Capstone Project – 2

Walmart Store Sales Forecasting

Overview

Walmart is an American multinational retail corporation that operates a chain of hypermarkets, department stores, and grocery stores. As of Jul 2019, Walmart has 11,200 stores in 27 countries with revenues exceeding $500 billion. A challenge facing the retail industry such as Walmart's is to ensure the supply chain and warehouse space usage is optimized to ensure supply meets demand effectively, especially during spikes such as the holiday seasons.

This is where accurate sales forecasting enables companies to make informed business decisions. Companies can base their forecasts on past sales data, industry-wide comparisons, and economic trends. However, a forecasting challenge is a need to make decisions based on limited history. If Christmas comes but once a year, so does the chance to see how strategic decisions impacted the bottom line.

A. Problem Statement

A retail store that has multiple outlets across the country is facing issues in managing the inventory - to match the demand with respect to supply. You are a data scientist, who has to come up with useful insights using the data and make prediction models to forecast the sales for X number of months/years.

Dataset Information:

The walmart.csv contains 6435 rows and 8 columns.

Feature Name	Description
Store	Store number
Date	Week of Sales
Weekly Sales	Sales for the given store in that week

Holiday Flag	If it is a holiday week
Temperature	The temperature on the day of the sale
Fuel Price	Cost of fuel in the region
CPI	Consumer Price Index
Unemployment	Unemployment Rate

B. Project Objectives

1. Using the above data, come up with useful insights that can be used by each of the stores to improve in various areas.

2. Forecast the sales for each store for the next 12 weeks.

C. Data Description & Analysis with some useful insights

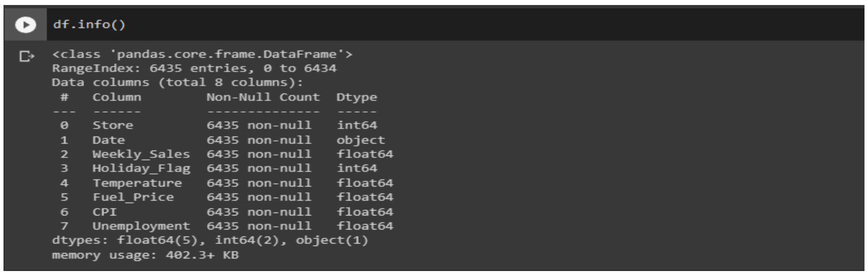
1. Data Exploration

The dataset is provided by Intellipaat itself, however, we can also download the same from Kaggle in CSV format by, clicking here.

This is the historical data that covers sales from 2010-02-05 to 2012-10-26, in the file "walmart.csv". within this file you will find the following fields:

- Store - the store number
- Date - the week of sales
- Weekly_Sales - sales for the given store
- Holiday_Flag - whether the week is a special holiday week 1 – Holiday week 0 – Non-holiday week
- Temperature - Temperature on the day of sale
- Fuel_Price - Cost of fuel in the region
- CPI – Prevailing consumer price index
- Unemployment - Prevailing unemployment rate

Observation: Information gathered about the dataset.



```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6435 entries, 0 to 6434
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Store         6435 non-null   int64
 1   Date          6435 non-null   object
 2   Weekly_Sales  6435 non-null   float64
 3   Holiday_Flag  6435 non-null   int64
 4   Temperature   6435 non-null   float64
 5   Fuel_Price    6435 non-null   float64
 6   CPI           6435 non-null   float64
 7   Unemployment  6435 non-null   float64
dtypes: float64(5), int64(2), object(1)
memory usage: 402.3+ KB
```

1. The dataset has 6435 entries, with 1 object and 7 numerical features.
2. Features have no missing values.
3. “Date” column is a string type and “Holiday_flag” is a Boolean type data.
4. Every column has several unique numbers of entries,

Holiday_Flag	2	CPI	2145
Store	45	Temperature	3528
Unemployment	349	Weekly_Sales	6435
Fuel_Price	892		

5. The dataset has no duplicate entries.
6. In the dataset, the “Weekly_Sales” column is already been resampled with weekly sales only for 45 different stores, each store having 143 weekly sales entries.
7. Weekly sales have not any negative or zero sales entries, as I have analyzed it with the pivot table and stored it in the variable named “store_dept_table”.

2. Evaluating and Fixing Outliers

2.1. Analyzing and fixing outliers

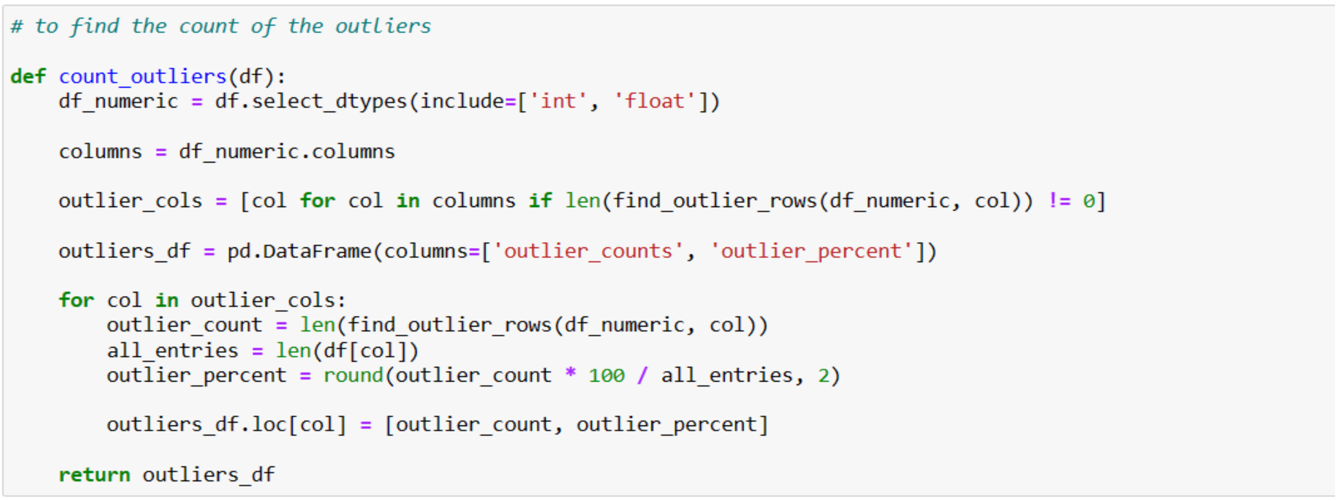
I have written two functions to find out the number of outliers and the rows with their index having outliers.

1. To find the rows having outliers:

```python
# to find the rows having outliers

def find_outlier_rows(df, col, level='both'):
    iqr = df[col].quantile(0.75) - df[col].quantile(0.25)

    lower_bound = df[col].quantile(0.25) - 1.5 * iqr
    upper_bound = df[col].quantile(0.75) + 1.5 * iqr

    if level == 'lower':
        return df[df[col] < lower_bound]
    elif level == 'upper':
        return df[df[col] > upper_bound]
    else:
        return df[(df[col] > upper_bound) | (df[col] < lower_bound)]
```

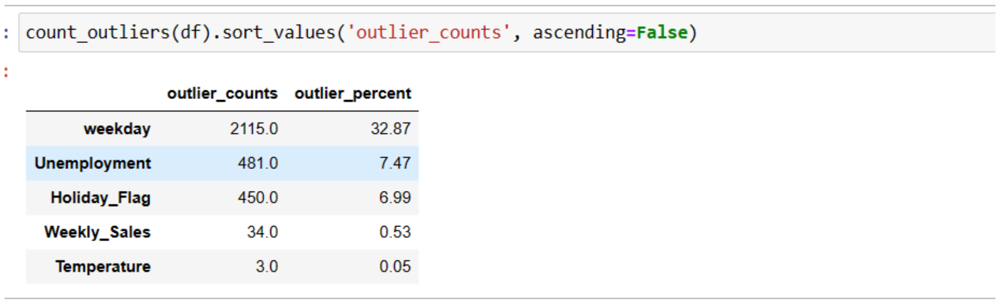
[Continues..]

2. To find the count of the outliers:



```
# to find the count of the outliers

def count_outliers(df):
    df_numeric = df.select_dtypes(include=['int', 'float'])

    columns = df_numeric.columns

    outlier_cols = [col for col in columns if len(find_outlier_rows(df_numeric, col)) != 0]

    outliers_df = pd.DataFrame(columns=['outlier_counts', 'outlier_percent'])

    for col in outlier_cols:
        outlier_count = len(find_outlier_rows(df_numeric, col))
        all_entries = len(df[col])
        outlier_percent = round(outlier_count * 100 / all_entries, 2)

        outliers_df.loc[col] = [outlier_count, outlier_percent]

    return outliers_df
```

3. Count the outliers in the columns of the dataset:



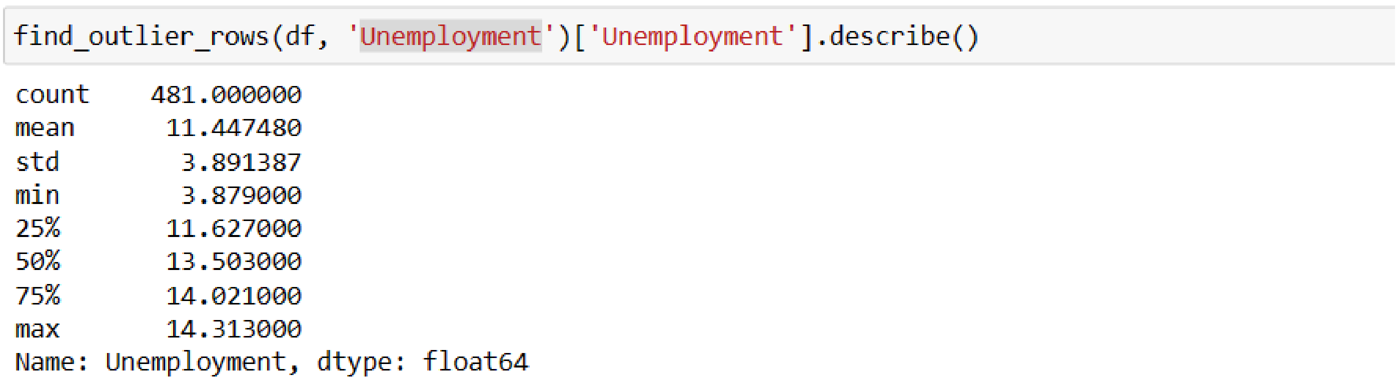
```
count_outliers(df).sort_values('outlier_counts', ascending=False)
```

	outlier_counts	outlier_percent
weekday	2115.0	32.87
Unemployment	481.0	7.47
Holiday_Flag	450.0	6.99
Weekly_Sales	34.0	0.53
Temperature	3.0	0.05

- As per the analysis, the obtained data frame shows that ‘weekday’, ‘weekly_sales’, ‘holiday_flag’, temperature, and unemployment columns all have outliers with unemployment having the largest outlier percentage, 7%.
- However, we can ignore the weekday as it is less relevant according to the problem statement and also, ‘weekly_sales’, as per the problem statement we will perform the time series analysis and forecast the weekly sales for the different 45 stores.

Let’s examine the outliers in each column except 'weekday' to decide how to handle them.

3.1. Unemployment column:



```
find_outlier_rows(df, 'Unemployment')['Unemployment'].describe()
```

```
count    481.000000
mean      11.447480
std        3.891387
min        3.879000
25%       11.627000
50%       13.503000
75%       14.021000
max       14.313000
Name: Unemployment, dtype: float64
```

- The minimum and maximum values of these outliers are 3.89% and 14.31% respectively.

- The majority, greater or equal to 75%, are more than or equal to 11.6% These values are obtainable in reality and will be left intact for the analysis. Thus, the median, which is robust to outliers, will be used to measure the center of the unemployment rate distribution.

3.2. Holiday_Flag column:

```
find_outlier_rows(df, 'Holiday_Flag')['Holiday_Flag'].describe()
```

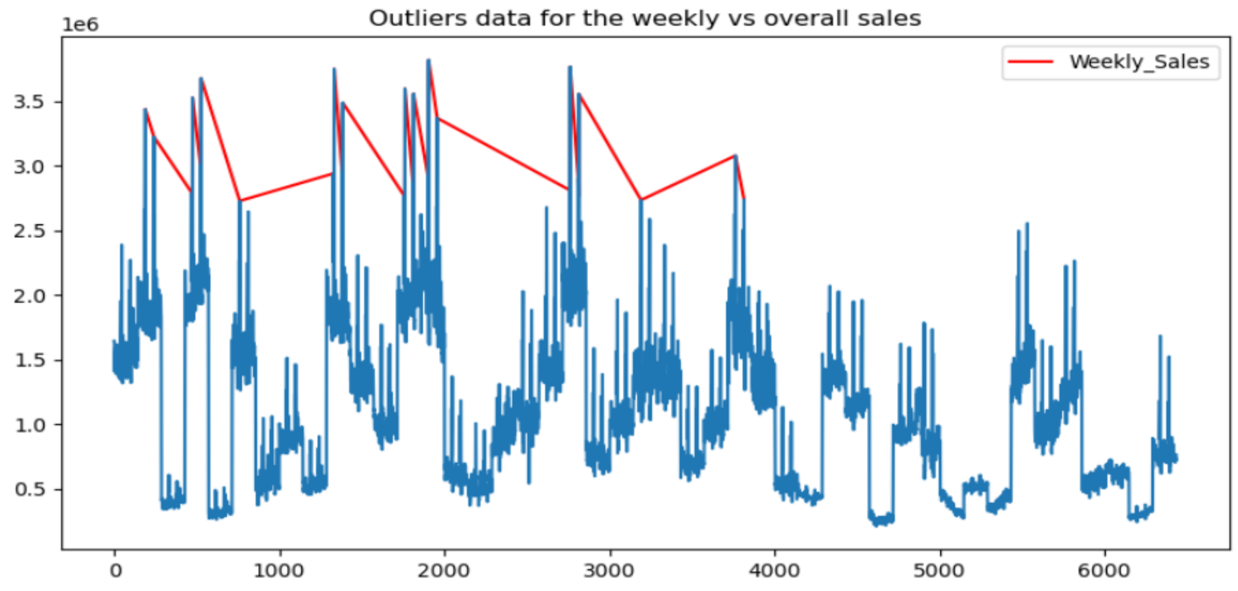
```
count    450.0
mean       1.0
std        0.0
min        1.0
25%        1.0
50%        1.0
75%        1.0
max        1.0
Name: Holiday_Flag, dtype: float64
```

- We can see that all special holiday weeks form outliers. This is a result of the fact that most of the weeks, 93%, are non-special holiday weeks. This will also be left intact.

3.3. Weekly_Sales column:

- We can note that all the weekly_sales outliers occur either in November or December for the years 2010 & 2011 of weekly sales and also, only one outlier was found in October for the years 2010 & 2011.
- Also, as per the plotted image, we can understand that we have seasonal occurrences in the data and greater sales in the months of November and December for the years 2010 & 2011.
- Plotted image with the sales obtained as outliers:

```
find_outlier_rows(df, 'Weekly_Sales')['Weekly_Sales'].plot(figsize=(10,5),
                                                            legend = True,
                                                            title = 'Outliers data for the weekly vs overall sales',
                                                            color='r')
df['Weekly_Sales'].plot()

plt.show() # to avoid the noise
```

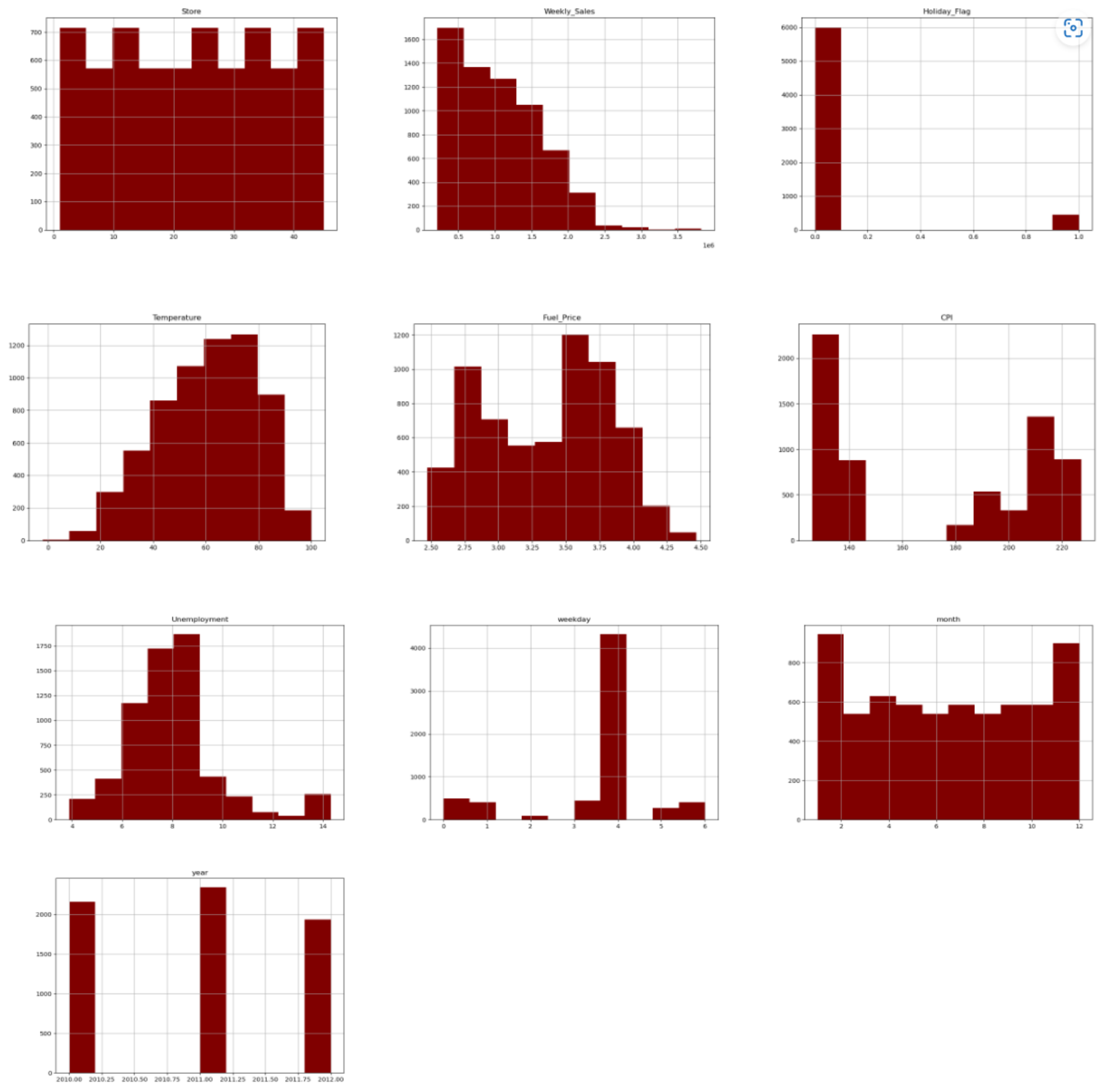


2.2. Distribution of data and its visualization

1. Distribution of overall data for all the columns

Analyzing Distribution

```
df.hist(figsize=(30,30), color='maroon')
plt.show()
```



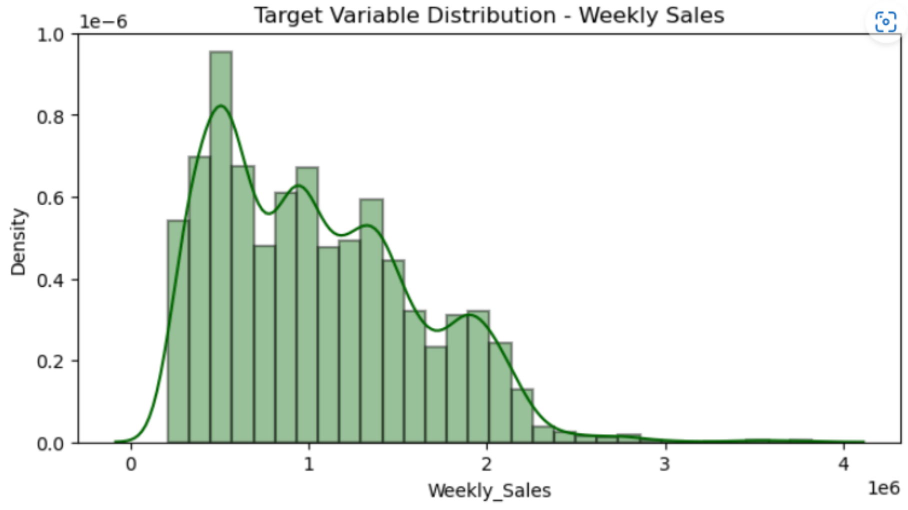
[Continues..]

From the above histograms, we can understand that:

- The number of transactions occurred almost evenly across various stores and years.
- The distribution of weekly_sales right-skewed. Only a few of the weekly sales are above 2 million USD.
- The distribution of temperature is approximately normal.
- The distribution of prices is bi-modal.
- CPI formed two clusters.
- The unemployment rate is near normally distributed.
- For Four consecutive months, November-February recorded the highest sales.

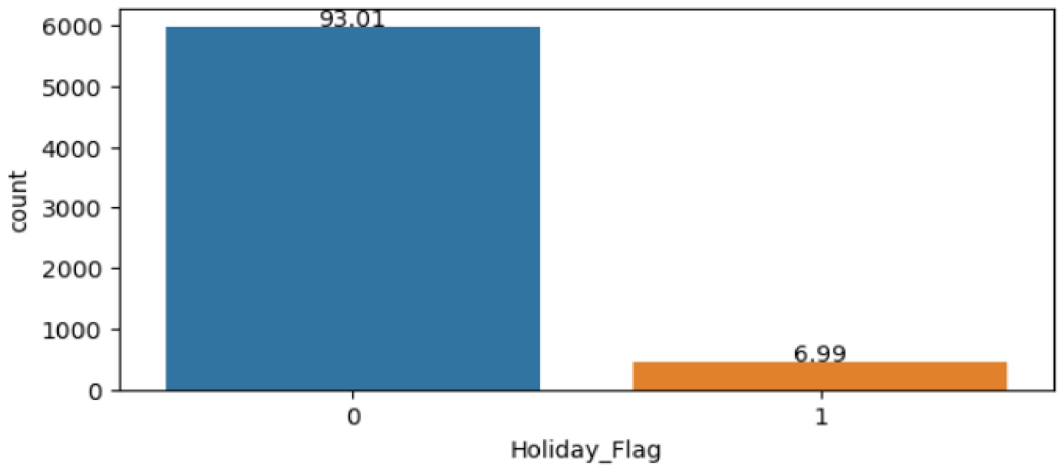
2. Overall weekly sales distribution for all the years:

```
#Let us first analyze the distribution of the target variable

plt.figure(figsize=[8,4])
sns.distplot(df[target], color='darkgreen',hist_kws=dict(edgecolor="black", linewidth=1.5), bins=30)
plt.title('Target Variable Distribution - Weekly Sales')
plt.show()
```



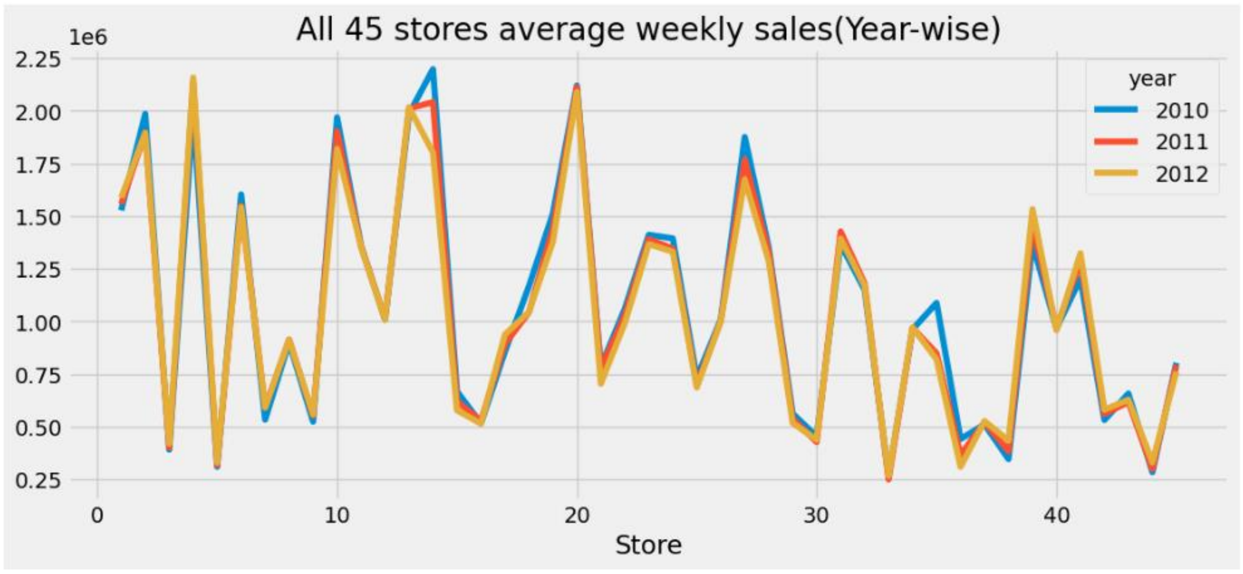
- The Target Variable seems to be normally distributed, averaging around 20 units.
- Sales over 2 million is minimal.

3. Overall sales distribution during holidays



- Most of the sales, which is 93.01% sales occurred during non-holiday dates and rest 6.99% during the holidays.

4. Average sales distribution of each store for years 2010, 2011 & 2012:



5. Stores and their data distribution:

- We have found that the dataset has a total of 45 stores.
- Each store has 143 weekly sales records dated from 2010-05-02 to 2012-10-26.

Further, the datasets seem that sales are already been resampled on a weekly basis. The total numbers of stores are 45 and each store has equal sales entries.

6. **Exploratory Data Analysis (EDA):** Perform a thorough analysis of the dataset to uncover patterns, trends, and relationships between variables. This step can involve visualizations, statistical summaries, etc., to gain insights into the data. The Following analysis I did on the dataset which are mentioned below(Refer to the Python Notebook File):
   - *Total features data distribution.*
   - *Holiday-based sales analysis*
   - *Average yearly sales analysis by each store*
   - *Monthly sales per store*
   - *Weekly sales per store (each year)*
   - *The highest and lowest average revenues over the years*
   - *Trend analysis over time*
   - *Feature effect analysis on sales (Fuel Price, CPI, Unemployment, Temperature)*

Overall findings and explorations:

- There are 45 stores and each stores have 143 entries.

- Although stores 10 and 35 have higher weekly sales values sometimes, on general average store 20 and store 4 are on the first and second rank. It means that some areas have higher seasonal sales.
- Also, it seems that the stores having larger sales are bigger and cover a populated area with a huge number of users. According to different factors, sales of the stores are changing.
- As expected, holiday average sales are higher than on normal dates.
- Christmas holiday introduces as the last days of the year. But people generally shop at the 51st week. So, when we look at the total sales of holidays, Thanksgiving has higher sales between them which were assigned by Walmart.
- The year 2010 has higher sales than 2011 and 2012. But November and December sales are not in the data for 2012. Even without the highest sales months, 2012 is not significantly less than 2010, so after adding the last two months, it can be first.
- It is obviously seen that weeks 51 and 47 have higher values and 50-48 weeks follow them. Interestingly, the 5th top sales belong to the 22nd week of the year. These results show that Christmas, Thanksgiving, and Black Friday are very important than other weeks for sales and 5th important time is the 22th week of the year and it is end of the May, when schools are closed. Most probably, people are preparing for a holiday at the end of May.
- January sales are significantly less than in other months. This is the result of November and December high sales. After two high sales months, people prefer to pay less in January.
- CPI, temperature, unemployment rate, and fuel price have no pattern on weekly sales.

[Continues..]

D. Data Pre-processing Steps and Inspiration

1. **Creating store-based sales dataset**: I am creating a store-wise sales data frame for time series analysis. The code is iterating over store numbers, selecting the weekly sales data for each store from the original Data Frame, and storing it in a new Data Frame (`**ts_df**`) with separate columns for each store's sales data. This allows for the analysis and modeling of store-wise sales patterns and forecasting.

Inspiration:

- I have already done the EDA and plotted the heatmap, which completely shows that the greatest number of features are not highly correlated with the weekly sales feature.
- So, I have decided to create this dataset for further forecasting and model building.
- In that case, the dataset is said to have completely irrelevant features and has less useful features & least linear relationship with other features to predict sales.
- I have decided to move forward and create time-based features like; day of the month, week or month, year, day, etc., as the sales are totally time-dependent and we can predict near-by sales value with it.

2. **Feature Engineering (Creation/Selection/Extraction):**

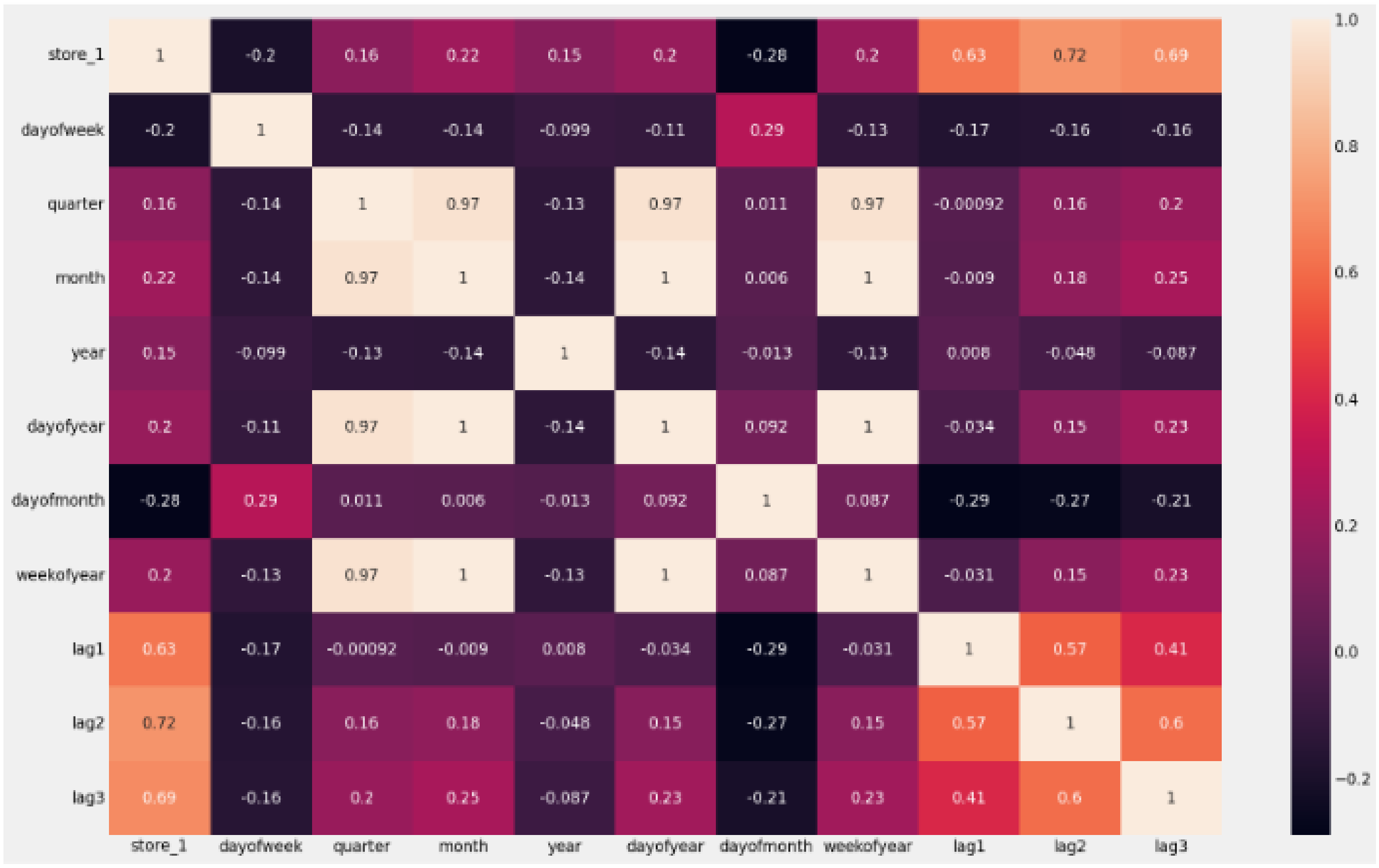
We have set the time as an index and created a function to create features on the basis of their sales date and the function operates on a copy of the input Data Frame to avoid modifying the original data.

Here is an explanation of the created features:

- **dayofweek**: It represents the day of the week (0 for Monday, 1 for Tuesday, ..., and 6 for Sunday) corresponding to each index in the time series.
- **quarter**: It indicates the quarter of the year (1, 2, 3, or 4) corresponding to each index in the time series.
- **month**: It represents the month (1 to 12) corresponding to each index in the time series.
- **year**: It indicates the year corresponding to each index in the time series.
- **dayofyear**: It represents the day of the year (1 to 365 or 366 for leap years) corresponding to each index in the time series.
- **dayofmonth**: It indicates the day of the month (1 to 31) corresponding to each index in the time series.
- **weekofyear**: It represents the week of the year (ISO week numbering, 1 to 53) corresponding to each index in the time series.
- **Creating the Lag feature:** Lag features are crucial for forecasting future values in time series analysis. By using lagged values as input features, we can train models to learn the relationship between past observations and future outcomes, enabling us to make predictions for upcoming time points and improve the predictive power of models by including relevant historical information.

By creating these features, the function provides additional temporal information that can be useful for time series analysis and forecasting. These features capture various time-related patterns and trends related to specific days, months, quarters, or years. This can enable predictive models to consider these patterns and make more accurate forecasts or predictions.

3. **Correlation matrix & heatmap for a new dataset:** Here, I have plotted the heatmap to visualize the correlation with the sales data and sales data is marked as 'store_1'**.**



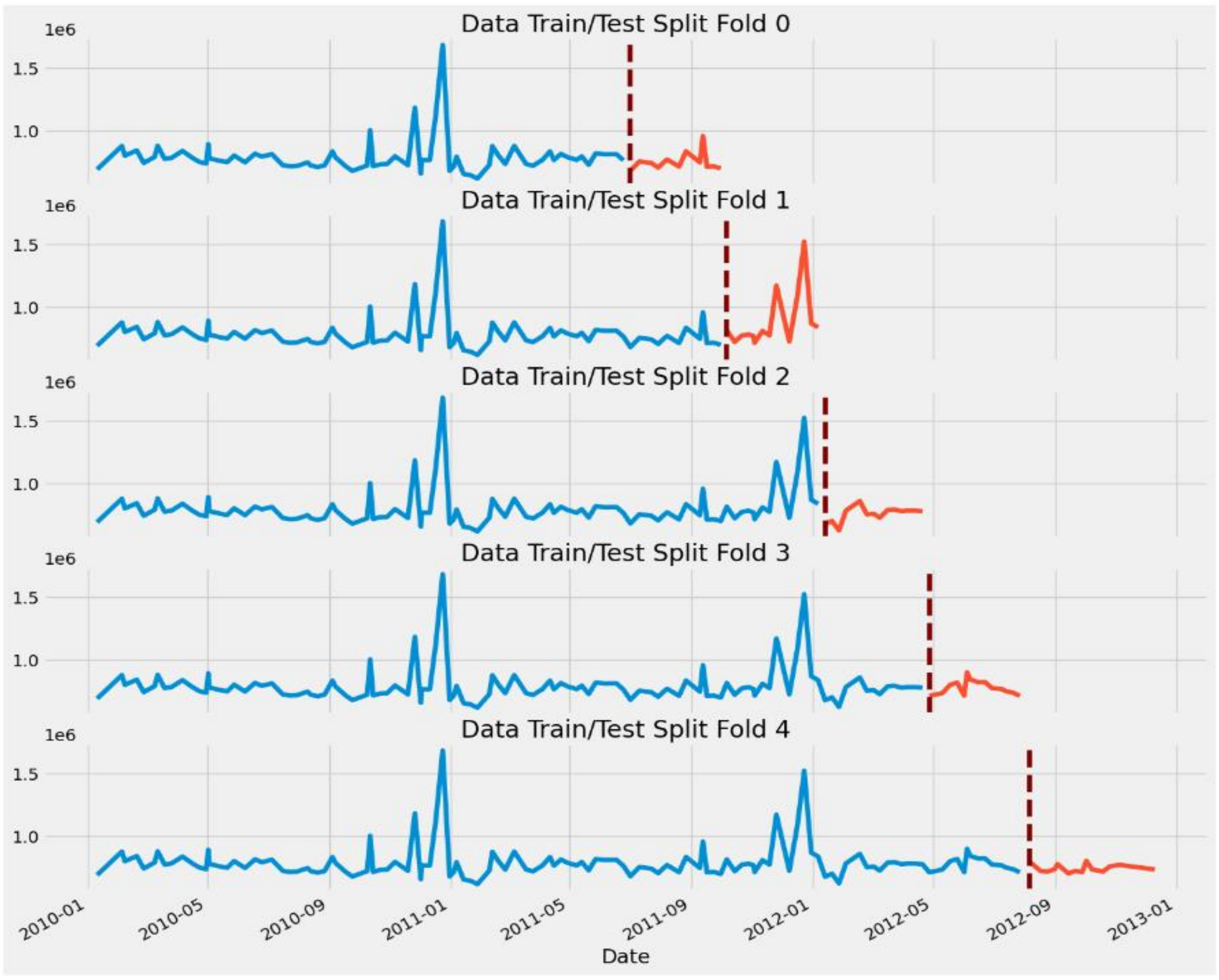
	store_1	dayofweek	quarter	month	year	dayofyear	dayofmonth	weekofyear	lag1	lag2	lag3
store_1	1	-0.2	0.16	0.22	0.15	0.2	-0.28	0.2	0.63	0.72	0.69
dayofweek	-0.2	1	-0.14	-0.14	-0.099	-0.11	0.29	-0.13	-0.17	-0.16	-0.16
quarter	0.16	-0.14	1	0.97	-0.13	0.97	0.011	0.97	-0.00092	0.16	0.2
month	0.22	-0.14	0.97	1	-0.14	1	0.006	1	-0.009	0.18	0.25
year	0.15	-0.099	-0.13	-0.14	1	-0.14	-0.013	-0.13	0.008	-0.048	-0.087
dayofyear	0.2	-0.11	0.97	1	-0.14	1	0.092	1	-0.034	0.15	0.23
dayofmonth	-0.28	0.29	0.011	0.006	-0.013	0.092	1	0.087	-0.29	-0.27	-0.21
weekofyear	0.2	-0.13	0.97	1	-0.13	1	0.087	1	-0.031	0.15	0.23
lag1	0.63	-0.17	-0.00092	-0.009	0.008	-0.034	-0.29	-0.031	1	0.57	0.41
lag2	0.72	-0.16	0.16	0.18	-0.048	0.15	-0.27	0.15	0.57	1	0.6
lag3	0.69	-0.16	0.2	0.25	-0.087	0.23	-0.21	0.23	0.41	0.6	1

Sales data shows a positive & negative correlation with the **'store_1'** sales data.

4. **Data cross-validation on the current data:** Here, I am using TimeSeriesSplit to check for data leakage and intersection in time series data. Data leakage occurs when information from the future is used to make predictions in the past, leading to overly optimistic performance estimates. Intersection refers to the overlapping of data between training and testing sets, which can also introduce data leakage.

   How it helps, TimeSeriesSplit helps to avoid data leakage and intersection by providing a way to split the time series data in a manner that preserves the temporal order. It ensures that the training data comes before the test data, simulating the real-world scenario of training on historical data and evaluating future data. So, we can say that no more manipulation required to our dataset.

   Note: I am only proceeding with 'store_1' sales data.



Here, In the above plot, I have checked and cross-validated the data after preprocessing, and found that we have no leakage, however, already we have less data for the sales of each store so no chance any intersection or leakage in it.

[Continues..]

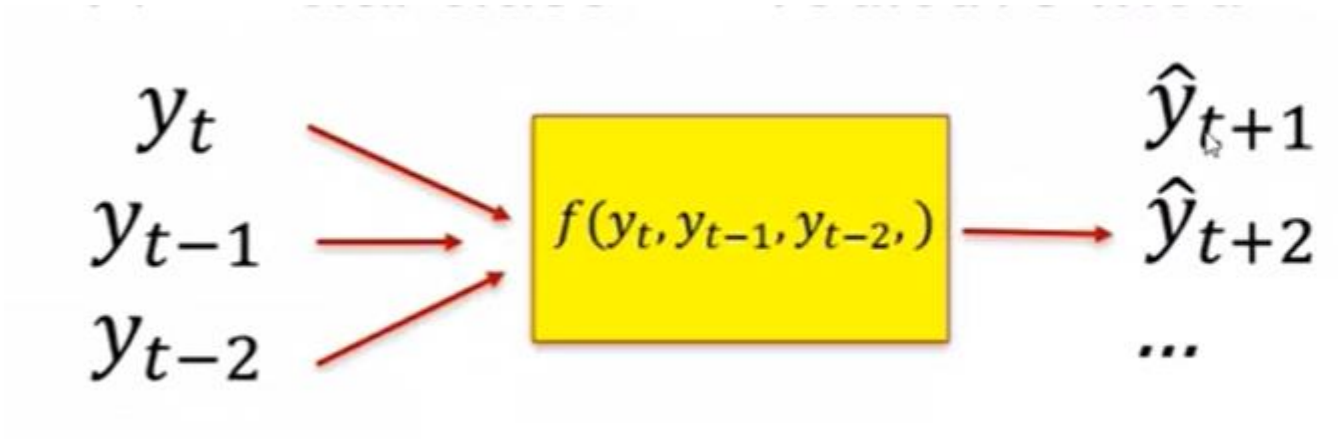
E. Choosing the Algorithm for the Project

Overview of Time series

Time-series forecasting is the process of using past data to predict future events. Time-series forecasting models are statistical models used to make predictions about future values based on historical data points arranged in chronological order. These models analyze trends and patterns in the data and extrapolate them to make predictions about future values. These models are commonly used in business and finance to predict sales or stock prices, and in science to predict weather patterns. Time-series forecasting models are a special class of predictive modeling that is used to forecast future events.

How are time series forecasting models different from other predictive modeling techniques? The answer is,

- Time-series models rely on historical data that is arranged in chronological order, whereas other models may use cross-sectional data or other non-time-related variables. The response variable to be forecasted is continuous in nature while input data are past values of the variable of interest (response variable). The following picture represents the same. The variable to be predicted and the input variable are one and the same. Yt, Yt−1, & Yt−2 are values of the past, and Yt+1, & Yt+2 are values in the future that are to be predicted.

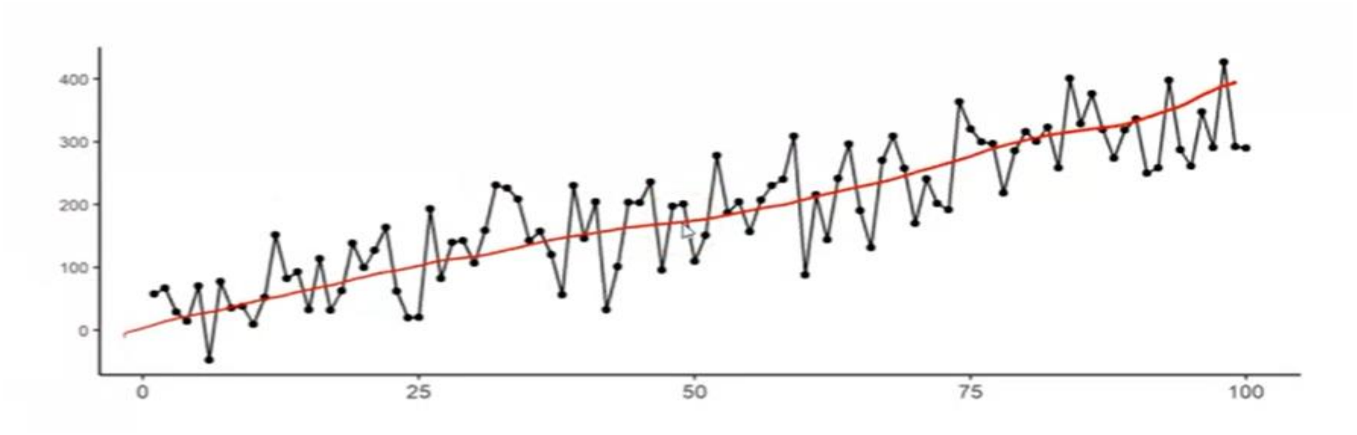


- Time-series models account for the temporal dependencies between data points, whereas other models assume that each data point is independent of the others.
- Time-series models often use specialized algorithms and techniques, such as ARIMA or exponential smoothing, which are designed specifically for time-series data.

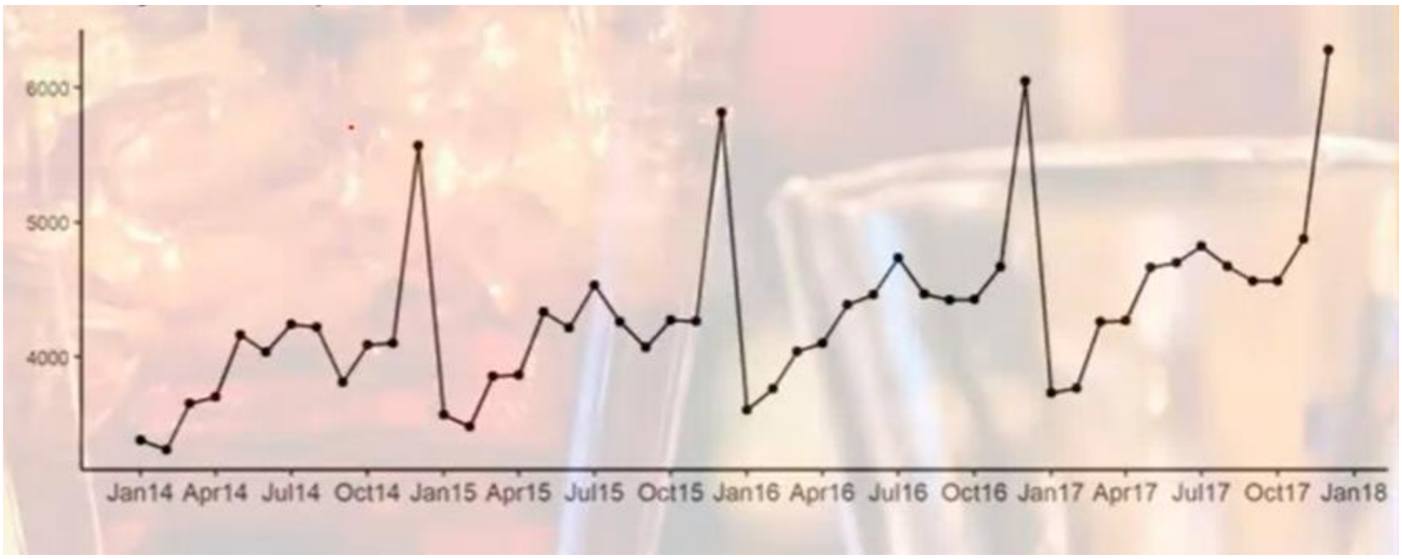
There are several characteristics of time-series data that need to be taken into account when modeling time-series data.

Time dependence: Time-series data is collected over time, with each observation representing a data point at a specific point in time. As a result, time-series data exhibits temporal dependence, meaning that each data point is influenced by the previous ones.

Trend: The most important one is the trend of the data. Time-series data often exhibit a trend, which is a gradual increase or decrease in the value of the data over time. The trend needs to be accounted for in the time-series forecasting model. The picture below shows an upward trend in the time-series data.



Seasonality: Another important characteristic of time-series data is the seasonal effect. Seasonal effects are patterns that occur regularly (yearly, quarterly, monthly, etc) in the data due to seasonal factors such as holidays or changes in weather. The seasonal effect needs to be accounted for in the forecasting model. The picture below shows the seasonality in the data:



Noise: The final characteristic of time-series data is the random error. Random error is the noise that is present in the data and affects the accuracy of the forecast. The random error needs to be accounted for in the forecasting model.

a. Autoregressive moving average (ARMA) model

The autoregressive moving average (ARMA) model is a combination of the autoregressive and moving average models. The ARMA model is defined as a regression model in which the dependent/response variable is a linear function of past values of both the dependent/response variable and the error term. The order of an ARMA model is represented by ‘p’ for the autoregressive part and ‘q’ for the moving average part. For example, if p=0 and q=0, then it means that we are predicting time-step (t) based on time-step (t) only. If p=n and q=m, then we are predicting time-step (t) based on n past time-steps of the dependent/response variable and m past time-steps of the error term.

The general form of an ARMA model can be represented as:

$$Yt=c+\phi pYt-p+\theta q\varepsilon t-q+\varepsilon t$$

b. Autoregressive integrated moving average (ARIMA) model

The autoregressive integrated moving average (ARIMA) model is a generalization of the ARMA model. The ARIMA model is defined as a regression model in which the dependent/response variable is a linear function of past values of both the dependent/response variable and the error term, where the error term has been differentiated ‘d’ times. The order of an ARIMA model is represented by ‘p’ for the autoregressive part, ‘q’ for the moving average part, and ‘d’ for the differencing part. For example, if p=0, q=0, and d=0, then it means that we are predicting time-step (t) based on time-step (t) only. If p=n, q=m, and d=k, then we are predicting time-step (t) based on n past time-steps of the dependent/response variable, m past time-steps of the error term, and k past time-steps of the differenced error term.

The general form of an ARIMA model can be represented as:

$$Yt=c+\phi pYt-p+\theta q\varepsilon t-q+\delta dYt$$

c. SARIMAX Model (Seasonal Autoregressive Integrated Moving Average with Exogenous Variables)

The SARIMAX model is an extension of SARIMA that allows for the inclusion of exogenous variables in the model. In addition to the seasonal and non-seasonal components, SARIMAX models can incorporate external factors that may influence the time series being forecasted. These exogenous variables can provide additional information and improve the accuracy of the forecasts.

Both SARIMA and SARIMAX models can be estimated and fitted to the data using various methods, such as maximum likelihood estimation. Once the model is fitted, it can be used to make forecasts for future time periods.

It's important to note that selecting the appropriate orders (p, d, q, P, D, Q, s) and identifying significant exogenous variables require careful analysis and model diagnostics. It often involves analyzing autocorrelation and partial autocorrelation plots, examining residual patterns, and conducting statistical tests.

The interpretation of autoregressive integrated moving average (ARIMA) models is similar to that of autoregressive moving average (ARMA) models. The main difference between ARIMA and ARMA models is that ARIMA models can be used to model time-series data that is non-stationary, whereas ARMA models can only be used to model time-series data that is stationary.

Further, as per the problem statement, an appropriate forecasting model for predicting sales for the next 12 weeks. Time series models, such as ARMA (Auto Regressive Moving Average), ARIMA (Auto Regressive Integrated Moving Average), or SARIMA (Seasonal ARIMA), are commonly used for such tasks. These models take into account the temporal nature of the data and can capture seasonality and trends.

However, we also created time-based features to predict the sales data and I will also test the data with some regressor models, which are mentioned below;

- XGB Regressor
- Boosted Tree Regression
- K-Nearest Neighbour Regression
- Random Forest Regression
- Decision Tree Regression
- Elastic Net Regression
- Polynomial Regression
- Lasso Regression
- Ridge Regression
- Support Vector Regression
- Spline Regression
- Linear Regression
- Neural Network Regression

F. Motivation and Reasons for Choosing the Algorithm

I have decided to build my final model with a stats-based ARMA, ARIMA or SARIMAX model instead of the regressor model, however, I have made some assumptions and for experiment purposes, I am also fitting the regression model with time-based features.

SARIMAX (Seasonal Autoregressive Integrated Moving Average) models are often preferred for time series analysis compared to regression models for several reasons:

1. **Handling Time Dependencies:** The SARIMAX model is an extension of SARIMA that allows for the inclusion of exogenous variables in the model. In addition to the seasonal and non-seasonal components, SARIMAX models can incorporate external factors that may influence the time series being forecasted. These exogenous variables can provide additional information and improve the accuracy of the forecasts.
2. **Nonlinear Relationships:** Time series data often exhibit nonlinear patterns and relationships over time. ARIMA models can capture such nonlinearities through the autoregressive and moving average terms, allowing for more flexible modelling compared to linear regression models.
3. **Trend and Seasonality:** Time series data commonly exhibit trend and seasonality components. ARIMA models can incorporate differencing to remove trend and seasonality from the data, making it easier to model the stationary residuals. Regression models typically assume a linear relationship between the predictors and the target variable, which may not adequately capture complex time-dependent patterns.
4. **Forecasting:** ARIMA models are well-suited for time series forecasting tasks. They can project future values based on the autoregressive, moving average & exogenous variables components, allowing for accurate

predictions. Regression models, on the other hand, may not account for the temporal dependencies and might not provide reliable forecasts.

5. **Model Interpretability:** SARIMAX models offer interpretability as they explicitly capture the autoregressive and moving average components. The coefficients in SARIMA models represent the strength and direction of the relationships along with the time and seasonal components, providing insights into the underlying dynamics of the time series. Regression models, while interpretable, may not capture the temporal aspects of the data as effectively.

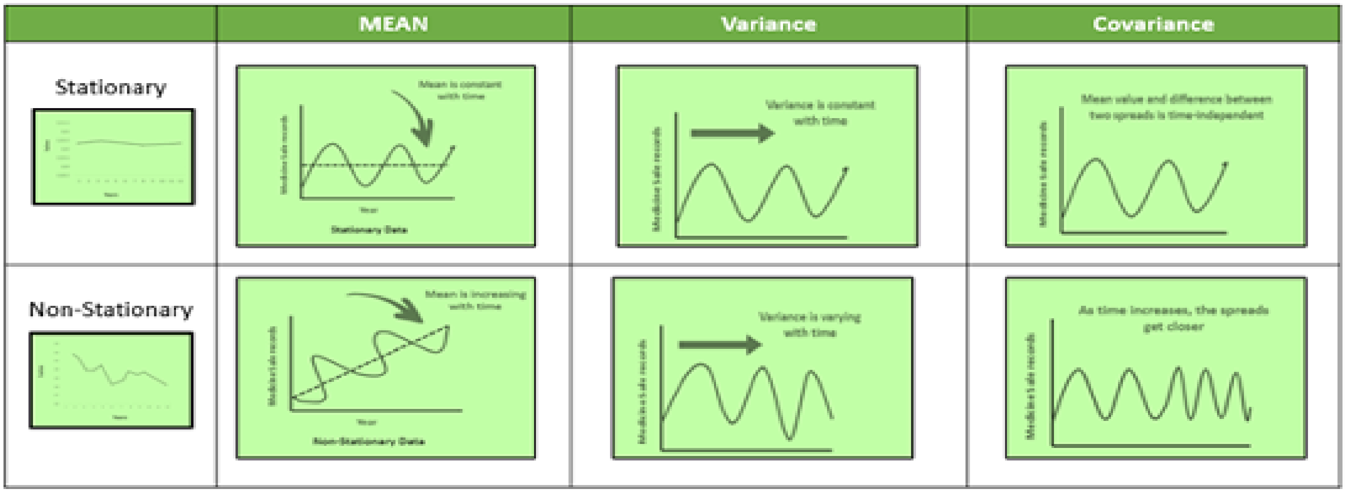
Limitations, Testing & Order:

Before fitting an ARIMA or SARIMA or SARIMAX model, whenever we come to the time series data it is required to follow the traditional way, and TS data have certain limitations & only two types of data types it has, which are, *stationary* and *non-stationary*.

Stationary: A dataset should follow the below thumb rules without having Trend, Seasonality, Cyclical, and Irregularity components of the time series.

- The mean value of them should be completely constant in the data during the analysis.
- The variance should be constant with respect to the time-frame
- Covariance measures the relationship between two variables.

Non- Stationary: If either the mean-variance or covariance is changing with respect to time, the dataset is called non-stationary.


MEAN
Variance
Covariance
Stationary
Mean is constant with time
Stationary Data
Variance is constant with time
Mean value and difference between two spreads is time-independent
Non-Stationary
Mean is increasing with time
Non-Stationary Data
Variance is varying with time
As time increases, the spreads get closer
Medicine Sale records
Year

Designed by Author (Shanthababu)

Hypothesis testing to validate data stationarity or non-stationarity:

a. **Augmented Dickey-Fuller (ADF) Test or Unit Root Test**

The ADF test is the most popular statistical test. It is done with the following assumptions:

- Null Hypothesis (H0): Series is non-stationary
- Alternate Hypothesis (HA): Series is stationary
    - p-value >0.05 Fail to reject (H0)
    - p-value <= 0.05 Accept (H1)

However, we have tested the data and found that our TS data is stationary by using the function '**adf_test**' to perform ADF.

b. **Seasonal Decomposition:** Decomposing the time series into its components (trend, seasonality, and residuals) using methods like additive or multiplicative decomposition can help identify and visualize the seasonality component. If the seasonality component exhibits clear and repeating patterns, it indicates the presence of seasonality.

Order

When dealing with TS data, there are multiple model options are available. In which the Autoregressive–Moving-Average (ARMA) or ARIMA models with [p, d, and q] are common. We have several parameters to find first which are as follows:

P➔ Autoregressive lags

q➔ Moving average lags

d➔ Difference in the order

and **P, D, Q, s are for the seasonal coponents.**

We have two different methods:

1. **Manual methods:**
   - Auto-Correlation Function (ACF)
   - Partial Auto-Correlation Function (PACF)
2. **Using the Auto Arima:**

   Related function: **pmdarima.auto_arima – Returns an optional order for the ARIMA model**

It's important to note that the choice between ARMA or ARIMA and regression models depends on the specific characteristics and requirements of the time series data and the modelling task at hand. In some cases, regression models may be appropriate, especially when the time dependencies are weak or when other variables, beyond time, have a significant impact on the target variable. However, for many time series analysis scenarios, where the focus is on temporal patterns and forecasting, ARIMA models are often more suitable and effective.

G. Assumptions

As per the overall feature analysis we have performed, from this it is clear that other features do not have a strong linear relationship with the weekly sales feature.

So, the assumption made here are as follow;

- To create the time-based feature, i.e., dayofweek, quarter, month, year etc.

- Check the correlation with the weekly sales feature.
- Try to fit the regressor model and get RMSE scores on the test & complete data.
- Also, saving the model which is the best of all.
- So, I have fitted different regressor models, however, XGB Regressor and Gradient Boosting Regressor performed better than all others.

H. Model Evaluation and Techniques

Model evaluation helps us to check the error and how our models performing. So, in the earlier order to avoid the e

There are several metrics commonly used to evaluate the performance of a stats-based ARIMA and different regressor models for time series forecasting. Some of the key metrics include:

1. **Mean Squared Error (MSE)**: MSE measures the average squared difference between the predicted and actual values. It provides an overall measure of the model's accuracy, with lower values indicating better performance.
2. **Root Mean Squared Error (RMSE):** RMSE is the square root of MSE and represents the average magnitude of the prediction errors. It is more interpretable than MSE as it is in the same unit as the target variable.
3. **Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC)**: AIC and BIC are model selection criteria that balance the goodness of fit with the complexity of the model. Lower values indicate better fit and parsimony.

I. Inferences from the Same

- The Dataset was quite small with just 6435 samples & after preprocessing 7.5% of the data samples were dropped.
- Visualizing the distribution of data & their relationships, helped us to get some insights on the feature-set.
- Also, performing EDA and visualizing the different sales help us to optimize the overall sales performance and segment our customer to create efficient business.
- The features had high multicollinearity, hence in the Feature Creation step, we created the appropriate time-based features with the help of given time columns.
- Testing multiple algorithms with default hyperparameters gave us some understanding of various models' performance on this specific dataset.
- It is safe to use multiple regression algorithms that performed better than other algorithms, as their scores were quite comparable & also, they're more generalizable.
- Using the stats model has an edge over the regression model if we want to forecast future prices.

J. Future Possibilities of the Project

1. **Demand forecasting:** By analyzing historical sales data, store owners can gain insights into the demand patterns and seasonality of their products. This information can be used to forecast future sales and optimize inventory management. Advanced forecasting techniques such as time series analysis, ARIMA models, or machine learning algorithms can be applied to predict sales for each store in the next 12 weeks.

2. **Holiday promotion planning:** The "Holiday_Flag" feature in the dataset indicates whether a given week is a holiday week. Store owners can analyze the impact of holidays on sales and plan targeted promotions or marketing campaigns to capitalize on increased consumer spending during those periods. This can help optimize promotional strategies and maximize sales during the holiday seasons.

3. **Store performance comparison:** By comparing sales data across different stores, store owners can identify top-performing stores and analyze the factors contributing to their success. This analysis can uncover best practices, strategies, or operational efficiencies that can be replicated in underperforming stores to improve their sales performance.

4. **Operational efficiency improvements:** Analyzing sales data can reveal operational inefficiencies or bottlenecks in the supply chain. By identifying patterns such as low sales during specific periods or stockouts, store owners can optimize inventory management, improve product availability, and streamline their operations.

[Continues..]

K. Conclusion

Based on the given dataset, here is a possible conclusion for the project:

- Store Performance Analysis: Analyze the sales performance of each store by examining the trends and patterns in weekly sales. Identify the top-performing stores and areas for improvement.
- Holiday Sales Impact: Investigate the impact of holidays on sales. Determine which holidays drive higher sales and strategize promotional activities accordingly.
- Fitting time based regression model with time based features also worked and the model predicting the sales with 0.008% Error.
- Using time series analysis techniques, such as ARIMA (AutoRegressive Integrated Moving Average), to forecast sales for each store for the next 12 weeks also bit promising.

- Continuously monitor the forecasted sales and compare them with the actual sales data. Adjust the models as needed based on the observed performance and any changing patterns or trends in the data.

Overall, the project aims to provide actionable insights for each store to improve their performance and efficiency. Additionally, by forecasting sales for the next 12 weeks, the project can assist in planning inventory, staffing, and marketing strategies to meet customer demand effectively.

L. References

a. https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-to-time-series-analysis/
b. https://www.kaggle.com/code/yasserh/bitcoin-prices-forecast-top-ml-model
c. https://analyticsindiamag.com/a-guide-to-different-evaluation-metrics-for-time-series-forecasting-models/
d. https://github.com/abhinav-bhardwaj/Walmart-Sales-Time-Series-Forecasting-Using-Machine-Learning
e. https://towardsdatascience.com/model-evaluation-in-time-series-forecasting-ae41993e267c