In [2]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
```

In [3]:
```
data = pd.read_csv("finance_data.csv")
```

In [4]:
```
data.head()
```

Out[4]:

	Identifier	Date	Time	Open	High	Low	Close	Volume	Extra
0	BANKNIFTY	2016/01/01	09:17	16912.60	16913.80	16899.00	16908.40	42	NaN
1	BANKNIFTY_F1	2016/01/01	09:17	16938.00	16940.00	16925.90	16934.80	13050	NaN
2	INDIAVIX	2016/01/01	09:17	13.95	14.04	13.93	14.00	37	NaN
3	NIFTY	2016/01/01	09:17	7928.65	7929.65	7926.30	7926.45	43200	NaN
4	NIFTY_F1	2016/01/01	09:17	7938.00	7938.10	7932.55	7933.25	43200	NaN

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2319695 entries, 0 to 2319694
Data columns (total 9 columns):
 #   Column      Dtype  
---  ------      -----  
 0   Identifier  object 
 1   Date        object 
 2   Time        object 
 3   Open        float64
 4   High        float64
 5   Low         float64
 6   Close       float64
 7   Volume      int64  
 8   Extra       float64
dtypes: float64(5), int64(1), object(3)
memory usage: 159.3+ MB
```

In [6]:
```
data.describe()
```

Out[6]:

	Open	High	Low	Close	Volume	Extra
count	2.319695e+06	2.319695e+06	2.319695e+06	2.319695e+06	2.319695e+06	1858466.0
mean	1.374949e+04	1.375366e+04	1.374532e+04	1.374947e+04	8.809004e+03	0.0
std	9.763408e+03	9.766586e+03	9.760215e+03	9.763395e+03	2.222699e+04	0.0
min	2.300000e+00	1.005750e+01	2.300000e+00	9.867500e+00	0.000000e+00	0.0
25%	8.226750e+03	8.228800e+03	8.224950e+03	8.226650e+03	0.000000e+00	0.0
50%	1.132000e+04	1.132280e+04	1.131770e+04	1.132000e+04	5.700000e+01	0.0
75%	2.280990e+04	2.281630e+04	2.280320e+04	2.280990e+04	8.840000e+03	0.0
max	3.276860e+04	3.277390e+04	3.275610e+04	3.276880e+04	1.664025e+06	0.0

In [7]:
```
data.isna().sum()
```

Out[7]:
```
Identifier         0
Date               0
Time               0
Open               0
High               0
Low                0
Close              0
Volume             0
Extra         461229
dtype: int64
```

In [8]:
```
data.drop("Extra",axis = 1)
```

Out[8]:

	Identifier	Date	Time	Open	High	Low	Close	Volume
0	BANKNIFTY	2016/01/01	09:17	16912.60	16913.80	16899.00	16908.40	42
1	BANKNIFTY_F1	2016/01/01	09:17	16938.00	16940.00	16925.90	16934.80	13050
2	INDIAVIX	2016/01/01	09:17	13.95	14.04	13.93	14.00	37
3	NIFTY	2016/01/01	09:17	7928.65	7929.65	7926.30	7926.45	43200
4	NIFTY_F1	2016/01/01	09:17	7938.00	7938.10	7932.55	7933.25	43200
...	...	...	...	...	...	...	...	...
2319690	BANKNIFTY	2020/12/31	15:32	31264.10	31264.10	31264.10	31264.10	0
2319691	NIFTY	2020/12/31	15:32	13981.80	13981.80	13981.80	13981.80	0
2319692	INDIAVIX	2020/12/31	15:33	21.09	21.09	21.09	21.09	0
2319693	BANKNIFTY_F1	2020/12/31	18:49	31264.10	31264.10	31264.10	31264.10	1
2319694	NIFTY_F1	2020/12/31	18:49	13981.80	13981.80	13981.80	13981.80	1

2319695 rows × 8 columns

In [9]:
```
data["Date"] = pd.to_datetime(data["Date"])
```

In [10]:
```
data["Day"] = data["Date"].dt.day
data["Month"] = data["Date"].dt.month
data["Year"] = data["Date"].dt.year
```

In [11]:
```
data
```

Out[11]:

	Identifier	Date	Time	Open	High	Low	Close	Volume	Extra	Day
0	BANKNIFTY	2016-01-01	09:17	16912.60	16913.80	16899.00	16908.40	42	NaN	1
1	BANKNIFTY_F1	2016-01-01	09:17	16938.00	16940.00	16925.90	16934.80	13050	NaN	1
2	INDIAVIX	2016-01-01	09:17	13.95	14.04	13.93	14.00	37	NaN	1
3	NIFTY	2016-01-01	09:17	7928.65	7929.65	7926.30	7926.45	43200	NaN	1
4	NIFTY_F1	2016-01-01	09:17	7938.00	7938.10	7932.55	7933.25	43200	NaN	1
...	...	...	...	...	...	...	...	...	...	...
2319690	BANKNIFTY	2020-12-31	15:32	31264.10	31264.10	31264.10	31264.10	0	0.0	31
2319691	NIFTY	2020-12-31	15:32	13981.80	13981.80	13981.80	13981.80	0	0.0	31
2319692	INDIAVIX	2020-12-31	15:33	21.09	21.09	21.09	21.09	0	0.0	31
2319693	BANKNIFTY_F1	2020-12-31	18:49	31264.10	31264.10	31264.10	31264.10	1	0.0	31
2319694	NIFTY_F1	2020-12-31	18:49	13981.80	13981.80	13981.80	13981.80	1	0.0	31

2319695 rows × 12 columns

In [12]:
```
data["Time"] = pd.to_datetime(data["Time"])
```

```
C:\Users\godde\AppData\Local\Temp\ipykernel_16208\3117657178.py:1: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent and as-expected, please specify a format.
  data["Time"] = pd.to_datetime(data["Time"])
```

In [13]:
```
data["Hours"] = data["Time"].dt.hour
data["Minutes"] = data["Time"].dt.minute
```

In [14]:
```
data
```

Out[14]:

	Identifier	Date	Time	Open	High	Low	Close	Volume	Extra	D
0	BANKNIFTY	2016-01-01	2024-10-21 09:17:00	16912.60	16913.80	16899.00	16908.40	42	NaN	
1	BANKNIFTY_F1	2016-01-01	2024-10-21 09:17:00	16938.00	16940.00	16925.90	16934.80	13050	NaN	
2	INDIAVIX	2016-01-01	2024-10-21 09:17:00	13.95	14.04	13.93	14.00	37	NaN	
3	NIFTY	2016-01-01	2024-10-21 09:17:00	7928.65	7929.65	7926.30	7926.45	43200	NaN	
4	NIFTY_F1	2016-01-01	2024-10-21 09:17:00	7938.00	7938.10	7932.55	7933.25	43200	NaN	
...	...	...	...	...	...	...	...	...	...	...
2319690	BANKNIFTY	2020-12-31	2024-10-21 15:32:00	31264.10	31264.10	31264.10	31264.10	0	0.0	
2319691	NIFTY	2020-12-31	2024-10-21 15:32:00	13981.80	13981.80	13981.80	13981.80	0	0.0	
2319692	INDIAVIX	2020-12-31	2024-10-21 15:33:00	21.09	21.09	21.09	21.09	0	0.0	
2319693	BANKNIFTY_F1	2020-12-31	2024-10-21 18:49:00	31264.10	31264.10	31264.10	31264.10	1	0.0	
2319694	NIFTY_F1	2020-12-31	2024-10-21 18:49:00	13981.80	13981.80	13981.80	13981.80	1	0.0	

2319695 rows × 14 columns

In [15]:
```
identifier = pd.get_dummies(data["Identifier"] , drop_first = True)
```

In [16]:
```
identifier = identifier.astype(int)
```

In [17]:
```
identifier
```

Out[17]:

	BANKNIFTY_F1	INDIAVIX	NIFTY	NIFTY_F1
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1
...	...	...	...	...
2319690	0	0	0	0
2319691	0	0	1	0
2319692	0	1	0	0
2319693	1	0	0	0
2319694	0	0	0	1

2319695 rows × 4 columns

In [18]:
```
data = pd.concat([data , identifier] , axis = 1)
```

In [19]:
```
data
```

Out[19]:

	Identifier	Date	Time	Open	High	Low	Close	Volume	Extra	D
0	BANKNIFTY	2016-01-01	2024-10-21 09:17:00	16912.60	16913.80	16899.00	16908.40	42	NaN	
1	BANKNIFTY_F1	2016-01-01	2024-10-21 09:17:00	16938.00	16940.00	16925.90	16934.80	13050	NaN	
2	INDIAVIX	2016-01-01	2024-10-21 09:17:00	13.95	14.04	13.93	14.00	37	NaN	
3	NIFTY	2016-01-01	2024-10-21 09:17:00	7928.65	7929.65	7926.30	7926.45	43200	NaN	
4	NIFTY_F1	2016-01-01	2024-10-21 09:17:00	7938.00	7938.10	7932.55	7933.25	43200	NaN	
...	...	...	...	...	...	...	...	...	...	...
2319690	BANKNIFTY	2020-12-31	2024-10-21 15:32:00	31264.10	31264.10	31264.10	31264.10	0	0.0	
2319691	NIFTY	2020-12-31	2024-10-21 15:32:00	13981.80	13981.80	13981.80	13981.80	0	0.0	
2319692	INDIAVIX	2020-12-31	2024-10-21 15:33:00	21.09	21.09	21.09	21.09	0	0.0	
2319693	BANKNIFTY_F1	2020-12-31	2024-10-21 18:49:00	31264.10	31264.10	31264.10	31264.10	1	0.0	
2319694	NIFTY_F1	2020-12-31	2024-10-21 18:49:00	13981.80	13981.80	13981.80	13981.80	1	0.0	

2319695 rows × 18 columns

In [20]:
```
df = data
```

In [21]:
```
data = data.drop(["Identifier" , "Date" , "Time" , "Extra"] , axis = 1)
```

In [22]:
```
x = data.drop(["High","Low" , "Close"] , axis = 1)
```

In [23]:
```
y = data[["High" ,"Low" ,"Close"]]
```

In [24]:
```
x
```

Out[24]:

	Open	Volume	Day	Month	Year	Hours	Minutes	BANKNIFTY_F1	INDIAVIX	NIFT
0	16912.60	42	1	1	2016	9	17	0	0	
1	16938.00	13050	1	1	2016	9	17	1	0	
2	13.95	37	1	1	2016	9	17	0	1	
3	7928.65	43200	1	1	2016	9	17	0	0	
4	7938.00	43200	1	1	2016	9	17	0	0	
...	...	...	...	...	...	...	...	...	...	
2319690	31264.10	0	31	12	2020	15	32	0	0	
2319691	13981.80	0	31	12	2020	15	32	0	0	
2319692	21.09	0	31	12	2020	15	33	0	1	
2319693	31264.10	1	31	12	2020	18	49	1	0	
2319694	13981.80	1	31	12	2020	18	49	0	0	

2319695 rows × 11 columns

In [25]:
```
y
```

Out[25]:

	High	Low	Close
0	16913.80	16899.00	16908.40
1	16940.00	16925.90	16934.80
2	14.04	13.93	14.00
3	7929.65	7926.30	7926.45
4	7938.10	7932.55	7933.25
...	...	...	...
2319690	31264.10	31264.10	31264.10
2319691	13981.80	13981.80	13981.80
2319692	21.09	21.09	21.09
2319693	31264.10	31264.10	31264.10
2319694	13981.80	13981.80	13981.80

2319695 rows × 3 columns

In [26]:
```
x_train , x_test , y_train , y_test = train_test_split(x,y, test_size = 0.2 , randc
```

In [27]:
```
x_train
```

Out[27]:

	Open	Volume	Day	Month	Year	Hours	Minutes	BANKNIFTY_F1	INDIAVIX	NIFT
456727	8085.75	0	28	12	2016	13	0	0	0	
298423	19318.40	0	24	8	2016	10	17	0	0	
421903	8155.35	50775	2	12	2016	9	25	0	0	
754882	9841.00	12300	18	8	2017	12	14	0	0	
1422280	10918.50	11850	25	1	2019	12	58	0	0	
...	...	...	...	...	...	...	...	...	...	
903324	15.37	35	13	12	2017	9	51	0	1	
2296453	13550.40	14775	14	12	2020	13	26	0	0	
794508	10146.10	53	19	9	2017	12	27	0	0	
446197	8084.45	0	20	12	2016	15	24	0	0	
794056	25030.30	1640	19	9	2017	10	57	1	0	

1855756 rows × 11 columns

In [28]:
```
x_train.shape
```

Out[28]:
```
(1855756, 11)
```

In [29]:
```
x_test
```

Out[29]:

	Open	Volume	Day	Month	Year	Hours	Minutes	BANKNIFTY_F1	INDIAVIX	NIFT
1955989	16631.90	16820	24	3	2020	9	59	1	0	
2297837	13513.50	36900	15	12	2020	11	46	0	0	
1685207	27497.30	0	22	8	2019	10	3	0	0	
983948	25427.70	59	14	2	2018	14	53	0	0	
980034	25588.60	2040	9	2	2018	14	24	1	0	
...	...	...	...	...	...	...	...	...	...	
140886	16622.60	77790	26	4	2016	10	9	1	0	
1012848	10278.50	56	9	3	2018	10	42	0	0	
1496573	11370.80	0	26	3	2019	9	21	0	0	
1967631	61.17	0	1	4	2020	11	4	0	1	
107573	7682.65	8550	28	3	2016	11	36	0	0	

463939 rows × 11 columns

In [30]:
```
x_test.shape
```

Out[30]:
```
(463939, 11)
```

In [31]:
```
y_train
```

Out[31]:

	High	Low	Close
456727	8086.15	8084.700	8084.8500
298423	19323.60	19317.900	19323.3000
421903	8156.30	8153.000	8156.1500
754882	9842.70	9839.200	9841.8000
1422280	10920.00	10916.300	10919.1000
...	...	...	...
903324	15.39	15.335	15.3575
2296453	13560.00	13550.400	13558.0000
794508	10146.10	10143.300	10143.5000
446197	8086.95	8083.950	8084.8500
794056	25030.30	25027.000	25029.0000

1855756 rows × 3 columns

In [32]:
```
y_train.shape
```

Out[32]:
```
(1855756, 3)
```

In [33]:
```
y_test
```

Out[33]:

	High	Low	Close
1955989	16670.00	16540.60	16637.10
2297837	13518.80	13513.30	13518.70
1685207	27515.60	27495.10	27512.60
983948	25439.30	25427.70	25433.00
980034	25595.00	25587.50	25595.00
...	...	...	...
140886	16624.90	16619.00	16619.80
1012848	10281.30	10277.90	10279.80
1496573	11374.00	11364.30	11365.00
1967631	61.21	61.17	61.20
107573	7684.65	7682.20	7684.05

463939 rows × 3 columns

In [34]:
```
y_test.shape
```

Out[34]:
```
(463939, 3)
```

In [35]:
```
model = LinearRegression()
```

In [36]:
```
model
```

Out[36]:
```
▾ LinearRegression
LinearRegression()
```

In [37]:
```
model = model.fit(x_train , y_train)
```

In [38]:
```
y_prediction = model.predict(x_test)
```

In [39]:
```
y_prediction
```

Out[39]:
```
array([[16644.24854433, 16619.56692044, 16631.94021643],
       [13518.53972812, 13508.43509943, 13513.50253232],
       [27506.16479413, 27488.16036862, 27497.19781382],
       ...,
       [11373.73529303, 11367.60055322, 11370.70364444],
       [   63.55933803,    58.66774187,    61.14283058],
       [ 7683.07780831,  7682.16539866,  7682.60244609]])
```

In [40]:
```
y_test
```

Out[40]:

	High	Low	Close
1955989	16670.00	16540.60	16637.10
2297837	13518.80	13513.30	13518.70
1685207	27515.60	27495.10	27512.60
983948	25439.30	25427.70	25433.00
980034	25595.00	25587.50	25595.00
...	...	...	...
140886	16624.90	16619.00	16619.80
1012848	10281.30	10277.90	10279.80
1496573	11374.00	11364.30	11365.00
1967631	61.21	61.17	61.20
107573	7684.65	7682.20	7684.05

463939 rows × 3 columns

In [ ]:
```
```

In [41]:
```
x_test
```

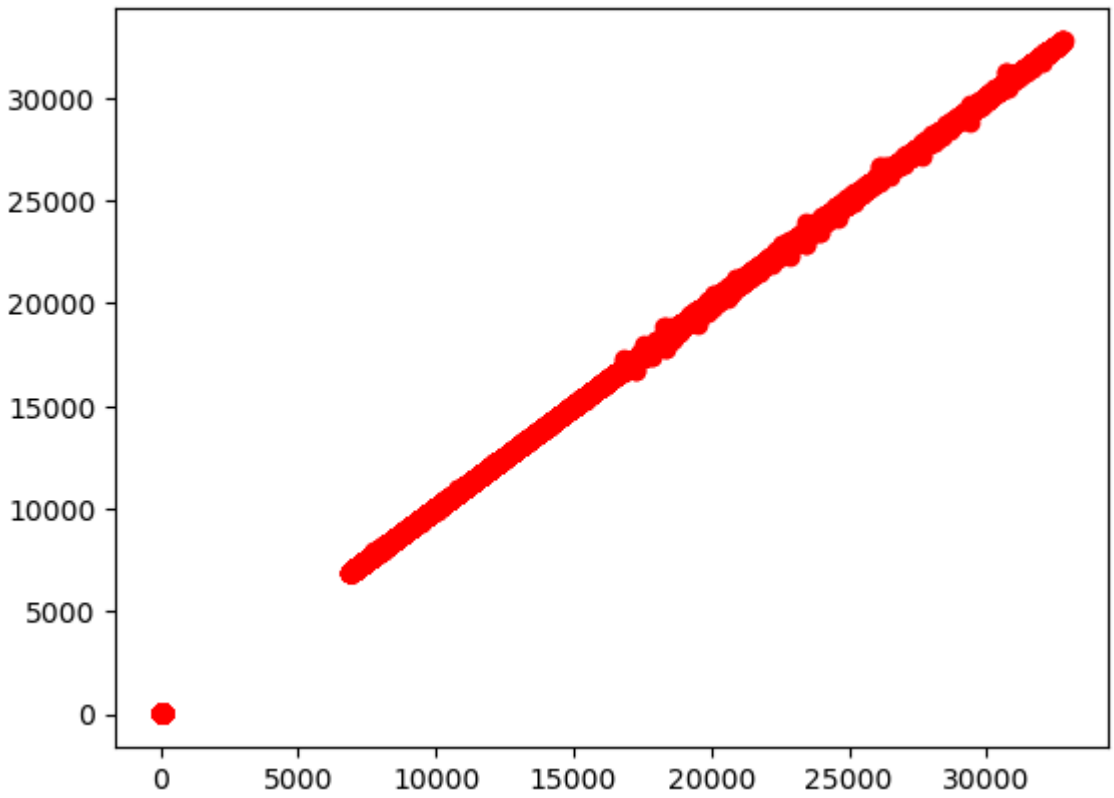
Out[41]:

	Open	Volume	Day	Month	Year	Hours	Minutes	BANKNIFTY_F1	INDIAVIX	NIFT
1955989	16631.90	16820	24	3	2020	9	59	1	0	
2297837	13513.50	36900	15	12	2020	11	46	0	0	
1685207	27497.30	0	22	8	2019	10	3	0	0	
983948	25427.70	59	14	2	2018	14	53	0	0	
980034	25588.60	2040	9	2	2018	14	24	1	0	
...	...	...	...	...	...	...	...	...	...	
140886	16622.60	77790	26	4	2016	10	9	1	0	
1012848	10278.50	56	9	3	2018	10	42	0	0	
1496573	11370.80	0	26	3	2019	9	21	0	0	
1967631	61.17	0	1	4	2020	11	4	0	1	
107573	7682.65	8550	28	3	2016	11	36	0	0	

463939 rows × 11 columns

In [43]:
```
plt.scatter(y_prediction , y_test , color = "red")
```

Out[43]:
```
<matplotlib.collections.PathCollection at 0x2824676ad10>
```



In [44]:
```
error = mean_squared_error(y_test , y_prediction)
```

In [45]:
```
error
```

Out[45]:
```
70.07467011460399
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: