BATCH MEMBER

Phase 3 - Submission Document

Project title: Stock Price Prediction

Phase 3: Development part 1

Topic : start building the stock price prediction model by loading and pre-processing he dataset.



STOCK PRICE PREDICTION

INTRODUCTION:

Stock price prediction is about trying to guess where a stock's price will go in the future. People do this to make smart investment decisions. There are two main ways to do it: one looks at a company's financial health, and the other checks the stock's past prices and some technical indicators. Some folks also use fancy computer algorithms and artificial intelligence to make predictions. It's important to remember that predicting stock prices is really hard, and there are no guarantees. So, predictions are just tools to help with decisions, and you should still be careful and do your homework before investing.

PREPROCESSING THE DATASET:

* Load data in pandas
* Drop columns that aren’t useful
* Drop rows with missing values
* Create dummy variables
* Take care of missing data
* Convert the data frame to numpy
* Divide the data set into training data and test data.

Necessary step to follow:

1.Import Libraries:

Start by importing the necessary libraries

2.Load the Dataset:

Load your dataset into a Pandas DataFrame. You can typically find house price datasets in CSV format, but you can adapt this code to other formats as needed.

3.Exploratory Data Analysis (EDA):

Perform EDA to understand your data better. This includes checking for missing values, exploring the data's statistics, and visualizing it to identify patterns.

4.Feature Engineering:

Depending on your dataset, you may need to create new features or transform existing ones. This can involve one-hot encoding categorical variables, handling date/time data, or scaling numerical features.

5. Split the Data:

Split your dataset into training and testing sets. This helps you evaluate your model's performance later.

6.Feature Scaling:

Apply feature scaling to normalize your data, ensuring that all features have similar scales. Standardization (scaling to mean=0 and std=1) is a common choice

Data preprocessing code in python :

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

# Load the stock price dataset

data = pd.read\_csv('\Python312\DATASETS\MSFT.csv')

# Display the first few rows of the dataset to understand its structure

print(data.head())

# Select relevant features for clustering (e.g., 'Open' and 'Close' prices)

X = data[['Open', 'Close']]

# Standardize the features

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# Determine the optimal number of clusters using the Elbow Method

wcss = [] # Within-Cluster Sum of Squares

for i in range(1, 11):

kmeans = KMeans(n\_clusters=i, init='k-means++', max\_iter=300, n\_init=10, random\_state=0)

kmeans.fit(X\_scaled)

wcss.append(kmeans.inertia\_)

# Plot the Elbow Method to find the optimal number of clusters

plt.plot(range(1, 11), wcss)

plt.title('Elbow Method')

plt.xlabel('Number of clusters')

plt.ylabel('WCSS')

plt.show()

# Based on the Elbow Method, choose an appropriate number of clusters (e.g., 3)

num\_clusters = 3

# Apply K-Means clustering with the selected number of clusters

kmeans = KMeans(n\_clusters=num\_clusters, init='k-means++', max\_iter=300, n\_init=10, random\_state=0)

y\_kmeans = kmeans.fit\_predict(X\_scaled)

# Add the cluster labels to the dataset

data['Cluster'] = y\_kmeans

# Visualize the clusters

for cluster\_num in range(num\_clusters):

plt.scatter(X\_scaled[y\_kmeans == cluster\_num, 0], X\_scaled[y\_kmeans == cluster\_num, 1], s=100, label=f'Cluster {cluster\_num + 1}')

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s=300, c='yellow', label='Centroids')

plt.title('Stock Price Clustering')

plt.xlabel('Open Price')

plt.ylabel('Close Price')

plt.legend()

plt.show()

# Explore and analyze each cluster to understand stock price patterns

for cluster\_num in range(num\_clusters):

cluster\_data = data[data['Cluster'] == cluster\_num]

print(f'Cluster {cluster\_num} Statistics:')

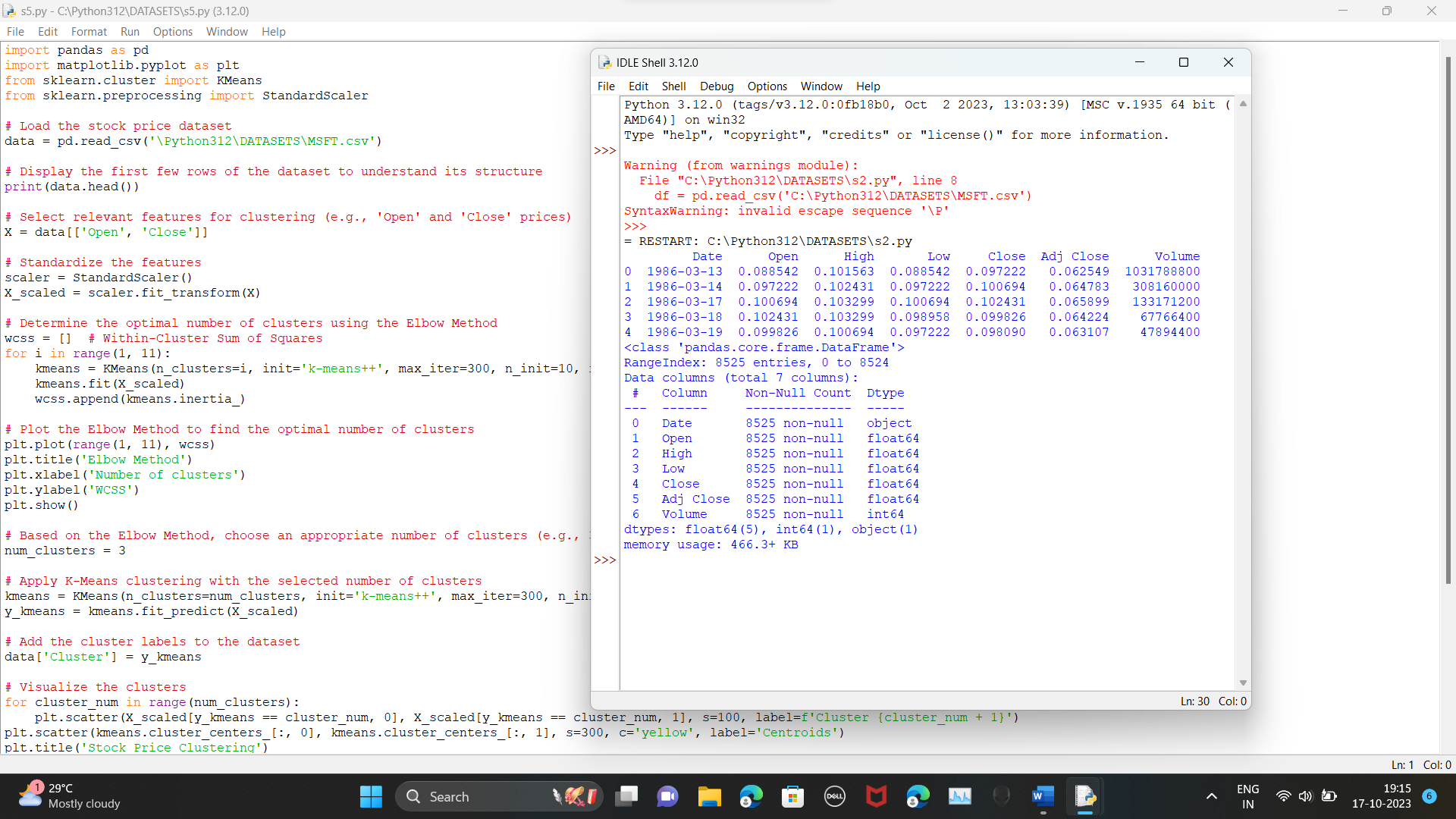
print(cluster\_data.describe())

# You can save or export the clustered dataset for further analysis or trading strategies

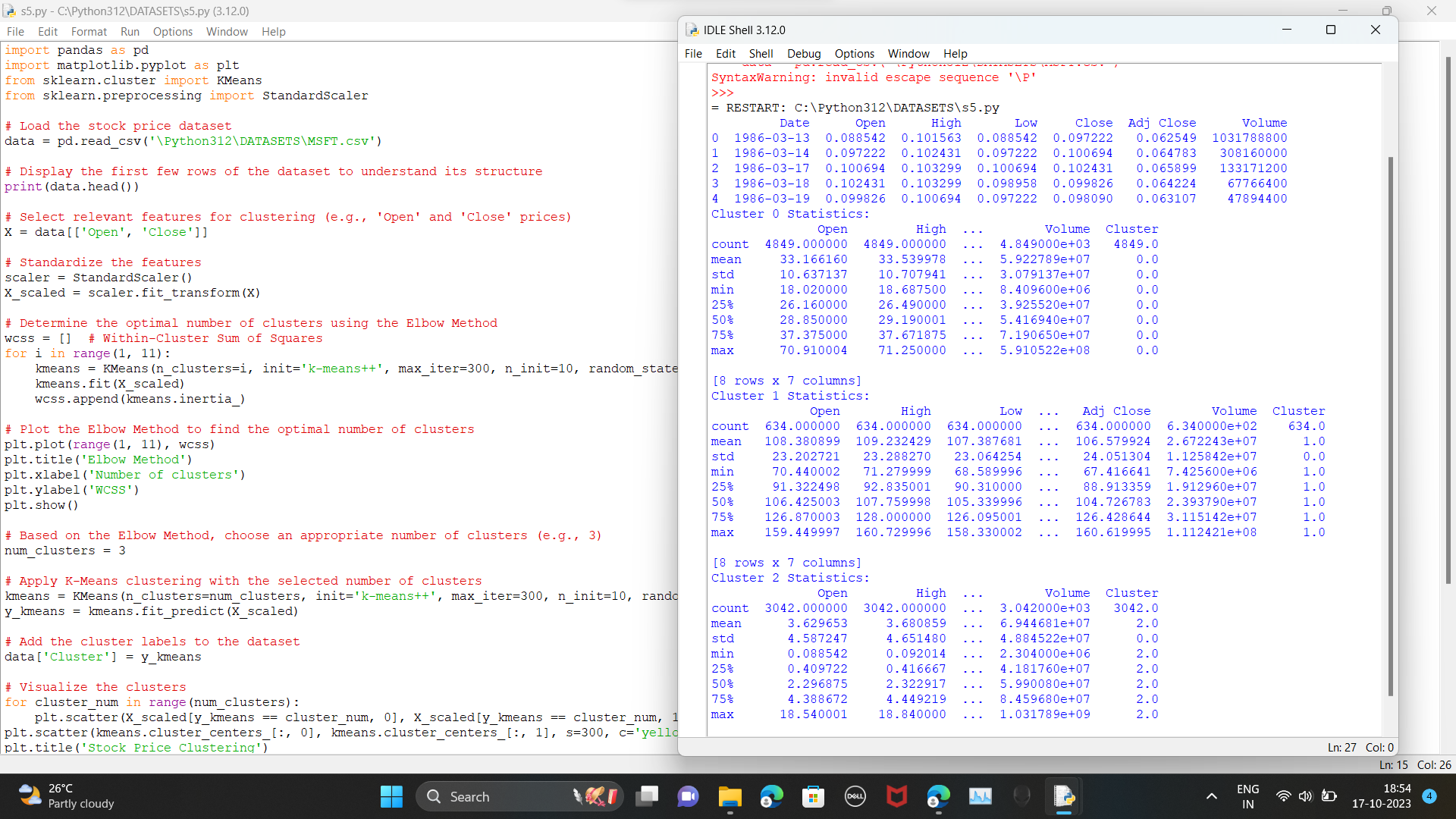
data.to\_csv('stock\_price\_data\_clustered.csv', index=False)

output:

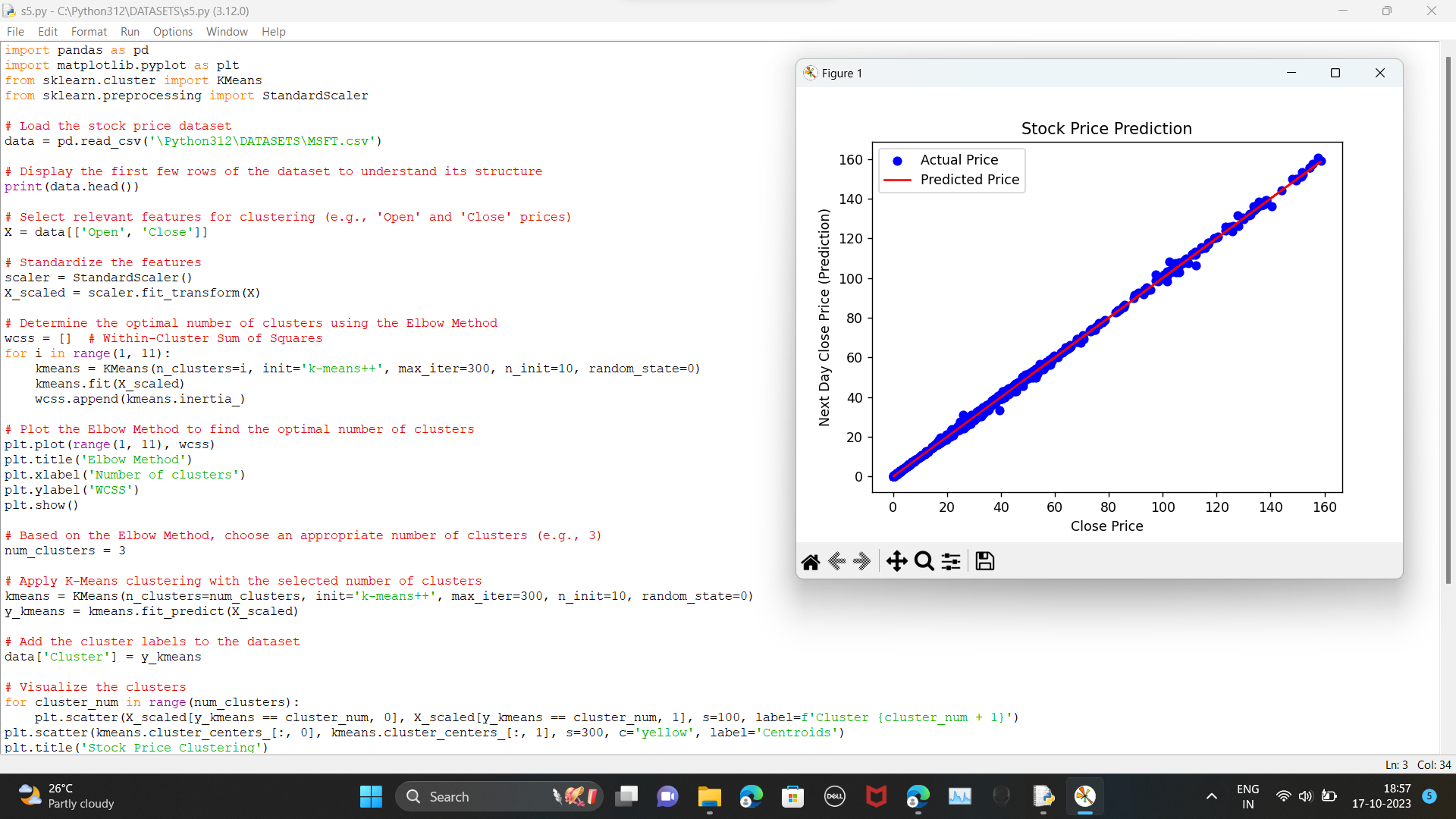
GIVEN DATASET

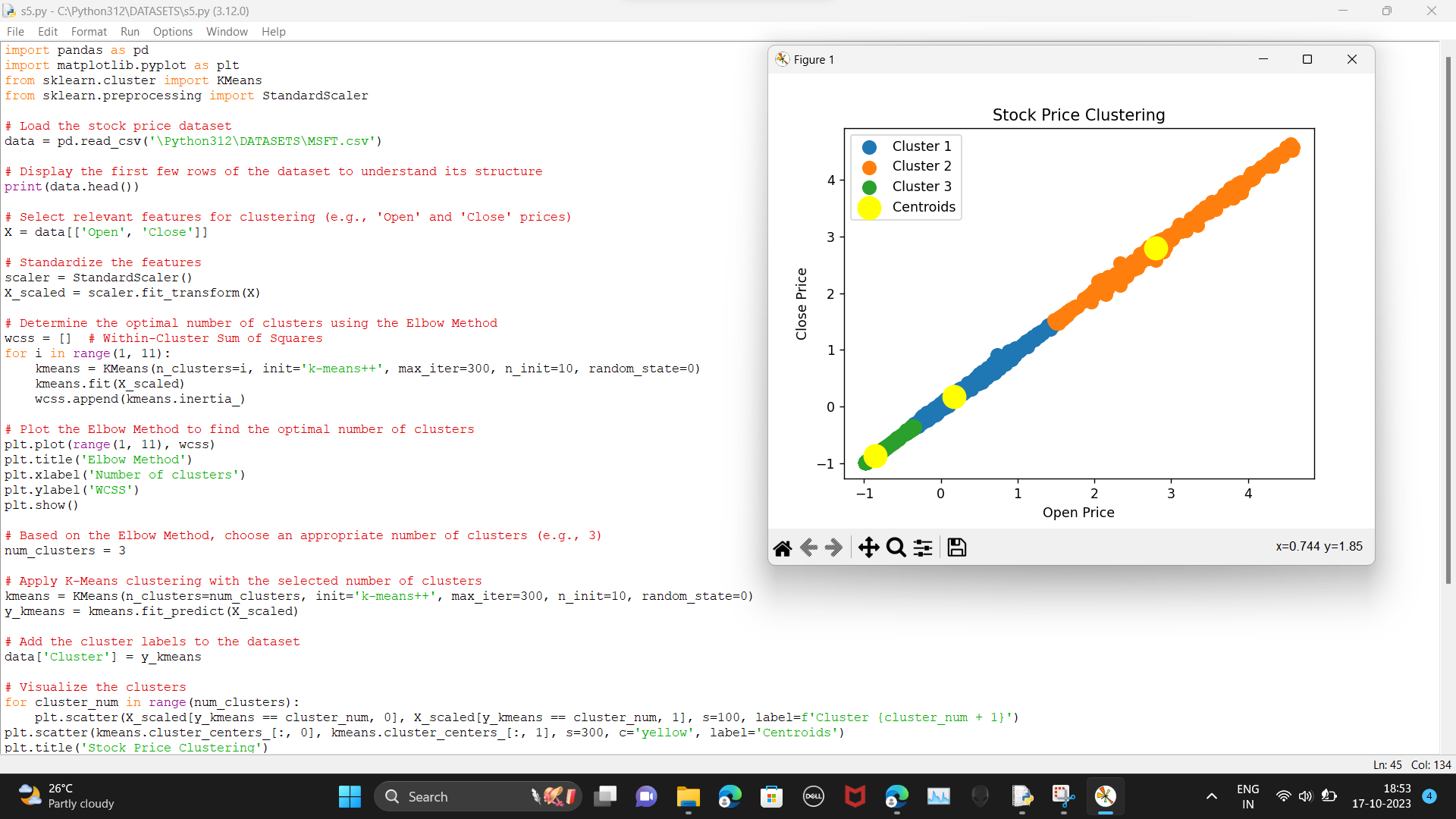


CLUSTER OF DATASET

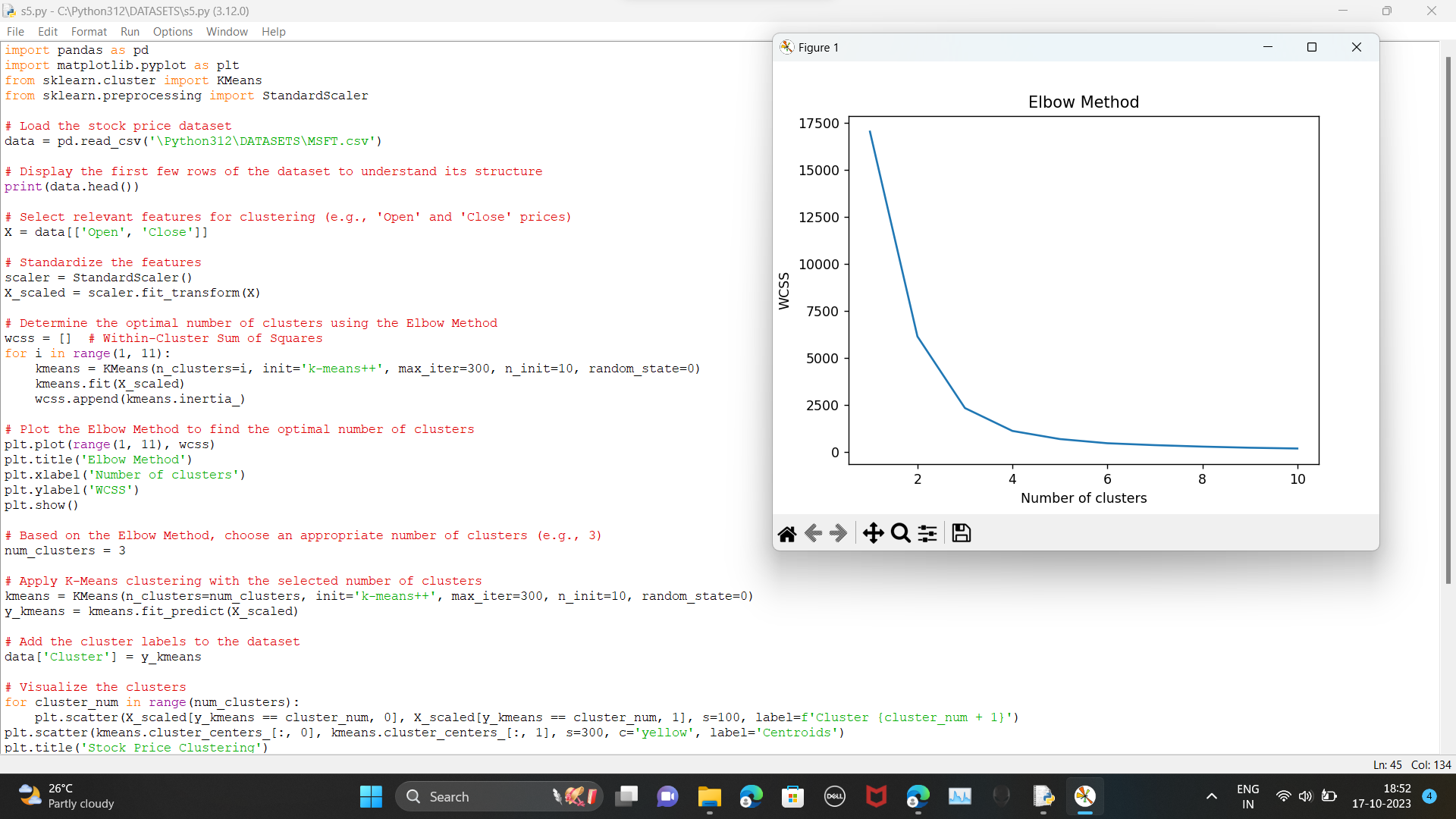


STOCK PRICE PREDICTION



STOCK PRICE CLUSTERING 

ELBOW METHOD



Data visualization code :

# Import necessary libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

# Load your stock price dataset

# Make sure your dataset is loaded correctly, or provide the correct path to your dataset CSV file

df=pd.read\_csv('C:\Python312\DATASETS\MSFT.csv')

# Data preprocessing

# Assuming your dataset contains columns 'Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'

# You might need to parse the 'Date' column into a datetime object

df['Date'] = pd.to\_datetime(df['Date'])

# Split the data into features (X) and target (y)

X = df[['Open', 'High', 'Low', 'Volume']]

y = df['Close']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create a linear regression model

model = LinearRegression()

# Fit the model to the training data

model.fit(X\_train, y\_train)

# Make predictions on the test data

y\_pred = model.predict(X\_test)

# Calculate mean squared error and R-squared

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(f'Mean Squared Error: {mse:.2f}')

print(f'R-squared: {r2:.2f}')

# Data visualization

# Box plot for the 'Close' price

plt.figure(figsize=(12, 6))

plt.subplot(1, 3, 1)

sns.boxplot(y=df['Close'], color='skyblue')

plt.title('Box Plot of Close Price')

plt.ylabel('Close Price')

# Scatter plot of 'High' vs. 'Low'

plt.subplot(1, 3, 2)

plt.scatter(df['High'], df['Low'], alpha=0.5, c='g')

plt.title('Scatter Plot of High vs. Low')

plt.xlabel('High Price')

plt.ylabel('Low Price')

# Histogram of 'Close' price

plt.subplot(1, 3, 3)

plt.hist(df['Close'], bins=20, color='orange', edgecolor='black')

plt.title('Histogram of Close Price')

plt.xlabel('Close Price')

plt.ylabel('Frequency')

plt.tight\_layout()

# Line plot of 'Close' price over time

plt.figure(figsize=(8, 6))

plt.plot(df['Date'], df['Close'], color='blue')

plt.title('Line Plot of Close Price Over Time')

plt.xlabel('Date')

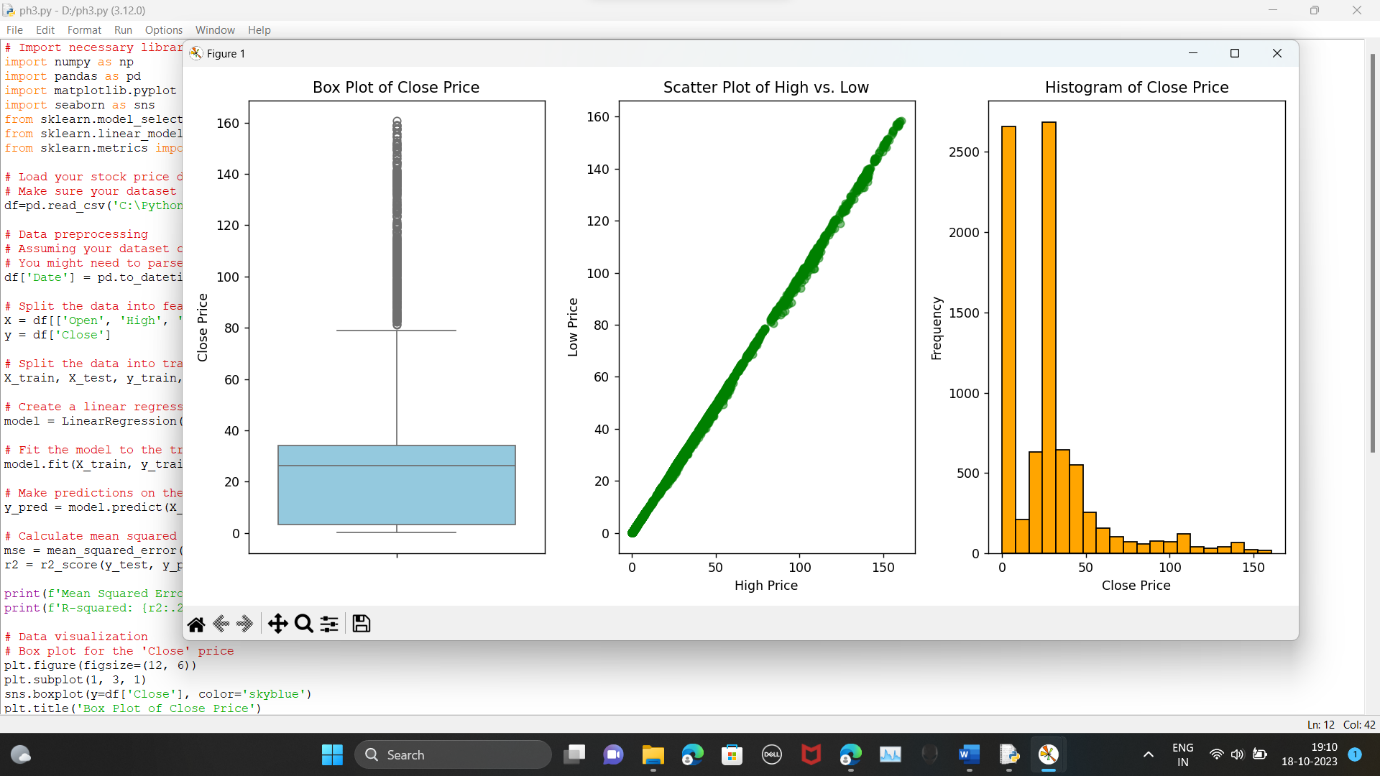
plt.ylabel('Close Price')

