Share price Predictor

Shahbaz S

Vishnu M

Dhanush V

Abstract:

Stock market prediction is the act of trying to determine the future value of a company stock. The successful prediction of a stock's future price could yield significant profit. There are multiple ways for stock predictions :

1. Fundamental analysis:
   This method is trying to find out the true value of a stock, which then can be compared with the value it is being traded with on stock markets and therefore finding out whether the stock on the market is undervalued or not. Based on analysis the stock is identified as under valued or overvalued hence predicting the buy/sell
2. Technical analysis:
   Technical analysts or chartists are not concerned with any of the company's fundamentals. They seek to determine the future price of a stock based solely on the trends of the past price (a form of time series analysis). Multiple patterns will be used to predict the future stock price
3. Machine Learning:
   With the advent of the digital computer, stock market prediction has since moved into the technological realm. The most prominent technique involves the use of artificial neural networks (ANNs) and Genetic Algorithms(GA)

Problem Statement

- The challenge of this project is to accurately predict the future closing value of a given company’s stock across a given period in the future
- GOALS:
  1. Explore stock prices
  2. Implement a suitable model
  3. Compare the results and submit the report

Data Collection

- Data is collected from Yahoo Finance
- For our project we have chosen stock price of **TATA POWER.**
- Data is downloaded from yahoo Finance website as csv file from here.

Data Cleanup

- CSV file is uploaded to code as dataframe and it is analyzed with df.info() and df.head()
- We saw some null values in data
- Upon inspection we found it to be date at which dividend was distributed, hence these null rows are removed as data cleanup step.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6873 entries, 0 to 6872
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       6873 non-null   object
 1   Open       6862 non-null   float64
 2   High       6862 non-null   float64
 3   Low        6862 non-null   float64
 4   Close      6862 non-null   float64
 5   Adj Close  6862 non-null   float64
 6   Volume     6862 non-null   float64
dtypes: float64(6), object(1)
memory usage: 376.0+ KB
None
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	1996-01-01	11.580205	11.662232	11.488529	11.628456	5.920809	33160.0
1	1996-01-02	11.628456	11.739433	11.396852	11.483704	5.847106	176162.0
2	1996-01-03	11.483704	11.599506	11.392027	11.411327	5.810253	104661.0
3	1996-01-04	11.411327	11.387202	11.155598	11.242449	5.724268	77718.0
4	1996-01-05	11.242449	11.483704	11.097697	11.464403	5.837279	113469.0

Jun 16, 2022	222.40	223.30	209.00	209.80	209.80	22,486,470
Jun 15, 2022	221.00	222.30	218.30	218.95	218.95	11,820,899
Jun 15, 2022	**1.75** Dividend					
Jun 14, 2022	214.70	224.40	214.65	220.40	218.65	16,235,651
Jun 13, 2022	223.45	226.35	218.90	219.50	217.76	19,190,225

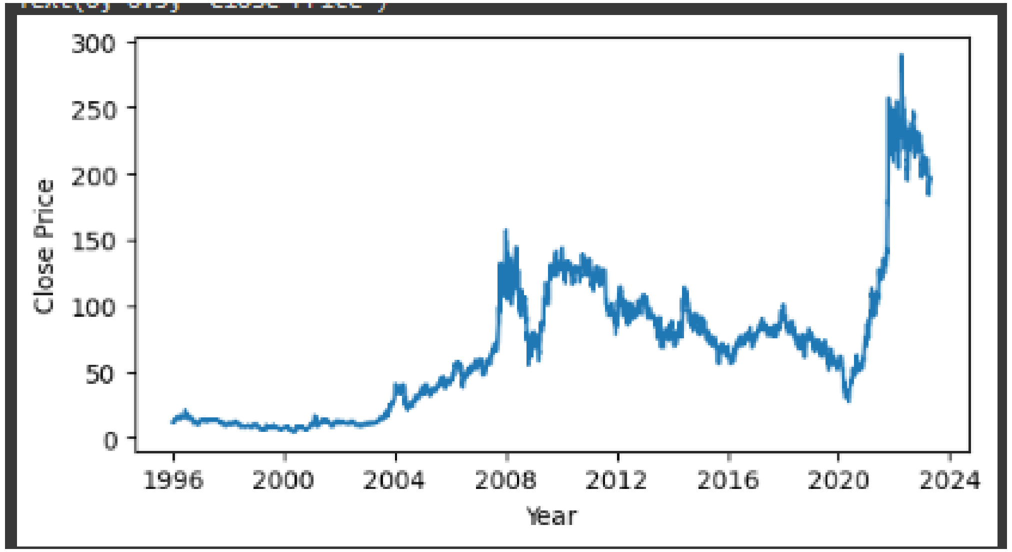
Feature set selection

- From available data points we have selected following tuple(s) of columns to be our data set and at end we have compared the results for each run to **predict closing price** of stock but ideally our code can be run to predict any column of data set (e.g., High, Low, Volume ..)
  - Run 1 : feature set = [Closing] *>> Single input*
  - Run 2 : feature set = [Closing, High] *>> Highly correlated Columns*
  - Run 3 : feature set = [Closing, Volume] *>> Poorly correlated Columns*

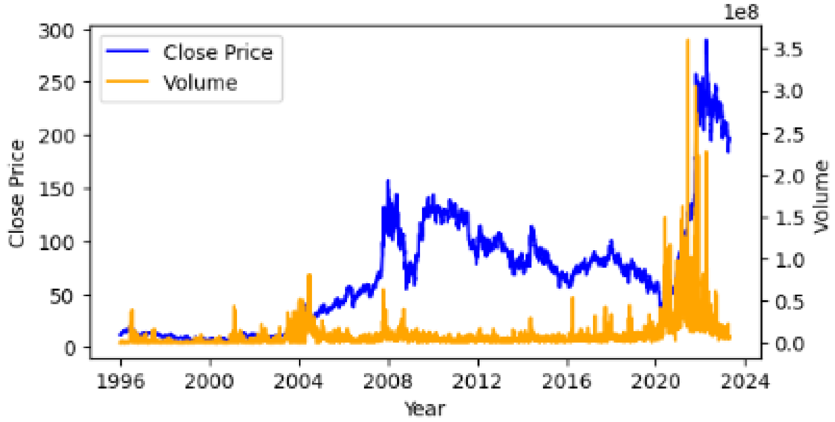
```
Saving TATAPOWER.NS.csv to TATAPOWER.NS (7).csv
Correlation between Open & Close: 0.9992904531180177
Correlation between Volume & Close: 0.30576907818730586
Correlation between High & Close: 0.9996803744618018
```

Data Visualization

Run 1 : Single feature

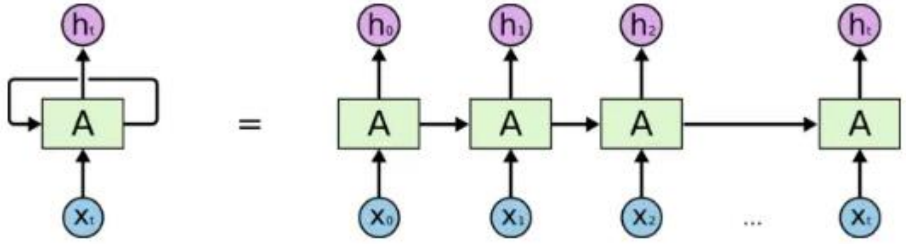


Run 2 & 3 : Two features



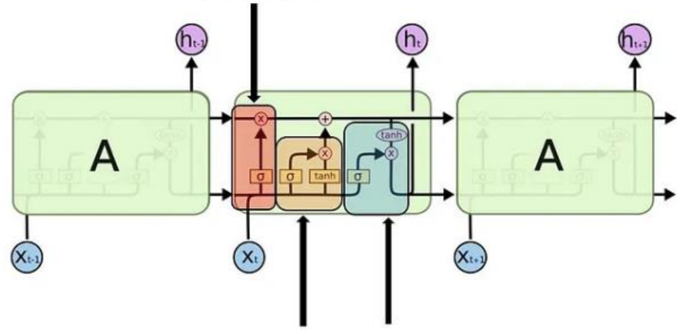
Model Selection

- Recurrent Neural Nets (RNN) are used specifically for sequence and pattern learning
- But Recurrent Neural Nets have vanishing Gradient descent problem which does not allow it to learn from past data as was expected.



An unrolled recurrent neural network.

- Long Short-Term Memory (LSTM) networks are a modified version of RNN, which makes it easier to remember past data in memory.
- The vanishing gradient problem of RNN is resolved here.
- LSTM is well-suited to classify, process and predict time series given time lags of unknown duration, It trains the model by using back-propagation



Model Training & Fitting

- Input Parameters

Preprocessing and Normalization

  Neural Network Architecture : LSTM

  Number of Layers (how many layers of nodes in the model; used 3)

  Number of Nodes (how many nodes per layer; used 50)

- Training Parameters

  Training / Test Split : 80 / 20

- Batch Size (kept at 1)
- Optimizer Function

  MSE and “Adam” Optimizer)

- Epochs (kept at 1)
- Model Fitting

```
#Build and train the LSTM model

lstm_model=Sequential()
lstm_model.add(LSTM(units=50,return_sequences=True,input_shape=(x_train_data.shape[1],1)))
lstm_model.add(LSTM(units=50))
lstm_model.add(Dense(1))

lstm_model.compile(loss='mean_squared_error',optimizer='adam')
lstm_model.fit(x_train_data,y_train_data,epochs=1,batch_size=1,verbose=2)
```

Model Prediction

- Predicting the close values from model by loading the test values

```
[ ]  X_test=[]

     X_test=np.array(scaled_data_test)
     print("Testing Data set Shape:",X_test.shape)

     Testing Data set Shape: (1373, 2)
```

```
[ ]  #Make a prediction using the LSTM model

     predicted_closing_price=lstm_model.predict(X_test)
     z = np.zeros((len(predicted_closing_price),1))

     predicted_closing_price = np.append(predicted_closing_price, z, axis=1)
     #print(predicted_closing_price)
     predicted_closing_price=scale.inverse_transform(predicted_closing_price)

     predicted_closing_price = np.delete(predicted_closing_price, 1, 1)  # delete second column of zeros
     print("Predicted Closing Price Sahep:",predicted_closing_price.shape)

     43/43 [==============================] - 1s 2ms/step
     Predicted Closing Price Sahep: (1373, 1)
```

