# **Importing modules**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error,mean\_absolute\_error

from sklearn.ensemble import RandomForestRegressorv

# **Reading the Dataset**

df=pd.read\_csv("D:\\miniproject\\stocks.csv")

df

**output:**



# **Dropping the unwanted column**

df.drop(['Ticker','Adj Close'],axis=1,inplace=True)

df

**output:**



# **Data checks to perform**

**#To check first five columns**

df.head()

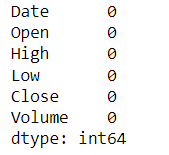
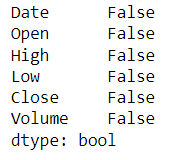
**#To check last five columns**

df.tail()

**#checking missing values**

df.isnull().sum() **or** df.isnull().any()

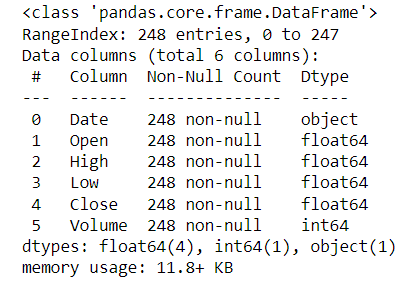
**output:**

**or **

**#checking information about dataset**

df.info()

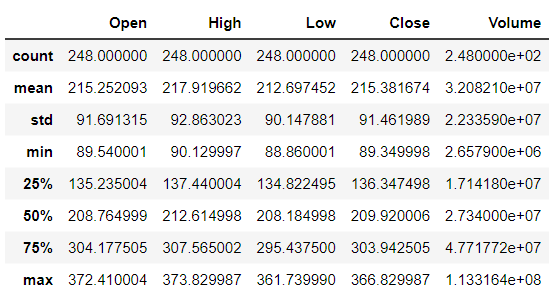
**output:**



**#check statistics of dataset**

df.describe()

**output:**

****

print('length of dataset:',len(df))

print('shape of the dataset',df.shape)

print("Number of columns in the dataset:",df.columns)

**output:**

lenghth of dataset: 248

shape of the dataset (248, 6)

Number of columns in the dataset: Index(['Date', 'Open', 'High', 'Low', 'Close', 'Volume']

# **Explorartory data analysis**

**#**

df['Open'].plot(figsize=(16,6))

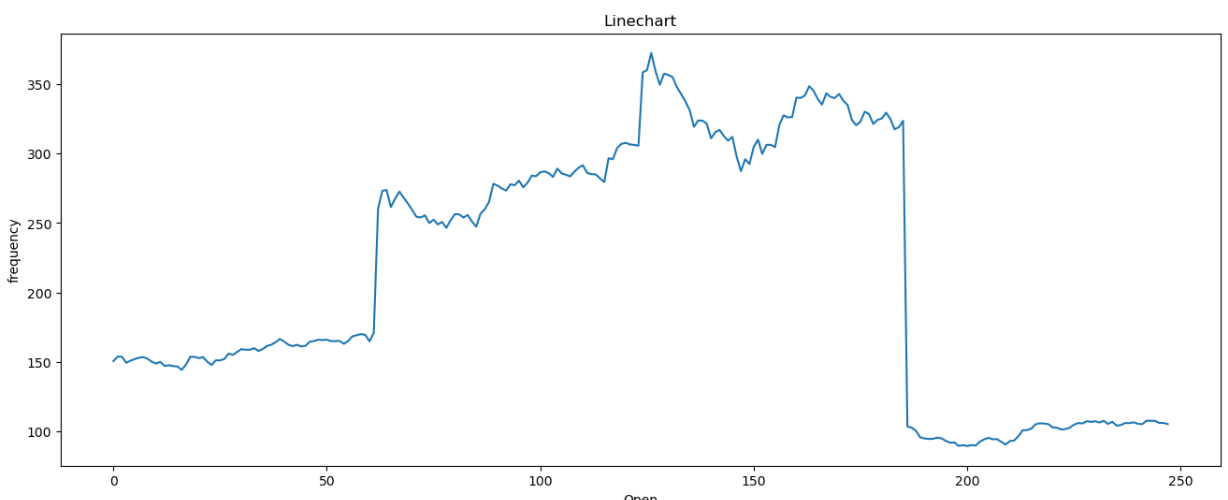
plt.title('Linechart')

plt.xlabel('Open')

plt.ylabel('frequency')

plt.show()

**output:**

****

**Summary:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# **Data preprocessing**

**#splitting the date column into year,month,day**

x=df

x['year']=pd.DatetimeIndex(x['Date']).year

x['month']=pd.DatetimeIndex(x['Date']).month

x['day']=pd.DatetimeIndex(x['Date']).day

x

**output:**

****

# **Training the model for MultipleLinearRegression**

**#splitting the dataset**

x=df.drop(['Close','Date'],axis=1)

y=df['Close']

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,random\_state=0)

print('xtrain:',x\_train.shape)

print('xtest:',x\_test.shape)

print('ytain:',y\_train.shape)

print('ytest:',y\_test.shape)

**output:**

xtrain: (186, 7)

xtest: (62, 7)

ytain: (186,)

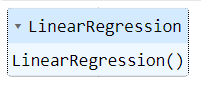
ytest: (62,)

**#model training**

regressor=LinearRegression()

regressor.fit(x\_train,y\_train)

**output:**

****

**#To check intercept and co-efficient**

print('intercept:',regressor.intercept\_)

print('Co-efficient:',regressor.coef\_)

**output:**

intercept: -0.6115052912800536

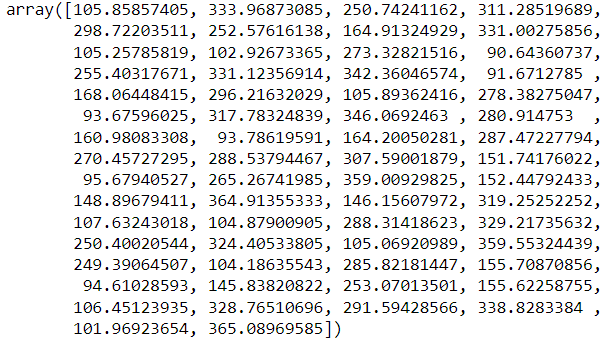
Co-efficient: [-5.45434461e-01 7.34498290e-01 8.12746520e-01 4.09319930e-096.34908792e-16 1.03687704e-01 1.15818167e-03]

**#Predictions**

y\_pred=regressor.predict(x\_test)

y\_pred

**output:**

****

**#Evaluating the model**

train\_accuracy=regressor.score(x\_train,y\_train)

print('train\_accuracy(R\_Squared):',train\_accuracy)

test\_accuracy=regressor.score(x\_test,y\_test)

print('test\_accuracy(R\_Squared):',test\_accuracy)

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:',math.sqrt(metrics.mean\_squared\_error(y\_test,y\_pred)))

**output:**

train\_accuracy(R\_Squared): 0.9996274457101879

test\_accuracy(R\_Squared): 0.9996351890659423

Mean Absolute Error: 1.3038307748950013

Mean Squared Error: 3.3155240230458447

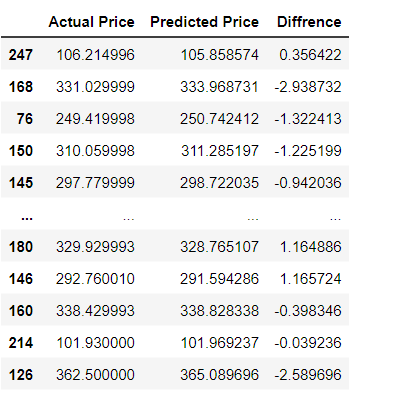
Root Mean Squared Error: 1.8208580458250567

**#To check actual price ,predicted price and difffrence**

dfr=pd.DataFrame({'Actual Price':y\_test,'Predicted Pric:y\_pred,'Diffrence':y\_test-y\_pred})

dfr

**output:**

****

**#Plotting the bar graph to check difference between actual price and predicted price**

graph=dfr.head(20)

graph.plot(kind='bar')

plt.title('BarGraph')

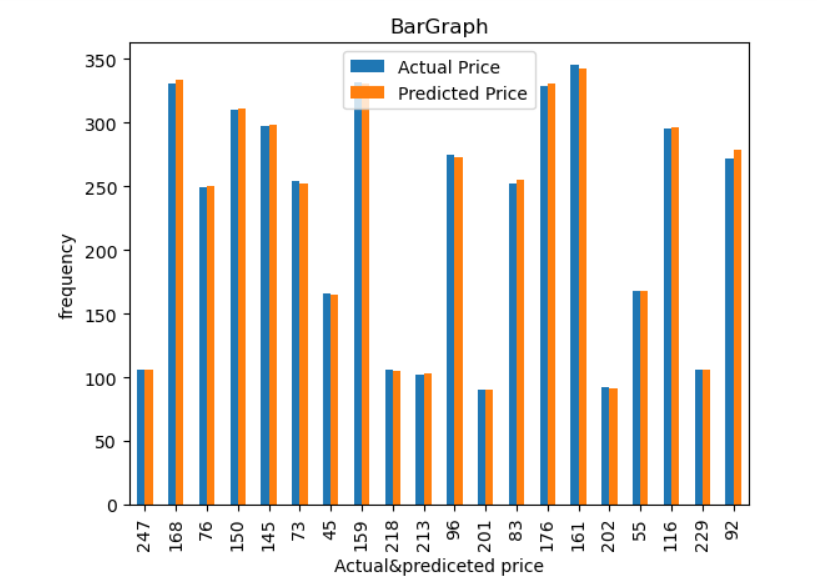
plt.xlabel('Actual&prediceted price')

plt.ylabel('frequency')

plt.show()

**output:**

**summary:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

****

# **Training the model for RandomForestRegressor**

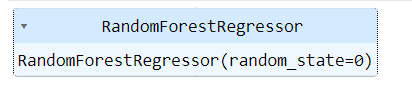
**#model training**

from sklearn.ensemble import RandomForestRegressor

regressor = RandomForestRegressor(n\_estimators=100,random\_state=0)

regressor.fit(x\_train,y\_train)

**output:**

****

**#Evaluating the model**

train\_accuracy=regressor.score(x\_train,y\_train)

print('train\_accuracy(R\_Squared):',train\_accuracy)

test\_accuracy=regressor.score(x\_test,y\_test)

print('test\_accuracy(R\_squared):',test\_accuracy)

**output:**

train\_accuracy(R\_Squared): 0.9998571271849735

test\_accuracy(R\_squared): 0.9987098858585105

**#Comparision between Linear and RandomForestRegression using barplot**

linear\_regression\_accuracy =0.9995657233927607

random\_forest\_accuracy =0.9987098858585105

accuracy\_scores = [linear\_regression\_accuracy, random\_forest\_accuracy]

model\_names = ['Linear Regression', 'Random Forest Regression']

plt.bar(model\_names, accuracy\_scores)

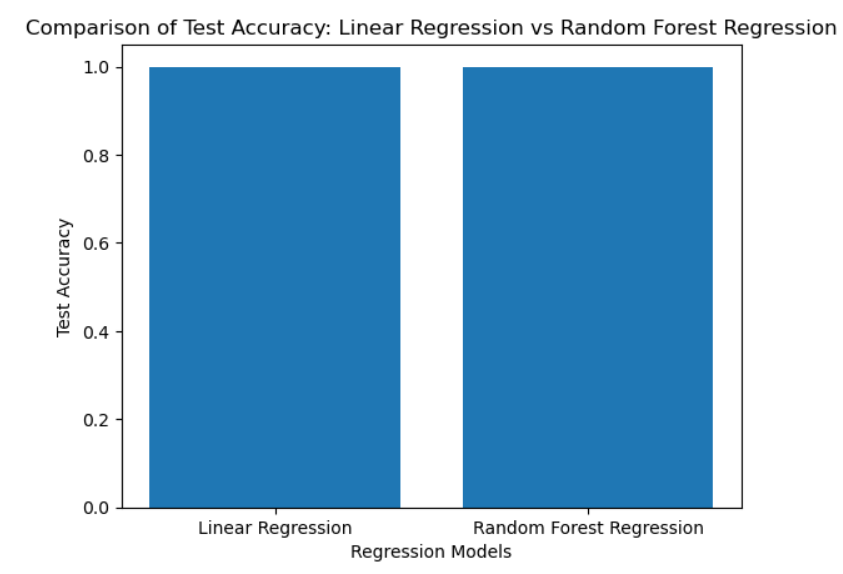
plt.xlabel('Regression Models')

plt.ylabel('Test Accuracy')

plt.title('Comparison of Test Accuracy: Linear Regression vs Random Forest Regression')

plt.show()

**output:**



**Summary:**