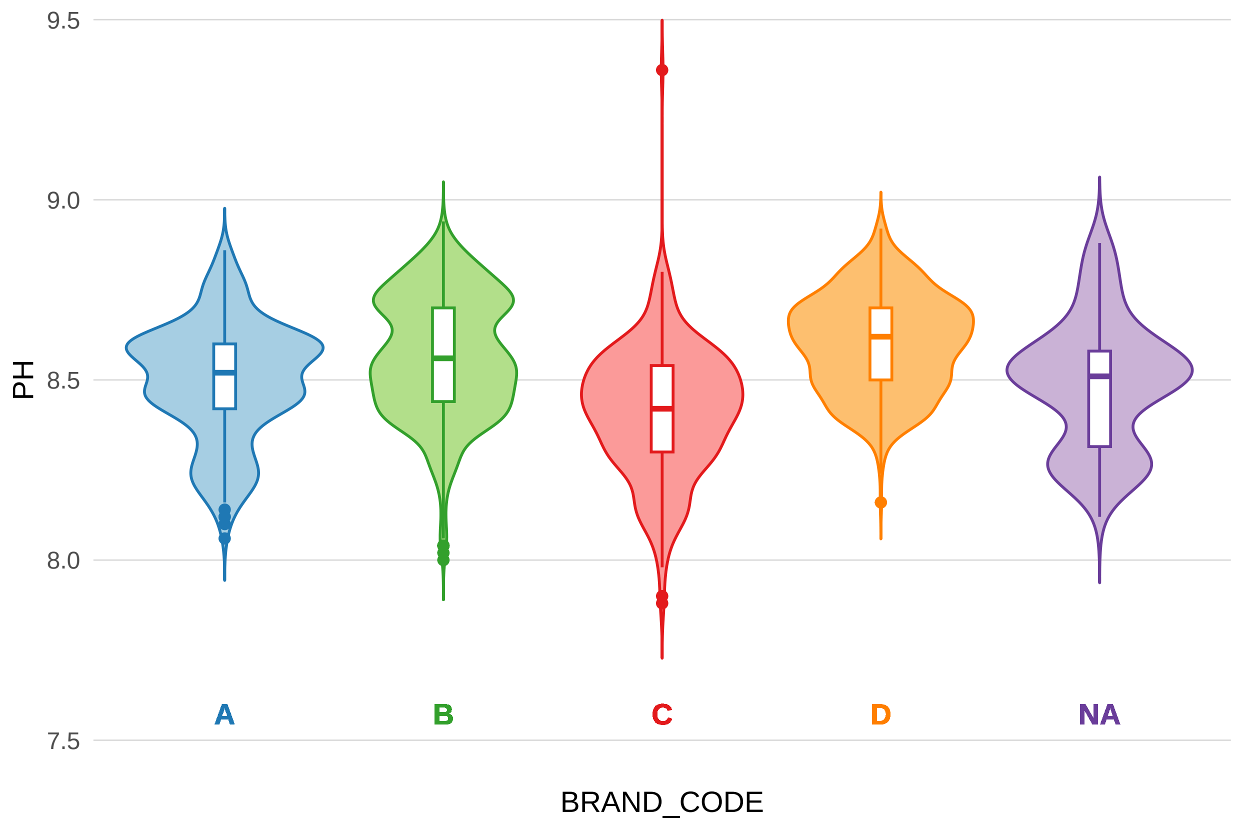
We take a closer look at where the missing values are.

```r
p5 <- p5 +
    scale_fill_brewer(palette = "Paired") +
    theme(plot.title.position = "plot")
p5
```



MFR, BRAND_CODE, and FILLER_SPEED are the predictors with the most missing values, but many other predictors are missing values as well. We coerce BRAND_CODE to a factor and add a level for NA values to handle missingness for this categorical predictor. Then we look at the distribution of PH by BRAND_CODE level to determine whether there are differences in variation between groups and outliers within groups.

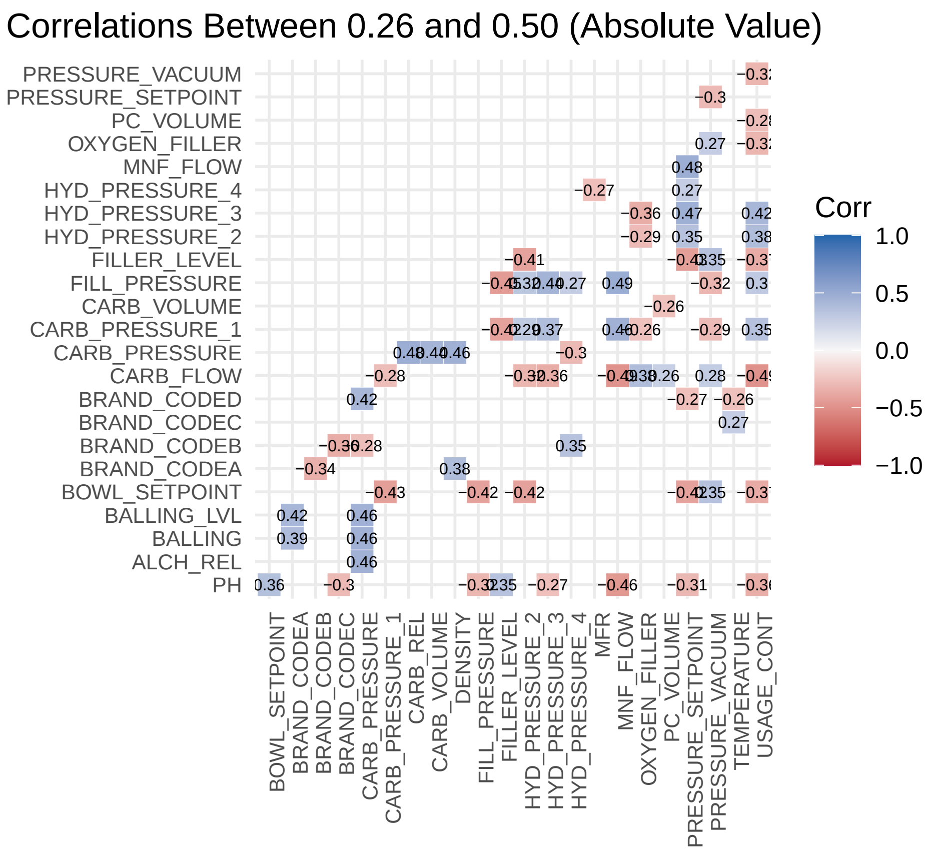
```r
main_df <- main_df |>
    mutate(BRAND_CODE = factor(BRAND_CODE, exclude = NULL))
palette <- brewer.pal(n = 12, name = "Paired")
col <- palette[c(2, 4, 6, 8, 10)]
fil <- palette[c(1, 3, 5, 7, 9)]
p6 <- main_df |>
    ggplot(aes(x = BRAND_CODE, y = PH, color = BRAND_CODE, fill = BRAND_CODE)) +
    geom_violin(trim = FALSE) +
    geom_boxplot(width = 0.1, fill = "white") +
    geom_text(aes(label = BRAND_CODE, color = BRAND_CODE), y = 7.5,
              vjust = -0.75, size = 4, fontface = "bold") +
    scale_y_continuous(limits = c(7.5, 9.5), breaks = seq(7.5, 9.5, 0.5)) +
    scale_color_manual(values = col) +
    scale_fill_manual(values = fil) +
    theme(legend.position = "none",
          axis.ticks = element_blank(),
          axis.text.x = element_blank(),
          axis.line = element_blank(),
          panel.grid.major.y = element_line(color = greys[3], linewidth = 0.25,
                                            linetype = 1))
p6
```



Level "D" has the highest median PH, level "C" has the only outlier on the high end, and all levels except the level representing NA values have outliers on the low end. Level "A" has the narrowest IQR, whereas as the level representing NA values has the widest IQR.

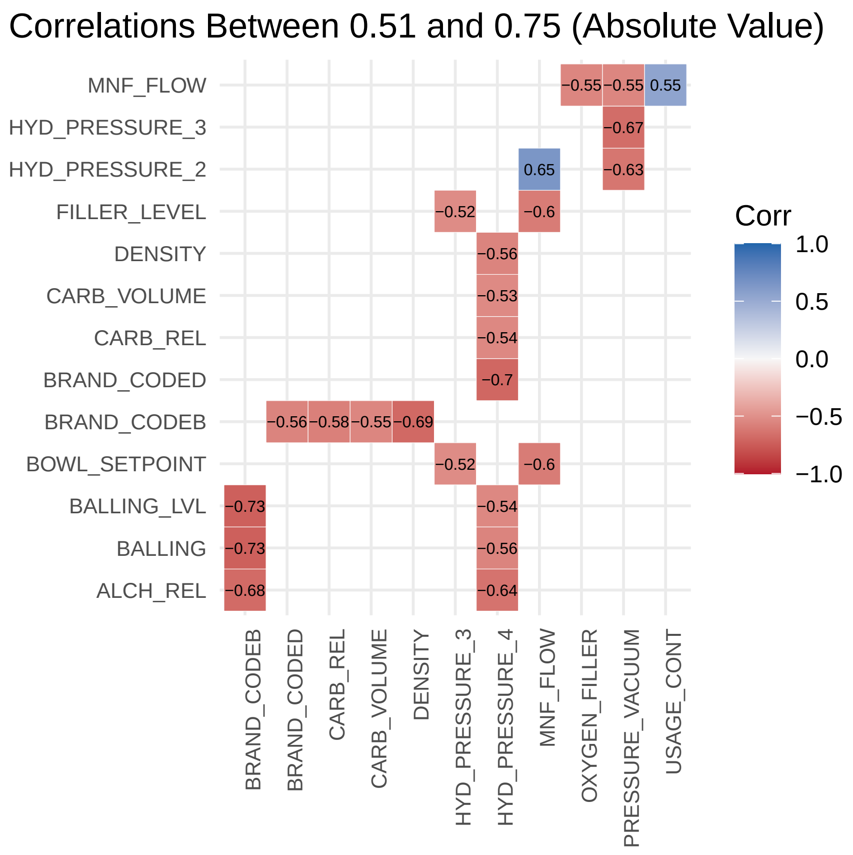
We will perform KNN imputation for the numeric predictors with missing data. We will also create a secondary version of the data where we perform list-wise deletion instead. Before we handle this remaining missing data, we first look at correlations between our predictors and the response variable. Because we have so many variables, it would be difficult to visualize all correlations at the same time without binning them. So we will bin absolute value correlations into four groups: 1) 0.00 to 0.25, 2) 0.26 to 0.50, 3) 0.51 to 0.75, and 4) 0.76 to 1.00. We won't visualize any correlations less than 0.26, but note that this doesn't imply those correlations are insignificant. Limiting what we examine most closely will simply help us hone in on a) the predictor variables we should expect any good model we develop to include and b) the predictor variables that are so highly correlated with one another that they could inhibit certain models' performance.

```r
incl <- c("PH", sort(colnames(main_df |> select(-PH))))
palette <- brewer.pal(n = 7, name = "RdBu")[c(1, 4, 7)]
r <- model.matrix(~0+., data = main_df |> select(all_of(incl))) |>
    cor(use = "pairwise.complete.obs")
is.na(r) <- abs(r) > 0.5
is.na(r) <- abs(r) < 0.26
p7 <- r |>
    ggcorrplot(show.diag = FALSE, type = "lower", lab = TRUE, lab_size = 2,
               tl.cex = 8, tl.srt = 90,
               colors = palette, outline.color = "white") +
    labs(title = "Correlations Between 0.26 and 0.50 (Absolute Value)") +
    theme(plot.title.position = "plot")
p7
```



Here, we see the predictors that are most positively correlated with PH are BOWL_SETPOINT and FILLER_LEVEL, and the predictors that are most negatively correlated with PH are BRAND_CODE level “C”, FILL_PRESSURE, HYD_PRESSURE_3, MNF_FLOW, PRESSURE_SETPOINT, and USAGE_CONT. While some of the predictors in this plot are moderately correlated with each other, we will focus on higher/more worrisome predictor-predictor correlation levels in the following two plots.

```r
r <- model.matrix(~0+., data = main_df |> select(all_of(incl))) |>
    cor(use = "pairwise.complete.obs")
is.na(r) <- abs(r) > 0.75
is.na(r) <- abs(r) < 0.51
p8 <- r |>
    ggcorrplot(show.diag = FALSE, type = "lower", lab = TRUE, lab_size = 2,
               tl.cex = 8, tl.srt = 90,
               colors = palette, outline.color = "white") +
    labs(title = "Correlations Between 0.51 and 0.75 (Absolute Value)") +
    theme(plot.title.position = "plot")
p8
```



