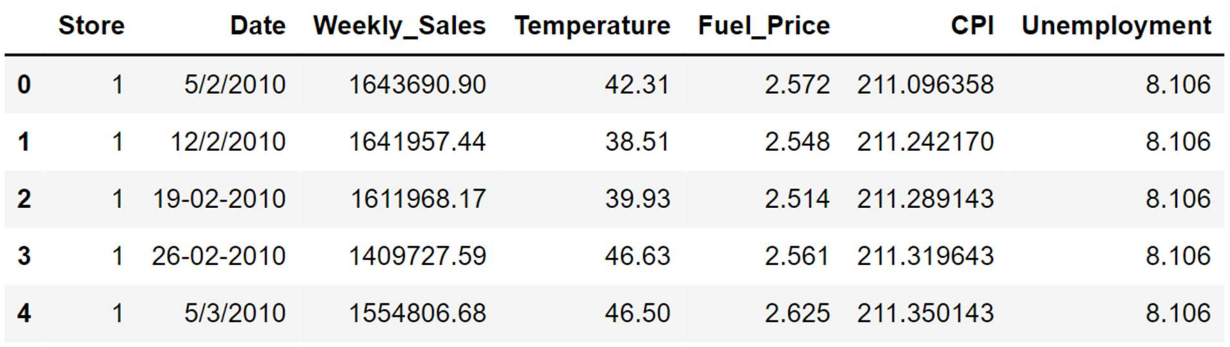
1. **Application-** Sales Prediction for better demand planning and inventory management. A lot of businesses and manufacturers face huge losses due to unwanted inventory costs or due to lost sales caused due to stockouts. My goal is to create a sales prediction model using Python and its machine-learning libraries to avoid excess inventory or stockout costs.

2. **Data**
   I used sales data for this project from Kaggle.com (dataset). The dataset is in the form of a '.csv' file and I imported it into python using Pandas library's 'pd.read_csv' function.

   The data have 8 columns which are Store, Date, Weekly_Sales, Temperature, Fuel_Price, CPI, and Unemployment. The data have 6435 rows, and I checked the shape of the data by using the '.shape' function on the data frame.

   After importing data, I cleaned the data by dropping the missing values from the data set, which can be done by using the pandas '.dropna' function. I also removed the columns which are not relevant to the model and study, I used the drop columns function to drop the columns.

   After cleaning data and removing null values and irrelevant columns, I had a look at how our data looked by using panda's '.head' function.

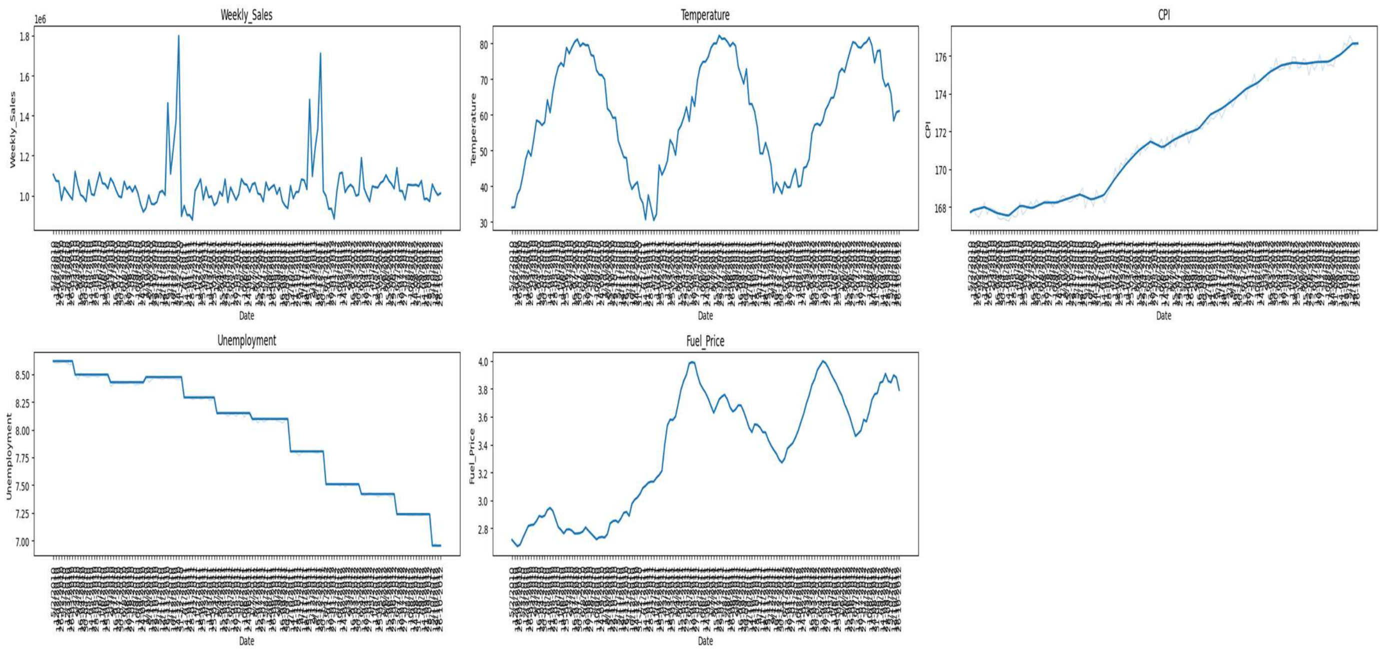


	Store	Date	Weekly_Sales	Temperature	Fuel_Price	CPI	Unemployment
0	1	5/2/2010	1643690.90	42.31	2.572	211.096358	8.106
1	1	12/2/2010	1641957.44	38.51	2.548	211.242170	8.106
2	1	19-02-2010	1611968.17	39.93	2.514	211.289143	8.106
3	1	26-02-2010	1409727.59	46.63	2.561	211.319643	8.106
4	1	5/3/2010	1554806.68	46.50	2.625	211.350143	8.106

3. Feature Exploration

After Cleaning the data, I was trying to under the data in more depth and explore how different features(columns) are moving with respect to time and look if they followed any trend, seasonality, or both.

This was done by visualizing the data, the simple line plot is used to understand the different features. These plots can be created in multiple ways; however, I chose two libraries which are Matplotlib and Seaborn.
I used seaborn to plot the line plot with Date on the x-axis and Features on the y-axis and used matplotlib for creating subplots and labels.



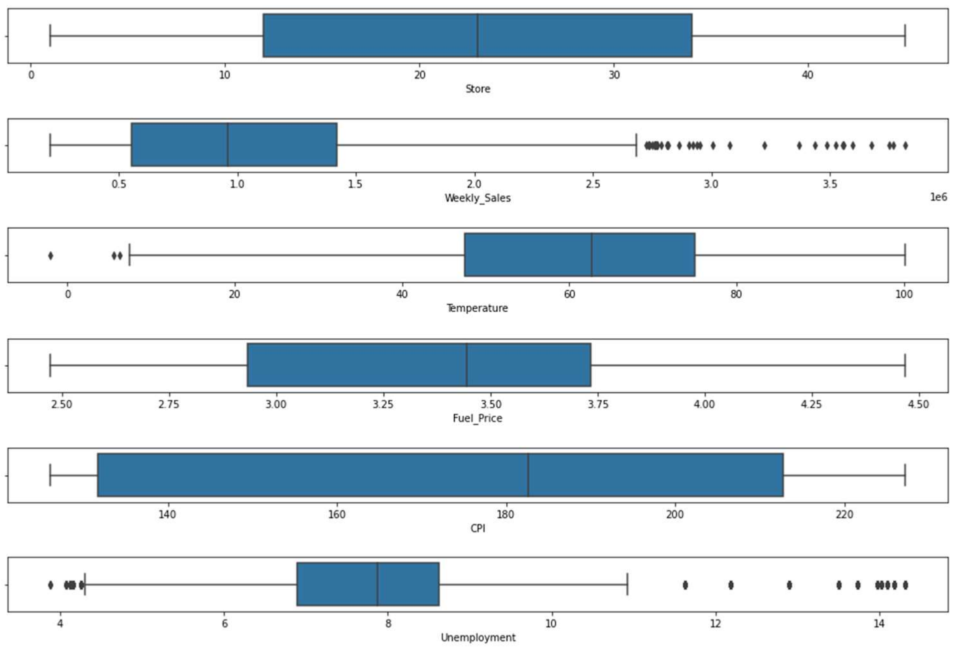
From the line graph we can see the following things:

1) The weekly sales are stable over a given period with a slightly declined trend, the weekly sales also have a seasonality rise every year. 2) The temperature shows a cyclic pattern or seasonality. 3) Fuel prices have an upward trend in them with some high variance. 4) CPI which is the consumer purchase Index has a steady upward trend. 5) Unemployment has a steady downward trend, which says that the unemployment rate is reducing.

4. Detecting and Removing Outliers.

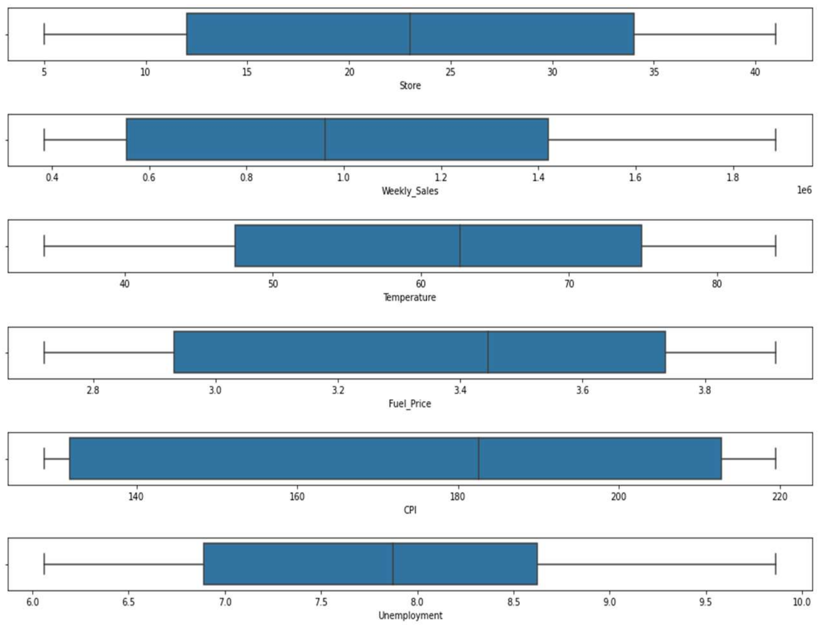
An Outlier is a data item that deviates significantly from the rest of the (so-called normal) objects. They can be caused by measurement or execution errors. The analysis for outlier detection is referred to as outlier mining. We have multiple methods to detect and remove outliers such as using scatter plots, Box Plot, using Z score, and using IQR interquartile.

In my work, I used a Box plot to detect the outliers through visualization, I used seaborn to plot a box plot.



From the above box plots, we can visually identify the outliers in the plots. The outliers are present in weekly_sales, Temperature, and Unemployment.

Now, Outliers should be removed for a better model, outliers can influence the model due to their extreme values. For removing Outliers by using the lower whisker and upper whisker, I also used visualization to check if the outliers were removed.



5. Correlation between Features.

Understanding the features that are involved in affecting sales. This can be done by understanding the correlation. To find which features are relevant to our model we have various methods like drop constant feature, using Pearson correlation, etc. For my project, I chose Pearson correlation and in this, we can use a single feature instead of multiple features that have a correlation of over 90%. This will help us to avoid overfitting the model.

The Pearson correlation coefficient model is calculated by following the formula.

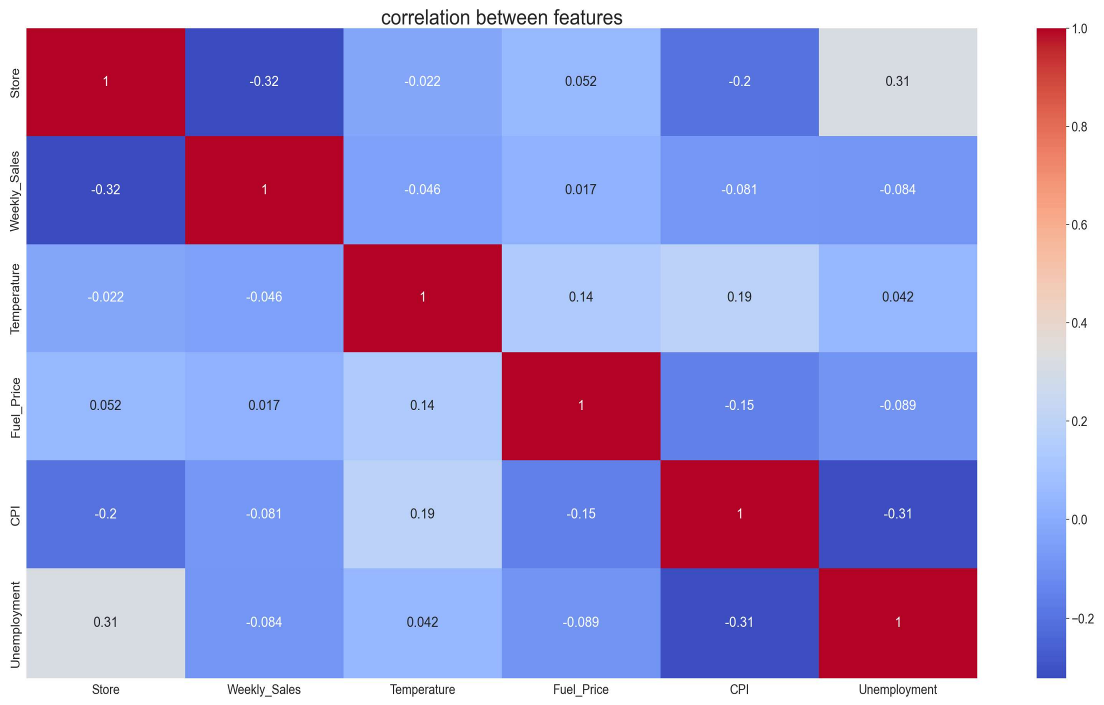
$$\begin{aligned}
\mu_X &= \mathbb{E}[\,X\,] \\
\mu_Y &= \mathbb{E}[\,Y\,] \\
\sigma_X^2 &= \mathbb{E}[\,(X-\mathbb{E}[X])^2\,] = \mathbb{E}[\,X^2\,] - (\mathbb{E}[\,X\,])^2 \\
\sigma_Y^2 &= \mathbb{E}[\,(Y-\mathbb{E}[Y])^2\,] = \mathbb{E}[\,Y^2\,] - (\,\mathbb{E}[\,Y\,])^2 \\
\mathbb{E}[\,(X-\mu_X)\,(Y-\mu_Y)\,] &= \mathbb{E}[\,(X-\mathbb{E}[\,X\,])\,(Y-\mathbb{E}[\,Y\,])\,] = \mathbb{E}[\,X\,Y\,] - \mathbb{E}[\,X\,]\,\mathbb{E}[\,Y\,],
\end{aligned}$$

the formula for ρ can also be written as

$$\rho_{X,Y} = \frac{\mathbb{E}[\,X\,Y\,] - \mathbb{E}[\,X\,]\,\mathbb{E}[\,Y\,]}{\sqrt{\mathbb{E}[\,X^2\,] - (\mathbb{E}[\,X\,])^2}\,\sqrt{\mathbb{E}[\,Y^2\,] - (\mathbb{E}[\,Y\,])^2}}.$$

Peason's correlation coefficient does not exist when either σ_X or σ_Y are zero, infinite or undefined.

In python, we can find correlation by using '.corr' function and I visualized it using the '.sns' heatmap.



From the heatmap, we can easily see that there is no feature having a correlation greater than 0.90 or 90%. So, we won't be eliminating any feature.

6. Standardization of feature Scaling.

We have two options when it comes to feature scaling first is normalization and the second is Standardization. I have used standardization in this the features will be transformed in such a way that it will have the properties of a standard normal distribution with mean=0 and standard deviation=1.

The formula used for a standardization (standard scalar) is,

$$z = \frac{x - \mu}{\sigma}$$

During feature scaling, I encountered an issue in which the 'Date 'column cannot be standardized (Dropping the date column is done because the date column cannot be in a float for standardization.) so I dropped the Date column to successfully run the standardization.

Before doing standardization, I separated the independent variables and dependent variables and stored them in a respective variable.

For standardization, I used the standard scalar function from the sklearn preprocessing library.

7. Splitting Data into train and test sets.

After Scaling the features, and separating the dependent and independent variables, the next step is to split the data into train and test sets.

A method for assessing a machine learning algorithm's performance is the train-test split. It can be applied to issues involving classification or regression as well as any supervised learning algorithm.

The process entails splitting the dataset into two subsets. The training dataset is the first subset, which is used to fit the model. The model is not trained using the second subset; rather, it is given the input element of the dataset, and its predictions are then made and contrasted with the expected values. The test dataset is the name given to this second dataset.

- **Train Dataset**: Used to fit the machine learning model.
- **Test Dataset**: Used to evaluate the fit machine learning model.

The objective is to estimate the performance of the machine learning model on new data: data not used to train the model. There is no optimal split ratio, for my model I used Train: 80%, and Test: 20% which is very commonly used as a standard split a lot of time.

The scikit-learn Python machine learning library provides an implementation of the train-test split evaluation procedure via the train_test_split() function. The function takes a loaded dataset as input and returns the dataset split into two subsets.

8. Using Machine Learning for Prediction Model.

There are multiple machine learning algorithms that can be used depending on the application and form of data.

Some types of machine learning tasks involve supervised learning, unsupervised learning, and reinforcement learning. Some of the commonly used ML algorithms are Linear regression, Logistics Regression, K-means, KNN, Random Forest, and Neural Networks.

After looking at the feature exploration and correlation result, I decided to use two models and compare their r-square score and mean squared error.

The r-square score is a statistical measure in a regression model that determines the proportion of variance in the dependent variable that can be explained by the independent variable.

$$R^2 = 1 - \frac{RSS}{TSS}$$

R^2	= coefficient of determination
RSS	= sum of squares of residuals
TSS	= total sum of squares

Formula

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

MSE	= mean squared error
n	= number of data points
Y_i	= observed values
$\hat{Y}_i$	= predicted values

Linear Regression algorithm

A statistical technique known as "linear regression" enables the analysis of relationships between two continuous (quantitative) variables. The independent variable is one variable, represented by the letter X.

$$\underset{\text{Dependent Variable}}{\underset{\uparrow}{Y_i}} = \overset{\text{Constant/Intercept}}{\overset{\downarrow}{\beta_0}} + \underset{\text{Slope/Coefficient}}{\underset{\uparrow}{\beta_1}} \overset{\text{Independent Variable}}{\overset{\downarrow}{X_i}}$$

Linear regression is carried out in python using ‘sklearn.linear’ model and LinearRegression function. We fit the linear regression model on the training dataset which in my case was 80% of the total dataset and then we use the ‘.predict’ function on test sets which is the remaining 20%.

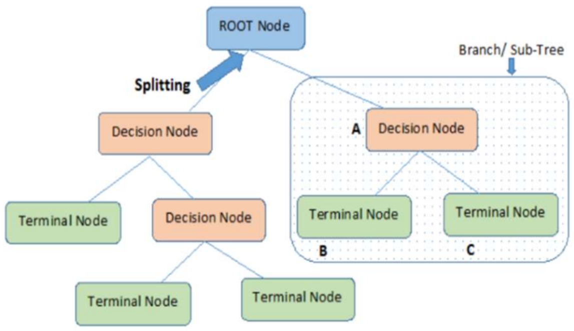
To check the R-square and mean squared error we use sklearn.metrics to import ‘r2_score and mean_squared_error’.

For Linear Regression r2_score = 0.125

For Linear Regression mean_squared_error = 231672732075.

Decision Tree Algorithm

Both classification and regression issues can be solved using decision trees. The term itself implies that it displays the predictions that come from a sequence of feature-based splits using a flowchart that resembles a tree structure. The decision is made by the leaves at the end, which follow the root node.



The formula for Entropy is shown below:

$$E(S) = -p_{(+)}\log p_{(+)} - p_{(-)}\log p_{(-)}$$

Here p_+ is the probability of positive class

p_- is the probability of negative class

S is the subset of the training example

Because decision trees are upside down, the root is at the top and is divided into multiple nodes. In simple words, decision trees are nothing more than a collection of if-else statements. It determines whether the condition is true, and if it is, it moves on to the next node associated with that choice.

Similarly to Linear regression, we can create a decision tree using sklearn.tree and importing decisionTreeRegressor.

For DecisionTree r2_score = 0.9280

For DecisionTree mean_squared_error = 19067548879

9. Future work-

In future work, I am planning to apply the GradientBoostingRegressor to the data set and check the R-square score and mean squared error for the predicted result.
I also plan to compare these three models, their R-square score, and mean squared error values.
I am also looking to identify the influence of various features on the dependent variable.