Netflix Stock Price Prediction with Machine Learning

Netflix is one of the most popular OTT streaming platforms. It offers a vast collection of television series and films and owns its productions known as Netflix Originals. People who are highly active in stock market investments always keep an eye on companies like Netflix because of its popularity. Now we are going to predict the stock prices of Netflix with machine learning.

In [1]:
```
import pandas as pd
import yfinance as yf
import datetime
from datetime import date, timedelta
```

In [2]:
```
today = date.today()

d1 = today.strftime("%Y-%m-%d")
end_date = d1
d2 = date.today() - timedelta(days=5000)
d2 = d2.strftime("%Y-%m-%d")
start_date = d2
```

In [3]:
```
data = yf.download('NFLX', start=start_date,end=end_date,progress=False)
```

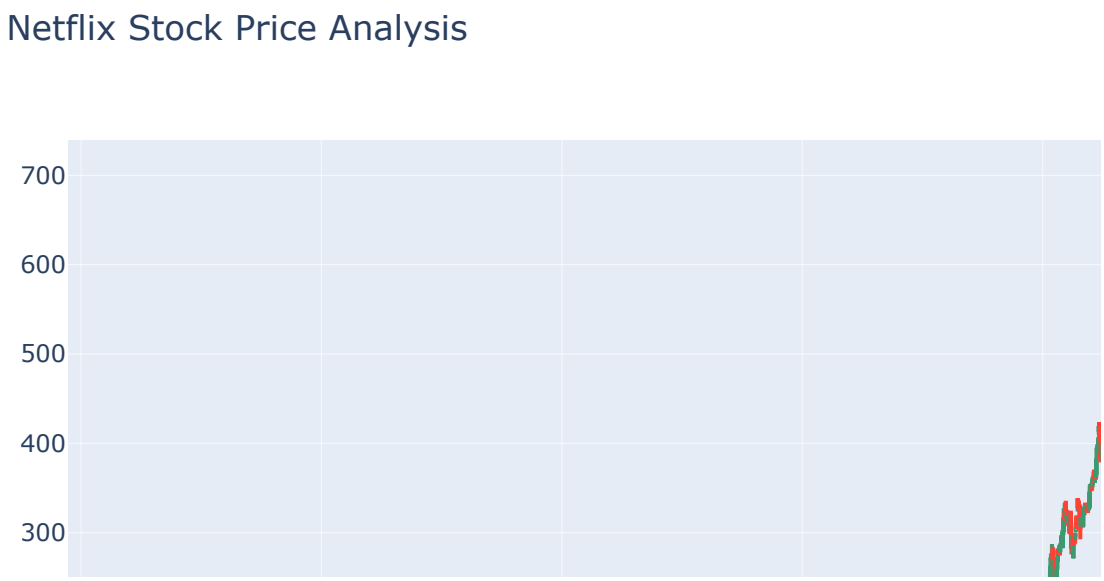
In [4]:
```
data["Date"] = data.index
data = data[["Date", "Open", "High", "Low", "Close", "Adj Close", "Volume"]]
data.reset_index(drop=True, inplace=True)
print(data.tail())
```

```
           Date        Open        High         Low       Close   Adj Close
\
3439 2023-07-26  424.200012  425.260010  415.589996  422.670013  422.670013
3440 2023-07-27  426.899994  427.519989  411.880005  413.170013  413.170013
3441 2023-07-28  415.559998  427.470001  413.760010  425.779999  425.779999
3442 2023-07-31  426.510010  439.130005  426.299988  438.970001  438.970001
3443 2023-08-01  437.369995  445.249908  431.420013  442.160004  442.160004

       Volume
3439  6009200
3440  6594500
3441  6424200
3442  6577400
3443  4434464
```

Now let's visualize the stock price data of Netflix by using a candlestick chart as it gives a clear picture of the increase and decrease of stock prices:

In [5]:
```
import plotly.graph_objects as go
figure = go.Figure(data=[go.Candlestick(x=data["Date"],
                                        open=data["Open"],
                                        high=data["High"],
                                        low=data["Low"],
                                        close=data["Close"])])
figure.update_layout(title = "Netflix Stock Price Analysis",
                     xaxis_rangeslider_visible=False)
figure.show()
```



Now let's have a look at the correlation of all the columns with the Close column:

In [6]:
```
correlation = data.corr()
print(correlation["Close"].sort_values(ascending=False))
```

```
Close        1.000000
Adj Close    1.000000
High         0.999798
Low          0.999781
Open         0.999540
Volume      -0.471121
Name: Close, dtype: float64
```

