PROJECT REPORT

TABLE OF CONTENT
Data Source
Libraries Used
Data Preprocessing Steps
Models Used
Polynomial Regression
LSTM

Data Source :

- Kaggle (click here)
- Dataset Contains Stock prices from 2002-02-18 to 2021-04-30
- 15 Features in Dataset :

```
0   Date                4774 non-null   object
1   Symbol              4774 non-null   object
2   Series              4774 non-null   object
3   Prev Close          4774 non-null   float64
4   Open                4774 non-null   float64
5   High                4774 non-null   float64
6   Low                 4774 non-null   float64
7   Last                4774 non-null   float64
8   Close               4774 non-null   float64
9   VWAP                4774 non-null   float64
10  Volume              4774 non-null   int64
11  Turnover            4774 non-null   float64
12  Trades              2456 non-null   float64
13  Deliverable Volume  4758 non-null   float64
14  %Deliverble         4758 non-null   float64
```

Libraries Used :

- Numpy
- Pandas
- Matplotlib
- Plotly
- Sklearn
- Keras

Data Preprocessing:

1. Changed Datatypes : date column to pandas datetime64
2. Dataset of parent company Airtel contains data of two sub companies 'BHARTI' and 'BHARTIARTL'
    - ‘Symbol’ contains 'BHARTI' and 'BHARTIARTL'
3. Extracted ‘BHARTI DF’ in variable ‘bharti_telecom’

4. Now all data processing is done on 'bharti_telecom DF'
5. Missing Values
6. Outliers : User defined functions to remove outliers
7. Feature Selection and Scaling

Model Used :

1. Polynomial Regression:

- Reason(Why?) : I have used this model because it can capture nonlinear relationships between variables. In the case of stock price prediction, it is often the case that the relationship between the predictors and the response variable is not linear. By using a polynomial regression, you can fit a curve to the data that better captures the underlying relationship, potentially improving the accuracy of your predictions.

```
In [41]: polyreg=make_pipeline(PolynomialFeatures(degree),LinearRegression())
         polyreg.fit(x_train,y_train)
```

```
Out[41]:
▸ Pipeline
  ▸ PolynomialFeatures
  ▸ LinearRegression
```

- Accuracy :

```
In [43]: r2_score(y_test,prediction)
Out[43]: 0.9999625596635408

In [44]: mse(y_test,prediction)
Out[44]: 0.6192539113495068
```

2. LSTM :

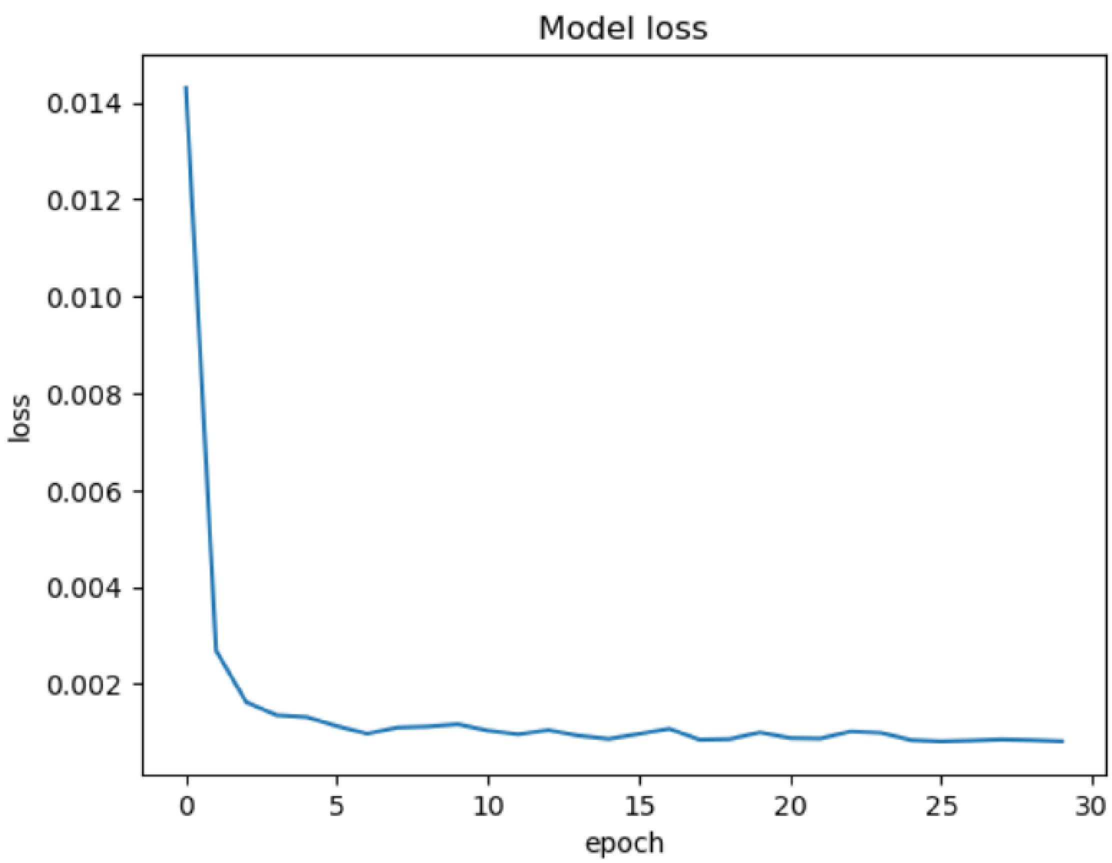
- Reason(Why?): I have used LSTM because LSTM is well-suited for time-series data analysis and prediction. LSTM networks have the ability to remember past information for long periods of time and selectively forget information that is no longer relevant, making them particularly effective for analyzing sequences of data with complex dependencies. For stock price prediction, LSTM can be used to analyze historical data and identify patterns that may be indicative of future trends in the stock market. By training the network on a dataset of past stock prices and associated market conditions, the LSTM can learn to recognize patterns and make predictions about future stock prices.

A. LSTM for single variable 'Closing Stock Price':

- Data Preparation for the model
- User defined function to extract input and output feature for LSTM
- User defined function to split the test train data
- Model:

```
model = Sequential()

model.add((LSTM(units=50,return_sequences=True,input_shape=(100, 1))))
model.add(Dropout(0.2))

model.add(LSTM(units=50,return_sequences=True))
model.add(Dropout(0.2))

model.add(LSTM(units=50,return_sequences=True))
model.add(Dropout(0.2))

model.add(LSTM(units=50,return_sequences=False))
model.add(Dropout(0.2))

model.add(Dense(units=1))
model.compile(optimizer='adam',loss='mean_squared_error')
```

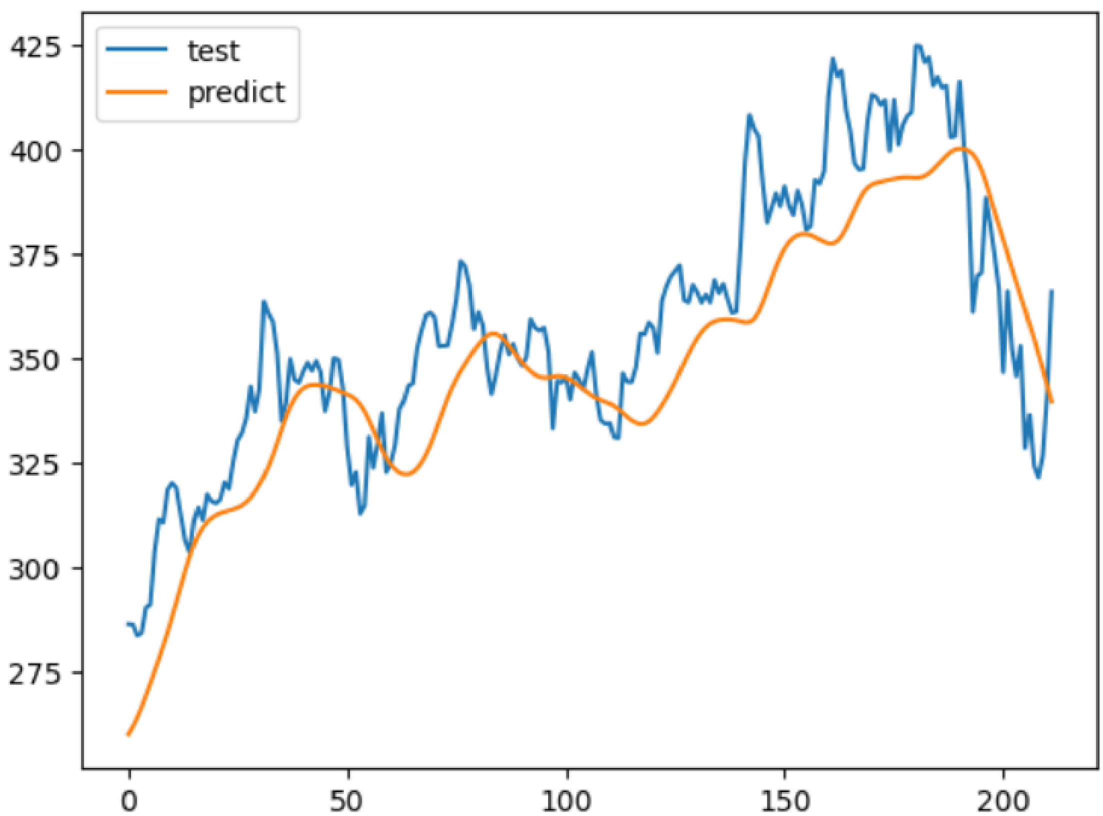
- Loss function:



Accuracy ¶

```
In [59]: import math
         math.sqrt(mse(ly_test,predict))

Out[59]: 18.63991957903472
```

- Prediction:



B. LSTM for multiple(11) features and 'closing stock price' as output feature :

- Data Preparation for the model

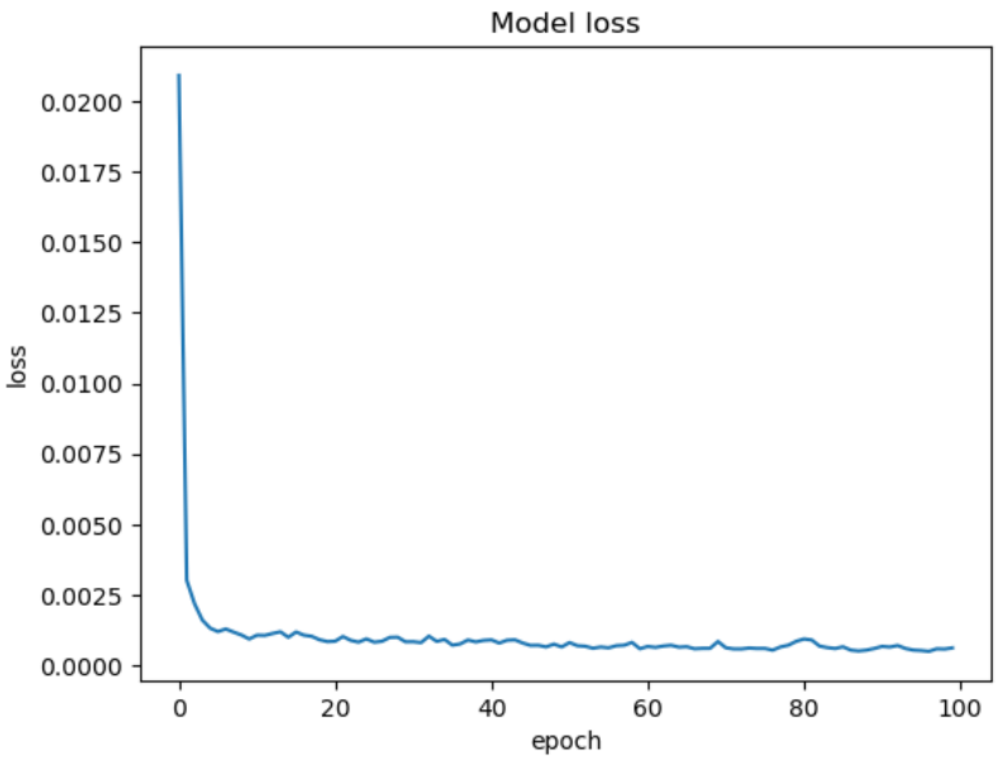
- User defined function to extract input and output feature for LSTM
- User defined function to split the test train data

- Model:

In [73]:
```
model1.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
click to scroll output; double click to hide  (None, 100, 50)  12400

 dropout_4 (Dropout)         (None, 100, 50)           0

 lstm_5 (LSTM)               (None, 100, 50)           20200

 dropout_5 (Dropout)         (None, 100, 50)           0

 lstm_6 (LSTM)               (None, 100, 50)           20200

 dropout_6 (Dropout)         (None, 100, 50)           0

 lstm_7 (LSTM)               (None, 50)                20200

 dropout_7 (Dropout)         (None, 50)                0

 dense_1 (Dense)             (None, 1)                 51

=================================================================
Total params: 73,051
Trainable params: 73,051
Non-trainable params: 0
_________________________________________________________________
```

- Model Loss:



Performance and Accuracy

In [111]:
```
import math
math.sqrt(mse(ly_test,predict))
```

Out[111]: 63.465944671847545

In [112]:
```
r2_score(ly_test,predict)
```

Out[112]: -2.922142397959334

- Prediction: