# **Methodology:**

**Dataset:**

The dataset used in this project was sourced from Kaggle, specifically from the dataset titled "Stock market analysis". It contains comprehensive information about stock market . The dataset encompasses 8 essential features, including Ticker, Date, open, close, adj close, low, high, volume, and the target variable—close.

In this project, we will employ data preprocessing, feature engineering, and various machine learning models to predict stock prices accurately. Our objective to build predictive models for market analysis.

**Algorithm:**

1)**Linear Regression:**

Linear regression is a statistical algorithm that helps to predict or visualize the relationship between two different features/variables. It involves examining two kinds of variables: the dependent variable and the independent variable. The independent variable is the variable that stands by itself, not impacted by the other variable. As the independent variable is adjusted, the levels of the dependent variable will fluctuate. The dependent variable is the variable that is being studied, and it is what the regression model solves for/attempts to predict.

**2)Random Forest Regressor:**

Random forest regression is a machine learning algorithm that uses an ensemble of decision trees to perform regression tasks. It is a bagging technique that involves training multiple decision trees on different subsets of the training data and then averaging their predictions to obtain the final output. This helps to reduce overfitting and improve the accuracy of the model. In the case of a regression problem, the final output is the mean of all the outputs.

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()

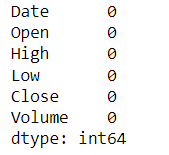
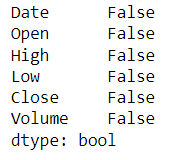
**#To check last five columns**

df.tail()

**#checking missing values**

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

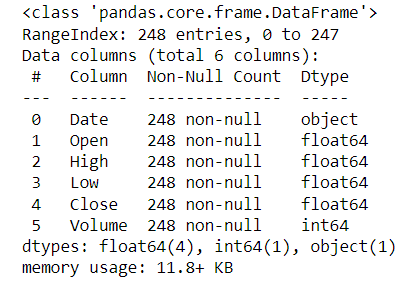
**output:**

**or **

**#checking information about dataset**

df.info()

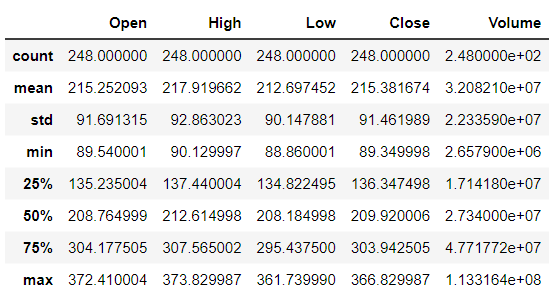
**output:**



**#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**

**#LineChart for open and close price**

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

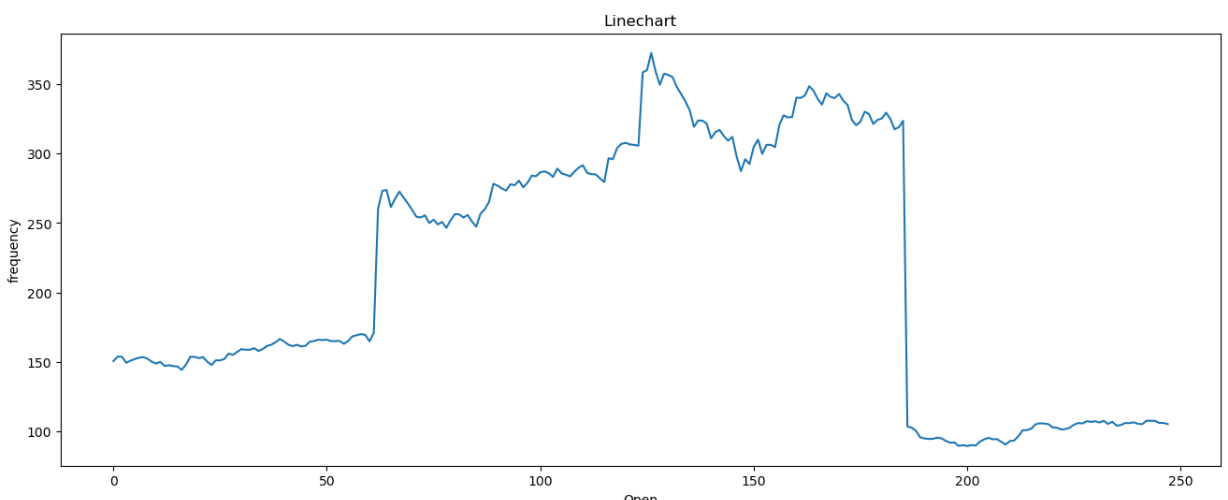
plt.title('Linechart')

plt.xlabel('Open')

plt.ylabel('frequency')

plt.show()

**output:**

****

**Summary:**

**In the above shown line graph x- axis represents current market value and y-axis represents current time period .The chart shows movement in market where we can easily track the highest point and lowest point by viewing the chart.**

# **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 LinearRegression**

**#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,)

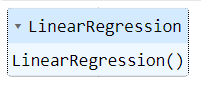
ytest: (62,)

**#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

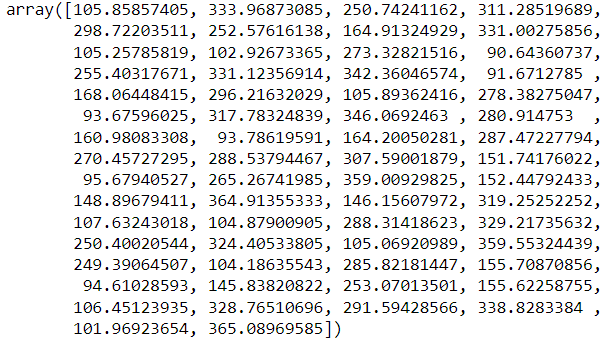
Co-efficient: [-5.45434461e-01 7.34498290e-01 8.12746520e-01 4.09319930e-096.34908792e-16 1.03687704e-01 1.15818167e-03]

**#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

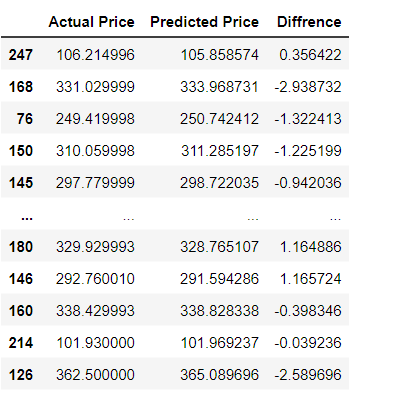
Root Mean Squared Error: 1.8208580458250567

**#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')

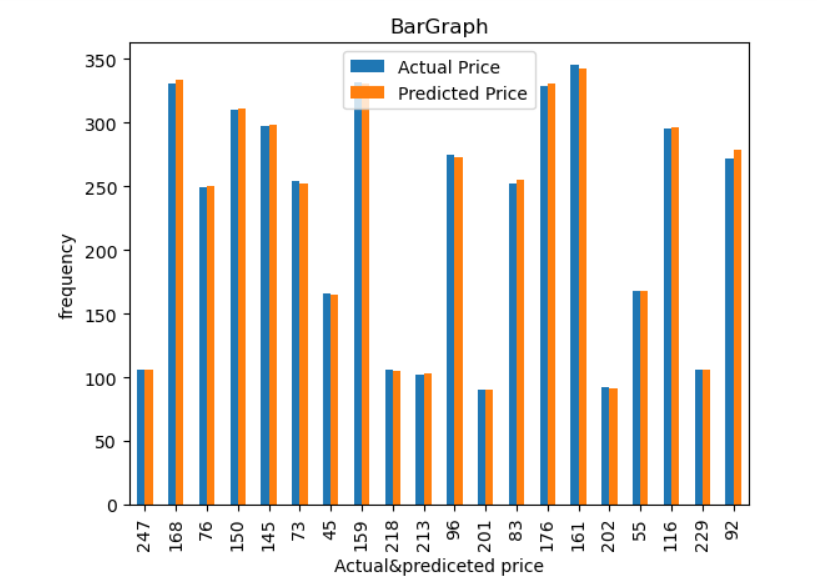
plt.ylabel('frequency')

plt.show()

**output:**

**summary:**

**In the below shown line graph x- axis represents predicted and actual price and y-axis represents categories .There we are going to compare actual and predicted price, It shows actual and predicted prices are same.Beacuse it has best accuracy.**

****

# **Training the model for RandomForestRegressor**

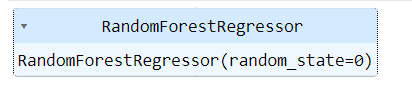
**#model training**

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')

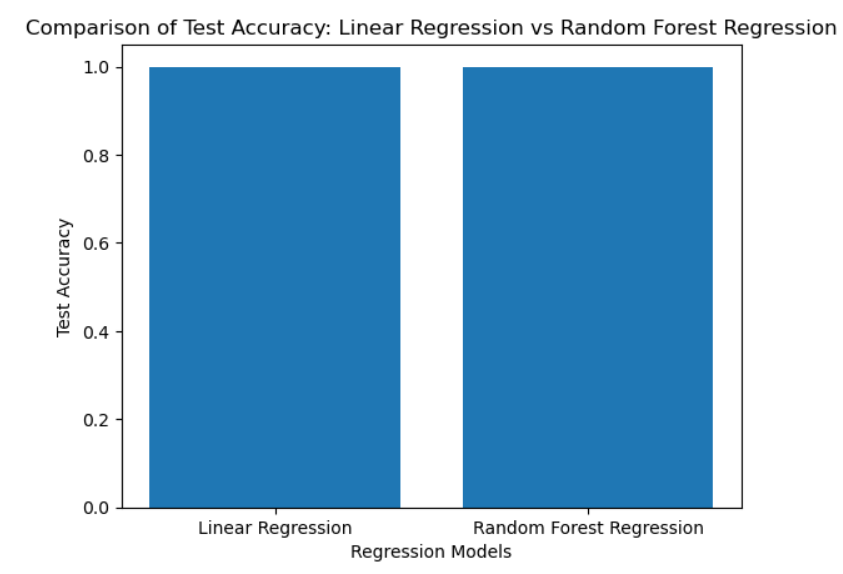
plt.title('Comparison of Test Accuracy: Linear Regression vs Random Forest Regression')

plt.show()

**Result:**

**In this case, the x-axis might represent the different models (linear regression and random forest regression), while the y-axis represents a performance metric or evaluation criterion.**

**The heights of the bars for linear regression and random forest regression are same.Because two models got best accuracy.**



Stock Market Analysis

Introduction:

Stock market analysis is a critical aspect of investment decision-making, aiming to identify patterns and trends in stock prices to make informed predictions about future movements. In this mini project, we will explore the application of regression algorithms in stock market analysis.

The objective of this mini project is to analyze historical stock market data and build regression models to predict future stock prices. Regression algorithms, such as linear regression, RandomForest regression can be employed to establish relationships between input variables (such as historical stock prices, trading volumes, or economic indicators) and the target variable.

Throughout the project, we will work with historical stock market data, preprocess and clean the data, select appropriate input variables, and split the dataset into training and testing sets. We will then apply various regression algorithms to train the models using the training set and evaluate their performance using the testing set.

The project will involve assessing the accuracy of the regression models by measuring metrics such as mean squared error (MSE), root mean squared error (RMSE), or R-squared value. These metrics will help us gauge how well the models fit the data and their ability to make reliable predictions.

By undertaking this mini project, you will gain hands-on experience in applying regression algorithms to stock market analysis. You will develop skills in data preprocessing, model training, evaluation, and interpretation of results. These skills can be further expanded and applied to real-world investment scenarios or extended to more advanced machine learning techniques.

Remember that this mini project serves as a learning opportunity, and any investment decisions should be made after careful consideration of multiple factors and consultation with a qualified financial professional.

**Conclusion:**

**both linear regression and random forest regression are commonly used techniques for stock market analysis in mini projects.**

**Linear regression is a simple and interpretable algorithm that assumes a linear relationship between the input features and the target variable. It can provide insights into the direction and magnitude of the relationships between variables. However, it may not capture complex nonlinear relationships and interactions among the features.**

**On the other hand, random forest regression is an ensemble method that combines multiple decision trees to make predictions. It can capture nonlinear relationships and interactions effectively, and it is less prone to overfitting compared to a single decision tree. Random forest regression also provides importance scores for the input features, which can be useful for feature selection and interpretation.**

**When choosing between linear regression and random forest regression for stock market analysis in a mini project, several factors should be considered. Linear regression may be suitable when the relationships between the features and the target variable are expected to be roughly linear and when interpretability and simplicity are important. Random forest regression may be more appropriate when there are complex nonlinear relationships and interactions among the variables, and when higher prediction accuracy is desired.**

**It is important to note that stock market analysis is a challenging task due to its nonstationary nature, high volatility, and the influence of numerous external factors. Therefore, the choice of regression algorithm is just one aspect of a comprehensive analysis. Other factors such as data preprocessing, feature engineering, model evaluation, and incorporating domain knowledge are also critical for accurate and reliable stock market predictions.**