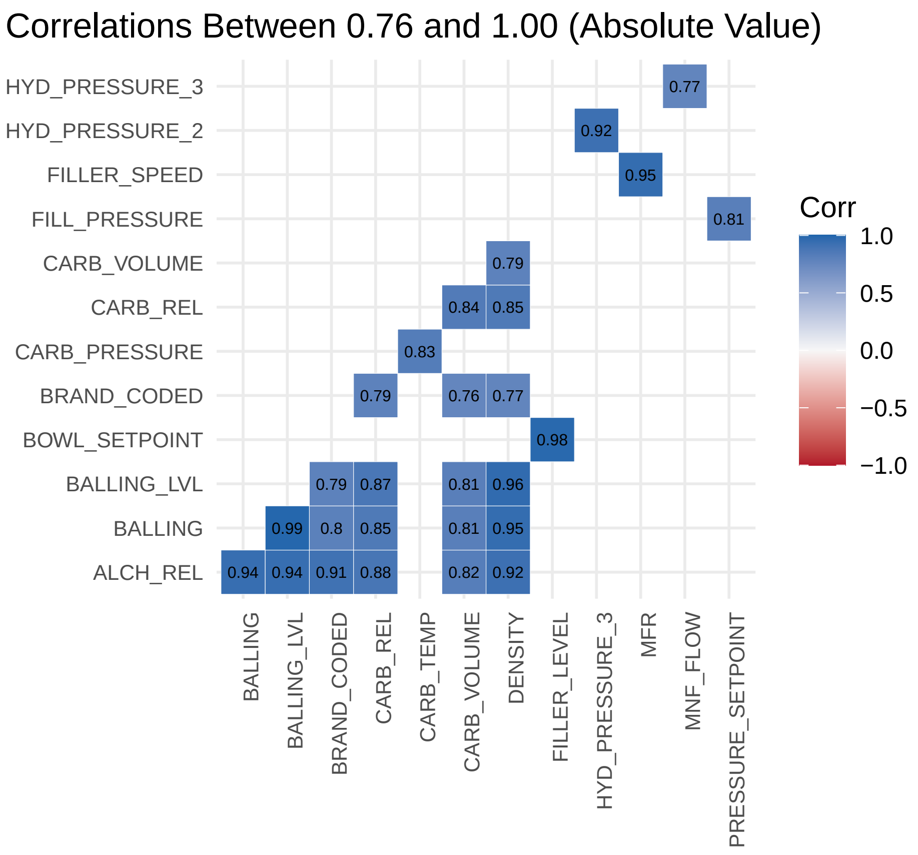
Here, we notice immediately that `PH` is missing from the plot and is therefore not correlated with any predictors at a level between 0.51 and 0.75 in absolute value. Although we could comment on all these correlation levels, we see high (> 0.6) positive predictor-predictor correlations between:

- `MNF_FLOW` and `HYD_PRESSURE_2`
- `HYD_PRESSURE_3` and `HYD_PRESSURE_2`
- `HYD_PRESSURE_2` and `HYD_PRESSURE_1`

We also see high (< -0.6) negative predictor-predictor correlations between:

- `PRESSURE_VACUUM` and `HYD_PRESSURE_3`/`HYD_PRESSURE_2`
- `HYD_PRESSURE_4` and `BRAND_CODE` level "D"/`ALCH_REL`
- `BRAND_CODE` level "B" and `DENSITY`/`BALLING_LVL`/`BALLING`/`ALCH_REL`

```r
r <- model.matrix(~0+., data = main_df |> select(all_of(incl))) |>
    cor(use = "pairwise.complete.obs")
is.na(r) <- abs(r) < 0.76
p9 <- r |>
    ggcorrplot(show.diag = FALSE, type = "lower", lab = TRUE, lab_size = 2,
               tl.cex = 8, tl.srt = 90,
               colors = palette, outline.color = "white") +
    labs(title = "Correlations Between 0.76 and 1.00 (Absolute Value)") +
    theme(plot.title.position = "plot")
p9
```



PH is again missing, so it is therefore not correlated with any predictors at a level between 0.76 and 1.00 in absolute value. Although we could again comment on all these correlation levels, we see extremely high (> 0.9) positive predictor-predictor correlations between

- MFR and FILLER_SPEED
- HYD_PRESSURE_3 and HYD_PRESSURE_2
- FILLER_LEVEL and BOWL_SETPOINT
- DENSITY and BALLING_LVL/BALLING/ALCH_REL
- BRAND_CODE level "D" and ALCH_REL
- BALLING_LVL and BALLING/ALCH_REL
- BALLING and ALCH_REL

There are no extremely high (< -0.9) negative predictor-predictor correlations.

When creating models that are not robust to collinearity later, we will definitely need to exclude one or more variables in each extremely correlated group, and we may have to do the same for less (but still highly) correlated groups as well.

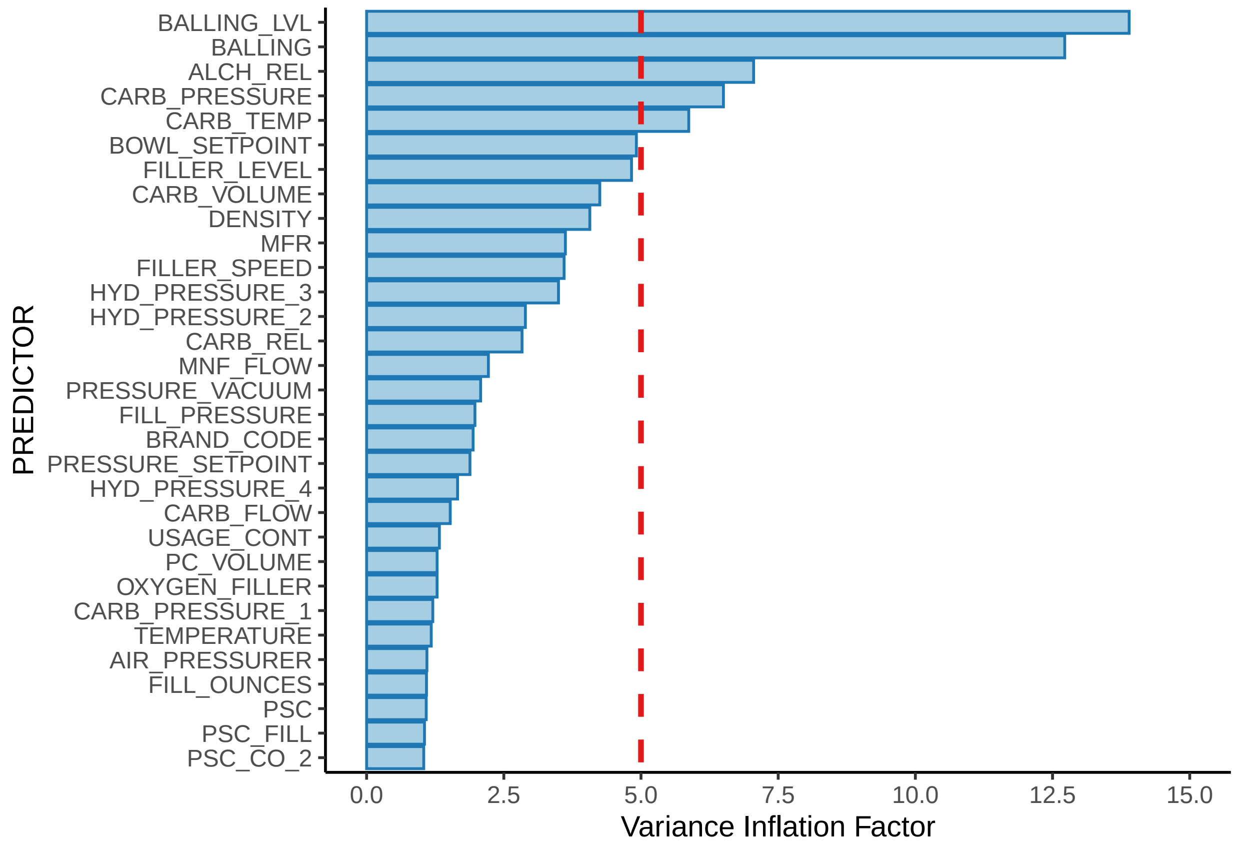
Data Preparation:

To create three versions of the data, one in which missing numeric values have been imputed, one in which observations with missing numeric values have been deleted, and one in which observations with missing numeric values have been imputed and skewed predictors have been transformed, we first set a seed and split the data into train and test sets.

```r
set.seed(417)
rows <- sample(nrow(main_df))
main_df <- main_df[rows, ]
sample <- sample(c(TRUE, FALSE), nrow(main_df), replace=TRUE,
                 prob=c(0.7,0.3))
train_df <- main_df[sample, ]
test_df <- main_df[!sample, ]
```

Next we perform KNN imputation on the missing numeric values for the primary train and test sets separately. To determine the variables this imputation method will use to calculate distance, and the weight of each distance variable used, we fit a random forest model to the training data, identify the 25 most important variables, and extract their variable importance scores. Since including highly correlated variables in a random forest model reduces all their variable importance scores, however, we're actually going to first fit a full multiple linear regression model, check the variance inflation factors to determine any sources of multicollinearity, and eliminate problematic predictors from consideration.

```r
mlr_model1 <- lm(PH ~ ., data = train_df)
mlr_model1_vif <- as.data.frame(vif(mlr_model1)) |>
    rownames_to_column()
cols <- c("PREDICTOR", "GVIF", "DF", "GVIF_ADJ_BY_DF")
colnames(mlr_model1_vif) <- cols
palette <- brewer.pal(n = 12, name = "Paired")
p10 <- mlr_model1_vif |>
    ggplot() +
    geom_col(aes(x = reorder(PREDICTOR, GVIF_ADJ_BY_DF), y = GVIF_ADJ_BY_DF),
             color = palette[2], fill = palette[1]) +
    geom_abline(intercept = 5, slope = 0, linewidth = 1, linetype = 2,
                color = palette[6]) +
    labs(x = "PREDICTOR",
         y = "Variance Inflation Factor") +
    scale_y_continuous(limits = c(0, 15), breaks = seq(0, 15, 2.5)) +
    coord_flip()
p10
```



The variables with variance inflation factors greater than five are `BALLING_LVL`, `BALLING`, `ALCH_REL`, `CARB_PRESSURE`, and `CARB_TEMP`. We remove `ALCH_REL` and `BALLING` from variable importance consideration for imputation because the information these variables provide is largely covered by `BALLING_LVL`, and we remove `CARB_TEMP` from variable importance consideration for imputation in favor of `CARB_PRESSURE`for the same reason.

```
rf_model1 <- randomForest(PH ~ . - ALCH_REL - BALLING - CARB_TEMP, data = train_df,
                          importance = TRUE,
                          ntree = 1000,
                          na.action = na.omit)
rf_imp1 <- varImp(rf_model1, scale = TRUE)
cols <- c("Predictor", "Importance")
rf_imp1 <- rf_imp1 |>
    rownames_to_column()
colnames(rf_imp1) <- cols
rf_imp1 <- rf_imp1 |>
    arrange(desc(Importance)) |>
    top_n(25)
knitr::kable(rf_imp1, format = "simple")
```

Predictor	Importance
BRAND_CODE	62.546944
USAGE_CONT	45.431027
MNF_FLOW	44.120070
PRESSURE_VACUUM	40.833090

Predictor	Importance
OXYGEN_FILLER	38.732888
CARB_REL	36.848281
BALLING_LVL	36.079182
TEMPERATURE	33.419371
DENSITY	31.057464
AIR_PRESSURER	30.507710
FILLER_SPEED	28.337795
FILLER_LEVEL	27.848629
CARB_FLOW	27.345460
BOWL_SETPOINT	23.776895
CARB_VOLUME	22.893358
HYD_PRESSURE_3	22.242977
CARB_PRESSURE_1	21.929859
HYD_PRESSURE_2	21.497959
HYD_PRESSURE_4	20.569592
FILL_PRESSURE	19.534525
MFR	17.209387
PC_VOLUME	16.985108
PRESSURE_SETPOINT	14.316484
CARB_PRESSURE	5.407045
FILL_OUNCES	4.167075

The above variables become the distance variables, and their variable importance scores become the weights in our KNN imputation. We perform said imputation on the primary train and test sets.

```r
dist_vars = rf_imp1$Predictor
wts = rf_imp1$Importance
find_cols_na <- function(df){
    col_sums_na <- colSums(is.na(df))
    cols <- names(col_sums_na[col_sums_na > 0])
    cols #returns column names vector
}
missing_val_cols <- find_cols_na(train_df)
primary_train_df <- train_df |>
    VIM::kNN(variable = missing_val_cols, k = 15, dist_var = dist_vars,
             weights = wts, numFun = median, imp_var = FALSE)
missing_val_cols <- find_cols_na(test_df)
primary_test_df <- test_df |>
    VIM::kNN(variable = missing_val_cols, k = 15, dist_var = dist_vars,
             weights = wts, numFun = median, imp_var = FALSE)
```

Then we create secondary train and test sets where observations with missing numeric values have been deleted.

```r
remove_rows_na <- function(df){
    na_row_sums <- rowSums(is.na(df))
    row_has_na <- ifelse(na_row_sums > 0, TRUE, FALSE)
    copy <- df[!row_has_na, ]
    copy
}
secondary_train_df <- remove_rows_na(train_df)
secondary_test_df <- remove_rows_na(test_df)
```

Finally, we create tertiary train and test sets where all observations with missing numeric values have been imputed and skewed predictors (excluding `PRESSURE_VACUUM` since it takes negative values) have been transformed. Below is a breakdown of the ideal lambdas proposed by Box-Cox for these skewed predictors and the reasonable, more commonly understood transformations we will make instead:

```r
tertiary_train_df <- primary_train_df
tertiary_test_df <- primary_test_df
skewed <- c("PSC", "PSC_CO_2", "PSC_FILL", "HYD_PRESSURE_4", "FILLER_SPEED",
            "MFR", "TEMPERATURE", "USAGE_CONT", "AIR_PRESSURER",
            "BOWL_SETPOINT", "OXYGEN_FILLER")
for (i in 1:(length(skewed))){
    #Add a small constant to columns with any 0 values
    if (sum(tertiary_train_df[[skewed[i]]] == 0) > 0){
        tertiary_train_df[[skewed[i]]] <-
            tertiary_train_df[[skewed[i]]] + 0.001
    }
}
for (i in 1:(length(skewed))){
    if (i == 1){
        lambdas <- c()
    }
    bc <- boxcox(lm(tertiary_train_df[[skewed[i]]] ~ 1),
                 lambda = seq(-2, 2, length.out = 81),
                 plotit = FALSE)
    lambda <- bc$x[which.max(bc$y)]
    lambdas <- append(lambdas, lambda)
}
lambdas <- as.data.frame(cbind(skewed, lambdas))
adj <- c("square root", "square root", "square root", "none", "square",
         "square", "inverse square", "square", "inverse square", "square",
         "log")
lambdas <- cbind(lambdas, adj)
cols <- c("Variable", "Ideal Lambda Proposed by Box-Cox", "Reasonable Alternative Transformation")
colnames(lambdas) <- cols
kable(lambdas, format = "simple")
```

Variable	Ideal Lambda Proposed by Box-Cox	Reasonable Alternative Transformation
PSC	0.45	square root
PSC_CO_2	0.4	square root
PSC_FILL	0.45	square root
HYD_PRESSURE_4	-0.3	none
FILLER_SPEED	2	square
MFR	2	square
TEMPERATURE	-2	inverse square
USAGE_CONT	2	square
AIR_PRESSURER	-2	inverse square
BOWL_SETPOINT	2	square
OXYGEN_FILLER	0.25	log

We make the transformations in the tertiary train and test sets, leaving in the lower order terms for anything we squared, but removing the original terms otherwise.

```r
remove <- c("PSC", "PSC_CO_2", "PSC_FILL", "TEMPERATURE", "AIR_PRESSURER",
            "OXYGEN_FILLER")
tertiary_train_df <- tertiary_train_df |>
    mutate(sqrt_PSC = PSC^0.5,
           sqrt_PSC_CO_2 = PSC_CO_2^0.5,
           sqrt_PSC_FILL = PSC_FILL^0.5,
           FILLER_SPEED_sq = FILLER_SPEED^2,
           MFR_sq = MFR^2,
           inv_sq_TEMPERATURE = TEMPERATURE^-2,
           USAGE_CONT_sq = USAGE_CONT^2,
           inv_sq_AIR_PRESSURER = AIR_PRESSURER^-2,
           BOWL_SETPOINT_sq = BOWL_SETPOINT^2,
           log_OXYGEN_FILLER = log(OXYGEN_FILLER)) |>
    select(-all_of(remove))
for (i in 1:(length(skewed))){
    #Add a small constant to columns with any 0 values
    if (sum(tertiary_test_df[[skewed[i]]] == 0) > 0){
        tertiary_test_df[[skewed[i]]] <-
            tertiary_test_df[[skewed[i]]] + 0.001
    }
}
tertiary_test_df <- tertiary_test_df |>
    mutate(sqrt_PSC = PSC^0.5,
           sqrt_PSC_CO_2 = PSC_CO_2^0.5,
           sqrt_PSC_FILL = PSC_FILL^0.5,
           FILLER_SPEED_sq = FILLER_SPEED^2,
           MFR_sq = MFR^2,
           inv_sq_TEMPERATURE = TEMPERATURE^-2,
           USAGE_CONT_sq = USAGE_CONT^2,
           inv_sq_AIR_PRESSURER = AIR_PRESSURER^-2,
           BOWL_SETPOINT_sq = BOWL_SETPOINT^2,
           log_OXYGEN_FILLER = log(OXYGEN_FILLER)) |>
    select(-all_of(remove))
```

Model Building:

We build a few models in each of three regression categories: linear, nonlinear, and tree-based.

Linear Regression Models:

We start with **Model LM:1**, a linear model that includes all predictors and uses the primary training set (in which missing numeric values have been imputed). The Adjusted R-Squared for the full model is:

```r
lm1 <- lm(PH ~ ., data=primary_train_df)
summary(lm1)$adj.r.squared
```

```
## [1] 0.4121808
```

We reduce **Model LM:1** via step-wise AIC (Akaike Information Criterion) selection in both directions. A summary of the reduced model is below.

```r
lm1 <- stepAIC(lm1, direction = "both", trace=FALSE)
summary(lm1)
```

```
##
## Call:
## lm(formula = PH ~ BRAND_CODE + CARB_VOLUME + PSC + MNF_FLOW +
##     CARB_PRESSURE_1 + HYD_PRESSURE_2 + HYD_PRESSURE_3 + FILLER_LEVEL +
##     TEMPERATURE + USAGE_CONT + CARB_FLOW + DENSITY + BALLING +
##     PRESSURE_VACUUM + OXYGEN_FILLER + BOWL_SETPOINT + PRESSURE_SETPOINT +
##     ALCH_REL + BALLING_LVL, data = primary_train_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.52101 -0.07777  0.00976  0.08602  0.42617
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        8.988e+00  3.906e-01  23.014  < 2e-16 ***
## BRAND_CODEB        1.049e-01  2.653e-02   3.953 8.01e-05 ***
## BRAND_CODEC       -4.464e-02  2.618e-02  -1.706 0.088266 .
## BRAND_CODED        6.708e-02  1.836e-02   3.654 0.000266 ***
## BRAND_CODENA       2.964e-02  2.969e-02   0.998 0.318189
## CARB_VOLUME       -1.336e-01  5.287e-02  -2.527 0.011580 *
## PSC               -1.340e-01  6.440e-02  -2.080 0.037626 *
## MNF_FLOW          -6.764e-04  5.409e-05 -12.506  < 2e-16 ***
## CARB_PRESSURE_1    6.283e-03  8.182e-04   7.678 2.66e-14 ***
## HYD_PRESSURE_2    -1.094e-03  5.441e-04  -2.011 0.044480 *
## HYD_PRESSURE_3     3.403e-03  6.640e-04   5.125 3.31e-07 ***
## FILLER_LEVEL      -1.036e-03  6.440e-04  -1.608 0.107909
## TEMPERATURE       -1.430e-02  2.671e-03  -5.353 9.78e-08 ***
## USAGE_CONT        -5.385e-03  1.317e-03  -4.088 4.54e-05 ***
## CARB_FLOW          1.422e-05  3.823e-06   3.718 0.000207 ***
## DENSITY           -1.056e-01  3.282e-02  -3.219 0.001311 **
## BALLING           -1.209e-01  2.654e-02  -4.555 5.60e-06 ***
## PRESSURE_VACUUM   -2.694e-02  8.373e-03  -3.217 0.001317 **
## OXYGEN_FILLER     -2.901e-01  8.995e-02  -3.225 0.001285 **
## BOWL_SETPOINT      3.316e-03  6.662e-04   4.977 7.09e-07 ***
## PRESSURE_SETPOINT -7.734e-03  1.904e-03  -4.062 5.09e-05 ***
## ALCH_REL           6.825e-02  2.705e-02   2.523 0.011710 *
## BALLING_LVL        1.735e-01  2.752e-02   6.305 3.64e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1313 on 1754 degrees of freedom
## Multiple R-squared:   0.42,  Adjusted R-squared:  0.4128
## F-statistic: 57.74 on 22 and 1754 DF,  p-value: < 2.2e-16
```

The Adjusted R-Squared did not change.

Next we build **Model LM:2**, a linear model that includes all predictors and uses the secondary training set (in which observations with missing numeric values have been deleted). The Adjusted R-Squared for the full model is:

```r
lm2 <- lm(PH ~ ., data=secondary_train_df)
summary(lm2)$adj.r.squared
```

```
## [1] 0.4314624
```

We reduce **Model LM:2** using step-wise AIC selection in both directions again. A summary of the reduced model is below.

```r
lm2 <- stepAIC(lm2, direction = "both", trace=FALSE)
summary(lm2)
```

```
##
## Call:
## lm(formula = PH ~ BRAND_CODE + CARB_VOLUME + FILL_OUNCES + CARB_PRESSURE +
##     CARB_TEMP + MNF_FLOW + CARB_PRESSURE_1 + HYD_PRESSURE_2 +
##     HYD_PRESSURE_3 + HYD_PRESSURE_4 + TEMPERATURE + USAGE_CONT +
##     CARB_FLOW + DENSITY + BALLING + PRESSURE_VACUUM + OXYGEN_FILLER +
##     BOWL_SETPOINT + PRESSURE_SETPOINT + CARB_REL + BALLING_LVL,
##     data = secondary_train_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.54066 -0.07215  0.00653  0.08647  0.43003
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        1.305e+01  1.437e+00   9.085  < 2e-16 ***
## BRAND_CODEB        1.792e-01  3.740e-02   4.791 1.83e-06 ***
## BRAND_CODEC        2.057e-02  3.601e-02   0.571 0.568022
## BRAND_CODED        1.079e-01  1.678e-02   6.429 1.74e-10 ***
## BRAND_CODENA       1.160e-01  4.023e-02   2.883 0.004001 **
## CARB_VOLUME       -3.947e-01  1.363e-01  -2.895 0.003844 **
## FILL_OUNCES       -9.681e-02  4.272e-02  -2.266 0.023594 *
## CARB_PRESSURE      1.226e-02  6.420e-03   1.909 0.056421 .
## CARB_TEMP         -9.288e-03  5.022e-03  -1.849 0.064594 .
## MNF_FLOW          -6.262e-04  5.923e-05 -10.571  < 2e-16 ***
## CARB_PRESSURE_1    5.918e-03  9.018e-04   6.562 7.36e-11 ***
## HYD_PRESSURE_2    -1.463e-03  5.835e-04  -2.507 0.012284 *
## HYD_PRESSURE_3     3.509e-03  7.331e-04   4.787 1.87e-06 ***
## HYD_PRESSURE_4     7.270e-04  4.817e-04   1.509 0.131453
## TEMPERATURE       -1.874e-02  3.763e-03  -4.981 7.09e-07 ***
## USAGE_CONT        -6.878e-03  1.480e-03  -4.647 3.68e-06 ***
## CARB_FLOW          2.013e-05  5.337e-06   3.772 0.000168 ***
## DENSITY           -1.149e-01  3.567e-02  -3.222 0.001303 **
## BALLING           -1.885e-01  4.217e-02  -4.469 8.45e-06 ***
## PRESSURE_VACUUM   -4.973e-02  1.111e-02  -4.476 8.22e-06 ***
## OXYGEN_FILLER     -3.267e-01  1.062e-01  -3.076 0.002139 **
## BOWL_SETPOINT      2.323e-03  3.361e-04   6.911 7.18e-12 ***
## PRESSURE_SETPOINT -7.642e-03  2.080e-03  -3.673 0.000248 ***
## CARB_REL           1.090e-01  7.484e-02   1.456 0.145535
## BALLING_LVL        3.049e-01  4.850e-02   6.288 4.26e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1283 on 1448 degrees of freedom
## Multiple R-squared:  0.4424, Adjusted R-squared:  0.4331
## F-statistic: 47.86 on 24 and 1448 DF,  p-value: < 2.2e-16
```

The Adjusted R-Squared increased very slightly to 0.4331.

Next we build **Model LM:3**, a linear model that uses the tertiary training set (in which observations with missing numeric values have been imputed and skewed predictors have been transformed). The Adjusted R-Squared for the full model is:

```
lm3 <- lm(PH ~ ., data=tertiary_train_df)
summary(lm3)$adj.r.squared
```

```
## [1] 0.4158026
```

We reduce **Model LM:3** using step-wise AIC selection in both directions again. A summary of the reduced model is below. (Note that `BOWL_SETPOINT`, which we squared during transformation, had to be manually added back to the model. If we had done the transformations within the model itself, the model would have kept all lower order terms by default during reduction, but since we transformed the data outside the model, we have to be more careful.)

```
lm3 <- stepAIC(lm3, direction = "both", trace=FALSE)
lm3 <- update(lm3, . ~ . + BOWL_SETPOINT)
summary(lm3)
```

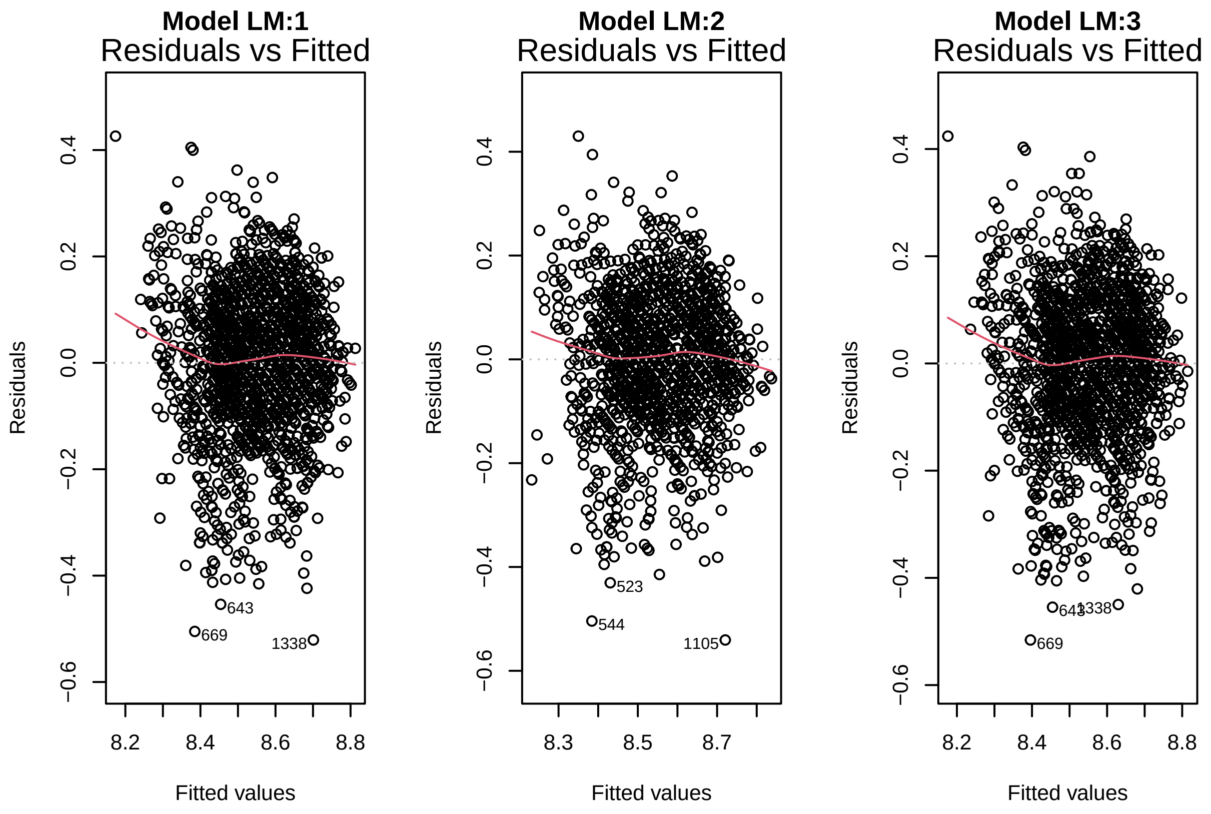
```
##
## Call:
## lm(formula = PH ~ BRAND_CODE + CARB_VOLUME + PC_VOLUME + MNF_FLOW +
##     CARB_PRESSURE_1 + HYD_PRESSURE_2 + HYD_PRESSURE_3 + USAGE_CONT +
##     CARB_FLOW + DENSITY + MFR + BALLING + PRESSURE_VACUUM + PRESSURE_SETPOINT +
##     ALCH_REL + BALLING_LVL + sqrt_PSC + MFR_sq + inv_sq_TEMPERATURE +
##     USAGE_CONT_sq + BOWL_SETPOINT_sq + BOWL_SETPOINT, data = tertiary_train_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.51590 -0.07899  0.00931  0.08659  0.42401
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          7.061e+00  4.535e-01  15.568  < 2e-16 ***
## BRAND_CODEB          1.192e-01  2.670e-02   4.464 8.55e-06 ***
## BRAND_CODEC         -3.018e-02  2.643e-02  -1.142 0.253802
## BRAND_CODED          6.711e-02  1.840e-02   3.647 0.000273 ***
## BRAND_CODENA         4.450e-02  2.981e-02   1.493 0.135657
## CARB_VOLUME         -1.109e-01  5.298e-02  -2.094 0.036384 *
## PC_VOLUME           -1.055e-01  6.258e-02  -1.686 0.091908 .
## MNF_FLOW            -5.758e-04  4.930e-05 -11.680  < 2e-16 ***
## CARB_PRESSURE_1      5.962e-03  8.188e-04   7.281 4.97e-13 ***
## HYD_PRESSURE_2      -1.063e-03  5.489e-04  -1.936 0.053051 .
## HYD_PRESSURE_3       3.423e-03  6.615e-04   5.174 2.55e-07 ***
## USAGE_CONT           6.737e-02  1.825e-02   3.691 0.000230 ***
## CARB_FLOW            1.063e-05  4.029e-06   2.639 0.008385 **
## DENSITY             -1.110e-01  3.312e-02  -3.352 0.000820 ***
## MFR                 -4.212e-04  2.421e-04  -1.740 0.082122 .
## BALLING             -1.100e-01  2.767e-02  -3.976 7.30e-05 ***
## PRESSURE_VACUUM     -2.741e-02  8.782e-03  -3.121 0.001829 **
## PRESSURE_SETPOINT   -6.978e-03  1.912e-03  -3.650 0.000270 ***
## ALCH_REL             6.038e-02  2.736e-02   2.207 0.027460 *
## BALLING_LVL          1.729e-01  2.794e-02   6.186 7.66e-10 ***
## sqrt_PSC            -7.006e-02  3.817e-02  -1.835 0.066638 .
## MFR_sq               3.939e-07  2.289e-07   1.721 0.085403 .
## inv_sq_TEMPERATURE  2.249e+03  4.083e+02   5.509 4.15e-08 ***
## USAGE_CONT_sq       -1.848e-03  4.599e-04  -4.019 6.09e-05 ***
## BOWL_SETPOINT_sq     2.009e-05  1.481e-05   1.357 0.175084
## BOWL_SETPOINT       -1.722e-03  3.015e-03  -0.571 0.567982
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.131 on 1751 degrees of freedom
## Multiple R-squared:  0.4242, Adjusted R-squared:  0.416
## F-statistic:  51.6 on 25 and 1751 DF,  p-value: < 2.2e-16
```

The Adjusted R-Squared increased very slightly to 0.416, but `BOWL_SETPOINT_sq` is no longer significant. We leave it in.

Looking at the significant predictors in the three linear models, there are some differences to note. `FILL_OUNCES` was deemed significant in only **Model LM:2**, and `PSC` and `ALCH_REL` were deemed significant in only **Model LM:1**.

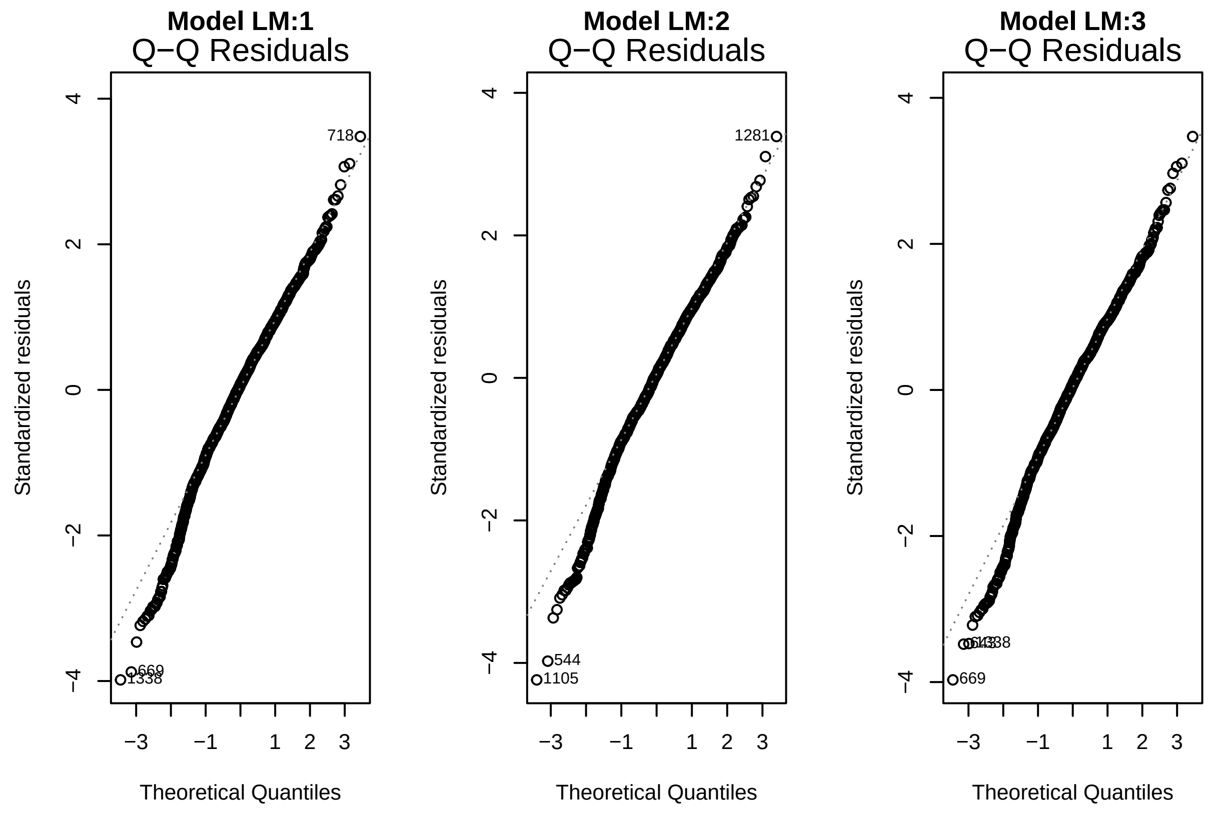
We examine diagnostic plots for the three linear models.

```r
par(mfrow = c(1, 3))
plot(lm1, 1, main = "Model LM:1")
plot(lm2, 1, main = "Model LM:2")
plot(lm3, 1, main = "Model LM:3")
```



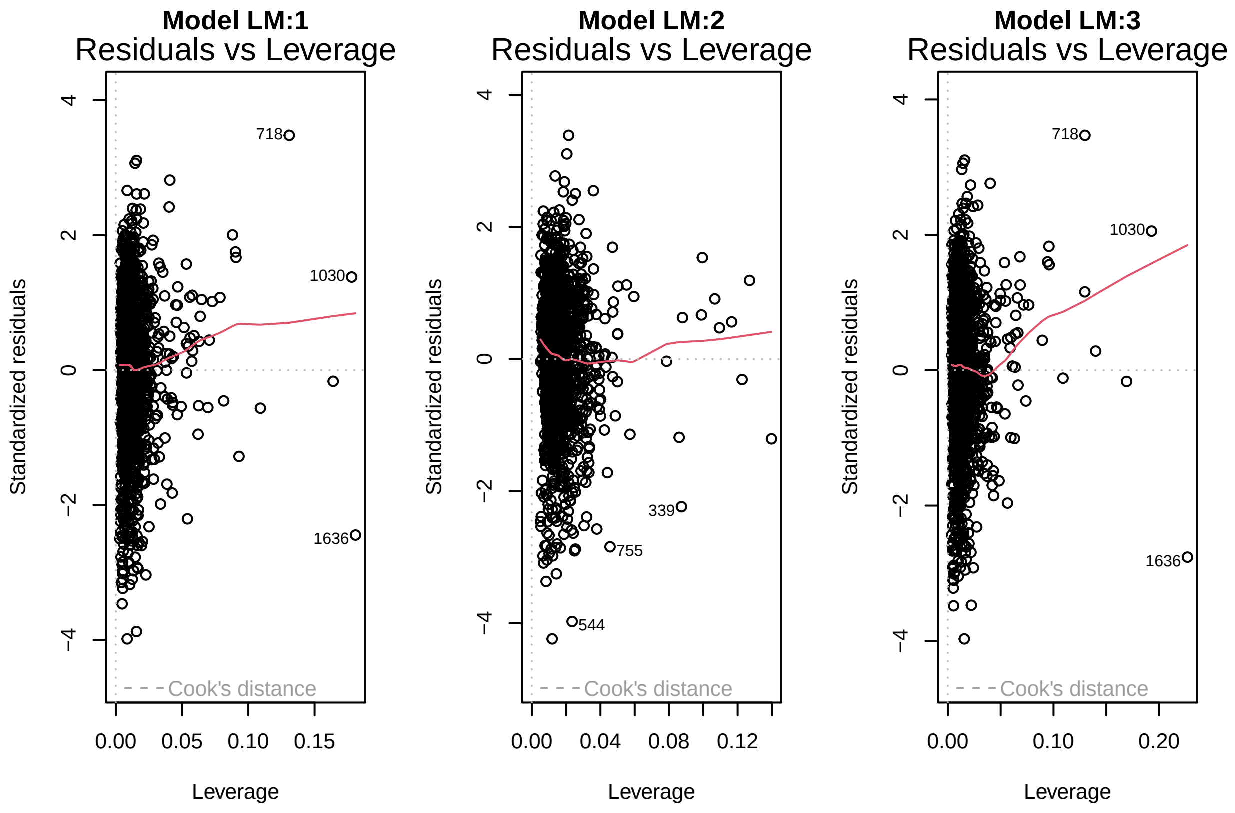
There’s a bit of a sigmoid pattern in the red line in the Residuals vs. Fitted Values plots for all three linear models, suggesting they share a fit issue.

```r
par(mfrow = c(1, 3))
plot(lm1, 2, main = "Model LM:1")
plot(lm2, 2, main = "Model LM:2")
plot(lm3, 2, main = "Model LM:3")
```



The Q-Q plots diverge from the normal line at the low-end for all three linear models.

```r
par(mfrow = c(1, 3))
plot(lm1, 5, main = "Model LM:1")
plot(lm2, 5, main = "Model LM:2")
plot(lm3, 5, main = "Model LM:3")
```

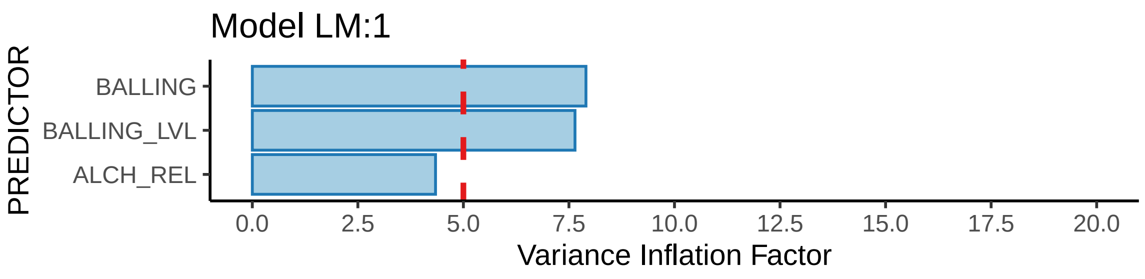


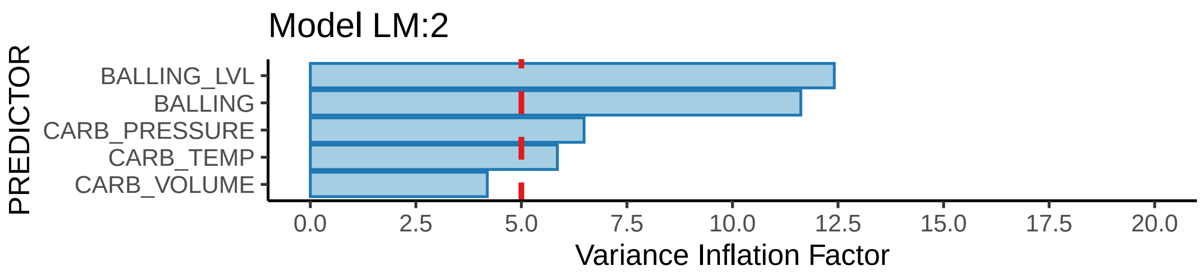
There are no points beyond the border of Cook's distance in any of the linear models, so there are no highly influential observations to investigate.

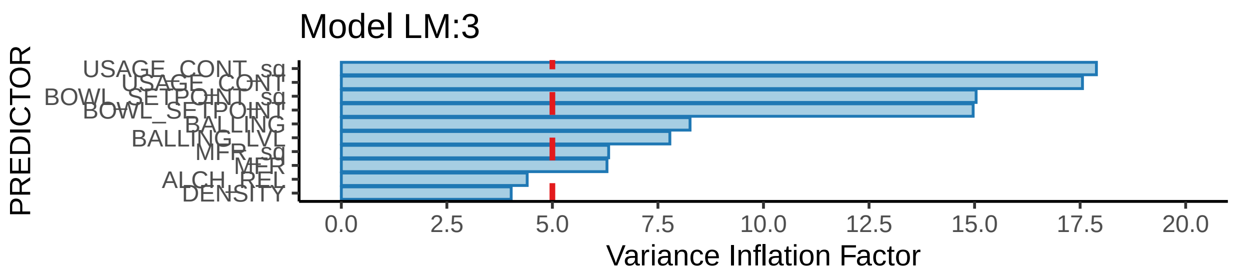
We check for multicollinearity in the three linear models. (Only variance inflation factors greater than 4 are displayed for readability.)

```
palette <- brewer.pal(n = 12, name = "Paired")
lm1_vif <- as.data.frame(vif(lm1)) |>
    rownames_to_column()
lm2_vif <- as.data.frame(vif(lm2)) |>
    rownames_to_column()
lm3_vif <- as.data.frame(vif(lm3)) |>
    rownames_to_column()
cols <- c("PREDICTOR", "GVIF", "DF", "GVIF_ADJ_BY_DF")
colnames(lm1_vif) <- cols
colnames(lm2_vif) <- cols
colnames(lm3_vif) <- cols
p11a <- lm1_vif |>
    filter(GVIF_ADJ_BY_DF > 4) |>
    ggplot() +
    geom_col(aes(x = reorder(PREDICTOR, GVIF_ADJ_BY_DF), y = GVIF_ADJ_BY_DF),
             color = palette[2], fill = palette[1]) +
    geom_abline(intercept = 5, slope = 0, linewidth = 1, linetype = 2,
                color = palette[6]) +
    labs(x = "PREDICTOR",
         y = "Variance Inflation Factor",
         title = "Model LM:1") +
```

```r
    scale_y_continuous(limits = c(0, 20), breaks = seq(0, 20, 2.5)) +
    coord_flip()
p11b <- lm2_vif |>
    filter(GVIF_ADJ_BY_DF > 4) |>
    ggplot() +
    geom_col(aes(x = reorder(PREDICTOR, GVIF_ADJ_BY_DF), y = GVIF_ADJ_BY_DF),
             color = palette[2], fill = palette[1]) +
    geom_abline(intercept = 5, slope = 0, linewidth = 1, linetype = 2,
                color = palette[6]) +
    labs(x = "PREDICTOR",
         y = "Variance Inflation Factor",
         title = "Model LM:2") +
    scale_y_continuous(limits = c(0, 20), breaks = seq(0, 20, 2.5)) +
    coord_flip()
p11c <- lm3_vif |>
    filter(GVIF_ADJ_BY_DF > 4) |>
    ggplot() +
    geom_col(aes(x = reorder(PREDICTOR, GVIF_ADJ_BY_DF), y = GVIF_ADJ_BY_DF),
             color = palette[2], fill = palette[1]) +
    geom_abline(intercept = 5, slope = 0, linewidth = 1, linetype = 2,
                color = palette[6]) +
    labs(x = "PREDICTOR",
         y = "Variance Inflation Factor",
         title = "Model LM:3") +
    scale_y_continuous(limits = c(0, 20), breaks = seq(0, 20, 2.5)) +
    coord_flip()
p11 <- plot_grid(p11a, p11b, p11c, ncol = 1, align = "v", axis = "l")
p11
```







There are different multicollinearity issues in each model. In **Model LM:1**, we remove `BALLING` and `ALCH_REL` because their information is largely covered by `BALLING_LVL`, as discussed previously. The same reasoning applies to removing `BALLING` in favor of `BALLING_LVL` and `CARB_TEMP` in favor of `CARB_PRESSURE` in **Model LM:2**. In **Model LM:3**, we expect issues from including lower and higher order terms, but that is what we want to do, so those can be ignored. We only remove `BALLING` and `ALCH_REL`. After these adjustments, the final Adjusted R-Squared metrics for the three linear models are below:

```
lm1 <- update(lm1, . ~ . - BALLING - ALCH_REL)
lm2 <- update(lm2, . ~ . - BALLING - CARB_TEMP)
lm3 <- update(lm3, . ~ . - BALLING - ALCH_REL)
lms <- c("Model LM:1", "Model LM:2", "Model LM:3")
rsq <- c(summary(lm1)$adj.r.squared, summary(lm2)$adj.r.squared,
         summary(lm3)$adj.r.squared)
summ <- as.data.frame(cbind(lms, round(rsq, 4)))
cols <- c("Model", "Adjust R-Squared")
colnames(summ) <- cols
knitr::kable(summ, format = "simple")
```

Model	Adjust R-Squared
Model LM:1	0.4054
Model LM:2	0.4245
Model LM:3	0.4107

Nonlinear Regression Models:

Next we build three versions of three types of nonlinear regression models, training each version on one of the three different training sets.

First, we tune three Multivariate Adaptive Regression Spline (MARS) models.

```r
marsGrid <- expand.grid(.degree = 1:2, .nprune = 2:20)
mars1 <- train(primary_train_df |> select(-PH), primary_train_df$PH,
                   method = "earth",
                   tuneGrid = marsGrid,
                   trControl = trainControl(method = "cv"))
mars2 <- train(secondary_train_df |> select(-PH), secondary_train_df$PH,
                   method = "earth",
                   tuneGrid = marsGrid,
                   trControl = trainControl(method = "cv"))
mars3 <- train(tertiary_train_df |> select(-PH), tertiary_train_df$PH,
                   method = "earth",
                   tuneGrid = marsGrid,
                   trControl = trainControl(method = "cv"))
```

A summary of the ideal tuning parameters and Generalized R-Squared values for the three MARS models is below:

```r
mars_mods <- c("Model MARS:1", "Model MARS:2", "Model MARS:3")
nprune <- c(mars1$bestTune$nprune, mars2$bestTune$nprune, mars3$bestTune$nprune)
degree <- c(mars1$bestTune$degree, mars2$bestTune$degree, mars3$bestTune$degree)
grsq <- c(mars1$finalModel$grsq, mars2$finalModel$grsq, mars3$finalModel$grsq)
summ <- as.data.frame(cbind(mars_mods, nprune, degree, round(grsq, 4)))
cols <- c("Model", "nprune", "degree", "Generalized R-Squared")
colnames(summ) <- cols
knitr::kable(summ, format = "simple")
```

Model	nprune	degree	Generalized R-Squared
Model MARS:1	16	2	0.4853
Model MARS:2	20	2	0.5296
Model MARS:3	14	2	0.4804

And here are summaries of the estimated feature importance for the ten most important features in each of these MARS models:

```r
mars1_feature_importance <- varImp(mars1, method = "gcv")
mars1_feature_importance <- mars1_feature_importance$importance |>
    arrange(desc(Overall)) |>
    rownames_to_column() |>
    top_n(10)
cols <- c("Predictor", "Model MARS:1 Importance")
colnames(mars1_feature_importance) <- cols
knitr::kable(mars1_feature_importance, format = "simple")
```

Predictor	Model MARS:1 Importance
MNF_FLOW	100.000000
BRAND_CODEC	66.716165
HYD_PRESSURE_3	52.879172
AIR_PRESSURER	52.879172
BALLING	48.230435
ALCH_REL	43.999619
BOWL_SETPOINT	26.673886
TEMPERATURE	22.344526
USAGE_CONT	11.440172
CARB_PRESSURE_1	6.685723

```r
mars2_feature_importance <- varImp(mars2, method = "gcv")
mars2_feature_importance <- mars2_feature_importance$importance |>
    arrange(desc(Overall)) |>
    rownames_to_column() |>
    top_n(10)
cols <- c("Predictor", "Model MARS:2 Importance")
colnames(mars2_feature_importance) <- cols
knitr::kable(mars2_feature_importance, format = "simple")
```

Predictor	Model MARS:2 Importance
MNF_FLOW	100.00000
BRAND_CODEC	74.45417
HYD_PRESSURE_3	74.45417
USAGE_CONT	63.63574
FILLER_LEVEL	59.72071
ALCH_REL	54.45987
CARB_PRESSURE_1	50.84616
PRESSURE_VACUUM	45.07385
BALLING	41.47829
TEMPERATURE	36.53317

```r
mars3_feature_importance <- varImp(mars3, method = "gcv")
mars3_feature_importance <- mars3_feature_importance$importance |>
    arrange(desc(Overall)) |>
    rownames_to_column() |>
    top_n(10)
cols <- c("Predictor", "Model MARS:3 Importance")
colnames(mars3_feature_importance) <- cols
knitr::kable(mars3_feature_importance, format = "simple")
```

Predictor	Model MARS:3 Importance
MNF_FLOW	100.00000
inv_sq_AIR_PRESSURER	100.00000
BRAND_CODEC	58.96943
BALLING	44.67704
inv_sq_TEMPERATURE	44.67704
FILLER_SPEED_sq	41.81319

Predictor	Model MARS:3 Importance
USAGE_CONT	36.39819
CARB_PRESSURE_1	30.52184
PRESSURE_VACUUM	30.52184
BRAND_CODED	23.80526
BOWL_SETPOINT_sq	23.80526

Next, we tune three K Nearest Neighbors (KNN) models.

```r
ctrl <- trainControl(method="repeatedcv",repeats = 3)
knn1 <- train(PH ~ ., data = primary_train_df, method = "knn", trControl=ctrl, preProcess = c("center",
knn2 <- train(PH ~ ., data = secondary_train_df, method = "knn", trControl=ctrl, preProcess = c("center
knn3 <- train(PH ~ ., data = tertiary_train_df, method = "knn", trControl=ctrl, preProcess = c("center"
```

A summary of the ideal k and R-Squared values for the three KNN models is below:

```r
knn_mods <- c("Model KNN:1", "Model KNN:2", "Model KNN:3")
k <- c(knn1$bestTune$k, knn2$bestTune$k, knn3$bestTune$k)
rsq <- c(knn1$results |> filter(k == knn1$bestTune$k) |> select(Rsquared) |> as.numeric(),
         knn2$results |> filter(k == knn2$bestTune$k) |> select(Rsquared) |> as.numeric(),
         knn3$results |> filter(k == knn3$bestTune$k) |> select(Rsquared) |> as.numeric())
summ <- as.data.frame(cbind(knn_mods, k, round(rsq, 4)))
cols <- c("Model", "k", "R-Squared")
colnames(summ) <- cols
knitr::kable(summ, format = "simple")
```

Model	k	R-Squared
Model KNN:1	7	0.5137
Model KNN:2	7	0.5543
Model KNN:3	7	0.49

The ideal k was 7 for all three models, so the different training sets did not impact the best boundary here. Below are summaries of the estimated feature importance for the ten most important features in each of these KNN models:

```r
orig_test_pred <- predict(knn1, primary_test_df |> select(-PH))
orig_pred_rsq <- as.numeric(R2(orig_test_pred, primary_test_df |> select(PH),
                               form = "traditional"))
features <- colnames(primary_test_df |> select(-PH))
feature_importance <- rep(0, length(features))
names(feature_importance) <- features
for (f in 1:length(features)){
    test_x_shuffled <- primary_test_df |> select(-PH)
    rows <- sample(nrow(test_x_shuffled))
    test_x_shuffled[, f] <- test_x_shuffled[rows, f]
    new_test_pred <- predict(knn1, test_x_shuffled)
    new_pred_rsq <- as.numeric(R2(new_test_pred, primary_test_df |> select(PH),
                                  form = "traditional"))
    feature_importance[f] <- orig_pred_rsq - new_pred_rsq
}
feature_importance <- sort(feature_importance, decreasing = TRUE) |>
    as.data.frame() |>
    rownames_to_column() |>
    top_n(10)
cols <- c("Predictor", "Model KNN:1 Importance")
colnames(feature_importance) <-  cols
knitr::kable(feature_importance, format = "simple")
```

Predictor	Model KNN:1 Importance
BRAND_CODE	0.2173181
AIR_PRESSURER	0.0647417
USAGE_CONT	0.0526153
OXYGEN_FILLER	0.0510969
BOWL_SETPOINT	0.0480915
MNF_FLOW	0.0378647
TEMPERATURE	0.0352188
CARB_FLOW	0.0339325
PRESSURE_VACUUM	0.0333629
FILLER_LEVEL	0.0298235

```
orig_test_pred <- predict(knn2, secondary_test_df |> select(-PH))
orig_pred_rsq <- as.numeric(R2(orig_test_pred, secondary_test_df |> select(PH),
                                  form = "traditional"))
features <- colnames(secondary_test_df |> select(-PH))
feature_importance <- rep(0, length(features))
names(feature_importance) <- features
for (f in 1:length(features)){
    test_x_shuffled <- secondary_test_df |> select(-PH)
    rows <- sample(nrow(test_x_shuffled))
    test_x_shuffled[, f] <- test_x_shuffled[rows, f]
    new_test_pred <- predict(knn2, test_x_shuffled)
    new_pred_rsq <- as.numeric(R2(new_test_pred,
                                  secondary_test_df |> select(PH),
                                  form = "traditional"))
    feature_importance[f] <- orig_pred_rsq - new_pred_rsq
}
feature_importance <- sort(feature_importance, decreasing = TRUE) |>
    as.data.frame() |>
    rownames_to_column() |>
    top_n(10)
cols <- c("Predictor", "Model KNN:2 Importance")
colnames(feature_importance) <-  cols
knitr::kable(feature_importance, format = "simple")
```

Predictor	Model KNN:2 Importance
BRAND_CODE	0.1998780
OXYGEN_FILLER	0.0563566
PRESSURE_VACUUM	0.0505829
CARB_FLOW	0.0454219

Predictor	Model KNN:2 Importance
FILLER_LEVEL	0.0439370
TEMPERATURE	0.0407374
AIR_PRESSURER	0.0380072
BOWL_SETPOINT	0.0352910
CARB_PRESSURE_1	0.0322369
USAGE_CONT	0.0303735

```r
orig_test_pred <- predict(knn3, tertiary_test_df |> select(-PH))
orig_pred_rsq <- as.numeric(R2(orig_test_pred, tertiary_test_df |> select(PH),
                               form = "traditional"))
features <- colnames(tertiary_test_df |> select(-PH))
feature_importance <- rep(0, length(features))
names(feature_importance) <- features
for (f in 1:length(features)){
    test_x_shuffled <- tertiary_test_df |> select(-PH)
    rows <- sample(nrow(test_x_shuffled))
    test_x_shuffled[, f] <- test_x_shuffled[rows, f]
    new_test_pred <- predict(knn3, test_x_shuffled)
    new_pred_rsq <- as.numeric(R2(new_test_pred,
                                  tertiary_test_df |> select(PH),
                                  form = "traditional"))
    feature_importance[f] <- orig_pred_rsq - new_pred_rsq
}
feature_importance <- sort(feature_importance, decreasing = TRUE) |>
    as.data.frame() |>
    rownames_to_column() |>
    top_n(10)
cols <- c("Predictor", "Model KNN:3 Importance")
colnames(feature_importance) <-  cols
knitr::kable(feature_importance, format = "simple")
```

Predictor	Model KNN:3 Importance
BRAND_CODE	0.2014817
inv_sq_AIR_PRESSURER	0.0609561
PRESSURE_VACUUM	0.0347035
log_OXYGEN_FILLER	0.0324806
MNF_FLOW	0.0310678
CARB_FLOW	0.0305449
CARB_PRESSURE_1	0.0285600
BOWL_SETPOINT	0.0279335
USAGE_CONT	0.0229364
inv_sq_TEMPERATURE	0.0188264

Next we train three Support Vector Machine: Radial Basis (SVM: RB) models. (Note that factors always have to be one-hot encoded prior to building these models).

```r
mm1 <- model.matrix(~0+., data = primary_train_df |> select(-PH))
svmRB1Tuned <- train(mm1, primary_train_df$PH,
                     method = "svmRadial",
                     preProc = c("center", "scale"),
                     tuneLength = 14,
                     trControl = trainControl(method = "cv"))
mm2 <- model.matrix(~0+., data = secondary_train_df |> select(-PH))
svmRB2Tuned <- train(mm2, secondary_train_df$PH,
                     method = "svmRadial",
                     preProc = c("center", "scale"),
                     tuneLength = 14,
                     trControl = trainControl(method = "cv"))
mm3 <- model.matrix(~0+., data = tertiary_train_df |> select(-PH))
svmRB3Tuned <- train(mm3, tertiary_train_df$PH,
                     method = "svmRadial",
                     preProc = c("center", "scale"),
                     tuneLength = 14,
                     trControl = trainControl(method = "cv"))
```

A summary of the ideal tuning parameters and R-Squared values for the three SVM:RB models is below:

```
svm_mods <- c("Model SVM:RB:1", "Model SVM:RB:2", "Model SVM:RB:3")
sigmas <- c(svmRB1Tuned$bestTune$sigma, svmRB2Tuned$bestTune$sigma,
            svmRB3Tuned$bestTune$sigma)
cs <- c(svmRB1Tuned$bestTune$C, svmRB2Tuned$bestTune$C,
            svmRB3Tuned$bestTune$C)
rsq <- c(svmRB1Tuned$results |> filter(C == svmRB1Tuned$bestTune$C) |> select(Rsquared) |> as.numeric()
svmRB2Tuned$results |> filter(C == svmRB2Tuned$bestTune$C) |> select(Rsquared) |> as.numeric(),
svmRB3Tuned$results |> filter(C == svmRB3Tuned$bestTune$C) |> select(Rsquared) |> as.numeric())
summ <- as.data.frame(cbind(svm_mods, round(sigmas, 4), cs, round(rsq, 4)))
cols <- c("Model", "sigma", "C", "R-Squared")
colnames(summ) <- cols
knitr::kable(summ, format = "simple")
```

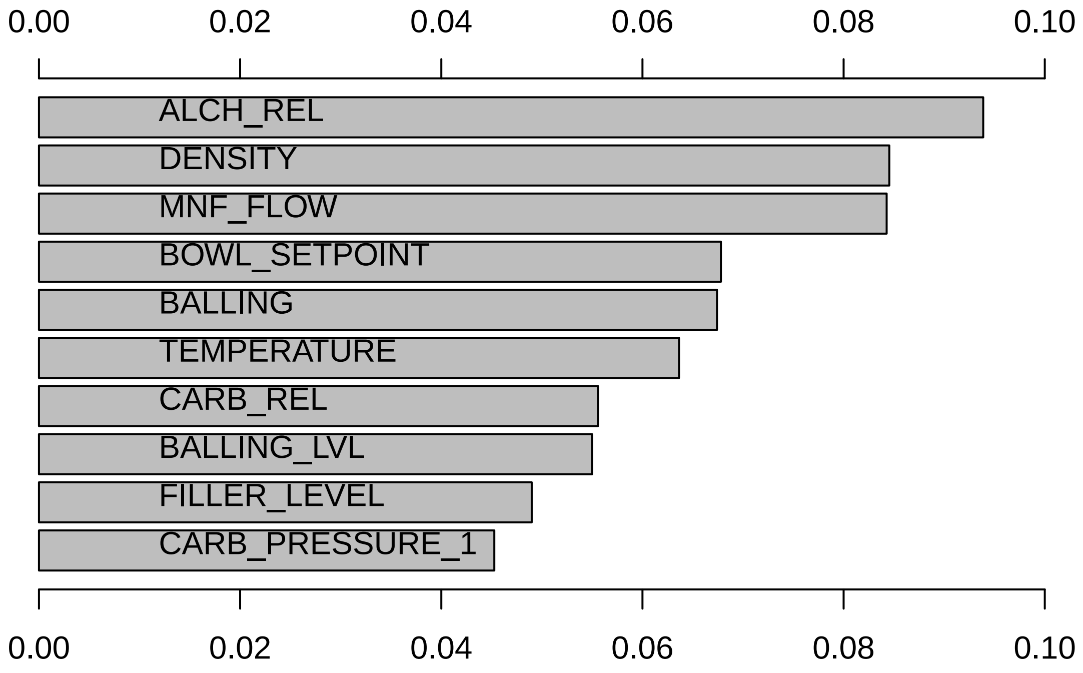
Model	sigma	C	R-Squared
Model SVM:RB:1	0.02	8	0.532
Model SVM:RB:2	0.0203	8	0.5757
Model SVM:RB:3	0.0167	8	0.5393

The ideal C was 8 for all three SVM models. Below are summaries of the estimated feature importance for the ten most important features in each of these SVM:RB models:

Model SVM:RB:1 Importance

```
y <- primary_train_df$PH
names(y) <- "y"
dat = cbind(primary_train_df |> select(-PH), y)
svmRB1Fit <- fit(y~., data = dat, model = "svm",
               kpar = list(sigma = .02), C = 8)
svmRB1.imp <- Importance(svmRB1Fit, data = dat)
L = list(runs = 1,sen = t(svmRB1.imp$imp),
         sresponses = svmRB1.imp$sresponses)
sen_vec <- as.numeric(L[["sen"]])
copy <- L
```

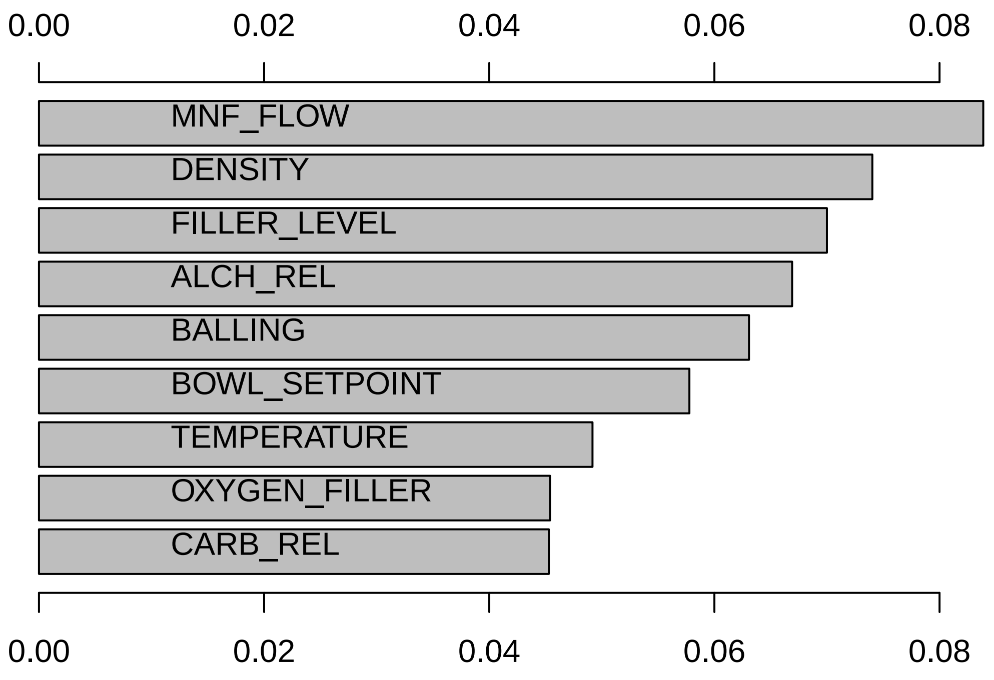
```r
delete <- c()
for (i in 1:length(sen_vec)){
    if (sen_vec[i] >= 0.045){
        next
    }else{
        delete <- append(delete, i)
    }
}
copy[["sen"]] <- t(as.matrix(copy[["sen"]][, -delete]))
copy[["sresponses"]] <- copy[["sresponses"]][-delete]
names <- c()
for (i in 1:length(copy[["sresponses"]])){
    n <- copy[["sresponses"]][[i]][["n"]]
    names <- append(names, n)
}
mgraph(copy, graph = "IMP", leg = names, col = "gray",
       PDF = "")
```



Model SVM:RB:2 Importance

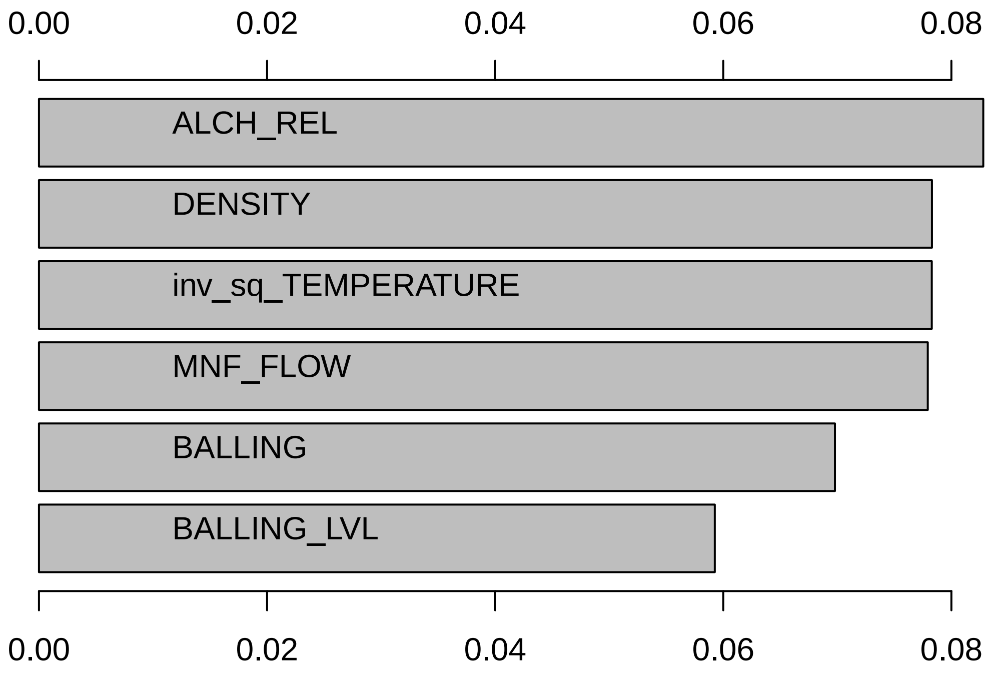
```r
y <- secondary_train_df$PH
names(y) <- "y"
dat = cbind(secondary_train_df |> select(-PH), y)
svmRB2Fit <- fit(y~., data = dat, model = "svm",
                kpar = list(sigma = .0203), C = 8)
svmRB2.imp <- Importance(svmRB2Fit, data = dat)
```

```r
L = list(runs = 1,sen = t(svmRB2.imp$imp),
         sresponses = svmRB2.imp$sresponses)
sen_vec <- as.numeric(L[["sen"]])
copy <- L
delete <- c()
for (i in 1:length(sen_vec)){
    if (sen_vec[i] >= 0.045){
        next
    }else{
        delete <- append(delete, i)
    }
}
copy[["sen"]] <- t(as.matrix(copy[["sen"]][, -delete]))
copy[["sresponses"]] <- copy[["sresponses"]][-delete]
names <- c()
for (i in 1:length(copy[["sresponses"]])){
    n <- copy[["sresponses"]][[i]][["n"]]
    names <- append(names, n)
}
mgraph(copy, graph = "IMP", leg = names, col = "gray",
       PDF = "")
```



Model SVM:RB:3 Importance

```r
y <- tertiary_train_df$PH
names(y) <- "y"
dat = cbind(tertiary_train_df |> select(-PH), y)
svmRB3Fit <- fit(y~., data = dat, model = "svm",
               kpar = list(sigma = .0167), C = 8)
svmRB3.imp <- Importance(svmRB3Fit, data = dat)
L = list(runs = 1,sen = t(svmRB3.imp$imp),
         sresponses = svmRB3.imp$sresponses)
sen_vec <- as.numeric(L[["sen"]])
copy <- L
delete <- c()
for (i in 1:length(sen_vec)){
    if (sen_vec[i] >= 0.045){
        next
    }else{
        delete <- append(delete, i)
    }
}
copy[["sen"]] <- t(as.matrix(copy[["sen"]][, -delete]))
copy[["sresponses"]] <- copy[["sresponses"]][-delete]
names <- c()
for (i in 1:length(copy[["sresponses"]])){
    n <- copy[["sresponses"]][[i]][["n"]]
    names <- append(names, n)
}
mgraph(copy, graph = "IMP", leg = names, col = "gray",
       PDF = "")
```



Tree Models:

Finally, we train three single tree regression models.

```
rpart1Tune <- train(primary_train_df |> select(-PH), primary_train_df$PH,
                    method = "rpart2",
                    tuneLength = 10,
                    trControl = trainControl(method = "cv"))
rpart2Tune <- train(secondary_train_df |> select(-PH), secondary_train_df$PH,
                    method = "rpart2",
                    tuneLength = 10,
                    trControl = trainControl(method = "cv"))
rpart3Tune <- train(tertiary_train_df |> select(-PH), tertiary_train_df$PH,
                    method = "rpart2",
                    tuneLength = 10,
                    trControl = trainControl(method = "cv"))
```

A summary of the best maxdepth and R-Squared for each of these tree models is below:

```
tree_mods <- c("Model Tree:1", "Model Tree:2", "Model Tree:3")
maxdepth <- c(rpart1Tune$bestTune$maxdepth, rpart2Tune$bestTune$maxdepth,
              rpart3Tune$bestTune$maxdepth)
rsq <- c(rpart1Tune$results |> filter(maxdepth == rpart1Tune$bestTune$maxdepth) |> select(Rsquared) |> a
         rpart2Tune$results |> filter(maxdepth == rpart2Tune$bestTune$maxdepth) |> select(Rsquared) |> a
         rpart3Tune$results |> filter(maxdepth == rpart3Tune$bestTune$maxdepth) |> select(Rsquared) |> a
```

```r
summ <- as.data.frame(cbind(tree_mods, maxdepth, round(rsq, 4)))
cols <- c("Model", "maxdepth", "R-Squared")
colnames(summ) <- cols
knitr::kable(summ, format = "simple")
```

Model	maxdepth	R-Squared
Model Tree:1	11	0.4738
Model Tree:2	11	0.4115
Model Tree:3	11	0.4659

All three tree models had a best maxdepth of 11. Below are summaries of the estimated feature importance for the ten most important features in each of these tree models:

```r
feature_importance <- varImp(rpart1Tune)
feature_importance <- feature_importance$importance |>
    rownames_to_column()
feature_importance <- feature_importance |>
    arrange(desc(Overall)) |>
    top_n(10)
cols <- c("Predictor", "Model Tree:1 Importance")
colnames(feature_importance) <- cols
knitr::kable(feature_importance, format = "simple")
```

Predictor	Model Tree:1 Importance
BRAND_CODE	100.00000
MNF_FLOW	79.72742
HYD_PRESSURE_3	76.87580
FILLER_SPEED	76.26406
CARB_PRESSURE_1	74.44439
USAGE_CONT	71.98577
PC_VOLUME	69.37967
OXYGEN_FILLER	65.27942
CARB_REL	57.61015
FILLER_LEVEL	49.09769

```r
feature_importance <- varImp(rpart2Tune)
feature_importance <- feature_importance$importance |>
    rownames_to_column()
feature_importance <- feature_importance |>
    arrange(desc(Overall)) |>
    top_n(10)
cols <- c("Predictor", "Model Tree:2 Importance")
colnames(feature_importance) <- cols
knitr::kable(feature_importance, format = "simple")
```

Predictor	Model Tree:2 Importance
PC_VOLUME	100.00000

Predictor	Model Tree:2 Importance
MNF_FLOW	81.18041
CARB_REL	81.13728
BRAND_CODE	80.01114
BOWL_SETPOINT	76.62495
FILLER_LEVEL	76.55623
TEMPERATURE	67.52719
CARB_PRESSURE_1	66.97522
ALCH_REL	63.45126
OXYGEN_FILLER	51.02602

```r
feature_importance <- varImp(rpart3Tune)
feature_importance <- feature_importance$importance |>
    rownames_to_column()
feature_importance <- feature_importance |>
    arrange(desc(Overall)) |>
    top_n(10)
cols <- c("Predictor", "Model Tree:3 Importance")
colnames(feature_importance) <- cols
knitr::kable(feature_importance, format = "simple")
```

Predictor	Model Tree:3 Importance
BRAND_CODE	100.00000
FILLER_SPEED	82.86299
FILLER_SPEED_sq	82.86299
MNF_FLOW	81.29959
USAGE_CONT	78.21451
USAGE_CONT_sq	78.21451
HYD_PRESSURE_3	77.57268
PC_VOLUME	75.38292
CARB_PRESSURE_1	73.52001
log_OXYGEN_FILLER	70.92788

Final Model Selection:

We will select the final model by looking at the models' Predictive R-Squared and Root Mean Squared Error (RMSE) measures using the test data we held out from the primary, secondary, and tertiary datasets. For each test dataset, we will select the model with the lowest RMSE (and hopefully the highest Predictive R-Squared). From that reduced selection of models, we will select the most appropriate model for the evaluation data, a determination that requires consideration of both the model itself and what changes were made to its underlying training data.

Below is a test data performance summary for all models operating on the primary dataset:

```r
test_pred1 <- predict(lm1, primary_test_df |> select(-PH))
test_rsq1 <- as.numeric(R2(test_pred1, primary_test_df$PH, form = "traditional"))
test_rmse1 <- as.numeric(RMSE(test_pred1, primary_test_df$PH))
row1 <- cbind("Model LM:1",
              as.character(round(test_rsq1, 4)),
              as.character(round(test_rmse1, 4)))
```

```r
test_pred2 <- predict(mars1, primary_test_df |> select(-PH))
test_rsq2 <- as.numeric(R2(test_pred2, primary_test_df$PH, form = "traditional"))
test_rmse2 <- as.numeric(RMSE(test_pred2, primary_test_df$PH))
row2 <- cbind("Model MARS:1",
              as.character(round(test_rsq2, 4)),
              as.character(round(test_rmse2, 4)))
test_pred3 <- predict(knn1, primary_test_df |> select(-PH))
test_rsq3 <- as.numeric(R2(test_pred3, primary_test_df$PH, form = "traditional"))
test_rmse3 <- as.numeric(RMSE(test_pred3, primary_test_df$PH))
row3 <- cbind("Model KNN:1",
              as.character(round(test_rsq3, 4)),
              as.character(round(test_rmse3, 4)))
mm_test <- model.matrix(~0+., data = primary_test_df |> select(-PH))
test_pred4 <- predict(svmRB1Tuned, mm_test)
test_rsq4 <- as.numeric(R2(test_pred4, primary_test_df$PH, form = "traditional"))
test_rmse4 <- as.numeric(RMSE(test_pred4, primary_test_df$PH))
row4 <- cbind("Model SVM:RB:1",
              as.character(round(test_rsq4, 4)),
              as.character(round(test_rmse4, 4)))
test_pred5 <- predict(rpart1Tune, primary_test_df |> select(-PH))
test_rsq5 <- as.numeric(R2(test_pred5, primary_test_df$PH, form = "traditional"))
test_rmse5 <- as.numeric(RMSE(test_pred5, primary_test_df$PH))
row5 <- cbind("Model Tree:1",
              as.character(round(test_rsq5, 4)),
              as.character(round(test_rmse5, 4)))
tbl <- as.data.frame(rbind(row1, row2, row3, row4, row5))
cols <- c("Model", "Predictive R-Squared", "RMSE")
colnames(tbl) <- cols
knitr::kable(tbl, format = "simple")
```

Model	Predictive R-Squared	RMSE
Model LM:1	0.3984	0.1357
Model MARS:1	0.4532	0.1294
Model KNN:1	0.4651	0.128
Model SVM:RB:1	0.567	0.1151
Model Tree:1	0.4432	0.1306

Model SVM:RB:1 has the lowest RMSE and highest Predictive R-Squared on the primary dataset.

Below is a test data performance summary for all models operating on the secondary dataset:

```r
test_pred1 <- predict(lm2, secondary_test_df |> select(-PH))
test_rsq1 <- as.numeric(R2(test_pred1, secondary_test_df$PH, form = "traditional"))
test_rmse1 <- as.numeric(RMSE(test_pred1, secondary_test_df$PH))
row1 <- cbind("Model LM:2",
              as.character(round(test_rsq1, 4)),
              as.character(round(test_rmse1, 4)))
test_pred2 <- predict(mars2, secondary_test_df |> select(-PH))
test_rsq2 <- as.numeric(R2(test_pred2, secondary_test_df$PH, form = "traditional"))
test_rmse2 <- as.numeric(RMSE(test_pred2, secondary_test_df$PH))
row2 <- cbind("Model MARS:2",
              as.character(round(test_rsq2, 4)),
              as.character(round(test_rmse2, 4)))
test_pred3 <- predict(knn2, secondary_test_df |> select(-PH))
test_rsq3 <- as.numeric(R2(test_pred3, secondary_test_df$PH, form = "traditional"))
test_rmse3 <- as.numeric(RMSE(test_pred3, secondary_test_df$PH))
row3 <- cbind("Model KNN:2",
              as.character(round(test_rsq3, 4)),
              as.character(round(test_rmse3, 4)))
mm_test <- model.matrix(~0+., data = secondary_test_df |> select(-PH))
test_pred4 <- predict(svmRB2Tuned, mm_test)
test_rsq4 <- as.numeric(R2(test_pred4, secondary_test_df$PH, form = "traditional"))
test_rmse4 <- as.numeric(RMSE(test_pred4, secondary_test_df$PH))
row4 <- cbind("Model SVM:RB:2",
              as.character(round(test_rsq4, 4)),
              as.character(round(test_rmse4, 4)))
test_pred5 <- predict(rpart2Tune, secondary_test_df |> select(-PH))
test_rsq5 <- as.numeric(R2(test_pred5, secondary_test_df$PH, form = "traditional"))
test_rmse5 <- as.numeric(RMSE(test_pred5, secondary_test_df$PH))
row5 <- cbind("Model Tree:2",
              as.character(round(test_rsq5, 4)),
              as.character(round(test_rmse5, 4)))
tbl <- as.data.frame(rbind(row1, row2, row3, row4, row5))
cols <- c("Model", "Predictive R-Squared", "RMSE")
colnames(tbl) <- cols
knitr::kable(tbl, format = "simple")
```

Model	Predictive R-Squared	RMSE
Model LM:2	0.4345	0.13
Model MARS:2	-2.7649	0.3355
Model KNN:2	0.5322	0.1183
Model SVM:RB:2	0.6224	0.1063
Model Tree:2	0.4217	0.1315

Model SVM:RB:2 has the lowest RMSE and highest Predictive R-Squared on the secondary dataset. **Model MARS:2** performs particularly poorly on this dataset.

```
test_pred1 <- predict(lm3, tertiary_test_df |> select(-PH))
test_rsq1 <- as.numeric(R2(test_pred1, tertiary_test_df$PH, form = "traditional"))
test_rmse1 <- as.numeric(RMSE(test_pred1, tertiary_test_df$PH))
row1 <- cbind("Model LM:3",
              as.character(round(test_rsq1, 4)),
              as.character(round(test_rmse1, 4)))
test_pred2 <- predict(mars3, tertiary_test_df |> select(-PH))
test_rsq2 <- as.numeric(R2(test_pred2, tertiary_test_df$PH, form = "traditional"))
test_rmse2 <- as.numeric(RMSE(test_pred2, tertiary_test_df$PH))
row2 <- cbind("Model MARS:3",
              as.character(round(test_rsq2, 4)),
              as.character(round(test_rmse2, 4)))
test_pred3 <- predict(knn3, tertiary_test_df |> select(-PH))
test_rsq3 <- as.numeric(R2(test_pred3, tertiary_test_df$PH, form = "traditional"))
test_rmse3 <- as.numeric(RMSE(test_pred3, tertiary_test_df$PH))
row3 <- cbind("Model KNN:3",
```

```r
               as.character(round(test_rsq3, 4)),
               as.character(round(test_rmse3, 4)))
mm_test <- model.matrix(~0+., data = tertiary_test_df |> select(-PH))
test_pred4 <- predict(svmRB3Tuned, mm_test)
test_rsq4 <- as.numeric(R2(test_pred4, tertiary_test_df$PH, form = "traditional"))
test_rmse4 <- as.numeric(RMSE(test_pred4, tertiary_test_df$PH))
row4 <- cbind("Model SVM:RB:3",
               as.character(round(test_rsq4, 4)),
               as.character(round(test_rmse4, 4)))
test_pred5 <- predict(rpart3Tune, tertiary_test_df |> select(-PH))
test_rsq5 <- as.numeric(R2(test_pred5, tertiary_test_df$PH, form = "traditional"))
test_rmse5 <- as.numeric(RMSE(test_pred5, tertiary_test_df$PH))
row5 <- cbind("Model Tree:3",
               as.character(round(test_rsq5, 4)),
               as.character(round(test_rmse5, 4)))
tbl <- as.data.frame(rbind(row1, row2, row3, row4, row5))
cols <- c("Model", "Predictive R-Squared", "RMSE")
colnames(tbl) <- cols
knitr::kable(tbl, format = "simple")
```

Model	Predictive R-Squared	RMSE
Model LM:3	0.3993	0.1356
Model MARS:3	0.4853	0.1255
Model KNN:3	0.4599	0.1286
Model SVM:RB:3	0.5571	0.1164
Model Tree:3	0.4432	0.1306

Model SVM:RB:3 has the lowest RMSE and highest Predictive R-Squared on the tertiary dataset.

While **Model SVM:RB:2** has the lowest RMSE (and the highest Predictive R-Squared), its underlying data uses only complete cases. Since the evaluation data contains `NA` values, and SVM models can't handle missing predictor data, we choose the second best performer: **Model SVM:RB:1**. It uses imputed data, and we can impute the missing values in the evaluation data the same way so that `PH` can be predicted for all observations.

Final Model Evaluation:

We make our `PH` predictions using **Model SVM:RB:1** and save the predictions made to an Excel file: "Student_Evaluation_w_Predictions.xlsx."

```r
missing_val_cols <- find_cols_na(eval_df |> select(-PH))
eval_df <- eval_df |>
    mutate(BRAND_CODE = factor(BRAND_CODE, exclude = NULL)) |>
    VIM::kNN(variable = missing_val_cols, k = 15, dist_var = dist_vars,
             weights = wts, numFun = median, imp_var = FALSE)
mm_pred <- model.matrix(~0+., data = eval_df |> select(-PH))
eval_df <- eval_df |>
    mutate(PH = predict(svmRB1Tuned, mm_pred))
write_xlsx(eval_df, "Student_Evaluation_w_Predictions.xlsx")
```