Institute for Advanced Computing and Software Development (IACSD)
Akurdi, Pune – 411044

Documentation On

"Market Analysis and Prediction Of Indian Stock Companies"

Submitted By:

Group No: 1

Abhay Sharma 220341225001

Swarnim Manvendra Vaidya 220341225055

Mr. Prashant Karhale
Centre Coordinator

Mr. Akshay Tilekar
Project Guide

INDEX

LIST OF TABLES

LIST OF FIGURES

Chapter No. 1

Introduction

1.1 Problem Statement

Market Analysis and Prediction Of Indian Stock Market Company

A stock exchange is an exchange where stockbrokers and traders can buy and sell securities, such as shares of stock, bonds, and other financial instruments. The National Stock Exchange of India Limited (NSE) is located in Mumbai, Maharashtra. The stock market value changes. The stock value changes every minute based on buys and sells of stock in market. As a technical investor, you would look at historical price patterns and form an opinion about market trends. Accordingly, you will decide whether you want to buy the stock or sell it.

We have tried to predict the trend of Close value of Nifty 50 stock of NSE which can used to analyse and understand the trend of stock market

1.2 Abstract

The project analyses Indian Stock Market Company using historical values. We have used Nifty 50 Dataset of stock market from 2017 to starting of 2021 which is per minute per hour data. Stock market changes are dependent various factors and very difficult to predict accurately.

We are using various ML Algorithm like Random Forest Regressor, Gradient Boosting Regressor and DL model like LSTM to predict the trend of closing value of company. We also have used Facebook Prophet Library for analysing and predicting close value of Nifty 50 stock in NSE. We have deployed the project on AWS and GitHub Repository.

1.3 Product Scope

The stock market changes are difficult to predict based on various factors

- Supply and Demand
- Inflation
- Domestic and Global changes
- Interest rate by RBI

- People Sentiment
- Social and Economic changes in India and International market

The factors make it difficult to predict the value of stock price but we have used historical data of Nifty 50 to predict trend of close value of Nifty 50.

The algorithm like Random Forest and Gradient Boosting Regressor and DL model like LSTM uses this historical value of stock to predict trend of close value of stock. This prediction can be used by stock market investor to analyse the trend take decision of buying or selling stock within short term.

1.4 Aims and Objective

- To use ML algorithm like Random Forest Regressor and Gradient Boosting Regressor to predict the trend of close value of stock
- To use DL model LSTM to predict trend of close value of stock
- To use Library like Facebook Prophet to predict the trend of close value of stock
- Evaluation metrics like r2, MAE, MSE are used to compare these models which predict value closer to the actual value of Nifty 50 stock
- The predicted value can be used by Stock Market Investor to analyse the trend and take decision of buying and selling of stock

Chapter No 2

Overall Description

2.1 Workflow of Project

Following steps were followed to solve the problem statement and predict the close value of Nifty 50 stock

