Modeling Product #3

Michael

April 19, 2024

Set up

Taking a sample of the whole dataset

```
df <- readRDS("swire_no_nas.rds")  #inject the data and we will sub-sample
```

```
regions_joinme <- read.csv("states_summary.csv")

unique(regions_joinme$REGION)
```

```
##  [1] "NORTHERN"    "DESERT_SW"   "PRAIRIE"     "CALI_NEVADA" "MOUNTAIN"
##  [6] "SOCAL"       "ARIZONA"     "NEWMEXICO"   "NOCAL"       "COLORADO"
## [11] "KANSAS"
```

```
# "NORTHERN"    "DESERT_SW"    "PRAIRIE"     "CALI_NEVADA"  "MOUNTAIN"    "SOCAL"    "ARIZONA"
"NEWMEXICO"   "NOCAL"    "COLORADO"    "KANSAS"

str(regions_joinme)
```

```
## 'data.frame':    200 obs. of  2 variables:
##  $ MARKET_KEY: int  13 70 179 197 272 352 32 33 44 50 ...
##  $ REGION    : chr  "NORTHERN" "NORTHERN" "DESERT_SW" "DESERT_SW" ...
```

```
# Perform a left join using the merge() function
df <- merge(df, regions_joinme[, c("MARKET_KEY", "REGION")], by = "MARKET_KEY", all.x = TRUE)
rm(regions_joinme)
```

Quick imputations

```
# Update CALORIC_SEGMENT values: 0 if 'DIET/LIGHT', otherwise 1
df$CALORIC_SEGMENT <- ifelse(df$CALORIC_SEGMENT == "DIET/LIGHT", 0, 1)
df$MARKET_KEY <- as.character(df$MARKET_KEY)
df <- df %>%
  mutate(
    MONTH = as.numeric(substr(DATE, 6, 7)),  # Extract the month from YYYY-MM-DD format
    SEASON = case_when(
      MONTH %in% c(12, 01, 02) ~ "WINTER",
      MONTH %in% c(03, 04, 05) ~ "SPRING",
      MONTH %in% c(06, 07, 08) ~ "SUMMER",
      MONTH %in% c(09, 10, 11) ~ "FALL",
      TRUE ~ NA_character_  # This is just in case there are any undefined values
    )
  )
```

```
str(df)
```

```
## 'data.frame':    24461424 obs. of  13 variables:
##  $ MARKET_KEY     : chr  "1" "1" "1" "1" ...
##  $ DATE           : chr  "2021-10-16" "2022-06-04" "2022-02-05" "2022-10-08" ...
##  $ CALORIC_SEGMENT: num  0 0 1 0 0 1 0 0 1 0 ...
##  $ CATEGORY       : chr  "ENERGY" "SSD" "SSD" "SSD" ...
##  $ UNIT_SALES     : num  434 28 42 1 26 161 6 5 68 90 ...
##  $ DOLLAR_SALES   : num  924.04 147.77 25.13 0.99 94.56 ...
##  $ MANUFACTURER   : chr  "PONYS" "SWIRE-CC" "COCOS" "JOLLYS" ...
##  $ BRAND          : chr  "MYTHICAL BEVERAGE ULTRA" "DIET PEPPY CF" "HANSENIZZLE'S ECO" "DIET
PAPI" ...
##  $ PACKAGE        : chr  "16SMALL MULTI CUP" "12SMALL 12ONE CUP" "12SMALL 6ONE CUP" "12SMALL
6ONE CUP" ...
##  $ ITEM           : chr  "MYTHICAL BEVERAGE ULTRA SUNRISE ENERGY DRINK UNFLAVORED ZERO SUGAR
CUP 16 LIQUID SMALL" "DIET PEPPY CAFFEINE FREE GENTLE DRINK RED  PEPPER COLA DIET CUP 12 LIQUID
SMALL X12" "HANSENIZZLE'S ECO GENTLE DRINK MANDARIN DURIAN  CUP 12 LIQUID SMALL" "DIET PAPI
GENTLE DRINK COLA DIET CUP 12 LIQUID SMALL" ...
##  $ REGION         : chr  "NORTHERN" "NORTHERN" "NORTHERN" "NORTHERN" ...
##  $ MONTH          : num  10 6 2 10 7 9 9 6 10 5 ...
##  $ SEASON         : chr  "FALL" "SUMMER" "WINTER" "FALL" ...
```

Making a 10% sample of the data to shrink it

```
# Assuming df is your dataframe
set.seed(123) # Set a random seed for reproducibility
sampled_df <- df[sample(1:nrow(df), 2446143), ]
rm(df)
```

```
df <- sampled_df
rm(sampled_df)
```

```
#skim(df)
```

```
summary(df)
```

```
##   MARKET_KEY            DATE           CALORIC_SEGMENT    CATEGORY        
##  Length:2446143     Length:2446143     Min.   :0.0000   Length:2446143    
##  Class :character   Class :character   1st Qu.:0.0000   Class :character  
##  Mode  :character   Mode  :character   Median :1.0000   Mode  :character  
##                                        Mean   :0.5025                     
##                                        3rd Qu.:1.0000                     
##                                        Max.   :1.0000                     
##    UNIT_SALES        DOLLAR_SALES      MANUFACTURER          BRAND          
##  Min.   :    0.04   Min.   :     0.0   Length:2446143     Length:2446143    
##  1st Qu.:   11.00   1st Qu.:    36.5   Class :character   Class :character  
##  Median :   40.00   Median :   135.1   Mode  :character   Mode  :character  
##  Mean   :  173.43   Mean   :   587.4                                        
##  3rd Qu.:  126.00   3rd Qu.:   427.4                                        
##  Max.   :91778.00   Max.   :409159.3                                        
##    PACKAGE              ITEM              REGION              MONTH       
##  Length:2446143     Length:2446143     Length:2446143     Min.   : 1.000  
##  Class :character   Class :character   Class :character   1st Qu.: 3.000  
##  Mode  :character   Mode  :character   Mode  :character   Median : 6.000  
##                                                           Mean   : 6.283  
##                                                           3rd Qu.: 9.000  
##                                                           Max.   :12.000  
##     SEASON         
##  Length:2446143    
##  Class :character  
##  Mode  :character  
##                    
##                    
##                    
```

Linear model on sampled data looks the same largely

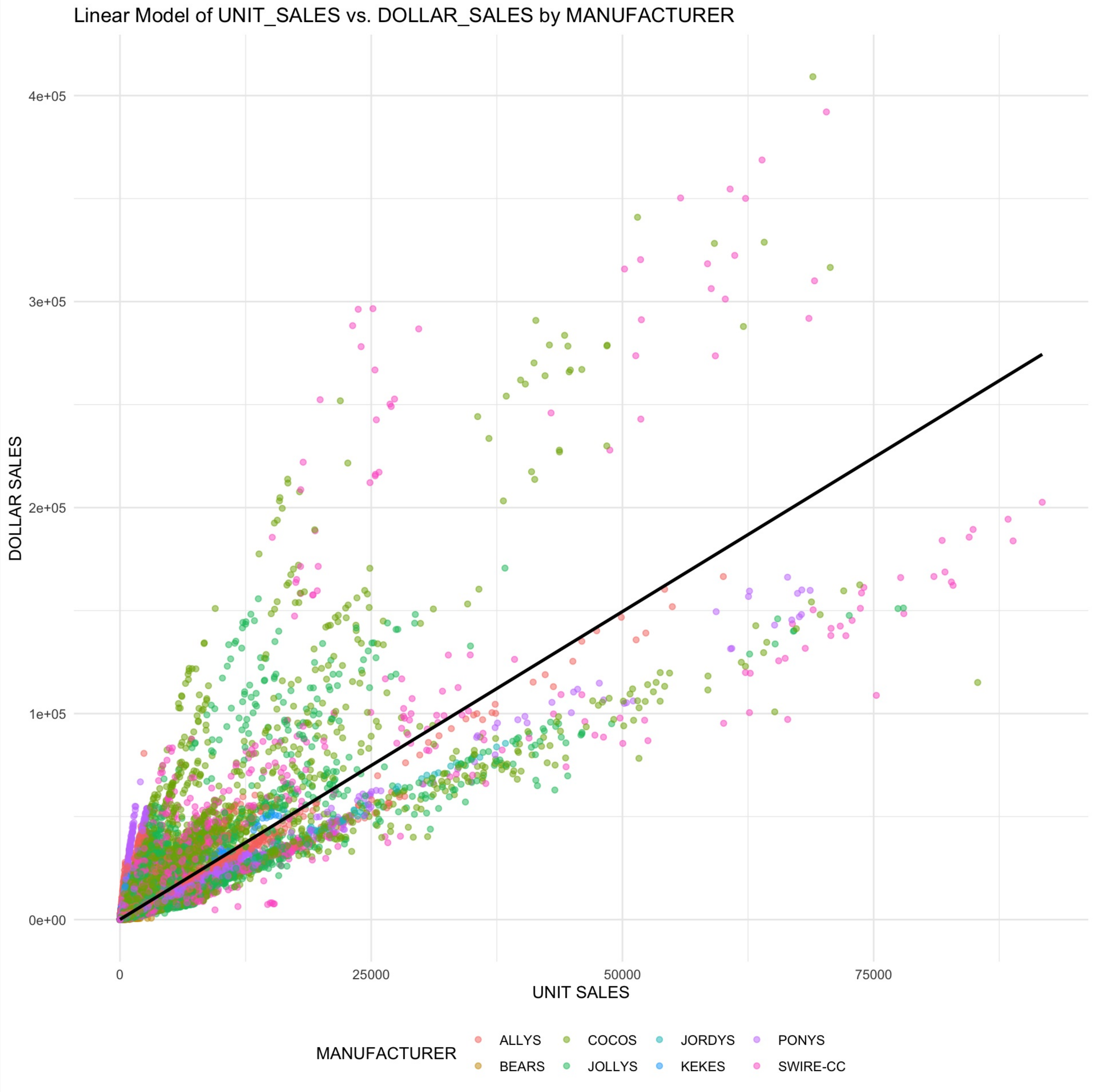
```
# Perform a linear regression with UNIT_SALES as the dependent variable
# and PRICE (or your chosen variable) as the independent variable
linear_model <- lm(DOLLAR_SALES ~ UNIT_SALES, data = df)

# Print the summary of the linear model to see the results
summary(linear_model)
```

```
## 
## Call:
## lm(formula = DOLLAR_SALES ~ UNIT_SALES, data = df)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -140089    -117     -68      -3  225329 
## 
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 69.056096   1.023439   67.47   <2e-16 ***
## UNIT_SALES   2.989060   0.001201 2489.17   <2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1567 on 2446141 degrees of freedom
## Multiple R-squared:  0.717,  Adjusted R-squared:  0.717
## F-statistic: 6.196e+06 on 1 and 2446141 DF,  p-value: < 2.2e-16
```

```
# Create a scatter plot with the regression line, colored by MANUFACTURER
ggplot(df, aes(x = UNIT_SALES, y = DOLLAR_SALES, color = MANUFACTURER)) +
  geom_point(alpha = 0.5) +  # Adjust alpha to avoid overplotting, if necessary
  geom_smooth(method = "lm", color = "black", se = FALSE) +  # Add linear regression line
without confidence band for clarity
  labs(title = "Linear Model of UNIT_SALES vs. DOLLAR_SALES by MANUFACTURER",
       x = "UNIT SALES",
       y = "DOLLAR SALES") +
  theme_minimal() +
  theme(legend.position = "bottom")  # Adjust legend position if needed
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
# create a table of total values by brand
brand_summary <- df %>%
  group_by(BRAND) %>%
  summarise(
    total_units_sold = sum(UNIT_SALES),
    total_revenue = sum(DOLLAR_SALES),
    avg_price = total_revenue / total_units_sold,
    total_days_sold = n() # Count the number of rows for each brand
  ) %>%
  arrange(desc(total_revenue)) %>%  # Order by revenue in descending order
  mutate(rank = row_number())

summary(brand_summary)
```

```
##     BRAND           total_units_sold   total_revenue        avg_price
##  Length:288         Min.   :       1   Min.   :        1   Min.   : 0.5315
```

```
##  Class :character   1st Qu.:    2310   1st Qu.:     7563   1st Qu.: 2.0861
##  Mode  :character   Median :   94691   Median :   266075   Median : 3.0291
##                     Mean   : 1473003   Mean   :  4989427   Mean   : 3.2661
##                     3rd Qu.:  651385   3rd Qu.:  2161764   3rd Qu.: 3.7252
##                     Max.   :40414038   Max.   :159387186   Max.   :42.9378
##  total_days_sold          rank
##  Min.   :     1.0   Min.   :  1.00
##  1st Qu.:   121.8   1st Qu.: 72.75
##  Median :  1988.0   Median :144.50
##  Mean   :  8493.5   Mean   :144.50
##  3rd Qu.:  8075.8   3rd Qu.:216.25
##  Max.   :124603.0   Max.   :288.00
```

```
print(brand_summary[brand_summary$BRAND == "VENOMOUS BLAST", ])
```

```
## # A tibble: 1 × 6
##   BRAND          total_units_sold total_revenue avg_price total_days_sold  rank
##   <chr>                     <dbl>         <dbl>     <dbl>           <int> <int>
## 1 VENOMOUS BLAST           360173       361370.      1.00            5188   130
```

> VENOMOUS BLAST does have a decent amount of sales ranking 130 of 288 in total revenue. They surprisingly have a low average price and a low total days sold.

Take a look at your brand..

