ECOLE NORMALE SUPÉRIEURE DE L'ENSEIGNEMENT
TECHNIQUE DE MOHAMMEDIA
UNIVERSITÉ HASSAN II DE CASABLANCA

Machine Learning : Dynamic Inventory Management

Réalisé par : ELFAKIR CHAIMAE

AIT JEDDI ASSIA

Encadré par : Mr. Soufiane Hamida

SUMMARY

Table of Contents

Project Summary

This project focuses on inventory management, utilizing predictive analytics and machine learning algorithms for sales forecasting and inventory optimization. By analyzing historical sales data and other relevant factors, sophisticated algorithms are employed to accurately forecast future sales.

These predictive models consider parameters such as seasonality, trends, and external factors impacting demand. Additionally, the project focuses on inventory optimization, dynamically adjusting stock levels based on demand forecasts to ensure optimal levels and minimize costs. Automation of key inventory management processes through machine learning enhances operational efficiency and enables real-time decision-making. Ultimately, this project offers a comprehensive solution for businesses to make data-driven decisions, adapt to market dynamics, and achieve sustainable growth.

1 Inventory Management

1.1 Introduction

Inventory management is a fundamental pillar of supply chain management, orchestrating the seamless flow of goods from manufacturers to warehouses and ultimately to end customers. It encompasses a diverse array of processes and strategies aimed at efficiently controlling, tracking, and organizing inventory levels to satisfy customer demand while minimizing costs and maximizing profitability.

1.2 Core Components of Inventory Management:

Inventory Tracking and Control: This involves the continuous monitoring of inventory levels and movements throughout the supply chain, including activities such as receiving, storing, picking, packing, and shipping goods, all while maintaining meticulous records of inventory transactions.

Demand Forecasting: Accurate demand forecasting is indispensable in inventory management. By scrutinizing historical sales data, market trends, and other pertinent factors, businesses can predict future demand patterns, enabling informed decisions on stock levels to avoid overstocking or stockouts.

Inventory Optimization: Optimal inventory levels are paramount to balancing supply and demand effectively. Establishing appropriate reorder points, safety stock levels, and economic order quantities (EOQ) minimizes costs while ensuring adequate stock availability. This entails leveraging data analysis techniques such as time series analysis and machine learning models.

Supplier Management: Effective inventory management hinges on robust supplier relationships. Selecting dependable suppliers, negotiating favorable terms, and ensuring punctual deliveries are pivotal for maintaining a consistent supply of goods.

Warehouse Management: Efficient warehouse management is imperative for inventory optimization. This encompasses optimizing layout, organizing inventory for facile access, implementing efficient picking and packing processes, and minimizing handling and storage costs.

1.3 Project Focus:

In addition to these core components, our project will specifically focus on two pivotal aspects of inventory management:

Sales Forecasting: Sales forecasting, a subset of demand forecasting, estimates customer demand based on historical sales data and other pertinent factors. It informs planning across supply chain processes, including material procurement, purchasing, logistics, and distribution. Our project will utilize machine learning models in conjunction with time series analysis to predict future sales accurately.

Inventory Optimization: Inventory optimization ensures judicious resource allocation by determining optimal stock levels. Leveraging data analysis and mathematical modeling, businesses can minimize costs while meeting customer demand effectively. Our project will further elucidate how accurate sales forecasts

inform inventory management decisions, enabling the calculation of safety stock and reorder points for optimal inventory management.

1.4 Conclusion:

Inventory management is an intricate discipline pivotal for business prosperity. By incorporating machine learning models for sales forecasting and integrating forecasting sales with inventory optimization, our project endeavors to enhance operational efficiency, mitigate costs, and maximize profitability in the dynamic landscape of supply chain management.

2 Methodology

2.1 Introduction

This section contains detailed information about the dataset, the exact techniques that have been used in forecasting weekly sales and the last section talks about how this study is significant in predicting the weekly sales for Walmart stores. It will also discuss the success of the applied models in identifying the effect of different factors on such weekly sales

2.2 About Dataset:

The dataset utilized in this study was obtained from a previous Kaggle competition hosted by Walmart, accessible at the following link: Walmart Recruiting - Store Sales Forecasting. It comprises historical weekly sales data from 45 Walmart stores situated across diverse regions in the country, alongside department-wide information pertaining to these stores.

The 'test.csv' file within this dataset is utilized solely for predicting values generated by the model with the lowest Weighted Mean Absolute Error (WMAE) score. As the dataset lacks a target variable, specifically the 'Weekly Sales' data in our analysis, it cannot be utilized for testing purposes. Instead, the 'train.csv' dataset is divided into training and validation datasets for the study.

The primary objective of this study is to forecast department-wide weekly sales for each of the stores.

The dataset is already segregated into distinct training and testing data sets, with the testing data mirroring the training dataset except for the absence of weekly sales information. The training dataset encompasses weekly sales data spanning from February 5, 2010, to November 1, 2012, encompassing information on stores, departments, and an indication of whether a specific date corresponds to a holiday.

There are 3 Datasets:

Stores:

- Type: Three types of stores ‘A’, ‘B’ or ‘C’.
- Size: Sets the size of a Store would be calculated by the no. of products available in the store ranging from 34,000 to 210,000.

primary key is **Store**

```
178] walmart_store.head()
```

	Store	Type	Size
0	1	A	151315
1	2	A	202307
2	3	B	37392
3	4	A	205863
4	5	B	34875

Sales:

- Date: The date of the week where this observation was taken.
- Weekly_Sales: The sales recorded during that Week.
- Store: The store which observation in recorded 1–45
- Dept: One of 1–99 that shows the department.
- IsHoliday: Boolean value representing a holiday week or not.

primary key is a combination of (**Store**,**Dept**,**Date**).

```
[37] walmart.head()
```

	Store	Dept	Date	Weekly_Sales	IsHoliday
0	1	1	2010-02-05	24924.50	False
1	1	1	2010-02-12	46039.49	True
2	1	1	2010-02-19	41595.55	False
3	1	1	2010-02-26	19403.54	False
4	1	1	2010-03-05	21827.90	False

Features:

- Temperature: Temperature of the region during that week.
- Fuel_Price: Fuel Price in that region during that week.
- MarkDown1:5: Represents the Type of markdown and what quantity was available during that week.
- CPI: Consumer Price Index during that week.

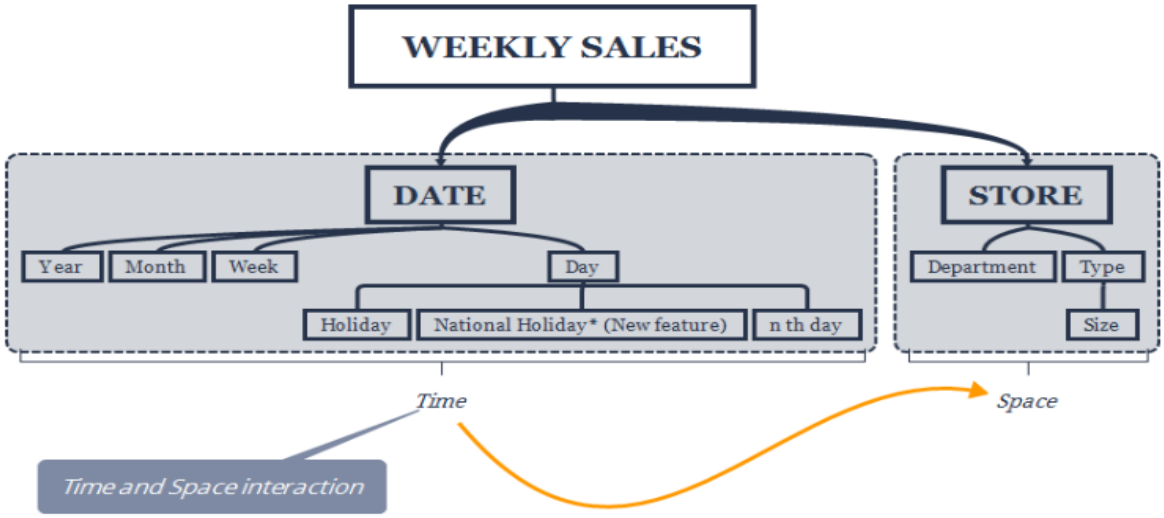
Unemployment: The unemployment rate during that week in the region of the store.

primary key here is a combination of (**Store**,**Date**)

```
[176] walmart_feature
```

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	IsHoliday
0	1	2010-02-05	42.31	2.572	NaN	NaN	NaN	NaN	NaN	211.096358	8.106	False
1	1	2010-02-12	38.51	2.548	NaN	NaN	NaN	NaN	NaN	211.242170	8.106	True
2	1	2010-02-19	39.93	2.514	NaN	NaN	NaN	NaN	NaN	211.289143	8.106	False
3	1	2010-02-26	46.63	2.561	NaN	NaN	NaN	NaN	NaN	211.319643	8.106	False
4	1	2010-03-05	46.50	2.625	NaN	NaN	NaN	NaN	NaN	211.350143	8.106	False
...	...	...	...	...	...	...	...	...	...	...	...	...
8185	45	2013-06-28	76.05	3.639	4842.29	975.03	3.00	2449.97	3169.69	NaN	NaN	False
8186	45	2013-07-05	77.50	3.614	9090.48	2268.58	582.74	5797.47	1514.93	NaN	NaN	False
8187	45	2013-07-12	79.37	3.614	3789.94	1827.31	85.72	744.84	2150.36	NaN	NaN	False
8188	45	2013-07-19	82.84	3.737	2961.49	1047.07	204.19	363.00	1059.46	NaN	NaN	False
8189	45	2013-07-26	76.06	3.804	212.02	851.73	2.06	10.88	1864.57	NaN	NaN	False

8190 rows × 12 columns



2.3 Exploratory Data Analysis

The initial step in the Exploratory Data Analysis (EDA) process involved merging three distinct datasets into a unified dataset for comprehensive analysis. Initially scattered across separate sources, the datasets were consolidated into a single cohesive entity to facilitate a holistic exploration of the data. Through this consolidation, redundancies were eliminated, and disparate pieces of information were integrated into a unified framework, streamlining the analysis process. As a result, the dataset underwent a significant reduction in size, decreasing from **421570** entries to **6435** entries. This consolidation not only simplified the dataset but also provided a solid foundation for subsequent EDA tasks, enabling a more focused and insightful exploration of key patterns and relationships within the data

```
[42]
data.head()
```

Store	Date	Weekly_Sales	Type	Size	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	IsHoliday
1	2010-02-05	1643690.90	A	151315	42.31	2.572	NaN	NaN	NaN	NaN	NaN	211.096358	8.106	False
1	2010-02-12	1641957.44	A	151315	38.51	2.548	NaN	NaN	NaN	NaN	NaN	211.242170	8.106	True
1	2010-02-19	1611968.17	A	151315	39.93	2.514	NaN	NaN	NaN	NaN	NaN	211.289143	8.106	False
1	2010-02-26	1409727.59	A	151315	46.63	2.561	NaN	NaN	NaN	NaN	NaN	211.319643	8.106	False
1	2010-03-05	1554806.68	A	151315	46.50	2.625	NaN	NaN	NaN	NaN	NaN	211.350143	8.106	False

2.3.1 Descriptive Statistics :

```
[49] data.describe().T
```

	count	mean	min	25%	50%	75%	max	std
Store	6435.0	23.0	1.0	12.0	23.0	34.0	45.0	12.988182
Date	6435	2011-06-17 00:00:00	2010-02-05 00:00:00	2010-10-08 00:00:00	2011-06-17 00:00:00	2012-02-24 00:00:00	2012-10-26 00:00:00	NaN
Weekly_Sales	6435.0	1046964.877562	209986.25	553350.105	960746.04	1420158.66	3818686.45	564366.622054
Size	6435.0	130287.6	34875.0	70713.0	126512.0	202307.0	219622.0	63117.022465
Temperature	6435.0	60.663782	-2.06	47.46	62.67	74.94	100.14	18.444933
Fuel_Price	6435.0	3.358607	2.472	2.933	3.445	3.735	4.468	0.45902
MarkDown1	2280.0	6855.58743	0.27	1679.19	4972.59	8873.5825	88646.76	8183.310015
MarkDown2	1637.0	3218.965504	-265.76	37.2	187.04	1785.29	104519.54	9268.082387
MarkDown3	2046.0	1349.853021	-29.1	4.7	22.7	99.9875	141630.61	9287.2428
MarkDown4	1965.0	3303.858142	0.22	483.27	1419.42	3496.08	67474.85	6211.203947
MarkDown5	2295.0	4435.26224	135.16	1702.565	3186.52	5422.08	108519.28	5868.933325
CPI	6435.0	171.578394	126.064	131.735	182.616521	212.743293	227.232807	39.356712
Unemployment	6435.0	7.999151	3.879	6.891	7.874	8.622	14.313	1.875885
IsHoliday	6435.0	0.06993	0.0	0.0	0.0	0.0	1.0	0.255049
Day	6435.0	15.678322	1.0	8.0	16.0	23.0	31.0	8.75578
Year	6435.0	2010.965035	2010.0	2010.0	2011.0	2012.0	2012.0	0.797019

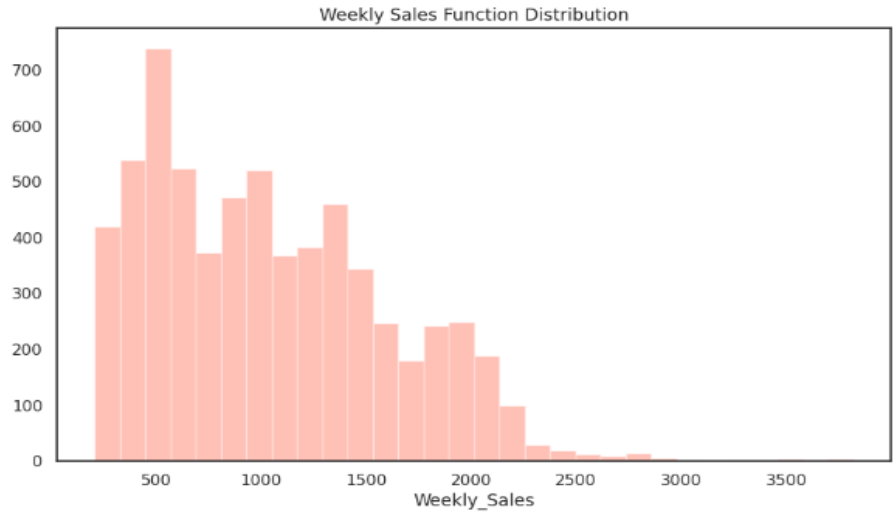
```
[51] numeric_data = data.select_dtypes(include=['float64', 'int64'])
     numeric_summary = numeric_data.describe().T.style.bar(subset=['mean'], color='#205ff2')\
                               .background_gradient(subset=['std'], cmap='Reds')\
                               .background_gradient(subset=['50%'], cmap='coolwarm')
     numeric_summary
```

	count	mean	std	min	25%	50%	75%	max
Store	6435.000000	23.000000	12.988182	1.000000	12.000000	23.000000	34.000000	45.000000
Weekly_Sales	6435.000000	1046964.877562	564366.622054	209986.250000	553350.105000	960746.040000	1420158.660000	3818686.450000
Size	6435.000000	130287.600000	63117.022465	34875.000000	70713.000000	126512.000000	202307.000000	219622.000000
Temperature	6435.000000	60.663782	18.444933	-2.060000	47.460000	62.670000	74.940000	100.140000
Fuel_Price	6435.000000	3.358607	0.459020	2.472000	2.933000	3.445000	3.735000	4.468000
MarkDown1	2280.000000	6855.587430	8183.310015	0.270000	1679.190000	4972.590000	8873.582500	88646.760000
MarkDown2	1637.000000	3218.965504	9268.082387	-265.760000	37.200000	187.040000	1785.290000	104519.540000
MarkDown3	2046.000000	1349.853021	9287.242800	-29.100000	4.700000	22.700000	99.987500	141630.610000
MarkDown4	1965.000000	3303.858142	6211.203947	0.220000	483.270000	1419.420000	3496.080000	67474.850000
MarkDown5	2295.000000	4435.262240	5868.933325	135.160000	1702.565000	3186.520000	5422.080000	108519.280000
CPI	6435.000000	171.578394	39.356712	126.064000	131.735000	182.616521	212.743293	227.232807
Unemployment	6435.000000	7.999151	1.875885	3.879000	6.891000	7.874000	8.622000	14.313000
IsHoliday	6435.000000	0.069930	0.255049	0.000000	0.000000	0.000000	0.000000	1.000000

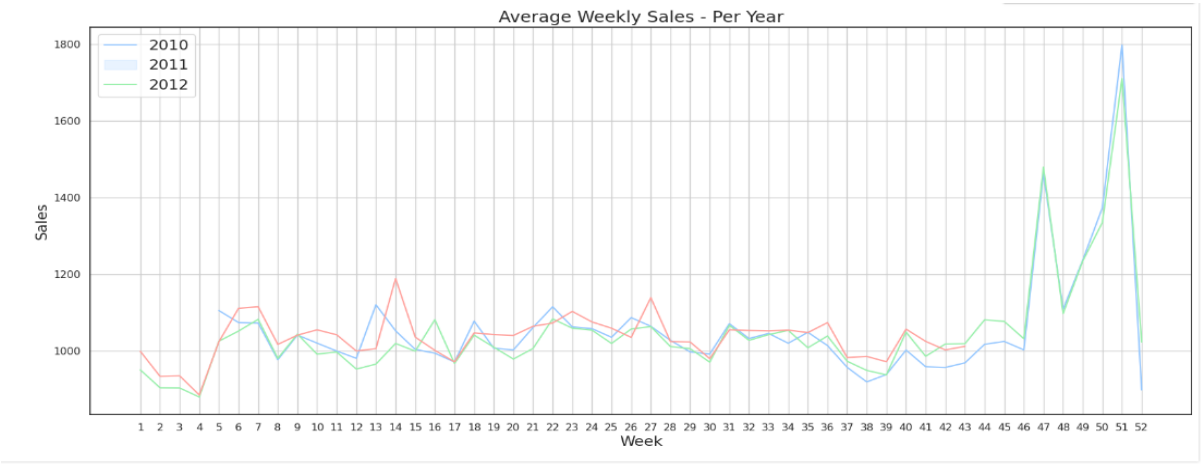
2-2- Data visualisation

2.3.1.1 Weekly Sales *Distribution* :

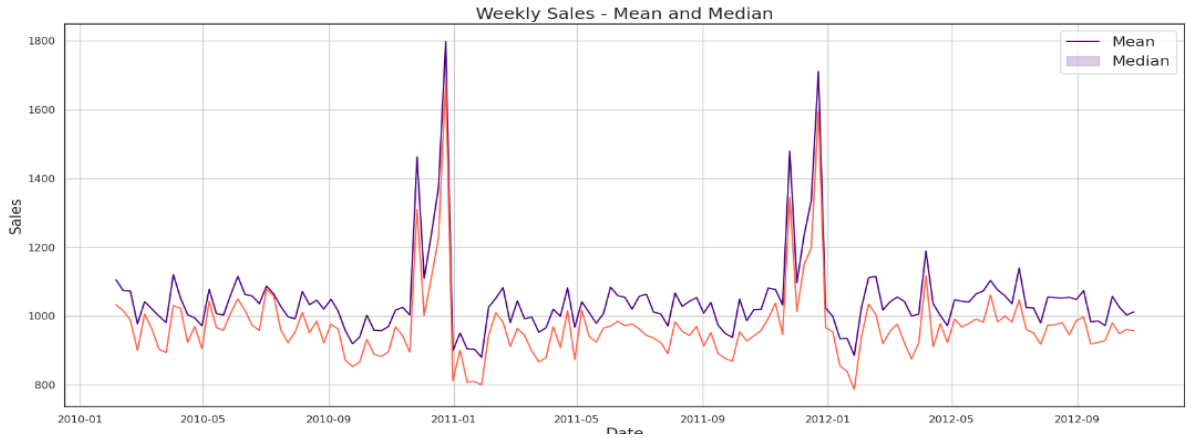
```
plt.figure(figsize=(10, 6))
data["Weekly_Sales"]=data.Weekly_Sales/1000

sns.distplot(data.Weekly_Sales, kde=False, bins=30, color = 'tomato')
plt.title('Weekly Sales Function Distribution')
plt.show()
```



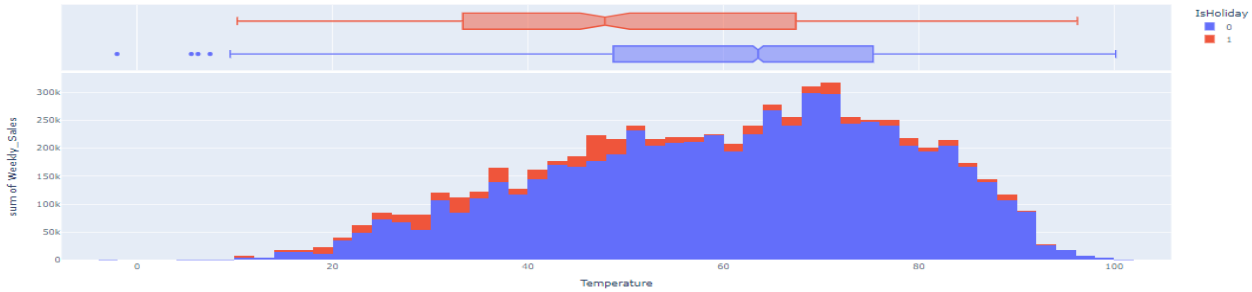
2.3.1.2 Average Weekly Sales –Per Year :



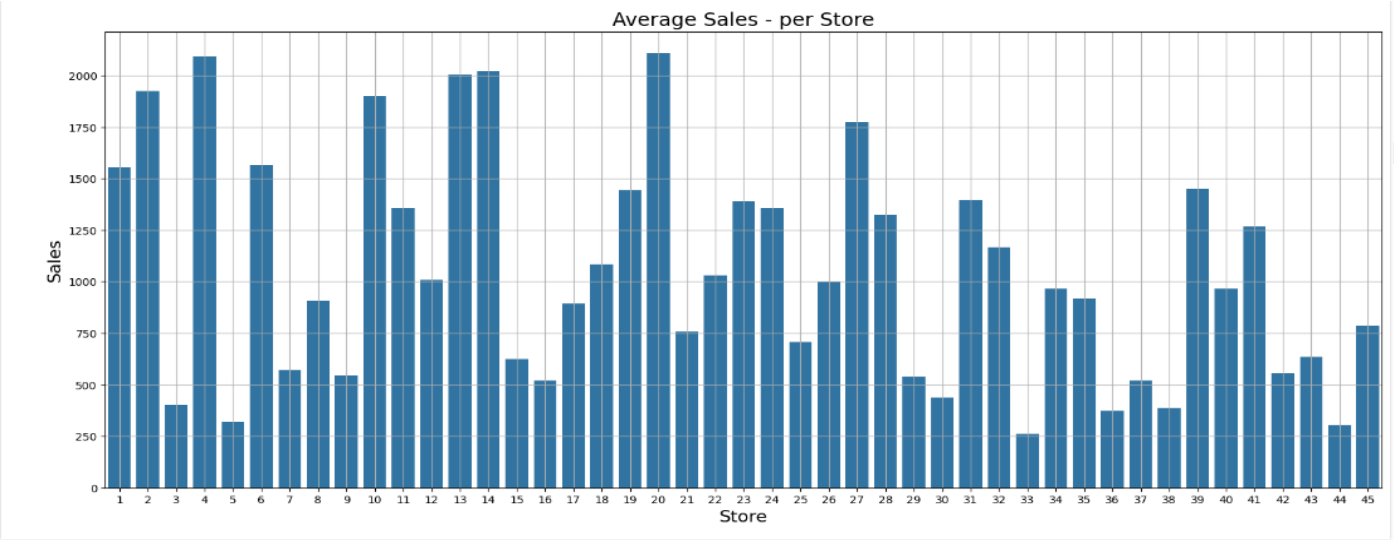
2.3.1.3 Weekly Sales – Mean and Median :



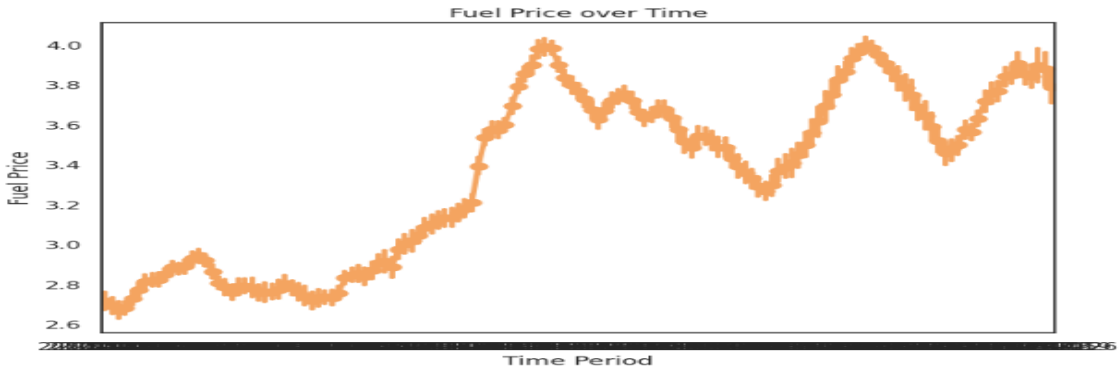
2.3.1.4 Sum of weekly sales by temperature:



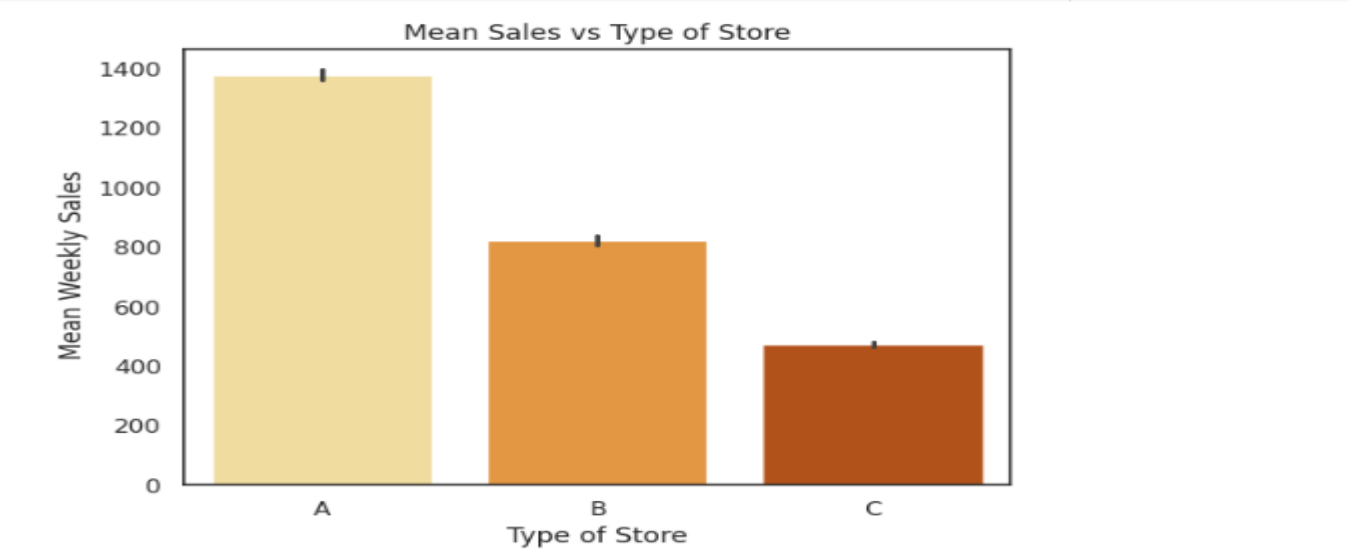
2.3.1.5 Average sales per store



2.3.1.6 Fuel price over time



2.3.1.7 Mean Sales vs Type of Store



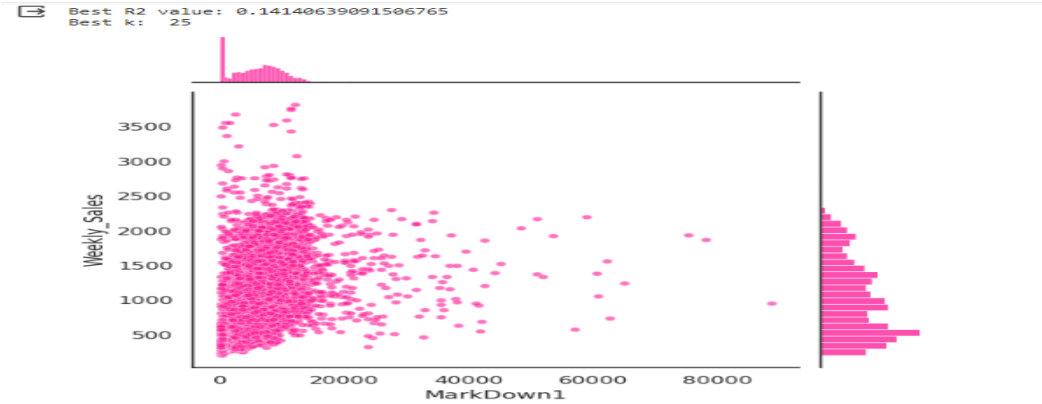
2.3.2 Missing Values

Since 65% of the markdown values are missing, we need to select an appropriate method for imputing these missing values.

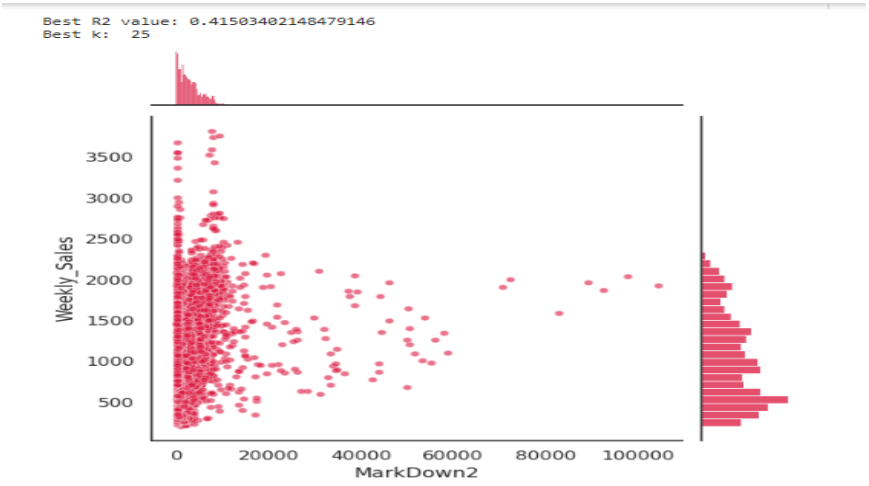
```
Store             0
Date              0
Weekly_Sales      0
Type              0
Size              0
Temperature       0
Fuel_Price        0
MarkDown1      4155
MarkDown2      4798
MarkDown3      4389
MarkDown4      4470
MarkDown5      4140
CPI               0
Unemployment      0
IsHoliday         0
Day               0
Month             0
Year              0
dtype: int64
```

The methodology employed for missing value imputation in the 'MarkDown' columns utilizes a k-nearest neighbors (KNN) regression approach. This method was chosen due to its ability to impute missing values based on the similarity between data points. By identifying similar instances with available 'MarkDown' values, the KNN algorithm predicts the missing values by considering the values of neighboring points. This approach is advantageous as it leverages the underlying patterns present in the dataset to estimate missing values, thereby preserving the integrity of the data. Additionally, KNN regression allows for flexibility in determining the number of neighbors (k) used for imputation, with the optimal k value selected through cross-validation to ensure robust performance. Overall, the KNN regression methodology provides an effective means of handling missing data while maintaining the integrity and accuracy of the dataset for subsequent analysis and modeling.

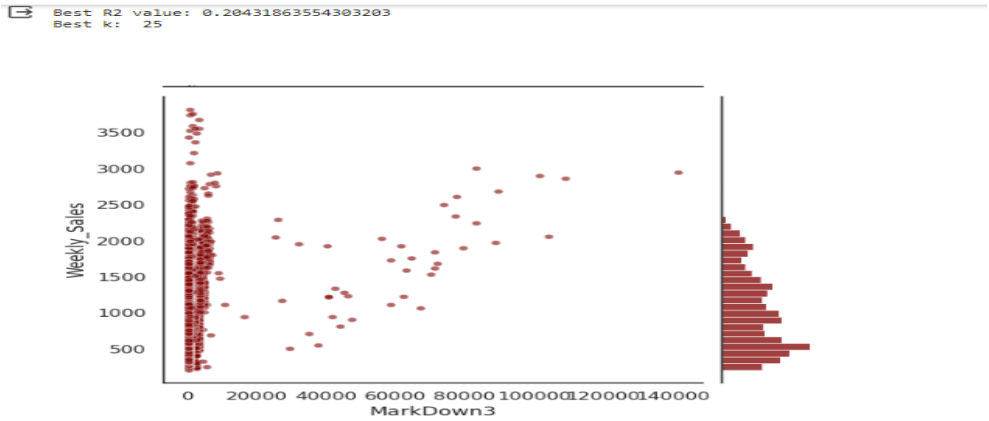
- Markdown 1 :



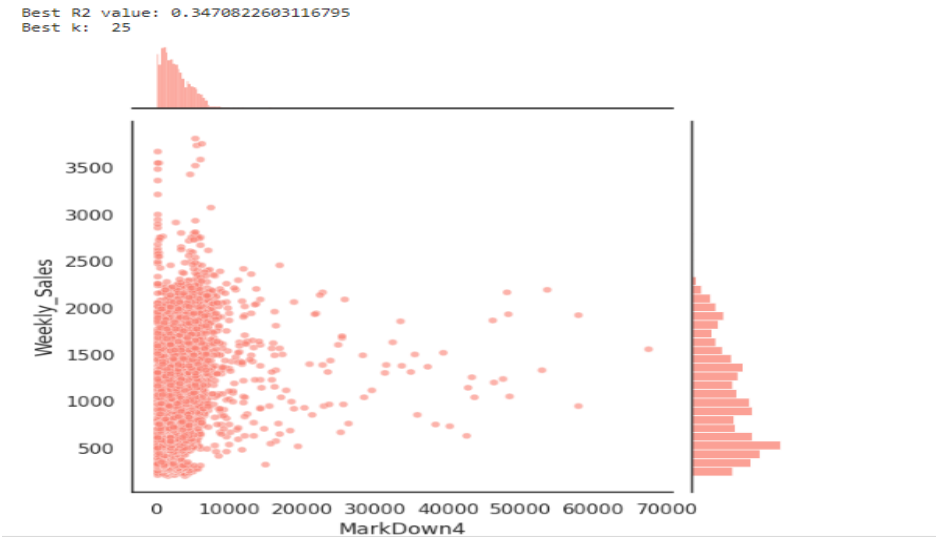
- Markdown 2 :



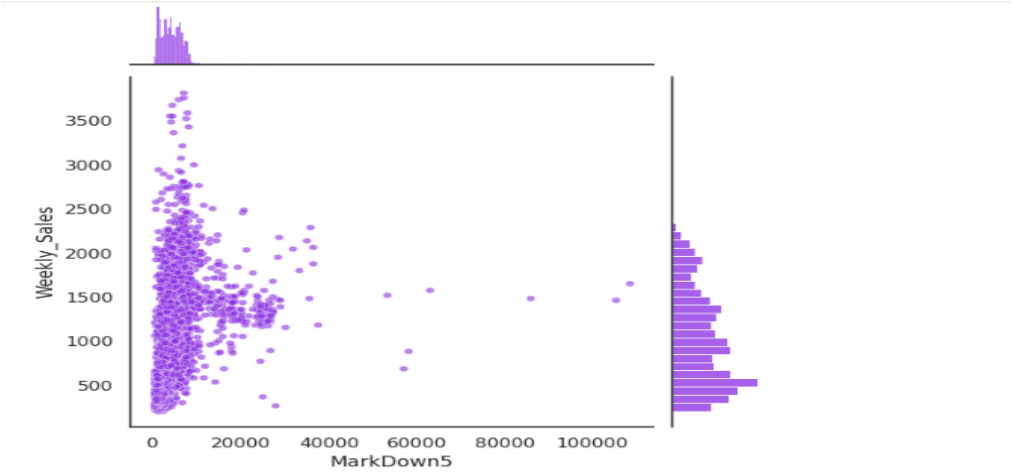
- Markdown 3 :



- Markdown 4 :



- Markdown 5 :



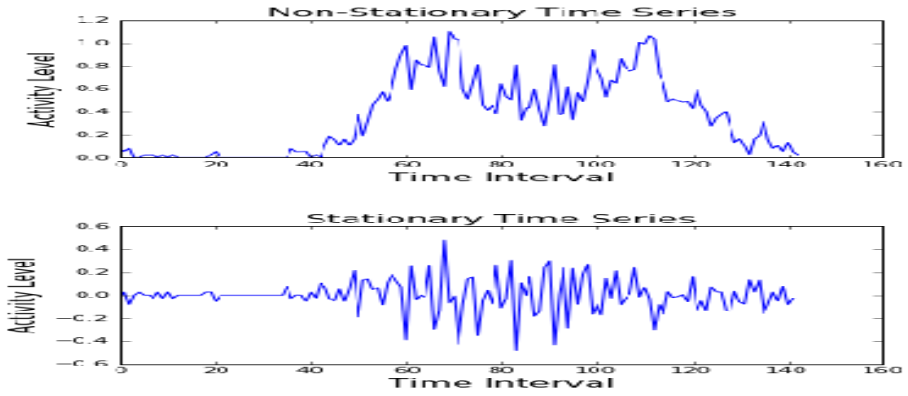
2.4 Time Series Analysis

Time Series is Any data recorded with some fixed interval of time is called as time series data. This fixed interval can be hourly, daily, monthly or yearly. e.g. hourly temp reading, daily changing fuel prices, monthly electricity bill, annul company profit report etc. In time series data, time will always be independent variable and there can be one or many dependent variable.

Objective of time series analysis is to understand how change in time affect the dependent variables and accordingly predict values for future time intervals.

2.4.1 Stationarity Data:

For accurate analysis and forecasting trend and seasonality is removed from the time series and converted it into **stationary series**. Time series data is said to be stationary when statistical properties like mean, standard deviation are constant and there is no seasonality. In other words statistical properties of the time series data should not be a function of time

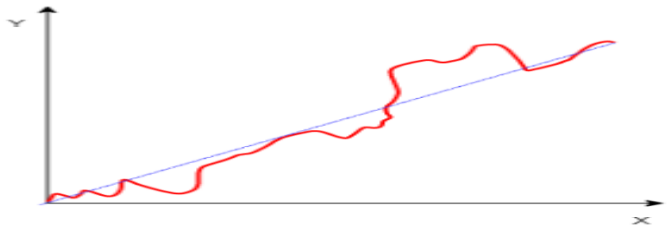


2.4.2 Time Series Components

The various reasons or the forces which affect the values of an observation in a time series are the components of a time series. The four categories of the components of time series are:

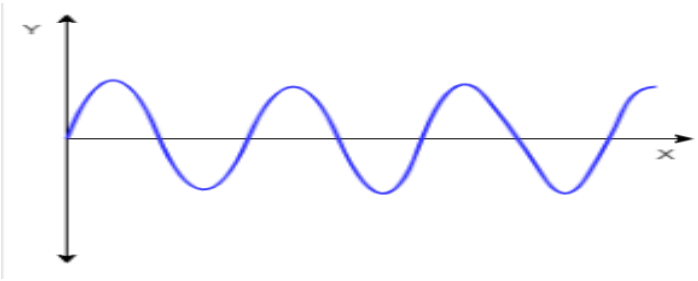
2.4.2.1 Trend

Trend represent the change in dependent variables with respect to time from start to end. In case of increasing trend dependent variable will increase with time and vice versa. It's not necessary to have definite trend in time series, we can have a single time series with increasing and decreasing trend. In short trend represent the varying mean of time series data.



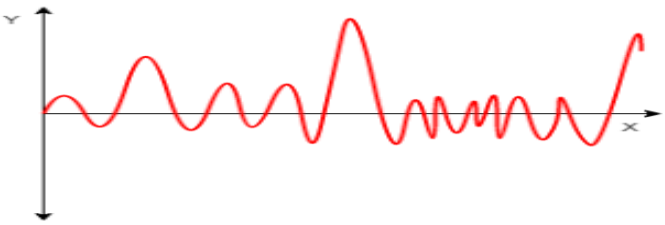
2.4.2.2 Seasonal Variations

If observations repeats after fixed time interval then they are referred as seasonal observations. These seasonal changes in data can occur because of natural events or man-made events. For example every year warm cloths sales increases just before winter season. So seasonality represent the data variations at fixed intervals.

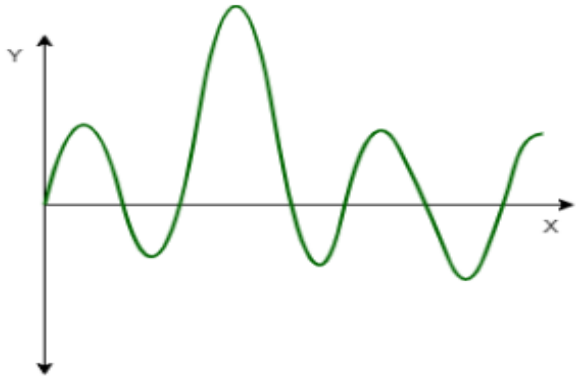


2.4.2.3 Cyclic Variations

Cyclicity occurs when observations in the series repeats in random pattern. Note that if there is any fixed pattern then it becomes seasonality, in case of cyclicity observations may repeat after a week, months or may be after a year. These kinds of patterns are much harder to predict.



2.4.2.4 Random or Irregular movements

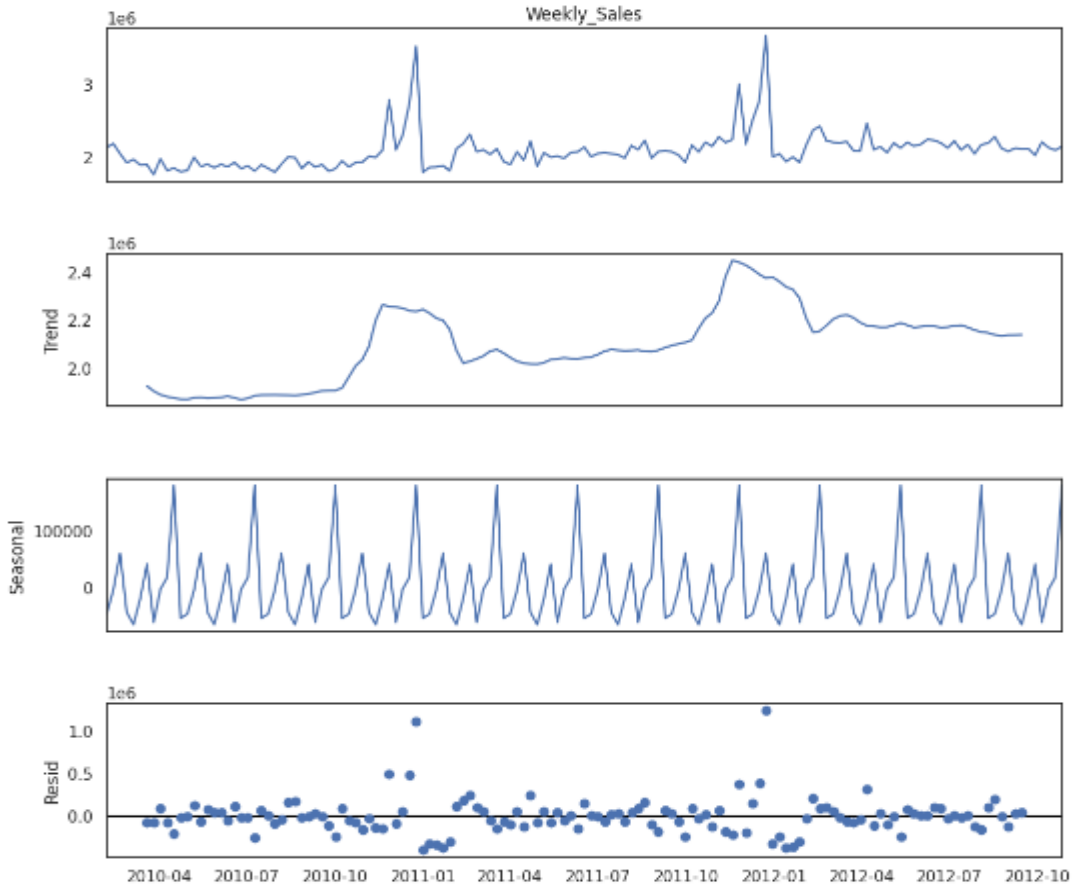


Seasonal and Cyclic Variations are the periodic changes or short-term fluctuations.

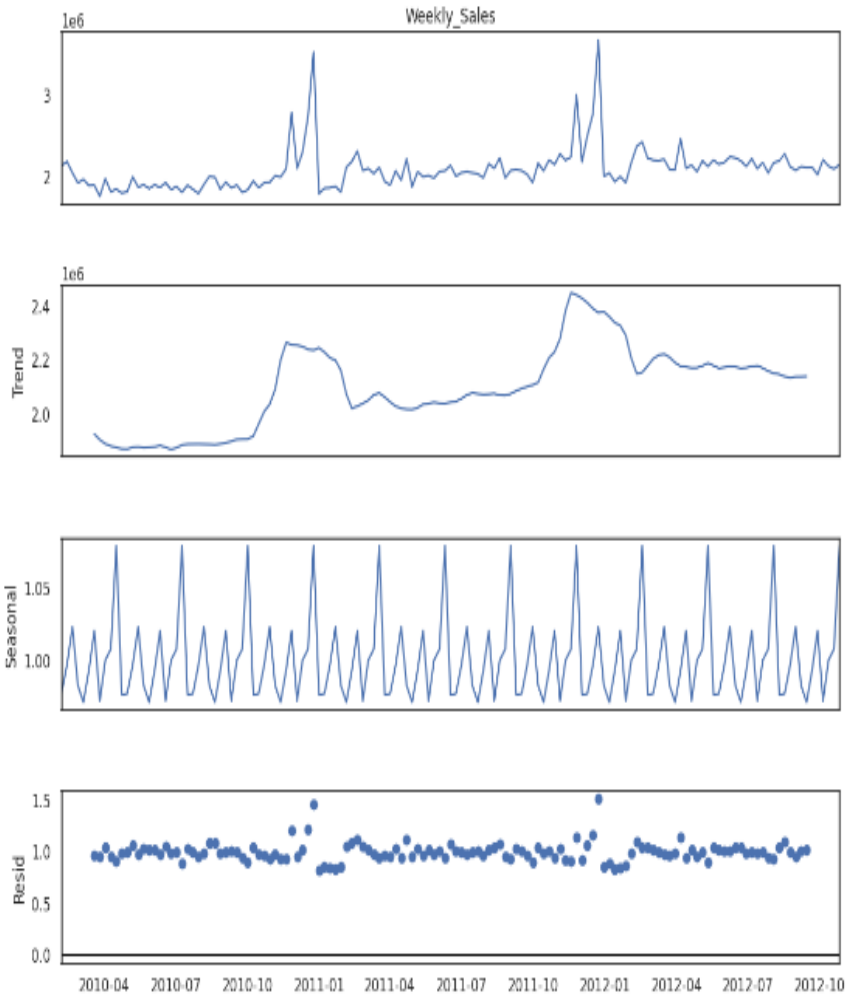
2.4.3 Time Series Decomposition

Time series decomposition helps to deconstruct the time series into several component like trend and seasonality for better visualization of its characteristics. Using time-series decomposition makes it easier to quickly identify a changing mean or variation in the data

2.4.3.1 Additive Decomposition:



2.4.3.2 Multiplicative Decomposition



2.5 Feature Engineering

Feature engineering is a critical step in the data preprocessing pipeline aimed at creating new, informative features from the existing dataset. In this section, we describe the feature engineering process undertaken to enhance the predictive power of our models. The feature engineering process consists of the following key steps:

1. Creation of Week Column:
   We introduced a new feature called "Week" to capture the week number of each month based on the "Day" column in the dataset. This allows us to analyze sales patterns and trends on a weekly basis.
2. Creation of Dummy Variables :

Dummy variables were generated for categorical variables such as "Store", "Month", "Year", and the newly created "Week". This was accomplished using the **pd.get_dummies()** function, which converts categorical variables into binary (0 or 1) indicator variables.

For each categorical variable, dummy variables were created with a prefix added to their names to distinguish them from the original variables.

These dummy variables encode categorical information into a format suitable for machine learning models, enabling us to incorporate categorical data into our predictive models effectively.

3. Concatenation of DataFrames:

The original DataFrame was concatenated with the dummy variable DataFrames created in the previous step using the **pd.concat()** function. This resulted in a single DataFrame containing the original features along with the newly created dummy variables.

The concatenated DataFrame serves as the input for subsequent modeling and analysis tasks, with the added benefit of enhanced feature representation.

	Date	Weekly_Sales	Size	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	...	Month_May	Month_Nov	Month_Oct	Month_Sep	Year_2011	Year_2012	Week_1.0	Week_2.
0	2010-02-05	1643.69090	151315	42.31	2.572	11093.5024	5086.0092	191.8768	11285.2364	5787.1100	...	False	False	False	False	False	False	True	Fals
1	2010-02-12	1641.95744	151315	38.51	2.548	9175.0396	6319.9408	2413.9308	7885.5248	6570.4340	...	False	False	False	False	False	False	False	Tru
2	2010-02-19	1611.96817	151315	39.93	2.514	7064.4916	5578.6700	147.6212	3686.7364	5574.9940	...	False	False	False	False	False	False	False	Fals
3	2010-02-26	1409.72759	151315	46.63	2.561	7525.9280	4976.8168	172.9892	4429.7088	5566.9504	...	False	False	False	False	False	False	False	Fals
4	2010-03-05	1554.80668	151315	46.50	2.625	9749.6264	4433.8064	169.8220	5213.9284	5286.0824	...	False	False	False	False	False	False	True	Fals
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
6430	2012-09-28	713.17395	118221	64.88	3.997	4556.6100	20.6400	1.5000	1601.0100	3288.2500	...	False	False	False	True	False	True	False	Fals
6431	2012-10-05	733.45507	118221	64.89	3.985	5046.7400	2984.0352	18.8200	2253.4300	2340.0100	...	False	False	True	False	False	True	True	Fals
6432	2012-10-12	734.46436	118221	54.47	4.000	1956.2800	3460.1736	7.8900	599.3200	3990.5400	...	False	False	True	False	False	True	False	Tru
6433	2012-10-19	718.12553	118221	56.47	3.969	2004.0200	3080.3176	3.1800	437.7300	1537.4900	...	False	False	True	False	False	True	False	Fals
6434	2012-10-26	760.28143	118221	58.85	3.882	4018.9100	58.0800	100.0000	211.9400	858.3300	...	False	False	True	False	False	True	False	Fals

6435 rows × 76 columns

2.6 Modeling

2.6.1 Split data:

We will forecasting for last 3 month (90 days) :

```
[85] # Setting the offset to finalize the test data.
     offset = timedelta(days=90)
     split_date=data.Date.max()-offset
     split_date

     Timestamp('2012-07-28 00:00:00')
```

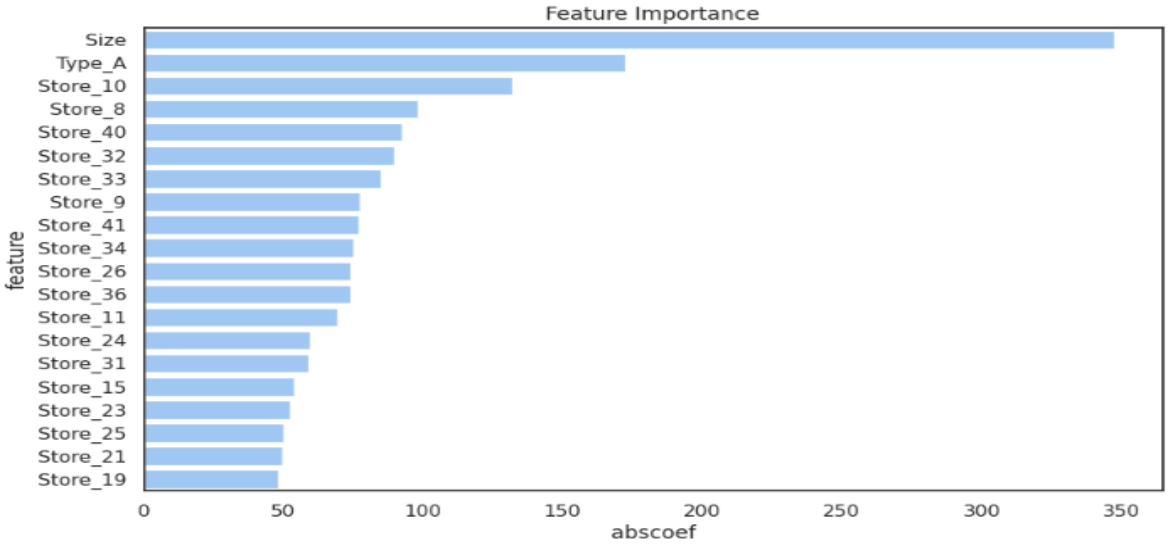
```
[86] data_train=data[data.Date < split_date]
     data_test=data[data.Date > split_date]
```

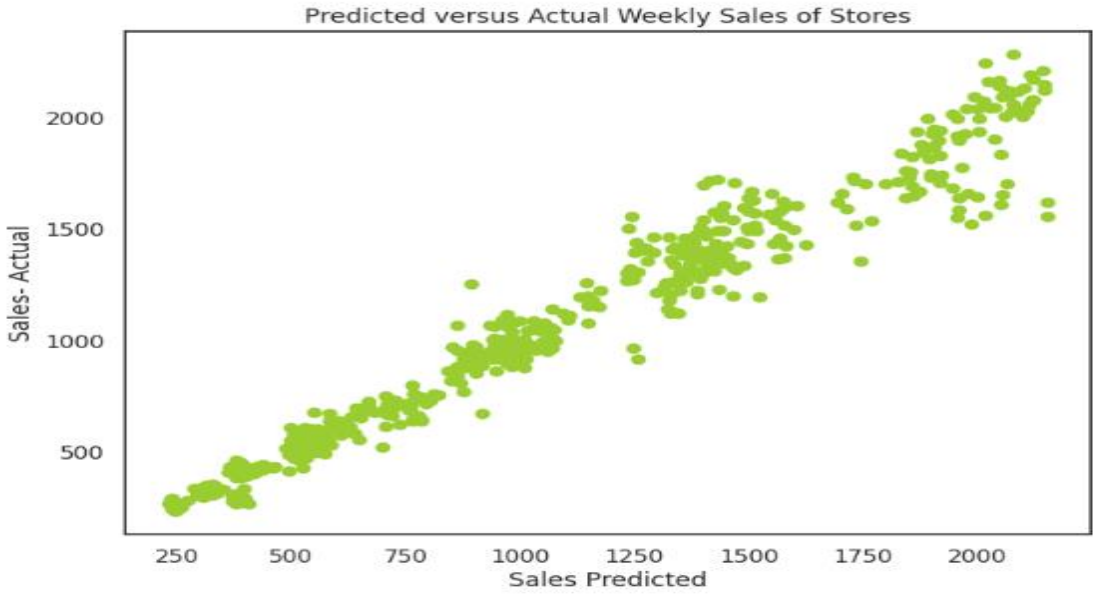
2.6.2 Lasso Regressor:

LASSO is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the statistical model it produces.

```
[103] lasso_cv = LassoCV(n_alphas=1000,max_iter=2000, cv=10, verbose=1)
      lasso_cv.fit(X_train_s, y_train)
```

```
LassoCV
LassoCV(cv=10, max_iter=2000, n_alphas=1000, verbose=1)
```



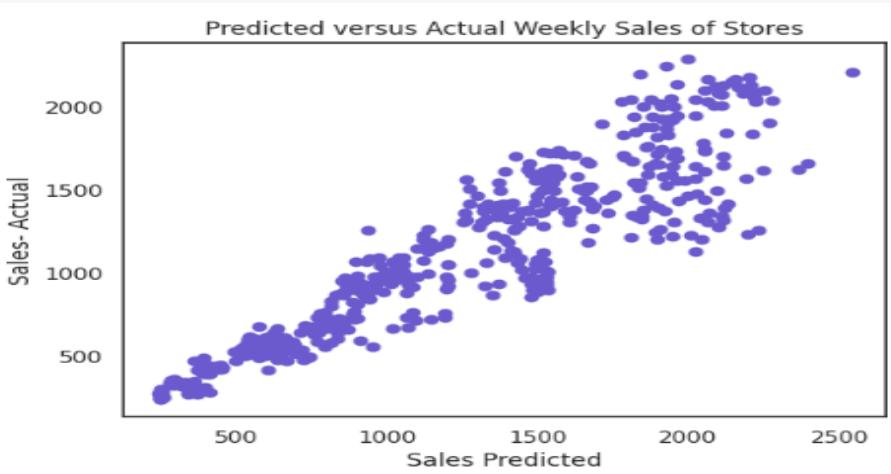


2.6.3 Gradient Boosting :

popular and powerful machine learning algorithm is based on the gradient boosting framework, which builds a predictive model by combining the predictions of multiple weak learners, typically decision trees. It optimizes a differentiable loss function by iteratively adding weak learners to minimize the loss.

2.6.4 Random Forest :

The Random Forest Regressor is a machine learning algorithm that constructs a collection of decision trees during training and makes predictions based on the average prediction of all individual trees. Each decision tree is built independently using a random subset of the training data and a random subset of features at each node



2.7 Model Evaluation:

For evaluating our sales forecasting models, we selected the R-squared (R^2) and Weighted Mean Absolute Error (WMAE) metrics. R^2 measures the proportion of variance in sales that is predictable from the model, providing insights into the goodness of fit. WMAE, on the other hand, enhances traditional mean absolute error by incorporating weights, reflecting the relative importance of different forecasts. Together, these metrics offer a comprehensive evaluation of model performance, balancing explanatory power with practical accuracy in predicting weekly sales for 45 stores.

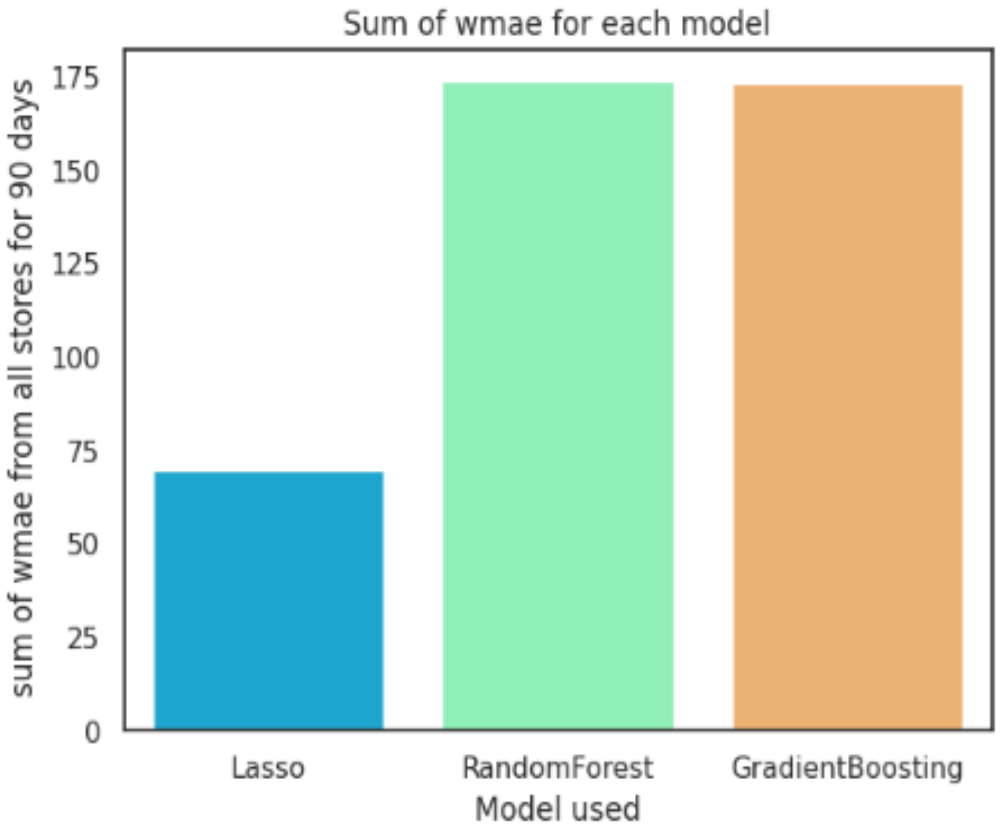
2.7.1 WMAE

The Weighted Mean Absolute Error (WMAE) is a valuable metric used extensively in demand forecasting scenarios, particularly within industries like retail and supply chain management. In demand forecasting, accurate predictions are crucial for optimizing inventory levels, production scheduling, and resource allocation. WMAE enhances traditional mean absolute error calculations by incorporating weights that reflect the relative importance of different forecasts

$$\text{WMAE} = \frac{1}{\sum w_i} \sum_{i=1}^{n} w_i |y_i - \hat{y}_i|$$

where

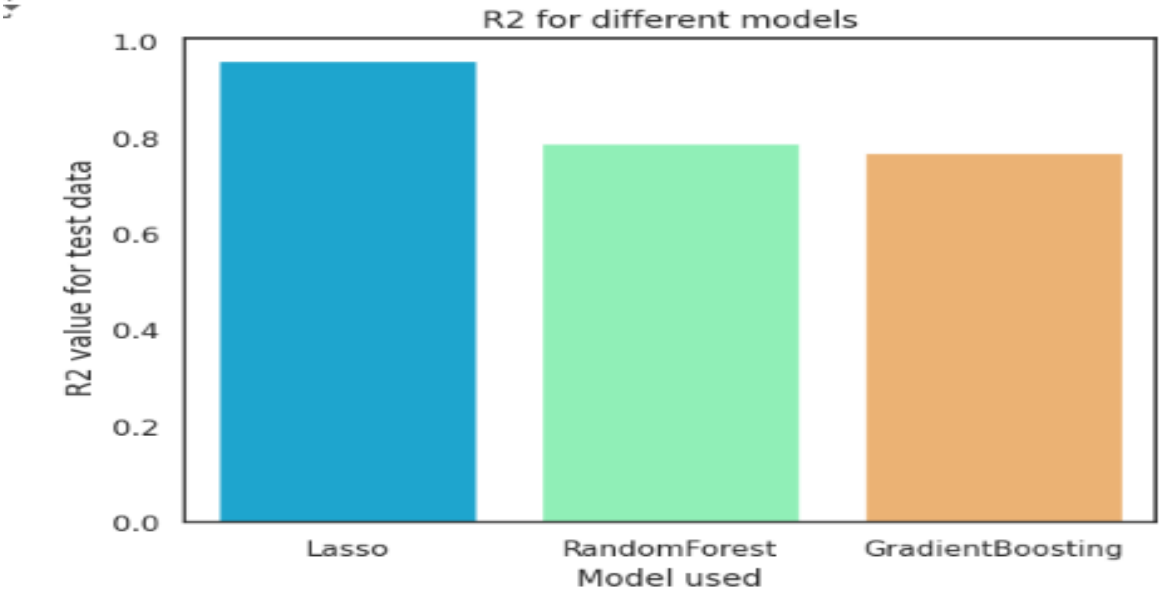
- n is the number of rows
- ŷ i is the predicted sales
- yi is the actual sales
- wi are weights. w = 5 if the week is a holiday week, 1 otherwise



2.7.2 R2

The R-squared (R^2) metric, also known as the coefficient of determination, is a widely used statistic in regression analysis, particularly within fields such as finance, economics, and the natural sciences. R^2 measures the proportion of the variance in the dependent variable that is predictable from the independent variables. It provides insights into the goodness of fit of a regression model, indicating how well the model's predictions match the actual data. High R^2 values signify a model that explains a substantial portion of the variance, thereby demonstrating strong predictive power and reliability. This metric is essential for model evaluation and selection, guiding decisions in areas like investment strategy, economic forecasting, and scientific research by highlighting the explanatory strength of the models used.

$$R2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})}, \qquad \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$$



Finally, after building multiple models to predict the weekly sales of 45 stores, we achieved an accuracy of approximately 94% over a period of three months (90 days). Among all the models tested, we selected the **Lasso Regressor** as our predictive model because it provided the highest prediction accuracy with the lowest residual error

3 Realization

3.1 Tools

Scikit-learn is a powerful and user-friendly machine learning library for Python, offering simple and efficient tools for data mining and data analysis. It is widely used for building and evaluating predictive models. In our project, we used scikit-learn for selecting and modeling the optimal algorithm to predict the weekly sales of 45 stores, ultimately choosing the Lasso Regressor for its superior accuracy and minimal residual error.

Pandas is a versatile and widely-used data manipulation and analysis library for Python, providing high-performance, easy-to-use data structures and data analysis tools. In our project, we used Pandas for data cleaning, transformation, and exploration, enabling us to efficiently prepare the dataset for building and evaluating predictive models to forecast weekly sales of 45 stores.

Plotly is an interactive graphing library for Python, known for its high-quality visualizations and ability to create complex, customizable plots with ease. In our project, we used Plotly to visualize data trends and model predictions, allowing us to gain deeper insights into the weekly sales patterns of 45 stores and effectively communicate our findings.

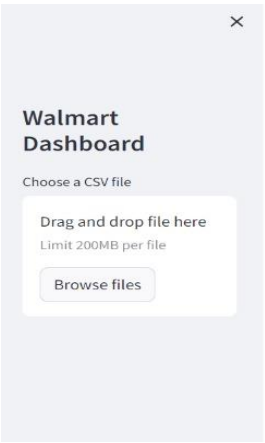
Streamlit is an open-source app framework for creating and sharing custom web applications for machine learning and data science projects. It enables fast and easy development of interactive dashboards with minimal coding. In our project, we used Streamlit to create our Walmart

dashboard, providing an intuitive interface to visualize and interact with the sales predictions and data trends for 45 stores.

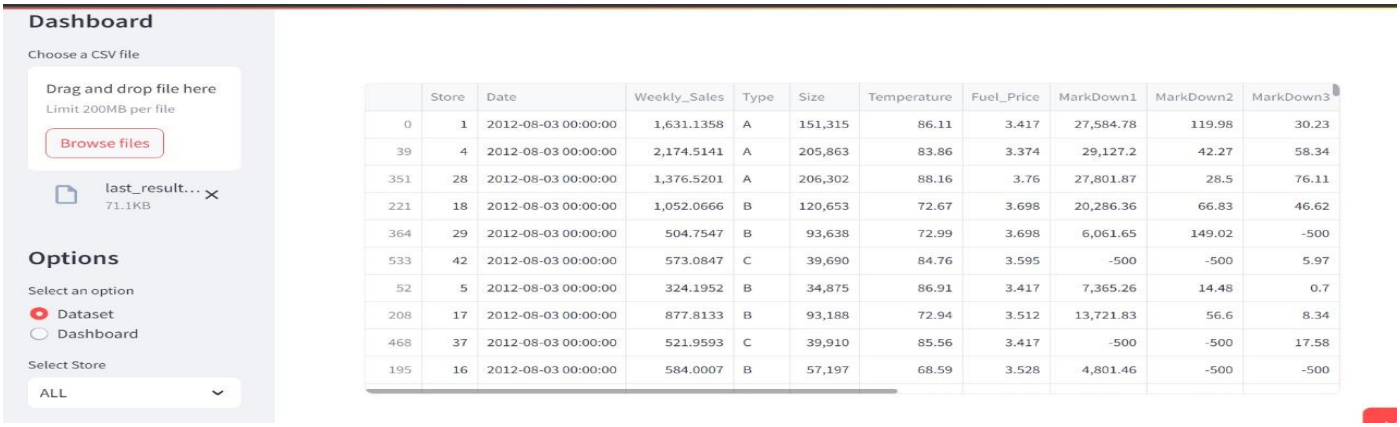
GitHub is a widely-used platform for version control and collaborative software development, enabling efficient code management and sharing. In our project, we used GitHub in conjunction with Streamlit to deploy our Walmart dashboard, ensuring seamless collaboration, version tracking, and easy updates for our interactive sales prediction tool.

3.2 Dashboard

3.2.1 choose the file that contains the prediction:

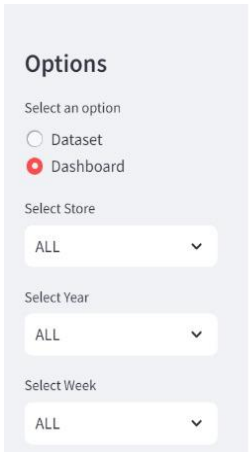


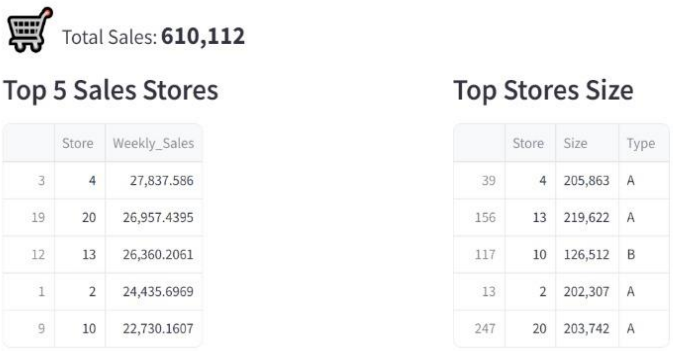
3.2.2 Visualize the data :

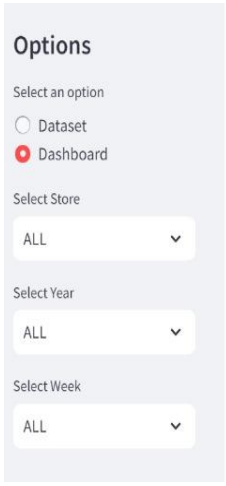


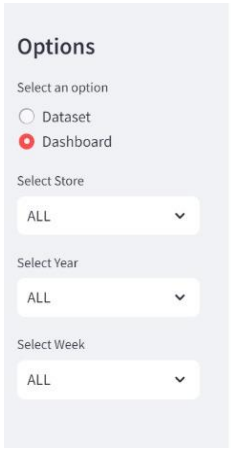
	Store	Date	Weekly_Sales	Type	Size	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3
0	1	2012-08-03 00:00:00	1,631.1358	A	151,315	86.11	3.417	27,584.78	119.98	30.23
39	4	2012-08-03 00:00:00	2,174.5141	A	205,863	83.86	3.374	29,127.2	42.27	58.34
351	28	2012-08-03 00:00:00	1,376.5201	A	206,302	88.16	3.76	27,801.87	28.5	76.11
221	18	2012-08-03 00:00:00	1,052.0666	B	120,653	72.67	3.698	20,286.36	66.83	46.62
364	29	2012-08-03 00:00:00	504.7547	B	93,638	72.99	3.698	6,061.65	149.02	-500
533	42	2012-08-03 00:00:00	573.0847	C	39,690	84.76	3.595	-500	-500	5.97
52	5	2012-08-03 00:00:00	324.1952	B	34,875	86.91	3.417	7,365.26	14.48	0.7
208	17	2012-08-03 00:00:00	877.8133	B	93,188	72.94	3.512	13,721.83	56.6	8.34
468	37	2012-08-03 00:00:00	521.9593	C	39,910	85.56	3.417	-500	-500	17.58
195	16	2012-08-03 00:00:00	584.0007	B	57,197	68.59	3.528	4,801.46	-500	-500

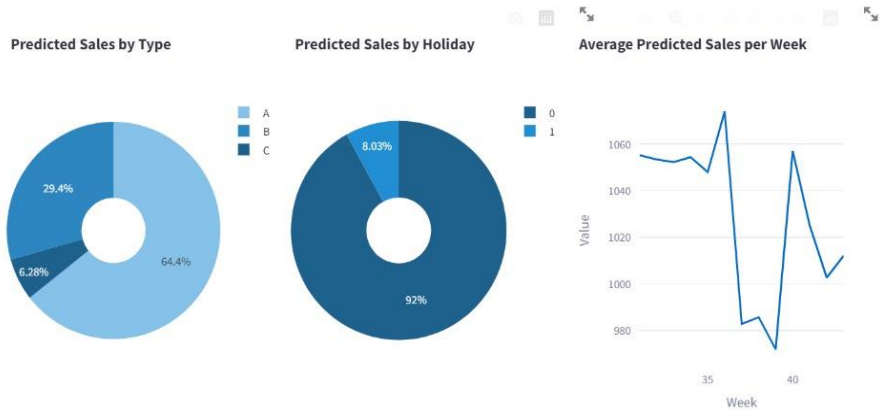
3.2.3 Visualize the graphs :











3.2.4 Select a store :