Visualizing and ForeCasting of Stocks using Dash

by Mahesh B V

Submission date: 15-Jul-2022 11:49AM (UTC+0530)
Submission ID: 1870767190
File name: Visualizing_and_ForeCasting_of_Stocks_using_Dash.pdf (704.89K)
Word count: 2474
Character count: 12445

These days, as the associations between overall economics are fixed by globalization, outer aggravations to the money related markets are never again residential. With developingcapital markets, an ever-increasing number of information is being made day by day. The inherent estimation of an organization's stock is the worth controlled by assessing the normal future incomes of a stock and limiting them to the present, which is known as the book value. This is distinct from share market estimation, which would be governed by the company's share.Share market trading is among the most frameworks and methodologies for investors to start growing their money, but it's indeed one of the costlier investment choices offered.

The very first stage in becoming a skilled investor is to master the principles of the financial markets. Whereas the share market is a sophisticated process, its underlying principles are certainly easier. In this approach, with the use of supervised learning classifier to predict stock price movement based on financial index report and evaluate their potency is proposed. Statistical analytic methods in financial market have become stock marketing. Here in this paper, we used Long short-term memory Algorithm. We tried to use different types offeatures and kernels in order to achieve the results. We took the Data Set from Kaggle an online data set provider

Through the years, Various machine learning algorithms are implemented in time series forecasting all throughout years, although many of the machine learning models were not able to handle the increased amount of data and expectation of more accurate prediction, the deep learning models are being used nowadays which have proven their advantage over traditional machine learning methods in terms of accuracy and speed of prediction. The Long-Short-Term Memory (LSTM) Recurrent Neural Network, one of the popular deep learning models, used in stock market prediction. In this task, we will fetch the historical data of stock automatically using python libraries and fit the LSTM model on this data to predict the future prices of the stock.

1.1 Purpose of the Project:

Stock market prediction is a software prediction method that highlights the risk that investors face while investing in the market. It forecasts stock prices and currency rates by emphasizing the importance of knowing and, as a result, doing statistical analysis ahead of time for consumers.

Data is viewed as a digital fuel source that presents opportunities for higher yearning as well as future terms. Knowledge is power, and the stock market is no exception. In nature, stock is unpredictably volatile and constantly changing. The rise and fall of an equivalent are uneven and difficult to characterize. Dependencies on the same dealing with flexible resources and the people that work behind the scenes.

The beginning stock exchange for the following day is determined by investments made during a fiscal day. It has its own set of dependencies and is fully integrated with the financial and revenue-generating aspects of the business. The stock market is colossal and frenetic. The project's major goal is to forecast turning curves, develop a prediction approach, and go through the process and algorithms to arrive at a sustainable resource source.

Everything follows a predictable pattern. The manner of derivation is pattern, and the same is true for the stock. In everyday life, stock moves in a predictable fashion. Increases in certain resources can enhance the value of certain resources while decreasingthe value of others. The source and hence the outcome are determined by the polarity of the flow, which can be positive, neutral, or negative. The specified polarity's correlation isdetermined, and a reliable and efficient source is developed.

This initiative aids in linking resources and empowering individuals to comprehend and trade the most in-demand items, as well as the generation and, as a result, the vulnerabilities that must be identified and forecast. The enhancement of an equivalent is finished by the resource graph, which allows a user or a client to analyse an equivalentand take into account the needs and key facts before dealing, as well as the yield that the person is prepared to invest on. The available data source is used to forecast stock predictions, which are made for the future week. Predictability may be difficult in and of itself.

1.2 Project Scope:

- The future scope of the market system must be constructed in such a way that ensures maximum accuracy and takes into account all relevant aspects that might impact the outcome.
- LSTMs is powerful in sequence prediction because they're able to store past information.
- We can add more companies to predict stock prices by fetching historical data.