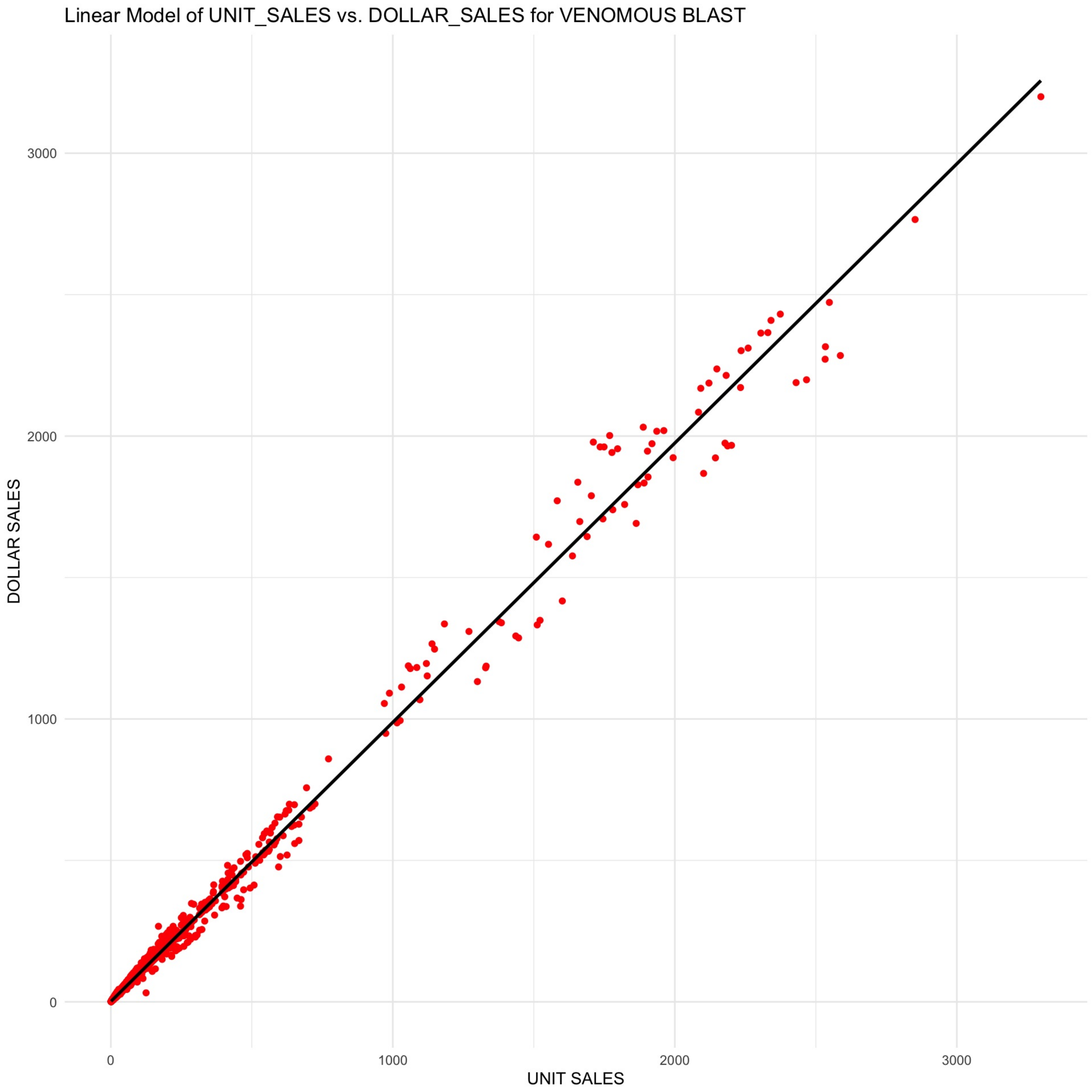
```
# Filter the dataframe for only 'Venomous Blast'
filtered_df <- df %>%
  filter(BRAND == "VENOMOUS BLAST")

summary(filtered_df)
```

```
##   MARKET_KEY            DATE           CALORIC_SEGMENT    CATEGORY
##  Length:5188        Length:5188        Min.   :0.0000   Length:5188
##  Class :character   Class :character   1st Qu.:0.0000   Class :character
##  Mode  :character   Mode  :character   Median :1.0000   Mode  :character
##                                        Mean   :0.7406
##                                        3rd Qu.:1.0000
##                                        Max.   :1.0000
##    UNIT_SALES        DOLLAR_SALES     MANUFACTURER          BRAND
##  Min.   :   1.00   Min.   :   0.50   Length:5188        Length:5188
##  1st Qu.:   6.00   1st Qu.:   5.92   Class :character   Class :character
##  Median :  16.00   Median :  16.64   Mode  :character   Mode  :character
##  Mean   :  69.42   Mean   :  69.66
##  3rd Qu.:  41.00   3rd Qu.:  42.20
##  Max.   :3298.00   Max.   :3199.67
##    PACKAGE              ITEM              REGION              MONTH
##  Length:5188        Length:5188        Length:5188        Min.   : 1.000
##  Class :character   Class :character   Class :character   1st Qu.: 3.000
##  Mode  :character   Mode  :character   Mode  :character   Median : 6.000
##                                                           Mean   : 6.174
##                                                           3rd Qu.: 9.000
##                                                           Max.   :12.000
##     SEASON
##  Length:5188
##  Class :character
```

```
##  Mode  :character
##
##
##
```

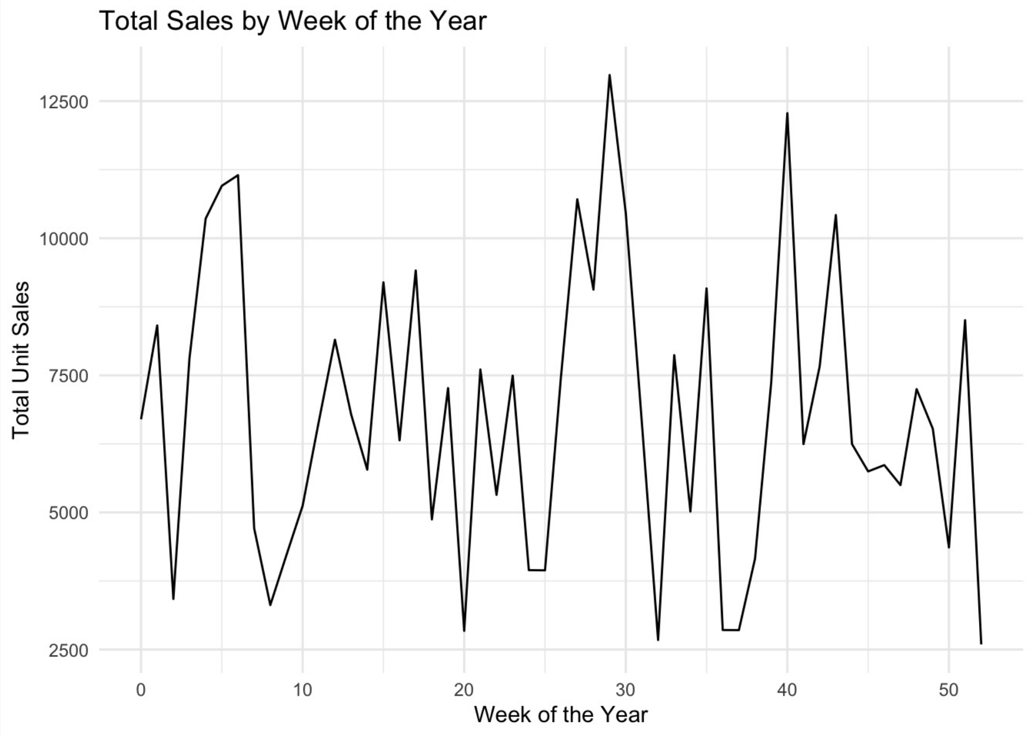
```
# Create the plot
ggplot(filtered_df, aes(x = UNIT_SALES, y = DOLLAR_SALES)) +
  geom_point(color = "red", alpha = 1) +  # Bright red points with full opacity
  geom_smooth(method = "lm", color = "black", se = FALSE) +  # Add linear regression line
without confidence band
  labs(title = "Linear Model of UNIT_SALES vs. DOLLAR_SALES for VENOMOUS BLAST",
       x = "UNIT SALES",
       y = "DOLLAR SALES") +
  theme_minimal() +
  theme(legend.position = "none")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Sales by Week of the year

```
filtered_df %>%
  mutate(DATE = as.Date(DATE)) %>%
  mutate(WEEK = as.integer(format(DATE, "%U"))) %>%
  group_by(WEEK) %>%
  summarise(total_sales = sum(UNIT_SALES)) %>%
  ggplot(aes(x = WEEK, y = total_sales)) +
  geom_line(color = "black") +  # Blue line connecting points
  labs(title = "Total Sales by Week of the Year",
       x = "Week of the Year",
       y = "Total Unit Sales") +
  theme_minimal()
```



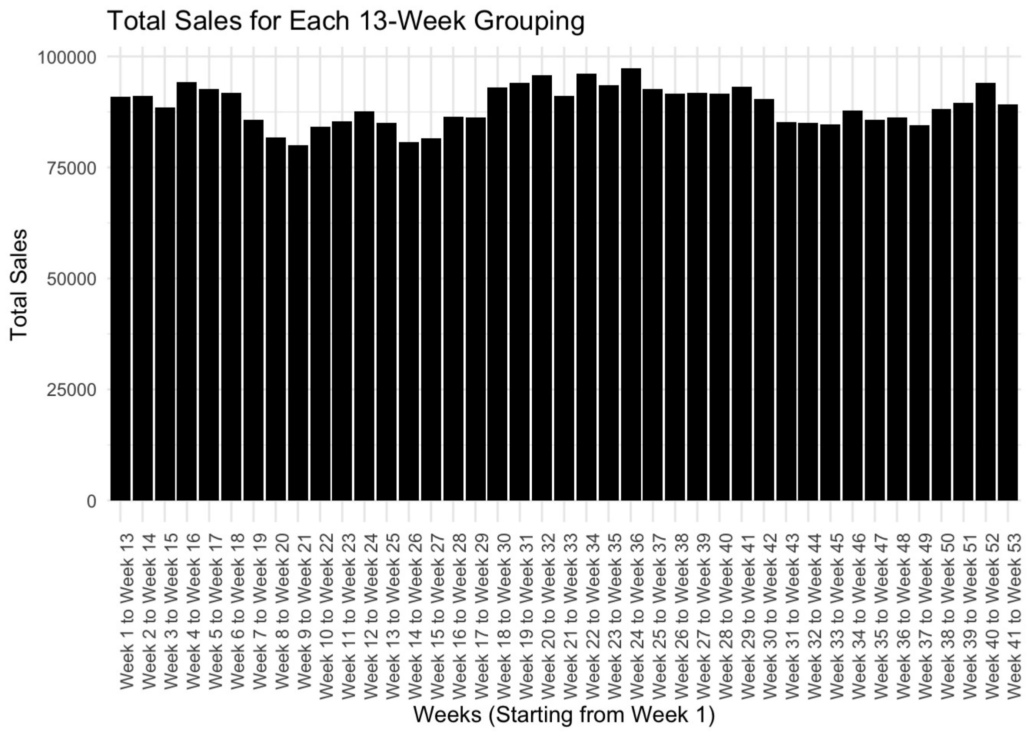
> Ths graph demonstrates that sales of venomus blast has a large amount of variance through out the year.

```
#find the best 13 weeks
library(zoo)
```

```
##
## Attaching package: 'zoo'
```

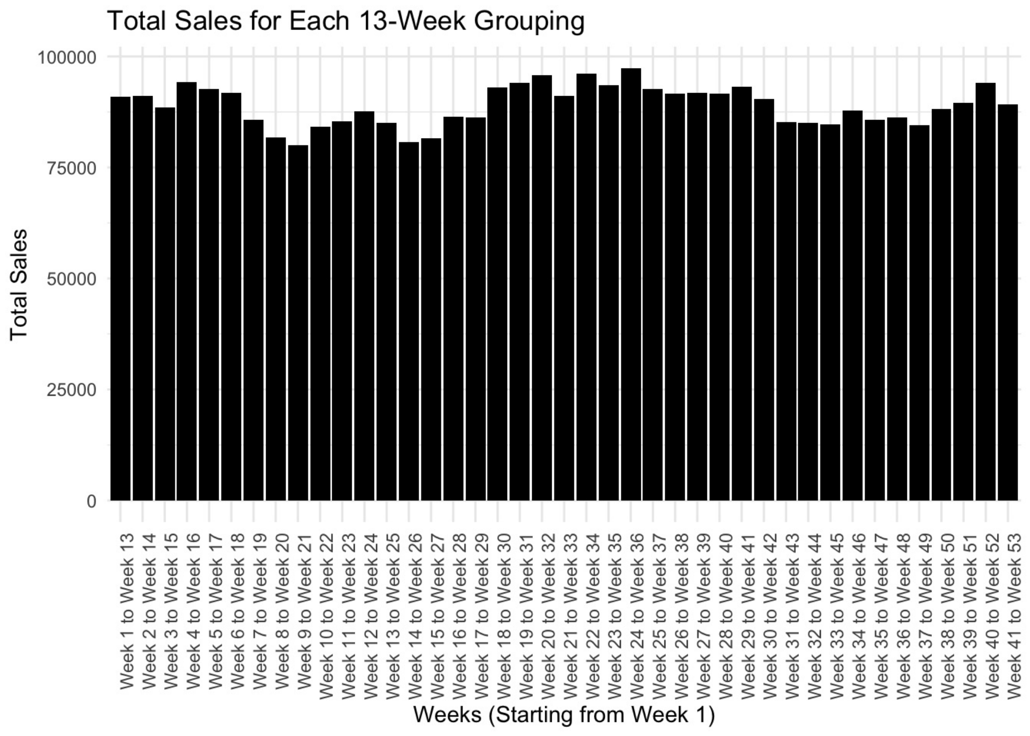
```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
# Calculate total sales for each group of 13 consecutive weeks
sales_by_group <- filtered_df %>%
  mutate(DATE = as.Date(DATE)) %>%
  mutate(WEEK = as.integer(format(DATE, "%U"))) %>%
  group_by(WEEK) %>%
  summarise(total_sales = sum(UNIT_SALES)) %>%
  mutate(sales_in_group = rollsum(total_sales, 13, align = "left", fill = NA)) %>%
  mutate(week_label = paste0("Week ", WEEK + 1, " to Week ", WEEK + 13)) %>%
  arrange(WEEK) %>%  # Order by WEEK
  filter(!is.na(sales_in_group))  # Remove rows with sales_in_group = NA

