Predicting Inventory Demand Based in Sales

Marcone

2020-07-23

```
setwd("C:/Projects")
set.seed(42)

# Import necessary libraries
library(e1071)
```

```
require(lubridate)
```

```
## Loading required package: lubridate
```

```
##
## Attaching package: 'lubridate'
```

```
library(readr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
library(ggplot2)
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```r
library(arules)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'arules'
```

```r
# Read .csv data from zipped folder
folder2 <- "cliente_tabla.zip"
folder3 <- "producto_tabla.zip"
df <- read_csv("sampled_train.csv")
```

```
## Parsed with column specification:
## cols(
##   X1 = col_double(),
##   Semana = col_double(),
##   Agencia_ID = col_double(),
##   Canal_ID = col_double(),
##   Ruta_SAK = col_double(),
##   Cliente_ID = col_double(),
##   Producto_ID = col_double(),
##   Venta_uni_hoy = col_double(),
##   Venta_hoy = col_double(),
##   Dev_uni_proxima = col_double(),
##   Dev_proxima = col_double(),
##   Demanda_uni_equil = col_double()
## )
```

```r
client <- read_csv(unz(folder2,"cliente_tabla.csv"))
```

```
## Parsed with column specification:
## cols(
##   Cliente_ID = col_double(),
##   NombreCliente = col_character()
## )
```

```r
product <- read_csv(unz(folder3,"producto_tabla.csv"))
```

```
## Parsed with column specification:
## cols(
##   Producto_ID = col_double(),
##   NombreProducto = col_character()
## )
```

File failed to load: /extensions/MathZoom.js

```
View(head(df))
View(head(client))
View(head(product))
length(df$Semana)
```

```
## [1] 10000
```

```
# Drop NA values presented in the Data Frame
df <- na.omit(df)

# Chane type of data for each feature/column
feat <- colnames(df)
df[feat[7:11]] <- mapply(as.numeric,df[feat[7:11]])
df[feat[1:6]] <- mapply(as.factor,df[feat[1:6]])

# Merge data frames in order to get the names of each client and product presented in the dat
a
df <- df %>% merge(client, by = "Cliente_ID", all.x = T)
df <- df %>% merge(product, by = "Producto_ID", all.x = T)
```

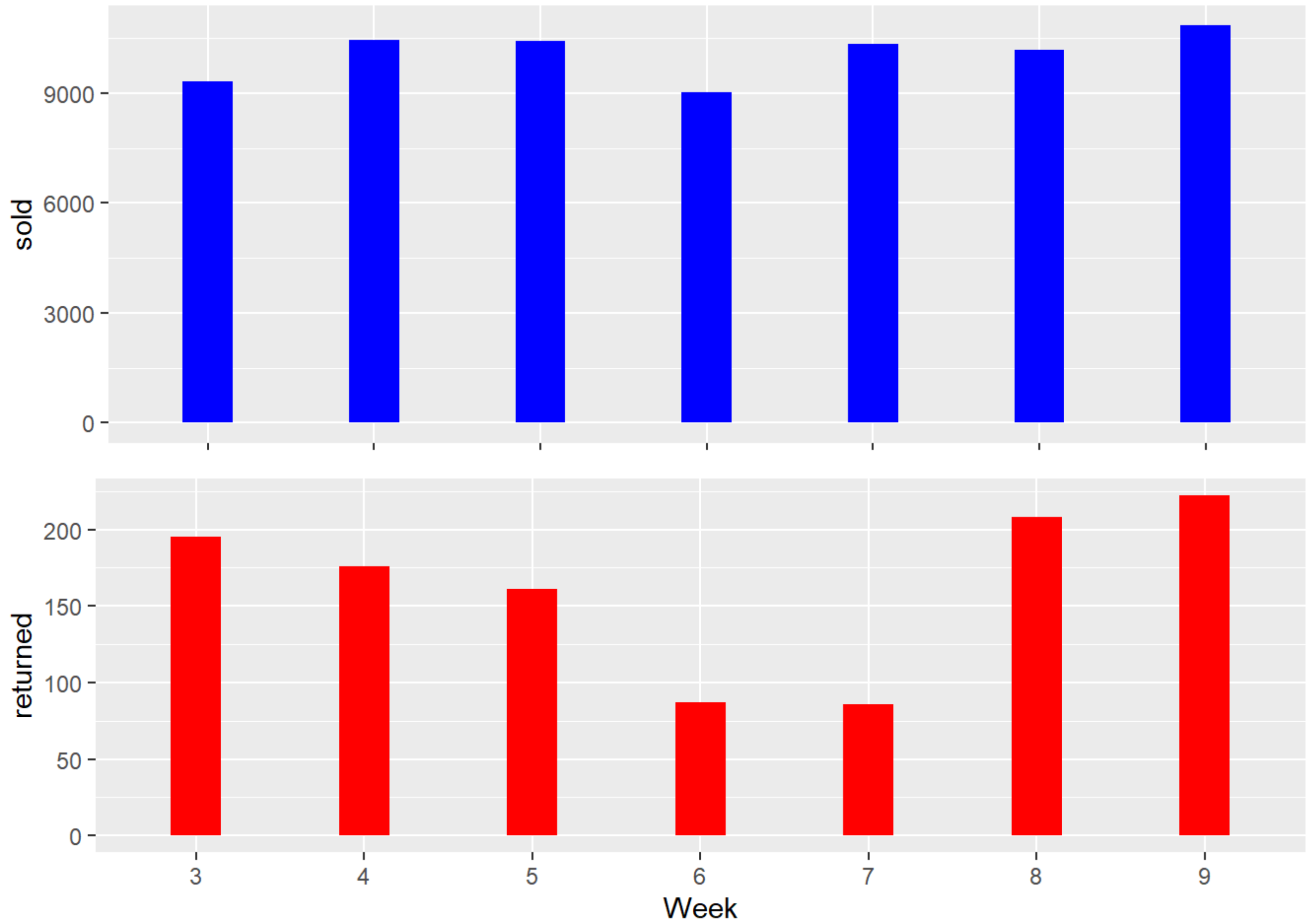
```
summary(df)
```

```
##   Producto_ID     Cliente_ID             X1               Semana
##  Min.   :   72   Length:10084       Length:10084       Length:10084
##  1st Qu.: 1242   Class :character   Class :character   Class :character
##  Median :30552   Mode  :character   Mode  :character   Mode  :character
##  Mean   :21017
##  3rd Qu.:37361
##  Max.   :49973
##   Agencia_ID          Canal_ID           Ruta_SAK          Venta_uni_hoy
##  Length:10084       Length:10084       Length:10084       Min.   :   0.000
##  Class :character   Class :character   Class :character   1st Qu.:   2.000
##  Mode  :character   Mode  :character   Mode  :character   Median :   3.000
##                                                           Mean   :   7.087
##                                                           3rd Qu.:   6.000
##                                                           Max.   :1493.000
##    Venta_hoy        Dev_uni_proxima    Dev_proxima      Demanda_uni_equil
##  Min.   :    0.00   Min.   : 0.0000   Min.   :  0.000   Min.   :   0.000
##  1st Qu.:   16.76   1st Qu.: 0.0000   1st Qu.:  0.000   1st Qu.:   2.000
##  Median :   30.00   Median : 0.0000   Median :  0.000   Median :   3.000
##  Mean   :   67.61   Mean   : 0.1125   Mean   :  1.113   Mean   :   6.998
##  3rd Qu.:   56.10   3rd Qu.: 0.0000   3rd Qu.:  0.000   3rd Qu.:   6.000
##  Max.   :23081.78   Max.   :70.0000   Max.   :433.160   Max.   :1493.000
##  NombreCliente      NombreProducto
##  Length:10084       Length:10084
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

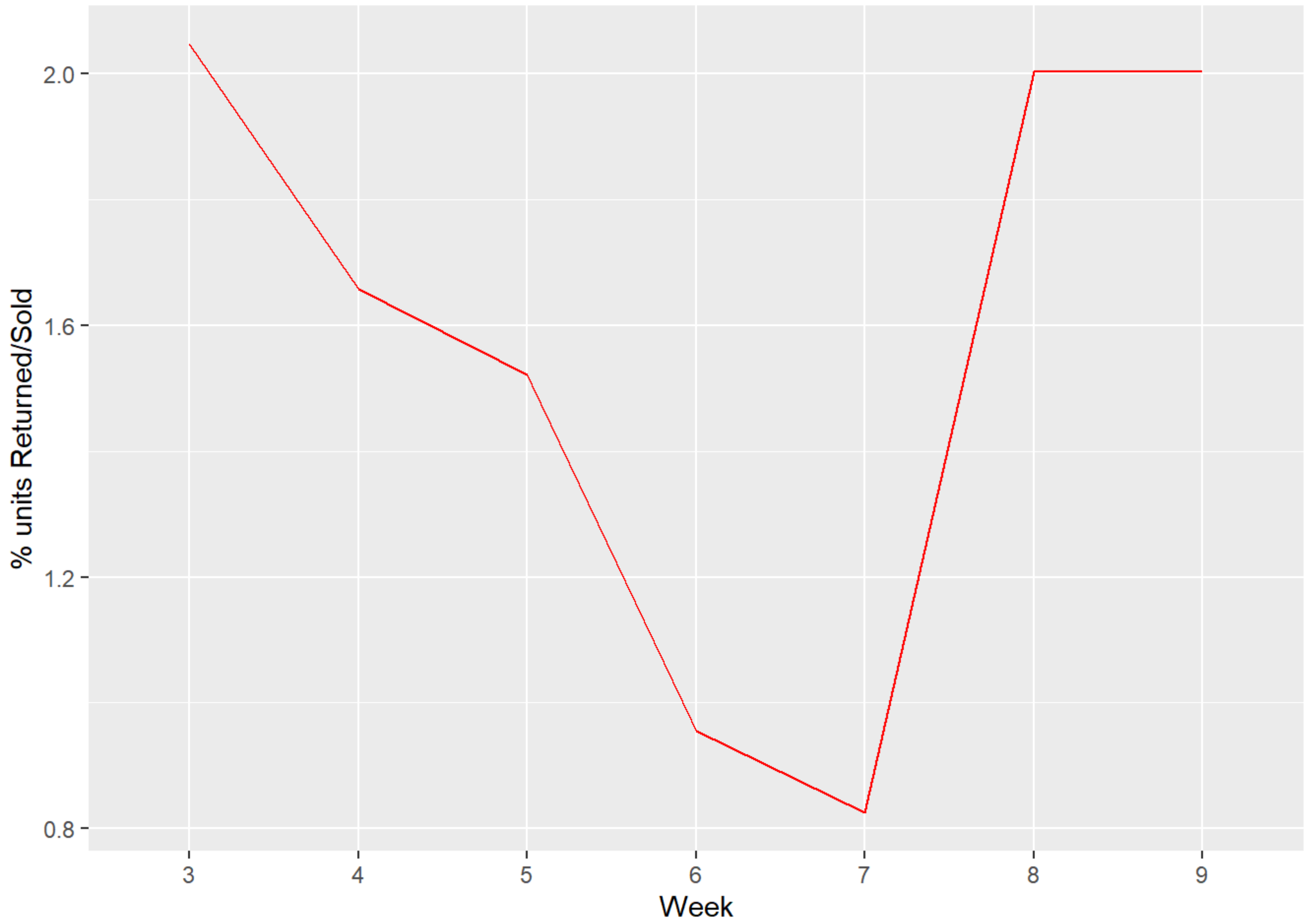
File failed to load: /extensions/MathZoom.js

```
# Generate the following graph: Selling demand vs lost vs weeks
df_sum <- merge(df %>% group_by(Semana) %>% summarise(sold=sum(Demanda_uni_equil)) ,
                     df %>% group_by(Semana) %>% summarise(returned=sum(Dev_uni_proxima)))
```

```
p1 <- ggplot(data=df_sum) + geom_bar(aes(x=Semana, y=sold),stat="identity",fill = "blue", wid
th = 0.3, position = "dodge") +
                            theme(axis.title.x=element_blank(),axis.text.x=element_blank())
p2 <- ggplot(data=df_sum) + geom_bar(aes(x=Semana, y=returned),fill = "red", width = 0.3,stat
="identity")+ labs(x="Week")
grid.arrange(p1, p2, ncol = 1)
```

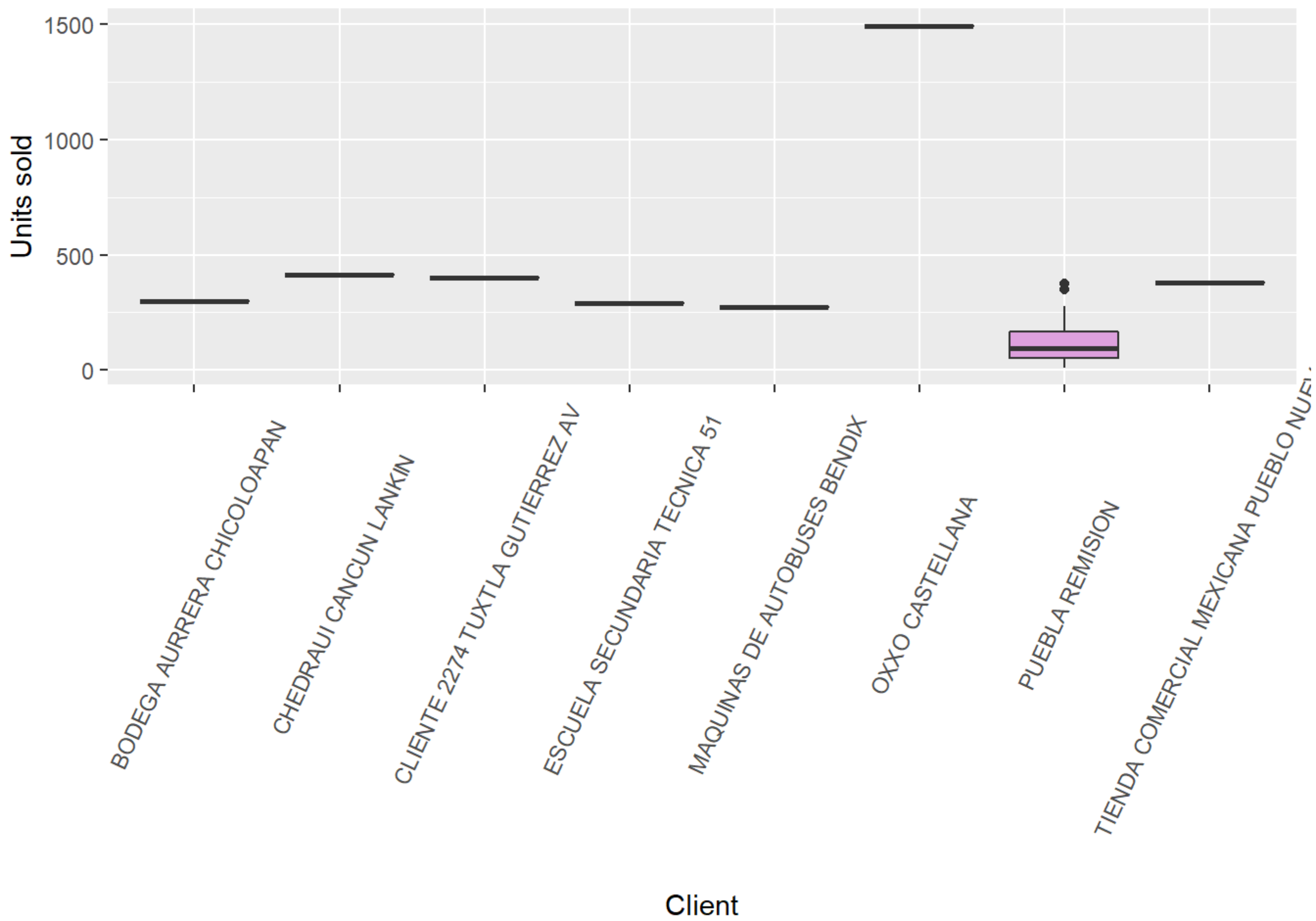



```
# Generate the following graph: weeks vs (units sold)/(total units)
df_sum <- df_sum %>% mutate(rate = 100*returned/(sold+returned))
ggplot(data=df_sum, aes(x=Semana,y=rate)) + geom_line(colour="red",aes(group = 1)) + labs(y=
"% units Returned/Sold", x="Week")
```



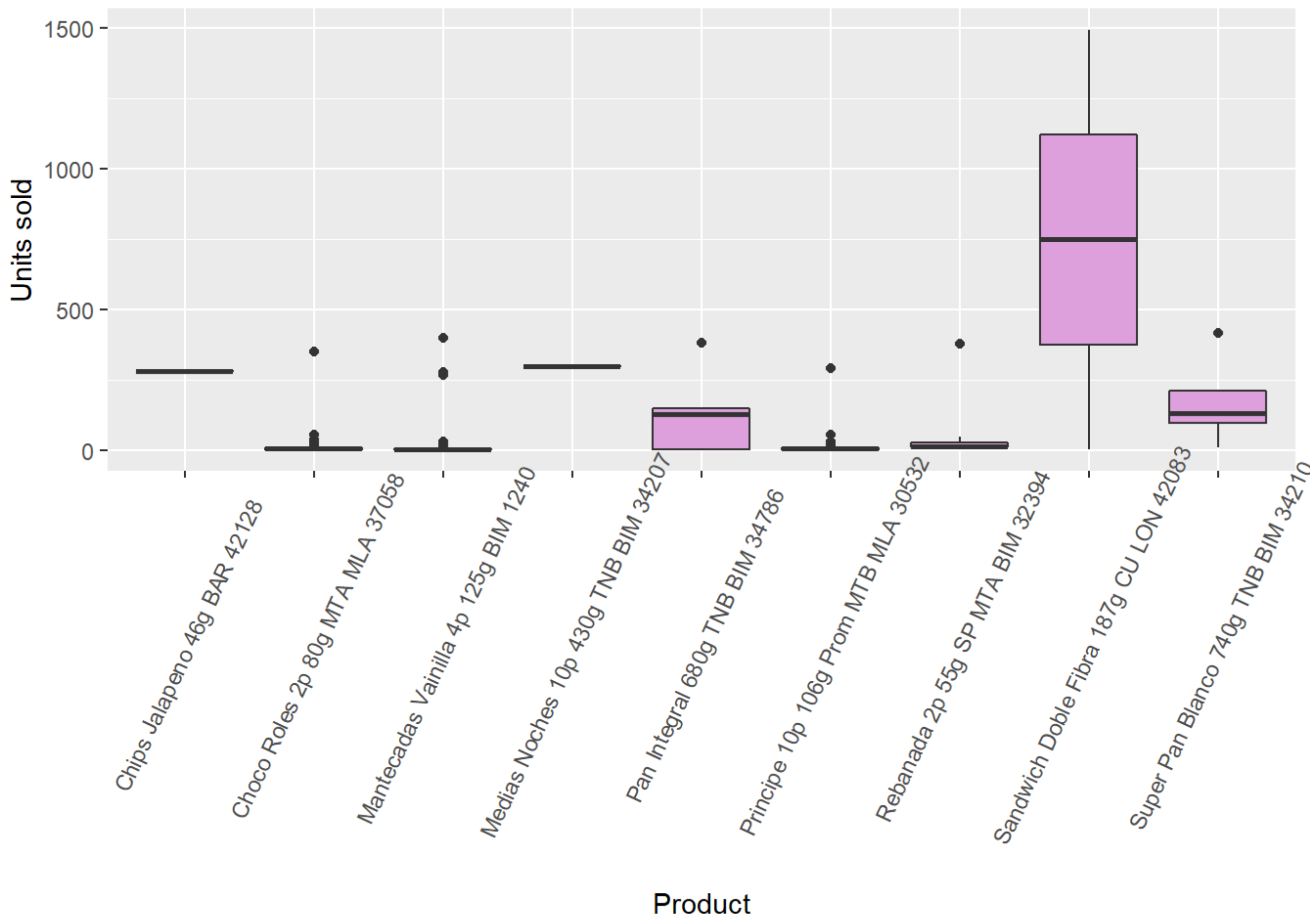

```
# Show the 10 Top Clients (customers who had more products sold)
top_clients <- df %>% slice_max(n=10,order_by=Demanda_uni_equil)
ggplot( df %>% subset(Cliente_ID %in% top_clients$Cliente_ID) ) +
        geom_boxplot( aes(NombreCliente,Demanda_uni_equil), varwidth=F, fill="plum") +
        theme(axis.text.x = element_text(angle=65, vjust=0.6)) + labs( x="Client", y="Units s
old")
```

File failed to load: /extensions/MathZoom.js




```
# Show 10 top selling products
top_products <- df %>% slice_max(n=10,order_by=Demanda_uni_equil)
ggplot( df %>% subset(Producto_ID %in% top_products$Producto_ID) ) +
        geom_boxplot( aes(NombreProducto,Demanda_uni_equil), varwidth=F, fill="plum") +
        theme(axis.text.x = element_text(angle=65, vjust=0.6)) + labs( x="Product", y="Units
 sold")
```

File failed to load: /extensions/MathZoom.js




```
df_new <- df

# Add information of the last 4 weeks:
# 1 week ago
to_merge <- df_new %>% select(Producto_ID,Semana,Demanda_uni_equil)
to_merge$Semana <- to_merge$Semana %>% sapply(function(i){ as.numeric(i)+1 })
to_merge <- to_merge %>% group_by(Producto_ID,Semana) %>% summarise("previous_demanda_1" = su
m(Demanda_uni_equil)) %>% subset(Semana<9)
```

```
to_merge$Semana <- to_merge$Semana %>% as.character()
df_new <- df_new %>% merge(to_merge, by = c("Producto_ID","Semana"), all.x = T)

# 2 weeks ago
to_merge <- df_new %>% select(Producto_ID,Semana,Demanda_uni_equil)
to_merge$Semana <- to_merge$Semana %>% sapply(function(i){ as.numeric(i)+2 })
to_merge <- to_merge %>% group_by(Producto_ID,Semana) %>% summarise("previous_demanda_2" = su
m(Demanda_uni_equil)) %>% subset(Semana<9)
```

```
to_merge$Semana <- to_merge$Semana %>% as.character()
df_new <- df_new %>% merge(to_merge, by = c("Producto_ID","Semana"), all.x = T)

# 3 weeks ago
to_merge <- df_new %>% select(Producto_ID,Semana,Demanda_uni_equil)
to_merge$Semana <- to_merge$Semana %>% sapply(function(i){ as.numeric(i)+3 })
to_merge <- to_merge %>% group_by(Producto_ID,Semana) %>% summarise("previous_demanda_3" = su
m(Demanda_uni_equil)) %>% subset(Semana<9)
```

```
to_merge$Semana <- to_merge$Semana %>% as.character()
df_new <- df_new %>% merge(to_merge, by = c("Producto_ID","Semana"), all.x = T)

# 4 weeks ago
to_merge <- df_new %>% select(Producto_ID,Semana,Demanda_uni_equil)
to_merge$Semana <- to_merge$Semana %>% sapply(function(i){ as.numeric(i)+4 })
to_merge <- to_merge %>% group_by(Producto_ID,Semana) %>% summarise("previous_demanda_4" = su
m(Demanda_uni_equil)) %>% subset(Semana<9)
```

```
to_merge$Semana <- to_merge$Semana %>% as.character()
df_new <- df_new %>% merge(to_merge, by = c("Producto_ID","Semana"), all.x = T)

# Add product weight column
df_new['weight'] <- df_new$Venta_hoy /df_new$Venta_uni_hoy

# Replace all NA values by zero in the data frame
df_new <- df_new %>% mutate(previous_demanda_1 = ifelse(is.na(previous_demanda_1),0,previous_
demanda_1)) %>%
           mutate(previous_demanda_2 = ifelse(is.na(previous_demanda_2),0,previous_demanda_
2)) %>%
           mutate(previous_demanda_3 = ifelse(is.na(previous_demanda_3),0,previous_demanda_
3)) %>%
           mutate(previous_demanda_4 = ifelse(is.na(previous_demanda_4),0,previous_demanda_
4)) %>%
           mutate(weight = ifelse(is.na(weight),0,weight))
```

```
# Change data type to Numeric for specific features and normalize them
df_norm <- df_new
numeric.vars <- c("Venta_uni_hoy", "Venta_hoy", "Dev_uni_proxima", "Dev_proxima", "Demanda_un
i_equil",
                  "previous_demanda_1", "previous_demanda_2", "previous_demanda_3", "previous
_demanda_4", "weight")
for (variable in numeric.vars){
  df_norm[[variable]] <- scale(df_norm[[variable]], center=T, scale=T)
}
```

```
# Drop unnecessary columns
df_norm <- df_norm%>%subset(select=-c(Dev_proxima,Dev_uni_proxima,Venta_hoy,Venta_uni_hoy,Nom
breCliente,NombreProducto,X1 ))

# Drop NA values
df_norm <- na.omit(df_norm)

# Change data type of specific columns to factor type
df[feat[1:6]] <- mapply(as.factor,df[feat[1:6]])

# Find best features to be used in the trainning step
summarise(df_norm)
```

```
## data frame with 0 columns and 1 row
```

File failed to load: /extensions/MathZoom.js

```
control <- rfeControl(functions = rfFuncs, method = "cv",
                      verbose = FALSE, returnResamp = "all",
                      number = 5)
results.rfe <- rfe(x = df_norm%>%subset(select=-Demanda_uni_equil),
                   y = as.matrix(df_norm%>%subset(select=Demanda_uni_equil)),
                   sizes = 1:10,
                   rfeControl = control)
```

```
results.rfe
```

```
##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (5 fold)
##
## Resampling performance over subset size:
##
##  Variables   RMSE Rsquared    MAE RMSESD RsquaredSD   MAESD Selected
##          1 0.8258   0.2445 0.2662 0.4603    0.12211 0.01960
##          2 0.8497   0.1897 0.2648 0.4482    0.09581 0.02248
##          3 0.8298   0.2398 0.2481 0.4669    0.12200 0.03412
##          4 0.8354   0.2261 0.2495 0.4591    0.11513 0.02962
##          5 0.8271   0.2491 0.2408 0.4606    0.12145 0.02680
##          6 0.8052   0.2858 0.2209 0.4663    0.13307 0.01940        *
##          7 0.8114   0.2737 0.2230 0.4633    0.12767 0.01954
##          8 0.8178   0.2590 0.2244 0.4578    0.11839 0.01921
##          9 0.8107   0.2780 0.2258 0.4675    0.13152 0.02059
##         10 0.8148   0.2693 0.2275 0.4658    0.12988 0.02000
##         11 0.8144   0.2671 0.2282 0.4616    0.12576 0.02062
##
## The top 5 variables (out of 6):
##    Canal_ID, Ruta_SAK, weight, previous_demanda_3, Producto_ID
```

```
varImp((results.rfe))
```

```
##                      Overall
## Canal_ID           20.560238
## Ruta_SAK           17.950066
## weight              8.656674
## previous_demanda_3  8.581060
## Agencia_ID          7.717503
## previous_demanda_2  7.481760
## previous_demanda_1  7.413589
## Producto_ID         6.872764
## previous_demanda_4  5.146007
```

```
# Select 8 best features
df_sel <- df_norm%>%subset(select=c(Canal_ID,Producto_ID,Ruta_SAK,weight,previous_demanda_3,p
revious_demanda_1,previous_demanda_2,
                                    Demanda_uni_equil))
```

```
str(df_sel)
```

```
## 'data.frame':    10084 obs. of  8 variables:
##  $ Canal_ID          : chr  "1" "1" "1" "1" ...
##  $ Producto_ID       : num  72 72 72 72 72 72 72 72 72 72 ...
##  $ Ruta_SAK          : chr  "1623" "1614" "1661" "5004" ...
##  $ weight            : num [1:10084, 1] -1.06 -1.06 -1.06 -1.06 -1.06 ...
##  $ previous_demanda_3: num [1:10084, 1] -0.411 -0.411 -0.411 -0.411 -0.411 ...
##  $ previous_demanda_1: num [1:10084, 1] -0.615 -0.615 -0.615 -0.615 -0.615 ...
##  $ previous_demanda_2: num [1:10084, 1] -0.493 -0.493 -0.493 -0.322 -0.322 ...
##  $ Demanda_uni_equil : num [1:10084, 1] -0.1379 -0.0919 0.1381 -0.2759 -0.1839 ...
```

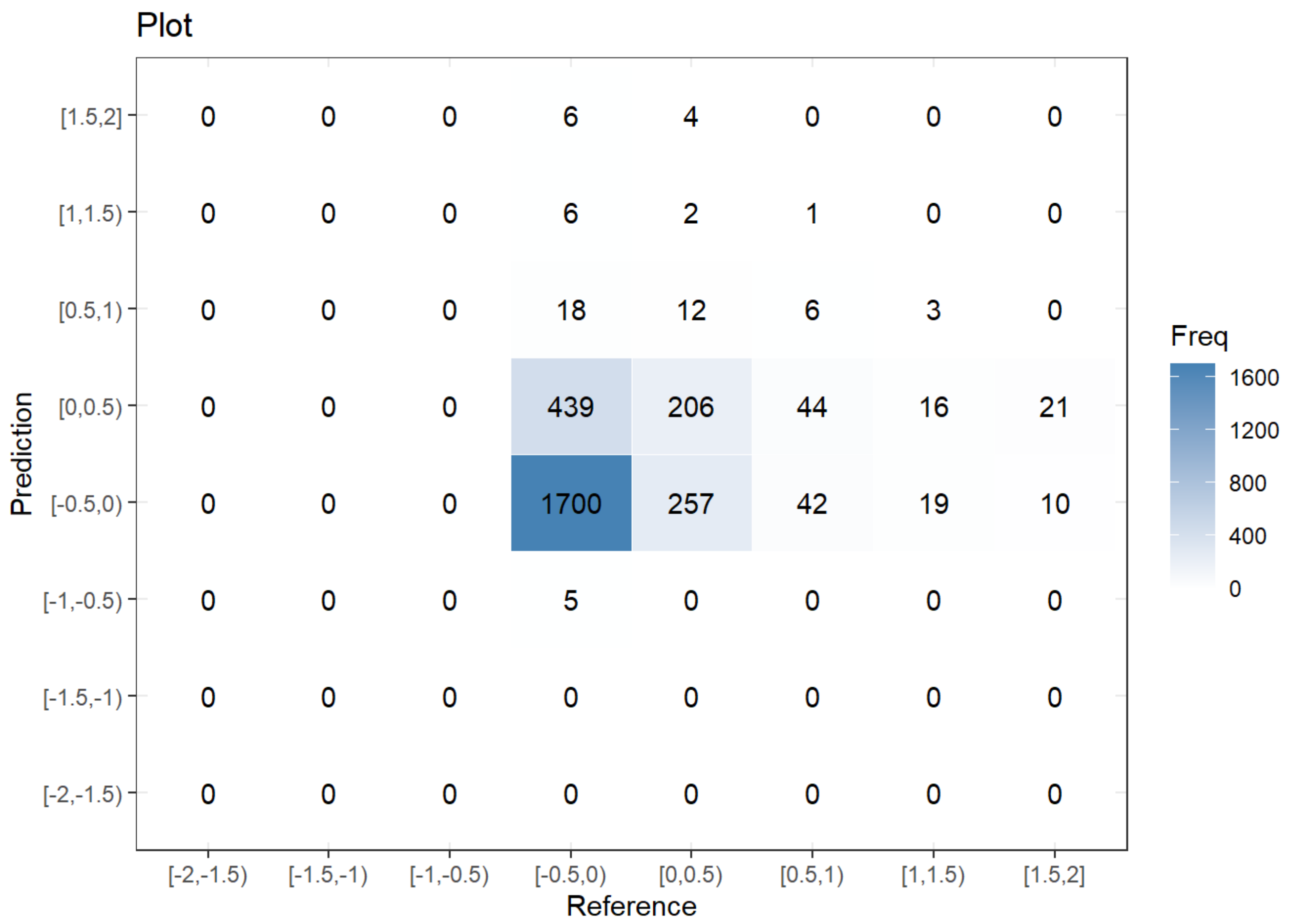
```
# Split data into train and test data
index <- createDataPartition(df_sel$Demanda_uni_equil,p=0.7,list=FALSE)
train_data <- df_sel[c(index),]
test_data <-  df_sel[-c(index),]

# Drop test data which its feature factors are not in train data
features = c(1,2,3)
for(f in features){
  test_data <- test_data %>% subset(test_data[,f] %in% levels(as.factor(train_data[,f])))
}

# Function that display results
show_results <- function(df_final){
  colnames(df_final) <- c("true","pred")
  df_final['resid'] <- df_final$true - df_final$pred

  ggplot(df_final) + geom_point(aes(x=true, y=pred))

  b <- c(2,1.5,1,0.5,0,-0.5,-1,-1.5,-2)
  pred <- discretize(x=df_final$pred, method="fixed", breaks=b)
  true <- discretize(x=df_final$true, method="fixed", breaks=b)
  cm <- confusionMatrix(pred,true)
  ggplot(data = as.data.frame(cm$table), mapping = aes(x = Reference, y = Prediction)) +
    geom_tile(aes(fill = Freq), colour = "white") +
    scale_fill_gradient(low = "white", high = "steelblue") +
    geom_text(aes(label = Freq), colour = "black") +
    ggtitle("Plot")  +
    theme_bw()
}


# Multiple Linear Regression
ctrl <- trainControl(method = "repeatedcv",
                     number = 5,
                     repeats = 5,
                     verboseIter = FALSE)
model_lm <- caret::train(Demanda_uni_equil ~ .,
                         data = train_data,
                         method = "lm",
                         trControl = ctrl)
```

```
final_lm <- data.frame(actual = test_data$Demanda_uni_equil,
                       predict(model_lm, newdata = test_data))
```

```
model_lm
```

```
## Linear Regression
##
## 7061 samples
##    7 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 5 times)
## Summary of sample sizes: 5647, 5649, 5649, 5650, 5649, 5649, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.9015992  0.2004982  0.2795324
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
show_results(final_lm)
```



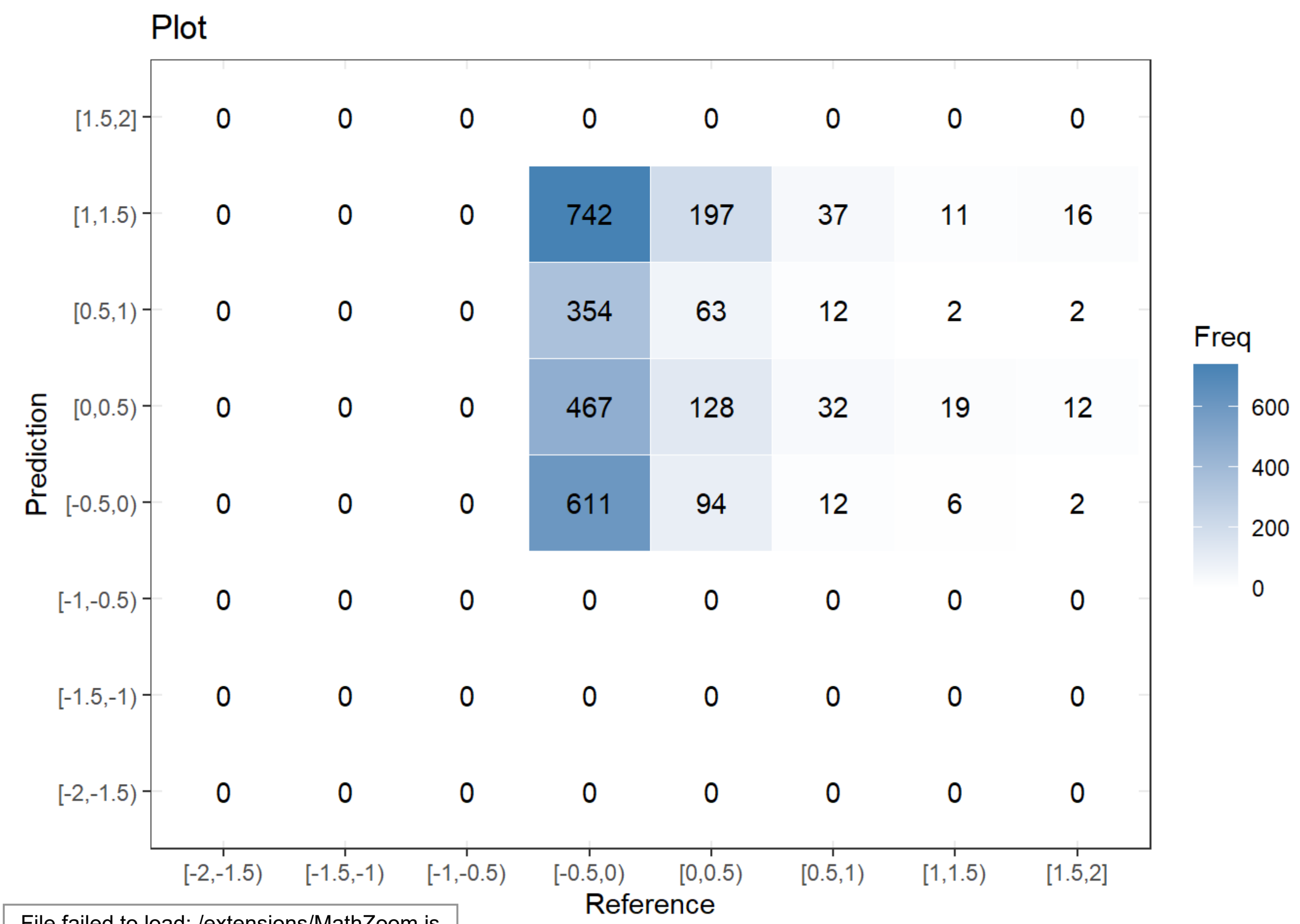

```
# SVM with Linear Kernel
model_svm <- caret::train(Demanda_uni_equil ~ .,
                         data = train_data,
                         method = 'svmLinear3',
                         trControl = ctrl)
```

File failed to load: /extensions/MathZoom.js

```
final_svm <- data.frame(actual = test_data$Demanda_uni_equil,
                        predict(model_svm, newdata = test_data))

model_svm
```

```
## L2 Regularized Support Vector Machine (dual) with Linear Kernel
##
## 7061 samples
##    7 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 5 times)
## Summary of sample sizes: 5649, 5648, 5649, 5649, 5649, 5647, ...
## Resampling results across tuning parameters:
##
##   cost  Loss  RMSE      Rsquared     MAE
##   0.25  L1    2.159118  0.007552525  1.3641108
##   0.25  L2    1.331872  0.003726343  0.6593471
##   0.50  L1    3.342522  0.004919144  2.2334317
##   0.50  L2    2.680436  0.003910356  1.7010524
##   1.00  L1    2.541064  0.005656591  1.6602927
##   1.00  L2    2.271036  0.004030972  1.4068568
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were cost = 0.25 and Loss = L2.
```

```
show_results(final_svm)
```



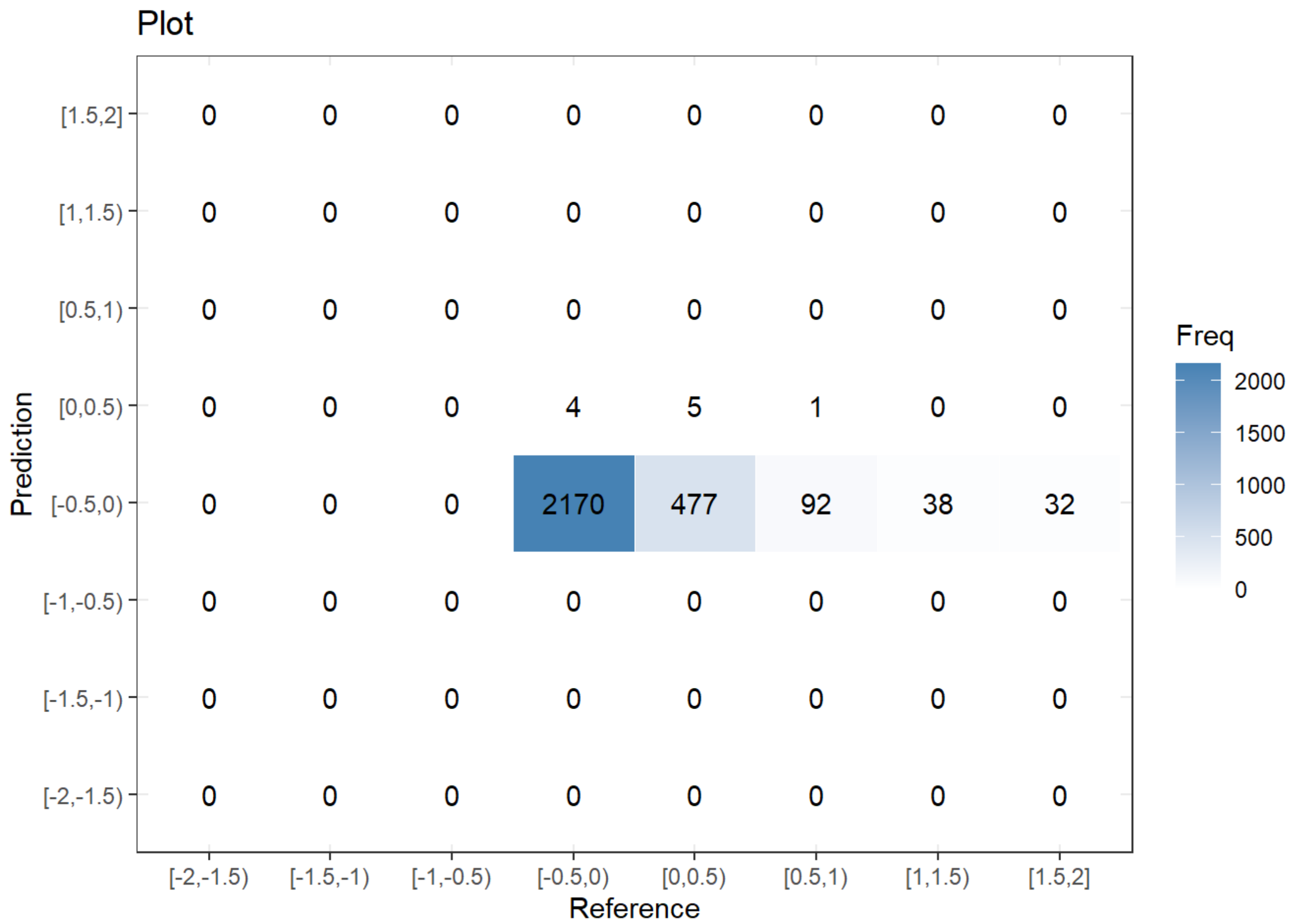
File failed to load: /extensions/MathZoom.js

```
# Random Forest Regression
model_rf <- caret::train(Demanda_uni_equil ~ .,
                         data = train_data,
                         method = 'cforest',
                         trControl = ctrl)
final_rf <- data.frame(actual = test_data$Demanda_uni_equil,
                       predict(model_rf, newdata = test_data))

model_rf
```

```
## Conditional Inference Random Forest
##
## 7061 samples
##    7 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 5 times)
## Summary of sample sizes: 5649, 5649, 5649, 5649, 5648, 5647, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE       Rsquared    MAE
##      2  0.9308272  0.05106515  0.2907452
##     47  0.9280834  0.10691369  0.2884068
##   1120  0.9261611  0.06651952  0.2877821
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 1120.
```

```
show_results(final_rf)
```

File failed to load: /extensions/MathZoom.js



File failed to load: /extensions/MathZoom.js