DJIA STOCK TIME SERIES

Abstract:

**The DJIA is a price-weighted index that tracks 30 large, publicly-owned companies trading on the New York Stock ExchangeTime-series analysis contains a set of techniques and methods to analyze time-series data and extract meaningful insights from it. On the other hand, time-series forecasting is a predictive analysis approach that predicts future values based on historical data which is collected over a period** **of time.The Dow Jones Industrial Average (DJIA), also commonly referred to as “the Dow Jones” or simply “the Dow,” is one of the most popular and widely-recognized stock market indices.. The DJIA is one of the stock indices created by Dow & Jones Company founder wall street and Journal editor Charles Dow.**

**The data is presented in a couple of formats to suit different individual's needs or computational limitations.**

**The DJIA is also used to calculate other important stock market indexes, such as the Dow Jones Transportation Average and the Dow Jones Utility Average.**

**Time series data is a sequence of data points in chronological order that is used by businesses to analyze past data and make future predictions. These data points are a set of observations at specified times and equal intervals, typically with a datetime index and corresponding value**

*METHODOLLOGY*

* ***MultiLinearRegression***

**Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable**

**Multiple Linear Regression is one of the important regression algorithms which models the linear relationship between a single dependent continuous variable and more than one**

**A multiple regression model extends to several explanatory variables.**

* ***RandomForestRegression***

**Random forest is a type of supervised learning algorithm that uses ensemble methods (bagging) to solve both regression and classification problems. The algorithm operates by constructing a multitude of decision trees at training time and outputting the mean/mode of prediction of the individual trees.**

**Random Forest is also a classic example of a bagging approach as we use different subsets of data in each model to make predictions.**

***Importing modules***

Import pandas as pd

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

**from** sklearn.linear\_model **import** LinearRegression

**from** sklearn.metrics **import** confusion\_matrix,accuracy\_score

**from** sklearn.ensemble **import** RandomForestRegressor

***Reading the Dataset***

**import** pandas **as** pd

df**=**pd**.**read\_csv("C:\\Users\\GPT BANTWAL\\AI&ML\\AABA\_2006-01-01\_to\_2018-01-01.csv")

df

**output:**



***check the five columns***

df**.**head()

**output:**



df**.**shape

**output:**



***Check the missing value***

df**.**isnull()**.**sum()

**output:**



df**.**isnull()**.**any()

**output:**



***Checking the information Dataset***

df**.**info()

**output:**



df**.**describe()

**output:**



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

**output:**



x**=**df**.**drop(['Open','Date','Name'],axis**=**1)

x

**output:**



y**=**df['Open']

y

**output:**



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

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

print(x\_train**.**shape)

print(x\_test**.**shape)

print(y\_train**.**shape)

print(y\_test**.**shape)

**output:**



**from** sklearn.linear\_model **import** LinearRegression

regressor**=**LinearRegression()

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

**output:**



print(regressor**.**coef\_)

**output:**



print(regressor**.**intercept\_)

**output:**



***Prediction***

predicted**=**regressor**.**predict(x\_test)

predicted

**output:**



print(x\_test)

print(x\_test)

**output:**



predicted**.**shape

**output:**



dframe**=**pd**.**DataFrame(y\_test,predicted)

dframe

**output:**



dfr**=**pd**.**DataFrame({'Actual Price':y\_test,'Predicted Price':predicted})

dfr

**Output:**



dfr**.**head(60)

**output:**



**from** sklearn.metrics **import** confusion\_matrix,accuracy\_score

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

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

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

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

**output:**



**import** math

**from** sklearn **import** metrics

print('Mean Absolute Error:',metrics**.**mean\_absolute\_error(y\_test,predicted))

print('Mean Squared Error:',metrics**.**mean\_squared\_error(y\_test,predicted))

print('Root Mean Squared Error:',math**.**sqrt(metrics**.**mean\_squared\_error(y\_test,predicted)))

**output:**



graph**=**dfr**.**head(20)

graph

**Output:**



***plotting the bargraph to check difference between actualprice and predicted price***

graph**.**plot(kind**=**'bar')

plt**.**title('BarGraph')

plt**.**xlabel('Actual&prediceted price')

plt**.**ylabel('frequency')

plt**.**show()

**Output:**



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

x\_train,x\_test,y\_train,y\_test**=**train\_test\_split(x,y,test\_size**=**0.90,random\_state**=**100)

print(x\_train**.**shape)

print(x\_test**.**shape)

print(y\_train**.**shape)

print(y\_test**.**shape)

**output:**



***Model trainning***

**from** sklearn.ensemble **import** RandomForestRegressor

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

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

**output:**



predicted**=**regressor**.**predict(x\_test)

predicted

**output:**



***Evaluating the model***

**from** sklearn.metrics **import** confusion\_matrix,accuracy\_score

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

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

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

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

**output:**



***Linear and Randomforestregression using bar plot***

**import** matplotlib.pyplot **as** plt

linear\_regression\_accuracy **=** 0.999700039585113

random\_forest\_accuracy **=** 0.998957807027415

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:**

**Result:**

**In this case, x-axis represents the regression models and y-axis represents the test accuracy it is comparison between linear regression and random forest regressor**

Linear Regression Accuracy:0.99

Random Forest Regressor Accuracy:0.99

**In this model Random Forest as given best accuracy, the pictorial representation of graph is shown below**

**Conclusion:**

**In this model,we have preprocessed the data using LabelEncoder and converted the data from categorical to the numerical data.In the stage of training the model,it has been splited the train and test data set,and we also applied a linear regression algorithm to model to train it.also we applied the random forest regressor algorithm and find the accuracy.Then we compared linear regresson and random forest regressor using graph,in this comparision we got the best accuracy in the random forest regressor**

**from sklearn.preprocessing import StandardScaler**

**scaler=StandardScaler()**

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

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

**from sklearn.neural\_network import MLPRegressor**

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

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

**output:**

**ann\_prediction = ann\_model.predict(x\_test)**

**ann\_prediction**

**output:**

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

**print('train\_accuracy(R\_squered):',train\_accuracy)**

**test\_accuracy=ann\_model.score(x\_test,y\_test)**

**print('test\_accuracy(R\_squered):',test\_accuracy)**

**output:**

**print('Mean absolute Error:',metrics.mean\_absolute\_error(y\_test,predicted))**

**print('Mean Squared Error:',metrics.mean\_squared\_error(y\_test,predicted))**

**print('Root Mean Squared Error:',math.sqrt(metrics.mean\_squared\_error(y\_test,predicted)))**

**output:**

**import matplotlib.pyplot as plt**

**linear\_regression\_accuracy = 0.999700039585113**

**random\_forest\_accuracy = 0.998957807027415**

**ANN=0.9998289267069702**

**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 vs ANN')**

**plt.show()**

**output:**