# Plot the bar chart
sales_by_group$week_label <- factor(sales_by_group$week_label, levels =
sales_by_group$week_label[order(sales_by_group$WEEK)])
ggplot(sales_by_group, aes(x = factor(week_label), y = sales_in_group)) +
  geom_bar(stat = "identity", fill = "black") +
  labs(title = "Total Sales for Each 13-Week Grouping",
       x = "Weeks (Starting from Week 1)",
       y = "Total Sales") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



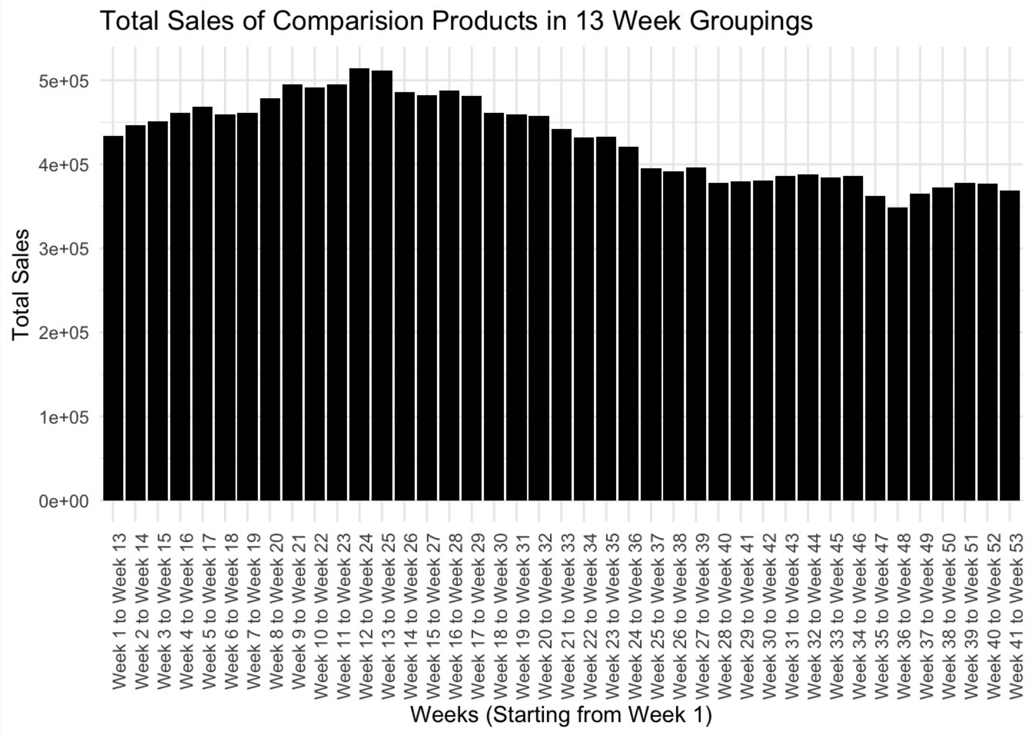
> From this graph we see that weeks 24 to 36 historically have the highest unit sales of VENOMOUS BLAST

```
#find the best 13 weeks for Kiwano sales
# Calculate total sales for each group of 13 consecutive weeks
sales_by_kiwano <- df %>%
  filter(str_detect(BRAND, "VENOMOUS BLAST"),
         CATEGORY == "ENERGY") %>%
  mutate(DATE = as.Date(DATE)) %>%
  mutate(WEEK = as.integer(format(DATE, "%U"))) %>%
  group_by(WEEK) %>%
  summarise(total_sales = sum(UNIT_SALES)) %>%
  mutate(sales_in_group = rollsum(total_sales, 13, align = "left", fill = NA)) %>%
  mutate(week_label = paste0("Week ", WEEK + 1, " to Week ", WEEK + 13)) %>%
  arrange(WEEK) %>%  # Order by WEEK
  filter(!is.na(sales_in_group))  # Remove rows with sales_in_group = NA

# Plot the bar chart
sales_by_kiwano$week_label <- factor(sales_by_kiwano$week_label, levels =
sales_by_kiwano$week_label[order(sales_by_kiwano$WEEK)])
ggplot(sales_by_kiwano, aes(x = factor(week_label), y = sales_in_group)) +
  geom_bar(stat = "identity", fill = "black") +
  labs(title = "Total Sales for Each 13-Week Grouping",
       x = "Weeks (Starting from Week 1)",
       y = "Total Sales") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



>This graph shows the best weeks sales of any kiwano drink is week 19 to 31.

```
#find the best 13 weeks for Kiwano sales
# Calculate total sales for each group of 13 consecutive weeks
sales_by_energy <- df %>%
  filter(CATEGORY == "ENERGY",
         str_detect(ITEM, "KIWANO"),
         str_detect(PACKAGE, "16")) %>%
  mutate(DATE = as.Date(DATE)) %>%
  mutate(WEEK = as.integer(format(DATE, "%U"))) %>%
  group_by(WEEK) %>%
  summarise(total_sales = sum(UNIT_SALES)) %>%
  mutate(sales_in_group = rollsum(total_sales, 13, align = "left", fill = NA)) %>%
  mutate(week_label = paste0("Week ", WEEK + 1, " to Week ", WEEK + 13)) %>%
  arrange(WEEK) %>%  # Order by WEEK
  filter(!is.na(sales_in_group))  # Remove rows with sales_in_group = NA

# Plot the bar chart
sales_by_energy$week_label <- factor(sales_by_energy$week_label, levels =
sales_by_energy$week_label[order(sales_by_energy$WEEK)])
ggplot(sales_by_energy, aes(x = factor(week_label), y = sales_in_group)) +
  geom_bar(stat = "identity", fill = "black") +
  labs(title = "Total Sales of Comparision Products in 13 Week Groupings",
       x = "Weeks (Starting from Week 1)",
       y = "Total Sales") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



>In this Graph we are shown the best weeks for sales of Energy drinks with Kiwano flavors and packageing 16 is weeks 11 to 23 which is March 11th to June 9th

Made a new smaller "innovation" data fram

```
innovation <- df %>%
  filter(CATEGORY == "ENERGY",
         CALORIC_SEGMENT == 0,
         str_detect(ITEM, "KIWANO"),
         str_detect(PACKAGE, "16"))

print(unique(innovation$ITEM))
```

```
##  [1] "MYTHICAL BEVERAGE ULTRA KIWANO  ENERGY DRINK UNFLAVORED ZERO SUGAR CUP 16 LIQUID SMALL X4"
##  [2] "SUPER-DUPER PURE ZERO ENERGY DRINK KIWANO  KEKE  SUGAR FREE CUP 16 LIQUID SMALL"
##  [3] "RAINING JUMPIN-FISH GAME FUEL ZERO ENERGY DRINK CHARGED KIWANO  SHOCK ZERO SUGAR CUP 16 LIQUID SMALL"
##  [4] "MYTHICAL BEVERAGE ULTRA KIWANO  ENERGY DRINK UNFLAVORED ZERO SUGAR CUP 16 LIQUID SMALL"
##  [5] "SUPER-DUPER PURE ZERO ENERGY DRINK KIWANO  ZERO SUGAR CUP 16 LIQUID SMALL"
##  [6] "MYTHICAL BEVERAGE ULTRA KIWANO  ENERGY DRINK UNFLAVORED ZERO SUGAR CUP 16 LIQUID SMALL X24"
##  [7] "VENOMOUS BLAST ENERGY DRINK KIWANO  DURIAN  CUP 16 LIQUID SMALL"
##  [8] "POW-POW GENTLE DRINK WYLDIN KIWANO  CUP 16 LIQUID SMALL X12"
##  [9] "MYTHICAL BEVERAGE REHAB ENERGY DRINK KIWANO  CUP 15.5 LIQUID SMALL X24"
## [10] "SUPER-DUPER PURE ZERO ENERGY DRINK KIWANO  ZERO SUGAR CUP 16 LIQUID SMALL X24"
```

```
#there are 10 items with energy, diet, kiwano that come in packs of 16, but none of them are from VENOMOUS BLAST.



library(dplyr)
library(lubridate)
```

```
innovation <- innovation %>%
  mutate(
    MONTH = month(ymd(DATE)),  # Extract month using lubridate's ymd function
    MONTH = as.factor(MONTH)   # Convert the extracted month into a factor
  )

str(innovation)
```

```
## 'data.frame':    8082 obs. of  13 variables:
##  $ MARKET_KEY     : chr  "504" "953" "133" "817" ...
##  $ DATE           : chr  "2022-02-26" "2022-08-20" "2020-12-19" "2022-02-05" ...
##  $ CALORIC_SEGMENT: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ CATEGORY       : chr  "ENERGY" "ENERGY" "ENERGY" "ENERGY" ...
##  $ UNIT_SALES     : num  11 13 20 194 8 176 87 300 4 102 ...
##  $ DOLLAR_SALES   : num  78.9 21.8 40.5 287.1 63.4 ...
##  $ MANUFACTURER   : chr  "PONYS" "JOLLYS" "JOLLYS" "JOLLYS" ...
##  $ BRAND          : chr  "MYTHICAL BEVERAGE ULTRA" "SUPER-DUPER PURE ZERO" "HILL MOISTURE
JUMPIN-FISH" "SUPER-DUPER PURE ZERO" ...
##  $ PACKAGE        : chr  "16SMALL 4ONE CUP" "16SMALL MULTI CUP" "16SMALL MULTI CUP" "16SMALL
MULTI CUP" ...
##  $ ITEM           : chr  "MYTHICAL BEVERAGE ULTRA KIWANO  ENERGY DRINK UNFLAVORED ZERO SUGAR
CUP 16 LIQUID SMALL X4" "SUPER-DUPER PURE ZERO ENERGY DRINK KIWANO  KEKE  SUGAR FREE CUP 16
LIQUID SMALL" "RAINING JUMPIN-FISH GAME FUEL ZERO ENERGY DRINK CHARGED KIWANO  SHOCK ZERO SUGAR
CUP 16 LIQUID SMALL" "SUPER-DUPER PURE ZERO ENERGY DRINK KIWANO  KEKE  SUGAR FREE CUP 16 LIQUID
SMALL" ...
##  $ REGION         : chr  "NORTHERN" "ARIZONA" "MOUNTAIN" "COLORADO" ...
##  $ MONTH          : Factor w/ 12 levels "1","2","3","4",..: 2 8 12 2 5 5 10 8 5 8 ...
##  $ SEASON         : chr  "WINTER" "SUMMER" "WINTER" "WINTER" ...
```

```
# Assuming 'innovation' is your data frame
model <- lm(DOLLAR_SALES ~ UNIT_SALES + CALORIC_SEGMENT + PACKAGE + SEASON + REGION, data =
innovation)
summary(model)
```

```
##
## Call:
## lm(formula = DOLLAR_SALES ~ UNIT_SALES + CALORIC_SEGMENT + PACKAGE +
##     SEASON + REGION, data = innovation)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -953.6  -35.4   -1.1   27.6 5847.8
##
## Coefficients: (1 not defined because of singularities)
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)              -3.571915  18.407162  -0.194 0.846141
## UNIT_SALES                2.179578   0.004342 501.957  < 2e-16 ***
## CALORIC_SEGMENT                 NA         NA      NA       NA
## PACKAGE16SMALL 24ONE CUP 178.333322  19.021309   9.375  < 2e-16 ***
## PACKAGE16SMALL 4ONE CUP   62.481782  18.270952   3.420 0.000630 ***
## PACKAGE16SMALL MULTI CUP  -9.985916  17.796875  -0.561 0.574742
## SEASONSPRING              7.749111   5.278459   1.468 0.142126
## SEASONSUMMER              0.158127   5.606383   0.028 0.977500
## SEASONWINTER             -5.957836   5.296196  -1.125 0.260653
## REGIONCALI_NEVADA        -6.656448  10.258577  -0.649 0.516443
```

```
## REGIONCOLORADO          19.756980   6.669432   2.962 0.003062 **
## REGIONDESERT_SW          0.165096   7.867662   0.021 0.983259
## REGIONKANSAS           170.758804  14.371366  11.882  < 2e-16 ***
## REGIONMOUNTAIN          -0.897751   7.130829  -0.126 0.899816
## REGIONNEWMEXICO         15.665066   9.744594   1.608 0.107970
## REGIONNOCAL            -18.993746   9.856390  -1.927 0.054009 .
## REGIONNORTHERN          -6.707730   5.438244  -1.233 0.217449
## REGIONPRAIRIE           40.817170  11.563916   3.530 0.000418 ***
## REGIONSOCAL            -14.067039   7.502472  -1.875 0.060831 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 164.2 on 8064 degrees of freedom
## Multiple R-squared:  0.975,  Adjusted R-squared:  0.975
## F-statistic: 1.852e+04 on 17 and 8064 DF,  p-value: < 2.2e-16
```

> This model is showing an R2 of .975. With like the other models Region kansas being significant and then the size of the cup.There was only 8082 observations to go off of in this category, but I am wondering if we combine the model with this flavor, size and then sales for the first 13 weeks we can then apply that with a sales factor built based on VENOMOUS BLASTS best selling weeks to get demand.

