Project Exhibition- II

Title : Stock price prediction using LSTM

Guide : Prof. Anand Motwani

Team Members:

- Aditya Kunal Bhate- 19BCE10046
- Sanjana Singh- 19BCE10274
- Rachita Jha- 19BCE10283
- Akshra Singh- 19BCE10295

CONTENT :

Introduction

- ➔ The direction of the financial market is always volatile and the security return is deemed to be unpredictable.
- ➔ In the field of quantitative trading, predicting the future security returns lies in the center of the industry.The quantitative trading strategy uses mathematical models to make the decision hence avoids the interruption of human subjectivity and emotion.
- ➔ In this research, we will construct and apply the deep learning sequential model, namely Long Short Term Memory Model (LSTM) into the prediction of stock prices.

Limitations

- ➔ Fundamental method makes the trading decision based on the subjective view on the industry or company's future direction, it mainly relies on the public information such as market news, corporate statistics.

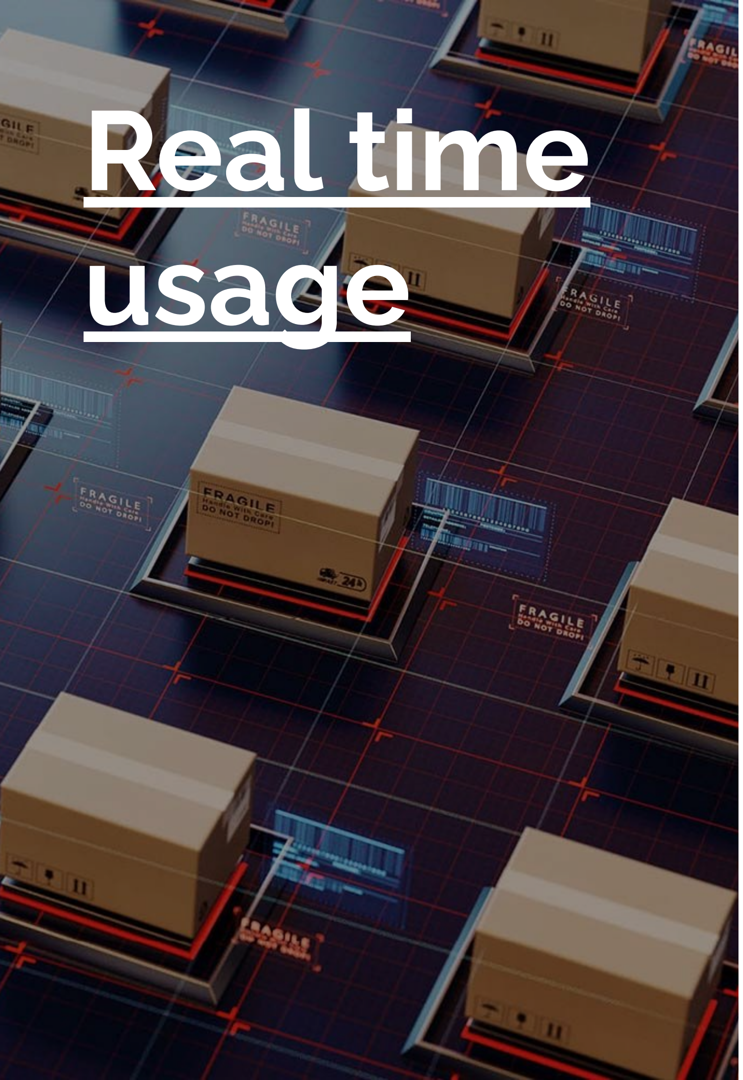
- ➔ Our trading strategy is an intraday trading strategy, meaning that we do not hold positions overnight, so we only take actions on the same day.

Proposed work and methodology

- ➔ Our main motive is to make a stock market predictor where we can predict the price of a particular stock for the next day based on the prices of that particular share.
- ➔ Well , the point to note here is that the price which will be predicted will not be exactly the same but will be approximately the same as the real price of stock.
- ➔ We will be using recurrent neural networks and machine learning using python.
- ➔ Also, since the prices are not accurate this project cannot be completely depended upon for making financial decisions , but is a very good way to learn neural networks.

Novelty of the project

- ➔ We observe from the previous works and find the gaps and proposed a solution architecture with a comprehensive feature engineering procedure before training the prediction model. With the success of feature extension method collaborating with recursive feature elimination algorithms, it opens doors for many other machine learning algorithms to achieve high accuracy scores for short-term price trend prediction.
- ➔ We used LSTM model and improved the prediction scores in all the evaluation metrics. The proposed solution outperformed the machine learning and deep learning-based models in similar previous works.
- ➔ The novelty of our proposed solution is not only to apply the technical method on raw data but also carry out the feature extensions that are used among stock prediction.



Real time usage

- ➔ Stock price prediction is the act of trying to determine the future value of a company stock.
- ➔ The accurate prediction of stock price movement will lead to more profit investors.
- ➔ Allows merchants to predict the ups and downs in the market.
- ➔ Allows traders to identify the suitable and profitable trading period.
- ➔ Dividend income

Requirements

HARDWARE REQUIREMENT

- Processor: Intel ® Core ™ i5-6440HQ CPU @ 2.60GHz 2.59GHz
- Installed Memory (RAM): 8.00 GB (7.64 GB Usable)
- System type: 64-bit Operating System

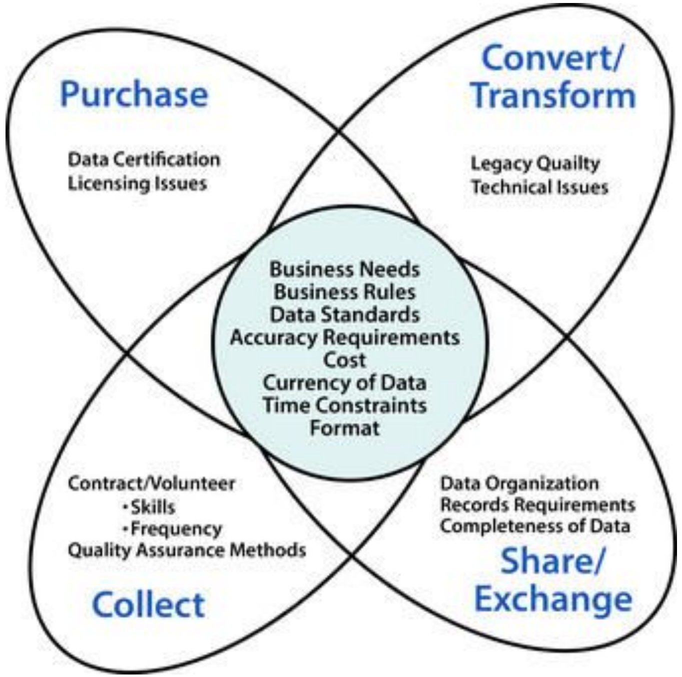
SOFTWARE REQUIREMENT

- Operating System: Windows 10
- Back-end Tool: Keras and tensorflow (above 2.0 version)
- Tools & Libraries: Stacked LSTM,Scikit-learn
- IDE: Google Colab

Data Acquisition

Data Acquisition Methods

There are four methods of acquiring data: collecting new data; converting/transforming legacy data; sharing/exchanging data; and purchasing data. This includes automated collection (e.g., of sensor-derived data), the manual recording of empirical observations, and obtaining existing data from other sources.



Common Data Acquisition Considerations

- Business Needs: The first thing to always consider is the business need - why are these data required? What will be done with them?

- Business Rules: A business rule identifies the constraints under which the business operates. For instance, where applicable, all geospatial data must have Federal Geographic Data Committee (FGDC) compliant metadata. These rules will affect your data acquisition decisions.

- Data Standards: Any Government, USGS, or industry standards that apply will need consideration.

- Accuracy Requirements: Among the most familiar accuracy requirements is the locational accuracy for spatial data; but there are other accuracy requirements that you may need to consider as well.

- Cost: Cost is always a consideration. Sometimes it's cheaper to buy than to collect.

- Currency of Data: For many types of work, the data need to be fairly current. For others, data may need to cover a specified time period. For others, data need to be in a specific season. If you are trying to determine vegetation coverage, for example, you may want photographs from the summer, when vegetation is at the highest. If you are trying to look for land forms, you may want winter photos.

- Time Constraints: You should determine how soon you need the data.

- Format: Do you need the data as spatial data, photos, flat files, Excel files, XML files? This may not apply, but you need to determine that for each project.

System Architecture Diagram

➔ **Download Data**

- Download the share data of particular company. The data such as date, open share price, high value, low value, last share value close share price, total trade and turn over values are used.

➔ **Dataset**

- Keep all our data in the form of csv files. Each line of the file is a data record.
- Our dataset is kept in tabular format in excel with values such as date, open, high, low, last, low ,total trade and turnover values.

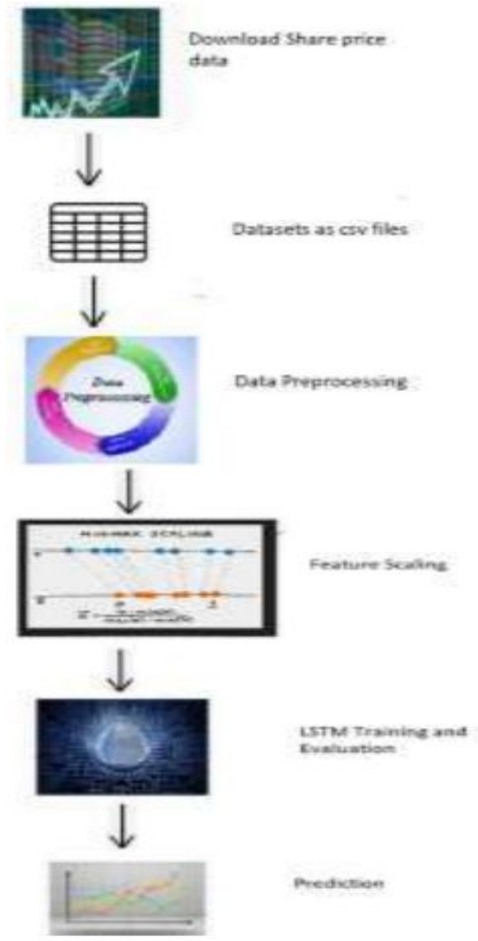


Fig 1 Architecture Overview

➔ **Data Preprocessing**

- Analyze data that has not been carefully screened for such problems can produce misleading results.
- Data preprocessing is the most important phase of a machine learning project, especially in computational data.
- Data preparation and filtering steps can take considerable amount of processing time.
- Data preprocessing includes cleaning, instance selection, normalization, transformation, feature extraction and selection etc. The product of data preprocessing is the final training set.

➔ **Feature Scaling**

- Feature scaling is a method used to standardize the range of independent variables or features of data. In data preprocessing.
- Therefore, the range of all features should be normalized so that each feature contributes approximately proportionately to the final distance.
- Simplest method in rescaling the range of features to scale the range in [0, 1].

Literature Review

Prediction through LSTM helps organizations and individuals to understand the behavior and trend in an effective way and provide accurate results. Long Short Term Memory(LSTM) recurrent neural networks belong to the field of deep learning algorithms. This model is chosen over other neural networks as it is very efficient in terms of time- series forecasting and processing the entire sequence data. After importing the required libraries, datas of a few days, months or years are taken and then visualized for training purposes. To preprocess the data and improve the accuracy of the project, LSTM comes into play for testing purposes. New data will be fetched for training purposes and LSTM will be tested on it. After fetching new data, test the LSTM project and predict the stock prices from the earlier one. This will increase the level of accuracy in statistical ways as it will discover the trends and dynamics of stock prediction. Our aim will be to cover the loopholes by using LSTM.

MODULE DESCRIPTION

1. Data Collection
2. Pre-Processing
3. Creation of LSTM Model
4. Predicting the Trend

Data Collection:

- ➔ We will be using the **dataset** from **Yahoo Finance** - Stock of **Reliance Industries** Limited NSE: RELIANCE: (https://finance.yahoo.com/quote/RELIANCE.NS/history?period1=1433635200&period2=1591488000&interval=1d&filter=history&frequency=1d)

- ➔ **Libraries**- Mainly Pandas,Keras and Tensorflow

- ➔ There are multiple **variables** in the dataset – **date, open, high, low, close, adj Close and volume**
- ➔ The columns *Open* and *Close* represent the starting and final price at which the stock is traded on a particular day.
- ➔ *High*, *Low* represent the maximum, minimum price of the share for the day.
- ➔ *Volume* represents the total number of shares traded in a security over a period.

Preprocessing

- The profit or loss calculation is usually determined by the closing price of a stock for the day, hence we will consider the adj closing price as the target variable.
- Since LSTM are sensitive to the scale of the data, so we apply MinMaxScaler to transform our values between 0 and 1
- Sklearn contains the preprocessing module that will allow us to scale our data and then fit it in our model.

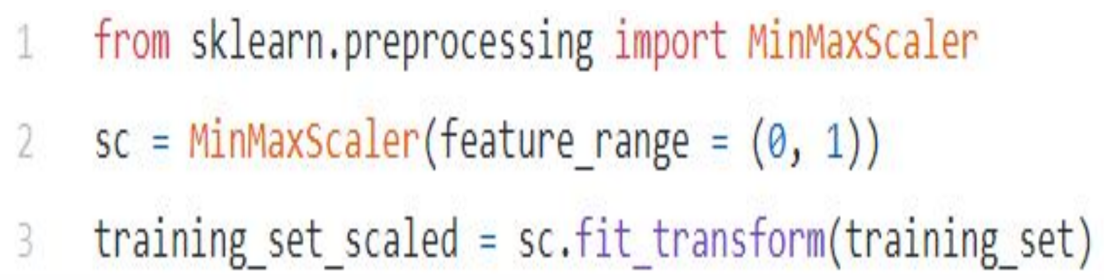
->Normalization

- Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.
- Here’s the formula for normalization:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Here, Xmax and Xmin are the maximum and the minimum values of the feature respectively.

- **When the value of X is the minimum value in the column, the numerator will be 0, and hence X' is 0**
- **On the other hand, when the value of X is the maximum value in the column, the numerator is equal to the denominator and thus the value of X' is 1**
- **If the value of X is between the minimum and the maximum value, then the value of X' is between 0 and 1**



```python
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range = (0, 1))
training_set_scaled = sc.fit_transform(training_set)
```

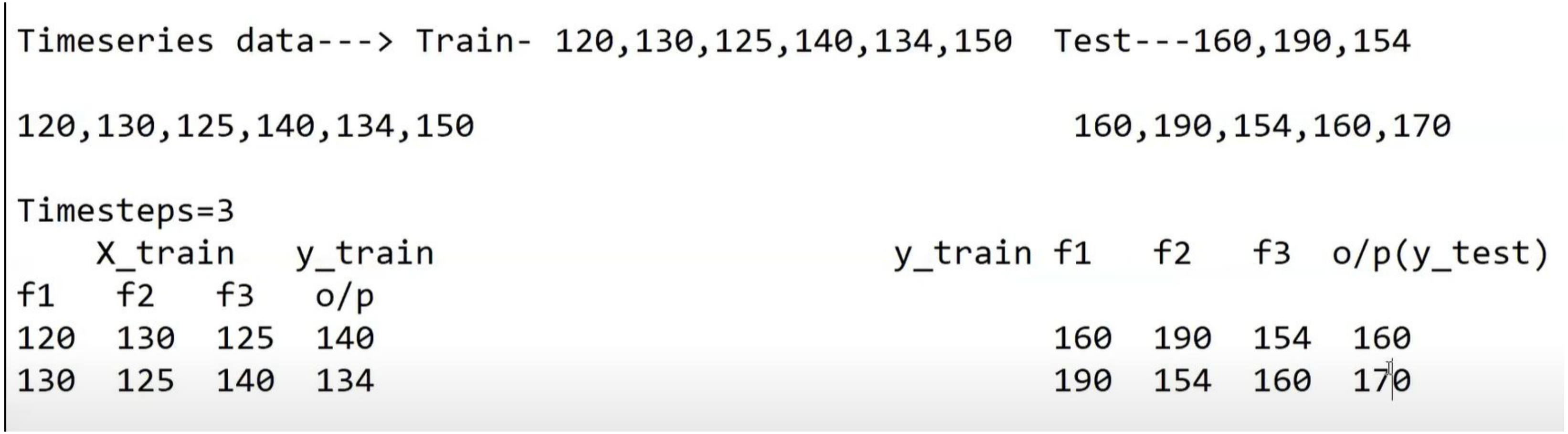
TRAINING AND TESTING DATASET

Splitting the data into training and test sets is crucial for getting a realistic estimate of our model’s performance. We have used 75% of the dataset as the training set and the remaining 25% as the testing set.

We will take 75% of the the total length of our data frame and store it as our training size and test size should be the length of (data frame-training size) of the data frame i.e 25%

Using this we will create our training and test data and store it in variables.

- TimeSteps-How many data points we are going to use to predict the next datapoint in the sequence. .
- X_Train Input on the basis of which we will get the Y_Train output in the Training dataset.
- Similarly we will do the same segregation under test data as X_Test and Y_Test.

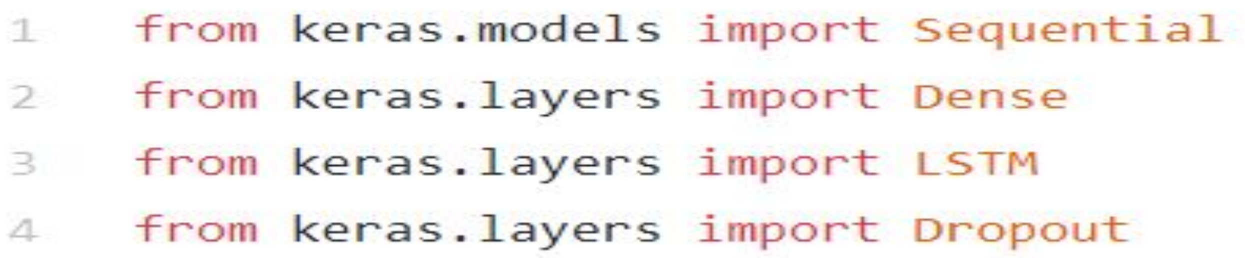


```
Timeseries data---> Train- 120,130,125,140,134,150  Test---160,190,154

120,130,125,140,134,150                              160,190,154,160,170

Timesteps=3
   X_train    y_train                           y_train f1    f2    f3   o/p(y_test)
f1    f2    f3    o/p
120   130   125   140                                   160   190   154   160
130   125   140   134                                   190   154   160   170
```

Creation of LSTM Model

In order to build the LSTM, we need to import a couple of modules from Keras:

1. Sequential for initializing the neural network
2. Dense for adding a densely connected neural network layer
3. LSTM for adding the Long Short-Term Memory layer
4. Dropout for adding dropout layers that prevent overfitting



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

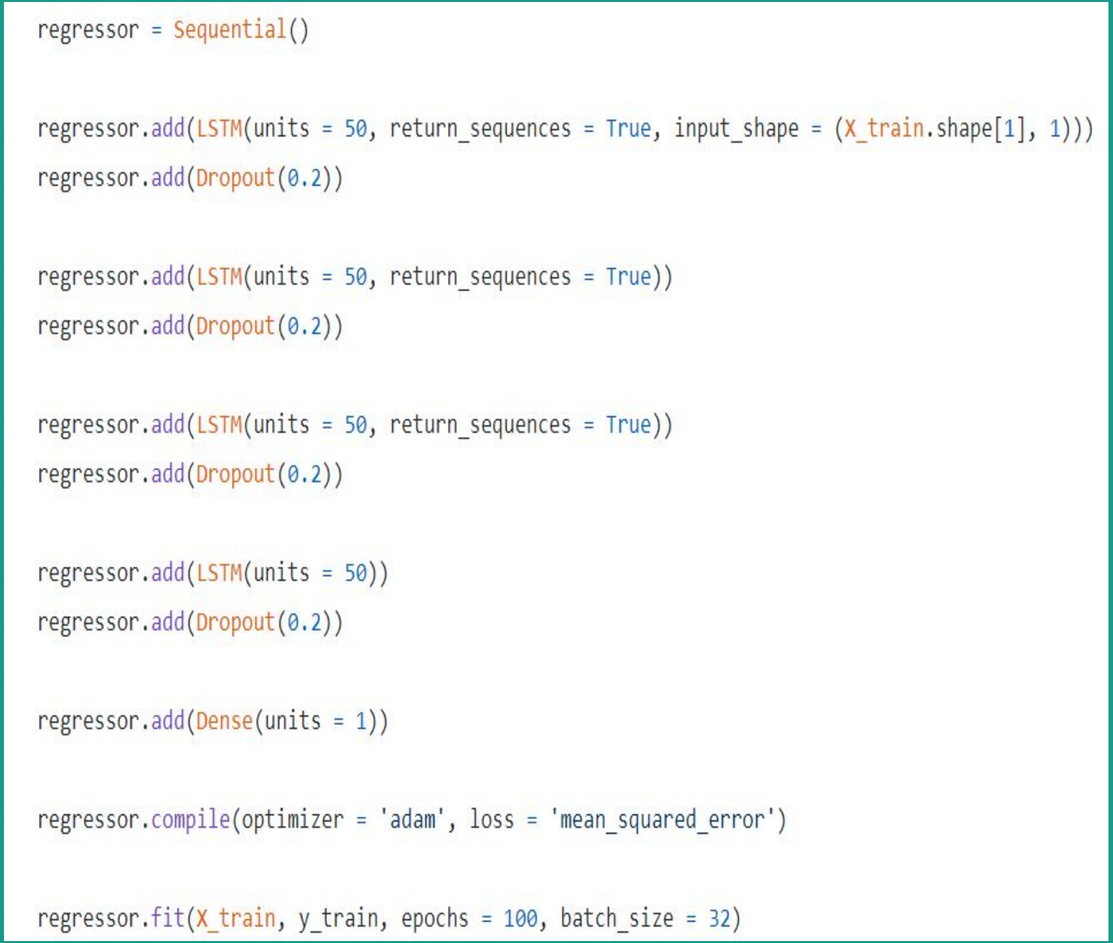
Evaluate the performance

We add the LSTM layer and later add a few Dropout layers to prevent overfitting. We add the LSTM layer with the following arguments:

1. *50 units* which is the dimensionality of the output space
2. return_sequences=True which determines whether to return the last output in the output sequence, or the full sequence
3. input_shape as the shape of our training set.

The model is tested based on various criteria to test its proper functionality. After this, we compile our model

Next, we fit the model to run on 100 epochs with a batch size of 32.

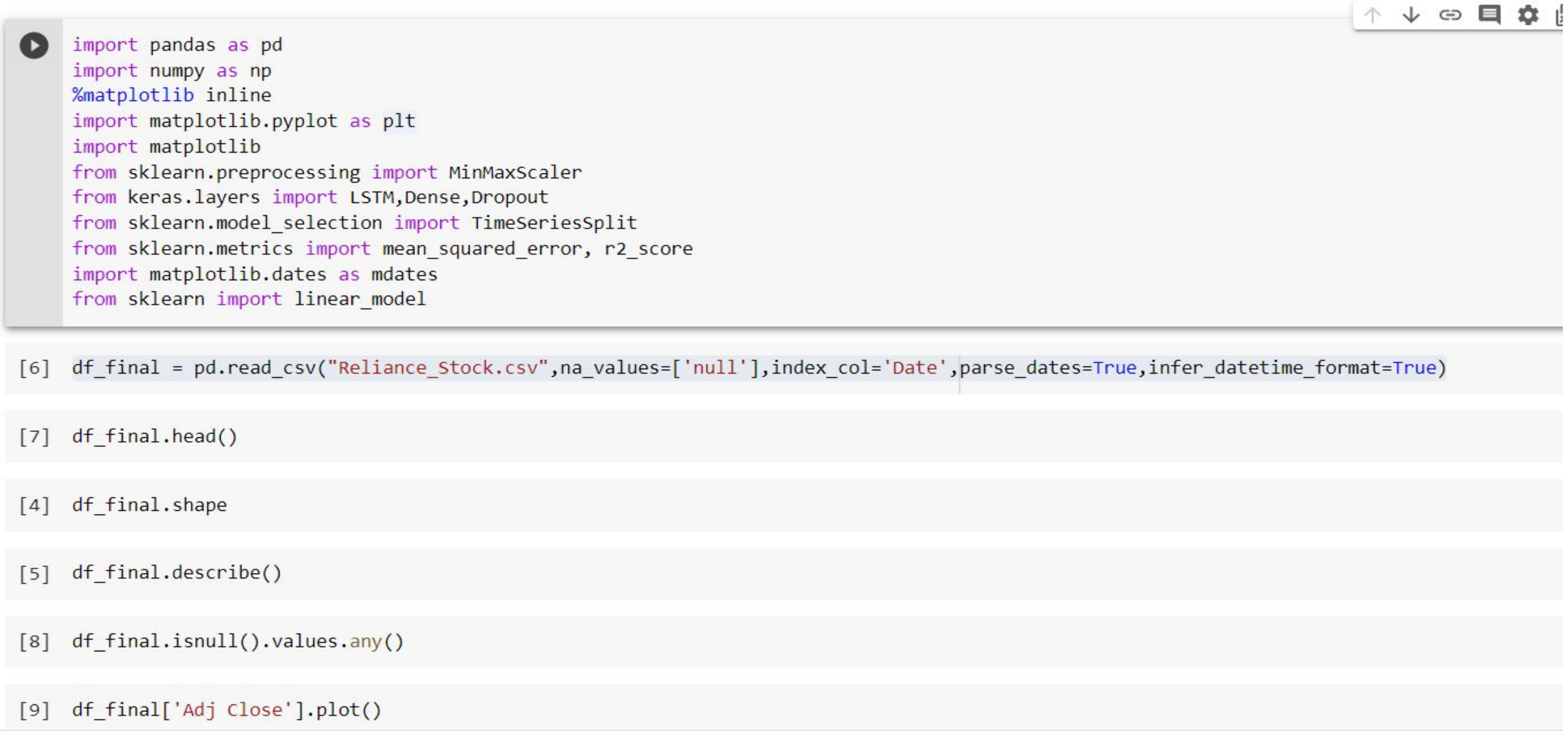


```
regressor = Sequential()


regressor.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.shape[1], 1)))

regressor.add(Dropout(0.2))


regressor.add(LSTM(units = 50, return_sequences = True))

regressor.add(Dropout(0.2))


regressor.add(LSTM(units = 50, return_sequences = True))

regressor.add(Dropout(0.2))


regressor.add(LSTM(units = 50))

regressor.add(Dropout(0.2))


regressor.add(Dense(units = 1))


regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')


regressor.fit(X_train, y_train, epochs = 100, batch_size = 32)
```

Predicting The Trend

Though not perfect, LSTMs seem to be able to predict stock price behavior correctly most of the time. Note that you are making predictions roughly in the range of 0 and 1.0 (that is, not the true stock prices). This is okay, because you're predicting the stock price movement, not the prices themselves.

Implementation & Coding



```
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib
from sklearn.preprocessing import MinMaxScaler
from keras.layers import LSTM,Dense,Dropout
from sklearn.model_selection import TimeSeriesSplit
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.dates as mdates
from sklearn import linear_model
```

```
[6]  df_final = pd.read_csv("Reliance_Stock.csv",na_values=['null'],index_col='Date',parse_dates=True,infer_datetime_format=True)
```

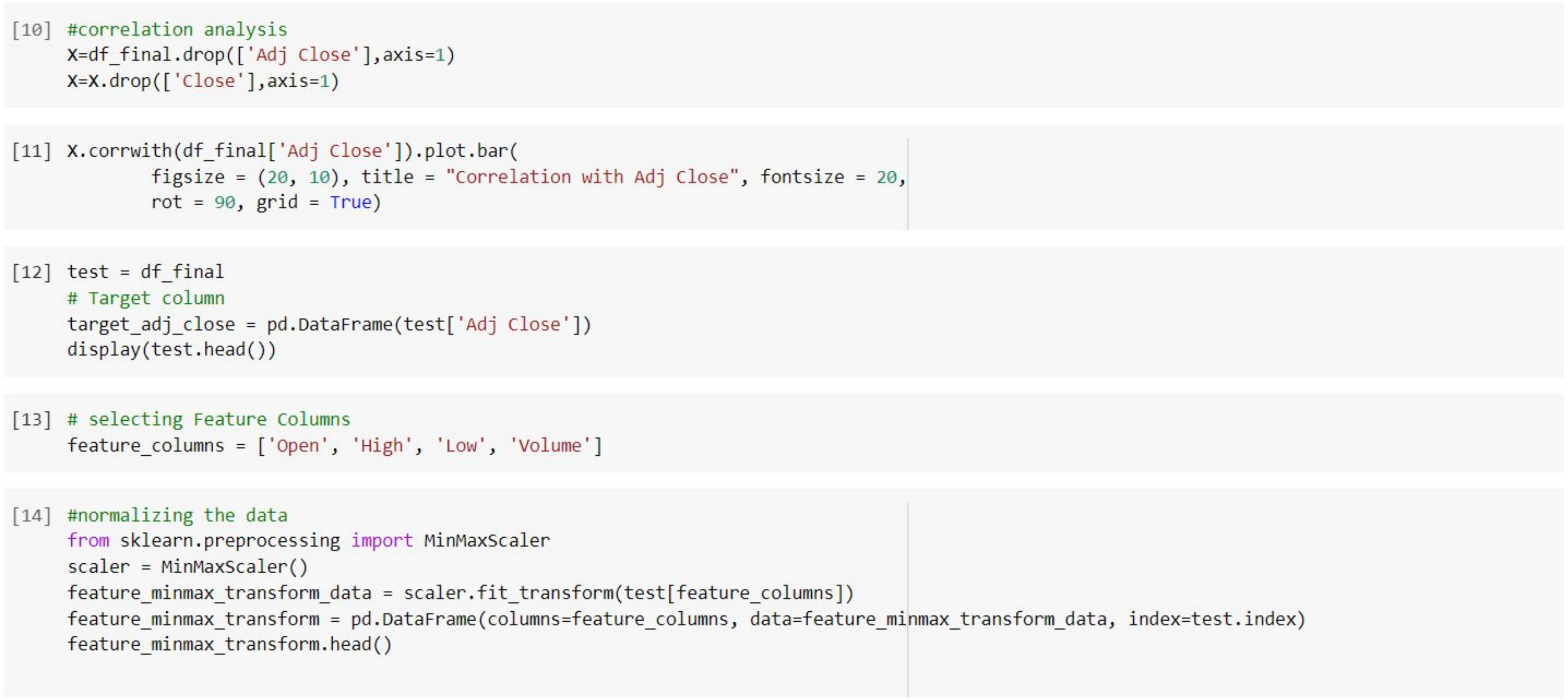
```
[7]  df_final.head()
```

```
[4]  df_final.shape
```

```
[5]  df_final.describe()
```

```
[8]  df_final.isnull().values.any()
```

```
[9]  df_final['Adj Close'].plot()
```



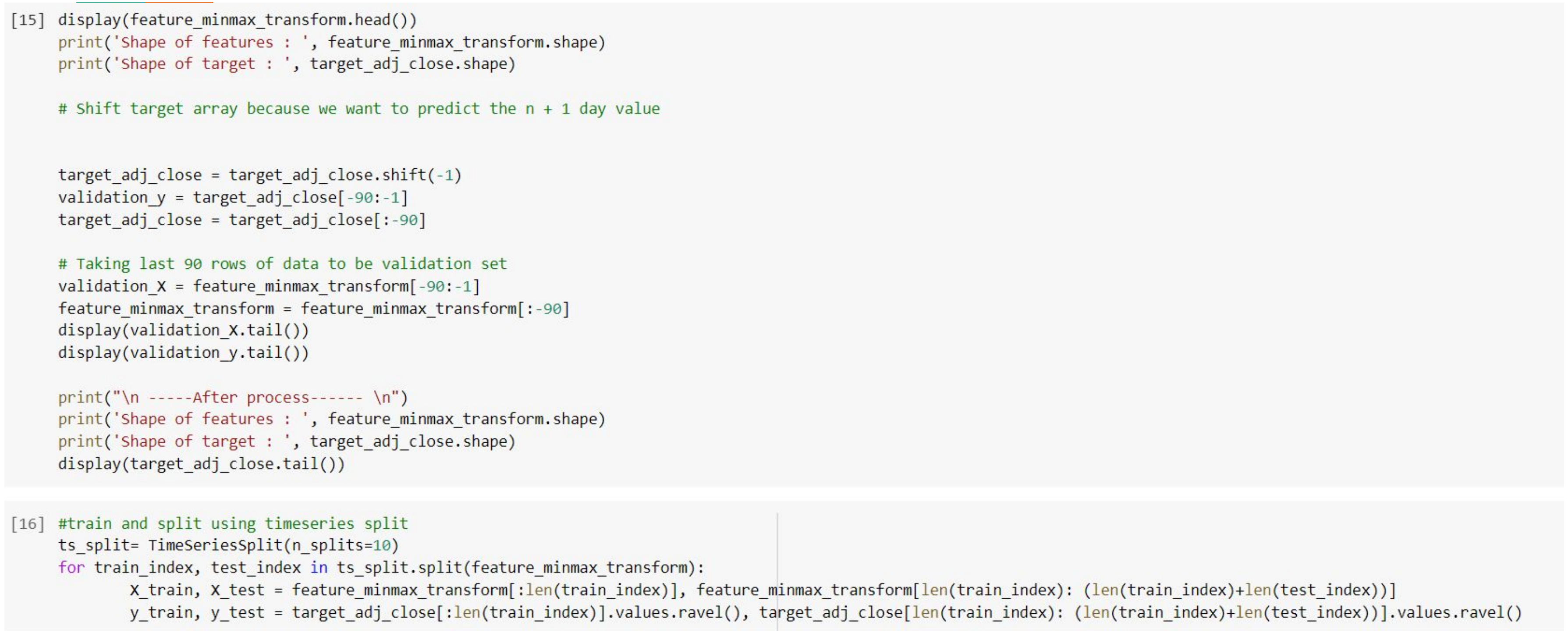
```
[10] #correlation analysis
     X=df_final.drop(['Adj Close'],axis=1)
     X=X.drop(['Close'],axis=1)
```

```
[11] X.corrwith(df_final['Adj Close']).plot.bar(
             figsize = (20, 10), title = "Correlation with Adj Close", fontsize = 20,
             rot = 90, grid = True)
```

```
[12] test = df_final
     # Target column
     target_adj_close = pd.DataFrame(test['Adj Close'])
     display(test.head())
```

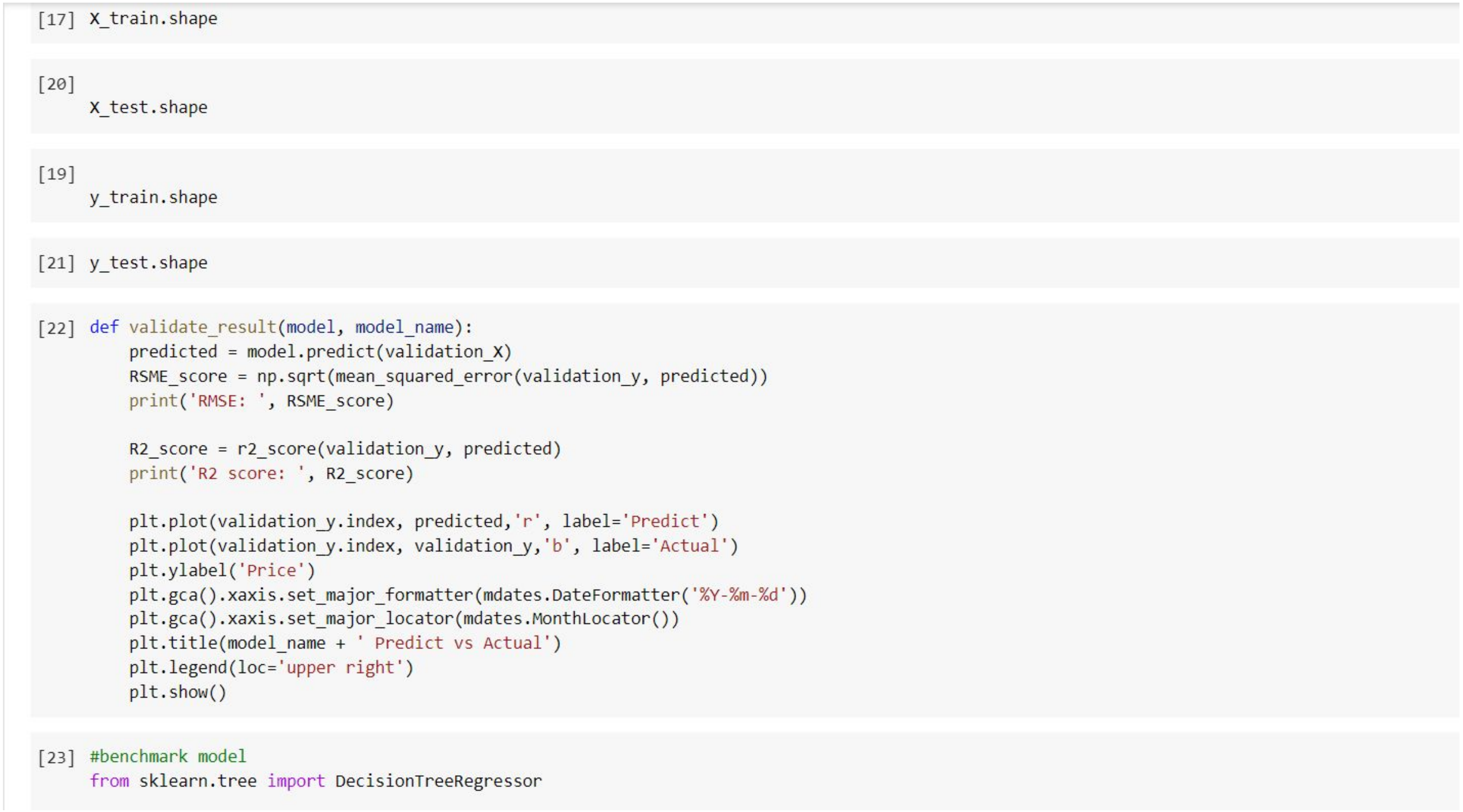
```
[13] # selecting Feature Columns
     feature_columns = ['Open', 'High', 'Low', 'Volume']
```

```
[14] #normalizing the data
     from sklearn.preprocessing import MinMaxScaler
     scaler = MinMaxScaler()
     feature_minmax_transform_data = scaler.fit_transform(test[feature_columns])
     feature_minmax_transform = pd.DataFrame(columns=feature_columns, data=feature_minmax_transform_data, index=test.index)
     feature_minmax_transform.head()
```



```
[15] display(feature_minmax_transform.head())
     print('Shape of features : ', feature_minmax_transform.shape)
     print('Shape of target : ', target_adj_close.shape)

     # Shift target array because we want to predict the n + 1 day value


     target_adj_close = target_adj_close.shift(-1)
     validation_y = target_adj_close[-90:-1]
     target_adj_close = target_adj_close[:-90]

     # Taking last 90 rows of data to be validation set
     validation_X = feature_minmax_transform[-90:-1]
     feature_minmax_transform = feature_minmax_transform[:-90]
     display(validation_X.tail())
     display(validation_y.tail())

     print("\n -----After process------ \n")
     print('Shape of features : ', feature_minmax_transform.shape)
     print('Shape of target : ', target_adj_close.shape)
     display(target_adj_close.tail())
```

```
[16] #train and split using timeseries split
     ts_split= TimeSeriesSplit(n_splits=10)
     for train_index, test_index in ts_split.split(feature_minmax_transform):
             X_train, X_test = feature_minmax_transform[:len(train_index)], feature_minmax_transform[len(train_index): (len(train_index)+len(test_index))]
             y_train, y_test = target_adj_close[:len(train_index)].values.ravel(), target_adj_close[len(train_index): (len(train_index)+len(test_index))].values.ravel()
```



```
[17] X_train.shape
```

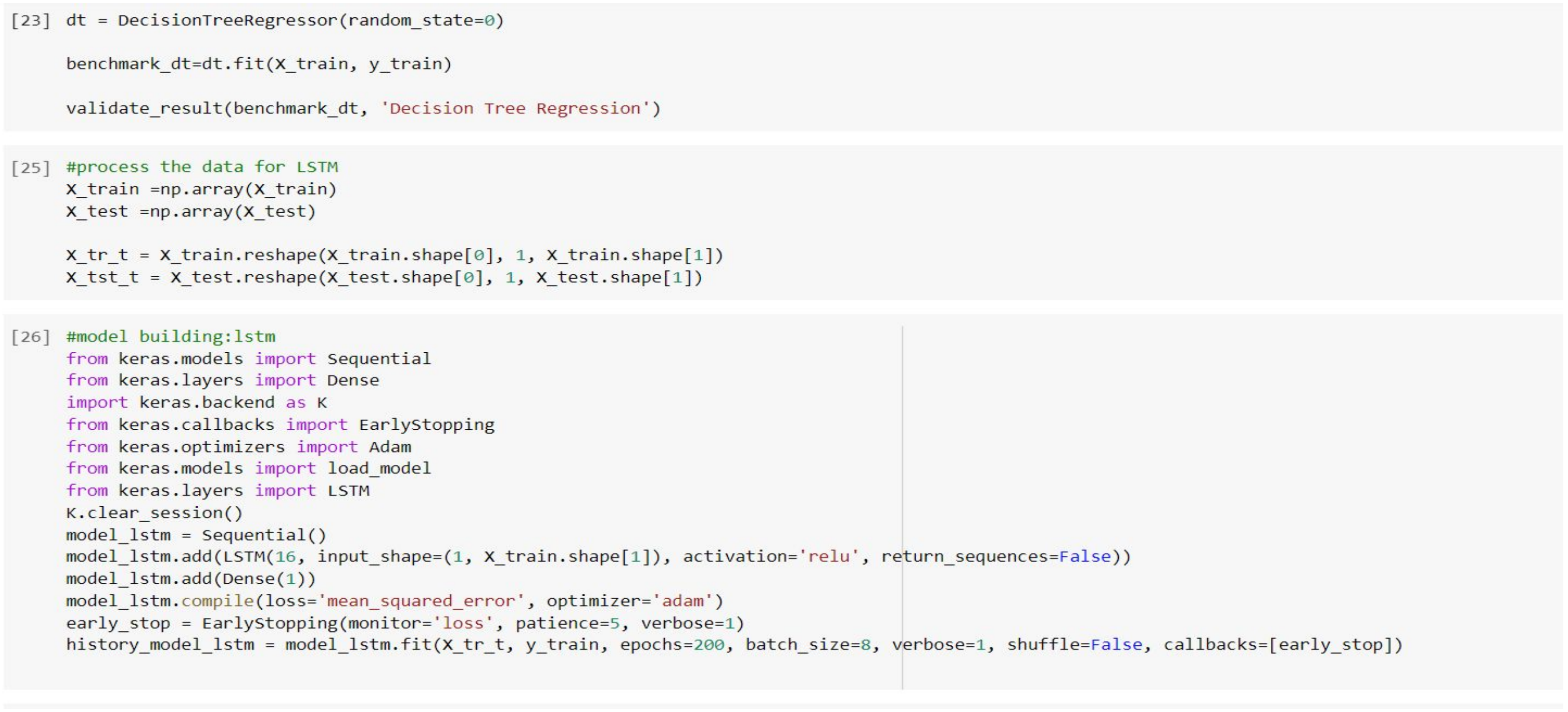
```
[20]
     X_test.shape
```

```
[19]
     y_train.shape
```

```
[21] y_test.shape
```

```
[22] def validate_result(model, model_name):
         predicted = model.predict(validation_X)
         RSME_score = np.sqrt(mean_squared_error(validation_y, predicted))
         print('RMSE: ', RSME_score)

         R2_score = r2_score(validation_y, predicted)
         print('R2 score: ', R2_score)

         plt.plot(validation_y.index, predicted,'r', label='Predict')
         plt.plot(validation_y.index, validation_y,'b', label='Actual')
         plt.ylabel('Price')
         plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
         plt.gca().xaxis.set_major_locator(mdates.MonthLocator())
         plt.title(model_name + ' Predict vs Actual')
         plt.legend(loc='upper right')
         plt.show()
```

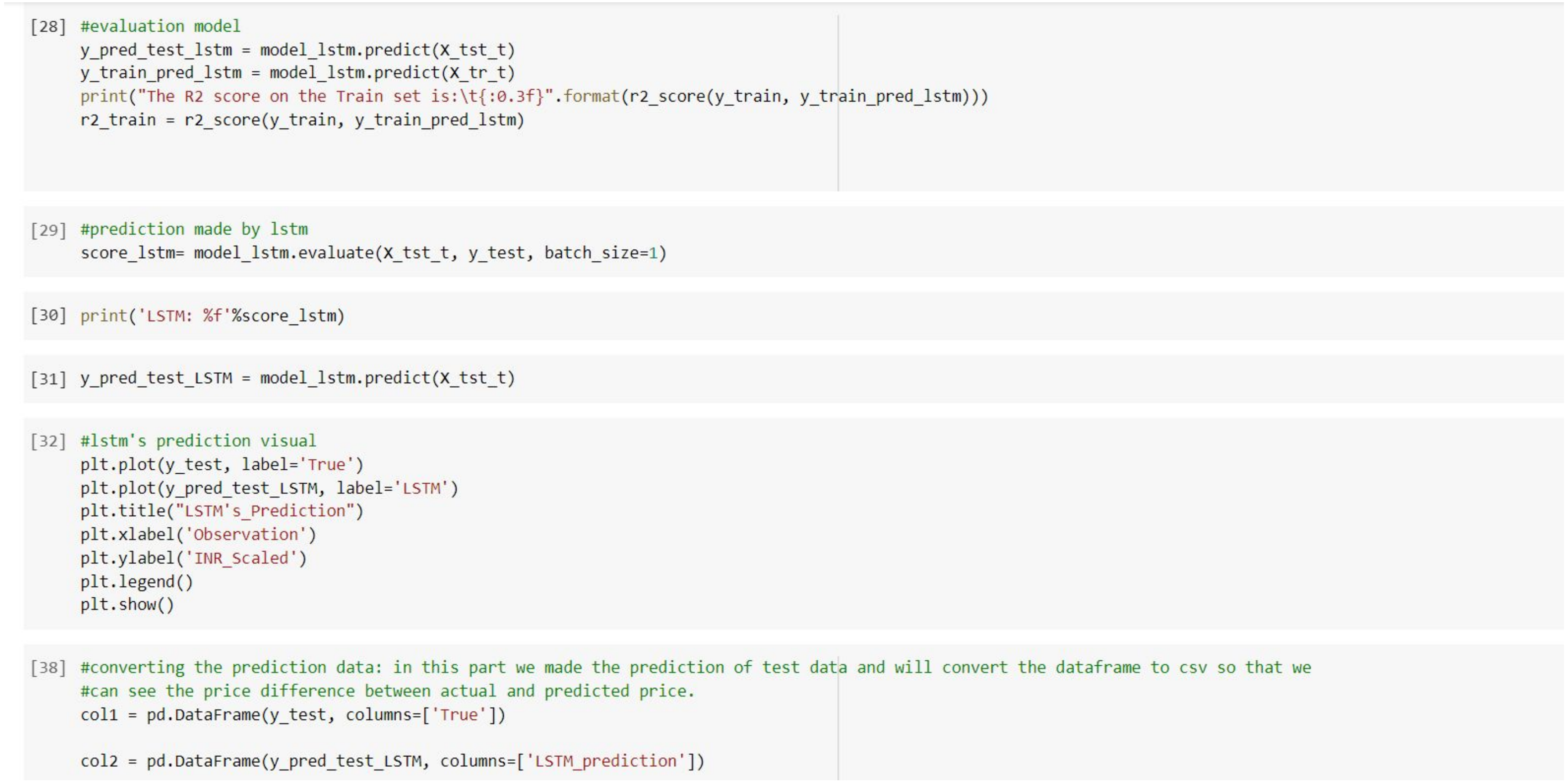
```
[23] #benchmark model
     from sklearn.tree import DecisionTreeRegressor
```



```
[23] dt = DecisionTreeRegressor(random_state=0)

     benchmark_dt=dt.fit(X_train, y_train)

     validate_result(benchmark_dt, 'Decision Tree Regression')
```

```
[25] #process the data for LSTM
     X_train =np.array(X_train)
     X_test =np.array(X_test)

     X_tr_t = X_train.reshape(X_train.shape[0], 1, X_train.shape[1])
     X_tst_t = X_test.reshape(X_test.shape[0], 1, X_test.shape[1])
```

```
[26] #model building:lstm
     from keras.models import Sequential
     from keras.layers import Dense
     import keras.backend as K
     from keras.callbacks import EarlyStopping
     from keras.optimizers import Adam
     from keras.models import load_model
     from keras.layers import LSTM
     K.clear_session()
     model_lstm = Sequential()
     model_lstm.add(LSTM(16, input_shape=(1, X_train.shape[1]), activation='relu', return_sequences=False))
     model_lstm.add(Dense(1))
     model_lstm.compile(loss='mean_squared_error', optimizer='adam')
     early_stop = EarlyStopping(monitor='loss', patience=5, verbose=1)
     history_model_lstm = model_lstm.fit(X_tr_t, y_train, epochs=200, batch_size=8, verbose=1, shuffle=False, callbacks=[early_stop])
```



```
[28] #evaluation model
     y_pred_test_lstm = model_lstm.predict(X_tst_t)
     y_train_pred_lstm = model_lstm.predict(X_tr_t)
     print("The R2 score on the Train set is:\t{:0.3f}".format(r2_score(y_train, y_train_pred_lstm)))
     r2_train = r2_score(y_train, y_train_pred_lstm)
```

```
[29] #prediction made by lstm
     score_lstm= model_lstm.evaluate(X_tst_t, y_test, batch_size=1)
```

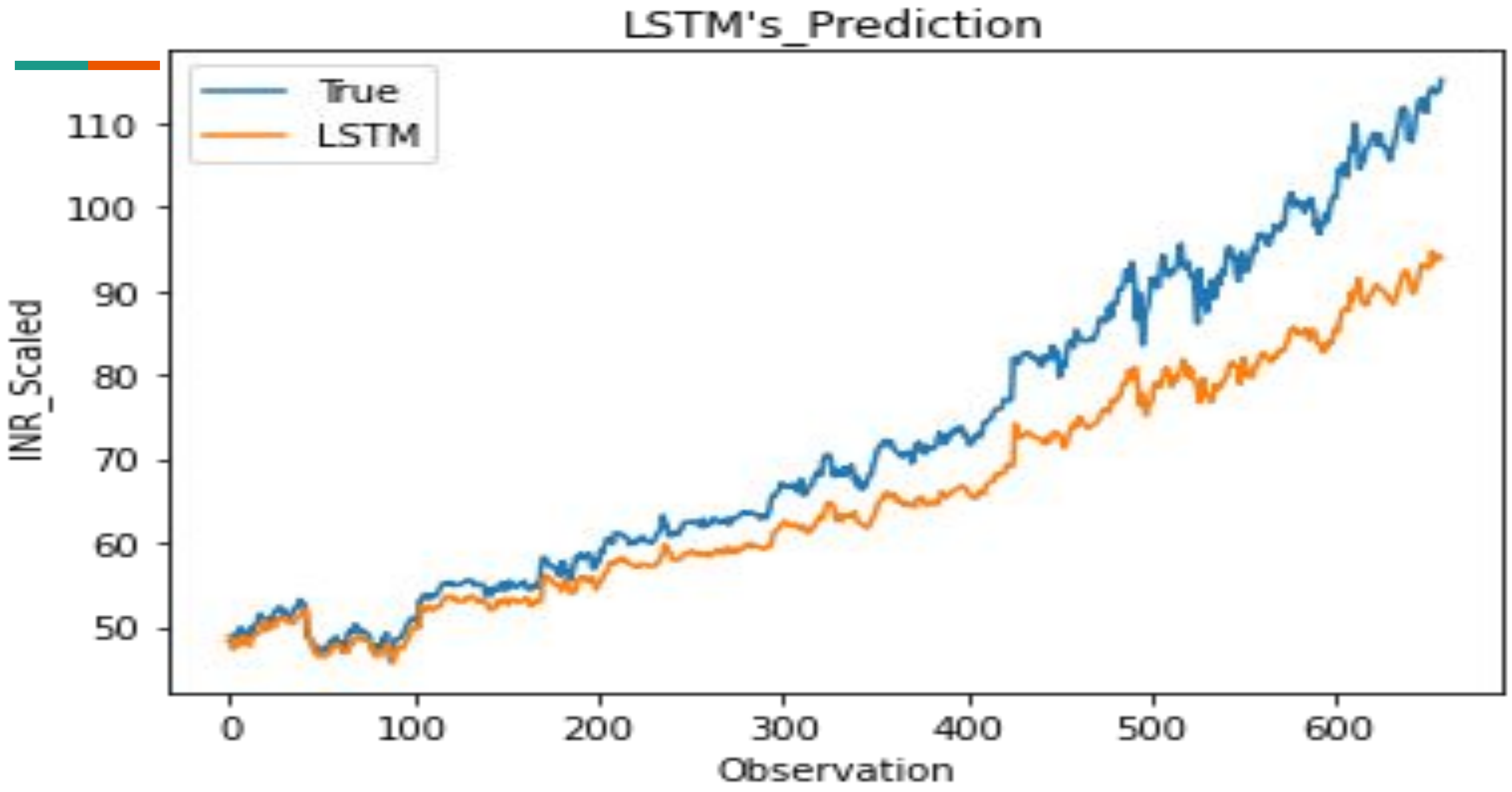
```
[30] print('LSTM: %f'%score_lstm)
```

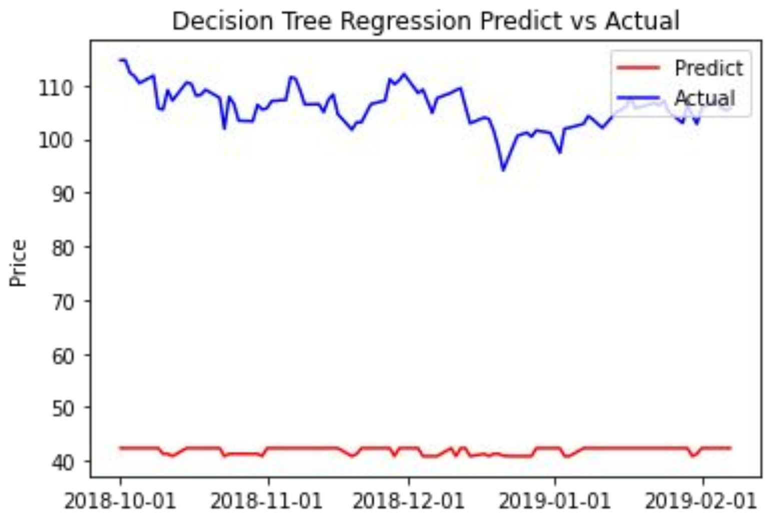
```
[31] y_pred_test_LSTM = model_lstm.predict(X_tst_t)
```

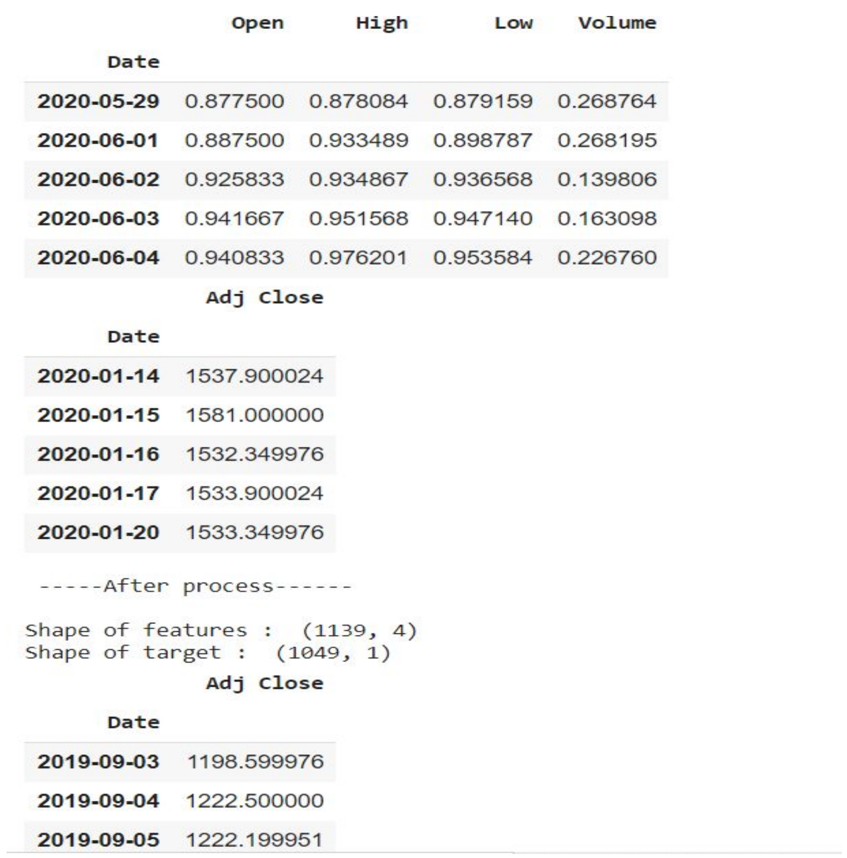
```
[32] #lstm's prediction visual
     plt.plot(y_test, label='True')
     plt.plot(y_pred_test_LSTM, label='LSTM')
     plt.title("LSTM's_Prediction")
     plt.xlabel('Observation')
     plt.ylabel('INR_Scaled')
     plt.legend()
     plt.show()
```

```
[38] #converting the prediction data: in this part we made the prediction of test data and will convert the dataframe to csv so that we
     #can see the price difference between actual and predicted price.
     col1 = pd.DataFrame(y_test, columns=['True'])

     col2 = pd.DataFrame(y_pred_test_LSTM, columns=['LSTM_prediction'])
```

OUTPUT


LSTM's_Prediction
True
LSTM
INR_Scaled
110
100
90
80
70
60
50
0
100
200
300
400
500
600
Observation





```
                Open      High       Low    Volume
Date
2020-05-29  0.877500  0.878084  0.879159  0.268764
2020-06-01  0.887500  0.933489  0.898787  0.268195
2020-06-02  0.925833  0.934867  0.936568  0.139806
2020-06-03  0.941667  0.951568  0.947140  0.163098
2020-06-04  0.940833  0.976201  0.953584  0.226760
              Adj Close
Date
2020-01-14  1537.900024
2020-01-15  1581.000000
2020-01-16  1532.349976
2020-01-17  1533.900024
2020-01-20  1533.349976

 -----After process------

Shape of features :  (1139, 4)
Shape of target :  (1049, 1)
              Adj Close
Date
2019-09-03  1198.599976
2019-09-04  1222.500000
2019-09-05  1222.199951
```

DEMO VIDEO

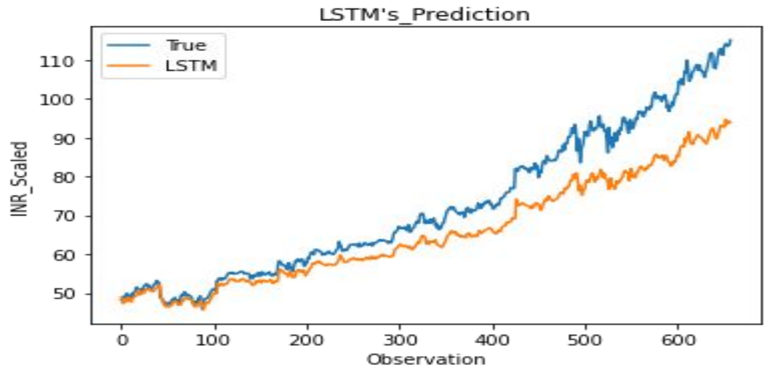
Demo

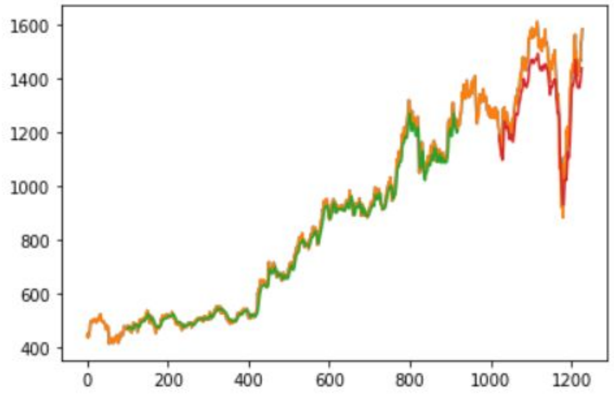
Result

The implementation of proposed LSTM model using python which predicts the future trends of Reliance_Stock share based on its historical data. The below visualization figure shows the visualization of Reliance_Stock prediction. In this report the implementation of an algorithm that predicts the stock price of a share for given time period, the below graph from our algorithm will show the predicted price. In the result shown in the below graph is the plotted form of our algorithm outcome by applying LSTM model.

Final Result

The given figure is drawn from the original dataset and show the result of the splitting of the train and test dataset. X-axis is the share price and y-axis is the number of days.





The given figure above is the graph representing the blue line is the actual stock prices and an orange line is the prediction of next few days stock trends which will be our final output.You can clearly see that our stock prediction model is able to capture the overall trend.

Conclusion

The popularity of stock market trading is growing rapidly, which is encouraging researchers to find out new methods for the prediction using new techniques. The forecasting technique is not only helping the researchers but it also helps investors and any person dealing with the stock market. In order to help predict the stock indices, a forecasting model with good accuracy is required. In this work, we have used one of the most precise forecasting technology using Long Short-Term Memory unit which helps investors, analysts or any person interested in investing in the stock market by providing them a good knowledge of the future situation of the stock market.

References

- https://en.wikipedia.org/wiki/Recurrent_neural_network
- https://corporatefinanceinstitute.com/resources/knowledge/trading-investing/stock-price/#:~:text=A%20stock%20price%20is%20a,and%20within%20the%20company%20itself
- https://www.investopedia.com/articles/investing/082614/how-stock-market-works.asp
- https://jfin-swufe.springeropen.com/articles/10.1186/s40854-019-0131-7
- https://www.irjet.net/archives/V5/i3/IRJET-V5I3788.pdf
- https://www.researchgate.net/publication/321503983_Stock_price_prediction_using_LSTM_RNN_and_CNN-sliding_window_model
- https://www.researchgate.net/publication/348390803_Stock_Price_Prediction_Using_LSTM
- https://link.springer.com/article/10.1007/s13042-019-01041-1
- http://cs230.stanford.edu/projects_winter_2020/reports/32066186.pdf
- https://paperswithcode.com/task/stock-price-prediction/codeless
- Lakshminarayanan, S.K.; McCrae, J. A comparative study of svm and lstm deep learning algorithms for stock market prediction. In Proceedings of the 27th AIAI Irish Conference on Artificial Intelligence and Cognitive Science (AICS 2019), Galway, Ireland, 5–6 December 2019.
- Achkar, R.; Elias-Sleiman, F.; Ezzidine, H.; Haidar, N. Comparison of BPA-MLP and LSTM-RNN for stock prediction. In Proceedings of the 2018 6th International Symposium on Computational and Business Intelligence (ISCBI), Basel, Switzerland,27–29 August 2018.
- Skehin, T.; Crane, M.; Bezbradica, M. Day ahead forecasting of FAANG stocks using ARIMA, LSTM networks and wavelets. In Proceedings of the 26th AIAI Irish Conference on Artificial Intelligence and Cognitive Science (AICS 2018), Dublin, Ireland, 6–7 December 2018.
- Jin, Z.; Yang, Y.; Liu, Y. Stock closing price prediction based on sentiment analysis and LSTM. Neural Comput. Appl. 2020, 32, 9713–9729.

THANK YOU