Subject: Fundamentals of Data Science

Title: Walmart Sale Forecasting

Author : Apeksha Chikane - 23030141014

Problem Statement:

We are forecasting Walmart sales based on historical data and various factors such as holidays, environmental conditions, geographic location, and the relationship between two or more product sales. The project aims to analyse weekly sales data from Walmart stores against features like holidays, temperature, fuel price, and CPI (Customer Price Index) using a secondary dataset. The goal is to perform Exploratory Data Analysis (EDA) to understand the relationships between these features and weekly sales, ultimately enabling accurate sales forecasting for the retail giant.

Colab Link :

https://shorturl.at/WktPa

Walmart Exploratory Data Analysis:

SecondaryDataSet: https://www.kaggle.com/datasets/yasserh/walmart-dataset

About this Dataset:

This is the historical data that covers sales from 2010-02-05 to 2012-11-01, in the file Walmart. 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 – Nonholiday 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

Total row count : 6436

No	Date	Weekly_Sa	Holiday_Fl	Temperatu	Fuel_Price	CPI	Unemployment
1	05-02-2010	1643691	0	42.31	2.572	211.0964	8.106
1	12-02-2010	1641957	1	38.51	2.548	211.2422	
1	19-02-2010	1611968	0	39.93	2.514	211.2891	
1	26-02-2010	1409728	0	46.63		211.3196	
1	05-03-2010		0	46.63	2.625	211.3501	8.106
1	12-03-2010	1439542	0	46.63	2.667	211.3806	8.106
1	19-03-2010	1472516	0	46.63	2.72	211.2156	8.106
1	26-03-2010	1404430	0	51.45	2.732	211.018	8.106
1	02-04-2010	1594968	0	62.27	2.719	210.8204	7.808
1	09-04-2010	1545419	0	65.86	2.77	210.6229	7.808
1	16-04-2010	1466058	0	66.32	2.808	210.4887	7.808
1	23-04-2010	1391256	0	64.84	2.795	210.4391	7.808
1	30-04-2010	1425101	0	67.41	2.78	210.3895	7.808
1	07-05-2010	1603955	0	72.55		210.34	7.808
1	14-05-2010	1494252	0	74.78	2.854	210.3374	7.808
1	21-05-2010	1399662	0	76.44	2.826	210.6171	7.808
1	28-05-2010	1432070	0	80.44	2.759	210.8968	7.808
1	04-06-2010	1615525	0	80.69	2.705	211.1764	7.808
1	11-06-2010	1542561	0	80.43	2.668	211.4561	7.808
1	18-06-2010	1503284	0	84.11	2.637	211.4538	7.808
1	25-06-2010	1422712	0	84.34	2.653	211.3387	
1	02-07-2010	1492418	0	80.91	2.669	211.2235	
1	09-07-2010	1546074	0	80.48	2.642	211.1084	
1	16-07-2010	1448939	0	83.15	2.623	211.1004	
1	23-07-2010		0	83.36	2.608	211.2351	
1	30-07-2010	1371987	0	81.84	2.64	211.3699	
1	06-08-2010	1605492	0	87.16	2.627	211.5047	7.787
1	13-08-2010	1508238	0	87	2.692	211.6394	7.787

Walmart

Ready Accessibility: Unavailable Average: 40711 Count: 6436 Sum: 261975285

Exploratory Data Analysis

Basic Statistics

- Explore distribution and relationships between variables
- Handling Outliers
- Handling Missing Values
- Correlation analysis/regression
- Visualization of data

Step 1 : Import libraries and dataset:

Import libraries and dataset

```
import numpy as np
import pandas as pd
# Visualization
import matplotlib.pyplot as plt
import seaborn as sns
```

```
[2] #importing the dataset
data= pd.read_csv("/content/Walmart .csv", encoding='latin-1')
```

Here we have imported important python libraries numpy,pandas and for visualization matplotlib.pyplot and seaborn

And then we have imported dataset using pd.read_csv command.

Step 2: Understanding the data processing and cleaning

Understand the data

- Data Preprocessing
- Data Cleaning

```
[3] data.shape
```

```
(6435, 8)
```

```
[4] data.duplicated().sum()
```

```
0
```

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

	No	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
0	1	05-02-2010	1643690.90	0	42.31	2.572	211.096358	8.106
1	1	12-02-2010	1641957.44	1	38.51	2.548	211.242170	NaN
2	1	19-02-2010	1611968.17	0	39.93	2.514	211.289143	NaN
3	1	26-02-2010	1409727.59	0	46.63	NaN	211.319643	NaN
4	1	05-03-2010	NaN	0	46.63	2.625	211.350143	8.106

- In this step we have perform data processing and data cleaning
- First we have count the no. of rows and columns (6435,8)
- Then we have count duplicate values we have 0 duplicate values Display data head

```
#data summary and data types
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6435 entries, 0 to 6434
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   No            6435 non-null   int64
 1   Date          6435 non-null   object
 2   Weekly_Sales  6393 non-null   float64
 3   Holiday_Flag  6435 non-null   int64
 4   Temperature   6435 non-null   float64
 5   Fuel_Price    6426 non-null   float64
 6   CPI           6435 non-null   float64
 7   Unemployment  6416 non-null   float64
dtypes: float64(5), int64(2), object(1)
memory usage: 402.3+ KB
```

```
[7] data.Holiday_Flag.unique()
```

```
array([0, 1])
```

```
[8] #separate numerical data
data_numerical= data.drop(['Holiday_Flag','Date'], axis=1)
data_numerical.columns
```

```
Index(['No', 'Weekly_Sales', 'Temperature', 'Fuel_Price', 'CPI',
       'Unemployment'],
      dtype='object')
```

- The we displayed all colums summary information like column no,column name, row count of each column data type
- Then for holidays we have given flag 0 for non holidays and 1 for holidays
- Then we have separated the numerical data holiday flag, no,week sales,temperature,fuel price,CPI

```
#separate numerical data
data_numerical= data.drop(['Holiday_Flag','Date'], axis=1)
data_numerical.columns
```

```
Index(['No', 'Weekly_Sales', 'Temperature', 'Fuel_Price', 'CPI',
       'Unemployment'],
      dtype='object')
```

```
[9] #converting Date column from object to datetime
data['Date'] = pd.to_datetime(data['Date'], format='%d-%m-%Y')
```

```
# Extracting year and month into separate columns
data['Year'] = data['Date'].dt.year
data['Month'] = data['Date'].dt.month
```

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

	No	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	Year	Month
0	1	2010-02-05	1643690.90	0	42.31	2.572	211.096358	8.106	2010	2
1	1	2010-02-12	1641957.44	1	38.51	2.548	211.242170	NaN	2010	2
2	1	2010-02-19	1611968.17	0	39.93	2.514	211.289143	NaN	2010	2
3	1	2010-02-26	1409727.59	0	46.63	NaN	211.319643	NaN	2010	2
4	1	2010-03-05	NaN	0	46.63	2.625	211.350143	8.106	2010	3

vailable

0s completed at 16:33

- Convert the date column from object to datetime the displayed data head

Step 3: Handlling Outliers

Handling Outliers

```
[12] #defined a function to find outliers in the numerical data type column
def find_outliers_IQR(df):
    q1=df.quantile(0.25)
    q3=df.quantile(0.75)
    IQR=q3-q1
    outliers = df[((df<(q1-1.5*IQR)) | (df>(q3+1.5*IQR)))]
    print(len(outliers))

#replacing outliers with
def remove_outlier(df):
    q1=df.quantile(0.25)
    q3=df.quantile(0.75)
    IQR=q3-q1
    df[((df<(q1-1.5*IQR)) | (df>(q3+1.5*IQR)))]=np.nan
```

```
[13] #finding number of the outliers in the numerical columns
column_num=data_numerical.columns
for i in column_num:
    print("Number of outliers in the column",i,":")
    find_outliers_IQR(data[i])

for i in column_num:
    remove_outlier(data[i])
    print("Value count after treatment in",i,":")
```

```
column_num=data_numerical.columns
for i in column_num:
    print("Number of outliers in the column",i,":")
    find_outliers_IQR(data[i])

for i in column_num:
    remove_outlier(data[i])
    print("Value count after treatment in",i,":")
    find_outliers_IQR(data[i])
```

```
Number of outliers in the column No :
0
Number of outliers in the column Weekly_Sales :
34
Number of outliers in the column Temperature :
2
Number of outliers in the column Fuel_Price :
0
Number of outliers in the column CPI :
0
Number of outliers in the column Unemployment :
481
Value count after treatment in No :
0
Value count after treatment in Weekly_Sales :
0
Value count after treatment in Temperature :
0
Value count after treatment in Fuel_Price :
0
Value count after treatment in CPI :
0
```

0s completed at 16:33

- We displayed outliers count present in each column
- As shown in above screen shots

```
#checking if na columns
data.isnull().sum()
```

```
No                0
Date              0
Weekly_Sales     76
Holiday_Flag      0
Temperature       2
Fuel_Price        9
CPI               0
Unemployment    500
Year              0
Month             0
dtype: int64
```

+ Code + Text

- We have count Null values present in each column
- Colum weekly sales and unemployment has most null values

Step 4: Handling Missing values

Handling Missing Values by replacing with mean

```
[15] #replacing na with mean for numerical data
     for i in column_num:
         data[i].fillna(data[i].mean(), inplace= True)
```

```
[16] #percentage of null values
     data.isnull().mean().round(3)*100
```

```
No              0.0
Date            0.0
Weekly_Sales    0.0
Holiday_Flag    0.0
Temperature     0.0
Fuel_Price      0.0
CPI             0.0
Unemployment    0.0
Year            0.0
Month           0.0
dtype: float64
```

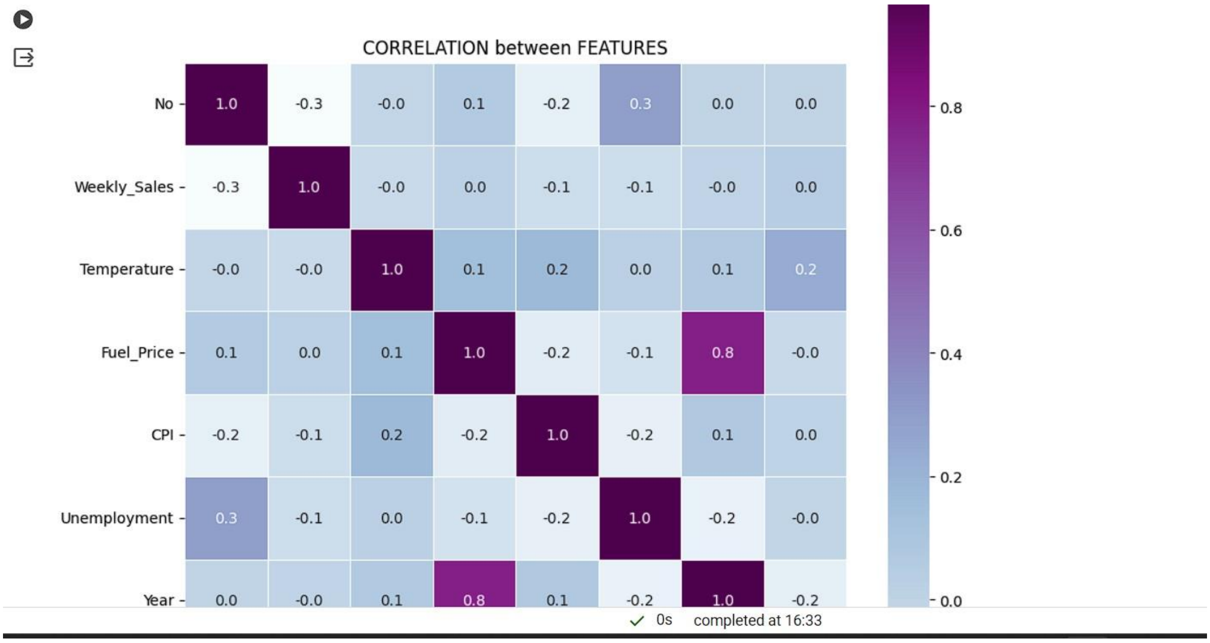
- We have replace all missing value with meen of each coulum
- Then again we count null value it was 0
- All value are replced it will help analys data easily

Step 5: We have calculated correlation between each features with each other

Correlation Analysis Between Features

```
data_1= data.drop(['Holiday_Flag','Date'], axis=1)
corr=data_1.corr()
f,ax=plt.subplots(figsize=(10,10))
sns.heatmap(corr,annot=True,linewidths=0.5,fmt=".1f",ax=ax,cmap="BuPu",square=True)
plt.title("CORRELATION between FEATURES")
plt.show()
```

Output:



Analysis: We have performed a correlation analysis between the features to check if there any features related to each other. As we can see some positive correlations between Temperature and Month, Fuel_price and Year, and Temperature and Year.

Step 6: Displayed summary statistics of the whole data

Statistics summary for the data

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

	No	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	Year	Month
count	6435.000000	6.435000e+03	6435.000000	6435.000000	6435.000000	6435.000000	6435.000000	6435.000000	6435.000000
mean	23.000000	1.035855e+06	0.069930	60.650902	3.358770	171.578394	7.720936	2010.965035	6.447552
std	12.988182	5.407040e+05	0.255049	18.426314	0.458663	39.356712	1.195058	0.797019	3.238308
min	1.000000	2.099862e+05	0.000000	6.230000	2.472000	126.064000	4.308000	2010.000000	1.000000
25%	12.000000	5.559849e+05	0.000000	47.340000	2.934000	131.735000	6.961000	2010.000000	4.000000
50%	23.000000	9.657888e+05	0.000000	62.630000	3.445000	182.616521	7.720936	2011.000000	6.000000
75%	34.000000	1.407870e+06	0.000000	74.940000	3.735000	212.743293	8.458000	2012.000000	9.000000
max	45.000000	2.685352e+06	1.000000	100.140000	4.468000	227.232807	10.926000	2012.000000	12.000000

Analysis as you see in Descriptive statistics table we have calculated row count, mean, standard deviation (how much that value is deviated from the mean)Q1, Q2, Q3, Mean nad max values of each column from these we can say it is the normal data

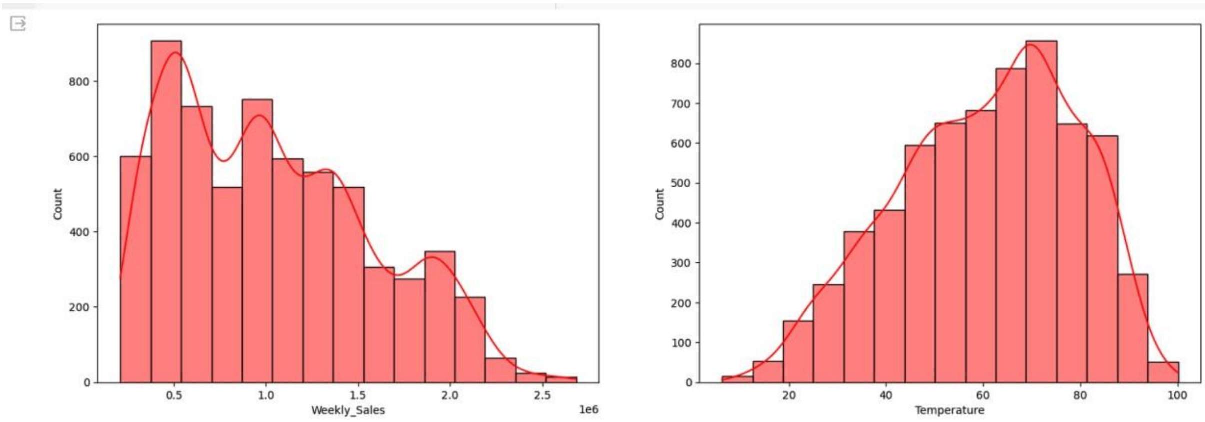
Step 7: Visualization of data

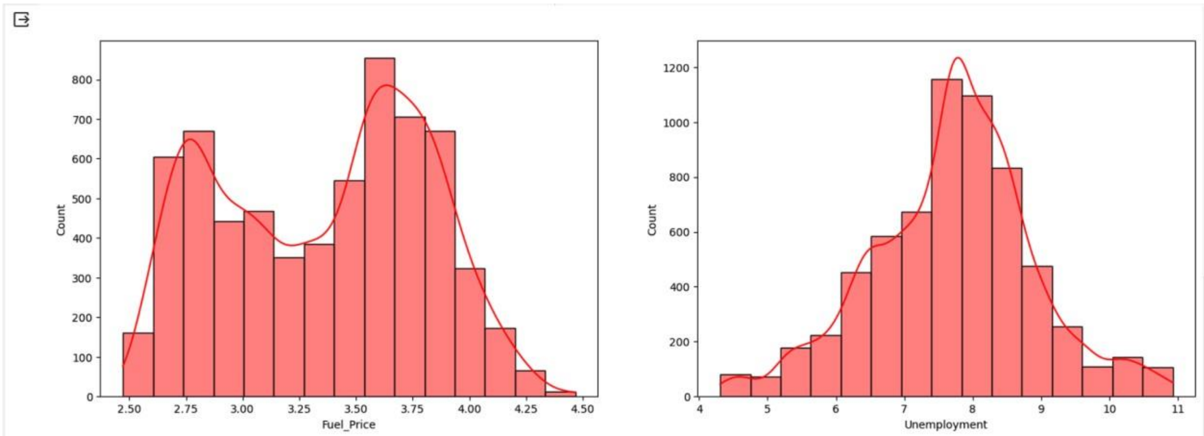
Visualization of data

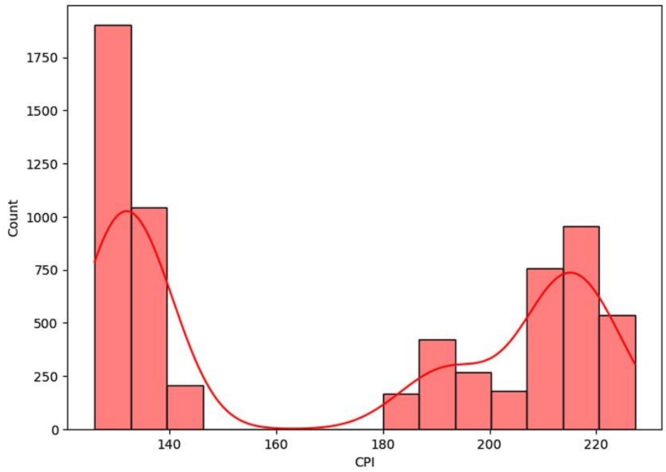
```
columns = ['Weekly_Sales', 'Temperature', 'Fuel_Price', 'Unemployment', 'CPI']
plt.figure(figsize=(18, 20))
for i,col in enumerate(columns):
    plt.subplot(3, 2, i+1)
    sns.histplot(data = data, x = col, kde = True, bins = 15, color = 'r')
plt.show()
```

We have plot weekly sales, temperature, fuel price, Unemployment, CPI against count

Output:







Conclusion:

- The distribution of Weekly_Sales is right skewed, this is normal because the weekly sales may be high in some time.
- Temperature and Unemployment have normal distribution.
- CPI and Fuel Price have bimodal distribution.

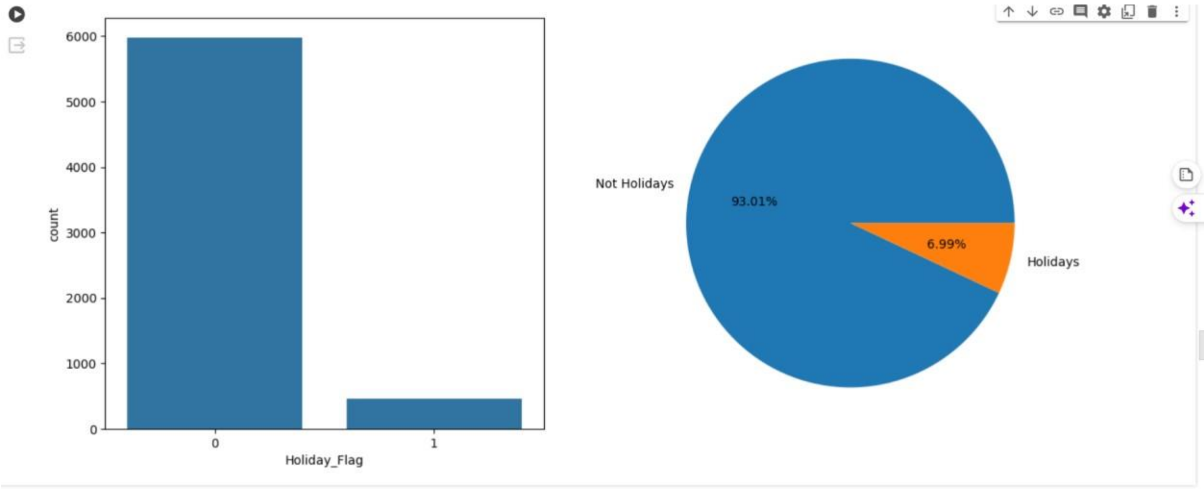
Step 8: visual Comparison of holidays and not holidays days

```
fig, ax = plt.subplots(1, 2, figsize = (14, 6))
sns.countplot(data = data, x = 'Holiday_Flag', ax = ax[0])

ax[1].pie(data['Holiday_Flag'].value_counts().values,
          labels = ['Not Holidays', 'Holidays'],
          autopct = '%1.2f%%')

plt.show()
```

Output:

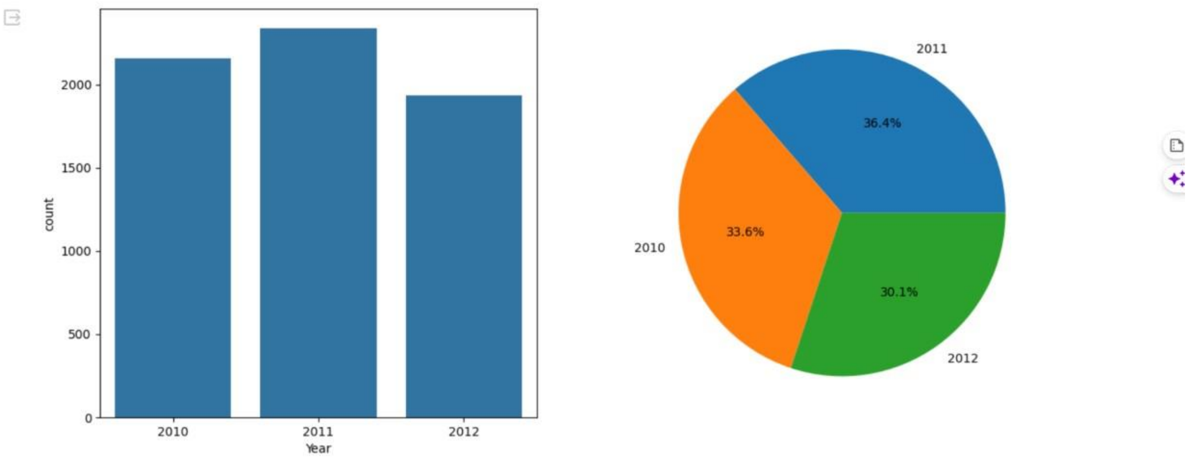


Analysis: Days of no holiday are the most frequent than days of holiday in the dataset with a percentage of 93 % and this is normal.

Step 9: Year Wise Analysis of Sales

```
fig, ax = plt.subplots(1, 2, figsize = (14, 6))
sns.countplot(data = data, x = 'Year', ax = ax[0])
ax[1].pie(data['Year'].value_counts().values,
          labels = data['Year'].value_counts().index,
          autopct = '%1.1f%%')
plt.show()
```

Output:



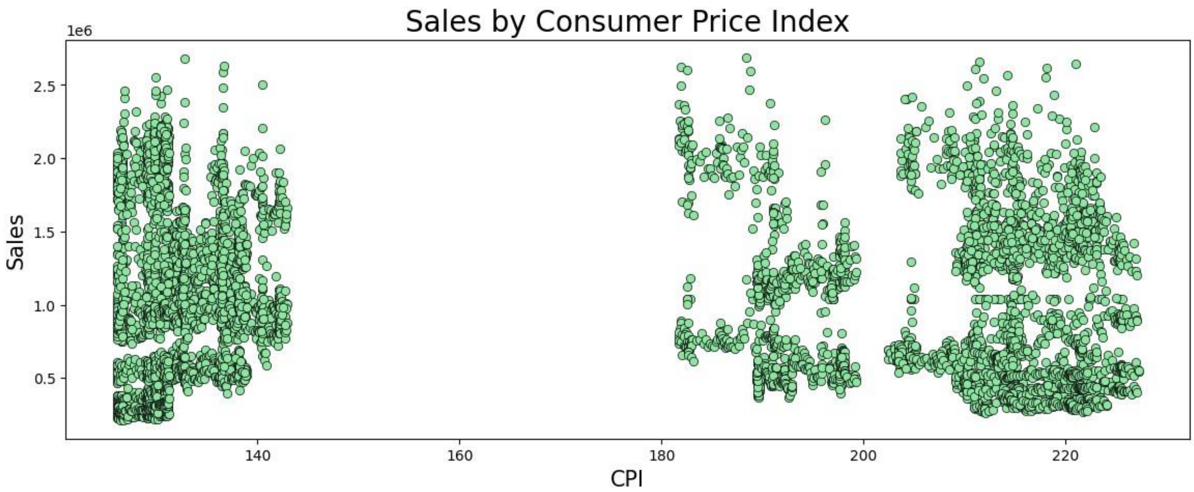
Analysis: 2011 is the highest in the dataset with 36.4% because most of the weekly sales were recorded during this year.

Step 10: Scatter diagram of Consumer Price Index (CPI) against Sales

```
[ ] plt.figure(figsize = (14, 5))
    sns.scatterplot(data = data,
                    x = 'CPI',
                    y = 'Weekly_Sales',
                    color = '#8de5a1',
                    edgecolor = "black")

    # Add labels and title
    plt.title('Sales by Consumer Price Index', size = 20)
    plt.xlabel('CPI', size = 15)
    plt.ylabel('Sales', size = 15)
    plt.show()
```

Output:



Analysis: Consumer Price Index (CPI) does not affect sales. And based on the distribution of average consumer prices in the above figure, customers can be divided into two categories: customers who pay from 120 to 150 (Middle-class customers). customers who pay from 180 to 230 (High-class customers).

Conclusion:

We have taken a secondary dataset from Kaggle of Walmart to analyse the weekly sales against the other features Holiday Flag, Temperature, Fuel Price, CPI, Unemployment.

Perform Exploratory Data Analysis on the dataset. Exploratory Data Analysis also known as EDA is the process of analysing the data using visual techniques. Exploratory Data Analysis (EDA) is a fundamental stage in any investigation.

We have performed data preprocessing such as checking for duplicates, cleaning, finding outliers, and replacing them with null values.

We have handled null values with mean for each column.

We have performed a correlation analysis between the features to check if there any features related to each other. As we can see some positive correlations between Temperature and Month, Fuel price and Year, and Temperature and Year.

And performed descriptive statistics for all the features to check the data description and understand the data more clearly.

Visualization data

1. Visualization distribution for different features and observed the following

- The distribution of Weekly Sales is right skewed, this is normal because the weekly sales may be high in some time.
- Temperature and Unemployment have normal distribution. CPI and Fuel Price have a bimodal distribution.

2. Visualization of the holiday effect.

Days of no holiday are the most frequent than days of holiday in the dataset with a percentage of 93 % and this is normal.

3. Yearly analysis of the sales

2011 is the most highest in the dataset with 36.4% because most of the weekly sales were recorded during this year.

4. CPI analysis of the sales

Consumer Price Index (CPI) does not affect sales. And based on the distribution of average consumer prices in the above figure, customers can be divided into two categories: customers who pay from 120 to 150 (Middle-class customers). customers who pay from 180 to 230 (High-class customers).

Limitations:

The scope of the dataset restricts the research, which only looks at Walmart's weekly sales data and ignores other variables like competitor activity or macroeconomic trends. The depth of insights may be limited if there is a lack of relevant store information, such as demographics unique to a given area or store size. The preprocessing technique of substituting the mean for null values could distort the data and affect the analysis's accuracy, particularly if the dataset contains a lot of significant variances. A brief description of the identification and handling of outliers could compromise the analysis's robustness. Because the correlation study only looks at linear correlations, it may miss nonlinear interactions between variables.