Training LSTM for Netflix Stock Price Prediction

Now I will train the LSTM neural network model for the task of Netflix stock price prediction using Python. Here I will first split the data into training and test sets:

In [7]:
```
x = data[["Open", "High", "Low", "Volume"]]
y = data["Close"]
x = x.to_numpy()
y = y.to_numpy()
y = y.reshape(-1, 1)
```

In [8]:
```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y,test_size=0.2,random_stat
```

Now I will prepare the LSTM neural network architecture:

In [9]:
```
from keras.models import Sequential
from keras.layers import Dense, LSTM
model = Sequential()
model.add(LSTM(128, return_sequences=True, input_shape= (xtrain.shape[1], 1)))
model.add(LSTM(64, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 4, 128)            66560

 lstm_1 (LSTM)               (None, 64)                49408

 dense (Dense)               (None, 25)                1625

 dense_1 (Dense)             (None, 1)                 26

=================================================================
Total params: 117,619
Trainable params: 117,619
Non-trainable params: 0
_________________________________________________________________
```

Now here's how we can train an LSTM neural network model for predicting the stock prices of Netflix using Python:

In [10]:
```
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(xtrain, ytrain, batch_size=1, epochs=30)
```

```
Epoch 1/30
2755/2755 [==============================] - 15s 4ms/step - loss: 7467.4570
Epoch 2/30
2755/2755 [==============================] - 13s 5ms/step - loss: 624.1381
Epoch 3/30
2755/2755 [==============================] - 14s 5ms/step - loss: 511.8553
Epoch 4/30
2755/2755 [==============================] - 14s 5ms/step - loss: 570.3959
Epoch 5/30
2755/2755 [==============================] - 13s 5ms/step - loss: 400.0124
Epoch 6/30
2755/2755 [==============================] - 12s 4ms/step - loss: 511.2804
Epoch 7/30
2755/2755 [==============================] - 13s 5ms/step - loss: 257.1790
Epoch 8/30
2755/2755 [==============================] - 12s 4ms/step - loss: 348.7872
Epoch 9/30
2755/2755 [==============================] - 13s 5ms/step - loss: 268.7497
Epoch 10/30
2755/2755 [==============================] - 12s 4ms/step - loss: 174.5135
Epoch 11/30
2755/2755 [==============================] - 14s 5ms/step - loss: 230.8017
Epoch 12/30
2755/2755 [==============================] - 14s 5ms/step - loss: 217.5912
Epoch 13/30
2755/2755 [==============================] - 14s 5ms/step - loss: 152.5512
Epoch 14/30
2755/2755 [==============================] - 14s 5ms/step - loss: 248.2458
Epoch 15/30
2755/2755 [==============================] - 15s 5ms/step - loss: 210.6700
Epoch 16/30
2755/2755 [==============================] - 14s 5ms/step - loss: 176.1658
Epoch 17/30
2755/2755 [==============================] - 14s 5ms/step - loss: 132.3721
Epoch 18/30
2755/2755 [==============================] - 14s 5ms/step - loss: 159.2290
Epoch 19/30
2755/2755 [==============================] - 14s 5ms/step - loss: 149.8534
Epoch 20/30
2755/2755 [==============================] - 14s 5ms/step - loss: 193.2155
Epoch 21/30
2755/2755 [==============================] - 13s 5ms/step - loss: 147.7409
Epoch 22/30
2755/2755 [==============================] - 13s 5ms/step - loss: 194.1890
Epoch 23/30
2755/2755 [==============================] - 14s 5ms/step - loss: 192.7515
Epoch 24/30
2755/2755 [==============================] - 13s 5ms/step - loss: 149.8256
Epoch 25/30
2755/2755 [==============================] - 13s 5ms/step - loss: 152.7651
Epoch 26/30
2755/2755 [==============================] - 14s 5ms/step - loss: 182.9975
Epoch 27/30
2755/2755 [==============================] - 13s 5ms/step - loss: 118.2971
Epoch 28/30
2755/2755 [==============================] - 14s 5ms/step - loss: 202.9468
Epoch 29/30
```

```
2755/2755 [==============================] - 13s 5ms/step - loss: 123.2539
Epoch 30/30
2755/2755 [==============================] - 14s 5ms/step - loss: 154.4326
```

Out[10]: <keras.callbacks.History at 0x248af50eb20>

Now let's test the model by giving the inputs according to the features that we used to train this model for predicting the final results:

In [11]:
```
import numpy as np
#features = [Open, High, Low, Adj Close, Volume]
features = np.array([[401.970001, 427.700012, 398.200012, 20047500]])
model.predict(features)
```

Out[11]:
```
array([[395.36432]], dtype=float32)
```

In [ ]: