Step by step execution is given considering random forest algorithm .

Dataset: ' NFLX\_dataset.csv'

**Random Forest Algorithm**

**//Stock Price Prediction of netflix**

from google.colab import files

uploaded = files.upload()

import pandas as pd

import io

nflx\_data= pd.read\_csv(io.BytesIO(uploaded[' NFLX\_dataset.csv']))

print(nflx\_data)

output:

Date Open High Low Close Adj Close \

0 05/02/2018 262.000000 267.899994 250.029999 254.259995 254.259995

1 06/02/2018 247.699997 266.700012 245.000000 265.720001 265.720001

2 07/02/2018 266.579987 272.450012 264.329987 264.559998 264.559998

3 08/02/2018 267.079987 267.619995 250.000000 250.100006 250.100006

4 09/02/2018 253.850006 255.800003 236.110001 249.470001 249.470001

... ... ... ... ... ... ...

1004 31/01/2022 401.970001 427.700012 398.200012 427.140015 427.140015

1005 01/02/2022 432.959991 458.480011 425.540009 457.130005 457.130005

1006 02/02/2022 448.250000 451.980011 426.480011 429.480011 429.480011

1007 03/02/2022 421.440002 429.260010 404.279999 405.600006 405.600006

1008 04/02/2022 407.309998 412.769989 396.640015 410.170013 410.170013

Volume

0 11896100

1 12595800

2 8981500

3 9306700

4 16906900

... ...

1004 20047500

1005 22542300

1006 14346000

1007 9905200

1008 7782400

[1009 rows x 7 columns]

nflx\_data.info()

nflx\_data.isna().sum() // to check null and summing

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1009 entries, 0 to 1008

Data columns (total 7 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Date 1009 non-null object

1 Open 1009 non-null float64

2 High 1009 non-null float64

3 Low 1009 non-null float64

4 Close 1009 non-null float64

5 Adj Close 1009 non-null float64

6 Volume 1009 non-null int64

dtypes: float64(5), int64(1), object(1)

memory usage: 55.3+ KB

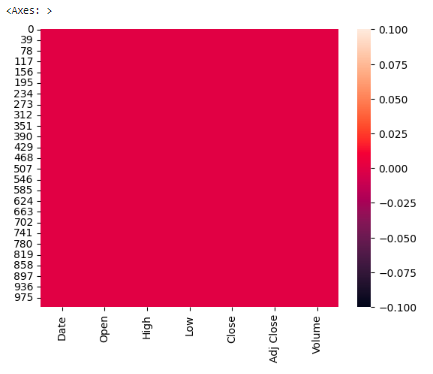
|  | **0** |
| --- | --- |
| **Date** | 0 |
| **Open** | 0 |
| **High** | 0 |
| **Low** | 0 |
| **Close** | 0 |
| **Adj Close** | 0 |
| **Volume** | 0 |

**Note:** Here the column is not having non null data

Now checking using heat map

import seaborn as sea

sea.heatmap(nflx\_data.isna())



**Note : you can see no white spaces this shows that it is not having non null value**

**// changing all column to small case**

nflx\_data.columns=[cols.lower() for cols in nflx\_data.columns]

nflx\_data.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1009 entries, 0 to 1008

Data columns (total 7 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 date 1009 non-null object

1 open 1009 non-null float64

2 high 1009 non-null float64

3 low 1009 non-null float64

4 close 1009 non-null float64

5 adj close 1009 non-null float64

6 volume 1009 non-null int64

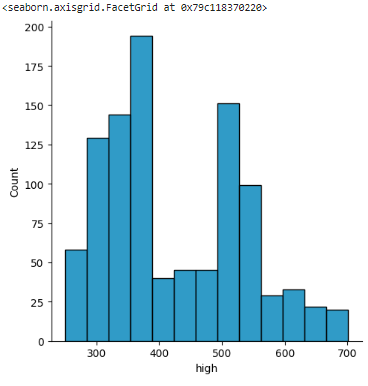
dtypes: float64(5), int64(1), object(1)

memory usage: 55.3+ KB

Note: After changing all column to small case, you can see above

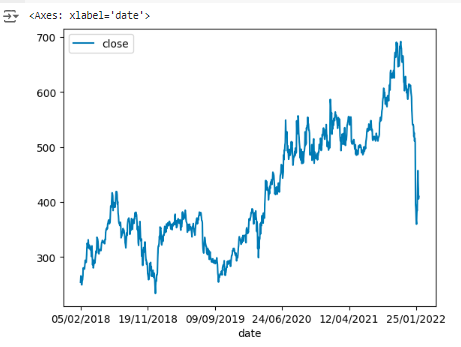
**// to see distribution plot**

**sea**.displot(x=’high’, data=nflx\_data)



**// Plotting the value for x and y**

nflx\_data.plot(x='date',y='close')



// now we are going to drop certain columns and put it in input data and output data

input\_data= nflx\_data.drop(columns=['date','close','adj close'])

output\_data=nflx\_data['close']

print(input\_data)

open high low volume

0 262.000000 267.899994 250.029999 11896100

1 247.699997 266.700012 245.000000 12595800

2 266.579987 272.450012 264.329987 8981500

3 267.079987 267.619995 250.000000 9306700

4 253.850006 255.800003 236.110001 16906900

... ... ... ... ...

1004 401.970001 427.700012 398.200012 20047500

1005 432.959991 458.480011 425.540009 22542300

1006 448.250000 451.980011 426.480011 14346000

1007 421.440002 429.260010 404.279999 9905200

1008 407.309998 412.769989 396.640015 7782400

[1009 rows x 4 columns]

print(output\_data)

0 254.259995

1 265.720001

2 264.559998

3 250.100006

4 249.470001

...

1004 427.140015

1005 457.130005

1006 429.480011

1007 405.600006

1008 410.170013

Name: close, Length: 1009, dtype: float64

**// to check how many columns present in input and output data**

print(input\_data.shape)

print(output\_data.shape)

**output:**

(1009, 4)

(1009,)

**// for training and spliting**

from sklearn.model\_selection import train\_test\_split // for training and spliting

input\_data\_train,input\_data\_test,output\_data\_train, output\_data\_test=train\_test\_split(input\_data,output\_data,test\_size=0.2)

print(input\_data\_train.shape)

print(output\_data\_train.shape)

print(input\_data\_test.shape)

print(output\_data\_test.shape)

**Output:**

(706, 4)

(706,)

(303, 4)

(303,)

// We are predicting stock price and it is continuous hence regression should be used.

// importing Random Forest Regression algorithm

from sklearn.ensemble import RandomForestRegressor

model= RandomForestRegressor()

model.fit(input\_data\_train,output\_data\_train)

**// Predicting input\_data\_test with random forest model**

nflx\_predict=model.predict(input\_data\_test)

//now machine learning has predicted.

// Now checking machine learning model prediction with output\_data

// using metrics for checking

from sklearn  import metrics

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

print("mean absolute error:",metrics. mean\_squared\_error(nflx\_predict,output\_data\_test))

print("mean squared error:",metrics. mean\_absolute\_error(nflx\_predict, output\_data\_test))

print("R2 score:",metrics.r2\_score(nflx\_predict, output\_data\_test))

**output:**

mean absolute error: 29.093397769864307

mean squared error: 3.829250509636968

R2 score: 0.9972980133594307

print(input\_data\_test.tail(3))

input\_random=input\_data\_test.tail(3)

input\_random

open high low volume

733 539.000000 540.799988 515.090027 4444400

553 431.000000 432.000000 414.700012 12616300

166 384.380005 386.799988 373.829987 8638700

|  | **open** | **high** | **low** | **volume** |
| --- | --- | --- | --- | --- |
| **733** | 539.000000 | 540.799988 | 515.090027 | 4444400 |
| **553** | 431.000000 | 432.000000 | 414.700012 | 12616300 |
| **166** | 384.380005 | 386.799988 | 373.829987 | 8638700 |

print(output\_data\_test.tail(3))

733 522.859985

553 422.959991

166 377.140015

Name: close, dtype: float64

stock\_predict\_test=model.predict(input\_random)

stock\_predict\_test

array([527.85420507, 425.58269806, 383.48749537])