I. Data Exploration and Visualization

In the data set we can find the following information:

- Store - the store number
- Dept - the department number
- Date - the week of sale
- Weekly_Sales - sales for the given department in the given store
- IsHoliday - whether the week is a special holiday week
- Temperature - average temperature in the region
- Fuel_Price - cost of fuel in the region
- MarkDown1-5 - anonymized data related to promotional markdowns that Walmart is running.
- CPI - the consumer price index
- Unemployment - the unemployment rate
- IsHoliday - whether the week is a special holiday week

In total there are 11 variables in the set. Although it is not a big number of variables, the data set has a lot of entries. Also, we want to determine obvious relations between the key variables and see if they had some kind of correlation:

```
> cor(df)
> corr[is.na(corr)] <- 0
> ggcorrplot(corr)
```

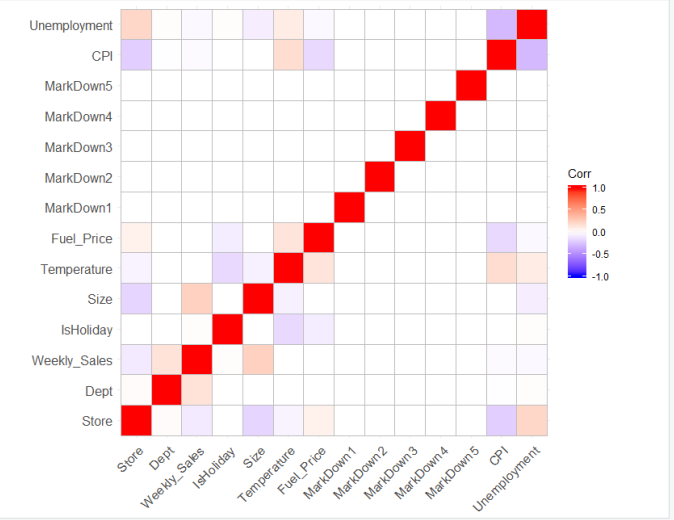


Figure 2 – Heat map of correlation matrix

The heat map (Figure 2) shows what variables have correlation between one another. It might be odd that, as the store increases, we have less weekly sales or the bigger the department number we have more sales. Also, there is a little relationship between holidays, temperatures or fuel prices with weekly sales.

Looking closer to the data, we plotted some variables against the historical weekly sales to better understand the relationship. Here are two examples:

```
> ggplot(df, aes(x=Fuel_Price, =Weekly_Sales))+geom_point(color="blue")
> ggplot(df, aes(x=Unemployment, y=Weekly_Sales))+geom_point(color="black")
```

