Final Report

Date: 12/04/2022

CECS 551 Advanced Artificial Intelligence

California State University Long Beach

Group ~ 02

Submitted To:

Prof. Mahshid Fardadi

Pragya Sharma

Almee Christian

Hasha Bhargav Bolem

Taher Vora

Suraj Shah

CONTENTS

Project Links

Colab URLs:

1. Sprint 1 -

https://colab.research.google.com/drive/1KLOtSwwLYHrW4Ruo7gr92OqsFWF-wQNH?usp=sharing

2. Sprint 2 -

    a. Dataset 1 -

https://colab.research.google.com/drive/1DuvTCiI-SZFvkJqjq-i_SmiK-yybJliG?usp=sharing

    b. Dataset 2 -

https://colab.research.google.com/drive/1faD5HFDRjBfbAUd-zB_8lKv7XdU7iSRY?usp=sharing

3. Sprint 3 -

https://colab.research.google.com/drive/1DwBmXI7T6eQESCXxtWc1WyjiYs0cLeRX?usp=sharing

Tableau Dashboard URL:

https://public.tableau.com/app/profile/pragya.sharma1552/viz/Group_02_Christian_Thube_Bolem_Sharma_Vora/Dashboard?publish=yes

Presentation URL:

Group2_FinalReport

Introduction of Datasets

1. Dataset - 1

The dataset has three parts: Stores features, train and test along with individual store details.

This dataset is of the retail store. Following are the features of the dataset:

- Store - the store number
- Date - the week
- Temperature - average temperature in the region
- Fuel_Price - cost of fuel in the region
- Discounts store - The number of stores
- type - Stores segregated into three types, i.e., A, B, and C
- size - Size of the store
- Weekly_Sales - weekly sales of the store.
- CPI - the consumer price index
- Unemployment - the unemployment rate in the region where the store is present.
- IsHoliday - whether the week is a special holiday week.

Below are the files that are used for the dataset:

- **“store_01.csv to store_11_35.csv” ~** Contains information about all the individual stores from 1 to 35 with the features and the dates:
  - Store: The id of the store.
  - Date: The date in a “m-d-y” format.(Before preprocessing).
  - IsHoliday: The date given is either holiday or not.
  - Temperature: The temperature recorded on that date.
  - gas_price: The gas price on that particular date.
  - discount_promotional: The promotional discount on that date.
  - discount_clearance: The clearance discount offered on that date.
  - discount_damaged_good: The discount on the no. of goods that are damaged.
  - discount_competetive: The discount based on the market competitive price.
  - CPI - the consumer price index
  - Unemployment - the unemployment rate in the region where the store is present.

- **“stores.csv” ~** Contains information about all the stores from 1 to 45 with their type (A,B,C) and the size of that store.
    - Store: The id of the store.
    - Type: The type of the store either A,B,C
    - Size: The size of that particular store.

- **“test.csv” ~** Contains information about all the stores from 1 to 45 with their specific department id along with the dates and holiday details.
    - Store: The id of the store.
    - Dept: The id of the specific department belonging to the store.
    - Date: The date in a “d-m-y” format.(Before preprocessing).
    - IsHoliday: The date given is either holiday or not.

- **“train.csv” ~** Contains information about all the stores from 1 to 45 with their specific department id along with the dates, weekly sales recorded and the holiday details.
    - Store: The id of the store.
    - Dept: The id of the specific department belonging to the store.
    - Date: The date in a “d-m-y” format.(Before preprocessing).
    - Weekly_Sales: The weekly sales recorded in the stores as per the date.
    - IsHoliday: The date given is either holiday or not.

2. Dataset - 2

This dataset represents the unit sales (in number/quantity) of various products sold in one year (2011-12) in the USA, organized in the form of grouped time series. More specifically, the dataset involves the unit sales of around 3000 products, classified in 3 product categories (Hobbies, Foods, and Household), and 7 product departments in which the above-mentioned categories are disaggregated.

The products are sold across 10 stores, located in 3 States (CA, TX, and WI).

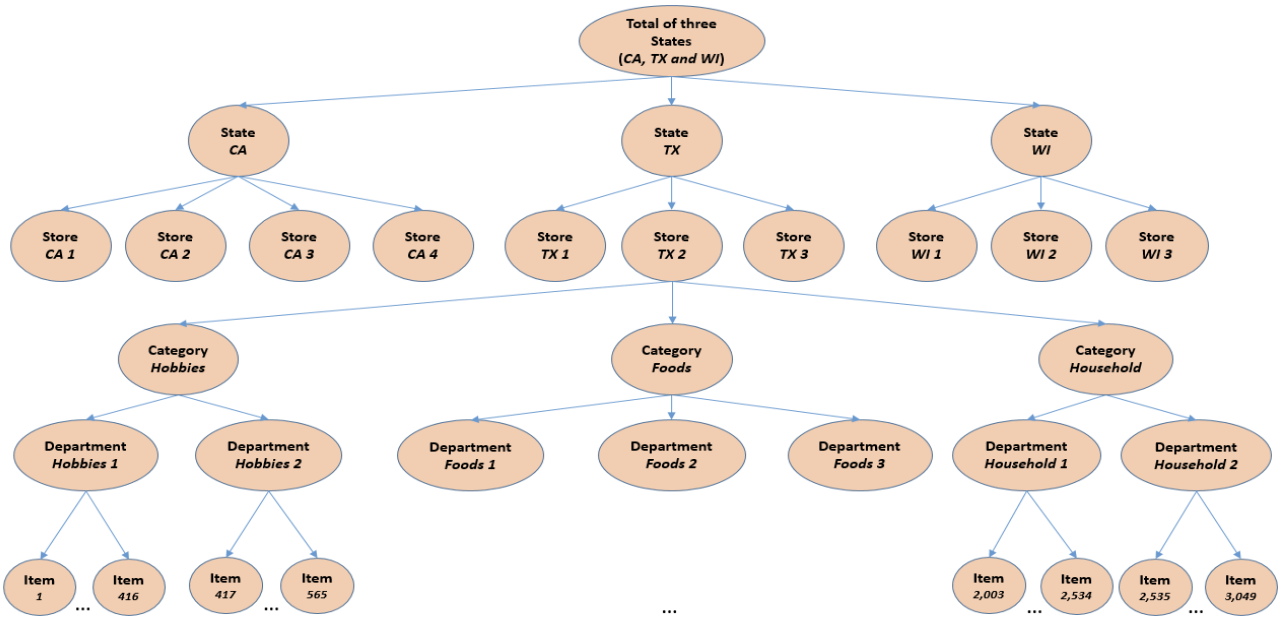


Fig. 1: An overview of how the dataset - 2 series are organized

This dataset consists of three files:

- **“calendar.csv” ~** Contains information about the dates the products are sold.
    - date: The date in a “y-m-d” format.
    - wm_yr_wk: The id of the week the date belongs to.
    - weekday: The type of the day (Saturday, Sunday, …, Friday).
    - wday: The id of the weekday, starting from Saturday.
    - month: The month of the date.
    - year: The year of the date.
    - event_name_1: If the date includes an event, the name of this event.
    - event_type_1: If the date includes an event, the type of this event.
    - event_name_2: If the date includes a second event, the name of this event.
    - event_type_2: If the date includes a second event, the type of this event.
    - snap_CA, snap_TX, and snap_WI: A binary variable (0 or 1) indicating whether the stores of CA, TX or WI allow SNAP purchases on the examined date. 1 indicates that SNAP (Supplement Nutrition Assistance Program) purchases are allowed.

- **“sell_prices.csv” ~** Contains information about the price of the products sold per store and date.
    - store_id: The id of the store where the product is sold.
    - item_id: The id of the product.
    - wm_yr_wk: The id of the week.
    - sell_price: The price of the product for the given week/store. The price is provided per week (average across seven days). If not available, this means that the product was not sold during the examined week. Note that although prices are constant on a weekly basis, they may change through time (both training and test set).

- **“sales_train.csv” ~** Contains the historical daily unit sales data per product and store.
    - item_id: The id of the product.
    - dept_id: The id of the department the product belongs to.
    - cat_id: The id of the category the product belongs to.
    - store_id: The id of the store where the product is sold.
    - state_id: The State where the store is located.
    - d_1, d_2, …, d_i, … d_1941: The number of units sold at day i, starting from 2011-01-29.

Sprint - 1

Retail analysis using Artificial Intelligence approach

2.1 Sprint 1: Data Visualization

2.1.1 Analyze the dataset for CECS551_dataset_01

A. Identify the key variables for the model using correlation plots, heatmaps, histograms, feature importance (SHAP).

The dataset has three parts: Stores features, train and test. The store features has 11 different
We built a correlation heatmap between different features of the store such as IsHoliday, Discount, temperature, Unemployment, etc. and Weekly_Sales, . For this we needed to join the data in the excel sheets for different stores and train data as it contains weekly sales for every department of every store. We joined these excel sheets based on Store, Dept and IsHoliday.

```
data = pd.merge(data_train, data_store, on=["Store", "Date", "IsHoliday"])
```

Using this new dataframe that had store, department as well as weekly sales we did preprocessing of data. To eliminate the null fields of the dataset, we replaced them with zeros. Also, as IsHoliday is a column with boolean values, we converted False to 0 and True to 1.

```
data.IsHoliday = data.IsHoliday.replace({True: 1, False: 0})
data.fillna(0, inplace=True)
```

- **Correlation Heatmap**

Using above preprocessed data we built a correlation heatmap.

```
import matplotlib.pyplot as plt
import seaborn as sns

corrmat = data.corr()
f, ax = plt.subplots(figsize=(12, 12))
sns.heatmap(corrmat, vmax=.200, square=True);
```

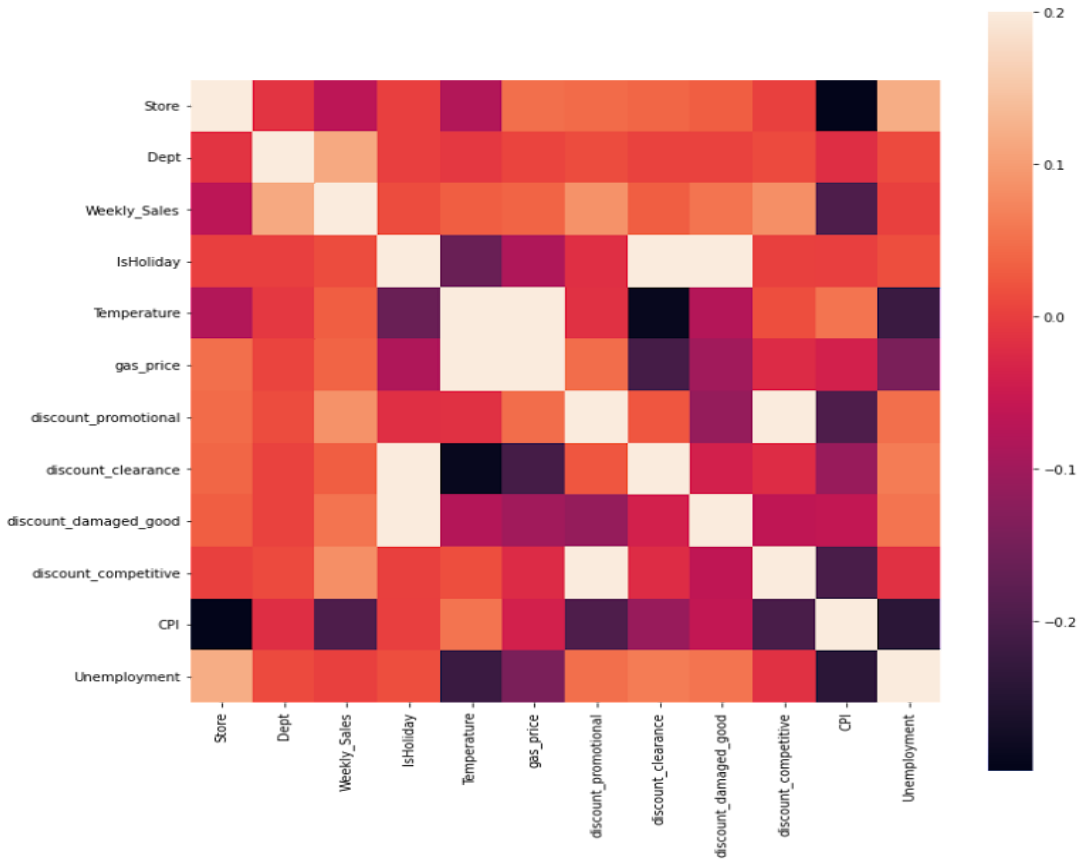


Fig.: Correlation Heatmap

As it is seen, there are several correlations that can be predicted. We can see that there's correlation between temperature and discount clearance. Also there's a positive correlation between discount correlation as weekly sales. This shows that as season changes and discount clearance is available, there's a rise in sales. On the other hand CPI has an inverse effect. As per the correlation map, as CPI increases, the weekly sales are reduced. Competitive discounts also have a positive correlation with weekly sales. Similarly, multiple relations can be established.

For the first 10 stores visualize the weekly and monthly sales patterns for the top 35% of the department sales.

Visualizing weekly and monthly sales for the top 35% needed us to perform calculations on the dataset. Similar to part A, we joined the store features excel sheets and train data based on Store, Dept and IsHoliday**.**

Feature Importance using SHAP (SHapley Additive exPlanation)

SHAP values are a unified measure of feature importance. These values are basically solutions to an equation mentioned in the paper 'A Unified Approach to Interpreting Model Predictions'.

Both figure 2 and 3 show importance of each importance based on their impact on the model output.

The Beeswarm plot (fig. 3) shows us that high CPI predicts low sales and vice versa. CPI is the most important feature for our model. Unemployment shows mixed impact on both negative and positive classes. Promotional and damaged goods discounts show higher impact on increasing the sales when these discounts increase. High gas price corresponds to slight negative impact on sales. Similar to unemployment, the effect of temperature on sales is a little ambiguous. The rest of the features seem to have a small impact on the sales. Similar feature importance is depicted in the bar plot Fig. 4.

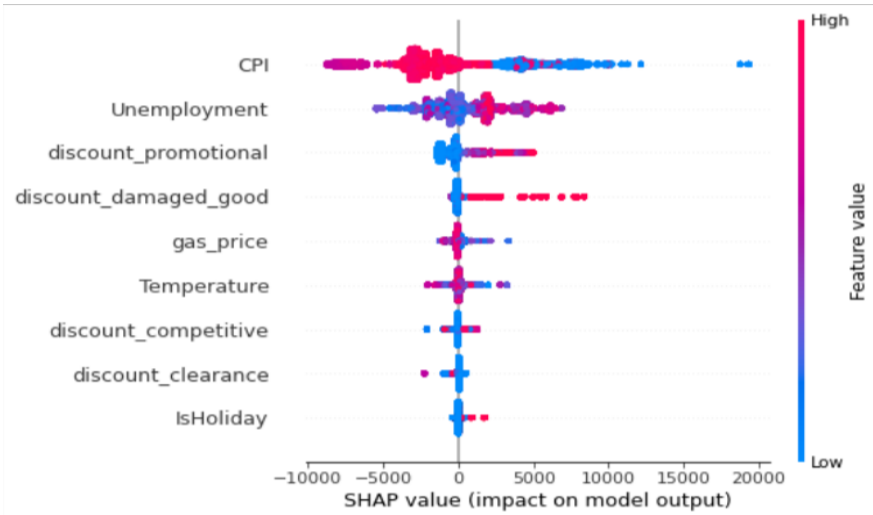


Fig.: SHAP value (impact on model output)

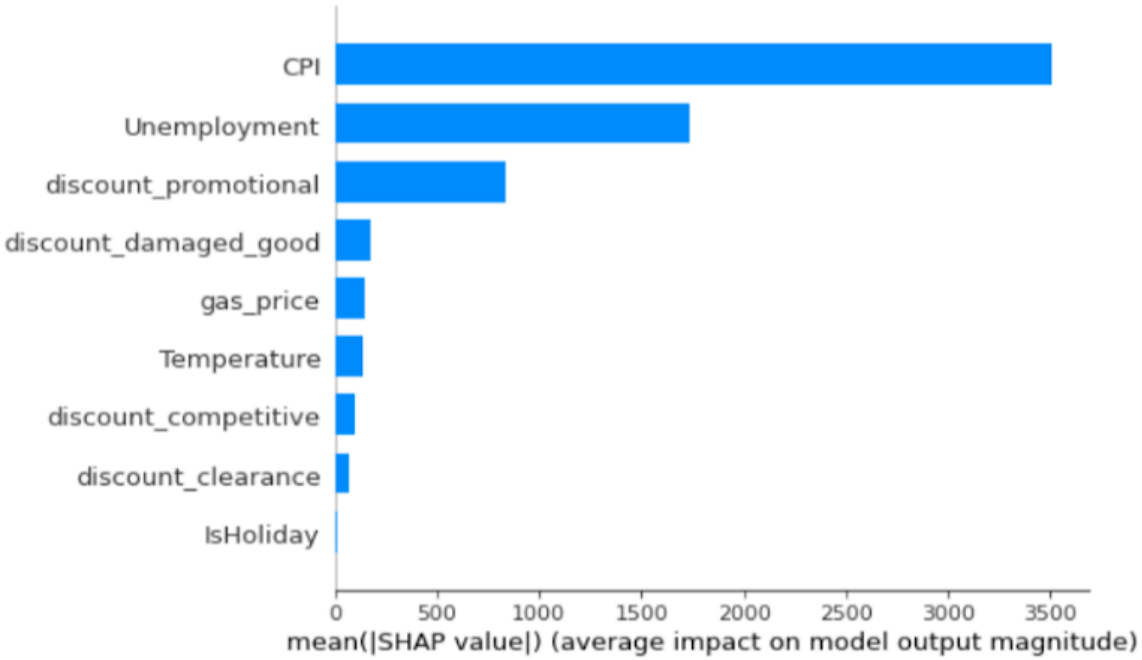


Fig.: mean(|SHAP value|) average impact on model output magnitude

B. For the first 10 stores visualize the weekly and monthly sales patterns for top 35% of the product sales and department sales.

- **Weekly Sales Pattern**

First we took a mean of weekly sales for every store per department per year.

```
df = pd.DataFrame({'Average_weekly_sales_per_store_per_year' :
data.groupby(['Store', 'Dept','year'])['Weekly_Sales'].mean()}).reset_index()
```

After this, we calculated average weekly sales for each department per year.

```
df1 = pd.DataFrame({'Average_Weekly_Sales' : df.groupby(['Dept',
'year'])['Average_weekly_sales_per_store_per_year'].mean()}).reset_index()
```

	Store	Dept	year	Average_weekly_sales_per_store_per_year
0	1	1	2010	23465.598542
1	1	1	2011	22529.823077
2	1	1	2012	21430.363953
3	1	2	2010	45844.475833
4	1	2	2011	45950.409423
...	...	...	...	...
2194	10	97	2011	5209.476538
2195	10	97	2012	5119.292326
2196	10	98	2010	180.492727
2197	10	98	2011	201.049762
2198	10	98	2012	225.580417

2199 rows × 4 columns

	Dept	year	Average_Weekly_Sales
0	1	2010	21408.970438
1	1	2011	20796.213212
2	1	2012	19907.404326
3	2	2010	46967.562625
4	2	2011	47733.620904
...	...	...	...
229	98	2011	5851.242527
230	98	2012	5824.112630
231	99	2010	-18.226667
232	99	2011	823.852808
233	99	2012	159.013271

234 rows × 3 columns

To understand the pattern of weekly sales for the top 35% of departments across all the stores, we plotted a 3D graph. X axis represents year, Y axis represents department and Z axis represents average weekly sales.

As we can see from the figures 3 to 8, there were 2 candidates for the departments with the best weekly sales. Department 95 had approximately 71k average weekly sales in 2010, 73k in 2012 and 77k in 2012. On the other hand, department 38 had approximately 75k average weekly sales in 2010, 74k in 2011 and 72k in 2012. Although department 38 had slightly better average weekly sales than department 95 in the years 2010 and 2011, we can observe that overall sales are decreasing for department 38 where sales are increasing for department 95 having the highest weekly average sales of all time in the most recent year 2012 which is 77k.

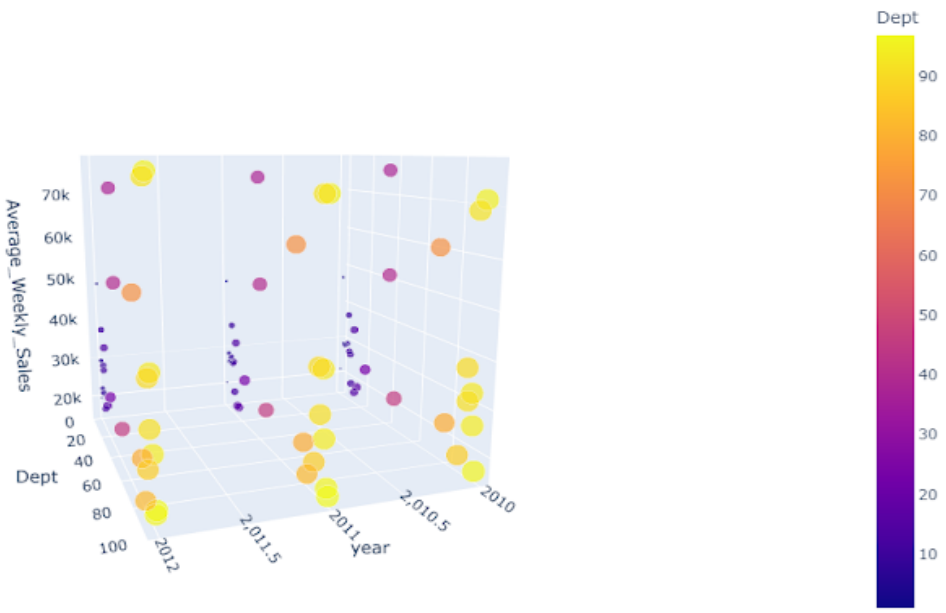


Fig.: 3D Graph for average weekly sales for top 35% departments across 10 stores

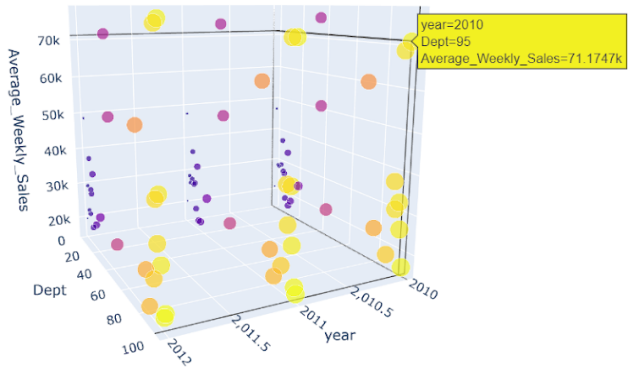


Fig.: Dept 95 avg weekly sales in 2010

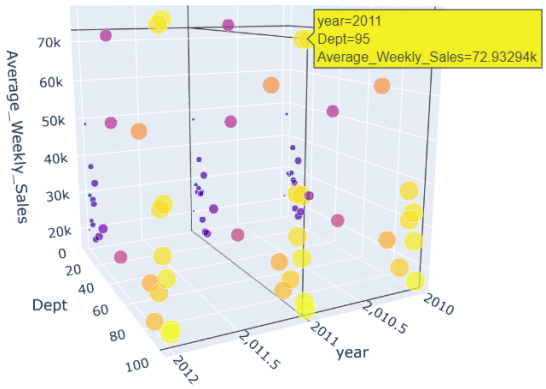


Fig.: Dept 95 avg weekly sales in 2011

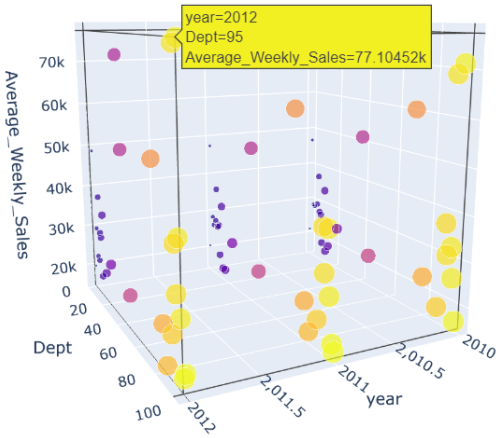


Fig.: Dept 95 avg weekly sales in 2012

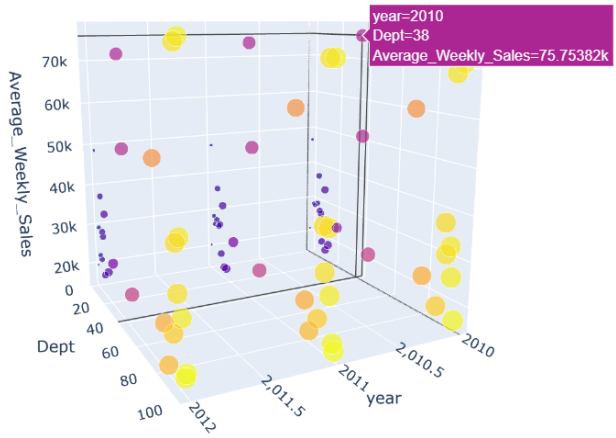


Fig.: Dept 38 avg weekly sales in 2010

Fig.: Dept 38 avg weekly sales in 2011

Fig : Dept 38 avg weekly sales in 2012

- **Monthly Sales Pattern**

To get monthly sales data we calculated the sum of weekly data for each month. Then, we took the average of all the months' sales to get average monthly sales per store for each department of each year.

```
df = pd.DataFrame({'Total_monthly_sales_per_store_per_year' :
data.groupby(['Store', 'Dept',
'month','year'])['Weekly_Sales'].sum()}).reset_index()
df1 = pd.DataFrame({'Average_monthly_sales_per_store_per_year' :
df.groupby(['Store',
'Dept','year'])['Total_monthly_sales_per_store_per_year'].mean()}).reset_
index()
```

Then we calculated average monthly sales for each department across all the stores for every year.

```
df2 = pd.DataFrame({'Average_monthly_sales' :
df1.groupby(['Dept','year'])['Average_monthly_sales_per_store_per_year'].
mean()}).reset_index()
```

	Store	Dept	year	Average_monthly_sales_per_store_per_year
0	1	1	2010	102395.339091
1	1	1	2011	97629.233333
2	1	1	2012	92150.565000
3	1	2	2010	200048.621818
4	1	2	2011	199118.440833
...	...	...	...	...
2194	10	97	2011	22574.398333
2195	10	97	2012	22012.957000
2196	10	98	2010	721.970909
2197	10	98	2011	703.674167
2198	10	98	2012	541.393000

2199 rows × 4 columns

	Dept	year	Average_monthly_sales
0	1	2010	93420.961909
1	1	2011	90116.923917
2	1	2012	85601.838600
3	2	2010	204949.364182
4	2	2011	206845.690583
...	...	...	...
229	98	2011	25324.186167
230	98	2012	24987.187376
231	99	2010	-18.226667
232	99	2011	1750.058333
233	99	2012	415.711481

234 rows × 3 columns

After preparing the data, we created a 3D graph for average monthly sales that is similar to average weekly sales for the top 35% of the departments.

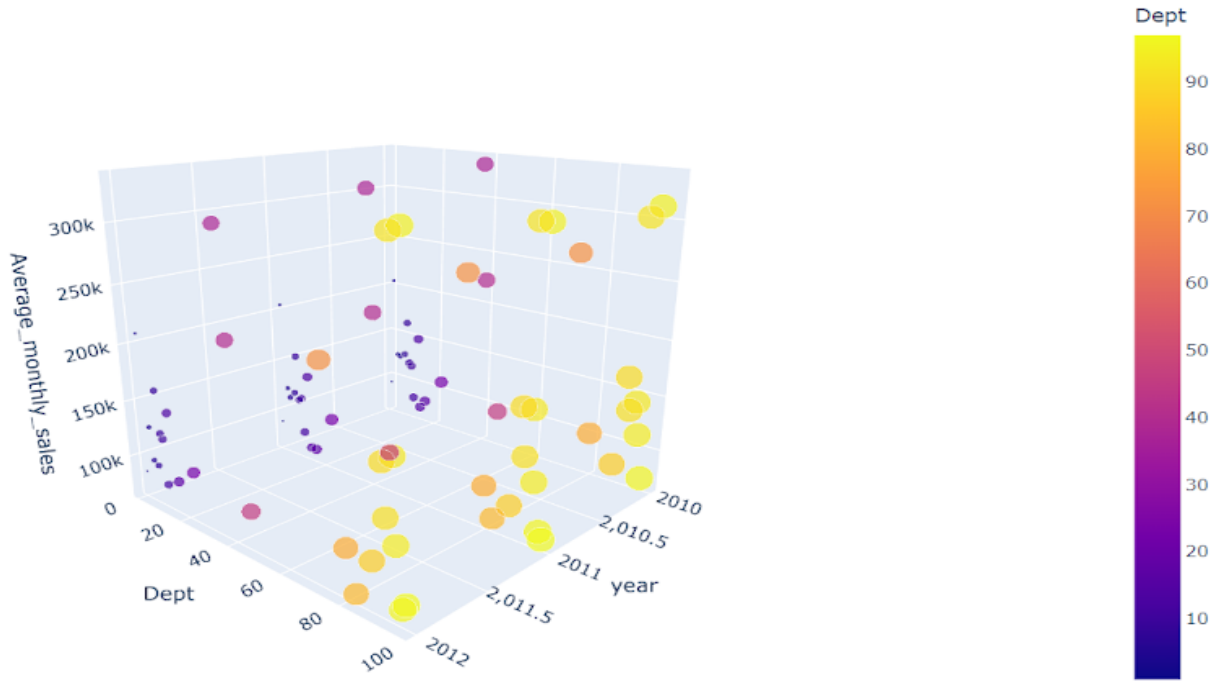


Fig: Average monthly sales for top 35% departments across all the stores

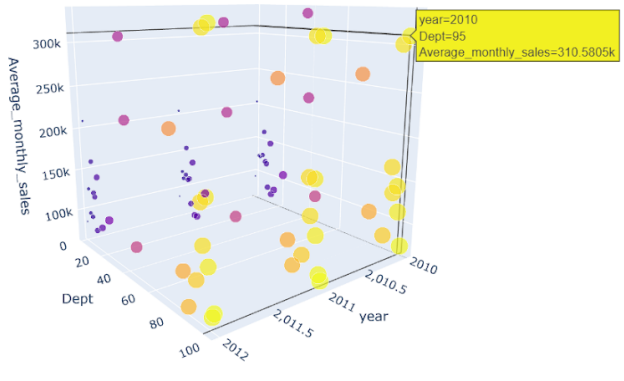


Fig: Dept 95 avg monthly sales in 2010

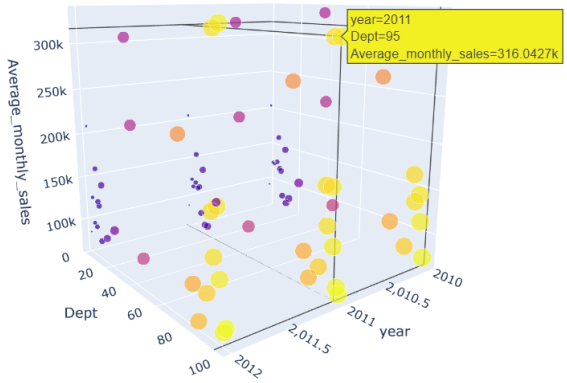


Fig: Dept 95 avg monthly sales in 2011

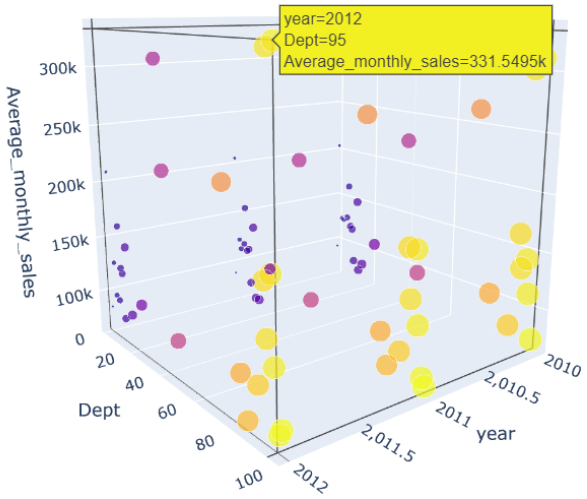


Fig: Dept 95 avg monthly sales in 2012

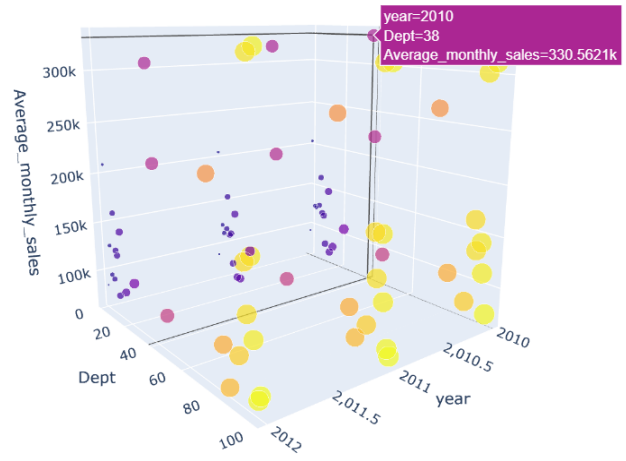


Fig.: Dept 38 avg monthly sales in 2010

Fig.: Dept 38 avg monthly sales in 2011

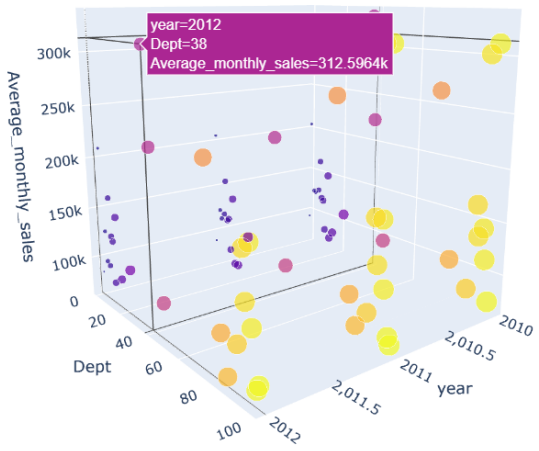


Fig. Dept 38 avg monthly sales in 2012

We see a similar pattern for average monthly sales as for average weekly sales. As we can see from the figures10 to 15, there were 2 candidates for the departments with the best monthly sales as well. Department 95 had approximately 310k average weekly sales in 2010, 316k in 2012 and 331k in 2012. On the other hand, department 38 had approximately 330k average weekly sales in 2010, 322k in 2011 and 312k in 2012. Although department 38 had slightly better average monthly sales than department 95 in the years 2010 and 2011, we can observe that overall monthly average sales are decreasing for department 38 where monthly sales are increasing for department 95 having the highest weekly average sales of all time in the most recent year 2012 which is 331.54k.

B.1. Identify the best department and product "type" across the first ten stores.

After observing average weekly sales and average monthly sales, we came to the conclusion that department 95 is the best department across all the 10 stores as it shows consistently increasing high average sales.

C. Investigate the relationship between weekly sales over CPI and unemployment for the first 10 stores. You can explore the what-if scenarios while writing the report.

We need to check the relationship between the Weekly sales over CPI and also Unemployment to see the changes.

First, let's check the relationship between Weekly Sales and CPI values. During the preprocessing the data, we have removed the null values from all the datasets and replaced the values with 0 and also we checked for the unique values and changed the data types of the date columns accordingly. Finally we have sorted the top 10 stores separately.

Before plotting our graphs we have taken three separate data frames and calculated the mean weekly sales, CPI and also the unemployment separately by grouping them with Store and Date.

	Store	Date	Weekly_Sales
0	1	2010-02-05	22516.313699
1	1	2010-02-12	22804.964444
2	1	2010-02-19	22081.755753
3	1	2010-02-26	19579.549861
4	1	2010-03-05	21298.721644
...	...	...	...
1425	10	2012-09-28	23539.755694
1426	10	2012-10-05	24095.498356
1427	10	2012-10-12	23477.933014
1428	10	2012-10-19	24094.928056
1429	10	2012-10-26	24227.070139

1430 rows × 3 columns

	Store	Date	Weekly_Sales	CPI
0	1	2010-02-05	22516.313699	223.659114
1	1	2010-02-12	22804.964444	223.753643
2	1	2010-02-19	22081.755753	223.917015
3	1	2010-02-26	19579.549861	224.132020
4	1	2010-03-05	21298.721644	224.347025
...	...	...	...	...
1425	10	2012-09-28	23539.755694	131.043000
1426	10	2012-10-05	24095.498356	131.075667
1427	10	2012-10-12	23477.933014	131.108333
1428	10	2012-10-19	24094.928056	131.149968
1429	10	2012-10-26	24227.070139	131.193097

1430 rows × 4 columns

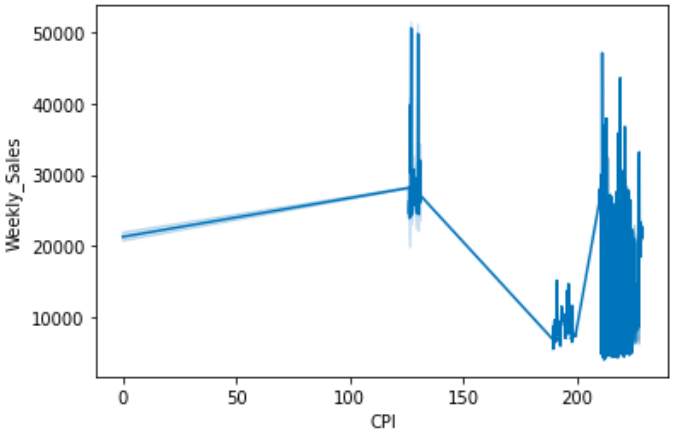
	Store	Date	Weekly_Sales	Unemployment
0	1	2010-02-05	22516.313699	6.833
1	1	2010-02-12	22804.964444	6.833
2	1	2010-02-19	22081.755753	6.833
3	1	2010-02-26	19579.549861	6.833
4	1	2010-03-05	21298.721644	6.833
...	...	...	...	...
1425	10	2012-09-28	23539.755694	7.170
1426	10	2012-10-05	24095.498356	6.943
1427	10	2012-10-12	23477.933014	6.943
1428	10	2012-10-19	24094.928056	6.943
1429	10	2012-10-26	24227.070139	6.943

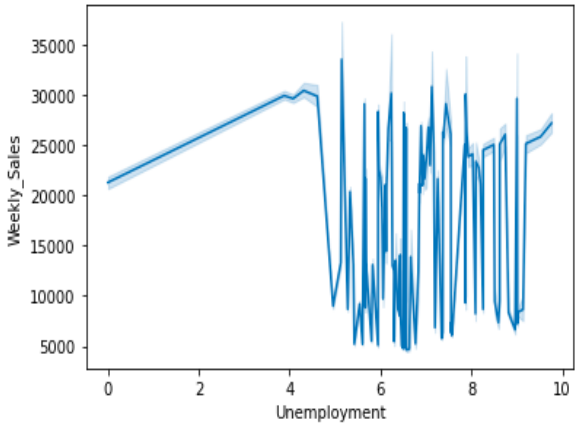
1430 rows × 4 columns

We have used the concat and the merge functions to take all the average values into a single column and then merge the results to check the relations between the Weekly sales over CPI and Weekly sales over Unemployment. Here is the below :

```
CPI_Avg_Data = pd.concat([store_01_cpi, store_02_cpi, store_03_cpi,
store_04_cpi, store_05_cpi, store_06_cpi, store_07_cpi, store_08_cpi,
store_09_cpi, store_10_cpi])

Weekly_Sales_Data= WeeklySales_Avg.head(1430)

Unemployment_Avg_Data = pd.concat([store_01_Unemployment,
store_02_Unemployment, store_03_Unemployment, store_04_Unemployment,
store_05_Unemployment, store_06_Unemployment, store_07_Unemployment,
store_08_Unemployment, store_09_Unemployment, store_10_Unemployment])

Weeklysales_CPI_data = pd.merge(Weekly_Sales_Data, CPI_Avg_Data)

Weeklysales_Unemployment_data = pd.merge(Weekly_Sales_Data,
Unemployment_Avg_Data)
```

Now, let's look at the below graphs of Weekly sales over CPI and Unemployment to draw the relationships.





By the above graphs, we can witness that the Weekly sales over the CPI has seen a regular increase but there is a sudden increase in the weekly sales for a period of time and later we see the weekly sales decrease and the increase in an irregular manner.

During the Weekly sales over the unemployment, we can witness that there is a large increase of the weekly sales when the unemployment is decreasing and then around the median of the graph we see an irregular increase and decrease of the weekly sales with respect to the increase in the unemployment.

The scenarios that we can draw from the above graph is, in both CPI and Unemployment graphs we can see that initially as the CPI and Unemployment is increasing our Weekly sales are increasing, whereas if we don't see any decrease in the unemployment, the graph for the weekly sales over the unemployment should have been positively increased. Also, For the CPi, if the weekly sales are not suddenly increased we would have seen a regular steep rise irrespective of the irregularity in the graph.

D. Investigate the impact of various types of discounts, for example, discount promotional, discount clearance, discount damaged good, discount competitive and discount employee on the overall sales.

D.1. Which type of discount is helpful in increasing the sales? Consider top 30% of the best performing stores (sales per 1000 square feet).

D.2. Does the observed behavior hold true for all the stores? Consider bottom 30% of the least performing store (sales per 1000 square feet).

To start off with the analysis, First imported the dataset and checked all different datasets to figure out which columns from one dataset needs to be merged into another dataset.

Data Preprocessing:

```
In [102]: df1.head()
```

Out[102]:

	Store	Date	IsHoliday	Temperature	gas_price	discount_promotional	discount_clearance	discount_damaged_good	discount_competitive	CPI	U
0	1	2/5/2010	False	59.33	3.360	9667.50	268.29	0.60	8368.15	223.659114	
1	1	2/12/2010	True	51.65	3.409	8687.47	1594.87	2.20	2144.87	223.753643	
2	1	2/19/2010	False	52.39	3.510	2706.87	3128.74	1.88	2396.68	223.917015	
3	1	2/26/2010	False	60.12	3.555	6129.28	1802.84	NaN	301.48	224.132020	
4	1	3/5/2010	False	61.65	3.630	3552.58	601.32	NaN	2666.22	224.347025	

From the analysis we got to know that a lot of values in store dataset are Nan(NULL) which needs to be filled.

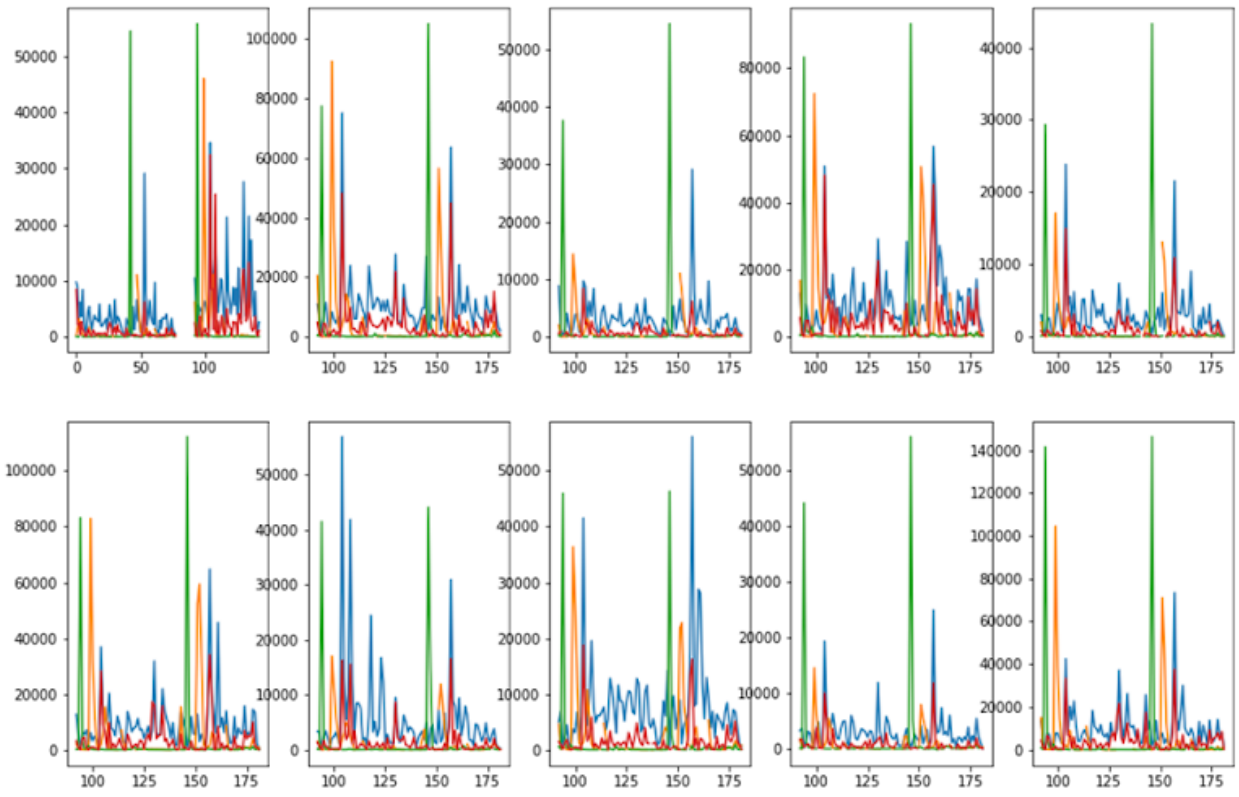
So we used df.fillna(0,inplace= True) to replace all the Nan values to 0.

```
In [103]: df1.fillna(0)
```

Out[103]:

	Store	Date	IsHoliday	Temperature	gas_price	discount_promotional	discount_clearance	discount_damaged_good	discount_competitive	CPI
0	1	2/5/2010	False	59.33	3.360	9667.50	268.29	0.60	8368.15	223.659114
1	1	2/12/2010	True	51.65	3.409	8687.47	1594.87	2.20	2144.87	223.753643
2	1	2/19/2010	False	52.39	3.510	2706.87	3128.74	1.88	2396.68	223.917015
3	1	2/26/2010	False	60.12	3.555	6129.28	1802.84	0.00	301.48	224.132020
4	1	3/5/2010	False	61.65	3.630	3552.58	601.32	0.00	2666.22	224.347025

Impacts of different discounts on stores1-stores10.



We can see from the graphs that even if there are some high spikes for discount_damaged_good Discount_promotional plays an important role in weekly sales as is it even throughout.

Calculating mean for all 35 stores to get the best performing store and worst performing store:

In [108]:
```
list1=[]
for i in range(1,36):
    store=train[(train['Store']== i) & (train['Dept']==1)]
    x=store['Weekly_Sales'].mean()
    list1.append((int(x),i))
print(list1)
print(max(list1),min(list1))
```

```
[(22513, 1), (30777, 2), (7328, 3), (36979, 4), (9774, 5), (23867, 6), (9542, 7), (14789, 8), (11846, 9), (39925, 10), (18860,
11), (17330, 12), (47020, 13), (30611, 14), (13845, 15), (11352, 16), (22801, 17), (21988, 18), (21504, 19), (40545, 20), (1495
0, 21), (21493, 22), (33186, 23), (18859, 24), (20145, 25), (19402, 26), (30437, 27), (20180, 28), (15504, 29), (9788, 30), (17
356, 31), (22852, 32), (2379, 33), (19947, 34), (17082, 35)]
(47020, 13) (2379, 33)
```

Analysis: From above values we can see that Store13 performed best and Store33 performed worst

Now as we don't have separate datasets for store 13 and store 33 we created data frames for those two stores which will give us weekly sales.

```
: # We create store13 from train dataset
store13=train[(train['Store']== 13) & (train['Dept']==1)]
store13
```

```
store33=store33=train[(train['Store']== 33) & (train['Dept']==1)]
store33
```

Also created df13 and df33 in similar way from store_11_35 dataset as we need those values to compare all the discounts.

The issue faced here was the dimensions of both the datasets to be merged were different so to equalize dimensions we dropped the last 42 values and combined both the datasets so that we get weekly_sales column which will be used to calculate top30% and bottom30% from best and worst performing store respectively.

Top performing store(Store 13):

```
In [149]: top30=df13.nlargest(math.floor(len(store13)*0.30),'Weekly_Sales')
          top30.head()
```

Out[149]:

oliday	Temperature	gas_price	discount_promotional	discount_clearance	discount_damaged_good	discount_competitive	CPI	Unemployment	Weekly_Sal
False	34.90	2.846	NaN	NaN	NaN	NaN	126.983581	7.795	172225.
False	24.76	3.186	686.24	193.26	2971.53	239.78	129.984548	6.392	168450.
False	47.55	3.655	NaN	NaN	NaN	NaN	128.955300	7.193	154137.
False	48.85	3.793	9249.62	NaN	59.94	3969.24	131.037548	5.965	150715.
False	42.15	2.842	NaN	NaN	NaN	NaN	126.603484	8.107	139870.

Calculating mean to get a glimpse of which discount helps in increasing the sale

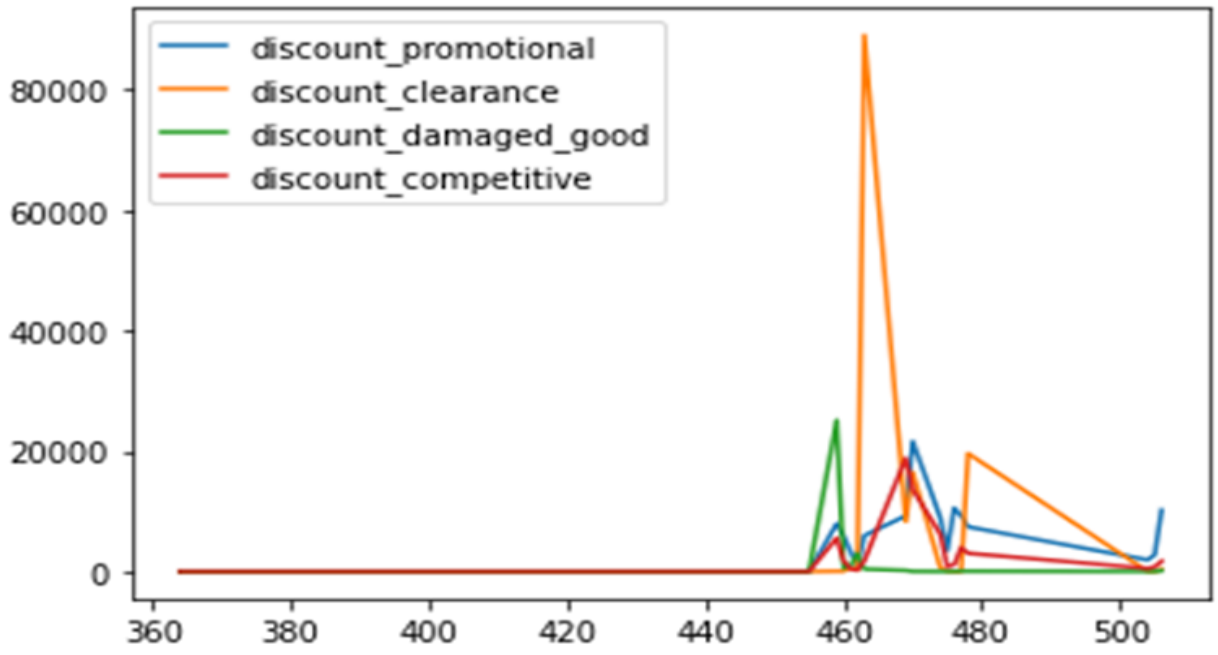
```
top30.mean()
```

```
<ipython-input-113-5de91d13c099>:1: Fut
time64 and datetime64tz columns in a fu
  top30.mean()
```

```
Store                        13.000000
IsHoliday                     0.095238
Temperature                  42.880476
gas_price                     3.251262
discount_promotional       7278.369333
discount_clearance        11292.103333
discount_damaged_good      2036.838000
discount_competitive       4024.842000
CPI                         128.727366
Unemployment                  6.994548
Weekly_Sales              78189.600238
dtype: float64
```

Plotting Graph:

```
In [116]: plt.plot(top30["discount_promotional"], label = "discount_promotional")
          plt.plot(top30["discount_clearance"], label = "discount_clearance")
          plt.plot(top30["discount_damaged_good"],  label = "discount_damaged_good")
          plt.plot(top30["discount_competitive"], label = "discount_competitive")
          plt.legend()
          plt.show()
```



From the mean and the plot we can see that discount_clearance impacts the most.

Worst Performing Store(Store33):

```
bottom30=df33.nsmallest(math.floor(len(store3)*0.30),'Weekly_Sales')
bottom30.head()
```

liday	Temperature	gas_price	discount_promotional	discount_clearance	discount_damaged_good	discount_competitive	CPI	Unemployment	Weekly_Sales
False	96.46	3.041	NaN	NaN	NaN	NaN	126.076645	9.495	711.11
False	90.82	3.087	NaN	NaN	NaN	NaN	126.101935	9.495	732.58
False	92.71	3.017	NaN	NaN	NaN	NaN	126.106903	9.495	765.00
False	94.00	3.022	NaN	NaN	NaN	NaN	126.089290	9.495	853.09
False	71.59	3.050	NaN	NaN	NaN	NaN	126.471333	9.849	912.81

Checking if there are any null values.

```
bottom30.isnull().sum()
```

```
Store                   0
Date                    0
IsHoliday               0
Temperature             0
gas_price               0
discount_promotional    42
discount_clearance      42
discount_damaged_good   42
discount_competitive    42
CPI                     0
Unemployment            0
Weekly_Sales            0
dtype: int64
```

Analysis: As we can see: All the values for discounts are Nan so we cannot tell anything from this dataset.

E. Identify the products which are highly impacted by external factors: “temperature”, “gas price”, and “holiday”. Is there any correlation between overall sales and holiday?

Since we don't have the products column we are plotting the graphs with respect to the department of the stores over the temperature, gas price and the holiday columns and look at the factors by taking the parameter as weekly sales to check the impact.

First, we have preprocessed the data by removing the null values, changed the holiday values from true to 1 and false to 0 for better analysation and later we have separated the store 1

departments and merged with the store 1 data to compare for three columns temperature, gas prices and holiday.

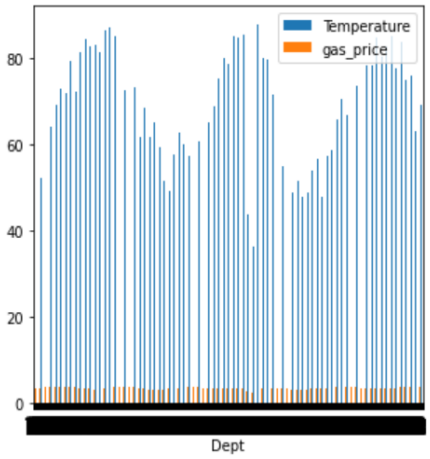
```
store1_dept = train.loc[train['Store'] == 1]
store1_dept.tail(5)
store1_dept['Date'] =  pd.to_datetime(store1_weekly['Date'])
store_01['Date'] =  pd.to_datetime(store_01['Date'])
df2 = pd.merge( store_01, store1_dept, on=['Date','Store'])
```

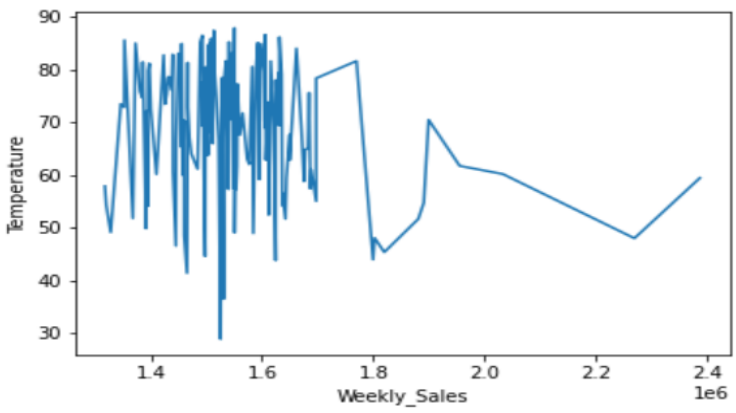
```
df2.head(5)
```

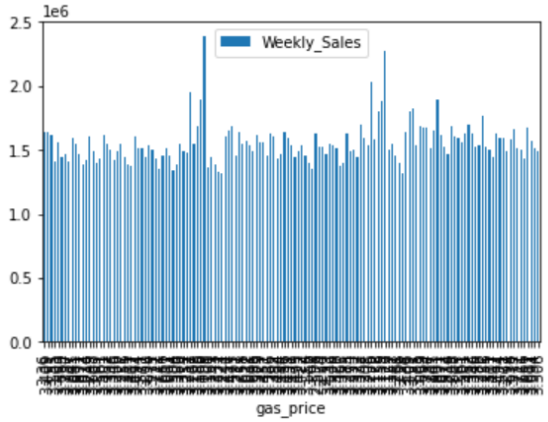
	Store	Date	IsHoliday_x	Temperature	gas_price	discount_promotional	discount_clearance	discount_damaged_good	discount_competitive	CPI	Une
0	1	2010-02-05	False	59.33	3.36	9667.5	268.29	0.6	8368.15	223.659114	
1	1	2010-02-05	False	59.33	3.36	9667.5	268.29	0.6	8368.15	223.659114	
2	1	2010-02-05	False	59.33	3.36	9667.5	268.29	0.6	8368.15	223.659114	
3	1	2010-02-05	False	59.33	3.36	9667.5	268.29	0.6	8368.15	223.659114	
4	1	2010-02-05	False	59.33	3.36	9667.5	268.29	0.6	8368.15	223.659114	

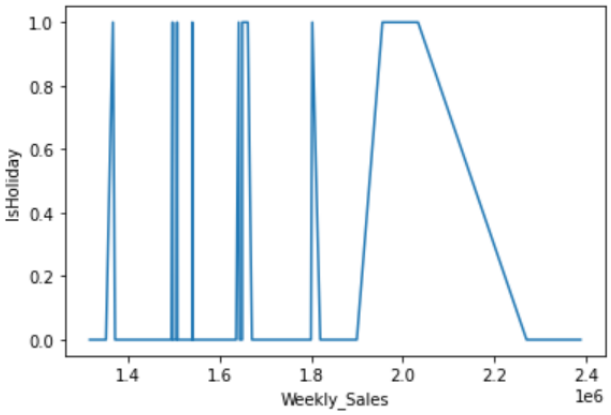
Now, since we have the four required columns in the same dataframe, we can identify the department that is highly impacting the temperature based on their weekly sales. Here, below we have checked the temperature and the gas price relationship. As per the bar graph we can witness that the temperatures are getting reduced when the gas prices are reduced and the temperatures are increasing when the gas prices starts to increase.

```
sns.lineplot(data=df2, x='Weekly_Sales', y='Temperature')
df2.plot(x = 'gas_price', y = 'Weekly_Sales', kind = 'bar')
sns.lineplot(data=df2, x='Weekly_Sales', y='IsHoliday')
```









For the Temperature and gas prices have a relation here we have lower gas prices when the temperature is lower and high sales when the temperature are medium and decrease in sales when the temperature are low for the departments 40 - 50 (according to the scale).

For the weekly sale date and the holiday relation there are spikes in the sales during holidays and most occasions in the graph two we can see the sales spike on holidays which is denoted with 1.0 and 0.0 is not a holiday.

2.1.2 Analyze the dataset for CECS551_dataset_02

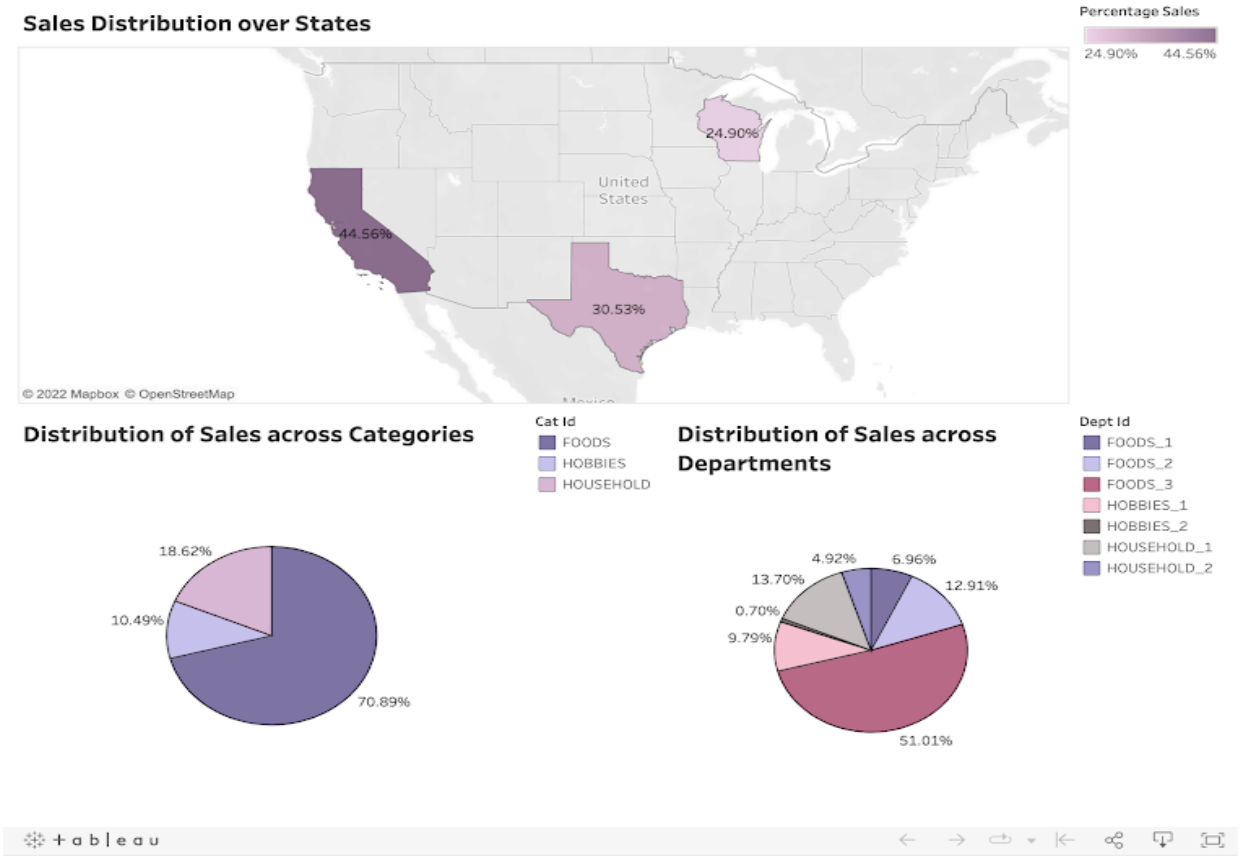
The dataset represents the unit sales (in number/quantity) of various products sold in the USA, organized in the form of grouped time series. More specifically, the dataset involves the unit sales of around 3000 products, classified in 3 product categories (Hobbies, Foods, and Household), and 7 product departments in which the above-mentioned categories are disaggregated.

The products are sold across 10 stores, located in 3 States (CA, TX, and WI). In this respect, the most disaggregated data, i.e., product- store unit sales, can be grouped based on either location (store and state) or product-related information (department and category).

The visualization of dataset_02 is performed on the data of 365 days across the stores of the three states. These visualizations are done using Tableau Desktop.

1. **Use Tableau to visualize the dataset_02.**

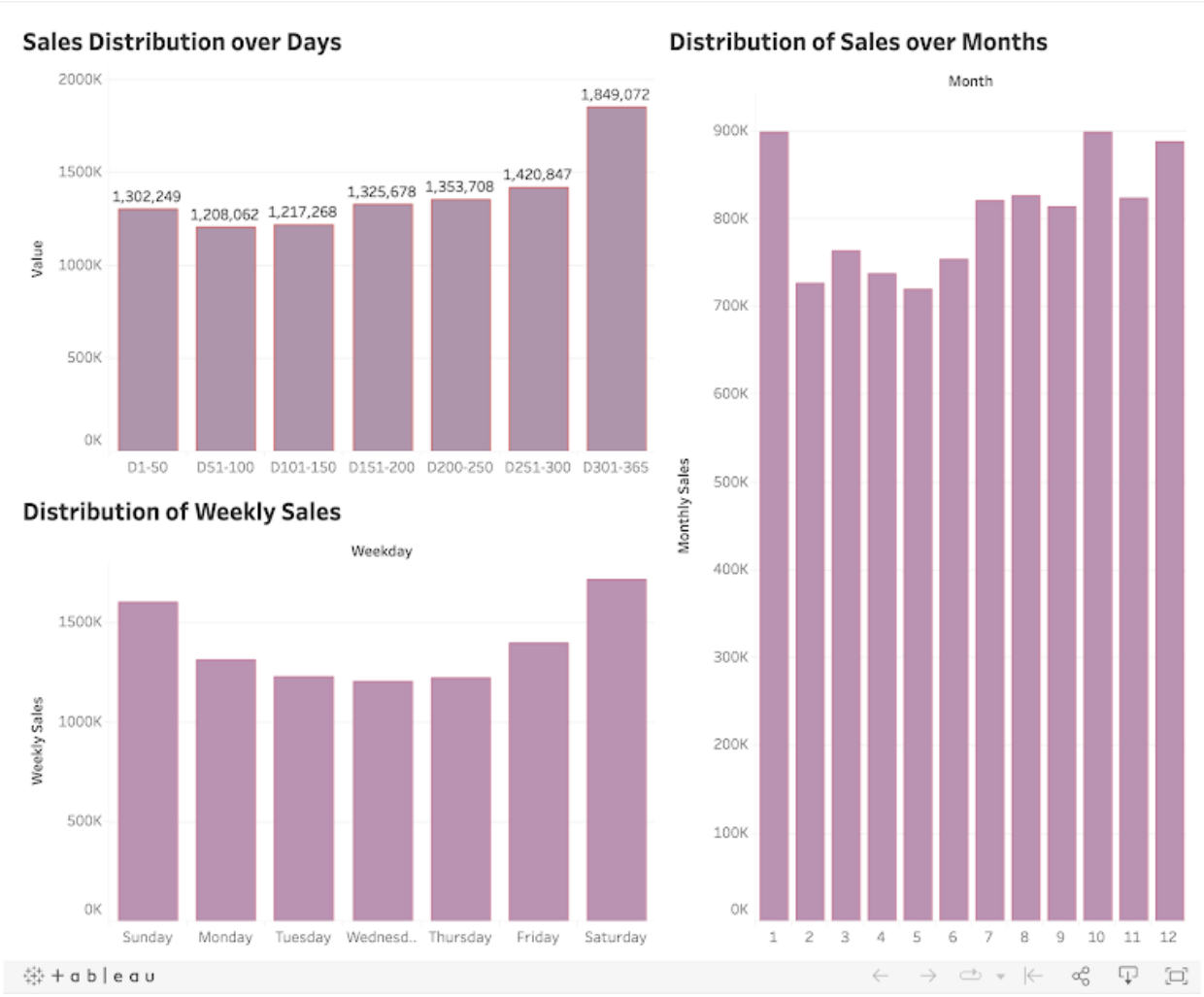
- **Figure - 1 Sales distribution across states, categories, departments**



The dataset_02 consists of stores from California, Texas and Wisconsin. The total sales from each of these locations are shown in the first sheet. It is interpreted that the highest sales are from CA (44.56 %) and the lowest is from WI (24.90 %) in a year.

The sales in the **Foods** category are 70.89%, the largest among all the other categories. The lowest sales are in the **Hobbies** categories in all the stores in the three locations.

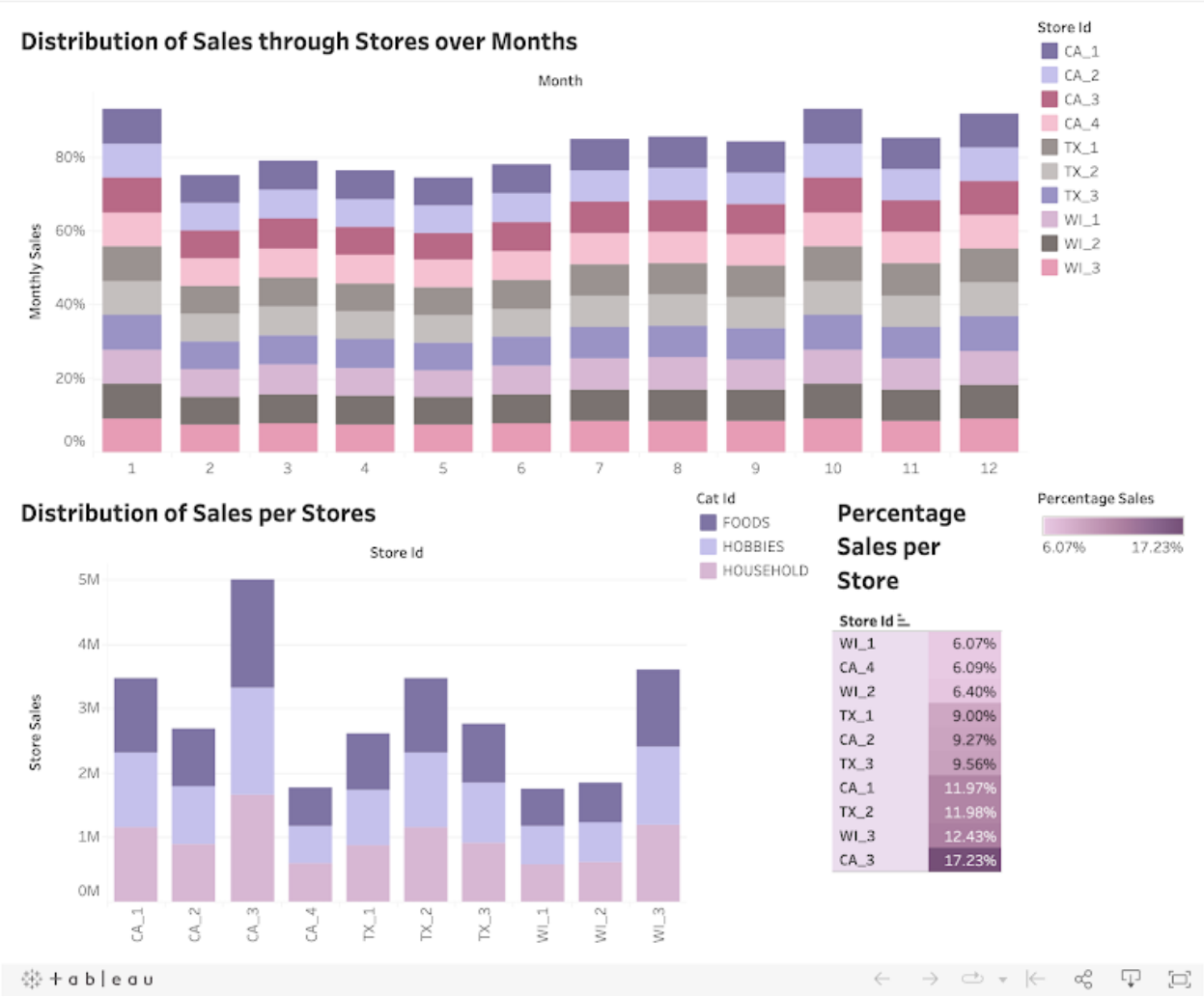
- **Figure - 2 Sales distribution over days, weeks and months**



The sales distribution plots according to days, months and weeks are shown. It is interpreted that the highest sales are done in the last 65 days of the year and people have purchased the most goods on Saturdays and in the month of October.

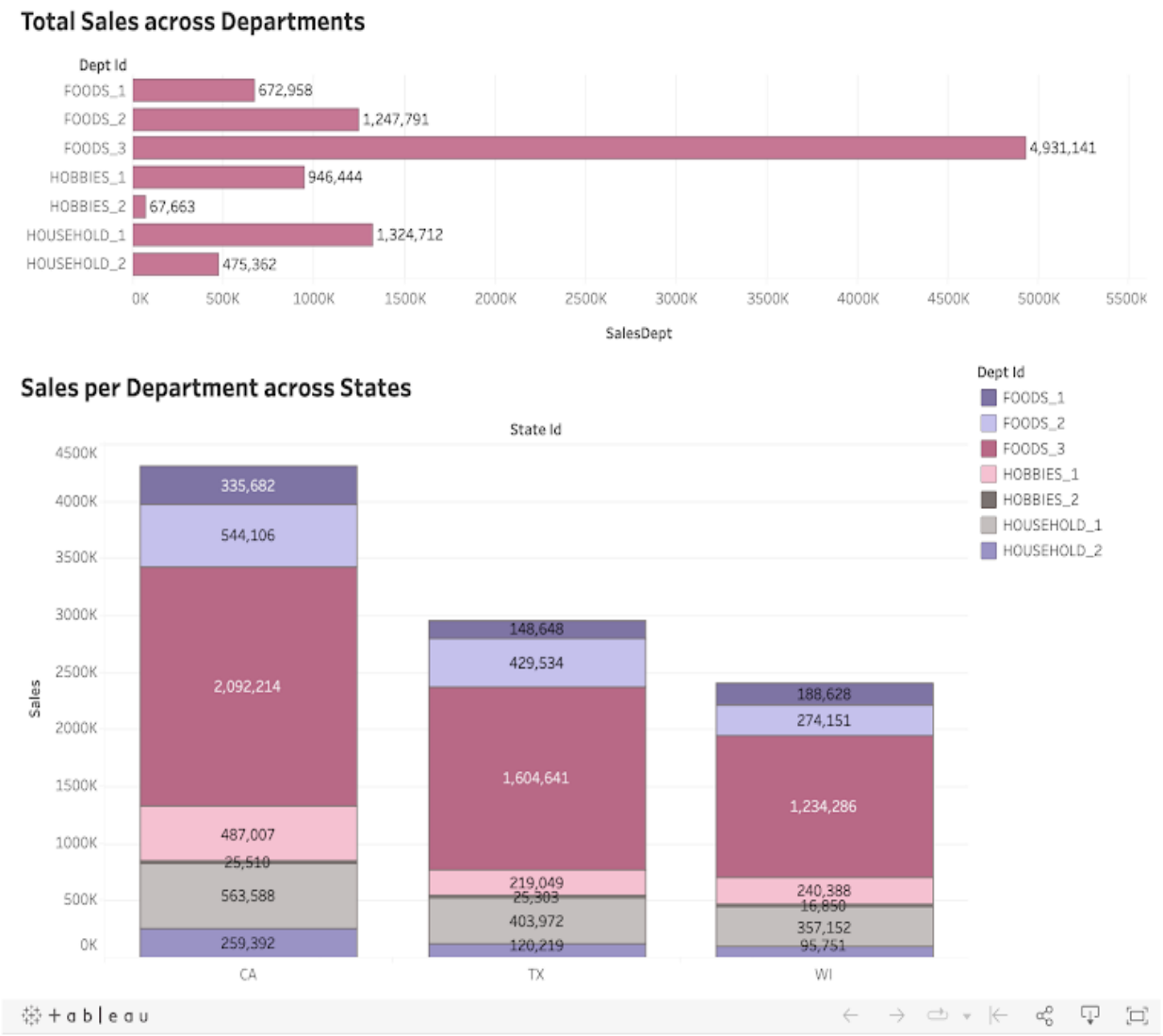
We encountered a challenge to show the plot for the sales distribution over days as there were 365 days. Showing sales of 365 days in a bar plot would have been difficult to understand and the plot would not have been readable. Hence, to simplify the plot a little, we grouped the days in 50 using the **'Group'** feature in Tableau.

- **Figure - 3 Sales distribution across stores, across stores over months and percentage sales per store**



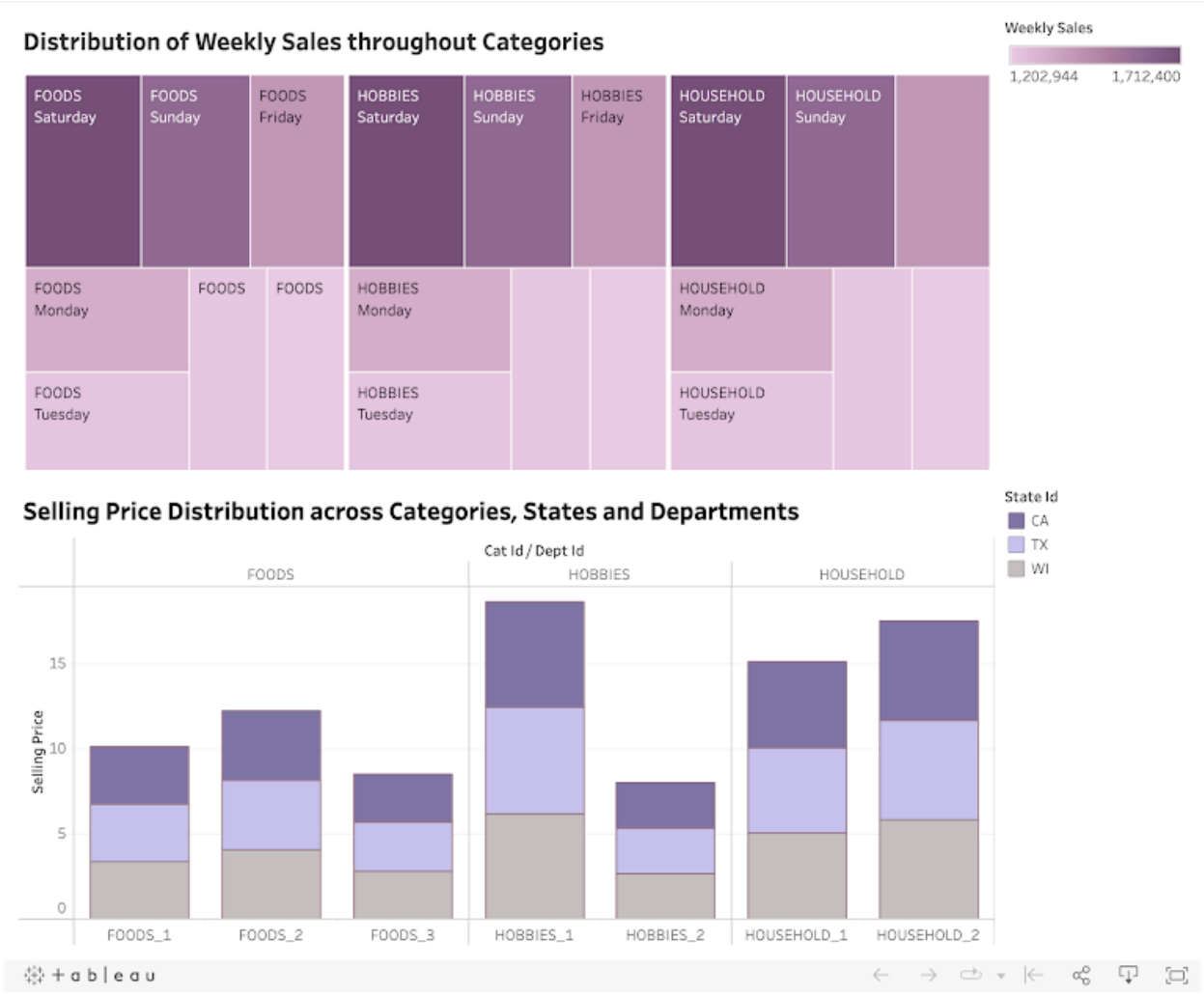
The distribution of sales in a year in each store is shown over the months and sales distribution in each store is also plotted. The largest sales are from the store CA_3 (17.23%) and the lowest sales are from WI_1 (6.07%).

- **Figure - 4 Sales across departments and per department across states**



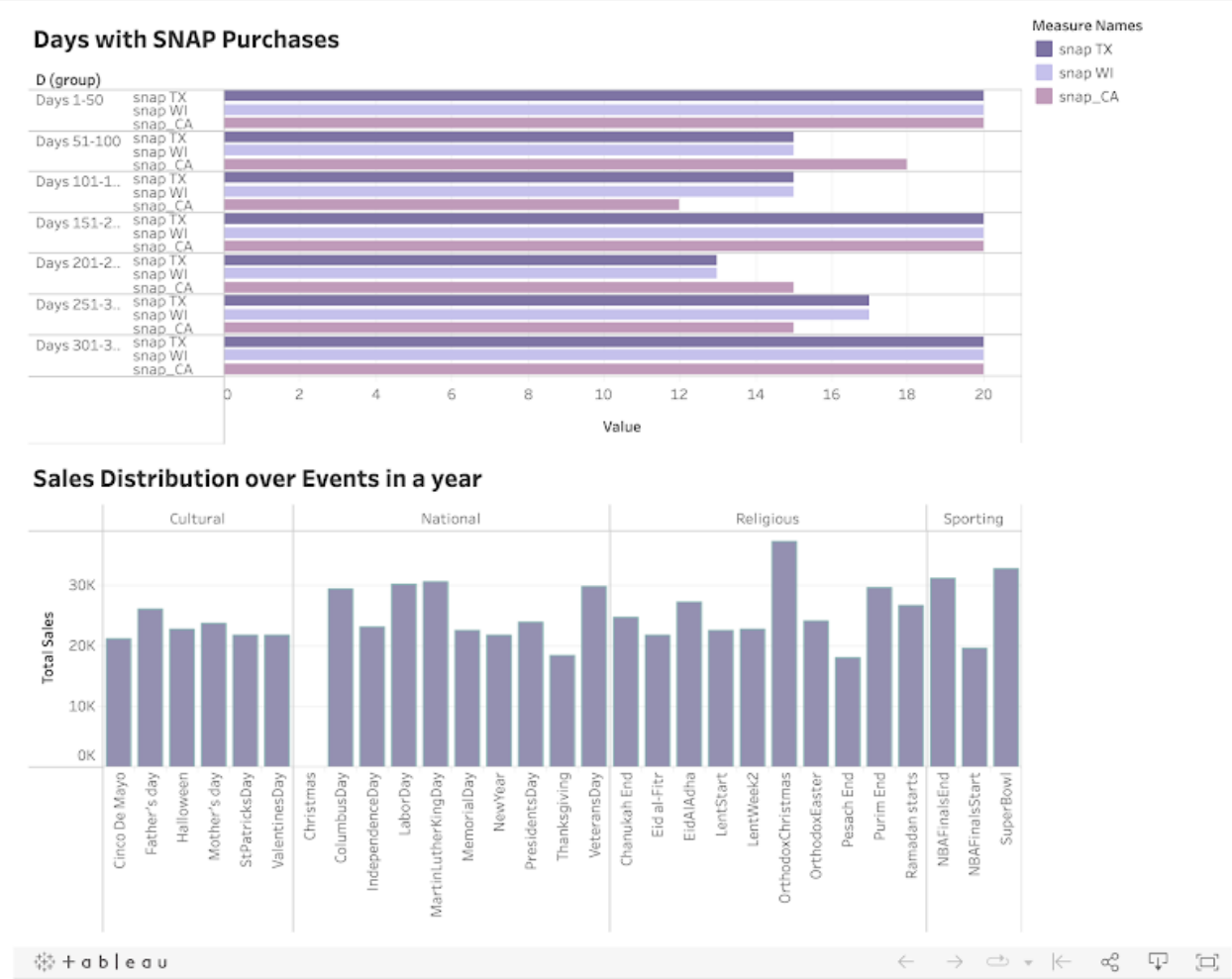
The total sales in each department are shown and the sales in each department across states is also plotted. It is interpreted that the largest sales is in **Foods_3** and the lowest sales are in **Hobbies_2**. As we noticed from the previous plots that California has the largest sales and from here we can conclude that the **Foods_3** department contributes the most in the sales from CA and **Hobbies_2** department contributes the least**.**

- **Figure - 5 Weekly sales distribution over categories and selling price distribution across categories, states and departments**



Dashboard 5 consists of 2 graphs. The topmost graph wherein depicts the distribution of weekly sales throughout categories, while the bottom graph shows the distribution of selling prices across categories, stores, and departments. It can be observed that most units of items from all the departments were sold over the weekends. It is worth noting how the selling price varies across departments based on categories.

- **Figure - 6 Plots to show days with SNAP purchases**



The above figure illustrates two graphs, viz, the number of days that saw SNAP purchases and the distribution of sales over the events in a year, on the top and bottom respectively. SNAP is called the Supplement Nutrition Assistance Program where the US federal government provides a nutrition assistance. Days with SNAP purchases are distributed somewhat evenly across the stores in the US. It can also be observed that the least and most sales happened on Christmas and Orthodox Christmas respectively.

2. Publish the Tableau dashboard on public server.

- **Tableau Dashboard URL:**

https://public.tableau.com/app/profile/pragya.sharma1552/viz/Group_02_Christian_Thube_Bolem_Sharma_Vora/Dashboard?publish=yes

Sprint - 2

2.2 Machine Learning model

2.2.1 CECS551_dataset_01

➔ Dataset_01: Regression, Classification

➔ Dataset_02: Time Series

1. **Linear regression, Ridge Regression, XGBoost, Random Forest Regressor, Arima with performance matrix(Mean squared error, Mean absolute error and R2 value)**

First we created a merged dataset named output.xlsx which contained all the merged data of 35 stores so that we can use it throughout our project.

	Unnamed: 0	Store	Dept	Weekly_Sales	IsHoliday	Type	Size	Temperature	gas_price	discount_promotional	discount_clearance	discount_damaged_good	di
0	0	1	1	24924.50	False	A	151315	59.33	3.360	9667.50	268.29	0.60	
1	1	1	1	46039.49	True	A	151315	51.65	3.409	8687.47	1594.87	2.20	
2	2	1	1	41595.55	False	A	151315	52.39	3.510	2706.87	3128.74	1.88	
3	3	1	1	19403.54	False	A	151315	60.12	3.555	6129.28	1802.84	0.00	
4	4	1	1	21827.90	False	A	151315	61.65	3.630	3552.58	601.32	0.00	

For cleaning the data we started with converting the Date column which is in integer to data time format so that we can access Day Month and Year.

```
df['Date'] = pd.to_datetime(df['Date'])
df['Day'] = df['Date'].dt.weekday
df['Month'] = df['Date'].dt.month
df['Year']  = df['Date'].dt.year
```

After converting to date format we dropped the Date column. Also we saw that we had a lot of null values and tried dropping them but due to more number of Null values the dataset became too small so rather than dropping we decided to convert all Null values to 0.

```
df.drop('Date',axis=1,inplace=True)
df.fillna(0,inplace=True)
```

❖ **Data Pre processing:**

Store	Dept	Weekly_Sales	IsHoliday	Type	Size	Temperature	gas_price	discount_promotional	discount_clearance	c
1	1	24924.50	False	A	151315	59.33	3.360	9667.50	268.29	
1	1	46039.49	True	A	151315	51.65	3.409	8687.47	1594.87	
1	1	41595.55	False	A	151315	52.39	3.510	2706.87	3128.74	
1	1	19403.54	False	A	151315	60.12	3.555	6129.28	1802.84	
1	1	21827.90	False	A	151315	61.65	3.630	3552.58	601.32	

As we can see from the above pic that we have some columns which have very high values such as 24924 whereas some of the values are very low like 3.3 so we need to convert all those values in a smaller range.

To do this we used **PowerTransformer.**

Also we converted Type which was string to Integer as well as year which were 2010,2011,2012 to 0,1,2 respectively so that they wont be considered as outliers.

Store	Dept	Weekly_Sales	IsHoliday	Type	Size	Temperature	gas_price	discount_promotional	discount_clearance	discount_damaged_good	dis
1	1	0.633019	-0.275106	1	0.22807	0.434446	0.383330	1.566556	0.233706	-0.189748	
1	1	1.440158	3.634961	1	0.22807	0.203730	0.440705	1.561992	0.856296	-0.038212	
1	1	1.283860	-0.275106	1	0.22807	0.226273	0.561469	1.500941	1.271129	-0.062360	
1	1	0.384527	-0.275106	1	0.22807	0.457793	0.616363	1.545995	0.922074	-0.283214	
1	1	0.496418	-0.275106	1	0.22807	0.502821	0.709349	1.517219	0.451048	-0.283214	

Next step was to remove all the outliers which were distorting the plots so we used quantiles to do that.

```
Q1 = df['Unemployment'].quantile(0.25)
Q3 = df['Unemployment'].quantile(0.75)
IQR = Q3-Q1

df = df[df['Unemployment'] >= Q1-1.5*(IQR)]
df = df[df['Unemployment'] <= Q3+1.5*(IQR)]
df.shape
```

```
(323714, 18)
```

```
Q1 = df['Temperature'].quantile(0.25)
Q3 = df['Temperature'].quantile(0.75)
IQR = Q3-Q1

df = df[df['Temperature'] >= Q1-1.5*(IQR)]

df = df[df['Temperature'] <= Q3+1.5*(IQR)]
df.shape
```

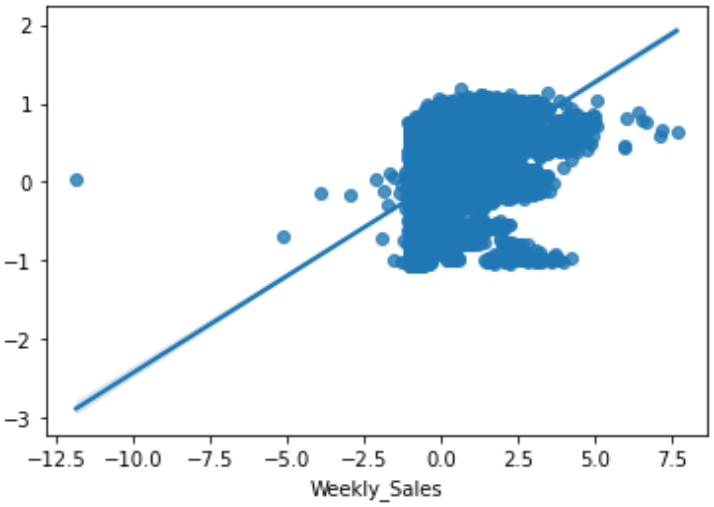
```
(323027, 18)
```

Now after all data cleaning and data preprocessing is done we started with modeling:

Note: All the analysis has been done with keeping all external features in the dataset.

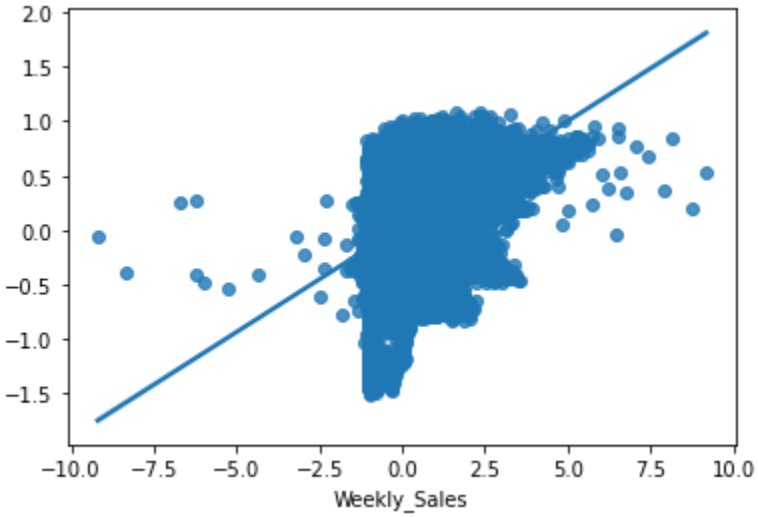
- **Linear Regression: Store1_10:**

```
MSE: 0.7880411373245544
MAE: 0.6246598472526079
R2 : 0.2508821045622597
```



- **Linear Regression: Store11_35:**

```
MSE: 0.7393404224768703
MAE: 0.6253634598854412
R2 : 0.19675135094369778
```

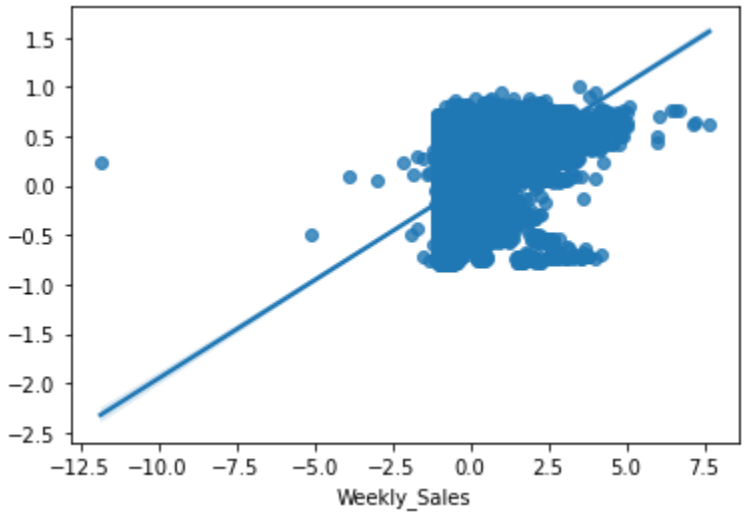


Observation:

- ❖ R2 value should be between 0 to 1 (0.25 and 0.19 in our case) and value closer to 0 states that it doesn't explain any variations.
- ❖ It can be seen from the graph that the predicted values(scatter plot) are far from actual value(line plot). In such cases we can't rely on this model.

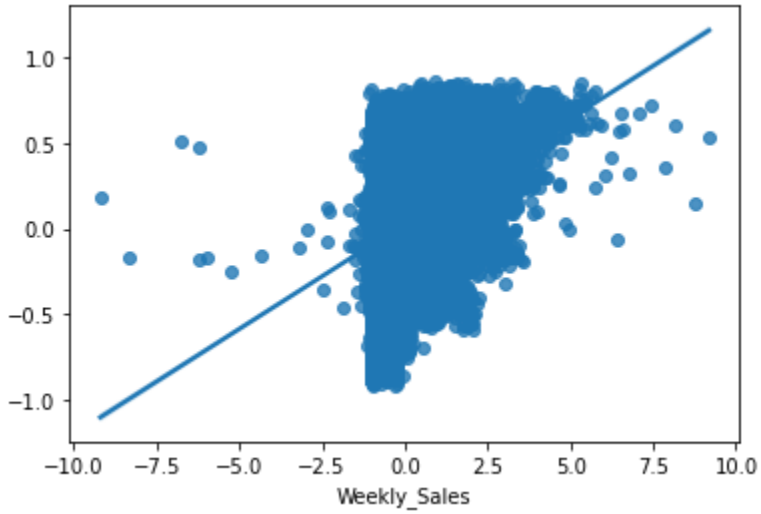
- **Ridge Regression: Store1_10:**

```
MSE: 0.8161764441536359
MAE: 0.6495531298298004
R2 : 0.22413646801991738
```



- **Ridge Regression: Store11_35:**

```
MSE: 0.79236680686
MAE: 0.6698936019286715
R2 : 0.1391413916810934
```



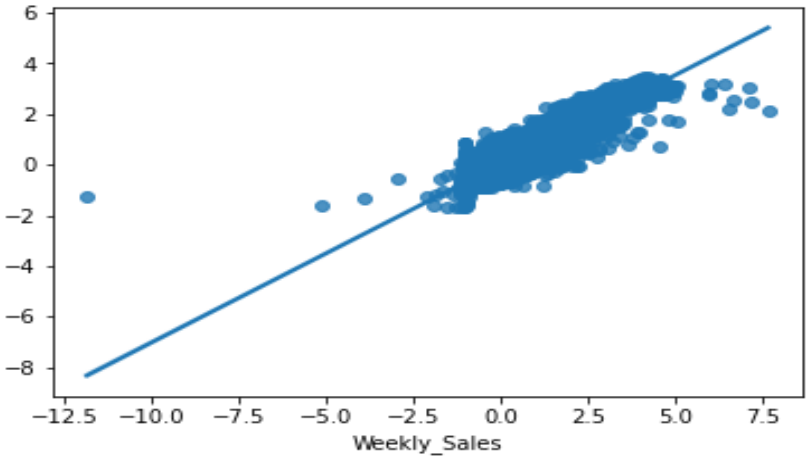
Observation:

- ❖ R2 value should be between 0 to 1 (0.22 and 0.13 in our case) and value closer to 0 states that it doesn't explain any variations.

❖ It can be seen from the graph that the predicted values(scatter plot) are far from actual value(line plot). In such cases we can't rely on this model.

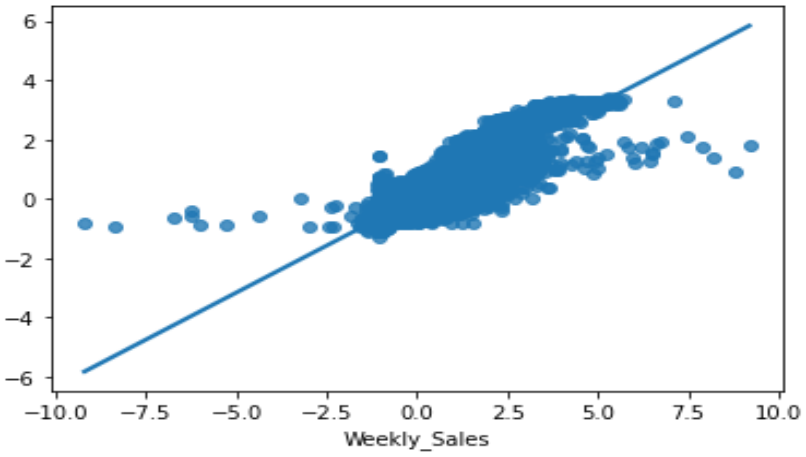
- **XGBoost: Store1_10:**

```
MSE: 0.20008958647788302
MAE: 0.31092388119724873
R2 : 0.8097933181125454
```



- **XGBoost: Store11_35:**

```
MSE: 0.2276660170837622
MAE: 0.3426508749719278
R2 : 0.7526546431129537
```



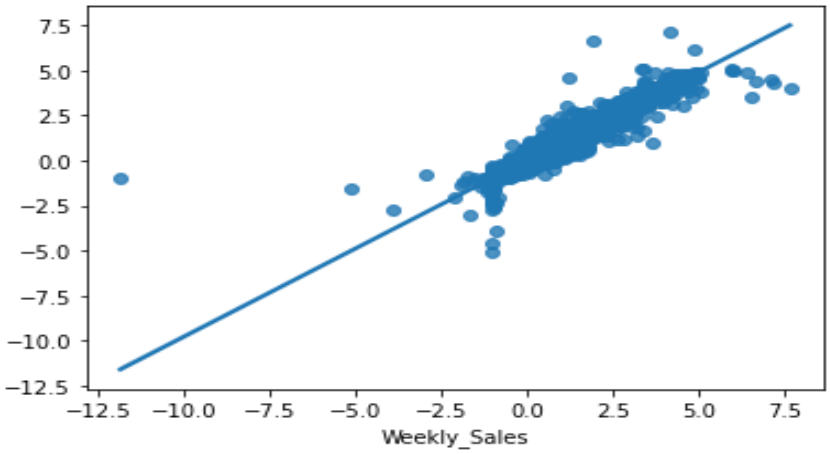
Observation:

❖ This model performs far more better than linear Regression and Ridge Regression.

- ❖ As we can see from R2 value which is close to 1 (0.81 and 0.75 in our case) it explains all the variation
- ❖ It can be seen from the graph that the predicted values(scatter plot) are clumped around the actual value(line) which shows that actual and predicted values are close to one another. In such cases we can rely on this model.

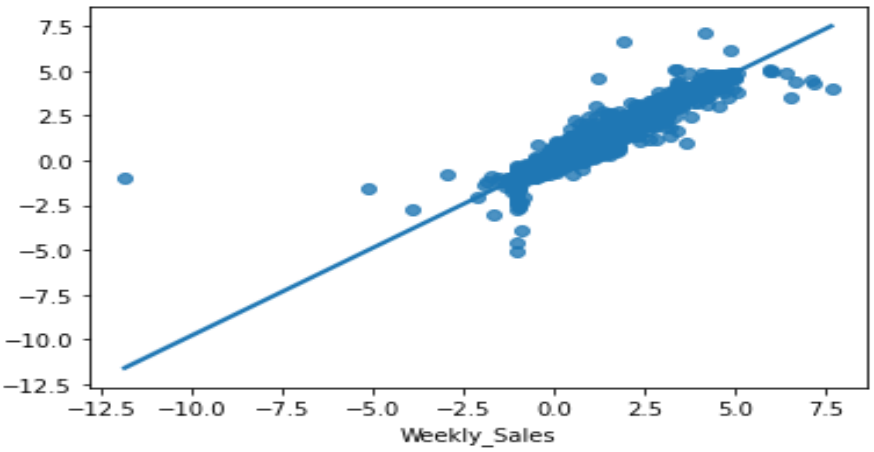
- **Random Forest Regressor: Store1_10:**

```
MSE: 0.03528361481900656
MAE: 0.08192329053644175
R2 : 0.9664591275445508
```



- **Random Forest Regressor: Store11_35:**

```
MSE: 0.03129321660979816
MAE: 0.08044913974656613
R2 : 0.9660018129642668
```

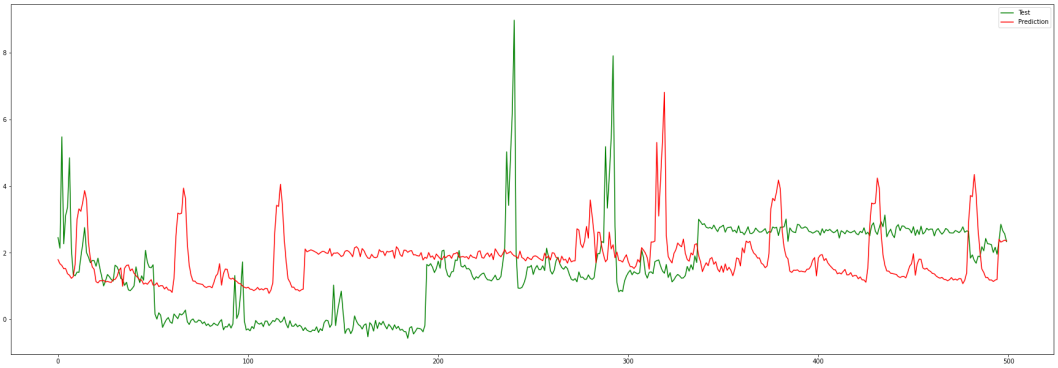


Observation:

- ❖ This is the best model.
- ❖ As we can see from R2 value which is close to 1 (0.96 and 0.96 in our case) it explains all the variation
- ❖ It can be seen from the graph that the predicted values(scatter plot) are clumped around the actual value(line) which shows that actual and predicted values are close to one another. In such cases we can rely on this model.

- **Random Forest Regressor: Store1_10:**

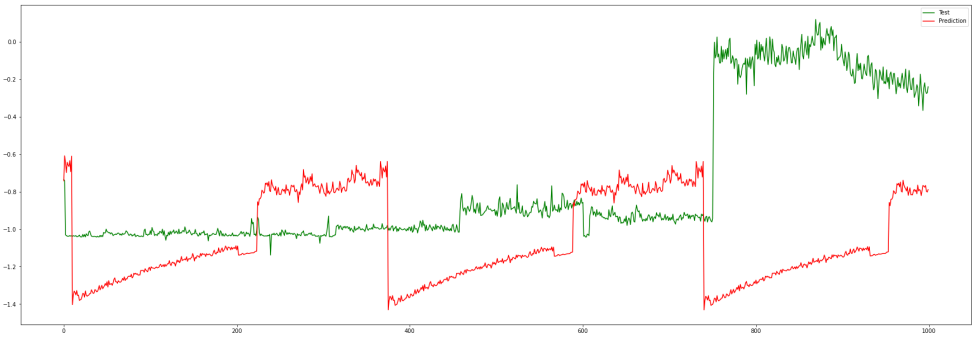
```
1. ADF :  -14.74187333715586
2. P-Value :  2.563295044567325e-27
3. Num Of Lags :  64
4. Num Of Observations Used For ADF Regression: 96443
5. Critical Values :
        1% :  -3.430417806622145
        5% :  -2.861569969452484
        10% :  -2.5667859516929794
```




```
MSE: 30.633598385018153
MAE: 4.902795855140299
R2 : -20.788963944926333
```

- **Random Forest Regressor: Store11_35:**

```
1. ADF :  -23.046542733740555
2. P-Value :  0.0
3. Num Of Lags :  74
4. Num Of Observations Used For ADF Regression: 226444
5. Critical Values :
        1% :  -3.4303788785489138
        5% :  -2.861552763944722
        10% :  -2.5667767937874477
```



```
MSE: 1.8384723934543519
MAE: 0.9341963836686561
R2 : -0.8047079119326921
```

Observation:

- ❖ If p< 0.05 ; Data is stationary
- ❖ if p>0.05; Data is not stationary
- ❖ Larger p value could indicate presence of certain trends(varying mean) or seasonality as well.
- ❖ Values of p for both the cases is less than 0.05, hence they are stationary. But values of R2 shows that the model is not reliable

- **Model performance metrics and comparison in report**

	Performance Matrix:Store 1_10	**Performance Matrix: Store11_35**
Linear Regression	MSE: 0.7880411373245544 MAE: 0.6246598472526079 R2 : 0.2508821045622597	MSE: 0.7393404224768703 MAE: 0.6253634598854412 R2 : 0.1967513509436977
Ridge Regression	MSE: 0.8161764441536359 MAE: 0.6495531298298004 R2 : 0.2241364680199173	MSE: 0.79236680686 MAE: 0.6698936019286715 R2 : 0.1391413916810934
XGBoost	MSE: 0.2000895864778830 MAE: 0.3109238811972487 R2 : 0.8097933181125454	MSE: 0.2276660170837622 MAE: 0.3426508749719278 R2 : 0.7526546431129537
Random Forest Regressor	MSE: 0.03528361481900656 MAE: 0.0819232905364417 R2 : 0.9664591275445506	MSE: 0.03129321660979816 MAE: 0.0804491397465661 R2 : 0.9660018129642668
ARIMA Model	MSE: 30.633598385018153 MAE: 4.902795855140299 R2 : -20.788963944926333	MSE: 1.8384723934543519 MAE: 0.9341963836686561 R2 : -0.8047079119326921

In stores.csv, we have a feature “type”, i.e., three types of stores – A (Super center), B (Discount center), and C (Neighborhood markets). Now, based on the given features in the dataset, can we predict the store type? Please consider only first 10 stores for this problem statement.

2. **Consider the problem statement as a multi-label classification problem. Use the below classification algorithms and perform hyper-parameter tuning for the Deep Learning models.**
    - **Ensemble models (3 statistical methods)**
    - **Recurrent neural network (RNN)**
    - **Convolutional Neural Networks (CNN)**

- **Ensemble models (3 statistical methods)**

For the Ensemble models, we have used three different models as below:

1. Random Forest
2. Bagging
3. Boosting(AdaBoost Classifier, XGBoost Classifier, LGBM Classifier)

At first, the required libraries were imported for the classifiers. Let's check the percentage of Type C stores before preprocessing our data.

```
Percentage of C type Stores 10.104371753208245
```

Next, pre-processing of the data has been done and the separation of the target variable Type has been done.

```
data = data_.dropna()
y = data.pop('Type')
data_ = data.drop('Date',axis=1)
X = data_
X_ = X.to_numpy().astype('float32')
```

Next the usage of the LabelEncoder to convert each value in categorical column to numerical value

```
le = LabelEncoder()
transformed = le.fit_transform(y.to_numpy().reshape(-1, 1))
```

Next, the usage of the MinMaxScaler is being done to scale all the features for a range and the scaled data is being stored in X_

```
scaler = MinMaxScaler()

data_scaled = scaler.fit_transform(X_)

X__ = data_scaled
```

Then, we have separated the train_test_split data like below:

```
X_train, X_test, y_train, y_test = train_test_split(X__, transformed, test_size=0.20)
```

- **Random Forest Classifier**

Firstly, We simply trained the model on our training dataset without applying any hyper-tuning parameters. Based on the results, we can see the accuracy score of the model.

After applying the tuning parameters the accuracy of the model increased by some extent. It recommends gini as splitting criteria with max depth of 60 and 50 n _estimators

```
Accuracy Score - Random Forest - Default: 0.6947656259208185
```

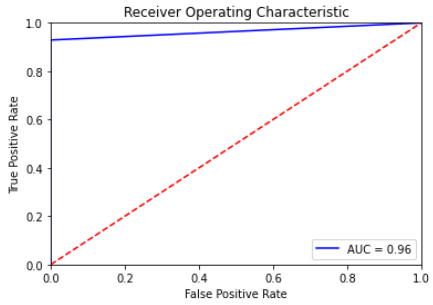
```
Best Parameter for Tunning:  {'criterion': 'entropy', 'max_depth': 160, 'min_samples_split': 1e-05, 'n_estimators': 50}
Accuracy Score - Random Forest - Tunned Model: 0.8400000000000001
```

The below figure depicts the confusion matrix for the random forest classifier indicating that most of the values predicted are correct.

	Predicted_Type_A	Predicted_Type_B	Predicted_Type_C
Type A	41499	3090	0
Type B	1332	37133	0
Type C	48	92	1649

	precision	recall	f1-score	support
1	0.97	0.93	0.95	44589
2	0.92	0.97	0.94	38465
3	1.00	0.92	0.96	1789
accuracy			0.95	84843
macro avg	0.96	0.94	0.95	84843
weighted avg	0.95	0.95	0.95	84843

Precision, recall and f1-score for different hyper-tuning parameters can be seen in figure above.



The Roc-Auc curve for the Random forest model can be seen which shows that 0.96 is the roc_auc value for the current model.

- **Bagging**

Here, We trained the Bagging model on our training dataset without applying any hyper-tuning parameters. The result shows us that this model needs to be tuned based on the accuracy.

After applying the tuning parameters the accuracy of the model increased by some extent. It suggests that there should be 0.7 max features and 1 n estimators.

```
Accuracy Score - Bagging - Default: 0.7290642716547034
```

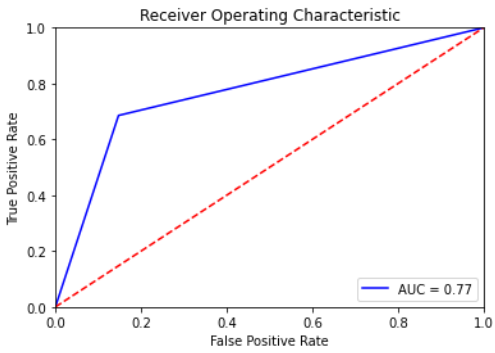
```
Best Parameter for Tunning:  {'max_features': 0.7, 'n_estimators': 1}
Accuracy Score - Bagging - Tunned Model: 0.85
```

The below figure depicts the confusion matrix for the random forest classifier indicating that most of the values predicted are correct.

	Predicted_Type_A	Predicted_Type_B	Predicted_Type_C
Type A	39539	5050	0
Type B	12088	26377	0
Type C	0	1789	0

	precision	recall	f1-score	support
1	0.77	0.89	0.82	44589
2	0.79	0.69	0.74	38465
3	0.00	0.00	0.00	1789
accuracy			0.78	84843
macro avg	0.52	0.52	0.52	84843
weighted avg	0.76	0.78	0.77	84843

Precision, recall and f1-score for different hyper-tuning parameters can be seen in figure above.



The Roc-Auc curve for the Random forest model can be seen which shows that 0.77 is the roc_auc value for the current model.

- **Boosting Techniques**

Now we have passed the respective parameters and tested the ensemble models.

```
model_params = {

    'ADA_Boost': {
        'model': AdaBoostClassifier(),
        'params': {
            'learning_rate': [0.2],
            'n_estimators': [100]
        }
    },
        'XGB_classifier' : {
        'model': XGBClassifier(),
        'params' : {
            'learning_rate': [0.1],
            'n_estimators': [1500]
        }
    },
         'LGBM_classifier' : {
        'model': LGBMClassifier(),
        'params' : {
            'learning_rate': [0.2],
            'n_estimators': [100],
        }

    }
}
```

Now, after checking the three models, we get the below scores for the models.

	model	best_score	best_params
0	ADA_Boost	0.998950	{'learning_rate': 0.2, 'n_estimators': 100}
1	XGB_classifier	0.999986	{'learning_rate': 0.1, 'n_estimators': 1500}
2	LGBM_classifier	1.000000	{'learning_rate': 0.2, 'n_estimators': 100}

We tried different parameters for the lgbm classifier to reduce the overfitting, but the lgbm score is quite high.

- **Convolutional Neural Networks (CNN)**

For, the CNN Model, first we are passing the input and the output values by adding two different layers to the model

We are passing the sigmoid function; it exists between 0 to 1 and it is a logistic activation function. Also, we are using the binary_crossentropy as it computes the cross-entropy loss between true labels and predicted labels.

```
def get_model(n_inputs, n_outputs):
    model = Sequential()
    model.add(Dense(20, input_dim=n_inputs, kernel_initializer='he_uniform', activation='relu'))
    model.add(Dense(n_outputs, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer='adam')
    return model
```

Now, in the evaluation model function, we are taking the X, y values and making them to list and for the evaluation procedure we are splitting the data using

```
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)
```

Later, we define the X-train and X_test and y_train and y_test values and we define the fitting for the model which we call at the end. Later. We rounded the probabilities and later and calculate the accuracy like below

```
acc = accuracy_score(y_test, yhat)
```

Now, since we have large data, we used the below function to get the random values so that CNN is executed properly

```
cnn_data = cnn_data.sample(frac=1).reset_index(drop=True)
```

And below, we have converted the dataset to float32 so that we wont have issues with the dataset and then, dropped the date from the dataset as it is not convertible to float32 and is of no use.

```
y = cnn_data.pop('Type')

data_ = cnn_data.drop('Date',axis=1)

X = data_
```

And Later, we have used One-hot encoding to convert the categorical values to dummy numerical values for the model.

Finally, we have evaluated the model and passed the X_, transformed_ values to the model and this is the below results

```
results = evaluate_model(X_, transformed_)
# summarize performance
print('Accuracy: %.3f (%.3f)' % (mean(results), std(results)))
```

```
7/7 [==============================] - 0s 2ms/step
>0.910
7/7 [==============================] - 0s 2ms/step
>0.870
Accuracy: 0.858 (0.054)
```

2.2.1 CECS551_dataset_01

Task 1: Design a machine learning model to make accurate predictions for product sales for next 10 days in advance (the data set includes daily unit sales per product) and compare the performance of different machine learning algorithms.

Background: The dataset also involves external variables, for example, calendar-related information and selling prices. Thus, apart from the past unit sales of the products and the corresponding timestamps (e.g., date, weekday, week number, month, and year), there is also information available about: special events and holidays (e.g., Super Bowl, Valentine's Day, and Orthodox Easter), organized into four classes, namely Sporting, Cultural, National, and Religious.

Selling prices, provided on a week-store level (average across seven days). If not available, this means that the product was not sold during the week examined. Although prices are constant on a weekly basis, they may change with time.

A. Perform data preprocessing and exploratory data analysis.

- **Data Preprocessing:**

The dataset 02 consists of four tables of data, sales_train_validation, sales_train_evaluation, calendar and sell_prices.

The data processing of sales_train_validation is done to obtain the training data.

```
print(salestv_data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30490 entries, 0 to 30489
Columns: 371 entries, id to d_365
dtypes: int64(365), object(6)
memory usage: 86.3+ MB
None
```

There are in total 30490 entries in this data and it consists of sales from d_1 to d_365, that is, the sales for 365 days (1 year).

The memory used in this data file is 86.3+ MB. Downcasting is performed on this dataset in order to save the memory and optimize its usage.

```
print(salestv_data_downcast.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30490 entries, 0 to 30489
Columns: 371 entries, id to d_365
dtypes: int16(365), object(6)
memory usage: 22.6+ MB
None
```

After performing downcasting, the memory usage is significantly reduced and optimized to 22.6+ MB.

The calendar dataset consists of information about the dates on which the products are sold. It also shows the dates of events if there are any and the SNAP analysis which is a binary variable (0 or 1) indicating whether the stores allow the SNAP purchases on the examined date.

```
cal_data.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

This dataset consists of null values in event name columns. These null values are handled through imputation and all the null values are replaced by the string “no_event”.

```
cal_data.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	no_event	no_event	no_event	no_event	0	0	0
1	2011-01-30	11101	Sunday	2	1	2011	d_2	no_event	no_event	no_event	no_event	0	0	0
2	2011-01-31	11101	Monday	3	1	2011	d_3	no_event	no_event	no_event	no_event	0	0	0
3	2011-02-01	11101	Tuesday	4	2	2011	d_4	no_event	no_event	no_event	no_event	1	1	0
4	2011-02-02	11101	Wednesday	5	2	2011	d_5	no_event	no_event	no_event	no_event	1	0	1

All the dataframes of the dataset 02 are optimized further to make a final dataset which contains all the necessary values in one dataframe for both the train and test data.

Updated calendar and updated train_validation data frames are combined on the feature ‘d’, that is, the days from d_1 to d_365. With this combined dataframe, sell_prices data is merged on the features "store_id", "item_id", "wm_yr_wk". This dataframe is stored in a csv file named “final_dataframe.csv”. This csv file now consists of the merged data that is further used to train the models.

Updated calendar and updated train_evaluation data frames are combined on the feature ‘d’, that is, the days from d_1 to d_365. With this combined dataframe, sell_prices data is merged on the features "store_id", "item_id", "wm_yr_wk". This dataframe is stored in a csv file named “final_dataframe_test.csv”. This csv file now consists of the merged data that is further used to test the models.

The train dataset is further stored in a dataframe named “data” and this dataframe is explored. Certainly it uses a significant amount of memory, that is, 979.4+ MB, which is reduced by eliminating extra features in feature engineering.

```
print(data.info())
```

```
Complete information about data Frame is:-

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5581600 entries, 0 to 5581599
Data columns (total 23 columns):
 #   Column        Dtype
---  ------        -----
 0   Unnamed: 0    int64
 1   id            object
 2   item_id       object
 3   dept_id       object
 4   cat_id        object
 5   store_id      object
 6   state_id      object
 7   d             object
 8   sales         int64
 9   date          object
 10  wm_yr_wk      int64
 11  weekday       object
 12  wday          int64
 13  month         int64
 14  year          int64
 15  event_name_1  object
 16  event_type_1  object
 17  event_name_2  object
 18  event_type_2  object
 19  snap_CA       int64
 20  snap_TX       int64
 21  snap_WI       int64
 22  sell_price    float64
dtypes: float64(1), int64(9), object(13)
memory usage: 979.4+ MB
```

```
print("There are these unique stores in this data=",data['store_id'].unique())
```

```
There are these unique stores in this data= ['CA_1' 'CA_2' 'CA_3' 'CA_4' 'TX_1' 'TX_2' 'TX_3' 'WI_1' 'WI_2' 'WI_3']
```

```
print("There are these sates in data=",data['state_id'].unique())
```

```
There are these sates in data= ['CA' 'TX' 'WI']
```

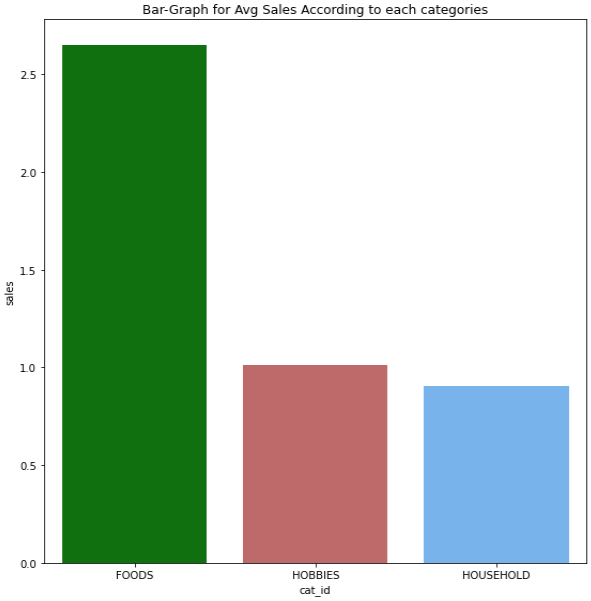
```
print("Unique values of wday features=",data['wday'].unique())
print("Unique values of weekday features=",data['weekday'].unique())
```

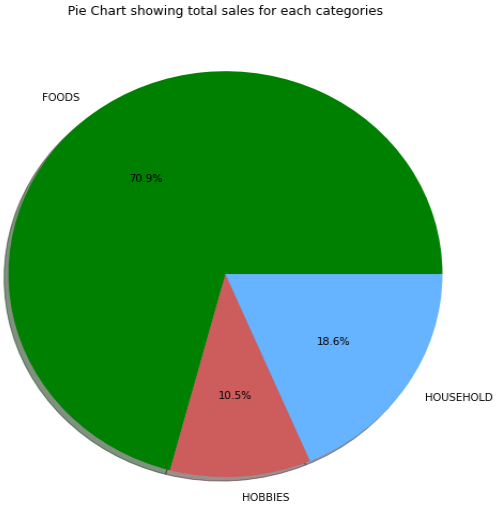
```
Unique values of wday features= [1 2 3 4 5 6 7]
Unique values of weekday features= ['Saturday' 'Sunday' 'Monday' 'Tuesday' 'Wednesday' 'Thursday' 'Friday']
```

It is inferred that there are 10 unique stores in the merged dataframe that are distributed across three states CA, TX, and WI. Here, 1 in wday corresponds to Saturday on weekdays, 2 for Sunday and so on.

- **Exploratory Data Analysis**

- **Figure - 1**

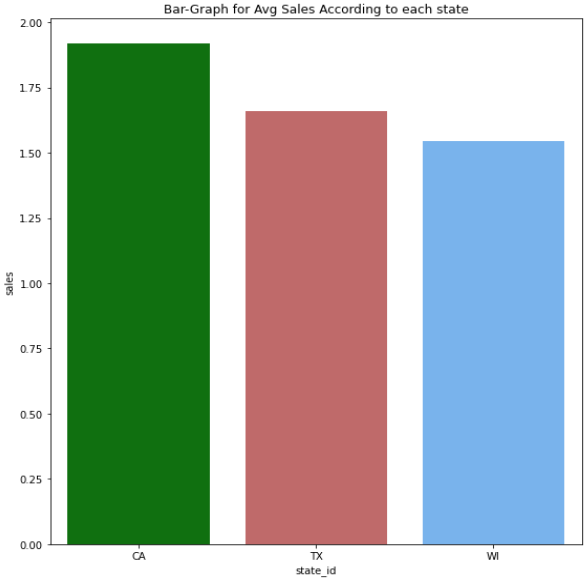


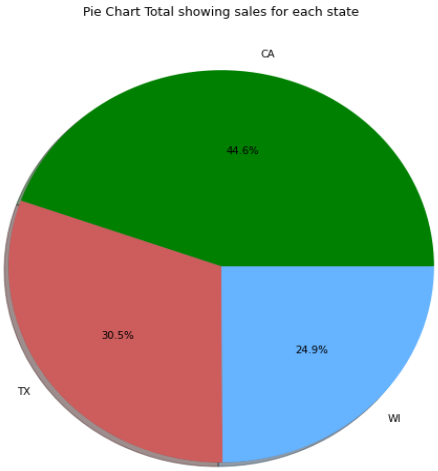


The bar graph is for average sales in each category and the pie chart shows the percentage of total sales in each category.

The sales in the **FOODS** category are 70.9%, the largest among all the other categories. The lowest sales are in the **HOBBIES** category in all the stores in the three locations.

- **Figure - 2**

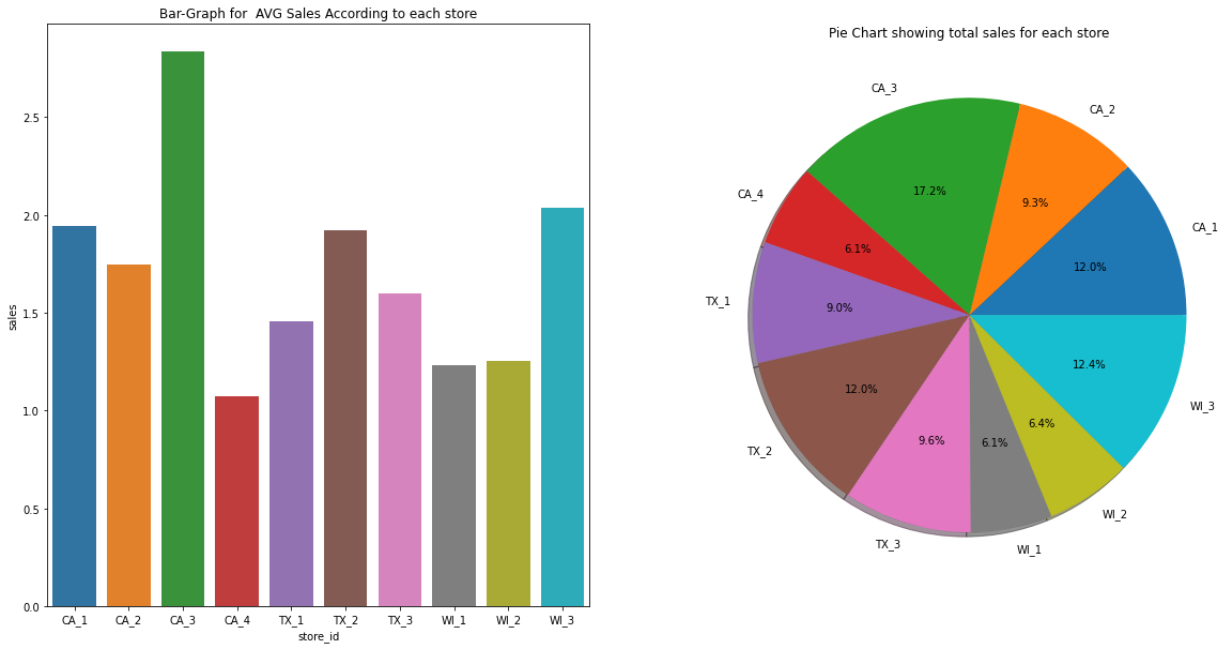




The bar graph shows the average sales according to each state and the pie chart shows the total sales according to each state.

The dataset_02 consists of stores from California (CA), Texas (TX) and Wisconsin (WI). It can be interpreted that the highest sales are from CA (44.6 %) and the lowest is from WI (24.90 %) in a year (2011-12).

- **Figure - 3**

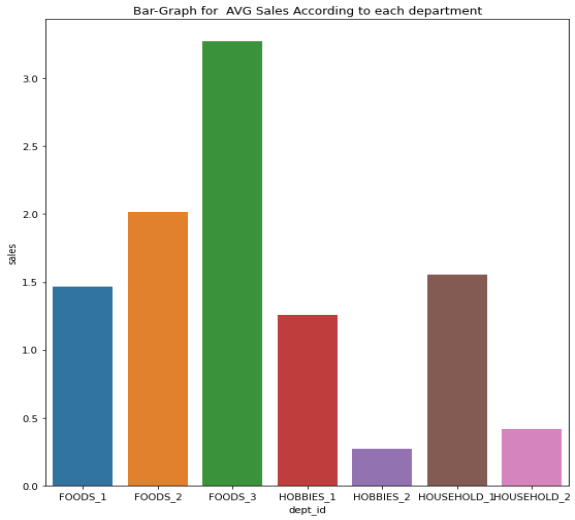


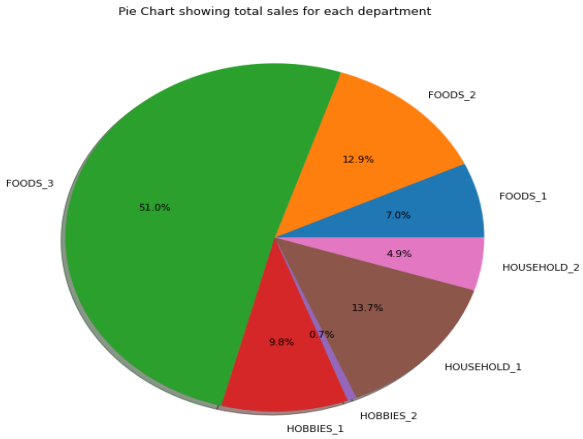
The bar graph shows the average sales according to each store and the pie chart shows the total sales according to each store.

The plots depict that there are in total 10 stores that are named as CA_1, CA_2, CA_3, CA_4, TX_1, TX_2, TX_3, WI_1, WI_2, WI_3.

The average and total sales are the maximum in the store CA_3. The total sales is the lowest in the store WI_1 and CA_4 with 6.1%.

- **Figure - 4**



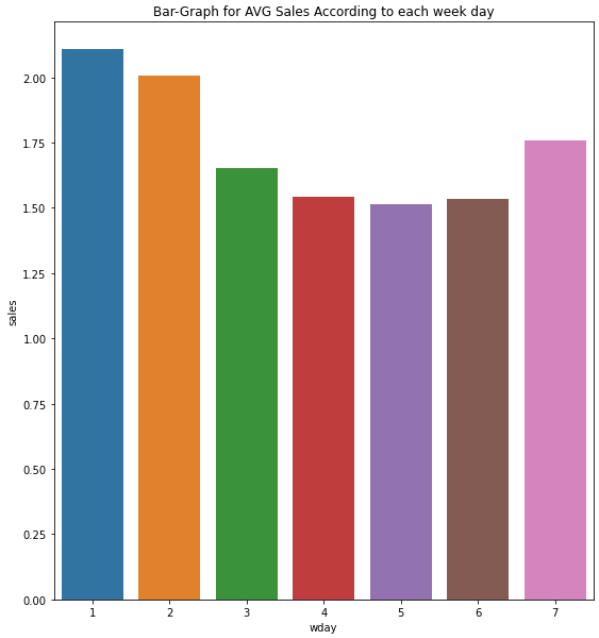


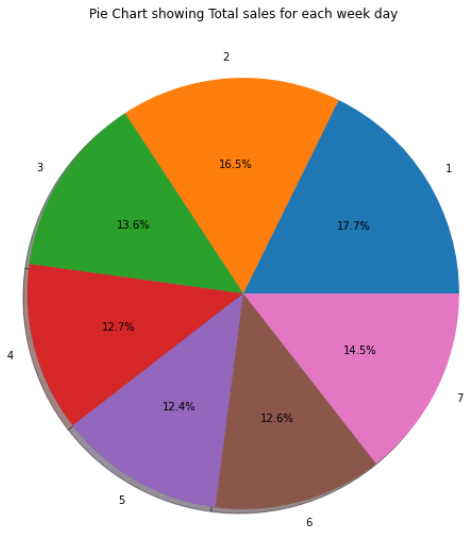
The bar graph shows the average sales according to each department and the pie chart shows the total sales according to each department.

It is interpreted that the largest sales is in **FOODS_3** and the lowest sales are in **HOBBIES_2**. As we noticed from the previous plots, California has the largest sales and from here we can conclude that the **FOODS_3** department contributes the most in the sales from CA and the Hobbies_2 department contributes the least.

It is also concluded that almost 50% of the sales are done by the **FOODS_3** department.

- **Figure - 5**



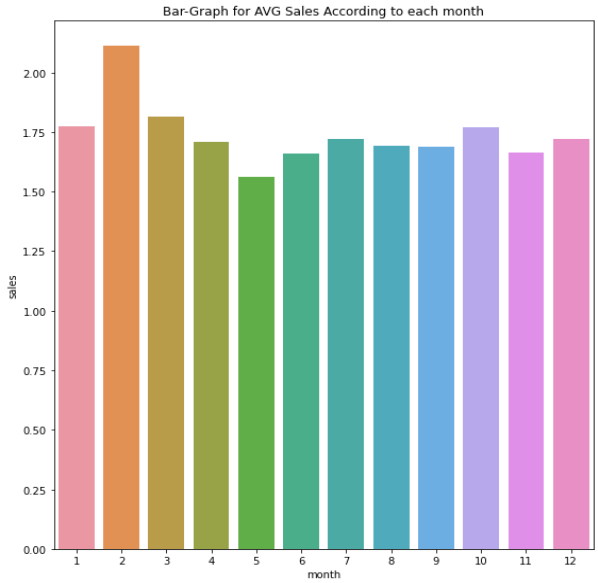


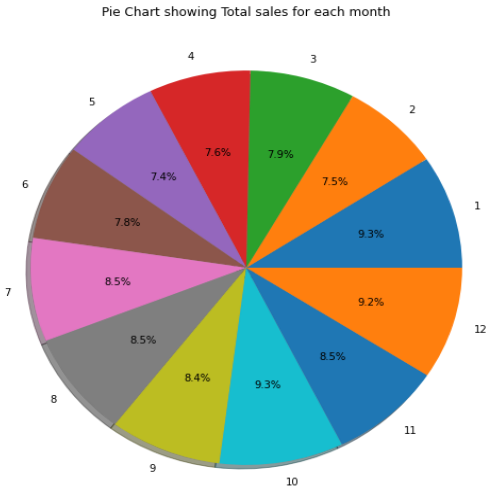
The bar graph shows the average sales according to each day in the week and the pie chart shows the total sales according to each day in the week.

It can be observed that most units of items from all the departments were sold over the weekends. It is worth noting how the selling price varies across departments based on categories.

Clearly sales are more on weekends than normal days. We tried to use this weekday feature while training the models.

- **Figure - 6**

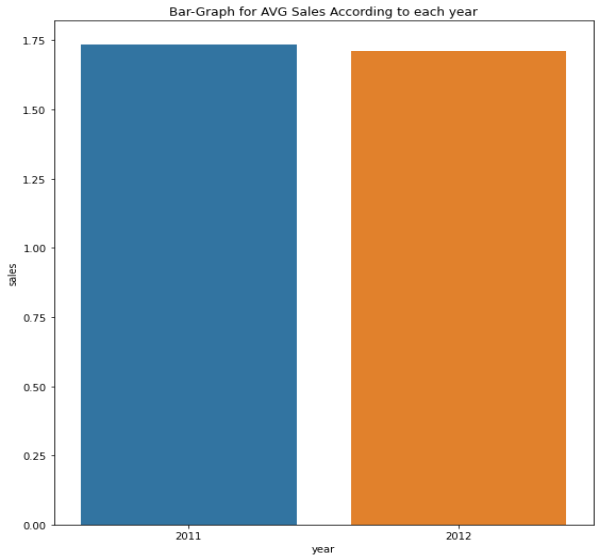


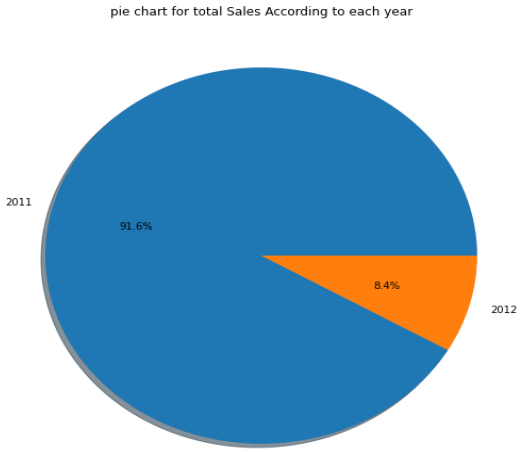


The bar graph shows the average sales in each month in the year of 2011-12. The pie chart shows the total sales in each month in the year of 2011-12. The month corresponding to 2, that is, February, has the most sales and the month of May corresponding to the numerical value 5 has the least sales.

It can also be observed that the least and most sales happened on Christmas and Orthodox Christmas respectively.

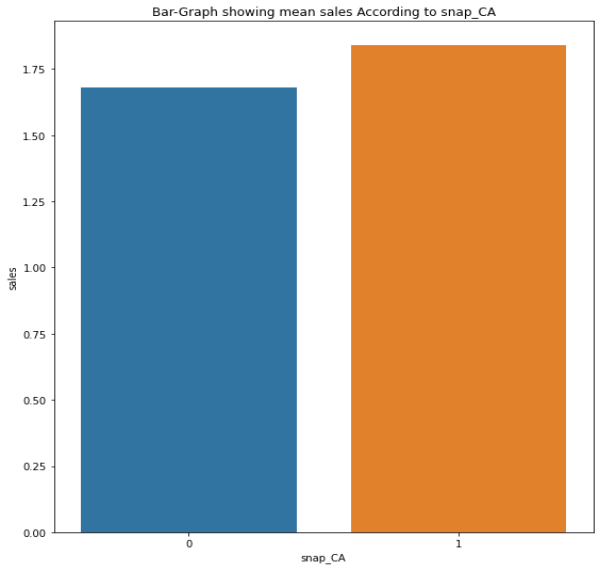
- **Figure - 7**

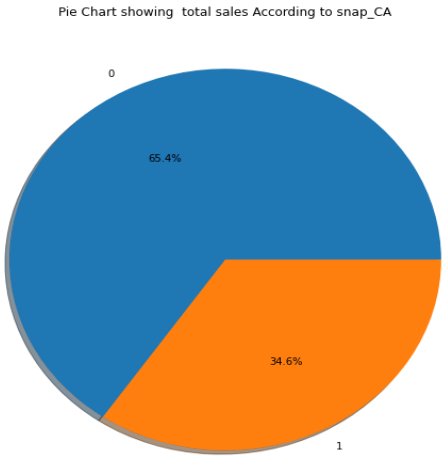




The above figure illustrates two graphs, viz, the number of days that saw SNAP purchases and the distribution of sales over the events in a year, on the top and bottom respectively. Days with SNAP purchases are distributed somewhat evenly across the stores in the US.

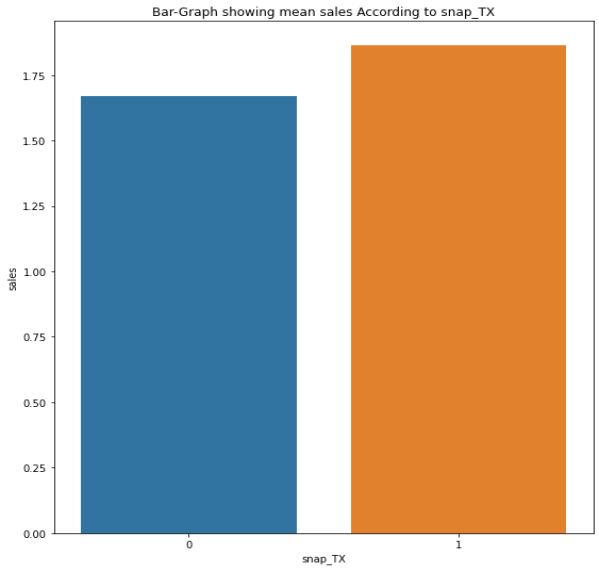
- **Figure - 8**

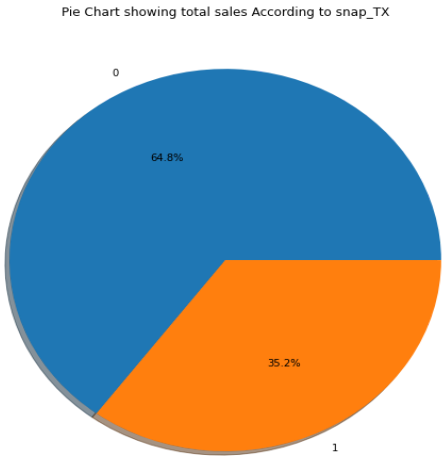




The bar graph shows the average sales according to SNAP purchases across all the stores in CA and the pie chart shows the total sales according to SNAP purchases across all the stores in CA. It is very evident that the SNAP program has led to more sales in CA.

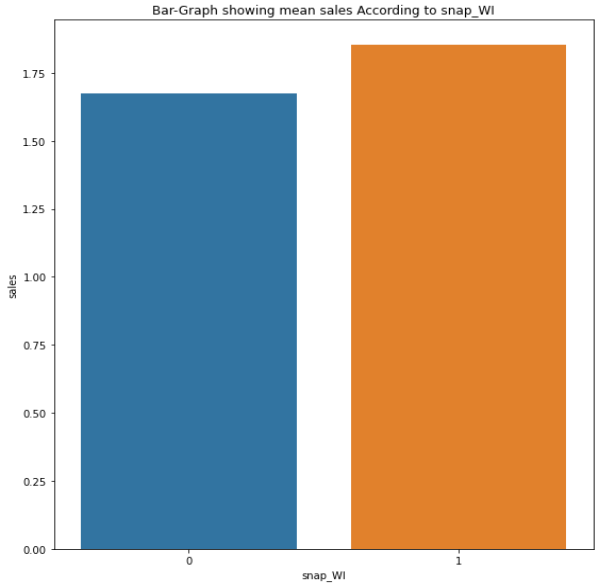
- **Figure - 9**

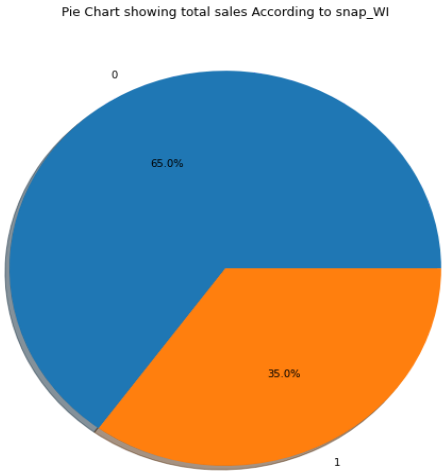




The bar graph shows the average sales according to SNAP purchases across all the stores in TX and the pie chart shows the total sales according to SNAP purchases across all the stores in TX. It is very evident that the SNAP program has led to more sales in TX.

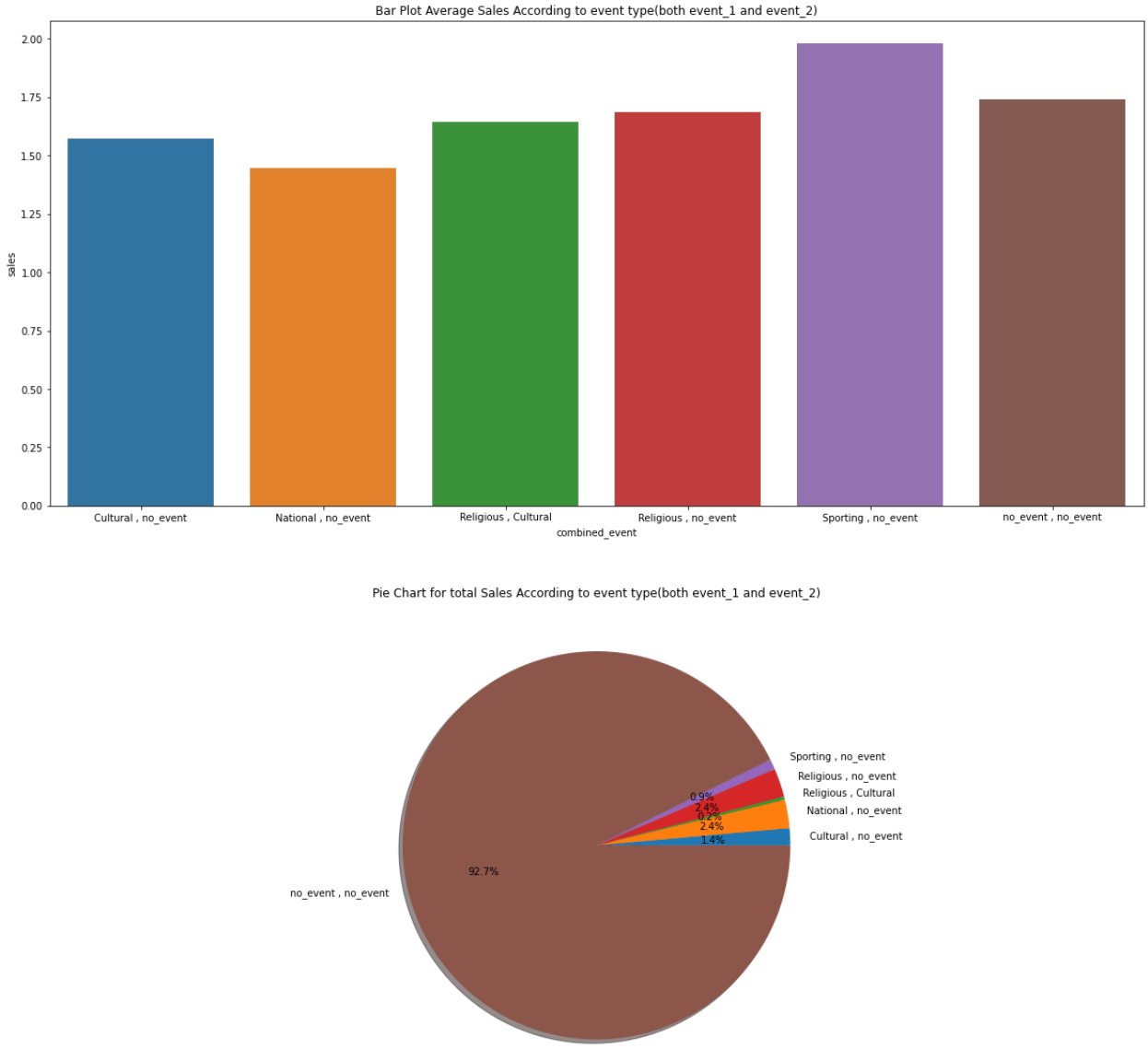
- **Figure - 10**





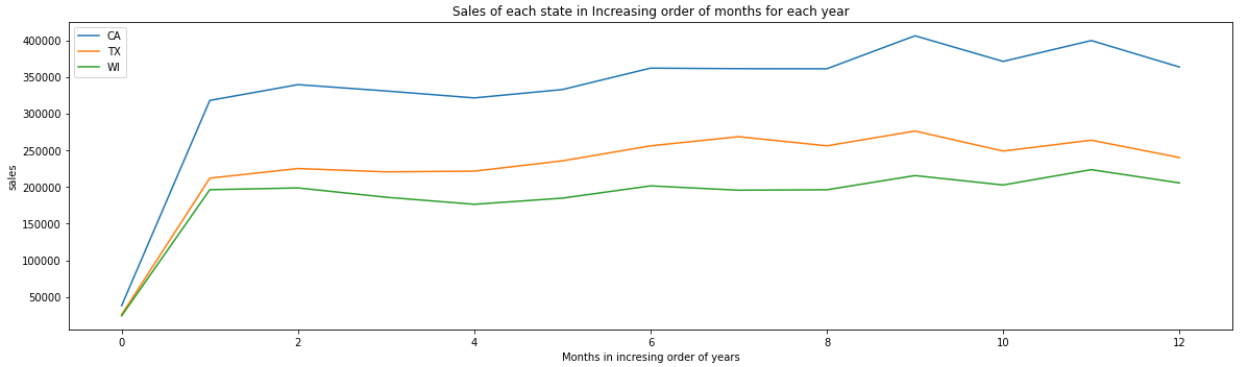
The bar graph shows the average sales according to SNAP purchases across all the stores in WI and the pie chart shows the total sales according to SNAP purchases across all the stores in WI. It is very evident that the SNAP program has led to more sales in WI.

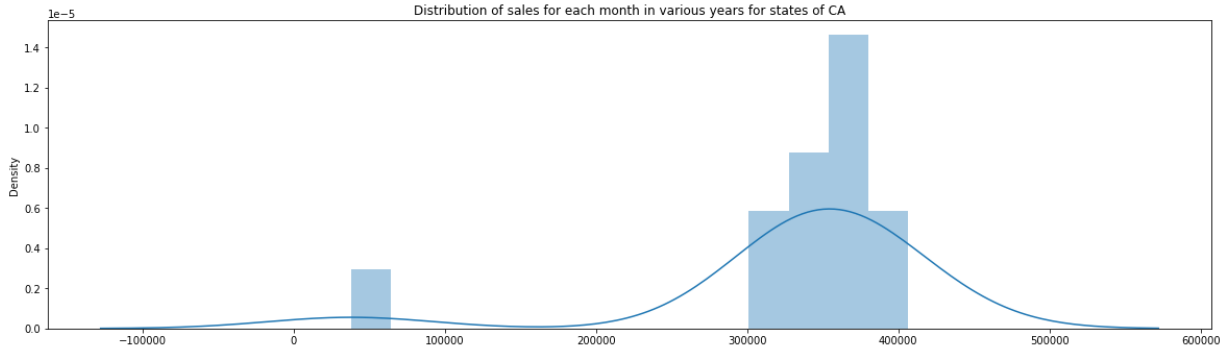
- **Figure - 11**

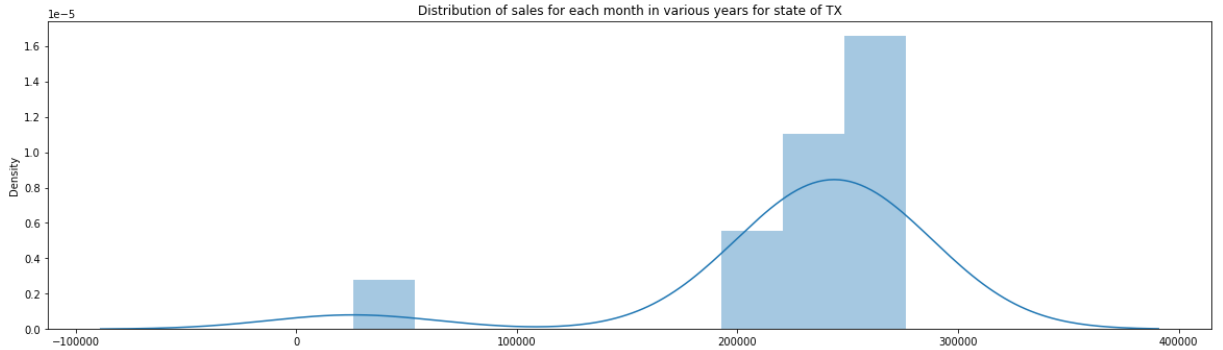


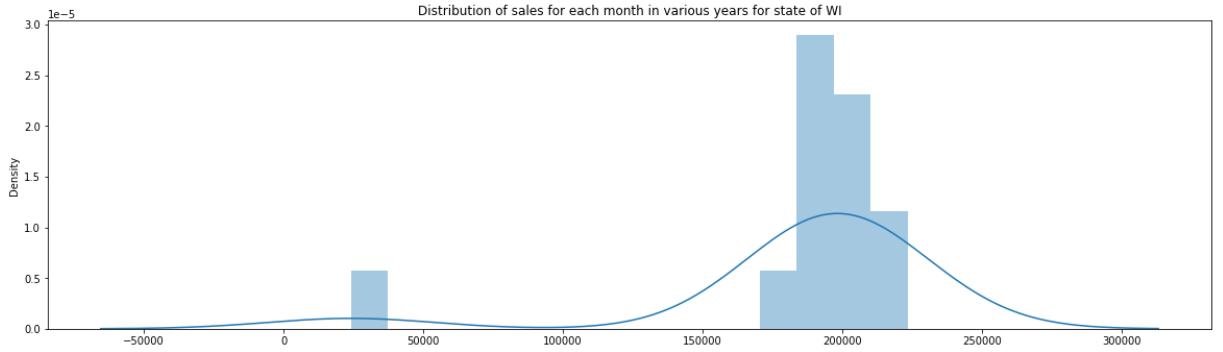
The above plots show the average and total sales according to the event types in all the three states. Average sales are maximum when we have 2 events of Religious and Cultural type. Thus sales get impacted according to the event.

- **Figure - 12**







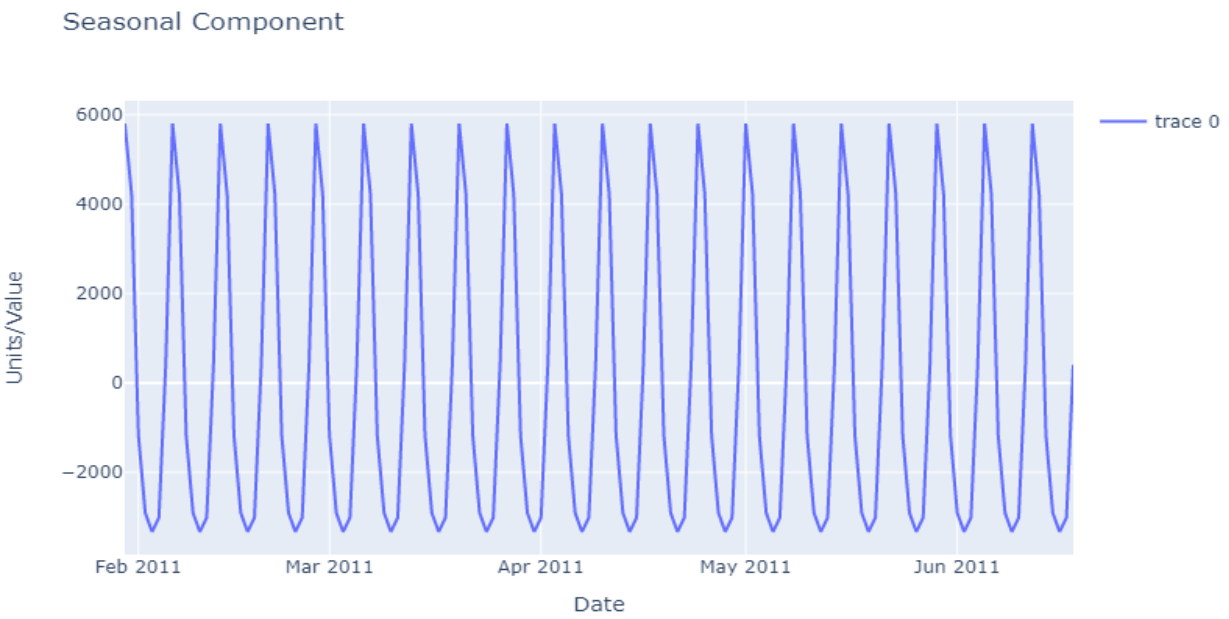


The plots above show the sales in each state in the increasing order of the months. Each of the plots are left skewed and show that the sales in each location increased with time.

- **Figure - 13**

The above plots show the seasonal decomposition of the time series into trend and seasonality. Clearly we can see the different components are at play. At this moment we can not be so sure if this is the exact decomposition we want. The weekly seasonalities & the Monthly seasonality were the sales peaks, shortly after the pay date(assuming monthly pay) are clearly captured.

- **Figure - 14**

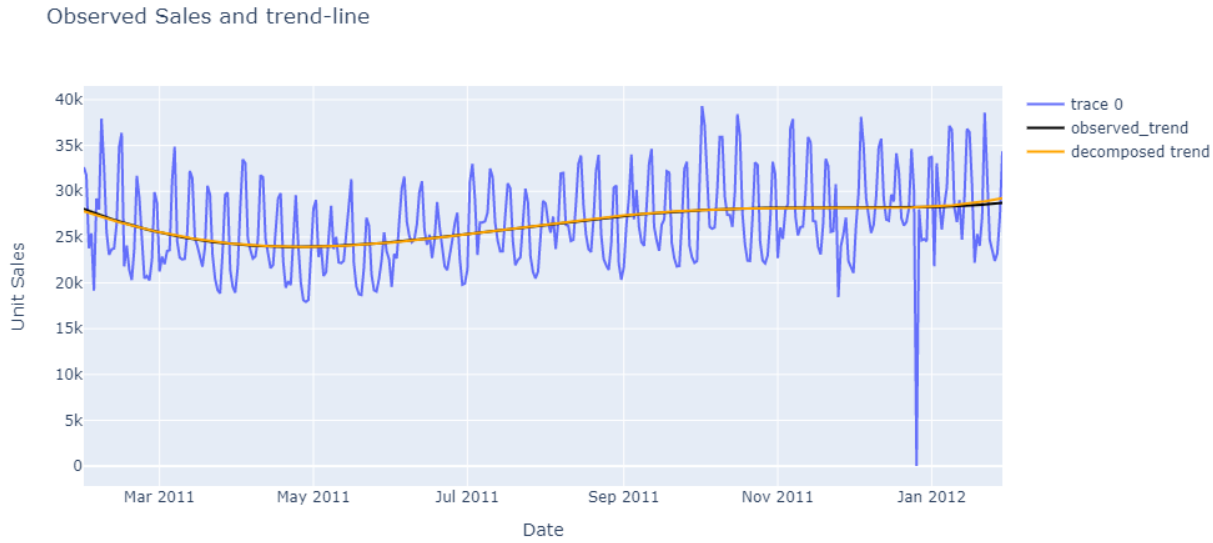


It is clear that it shows the weekly fluctuations in the sales data where the sales peaks at weekends(Saturday and Sunday).

Moving to the noise component(noise), the dip at year end is captured there rather than being in seasonal trend.

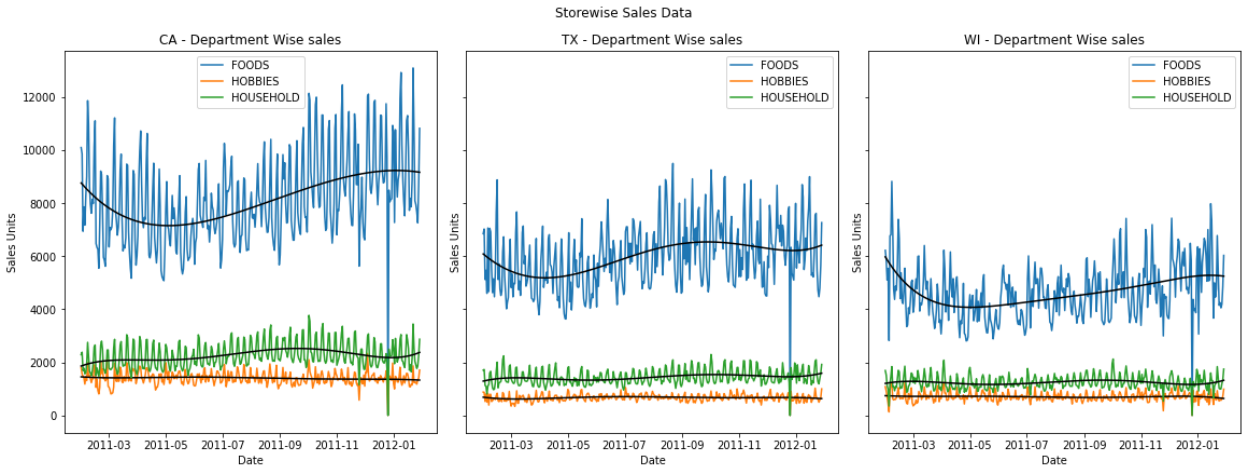
It would be a good comparison between the Trend part of the decomposition and the overall trend for Observed data. It can bring in an idea how effective our decomposition algorithm is here. Let's bring in polynomial regression to make a trendline.

- **Figure - 15**



It's clear from the plots that the trendline from seasonal decomposition is quite aligned with the observed trendline. Hence the naive decomposition does a very good job in getting the trend right.

- **Figure - 16**



We can observe similar trends statewise as well in the above graph. Food is the wild card here with maximum fluctuations observed.

B. Feature engineering: create two new features

We try to create two new features, weather data and median income through feature engineering. We have taken the weather data from the National Oceanic and Atmospheric Administration website.

- **weather data - feature 1**

```
weather.sample(5)
```

	STATION	NAME	LATITUDE	LONGITUDE	ELEVATION	DATE	WT01	WT03	WT04	WT05	WT06	WT11
14794	USR0000CLGA	LOS GATOS CALIFORNIA, CA US	37.202800	-121.942800	609.6	2011-11-01	NaN	NaN	NaN	NaN	NaN	NaN
8132	USC00478919	WATERTOWN WWTP, WI US	43.174240	-88.736330	251.5	2012-12-15	NaN	NaN	NaN	NaN	NaN	NaN
5640	US1WIJF0002	WATERTOWN 1.6 ESE, WI US	43.179157	-88.697222	253.0	2011-06-25	NaN	NaN	NaN	NaN	NaN	NaN
5859	US1WIJF0002	WATERTOWN 1.6 ESE, WI US	43.179157	-88.697222	253.0	2012-09-29	NaN	NaN	NaN	NaN	NaN	NaN
12077	US1WIOZ0005	CEDARBURG 3.7 WSW, WI US	43.278763	-88.056643	267.9	2012-12-02	NaN	NaN	NaN	NaN	NaN	NaN

Here, we can see the fields of weather data, it shows weather according to the zip code of the location, and also takes into consideration the elevation of that location. It can be observed that weather is classified into 6 different classes, viz, WT01, WT03, WT04, WT05, WT06 and WT11. We can also observe that there is no common field in the weather dataset and the other given datasets.

To make this connection, we map unique entities of the column “NAME” in the weather dataset to the column “store_id” in the sales_train_validation dataset.

We merge this data with the original merged dataset that we are going to use for training and testing.

```
display(weather)
```

	STATION	NAME	LATITUDE	LONGITUDE	ELEVATION	DATE	WT01	WT03	WT04	WT05	WT06	WT11	store_id	date	d
0	US1CAAL0024	FREMONT 3.7 NE, CA US	37.571972	-121.95903	31.1	2014-12-18	0.0	0.0	0.0	0.0	1.0	0.0	CA_4	2014-12-18	d_1420
1	US1CAAL0024	FREMONT 3.7 NE, CA US	37.571972	-121.95903	31.1	2014-12-19	0.0	0.0	0.0	1.0	0.0	0.0	CA_4	2014-12-19	d_1421
2	US1CAAL0024	FREMONT 3.7 NE, CA US	37.571972	-121.95903	31.1	2014-12-20	1.0	0.0	0.0	0.0	0.0	0.0	CA_4	2014-12-20	d_1422
3	US1CAAL0024	FREMONT 3.7 NE, CA US	37.571972	-121.95903	31.1	2014-12-21	1.0	0.0	0.0	0.0	0.0	0.0	CA_4	2014-12-21	d_1423
4	US1CAAL0024	FREMONT 3.7 NE, CA US	37.571972	-121.95903	31.1	2014-12-25	0.0	1.0	0.0	0.0	0.0	0.0	CA_4	2014-12-25	d_1427
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
18389	USC00043244	FREMONT, CA US	37.542200	-122.01580	11.6	2016-06-15	1.0	0.0	0.0	0.0	0.0	0.0	CA_4	2016-06-15	d_1965
18390	USC00043244	FREMONT, CA US	37.542200	-122.01580	11.6	2016-06-16	0.0	1.0	0.0	0.0	0.0	0.0	CA_4	2016-06-16	d_1966
18391	USC00043244	FREMONT, CA US	37.542200	-122.01580	11.6	2016-06-17	0.0	0.0	1.0	0.0	0.0	0.0	CA_4	2016-06-17	d_1967
18392	USC00043244	FREMONT, CA US	37.542200	-122.01580	11.6	2016-06-18	0.0	0.0	0.0	0.0	0.0	1.0	CA_4	2016-06-18	d_1968
18393	USC00043244	FREMONT, CA US	37.542200	-122.01580	11.6	2016-06-19	0.0	0.0	1.0	0.0	0.0	0.0	CA_4	2016-06-19	d_1969

18394 rows × 15 columns

- **median income - feature 2**

We obtain the median income from the following website (United States Census Bureau): https://data.census.gov/cedsci/table?q=ZCTA5%2090804%20Income%20and%20Poverty&tid=ACSST5Y2020.S1903

The data we obtained by default contained the median income based on factors like household strength, race, age, etc. The data was quite big and impossible to show here. We removed the fields that were unnecessary and kept only the necessary features, like zip code, average household median income, and year.

We also renamed some of the columns so that it would be easier to merge this dataset with the original dataset as added “feature”. In addition to this, we had to merge the two tables of median income of 2011 and 2012 respectively, since we the data we got was separate according to the years.

After all this preprocessing, the data looked like this:

	NAME	S1903_C02_001E	Year	store_ID
0	ZCTA5 53091	56830	2011	WI_3
1	ZCTA5 53092	99123	2011	WI_2
2	ZCTA5 53094	51205	2011	WI_1
3	ZCTA5 75032	92526	2011	TX_1
4	ZCTA5 75048	87721	2011	TX_3
5	ZCTA5 90804	43489	2011	CA_1
6	ZCTA5 92107	60673	2011	CA_3
7	ZCTA5 94536	84840	2011	CA_4
8	ZCTA5 95032	118713	2011	CA_2
9	ZCTA5 75033	77235	2011	TX_2
0	ZCTA5 53091	55052	2012	WI_3
1	ZCTA5 53092	96083	2012	WI_2
2	ZCTA5 53094	50422	2012	WI_1
3	ZCTA5 75032	91456	2012	TX_1
4	ZCTA5 75048	88691	2012	TX_3
5	ZCTA5 90804	41642	2012	CA_1
6	ZCTA5 92107	61680	2012	CA_3
7	ZCTA5 94536	85196	2012	CA_4
8	ZCTA5 95032	122512	2012	CA_2

We again merge this dataset to the main data frame:

```
data_features = pd.merge(data_features,df)
```

```
data_features.head()
```

	cat_id	store_id	d	sales	date	weekday	month	event_name_1	event_type_1	snap_CA	snap_TX	snap_WI	NAME	WT01	WT03	WT04	WT05	WT06	WT11	Med_Income
0	HOBBIES	CA_2	d_1	6	2011-01-29	Saturday	1	no_event	no_event	0	0	0	LOS GATOS, CA US	0.0	0.0	1.0	0.0	0.0	0.0	118713
1	HOBBIES	CA_2	d_1	6	2011-01-29	Saturday	1	no_event	no_event	0	0	0	LOS GATOS CALIFORNIA, CA US	0.0	0.0	1.0	0.0	0.0	0.0	118713
2	HOBBIES	CA_2	d_1	2	2011-01-29	Saturday	1	no_event	no_event	0	0	0	LOS GATOS, CA US	0.0	0.0	1.0	0.0	0.0	0.0	118713
3	HOBBIES	CA_2	d_1	2	2011-01-29	Saturday	1	no_event	no_event	0	0	0	LOS GATOS CALIFORNIA, CA US	0.0	0.0	1.0	0.0	0.0	0.0	118713
4	HOBBIES	CA_2	d_1	23	2011-01-29	Saturday	1	no_event	no_event	0	0	0	LOS GATOS, CA US	0.0	0.0	1.0	0.0	0.0	0.0	118713

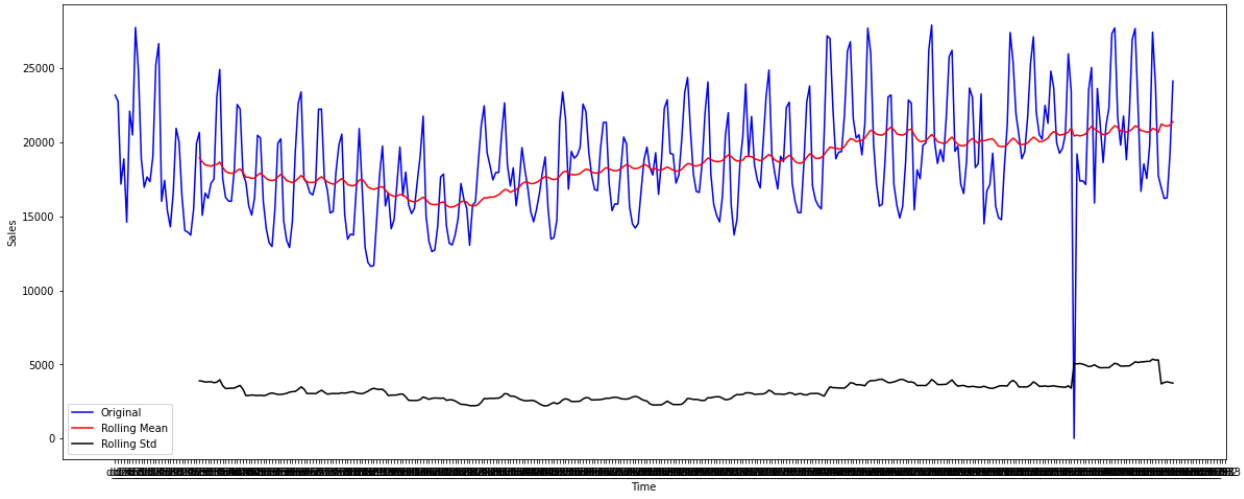
C. Machine learning algorithms to model n-step ahead forecasting (n = 10)

We performed the time series analysis using ARIMA model. ARIMA is an autoregressive integrated moving average model based on a description of the trend and seasonality in the data, ARIMA models aim to describe the autocorrelations in the data. The final objective to use this model is to predict future time series movement of each category (FOODS, HOBBIES, HOUSEHOLD) by examining the differences between values in the series instead of through actual values. We would first perform the time series data analysis for each of the category in the dataset.

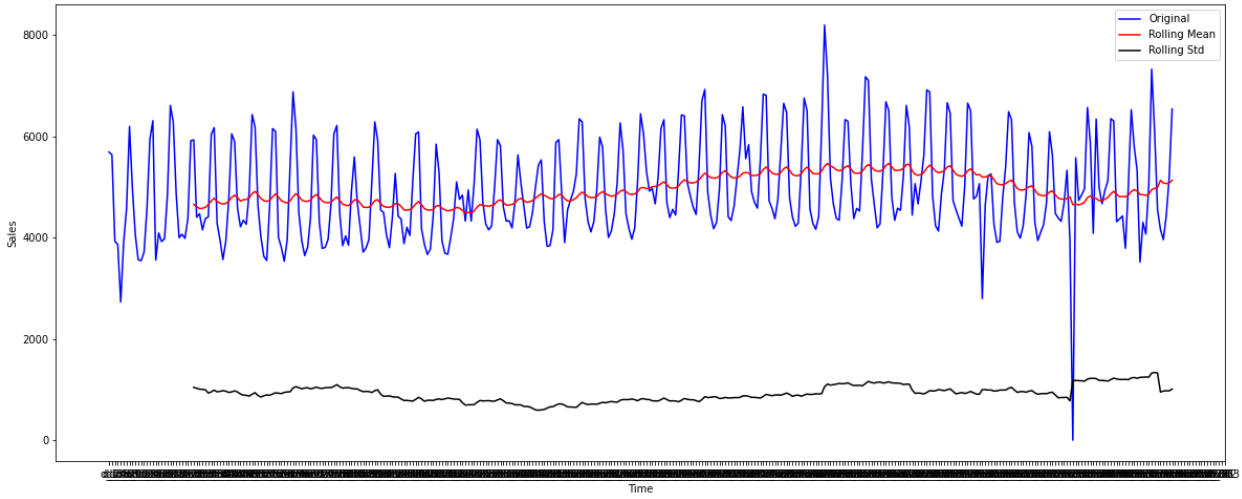
- **Time Series Data Analysis**

To get an idea of the modeling, we will try to pridict result using Time series. Further, predicting total sales in each category Foods, Hobbies & Household.

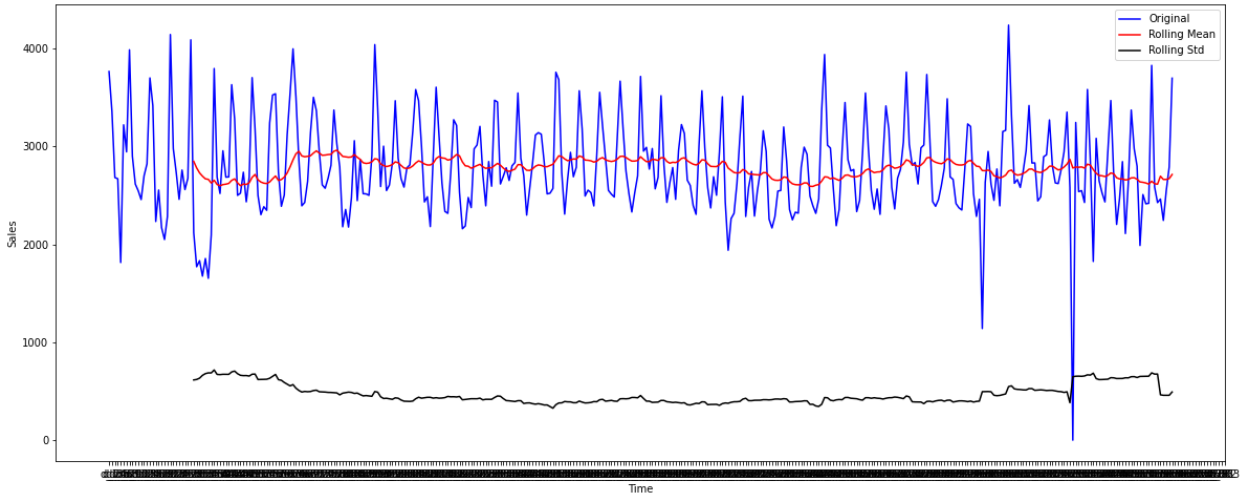
- **FOODS Category**



- **HOBBIES Category**



- **HOUSEHOLD Category**

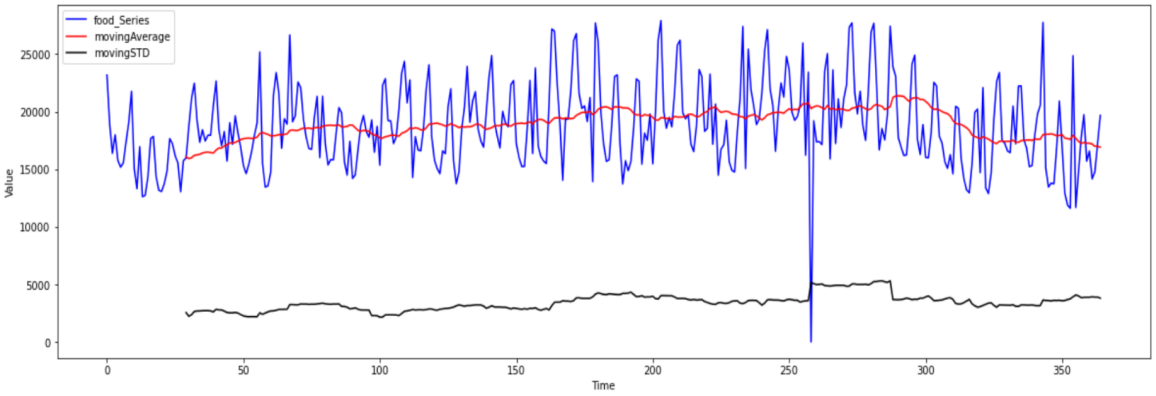


As we convert the the data for sales in each category into time series, we need to check wheter these series are stationary or not. In time series analysis, non-stationary data are always transformed into stationary data. Time series with trends, or with seasonality, are not stationary. The trend and seasonality will affect the value of the time series at different times.

Hence, we have used the plots for each series along with their moving average and moving standard deviation to determine whether the series for each category tend to be stationary or not.

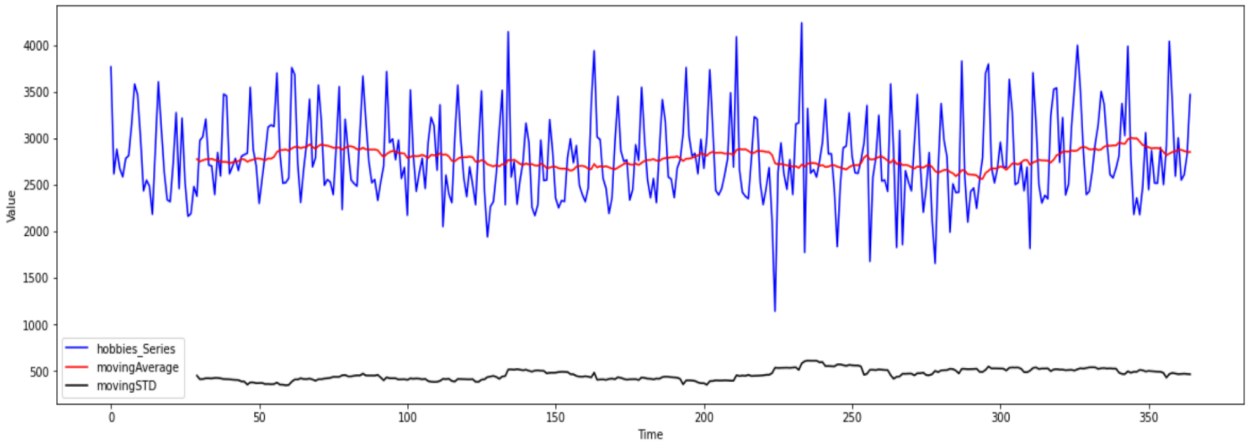
- **FOODS Category**

```
Test Statistic                -4.685564
p-value                        0.000090
#Lags Used                    12.000000
No of Observesations Used    352.000000
Critical Value (1%)           -3.449065
Critical Value (5%)           -2.869786
Critical Value (10%)          -2.571163
dtype: float64
```



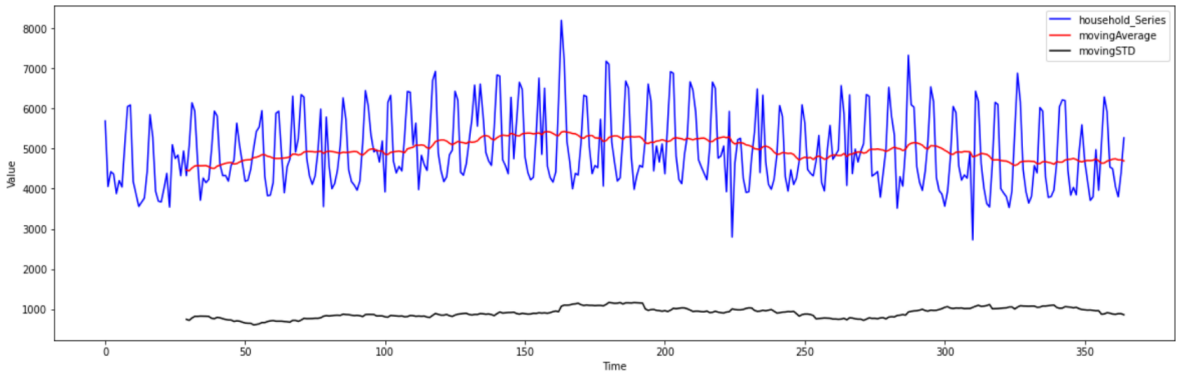
- **HOBBIES Category**

```
Test Statistic                -3.603406
p-value                        0.005696
#Lags Used                    15.000000
No of Observesations Used    349.000000
Critical Value (1%)           -3.449227
Critical Value (5%)           -2.869857
Critical Value (10%)          -2.571201
dtype: float64
```



- **HOUSEHOLD Category**

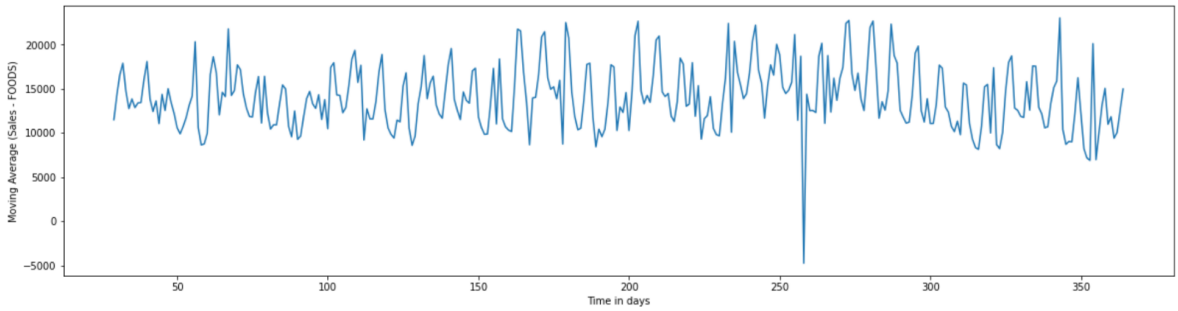
```
Test Statistic                -2.640144
p-value                        0.084981
#Lags Used                    15.000000
No of Observesations Used    349.000000
Critical Value (1%)           -3.449227
Critical Value (5%)           -2.869857
Critical Value (10%)          -2.571201
dtype: float64
```



We used the ADF (Augmented Dickey-Fuller) test to determine the stationarity of the time series. It is a statistical significance test which means the test will give results in hypothesis tests with null and alternative hypotheses. We observed that test statistic for all the three categories is not below the 1% of the Critical value. Hence we conclude that all, the three series are not stationary and hence the ARIMA model cannot be implemented on these moving averages directly.

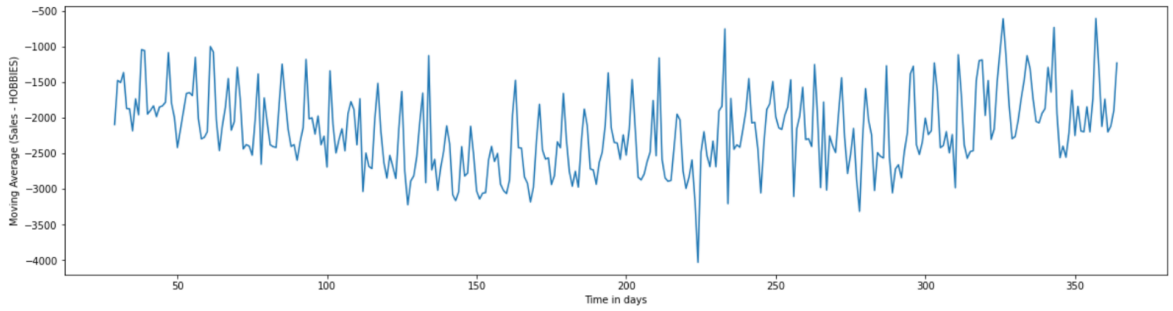
First the time series for all the three categories has to be made stationary and the test statistic value is reduced. Then we implemented the ARIMA model. For this, we have calculated the differencing moving averages for each series and plotted the mean values of them.

- **FOODS Category**



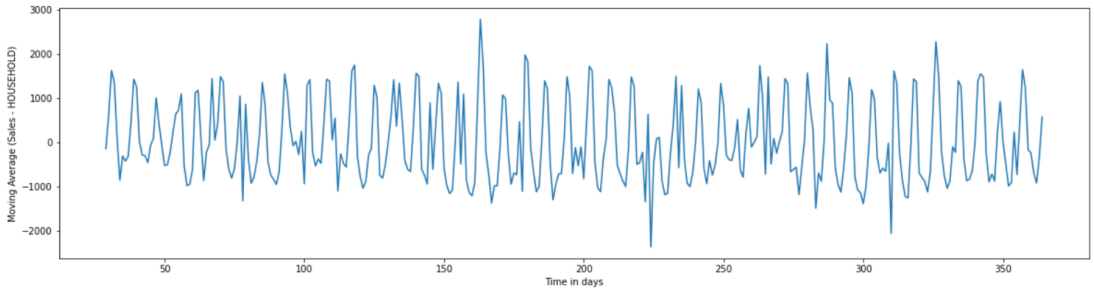

```
Test Statistic                -5.23
p-value                        0.00
#Lags Used                    12.00
No of Observesations Used    352.00
Critical Value (1%)           -3.45
Critical Value (5%)           -2.87
Critical Value (10%)          -2.57
dtype: float64
```

- **HOBBIES Category**




```
Test Statistic                -1.58441
p-value                        0.49142
#Lags Used                    15.00000
No of Observesations Used    349.00000
Critical Value (1%)           -3.44923
Critical Value (5%)           -2.86986
Critical Value (10%)          -2.57120
dtype: float64
```

- **HOUSEHOLD Category**



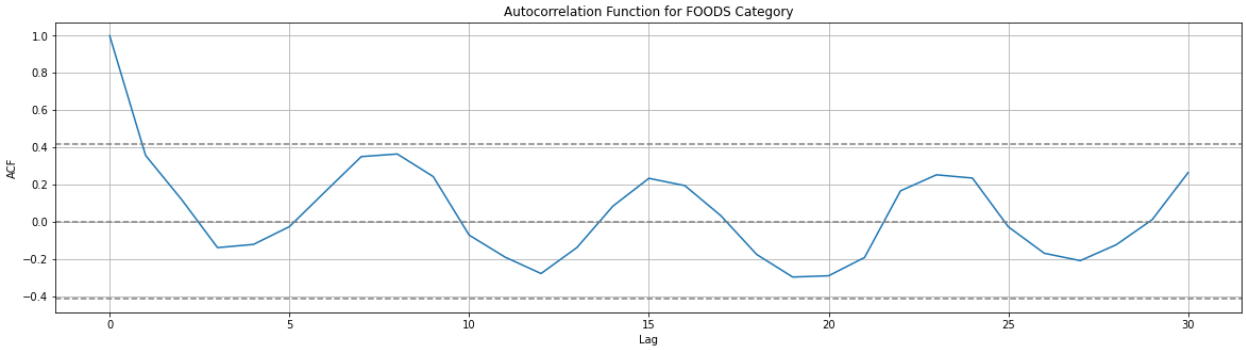

```
Test Statistic                -5.08518
p-value                        0.00002
#Lags Used                    13.00000
No of Observesations Used    351.00000
Critical Value (1%)           -3.44912
Critical Value (5%)           -2.86981
Critical Value (10%)          -2.57118
dtype: float64
```

- **Autocorrelation (ACF) and Partial autocorrelation function (PACF)**

After the time series has been stationarized by differencing, the next step in fitting an ARIMA model is to determine whether AR or MA terms are needed to correct any autocorrelation that remains in the differenced series. By looking at the autocorrelation function (ACF) and partial autocorrelation (PACF) plots of the differenced series, we can tentatively identify the numbers of AR and MA terms that are needed.

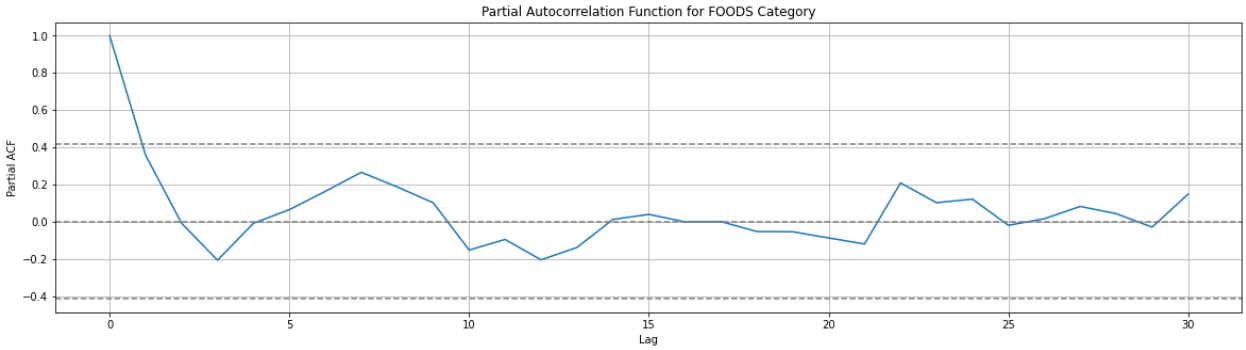
We now plot the ACF and PACF plots for each of the stationery time series. ACF is a chart of the coefficients of correlation between a time series and lags of itself. The PACF plot is a plot of the partial correlation coefficients between the series and lags of itself.

- **ACF for FOODS Category**



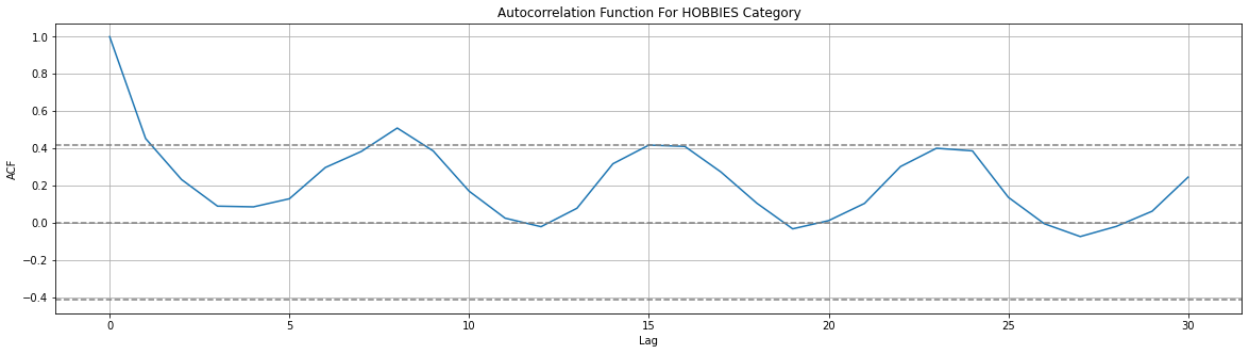
Autocorrelation function cross upper confidence value between 1 and 2. Hence, we have taken p = 2 for ARIMA for FOODS Category.

- **PACF for FOODS Category**



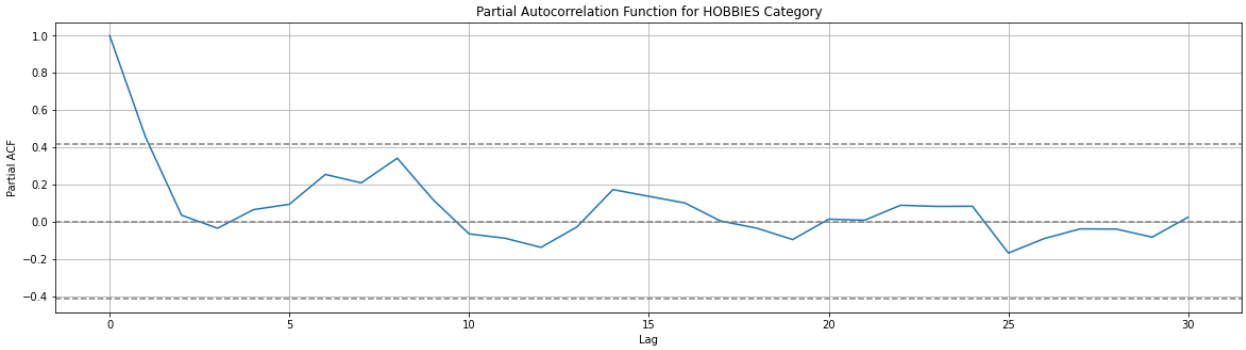
Partial Autocorrelation function drops to 0 when value is between 1 and 2. Hence we have taken q = 2 for ARIMA for FOODS Category.

- **ACF for HOBBIES Category**



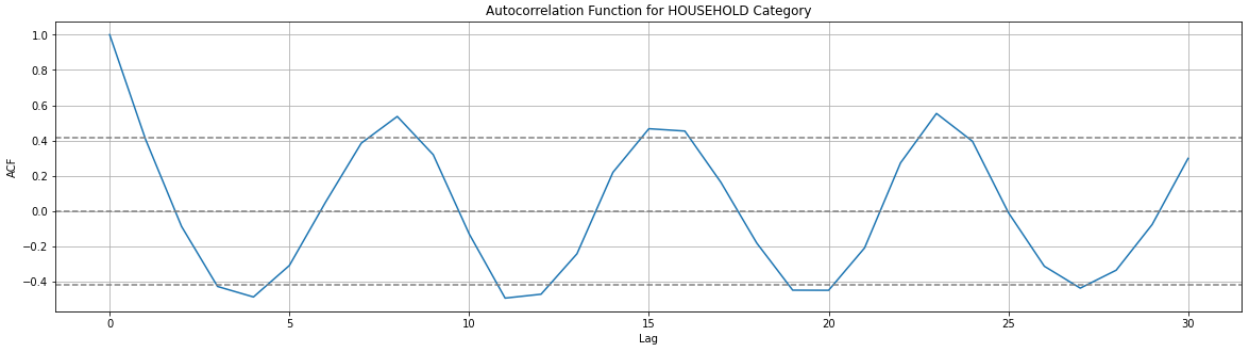
Autcorrelation function cross upper confidence value between 1 and 2. Hence, we have taken p = 2 for ARIMA for HOBBIES Category.

- **PACF for HOBBIES Category**



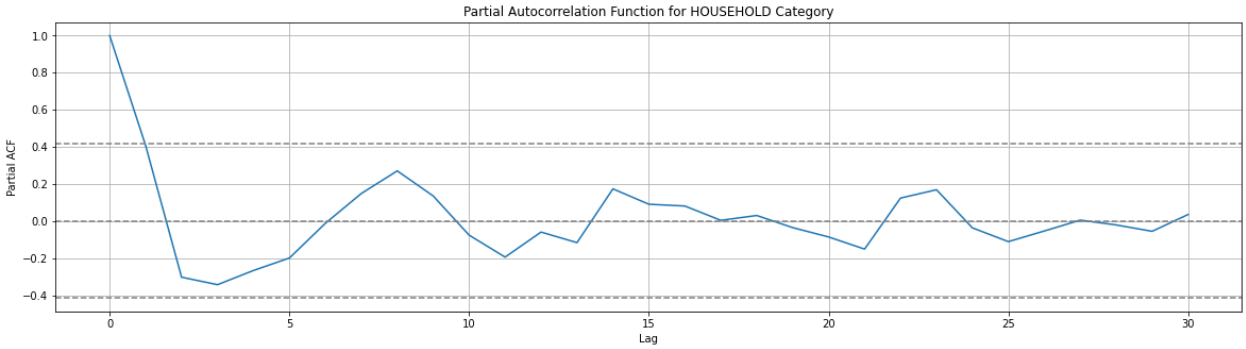
Partial Autocorrelation function drops to 0 when value is between 1 and 2. Hence we have taken q = 2 for ARIMA for HOBBIES Category.

- **ACF for HOUSEHOLD Category**



Autocorrelation function cross upper confidence value between 1 and 2. Hence, we have taken p = 2 for ARIMA for HOUSEHOLD Category.

- **PACF for HOUSEHOLD Category**



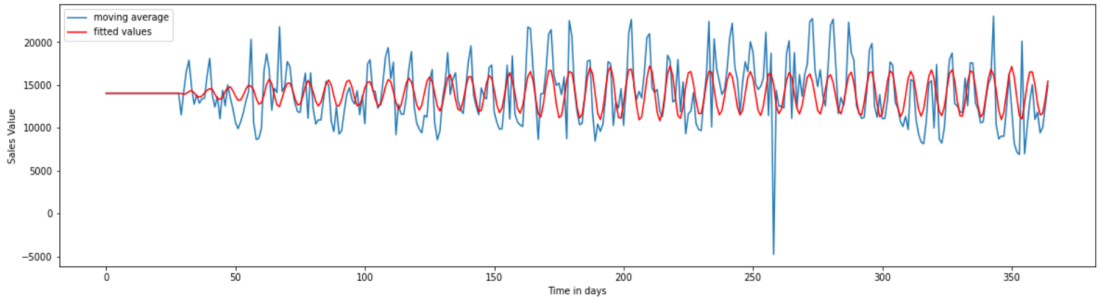
Partial Autocorrelation function drops to 0 when value is between 1 and 2. Hence we have taken q = 2 for ARIMA for HOUSEHOLD Category.

Now we fit the ARIMA model on the time series of moving averages for each of the categories.

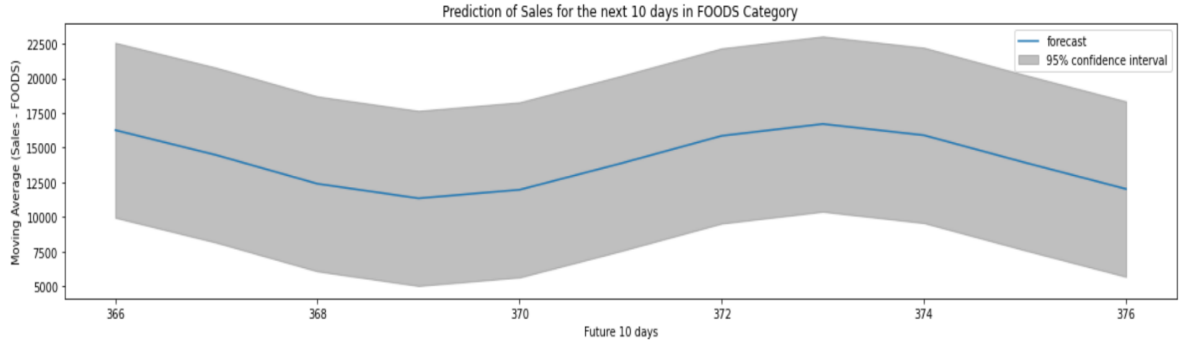
- **ARIMA Model Implementation**

- **ARIMA Model for FOODS Category**

```
from statsmodels.tsa.arima.model import ARIMA
model_foods = ARIMA(foodSeriesDiff,order=(2,0,2))
results_ARIMA_foods = model_foods.fit()
plt.figure(figsize=(20,5))
plt.xlabel("Time in days")
plt.ylabel("Sales Value")
plt.plot(foodSeriesDiff,label='moving average')
plt.plot(results_ARIMA_foods.fittedvalues, color='red',label='fitted values')
plt.legend(loc='best')
plt.show()
```



Now we used this fitted model to predict the moving averages for the further 10 days of the data for FOODS Category.

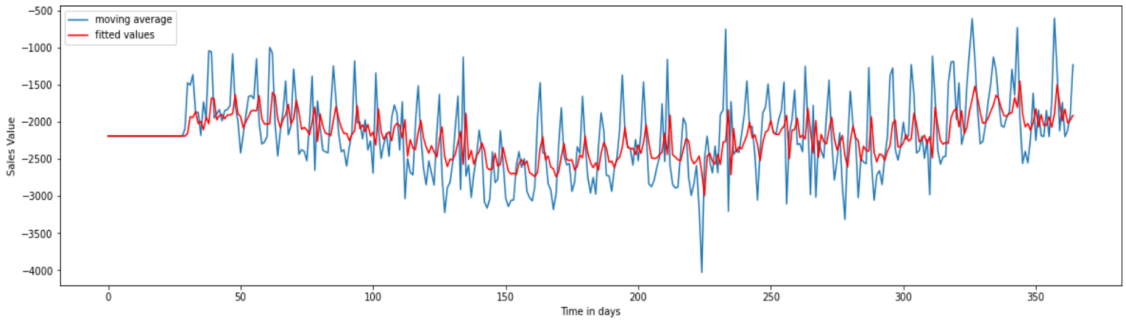


- We then calculated the root mean square error (RMSE) score for this category:

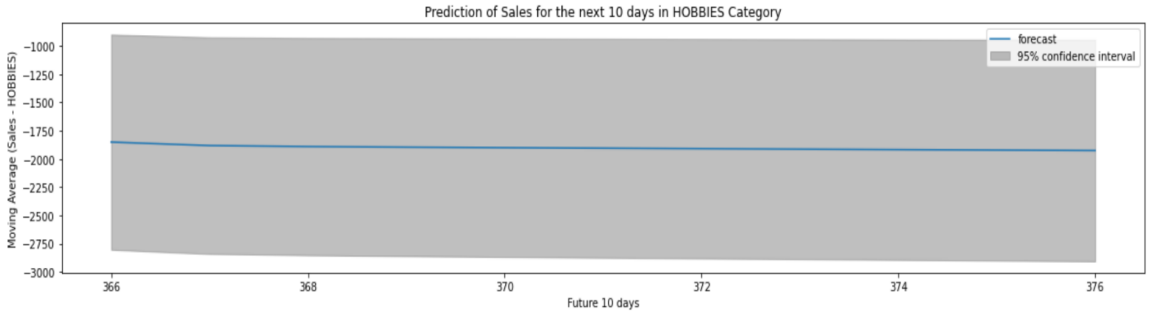
```
The Root Mean Squared Error of our forecasts for FOODS Category is 54.76.
```

- **ARIMA Model for HOBBIES Category**

```
model_hobbies = ARIMA(hobbiesSeriesDiff,order=(2,0,2))
results_ARIMA_hobbies = model_hobbies.fit()
plt.figure(figsize=(20,5))
plt.xlabel("Time in days")
plt.ylabel("Sales Value")
plt.plot(hobbiesSeriesDiff,label='moving average')
plt.plot(results_ARIMA_hobbies.fittedvalues, color='red',label='fitted values')
plt.legend(loc='best')
plt.show()
```



Now we used this fitted model to predict the moving averages for the further 10 days of the data for HOBBIES Category.

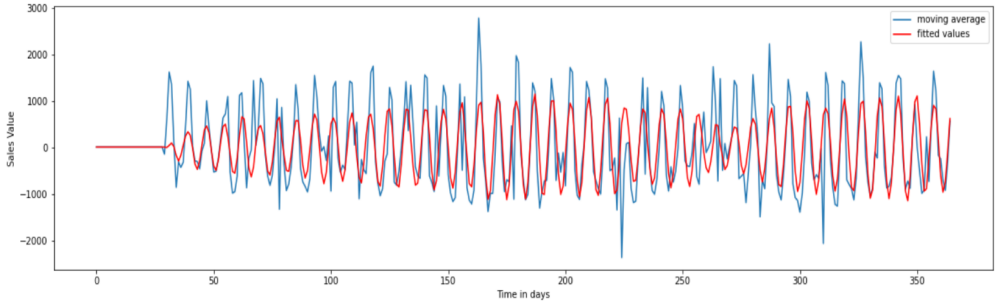


- We then calculated the root mean square error (RMSE) score for this category:

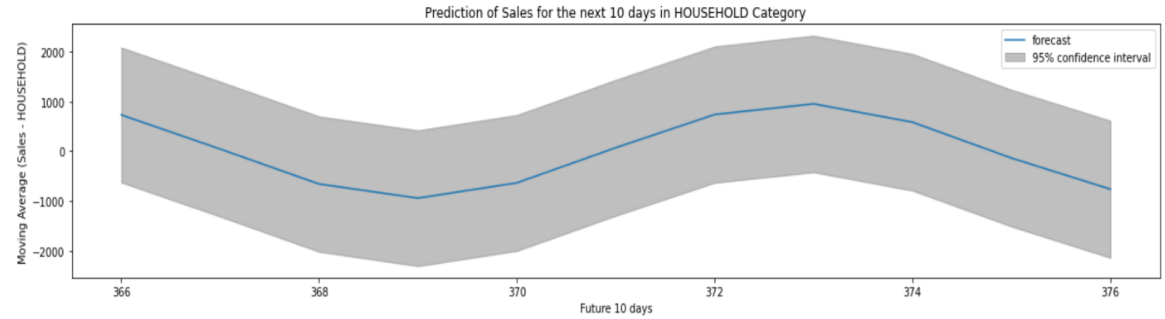
`The Root Mean Squared Error of our forecasts for HOBBIES Category is 21.38.`

- **ARIMA Model for HOUSEHOLD Category**

```
[ ] model_household = ARIMA(householdSeriesDiff,order=(2,0,2))
    results_ARIMA_household = model_household.fit()
    plt.figure(figsize=(20,5))
    plt.xlabel("Time in days")
    plt.ylabel("Sales Value")
    plt.plot(householdSeriesDiff,label='moving average')
    plt.plot(results_ARIMA_household.fittedvalues, color='red',label='fitted values')
    plt.legend(loc='best')
    plt.show()
```



Now we used this fitted model to predict the moving averages for the further 10 days of the data for HOUSEHOLD Category.



- We then calculated the root mean square error (RMSE) score for this category:

The Root Mean Squared Error of our forecasts for FOODS Category is 25.03.

- **LSTM Model Implementation**

We used Tensorflow LSTM model for this dataset and predicted the toal sales for further 10 days. The Model has one LSTM layer and one dense layer. Hyperparameter tuning is applied to improve the model.

```
model = tf.keras.Sequential()
model.add(tf.keras.layers.LSTM(units =64,input_shape = (np.array(X_train).shape[1], np.array(X_train).shape[2])))
model.add(tf.keras.layers.Dense(19))

model.compile(
  loss='mean_squared_error',
  optimizer=tf.keras.optimizers.Adam(0.001)
)
model.summary()
```

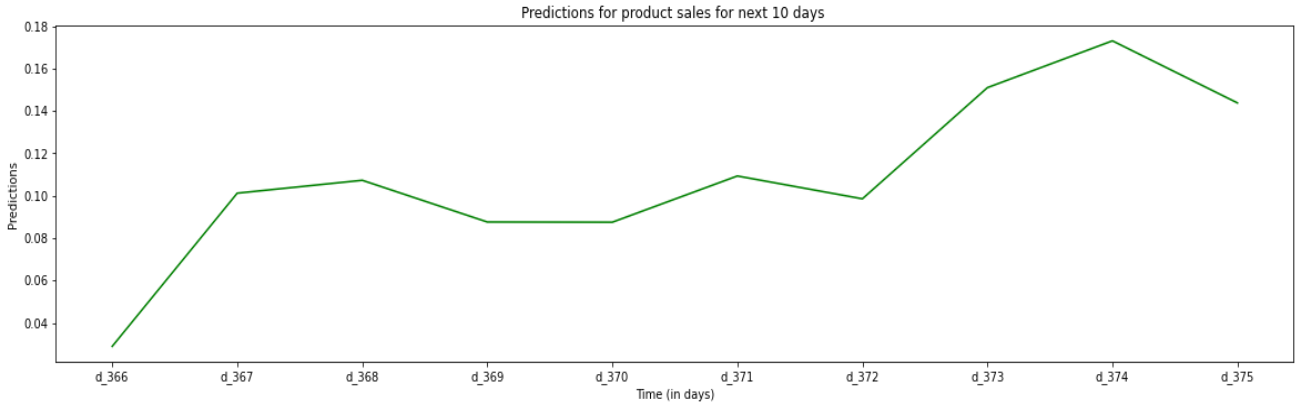
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 64)                21504

 dense (Dense)               (None, 19)                1235

=================================================================
Total params: 22,739
Trainable params: 22,739
Non-trainable params: 0
_________________________________________________________________
```

Model is then fitted with 10 epochs and 10 batch size. If the performance of the model degrades, then the model is stopped.

```
model.fit(X_train, y_train, epochs = 10, batch_size = 10)
```

```
Epoch 1/10
36/36 [==============================] - 7s 3ms/step - loss: 0.1695
Epoch 2/10
36/36 [==============================] - 0s 3ms/step - loss: 0.0987
Epoch 3/10
36/36 [==============================] - 0s 3ms/step - loss: 0.0938
Epoch 4/10
36/36 [==============================] - 0s 3ms/step - loss: 0.0903
Epoch 5/10
36/36 [==============================] - 0s 3ms/step - loss: 0.0877
Epoch 6/10
36/36 [==============================] - 0s 3ms/step - loss: 0.0850
Epoch 7/10
36/36 [==============================] - 0s 3ms/step - loss: 0.0831
Epoch 8/10
36/36 [==============================] - 0s 3ms/step - loss: 0.0836
Epoch 9/10
36/36 [==============================] - 0s 3ms/step - loss: 0.0817
Epoch 10/10
36/36 [==============================] - 0s 3ms/step - loss: 0.0812
<keras.callbacks.History at 0x7efda68c4e20>
```

The plot below shows the prediction of total sales for further 10 days using LSTM model -



Then the RMSE score for LSTM model is determined -

RMSE score for LSTM model: 0.31

- **Performance Metrics for ARIMA and LSTM Models**

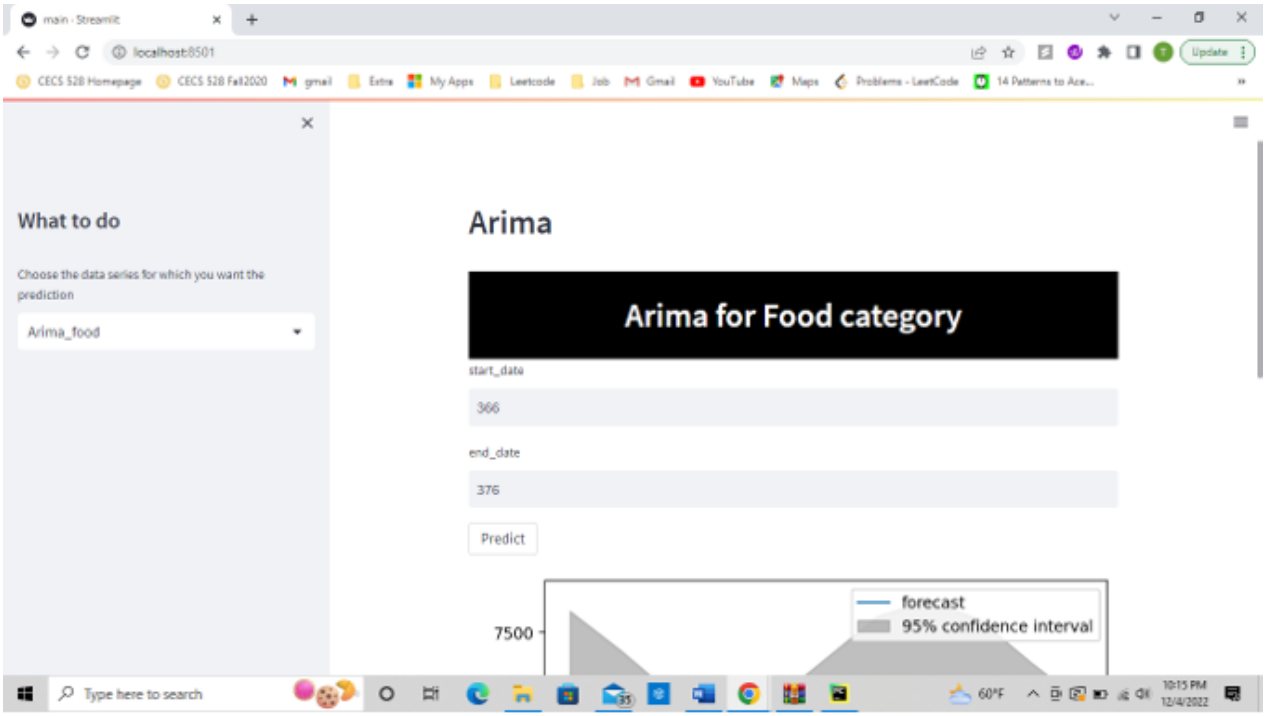
	Prediction Category	MSE Score	RMSE Score
ARIMA Model	Food Category	2999.07	54.76
	Hobbies Category	456.9	21.38
	Household Category	626.73	25.03
LSTM Model	Total Sales	0.09	0.31

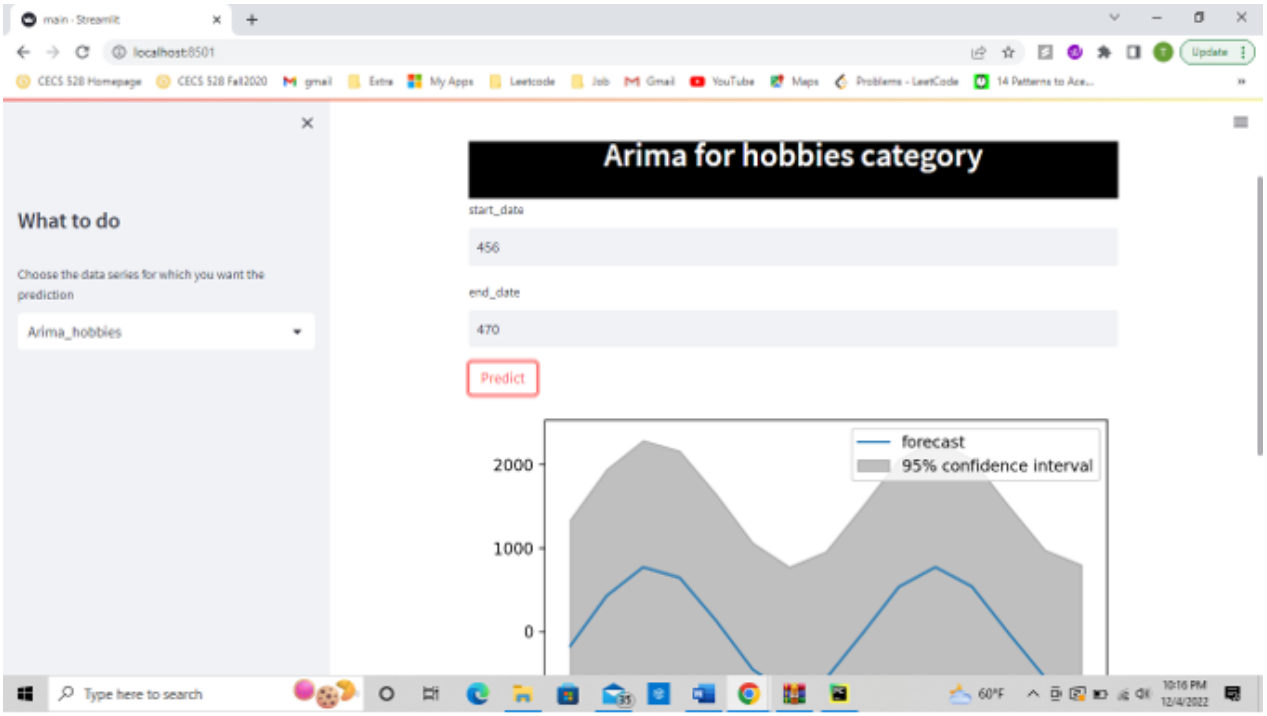
Sprint - 3

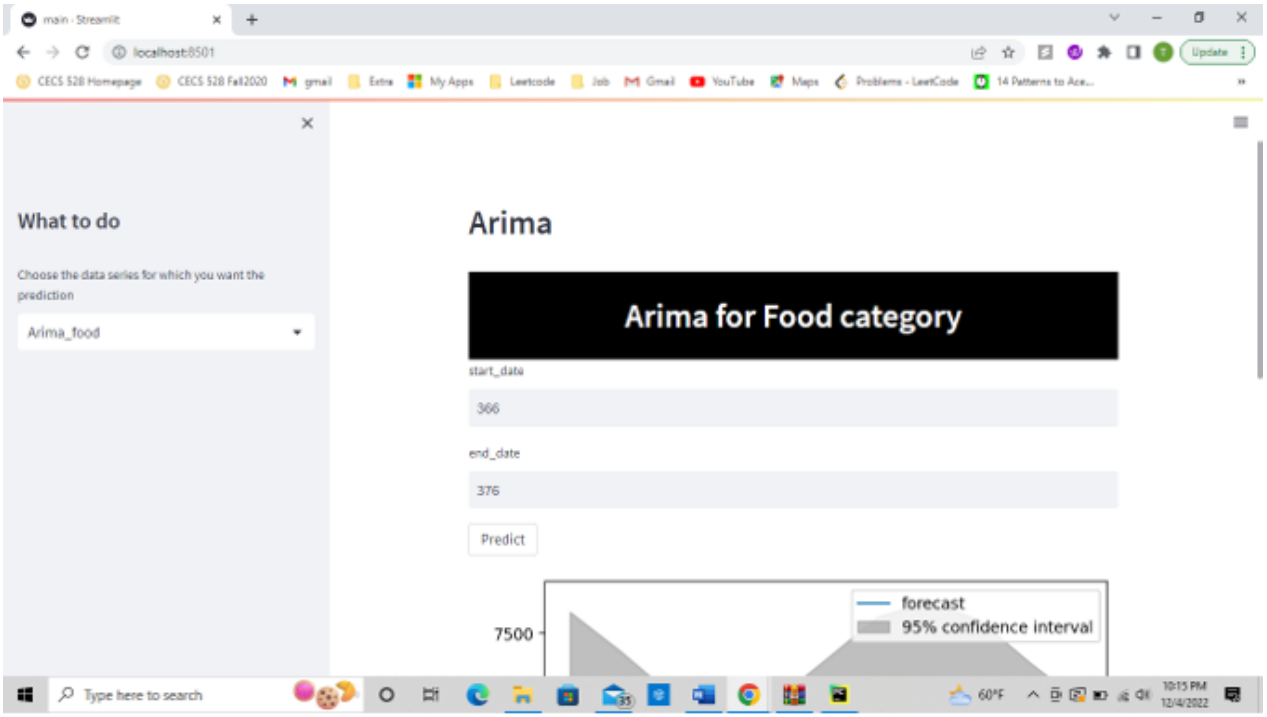
1. Deploy the machine learning model using Streamlit for dataset_02 (ARIMA)

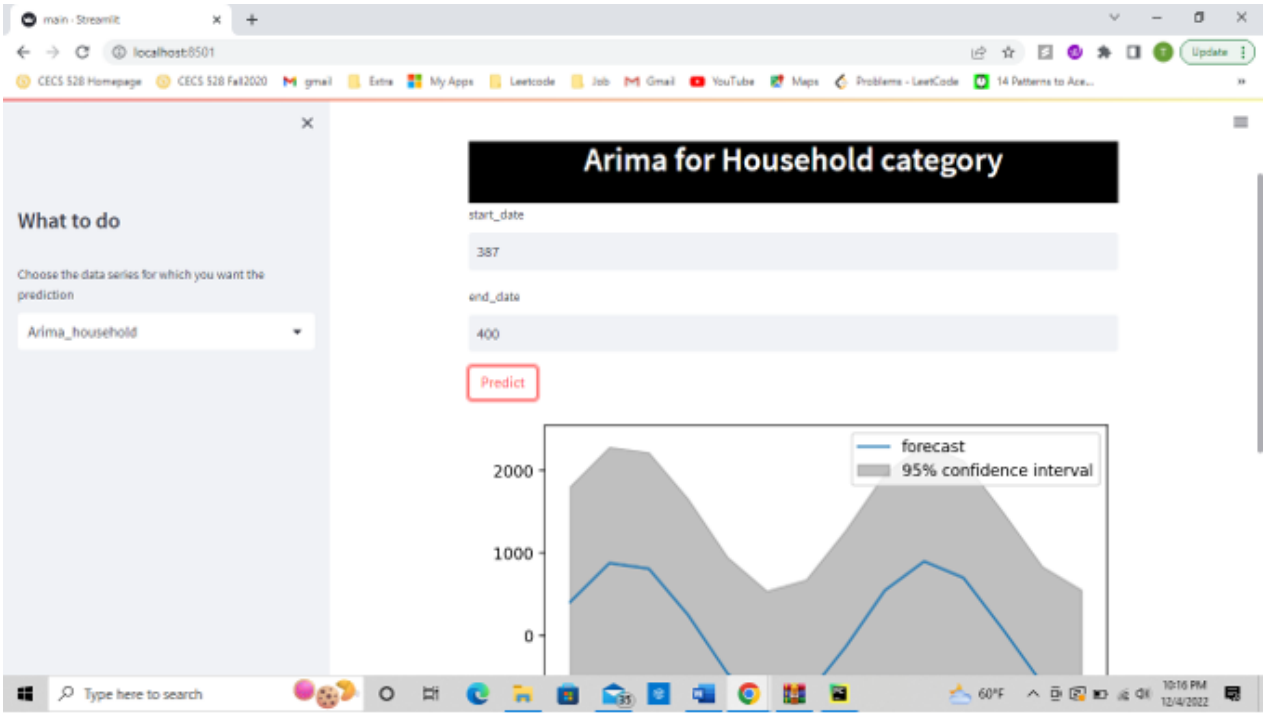
We deployed the ARIMA model using streamlit. From the left section, the category (Foods, Hobbies & Household) can be chosen through the drop down. Further the start date and the end date can be entered and a prediction of sales (moving averages) through ARIMA model in that category will be shown through the graph.

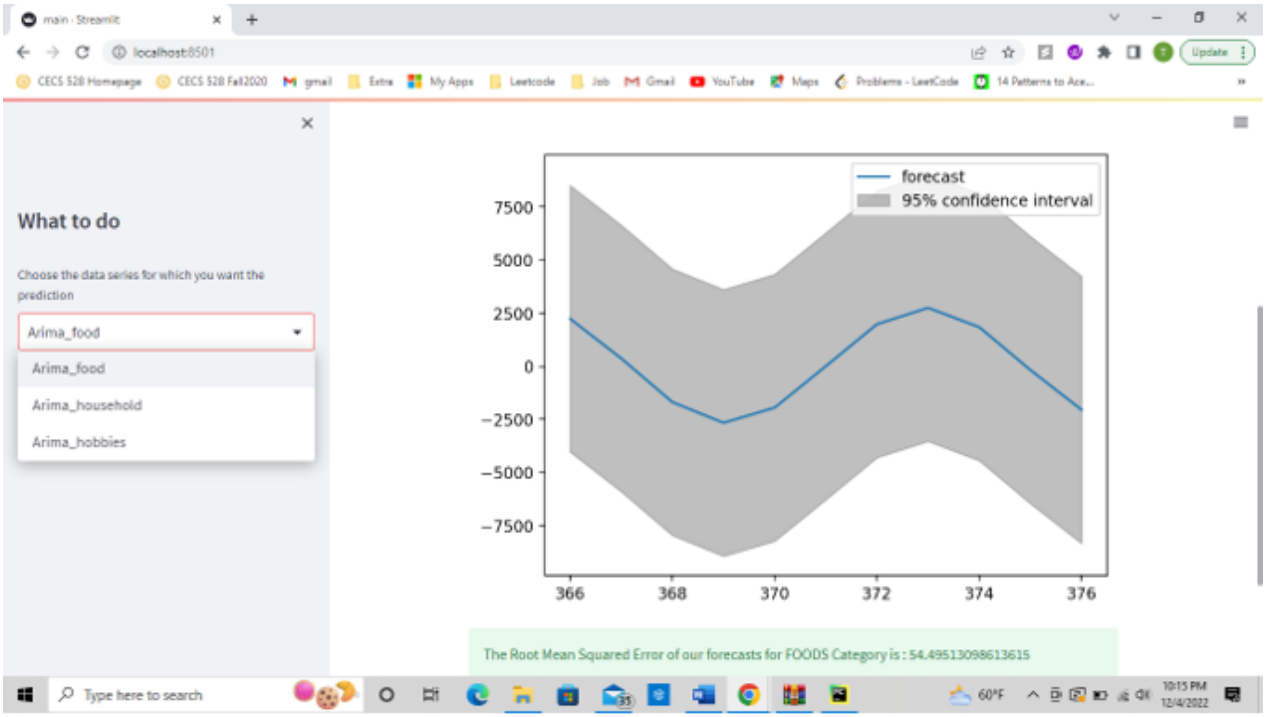
The screenshots show the predicted graphs with different start and end dates.




main · Streamlit
localhost:8501
What to do
Choose the data series for which you want the prediction
Arima_hobbies
Arima for hobbies category
start_date
456
end_date
470
Predict
forecast
95% confidence interval
2000
1000
0
Type here to search
60°F
10:16 PM
12/4/2022


main · Streamlit
localhost:8501
What to do
Choose the data series for which you want the prediction
Arima_food
Arima
Arima for Food category
start_date
366
end_date
376
Predict
forecast
95% confidence interval
7500
Type here to search
60°F
10:15 PM
12/4/2022


main · Streamlit
localhost:8501
Arima for Household category
What to do
Choose the data series for which you want the prediction
Arima_household
start_date
387
end_date
400
Predict
forecast
95% confidence interval
2000
1000
0
Type here to search


main · Streamlit
localhost:8501
What to do
Choose the data series for which you want the prediction
Arima_food
Arima_food
Arima_household
Arima_hobbies
forecast
95% confidence interval
7500
5000
2500
0
−2500
−5000
−7500
366
368
370
372
374
376
The Root Mean Squared Error of our forecasts for FOODS Category is : 54.49513098613615
Type here to search

2. Product segmentation based on demand variability (ABC Analysis).(dataset_02)

A. Use first year data of Household category to create ABC Analysis and interpret the graph.

We now calculate the demand variability for the HOUSEHOLD category, as mentioned in the problem statement. We perform ABC Analysis to get the variability in demand of each item.

What is ABC Analysis:

The Pareto Law states that 80% of the revenue is generated by the top 20% of the items and the remaining 20% is generated by the other 80% of the items.

We divide the items in the household category into three classes, viz, A, B and C.

- Class A: Very Fast Movers: top 5%
- Class B: The following 15% of fast movers
- Class C: The remaining 80% of very slow movers

For this project, we took into consideration the following categories: ***'cat_id'***, ***'sales'*** and ***'sell_price'.*** We then created a new column, ***'revenue'***, which states the revenue generated by each column and a column percentage that shows the revenue percentage for each HOUSEHOLD item.

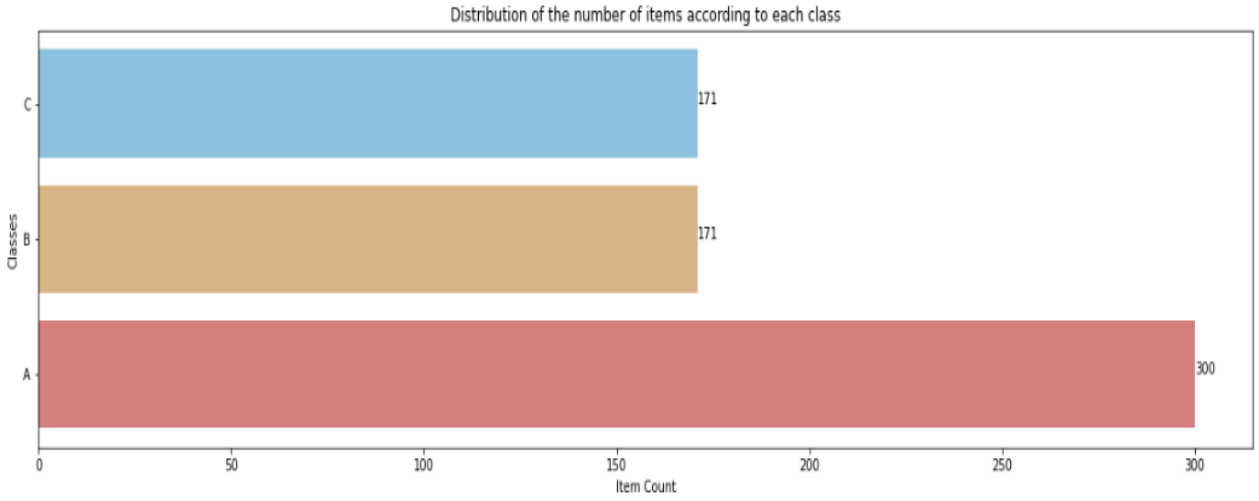
item_id	sell_price	sales	revenue	percentage	Class
HOUSEHOLD_1_272	9.97	11045	110118.65	1.76	A
HOUSEHOLD_1_535	6.97	12046	83960.62	1.34	A
HOUSEHOLD_1_053	14.97	5401	80852.97	1.30	A
HOUSEHOLD_1_537	15.97	3866	61740.02	0.99	A
HOUSEHOLD_1_177	7.97	7636	60858.92	0.97	A

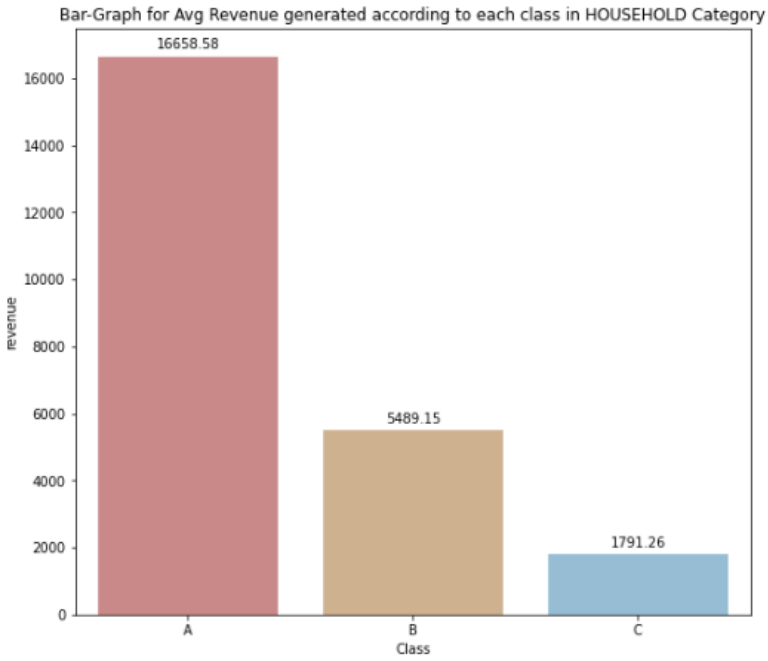
For this project, we tried to classify the items that constitute 80% of the total revenue as 'A', remaining 15% as, 'B' and the rest 5% as 'C'.

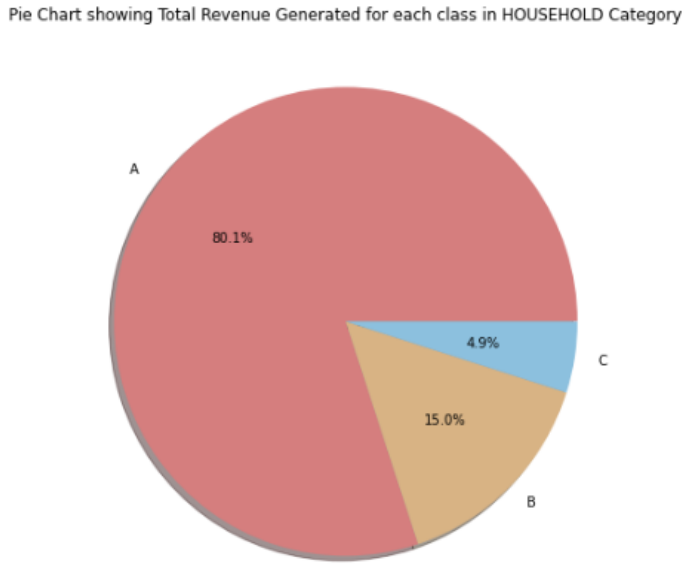
OBSERVATIONS: Here, we can observe that the number of items that constitute 80% of the total revenue are way more than 20% of the total items (46.73% of the total items). It is safe to say that the items in the HOUSEHOLD category across 10 stores in 3 states, do not follow the Pareto Law.

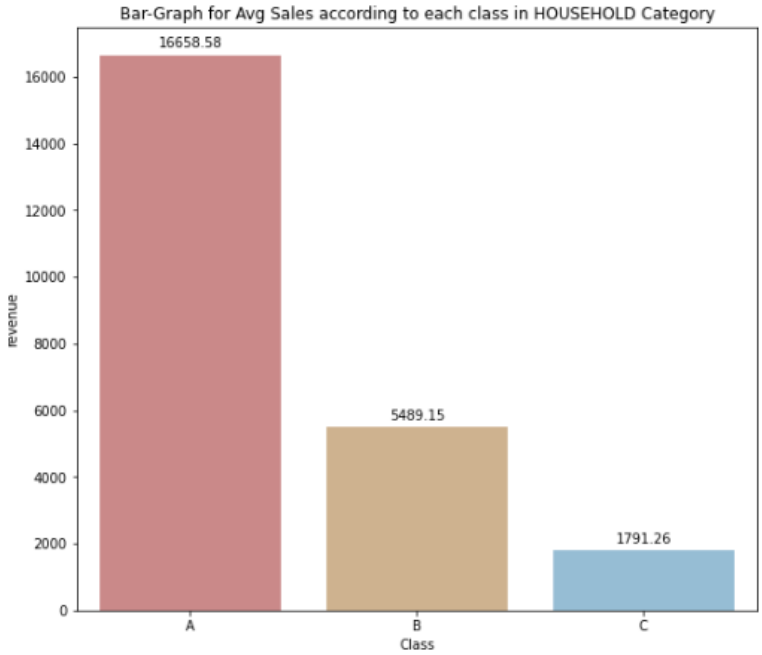
These observations will help us organize the inventory better.

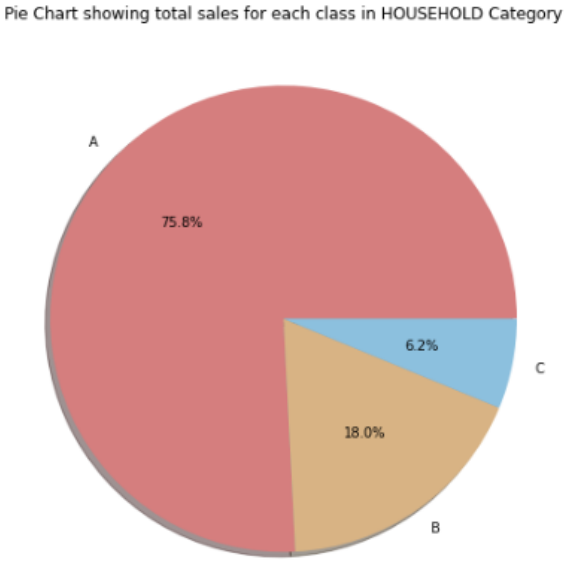
The graphs below further illustrate this classification:

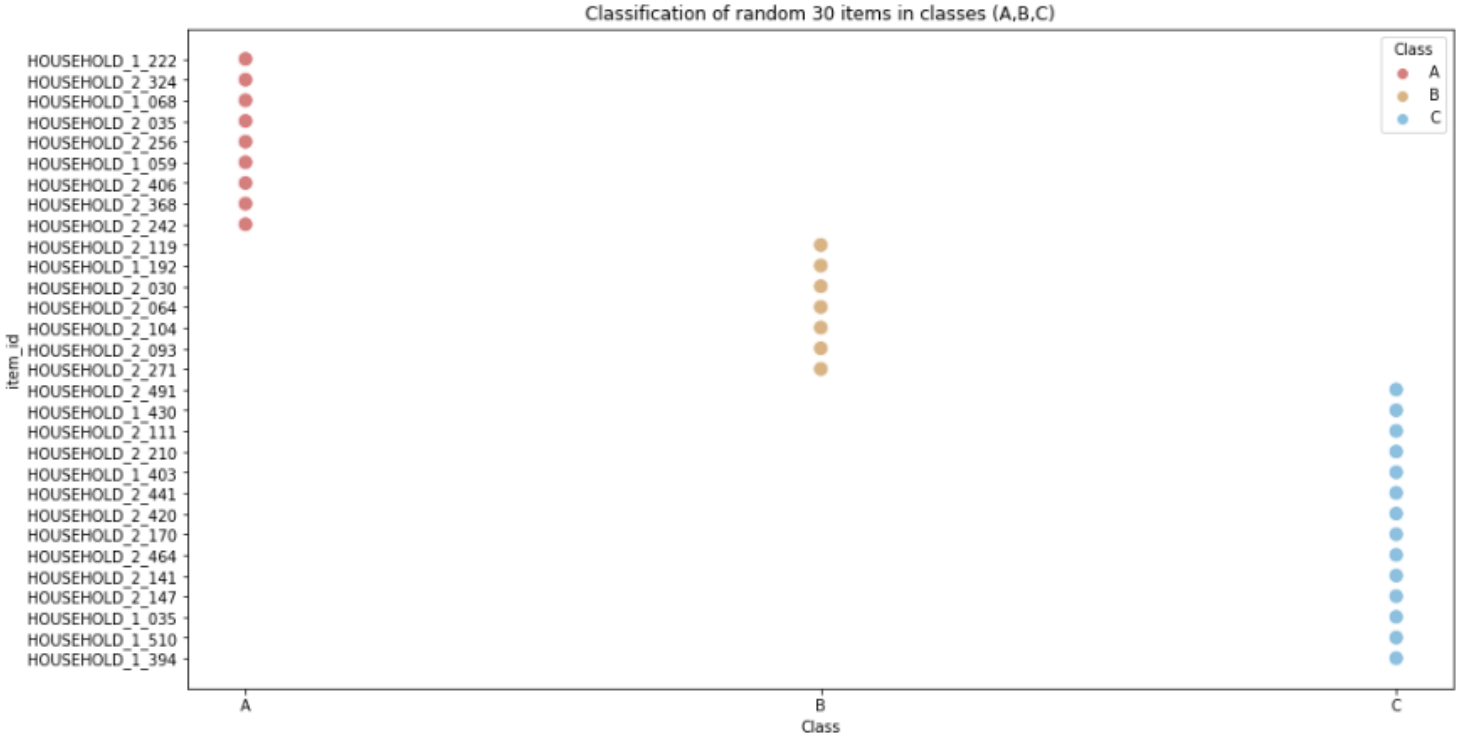


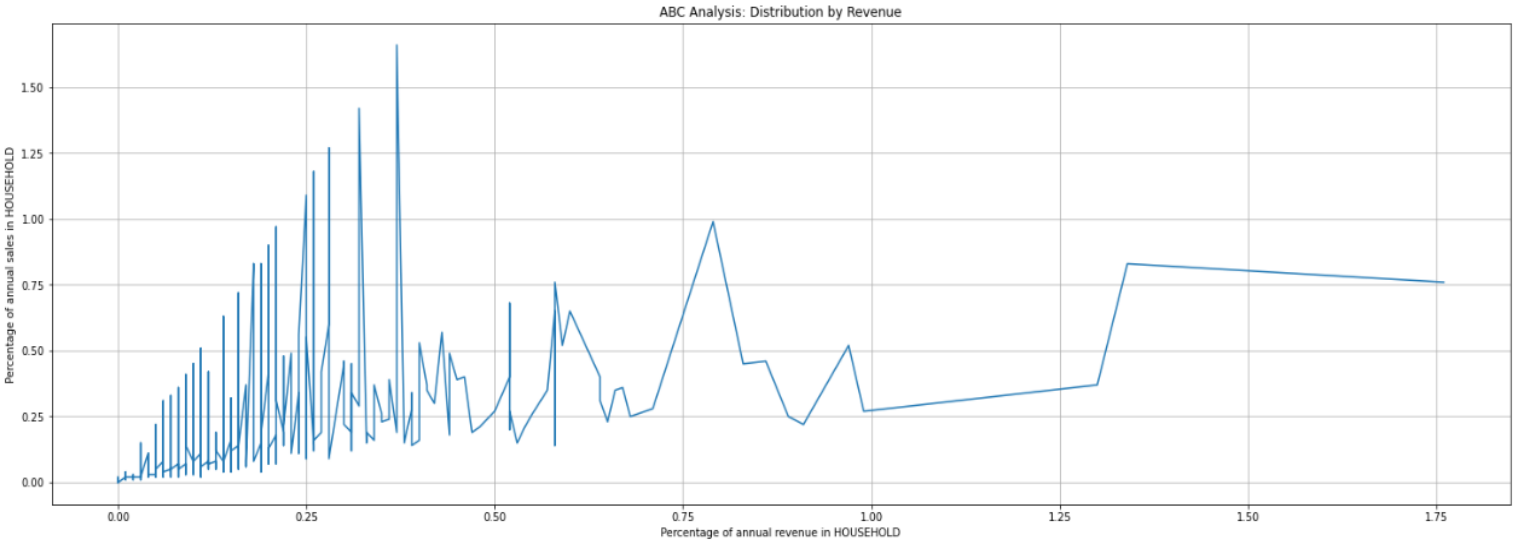





Bar-Graph for Avg Sales according to each class in HOUSEHOLD Category
16658.58
5489.15
1791.26
revenue
Class
A
B
C


Pie Chart showing total sales for each class in HOUSEHOLD Category
A
75.8%
6.2%
C
18.0%
B

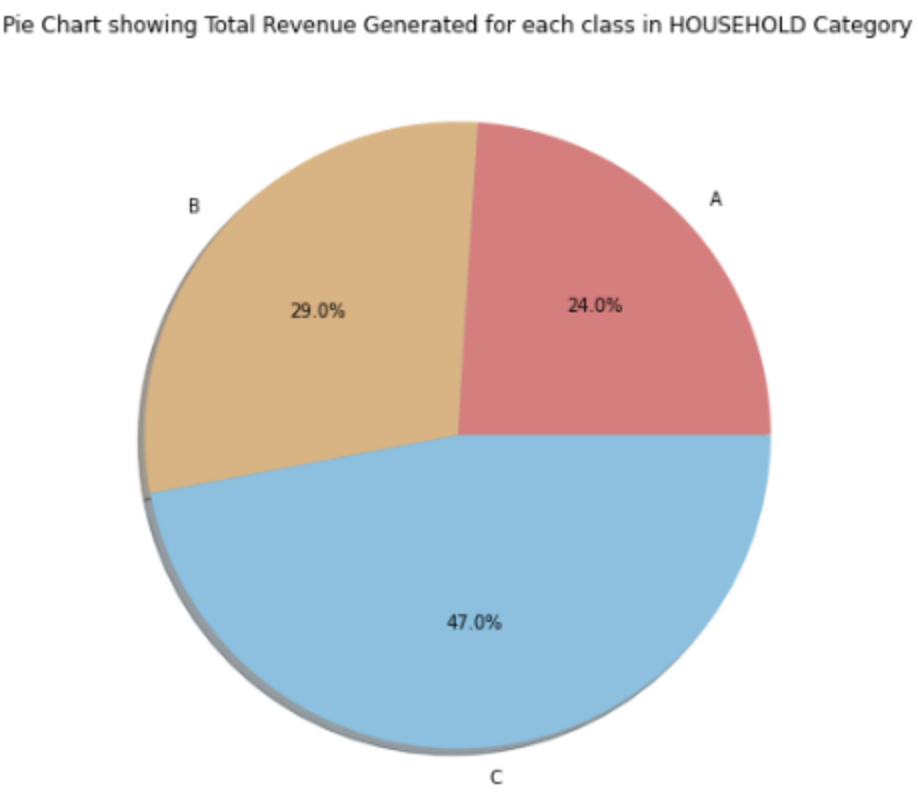

Classification of random 30 items in classes (A,B,C)
Class
A
B
C
item_id
HOUSEHOLD_1_222
HOUSEHOLD_2_324
HOUSEHOLD_1_068
HOUSEHOLD_2_035
HOUSEHOLD_2_256
HOUSEHOLD_1_059
HOUSEHOLD_2_406
HOUSEHOLD_2_368
HOUSEHOLD_2_242
HOUSEHOLD_2_119
HOUSEHOLD_1_192
HOUSEHOLD_2_030
HOUSEHOLD_2_064
HOUSEHOLD_2_104
HOUSEHOLD_2_093
HOUSEHOLD_2_271
HOUSEHOLD_2_491
HOUSEHOLD_1_430
HOUSEHOLD_2_111
HOUSEHOLD_2_210
HOUSEHOLD_1_403
HOUSEHOLD_2_441
HOUSEHOLD_2_420
HOUSEHOLD_2_170
HOUSEHOLD_2_464
HOUSEHOLD_2_141
HOUSEHOLD_2_147
HOUSEHOLD_1_035
HOUSEHOLD_1_510
HOUSEHOLD_1_394
A
B
C
Class



We further classify the items into A, B and C categories, just we consider the top 5% of the items in class A, next 15% in class B and the remaining 80% in class C to see if the division of revenue over the items follow the Pareto Law or not.

Following is the sample of the data after implementing the above mentioned:

item_id	sell_price	sales	revenue	percentage	Class	Pareto_Classification
HOUSEHOLD_1_272	9.97	11045	110118.65	1.76	A	A
HOUSEHOLD_1_535	6.97	12046	83960.62	1.34	A	A
HOUSEHOLD_1_053	14.97	5401	80852.97	1.30	A	A
HOUSEHOLD_1_537	15.97	3866	61740.02	0.99	A	A
HOUSEHOLD_1_177	7.97	7636	60858.92	0.97	A	A



OBSERVATION: The above graph shows that the annual revenue of the household items is not divided according to the Pareto Law, which states that 80% of the revenue comes from the top 5% of the items.

However, we can observe that 80% of the revenue comes from the items from classes A and B combined, some items from class, in contradiction to the Pareto Law.

We can use these observations to make changes to the business model and the way the inventory is organized.

B. Customers' demand stability (Coefficient of Variation). Computation of coefficient of variation of the yearly distribution of sales of each reference to understand which products will bring planning and distribution challenges.

We calculate the Coefficient of Variance to gauge the demand for which items tend to vary more and which items need to be kept a close eye on. Below table shows a sample of first 5 items with their standard deviation and Coefficient of Variance:

item_id	std	CV
HOUSEHOLD_1_001	2.063595	0.648761
HOUSEHOLD_1_002	2.171559	0.649156
HOUSEHOLD_1_004	12.215944	1.333379
HOUSEHOLD_1_005	5.464366	0.798756
HOUSEHOLD_1_006	4.543668	1.312056

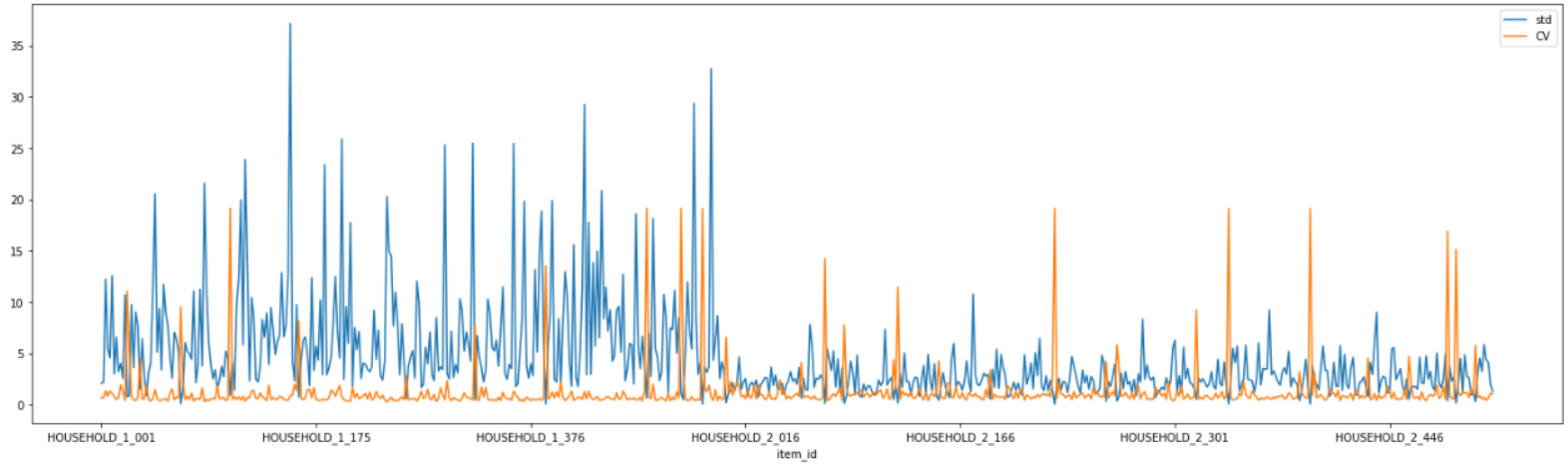


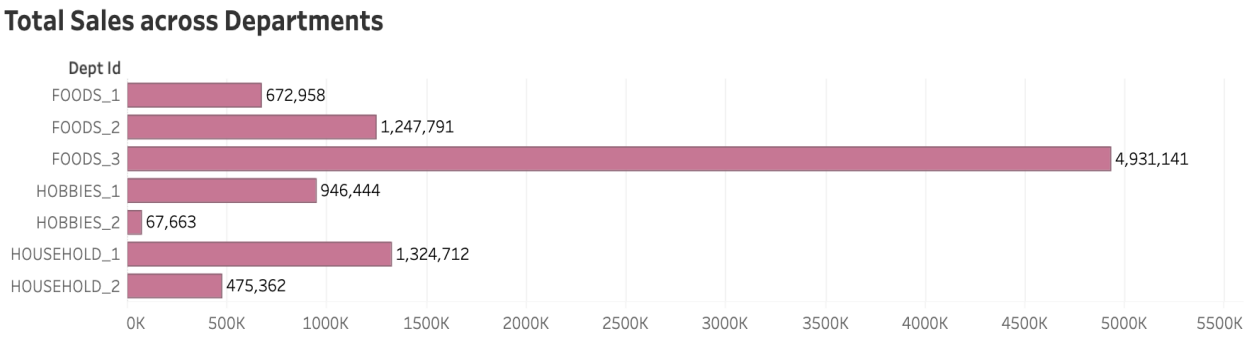
Fig: Distribution of Standard Deviation and CoV over the HOUSEHOLD items

OBSERVATIONS: From the above graph, we can observe that the sales of items in the range of about 125 to 150 deviate the most from the mean sales and would constitute most to the overall sales.

We also observe that sales of some of the items like item HOUSEHOLD_1_100, HOUSEHOLD_1_180, HOUSEHOLD_2_001, etc tend to vary more than other items. These items need to be accounted for more and their counting should be done more often compared to other items.

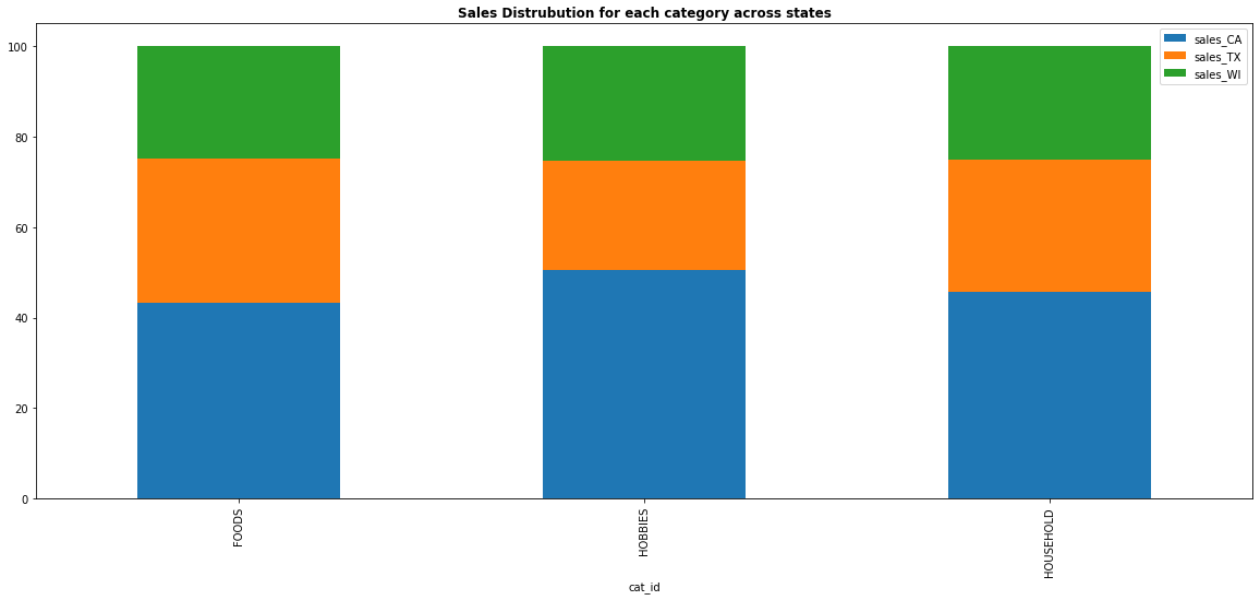
c. Initiatives and recommendations for improving the retail business for dataset_02.

1. The plot below shows the total sales across each department in the three states CA, TX and WI in one year (365 days).



From the above sales distribution plot, we can observe that the maximum sales is in the FOOD_3 department. We can assume that these could be regular foods such as milk, bread, vegetables, fruits, salt, meat, etc. Since the consumption of these food items is done on a regular basis, they are prone to run out of stock often. We recommend an increase in the restocking of the items in this department and this stocking should be done on a regular basis. Hence, even at odd times, the demand of the customers for these items will be met ensuring more customer satisfaction and increase in profit.

2. The plot below shows the sales distribution in each category (FOODS, HOBBIES, HOUSEHOLD) across the states (CA, TX, WI).



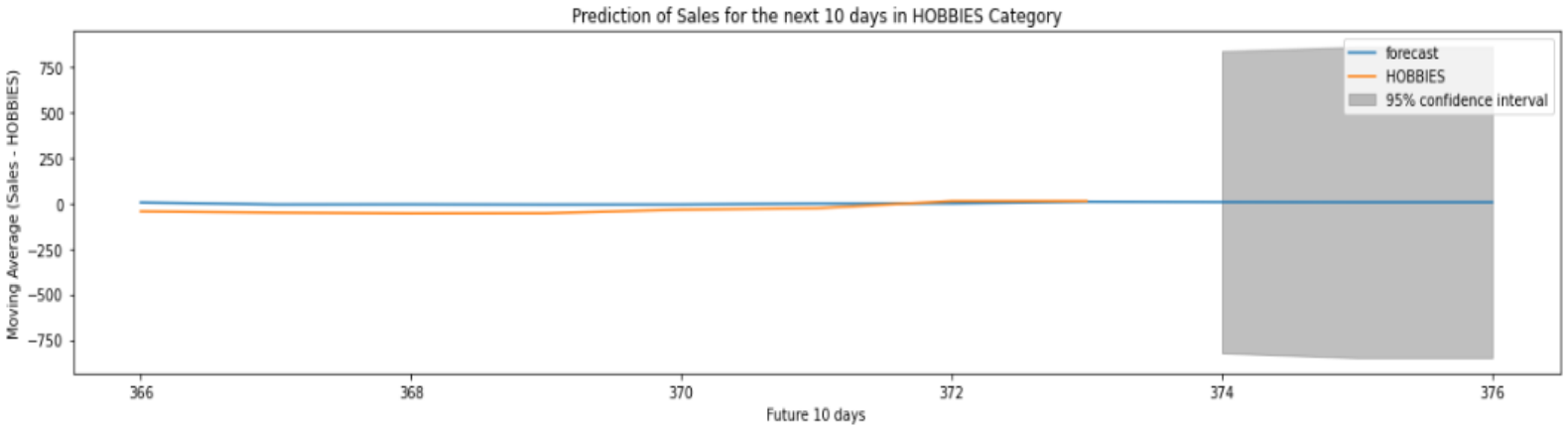
The table below shows the sales in percentages for each category (FOODS, HOBBIES, HOUSEHOLD) across the states (CA, TX, WI).

```
data_in
```

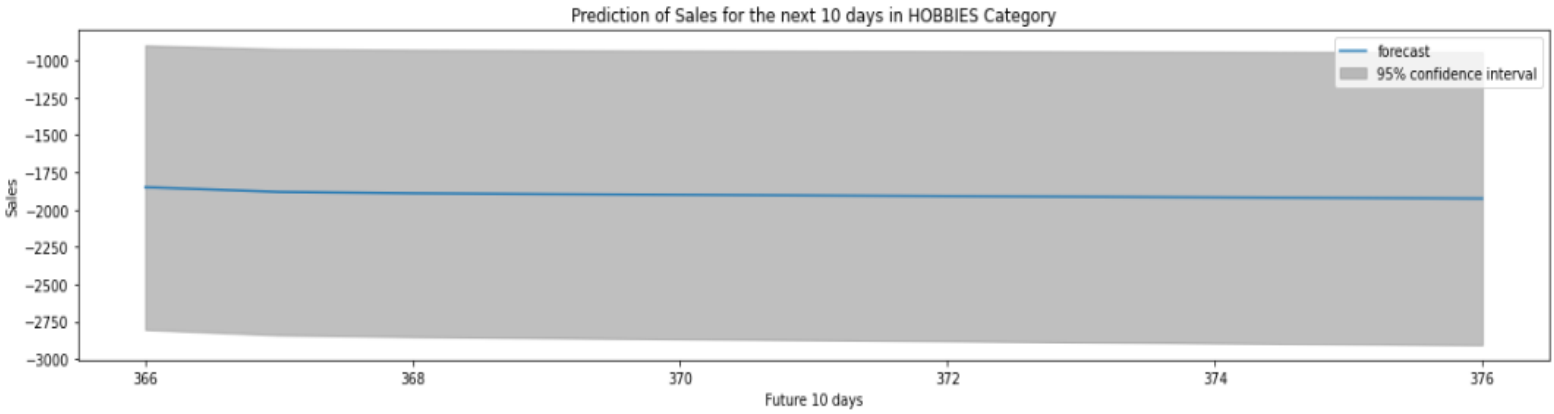
cat_id	sales_CA	sales_TX	sales_WI
FOODS	43.37492	31.85724	24.76784
HOBBIES	50.53875	24.09529	25.36596
HOUSEHOLD	45.71923	29.12053	25.16024

It can be observed that California constitutes 50% of the total sales in the HOBBIES category. This shows that CA residents are more willing to spend on their hobbies. Hence, we recommend to include more diverse items in this category (in the average and above average price range) to attract all sections of people. This will enhance the sales associated with Hobbies and help in increase in the overall profit.

We tried mutating a part of the original dataset and added a few more dummy items for the HOBBIES category and ran the ARIMA model on the new series to forecast the sales for the next 10 days. The model gave the following output:

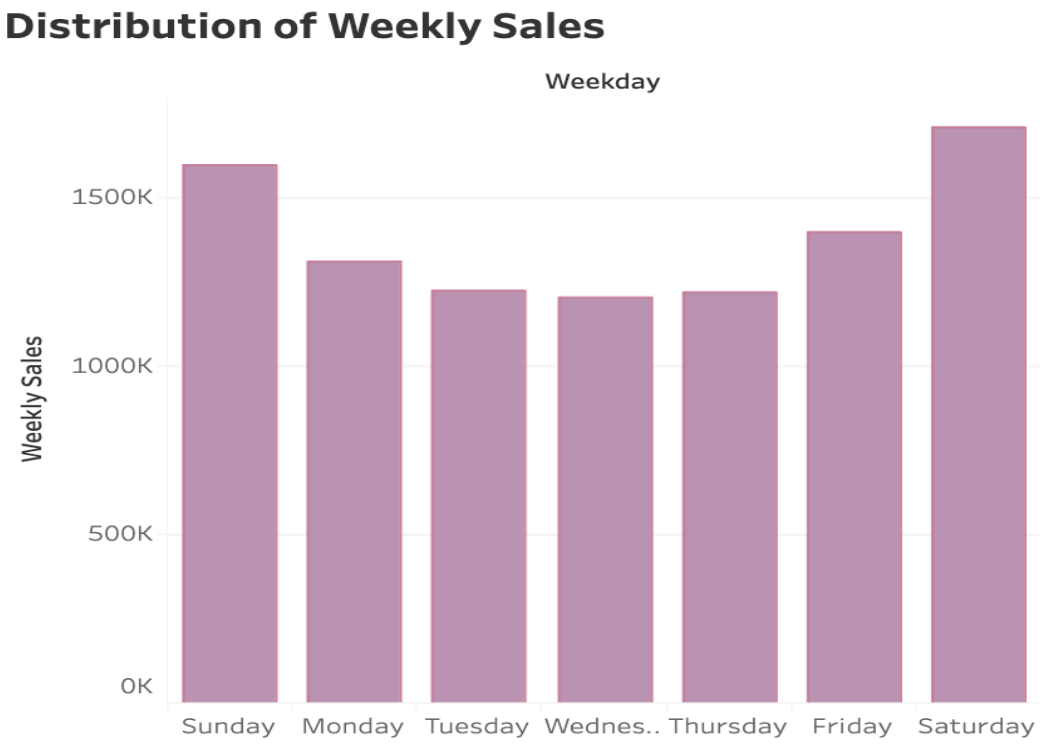


When we compare this to the original prediction without the new items as shown below:



We can see a significant increase in the number of forecasted sales. Hence, this initiative will certainly yield more sales in the future.

3. The plot below shows the weekly sales distribution in one year in all the categories of the items.



From the plot, it is observed that the overall sales decrease on weekdays as compared to the weekends. The given data does not provide any information about provision of delivery services across the stores. We assume delivery services are not available. We recommend the provision of free delivery service on an order of above $25 during weekdays and a discount of 10% on the

order consisting of top 20% items that are in demand. This would aid in the increase in the total sales during weekdays and hence will increase the average weekly sales too.

Contribution :

	Sprint1	Sprint2	Sprint3
Almee Christian	Tableau Dashboard	Feature Engineering,LSTM, performance matrix	ABC analysis,Initiatives
Harsha Bhargav	Q2 C,E	ADA Boost, CNN,Ensemble models	Coefficient of variance
Pragya Sharma	Tableau Dashboard	Data Preprocessing, EDA, Arima for all 3 categories	Product segmentation,Initiatives
Suraj Shah	Q2 A(EDA)	Arima, Ensemble models	Plots for ABC analysis, Initiatives
Taher Vora	Q2 B,D	Linear regression, XGBoost, Ridge regression, Random Forest Classifier.	Model deployment in Streamlit

---