Chapter 2

LITERATURE SURVEY

2.1 Survey of Stock Market Prediction Using Machine Learning:

Author: Ashish Sharma, Dinesh Bhuriya, Upendra Singh

Year: 2019

This paper was published in 2019 by ashish Sharma, Dinesh buriya and upendra Singh. In this paper they will be using linear regression methodology to implement stock market prediction. But what we observed in this method is the linear regression model has some disadvantages in it where the linear regression model is only for limited data sets as it cannot handle larger datasets. This is the problem we faced while using this model and the accuracy of the stock price prediction is less compared to another model. The technique that was employed that was employed in this instance was a regression. Since financial stock marks generate enormous amounts of data at any given time a great volume of data needs to undergo analysis before a prediction canbe made. Each of the techniques listed under regression has its own advantages and limitations over its other counterparts.

2.2 Stock Market Prediction Using Machine Learning Algorithm:

Author: K. Hiba Sadia, Aditya Sharma, Adarsh Paul, Saurav Sanyal

Year:2019

This paper was published in the year 2019 in the IJEAT journal. In this paper they used the random forest classifier and svm classifier where both algorithms cannot give the prediction precisely which may lead to the improper prices of stocks.so this is the problem we faced in these algorithms.

The stock prediction using few classifiers includes the Random Forest Classifier, SVM Classifier. The outcome of the paper is to sum it up, the accuracy of the SCM Model to Test Set is 0.787 whereas the Random Forest Classifier is calculated to 0.808.

2.3 Stock Market Prediction using LDA-Online Learning Model:

Author: Tanapon Tantisrip, Nuanwan Sonthornphisaj

Year:2020

This paper was published in the year 2020 on the journal IEEE Where they will be using a linear discriminant analysis (LDA) where the Accuracy of the prediction of stocks is less, and it uses ANN where it has a vanishing gradient problem that it could not handle historical data and whereas in this time series model, we will be predicting the stocks using the historical data.so this the problem we faced in this model.

Financial organizations and merchants have made different exclusive models to attempt and beat the market for themselves or their customers, yet once in a while has anybody accomplished reliably higher-than-normal degrees of profitability. Nevertheless, the challenge of stock forecasting is so engaging in light of the fact that the improvement of only a couple of rate focuses can build benefit by a large number of dollars for these organizations.

2.4 Data Visualization and Stock Market and Prediction:

Author: Ashutosh Sharma, Sanket Model, Eashwaran Sridhar

Year:2019

Ashutosh Sharma, Sanket Model, Eashwaran Sridhar published the Data Visualization and Stock Market and Prediction paper in IRJET. In the paper they compared the benchmark model- Linear regression to the final improved LSTM Model, Mean Squared Error. Output graph showing the pattern prediction by LSTM model and the actual pattern observed in the dataset of closing price.

2.5 Stock Market Prediction Using Machine Learning:

Author: V kranthi sai reddy

Year:2019

He proposed the paper on Stock Market Prediction using ML in IRJET Journal. In this paper the prediction of stock market is done by the Support Vector Machine (SVM), Radial Basis Function (RBF). The model generates higher profit compared to the selected benchmarks. SVM does not give over fitting and results are highly efficiency.

2.6 Stock Market Prediction via Multi-Source Multiple Instance Learning:

Author: Xi Zhang, Siyu Qu , Jieyun Huang, Binxing Fang, Philip Yu

Year:2019

Accurately predicting the stock market is a challenging task, but the modern web has proved to be a very useful tool in making this task easier. Due to the interconnected format of data, it is easy to extract certain sentiments thus making it easier to establish relationships between various variable and roughly scope out a pattern of investment. Investment pattern from various firms show sign of similarity, and the key to successfully predicting the stock market is to exploit these same consistencies between the data sets. The way stock market information can be predicted successfully is by using more than just technical historical data and using other methods like the use of sentiment analyzer to derive an important connection between people's emotions and how they are influenced by investment in specific stocks. One more important segment of the prediction process was the extraction of important events from web news to see how it affected stock prices.

2.7 A Survey on Stock Market Prediction Using SVM:

Author: Sachin Sampat Patil, Prof. Kailash Patidar, Asst. Prof. Megha Jain

Year:2019

The recent studies provide a well-grounded proof that most of the predictive regression models are inefficient in out of sample predictability test. The reason for this inefficiency was parameter instability and model uncertainty. The studies also concluded the traditional strategies that promise to solve this problem. Support vector machine commonly known as SVM provides with the kernel, decision function, and sparsity of the solution. It is used to learn polynomial radial basis function and the multi-layer perceptron classifier. It is a training algorithm for classification and regression, which works on a larger dataset. There are many algorithms in the market, but SVM provides with better efficiency and accuracy. The correlation analysis between SVM and stock market indicates strong interconnection between the stock prices and the market index.

Chapter 3

SYSTEM REQUIREMENT SPECIFICATION

3.1 Hardware Requirements Analysis:

System Requirements analysis, the term "analysis" refers to the process of examining and identifying a comprehensive set of functional, operational, performance, interface, quality, design, criticality, and test requirements.

Minimum Hardware Requirements:

Intel Core i5 6th Generation processor or above as the central processing unit (CPU). It will also be best to use an AMD comparable CPU. RAM should be at least 8 GB. Memory — 100 GB minimum free space Operating System — Ubuntu or Windows.

3.2 Software Requirements Analysis:

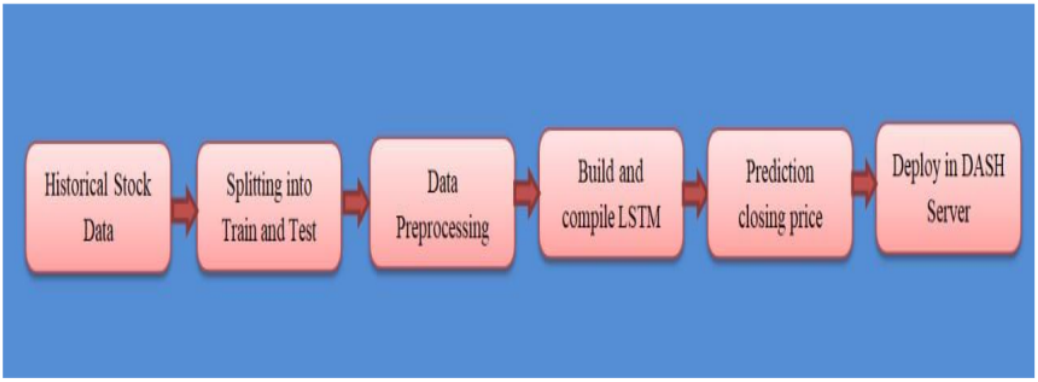
A software requirements specification is an abstract description of the services that the system should deliver as well as the limitations that it must function under. It should solely describe the system's exterior behavior's and should not include any information about the system's architecture. It's a solution that explains what services the system is intended to deliver and the limitations under which it must function in natural language with graphics.

Minimum Software Requirements:

Jupyter notebook / Anaconda, Python, Libraries: numpy, pandas, matplotlib, seaborn, streamlit, sklearn, scipy, gunicorn, dash Collections.

Chapter 4

DESIGN



A. **Historical stock data: -** The historical data taken from the Yahoo finance for the past five years for different companies and different attributes of the company like open and closing price if the stock and the highest and lowest values of the stock

B. **Train and test: -** Taking the whole dataset and dividing certain amount of data into the training data and the remaining data into testing because based on the splitting of the data gives the relevant accuracy

C. **DATA PREPROCESSING: -** It Acquires the dataset and imports all the crucial libraries and imports the dataset and identify and handling the missing values and encoding the categorical data and features the scaling

D. BUILD AND COMPILE LSTM: - First thing to do is to import the right modules and adding the layers to the LSTM model and load the dataset and then compile the LSTM neural network and train and fit the LSTM model and test the LSTM model

E. PREDICT THE CLOSING PRICE: - After building a LSTM model now based on the tested data we will predict the closing price of each stock price accurately. The perfect price is based on the way we test and train the data.

F. DEPLOY IN DASH SERVER: - After all the process is done then the prices of the stocks and even the prices of the past 2 years can be visualized on the dash server. DASH server is a plotty server where it is mostly used to plot the time series model

Chapter 5

IMPLEMENTATION

5.1 Machine Learning Model Implementation

1 Importing required modules

In [2]:
```
import pandas_datareader as web
import math
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, LSTM
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
```

2 Fetching data from yahoo finance

In [3]:
```
df = web.DataReader('AAPL',data_source='yahoo',start='2012-02-02', end='2022-06-24')
df
```

Out[3]:

Date	High	Low	Open	Close	Volume	Adj Close
2012-02-01	16.392500	16.269644	16.371786	16.292500	270046000.0	13.930380
2012-02-02	16.327499	16.213572	16.282143	16.254286	186796400.0	13.897708
2012-02-03	16.428572	16.270000	16.332144	16.417143	286599600.0	14.036956
2012-02-06	16.606428	16.364286	16.370714	16.570356	249412800.0	14.167948
2012-02-07	16.776787	16.592142	16.616072	16.743929	316223600.0	14.316356
...	...	...	...	...	...	...
2022-06-17	133.080002	129.809998	130.070007	131.559998	134118500.0	131.559998
2022-06-21	137.059998	133.320007	133.419998	135.869995	81000500.0	135.869995
2022-06-22	137.759995	133.910004	134.789993	135.350006	73409200.0	135.350006
2022-06-23	138.589996	135.630005	136.820007	138.270004	72433800.0	138.270004

In [5]:
```
plt.figure(figsize=(16,8))
plt.title('Close Price History')
plt.plot(df['Close'])
plt.xlabel('Date',fontsize=18)
plt.ylabel('Close Prize USD ($)',fontsize=18)
plt.show()
```

In [6]:
```
data= df.filter(['Close'])
dataset = data.values
training_data_len = math.ceil(len(dataset) * .8)
training_data_len
```

Out[6]: 2094

In [7]:
```
scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(dataset)
scaled_data
```

Out[7]:
```
array([[0.01395314],
       [0.01372576],
       [0.01469479],
       ...,
       [0.72236525],
       [0.73973973],
       [0.75991079]])
```

In [8]:
```
train_data = scaled_data[0:training_data_len, :]
x_train = []
y_train = []

for i in range(60,len(train_data)):
    x_train.append(train_data[i-60:i, 0])
    y_train.append(train_data[i, 0])
    if i<=60:
        print(x_train)
        print(y_train)
        print()

```

```
[array([0.01395314, 0.01372576, 0.01469479, 0.01560643, 0.01663922,
       0.01830739, 0.02181161, 0.02186474, 0.02381555, 0.02527333,
       0.0227679 , 0.02373267, 0.02371354, 0.02641875, 0.02603411,
       0.026746  , 0.02802528, 0.02873719, 0.03078787, 0.03228178,
       0.03271317, 0.03286405, 0.03030973, 0.02969346, 0.02978484,
       0.03218616, 0.03286193, 0.03431335, 0.03773469, 0.04229932,
       0.04144504, 0.04144716, 0.04474738, 0.04578017, 0.04504489,
       0.04437338, 0.04367423, 0.04599691, 0.04759072, 0.04825798,
       0.04660893, 0.044418  , 0.04847262, 0.0507443 , 0.04967965,
       0.05167084, 0.05221272, 0.0505573 , 0.05008129, 0.04935238,
       0.04562503, 0.04029114, 0.04657493, 0.04628593, 0.04184455,
       0.03877172, 0.0384997 , 0.03607289, 0.04663868, 0.04614992])]
[0.045151138566163374]
```

In [9]:
```
x_train,y_train = np.array(x_train), np.array(y_train)

```

In [10]:
```
x_train= np.reshape(x_train,(x_train.shape[0],x_train.shape[1],1))
x_train.shape
```

Out[10]: (2034, 60, 1)

In [12]:
```
model.compile(optimizer='adam', loss='mean_squared_error')
```

In [13]:
```
model.fit(x_train,y_train,batch_size=1, epochs=1)

```

```
2034/2034 [==============================] - 43s 19ms/step - loss: 2.8499e-04
```

Out[13]: <keras.callbacks.History at 0x28d40b5f190>

In [14]:
```
test_data = scaled_data[training_data_len - 60: , :]

x_test = []
y_test = dataset[training_data_len:, :]
for i in range(60,len(test_data)):
    x_test.append(test_data[i-60:i,0])
```

In [15]:
```
x_test = np.array(x_test)

```

In [16]:
```
x_test = np.reshape(x_test,(x_test.shape[0], x_test.shape[1],1))
```

In [17]:
```
predictions= model.predict(x_test)
predictions = scaler.inverse_transform(predictions)

```

In [ ]:
```

```

In [18]:
```
rmse=np.sqrt(np.mean(((predictions-y_test)**2)))
rmse=np.sqrt(np.mean(np.power((np.array(y_test)-np.array(predictions)),2)))
rmse=np.sqrt(((predictions-y_test)**2).mean())
rmse
```

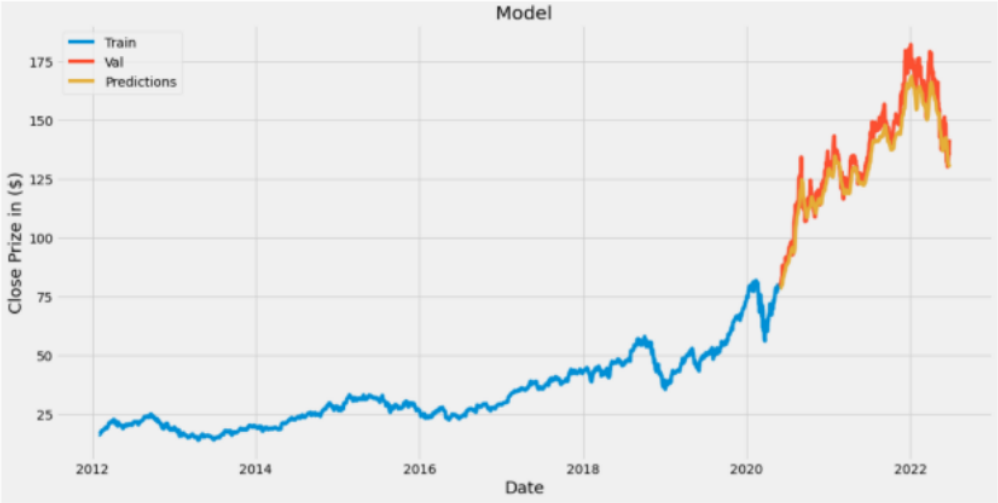
Out[18]: 6.838292279759221

In [19]:
```
mape = np.mean(np.abs((y_test - predictions)/y_test))*100
mape

```

Out[19]: 3.9559898830302855

In [20]:
```
train = data[:training_data_len]
valid = data[training_data_len:]
valid['Predictions']= predictions
plt.figure(figsize=(16,8))
plt.title('Model')
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Prize in ($)', fontsize=18)
plt.plot(train['Close'])
plt.plot(valid[['Close','Predictions']])
plt.legend(['Train','Val','Predictions'], loc ='upper left')
plt.show()
```

```
C:\Users\tharu\AppData\Local\Temp/ipykernel_8892/2839316381.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  valid['Predictions']= predictions
```



In [25]:
```
apple_quote = web.DataReader('AAPL',data_source='yahoo',start='2012-12-01', end='2022-06-23')
new_df = apple_quote.filter(['Close'])
last_60_days = new_df[-60:].values
last_60_days_scaled = scaler.transform(last_60_days)

X_test = []
X_test.append(last_60_days_scaled)
X_test =np.array(X_test)
X_test = np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))

pred_price = model.predict(X_test)
pred_price = scaler.inverse_transform(pred_price)
print(pred_price)
```

```
[[130.99954]]
```

In [23]:
```
apple_quote2 = web.DataReader('AAPL',data_source='yahoo',start='2022-06-24', end='2022-06-24')
print(apple_quote2['Close'])
```

```
Date
2022-06-23    138.270004
2022-06-24    141.660004
Name: Close, dtype: float64
```

```
In [26]: 1 valid
```

Out[26]:

Date	Close	Predictions
2020-05-29	79.485001	78.414932
2020-06-01	80.462502	78.536797
2020-06-02	80.834999	78.741890
2020-06-03	81.279999	79.005875
2020-06-04	80.580002	79.320328
...	...	...
2022-06-17	131.559998	132.277939
2022-06-21	135.869995	130.941910
2022-06-22	135.350006	130.510559
2022-06-23	138.270004	130.472870
2022-06-24	141.660004	130.999557

523 rows × 2 columns

3 Implementing Dash server

```
app = dash.Dash()
server = app.server

scaler=MinMaxScaler(feature_range=(0,1))



df_nse = pd.read_csv("./AAPL22.csv")

df_nse["Date"]=pd.to_datetime(df_nse.Date,format="%Y-%m-%d")
df_nse.index=df_nse['Date']


data=df_nse.sort_index(ascending=True,axis=0)
new_data=pd.DataFrame(index=range(0,len(df_nse)),columns=['Date','Close'])

for i in range(0,len(data)):
    new_data["Date"][i]=data['Date'][i]
    new_data["Close"][i]=data["Close"][i]

new_data.index=new_data.Date
new_data.drop("Date",axis=1,inplace=True)

dataset=new_data.values

train=dataset[0:987,:]
valid=dataset[987:,:]

scaler=MinMaxScaler(feature_range=(0,1))
scaled_data=scaler.fit_transform(dataset)
```

```
x_train,y_train=[],[]

for i in range(60,len(train)):
    x_train.append(scaled_data[i-60:i,0])
    y_train.append(scaled_data[i,0])

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))

model=load_model("saved_lstm_model.h5")
print(model)
inputs=new_data[len(new_data)-len(valid)-60:].values
inputs=inputs.reshape(-1,1)
inputs=scaler.transform(inputs)

X_test=[]
for i in range(60,inputs.shape[0]):
    X_test.append(inputs[i-60:i,0])
X_test=np.array(X_test)

X_test=np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))
closing_price=model.predict(X_test)
closing_price=scaler.inverse_transform(closing_price)

train=new_data[:987]
valid=new_data[987:]
valid['Predictions']=closing_price
```

```
df= pd.read_csv("./stock_data.csv")

app.layout = html.Div([

    html.H1("Stock Price Analysis Dashboard", style={"textAlign": "center"}),

    dcc.Tabs(id="tabs", children=[

        dcc.Tab(label='Stock Data',children=[
            html.Div([
                html.H2("Actual closing price",style={"textAlign": "center"}),
                dcc.Graph(
                    id="Actual Data",
                    figure={
                        "data":[
                            go.Scatter(
                                x=train.index,
                                y=valid["Close"],
                                mode='markers'
                            )

                        ],
                        "layout":go.Layout(
                            title='scatter plot',
                            xaxis={'title':'Date'},
                            yaxis={'title':'Closing Rate'}
                        )
                    }
```

```
        ),
        html.H2("LSTM Predicted closing price",style={"textAlign": "center"}),
        dcc.Graph(
            id="Predicted Data",
            figure={
                "data":[
                    go.Scatter(
                        x=valid.index,
                        y=valid["Predictions"],
                        mode='markers'
                    )

                ],
                "layout":go.Layout(
                    title='scatter plot',
                    xaxis={'title':'Date'},
                    yaxis={'title':'Closing Rate'}
                )
            }

        )
    ])
```

```
        ]),
        dcc.Tab(label='Facebook Stock Data', children=[
            html.Div([
                html.H1("Stocks High vs Lows",
                        style={'textAlign': 'center'}),

                dcc.Dropdown(id='my-dropdown',
                             options=[{'label': 'Tesla', 'value': 'TSLA'},
                                      {'label': 'Apple','value': 'AAPL'},
                                      {'label': 'Facebook', 'value': 'META'},
                                      {'label': 'Microsoft','value': 'MSFT'}],
                             multi=True,value=['META'],
                             style={"display": "block", "margin-left": "auto",
                                    "margin-right": "auto", "width": "60%"}),
                dcc.Graph(id='highlow'),
                html.H1("Stocks Market Volume", style={'textAlign': 'center'}),

                dcc.Dropdown(id='my-dropdown2',
                             options=[{'label': 'Tesla', 'value': 'TSLA'},
                                      {'label': 'Apple','value': 'AAPL'},
                                      {'label': 'Facebook', 'value': 'META'},
                                      {'label': 'Microsoft','value': 'MSFT'}],
                             multi=True,value=['META'],
                             style={"display": "block", "margin-left": "auto",
                                    "margin-right": "auto", "width": "60%"}),
                dcc.Graph(id='volume')
            ], className="container"),
        ])


    ])
])
```

```python
@app.callback(Output('highlow', 'figure'),
              [Input('my-dropdown', 'value')])
def update_graph(selected_dropdown):
    dropdown = {"TSLA": "Tesla","AAPL": "Apple","META": "Facebook","MSFT": "Microsoft",}
    trace1 = []
    trace2 = []
    for stock in selected_dropdown:
        trace1.append(
          go.Scatter(x=df[df["Stock"] == stock]["Date"],
                     y=df[df["Stock"] == stock]["High"],
                     mode='lines', opacity=0.7,
                     name=f'High {dropdown[stock]}',textposition='bottom center'))
        trace2.append(
          go.Scatter(x=df[df["Stock"] == stock]["Date"],
                     y=df[df["Stock"] == stock]["Low"],
                     mode='lines', opacity=0.6,
                     name=f'Low {dropdown[stock]}',textposition='bottom center'))
    traces = [trace1, trace2]
    data = [val for sublist in traces for val in sublist]
    figure = {'data': data,
              'layout': go.Layout(colorway=["#5E0DAC", '#FF4F00', '#375CB1',
                                            '#FF7400', '#FFF400', '#FF0056'],
            height=600,
            title=f"High and Low Prices for {', '.join(str(dropdown[i]) for i in selected_dropdown)} Over Time",
            xaxis={"title":"Date",
                   'rangeselector': {'buttons': list([{'count': 1, 'label': '1M',
                                                       'step': 'month',
                                                       'stepmode': 'backward'},
                                                      {'count': 6, 'label': '6M',
                                                       'step': 'month',
                                                       'stepmode': 'backward'},
                                                      {'step': 'all'}])},
                   'rangeslider': {'visible': True}, 'type': 'date'},
             yaxis={"title":"Price (USD)"})}
    return figure
```

```python
@app.callback(Output('volume', 'figure'),
              [Input('my-dropdown2', 'value')])
def update_graph(selected_dropdown_value):
    dropdown = {"TSLA": "Tesla","AAPL": "Apple","META": "Facebook",}
    trace1 = []
    for stock in selected_dropdown_value:
        trace1.append(
          go.Scatter(x=df[df["Stock"] == stock]["Date"],
                     y=df[df["Stock"] == stock]["Volume"],
                     mode='lines', opacity=0.7,
                     name=f'Volume {dropdown[stock]}', textposition='bottom center'))
    traces = [trace1]
    data = [val for sublist in traces for val in sublist]
    figure = {'data': data,
              'layout': go.Layout(colorway=["#5E0DAC", '#FF4F00', '#375CB1',
                                            '#FF7400', '#FFF400', '#FF0056'],
            height=600,
            title=f"Market Volume for {', '.join(str(dropdown[i]) for i in selected_dropdown_value)} Over Time",
            xaxis={"title":"Date",
                   'rangeselector': {'buttons': list([{'count': 1, 'label': '1M',
                                                       'step': 'month',
                                                       'stepmode': 'backward'},
                                                      {'count': 6, 'label': '6M',
                                                       'step': 'month',
                                                       'stepmode': 'backward'},
                                                      {'step': 'all'}])},
                   'rangeslider': {'visible': True}, 'type': 'date'},
             yaxis={"title":"Transactions Volume"})}
    return figure
```

```
if __name__=='__main__':
    app.run_server(debug=True,port = 8051,use_reloader=False)
```

```
C:\Users\tharu\AppData\Local\Temp/ipykernel_25192/3290210262.py:3: UserWarning:
The dash_core_components package is deprecated. Please replace
`import dash_core_components as dcc` with `from dash import dcc`
  import dash_core_components as dcc
C:\Users\tharu\AppData\Local\Temp/ipykernel_25192/3290210262.py:4: UserWarning:
The dash_html_components package is deprecated. Please replace
`import dash_html_components as html` with `from dash import html`
  import dash_html_components as html
```

```
<keras.engine.sequential.Sequential object at 0x000001E1D0D51AC0>
Dash is running on http://127.0.0.1:8051/

 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
```

```
C:\Users\tharu\AppData\Local\Temp/ipykernel_25192/3290210262.py:177: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  valid['Predictions']=closing_price
```

Chapter 6

TESTING AND RESULTS

Case 1:

X_train			y_train	x_train			y_test
F1	F2	F3	o/p	F1	F2	F3	o/p
120	130	125	140	160	190	154	160
130	125	140	134	190	154	160	174

Case 2:

*X*_train			y_train		x_train			y_test
F1	F2	F3	o/p		F1	F2	F3	o/p
125	140	134	135	184	154	160	174	
140	134	135	140	190	160	174	184	

```
rmse=np.sqrt(np.mean(((predictions-y_test)**2)))
rmse=np.sqrt(np.mean(np.power((np.array(y_test)-np.array(predictions)),2)))
rmse=np.sqrt(((predictions-y_test)**2).mean())
rmse
```

```
6.838292279759221
```

```
mape = np.mean(np.abs((y_test - predictions)/y_test))*100
mape
```

```
3.9559898830302855
```

Chapter 7

SNAPSHOTS

Chapter 8

CONCLUSION

We have implemented a model which predicts and visualizes the stocks with an accuracy of 95.6%. This model provides a good platform for visualizing and analyzing stocks where there is a lack of these software and there is no better platform to visualize the stocks. And we implemented this model for the intra-day prediction where closing price of each stock will be visualized and predicted. And we have used root-mean squared method for. And used dash plotty server for deploying our model and used for analyzing the price of the stock using DASH server. And LSTM methodology is used to implement the model because most of the time-series models are developed using LSTM. And our model gives good experience in visualizing the stocks.

Chapter 9

FUTURE ENHANCEMENT

For the further development a brief study of adding more companies. For visualizing the stock and will be adding more features based on the user comfort and will update to visualize the stock for more number of days and also improving the accuracy of the stock price And will implement the stocks for the inter-day trading. And will less size of data will improve to get more stable and proper stock price. Moreover will be studying to introduce different models to implement the project

Visualizing and ForeCasting of Stocks using Dash

ORIGINALITY REPORT

14%	13%	3%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.irjmets.com Internet Source	6%
2	Submitted to Indian Institute of Technology, Ropar Student Paper	2%
3	temeculaspeedandmarine.com Internet Source	1%
4	Submitted to Sydney Institute of Technology and Commerce Student Paper	1%
5	www.ijeat.org Internet Source	1%
6	pure.iiasa.ac.at Internet Source	1%
7	Submitted to Liverpool John Moores University Student Paper	<1%
8	delving.in Internet Source	<1%

9	Submitted to nsbm Student Paper	<1%
10	www.gradesaver.com Internet Source	<1%
11	"Proceedings of the International Conference on Paradigms of Computing, Communication and Data Sciences", Springer Science and Business Media LLC, 2021 Publication	<1%
12	oar.icrisat.org Internet Source	<1%
13	www.researchgate.net Internet Source	<1%

Exclude quotes	On	Exclude matches	< 5 words
Exclude bibliography	On		