# Capstone Project – Walmart Sales Analysis

## Table of Contents

## Problem Statement 1:

The retail store chain is facing challenges in managing its inventory to match supply with demand. The aim is to analyze the weekly sales data from multiple outlets and provide insights to improve inventory management and sales forecasting.

**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 | Temperature on the day of the sale |
| Fuel_Price | Cost of fuel in the region |
| CPI | Consumer Price Index |
| Unemployment | Unemployment Rate |

1. You are provided with the weekly sales data for their various outlets. Use statistical analysis, EDA, outlier analysis, and handle the missing values to come up with various insights that can give them a clear perspective on the following:
    a. If the weekly sales are affected by the unemployment rate, if yes - which stores are suffering the most?
    b. If the weekly sales show a seasonal trend, when and what could be the reason?
    c. Does temperature affect the weekly sales in any manner?
    d. How is the Consumer Price Index affecting the weekly sales of various stores?
    e. Top performing stores according to the historical data.
    f. The worst performing store, and how significant is the difference between the highest and lowest performing stores?

2. Use predictive modelling techniques to forecast the sales for each store for the next 12 weeks.

## Project Objective

- Analyze weekly sales data to uncover patterns and trends.
- Assess the impact of external factors (unemployment rate, temperature, Consumer Price Index) on sales.
- Identify top and worst-performing stores.
- Forecast future sales for better inventory management.

## Data Description

Dataset Download Link – [Walmart DataSet - Google Drive]

The dataset contains the following columns:

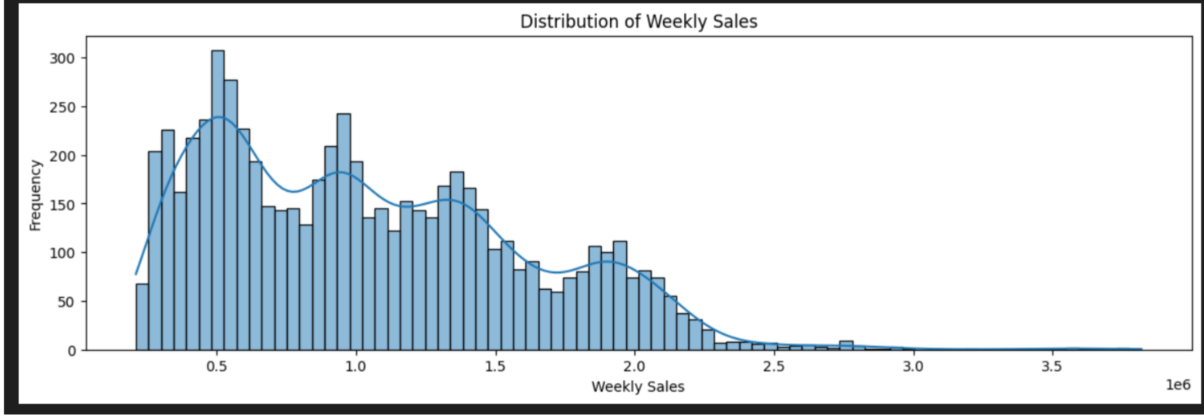
- Store: Store number.
- Date: Date of the sales entry.
- Weekly_Sales: Weekly sales amount.
- Holiday_Flag: Indicates if the week includes a special holiday.
- Temperature: Temperature in the region.
- Fuel_Price: Cost of fuel in the region.
- CPI: Consumer Price Index.
- Unemployment: The unemployment rate in the region.

## Various insights from the data.

### Plot the distribution of weekly sales

```
# Plot the distribution of weekly sales
plt.figure(figsize=(14, 4))
sns.histplot(data['Weekly_Sales'], bins=80, kde=True)
plt.title('Distribution of Weekly Sales')
plt.xlabel('Weekly Sales')
plt.ylabel('Frequency')
plt.show()
```

✓ 0.6s

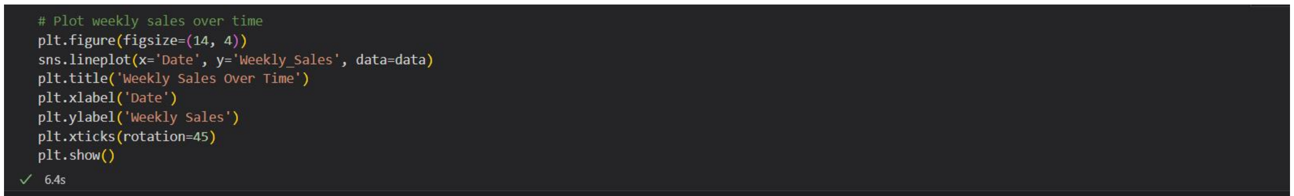


### Inferences

- **Central Tendency**: The peak of the histogram is around the 0.5 million mark. This suggests that the average weekly sales are around 0.5 million.
- **Spread:** The spread of the distribution shows a wide range of sales values from around 0.1 million to over 3.5 million. This indicates a high level of variability in weekly sales across different stores and weeks. There are some stores/weeks that perform significantly better or worse than others.
- **Skewness:** The sales distribution is right-skewed, meaning that there are weeks with exceptionally high sales, but these are less common.

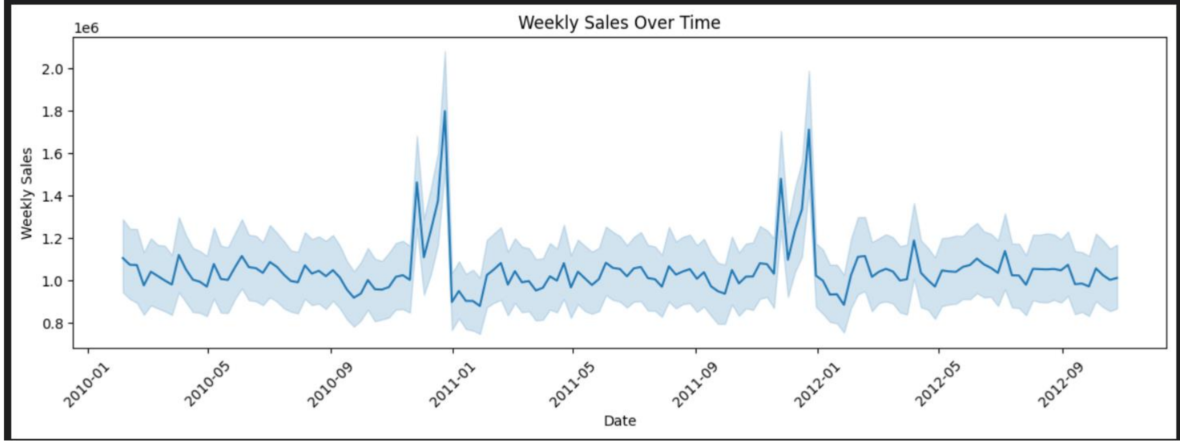
- **Kurtosis:** This higher peak indicates that the majority of the sales data points are close to the mean, with fewer extreme values than expected in a normal distribution.
- **Outliers:** The long tail on the right side of the histogram indicates the presence of outliers—weeks with exceptionally high sales. These outliers could be due to special events, promotions, holidays, or other factors that cause a spike in sales.

## Plot weekly sales over time



```
# Plot weekly sales over time
plt.figure(figsize=(14, 4))
sns.lineplot(x='Date', y='Weekly_Sales', data=data)
plt.title('Weekly Sales Over Time')
plt.xlabel('Date')
plt.ylabel('Weekly Sales')
plt.xticks(rotation=45)
plt.show()
```

✓ 6.4s



### Inferences

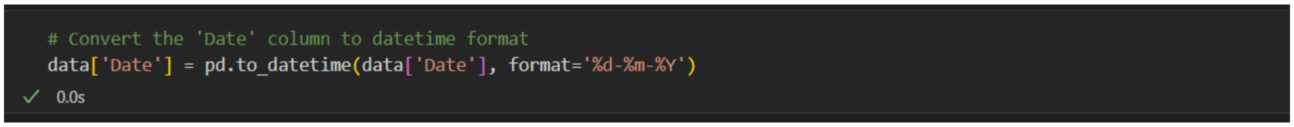
- **Trend Analysis:** The plot shows a generally stable trend in weekly sales with no significant upward or downward long-term movement. There are minor fluctuations around the mean, but no clear directional trend over the years 2010-2012.
- **Seasonality**: There are prominent spikes in sales around the same time each year, notably towards the end of the year (November-December). This suggests strong seasonal effects, likely due to the holiday shopping season. These spikes appear consistently each year, indicating a predictable seasonal pattern.
- **Anomalies and Outliers**: The most noticeable anomalies are the sharp spikes in sales around the end of each year. These are not outliers in a negative sense but rather expected peaks due to seasonal demand. There don't appear to be unexpected outliers outside of these seasonal spikes. The variations within the non-peak periods seem to be within a normal range.
- **Business Insights**: The end-of-year spikes indicate the holiday season as a period of high demand. The business can prepare for this with increased inventory, staffing, and targeted marketing efforts. Outside of the holiday season, sales appear relatively consistent. This can help in maintaining a steady inventory and regular operations.
- **Modelling Decisions**: Given the clear seasonality, models like SARIMAX that can handle seasonal effects should be considered over simpler models like ARIMA. Including holiday indicators as exogenous variables in the model can help improve

forecasting accuracy during peak seasons. A stable trend with minor fluctuations suggests that a model focusing on capturing the seasonal spikes and the consistent baseline sales will be effective.

# Data Preprocessing Steps and Inspiration

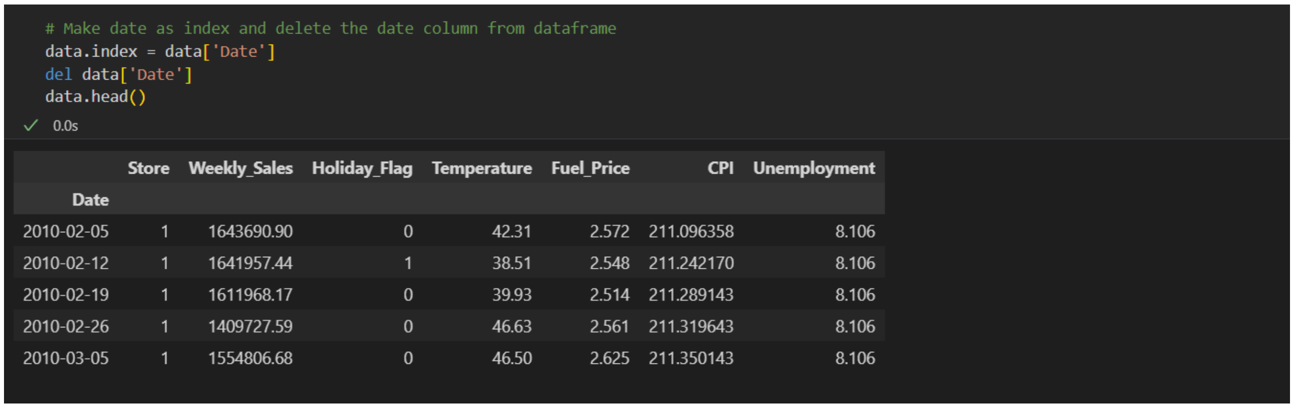
The preprocessing of the data included the following steps:

1. Convert the 'Date' column to datetime format



```
# Convert the 'Date' column to datetime format
data['Date'] = pd.to_datetime(data['Date'], format='%d-%m-%Y')
✓ 0.0s
```

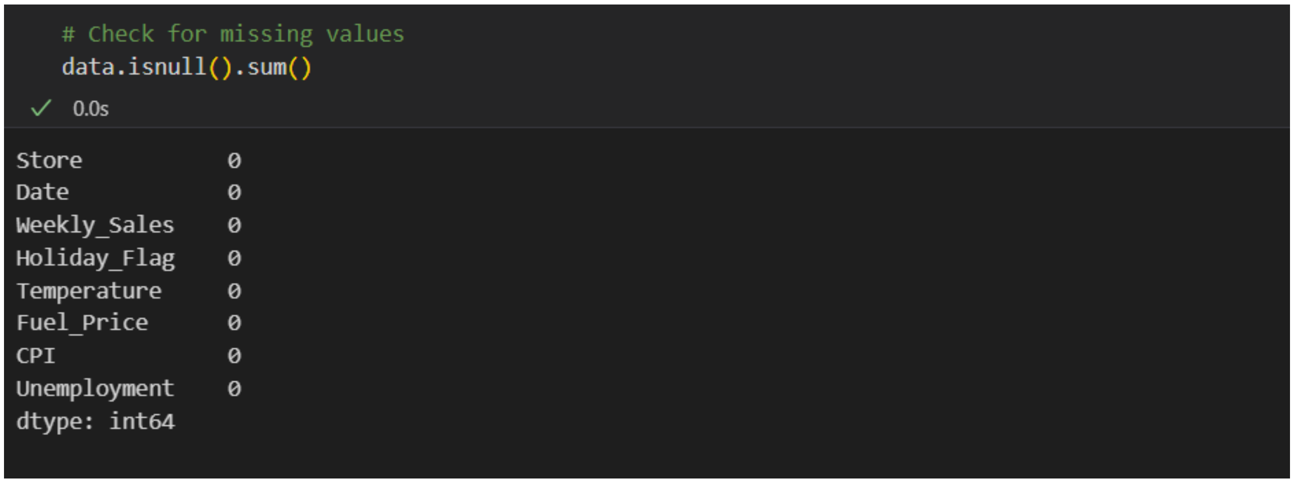
2. Make date as index and delete the date column from dataframe



```
# Make date as index and delete the date column from dataframe
data.index = data['Date']
del data['Date']
data.head()
✓ 0.0s
```

| Date | Store | Weekly_Sales | Holiday_Flag | Temperature | Fuel_Price | CPI | Unemployment |
|---|---|---|---|---|---|---|---|
| 2010-02-05 | 1 | 1643690.90 | 0 | 42.31 | 2.572 | 211.096358 | 8.106 |
| 2010-02-12 | 1 | 1641957.44 | 1 | 38.51 | 2.548 | 211.242170 | 8.106 |
| 2010-02-19 | 1 | 1611968.17 | 0 | 39.93 | 2.514 | 211.289143 | 8.106 |
| 2010-02-26 | 1 | 1409727.59 | 0 | 46.63 | 2.561 | 211.319643 | 8.106 |
| 2010-03-05 | 1 | 1554806.68 | 0 | 46.50 | 2.625 | 211.350143 | 8.106 |

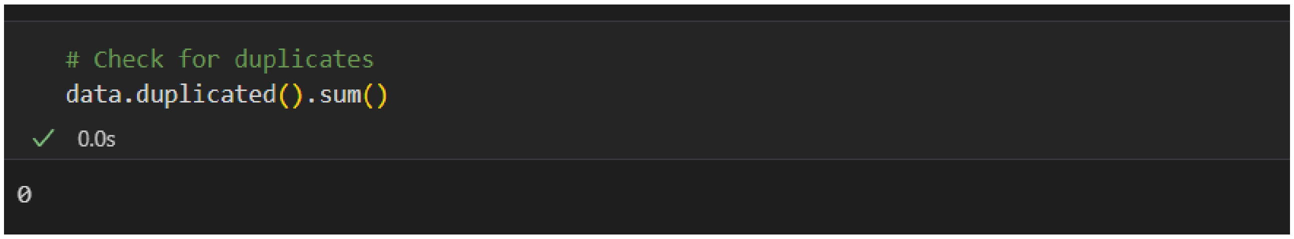
3. Handle Missing Values



```
# Check for missing values
data.isnull().sum()
✓ 0.0s
```

```
Store           0
Date            0
Weekly_Sales    0
Holiday_Flag    0
Temperature     0
Fuel_Price      0
CPI             0
Unemployment    0
dtype: int64
```

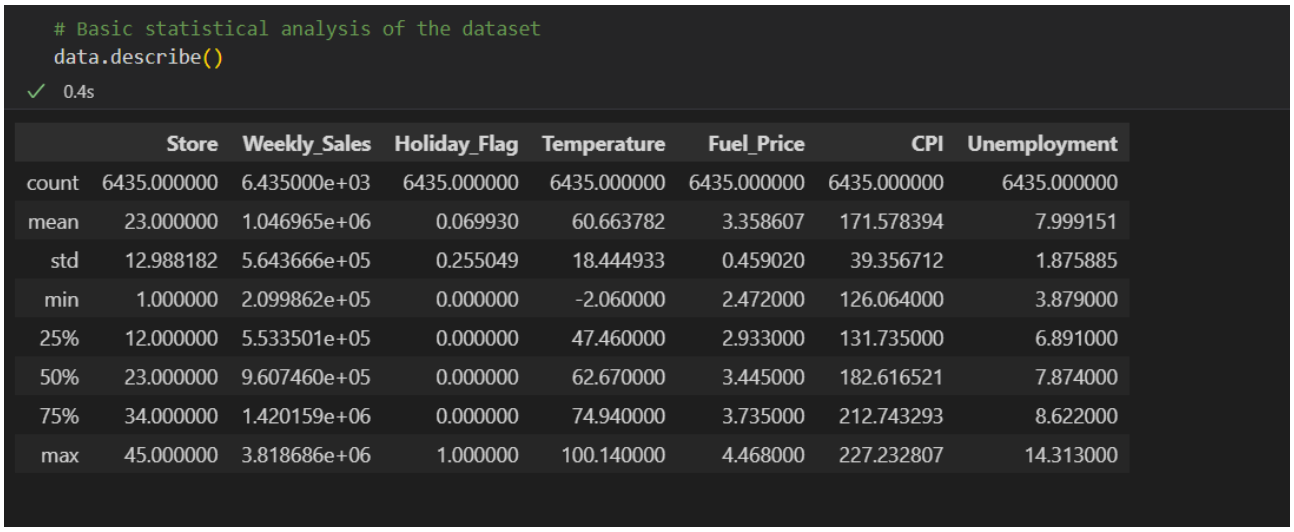
4. Handle Duplicates



```
# Check for duplicates
data.duplicated().sum()
✓ 0.0s
```

```
0
```

## 5. Statistical Analysis

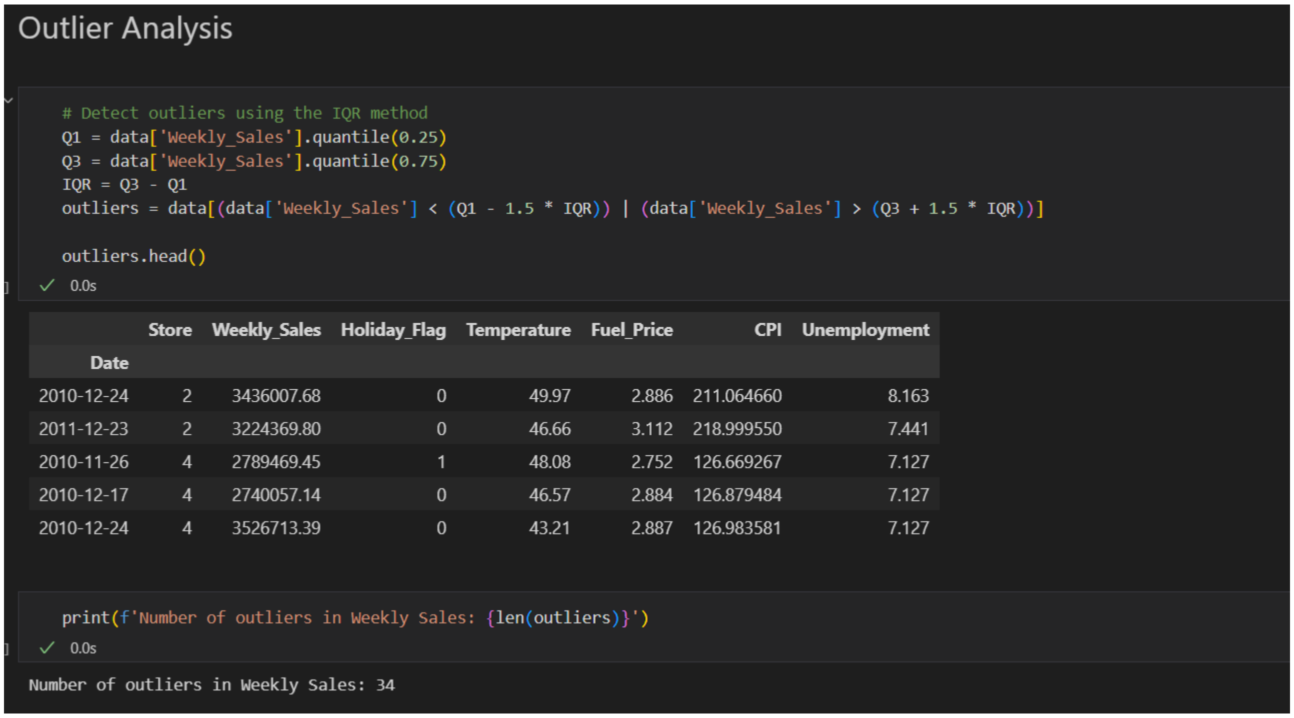


```python
# Basic statistical analysis of the dataset
data.describe()
```

✓ 0.4s

| | Store | Weekly_Sales | Holiday_Flag | Temperature | Fuel_Price | CPI | Unemployment |
|---|---|---|---|---|---|---|---|
| count | 6435.000000 | 6.435000e+03 | 6435.000000 | 6435.000000 | 6435.000000 | 6435.000000 | 6435.000000 |
| mean | 23.000000 | 1.046965e+06 | 0.069930 | 60.663782 | 3.358607 | 171.578394 | 7.999151 |
| std | 12.988182 | 5.643666e+05 | 0.255049 | 18.444933 | 0.459020 | 39.356712 | 1.875885 |
| min | 1.000000 | 2.099862e+05 | 0.000000 | -2.060000 | 2.472000 | 126.064000 | 3.879000 |
| 25% | 12.000000 | 5.533501e+05 | 0.000000 | 47.460000 | 2.933000 | 131.735000 | 6.891000 |
| 50% | 23.000000 | 9.607460e+05 | 0.000000 | 62.670000 | 3.445000 | 182.616521 | 7.874000 |
| 75% | 34.000000 | 1.420159e+06 | 0.000000 | 74.940000 | 3.735000 | 212.743293 | 8.622000 |
| max | 45.000000 | 3.818686e+06 | 1.000000 | 100.140000 | 4.468000 | 227.232807 | 14.313000 |

## 6. Outlier Analysis



Outlier Analysis

```python
# Detect outliers using the IQR method
Q1 = data['Weekly_Sales'].quantile(0.25)
Q3 = data['Weekly_Sales'].quantile(0.75)
IQR = Q3 - Q1
outliers = data[(data['Weekly_Sales'] < (Q1 - 1.5 * IQR)) | (data['Weekly_Sales'] > (Q3 + 1.5 * IQR))]

outliers.head()
```

✓ 0.0s

| | Store | Weekly_Sales | Holiday_Flag | Temperature | Fuel_Price | CPI | Unemployment |
|---|---|---|---|---|---|---|---|
| Date | | | | | | | |
| 2010-12-24 | 2 | 3436007.68 | 0 | 49.97 | 2.886 | 211.064660 | 8.163 |
| 2011-12-23 | 2 | 3224369.80 | 0 | 46.66 | 3.112 | 218.999550 | 7.441 |
| 2010-11-26 | 4 | 2789469.45 | 1 | 48.08 | 2.752 | 126.669267 | 7.127 |
| 2010-12-17 | 4 | 2740057.14 | 0 | 46.57 | 2.884 | 126.879484 | 7.127 |
| 2010-12-24 | 4 | 3526713.39 | 0 | 43.21 | 2.887 | 126.983581 | 7.127 |

```python
print(f'Number of outliers in Weekly Sales: {len(outliers)}')
```

✓ 0.0s

```
Number of outliers in Weekly Sales: 34
```

## Choosing the Algorithm for the Project

The SARIMAX (**Seasonal Autoregressive Integrated Moving Average with Exogenous Factors**) model was chosen for time series forecasting. This is an extension of SARIMA that incorporates exogenous variables for seasonal time series forecasting.

### Motivation and Reasons for Choosing the Algorithm

- ARIMA is not suitable for datasets with seasonality and external regressors like this one.
- SARIMAX can handle both seasonality and exogenous variables like Unemployment, Holiday_Flag, etc.

## Assumptions

The following assumptions were made to create the model for Walmart project.

- External factors (Unemployment, Holiday_Flag) have a linear relationship with sales.
- The time series data is stationary after differencing.
- Seasonal patterns exist in the sales data.

## Sales Forecasting

- Model Identification: Used auto-arima to determine the order of the ARIMA terms.
- Model Fitting: Fitted the SARIMA model to historical sales data for each store.
- Forecasting: Predicted sales for the next 12 weeks for each store.

## Inferences from the Project

- Seasonal trends and patterns in sales were identified.
- External factors like unemployment, Holiday impact on sales were identified.
- Top and worst-performing stores were identified, providing insights for targeted interventions.

## Future Possibilities

- **Enhanced Models**: More advanced models like below can be used for forecasting to improve result.

    - **Prophet**: Developed by Facebook, for forecasting time series data that exhibit daily, weekly, and yearly seasonality, along with holiday effects.
    - **XGBoost**: Time series forecasting algorithm by treating the problem as a supervised learning task. It captures complex patterns and non-linear relationships.

- **Real-Time Forecasting**: A real-time forecasting system integrated with the store's inventory management system can be developed.

## Conclusion

The project successfully analyzed the sales data, identified key trends, and provided a robust forecasting model to assist with inventory management. The insights gained can help the retail chain optimize its operations and meet customer demand more effectively.

# References

1. Geeksforgeeks time series forecasting
2. Walmart Sales Forecasting - Kaggle
3. Hands-On Time Series Analysis with Python - by B V Vishwas
4. Numerical Python: Scientific Computing and Data Science Applications with NumPy, SciPy, and Matplotlib - by Robert Johansson