Model Prediction

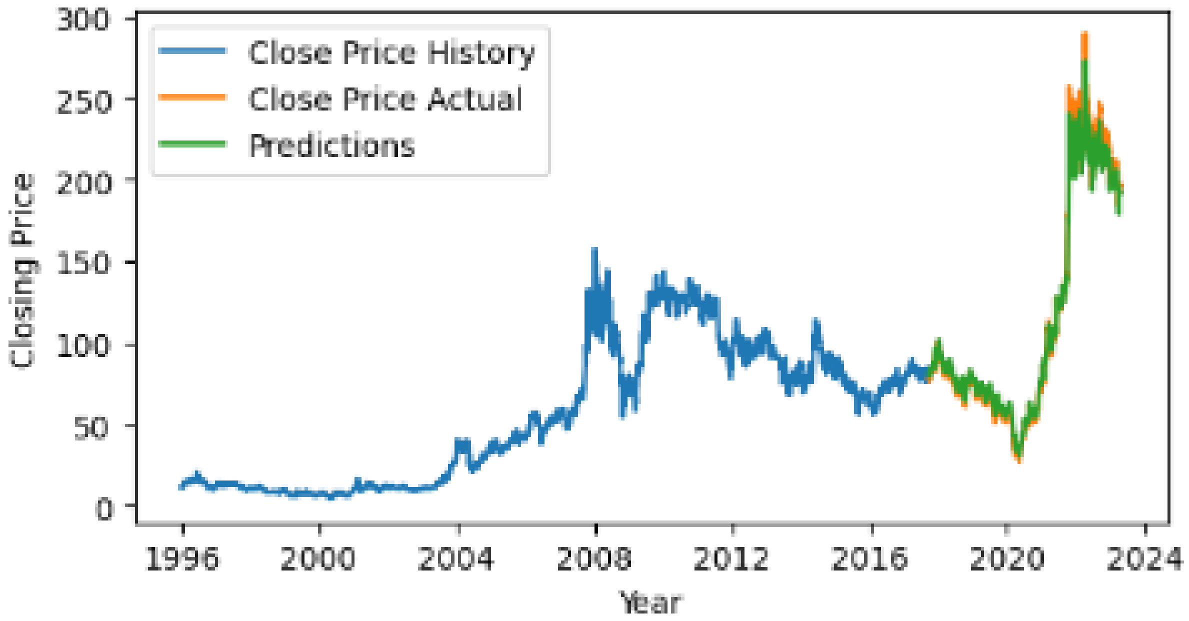
- Predicting the close values from model by loading the test values

```
[ ] X_test=[]

    X_test=np.array(scaled_data_test)
    print("Testing Data set Shape:",X_test.shape)

    Testing Data set Shape: (1373, 2)
```

```
[ ] #Make a prediction using the LSTM model

    predicted_closing_price=lstm_model.predict(X_test)
    z = np.zeros((len(predicted_closing_price),1))

    predicted_closing_price = np.append(predicted_closing_price, z, axis=1)
    #print(predicted_closing_price)
    predicted_closing_price=scale.inverse_transform(predicted_closing_price)

    predicted_closing_price = np.delete(predicted_closing_price, 1, 1)  # delete second column of zeros
    print("Predicted Closing Price Sahep:",predicted_closing_price.shape)

    43/43 [==============================] - 1s 2ms/step
    Predicted Closing Price Sahep: (1373, 1)
```

Results Run -1

- For run 1 we had only used Closing price of stock as training input
- Training Model took around **260** seconds
- Avg differences between actual & predicted : **-2.156**
- Variance between actual & predicted : **17.009**




```
Avg differences between actual & predicted : -2.156
Variance between actual & predicted : 17.009
```

Results Run -2

- For run 2 we had used Closing price and High of stock as training input
- Training Model took around **21** seconds
- Avg differences between actual & predicted : **-0.097**
- Variance between actual & predicted : **0.405**


```
Avg differences between actual & predicted : -0.097
Variance between actual & predicted : 0.405
```

Results Run -3

- For run 3 we had used Closing price and Volume of stock as training input
- Training Model took around 26 seconds
- Avg differences between actual & predicted : **0.51**
- Variance between actual & predicted : **1.961**


```
Avg differences between actual & predicted : 0.51
Variance between actual & predicted : 1.961
```

Conclusion

- In this project we were able to predict stock price of a company using LSTM and we saw that using multiple features improves our average error between predicted and actual value.
- Using highly correlated feature set tuples gave us slightly better results.
- In this project we did not take into account of real time news which could suddenly impact stock price (e.g. dip in recent Adani stocks)

Reference Links

- https://github.com/Mohit-Vernekar/stock-price-predictor
- https://en.wikipedia.org/wiki/Stock_market_prediction
- https://www.nirmalbang.com/knowledge-center/stock-market-terminology.html
- https://finance.yahoo.com/quote/ADANIENT.NS/history?p=ADANIENT.NS
- Understanding RNN and LSTM. What is Neural Network? | by Aditi Mittal | Medium