Modeling

Alexia Wells

2025-04-19

Contents

Read Me **2**

Introduction **2**
Project Goal and Purpose of the Notebook . . . 2
Business Problem . . . 2
Analytical Problem . . . 2

Data Preparation **2**
Checking the Data and Feature Engineering . . . 3
Visualizations and Tables . . . 5
Bar Plots . . . 5
Distribution of Units Ordered, Loaded, and Delivered . . . 6

Seperate DataFrames **6**
Running Data on the Server vs. Not . . . 7
When Using the Server . . . 7
Method When Not Using Server . . . 7

Modeling Process **7**
There is Multicollinearity Present in the Data . . . 7
Transformations Do Not Help . . . 9
Linear Regression on Total Units Ordered for 2023 . . . 9

Linear Regression on Total Units Ordered for 2024 **11**

Model Performance **12**
2023 . . . 12
Results . . . 12
2024 . . . 12
Results . . . 13
Similarities Between 2023 and 2024 Linear Regression Models . . . 13
Differences Between 2023 and 2024 Linear Regression Models . . . 13

Results (do this as a group too) **13**
Current Concerns with Modeling . . . 13
Call to Action/What Reccomendations Can We Give Anyway . . . 14
Next Steps/ Where could we take modeling in the future . . . 14
Improvements to the Data . . . 14

Read Me

If you would like to run the contents of this document on your own, please remove eval=FALSE in the modeling code chunks. They are set to false because these models take a long time computationally. To get the same results, please run the data and models on a server. Thank you.

Introduction

Project Goal and Purpose of the Notebook

The purpose of this notebook is to develop a predictive analytics model that estimates a client's annual sales volume, helping to identify those likely to exceed the optimal growth threshold of 400 gallons. Moreover, the notebook will start with data preparation and go into model building, evaluation, and results.

Business Problem

Swire Coca-Cola is a leading tech company that puts clients at the center of their work. As a bottler and distributor throughout 13 Western states, operational efficiency is a necessity to remain successful and create long-term relationships with clientele. Swire Coca-Cola must avoid prematurely moving high-growth potential customers to white truck delivery (Alternate Route to Market) as this could severely impact revenue growth. The purpose of this project is to reasonably maximize sales by predicting which clients have the potential to reach the high-performing SCCU optimal threshold of 400 gallons annually.

Analytical Problem

Several models will be created in attempt to address Swire Coca-Cola's business needs. A balanced approach will be used to inform client growth by taking into account historical sales data and other customer characteristics.

Data Preparation

A majority of the data preparation was completed in the exploratory data analysis portion of the project. However, there were still important steps taken to fit model requirements. To start off, I had to drop several columns: ordered cases, loaded cases, delivered cases, ordered gallons, loaded gallons, delivered gallons, transaction date, first delivery date, on boarding date, full address, zip code, and primary group number.

I removed several of those columns to avoid issues of multicollinearity since it was redundant information. Other columns like full address, zip code, and primary group number were removed out of simplifying the modeling process.

Considering that I had to remove several columns for multicollinearity, I wanted to avoid data loss as much as I could. For that reason, I feature engineered 3 columns: month, load difference, and shipment difference. Month was extracted from transaction date while loaded difference is a calculation representing the amount of ordered - loaded gallons and cases. This was similar for shipment difference, loaded - delivered. I also factored several columns.

The datasets that I read in were originally randomly split. This method is encouraged for when time dependencies may not impact the data. My belief is that these time dependencies would be important for our modeling, so I ended up filtering the datasets for train and test by year. I also was able to get access to a server to run my data faster, so I was able to use the full datasets instead of splitting them down into smaller ones.

```
# Load the libraries
library(vroom) # for quickly reading in/writing out
library(tidyverse)
library(tidymodels)
```

```r
library(lubridate) # to seperate date
library(DataExplorer)
library(caret)
library(car) # helps check for multicollinarity
library(MASS) # for boxcox
```

```r
# Read in the data, train and test should be included here
train <- vroom("train_set.csv")
test <- vroom("test_set.csv")
```

Checking the Data and Feature Engineering

Removed some columns due to redundancy (I.E Transaction_Date, All ordered, loaded, delivered gallons and casese). I actually ran into memory issues while running linear models - "Error: vector memory limit of 16.0 Gb reached, see mem.maxVSize()."

Feature engineered month column and two columns representing disparity between ordered/loaded, and loaded/delivered

```r
# Checking Column Names
colnames(train)
colnames(test)

# Check for duplicated data, there is none
anyDuplicated(train)

# Will need to factor the train and test set
str(train)
```

```r
# LOADED_DIFFERENCE = Ordered-loaded
# SHIPMENT_DIFFERENCE = loaded-delivered
train_set <- train |>
   mutate(across(c(CUSTOMER_NUMBER, ORDER_TYPE, PRIMARY_GROUP_NUMBER,
                   FREQUENT_ORDER_TYPE, COLD_DRINK_CHANNEL,
                   TRADE_CHANNEL, SUB_TRADE_CHANNEL, LOCAL_MARKET_PARTNER,
                   CO2_CUSTOMER, ZIP_CODE, ORDER_GROUPING), as.factor)) |>
  mutate(across(c(TRANSACTION_DATE, FIRST_DELIVERY_DATE, ON_BOARDING_DATE),
                ~ as.Date(., format = "%m/%d/%Y"))) |>
  mutate(MONTH = month(TRANSACTION_DATE), LOADED_DIFFERENCE =
           TOTAL_UNITS_ORDERED - TOTAL_UNITS_LOADED, SHIPMENT_DIFFERENCE =
           TOTAL_UNITS_LOADED - TOTAL_UNITS_DELIVERED) |>
  mutate(ORDER_GROUPING = ifelse(ORDER_GROUPING == "Customers Ordering Less than 400", 0, 1)) |> # if i
  mutate(across(c(WEEK, MONTH), as.factor)) |>
  mutate(TOTAL_UNITS_ORDERED= ifelse(
    TOTAL_UNITS_ORDERED == min(TOTAL_UNITS_ORDERED),
    TOTAL_UNITS_ORDERED + abs(min(TOTAL_UNITS_ORDERED)) + 0.01,
    TOTAL_UNITS_ORDERED)) |>
  dplyr::select(-c(ORDERED_CASES,LOADED_CASES,DELIVERED_CASES,ORDERED_GALLONS,
                   LOADED_GALLONS, DELIVERED_GALLONS, TRANSACTION_DATE,
                   full.address, ZIP_CODE, PRIMARY_GROUP_NUMBER))

# Now do the same to test
test_set <- test |>
   mutate(across(c(CUSTOMER_NUMBER, ORDER_TYPE, PRIMARY_GROUP_NUMBER,
                  FREQUENT_ORDER_TYPE, COLD_DRINK_CHANNEL,
```

```
                  TRADE_CHANNEL, SUB_TRADE_CHANNEL, LOCAL_MARKET_PARTNER,
                  CO2_CUSTOMER, ZIP_CODE, ORDER_GROUPING), as.factor)) |>
  mutate(across(c(TRANSACTION_DATE, FIRST_DELIVERY_DATE, ON_BOARDING_DATE),
                ~ as.Date(., format = "%m/%d/%Y"))) |>
  mutate(MONTH = month(TRANSACTION_DATE), LOADED_DIFFERENCE =
           TOTAL_UNITS_ORDERED - TOTAL_UNITS_LOADED, SHIPMENT_DIFFERENCE =
           TOTAL_UNITS_LOADED - TOTAL_UNITS_DELIVERED) |>
  mutate(ORDER_GROUPING = ifelse(ORDER_GROUPING == "Customers Ordering Less than 400", 0, 1)) |> # if i
  mutate(across(c(WEEK, MONTH), as.factor)) |>
  mutate(TOTAL_UNITS_ORDERED= ifelse(
    TOTAL_UNITS_ORDERED == min(TOTAL_UNITS_ORDERED),
    TOTAL_UNITS_ORDERED + abs(min(TOTAL_UNITS_ORDERED)) + 0.01,
    TOTAL_UNITS_ORDERED)) |>
  dplyr::select(-c(ORDERED_CASES,LOADED_CASES,DELIVERED_CASES,ORDERED_GALLONS,
                   LOADED_GALLONS, DELIVERED_GALLONS, TRANSACTION_DATE,
                   full.address, ZIP_CODE, PRIMARY_GROUP_NUMBER))

# There is near zero variance for shipment difference, so will need to consider whether it should be ke
nearZeroVar(train_set, saveMetrics = TRUE)
```

```
##                               freqRatio percentUnique zeroVar   nzv
## WEEK                           1.014752  0.0071295480   FALSE FALSE
## YEAR                           1.014815  0.0002742134   FALSE FALSE
## CUSTOMER_NUMBER                1.228125  2.9099524377   FALSE FALSE
## ORDER_TYPE                     1.119873  0.0009597468   FALSE FALSE
## FREQUENT_ORDER_TYPE            3.937187  0.0008226402   FALSE FALSE
## FIRST_DELIVERY_DATE            1.037126  0.3195956998   FALSE FALSE
## ON_BOARDING_DATE               1.279048  0.7927508950   FALSE FALSE
## COLD_DRINK_CHANNEL             3.036313  0.0012339602   FALSE FALSE
## TRADE_CHANNEL                  1.502847  0.0035647740   FALSE FALSE
## SUB_TRADE_CHANNEL              1.872574  0.0065811212   FALSE FALSE
## LOCAL_MARKET_PARTNER           4.768100  0.0002742134   FALSE FALSE
## CO2_CUSTOMER                   1.590430  0.0002742134   FALSE FALSE
## TOTAL_UNITS_ORDERED            1.020676  2.0228721384   FALSE FALSE
## TOTAL_UNITS_LOADED             1.040565  1.9112672909   FALSE FALSE
## TOTAL_UNITS_DELIVERED          1.035681  2.1834240751   FALSE FALSE
## ORDER_GROUPING                 1.297773  0.0002742134   FALSE FALSE
## CUSTOMER_GROUP_NUMBER_ROLLUP   3.201493  1.8577956809   FALSE FALSE
## Total_Delivery_Cost            4.634103  2.6931867571   FALSE FALSE
## MONTH                          1.091506  0.0016452803   FALSE FALSE
## LOADED_DIFFERENCE             12.598464  0.4085779431   FALSE FALSE
## SHIPMENT_DIFFERENCE           28.998827  0.2954649219   FALSE  TRUE
```

```
nearZeroVar(test_set, saveMetrics = TRUE)
```

```
##                               freqRatio percentUnique zeroVar   nzv
## WEEK                           1.010082  0.0164462760   FALSE FALSE
## YEAR                           1.011995  0.0006325491   FALSE FALSE
## CUSTOMER_NUMBER                1.009554  2.8774657554   FALSE FALSE
## ORDER_TYPE                     1.079598  0.0022139218   FALSE FALSE
## FREQUENT_ORDER_TYPE            4.174676  0.0018976472   FALSE FALSE
## FIRST_DELIVERY_DATE            1.010263  0.6581673156   FALSE FALSE
## ON_BOARDING_DATE               1.071346  1.2660469794   FALSE FALSE
## COLD_DRINK_CHANNEL             3.064387  0.0028464709   FALSE FALSE
## TRADE_CHANNEL                  1.501418  0.0079068635   FALSE FALSE
```

```
## SUB_TRADE_CHANNEL              2.030238  0.0139160797   FALSE FALSE
## LOCAL_MARKET_PARTNER           4.414615  0.0006325491   FALSE FALSE
## CO2_CUSTOMER                   1.511805  0.0006325491   FALSE FALSE
## TOTAL_UNITS_ORDERED            1.002148  2.5033129758   FALSE FALSE
## TOTAL_UNITS_LOADED             1.032266  2.3992586525   FALSE FALSE
## TOTAL_UNITS_DELIVERED          1.032654  2.7142680933   FALSE FALSE
## ORDER_GROUPING                 1.333956  0.0006325491   FALSE FALSE
## CUSTOMER_GROUP_NUMBER_ROLLUP   2.659827  1.9419256692   FALSE FALSE
## Total_Delivery_Cost            2.579747  3.1447177408   FALSE FALSE
## MONTH                          1.089678  0.0037952945   FALSE FALSE
## LOADED_DIFFERENCE             12.599481  0.4984486734   FALSE FALSE
## SHIPMENT_DIFFERENCE           29.403838  0.3924967028   FALSE  TRUE
```

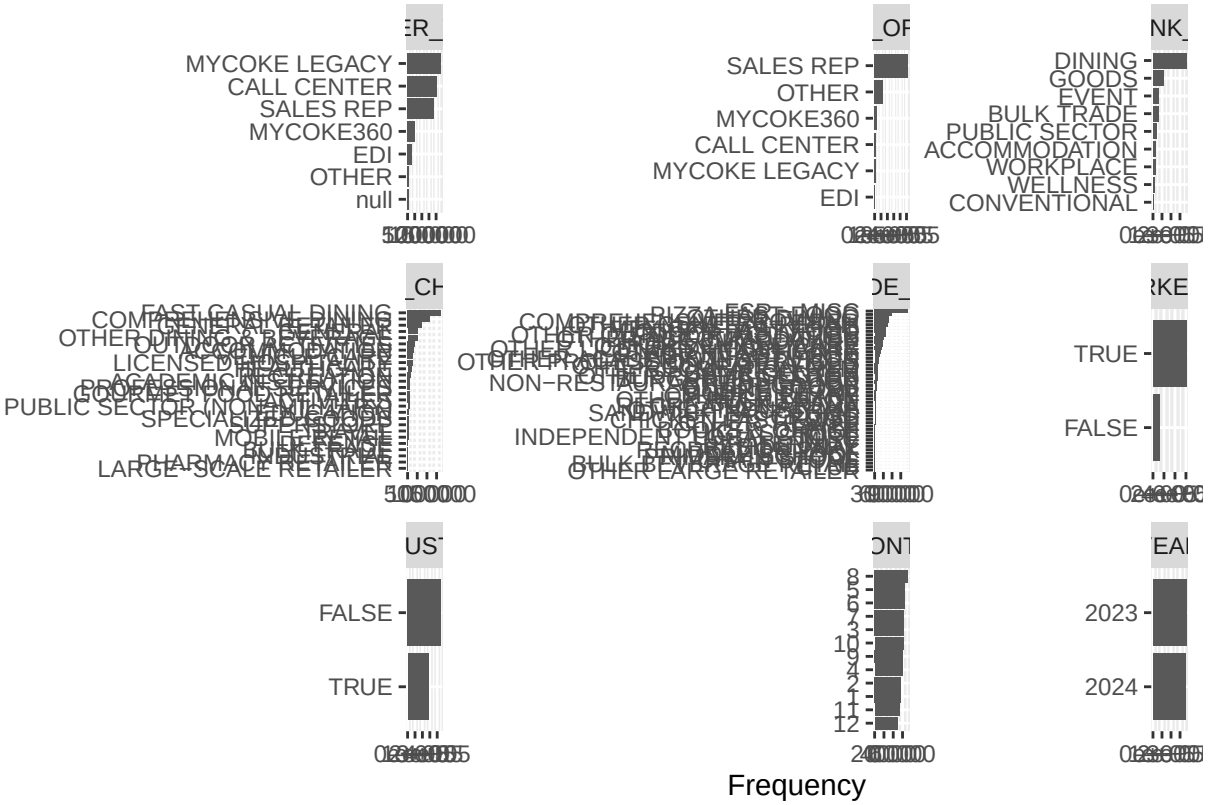
```
# Check that both datasets look correct
str(train_set)
str(test_set)
```

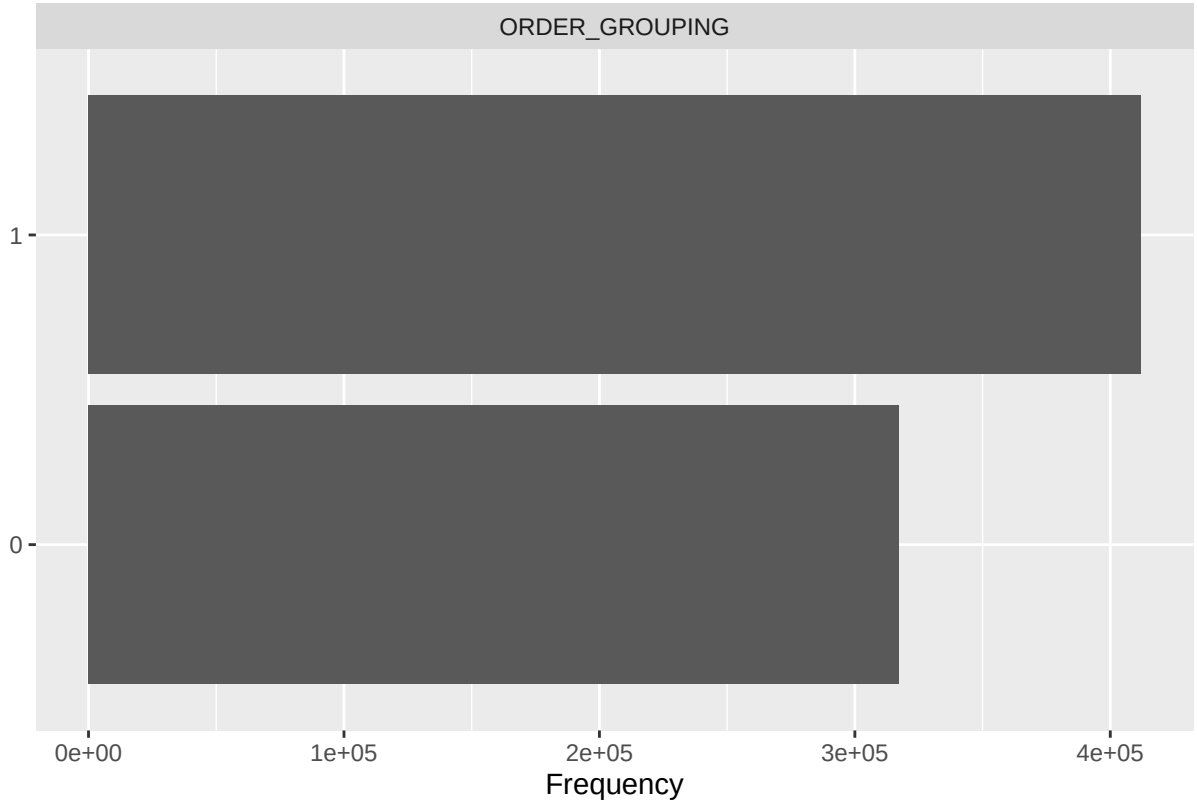
Visualizations and Tables

Bar Plots

```
plot_bar(train_set)
```

```
## 4 columns ignored with more than 50 categories.
## WEEK: 52 categories
## CUSTOMER_NUMBER: 21224 categories
## FIRST_DELIVERY_DATE: 2331 categories
## ON_BOARDING_DATE: 5782 categories
```





Distribution of Units Ordered, Loaded, and Delivered

```
summary(train_set[13:15])
```

```
##  TOTAL_UNITS_ORDERED TOTAL_UNITS_LOADED TOTAL_UNITS_DELIVERED
##  Min.   :   0.01     Min.   :   0.00    Min.   :-1894.00
##  1st Qu.:   9.00     1st Qu.:   8.00    1st Qu.:    7.50
##  Median :  15.50     Median :  15.00    Median :   15.00
##  Mean   :  36.54     Mean   :  35.35    Mean   :   34.19
##  3rd Qu.:  30.00     3rd Qu.:  29.00    3rd Qu.:   28.00
##  Max.   :8479.89     Max.   :8171.56    Max.   : 8069.48
```

Seperate DataFrames

```
# Separate the data into 2023 and 2024 subsets

# By year 2023
train_2023 <- train_set %>% filter(YEAR == 2023)
test_2023 <- test_set %>% filter(YEAR == 2023)

# filtered by year 2024
train_2024 <- train_set %>% filter(YEAR == 2024)
test_2024 <- test_set %>% filter(YEAR == 2024)

# Full 2024 dataset
full_2024 <- dplyr::bind_rows(train_2024, test_2024)
```

Running Data on the Server vs. Not

When Using the Server

```
set.seed(123)

# 2023
train_2023_data <- train_2023 |>
  dplyr::select(-YEAR)

test_2023 <- test_2023 |>
  dplyr::select(-YEAR)

# 2024
train_2024_data <- train_2024 |>
  dplyr::select(-YEAR)

test_2024 <- test_2024 |>
  dplyr::select(-YEAR)
```

Method When Not Using Server

Keep in mind this section of code was run using a server, so results will be different if you don't use the full datasets.

```
set.seed(123)

# Get samples of the data, 367,361
train_2023_data <- train_2023[sample(nrow(train_2023), 6000), ] |>
  dplyr::select(-YEAR)

# test data 2023
test_2023 <- test_2023[sample(nrow(test_2023), 6000), ] |>
  dplyr::select(-YEAR)

# Get samples of the data
train_2024_data <- train_2024[sample(nrow(train_2024), 6000), ] |>
  dplyr::select(-YEAR)

# test data 2023
test_2024 <- test_2024[sample(nrow(test_2024), 6000), ] |>
  dplyr::select(-YEAR)
```

Modeling Process

Discuss the candidate models you considered, the process for selecting the best model, cross-validation procedures, hyperparameter tuning.

There is Multicollinearity Present in the Data

With the current data, it gives an error: "there are aliased coefficients in the model". One way to avoid this is to get rid of one of the delivered/ordered/loaded columns.

```r
# Train the model on 2023 data
simple.lm <- lm(TOTAL_UNITS_ORDERED ~ ., data = train_2023_data)

# Write out the summary because it is too long to be normally displayed
# capture.output(summary(simple.lm), file = "linear_model_summary.txt")
```

```r
# Check if there are aliased coefficients
alias(simple.lm)

# Check high collinearity, commented out since it will cause an error
# vif(simple.lm)

# Check for correlation
cor_matrix <- cor(train_2023_data[, sapply(train_2023_data, is.numeric)])
cor_matrix
```

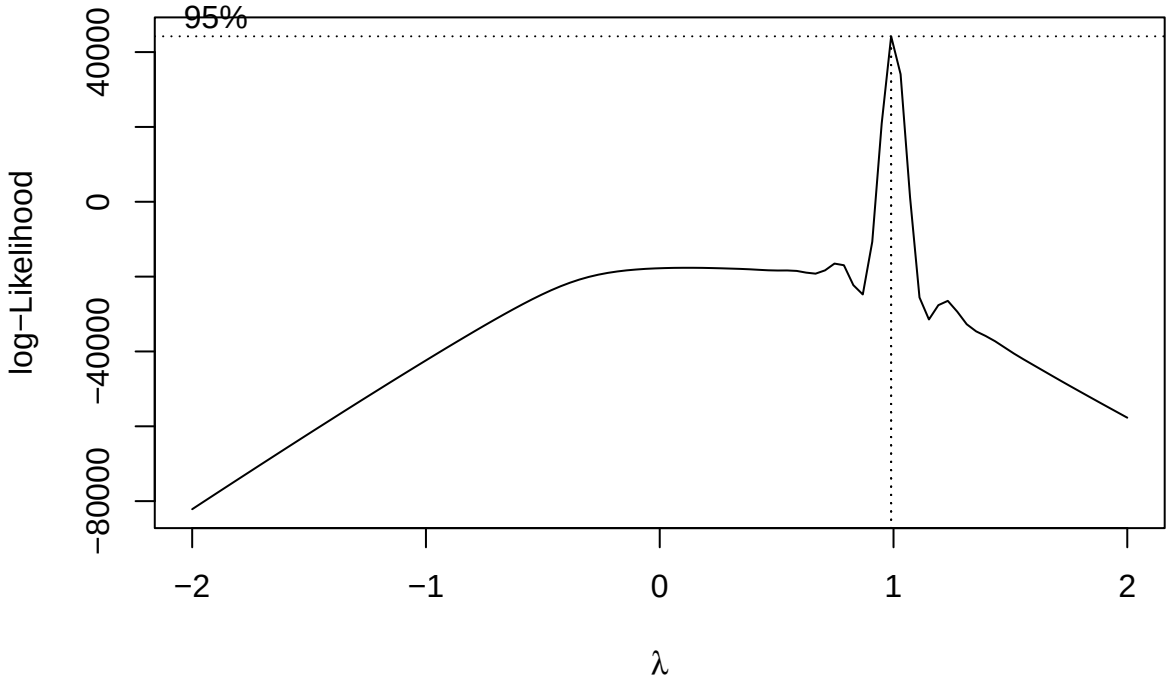
```r
# The minimum value is 0, and it needs to be a positive number
summary(train_2023_data$TOTAL_UNITS_ORDERED)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##    0.010    7.908   15.000   36.058   27.625 4421.620
```

```r
min(train_2023_data$TOTAL_UNITS_ORDERED)
```

```
## [1] 0.01
```

```r
# Here just adding slight change to the minimum
# Getting boxcox result
boxcox_results <-boxcox(simple.lm)
```




```r
# Lambda is 0.1414141
lambda <- boxcox_results$x[which.max(boxcox_results$y)]
lambda
```

```
## [1] 0.989899
```

Transformations Do Not Help

I attempted a transformation using step_log and step_boxcox, in my recipes but both ended ended up lowering results. They also did not improve model assumptions like normality and equal variance.

Linear Regression on Total Units Ordered for 2023

For reference, the above model contained all the variables from the merged dataset. The recipe below removes total units loaded and total units delivered. If only delivered is kept, r-squared is around 0.97 which is unreasonable for real world data and indicates significant overfitting. This is because those columns are extremely similar to total units ordered, which is the response variable. If both are removed, r-squared for the 2023 train and test set is around 0.6332.

Other removed variables in the recipe were due to high multicollinearity and aliased coefficients. The aliased coefficients were: sub_trade_channel_fsr..misc,sub_trade_channel_other.dining. The high vif/multicollinearity variables are: trade_channel_general (28.59), trade_channel_outdoor.activities (20.78), sub_trade_channel_comprehensive.provider (34.31), total_units_loaded (34.340290), and total_units_delivered is (34.150576).

```
my_recipe <- recipe(TOTAL_UNITS_ORDERED ~ ., data = train_2023_data) %>%
  step_novel(all_nominal(), new_level = "Unknown") %>%
  step_dummy(all_nominal(), -all_outcomes()) %>% # should prevent dummy variable trap
  step_nzv(all_predictors()) |>
  step_select(-TOTAL_UNITS_LOADED, -TOTAL_UNITS_DELIVERED,
              -SUB_TRADE_CHANNEL_FSR...MISC, -SUB_TRADE_CHANNEL_OTHER.DINING,
              -TRADE_CHANNEL_GENERAL, -TRADE_CHANNEL_OUTDOOR.ACTIVITIES,
              -SUB_TRADE_CHANNEL_COMPREHENSIVE.PROVIDER) %>%
  step_normalize(all_numeric_predictors())

# Prepare the recipe with the training data
prepped <- prep(my_recipe, training = train_2023_data)

# Apply the recipe to the training and test data
train_transformed23 <- bake(prepped, new_data = train_2023_data)
test_transformed23 <- bake(prepped, new_data = test_2023)

# Now fit the model on the training set
model <- lm(TOTAL_UNITS_ORDERED ~ ., data = train_transformed23)
summary(model) # 0.791, Adjusted R-squared:  0.7897

# Make predictions on the test set
predictions_2023 <- predict(model, newdata = test_transformed23)
# predictions_2023

# Compare predictions to actual value
comparison <- data.frame(Actual = test_transformed23$TOTAL_UNITS_ORDERED,
                         Predicted = predictions_2023)

# Evaluate the model performance
print("R-squared of test data vs. actual data:")
r2_value <- cor(test_transformed23$TOTAL_UNITS_ORDERED, predictions_2023)^2
r2_value # 0.6027491

# RMSE calculation
print("RMSE calculation:")
```

```r
rmse_value <- sqrt(mean((test_transformed23$TOTAL_UNITS_ORDERED - predictions_2023)^2))
print(rmse_value)

# Check what concerns are going on
print("Aliased coefficients:")
alias(model)

print("Multicollinearity:")
vif(model)

# Done with checking
end_time <- Sys.time()

# Print runtime
execution_time <- end_time - start_time
print("Execution time for first model:")
print(execution_time)
print("End of 2023 training and test, on to testing 2024:")

# Want to check how long it takes to run the predictions on the 2023 model
start_time_2 <- Sys.time()

# Apply the recipe to the 2024 full dataset
full_transformed24 <- bake(prepped, new_data = full_2024)

# Make predictions on the test set
predictions_2024 <- predict(model, newdata = full_transformed24)

# Compare predictions to actual value
comparison <- data.frame(Actual = full_2024$TOTAL_UNITS_ORDERED,
                         Predicted = predictions_2024)

# Evaluate the model performance
print("R-squared of test data vs. actual data:")
r2_value <- cor(full_2024$TOTAL_UNITS_ORDERED, predictions_2024)^2
r2_value # 0.6027491

# RMSE calculation
print("RMSE calculation:")
rmse_value <- sqrt(mean((full_2024$TOTAL_UNITS_ORDERED - predictions_2024)^2))
print(rmse_value)

# Check what concerns are going on
print("Aliased coefficients:")
alias(model)

print("Multicollinearity:")
vif(model)

# Done with checking
end_time_2 <- Sys.time()

# Print runtime
```

```r
execution_time_2 <- end_time_2 - start_time_2
print("Execution time for 2024 predictions")
print(execution_time_2)
```

```r
### Checking Model Assumptions
# Normality
print("Normality:")
qqnorm(residuals(model))
qqline(residuals(model), col = "red")

# Equal Variance
print("Equal Variance:")
ggplot(data = train_2023_data, mapping = aes(x = stdres(model), y = fitted(model))) + geom_point() +
  geom_smooth(method = "loess", se = FALSE, color = "red")  # Loess smoothing line
```

Linear Regression on Total Units Ordered for 2024

Since we already know there is multicollinearity in the data, the recipe below removes total units loaded and total units delivered. The r-squared for the 2024 train and test set is around 0.5744332. Since we were not given 2025 data, we cannot test the model in the same way we did for the 2023 model where we used the previous years full data as a test set.

Additionally, the same method of identifying high vif and aliased coefficients was used. The aliased coefficients were: sub_trade_channel_fsr..misc,sub_trade_channel_other.dining. The high vif/multicollinearity variables are: trade_channel_general(52.88), trade_channel_outdoor.activties (15.06), sub_trade_channel_comprehensive.provider (59.30), and sub_trade_channel_other.outdoor.activities (15.060849).

```r
# Want to check how long it takes to run the predictions for the 2024 model
start_time_3 <- Sys.time()

my_recipe <- recipe(TOTAL_UNITS_ORDERED ~ ., data = train_2024_data) %>%
  step_novel(all_nominal(), new_level = "Unknown") %>%
  step_dummy(all_nominal(), -all_outcomes()) %>% # should prevent dummy variable trap
  step_nzv(all_predictors()) |>
  step_select(-TOTAL_UNITS_LOADED, -TOTAL_UNITS_DELIVERED, - SUB_TRADE_CHANNEL_FSR...MISC,
              - SUB_TRADE_CHANNEL_OTHER.DINING, -SUB_TRADE_CHANNEL_COMPREHENSIVE.PROVIDER,-TRADE_CHANNEL
              -TRADE_CHANNEL_OUTDOOR.ACTIVITIES, -SUB_TRADE_CHANNEL_OTHER.OUTDOOR.ACTIVITIES) %>%
  step_normalize(all_numeric_predictors())

# Prepare the recipe with the training data
prepped <- prep(my_recipe, training = train_2024_data)

# Apply the recipe to the training and test data
train_transformed24 <- bake(prepped, new_data = train_2024_data)
test_transformed24 <- bake(prepped, new_data = test_2024)

# Now fit the model on the training set
model <- lm(TOTAL_UNITS_ORDERED ~ ., data = train_transformed24)
summary(model) #

# Make predictions on the test set
predictions_2024 <- predict(model, newdata = test_transformed24)
```

```r
# Evaluate the model performance
print("R-squared calculation:")
r2_value <- cor(test_transformed24$TOTAL_UNITS_ORDERED, predictions_2024)^2
r2_value

# RMSE calculation
print("RMSE calculation:")
rmse_value <- sqrt(mean((test_transformed24$TOTAL_UNITS_ORDERED - predictions_2024)^2))
print(rmse_value)

# Done with checking
end_time_3 <- Sys.time()

# Print runtime
execution_time_3 <- end_time_3 - start_time_3
print("Execution time for 2024 model, using 2024 train and test")
print(execution_time_3)
```

Model Performance

Describe the performance characteristics of your best model, including run time. What is the train set and test set performance?

2023

The 2023 model takes 2.195178 hours to run using the full data on a server. The Multiple R-squared and Adjusted R-squared for the training data have the same value of 0.6332. The r-squared for the test predictions minus the actual data is 0.5634993 while the RMSE is 81.87398. I would argue these results are decent considering the issues found within the data.

The next step I took was running the full 2024 data as the test_set using the same model. Turns out, the 2024 predictions take 38.02309 minutes to run using the full data on a server. The r-squared for the test predictions minus the actual data is 0.572642 while the RMSE is 92.60317. The good news is that we are seeing consistency here!

Results

< 0: FIRST_DELIVERY_DATE, ON_BOARDING_DATE, CUSTOMER_GROUP_NUMBER_ROLLUP, Total_Delivery_Cost, LOADED_DIFFERENCE ORDER_TYPE_MYCOKE.LEGACY, ORDER_TYPE_SALES.REP, COLD_DRINK_CHANNEL_BULK.TRADE, COLD_DRINK_CHANNEL_DINING, COLD_DRINK_CHANNEL_EVENT COLD_DRINK_CHANNEL_GOODS, TRADE_CHANNEL_GENERAL.RETAILER, TRADE_CHANNEL_OTHER.DINI SUB_TRADE_CHANNEL_MEXICAN.FAST.FOOD, SUB_TRADE_CHANNEL_OTHER.FAST.FOOD, SUB_TRADE_CHANNEL_OTHER.GENERAL.RETAIL, SUB_TRADE_CHANNEL_OTHER.OUTDOOR.ACTIVITIES, SUB_TRADE_CHANNEL_PIZZA.FAST.FOOD, LOCAL_MARKET_PARTNER_TRUE., LOCAL_MARKET_PARTNER_TRUE., CO2_CUSTOMER_TRUE., ORDER_GROUPING_Customers.Ordering.More.than.4 MONTH_X5, MONTH_X6, MONTH_X7, MONTH_X8, MONTH_X9, MONTH_X10, MONTH_X11, MONTH_X12 < 0.001: COLD_DRINK_CHANNEL_PUBLIC.SECTOR, MONTH_X4 < 0.01: FREQUENT_ORDER_TYPE_OTHER, MONTH_X3 < 0.05: FREQUENT_ORDER_TYPE_SALES.REP

2024

The Multiple R-squared and Adjusted R-squared for the training data have the same value of 0.5962. The r-squared for the test predictions minus the actual data is 0.5744332 while the RMSE is 80.42996. The

execution time for the 2024 model which uses a train and test set from that year, took 2.038676 hours to run on the server.

