CAPSTONE- PROJECT

Analysis and forecasting of Sales of Walmart store data on account of impact of various Factors using Time Series Model:

Submitted to:

Intellipaat

For the certification of

MITxMicroMasters- PGP in Data Science and Machine Learning

Submitted By:

GUNVANTI KUMARI

Batch 2023-2024

Contents:

CAPSTONE WALMART

About Walmart Store

Walmart is an American multinational retail corporation that operates a chain of hypermarkets, discount department stores, and grocery stores in the United States, headquartered in Bentonville, Arkansas, Walmart is one of the largest private employers globally, providing jobs to a vast number of people. Walmart was founded by Sam Walton in 1962 in Rogers, Arkansas, USA.

The first Walmart store, named Wal-Mart Discount City, opened in Rogers. Walmart is known for its low-cost retail strategy, offering a wide range of products at affordable prices. The company operates on a massive scale, leveraging economies of scale to keep costs low. Walmart operates various store formats, including Walmart Supercenters, Walmart Discount Stores, Walmart Neighborhood Markets, and Sam's Club (a membership-based warehouse club). Walmart Supercenters are large retail stores that combine a supermarket and general merchandise. They typically include groceries, electronics, clothing, home goods, and more. These are traditional discount department stores that offer a variety of products at lower prices. Neighborhood Markets are smaller-sized grocery stores that focus on providing groceries and household essentials to local communities

Walmart has invested heavily in e-commerce, providing online shopping options through its website, The Company has also acquired various e-commerce platforms to enhance its digital presence. Walmart is one of the largest private employers globally, providing jobs to a vast number of people.

It's important to note that Walmart's business model and operations may vary based on the specific store format and location. The company has had a significant impact on the retail industry and is often studied for its innovative approaches to supply chain management and retail operations.

Problem Statement:

Problem Statement 1:

A retail store that has multiple outlets across the country are facing issues in managing the inventory - to match the demand with respect to supply.

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 the 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 modeling techniques to forecast the sales for each store for the next 12 week**

Project Objective:

Analase the Walmart store data and forecast the sale for upcoming 12 weeks by using Predictive modeling technique.

Data Description:

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 the fuel in the region
CPI	Consumer Price Index
Unemployment	Unemployment Rate

Data Analysis

Exploratory Data Analysis (EDA) : It is a crucial step in the data analysis process that involves exploring and understanding the characteristics of a dataset. In Python, using pandas, we can perform EDA efficiently. Below are some common EDA tasks we can perform using a pandas DataFrame.

1. **Load Dataset:** Using Pandas we will Load and read the data on Jupyter Notebook. We will import pandas as pd
   Walmart_data = pd.read_csv("Walmart(1).csv")
2. **Basic Information:**
   Get a quick overview of the DataFrame, including the number of rows and columns, data types, and non-null counts.
   Walmart_data.info ()

3. Missing values: Identify and handle missing values.
   Walmart_data.isnull ().sum ()
4. Descriptive Statistics: Calculate summary statistics for numerical columns.
   Walmart_data.describe ()
5. Duplicated Rows: Identify and handle duplicated rows.
   Walmart_data.duplicated ().sum ()
6. Column Distribution: Examine the distribution of values in categorical columns.
   Walmart data ["column"].values. Counts ()
7. Correlation: Analyze the correlation between numerical columns.
   Walmart_data.corr ()
8. Visualization: Use visualizations to explore relationships and distributions of the dataset using Matplotlib and seaborne library. For that import matplotlib.pyplot as plt and seaborn as sns.
   Pairplot for numerical column:
   Sns.pairplot("Walmart_data")
   Plt.show ()
   Histogram for Numerical columns:
   Plt.hist (**Walmart_data ["weekly sales"], bins =20, color = 'blue'**)
   Plt.xlabel ()
   Plt.ylabel ()
   Plt. title ()
   Plt.show ()

9. Grouping and Aggregation: Group the data and perform aggregations
   Grouped_data = **Walmart_data.groupby ('store') ['weekely_sale'].mean ()**

Sales Forecasting using Predictive Modeling

Time Series Model: Time series analysis involves analyzing data points collected over time to understand patterns, trends, and make predictions about future values.

Basic time series concepts are **trend, seasonality, and autocorrelation**. Using pandas for handling Time Series Data and statsmodels or scikit learn for time series analysis. Try different time series models such as ARIMA, SARIMA, Exponential Smoothing, or machine learning models (e.g., LSTM for neural networks).

statsmodels.tsa contains model classes and functions that are useful for time series analysis. Basic models include univariate autoregressive models (AR), vector autoregressive models (VAR) and univariate autoregressive moving average models (ARMA).Autocorrelations, partial autocorrelation function and periodogram, as well as the corresponding theoretical properties of ARMA or related processes.

Trend: To analyze the trend in a time series dataset, such as weekly sales over time, you can use techniques like moving averages, polynomial regression, or statistical decomposition. Here, I'll demonstrate using a simple moving average and a polynomial regression to identify and visualize the trend in weekly sales.

Moving Average: A moving average smoothens out fluctuations in data and helps reveal the underlying trend.

```
Import pandas as pd

Import matplotlib.pyplot as plt


# Assuming Walmart_data is my DataFrame with 'Date' and 'Weekly Sales' columns

Walmart_data ['Date'] = pd.to_datetime(Walmart_data ['Date'])

Walmart_data.set_index ('Date', inplace=True)


# Calculate the 4-week moving average

Walmart_data ['MA_4weeks'] = Walmart_data ['Weekly Sales'].rolling (window=4).mean ()


# plotting the original data and the moving average

Plt. Figure (figsize= (12, 6))

plt.plot(Walmart_data ['Weekly Sales'], label='Weekly Sales', marker='o', linestyle='-', color='b')

plt.plot(Walmart_data ['MA_4weeks'], label='4-Week Moving Average', linestyle='--', color='r')

plt.title ('Weekly Sales with 4-Week Moving Average')

plt.xlabel ('Date')

plt.ylabel ('Weekly Sales')

plt. legend ()

plt. Show()
```

Seasonality: Identifying seasonality in time series data is important for understanding recurring patterns or fluctuations that repeat at regular intervals. Seasonality often corresponds to specific time frames, such as daily, weekly, or yearly cycles. Let's explore how to identify and visualize seasonality in the Walmart dataset using Python.

Autocorrelation: One common approach to identifying seasonality is by decomposing the time series into its components: trend, seasonality, and residual. The statsmodels library provides a convenient function for seasonal decomposition.

from statsmodels.tsa. Seasonal import seasonal decompose

assuming Walmart_data is your DataFrame with 'Date' and 'Weekly_Sales' columns

Walmart_data ['Date'] = pd.to_datetime (Walmart_data ['Date'])

Walmart_data.set_index ('Date', inplace=True)

Perform seasonal decomposition

Result = seasonal_decompose (Walmart_data ['Weekly_Sales'], model='additive', period=52)

Adjust the period based on the expected seasonality

Plot the components: trend, seasonality, and residual

plt. figure (figsize= (12, 8))

The first subplot (observed) shows the original time series data

plt. Subplot(4, 1, 1)

plt.plot (result.observed, label='Observed')

plt.legend ()

The second subplot (trend) displays the trend component.

plt.subplot (4, 1, 2)

plt.plot (result.trend, label='Trend')

plt.legend ()

The third subplot (seasonal) illustrates the seasonality component.

plt.subplot (4, 1, 3)

plt.plot (result.seasonal, label='Seasonal')

plt.legend ()

The fourth subplot (residual) represents the residual component.

plt.subplot (4, 1, 4)

plt.plot (result.resid, label='Residual')

plt.legend ()

plt.tight_layout ()

Plt. Show ()

SEASONAL PATTERNSS

We can also explore seasonal patterns by aggregating weekly sales over different time periods (e.g., days of the week or months) and visualizing the aggregated results.

assuming df is your DataFrame with 'Date' and 'Weekly_Sales' columns

Walmart_data ['Date'] = pd.to_datetime (Walmart_data ['Date'])

Walmart_data ['Day_of_Week'] = Walmart_data ['Date'].dt.day_name ()

Aggregate weekly sales by day of the week

weekly_sales_by_day = Walmart_data.groupby('Day_of_Week')['Weekly_Sales'].mean()

Plot the aggregated weekly sales

plt. Figure(figsize= (10, 6))

weekly_sales_by_day.reindex(['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']).plot(kind='bar', color='skyblue')

plt.title ('Average Weekly Sales by Day of the Week')

plt.xlabel('Day of the Week')

plt.ylabel('Average Weekly Sales')

plt.show()

 # Train-test split

Split data into Train dataset (80%) and Test dataset (20%) to apply the time series model, we will train the train dataset on model and then test the accuracy of the result on test dataset

train_size = int(len(new_data) * 0.8)

train, test = new_data[:train_size], new_data[train_size:]

Now fit the data in ARIMA model

from statsmodels.tsa.arima.model import ARIMA

train = new_data.iloc [:105]['Weekly_Sales']

test = new_data.iloc[105:]['Weekly_Sales']

model = ARIMA (train, order=(1,0,2))

model_fit = model.fit()

print (model_fit. Summary ())

new_data ['predict'] = model_fit.predict (start = 75, end=len (train) +len (test)-1)

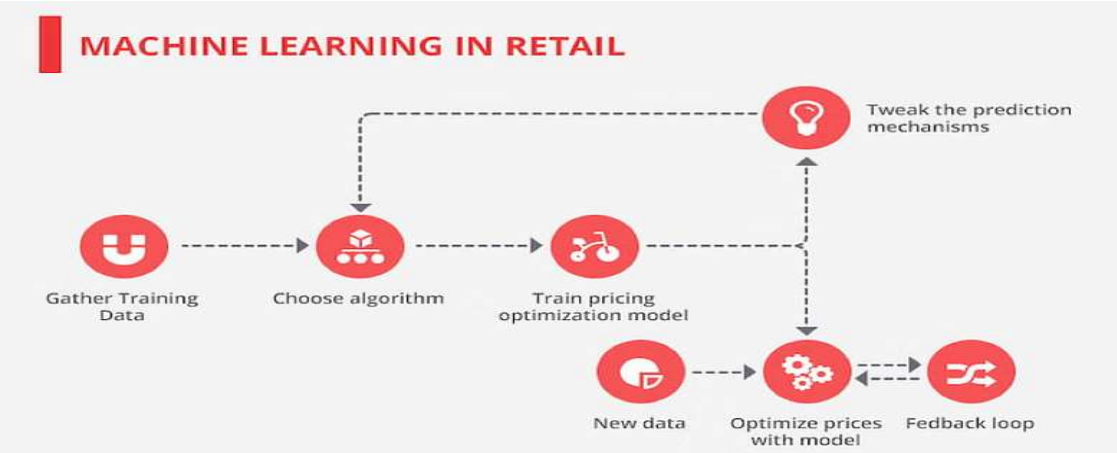
new_data [['Weekly_Sales', 'predict']].plot ()

FORCAST: We can predict future sale on the basis of past Outcomes data.

forecast = model_fit.forecast (steps = 12)

new_data.plot ()

forcast.plot()



Use Case of Time Series Model

Time series models are used to analyze and predict trends and patterns in sequential data over time. These models are particularly useful when dealing with data that exhibits temporal dependencies and where the order of observations matters. Some common use cases for time series models include forecasting, anomaly detection, and understanding the underlying structure of time-varying data.

1. **Forecasting: ARIMA(AutoRegressive Integration Moving Average):**
   ARIMA models are effective for forecasting time series data. They include autoregressive (AR) and moving average (MA) components and can handle trends and seasonality.

2. **Seasonal Decomposition**: Seasonal and Trend decomposition using losses (STL): STL decomposes time series data into seasonal, trend, and residual components, providing insights into each component.

3. **Exponential Smoothing**: Holt Winter Exponential Smoothing:
   This method is useful for capturing trends and seasonality in time series data.

   ```
   from statsmodels.tsa.holtwinters import ExponentialSmoothing

   # Example: Holt-Winters Exponential Smoothing
   model = ExponentialSmoothing(df['Weekly_Sales'], seasonal='add', seasonal_periods=52)
   results = model.fit()
   forecast = results.forecast (steps=12)
    # Adjust the number of steps as needed
   ```

4. **Dynamic Time Warping**(DTW): DTW measures the similarity between two time series sequences, which is useful for tasks like pattern recognition, clustering, and anomaly detection

   ```
   from fastdtw import fastdtw

   # Example: Dynamic Time Warping
   distance, path = fastdtw (time_series_1, time_series_2)
   ```

5. **Anomaly Detection**: Isolation Forest: Isolation Forest is a machine learning algorithm that can be used for detecting anomalies in time series data

   ```
   from sklearn.ensemble import IsolationForest

   # Example: Isolation Forest for anomaly detection
   model = IsolationForest (contamination=0.05)
   anomalies = model.fit_predict (df['Weekly_Sales'].values.reshape(-1, 1))
   ```

6. **Recurrent Neural Network** (RNN): RNNs are a type of neural network architecture that can capture temporal dependencies in sequential data.

```
from keras.models import Sequential
from keras.layers import LSTM, Dense

# Example: LSTM for time series forecasting
model = Sequential ()
model.add (LSTM (units=50, activation='relu', input shape=(n_steps, n_features)))
model. add (Dense (1))
```

7. **Facebook Prophet** : Prophet is an open-source forecasting tool designed for forecasting time series data with daily observations that display patterns on different time scales.

```
from fbprophet import Prophet

# Example: Facebook Prophet
model = Prophet ()
model.fit(df[['ds', 'y']])
future = model.make_future_dataframe (periods=365)

# Extend the time horizon as needed
forecast = model.predict(future)
```

Note: Choose a time series model based on the specific characteristics of your data, the nature of the problem, and the insights you want to gain or predictions you want to make. Keep in mind that the choice of model parameters and hyperparameters may require tuning based on your specific use case.

REFRENCES:

https://www.statsmodels.org/stable/tsa.html

https://www.kaggle.com/

https://www.kaggle.com/search?q=arima+model+date%3A90+in%3Atopics

https://www.analyticsvidhya.com/blog/2021/12/time-series-forecasting-with-extreme-learning-machines/