Stock Price Prediction

A COURSE PROJECT REPORT

By

AKASH KUMAR PANDIT (RA2011027010122)

SHIVANSH SHARMA (RA2011027010081)

RUDRANSH PANDEY (RA2011027010071)

Under the guidance of

Ms.D.Hemavathi
Assistant professor

Department of Data Science and Business Systems

In partial fulfilment for the Course

of

18CSE396T – DATA SCIENCE

in

Department of Data Science and Business Systems

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur, Chengalpattu District

NOVEMBER 2022

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this mini project report " **Stock Price Prediction**" is the bonafide work of Akash kumar pandit(RA2011027010122), Shivansh Sharma (RA2011027010081) , Rudransh Pandey(RA2011027010071) who carried out the project work under my supervision.

Ms.D.Hemavathi
Assistant professor
Department of Data Science and Business Systems
SRM institute of science and technology

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our Registrar **Dr. S. Ponnusamy**, for his encouragement.

We express our profound gratitude to our Dean of College of Engineering and Technology, **Dr. T. V.Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to the Chairperson, School of Computing **Dr. Revathi Venkataraman**, for imparting confidence to complete my course project

We wish to express my sincere thanks to Course Audit Professor **Dr. Annapurani Panaiyappan**, Professor and Head, Department of Networking and Communications and Course Coordinators for their constant encouragement and support.

We are highly thankful to our course project faculty , Assistant Professor **Ms.D.Hemavathi** , Department of DSBS for her assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our HoD, Professor **Dr. M. Lakshmi ,** Department of DSBS and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

18CSE396T
DATA SCIENCE PROJECT

On

Stock Price Prediction

INDEX

ABSTRACT

In Stock Market Prediction, the aim is to predict the future value of the financial stocks of a company. The recent trend in stock market prediction technologies is the use of machine learning which makes predictions based on the values of current stock market indices by training on their previous values. Machine learning itself employs different models to make prediction easier and authentic. The project focuses on the use of LSTM based Machine learning algorithm to predict stock values. Factors considered are open, close, low, high and volume. The prediction is done based on the close values.

Datasets used:

To build the stock price prediction model, we will use the Apple dataset. We will get all values of its open, close, high, low and volume from the keras API function and receive data from yahoo finance. The model will predict data based on the previous data provided and shift in the array as per requirement and new day values.

Jupyter Notebook Code:

jupyter ShareMarketPricePrediction Last Checkpoint: Last Tuesday at 17:32 (autosaved)

File Edit View Insert Cell Kernel Help

In [8]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pandas_datareader as data
```

In [9]:
```
start ='2010-01-01'
end = '2019-12-31'

df=data.DataReader('AAPL','yahoo', start, end )
df.head()
```

Out[9]:

Date	High	Low	Open	Close	Volume	Adj Close
2009-12-31	7.619643	7.520000	7.611786	7.526071	352410800.0	6.434927
2010-01-04	7.660714	7.585000	7.622500	7.643214	493729600.0	6.535086
2010-01-05	7.699643	7.616071	7.664286	7.656429	601904800.0	6.546382
2010-01-06	7.686786	7.526786	7.656429	7.534643	552160000.0	6.442254
2010-01-07	7.571429	7.466071	7.562500	7.520714	477131200.0	6.430346

In [10]:
```
df=df.reset_index()
df.head()
```

Out[10]:

	Date	High	Low	Open	Close	Volume	Adj Close
0	2009-12-31	7.619643	7.520000	7.611786	7.526071	352410800.0	6.434927
1	2010-01-04	7.660714	7.585000	7.622500	7.643214	493729600.0	6.535086
2	2010-01-05	7.699643	7.616071	7.664286	7.656429	601904800.0	6.546382
3	2010-01-06	7.686786	7.526786	7.656429	7.534643	552160000.0	6.442254
4	2010-01-07	7.571429	7.466071	7.562500	7.520714	477131200.0	6.430346

jupyter ShareMarketPricePrediction Last Checkpoint: Last Tuesday at 17:32 (autosaved)

File Edit View Insert Cell Kernel Help

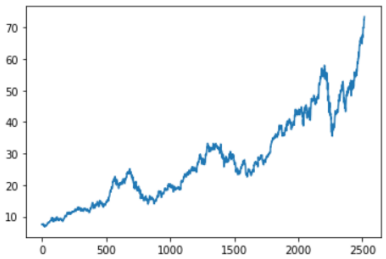
In [11]:
```
df=df.drop(['Date','Adj Close'],axis=1)
df.head()
```

Out[11]:

	High	Low	Open	Close	Volume
0	7.619643	7.520000	7.611786	7.526071	352410800.0
1	7.660714	7.585000	7.622500	7.643214	493729600.0
2	7.699643	7.616071	7.664286	7.656429	601904800.0
3	7.686786	7.526786	7.656429	7.534643	552160000.0
4	7.571429	7.466071	7.562500	7.520714	477131200.0

In [12]:
```
plt.plot(df.Close)
```

Out[12]:
```
[<matplotlib.lines.Line2D at 0x1d590542bc8>]
```



In [13]:
```
df
```

jupyter ShareMarketPricePrediction Last Checkpoint: Last Tuesday at 17:32 (autosaved)

Logout

File Edit View Insert Cell Kernel Help Not Trusted Python 3 (ipykernel)

Run Code

Out[13]:

	High	Low	Open	Close	Volume
0	7.619643	7.520000	7.611786	7.526071	352410800.0
1	7.660714	7.585000	7.622500	7.643214	493729600.0
2	7.699643	7.616071	7.664286	7.656429	601904800.0
3	7.686786	7.526786	7.656429	7.534643	552160000.0
4	7.571429	7.466071	7.562500	7.520714	477131200.0
...	...	...	...	...	...
2512	71.222504	70.730003	71.172501	71.067497	48478800.0
2513	72.495003	71.175003	71.205002	72.477501	93121200.0
2514	73.492500	72.029999	72.779999	72.449997	146266000.0
2515	73.172501	71.305000	72.364998	72.879997	144114400.0
2516	73.419998	72.379997	72.482498	73.412498	100805600.0

2517 rows × 5 columns

In [14]:
```
ma100=df.Close.rolling(100).mean()
ma100
```

Out[14]:
```
0             NaN
1             NaN
2             NaN
3             NaN
4             NaN
          ...
2512    59.401700
2513    59.643125
2514    59.875125
2515    60.106325
2516    60.331875
```

jupyter ShareMarketPricePrediction Last Checkpoint: Last Tuesday at 17:32 (autosaved)

Logout

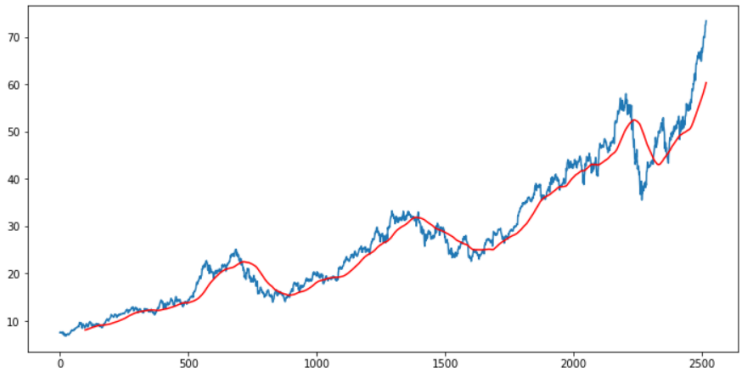
File Edit View Insert Cell Kernel Help Not Trusted Python 3 (ipykernel)

Run Code

```
Name: Close, Length: 2517, dtype: float64
```

In [15]:
```
plt.figure(figsize = (12,6))
plt.plot(df.Close)
plt.plot(ma100, 'r')
```

Out[15]:
```
[<matplotlib.lines.Line2D at 0x1d590934d88>]
```



In [16]:
```
ma200=df.Close.rolling(200).mean()
ma200
```

Out[16]:
```
0             NaN
1             NaN
2             NaN
```

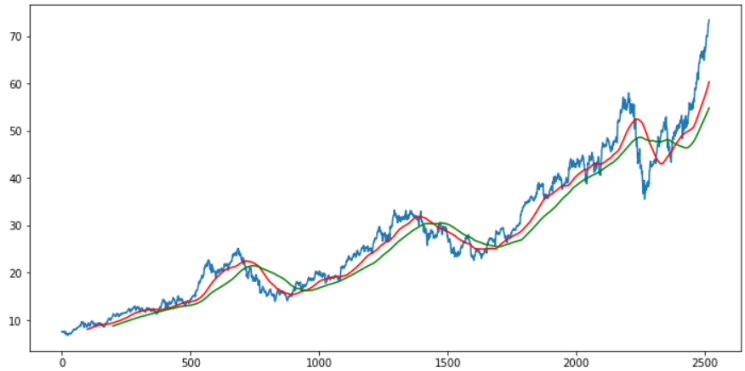
jupyter ShareMarketPricePrediction Last Checkpoint: Last Tuesday at 17:32 (autosaved)

File Edit View Insert Cell Kernel Help

```
...
2512    54.261513
2513    54.396763
2514    54.529350
2515    54.661100
2516    54.793137
Name: Close, Length: 2517, dtype: float64
```

In [17]:
```
plt.figure(figsize = (12,6))
plt.plot(df.Close)
plt.plot(ma100, 'r')
plt.plot(ma200, 'g')
```

Out[17]:
```
[<matplotlib.lines.Line2D at 0x1d5909c97c8>]
```



jupyter ShareMarketPricePrediction Last Checkpoint: Last Tuesday at 17:32 (autosaved)

File Edit View Insert Cell Kernel Help

In [18]:
```
df.shape
```

Out[18]:
```
(2517, 5)
```

In [20]:
```
# Spliting data into two DataFrames ie. Testing and Traning

data_training = pd.DataFrame(df['Close'][0:int(len(df)*0.70)])
data_testing = pd.DataFrame(df['Close'][int(len(df)*0.70):int(len(df))])

print(data_training.shape)
print(data_testing.shape)
```

```
(1761, 1)
(756, 1)
```

In [21]:
```
data_training.head()
```

Out[21]:

	Close
0	7.526071
1	7.643214
2	7.656429
3	7.534643
4	7.520714

In [22]:
```
data_testing.head()
```

Out[22]:

	Close
1761	29.182501
1762	28.955000

jupyter ShareMarketPricePrediction Last Checkpoint: Last Tuesday at 17:32 (autosaved)

File Edit View Insert Cell Kernel Help

```
1763  29.037500
1764  29.004999
1765  29.152500
```

In [24]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0,1))
```

In [25]:
```
data_training_array = scaler.fit_transform(data_training)
data_training_array
```

Out[25]:
```
array([[0.02527908],
       [0.02971782],
       [0.03021854],
       ...,
       [0.84388656],
       [0.85089656],
       [0.84616011]])
```

In [31]:
```
x_train = []
y_train = []

for i in range(100, data_training_array.shape[0]):
    x_train.append(data_training_array[i-100: i])
    y_train.append(data_training_array[i, 0])

x_train, y_train = np.array(x_train), np.array(y_train)
```

In [32]:
```
# ML model
```

In [34]:
```
from keras.layers import Dense, Dropout, LSTM
from keras.models import Sequential
```

jupyter ShareMarketPricePrediction Last Checkpoint: Last Tuesday at 17:32 (autosaved)

File Edit View Insert Cell Kernel Help

In [37]:
```
model = Sequential()
model.add(LSTM(units = 50, activation ='relu', return_sequences = True, input_shape = (x_train.shape[1], 1)))
model.add(Dropout(0.2))

model.add(LSTM(units = 60, activation ='relu', return_sequences = True))
model.add(Dropout(0.3))

model.add(LSTM(units = 80, activation ='relu', return_sequences = True))
model.add(Dropout(0.4))

model.add(LSTM(units = 120, activation ='relu'))
model.add(Dropout(0.5))

model.add(Dense(units = 1))
```

In [38]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 100, 50)           10400

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

 lstm_1 (LSTM)               (None, 100, 60)           26640

 dropout_1 (Dropout)         (None, 100, 60)           0

 lstm_2 (LSTM)               (None, 100, 80)           45120

 dropout_2 (Dropout)         (None, 100, 80)           0

 lstm_3 (LSTM)               (None, 120)               96480
```

jupyter ShareMarketPricePrediction Last Checkpoint: Last Tuesday at 17:32 (autosaved)

File Edit View Insert Cell Kernel Help

```
 dropout_3 (Dropout)         (None, 120)               0

 dense (Dense)               (None, 1)                 121

=================================================================
Total params: 178,761
Trainable params: 178,761
Non-trainable params: 0
_________________________________________________________________
```

In [39]:
```
model.compile(optimizer = 'adam', loss = 'mean_squared_error')
model.fit(x_train, y_train, epochs = 50)
```

```
Epoch 1/50
52/52 [==============================] - 27s 330ms/step - loss: 0.0713
Epoch 2/50
52/52 [==============================] - 16s 307ms/step - loss: 0.0134
Epoch 3/50
52/52 [==============================] - 16s 308ms/step - loss: 0.0124
Epoch 4/50
52/52 [==============================] - 16s 310ms/step - loss: 0.0105
Epoch 5/50
52/52 [==============================] - 16s 309ms/step - loss: 0.0097
Epoch 6/50
52/52 [==============================] - 16s 308ms/step - loss: 0.0092
Epoch 7/50
52/52 [==============================] - 16s 305ms/step - loss: 0.0088
Epoch 8/50
52/52 [==============================] - 16s 310ms/step - loss: 0.0082
Epoch 9/50
52/52 [==============================] - 16s 310ms/step - loss: 0.0074
Epoch 10/50
52/52 [==============================] - 18s 347ms/step - loss: 0.0076
Epoch 11/50
52/52 [==============================] - 18s 338ms/step - loss: 0.0077
```

jupyter ShareMarketPricePrediction Last Checkpoint: Last Tuesday at 17:32 (autosaved)

File Edit View Insert Cell Kernel Help

```
Epoch 50/50
52/52 [==============================] - 17s 320ms/step - loss: 0.0031
```

Out[39]:
```
<keras.callbacks.History at 0x1d5ab628208>
```

In [40]:
```
model.save('Ml_model_LSTM')
```

```
INFO:tensorflow:Assets written to: Ml_model_LSTM\assets
```

In [41]:
```
data_training.tail(100)
```

Out[41]:

	Close
1661	27.092501
1662	27.202499
1663	27.000000
1664	26.982500
1665	27.045000
...	...
1756	29.264999
1757	29.072500
1758	29.129999
1759	29.315001
1760	29.190001

100 rows × 1 columns

In [42]:
```
data_testing.head()
```

Out[42]:

	Close

jupyter ShareMarketPricePrediction Last Checkpoint: Last Tuesday at 17:32 (autosaved)

File Edit View Insert Cell Kernel Help Not Trusted Python 3 (ipykernel)

Out[42]:

	Close
1761	29.182501
1762	28.955000
1763	29.037500
1764	29.004999
1765	29.152500

In [43]:
```
# we need to append the values from tail to the values of head only then we will be able to predict the values based on the head
#position
```

In [44]:
```
past_100_days = data_training.tail(100)
```

In [45]:
```
final_df = past_100_days.append(data_testing, ignore_index = True)
```

In [46]:
```
final_df.head()
```

Out[46]:

	Close
0	27.092501
1	27.202499
2	27.000000
3	26.982500
4	27.045000

In [47]:
```
input_data = scaler.fit_transform(final_df)
input_data
```

Out[47]:
```
array([[0.0275037 ],
```

jupyter ShareMarketPricePrediction Last Checkpoint: Last Tuesday at 17:32 (autosaved)

File Edit View Insert Cell Kernel Help Not Trusted Python 3 (ipykernel)

In [47]:
```
input_data = scaler.fit_transform(final_df)
input_data
```

Out[47]:
```
array([[0.0275037 ],
       [0.02981315],
       [0.02556164],
       [0.02519422],
       [0.02650642],
       [0.03332987],
       [0.03280496],
       [0.03196517],
       [0.03123034],
       [0.0327    ],
       [0.02823853],
       [0.0300231 ],
       [0.02571909],
       [0.02330465],
       [0.01999793],
       [0.01936806],
       [0.01506405],
       [0.01558892],
       [0.01889569],
```

In [48]:
```
input_data.shape
```

Out[48]: (856, 1)

In [50]:
```
x_test = []
y_test = []

for i in range(100, input_data.shape[0]):
    x_test.append(input_data[i-100: i])
    y_test.append(input_data[i, 0])

x_test, y_test = np.array(x_test), np.array(y_test)
```

jupyter ShareMarketPricePrediction Last Checkpoint: Last Tuesday at 17:32 (autosaved)

File Edit View Insert Cell Kernel Help Not Trusted Python 3 (ipykernel)

In [51]:
```
print(x_test.shape)
print(y_test.shape)
```

```
(756, 100, 1)
(756,)
```

In [52]:
```
#making Predictions

y_predicted = model.predict(x_test)
```

```
24/24 [==============================] - 4s 114ms/step
```

In [53]:
```
y_predicted.shape
```

Out[53]:
```
(756, 1)
```

In [54]:
```
y_test
```

```
       0.36552597, 0.34227378, 0.33251102, 0.33933447, 0.33356076,
       0.34253623, 0.35943735, 0.36568343, 0.34248377, 0.35124925,
       0.36804537, 0.36379386, 0.37271679, 0.37581356, 0.38153481,
       0.39423688, 0.39213731, 0.36568343, 0.32852197, 0.32600257,
       0.31393033, 0.31765693, 0.32064878, 0.3106761 , 0.32610749,
       0.34626291, 0.3854714 , 0.38715097, 0.42357761, 0.43055851,
       0.43522994, 0.44210584, 0.45617257, 0.44856183, 0.44625235,
       0.43727696, 0.44640981, 0.44016381, 0.43659459, 0.44352304,
       0.4410561 , 0.44735463, 0.44625235, 0.44850937, 0.44494015,
       0.44284067, 0.4395339 , 0.45722239, 0.46556795, 0.47333614,
       0.47685282, 0.47412351, 0.46488558, 0.46241864, 0.46792989,
       0.45963679, 0.4601617 , 0.44987403, 0.4493492 , 0.43334037,
       0.43759188, 0.43213317, 0.42929877, 0.41486459, 0.42672684,
       0.42530972, 0.43234308, 0.43029606, 0.44116102, 0.42404998,
       0.43181817, 0.44530761, 0.45900696, 0.45779976, 0.44483524,
       0.4613689 , 0.46294356, 0.46073907, 0.46357338, 0.45806214,
       0.46583041, 0.46352093, 0.46441321, 0.47170903, 0.48126187,
```

jupyter ShareMarketPricePrediction Last Checkpoint: Last Tuesday at 17:32 (autosaved)

File Edit View Insert Cell Kernel Help Not Trusted Python 3 (ipykernel)

In [55]:
```
y_predicted
```

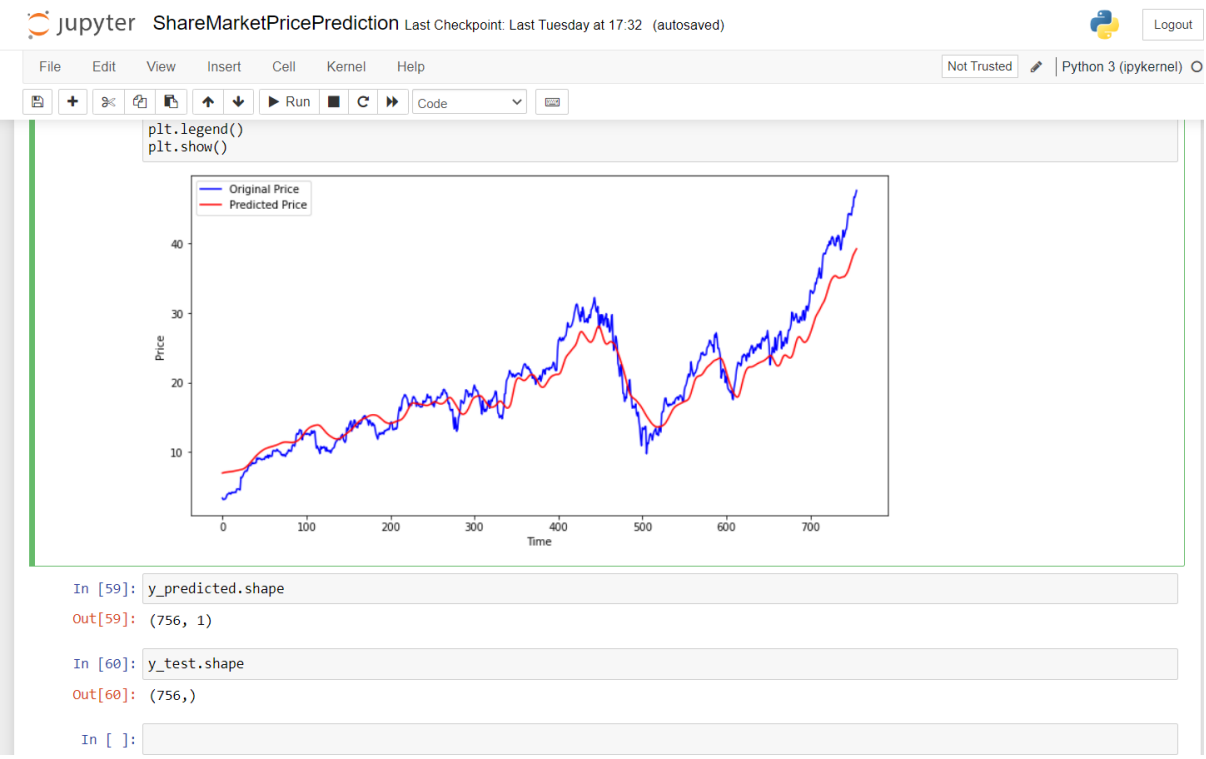
Out[55]:
```
array([[0.14695224],
       [0.14766397],
       [0.14830863],
       [0.1488624 ],
       [0.149315  ],
       [0.1496602 ],
       [0.14992033],
       [0.15013975],
       [0.1503625 ],
       [0.15062165],
       [0.15093747],
       [0.15130827],
       [0.15172789],
       [0.15219384],
       [0.15269521],
       [0.15321705],
       [0.15374589],
       [0.15426776],
       [0.15478426],
```

In [56]:
```
scaler.scale_
```

Out[56]:
```
array([0.02099517])
```

In [57]:
```
scale_factor = 1/0.02099517
y_predicted = y_predicted * scale_factor
y_test = y_test * scale_factor
```

In [58]:
```
plt.figure(figsize=(12,6))
plt.plot(y_test, 'b', label = 'Original Price')
plt.plot(y_predicted, 'r', label = 'Predicted Price')
plt.xlabel('Time')
plt.ylabel('Price')
```



Result Summary:

Stock price prediction is a machine learning project; I develop a stock cost prediction model and implemented stock market prediction using the LSTM model. The prediction model's accuracy is around 98%.

Conclusion

LSTM technique is used in the model. the technique have shown an improvement in the accuracy of predictions, thereby yielding positive results. Use of recently introduced machine learning techniques in the prediction of stocks have yielded promising results and thereby marked the use of them in profitable exchange schemes. It has led to the conclusion that it is possible to predict stock market with more accuracy and efficiency using machine learning techniques. In the future, the stock market prediction system can be further improved by utilizing a much bigger dataset than the one being utilized currently. This would help to increase the accuracy of our prediction models. Furthermore, other models of Machine Learning could also be studied to check for the accuracy rate resulted by them.

Reference

1. www.youtube.com
2. www.slimplilearn.com
3. www.zerodha.com

THANK YOU