Capstone Project on “walmart” Dataset

Problem Statement:

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. 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. Irrespective of the size of the business, inventory management is one of the most challenging processes in the retail sector. The data contains historical sales data of 6435 stores across the country. Each store shows weekly sales in contrast to temperature, fuel price, consumer price index (CPI) and unemployment which affects the overall sales of the store. The task here is to find ways to improve in various areas and to forecast sales for each store for the next twelve weeks. Part of the challenge presented by this competition is modelling the effects of holiday weeks in the absence of complete/ideal historical data.

Summary & Objective :

Historical sales data for 45 Walmart stores located in different regions are available. There are certain events and holidays which impact sales on each day. The business is facing a challenge due to unforeseen demands and runs out of stock some times, due to inappropriate machine learning algorithm. Walmart would like to predict the sales and demand accurately. An ideal ML algorithm will predict demand accurately and ingest factors like economic conditions including CPI, Unemployment Index, etc. The objective is to determine the factors affecting the sales and to analyze the impact of markdowns around holidays on the sales. This is the historical data that covers sales from 2010-02-05 to 2012-11-01, in which 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

Project Objective:

The project objective is to categorize the stores into different clusters. The clusters are categorized based on the income; we manage the inventory to decide which stores require more stocks.

Data Description:

The Walmart dataset consisting of 6435 rows and 8 columns, which has the target column Weekly Sales. We could say the dataset is quite clean, where there are no missing values nor duplicate values. This file contains additional data related to the store, department, and regional activity for the given dates. It contains the following fields: • Store — the store number • Date — the week • Temperature — average temperature in the region • Fuel Price — cost of fuel in the region • CPI — the consumer price index • Unemployment — the unemployment rate

There are outliers in the Temperature column with the temperature ranging between -2.04 to 100.14. There are outliers in unemployment where the lower value is 3.879 and the upper value is 14.313.

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 Rat

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,mean_squared_error,confusion_matrix
%matplotlib inline
walmart_data = pd.read_csv('/content/walmart_data_set_prob1.csv')
walmart_data.head()
```

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Un
0	1	05-02-2010	1643690.90	0	42.31	2.572	211.096358	
1	1	12-02-	1641957.44	1	38.51	2.548	211.242170	

Data Pre-processing Steps and Inspiration: The pre-processing steps are: • Checking the unique values present in each column. • Checking the information of each column for missing values and data types present in the dataset. • Checking the descriptive statistics of the dataset. • Checking the correlation of each column with other columns. From the pre-processing steps, we don't have any missing data and there is no strong positive or negative correlation between the columns. We categorize the outlets based on the generated income among the 45 stores using the cluster analysis. This helps in the prediction of the income generated from the stores, then predict the stores weekly sales using various machine learning algorithms and then forecast the weekly sales with the time series models.

Handling missing values of features dataset

```
walmart_data["CPI"].fillna(walmart_data["CPI"].median(),inplace=True)
walmart_data["Unemployment"].fillna(walmart_data["Unemployment"].median(),inplace=True)
```

```
# Convert date to datetime format
walmart_data['Date'] =  pd.to_datetime(walmart_data['Date'])
walmart_data.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   datetime64[ns]
 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: datetime64[ns](1), float64(5), int64(2)
memory usage: 402.3 KB
```

```
walmart_data.isnull().sum()
```

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

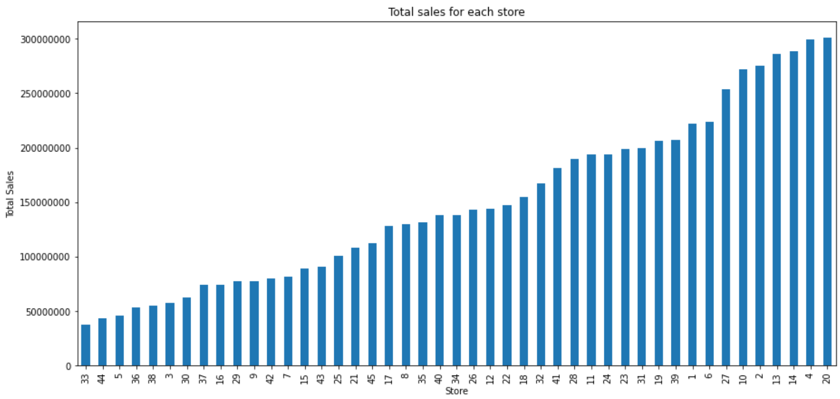
```
walmart_data["Day"]= pd.DatetimeIndex(walmart_data['Date']).day
walmart_data['Month'] = pd.DatetimeIndex(walmart_data['Date']).month
walmart_data['Year'] = pd.DatetimeIndex(walmart_data['Date']).year
walmart_data
```

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI
0	1	2010-05-02	1643690.90	0	42.31	2.572	211.096358
		2010-02-19					

store which have maximum sales

```
total_sales= walmart_data.groupby('Store')['Weekly_Sales'].sum().sort_values()
total_sales_array = np.array(total_sales)
plt.figure(figsize=(15,7))
plt.xticks(rotation=0)
plt.ticklabel_format(useOffset=False, style='plain', axis='y')
plt.title('Total sales for each store')
plt.xlabel('Store')
plt.ylabel('Total Sales')
total_sales.plot(kind='bar')
```

```
<AxesSubplot:title={'center':'Total sales for each store'}, xlabel='Store',
ylabel='Total Sales'>
```



Fthe above graph, Store which has maximum sales is store number 20 and the store which has minimum sales is the store number 33.

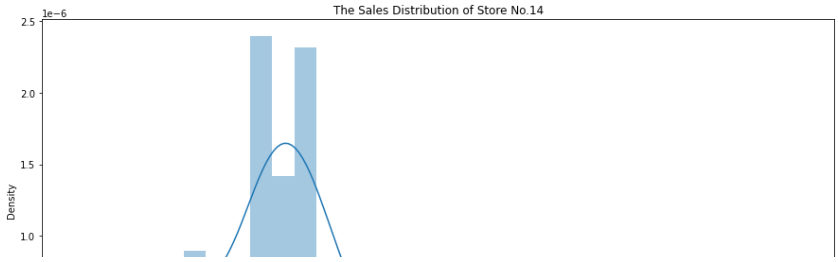
Which store has maximum standard deviation? i.e. the sales vary a lot. Also, find out the coefficient of mean to standard deviation.

```
walmart_data_std = pd.DataFrame(walmart_data.groupby('Store')['Weekly_Sales'].std().sort_values(ascending=False))
walmart_data_std.head(1).index[0] , walmart_data_std.head(1).Weekly_Sales[walmart_data_std.head(1).index[0]]
```

```
(14, 317569.9494755081)
```

store number 14 has maximum standard deviation is

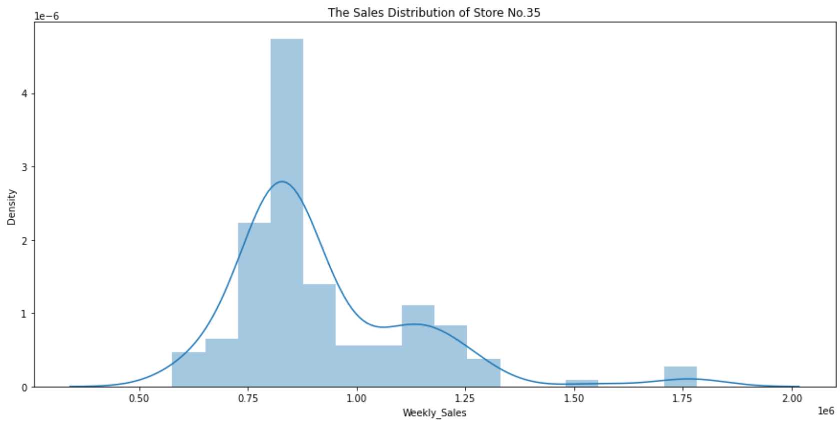
```
# Extracting the sales data for store number 14 and plotting its distribution
plt.figure(figsize=(15,7))
sns.distplot(walmart_data[walmart_data['Store'] == walmart_data_std.head(1).index[0]]['Weekly_Sales'])
plt.title('The Sales Distribution of Store No.'+ str(walmart_data_std.head(1).index[0]))
import warnings
warnings.filterwarnings('ignore')
```




```
#Calculating the coefficient of mean to standard deviation
coef = pd.DataFrame(walmart_data.groupby('Store')['Weekly_Sales'].std() / walmart_data.groupby('Store')['Weekly_Sales'].mean())
coef = coef.rename(columns={'Weekly_Sales':'Coefficient of mean to standard deviation'})
coef_max = coef.sort_values(by='Coefficient of mean to standard deviation',ascending=False)
coef_max.head(7)
```

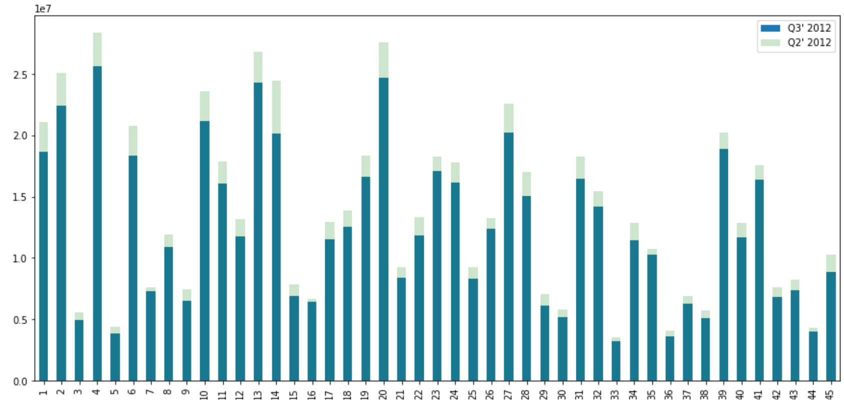
Store	Coefficient of mean to standard deviation
35	0.229681
7	0.197305
15	0.193384
29	0.183742
23	0.179721
21	0.170292
45	0.165613

```
# Distribution of store 35 has maximum coefficient of mean to standard deviation
plt.figure(figsize=(15,7))
sns.distplot(walmart_data[walmart_data['Store'] == coef_max.head(1).index[0]]['Weekly_Sales'])
plt.title('The Sales Distribution of Store No.'+str(coef_max.head(1).index[0]))
import warnings
warnings.filterwarnings('ignore')
```




```
# Sales for second and third quarter in 2012
quarter_2_sales = walmart_data[(walmart_data['Date'] >= '2012-04-01') & (walmart_data['Date'] <= '2012-06-30')].groupby('Store')['Weekly_
quarter_3_sales= walmart_data[(walmart_data['Date'] >= '2012-07-01') & (walmart_data['Date'] <= '2012-09-30')].groupby('Store')['Weekly_S

# Plotting the difference between sales for second and third quarterly
plt.figure(figsize=(15,7))
quarter_2_sales.plot(ax=quarter_3_sales.plot(kind ='bar'),kind='bar',color='g',alpha=0.2,legend=True)
plt.legend(["Q3' 2012", "Q2' 2012"])
```

```
<matplotlib.legend.Legend at 0x7f4765dab910>
```

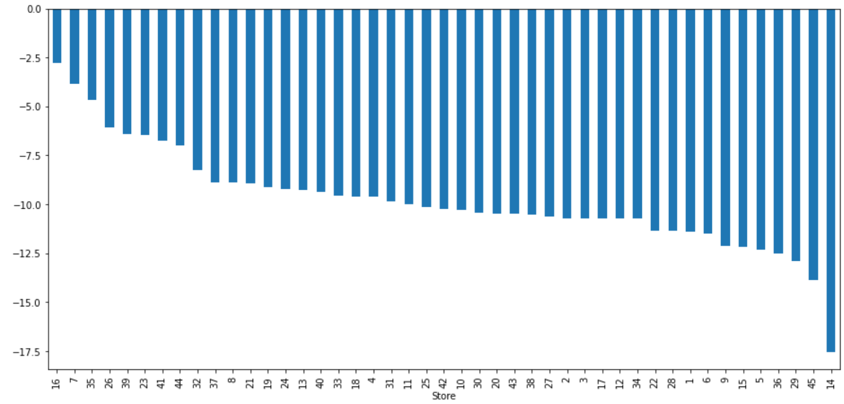



```
#Calculating Growth rate in Q3'2012
quarter_2_sales= walmart_data[(walmart_data['Date'] >= '2012-04-01') & (walmart_data['Date'] <= '2012-06-30')].groupby('Store')['Weekly_S
quarter_3_sales= walmart_data[(walmart_data['Date'] >= '2012-07-01') & (walmart_data['Date'] <= '2012-09-30')].groupby('Store')['Weekly_S
quarterly_growth_rate = ((quarter_3_sales - quarter_2_sales )/quarter_2_sales)*100
quarterly_growth_rate.sort_values(ascending=False).head()
```

```
Store
16   -2.789294
7    -3.824738
35   -4.663086
26   -6.057624
39   -6.396875
Name: Weekly_Sales, dtype: float64
```

```
plt.figure(figsize=(15,7))
quarterly_growth_rate.sort_values(ascending=False).plot(kind='bar')
```

```
<AxesSubplot:xlabel='Store'>
```



Here, there is no store which has performed better in the 3rd quarter as compared to the 2nd quarter.

4) Some holidays have a negative impact on sales. Find out holidays which have higher sales than the mean sales in non-holiday season for all stores together.

Holiday Events:

Super Bowl: 12-Feb-10, 11-Feb-11, 10-Feb-12, 8-Feb-13

Labour Day: 10-Sep-10, 9-Sep-11, 7-Sep-12, 6-Sep-13

Thanksgiving: 26-Nov-10, 25-Nov-11, 23-Nov-12, 29-Nov-13

Christmas: 31-Dec-10, 30-Dec-11, 28-Dec-12, 27-Dec-13

```
#Defining holiday dates
Super_Bowl =['12-2-2010', '11-2-2011', '10-2-2012']
Labour_Day =  ['10-9-2010', '9-9-2011', '7-9-2012']
Thanksgiving =  ['26-11-2010', '25-11-2011', '23-11-2012']
Christmas = ['31-12-2010', '30-12-2011', '28-12-2012']
```

```
#Calculating mean sales on holidays :
Super_Bowl_Sales = (pd.DataFrame(walmart_data.loc[walmart_data.Date.isin(Super_Bowl)]))['Weekly_Sales'].mean()
Labour_Day_Sales = (pd.DataFrame(walmart_data.loc[walmart_data.Date.isin(Labour_Day)]))['Weekly_Sales'].mean()
Thanksgiving_Sales = (pd.DataFrame(walmart_data.loc[walmart_data.Date.isin(Thanksgiving)]))['Weekly_Sales'].mean()
Christmas_Sales = (pd.DataFrame(walmart_data.loc[walmart_data.Date.isin(Christmas)]))['Weekly_Sales'].mean()
Super_Bowl_Sales,Labour_Day_Sales,Thanksgiving_Sales,Christmas_Sales
```

```
(1079127.9877037038, 1042427.293925926, 1471273.427777778, 960833.1115555555)
```

```
#Calculating mean sales on non-holidays :
Non_Holiday_Sales = walmart_data[walmart_data['Holiday_Flag'] == 0 ]['Weekly_Sales'].mean()
Non_Holiday_Sales
```

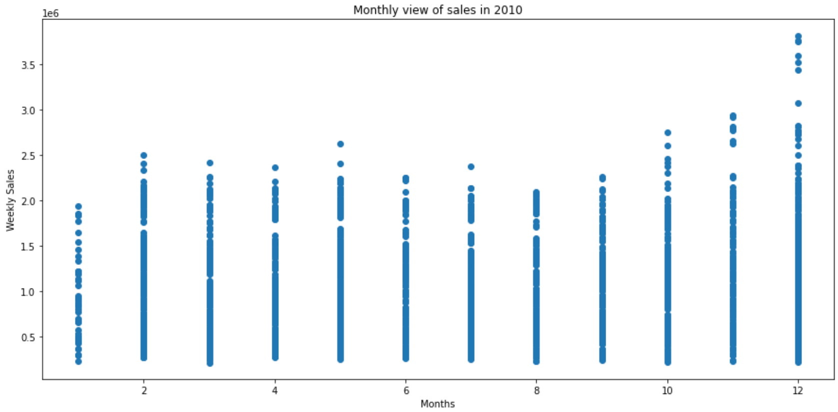
```
1041256.3802088555
```

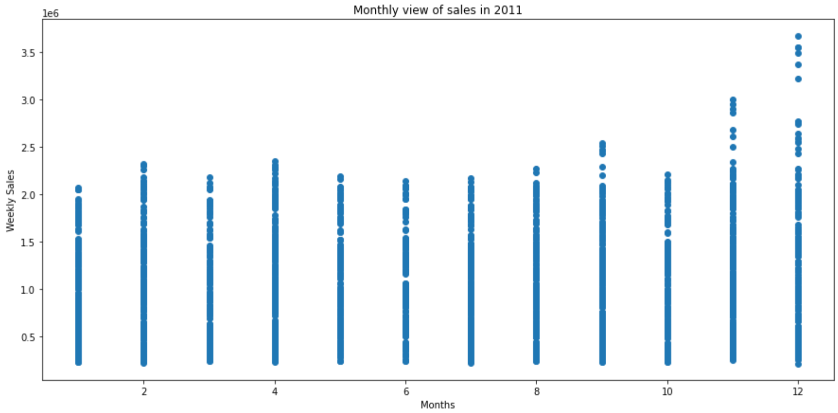
```
Mean_Sales = {'Super_Bowl_Sales' : Super_Bowl_Sales,
              'Labour_Day_Sales': Labour_Day_Sales,
              'Thanksgiving_Sales':Thanksgiving_Sales,
              'Christmas_Sales': Christmas_Sales,
              'Non_Holiday_Sales': Non_Holiday_Sales}
Mean_Sales
```

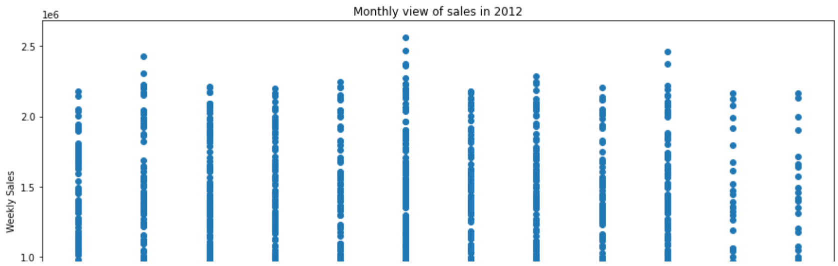
```
{'Super_Bowl_Sales': 1079127.9877037038,
 'Labour_Day_Sales': 1042427.293925926,
 'Thanksgiving_Sales': 1471273.427777778,
 'Christmas_Sales': 960833.1115555555,
 'Non_Holiday_Sales': 1041256.3802088555}
```

Clearly, Thanksgiving has higher sales than the mean sales on non-holidays.

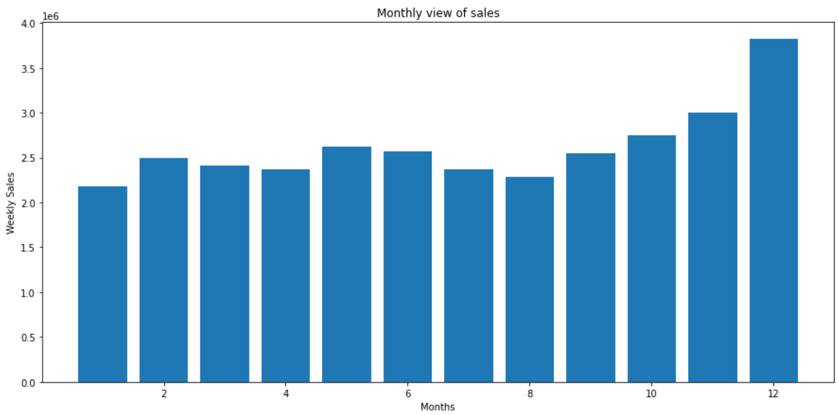
```
#Year-wise Monthly Sales

plt.figure(figsize=(15,7))
plt.scatter(walmart_data[walmart_data.Year==2010]["Month"],walmart_data[walmart_data.Year==2010]["Weekly_Sales"])
plt.xlabel("Months")
plt.ylabel("Weekly Sales")
plt.title("Monthly view of sales in 2010")
plt.show()

plt.figure(figsize=(15,7))
plt.scatter(walmart_data[walmart_data.Year==2011]["Month"],walmart_data[walmart_data.Year==2011]["Weekly_Sales"])
plt.xlabel("Months")
plt.ylabel("Weekly Sales")
plt.title("Monthly view of sales in 2011")
plt.show()

plt.figure(figsize=(15,7))
plt.scatter(walmart_data[walmart_data.Year==2012]["Month"],walmart_data[walmart_data.Year==2012]["Weekly_Sales"])
plt.xlabel("Months")
plt.ylabel("Weekly Sales")
plt.title("Monthly view of sales in 2012")
plt.show()
```







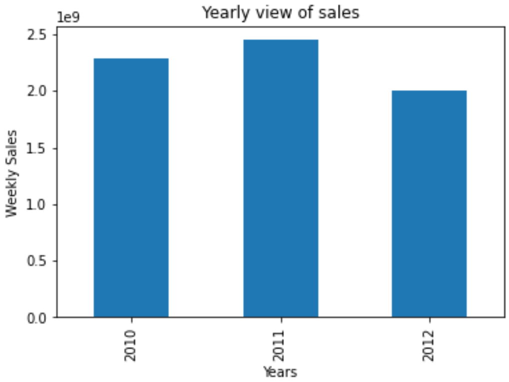

```
#Overall Monthly Sales
plt.figure(figsize=(15,7))
plt.bar(walmart_data["Month"],walmart_data["Weekly_Sales"])
plt.xlabel("Months")
plt.ylabel("Weekly Sales")
plt.title("Monthly view of sales")
plt.show()
```




```
#Yearly Sales
plt.figure(figsize=(15,7))
walmart_data.groupby("Year")[["Weekly_Sales"]].sum().plot(kind='bar',legend=False)
plt.xlabel("Years")
plt.ylabel("Weekly Sales")
```

```
plt.title("Yearly view of sales")
plt.show()
```

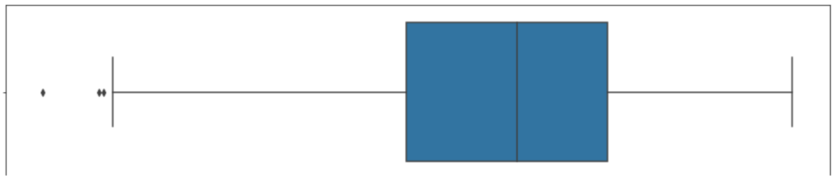
```
<Figure size 1080x504 with 0 Axes>
```



Here, overall monthly sales are higher in the month of December while the yearly sales in the year 2011 are the highest.

Build prediction models to forecast demand : Linear Regression – Utilize variables like date and restructure dates as 1 for 5 Feb 2010 (starting from the earliest date in order). Hypothesize if CPI, unemployment, and fuel price have any impact on sales. Change dates into days by creating new variable. Select the model which gives best accuracy

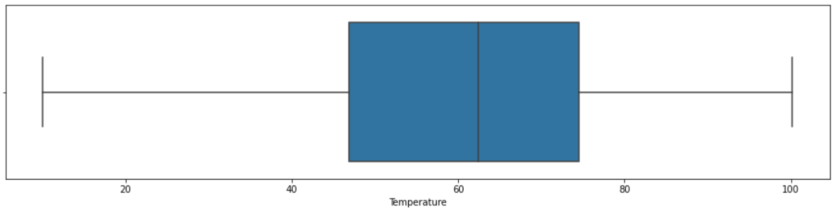
```
#Detecting outliers :
fig, axis = plt.subplots(4,figsize=(16,16))
X = walmart_data[['Temperature','Fuel_Price','CPI','Unemployment']]
for i,column in enumerate(X):
    sns.boxplot(walmart_data[column],ax=axis[i])

import warnings
warnings.filterwarnings('ignore')
```

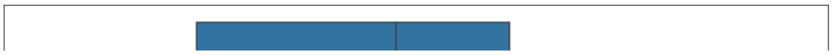


```
# Dropping outliers
walmart_data_clean = walmart_data[(walmart_data['Unemployment']<10) & (walmart_data['Unemployment']>4.5) & (walmart_data['Temperature']>1
walmart_data_clean
```

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI
0	1	2010-05-02	1643690.90	0	42.31	2.572	211.096358
1	1	2010-12-02	1641957.44	1	38.51	2.548	211.242170
2	1	2010-02-19	1611968.17	0	39.93	2.514	211.289143
3	1	2010-02-26	1409727.59	0	46.63	2.561	211.319643
4	1	2010-05-03	1554806.68	0	46.50	2.625	211.350143
...	...	...	...	...	...	...	...
6430	45	2012-09-28	713173.95	0	64.88	3.997	192.013558
6431	45	2012-05-10	733455.07	0	64.89	3.985	192.170412

```
#Checking data for outliers
fig, axis = plt.subplots(4,figsize=(16,16))
X = walmart_data_clean[['Temperature','Fuel_Price','CPI','Unemployment']]
for i,column in enumerate(X):
    sns.boxplot(walmart_data_clean[column], ax=axis[i])

import warnings
warnings.filterwarnings('ignore')
```

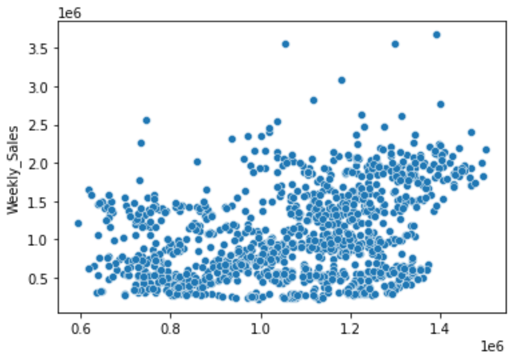




```
# Linear Regression :
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.linear_model import LinearRegression
X = walmart_data_clean[['Store','Fuel_Price','CPI','Unemployment','Day','Month','Year']]
Y = walmart_data_clean['Weekly_Sales']
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.2)
```

```
print('Linear Regression:')
print()
reg = LinearRegression()
reg.fit(X_train, Y_train)
Y_pred = reg.predict(X_test)
print('Accuracy:',reg.score(X_train, Y_train)*100)
print('Mean Absolute Error:', metrics.mean_absolute_error(Y_test, Y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(Y_test, Y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(Y_test, Y_pred)))
sns.scatterplot(Y_pred, Y_test)

import warnings
warnings.filterwarnings('ignore')
```

```
Linear Regression:

Accuracy: 12.918670991640136
Mean Absolute Error: 446056.9926640552
Mean Squared Error: 291884691367.2987
Root Mean Squared Error: 540263.5388098096
```




```
# Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor
print('Random Forest Regressor:')
print()
rfr = RandomForestRegressor()
rfr.fit(X_train,Y_train)
Y_pred = rfr.predict(X_test)
print('Accuracy:',rfr.score(X_test, Y_test)*100)
print('Mean Absolute Error:', metrics.mean_absolute_error(Y_test, Y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(Y_test, Y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(Y_test, Y_pred)))
sns.scatterplot(Y_pred, Y_test)
```

```
Random Forest Regressor:

Accuracy: 95.22884692872243
Mean Absolute Error: 68221.23311837453
Mean Squared Error: 15955201934.715458
Root Mean Squared Error: 126313.9023809947
<AxesSubplot:ylabel='Weekly_Sales'>
```

```
#print(f"Confusion Matrix :- \n{confusion_matrix(Y_test, Y_pred)}\n")
#confusion = confusion_matrix(Y_test, Y_pred)
#tn, fp, fn, tp = confusion.ravel()
```

Multiple logistic regression HAVE 0 accuracy

```
X1 = walmart_data_clean[['Store','Fuel_Price','CPI','Unemployment','Day','Month','Year']]
Y1 = walmart_data_clean['Weekly_Sales']
```

```
from sklearn import preprocessing
from sklearn import utils

#convert y values to categorical values
lab = preprocessing.LabelEncoder()
y_transformed = lab.fit_transform(Y1)

#view transformed values
print(y_transformed)
```

```
[4630 4626 4565 ... 2089 2031 2158]
```

```
MulLogRegr_x_train,MulLogRegr_x_test,MulLogRegr_y_train,MulLogRegr_y_test=train_test_split(X1,y_transformed ,train_size=0.30,random_state
```

```
MulLogRegr=LogisticRegression()
MulLogRegr.fit(MulLogRegr_x_train,MulLogRegr_y_train)
MulLogRegr_y_pred=MulLogRegr.predict(MulLogRegr_x_test)
MulLogRegr_cm=confusion_matrix(MulLogRegr_y_pred,MulLogRegr_y_test)
MulLogRegr_score=accuracy_score(MulLogRegr_y_pred,MulLogRegr_y_test)
print(f"Test Accuracy of multiple logistic regression is {MulLogRegr_score} \n")  #accuracy score
```

```
Test Accuracy of multiple logistic regression is 0.0
```

```
print('Mean Absolute Error:', metrics.mean_absolute_error(MulLogRegr_y_pred,MulLogRegr_y_test))
print('Mean Squared Error:', metrics.mean_squared_error(MulLogRegr_y_pred,MulLogRegr_y_test))
```

```
Mean Absolute Error: 1096.710679121434
Mean Squared Error: 2847334.793991416
```

Double-click (or enter) to edit

```
dTreeClassifier_x=walmart_data_clean[['Store','Fuel_Price','CPI','Unemployment','Day','Month','Year']]
dTreeClassifier_y= walmart_data_clean['Weekly_Sales']
```

```
dTreeClassifier_x_train,dTreeClassifier_x_test,dTreeClassifier_y_train,dTreeClassifier_y_test=train_test_split(dTreeClassifier_x,y_transf
```

```
dtc = DecisionTreeClassifier()
model = dtc.fit(dTreeClassifier_x_train,dTreeClassifier_y_train)
dtc_acc = accuracy_score(dTreeClassifier_y_test, dtc.predict(dTreeClassifier_x_test))
print(f"Test Accuracy of Decision Tree Classifier is {dtc_acc} \n")

print(f"Confusion Matrix :- \n{confusion_matrix(dTreeClassifier_y_test, dtc.predict(dTreeClassifier_x_test))}\n")
confusion = confusion_matrix(dTreeClassifier_y_test, dtc.predict(dTreeClassifier_x_test))
```

```
Test Accuracy of Decision Tree Classifier is 0.0

Confusion Matrix :-
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

Summary:

Here, Linear Regression, Decision tree, multiple regresssion is not an appropriate model to use which is clear from it's low accuracy. However, Random Forest Regression gives accuracy of over 95% , so, it is the best model to forecast demand.**