More exploration

```
library(dplyr)

small_group <- df %>%
  filter(UNIT_SALES < 3300, DOLLAR_SALES < 3200)

skim(small_group)
```

Data summary

Name	small_group
Number of rows	2372840
Number of columns	13
_______________________	
Column type frequency:	
character	9
numeric	4
________________________	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
MARKET_KEY	0	1	1	4	0	200	0
DATE	0	1	10	10	0	152	0
CATEGORY	0	1	3	18	0	5	0

MANUFACTURER	0	1	5	8	0	8	0
BRAND	0	1	4	56	0	288	0
PACKAGE	0	1	11	26	0	95	0
ITEM	0	1	26	142	0	2999	0
REGION	0	1	5	11	0	11	0
SEASON	0	1	4	6	0	4	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
CALORIC_SEGMENT	0	1	0.50	0.50	0.00	0.00	0.00	1.00	1.00	
UNIT_SALES	0	1	104.83	183.49	0.04	10.00	38.00	113.00	3298.00	
DOLLAR_SALES	0	1	332.69	516.65	0.01	34.92	126.27	380.55	3199.98	
MONTH	0	1	6.28	3.44	1.00	3.00	6.00	9.00	12.00	

```
# Create a scatter plot with the regression line, colored by MANUFACTURER
ggplot(small_group, aes(x = UNIT_SALES, y = DOLLAR_SALES, color = MANUFACTURER)) +
  geom_point(alpha = 0.5) +  # Adjust alpha to avoid overplotting, if necessary
  geom_smooth(method = "lm", color = "black", se = FALSE) +  # Add linear regression line
without confidence band for clarity
  labs(title = "Linear Model of UNIT_SALES vs. DOLLAR_SALES by MANUFACTURER",
       x = "UNTI SALES",
       y = "DOLLAR SALES") +
  theme_minimal() +
  theme(legend.position = "bottom")  # Adjust legend position if needed
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

> Basically this is where Venomous Blast lives in this realm. Notice still that certain items just sell way better than others in terms of dollars.

#Make the small Kiwanao df > Create a Kiwano Small Data set

```
kiwano_small <- df[grep("kiwano", df$ITEM, ignore.case = TRUE), ]
```

```
skim(kiwano_small)
```

Data summary

Name	kiwano_small
Number of rows	71256
Number of columns	13

Column type frequency:	
character	9
numeric	4
________________________	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
MARKET_KEY	0	1	1	4	0	200	0
DATE	0	1	10	10	0	152	0
CATEGORY	0	1	3	18	0	4	0
MANUFACTURER	0	1	5	8	0	7	0
BRAND	0	1	5	41	0	27	0
PACKAGE	0	1	12	23	0	28	0
ITEM	0	1	46	105	0	68	0
REGION	0	1	5	11	0	11	0
SEASON	0	1	4	6	0	4	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
CALORIC_SEGMENT	0	1	0.34	0.48	0.00	0.00	0.0	1.00	1.00	
UNIT_SALES	0	1	101.93	384.04	0.50	8.00	26.0	76.00	16851.00	
DOLLAR_SALES	0	1	280.74	1016.57	0.01	28.25	86.8	221.68	45991.65	
MONTH	0	1	6.32	3.44	1.00	3.00	6.0	9.00	12.00	

```
# Assuming 'innovation' is your data frame
model <- lm(DOLLAR_SALES ~ UNIT_SALES + CALORIC_SEGMENT + PACKAGE + CATEGORY + SEASON + REGION,
data = kiwano_small)
summary(model)
```

```
##
## Call:
## lm(formula = DOLLAR_SALES ~ UNIT_SALES + CALORIC_SEGMENT + PACKAGE +
##     CATEGORY + SEASON + REGION, data = kiwano_small)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4689.1   -40.8    -5.9    38.3  6971.0
##
## Coefficients:
##                                    Estimate Std. Error  t value Pr(>|t|)
```

```
## (Intercept)                       1.393e+02  1.103e+01   12.625  < 2e-16 ***
## UNIT_SALES                        2.568e+00  1.992e-03 1288.714  < 2e-16 ***
## CALORIC_SEGMENT                   1.117e+02  2.414e+00   46.279  < 2e-16 ***
## PACKAGE.5L 6ONE JUG              -8.551e+01  6.049e+00  -14.137  < 2e-16 ***
## PACKAGE.5L MULTI JUG             -1.233e+02  1.111e+01  -11.092  < 2e-16 ***
## PACKAGE1.25L MULTI JUG           -3.811e+02  2.084e+01  -18.287  < 2e-16 ***
## PACKAGE12SMALL 12ONE CUP         -8.777e+01  9.668e+00   -9.079  < 2e-16 ***
## PACKAGE12SMALL 24ONE CUP         -2.348e+02  1.484e+01  -15.825  < 2e-16 ***
## PACKAGE12SMALL 8ONE CUP          -8.347e+01  1.062e+01   -7.857 4.00e-15 ***
## PACKAGE12SMALL MLT BUMPY CUP     -1.712e+02  7.514e+00  -22.790  < 2e-16 ***
## PACKAGE12SMALL MLT MEDIUM CUP    -1.833e+02  4.247e+01   -4.317 1.58e-05 ***
## PACKAGE12SMALL MULTI CUP         -1.020e+02  1.107e+01   -9.209  < 2e-16 ***
## PACKAGE16SMALL 12ONE CUP         -1.417e+02  2.247e+01   -6.308 2.85e-10 ***
## PACKAGE16SMALL 24ONE CUP          3.268e+01  1.338e+01    2.442 0.014617 *
## PACKAGE16SMALL 4ONE CUP          -8.254e+01  1.206e+01   -6.842 7.86e-12 ***
## PACKAGE16SMALL MULTI CUP         -2.411e+02  1.079e+01  -22.335  < 2e-16 ***
## PACKAGE18SMALL 6ONE              -3.718e+01  5.306e+00   -7.007 2.46e-12 ***
## PACKAGE18SMALL MULTI JUG         -1.642e+02  4.463e+00  -36.784  < 2e-16 ***
## PACKAGE1L MULTI JUG              -1.883e+02  2.070e+01   -9.096  < 2e-16 ***
## PACKAGE20SMALL 12ONE JUG         -2.342e+02  6.947e+01   -3.371 0.000749 ***
## PACKAGE20SMALL MULTI JUG         -2.175e+02  4.431e+00  -49.091  < 2e-16 ***
## PACKAGE24 - 25SMALL MULTI JUG    -1.819e+02  5.587e+00  -32.557  < 2e-16 ***
## PACKAGE24SMALL MLT SHADYES JUG   -1.985e+02  4.364e+01   -4.548 5.42e-06 ***
## PACKAGE2L MULTI JUG              -2.168e+02  7.749e+00  -27.981  < 2e-16 ***
## PACKAGE7.5SMALL 10ONE            -1.313e+02  1.064e+02   -1.234 0.217097
## PACKAGE8SMALL 12ONE CUP          -3.811e+00  1.199e+01   -0.318 0.750544
## PACKAGE8SMALL 24ONE CUP          -2.551e+02  2.122e+01  -12.022  < 2e-16 ***
## PACKAGE8SMALL 4ONE CUP           -1.213e+02  1.151e+01  -10.540  < 2e-16 ***
## PACKAGE8SMALL MULTI CUP          -3.477e+02  1.148e+01  -30.292  < 2e-16 ***
## PACKAGEALL OTHER ONES            -2.758e+01  1.107e+01   -2.490 0.012769 *
## CATEGORYING ENHANCED WATER       -1.694e+01  1.022e+01   -1.658 0.097326 .
## CATEGORYSPARKLING WATER          -3.909e+00  3.442e+00   -1.136 0.256005
## CATEGORYSSD                      -6.247e+01  9.538e+00   -6.550 5.80e-11 ***
## SEASONSPRING                     -1.027e+01  1.944e+00   -5.282 1.28e-07 ***
## SEASONSUMMER                     -1.143e+01  2.017e+00   -5.667 1.46e-08 ***
## SEASONWINTER                     -7.458e+00  1.976e+00   -3.774 0.000161 ***
## REGIONCALI_NEVADA                 4.919e+00  4.000e+00    1.230 0.218800
## REGIONCOLORADO                    1.046e+01  2.480e+00    4.220 2.45e-05 ***
## REGIONDESERT_SW                   2.392e+00  2.934e+00    0.815 0.414922
## REGIONKANSAS                      1.757e+02  5.232e+00   33.571  < 2e-16 ***
## REGIONMOUNTAIN                    1.032e+01  2.691e+00    3.835 0.000126 ***
## REGIONNEWMEXICO                   1.735e+01  3.511e+00    4.943 7.71e-07 ***
## REGIONNOCAL                       2.368e+00  3.761e+00    0.630 0.528893
## REGIONNORTHERN                    8.657e+00  2.026e+00    4.273 1.93e-05 ***
## REGIONPRAIRIE                     2.289e+01  4.318e+00    5.301 1.15e-07 ***
## REGIONSOCAL                       1.705e+00  2.842e+00    0.600 0.548662
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 183.4 on 71210 degrees of freedom
## Multiple R-squared:  0.9675, Adjusted R-squared:  0.9675
## F-statistic: 4.707e+04 on 45 and 71210 DF,  p-value: < 2.2e-16
```

> r2 even higher than before of .9675. This one has many different factors that are showing significant and had about 10x the observations as the first grouping.

Cleaning

Create redefined KIWANO set for modeling

```
kiwano_small <- df %>%
  filter(CATEGORY == "ENERGY",
         str_detect(ITEM, "KIWANO"),
         CALORIC_SEGMENT == 0,
          str_detect(PACKAGE, "16"))
```

> Rework kiwanao for more features for XGboost Model

```
kiwano_small <- kiwano_small %>%
  mutate(
    PACKAGE2 = str_extract(ITEM, "(CUP|JUG).*"),  # Extracts the part from CUP or JUG to the
end.
    ITEM = str_replace(ITEM, "(CUP|JUG).*", "")  # Replaces the CUP/JUG and everything after it
with empty string in ITEM.
  )
```

```
kiwano_small <- kiwano_small %>%
  mutate(
    TEMP = str_extract(ITEM, "\\d+\\.?\\d*.*"), # Extracts the part from the first number to
the end.
    PACKAGE2 = if_else(is.na(PACKAGE2), TEMP, paste(PACKAGE2, TEMP)), # Combines existing
PACKAGE2 with new extraction if needed.
    ITEM = str_replace(ITEM, "\\d+\\.?\\d*.*", ""), # Removes the numeric part and everything
after it from ITEM.
    TEMP = NULL  # Removes the temporary column.
  )
```

```
na_rows <- kiwano_small %>%
  filter(is.na(PACKAGE2))
#na_rows
#the above steps excised all packaging out of ITEM column
```

```
kiwano_small <- kiwano_small %>%
  mutate(
    GENTLE_DRINK = if_else(str_detect(ITEM, "GENTLE DRINK"), 1, 0), # Assigns 1 if "GENTLE
DRINK" exists, otherwise 0.
    ITEM = str_replace(ITEM, "GENTLE DRINK", "") # Removes "GENTLE DRINK" from ITEM.
  )
```

```
kiwano_small <- kiwano_small %>%
  mutate(
    ENERGY_DRINK = if_else(str_detect(ITEM, "ENERGY DRINK"), 1, 0), # Assigns 1 if "ENERGY
DRINK" exists, otherwise 0.
    ITEM = str_replace(ITEM, "ENERGY DRINK", "") # Removes "ENERGY DRINK" from ITEM.
  )
```

```
library(stringr)
# Define the pattern as a regular expression
pattern <- "ZERO CALORIES|ZERO CALORIE|ZERO SUGAR|SUGAR FREE|NO CALORIES"
```

```
kiwano_small <- kiwano_small %>%
  mutate(
    CALORIC_SEGMENT_TEXT = str_extract(ITEM, pattern), # Extracts matching text based on the
pattern.
    ITEM = str_replace_all(ITEM, pattern, "") # Removes extracted text from ITEM.
  )
```

```
kiwano_small <- kiwano_small %>%
  mutate(
    CALORIC_SEGMENT_TEXT = if_else(str_detect(ITEM, "\\bDIET\\b"),
                                   if_else(is.na(CALORIC_SEGMENT_TEXT), "DIET",
paste(CALORIC_SEGMENT_TEXT, "DIET", sep=", ")),
                                   CALORIC_SEGMENT_TEXT)
  )
```

```
# Function to remove the second instance of any repeating word
remove_second_instance <- function(item) {
  words <- unlist(str_split(item, "\\s+")) # Split item into words
  unique_words <- unique(words) # Get unique words to check for repeats
  for (word in unique_words) {
    word_indices <- which(words == word) # Find all indices of the current word
    if (length(word_indices) > 1) { # If there is more than one occurrence
      words[word_indices[2]] <- "" # Remove the second occurrence
    }
  }
  return(paste(words, collapse = " ")) # Reconstruct sentence without the second instance
}

# Apply the function to the 'ITEM' column
kiwano_small <- kiwano_small %>%
  mutate(ITEM = sapply(ITEM, remove_second_instance))


# Remove specific columns
kiwano_small <- select(kiwano_small, -PACKAGE2, -GENTLE_DRINK, -ENERGY_DRINK, -
CALORIC_SEGMENT_TEXT)
```

```
head(kiwano_small)
```

```
##   MARKET_KEY       DATE CALORIC_SEGMENT CATEGORY UNIT_SALES DOLLAR_SALES
## 1        504 2022-02-26               0   ENERGY         11        78.89
## 2        953 2022-08-20               0   ENERGY         13        21.83
## 3        133 2020-12-19               0   ENERGY         20        40.55
## 4        817 2022-02-05               0   ENERGY        194       287.06
## 5        733 2022-05-07               0   ENERGY          8        63.42
## 6        754 2023-05-13               0   ENERGY        176       258.79
##   MANUFACTURER                     BRAND           PACKAGE
## 1        PONYS   MYTHICAL BEVERAGE ULTRA  16SMALL 4ONE CUP
## 2       JOLLYS     SUPER-DUPER PURE ZERO 16SMALL MULTI CUP
## 3       JOLLYS HILL MOISTURE JUMPIN-FISH 16SMALL MULTI CUP
## 4       JOLLYS     SUPER-DUPER PURE ZERO 16SMALL MULTI CUP
## 5        PONYS   MYTHICAL BEVERAGE ULTRA  16SMALL 4ONE CUP
## 6       JOLLYS     SUPER-DUPER PURE ZERO 16SMALL MULTI CUP
##                                                         ITEM    REGION MONTH
```

```
## 1                MYTHICAL BEVERAGE ULTRA KIWANO UNFLAVORED   NORTHERN     2
## 2                        SUPER-DUPER PURE ZERO KIWANO KEKE    ARIZONA     8
## 3 RAINING JUMPIN-FISH GAME FUEL ZERO CHARGED KIWANO SHOCK   MOUNTAIN    12
## 4                        SUPER-DUPER PURE ZERO KIWANO KEKE   COLORADO     2
## 5                MYTHICAL BEVERAGE ULTRA KIWANO UNFLAVORED    ARIZONA     5
## 6                        SUPER-DUPER PURE ZERO KIWANO KEKE  DESERT_SW     5
##   SEASON
## 1 WINTER
## 2 SUMMER
## 3 WINTER
## 4 WINTER
## 5 SPRING
## 6 SPRING
```

```
write.csv(kiwano_small, "kiwano_small.csv", row.names = FALSE)
```

FINAL THOUGHTS

> Though Kiwano and energy drinks have very few rows. I do think there is potential here to find a good fitting model that can predict launch sales. I am thinking that if we can get a model that will predict the sales of uints of energy drinks, with size 16, and kiwano flavor we can then use that combined with the current sales rate of VENOMUS BLAST launches to get an accurate forecast. As far as selection of what weeks would be best to sell I don't see any other way than by using historical best 13 weeks sales of either Venmous Blast, energy drinks, or kiwano flavored drinks.

XGBoost

```
# Set up
if (!require("pacman")) install.packages("pacman")
pacman::p_load(tidyverse, skimr, knitr, caret, readr,
               ggplot2, dplyr, tidymodels, pROC, xgboost, doParallel, vip, DALEXtra)
```

Create One Hot Encoded DF for Model

```
# Read the CSV file
kiwano_small <- read.csv("kiwano_small.csv")

# Convert 'Date' column to Date format
kiwano_small$DATE <- as.Date(kiwano_small$DATE)

# List to store unique values for each variable
unique_values_list <- list()

# Columns to get unique values for
columns_to_get_unique_values <- c("BRAND", "PACKAGE", "ITEM", "REGION", "SEASON")

# Get unique values for each variable and store in the list
for (col in columns_to_get_unique_values) {
  unique_values_list[[col]] <- unique(kiwano_small[[col]])
}

# Loop over unique regions and create new columns
for (region in unique_values_list$REGION) {
```

```
  kiwano_small[[region]] <- as.integer(grepl(region, kiwano_small$REGION))
}

# Loop over unique brands and create new columns
for (brand in unique_values_list$BRAND) {
  kiwano_small[[brand]] <- as.integer(grepl(brand, kiwano_small$BRAND))
}

# Loop over unique brands and create new columns
for (item in unique_values_list$ITEM) {
  kiwano_small[[item]] <- as.integer(grepl(item, kiwano_small$ITEM))
}

# Loop over unique regions and create new columns
for (package in unique_values_list$PACKAGE) {
  kiwano_small[[package]] <- as.integer(grepl(package, kiwano_small$PACKAGE))
}

# Loop over unique regions and create new columns
for (season in unique_values_list$SEASON) {
  kiwano_small[[season]] <- as.integer(grepl(season, kiwano_small$SEASON))
}

# Add new columns for week since launch and week of the year
kiwano_small <- kiwano_small %>%
  mutate(
    Week_Of_Year = week(DATE)
  ) %>%
  group_by(ITEM) %>%
  mutate(
    Week_Since_Launch = as.integer((DATE - min(DATE)) / 7) + 1
  ) %>%
  ungroup()  # Ungroup the data to ensure the next operation applies to the entire data frame

# Remove unnecessary columns
one_hot_kiwano <- kiwano_small %>%
  select(-MARKET_KEY, -CALORIC_SEGMENT, -CATEGORY, -MANUFACTURER, -BRAND, -REGION, -PACKAGE, -
SEASON, -ITEM)

head(one_hot_kiwano)
```

```
## # A tibble: 6 × 38
##   DATE       UNIT_SALES DOLLAR_SALES MONTH NORTHERN ARIZONA MOUNTAIN COLORADO
##   <date>          <dbl>        <dbl> <int>    <int>   <int>    <int>    <int>
## 1 2022-02-26         11         78.9     2        1       0        0        0
## 2 2022-08-20         13         21.8     8        0       1        0        0
## 3 2020-12-19         20         40.6    12        0       0        1        0
## 4 2022-02-05        194        287.      2        0       0        0        1
## 5 2022-05-07          8         63.4     5        0       1        0        0
## 6 2023-05-13        176        259.      5        0       0        0        0
## # ℹ 30 more variables: DESERT_SW <int>, NOCAL <int>, SOCAL <int>, KANSAS <int>,
## #   NEWMEXICO <int>, CALI_NEVADA <int>, PRAIRIE <int>,
## #   `MYTHICAL BEVERAGE ULTRA` <int>, `SUPER-DUPER PURE ZERO` <int>,
## #   `HILL MOISTURE JUMPIN-FISH` <int>, `VENOMOUS BLAST` <int>, `POW-POW` <int>,
## #   `MYTHICAL BEVERAGE REHAB` <int>,
## #   `MYTHICAL BEVERAGE ULTRA KIWANO UNFLAVORED ` <int>,
```

```
## #   `SUPER-DUPER PURE ZERO KIWANO KEKE ` <int>, …
```

```
write.csv(one_hot_kiwano, "one_hot_kiwano.csv", row.names = FALSE)
```

Load and Prepare the data

```
# Load and prepare dataset
df1 <- read.csv("one_hot_kiwano.csv")
df1 <- df1 %>%
  select(-DATE, -MONTH, -WINTER, -SPRING, -FALL, -DOLLAR_SALES, -SUMMER)
```

```
# Summarize the dataset
skimr::skim(df1)
```

Data summary

Name	df1
Number of rows	8082
Number of columns	31
_______________________	
Column type frequency:	
numeric	31
________________________	
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
UNIT_SALES	0	1	171.20	468.61	0.5	10	66	214	10621	▇▁▁▁▁
NORTHERN	0	1	0.25	0.43	0.0	0	0	0	1	▇▁▁▁▂
ARIZONA	0	1	0.21	0.41	0.0	0	0	0	1	▇▁▁▁▂
MOUNTAIN	0	1	0.10	0.29	0.0	0	0	0	1	▇▁▁▁▁
COLORADO	0	1	0.12	0.32	0.0	0	0	0	1	▇▁▁▁▁
DESERT_SW	0	1	0.07	0.26	0.0	0	0	0	1	▇▁▁▁▁
NOCAL	0	1	0.04	0.20	0.0	0	0	0	1	▇▁▁▁▁
SOCAL	0	1	0.09	0.28	0.0	0	0	0	1	▇▁▁▁▁
KANSAS	0	1	0.02	0.14	0.0	0	0	0	1	▇▁▁▁▁
NEWMEXICO	0	1	0.04	0.20	0.0	0	0	0	1	▇▁▁▁▁
CALI_NEVADA	0	1	0.04	0.19	0.0	0	0	0	1	▇▁▁▁▁
PRAIRIE	0	1	0.03	0.17	0.0	0	0	0	1	▇▁▁▁▁
MYTHICAL.BEVERAGE.ULTRA	0	1	0.55	0.50	0.0	0	1	1	1	▆▁▁▁▇

SUPER.DUPER.PURE.ZERO	0	1	0.37	0.48	0.0	0	0	1	1	
HILL.MOISTURE.JUMPIN.FISH	0	1	0.04	0.19	0.0	0	0	0	1	
VENOMOUS.BLAST	0	1	0.03	0.17	0.0	0	0	0	1	
POW.POW	0	1	0.01	0.10	0.0	0	0	0	1	
MYTHICAL.BEVERAGE.REHAB	0	1	0.00	0.02	0.0	0	0	0	1	
MYTHICAL.BEVERAGE.ULTRA.KIWANO.UNFLAVORED.	0	1	0.55	0.50	0.0	0	1	1	1	
SUPER.DUPER.PURE.ZERO.KIWANO.KEKE.	0	1	0.30	0.46	0.0	0	0	1	1	
RAINING.JUMPIN.FISH.GAME.FUEL.ZERO.CHARGED.KIWANO.SHOCK.	0	1	0.04	0.19	0.0	0	0	0	1	
SUPER.DUPER.PURE.ZERO.KIWANO.	0	1	0.37	0.48	0.0	0	0	1	1	
VENOMOUS.BLAST.KIWANO.DURIAN.	0	1	0.03	0.17	0.0	0	0	0	1	
POW.POW.WYLDIN.KIWANO.	0	1	0.01	0.10	0.0	0	0	0	1	
MYTHICAL.BEVERAGE.REHAB.KIWANO.	0	1	0.00	0.02	0.0	0	0	0	1	
X16SMALL.4ONE.CUP	0	1	0.15	0.36	0.0	0	0	0	1	
X16SMALL.MULTI.CUP	0	1	0.77	0.42	0.0	1	1	1	1	
X16SMALL.24ONE.CUP	0	1	0.07	0.25	0.0	0	0	0	1	
X16SMALL.12ONE.CUP	0	1	0.01	0.10	0.0	0	0	0	1	
Week_Of_Year	0	1	25.13	15.37	1.0	12	24	39	53	
Week_Since_Launch	0	1	65.58	40.11	1.0	29	66	99	152	

> One Hot encoded down to just over 8000 rows from sampled data and up to 33 features.

```
#Remove outliers in top 1% of Unit Sales.
df1 <- df1 %>% filter(UNIT_SALES < quantile(UNIT_SALES, 0.99))
```

```
# Split the data
set.seed(123)
df_testtrn <- initial_split(df1, prop = 0.8, strata = UNIT_SALES)
Train <- training(df_testtrn)
Test <- testing(df_testtrn)

# Prepare features and labels for XGBoost
train_features <- Train[, -which(names(Train) == "UNIT_SALES")]
train_labels <- Train$UNIT_SALES
test_features <- Test[, -which(names(Test) == "UNIT_SALES")]
test_labels <- Test$UNIT_SALES

# Convert data to DMatrix format
dtrain <- xgb.DMatrix(data = as.matrix(train_features), label = train_labels)
dtest <- xgb.DMatrix(data = as.matrix(test_features), label = test_labels)
```

```
# Define XGBoost parameters
set.seed(123)
params <- list(
```

```
  booster = "gbtree",
  objective = "reg:squarederror",
  eval_metric = "rmse",
  eta = 0.05,
  max_depth = 4,
  min_child_weight = 3,
  subsample = 0.7,
  colsample_bytree = 0.6,
  lambda = 1,
  alpha = 1
)
```

```
# Perform cross-validation to find the optimal number of boosting rounds
cv_results <- xgb.cv(
  params = params,
  data = dtrain,
  nfold = 5,
  nrounds = 500,  # Changed from 'num_boost_round' to 'nrounds'
  early_stopping_rounds = 10,
  metrics = "rmse",
  seed = 123
)
```

```
## [1]  train-rmse:217.360255+1.554030  test-rmse:217.293864+6.138701
## Multiple eval metrics are present. Will use test_rmse for early stopping.
## Will train until test_rmse hasn't improved in 10 rounds.
##
## [2]  train-rmse:209.961666+1.043515  test-rmse:209.885941+6.640715
## [3]  train-rmse:202.915207+1.030470  test-rmse:202.905051+6.711466
## [4]  train-rmse:196.207610+1.017242  test-rmse:196.202576+6.721541
## [5]  train-rmse:189.912581+1.038598  test-rmse:189.955217+6.696027
## [6]  train-rmse:185.004751+1.524953  test-rmse:185.045603+6.601243
## [7]  train-rmse:179.366715+1.455260  test-rmse:179.449307+6.649602
## [8]  train-rmse:174.537922+1.135570  test-rmse:174.673142+6.594987
## [9]  train-rmse:170.068956+0.905460  test-rmse:170.175863+7.077023
## [10] train-rmse:165.568663+0.925538  test-rmse:165.679325+6.897081
## [11] train-rmse:161.551248+1.027171  test-rmse:161.637536+6.919887
## [12] train-rmse:157.558015+1.222399  test-rmse:157.649809+6.717110
## [13] train-rmse:154.173442+1.580461  test-rmse:154.313175+6.313114
## [14] train-rmse:150.637531+1.566631  test-rmse:150.832064+6.261417
## [15] train-rmse:147.424726+1.423120  test-rmse:147.627254+6.261076
## [16] train-rmse:144.893609+1.515872  test-rmse:145.145605+6.474148
## [17] train-rmse:142.079329+1.492156  test-rmse:142.323735+6.428708
## [18] train-rmse:139.604694+1.504110  test-rmse:139.871289+6.521195
## [19] train-rmse:137.229412+1.350678  test-rmse:137.519038+6.496247
## [20] train-rmse:134.982804+1.382129  test-rmse:135.294056+6.377677
## [21] train-rmse:133.022947+1.410864  test-rmse:133.380242+6.433893
## [22] train-rmse:131.075874+1.327285  test-rmse:131.447363+6.426331
## [23] train-rmse:129.523206+1.343229  test-rmse:129.924215+6.428968
## [24] train-rmse:127.978419+1.416049  test-rmse:128.413026+6.298471
## [25] train-rmse:126.472364+1.524174  test-rmse:126.943370+6.180198
## [26] train-rmse:125.202023+1.607562  test-rmse:125.690935+5.991786
## [27] train-rmse:123.902173+1.402830  test-rmse:124.395792+6.097224
## [28] train-rmse:122.762665+1.526602  test-rmse:123.277907+5.889761
## [29] train-rmse:121.654197+1.470335  test-rmse:122.169469+5.880672
## [30] train-rmse:120.641471+1.350852  test-rmse:121.179148+5.893031
```

```
## [31] train-rmse:119.579547+1.307540  test-rmse:120.136370+5.837420
## [32] train-rmse:118.610219+1.279854  test-rmse:119.178132+5.781013
## [33] train-rmse:117.731891+1.307944  test-rmse:118.388016+5.716035
## [34] train-rmse:116.918090+1.336000  test-rmse:117.603366+5.644581
## [35] train-rmse:116.236564+1.317775  test-rmse:116.951541+5.586579
## [36] train-rmse:115.613887+1.328873  test-rmse:116.367394+5.538723
## [37] train-rmse:114.969078+1.365188  test-rmse:115.756148+5.471661
## [38] train-rmse:114.348154+1.377897  test-rmse:115.167330+5.373315
## [39] train-rmse:113.751339+1.369212  test-rmse:114.621706+5.302633
## [40] train-rmse:113.236290+1.356584  test-rmse:114.144036+5.227821
## [41] train-rmse:112.723290+1.328107  test-rmse:113.647114+5.163184
## [42] train-rmse:112.235027+1.355114  test-rmse:113.172577+5.085923
## [43] train-rmse:111.784884+1.336737  test-rmse:112.746740+5.074090
## [44] train-rmse:111.362021+1.307900  test-rmse:112.369615+5.039886
## [45] train-rmse:110.981580+1.246371  test-rmse:112.026961+5.076612
## [46] train-rmse:110.627002+1.238048  test-rmse:111.675433+5.017185
## [47] train-rmse:110.292031+1.237893  test-rmse:111.398181+4.984354
## [48] train-rmse:110.006497+1.217585  test-rmse:111.148723+4.959111
## [49] train-rmse:109.712695+1.234352  test-rmse:110.888276+4.894933
## [50] train-rmse:109.429788+1.186899  test-rmse:110.631836+4.882064
## [51] train-rmse:109.200934+1.181311  test-rmse:110.413114+4.846753
## [52] train-rmse:108.998921+1.153347  test-rmse:110.227842+4.857321
## [53] train-rmse:108.734361+1.150690  test-rmse:110.015857+4.836934
## [54] train-rmse:108.513340+1.117215  test-rmse:109.841681+4.852190
## [55] train-rmse:108.334809+1.101925  test-rmse:109.690141+4.855627
## [56] train-rmse:108.111765+1.102506  test-rmse:109.503466+4.818088
## [57] train-rmse:107.933043+1.104581  test-rmse:109.373979+4.778073
## [58] train-rmse:107.782752+1.094942  test-rmse:109.237878+4.760508
## [59] train-rmse:107.622530+1.100039  test-rmse:109.110197+4.731462
## [60] train-rmse:107.460659+1.089963  test-rmse:108.952320+4.729611
## [61] train-rmse:107.312859+1.080224  test-rmse:108.846782+4.741422
## [62] train-rmse:107.175157+1.078931  test-rmse:108.731721+4.722889
## [63] train-rmse:107.030002+1.063585  test-rmse:108.629616+4.734004
## [64] train-rmse:106.903749+1.049886  test-rmse:108.512578+4.734958
## [65] train-rmse:106.780158+1.058395  test-rmse:108.403697+4.711022
## [66] train-rmse:106.649292+1.060358  test-rmse:108.316702+4.672836
## [67] train-rmse:106.544610+1.042611  test-rmse:108.219648+4.670167
## [68] train-rmse:106.423862+1.020642  test-rmse:108.132584+4.679449
## [69] train-rmse:106.331050+1.031526  test-rmse:108.045375+4.662445
## [70] train-rmse:106.221177+1.040694  test-rmse:107.948353+4.658498
## [71] train-rmse:106.117435+1.039078  test-rmse:107.847232+4.647410
## [72] train-rmse:106.029197+1.046324  test-rmse:107.781227+4.645634
## [73] train-rmse:105.943940+1.053722  test-rmse:107.695007+4.626651
## [74] train-rmse:105.847249+1.038886  test-rmse:107.618468+4.630071
## [75] train-rmse:105.741791+1.031579  test-rmse:107.554973+4.614412
## [76] train-rmse:105.641216+1.008827  test-rmse:107.496592+4.605641
## [77] train-rmse:105.535785+1.018294  test-rmse:107.435202+4.587643
## [78] train-rmse:105.452102+1.019996  test-rmse:107.394942+4.551735
## [79] train-rmse:105.337801+1.011757  test-rmse:107.333303+4.542955
## [80] train-rmse:105.272222+1.004337  test-rmse:107.267733+4.540073
## [81] train-rmse:105.192001+0.998100  test-rmse:107.200580+4.548317
## [82] train-rmse:105.108544+0.993164  test-rmse:107.166407+4.526107
## [83] train-rmse:105.015570+1.008158  test-rmse:107.118452+4.503468
## [84] train-rmse:104.932408+1.002655  test-rmse:107.033451+4.492657
## [85] train-rmse:104.866123+0.995255  test-rmse:106.979789+4.490348
## [86] train-rmse:104.790773+0.994574  test-rmse:106.910228+4.490144
## [87] train-rmse:104.730471+0.984160  test-rmse:106.860499+4.483792
```

```
## [88] train-rmse:104.655813+0.988393  test-rmse:106.788952+4.475099
## [89] train-rmse:104.581253+0.996125  test-rmse:106.731611+4.472623
## [90] train-rmse:104.524967+0.999808  test-rmse:106.705235+4.471444
## [91] train-rmse:104.474715+0.993609  test-rmse:106.673815+4.473147
## [92] train-rmse:104.386729+0.994523  test-rmse:106.615075+4.460088
## [93] train-rmse:104.336628+0.998422  test-rmse:106.575322+4.458317
## [94] train-rmse:104.258353+1.003485  test-rmse:106.561357+4.434379
## [95] train-rmse:104.189010+1.013041  test-rmse:106.520165+4.410251
## [96] train-rmse:104.129404+1.014710  test-rmse:106.486185+4.424350
## [97] train-rmse:104.076297+1.011025  test-rmse:106.444238+4.420954
## [98] train-rmse:104.009894+1.022112  test-rmse:106.419966+4.384238
## [99] train-rmse:103.960181+1.025132  test-rmse:106.373462+4.375097
## [100]    train-rmse:103.909431+1.030437  test-rmse:106.344093+4.364590
## [101]    train-rmse:103.854588+1.039341  test-rmse:106.297992+4.356818
## [102]    train-rmse:103.785752+1.001281  test-rmse:106.248053+4.377966
## [103]    train-rmse:103.717114+0.992282  test-rmse:106.204561+4.378611
## [104]    train-rmse:103.661938+0.991903  test-rmse:106.190650+4.385458
## [105]    train-rmse:103.627512+0.982138  test-rmse:106.171539+4.379724
## [106]    train-rmse:103.566078+0.975684  test-rmse:106.134831+4.366116
## [107]    train-rmse:103.520901+0.977032  test-rmse:106.109050+4.356163
## [108]    train-rmse:103.469431+0.984242  test-rmse:106.096363+4.381008
## [109]    train-rmse:103.388053+0.975563  test-rmse:106.039944+4.404025
## [110]    train-rmse:103.344622+0.978713  test-rmse:106.027838+4.391944
## [111]    train-rmse:103.288371+0.983694  test-rmse:105.988407+4.365553
## [112]    train-rmse:103.228487+1.000494  test-rmse:105.953051+4.335968
## [113]    train-rmse:103.175833+0.993622  test-rmse:105.934773+4.341889
## [114]    train-rmse:103.110438+1.004290  test-rmse:105.915251+4.313697
## [115]    train-rmse:103.063452+1.014282  test-rmse:105.886714+4.307094
## [116]    train-rmse:103.007479+1.011217  test-rmse:105.883297+4.290814
## [117]    train-rmse:102.971302+1.018176  test-rmse:105.866060+4.290082
## [118]    train-rmse:102.928029+1.034638  test-rmse:105.822685+4.274057
## [119]    train-rmse:102.877733+1.021813  test-rmse:105.804543+4.287448
## [120]    train-rmse:102.831669+1.033473  test-rmse:105.803135+4.269271
## [121]    train-rmse:102.793184+1.028124  test-rmse:105.795582+4.269625
## [122]    train-rmse:102.750890+1.028588  test-rmse:105.782692+4.251959
## [123]    train-rmse:102.708742+1.038277  test-rmse:105.747815+4.237477
## [124]    train-rmse:102.656623+1.032563  test-rmse:105.716632+4.232614
## [125]    train-rmse:102.618892+1.024486  test-rmse:105.710269+4.214318
## [126]    train-rmse:102.572592+1.033707  test-rmse:105.695871+4.219118
## [127]    train-rmse:102.521134+1.037965  test-rmse:105.658790+4.193492
## [128]    train-rmse:102.474128+1.026091  test-rmse:105.646656+4.200731
## [129]    train-rmse:102.437232+1.016678  test-rmse:105.632236+4.191049
## [130]    train-rmse:102.395422+1.015498  test-rmse:105.602903+4.175004
## [131]    train-rmse:102.349011+1.017557  test-rmse:105.585602+4.169131
## [132]    train-rmse:102.316906+1.016105  test-rmse:105.581606+4.145101
## [133]    train-rmse:102.272762+1.006048  test-rmse:105.569334+4.136833
## [134]    train-rmse:102.236333+1.001512  test-rmse:105.564544+4.137326
## [135]    train-rmse:102.188406+0.991955  test-rmse:105.562437+4.136805
## [136]    train-rmse:102.148351+0.992707  test-rmse:105.551784+4.135809
## [137]    train-rmse:102.109242+0.985014  test-rmse:105.537104+4.140476
## [138]    train-rmse:102.068653+0.996073  test-rmse:105.508292+4.109137
## [139]    train-rmse:102.035334+0.988909  test-rmse:105.510325+4.092499
## [140]    train-rmse:101.995723+0.979024  test-rmse:105.508204+4.081129
## [141]    train-rmse:101.960939+0.978380  test-rmse:105.487657+4.073747
## [142]    train-rmse:101.923792+0.973016  test-rmse:105.474605+4.074237
## [143]    train-rmse:101.873908+0.971191  test-rmse:105.450794+4.075994
## [144]    train-rmse:101.838662+0.978781  test-rmse:105.441942+4.056984
```

```
## [145]	train-rmse:101.811505+0.983418	test-rmse:105.419699+4.054022
## [146]	train-rmse:101.769985+0.977941	test-rmse:105.404921+4.058857
## [147]	train-rmse:101.710334+0.976514	test-rmse:105.395853+4.076458
## [148]	train-rmse:101.676151+0.983626	test-rmse:105.382296+4.072826
## [149]	train-rmse:101.650528+0.979141	test-rmse:105.374901+4.067017
## [150]	train-rmse:101.612983+0.975917	test-rmse:105.358027+4.061669
## [151]	train-rmse:101.577076+0.969239	test-rmse:105.348437+4.067003
## [152]	train-rmse:101.532004+0.953754	test-rmse:105.342215+4.072343
## [153]	train-rmse:101.503294+0.950697	test-rmse:105.337528+4.076989
## [154]	train-rmse:101.459430+0.934362	test-rmse:105.332953+4.071023
## [155]	train-rmse:101.429511+0.924060	test-rmse:105.319022+4.074265
## [156]	train-rmse:101.401590+0.926597	test-rmse:105.301297+4.062705
## [157]	train-rmse:101.364559+0.941475	test-rmse:105.296007+4.055449
## [158]	train-rmse:101.333558+0.943358	test-rmse:105.290568+4.063727
## [159]	train-rmse:101.300026+0.949426	test-rmse:105.281461+4.048562
## [160]	train-rmse:101.262241+0.949586	test-rmse:105.283085+4.042226
## [161]	train-rmse:101.236295+0.951559	test-rmse:105.279289+4.027357
## [162]	train-rmse:101.207699+0.952200	test-rmse:105.262229+4.026568
## [163]	train-rmse:101.171520+0.941064	test-rmse:105.255794+4.019354
## [164]	train-rmse:101.141551+0.943195	test-rmse:105.249425+4.028841
## [165]	train-rmse:101.091790+0.933623	test-rmse:105.230038+4.012656
## [166]	train-rmse:101.062565+0.934304	test-rmse:105.206876+4.008377
## [167]	train-rmse:101.032856+0.941756	test-rmse:105.215568+3.991930
## [168]	train-rmse:101.004854+0.929112	test-rmse:105.219441+3.984197
## [169]	train-rmse:100.975163+0.928116	test-rmse:105.204518+3.985835
## [170]	train-rmse:100.954551+0.920970	test-rmse:105.209788+3.981172
## [171]	train-rmse:100.932495+0.923737	test-rmse:105.213487+3.988734
## [172]	train-rmse:100.899051+0.941276	test-rmse:105.196212+3.987949
## [173]	train-rmse:100.869720+0.936706	test-rmse:105.200657+3.980179
## [174]	train-rmse:100.836641+0.940116	test-rmse:105.191904+3.982627
## [175]	train-rmse:100.808636+0.927988	test-rmse:105.203999+3.975997
## [176]	train-rmse:100.775988+0.933001	test-rmse:105.223098+3.965243
## [177]	train-rmse:100.753351+0.928336	test-rmse:105.199056+3.958170
## [178]	train-rmse:100.714652+0.935254	test-rmse:105.180758+3.951215
## [179]	train-rmse:100.687634+0.928764	test-rmse:105.169464+3.946365
## [180]	train-rmse:100.661680+0.916983	test-rmse:105.145819+3.958059
## [181]	train-rmse:100.637991+0.923787	test-rmse:105.137384+3.957639
## [182]	train-rmse:100.616686+0.924841	test-rmse:105.133660+3.953471
## [183]	train-rmse:100.593796+0.927075	test-rmse:105.135133+3.953567
## [184]	train-rmse:100.571169+0.927013	test-rmse:105.128883+3.936286
## [185]	train-rmse:100.547877+0.927629	test-rmse:105.113502+3.935448
## [186]	train-rmse:100.524132+0.930284	test-rmse:105.110521+3.946861
## [187]	train-rmse:100.499036+0.933743	test-rmse:105.107644+3.939000
## [188]	train-rmse:100.475305+0.936061	test-rmse:105.102987+3.932346
## [189]	train-rmse:100.445939+0.931409	test-rmse:105.076154+3.929965
## [190]	train-rmse:100.423833+0.925547	test-rmse:105.065751+3.932928
## [191]	train-rmse:100.399728+0.924915	test-rmse:105.081433+3.928621
## [192]	train-rmse:100.379905+0.922316	test-rmse:105.079550+3.930478
## [193]	train-rmse:100.350551+0.926717	test-rmse:105.074291+3.928314
## [194]	train-rmse:100.332808+0.925513	test-rmse:105.079066+3.924849
## [195]	train-rmse:100.316628+0.923353	test-rmse:105.079985+3.922235
## [196]	train-rmse:100.278531+0.923287	test-rmse:105.072979+3.926553
## [197]	train-rmse:100.250699+0.924200	test-rmse:105.079120+3.926247
## [198]	train-rmse:100.227501+0.921685	test-rmse:105.071988+3.937932
## [199]	train-rmse:100.199090+0.921012	test-rmse:105.053004+3.932809
## [200]	train-rmse:100.184993+0.916710	test-rmse:105.049874+3.926577
## [201]	train-rmse:100.163042+0.913585	test-rmse:105.053422+3.931739
```

```
## [202]	train-rmse:100.140621+0.909076	test-rmse:105.055969+3.934605
## [203]	train-rmse:100.120766+0.906004	test-rmse:105.051717+3.922006
## [204]	train-rmse:100.089851+0.905941	test-rmse:105.049733+3.902391
## [205]	train-rmse:100.068420+0.907145	test-rmse:105.025565+3.902986
## [206]	train-rmse:100.038110+0.904924	test-rmse:105.018105+3.901082
## [207]	train-rmse:100.011403+0.911147	test-rmse:105.021181+3.904935
## [208]	train-rmse:99.993877+0.914025	test-rmse:105.025434+3.913648
## [209]	train-rmse:99.968086+0.917852	test-rmse:105.006846+3.932061
## [210]	train-rmse:99.943647+0.919612	test-rmse:105.005253+3.944709
## [211]	train-rmse:99.917861+0.911289	test-rmse:104.992500+3.949810
## [212]	train-rmse:99.900159+0.906863	test-rmse:104.988020+3.944087
## [213]	train-rmse:99.868428+0.908866	test-rmse:105.004346+3.934326
## [214]	train-rmse:99.843549+0.916574	test-rmse:104.995203+3.919110
## [215]	train-rmse:99.817815+0.910998	test-rmse:104.985636+3.926648
## [216]	train-rmse:99.799442+0.908894	test-rmse:104.994856+3.919838
## [217]	train-rmse:99.778567+0.915381	test-rmse:104.999397+3.918101
## [218]	train-rmse:99.747183+0.922440	test-rmse:104.977227+3.897420
## [219]	train-rmse:99.726412+0.923485	test-rmse:104.988142+3.892196
## [220]	train-rmse:99.704487+0.922033	test-rmse:104.991317+3.888281
## [221]	train-rmse:99.677993+0.916716	test-rmse:104.983141+3.889095
## [222]	train-rmse:99.657252+0.908815	test-rmse:104.985413+3.877679
## [223]	train-rmse:99.628451+0.903684	test-rmse:104.988353+3.885696
## [224]	train-rmse:99.600048+0.901580	test-rmse:104.986457+3.897182
## [225]	train-rmse:99.572526+0.902859	test-rmse:104.985545+3.891857
## [226]	train-rmse:99.552160+0.902181	test-rmse:105.002374+3.878769
## [227]	train-rmse:99.528547+0.899692	test-rmse:104.999507+3.867543
## [228]	train-rmse:99.510698+0.897851	test-rmse:104.990621+3.855139
## Stopping. Best iteration:
## [218]	train-rmse:99.747183+0.922440	test-rmse:104.977227+3.897420
```

```
best_nrounds <- cv_results$best_iteration
```

Create Model

```
# Train the final model using the best number of rounds found
model_xgb <- xgb.train(
  params = params,
  data = dtrain,
  nrounds = best_nrounds
)
```

Setup Train and Test

```
# Make predictions and evaluate the model
train_pred <- predict(model_xgb, dtrain)
test_pred <- predict(model_xgb, dtest)
train_rmse <- sqrt(mean((train_labels - train_pred)^2))
test_rmse <- sqrt(mean((test_labels - test_pred)^2))
```

Create Model Metrics

```
# Calculate R-squared for the training set
sst_train <- sum((train_labels - mean(train_labels)) ^ 2)
ssr_train <- sum((train_labels - train_pred) ^ 2)
r_squared_train <- 1 - (ssr_train / sst_train)

# Calculate R-squared for the test set
sst_test <- sum((test_labels - mean(test_labels)) ^ 2)
ssr_test <- sum((test_labels - test_pred) ^ 2)
r_squared_test <- 1 - (ssr_test / sst_test)

train_mape <- mean(abs((train_labels - train_pred) / train_labels)) * 100
test_mape <- mean(abs((test_labels - test_pred) / test_labels)) * 100
train_mae <- mean(abs(train_labels - train_pred))
test_mae <- mean(abs(test_labels - test_pred))
```

Output Results

```
cat("Model Performance Metrics:\n",
    "-------------------------\n",
    "Training RMSE: ", train_rmse, "\n",
    "Test RMSE: ", test_rmse, "\n",
    "Training R-squared: ", r_squared_train, "\n",
    "Test R-squared: ", r_squared_test, "\n",
    "Training MAE: ", train_mae, "\n",
    "Test MAE: ", test_mae, "\n",
    "Training MAPE: ", train_mape, "%\n",
    "Test MAPE: ", test_mape, "%\n", sep="")
```

```
## Model Performance Metrics:
## -------------------------
## Training RMSE: 100.2922
## Test RMSE: 109.3214
## Training R-squared: 0.6834959
## Test R-squared: 0.6279935
## Training MAE: 59.00142
## Test MAE: 62.41945
## Training MAPE: 232.4294%
## Test MAPE: 224.7447%
```

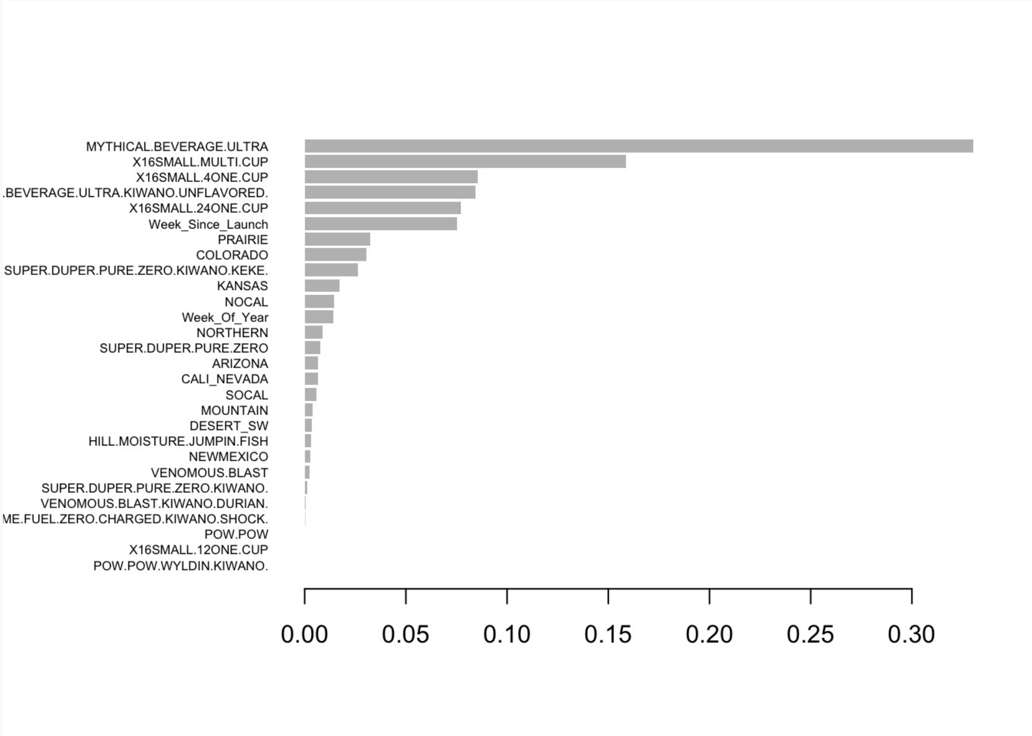
> For the Kiwano Energy model, Our train RMSE is 100.29 and test 109.32. We expect to see the drop from train to test. With the difference we may need to check if there is slight overfitting. With the R2 for test and train are both moderate at .68% training .67% testing, this indicates there is some but not all variance eplained by our model. Our MAE also is low and does not contain a significant difference between training and test. The last metric, MAPE, both values are at 232% meaning that we are with about 224% of the actual values. Overall this model does show some predictive power but with more features we maybe able to get stronger predictive power.

```
# Calculate feature importance
importance_matrix2 <- xgb.importance(feature_names = colnames(train_features), model =
model_xgb)

# View the feature importance scores
print(importance_matrix2)
```

```
##                                                       Feature         Gain
##  1:                                    MYTHICAL.BEVERAGE.ULTRA 3.304469e-01
##  2:                                         X16SMALL.MULTI.CUP 1.587434e-01
##  3:                                          X16SMALL.4ONE.CUP 8.550262e-02
##  4:                MYTHICAL.BEVERAGE.ULTRA.KIWANO.UNFLAVORED. 8.428400e-02
##  5:                                         X16SMALL.24ONE.CUP 7.729816e-02
##  6:                                          Week_Since_Launch 7.542692e-02
##  7:                                                    PRAIRIE 3.223505e-02
##  8:                                                   COLORADO 3.033846e-02
##  9:                        SUPER.DUPER.PURE.ZERO.KIWANO.KEKE. 2.614484e-02
## 10:                                                     KANSAS 1.719257e-02
## 11:                                                      NOCAL 1.450320e-02
## 12:                                               Week_Of_Year 1.412197e-02
## 13:                                                   NORTHERN 8.905549e-03
## 14:                                      SUPER.DUPER.PURE.ZERO 7.738604e-03
## 15:                                                    ARIZONA 6.605826e-03
## 16:                                                CALI_NEVADA 6.592295e-03
## 17:                                                      SOCAL 5.962191e-03
## 18:                                                   MOUNTAIN 3.982386e-03
## 19:                                                  DESERT_SW 3.526797e-03
## 20:                                  HILL.MOISTURE.JUMPIN.FISH 3.255008e-03
## 21:                                                  NEWMEXICO 2.808774e-03
## 22:                                             VENOMOUS.BLAST 2.249979e-03
## 23:                             SUPER.DUPER.PURE.ZERO.KIWANO. 1.416933e-03
## 24:                             VENOMOUS.BLAST.KIWANO.DURIAN. 5.235634e-04
## 25: RAINING.JUMPIN.FISH.GAME.FUEL.ZERO.CHARGED.KIWANO.SHOCK. 1.807011e-04
## 26:                                                    POW.POW 8.304176e-06
## 27:                                         X16SMALL.12ONE.CUP 3.906587e-06
## 28:                                   POW.POW.WYLDIN.KIWANO. 1.083434e-06
##                                                       Feature         Gain
##            Cover    Frequency
##  1: 7.316911e-02 0.0720268007
##  2: 7.392576e-02 0.0613065327
##  3: 1.955680e-02 0.0274706868
##  4: 2.445799e-02 0.0261306533
##  5: 1.610995e-02 0.0264656616
##  6: 2.436876e-01 0.2415410385
##  7: 4.494324e-02 0.0341708543
##  8: 3.546342e-02 0.0338358459
##  9: 3.319167e-02 0.0415410385
## 10: 5.572277e-02 0.0398659966
## 11: 3.706493e-02 0.0190954774
## 12: 1.244102e-01 0.1641541039
## 13: 2.540823e-02 0.0328308208
## 14: 8.260354e-03 0.0214405360
## 15: 2.532080e-02 0.0288107203
## 16: 3.513035e-02 0.0167504188
## 17: 2.672667e-02 0.0190954774
## 18: 3.166683e-02 0.0274706868
## 19: 3.602110e-02 0.0204355109
## 20: 2.917123e-03 0.0050251256
## 21: 1.853964e-02 0.0167504188
## 22: 4.562733e-03 0.0073701843
## 23: 1.473561e-03 0.0080402010
## 24: 1.986372e-03 0.0030150754
## 25: 1.858939e-04 0.0026800670
```

```
## 26: 7.435754e-05 0.0016750419
## 27: 1.512791e-05 0.0006700168
## 28: 7.435754e-06 0.0003350084
##            Cover    Frequency
```

```
xgb.plot.importance(importance_matrix = importance_matrix2)
```



>From this Importance matrix we see that brand and size seem to be the two biggest contributors to our model. We also see that the created featrure Week_Since_Launche is playing a large part in the creation of predictions.

Create Dummy Data and attempt prediction

```
# Define vectors for each category
regions <- 1:11
brands <- 1:6
items <- 1:7
package_options <- 1:4

# Create data frame with all combinations of categories
combinations <- expand.grid(Region = regions, Brand = brands, Item = items, Package =
package_options)

# Duplicate each combination 52 times to represent each week of the year
final_df_replicated <- combinations[rep(row.names(combinations), each = 52), ]

# Add a column with values from 1 to 52 for each combination
final_df_replicated$Week_of_Year <- rep(1:52, times = nrow(combinations))

# Duplicate each combination 52 times to represent each week of the year
final_df_replicated <- final_df_replicated[rep(row.names(final_df_replicated), each = 13), ]

# Add a column with values from 1 to 13 for each combination
final_df_replicated$Week_Since_Launch <- rep(1:13, times = nrow(combinations))

final_df_replicated$Region <- unique_values_list$REGION[final_df_replicated$Region]
final_df_replicated$Brand <- unique_values_list$BRAND[final_df_replicated$Brand]
```

```
final_df_replicated$Item <- unique_values_list$ITEM[final_df_replicated$Item]
final_df_replicated$Package <- unique_values_list$PACKAGE[final_df_replicated$Package]

# List to store unique values for each variable
new_unique_values_list <- list()

# Columns to get unique values for
new_columns_to_get_unique_values <- c("Region", "Brand", "Item", "Package")

# Get unique values for each variable and store in the list
for (col in new_columns_to_get_unique_values) {
  new_unique_values_list[[col]] <- unique(final_df_replicated[[col]])
}

# Loop over unique regions and create new columns
for (Region in new_unique_values_list$Region) {
  final_df_replicated[[Region]] <- as.integer(final_df_replicated$Region == Region)
}

# Loop over unique regions and create new columns
for (Brand in new_unique_values_list$Brand) {
  final_df_replicated[[Brand]] <- as.integer(final_df_replicated$Brand == Brand)
}

# Loop over unique regions and create new columns
for (Item in new_unique_values_list$Item) {
  final_df_replicated[[Item]] <- as.integer(final_df_replicated$Item == Item)
}

# Loop over unique regions and create new columns
for (Package in new_unique_values_list$Package) {
  final_df_replicated[[Package]] <- as.integer(final_df_replicated$Package == Package)
}

#Create dummy_data and remove non one hot encoded data
dummy_data <- final_df_replicated %>%
  select(-Region, -Brand, -Item, -Package)

#add a Unit sales column
dummy_data$UNIT_SALES <- NA
dummy_data$UNIT_SALES <- as.numeric(dummy_data$UNIT_SALES)
```

Assure Features are Matching and Predict

```
#rename columes to match original features
dummy_data <- dummy_data %>%
  rename(
    `MYTHICAL.BEVERAGE.ULTRA` = `MYTHICAL BEVERAGE ULTRA`,
    `SUPER.DUPER.PURE.ZERO` = `SUPER-DUPER PURE ZERO`,
    `HILL.MOISTURE.JUMPIN.FISH` = `HILL MOISTURE JUMPIN-FISH`,
    `VENOMOUS.BLAST` = `VENOMOUS BLAST`,
    `POW.POW` = `POW-POW`,
    `MYTHICAL.BEVERAGE.REHAB` = `MYTHICAL BEVERAGE REHAB`,
    `MYTHICAL.BEVERAGE.ULTRA.KIWANO.UNFLAVORED.` = `MYTHICAL BEVERAGE ULTRA KIWANO UNFLAVORED
`,
```

```
    `SUPER.DUPER.PURE.ZERO.KIWANO.KEKE.` = `SUPER-DUPER PURE ZERO KIWANO KEKE `,
    `RAINING.JUMPIN.FISH.GAME.FUEL.ZERO.CHARGED.KIWANO.SHOCK.` = `RAINING JUMPIN-FISH GAME FUEL ZERO CHARGED KIWANO SHOCK `,
    `SUPER.DUPER.PURE.ZERO.KIWANO.` = `SUPER-DUPER PURE ZERO KIWANO `,
    `VENOMOUS.BLAST.KIWANO.DURIAN.` = `VENOMOUS BLAST KIWANO DURIAN `,
    `POW.POW.WYLDIN.KIWANO.` = `POW-POW WYLDIN KIWANO `,
    `MYTHICAL.BEVERAGE.REHAB.KIWANO.` = `MYTHICAL BEVERAGE REHAB KIWANO `,
    `X16SMALL.4ONE.CUP` = `16SMALL 4ONE CUP`,
    `X16SMALL.MULTI.CUP` = `16SMALL MULTI CUP`,
    `X16SMALL.24ONE.CUP` = `16SMALL 24ONE CUP`,
    `X16SMALL.12ONE.CUP` = `16SMALL 12ONE CUP`,
    `Week_Of_Year`  = `Week_of_Year`
  )

# Check for Matching Features
#Get the column names of Test and dummy_data
names_Test <- names(Test)
names_dummy_data <- names(dummy_data)

# Find the matching column names
matching_names <- intersect(names_Test, names_dummy_data)

# Find the non-matching column names
non_matching_names_Test <- setdiff(names_Test, matching_names)
non_matching_names_dummy_data <- setdiff(names_dummy_data, matching_names)

#Print the matching and non-matching column names
cat("Matching column names:", paste(matching_names, collapse = ", "), "\n")
```

```
## Matching column names: UNIT_SALES, NORTHERN, ARIZONA, MOUNTAIN, COLORADO, DESERT_SW, NOCAL, SOCAL, KANSAS, NEWMEXICO, CALI_NEVADA, PRAIRIE, MYTHICAL.BEVERAGE.ULTRA, SUPER.DUPER.PURE.ZERO, HILL.MOISTURE.JUMPIN.FISH, VENOMOUS.BLAST, POW.POW, MYTHICAL.BEVERAGE.REHAB, MYTHICAL.BEVERAGE.ULTRA.KIWANO.UNFLAVORED., SUPER.DUPER.PURE.ZERO.KIWANO.KEKE., RAINING.JUMPIN.FISH.GAME.FUEL.ZERO.CHARGED.KIWANO.SHOCK., SUPER.DUPER.PURE.ZERO.KIWANO., VENOMOUS.BLAST.KIWANO.DURIAN., POW.POW.WYLDIN.KIWANO., MYTHICAL.BEVERAGE.REHAB.KIWANO., X16SMALL.4ONE.CUP, X16SMALL.MULTI.CUP, X16SMALL.24ONE.CUP, X16SMALL.12ONE.CUP, Week_Of_Year, Week_Since_Launch
```

```
cat("Non-matching column names in Test:", paste(non_matching_names_Test, collapse = ", "), "\n")
```

```
## Non-matching column names in Test:
```

```
cat("Non-matching column names in dummy_data:", paste(non_matching_names_dummy_data, collapse = ", "), "\n")
```

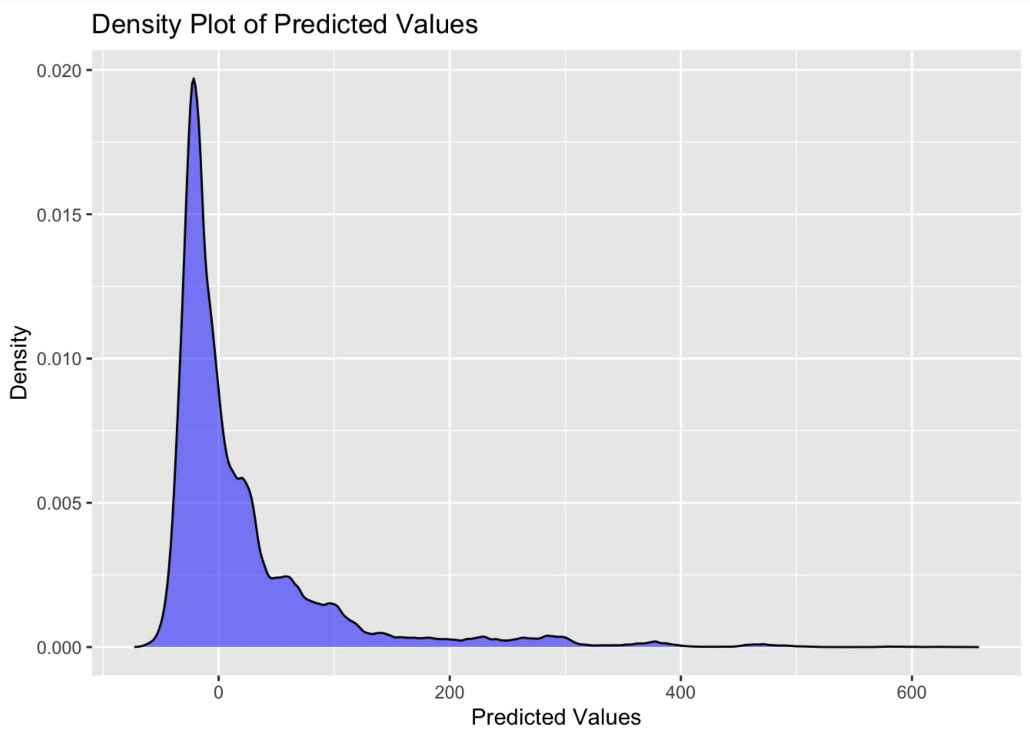
```
## Non-matching column names in dummy_data:
```

```
# Get the column names of the Test dataframe
test_colnames <- colnames(Test)

# Reorder columns of dummy_data to match the order of columns in Test
dummy_data <- dummy_data %>%
  select(all_of(test_colnames))
```

```
# Prepare features for XGBoost
dummy_features <- dummy_data[, -which(names(dummy_data) == "UNIT_SALES")]

# Convert data to DMatrix format
dummy_dmatrix<- xgb.DMatrix(data = as.matrix(dummy_features))

dummy_pred <- predict(model_xgb, dummy_dmatrix)

# Add the predictions to dummy_data
dummy_data$Predictions <- dummy_pred

# Convert predictions to integers
dummy_data$Predictions <- round(dummy_pred)

# Convert to integer data type
dummy_data$Predictions <- as.integer(dummy_data$Predictions)

summary(dummy_data$Predictions)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -73.00  -22.00   -6.00   21.07   31.00  658.00
```

```
ggplot(dummy_data, aes(x = Predictions)) +
  geom_density(fill = "blue", alpha = 0.5) +
  labs(title = "Density Plot of Predicted Values",
       x = "Predicted Values",
       y = "Density")
```




```
dummy_data %>%
  group_by(Week_Since_Launch, Week_Of_Year) %>%
  summarize(Total_Prediction = sum(Predictions, na.rm = TRUE))%>%
  filter(Week_Since_Launch <=13,
         Week_Of_Year <=24)
```

```
## `summarise()` has grouped output by 'Week_Since_Launch'. You can override using
## the `.groups` argument.
```

```
## # A tibble: 312 × 3
## # Groups:   Week_Since_Launch [13]
##    Week_Since_Launch Week_Of_Year Total_Prediction
##                <int>        <int>            <int>
##  1                 1            1            13114
##  2                 1            2            14329
##  3                 1            3            20001
##  4                 1            4            26952
##  5                 1            5            11969
##  6                 1            6            12265
##  7                 1            7            10152
##  8                 1            8             1138
##  9                 1            9             2919
## 10                 1           10            -2271
## # ℹ 302 more rows
```

> In this first round of predcition we attempted to have our model predict a value for every combination of Item, Region, Brand, and Package, Week of the year and week since launch (1-13) from our comparable data report. The model will need some refinement as it is predicting many negative values, but once tuned this could provide a way to make comparable predictions of a new product launching for 13 weeks in any range during the year.