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 project, we will explore the application of regression algorithms in stock market analysis.
* Regression models are statistical techniques used to establish relationships between one dependent variable (in this case, stock prices) and one or more independent variables (such as economic indicators, company financials, or market sentiment). By analyzing historical data and identifying patterns, regression models can help predict future stock price movements.
* To build a regression model for stock market analysis, relevant data needs to be collected and preprocessed. This includes historical stock prices, as well as data on independent variables that might impact stock prices. Data may be obtained from various sources such as financial databases, stock exchanges, or specialized market data providers.
* we will explore the fundamentals of stock market analysis using regression models, with a focus on the column names commonly associated with financial datasets: ticker, date, open, high, low, adjclose, volume, and close. These columns provide essential information for analyzing stock market data and building regression models to estimate and predict stock prices.
* After training and evaluating the model, insights can be derived from its results. The coefficients of the independent variables in the regression model provide information about their impact on stock prices. Positive coefficients indicate a positive relationship, while negative coefficients indicate a negative relationship. This information can help investors make informed decisions.

# **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.**

**The random forest algorithm is an extension of the bagging method as it utilizes both bagging and feature randomness to create an uncorrelated forest of decision trees.**

**3)ANN Model:**

**An Artificial Neural Network (ANN) is a computational model inspired by the structure and functioning of the human brain. It is a type of machine learning algorithm that is designed to recognize patterns and make predictions or decisions based on input data. ANNs can be used for various tasks, including classification, regression, and pattern recognition. They have been successfully applied in areas such as image and speech recognition, natural language processing, financial forecasting, and many other domains.**

**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 RandomForestRegressor

from sklearn.neural\_network import MLPRegressor

from sklearn.preprocessing import StandardScaler

# **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()

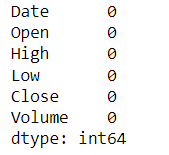
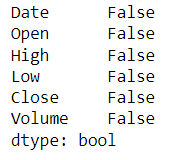
**#To check last five columns**

df.tail()

**#checking missing values**

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

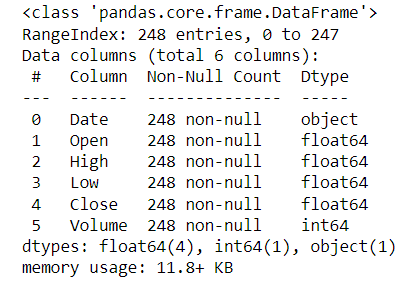
**output:**

**or **

**#checking information about dataset**

df.info()

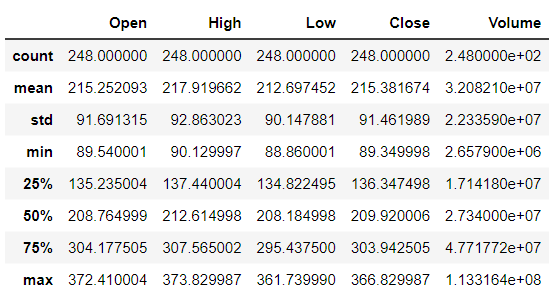
**output:**



**#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 openinig the stock market**

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

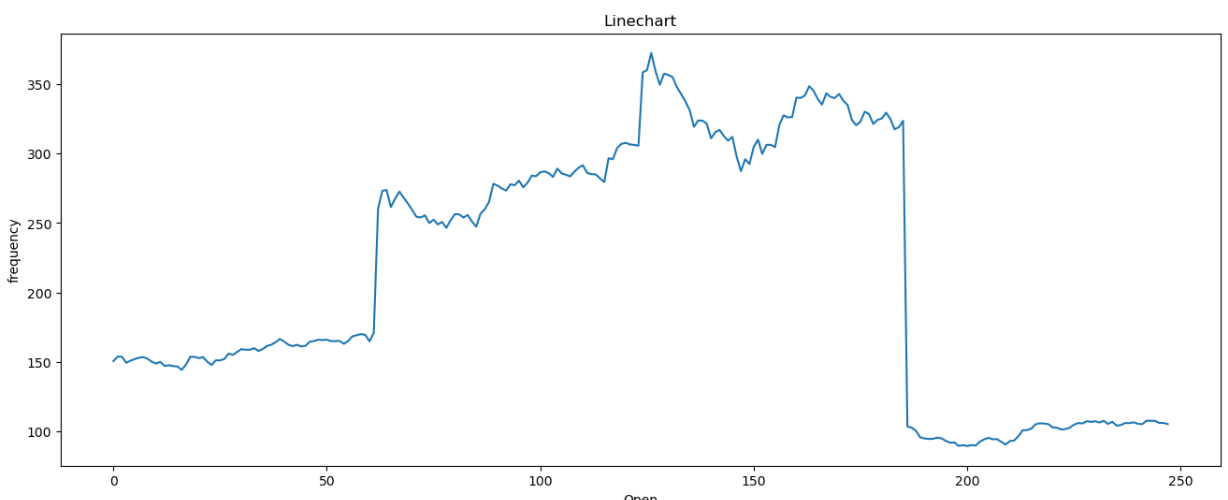
plt.title('Linechart')

plt.xlabel('Open')

plt.ylabel('frequency')

plt.show()

**output:**

****

**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 Using Linear Regression**

**#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,)

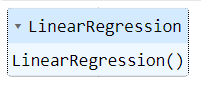
ytest: (62,)

**#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

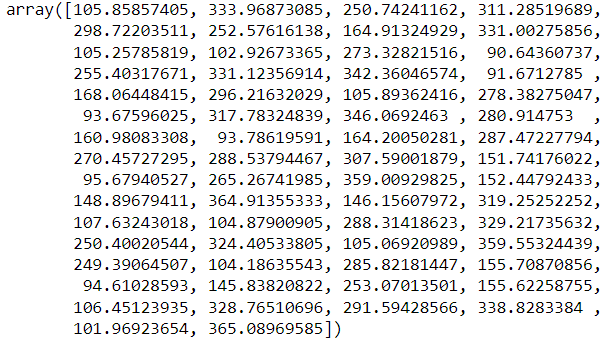
Co-efficient: [-5.45434461e-01 7.34498290e-01 8.12746520e-01 4.09319930e-096.34908792e-16 1.03687704e-01 1.15818167e-03]

**#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

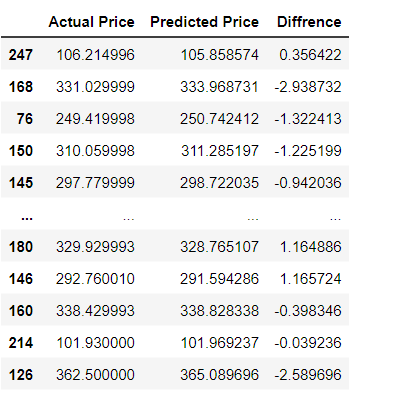
Root Mean Squared Error: 1.8208580458250567

**#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')

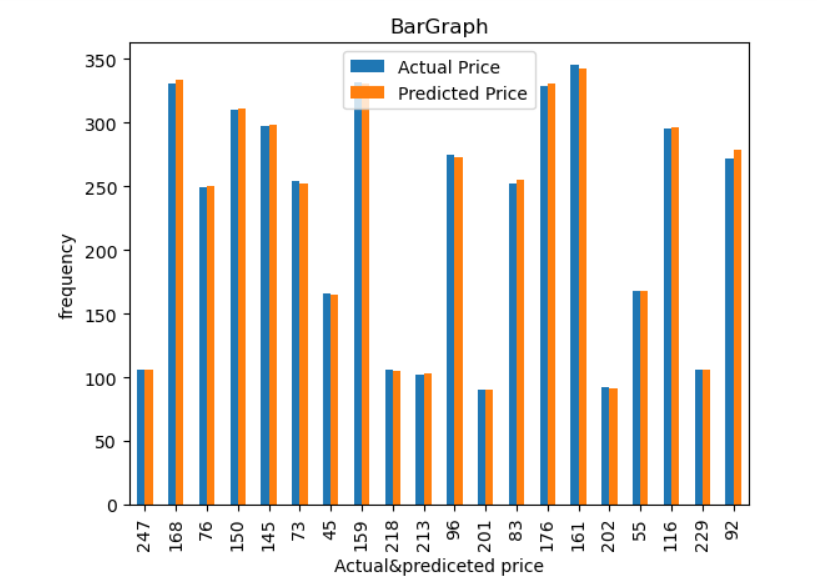
plt.xlabel('Actual&prediceted price')

plt.ylabel('frequency')

plt.show()

**output:**

**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. Because it has best accuracy.**

****

# **Training the model using RandomForestRegressor**

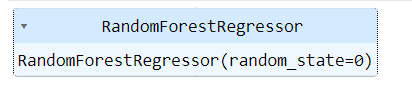
**#model training**

from sklearn.ensemble import RandomForestRegressor

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

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

**output:**

****

**#Prediction**

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

y\_pred

**#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

**#Preprocessing the x\_train and x\_test**

scaler = StandardScaler()

x\_train = scaler.fit\_transform(x\_train)

x\_test = scaler.transform(x\_test)

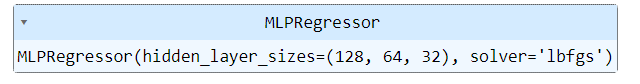
**ANN Model**

**#Training the model using ANN model**

ann\_model= MLPRegressor(hidden\_layer\_sizes=(128,64,32),activation='relu',solver='lbfgs')

ann\_model.fit(x\_train, y\_train)

**output:**

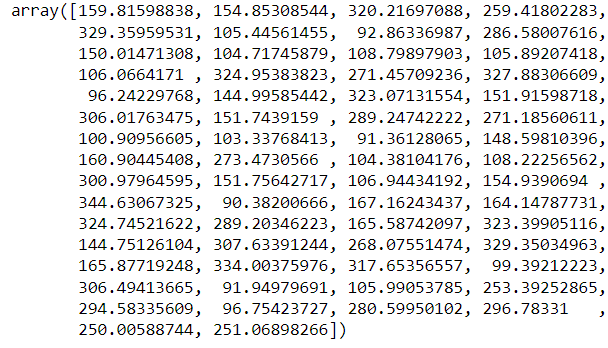
****

**#Prediction**

y\_pred = ann\_model.predict(x\_test)

y\_pred

**output:**



**#Evaluating the model**

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

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

test\_accuracy=ann\_model.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): 0.999662262302119

test\_accuracy(R\_squared): 0.9996151534568192

Mean Absolute Error: 1.3601397944771283

Mean Squared Error: 3.139341486967782

Root Mean Squared Error: 1.771818694722398

**#Comparision between Linear Regression, RandomForestRegression and ANN Model using barplot.**

linear\_regression\_accuracy =0.9995657233927607

Random\_forest\_accuracy =0.9987098858585105

ANN=0.9996151534568192

accuracy\_scores = [linear\_regression\_accuracy, Random\_forest\_accuracy,ANN]

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

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

plt.xlabel('Regression Models')

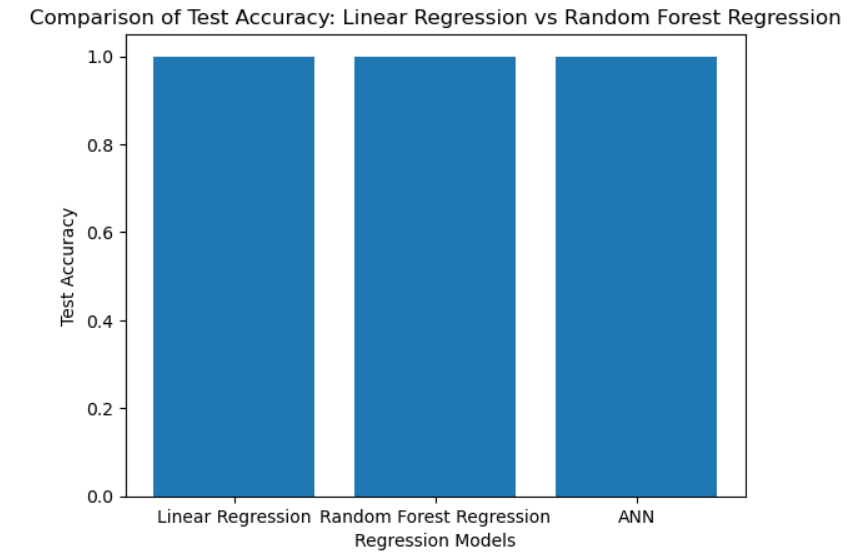
plt.ylabel('Test Accuracy')

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, Random forest regression and ANN model), while the y-axis represents a performance metric or evaluation criterion. We have compared test accuracy(R-squared) between Linear Regression model, Random Forest Regression model and ANN model using pictorial representation is shown below.**

****

**Linear Regression Accuracy:0.9995**

**Random Forest Regressor Accuracy: 0.9987**

**ANN Model:0.9996**

**After comparing the test accuracy between these models -**

**The ANN Model has given the best accuracy.**

**Conclusion**

**In this model , we have deleted the unwanted columns which contains the NULL values. Also we have the checked whether the dataset contains missing values, information about the dataset, statistics of the dataset. Also we have preprocessed the data.**

**In the stage of training the model , we have splited the dataset into train and test dataset, and we applied a linearRegression , Randomforest regressor and ANN model and we found train and test accuracy for all three models. Then we compared linear regression, random forest regressor and ANN model using graph.**

**In this comparision, we got best accuracy in ANN Model.**