WALMART SALES FORECAST PROJECT

CAPSTONE PROJECT 2

YOHEITA YOSHIMURA | CAPSTONE PROJECT2 | MAY 30, 2024

INTRODUCTION

Walmart, the largest company by revenue in the world, operates over 10,500 stores in 19 countries, offering more than 75 million products. [1] Efficient sales prediction plays a pivotal role in managing Walmart's diverse operations, influencing decisions related to financing, business planning, inventory management, and human resources allocation. The higher the accuracy of the sales prediction model, the greater its contribution to the company's strategic direction.

[1] Source: https://corporate.walmart.com/about

In this project, we utilize data provided by The Makridakis Open Forecasting Center (MOFC) at the University of Nicosia. The dataset comprises three tables. The first table contains sales data for approximately 3000 items across 10 stores located in Texas (TX), Wisconsin (WI), and California (CA), with three stores in TX, three in WI, and four in CA. The second table provides pricing information for each item in the respective stores, while the third table contains calendar data, including major events that occurred in the USA.

With a focus on time series forecasting, our objective is to leverage advanced forecasting algorithms to accurately predict sales trends, providing valuable insights for optimizing inventory management and strategic decision-making at Walmart.

APPROACH

DATA ACQUISITION AND WRANGLING

Evaluating the provided data to understand, clean, and transform it are essential steps to ensure consistency and suitability for analysis and modeling. We will begin by examining the characteristics of the tables and the data they contain. Understanding the data's characteristics will enable us to identify relevant information for our model.

Once we have a clearer understanding of the data, we will check for any missing values or time gaps. Removing missing values ensures that the dataset is complete and reduces the risk of biased conclusions. Identifying and addressing any time gaps enhances the consistency and reliability of the data.

After cleaning the data, we will proceed to transform it to gain deeper insights. This transformation involves removing irrelevant data and addressing missing values, allowing us to provide better quality and consistency of data for further analysis and modeling.

By conducting these steps diligently, we aim to prepare the data effectively for analysis and ensure that it is well-suited for feeding into our forecasting model.

Evaluate each tables

Sales_train_evaluation.csv

```
df_evaluation.head()
```

	id	item_id	dept_id	cat_id	store_id	state_id	d_1	d_2	d_3	d_4	...	d_1932	d_1933	d_1934	d_1935	d_1936	d_1937	d_1938	d_1939	d_1940	d_1941
0	HOBBIES_1_001_CA_1_evaluation	HOBBIES_1_001	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	2	4	0	0	0	0	3	3	0	1
1	HOBBIES_1_002_CA_1_evaluation	HOBBIES_1_002	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	0	1	2	1	1	0	0	0	0	0
2	HOBBIES_1_003_CA_1_evaluation	HOBBIES_1_003	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	1	0	2	0	0	0	2	3	0	1
3	HOBBIES_1_004_CA_1_evaluation	HOBBIES_1_004	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	1	1	0	4	0	1	3	0	2	6
4	HOBBIES_1_005_CA_1_evaluation	HOBBIES_1_005	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	0	0	0	2	1	0	0	2	1	0

5 rows × 1947 columns

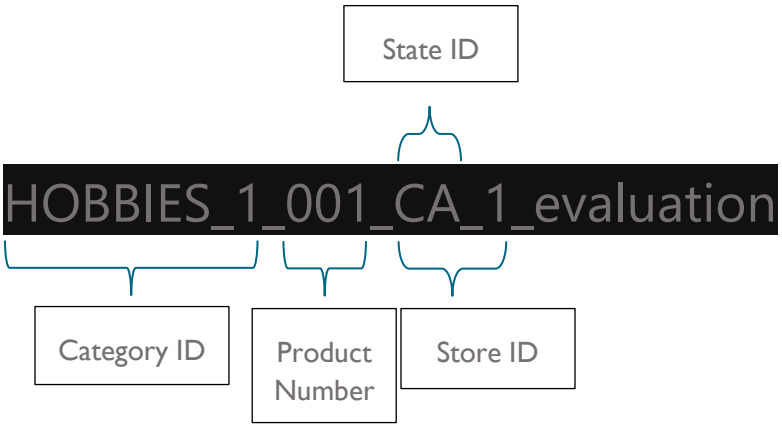
The dataset comprises product information and sales data, detailing the quantity of items sold each day. Product information includes unique identifiers such as item ID, department ID, and category ID, as well as store ID. Each of these identifiers has corresponding columns in the dataset.

Additionally, the dataset contains columns labeled with "d_" followed by a number, representing each date. These columns contain the daily sales data, with the value in each column indicating the number of items sold on that specific day. For instance, "d_1" corresponds to the first date in the dataset, "d_2" to the second date, and so on.

Overall, there are no missing data and the dataset consists of 1941 columns of sales data, each corresponding to a specific date, providing a comprehensive view of item sales over time. Additionally, we observe that there are 30,490 unique products across all stores. Notably, the dataset contains complete product information for all 30,490 products with an equal distribution of 3049 products' information available for each store.

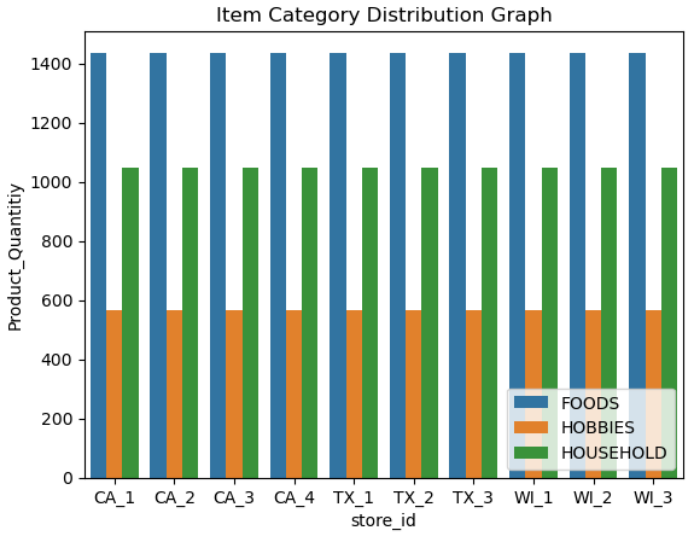
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30490 entries, 0 to 30489
Columns: 1947 entries, id to d_1941
dtypes: int64(1941), object(6)
memory usage: 452.9+ MB
```

The “id” has below format:

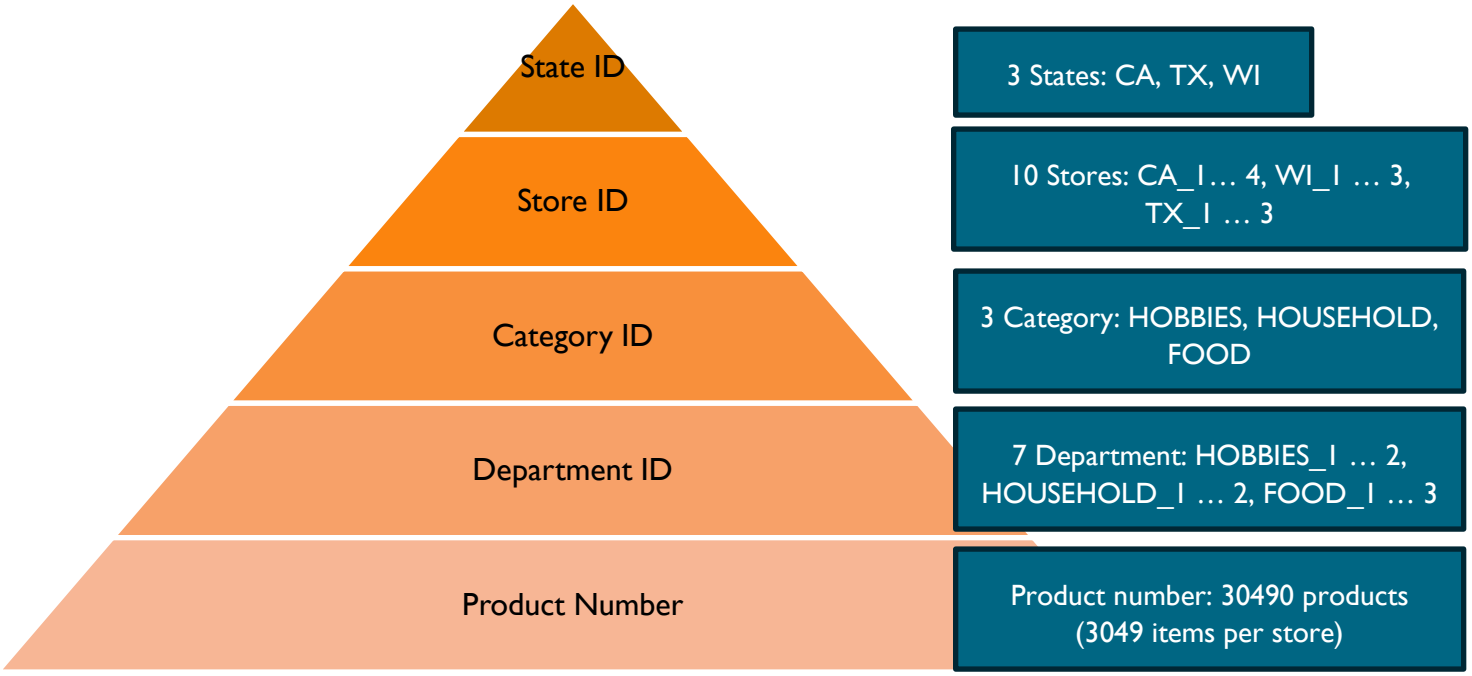


Products in each store.

store_id	Product_number
CA_1	3049
CA_2	3049
CA_3	3049
CA_4	3049
TX_1	3049
TX_2	3049
TX_3	3049
WI_1	3049
WI_2	3049
WI_3	3049



There are 4 stores in CA and 3 stores each in TX and WI, with 3049 items sold in each store (Fig. 1). Furthermore, the distribution of item categories is consistent across all stores, as depicted in Fig. 2.



sell_prices.csv evaluation

The table contains data of the price of each product in each store by a weekly bases.

	store_id	item_id	wm_yr_wk	sell_price
0	CA_1	HOBBIES_1_001	11325	9.58
1	CA_1	HOBBIES_1_001	11326	9.58
2	CA_1	HOBBIES_1_001	11327	8.26
3	CA_1	HOBBIES_1_001	11328	8.26
4	CA_1	HOBBIES_1_001	11329	8.26

Here is the data type for the each column data

```
df_sell_price_original.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6841121 entries, 0 to 6841120
Data columns (total 4 columns):
 #   Column      Dtype
---  ------      -----
 0   store_id    object
 1   item_id     object
 2   wm_yr_wk    int64
 3   sell_price  float64
dtypes: float64(1), int64(1), object(2)
memory usage: 208.8+ MB
```

There is no missing values in this table.

```
store_id      0
item_id       0
wm_yr_wk      0
sell_price    0
dtype: int64
```

Calender.csv evaluation

```
df_calender.head()
```

	date	wm_yr_wk	weekday	wday	month	year	d	event_name_1	event_type_1	event_name_2	event_type_2	snap_CA	snap_TX	snap_WI
0	2011-01-29	11101	Saturday	1	1	2011	d_1	NaN	NaN	NaN	NaN	0	0	0
1	2011-01-30	11101	Sunday	2	1	2011	d_2	NaN	NaN	NaN	NaN	0	0	0
2	2011-01-31	11101	Monday	3	1	2011	d_3	NaN	NaN	NaN	NaN	0	0	0
3	2011-02-01	11101	Tuesday	4	2	2011	d_4	NaN	NaN	NaN	NaN	1	1	0
4	2011-02-02	11101	Wednesday	5	2	2011	d_5	NaN	NaN	NaN	NaN	1	0	1

```
df_calender.tail()
```

	date	wm_yr_wk	weekday	wday	month	year	d	event_name_1	event_type_1	event_name_2	event_type_2	snap_CA	snap_TX	snap_WI
1964	2016-06-15	11620	Wednesday	5	6	2016	d_1965	NaN	NaN	NaN	NaN	0	1	1
1965	2016-06-16	11620	Thursday	6	6	2016	d_1966	NaN	NaN	NaN	NaN	0	0	0
1966	2016-06-17	11620	Friday	7	6	2016	d_1967	NaN	NaN	NaN	NaN	0	0	0
1967	2016-06-18	11621	Saturday	1	6	2016	d_1968	NaN	NaN	NaN	NaN	0	0	0
1968	2016-06-19	11621	Sunday	2	6	2016	d_1969	NBAFinalsEnd	Sporting	Father's day	Cultural	0	0	0

This table contains essential date-related information, including events and the availability of the Supplemental Nutrition Assistance Program (SNAP), across the three related states (CA, WI, TX). With this calendar data, we can establish connections between the "Sales_train_evaluation.csv" and "wm_yr_wk" datasets to link with "sell_prices.csv".

What is SNA?

The Supplemental Nutrition Assistance Program (SNAP) is the largest federal nutrition assistance program, providing benefits to eligible low-income individuals and families through an Electronic Benefits Transfer card. This card functions like a debit card, allowing recipients to purchase eligible food items at authorized retail food stores.[2]

Source: [2] https://www.benefits.gov/benefit/361

The table consists of 13 columns and 1969 rows of data. Notably, the event-related columns contain many null values, indicating dates without events.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1969 entries, 0 to 1968
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   date          1969 non-null   object
 1   wm_yr_wk      1969 non-null   int64 
 2   weekday       1969 non-null   object
 3   wday          1969 non-null   int64 
 4   month         1969 non-null   int64 
 5   year          1969 non-null   int64 
 6   d             1969 non-null   object
 7   event_name_1  162 non-null    object
 8   event_type_1  162 non-null    object
 9   event_name_2  5 non-null      object
 10  event_type_2  5 non-null      object
 11  snap_CA       1969 non-null   int64 
 12  snap_TX       1969 non-null   int64 
 13  snap_WI       1969 non-null   int64 
dtypes: int64(7), object(7)
memory usage: 215.5+ KB
```

```
date               0
wm_yr_wk           0
weekday            0
wday               0
month              0
year               0
d                  0
event_name_1    1807
event_type_1    1807
event_name_2    1964
event_type_2    1964
snap_CA            0
snap_TX            0
snap_WI            0
dtype: int64
```

The event data includes types and descriptions of events, denoted as event_1 and event_2 to represent overlapping events. Null values within this data will be replaced with the object "no_event." The list of event types and descriptions is provided below:

event_type_1	Number of Event
Cultural	37
National	52
No_Event	1807
Religious	55
Sporting	18

event_type_2	Number of Event
Cultural	4
No_Event	1964
Religious	1

event_name_1	Number of Event Occured
Father's day	4
Chanukah End	5
Thanksgiving	5
OrthodoxEaster	5
OrthodoxChristmas	5
NewYear	5
MartinLutherKingDay	5
LaborDay	5
IndependenceDay	5
VeteransDay	5
EidAlAdha	5
Christmas	5
Cinco De Mayo	5
Halloween	5
Easter	5
ColumbusDay	5
Eid al-Fitr	5
NBAFinalsEnd	6
NBAFinalsStart	6
MemorialDay	6
LentWeek2	6
Pesach End	6
PresidentsDay	6
Purim End	6
Ramadan starts	6
StPatricksDay	6
SuperBowl	6
LentStart	6
ValentinesDay	6
Mother's day	6
No_Event	1807

event_name_2	Number of Event Occured
Cinco De Mayo	1
Easter	1
OrthodoxEaster	1
Father's day	2
No_Event	1964

Transformation of the data

To enhance the calendar data and introduce additional exogenous variables, we will augment the existing table with the following additions:

"is_weekend": A binary indicator denoting whether the date falls on a weekend.

"Month_day": The day of the month.

"week_number": The week number of the year.

"day": The sequential day count throughout the dataset.

The left side represents the original table, while the right side illustrates the augmented table with the added variables.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1969 entries, 0 to 1968
Data columns (total 15 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   date          1969 non-null   datetime64[ns]
 1   wm_yr_wk      1969 non-null   int64
 2   weekday       1969 non-null   object
 3   wday          1969 non-null   int64
 4   month         1969 non-null   int64
 5   year          1969 non-null   int64
 6   d             1969 non-null   object
 7   event_name_1  1969 non-null   object
 8   event_type_1  1969 non-null   object
 9   event_name_2  1969 non-null   object
 10  event_type_2  1969 non-null   object
 11  snap_CA       1969 non-null   int64
 12  snap_TX       1969 non-null   int64
 13  snap_WI       1969 non-null   int64
 14  day           1969 non-null   object
dtypes: datetime64[ns](1), int64(7), object(7)
memory usage: 230.9+ KB
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1969 entries, 0 to 1968
Data columns (total 19 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   date            1969 non-null   datetime64[ns]
 1   wm_yr_wk        1969 non-null   int64
 2   weekday         1969 non-null   object
 3   wday            1969 non-null   int64
 4   month           1969 non-null   int64
 5   year            1969 non-null   int64
 6   d               1969 non-null   object
 7   event_name_1    1969 non-null   object
 8   event_type_1    1969 non-null   object
 9   event_name_2    1969 non-null   object
 10  event_type_2    1969 non-null   object
 11  snap_CA         1969 non-null   int64
 12  snap_TX         1969 non-null   int64
 13  snap_WI         1969 non-null   int64
 14  day             1969 non-null   object
 15  is_weekend      1969 non-null   int64
 16  month_day       1969 non-null   int32
 17  week_number     1969 non-null   UInt32
 18  events_per_day  1969 non-null   int64
dtypes: UInt32(1), datetime64[ns](1), int32(1), int64(9), object(7)
memory usage: 278.9+ KB
```

Upon merging all three tables into one, we observed the addition of NaN values. Subsequent investigation revealed that these missing values correspond to dates when products were not available on the shelf. To ensure the data does not bias the model by falsely indicating that products were not picked up due to customer choice, we will remove these added NaN values. Additionally, to represent the total sales of each product per day accurately, we will introduce a new variable, "daily_sell." This variable will be calculated by multiplying the product's price by the quantity sold on that day.

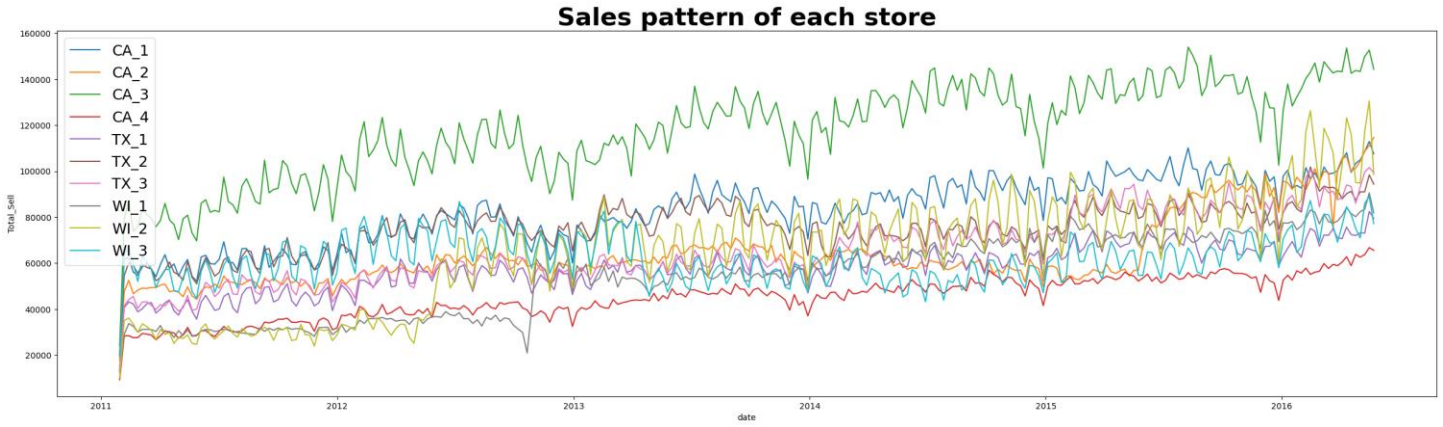
```
<class 'pandas.core.frame.DataFrame'>
Index: 46881677 entries, 7 to 59181089
Data columns (total 28 columns):
 #   Column          Dtype
---  ------          -----
 0   id              object
 1   item_id         object
 2   dept_id         object
 3   cat_id          object
 4   store_id        object
 5   state_id        object
 6   d               object
 7   sell_quantity   int32
 8   date            datetime64[ns]
 9   wm_yr_wk        int32
 10  weekday         object
 11  wday            int8
 12  month           int8
 13  year            int16
 14  event_name_1    object
 15  event_type_1    object
 16  event_name_2    object
 17  event_type_2    object
 18  snap_CA         int8
 19  snap_TX         int8
 20  snap_WI         int8
 21  day             int32
 22  is_weekend      int8
 23  month_day       int32
 24  week_number     int32
 25  events_per_day  int8
 26  sell_price      float64
 27  daily_sell      float64
dtypes: datetime64[ns](1), float64(2), int16(1), int32(5), int8(7), object(12)
memory usage: 6.9+ GB
```

EXPLORATORY DATA ANALYSIS (EDA)

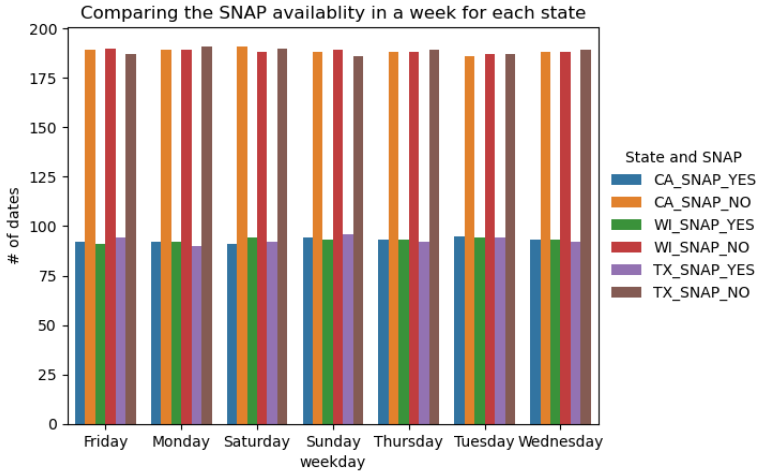
Now, leveraging the data obtained from previous steps, we embark on exploratory data analysis (EDA) to gain deeper insights into the characteristics and behavior of the dataset. Through EDA, we will delve into the patterns, trends, and relationships within the data, harnessing the power of visualization techniques to unveil hidden insights.

Here, we present the sales figures for each store within each state. Remarkably, a consistent trend in sales is observed across all states.

Upon comparing the sales figures across all stores, "CA_1" emerges with the highest sales among the 10 stores, while "CA_4" records the lowest sales. Notably, "TX_1" and "WI_3" stores exhibit similar sales figures, suggesting a narrower range compared to California, indicative of potentially higher economic disparity within the state.



The table below illustrates the availability of SNAP across each state, allowing us to discern any variations in the data. Remarkably, the number of SNAP dates appears to be nearly equal across all three states.



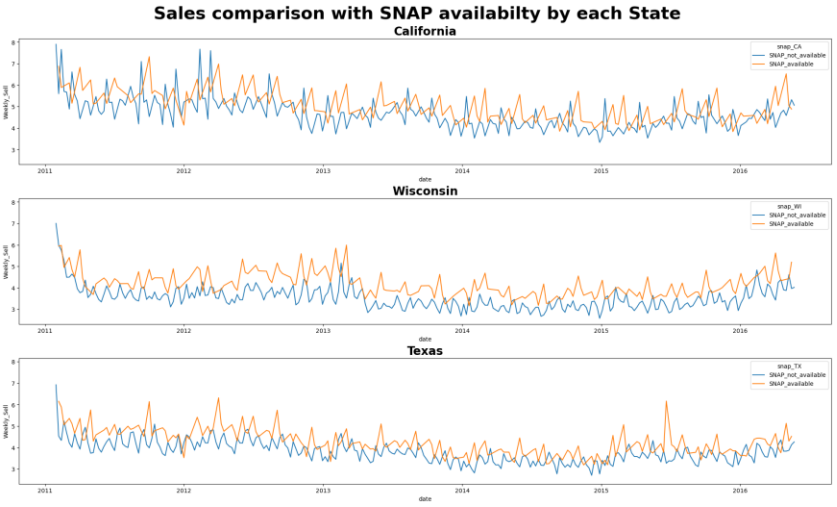
Comparing the SNAP availability sales trends across each state reveals that Wisconsin exhibits the largest sales gap, indicating the highest need for the SNAP program in the state. Given that SNAP benefits are exclusively applicable to food groceries [3], it follows that the sales difference is most pronounced in the food category across the state. Conversely, product categories other than food display smaller sales differences.

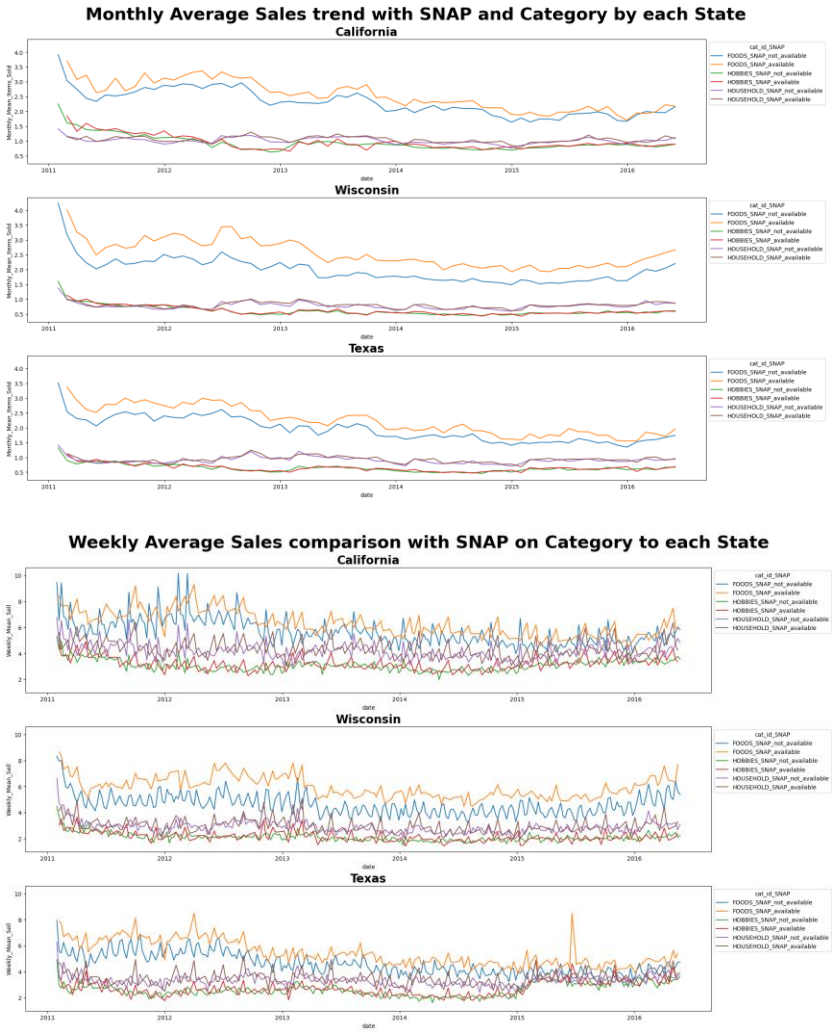
[3] Source: [What Can SNAP Buy? | Food and Nutrition Service (usda.gov)]

However, when analyzing sales trends by averaging across categories, a similar pattern emerges, indicating that individuals relying on SNAP benefits also purchase other groceries at Walmart. This suggests a consistent shopping pattern among SNAP recipients, regardless of product category.

The impact of SNAP availability on sales is evident in the presented table, with the significant difference in the food category affirming SNAP's focus on grocery items. Furthermore, this disparity in SNAP availability reflects the economic status of each state, with Wisconsin exhibiting the greatest difference, implying a lower economic standing compared to the other states.

Moreover, the observation that customers tend to purchase non-food items alongside food groceries when SNAP benefits are available highlights the influence of SNAP availability on overall sales patterns. Consequently, the sales difference attributed to SNAP availability serves as an exogenous variable in the dataset.

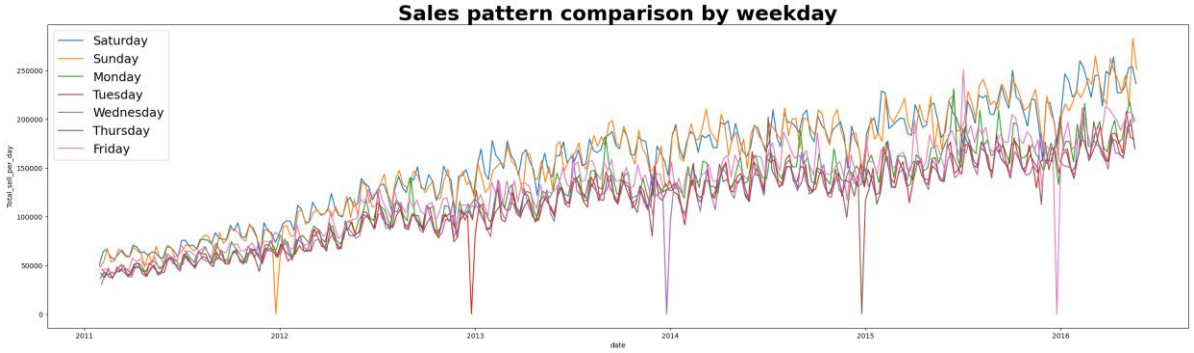


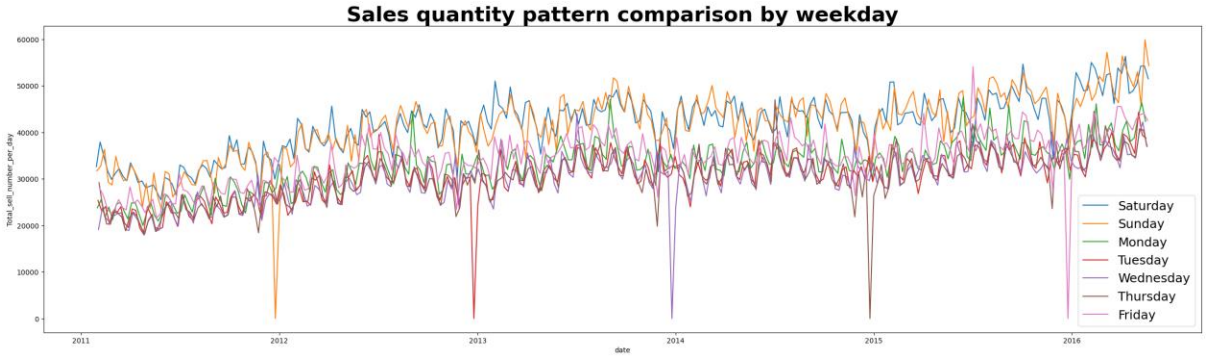


Presented here is a time series graph depicting sales and sales quantity over time. Notably, there is a consistent sales difference observed on a daily basis, with Saturday and Sunday consistently registering the highest sales figures in both quantity and revenue.

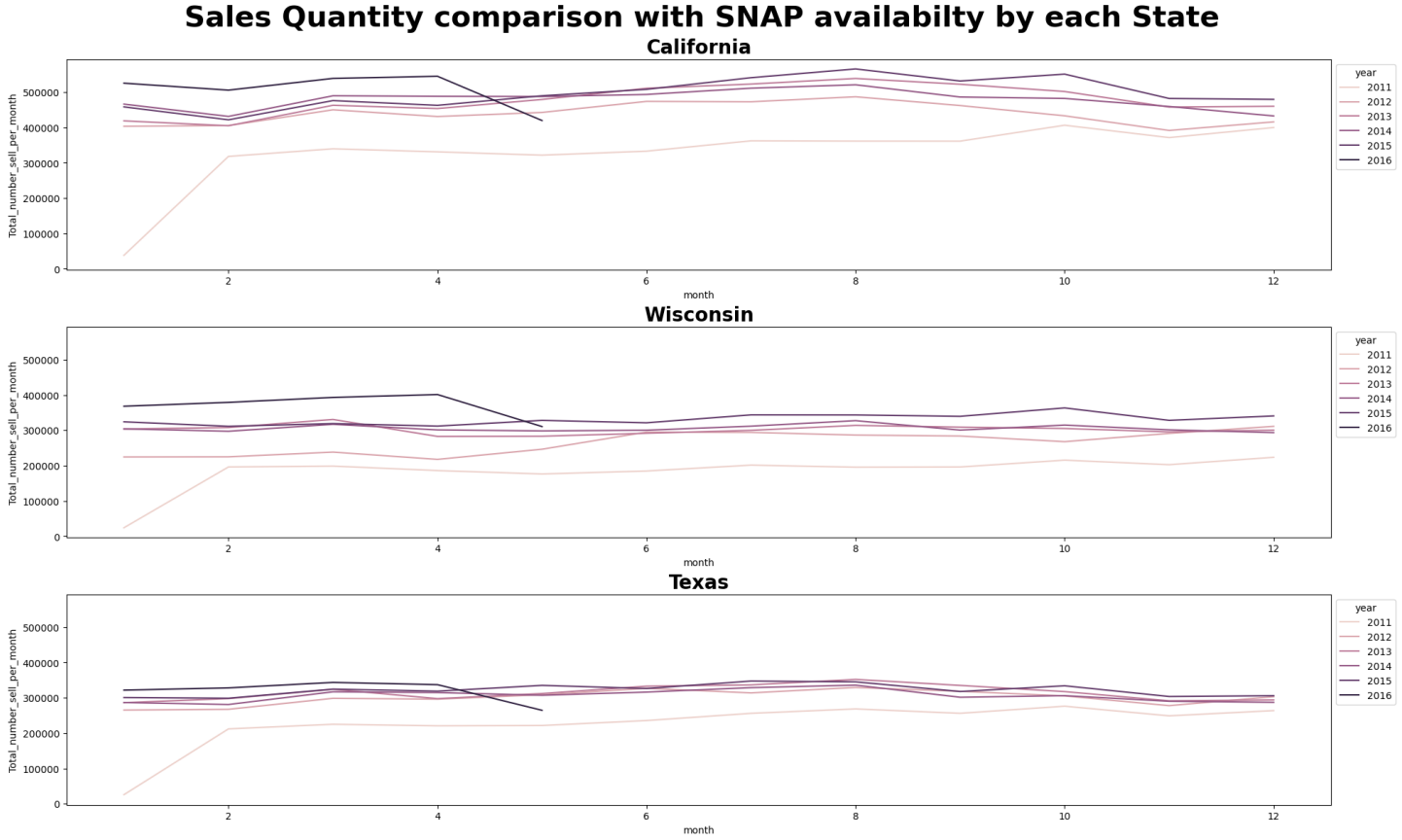
An intriguing observation is the gradual increase in sales over the years, particularly evident from 2011 onwards. However, the quantity of sales has not exhibited a corresponding increase during this period. This discrepancy may be attributed to a rise in the average item price over the past five years, influencing the overall sales revenue while maintaining a relatively stable quantity of items sold.

Furthermore, the sales difference observed on weekdays underscores the role of weekday data as an exogenous variable, contributing to variations in overall sales trends.





Graphing the monthly sales of each state reveals a consistent trend across the year, indicating similar sales patterns regardless of the month.



During the exploratory data analysis, we observed numerous zero values with intermittent stretches of continuous zeros in the sales data. These zeros could result from various factors such as human error, environmental conditions, software issues, or stock shortages.

To manage these zero values, we introduced a column to detect the length of gaps (consecutive zeros) with thresholds of 7 days and 70 days. This helps identify significant gaps and ensures the training data includes periods where the gap length exceeds 70 days, reducing noise in the model.

For the regression model, we added several features:

Lag Features: Lag1 to Lag49 to incorporate historical sales data.

Rolling Statistics: Rolling median and standard deviation for periods ranging from 7 to 56 days, capturing the central tendency and variability in sales.

These features aim to enhance the predictive power of the model by providing comprehensive historical context and statistical insights.

BASELINE MODELING

For this model, we focus on a single product ID, 'FOODS_3_083_WI_3_evaluation', utilizing its daily sales data to analyze the sales trend and generate parameter values for ARIMA (Autoregressive Integrated Moving Average). ARIMA is a popular time series forecasting model that combines autoregression, differencing, and moving components to capture trends and seasonality in complex data patterns. However, selecting appropriate values for parameters (p, d, q) requires careful analysis and model tuning.

We begin by decompressing the data to observe its seasonal and trend relationships. Subsequently, we check for stationarity using the Dickey-Fuller test on the time series data. Once stationarity is confirmed, we analyze the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) graphs to determine the values of p, d, and q for ARIMA. We then observe the predicted and actual results for the last 30 days of the data to assess accuracy.

ACF (Autocorrelation Function) represents the correlation of the time series with itself at different lags, aiding in identifying the order of the autoregressive (AR) model (p).

PACF (Partial Autocorrelation Function) represents the correlation of the time series with itself at different lags, excluding the effect of previous lags, and helps identify the order of the moving average (MA) model (q).

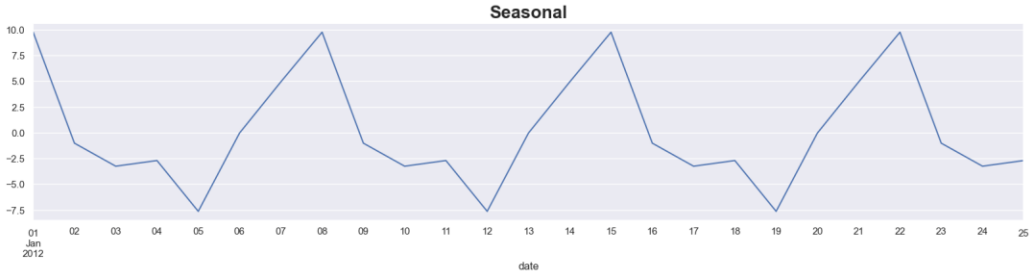
For testing purposes, we initially train the model without slicing the evaluation data containing long series of continuous zeros. Subsequently, we evaluate the model again after slicing the evaluation data based on the length of continuous zeros.

Seasonal and Trend Decompression

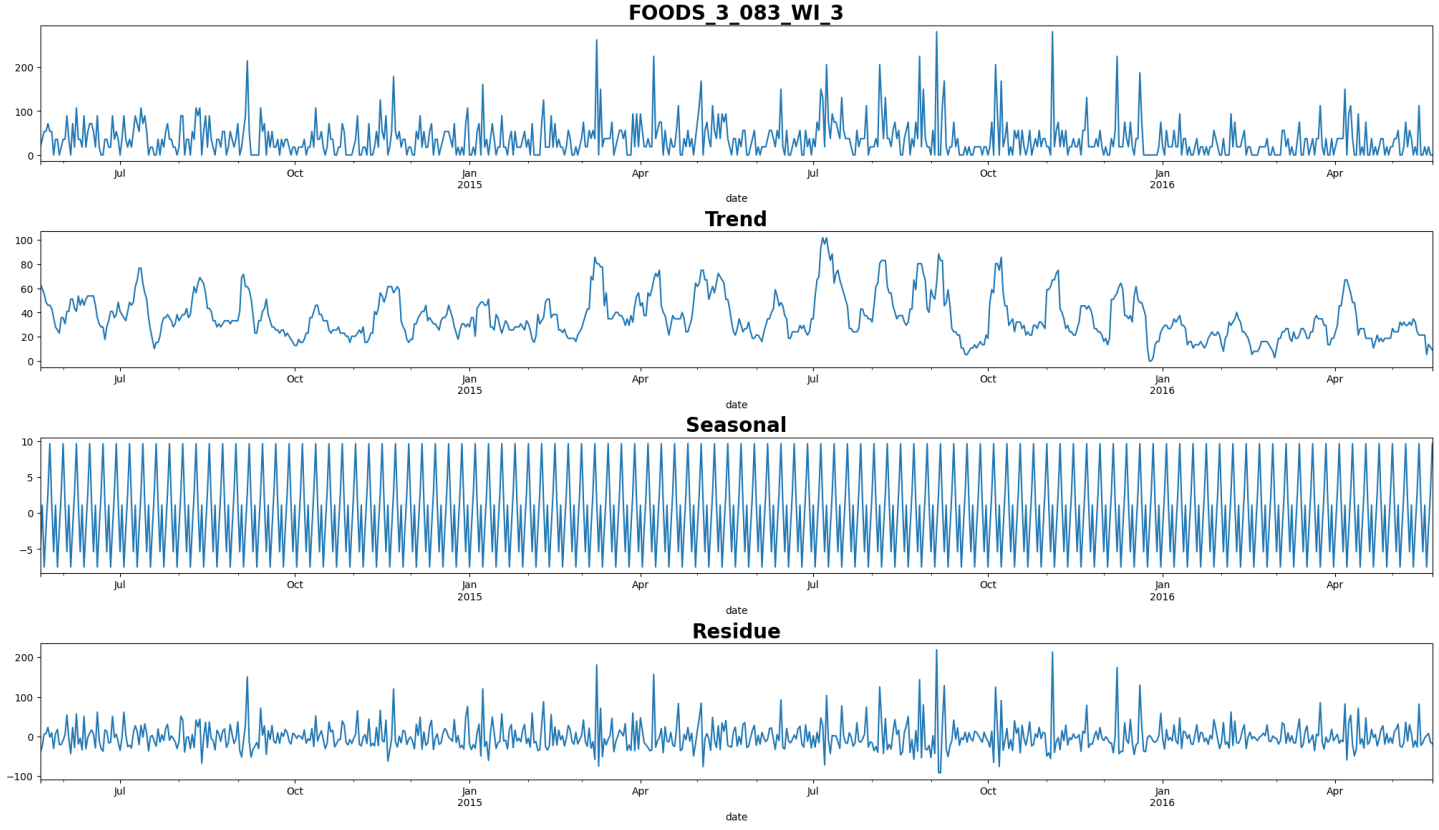
The below graph illustrates the decompressed seasonal trend for both the original and sliced data. With the original data, discerning the trend proves challenging as the plot lacks a clear upward or downward slope. However, continuous seasonal data suggests a recurring pattern within the dataset. The trend graph exhibits a slight decreasing trend with considerable fluctuation. Most data points are spread near zero to plus/minus 100, indicating significant noise in the data that cannot be captured by trend or seasonal patterns.

In contrast, the sliced data reveals a discernible trend where sales increase around the beginning to the first quarter of the month. However, the peak is not clearly defined, making trend determination difficult. The seasonal pattern mirrors the previous plot, with line fluctuations occurring every seven days. The residue plot exhibits peaks at the beginning of April, the beginning of September, and the middle of December, hinting at potential seasonal patterns on a yearly basis.

For the next step of evaluation, we will assess the stationarity of the data and plot the ACF and PACF to determine the values of p, d, and q for the ARIMA model.



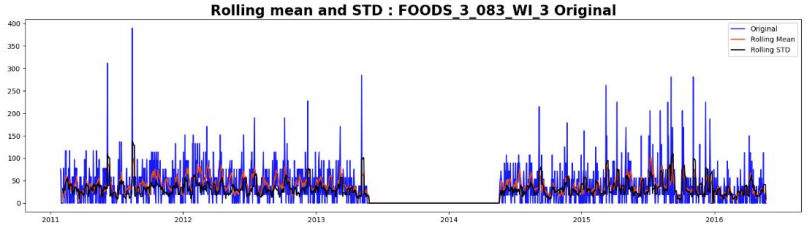
Seasonal and Trend Decompression of FOODS_3_083_WI_3 with Trimed data



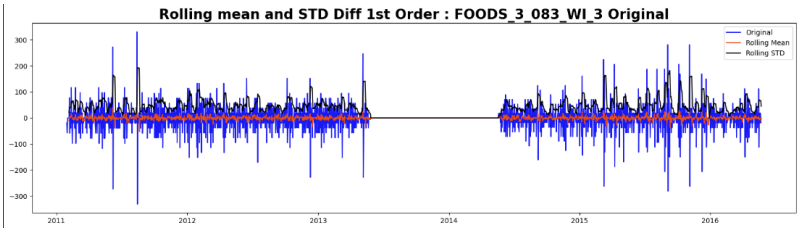
Stationarity Test and ACF & PACF graph

For the original data, we computed and plotted the rolling mean and standard deviation of the sales trend. The p-value obtained from the Dickey-Fuller test is less than 5% (less than 1%), indicating stationarity. However, upon plotting the ACF and PACF, we observed that the ACF does not decay as the lag increases, suggesting a trend in the dataset. To address this, differencing was applied to detrend the data, resulting in both ACF and PACF exhibiting a wave pattern.

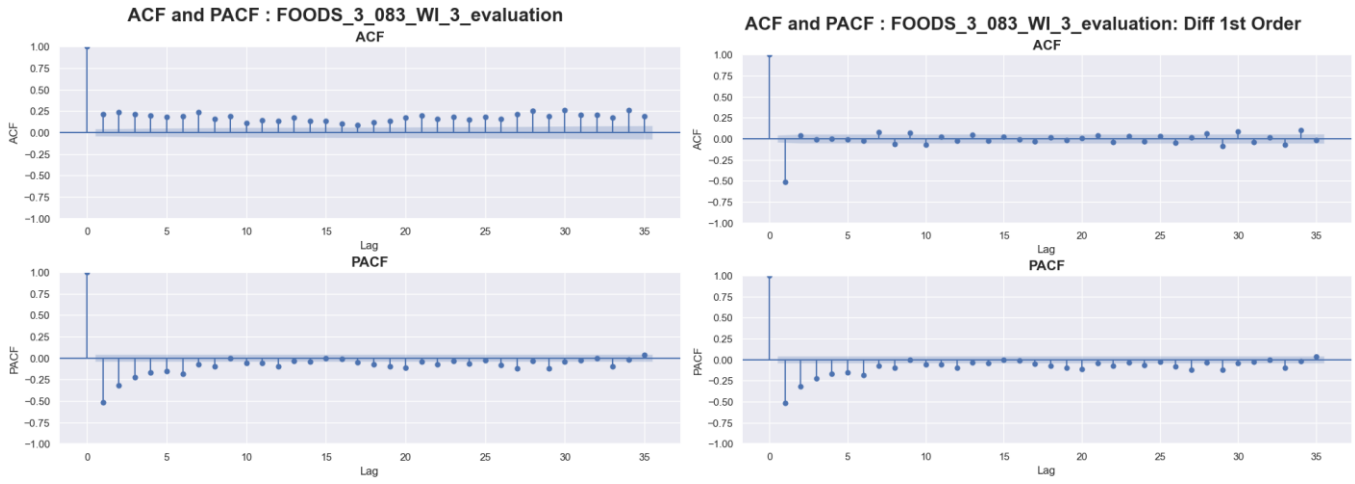
After applying first-order differencing, the data became more stationary, with a significantly low p-value of 8.52e-30 and a decay in ACF with higher lags. Nonetheless, notable features emerged, such as a significant negative spike at lag 1 in ACF and negative correlations in PACF. While the last value outside the negative confidence band appears at lag 8, correlations exceeding the negative confidence band are present with other lags. For simplicity, we opt to use an MA model, resulting in ARIMA parameters of (1,1,0).



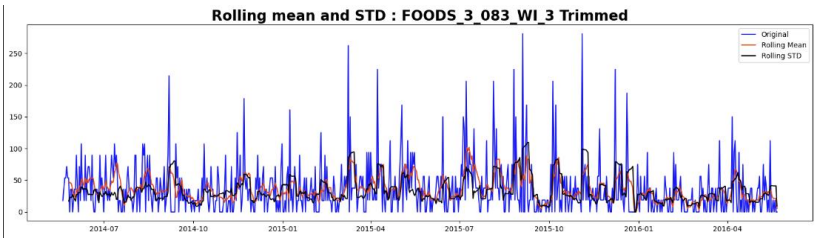
Dickey-Fuller Test:	
ADF Test Static	-3.621516
p-value	0.005363
# of Legs used	26.000000
Number of Observation Used	1914.000000
Critical Value (1%)	-3.433771
Critical Value (5%)	-2.863051
Critical Value (10%)	-2.567575



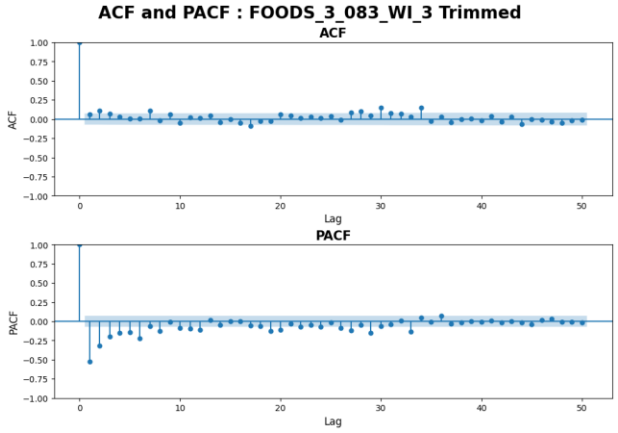
Dickey-Fuller Test:	
ADF Test Static	-1.701619e+01
p-value	8.521376e-30
# of Legs used	2.600000e+01
Number of Observation Used	1.913000e+03
Critical Value (1%)	-3.433773e+00
Critical Value (5%)	-2.863052e+00
Critical Value (10%)	-2.567575e+00



In the sliced data, the obtained p-value was 2.62e-13, indicating stationary data. Upon plotting the ACF and PACF, there were no obvious spikes observed in the ACF plot. However, the PACF plot displayed a significant spike outside the range at lag 6. As a result, we will apply (6,0,0) to the ARIMA model for this dataset.



Dickey-Fuller Test:	
ADF Test Static	-8.373310e+00
p-value	2.626541e-13
# of Legs used	6.000000e+00
Number of Observation Used	7.270000e+02
Critical Value (1%)	-3.439377e+00
Critical Value (5%)	-2.865524e+00
Critical Value (10%)	-2.568891e+00

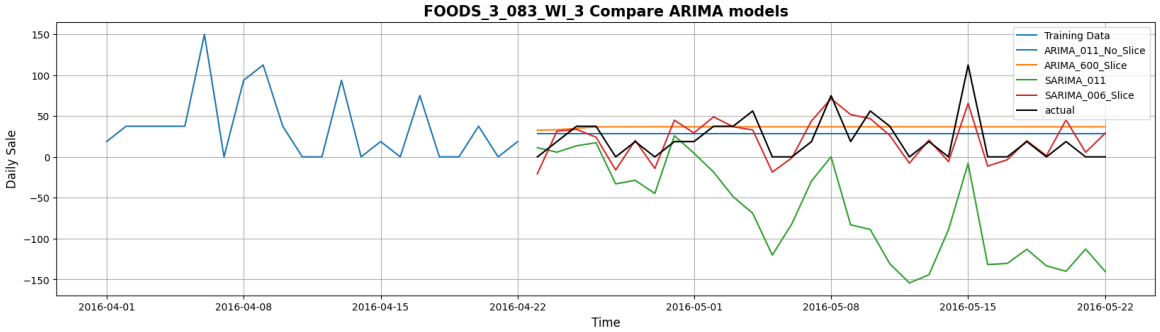


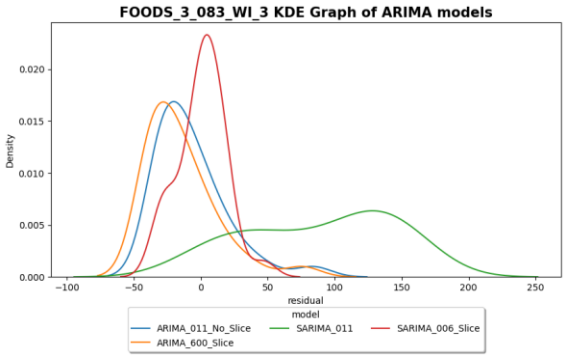
Training ARIMA models.

For training, both ARIMA without exogenous variables and SARIMAX with exogenous variables were utilized. The predicted values obtained from the original and sliced data using ARIMA and SARIMAX models were compared.

SARIMAX with sliced data demonstrated the ability to capture most of the peaks in the testing data, while SARIMAX with the original data exhibited negative trends but managed to capture some peaks. The negative trend observed in the latter case may be attributed to the presence of gaps of zeros in the training data, which increased the difficulty of determining the correct p, d, q values.

Including exogenous variables in SARIMAX models allowed for more data to be provided to the model during training, resulting in higher accuracy compared to ARIMA models. This higher accuracy was also evident in the KDE plot, where SARIMAX with sliced training data showed most of the data closely aligned with the original test values.





EXTENDED MODELING

In addition to ARIMA models, AutoARIMA and regression models (Linear Regression, Decision Tree Regression, XGBoost Regressor) were employed. Hyperparameter tuning was conducted for Decision Tree Regression and XGBoost Regressor to enhance model accuracy. AutoML techniques, including mljar-supervised and H2O, were also utilized for practice purposes.

AutoARIMA identified the best model for the original data as (2,1,0)(2,1,0)7 and for the sliced data as (6,1,0)(2,1,0)7.

When evaluating model performance using MAPE, Decision Tree Regression exhibited the highest accuracy with 26.047. However, it's important to note that the presence of zeros in the actual data may have artificially decreased the MAPE values. To mitigate this, RMSE was also considered, revealing that XGBoost Regressor achieved the most accurate result with an RMSE of 14.335. Visual inspection of the forecasted data compared to the actual data indicated that XGBoost Regressor captured most of the fluctuation patterns.

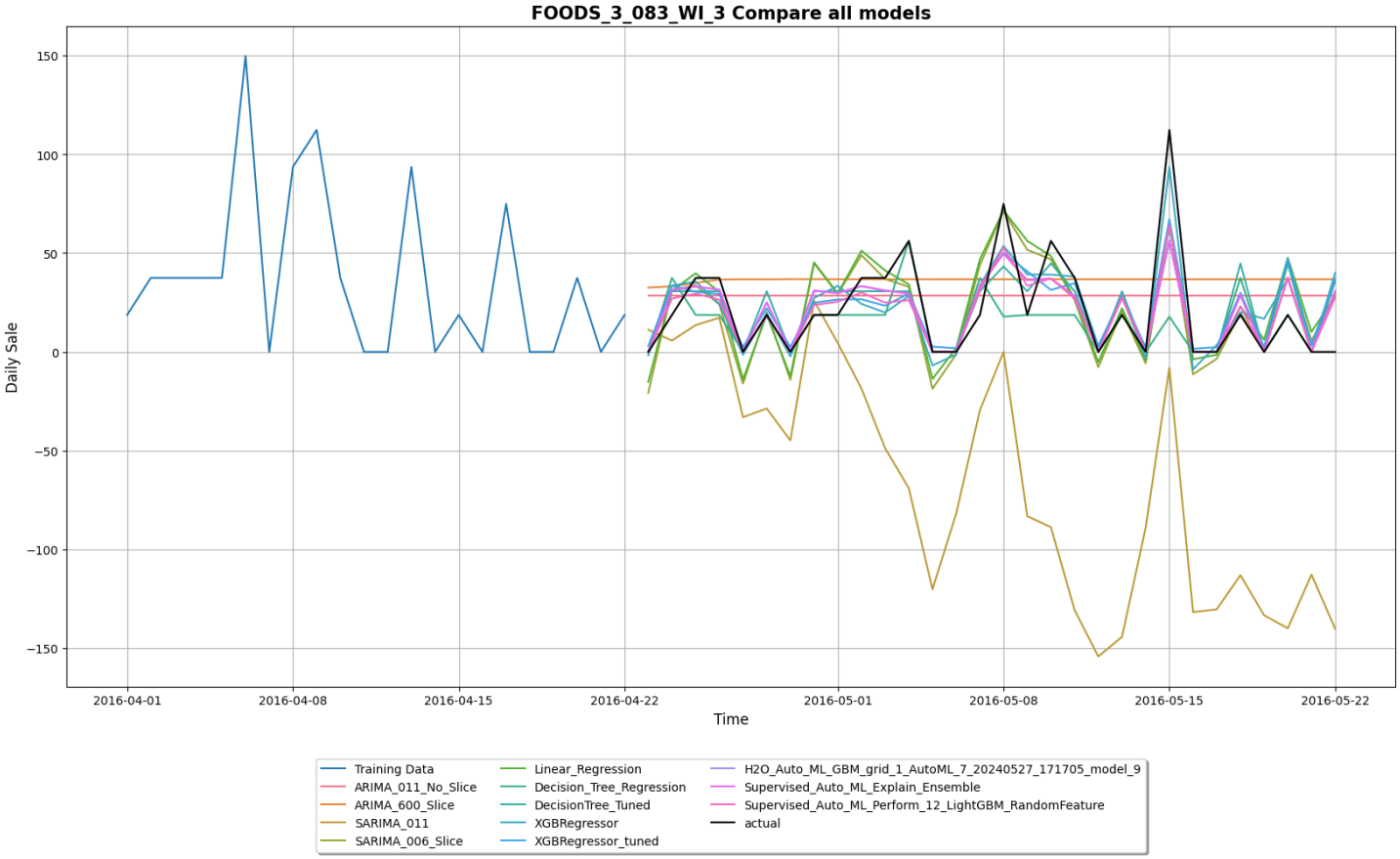
Furthermore, the KDE plot highlighted that the LightGBM Random Forest model generated by mljar supervised AutoML displayed the least difference between predicted and actual data, suggesting strong performance across all models.

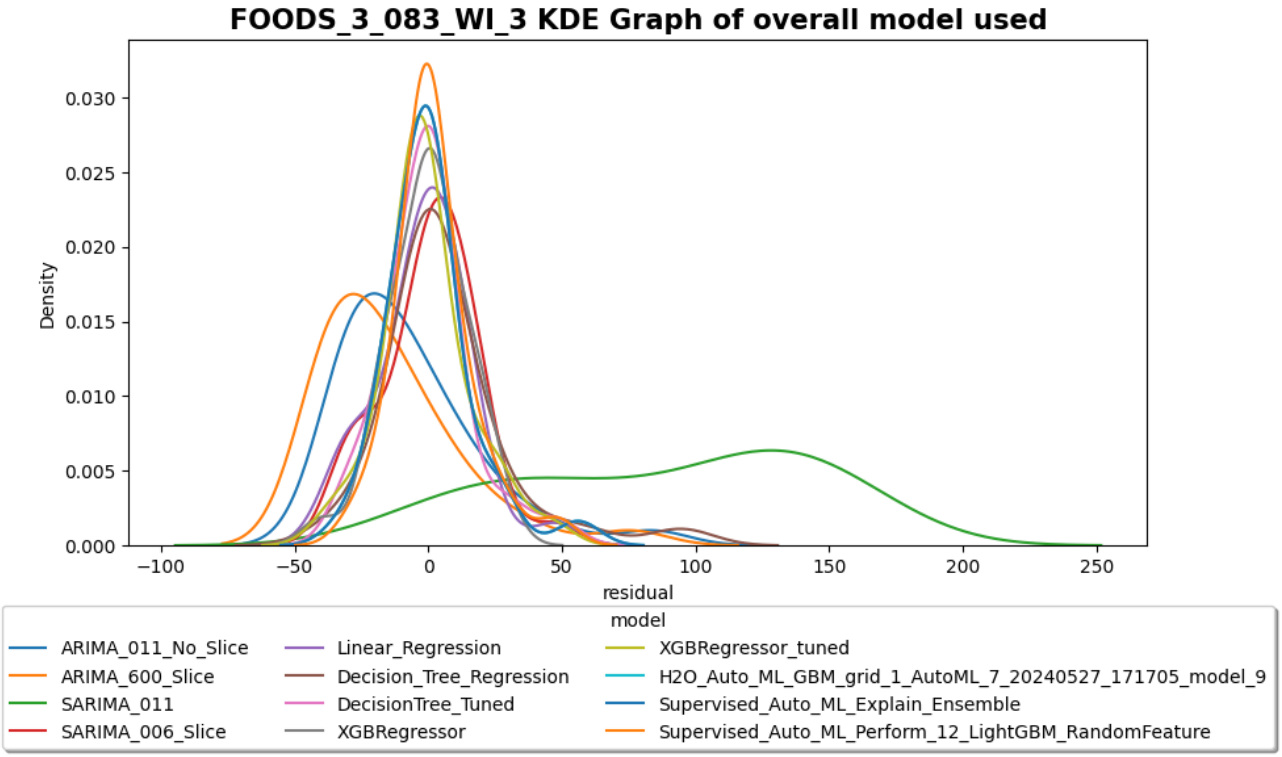
Table of MAPE

	id	model	adjust_r2	mape	r2	rmse
5	FOODS_3_083_WI_3_evaluation	Decision_Tree_Regression	0.083680	26.047636	0.368055	24.271428
0	FOODS_3_083_WI_3_evaluation	ARIMA_011_No_Slice	-0.351559	27.615579	0.067891	26.951169
13	FOODS_3_083_WI_3_evaluation	XGBRegressor_tuned	0.194988	30.105138	0.444819	16.211480
4	FOODS_3_083_WI_3_evaluation	DecisionTree_Tuned	0.270323	32.239183	0.496775	16.005696
1	FOODS_3_083_WI_3_evaluation	ARIMA_600_Slice	0.007167	35.001550	0.315288	29.831409
11	FOODS_3_083_WI_3_evaluation	Supervised_Auto_ML_Perform_12_LightGBM_RandomF...	0.192126	37.265888	0.442845	14.494863
6	FOODS_3_083_WI_3_evaluation	H2O_Auto_ML_GBM_grid_1_AutoML_7_20240527_17170...	0.187311	41.506044	0.439525	15.826093
10	FOODS_3_083_WI_3_evaluation	Supervised_Auto_ML_Explain_Ensemble	0.187311	41.506044	0.439525	15.826093
12	FOODS_3_083_WI_3_evaluation	XGBRegressor	0.556164	73.397855	0.693906	14.334725
7	FOODS_3_083_WI_3_evaluation	Linear_Regression	0.864611	87.044692	0.906629	17.327096
2	FOODS_3_083_WI_3_evaluation	Auto_ARIMA_no_zero_mod	0.946537	91.708215	0.963129	17.349737
8	FOODS_3_083_WI_3_evaluation	SARIMA_006_Slice	0.946537	91.708215	0.963129	17.349737
3	FOODS_3_083_WI_3_evaluation	Auto_ARIMA_sliced	1.732753	95.426566	1.505347	37.947214
9	FOODS_3_083_WI_3_evaluation	SARIMA_011	23.537154	249.770264	16.542865	103.543732

Table of RMSE

	id	model	adjust_r2	mape	r2	rmse
12	FOODS_3_083_WI_3_evaluation	XGBRegressor	0.556164	73.397855	0.693906	14.334725
11	FOODS_3_083_WI_3_evaluation	Supervised_Auto_ML_Perform_12_LightGBM_RandomF...	0.192126	37.265888	0.442845	14.494863
6	FOODS_3_083_WI_3_evaluation	H2O_Auto_ML_GBM_grid_1_AutoML_7_20240527_17170...	0.187311	41.506044	0.439525	15.826093
10	FOODS_3_083_WI_3_evaluation	Supervised_Auto_ML_Explain_Ensemble	0.187311	41.506044	0.439525	15.826093
4	FOODS_3_083_WI_3_evaluation	DecisionTree_Tuned	0.270323	32.239183	0.496775	16.005696
13	FOODS_3_083_WI_3_evaluation	XGBRegressor_tuned	0.194988	30.105138	0.444819	16.211480
7	FOODS_3_083_WI_3_evaluation	Linear_Regression	0.864611	87.044692	0.906629	17.327096
2	FOODS_3_083_WI_3_evaluation	Auto_ARIMA_no_zero_mod	0.946537	91.708215	0.963129	17.349737
8	FOODS_3_083_WI_3_evaluation	SARIMA_006_Slice	0.946537	91.708215	0.963129	17.349737
5	FOODS_3_083_WI_3_evaluation	Decision_Tree_Regression	0.083680	26.047636	0.368055	24.271428
0	FOODS_3_083_WI_3_evaluation	ARIMA_011_No_Slice	-0.351559	27.615579	0.067891	26.951169
1	FOODS_3_083_WI_3_evaluation	ARIMA_600_Slice	0.007167	35.001550	0.315288	29.831409
3	FOODS_3_083_WI_3_evaluation	Auto_ARIMA_sliced	1.732753	95.426566	1.505347	37.947214
9	FOODS_3_083_WI_3_evaluation	SARIMA_011	23.537154	249.770264	16.542865	103.543732

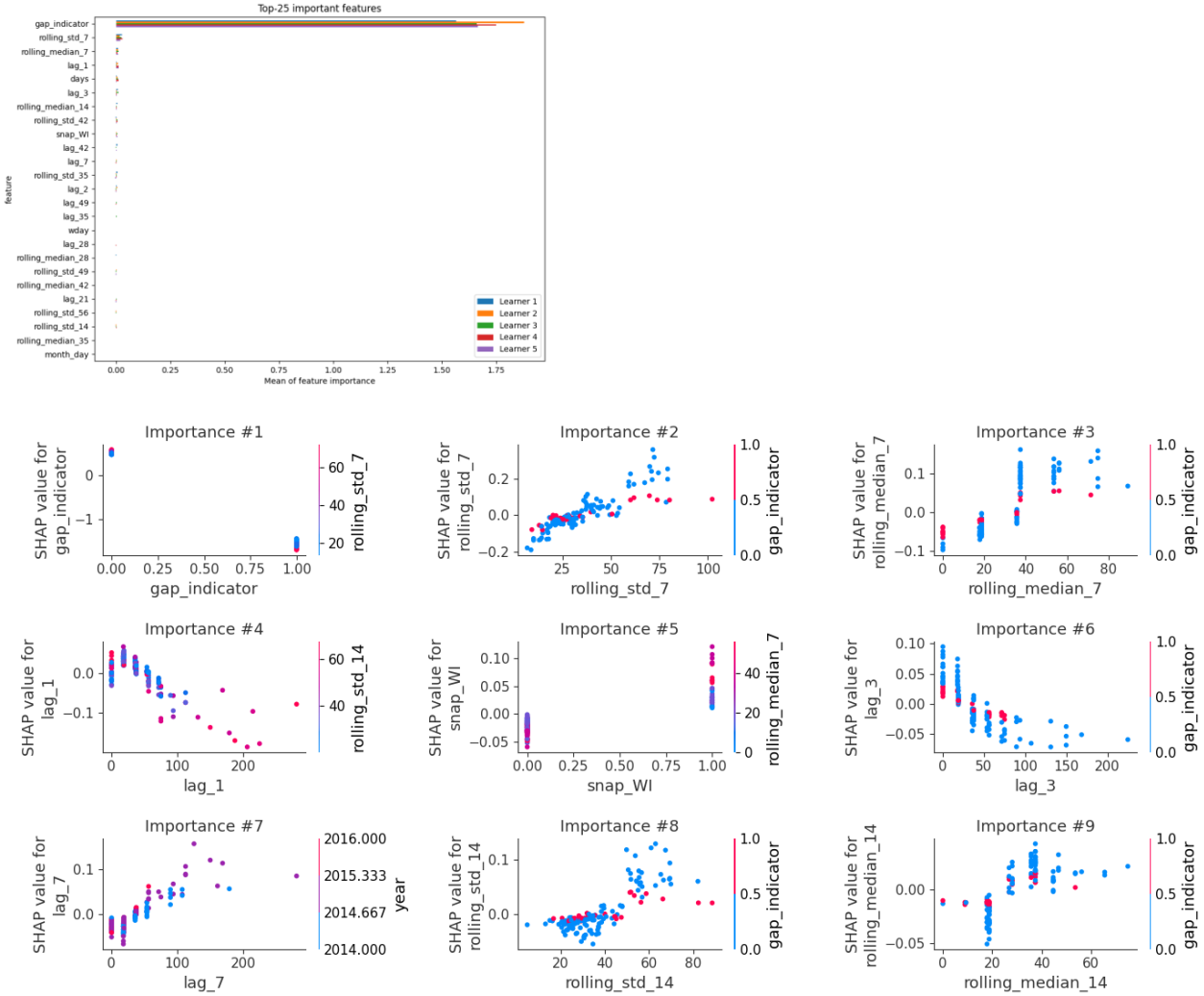

FOODS_3_083_WI_3 Compare all models
Daily Sale
Time
Training Data
ARIMA_011_No_Slice
ARIMA_600_Slice
SARIMA_011
SARIMA_006_Slice
Linear_Regression
Decision_Tree_Regression
DecisionTree_Tuned
XGBRegressor
XGBRegressor_tuned
H2O_Auto_ML_GBM_grid_1_AutoML_7_20240527_171705_model_9
Supervised_Auto_ML_Explain_Ensemble
Supervised_Auto_ML_Perform_12_LightGBM_RandomFeature
actual


FOODS_3_083_WI_3 KDE Graph of overall model used
Density
residual
model
ARIMA_011_No_Slice
ARIMA_600_Slice
SARIMA_011
SARIMA_006_Slice
Linear_Regression
Decision_Tree_Regression
DecisionTree_Tuned
XGBRegressor
XGBRegressor_tuned
H2O_Auto_ML_GBM_grid_1_AutoML_7_20240527_171705_model_9
Supervised_Auto_ML_Explain_Ensemble
Supervised_Auto_ML_Perform_12_LightGBM_RandomFeature

The evaluation of LightGBM with Random Features and the SHAP plots of feature importance revealed insightful findings. The gap indicator emerged as the most important feature, indicating its significant contribution to the model's predictive performance. Additionally, both rolling_std_7 and rolling_median_7 exhibited predominantly positive SHAP values, suggesting their importance in predicting sales data.

Notably, the model demonstrated proficiency in predicting small values when leveraging rolling std features, particularly those with window sizes of 7 and 14. Conversely, higher values were predicted more accurately with the utilization of rolling median features.

These observations underscore the nuanced relationship between different features and their impact on sales prediction. By understanding the significance of various features, model performance can be optimized to achieve more accurate forecasts.



CONCLUSION AND FUTURE WORK

The time series analysis conducted on the single product has yielded valuable insights. While ARIMA and SARIMAX were initially expected to provide accurate predictions, regression models demonstrated higher accuracy. This discrepancy may be attributed to the presence of zero values in the data, which ARIMA models struggle to effectively handle.

Understanding of necessity of Exogeneous variables and unusual gap data should not be included in the training table decreasing the accuracy by increasing the noise of the data.

For future work, several avenues could be explored. Firstly, experimenting with different data resolutions (such as weekly mean data) reducing the noise of zero values and test the effect on accuracy of the forecast models. Additionally, incorporating hyperparameter tuning into AutoML models could further enhance prediction accuracy.

Furthermore, transitioning the project into a pipeline format would enable seamless integration with cloud computing services. This would facilitate the utilization of advanced machine learning models such as Azure AutoML and AWS AutoML. Moreover, exploring deep learning techniques for forecasting could unlock additional insights and improve prediction accuracy.

By pursuing these avenues, the forecasting model can be refined and optimized, leading to more accurate predictions and deeper insights into sales trends.