Results

< 0: FIRST_DELIVERY_DATE, CUSTOMER_GROUP_NUMBER_ROLLUP, Total_Delivery_Cost, LOADED_DIFFERENCE ORDER_TYPE_EDI, ORDER_TYPE_MYCOKE.LEGACY, ORDER_TYPE_MYCOKE360, ORDER_TYPE_SALES.REP, FREQUENT_ORDER_TYPE_SALES.REP, COLD_DRINK_CHANNEL_BULK.TRADE, COLD_DRINK_CHANNEL_DINING, COLD_DRINK_CHANNEL_EVENT COLD_DRINK_CHANNEL_GOODS, COLD_DRINK_CHANNEL_PUBLIC.SECTOR, TRADE_CHANNEL_FAST.CASU TRADE_CHANNEL_GENERAL.RETAILER, SUB_TRADE_CHANNEL_MEXICAN.FAST.FOOD, SUB_TRADE_CHANNEL_OTHER.FAST.FOOD, SUB_TRADE_CHANNEL_OTHER.GENERAL.RETAIL, SUB_TRADE_CHANNEL_PIZZA.FAST.FOOD, LOCAL_MARKET_PARTNER_TRUE, CO2_CUSTOMER_TRUE. < 0.001: ORDER_GROUPING_Customers.Ordering.More.than.400, MONTH_X4, MONTH_X5, MONTH_X6, MONTH_X7, MONTH_X8, MONTH_X9, MONTH_X10, MONTH_X11, MONTH_X12 < 0.01: < 0.05: MONTH_X3

Similarities Between 2023 and 2024 Linear Regression Models

We see consistency among both models.

1. The model times for both versions of the model were extremely similar both taking just above 2 hours.
2. The r-squared for year specific test data was 0.6332 and 0.5744332, respectively
3. The RMSE values are also close, indicating comparable performance: 81.87398 and 80.42996
4. Significant variables in common: order_grouping_customers.ordering.more.than.400, first_delivery_date, customer_group_number_rollup, total_delivery_cost, and various cold_drink_channel, sub_trade_channel, and month variables. - Cold Drink Channels: bulk.trade, dining, event, goods - Sub Trade Channels: mexican.fast.food, other.fast.food, other.general.retail, pizza.fast.food - Months: Variables for months 4-12 are consistent between the 2023 and 2024 models.

Differences Between 2023 and 2024 Linear Regression Models

1. The 2023 model has a slightly better model performance in regards to r-squared and rmse.
2. The 2023 model is able to use the full 2024 data as a test set whereas the 2024 model cannot.
3. Distributions and p-values for the significant variables for both years are not exactly the same
4. 2023 unique significant variables: frequent order type other
5. 2024 unique significant variables: FREQUENT order type sales.rep, order_type_edi, and order_type_mycoke360

Results (do this as a group too)

Summarize and discuss your findings.

Identify a performance benchmark. What is a minimum threshold for model performance? Identify appropriate models to explore (given the business problem and the project objective).

- Perform cross-validation to develop performance metrics for each model, appropriate for the context.
- Optimize model performance with hyperparameter tuning, if appropriate.
- Evaluate the strengths and weaknesses of each model and select the best one.

Current Concerns with Modeling

The regular dataset has aliased coefficients and high multicollinearity. To get around this, they were removed from the recipe used to train the data. For reference, the 2023 and 2024 models had sub_trade_channel_fsr..misc and sub_trade_channel_other.dining as aliased coefficients. They also had sub_trade_channel_comprehensive.provider, trade_channel_general, and trade_channel_outdoor.activities

as having a high vif. However, the 2024 model identified sub_trade_channel_other.outdoor.activities as also having high vif. The question is whether that small difference indicates if the models are considered less generalizable.

One way to address this challenge is by using a penalized linear regression model that tunes the penalty parameters, leveraging cross-validation to select the best outcome. This approach allows the model to automatically determine whether Lasso, Ridge, or Elastic Net would be the most beneficial for the given data. A key advantage of using a penalized linear regression model is that it prevents overfitting while maintaining interpretability and performing automatic feature selection.

Feature engineered columns representing loaded difference and shipment difference were created. It is important to note the possibility that orders may not be fully processed (loaded and delivered) on the same transaction date. Interestingly, these numbers did seem to align for the most part, but it is something worth considering while interpreting model results.

Call to Action/What Reccomendations Can We Give Anyway

Next Steps/ Where could we take modeling in the future

Due to time constraints, clustering analysis was not completed on the data. A K-means clustering approach would be ideal to determine similarity between data points. If implemented, this analysis has the potential to identify more characteristics that distinguish customers with annual sales exceeding the determined volume threshold and those below. This method could use the full data Swire Coca Cola offered, both years 2023 and 2024.

Moreover, while the current regression models are decent a good next step would be to attempt black-box models. These are machine learning models that typically have high predictive power and efficiency, but lower levels of interpretability. However, Coca-Cola could easily use their historical data to predict which ARTM customers have the potential to grow beyond the annual volume threshold. To ensure the model works well, cross validation should be done and model metrics should be evaluated. It is important to note there is a tradeoff between traditional models and black-box models, Coca Cola would need to determine what works best for them and their business needs.

Improvements to the Data

To enhance the analysis of fulfillment rates, delivery efficiency, and customer behavior, additional data could be beneficial.

Key Data Improvements: • Customer-Level Data: More granular insights into purchasing patterns and preferences would enable better segmentation of customers by the defined threshold. This could improve marketing efforts and help forecast demand more accurately. • Order-Level Data: The data is currently aggregated by date and not Order ID. Adding additional columns such as Order IDs and timestamps to better track fulfillment trends and delivery performance. • Zip Code-Level Data: Instead of blinded full addresses and zip codes, removing full addresses and having accurate zip codes would allow for demographic comparisons while maintaining data privacy.