gq4gtos2o

January 23, 2025

[4]:
```
#importing the packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
```

[7]:
```
#loading the dataset
df=pd.read_csv("C:/Users/91703/OneDrive/Desktop/AAPL HISTORICAL DATA SET.csv")
```

[8]:
```
df.head()
```

[8]:

	Date	Close/Last	Volume	Open	High	Low
0	02/28/2020	$273.36	106721200	$257.26	$278.41	$256.37
1	02/27/2020	$273.52	80151380	$281.1	$286	$272.96
2	02/26/2020	$292.65	49678430	$286.53	$297.88	$286.5
3	02/25/2020	$288.08	57668360	$300.95	$302.53	$286.13
4	02/24/2020	$298.18	55548830	$297.26	$304.18	$289.23

[9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2518 entries, 0 to 2517
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Date         2518 non-null   object
 1    Close/Last  2518 non-null   object
 2    Volume      2518 non-null   int64 
 3    Open        2518 non-null   object
 4    High        2518 non-null   object
 5    Low         2518 non-null   object
dtypes: int64(1), object(5)
memory usage: 118.2+ KB
```

[10]:
```
df.tail()
```

[10]:

	Date	Close/Last	Volume	Open	High	Low
2513	03-05-2010	$31.2786	224647427	$30.7057	$31.3857	$30.6614
2514	03-04-2010	$30.1014	89591907	$29.8971	$30.1314	$29.8043
2515	03-03-2010	$29.9043	92846488	$29.8486	$29.9814	$29.7057
2516	03-02-2010	$29.8357	141486282	$29.99	$30.1186	$29.6771
2517	03-01-2010	$29.8557	137312041	$29.3928	$29.9286	$29.35

[11]:
```
df.describe()
```

[11]:

	Volume
count	2.518000e+03
mean	7.258009e+07
std	5.663113e+07
min	1.136205e+07
25%	3.053026e+07
50%	5.295469e+07
75%	9.861006e+07
max	4.624423e+08

[13]:
```
df.shape
```

[13]: (2518, 6)

[70]:
```
#Data Cleaning and Pre-Processing
#now we are changing the Date column into datetime formate and also we will␣
 ↪find out the outliers and anamolies
```

[71]:
```
#setting date as index
#we are setting data column to dataframe index  because we can easily do␣
 ↪date-based operations
```

[30]:
```
print(df.columns)
df.head()
```

```
Index([' Close/Last', ' Volume', ' Open', ' High', ' Low'], dtype='object')
```

[30]:

Date	Close/Last	Volume	Open	High	Low
2020-02-28	$273.36	106721200	$257.26	$278.41	$256.37
2020-02-27	$273.52	80151380	$281.1	$286	$272.96
2020-02-26	$292.65	49678430	$286.53	$297.88	$286.5
2020-02-25	$288.08	57668360	$300.95	$302.53	$286.13
2020-02-24	$298.18	55548830	$297.26	$304.18	$289.23

[35]:
```
df.set_index(df.iloc[:, 0],inplace=True)
```

[36]:
```
df.head()
```

[36]:

	Close/Last	Volume	Open	High	Low	Close/Last
$273.36	$273.36	106721200	$257.26	$278.41	$256.37	
$273.52	$273.52	80151380	$281.1	$286	$272.96	
$292.65	$292.65	49678430	$286.53	$297.88	$286.5	
$288.08	$288.08	57668360	$300.95	$302.53	$286.13	
$298.18	$298.18	55548830	$297.26	$304.18	$289.23	

[38]:
```
df.columns
```

[38]:
```
Index([' Close/Last', ' Volume', ' Open', ' High', ' Low'], dtype='object')
```

[42]:
```
df_values = df.values
train_size = int(len(df_values) * 0.8)
train_data, test_data = df_values[:train_size], df_values[train_size:]
```

[55]:
```
df = df.apply(lambda col: col.astype(str).str.replace('$', '').str.replace(',',␣
↪'')).astype(float)
```

[57]:
```
df_values = df.values
train_data_array = df_values[:-1, 0]  # features
y = df_values[1:, 0]  # target
train_data_array = train_data_array.reshape(-1, 1)
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor()
model.fit(train_data_array, y)
```

[57]:
```
RandomForestRegressor()
```

[58]:
```
test_data_array = df_values[-1:, 0].reshape(-1, 1)
prediction = model.predict(test_data_array)
print("Prediction:", prediction)
print("Actual:", df_values[-1, 0])
```

```
Prediction: [29.875189]
Actual: 29.8557
```

[65]:
```
from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error([df_values[-1, 0]], prediction)
```

[68]:
```
from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error([df_values[-1, 0]], [prediction[0]])
```

[69]:
```
import matplotlib.pyplot as plt
actual_values = df_values[:-1, 0]
predicted_values = model.predict(df_values[:-1, 0].reshape(-1, 1))
plt.plot(actual_values, label='Actual')
```

```
plt.plot(predicted_values, label='Predicted')
plt.plot([None for i in range(len(df_values)-1)] + [prediction[0]], label='Last␣
 ↪Predicted', marker='o')
plt.legend()
plt.show()
```