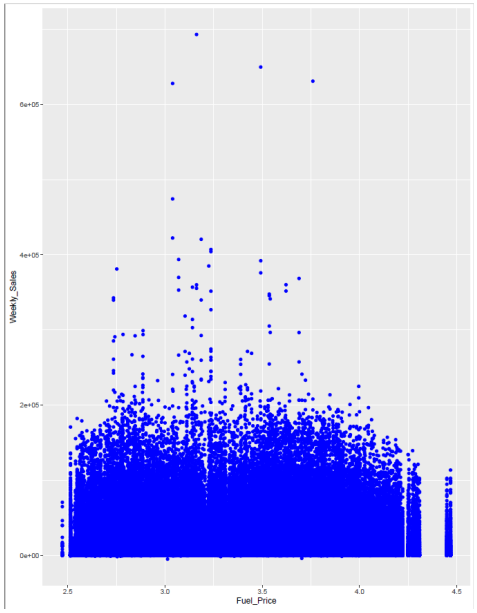


Figure 3 - Evolution of Weekly Sales vs. Fuel Prices

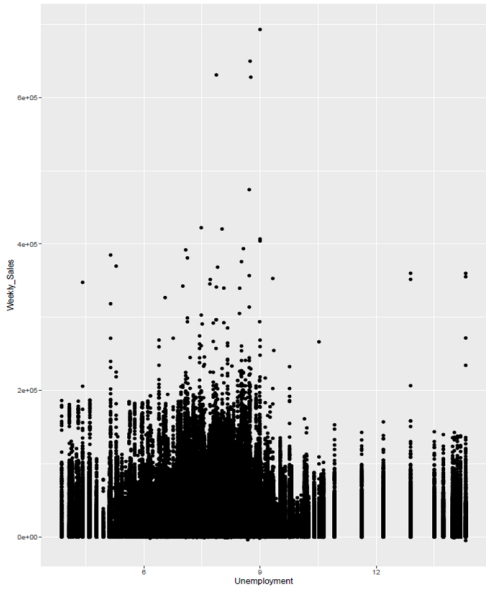


Figure 4 - Evolution of Weekly Sales vs. Unemployment

From this we can how the fuel prices have little to no relation with our weekly sales or that the unemployment rate could have relation at very high rates. But as seen before there is little to none correlation to it. From this analysis we were not looking at the variables have a strong or weak relation to our outcome variable, we just want to see what variables are more suitable to use in our model.

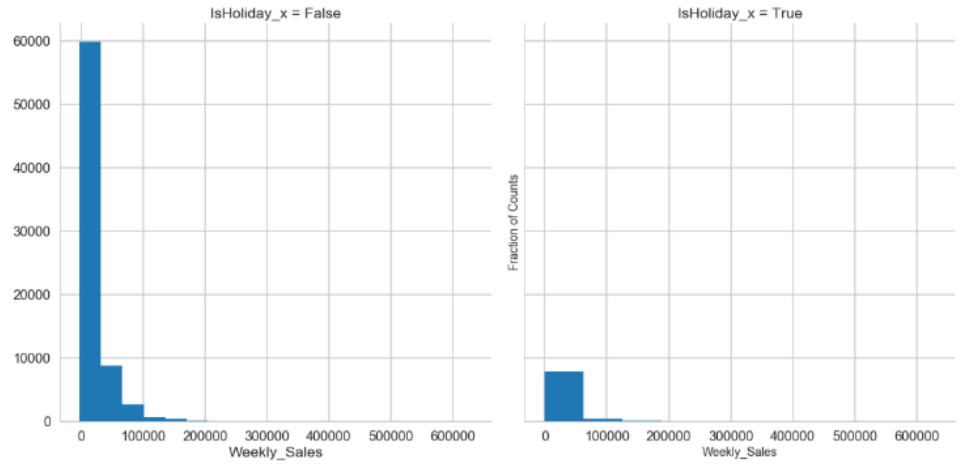


Figure 5 – Histogram of weekly sales to holiday season variable

Another important relation that we foresee is the Weekly Sales in relation to the variable "IsHoliday" (Figure 5). This is has a logical outcome for each date, adopting the value "true" if the date is within any of the nation holiday windows or "false" if it is not. Looking at the histogram, within the holiday dates, sales drop. Although our assumption is that prior to these dates the sales should spike.

We also wanted to see the relationship between stores and departments. We generated a time series plot of each store in dept 1 as shown in figure 6. By doing this, it was evident that the relation within the department in different stores is higher than doing it for each store with all the departments. in different stores is higher than doing it for each store with all the departments. And this actually is logical, since usually a department sales the same product type and so would present similar behavior.

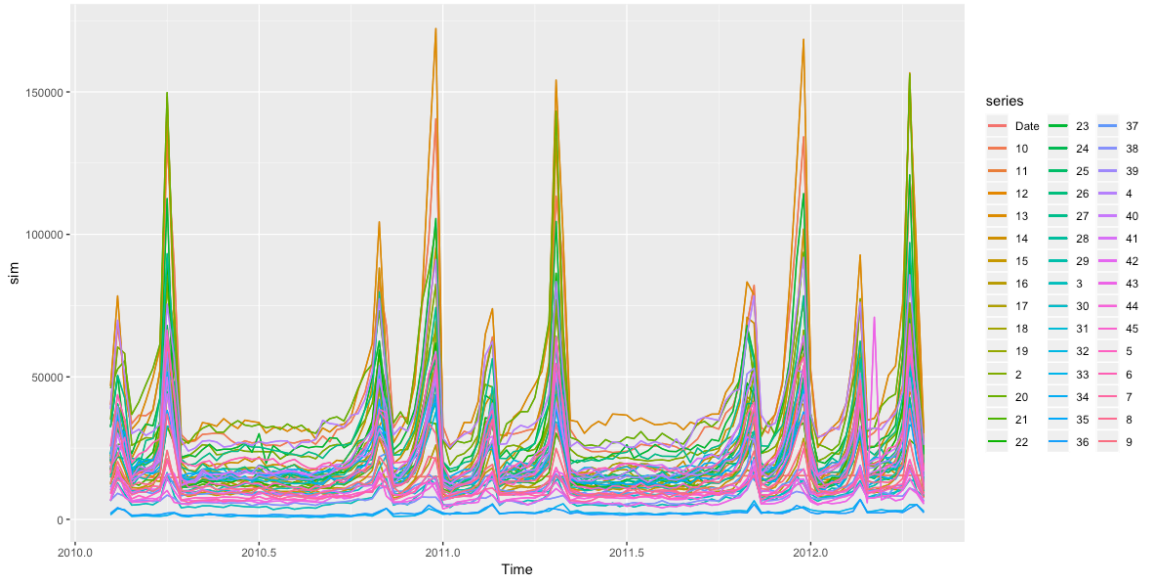


Figure 6 – Time series plot of department 1 of all stores

As we can see, in both graphs (figure 6) , a lot of stores present spikes of weekly sales. This can be related to the season sales of products in the departments according to the products they sale. And confirms what we assumed in the previous paragraph.

While exploring the data, we learned that the number of NAs in the "MarkDown" feature are too huge to simply fill in with 0 and average. Due to the markdown effects are also reflected on the time series data, the performance would be better if we drop the features and focus only on the sales itself since, in this way, we reduce the redundancy.

Once we understood the data and how they interacted, we decided to drop some variables and deal with what for us are the most important ones for our models. This can be seen in the summary of the training set (Figure 8).

```
> summary(train)
     Store            Dept             Date              Weekly_Sales      IsHoliday
 13     : 10474   1      :  6435   Min.   :2010-02-05   Min.   : -4989   Mode :logical
 10     : 10315   10     :  6435   1st Qu.:2010-10-08   1st Qu.:  2080   FALSE:391909
 4      : 10272   13     :  6435   Median :2011-06-17   Median :  7612   TRUE :29661
 1      : 10244   14     :  6435   Mean   :2011-06-18   Mean   : 15981
 2      : 10238   16     :  6435   3rd Qu.:2012-02-24   3rd Qu.: 20206
 24     : 10228   2      :  6435   Max.   :2012-10-26   Max.   :693099
 (Other):359799   (Other):382960
```

Figure 8 – Summary of training set

II. Data Preparation and Preprocessing

The set contains a total of 45 stores, that are distributed over the country. Each store is divided into 99 departments (recall Figures 6 and 7). From the exploration of our data, to make the data set more manageable both in time consumption and resource usage, and because we found little to none correlation, we reduced the set to only the most important variables needed to forecast sales.

While exploring the data, we learned that the number of NAs in the "MarkDown" feature (Figure 9) are too huge (sum is equal to 24040) to simply fill in with 0 and average. Due to the markdown effects are also reflected on the time series data, the performance would be better if we drop the features and focus only on the sales itself since, in this way, we reduce the redundancy.

```
> data <- read.csv("features.csv")
> summary(data$MarkDown1)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  -2781    1578    4744    7032    8923  103185    4158
> summary(data$MarkDown2)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.      NA's
  -265.76     68.88    364.57   3384.18   2153.35 104519.54      5269
> summary(data$MarkDown3)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.      NA's
  -179.26     6.60     36.26   1760.10    163.15 149483.31      4577
> summary(data$MarkDown4)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
    0.22   304.69  1176.42  3292.94  3310.01 67474.85     4726
> summary(data$MarkDown5)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
  -185.2   1440.8   2727.1   4132.2   4832.6 771448.1     4140
```

Figure 9 – Summary of Markdown variables 1 through 5

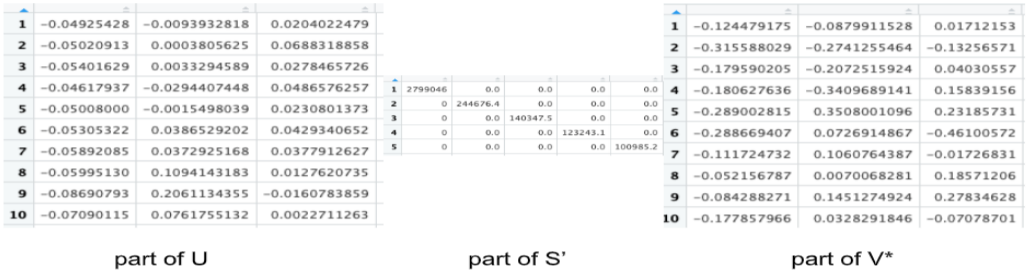
```
> sum(is.na(data))
[1] 24040
```

Then the set was divided in both a training and validation set (see appendix for code). From the summary of either our training or test set the five selected variables are shown (Figure 8). Here we have the historical sales data (in weeks) from different stores. Every store is divided into departments that sale different type of products. We assume that one department sales the same type of products and they have a similar behavior. In the set, we also have the historical data of weekly sales by department and corresponding store

for each date. An important variable is “IsHoliday” which is either True if the date coincides with a holiday in the U.S. or False, if it does not.

The preprocessing part consisted of building a pre-structure of what we wanted to have, that is the date of each sale and the store number for every department. We did this to treat the departments as an individual unit that has a unique product.

For dimensionality reduction we use the single value decomposition function, svd() (refer to the appendix for code). The store sales share a similar pattern within each department, the correlation is high between each stores, in this case, we can use SVD determine important features, and reduce the noise. We perform SVD on the data store structure we casted and reconstruct the train dataset using A' = US’V* (Figure 10).



1	−0.04925428	−0.0093932818	0.0204022479
2	−0.05020913	0.0003805625	0.0688318858
3	−0.05401629	0.0033294589	0.0278465726
4	−0.04617937	−0.0294407448	0.0486576257
5	−0.05008000	−0.0015498039	0.0230801373
6	−0.05305322	0.0386529202	0.0429340652
7	−0.05892085	0.0372925168	0.0377912627
8	−0.05995130	0.1094143183	0.0127620735
9	−0.08690793	0.2061134355	−0.0160783859
10	−0.07090115	0.0761755132	0.0022711263

part of U

1	2799046	0.0	0.0	0.0	0.0
2	0	244676.4	0.0	0.0	0.0
3	0	0.0	140347.5	0.0	0.0
4	0	0.0	0.0	123243.1	0.0
5	0	0.0	0.0	0.0	100985.2

part of S’

1	−0.124479175	−0.0879911528	0.01712153
2	−0.315588029	−0.2741255464	−0.13256571
3	−0.179590205	−0.2072515924	0.04030557
4	−0.180627636	−0.3409689141	0.15839156
5	−0.289002815	0.3508001096	0.23185731
6	−0.288669407	0.0726914867	−0.46100572
7	−0.111724732	0.1060764387	−0.01726831
8	−0.052156787	0.0070068281	0.18571206
9	−0.084288271	0.1451274924	0.27834628
10	−0.177857966	0.0328291846	−0.07078701

part of V*

Figure 10 – SVD parts

With SVD, the features that are shared across different stores are signal, while those that are not are more likely to be noise, which are reduced. Even though we don’t reduce the dimension of our set, this helps reduce the noise by replacing the data with a rank reduced approximation of itself.

IV. Data Mining Techniques and Implementation

After analyzing the data set and determining what our variables were, we continued with the construction of our model. Our output is the prediction of the weekly sales and this will be compared to the historical data.

We used time series for our model since we need to forecast sales in time or for different dates. As seen before, our series have trends or seasonal behavior. We expect spikes in sales and downturns, especially during holiday season (recall figure 6 and 7). That is why we chose the stlf function since it decomposes seasonal time series into STL components and then makes a non-seasonal forecast over the seasonally adjusted data, before adding back the naively extended seasonal component. It is defined as season and trend decomposition function that uses Loess forecasting model. As mentioned, it also does forecasting in one step and, after research, it is a simple approach to what we need. After

the model is created we need to adjust it due to the holiday seasons and how they affect the sales in specific dates. This was done by shifting the sales during the holiday period by a number of days according to the historical data. The forecasted sales are then saved into a table.

To test the model, we do a residual analysis by running the OLS normality test on the residuals of the model:

```
> ols_test_normality(model$residuals)
-----------------------------------------------
       Test             Statistic       pvalue
-----------------------------------------------
Shapiro-Wilk              0.3744         0.0000
Kolmogorov-Smirnov        0.4247         0.0000
Cramer-von Mises          9.7301         0.0000
Anderson-Darling         30.4612         0.0000
-----------------------------------------------
```

Figure 12 – OLS normality test

Since the p-value is less than 0.5, in our case it is zero, we can say that the residual of our model is not normally distributed, fully rejecting the H0 hypothesis. Since a non- normal distribution for the forecast errors is an unreasonable assumption, we did bootstrap aggregating to improve the model.

After running the model with bootstrapping (see added code in Annex) we generated a bar graph to compare the output of both models (Figure 13). The black line is the original data. The color lines are ten different data sets generated by the bootstrapping method. The average of these forecasts gives the bagged forecasts of the original data.

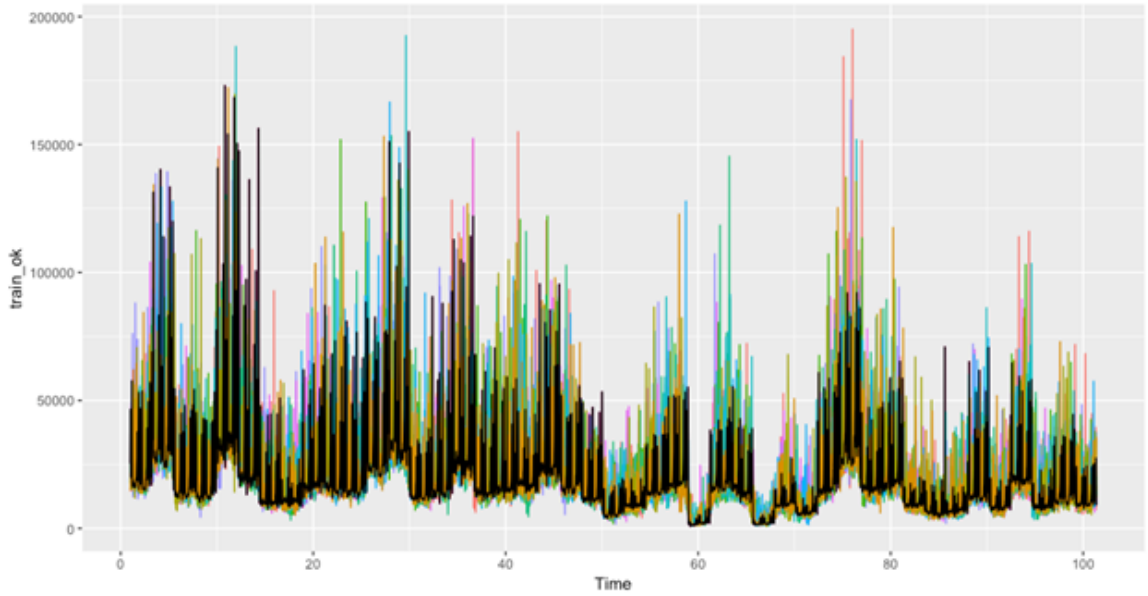


Figure 13 – Comparison between original and bootstrapped data

After completing the forecast model, the final output of our model is a pdf file for every department. In each file, we can see all the stores that have sales for that specific department and the predicted sales in red compared against the original data set. For department 1 and 42 we can see the results in figure 14 and 15.

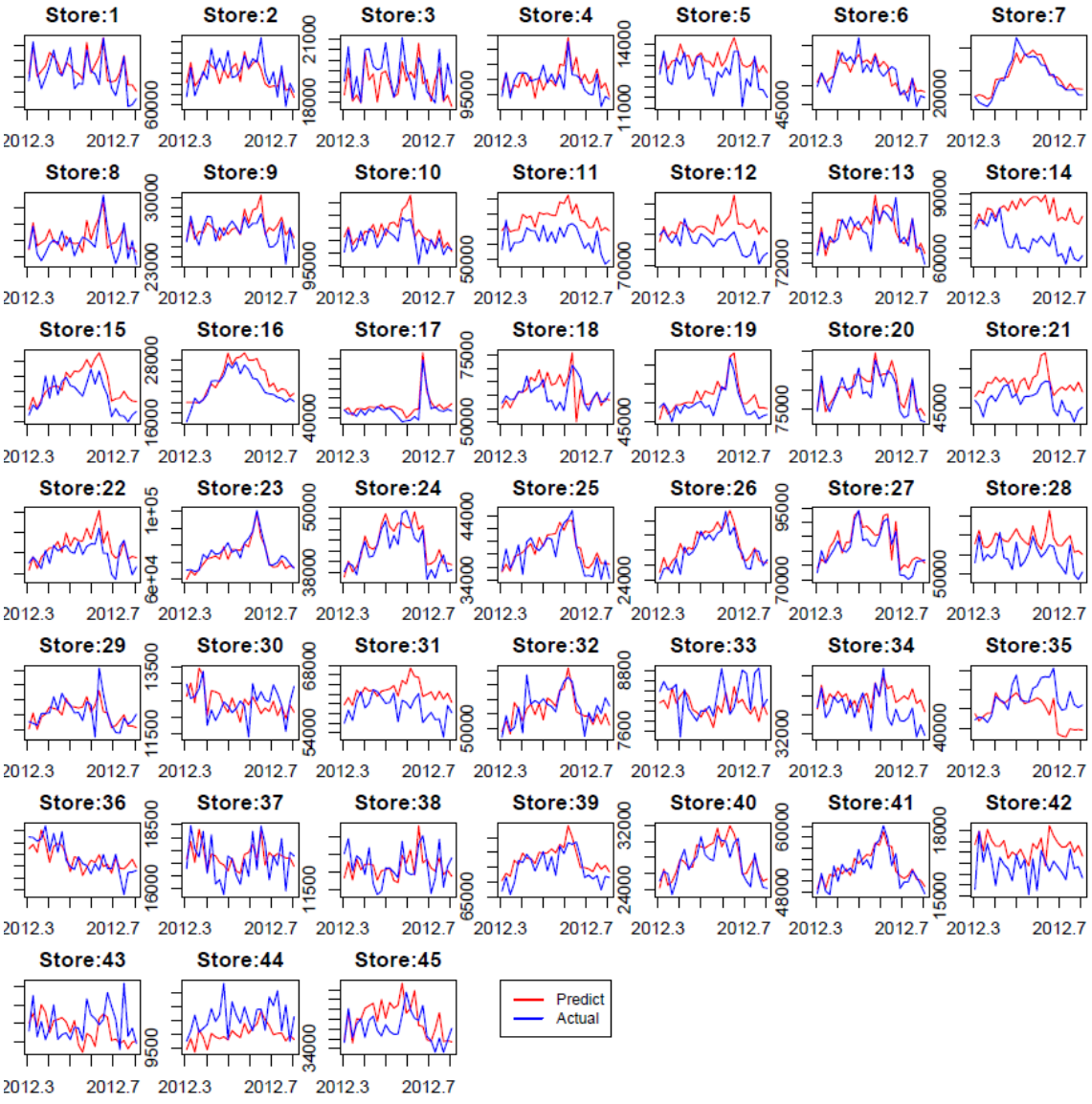


Figure 14 – Output: Department 2 visualization - Actual vs Predicted values after bootstrapping

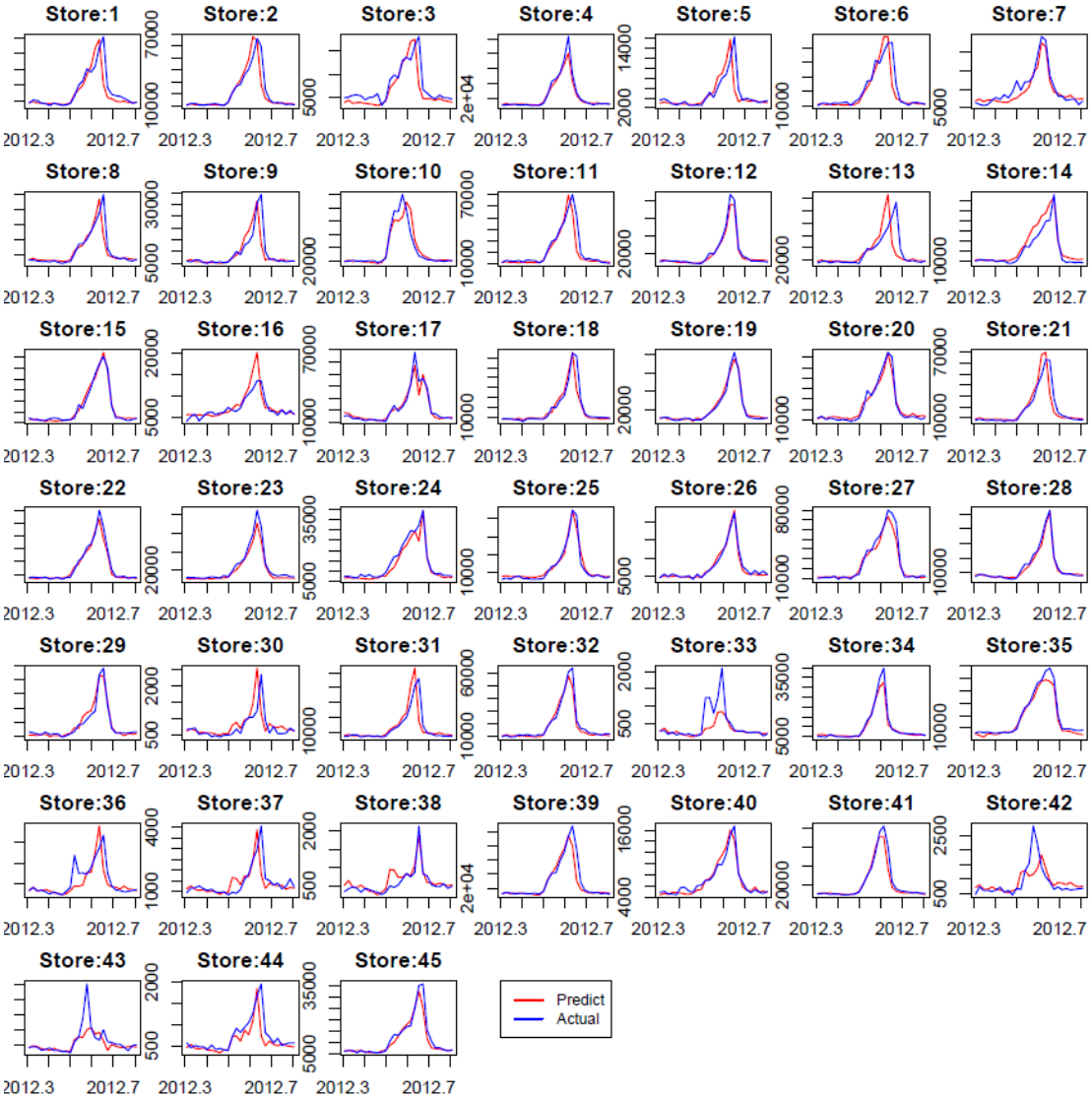


Figure 15 – Output: Department 3 visualization - Actual vs Predicted values after bootstrapping

V. Performance Evaluation

To measure the performance of both models, we obtained the accuracy of the models to compare the RMSE and see which one is better. The results are as following:

- **Original model**

```
> accuracy(result_acc,test.ori_acc)
                ME     RMSE     MAE MPE MAPE      ACF1 Theil's U
Test set 63.62552 3341.994 1610.55 NaN  Inf 0.4078515       NaN
```

- **Bootstrap aggregate model**

```
> accuracy(result_acc,test.ori_acc)
                ME        RMSE       MAE  MPE MAPE      ACF1 Theil's U
Test set -123.5797 2978.699 1448.362 NaN  Inf 0.4139966       NaN
```

Comparing the RMSE, after applying bootstrap, it improved 9%, which shows a significant improvement.

Below we can see on the right the outcome plots using the original model (figure 16 and 18) and on the left plots using the bootstrapping model (figure 17 and 19). By comparing both, we see that with boostrapping the prediction improves significantly. Both graph on the left, follow the actual very good while on the right we see a big gap between the prediction and the actual data.

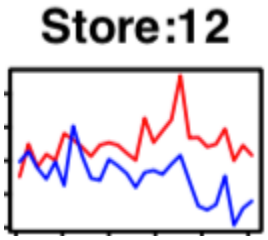


Figure 16 – Plot of predicted (red) vs actual data using original model

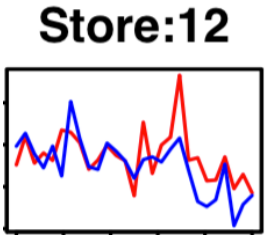


Figure 17 – Plot of predicted (red) vs actual data using bootsrapping model

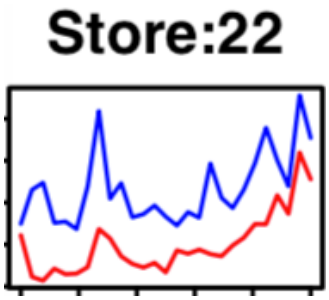
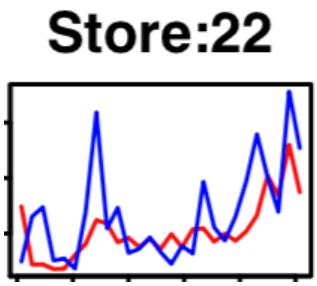


Figure 18 – Plots of predicted (red) vs actual data using original model



Figuree 19 – Plots of predicted (red) vs actual data using bootsrapping model

VI. Discussion and Recommendation

Overall, the model is accurate and has a good performance. It was beneficial to apply data visualization to understand the data set that we had and the kinds of variables. By doing this, we reduced our set by 11 to 5 variables since we saw how some variables were not going to help our model because of little to none correlations.

Our approach was very straightforward since we were dealing with a prediction problem. We used the stlf model since it decomposes the seasonal data and forecasts the sales at the same time. This was an advantage because it simplifies coding and resource usage as well as considering season and trends to predict. The model was improved significantly with bootstrapping allowing an error reduction of 9%. And the results obtained were accurate (see Figure 13 - 18). For anyone using this model, they can make relations with what products sells best in which store.

For recommendations, further investigation and predictions can be done using other functions to model the problem, like the ARIMA model using either fourier series or the seasonal arima, the tslm function that is used to fit linear models to time series including trend and seasonality components or the seasonal naïve function that computes season naïve forecast. All methods can be compared to each other and see which the best fit for this type of problem and how the performance is compared one to the other.

VII. Summary

The goal of the study was to forecast sales for Walmart's store. The problem was modeled with the stlf function that decomposes time series into seasonal, trend and irregular components and forecasts using Loess model. It was later improved by bootstrapping reducing the RMSE significantly and improving the performance of he overall model. The outcome are the predicted sales for each department on every store.