Date- 11 June 2024

Department of Skill Faculty of Engineering & Technology

Shri Vishwakarma Skill University

Project Report--

Stock Price Prediction Application

Wrriten by-

Rupesh Sharma (22BTC36156)

Introduction

1.1)

The primary goal of this project is to predict the future stock prices of a given stock, by studying historical stock market data and applying machine learning techniques. Accurate 2 | stock price predictions can be highly valuable for investors, traders, and financial analysts in making informed decisions.

1.2) Problem statement

Our project addresses several key problems in stock market prediction, including the challenge of accurately forecasting stock prices using historical data.

1.3) Objectives

Key objectives include mitigating investment risks, providing decision support to stakeholders, and contributing to research in stock market prediction. By leveraging historical stock data and advanced algorithms, we seek to empower investors with actionable insights and enhance their investment outcomes. Additionally, we aim to bridge the gap between academic research and practical applications in financial markets, thus offering a valuable tool for informed decision-making.

1.4) Scope & Limits

Scopes -

- **Stock Price Prediction**: Developing a machine learning model to forecast the future prices of selected stocks.
- **Data Analysis and Feature Engineering**: Exploring historical stock data, performing deep analysis, and engineering relevant features to increse the predictive capability of the model.
- **Evaluation and Validation**: Evaluvating the performance of the model using appropriate metrics and validation techniques to ensure its accuracy.

Limits-

• **Data Quality**: The accuracy of predictions depends upon the quality and completeness of the historical data available for training and testing the model.

• **Generalization**: While the model may perform well on some stocks ,but on some stocks its efficency on other stocks or in different market conditions may vary.

• **Investment Advice**: The project does not provide personalized investment advice and should be used as a simple project for decision-making, rather than as the sole basis for investment decisions.

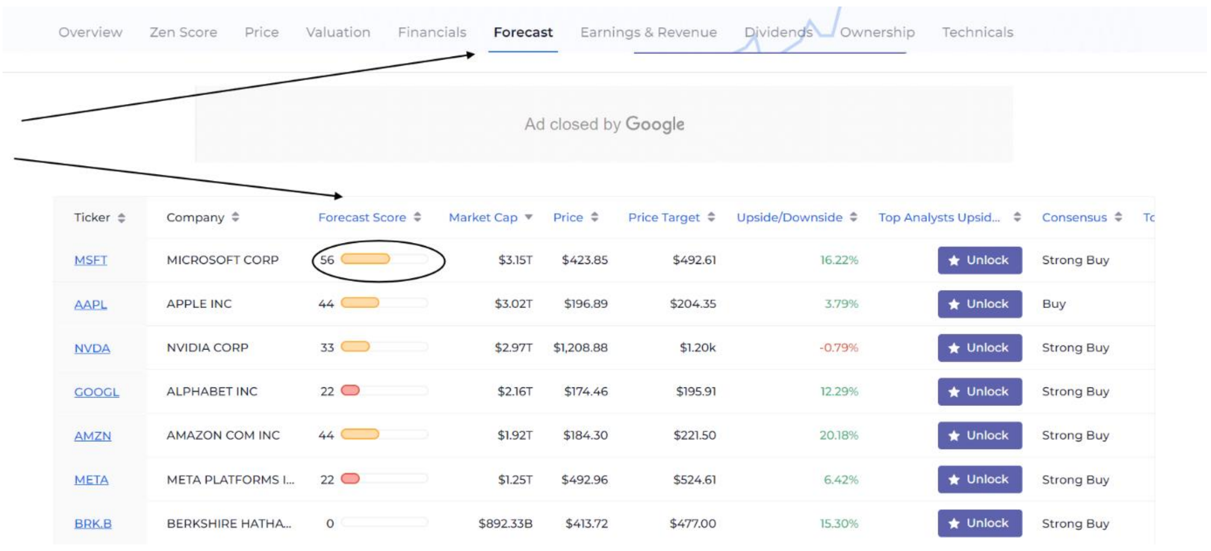
1.5) Methodology

- **Data Collection**: Acquired historical stock data of TCS(Tata consultancy service's).
- **Data Preprocessing**: Cleaned and prepared the data for analysis.
- **Feature Engineering**: Enhanced the dataset with additional relevant features(Matplotlib, minmaxscaler).
- **Data splitting** : splitted data in two parts -> Training set and Testing set.
- **Model Training**: Experimented with various machine learning models suitable for time series prediction.
- **Model Testing**: Tested the model's predictive performance using appropriate metrics.
- **Prediction**: Generated future stock price predictions.
- **Visualization**: Visualized both historical and predicted stock prices for better understanding.

Related works

Many applications such as Angle one, Zerodha , Wall street genz, Trading view, yahoo finance have the features to predict the prices of different-different stocks.

Example of existing work similar to project -



Ticker	Company	Forecast Score	Market Cap	Price	Price Target	Upside/Downside	Top Analysts Upsid...	Consensus	Tc
MSFT	MICROSOFT CORP	56	$3.15T	$423.85	$492.61	16.22%	Unlock	Strong Buy	
AAPL	APPLE INC	44	$3.02T	$196.89	$204.35	3.79%	Unlock	Buy	
NVDA	NVIDIA CORP	33	$2.97T	$1,208.88	$1.20k	-0.79%	Unlock	Strong Buy	
GOOGL	ALPHABET INC	22	$2.16T	$174.46	$195.91	12.29%	Unlock	Strong Buy	
AMZN	AMAZON COM INC	44	$1.92T	$184.30	$221.50	20.18%	Unlock	Strong Buy	
META	META PLATFORMS I...	22	$1.25T	$492.96	$524.61	6.42%	Unlock	Strong Buy	
BRK.B	BERKSHIRE HATHA...	0	$892.33B	$413.72	$477.00	15.30%	Unlock	Strong Buy	

Source - https://www.wallstreetzen.com/stock-screener/stock-forecast

Project Design

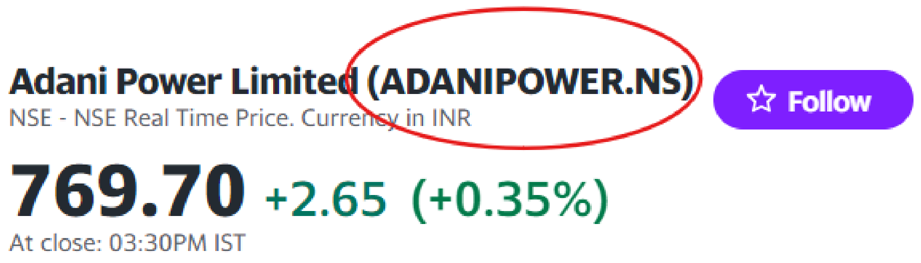
Our project has these parts -

Model Creation and Testing – We created a keras model by using the keras library from TCS Stock data . we trained that model and then tested that model after that we saved our model.

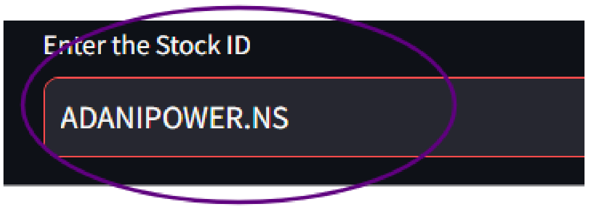
Model implementation – After saving our model we created a another python file that is using our trained model on which it is predicting values of different-different stocks that are unseen/unfamiliar to our model and given as input by the user.

Project execution – Now in the end we have to open our terminal and write the execution code in to the terminal to run our project example – "Streamlit run "file name".

Input formart – How input is given to the project it is also a crucial part on streamlit webpage. For this we have to open yahoo finance website and select any one stock . you have to just copy the symbol of that stock from the website and paste it on to the webpage. Every stock has a unique stock symbol.


Adani Power Limited (ADANIPOWER.NS)
NSE - NSE Real Time Price. Currency in INR
769.70 +2.65 (+0.35%)
At close: 03:30PM IST

Source – www.yahoofinance.com


Enter the Stock ID
ADANIPOWER.NS

4) Implementation

4.1) Tools and Technologies –

- **Programming Languages**: Python
- **Libraries**: Yahoo finance, Pandas, NumPy, Scikit-learn minmax, TensorFlow/Keras, Matplotlib,Streamlit.

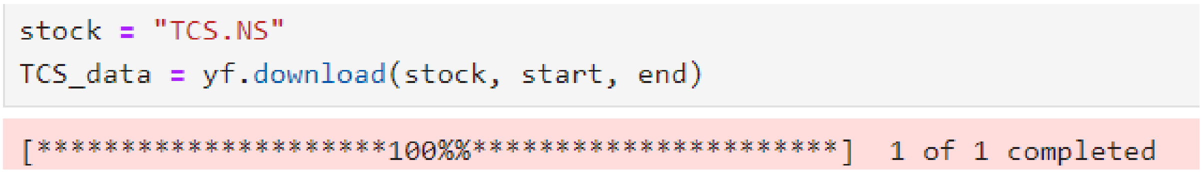
4.2) Software Enviroment -

- **Environment**: Jupyter Notebook ,VS code.
- **Other requirements** : Internet connection.

4.3) Code structure –

Now we explain that how our project is implemented in some steps-

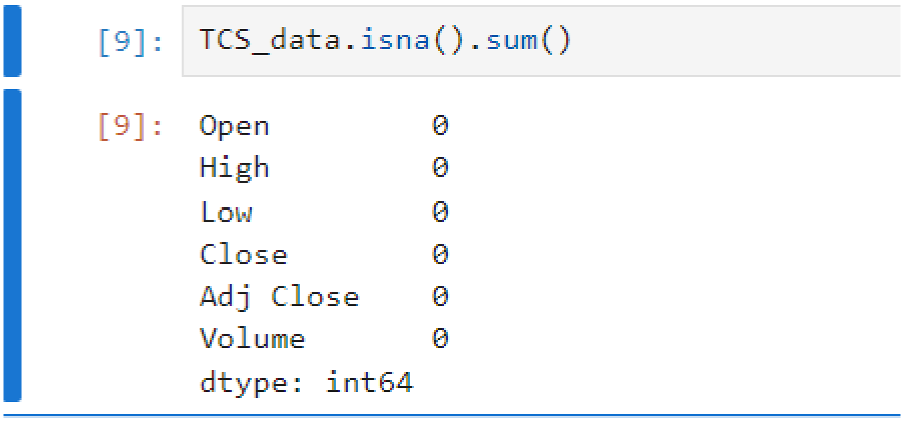
1.) Data collection – It is done by using the yahoo finance library . Here we are using this library because it is fast , free of cost and offers a large amount of realtime data. In this case we have used TCS data for our model training and testing part.



```
stock = "TCS.NS"
TCS_data = yf.download(stock, start, end)
```

```
[*********************100%%***********************]  1 of 1 completed
```

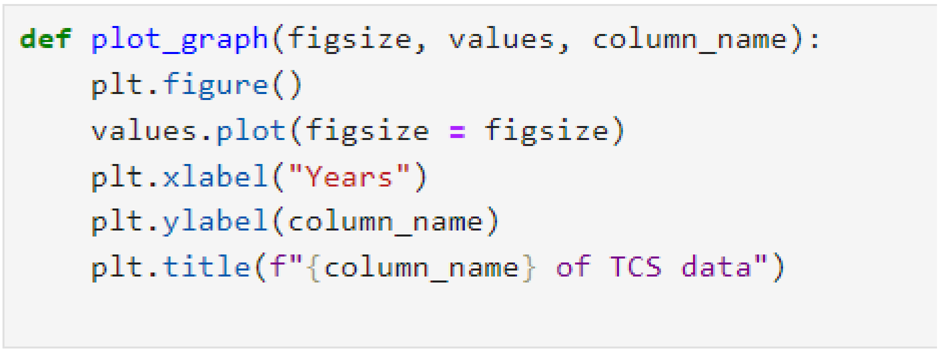
2) Data preprocessing – Before we start working on our project we have to also check is our data complete and correct because id any of this problem occurs in our data then it will defintely cause problem.



```
[9]: TCS_data.isna().sum()
```

```
[9]: Open         0
     High         0
     Low          0
     Close        0
     Adj Close    0
     Volume       0
     dtype: int64
```

3) Other important features – **Matplotlib** is a important library that we have added to our project to plot the data on the graph. We created a function to plot the values so we don't have to write the matplotlib library again and again . Just have to call that function by passing values in it.



```
def plot_graph(figsize, values, column_name):
    plt.figure()
    values.plot(figsize = figsize)
    plt.xlabel("Years")
    plt.ylabel(column_name)
    plt.title(f"{column_name} of TCS data")
```

Other important feature that is used by us is **Minmaxscaler**. If our data is in small values than it can be easily used by our model. The Minmaxscaler is a part of the sklearn.preprocessing module in the Scikit-learn library, which is used for feature scaling.

```
from sklearn.preprocessing import MinMaxScaler

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

This creates an instance of the Scaled_data class with the feature range set between 0 and 1. This means that the scaler will transform the data such that all features will lie in this range.

Example-

Suppose `Adj_close_price` looks like this:

Day	Adj_Close
1	100
2	150
3	200

After applying the `MinMaxScaler`, it might transform to:

Day	Adj_Close_Scaled
1	0
2	0.5
3	1

This ensures that all values are within the [0, 1] range, making it easier for many machine learning algorithms to process the data.

4) Data splitting – After transforming our data in (0,1) shape we have splitted in two parts 70% for the training purpose an remaining 30% for the testing purpose.

```
splitting_len = int(len(x_data)*0.7)
x_train = x_data[:splitting_len]
y_train = y_data[:splitting_len]

x_test = x_data[splitting_len:]
y_test = y_data[splitting_len:]
```

5) Model selection –

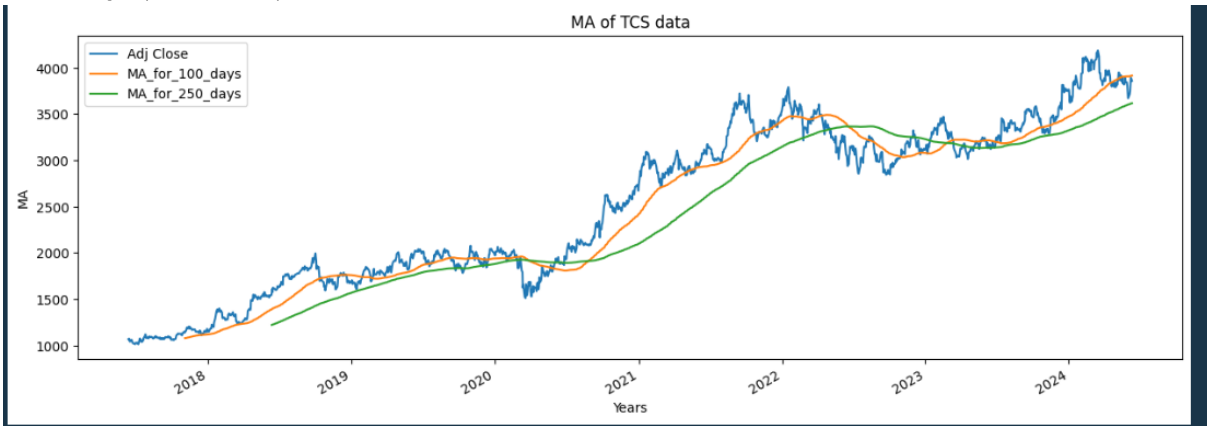
For our project we have used an **LSTM (Long Short-Term Memory)** model, which is a type of Recurrent Neural Network (RNN). LSTMs are particularly well-suited for sequence prediction problems due to their ability to learn long-term dependencies in data. LSTM are very effective in this type of data because they have the ability to capture long term data sequence.

There are many other models that are also available that can also be used for example – Support vector regression, quadratic regression ,Random forest. Each of these models has its own strengths and weaknesses, and the choice of model depends on the specific characteristics of your data and the nature of the prediction problem.

6) Training phase

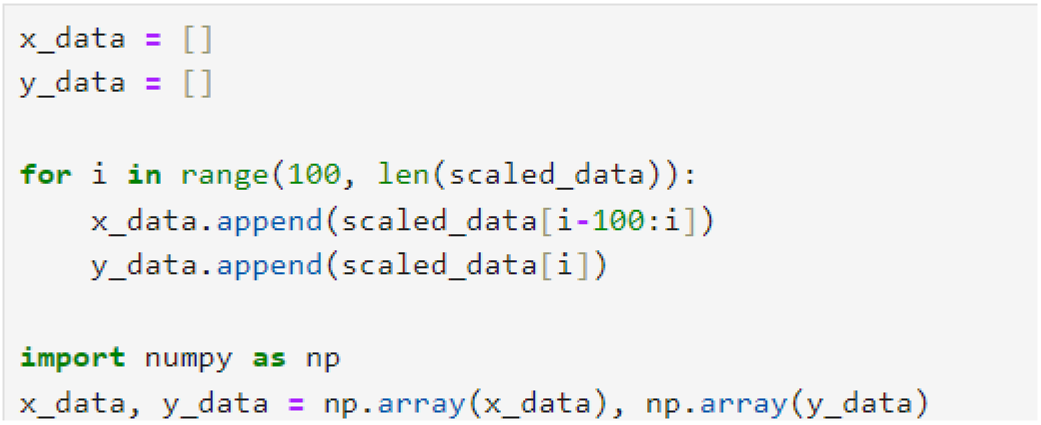
This is from one of the crucial section of our project where we trained our model.

A) When we train our model there are a high no of chances of our model that it will underfit , overfit which means we have to give data carefully to our model to do the train. If we provide a large amount of data it may not do prediction correctly and if we provide less data then our model is not able to capture the sequences and prediction accurately
For the solution of this we have used moving average concept to get idea before giving data to train our model.
We have done the moving average of 250 days and 100 days and plotted it on a same graph to compare them.



So you can see that our monthly average of 100 days is more correct if we compare it with the monthly average of 250 days . Hence it is clear that if we will use the monthly average of 100 days for training phase than it will give more accuracy to our model.

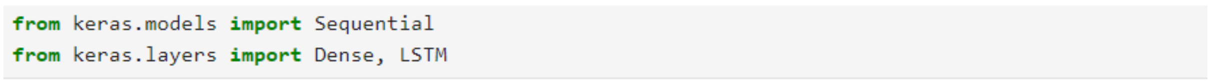
B) After finalizing the amount of data for the training purpose we stored our data in two arrays - x_data,y_data.



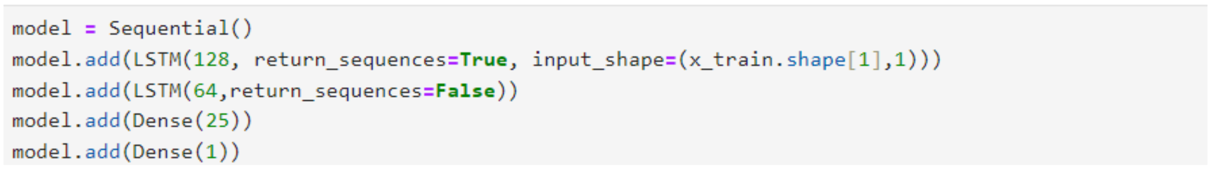
```
x_data = []
y_data = []

for i in range(100, len(scaled_data)):
    x_data.append(scaled_data[i-100:i])
    y_data.append(scaled_data[i])

import numpy as np
x_data, y_data = np.array(x_data), np.array(y_data)
```

Here our x_data consists all the value from day 1 to n-100 days and y_data consists values from day 100 to day n.

C) After this we used Keras sequential models to create Design of the model



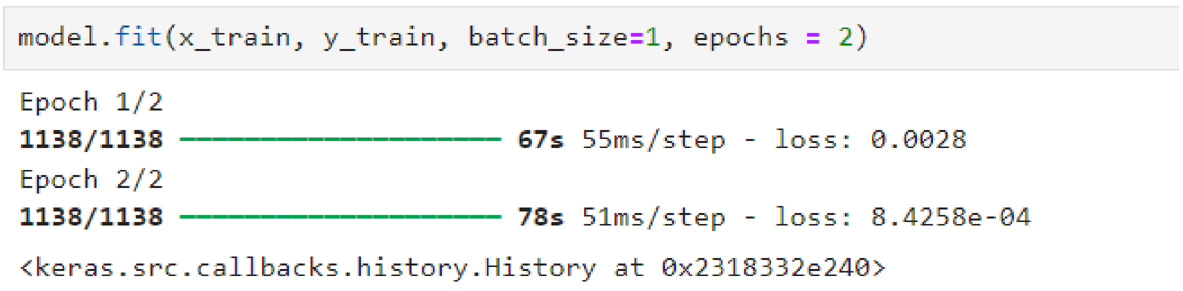
```
from keras.models import Sequential
from keras.layers import Dense, LSTM
```



```
model = Sequential()
model.add(LSTM(128, return_sequences=True, input_shape=(x_train.shape[1],1)))
model.add(LSTM(64,return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
```

1. **Initialize the Model**:
   - `model = Sequential()`: This creates a new sequential model, which is a linear stack of layers.
2. **First LSTM Layer**:
   - `model.add(LSTM(128, return_sequences=True, input_shape=(x_train.shape[1],1)))`: Adds an LSTM layer with 128 units.
     - `return_sequences=True`: Ensures that the LSTM layer outputs the full sequence of outputs for the next LSTM layer.
     - `input_shape=(x_train.shape[1], 1)`: Specifies the shape of the input data (number of time steps, number of features).
3. **Second LSTM Layer**:
   - `model.add(LSTM(64, return_sequences=False))`: Adds another LSTM layer with 64 units.
     - `return_sequences=False`: Outputs only the last output in the sequence, since this is the final LSTM layer.
4. **First Dense Layer**:
   - `model.add(Dense(25))`: Adds a fully connected (dense) layer with 25 units. This layer helps in transforming the data from the LSTM layers.
5. **Output Dense Layer**:
   - `model.add(Dense(1))`: Adds the final fully connected layer with 1 unit to produce the single output value (predicted stock price).

D) In the end we trained our model

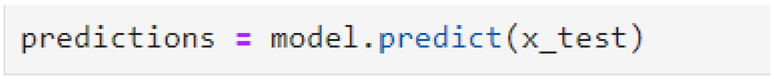


```
model.fit(x_train, y_train, batch_size=1, epochs = 2)
```

```
Epoch 1/2
1138/1138 ━━━━━━━━━━━━━━━━━━━━ 67s 55ms/step - loss: 0.0028
Epoch 2/2
1138/1138 ━━━━━━━━━━━━━━━━━━━━ 78s 51ms/step - loss: 8.4258e-04

<keras.src.callbacks.history.History at 0x2318332e240>
```

Here X_train (day 1 to n-100days data),Y_train(day 100 to n days data) are the lables which are given to our model. Batch size = 1 means the model updates its weights after processing each sample, epochs = 2 means the entire process is repeated 2 times.

Now our model is ready to do predictions our training phase is completed success fully. Now we can predict the price of y by using the data set



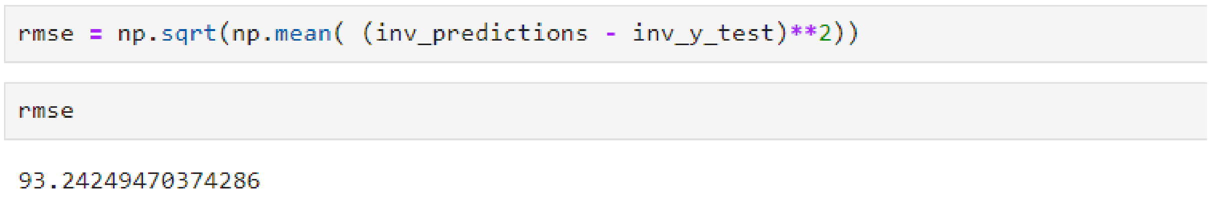
```
predictions = model.predict(x_test)
```

7) Testing phase

Testing phase is also a important part of our model same as the training phase.

- First we calculate the errors b/w the actual adj closed price and predicted price. we have different types of error calculation techninques like MAE, MSE and RMSE

Here we have used RMSE which is accurate than the other techniques.



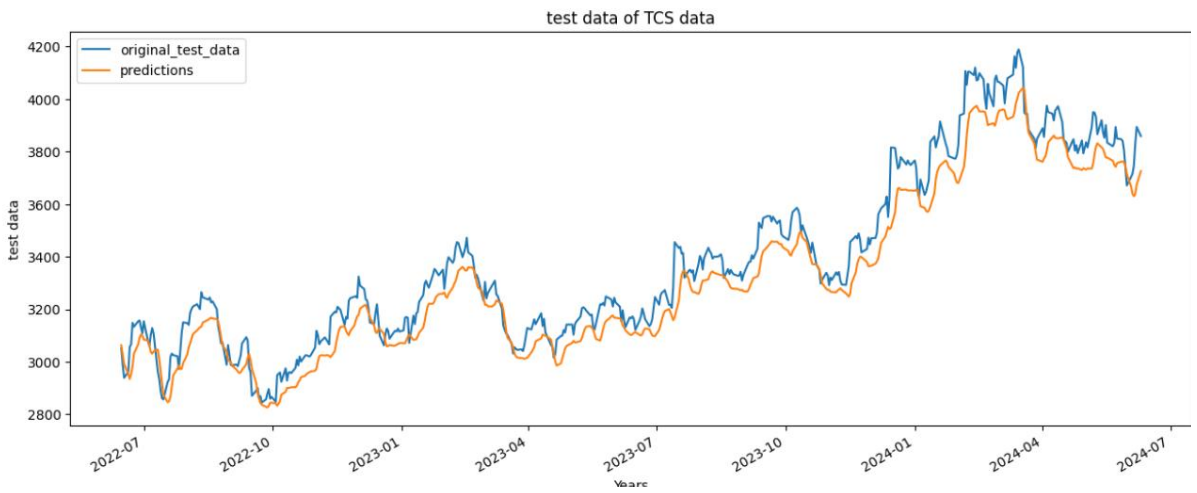
```
rmse = np.sqrt(np.mean( (inv_predictions - inv_y_test)**2))
```

```
rmse
```

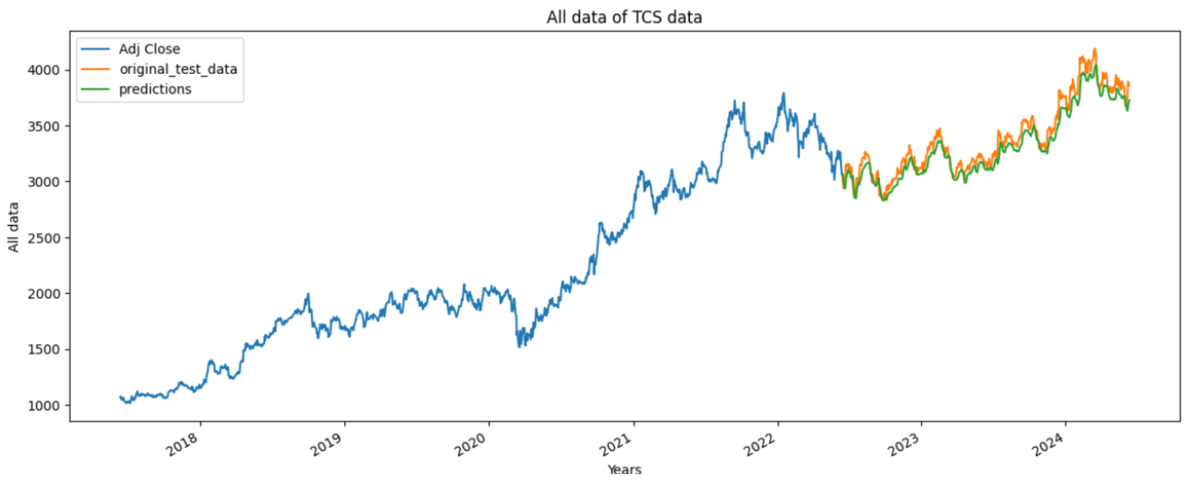
```
93.24249470374286
```

So RMSE = 93.24—which means on average the error of a days prediction is loss with 93.24—rupees which is not a large no means our model has not so much errors our model is working perfectly.

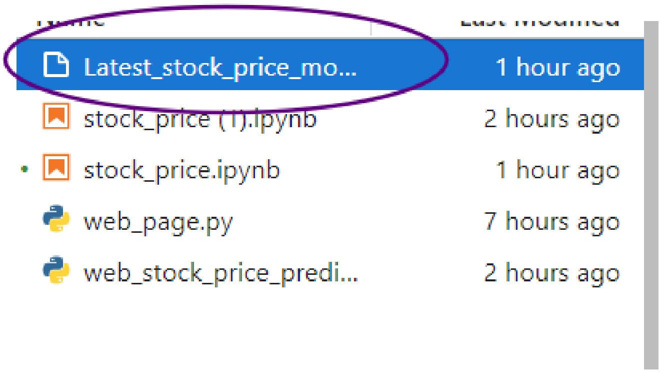
- Now we visualized our data on a graph to show the difference between the actual price and the predicted price.



- In last we visualized our whole data Training data , Testing data and predicted price in a same graph.



8) Model Save : In the end we saved our model . After this our model is stored in our same folder.



Latest_stock_price_mo...	1 hour ago
stock_price (1).ipynb	2 hours ago
stock_price.ipynb	1 hour ago
web_page.py	7 hours ago
web_stock_price_predi...	2 hours ago

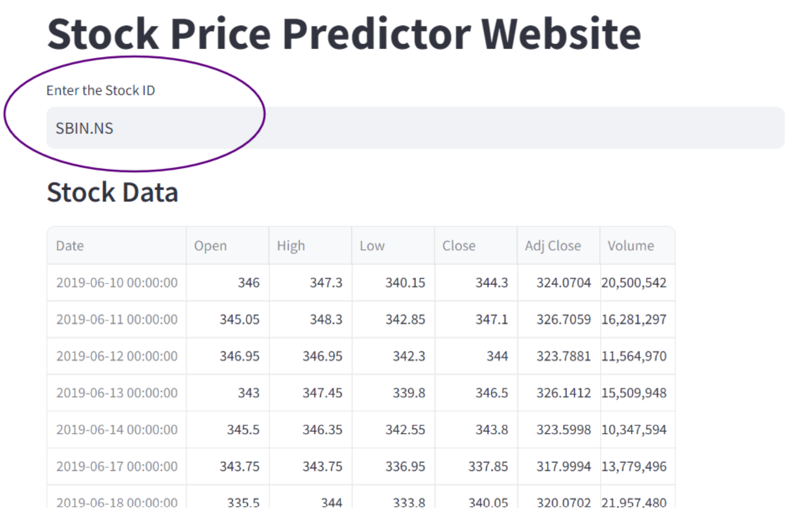
Now we can use our model to predict the prices of other stocks by using the same calculations . We have to just load this model in other files to use it for predictions .

5) Final model implementation/User Interface

For finally implementing our model we have to give inputs stock data on which our model will do predictions. We have also added future price price forecasting upto 60 days. And also added next day predicted price where you can enter your next day investment and it will calculate the the next days predicted loss and profit.

We have used streamlit library to implement these feature on a local webpage.

1. You have enter to the symbol of the stock price(you can find symbols on www.yahoofinance.com) that you want to predict data will we automatically downloaded from yahoo finance website



Stock Price Predictor Website

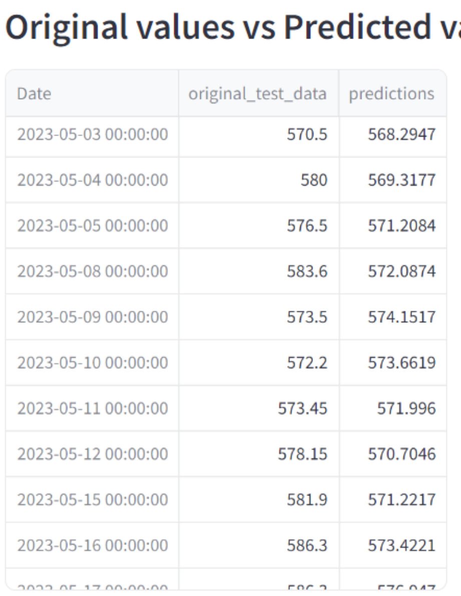
Enter the Stock ID

SBIN.NS

Stock Data

Date	Open	High	Low	Close	Adj Close	Volume
2019-06-10 00:00:00	346	347.3	340.15	344.3	324.0704	20,500,542
2019-06-11 00:00:00	345.05	348.3	342.85	347.1	326.7059	16,281,297
2019-06-12 00:00:00	346.95	346.95	342.3	344	323.7881	11,564,970
2019-06-13 00:00:00	343	347.45	339.8	346.5	326.1412	15,509,948
2019-06-14 00:00:00	345.5	346.35	342.55	343.8	323.5998	10,347,594
2019-06-17 00:00:00	343.75	343.75	336.95	337.85	317.9994	13,779,496
2019-06-18 00:00:00	335.5	344	333.8	340.05	320.0702	21,957,480

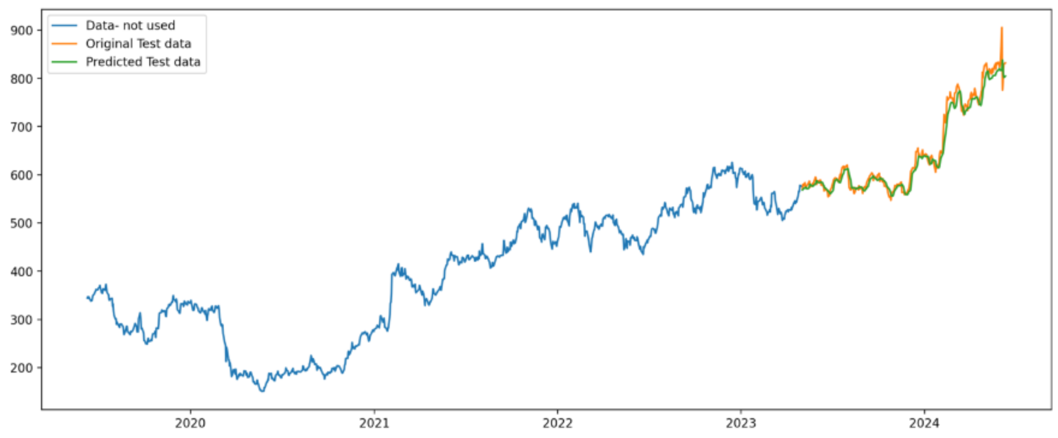
2. After this your model will predict the values of the data that you can see in a table & graph to compare the predictions .



Original values vs Predicted values

Date	original_test_data	predictions
2023-05-03 00:00:00	570.5	568.2947
2023-05-04 00:00:00	580	569.3177
2023-05-05 00:00:00	576.5	571.2084
2023-05-08 00:00:00	583.6	572.0874
2023-05-09 00:00:00	573.5	574.1517
2023-05-10 00:00:00	572.2	573.6619
2023-05-11 00:00:00	573.45	571.996
2023-05-12 00:00:00	578.15	570.7046
2023-05-15 00:00:00	581.9	571.2217
2023-05-16 00:00:00	586.3	573.4221
[illegible]	[illegible]	[illegible]

Original Close Price vs Predicted Close price

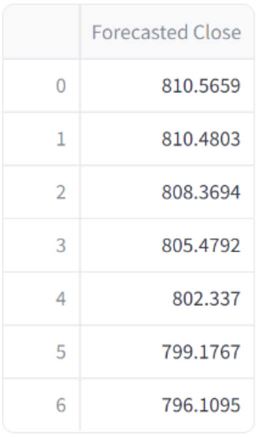


3) In the end here comes our important part in the last parts we are predicting price of past data or on test data. But now we will predict the future stock price on the basis our calculations that we have done until now.
Select the number of days predictions you want to forecast. And it will predict the price of next days.

Select the prediction period

7

Forecasted values for the next 7 days



	Forecasted Close
0	810.5659
1	810.4803
2	808.3694
3	805.4792
4	802.337
5	799.1767
6	796.1095

Results

In the end we are successful to make a user interface that can provides user the following -

1. **User Input**:
   - The user inputs the stock ID (e.g., "RELIANCE.NS") and the amount they want to invest, Number of days of price forecast.
2. **Data Retrieval**:
   - It retrieves historical stock price data for the specified stock using the Yahoo Finance API (`yfinance`).
   - The data is then split into training and testing sets.

3. **Model Loading**:
    - We are able to load a pre-trained LSTM model from the file "Latest_stock_price_model.h5".
4. **Prediction**:
    - The model predicts the stock prices for the next day and calculates the potential profit or loss based on the user's investment amount.
5. **Visualization**:
    - It displays various visualizations, including the original close price, moving averages, and the predicted close price compared to the actual close price.
    - It also provides forecasted values for a user-selected number of days.
6. **Results**:
    - The application displays the original and predicted stock prices, potential profit or loss, and **forecasted values for the next specified number of days**.

Limitations

- **Market Ireggularity**: Stock markets are influenced by various factors, including economic conditions, geopolitical events, and investor sentiment. Predicting stock prices accurately in such dynamic environments is challenging.

- **Data Quality**: The accuracy of predictions heavily depends on the quality and reliability of the input data. Financial data may contain errors, missing values, or outliers, which can adversely affect the model's performance.

- **Model Complexity**: While LSTM models are capable of capturing temporal dependencies in sequential data, they can also be complex and prone to overfitting, especially when trained on limited data.

- **Limited Scope**: Stock price prediction models typically focus on historical price data and may not incorporate fundamental factors such as company financials, industry trends, or macroeconomic indicators, which can also influence stock prices.

- **Risk and Uncertainty**: Stock market investments carry risk, and predictions generated by models should be interpreted with caution. Unexpected events, market shocks, or sudden changes in investor behavior can lead to huge difference in predicted price and original price.

- **Human Factors**: Market behavior is influenced by human emotions, biases, and irrationality, which may not be fully captured by mathematical models. Psychological factors such as fear, greed, and market sentiment can drive price movements in unpredictable ways.

Conclusion

Despite the promising results demonstrated by the model, it's important to acknowledge the uncertainties and limitations in stock price prediction. Financial markets are complex and influenced by multiple factors, including economic conditions, geopolitical events, and investor sentiment. While machine learning models can analyze historical trends and identify patterns, they may struggle to account for sudden changes or unseen events that impact market.

Investors should exercise caution and complement quantitative analysis with qualitative research, market expertise, and risk management strategies.

The addition of future price forecasts based on past data provides users with valuable insights into potential market trends and investment opportunities. However, it's crucial to approach to consider multiple sources of information when making investment decisions.

In summary, while predictive price modeling can offer valuable insights and aid in decision-making processes, it should be used as one tool among many in a comprehensive investment strategy. By combining data-driven analysis with a deep understanding of market fundamentals and risk management principles, investors can navigate the complexities of financial markets more effectively and optimize their investment portfolios for long-term success.