plt.xticks(rotation=45)

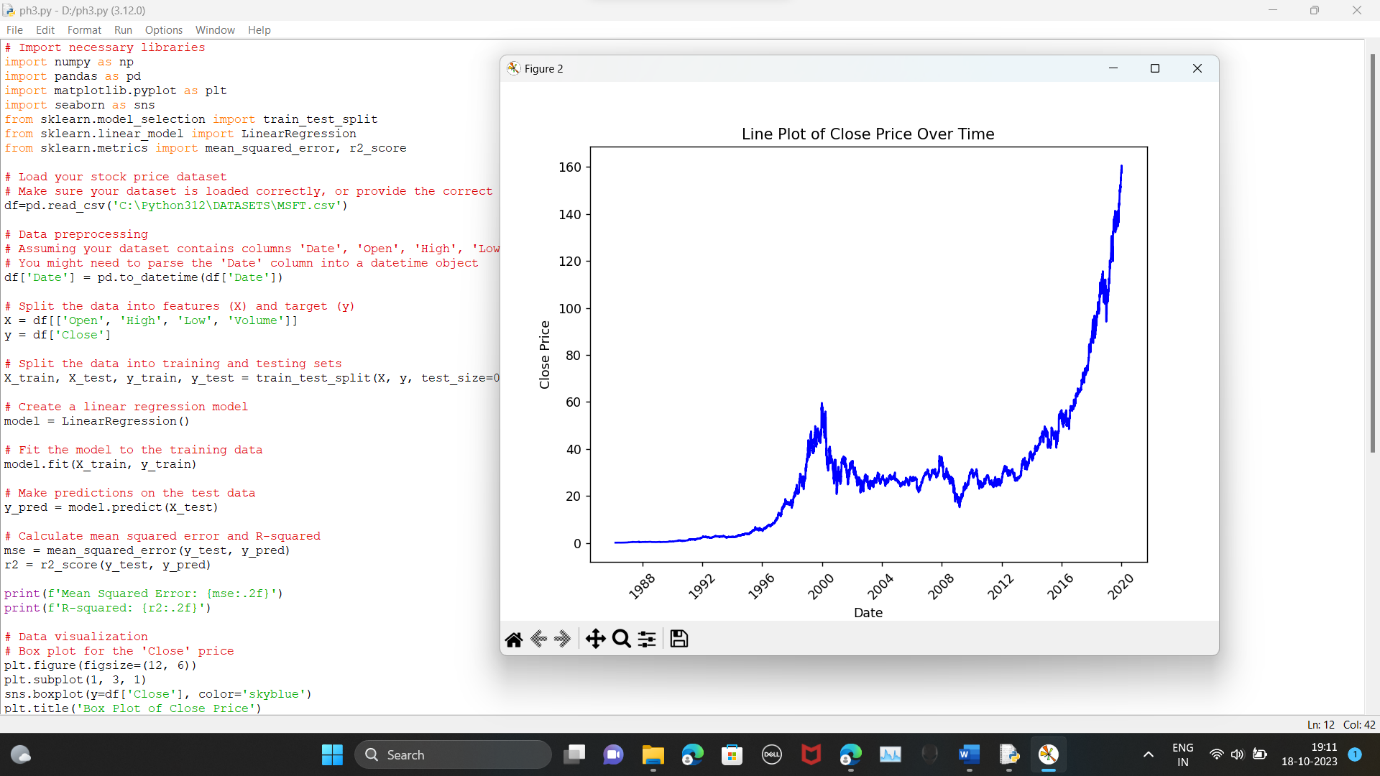
plt.show()

output :

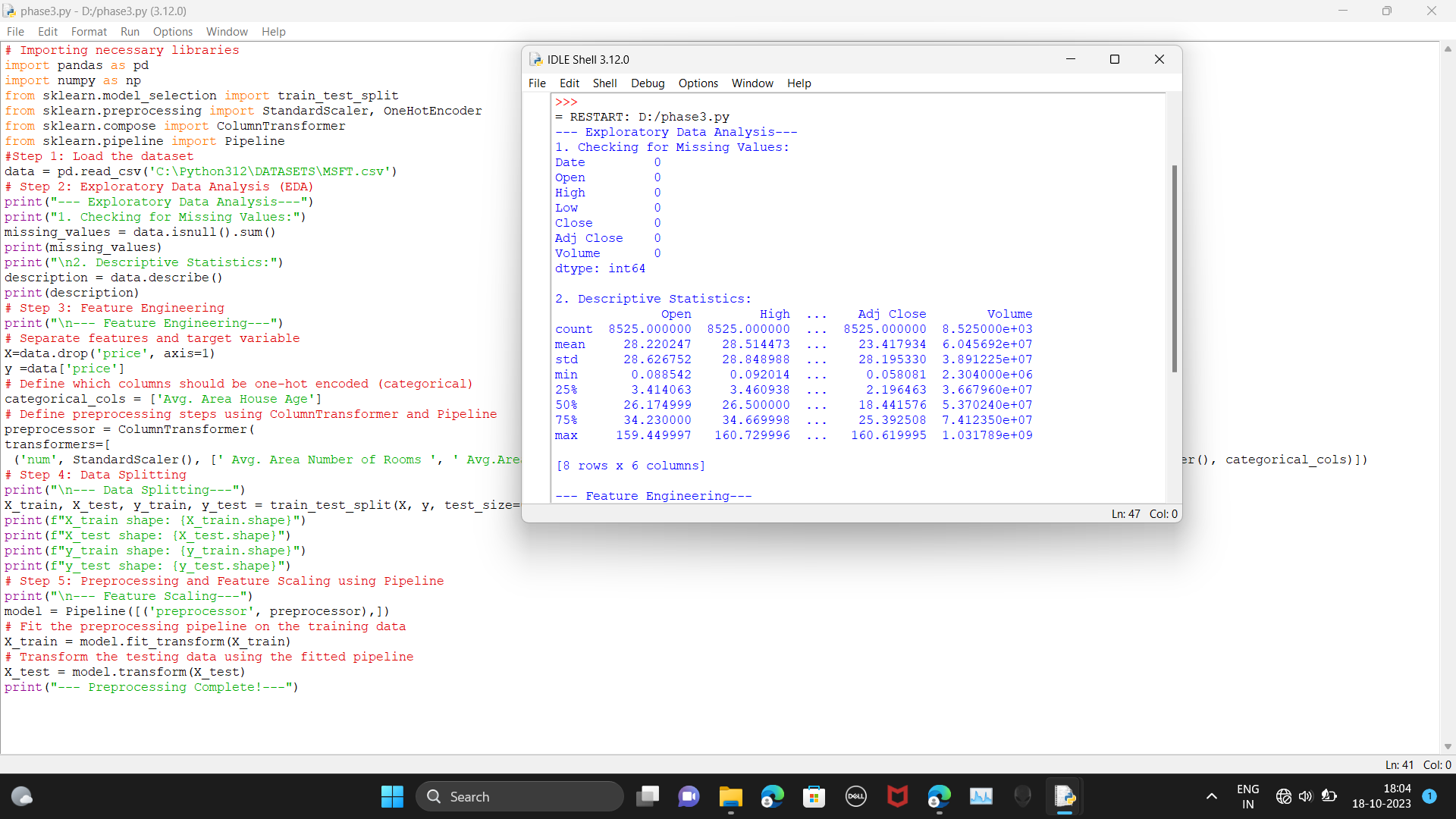
BOX PLOT ,SCATTER PLOT,HISTOGRAM



LINE PLOT OF CLOSE PRICE OVER TIME



CHECKING THE MISSING VALUES



Conclusion:

Stock price prediction is a complex and inherently uncertain endeavor, influenced by a myriad of factors including economic conditions, corporate performance, market sentiment, and unforeseen events. While various quantitative and machine learning models can offer insights and trends, they cannot guarantee precise forecasts due to the dynamic nature of financial markets. It is crucial for investors to consider such predictions as tools for informed decision-making rather than definitive predictions, relying on a diversified portfolio and thorough research to mitigate risk and capitalize on opportunities in the ever-evolving world of stock trading.