Asset bubble solution -

Objective:

Help the investors by building a predictive analytics model or providing actionable analytical insights to this problem. Output should be a confidence score on whether stock market will crash in 2022 or not and if it does find an approximate time interval in which this can possibly occur.

Data analysis and Methodology–

Source - finance.yahoo.com/quote/NFLX/history?p=NFLX

Stock Price Prediction using Machine Learning -

Stock Price Prediction using machine learning is the process of predicting the future value of a stock traded on a stock exchange for reaping profits. With multiple factors involved in predicting stock prices, it is challenging to predict stock prices with high accuracy, and this is where machine learning plays a vital role.

Stock Price as a Time Series Data -

Treating stock data as time-series, one can use past stock prices (and other parameters) to predict the stock prices for the next day or week. Machine learning models such as Recurrent Neural Networks (RNNs) or LSTMs are popular models applied to predicting time series data such as weather forecasting, election results, house prices, and, of course, stock prices. The idea is to weigh out the importance of recent and older data and determine which parameters affect the “current” or “next” day prices the most. The machine learning model assigns weights to each market feature and determines how much history the model should look at to predict future stock prices.

Approaches to obtain desired outcome –

1. **Moving averages** -

$$SMA = \frac{P1 + P2 + ... + Pn}{N}$$

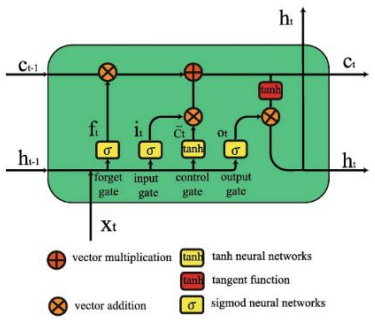
To begin with, we can use moving averages (or MA) to understand how the amount of history (or the number of past data points) considered affects the model's performance. A simple moving average computes the mean of the past N data points and takes this value as the predicted N+1 value.

Another moving average is the exponential moving average (EMA), giving more weight to the more recent samples. With this, we can look at more data points in the past and still not diminish the more recent trends in fluctuations.

$$EMA = P_t * k + EMA_{t-1} * (1 - k)$$

2. **Long Short Term Memory Network** –

LSTM is a Recurrent Neural Network that works on data sequences, learning to retain only relevant information from a time window. New information the network learns is added to a "memory" that gets updated with each timestep based on how significant the new sample seems to the model. Over the years, LSTM has revolutionized speech and handwriting recognition, language understanding, forecasting, and several other applications that have become the new normal today.



Analysis of the dataset –

	Date	Open	High	Low	Close	Adj Close	Volume
0	2021-11-01	689.059998	689.969971	676.539978	681.169983	681.169983	3110900
1	2021-11-02	683.109985	687.679993	673.820007	677.719971	677.719971	3888600
2	2021-11-03	677.270020	689.390015	677.270020	688.289978	688.289978	2334900
3	2021-11-04	685.890015	685.940002	665.500000	668.400024	668.400024	4865000
4	2021-11-05	663.969971	665.640015	645.010010	645.719971	645.719971	5283500

	Date	Open	High	Low	Close	Adj Close	Volume
247	2022-10-25	286.950012	297.589996	285.549988	291.019989	291.019989	15100700
248	2022-10-26	290.040009	305.630005	288.040009	298.619995	298.619995	15714100
249	2022-10-27	298.329987	305.209991	294.779999	296.940002	296.940002	14612600
250	2022-10-28	297.700012	301.190002	292.290009	295.720001	295.720001	9954900
251	2022-10-31	295.130005	297.619995	289.500000	291.880005	291.880005	7497800

We have taken an “Open-source licensed” dataset for our analysis. And it is freely available for anyone who wish to use it.

As with any other machine learning model, it is always good to normalize or rescale the data within a fixed range when dealing with real data. This will avoid features with larger numeric values to unjustly interfere and bias the model and help achieve rapid convergence.

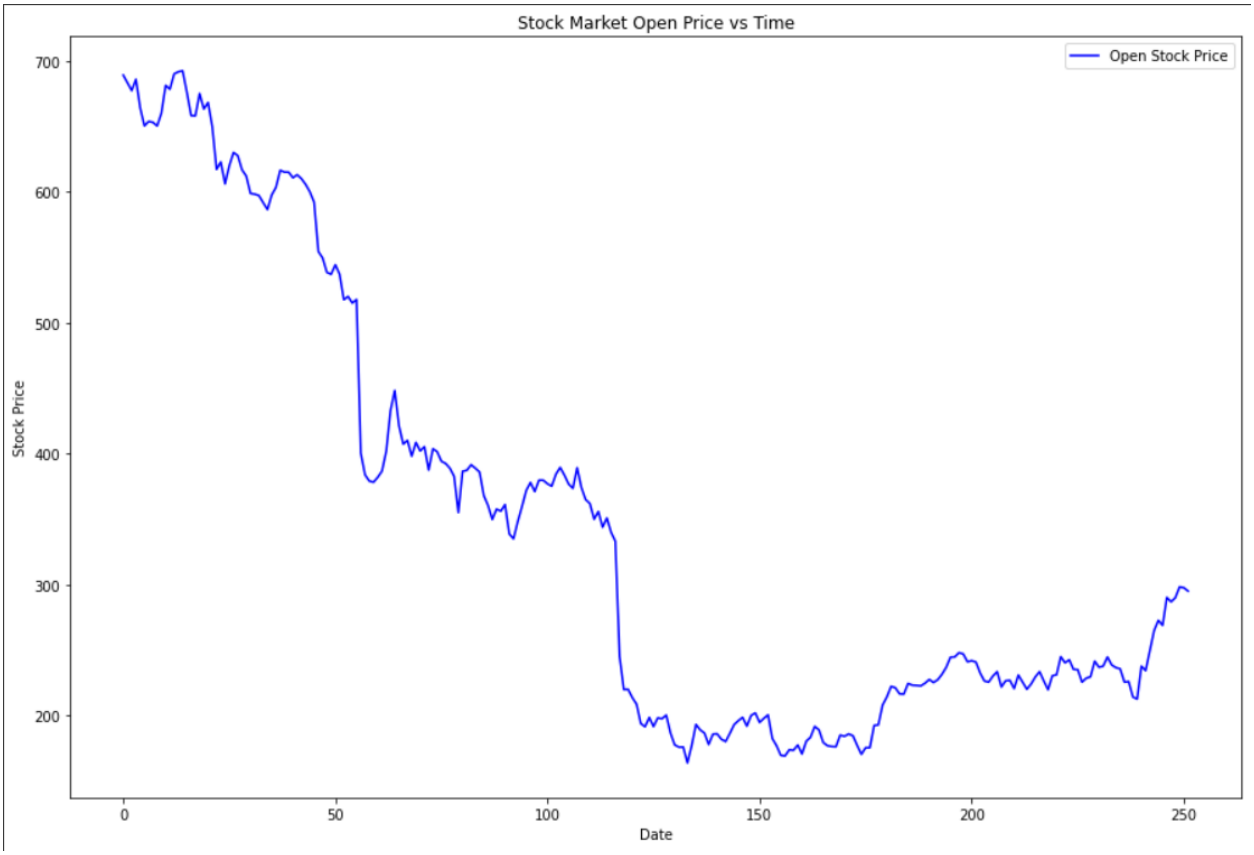
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(train_data['Open'].values.reshape(-1,1))
```

Scikit-learn also provides a popular MinMaxScaler preprocessing module. However, considering the context, stock prices might max out or minimise on different days, and using those values to influence others might not be great. The change in values from using either of these methods would not be much, so we stick to StandardScaler.

```
prediction_days = 30

x_train = []
y_train = []

for x in range(prediction_days, len(scaled_data)-10):      ######
    x_train.append(scaled_data[x-prediction_days:x, 0])
    y_train.append(scaled_data[x+10, 0])      ###### predict 10 days after

x_train, y_train = np.array(x_train), np.array(y_train)
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
```

We split our data into training and testing sets. Shuffling is not permitted in time-series datasets. In the beginning, we take two steps worth of past data to predict the current value. Thus, the model will look at yesterday's and today's values to predict today's closing price.

```
# Graph of Stock Market Open Price vs Time
plt.figure(figsize=(15,10))
plt.plot(df['Open'], color='blue', label='Open Stock Price')
plt.title('Stock Market Open Price vs Time')
plt.xlabel('Date')
plt.ylabel('Stock Price')
plt.legend()
plt.show()
```



We have used the Sequential and LSTM modules provided by Tensorflow Keras to build a simple, single-unit LSTM model.

```
def LSTM_model():

    model = Sequential()
    model.add(LSTM(units = 50, return_sequences = True, input_shape = (x_train.shape[1],1)))
    model.add(Dropout(0.2))
    model.add(LSTM(units = 50, return_sequences = True))
    model.add(Dropout(0.2))
    model.add(LSTM(units = 50))
    model.add(Dropout(0.2))
    model.add(Dense(units=1))

    return model
```

```
model = LSTM_model()
model.summary()
model.compile(optimizer='adam', loss='mean_squared_error', metrics = ['accuracy'])
```

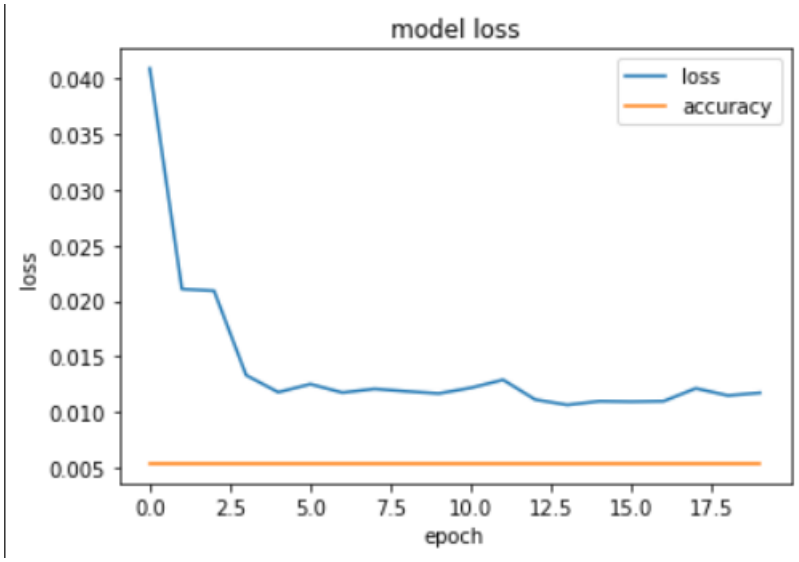
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 30, 50)            10400

 dropout (Dropout)           (None, 30, 50)            0

 lstm_1 (LSTM)               (None, 30, 50)            20200

 dropout_1 (Dropout)         (None, 30, 50)            0

 lstm_2 (LSTM)               (None, 50)                20200

 dropout_2 (Dropout)         (None, 50)                0

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
_________________________________________________________________
```

Now we can fit this simple model to the training data.

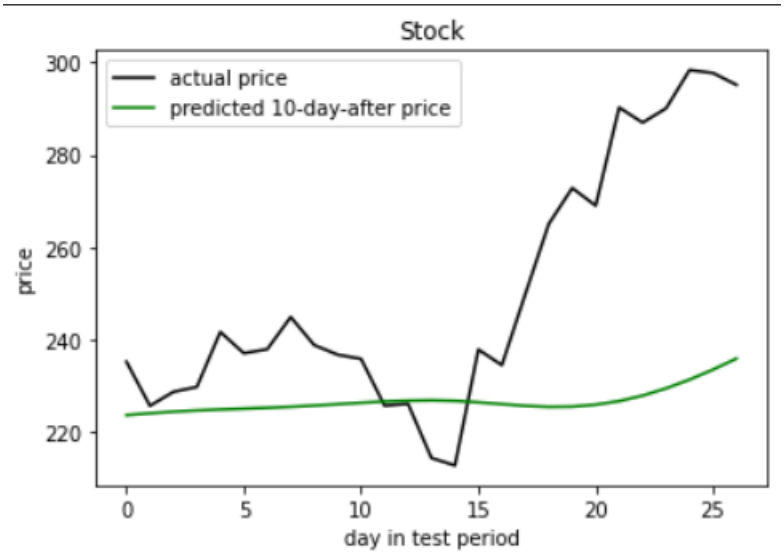
```
checkpointer = ModelCheckpoint(filepath = 'weights_best.hdf5', verbose = 1, save_best_only = True)
his=model.fit(x_train,y_train,epochs=20,batch_size=32,callbacks=[checkpointer])
```

```
6/6 [==============================] - 7s 56ms/step - loss: 0.0409 - accuracy: 0.0054
Epoch 2/20
6/6 [==============================] - ETA: 0s - loss: 0.0211 - accuracy: 0.0054WARNING:tensorflow:Can save best model only with val_loss available, skipping.
6/6 [==============================] - 0s 51ms/step - loss: 0.0211 - accuracy: 0.0054
Epoch 3/20
6/6 [==============================] - ETA: 0s - loss: 0.0209 - accuracy: 0.0054WARNING:tensorflow:Can save best model only with val_loss available, skipping.
6/6 [==============================] - 0s 53ms/step - loss: 0.0209 - accuracy: 0.0054
Epoch 4/20
6/6 [==============================] - ETA: 0s - loss: 0.0133 - accuracy: 0.0054    WARNING:tensorflow:Can save best model only with val_loss available, skipping.
6/6 [==============================] - 0s 51ms/step - loss: 0.0133 - accuracy: 0.0054
Epoch 5/20
5/6 [========================>.....] - ETA: 0s - loss: 0.0117 - accuracy: 0.0063    WARNING:tensorflow:Can save best model only with val_loss available, skipping.
6/6 [==============================] - 0s 47ms/step - loss: 0.0118 - accuracy: 0.0054
Epoch 6/20
5/6 [========================>.....] - ETA: 0s - loss: 0.0127 - accuracy: 0.0063    WARNING:tensorflow:Can save best model only with val_loss available, skipping.
6/6 [==============================] - 0s 48ms/step - loss: 0.0125 - accuracy: 0.0054
Epoch 7/20
5/6 [========================>.....] - ETA: 0s - loss: 0.0119 - accuracy: 0.0000e+00WARNING:tensorflow:Can save best model only with val_loss available, skipping.
6/6 [==============================] - 0s 52ms/step - loss: 0.0118 - accuracy: 0.0054
Epoch 8/20
5/6 [========================>.....] - ETA: 0s - loss: 0.0117 - accuracy: 0.0063    WARNING:tensorflow:Can save best model only with val_loss available, skipping.
6/6 [==============================] - 0s 52ms/step - loss: 0.0121 - accuracy: 0.0054
Epoch 9/20
6/6 [==============================] - ETA: 0s - loss: 0.0119 - accuracy: 0.0054WARNING:tensorflow:Can save best model only with val_loss available, skipping.
6/6 [==============================] - 0s 50ms/step - loss: 0.0119 - accuracy: 0.0054
Epoch 10/20
5/6 [========================>.....] - ETA: 0s - loss: 0.0115 - accuracy: 0.0063WARNING:tensorflow:Can save best model only with val_loss available, skipping.
6/6 [==============================] - 0s 52ms/step - loss: 0.0117 - accuracy: 0.0054
```

Given the simplicity of the model and the data, we note that the loss reduction stagnates after only 20 epochs. We can observe this by plotting the training loss against the number of epochs, and LSTM does not learn much after 10-20 epochs.



Thus, we can see that LSTM can emulate the trends of the stock prices to a certain extent. Based on the recent dip in prices, it has also fit the dropping curve well.

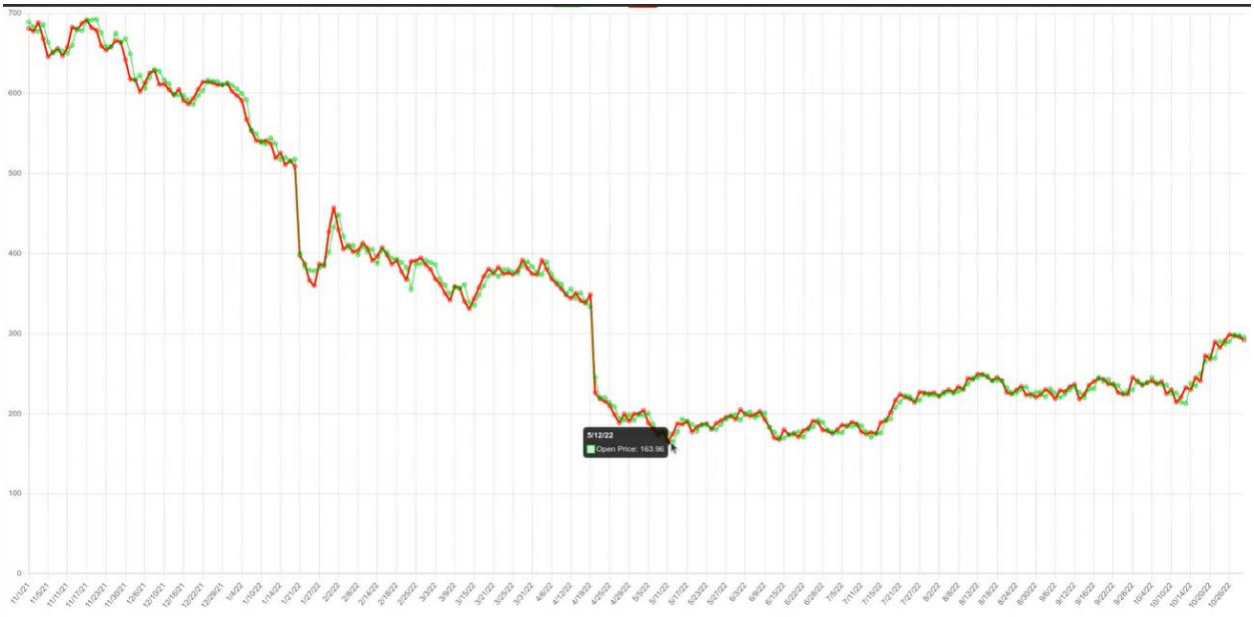


Expected outcome:

So in the case of predicting stock price data, SMA and EMA have similar performance and are far behind when compared to LSTMs. We can improve our

LSTM model by finetuning the hyperparameters such as the number of cells, batch size, or the loss function. However, using data beyond 2019 (up to 5-10 years' worth of data) would greatly help the model.

Definition of Crash:



By deduced analysis of the whole prediction model. It’s showing the lowest point on 5/12/2022. Thus, we conclude that this interval is when a crash may happen.