DATA ANALYTICS SEMESTER LONG ASSIGNMENT

Project Outline

Department of Computer Applications

National Instituteof Technology Kurukshetra

(2023-2024)

Submittedby:

Aman Bansal(52222206)

Mukul Chauhan(52212113)

Semester- 4th

Submitted to: Dr.

Kapil Gupta

Assistant Professor

Dataset description

Dataset: SBI stock Data From the Date of inspection (01-jan-1996) to the 19-apr-2024

Source: Yahoo! Finance

Context

The data is the price history and trading volumes of SBI stock in the index NIFTY 50 from NSE (National Stock Exchange) India. All datasets are at a day-level with pricing and trading values split across .cvs files for each stock along with a metadata file with some macro-information about the stocks itself. And data is from 01-jan-1996 to 19-apr-2024

Metadata

Date - Trade Data: Represents the date of the trading data, indicating when the stock market activity occurred.

Open - Opening Price for the Day: Represents the initial price at which a stock is traded on a given day.

High - Highest Price for the Day: Denotes the highest trading price reached by the stock during the trading day.

Low - Lowest Price for the Day: Represents the lowest trading price reached by the stock during the trading day.

Adj Close - * The adjusted closing price amends a stock's closing price to reflect that stock's value after accounting for any corporate actions. *The closing price is the raw price, which is just the cash value of the last transacted price before the market closes. *The adjusted closing price factors in corporate actions, such as stock splits, dividends, and rights offerings.

Close - Closing Price: Indicates the final trading price of the stock at the end of the trading day.

VWAP - Volume-Weighted Average Price: VWAP is a ratio of the cumulative share price to the cumulative volume traded over a given time period. It provides insight into the average price at which a stock is traded, weighted by the volume of trades.

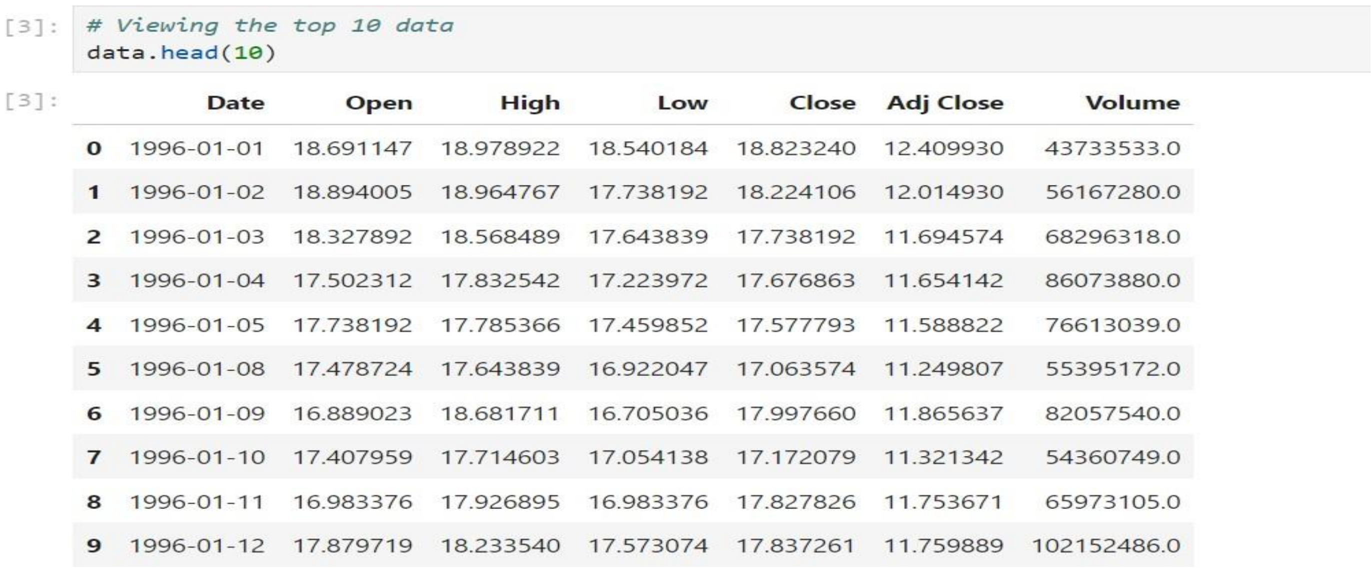
Volume - Volume Traded for the Day: Represents the total number of shares or contracts traded during a specific time period, typically a trading day.

Note: All the prices are denoted in Indian Rupees (INR), as mentioned at the end of the description. This dataset provides comprehensive information about the trading activity of various stocks, allowing users to analyze and understand the market trends and stock behavior.

The project aims to answer below questions:

- What are the data types?
- Are there missing values?
- Which independent variables are useful to predict a target (dependent variable)?
- Which independent variables have missing data? How much?
- What are the distributions of the predictor variables?
- Remove outliers and keep outliers (does if have an effect of the final predictive model)?

Starting 10 row of the database

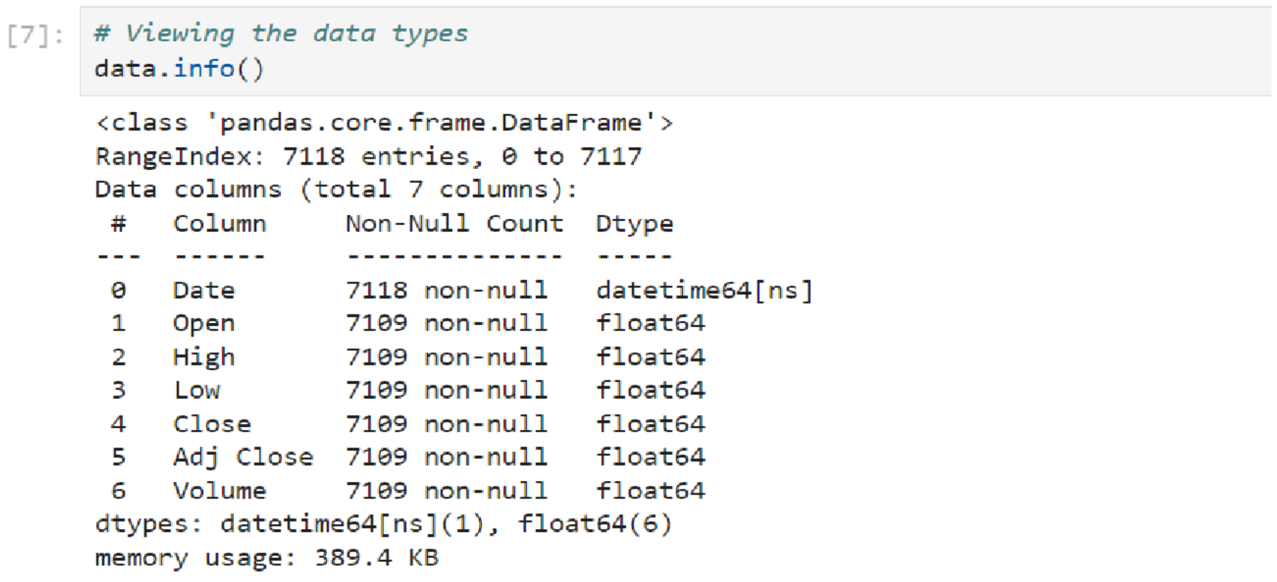


```
[3]: # Viewing the top 10 data
     data.head(10)
```

[3]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	1996-01-01	18.691147	18.978922	18.540184	18.823240	12.409930	43733533.0
1	1996-01-02	18.894005	18.964767	17.738192	18.224106	12.014930	56167280.0
2	1996-01-03	18.327892	18.568489	17.643839	17.738192	11.694574	68296318.0
3	1996-01-04	17.502312	17.832542	17.223972	17.676863	11.654142	86073880.0
4	1996-01-05	17.738192	17.785366	17.459852	17.577793	11.588822	76613039.0
5	1996-01-08	17.478724	17.643839	16.922047	17.063574	11.249807	55395172.0
6	1996-01-09	16.889023	18.681711	16.705036	17.997660	11.865637	82057540.0
7	1996-01-10	17.407959	17.714603	17.054138	17.172079	11.321342	54360749.0
8	1996-01-11	16.983376	17.926895	16.983376	17.827826	11.753671	65973105.0
9	1996-01-12	17.879719	18.233540	17.573074	17.837261	11.759889	102152486.0

1. What are the data types?



```
[7]: # Viewing the data types
     data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7118 entries, 0 to 7117
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       7118 non-null   datetime64[ns]
 1   Open       7109 non-null   float64
 2   High       7109 non-null   float64
 3   Low        7109 non-null   float64
 4   Close      7109 non-null   float64
 5   Adj Close  7109 non-null   float64
 6   Volume     7109 non-null   float64
dtypes: datetime64[ns](1), float64(6)
memory usage: 389.4 KB
```

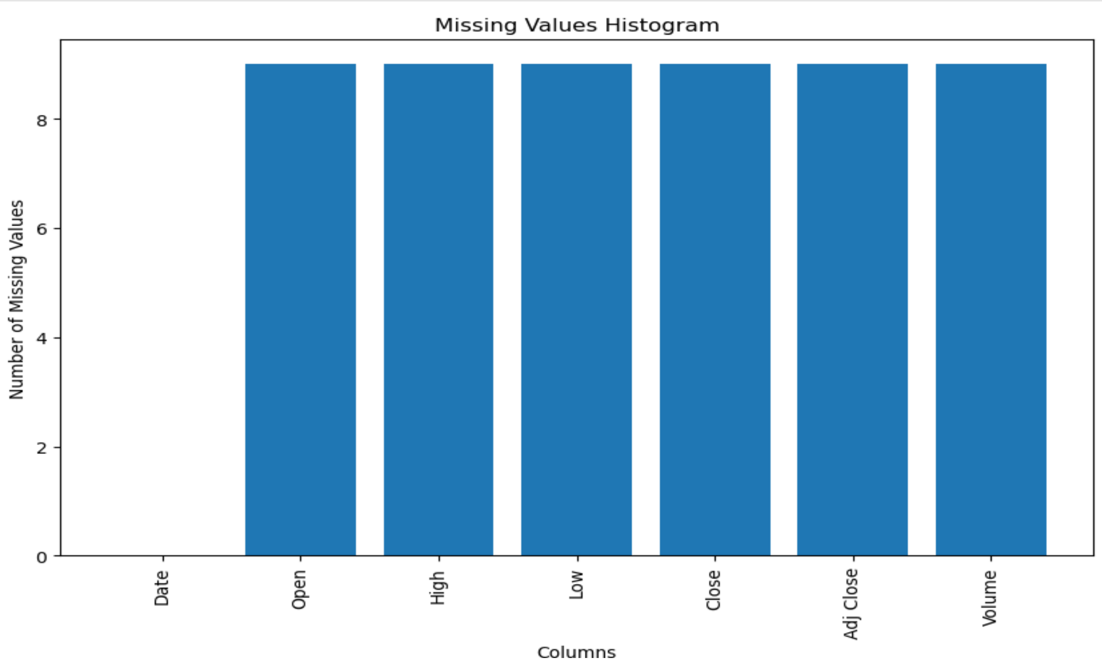
Conclusion derived

Dataset has ***1*** datetime data column, and ***6*** numerical data columns.

DareTime: Date

Numerical data columns: Open, High, Low, Adj Close, Close, Volume

#2. Are there missing values?

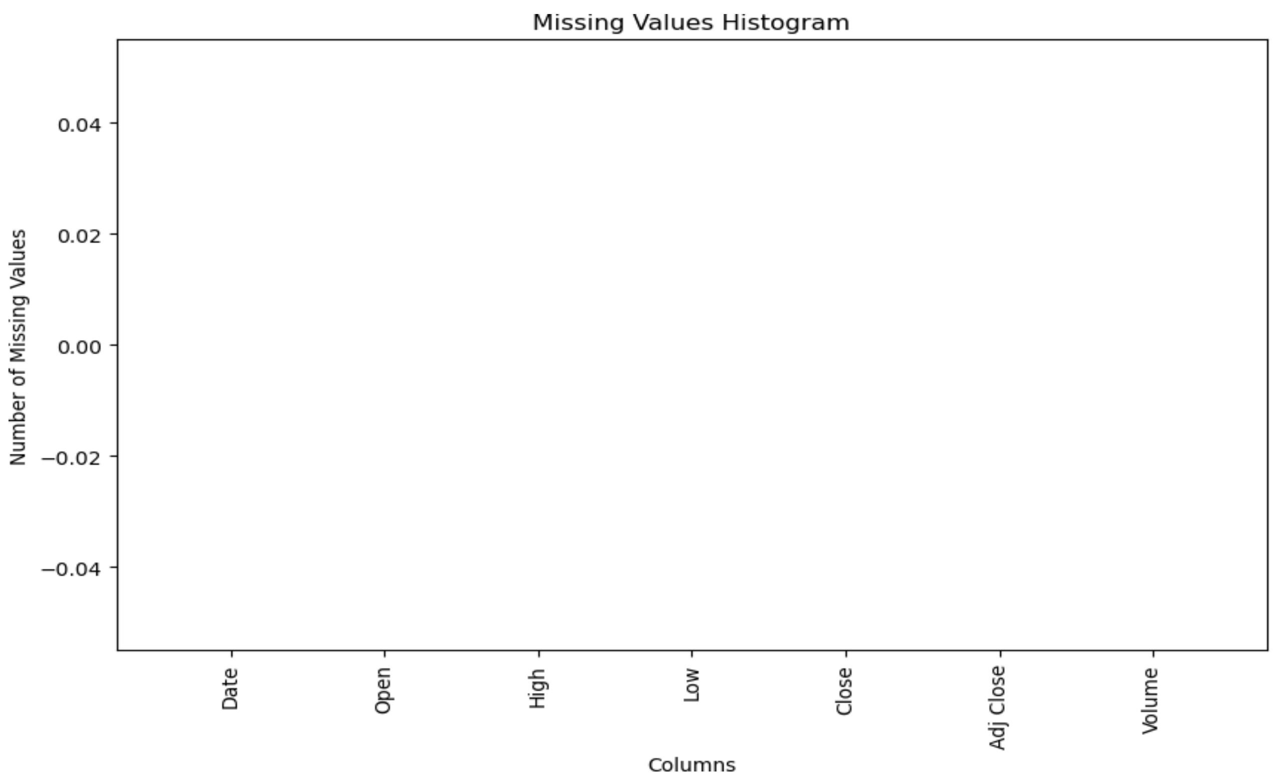


Conclusion derived: There are * missing values* in the dataset column open, high , low ,close Adj close ,volume and we are removing row with NULL values.

Reason for Deleting the NULL row:

There are only 9 row with NULL values which is less compare to full dataset (7100 rows) so on removing these lines there will be no effect on analysis

#Now our dataset has no NULL values in any column.



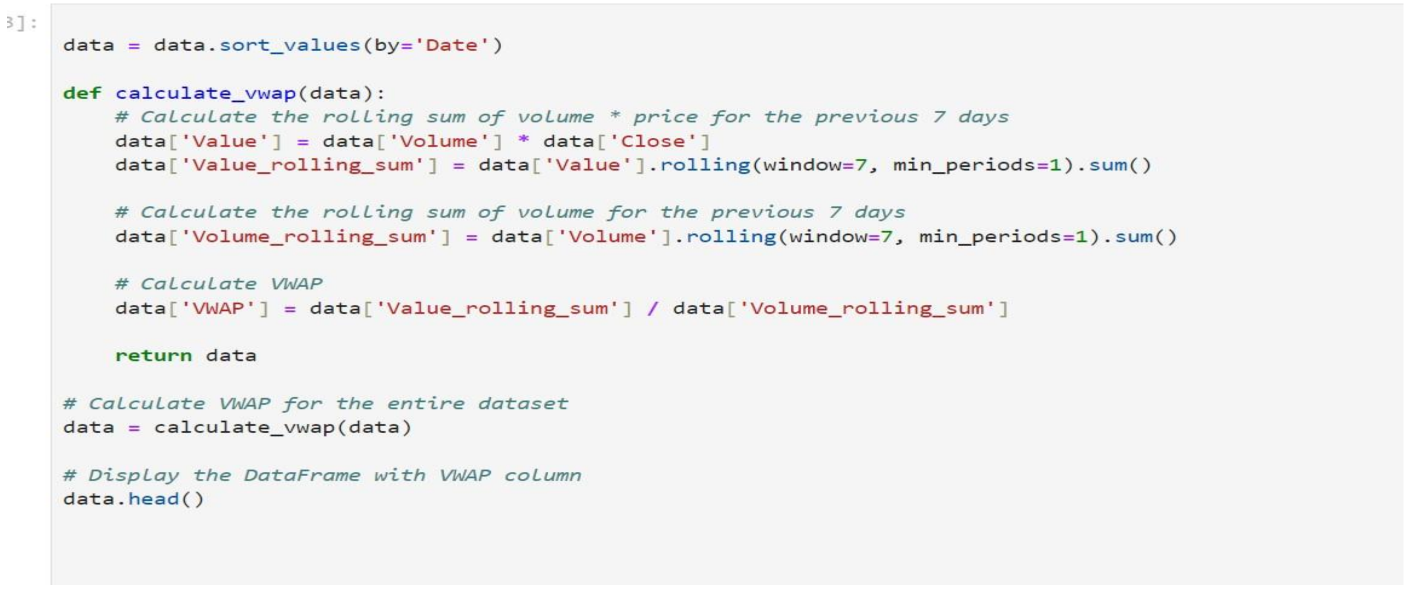
Rows that have null values:

[9]:

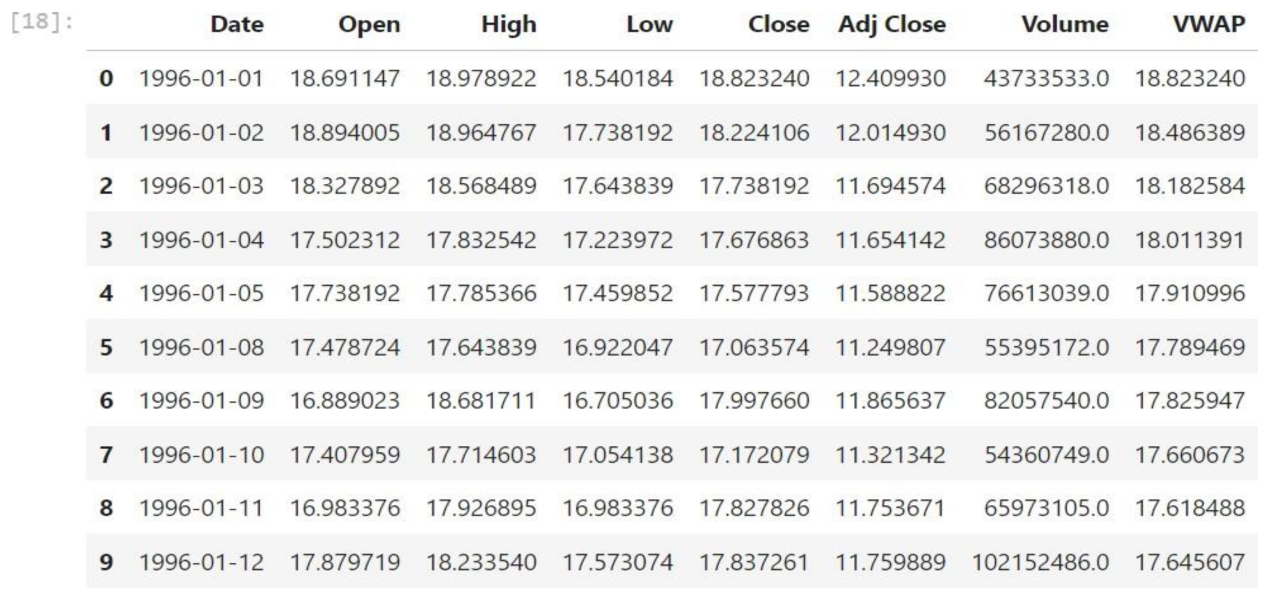
```
import pandas as pd
import numpy as np


# Find all rows with at least one missing value
rows_with_nulls = data[data.isnull().any(axis=1)]

print("Rows with at least one null value:")
print(rows_with_nulls)
```

```
Rows with at least one null value:
            Date  Open  High  Low  Close  Adj Close  Volume
2170  2004-04-26   NaN   NaN  NaN    NaN        NaN     NaN
2292  2004-10-13   NaN   NaN  NaN    NaN        NaN     NaN
3608  2010-02-06   NaN   NaN  NaN    NaN        NaN     NaN
4086  2012-01-07   NaN   NaN  NaN    NaN        NaN     NaN
4125  2012-03-03   NaN   NaN  NaN    NaN        NaN     NaN
4255  2012-09-08   NaN   NaN  NaN    NaN        NaN     NaN
4297  2012-11-11   NaN   NaN  NaN    NaN        NaN     NaN
4634  2014-03-22   NaN   NaN  NaN    NaN        NaN     NaN
4863  2015-02-28   NaN   NaN  NaN    NaN        NaN     NaN
```

# we are calculating The VWAP(volume weighted average price) from the last 7 days closing price and volume it is an indicator which helps in technical analysis and indicate the average price on a given time frame based on the average volume



```
3]:
data = data.sort_values(by='Date')

def calculate_vwap(data):
    # Calculate the rolling sum of volume * price for the previous 7 days
    data['Value'] = data['Volume'] * data['Close']
    data['Value_rolling_sum'] = data['Value'].rolling(window=7, min_periods=1).sum()

    # Calculate the rolling sum of volume for the previous 7 days
    data['Volume_rolling_sum'] = data['Volume'].rolling(window=7, min_periods=1).sum()

    # Calculate VWAP
    data['VWAP'] = data['Value_rolling_sum'] / data['Volume_rolling_sum']

    return data

# Calculate VWAP for the entire dataset
data = calculate_vwap(data)

# Display the DataFrame with VWAP column
data.head()
```



[18]:

	Date	Open	High	Low	Close	Adj Close	Volume	VWAP
0	1996-01-01	18.691147	18.978922	18.540184	18.823240	12.409930	43733533.0	18.823240
1	1996-01-02	18.894005	18.964767	17.738192	18.224106	12.014930	56167280.0	18.486389
2	1996-01-03	18.327892	18.568489	17.643839	17.738192	11.694574	68296318.0	18.182584
3	1996-01-04	17.502312	17.832542	17.223972	17.676863	11.654142	86073880.0	18.011391
4	1996-01-05	17.738192	17.785366	17.459852	17.577793	11.588822	76613039.0	17.910996
5	1996-01-08	17.478724	17.643839	16.922047	17.063574	11.249807	55395172.0	17.789469
6	1996-01-09	16.889023	18.681711	16.705036	17.997660	11.865637	82057540.0	17.825947
7	1996-01-10	17.407959	17.714603	17.054138	17.172079	11.321342	54360749.0	17.660673
8	1996-01-11	16.983376	17.926895	16.983376	17.827826	11.753671	65973105.0	17.618488
9	1996-01-12	17.879719	18.233540	17.573074	17.837261	11.759889	102152486.0	17.645607

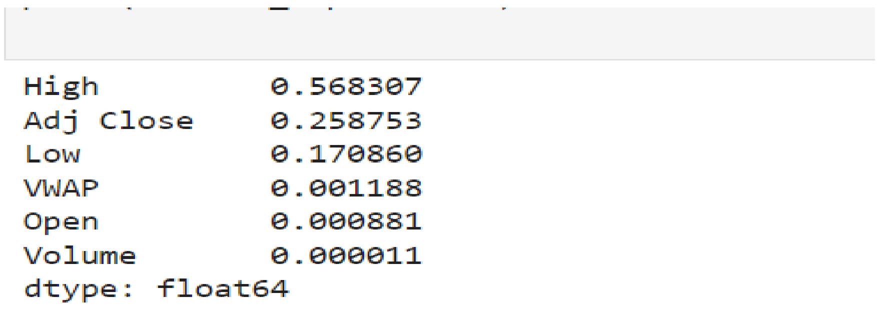
# Now dataset has one column VWAP which is calculated based on last 7 days

4 Which independent variables are useful to predict a target (dependent variable)?

Two methods to find the importance level of column to predict the target:

- **RandomForestRegresssor**
- **SelectKBest**

RandomForestRegresssor:



```
High          0.568307
Adj Close     0.258753
Low           0.170860
VWAP          0.001188
Open          0.000881
Volume        0.000011
dtype: float64
```

#This is score given by the Model according to the column contribution for predicting target

Key Takeaway:

- High is overwhelmingly the most influential feature in predicting 'Close.'
- VWAP has a minor but discernible impact on the model.
- Low, Open contribute significantly to the model.
- Volume provide moderate but less significant contributions.

SelectKBest:

For 2 selected features, selected indices: ['High', 'Low'], score: 4149173.4642471895

For 3 selected features, selected indices: ['Open', 'High', 'Low'], score: 4149173.4642471895

For 4 selected features, selected indices: ['Open', 'High', 'Low', 'Adj Close'], score: 5149173.4642471895

For 5 selected features, selected indices: ['Open', 'High', 'Low', 'Adj Close', 'Volume'], score: 4149173.4642471895

Best features: ['High', 'Low',open,Adj close,VWAP,]

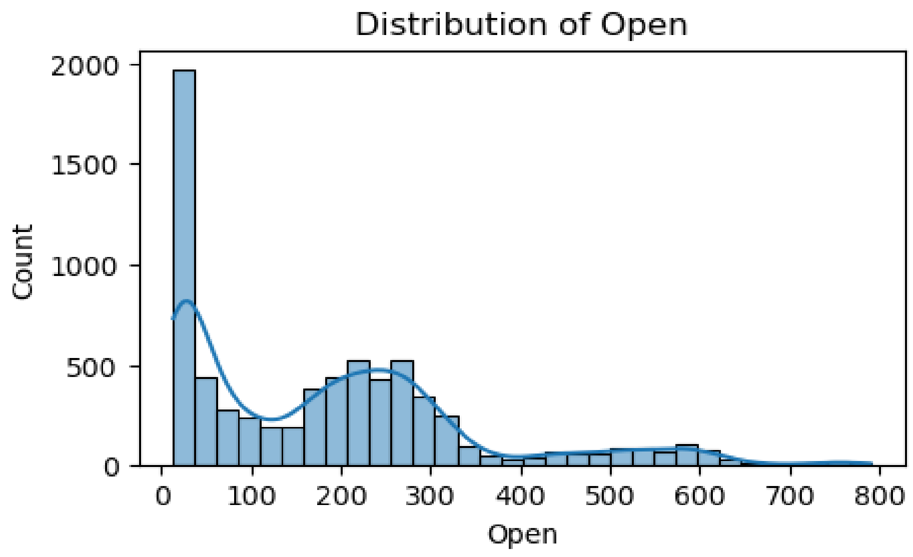
Best score: 5149173.4642471895

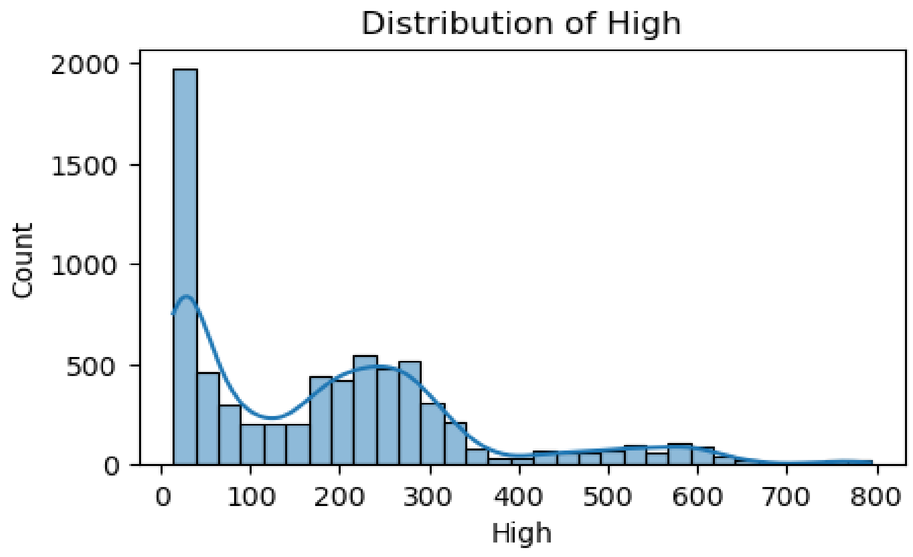
Conclusions derived:

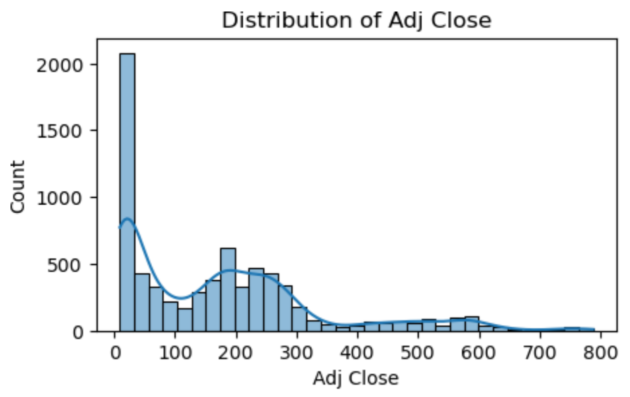
- Based on the above 2 feature selection methods we can observe that there are 6 features that are highly correlated to the target variable
- The rest of the features show slight correlation with the target variable.mu

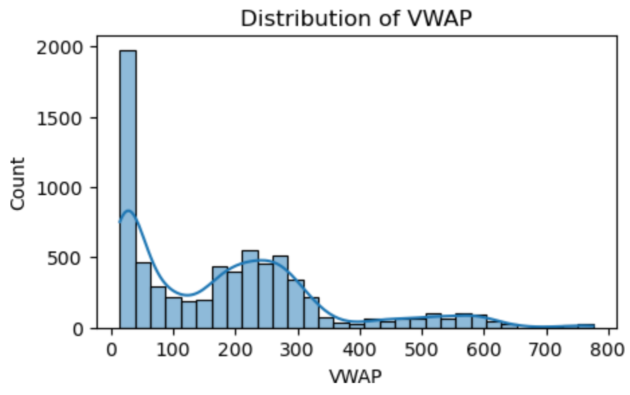
The selected features are: 'Open', 'High', 'Low', 'Adj close', 'VWAP'

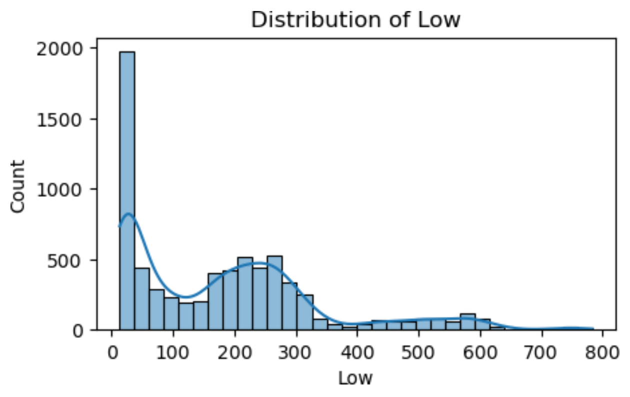
6. What are the distributions of the predictor variables?







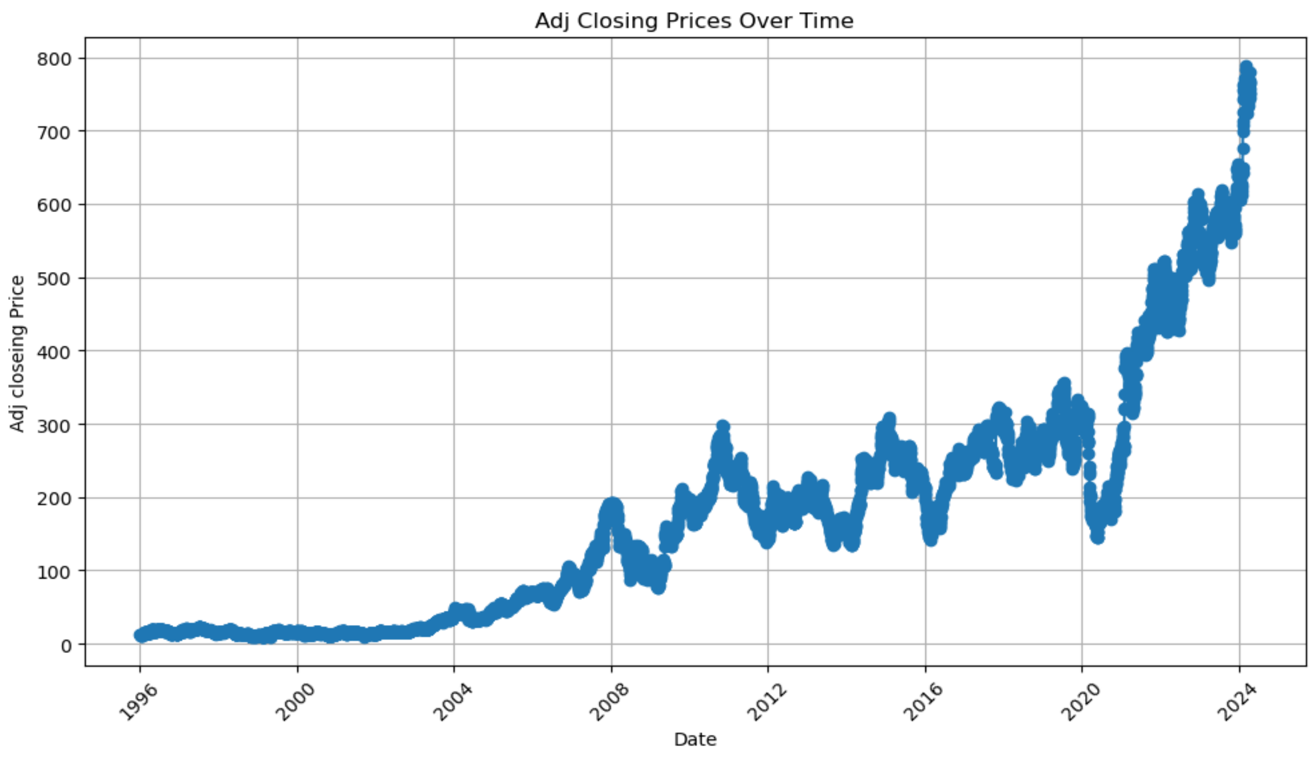




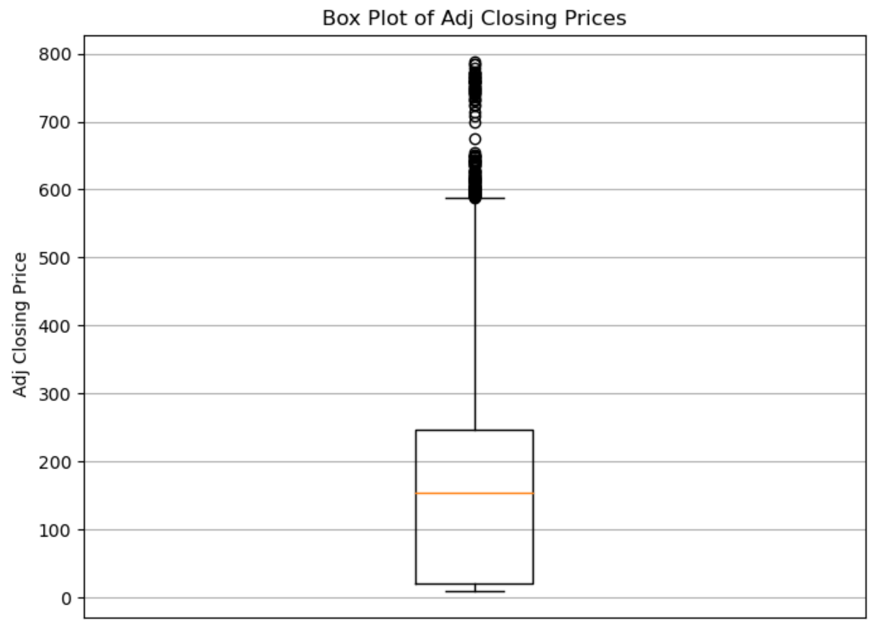
Conclusion derived

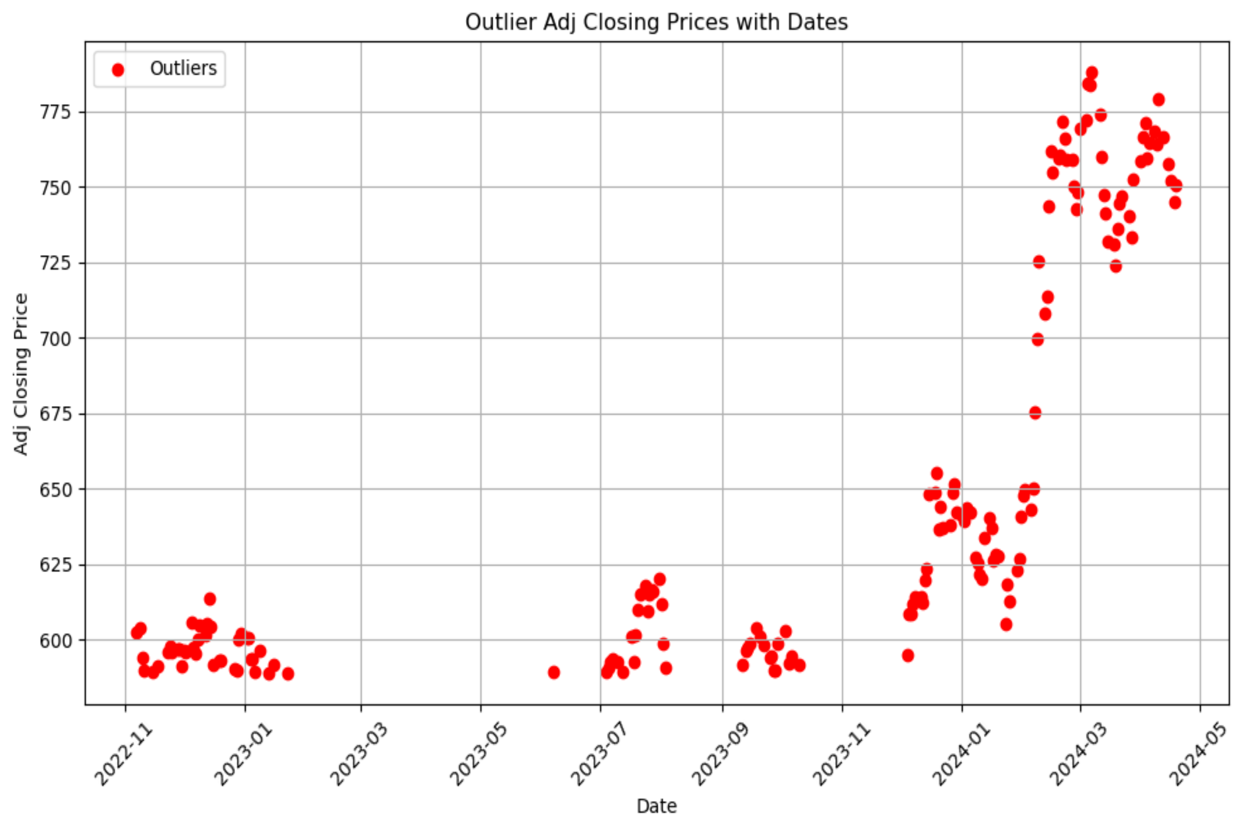
- The Distributions of Open, High, Low, Adj Close, VWAP, are extremely right skewed.
- This indicates that there are relatively fewer data points with higher values and majority of the data is concentrated towards the lower values. This also suggests the presence of outliers on the higher end of the data range.

# Ploting the Adj Close with respect to the date



#Ploting the box plot to analyze the outliers:





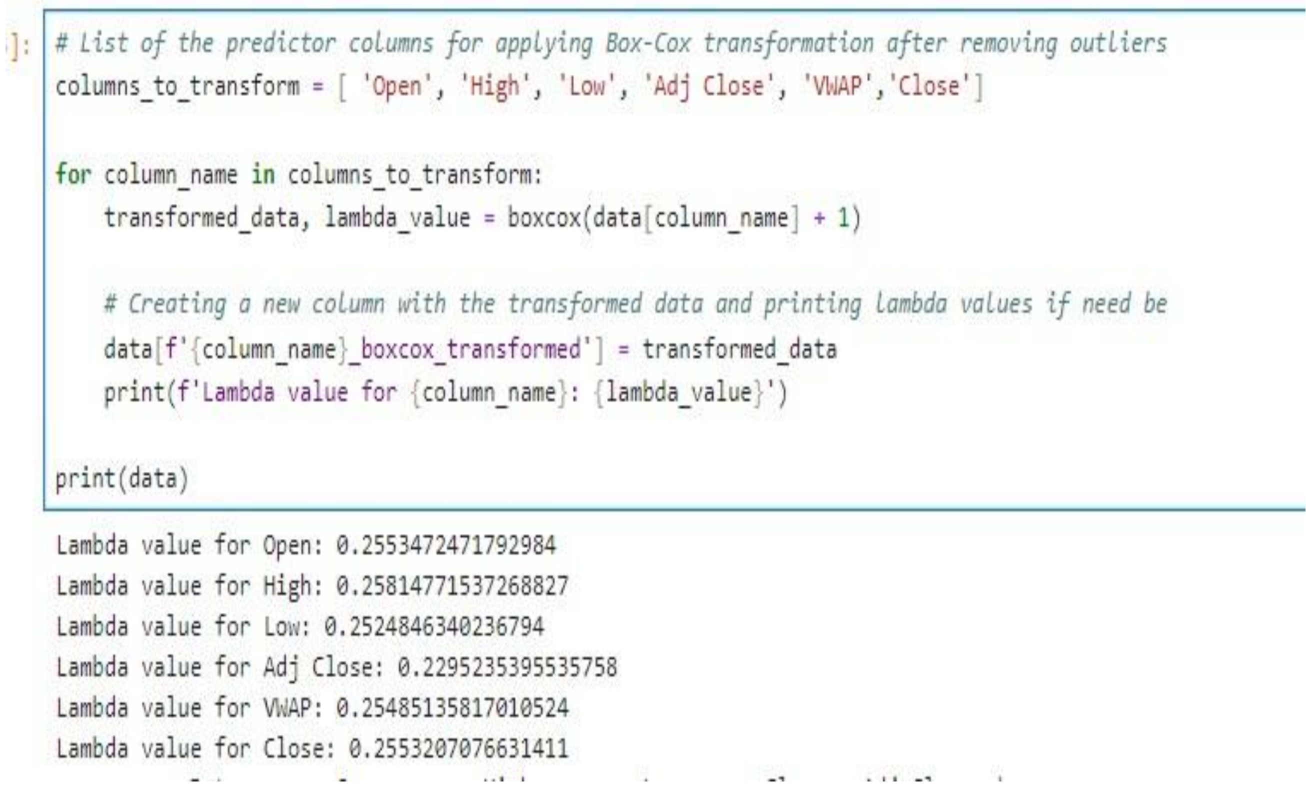
```
Outliers using Interquartile Range (IQR):
```

# Here points represents the outlier in the dataset

<u>Conclusion Derived:</u>

- **There are outlier in dataset first we will predict the target variable with outlier and the then without outlier**
- **We are using Box-Cox transformation to remove the outlier.**

-



```
# List of the predictor columns for applying Box-Cox transformation after removing outliers
columns_to_transform = [ 'Open', 'High', 'Low', 'Adj Close', 'VWAP','Close']

for column_name in columns_to_transform:
    transformed_data, lambda_value = boxcox(data[column_name] + 1)

    # Creating a new column with the transformed data and printing lambda values if need be
    data[f'{column_name}_boxcox_transformed'] = transformed_data
    print(f'Lambda value for {column_name}: {lambda_value}')

print(data)
```

```
Lambda value for Open: 0.2553472471792984
Lambda value for High: 0.25814771537268827
Lambda value for Low: 0.2524846340236794
Lambda value for Adj Close: 0.2295235395535758
Lambda value for VWAP: 0.25485135817010524
Lambda value for Close: 0.2553207076631411
```

The lambda values printed indicate how much each predictor column was transformed to achieve a more normal distribution. A lambda value of 0 would represent a log transformation, while values closer to 1 indicate less transformation. These values can provide insights into the nature of the transformation applied to each column.

Difference between box-cox and standard scaler :

Box-cox and standard Scaler are both techniques used for data transformation and normalization, but they serve different purpose and operate differently.

How it Works: Box-Cox applies a power transformation to the data. The transformation has a parameter, lambda (�λ), which controls the degree of transformation. Different lambda values correspond to different transformations:

Box-Cox Transformation

Constraints: Box-Cox can only be applied to strictly positive data. It requires finding the optimal lambda to best normalize the data.

Key Differences

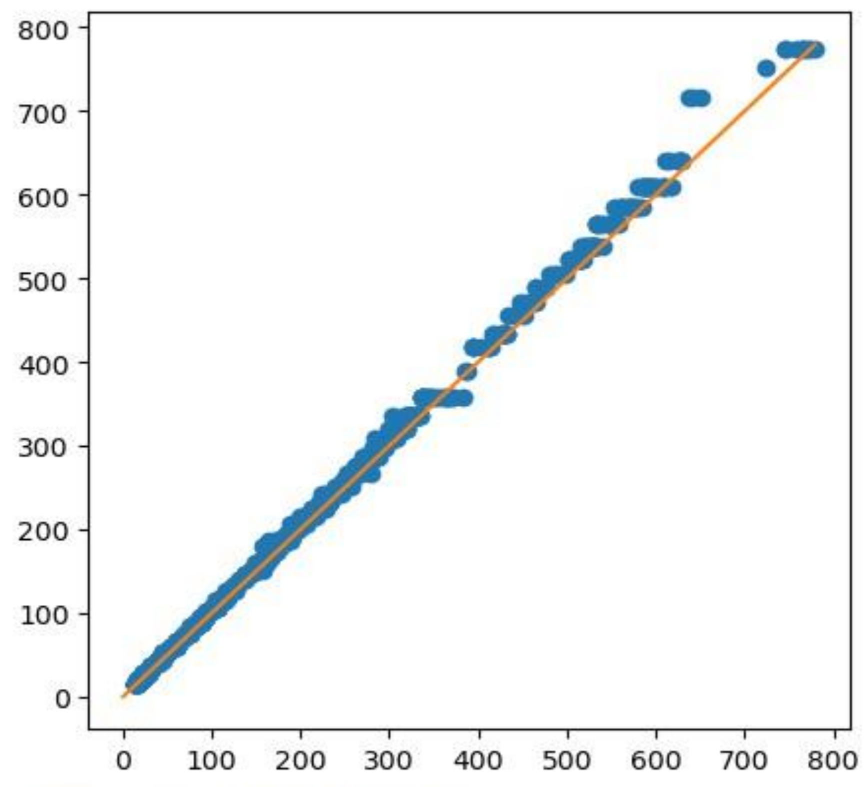
Goal: Box-Cox aims to normalize data, while Standard Scaler aims to standardize data.
Applicability: Box-Cox requires positive data and is typically used to correct skewness and make data more normally distributed. Standard Scaler can be used on any numerical data and doesn't change the
shape of the distribution, just the scale.
Usage in ML: Box-Cox is more common in statistical

contexts where normality is required, while Standard Scaler is widely used in machine learning to normalize feature scales.

7-Are there any differences in predicting the target variable with outlier and without outlier?

#With Outlier:

Since the dataset is divided in 80:20 proportions for training and testing respectively, the two sets have ***DIFFERENT*** data.



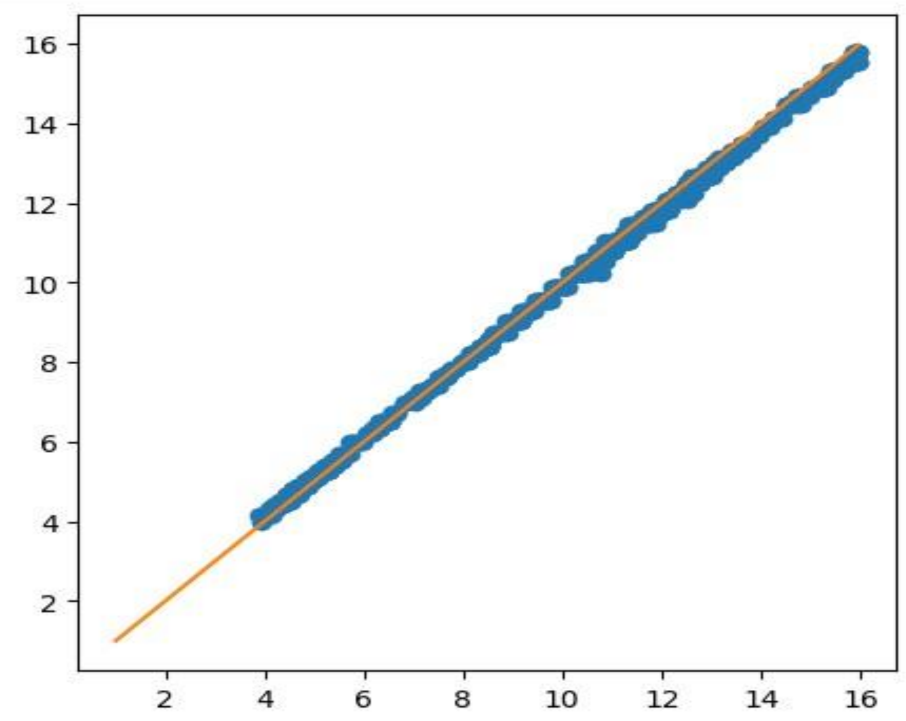
```
Train score:  0.9995241479406052
Test score:  0.9969188959602222
Regression score:  0.9969188959602222
Mean squared error:  76.59347087081709
Mean absolute error:  6.268757618293362
Root mean squared error:  8.75176958510775
```

Yellow line represents the predicted value

Conclusion derived:

The Decision Tree Regressor achieved exceptional performance on both training and test datasets, with R-squared values of approximately 99.95%. This indicates a strong ability to capture patterns in the data. The low Mean Squared Error (76.59), Mean Absolute Error (6.26), and Root Mean Squared Error (8.75) suggest accurate predictions with minimal deviation from actual values. While these results showcase the model's effectiveness, it's important to be mindful of potential overfitting, especially with a highly complex model.

#Without Outlier:



```
Train score:  0.9996646660757569
Test score:  0.9982381987392775
Regression score:  0.9982381987392775
Mean squared error:  0.023379411900499743
Mean absolute error:  0.12914806972222853
Root mean squared error:  0.1529032762909276
```

Yellow line represents the predicted value

Conclusion derived:

The Decision Tree Regressor demonstrated exceptional performance on both the training and test datasets, with R-squared values of approximately 99.94%. These values suggest that the model captures a significant portion of the variance in the target variable. The low Mean Squared Error (MSE) of .023, Mean Absolute Error (MAE) of 0.1291, and Root Mean Squared Error (RMSE) of .152 signify accurate predictions with minimal deviation from the actual values. These metrics indicate a close fit of the model to the data. *Conclusions derived:

The presence or removal of outliers has a noticeable impact on the final predictive model.

With Outliers:

Train score: 0.9995241479406052

Test score: 0.9969188959602222

Regression score: 0.9969188959602222

Mean squared error: 76.59347087081709

Mean absolute error: 6.268757618293362

Root mean squared error: 8.75176958510775

suggesting accurate predictions but with some deviation.

After Removing Outliers:

Train score: 0.9996646660757569

Test score: 0.9982381987392775

Regression score: 0.9982381987392775

Mean squared error: 0.023379411900499743

Mean absolute error: 0.12914806972222853

Root mean squared error: 0.1529032762909276

indicating a more accurate fit with reduced deviation.

The removal of outliers appears to enhance the model's predictive performance resulting in lower error metrics and potentially reducing the risk of overfitting. This underscores the importance of outlier analysis and removal in refining and improving the predictive capabilities of the model.

Train Score and Test Score

Train Score: This typically represents the model's performance on the training dataset. If you're using a regression model like Random Forest, this score is likely the R^2 (coefficient of determination), which measures the proportion of variance in the target variable that is predictable from the features. A score close to 1 indicates a very high level of predictability. However, an extremely high train score (like 0.999) can indicate overfitting, meaning the model may have memorized the training data rather than generalizing patterns.

Test Score: This reflects the model's performance on unseen test data. It is also likely the R^2 score. If the test score is significantly lower than the train score, it can indicate overfitting. In your case, a test score of 0.997 is quite high, suggesting a good level of generalization, though a small drop from the train score does indicate some degree of overfitting.

Regression Score

This score seems to be the same as the test score, suggesting it's an R^2 score. It indicates how well the model's predictions align with the actual values in the test dataset. The closer the score is to 1, the better the fit.

Mean Squared Error (MSE)

MSE: This measures the average squared difference between the actual values and the predicted values. The squaring aspect means that larger errors have a disproportionately large impact, which can be useful for detecting significant discrepancies. A smaller MSE is preferable, indicating more accurate predictions. An MSE of 76.593 suggests that on average, the squared error for predictions is about 76.593, indicating moderate error.

Mean Absolute Error (MAE)

MAE: This calculates the average absolute difference between actual values and predicted values. Unlike MSE, it does not square the differences, making it less sensitive to outliers. It represents the typical magnitude of prediction errors in the original unit of measure. An MAE of 6.269 means that, on average, the predictions are off by about 6.269 units.

Root Mean Squared Error (RMSE)

RMSE: This is the square root of the mean squared error. It brings the error back to the same unit as the original data, making it easier to interpret. An RMSE of 8.752 suggests that, on average, the predictions deviate from the true values by about 8.752 units. RMSE is useful because it is more sensitive to large errors compared to MAE.