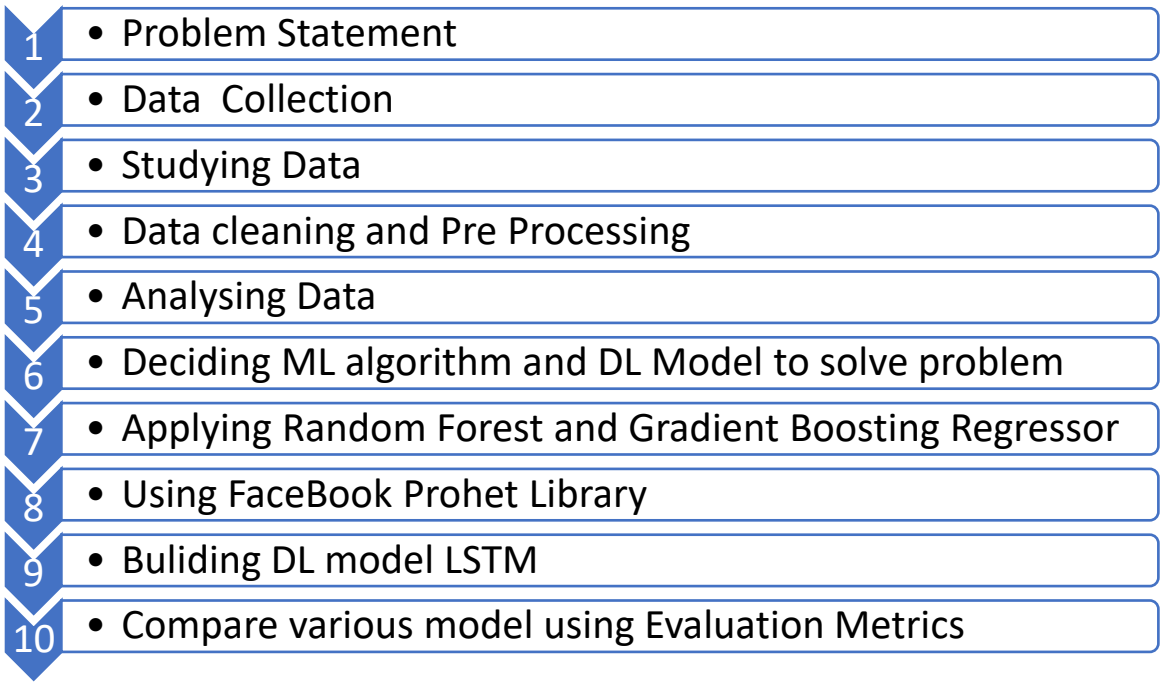


Fig 2.1 Flowchart for workflow of project

2.2.1 Data Description

The original dataset of Nifty 50 is csv file name as NIFTY_50__EQ__INDICES__NSE__MINUTE.csv available on Kaggle website. The data size is of 370740 rows consisting of 6 columns namely timestamp, open, high, low, close and volume. The dataset consists of per minute data from 9.15 am when NSE stock market opens to 3.29 pm when market closes. The data is from 20170102 09:15:00 am to 20210101 15:29:00 pm

2.2.2 Data Pre-processing and Cleaning

1. Removing UTC time zone from timestamp

The timestamp consist of datetime and “+5.30” UTC stamp which was remove to convert the timestamp to datetime format for further processing

2. Checking of Na or Null values

The dataset consisted of 63 rows having Na values at different timestamp present in original dataset

3. Filling the null values

The Na values were filled using ffill method available in Pandas library to fill the Na rows

2.3 Exploratory Data Analysis of Data

Various graphs plots were used provided in Pandas Library to analyse the Close value of Nifty 50 stock

1. Line graph of close value

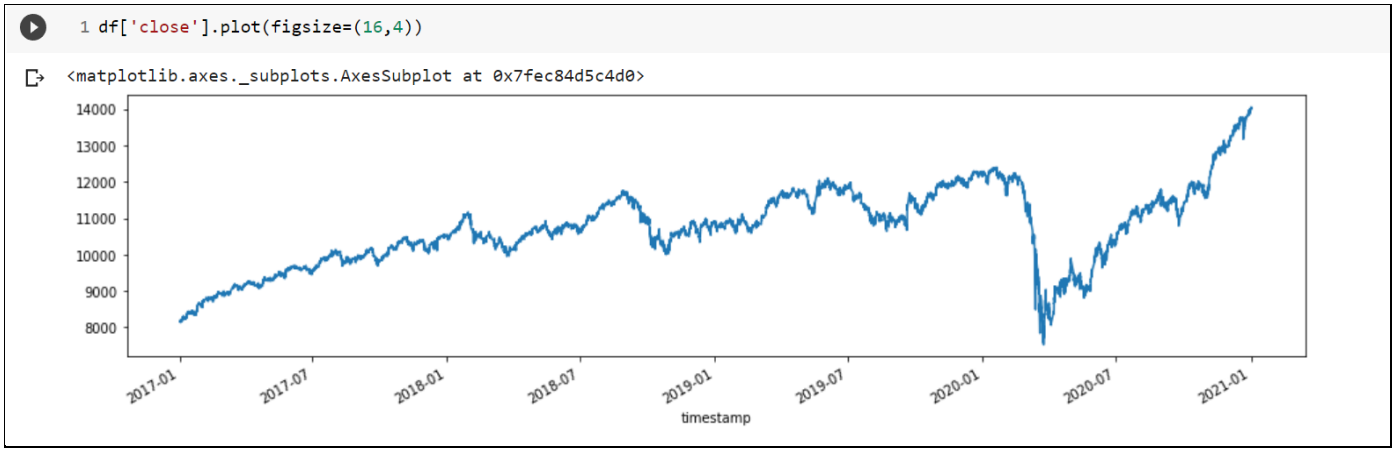


Fig 2.3.1 Line Graph of close value

2. Quarterly Histogram of close value

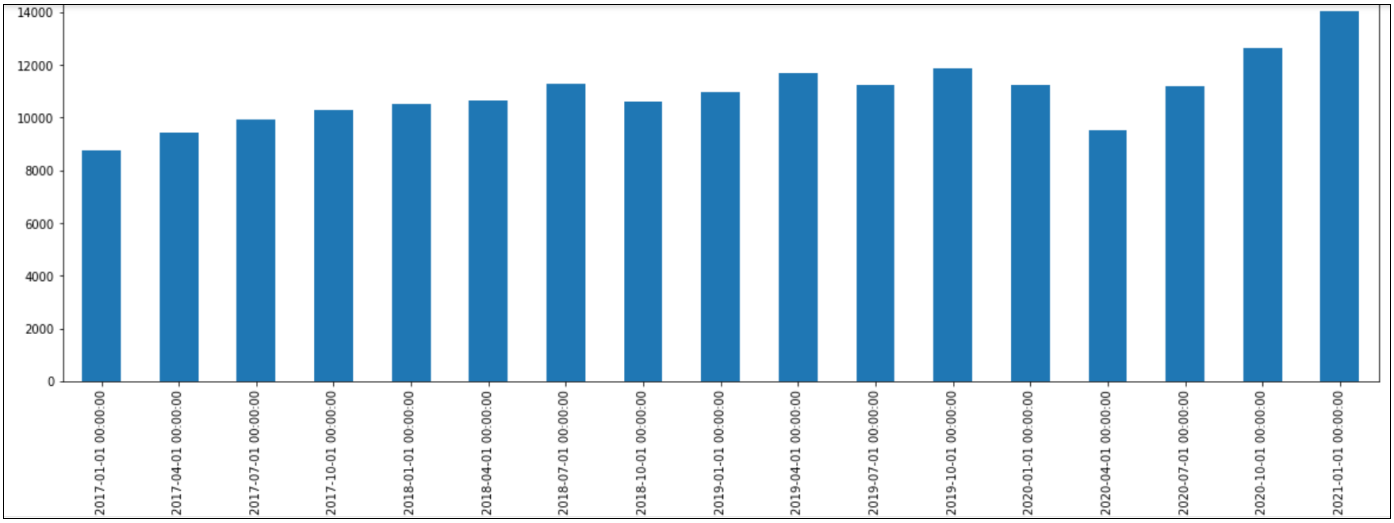


Fig 2.3.2 Quarterly Histogram of close value

3. Box Plot to determine median, outlier and IQR

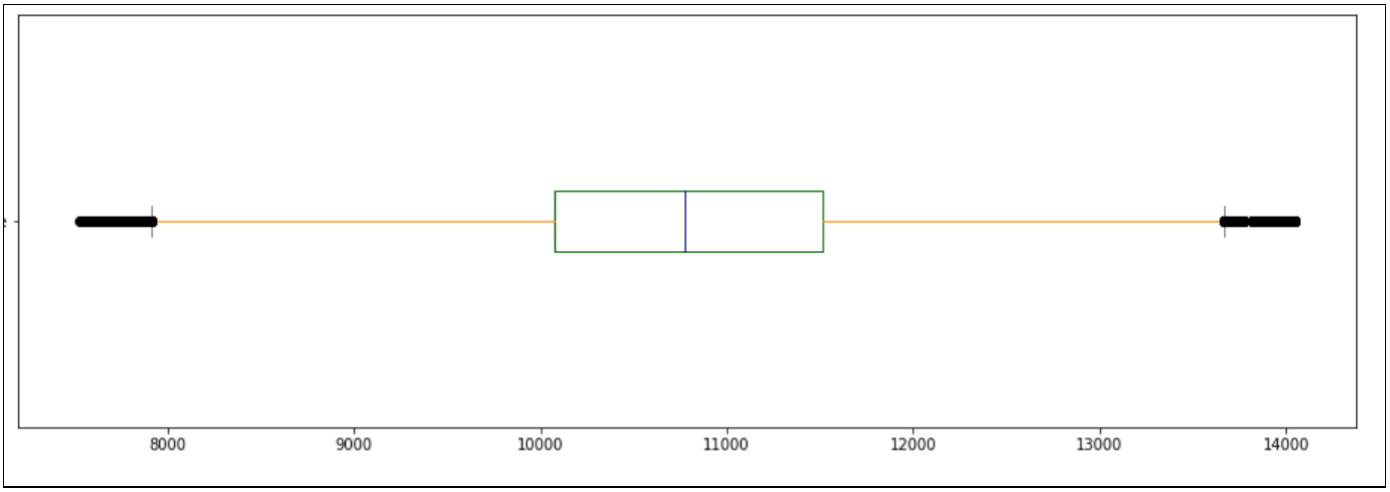


Fig 2.3.3 Box Plot

Q1 = Rs 10100

Q2 = Median value of Rs Rs 10900

Q3 = Rs 11600

IQR = Q3 Q1 = Rs 1500

Min Valid Value = Q1 1.5 IQR = Rs 7850

Max Valid Value = Q3 + 1.5 IQR = Rs 13850

Outlier = value less than Rs 7850 and more than Rs 13850

4. Various other function were used to find Min, Max values, Median and other statistics

For analysing data

5. Rolling Average

Rolling average is also calculated with rolling window of 10, 20 and 50 min to analyse the line graph previously plotted. This helps to smoothen the graph. Smoothing of graph is used to analyse the trend of stock market for short term which helps to decide to buy or sell the stock

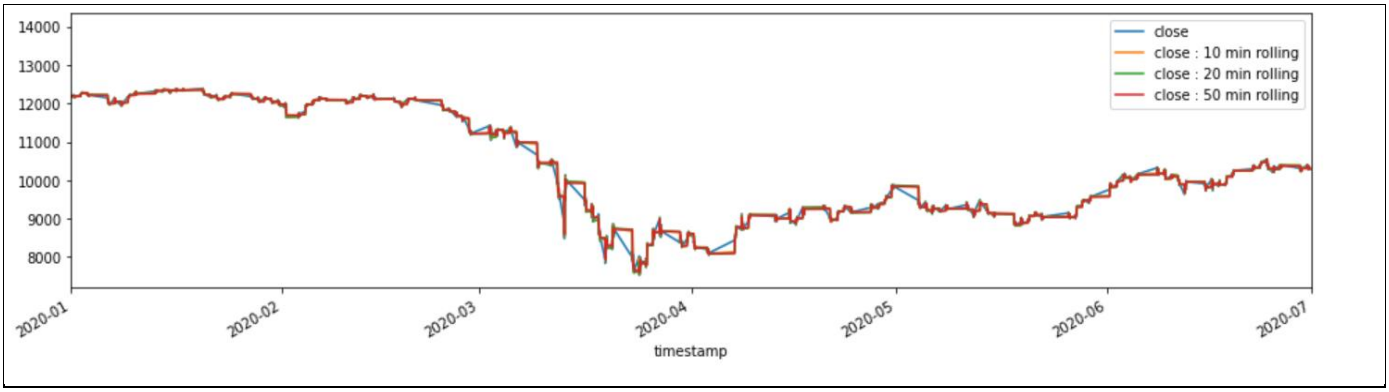


Fig 2.3.5 Rolling Average of line graph of close value

6. Exponential Weighted Moving Average

Exponential Weighted Moving Average (EWMA) is a type of moving average (MA) that places a greater weight and significance on the most recent data points. The exponential moving average is also referred to as the exponentially weighted moving average. An

exponentially weighted moving average reacts more significantly to recent price changes than a simple moving average simple moving average (SMA), which applies an equal weight to all observations in the period.

EWMA Formula

The EWMA's simple mathematical formulation described below:

$$EWMA_t = \alpha * r_t + (1 - \alpha) * EWMA_{t-1}$$

Where:

- **Alpha** = The weight decided by the user
- **r** = Value of the series in the current period

Fig 2.3.6 EWMA Formula

EWMA graph for close value of Nifty 50 stock

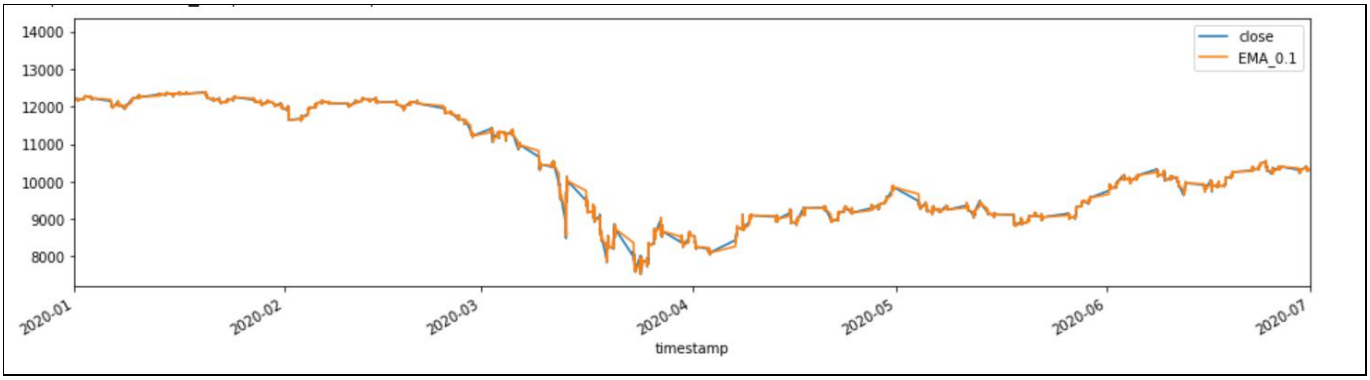


Fig 2.3.7 EWMA graph of close value

Chapter No 3

Model Building

3 Model Building

Following model were build for calculating prediction values of close trend of Nifty 50 stock

Model Name	Algorithm	Data size	X column or features	r2	mse	mae
model 1	Random Forest Regressor	50000	close t_1 to close t_4 lag value 1 min to 4 min	0.999957	41.33685	4.098979
model 2	Gradient Boosting Regressor	50000		0.999919	77.85788	6.222007
trial model 1	Random Forest Regressor	10000	ewm 0.1 and ewm 0.2	0.999784	22.78149	2.528047
trial model 2	Random Forest Regressor	10000	close t_1 (60 min)	0.971942	2978.179	34.86378
trial model 3	Random Forest Regressor	10000	close t_1 (180 min)	0.911749	9014.927	62.37577
trial model 4	Random Forest Regressor	10000	close t_1 (374 min)	0.853046	12680.43	68.51588
model 3	Random Forest Regressor	50000	close t_1 (374 min)	0.986207	13000.9	80.89605
model 4	Gradient Boosting Regressor	50000	close t_1 (374 min)	0.989716	9693.396	73.19261
model 5	Fb Prophet	50000	Done by library	0.986225	12984.58	80.97747
model 6	LSTM	50000	single layer			
model 7	LSTM	50000	multi layer			

Table 3.1 Models build

Following model were used to compare and evaluate the method

Model Name	Algorithm	Data size	X column or features
model 1	Random Forest Regressor	50000	close t_1 to close t_4 lag value 1 min to 4 min
model 2	Gradient Boosting Regressor	50000	
model 3	Random Forest Regressor	50000	close t_1 (374 min)
model 4	Gradient Boosting Regressor	50000	close t_1 (374 min)
model 5	Fb Prophet	50000	Done by library
model 6	LSTM	50000	single layer
model 7	LSTM	50000	multi layer

Table 3.1.1 Models Compare

3.1. Random Forest Regressor (model 1)

Since a random forest is an ensemble of decision trees, it has

- Lower variance
- High Accuracy
- Strong Algorithm
- Automates missing values

We have created 2 models of Random Forest Regressor model namely Model 1 and Model 3

Model 1 of Random Forest Regressor consists of 50000 rows of close column. The features were created for Model using lag values with help of shift() function available in Pandas Library. Shift function was used from 1 min to 4 min to create features from close t_1, close t_2, close t_3 and close t_4 resp. These close values were added in X columns and Y column which is target class consists of close value of stock.

```
[92]  1 df_rfe['close_t_1'] = df_rfe['close'].shift(1) # shift one value below
      2 df_rfe['close_t_2'] = df_rfe['close'].shift(2) # shift one value below
      3 df_rfe['close_t_3'] = df_rfe['close'].shift(3) # shift one value below
      4 df_rfe['close_t_4'] = df_rfe['close'].shift(4) # shift one value below
      5 #df_rfe['close_t_5'] = df_rfe['close'].shift(5) # shift one value below

[93]  1 df_rfe
```

timestamp	close	close_t_1	close_t_2	close_t_3	close_t_4
2017-01-02 09:15:00	8189.55	NaN	NaN	NaN	NaN
2017-01-02 09:16:00	8189.95	8189.55	NaN	NaN	NaN
2017-01-02 09:17:00	8173.70	8189.95	8189.55	NaN	NaN
2017-01-02 09:18:00	8177.55	8173.70	8189.95	8189.55	NaN
2017-01-02 09:19:00	8174.40	8177.55	8173.70	8189.95	8189.55
...	...	...	...	...	...
2021-01-01 15:25:00	14019.00	14022.70	14018.20	14018.80	14007.30

Fig 3.1 Features created

Then this becomes regular regression problem which cloud be solved with help Random Forest Regressor. GridSearch CV was used to fine tune the Hyper parameters to find the best parameters. Once best parameters were found then model was again trained to fit with best parameters.

Then evaluation parameters like R2, Mean Squared Error (MSE) and Mean Absolute Error (MAE) were calculated using SkLearn Library. These parameters are used to compare different models formed to find best model

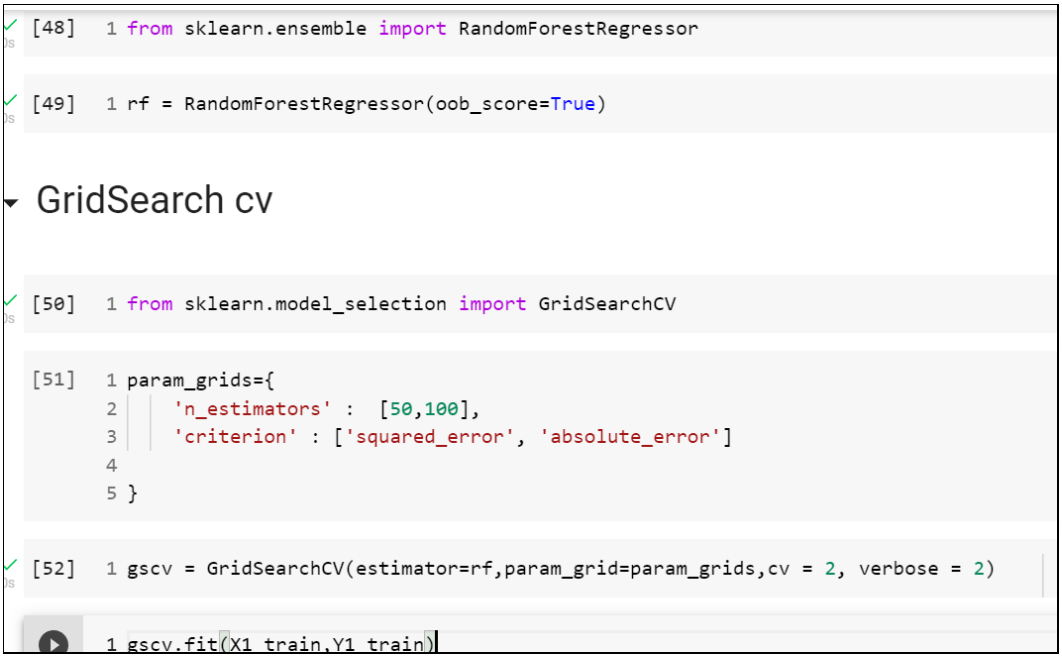


Fig3.1.2 Screenshot of Random Forest Regressor

3.2. Gradient Boosting Regressor (model 2)

It is Ensemble learning

- give a more accurate prediction
- It is much more flexible than other algorithms
- It does not require data pre-processing because it is suitable for both numeric as well as categorical variables.
- it handles missing data automatically.

Model 2 of Gradient Boosting Regressor consists of 50000 rows of close column. The features were created for Model using lag values with help of shift() function available in Pandas Library. Shift function was used from 1 min to 4 min to create features from close t_1, close t_2, close t_3 and close t_4 resp. These close values were added in X columns and Y column which is target class consists of close value of stock.

```
[92]  1 df_rfe['close_t_1'] = df_rfe['close'].shift(1) # shift one value below
      2 df_rfe['close_t_2'] = df_rfe['close'].shift(2) # shift one value below
      3 df_rfe['close_t_3'] = df_rfe['close'].shift(3) # shift one value below
      4 df_rfe['close_t_4'] = df_rfe['close'].shift(4) # shift one value below
      5 #df_rfe['close_t_5'] = df_rfe['close'].shift(5) # shift one value below

[93]  1 df_rfe
```

timestamp	close	close_t_1	close_t_2	close_t_3	close_t_4
2017-01-02 09:15:00	8189.55	NaN	NaN	NaN	NaN
2017-01-02 09:16:00	8189.95	8189.55	NaN	NaN	NaN
2017-01-02 09:17:00	8173.70	8189.95	8189.55	NaN	NaN
2017-01-02 09:18:00	8177.55	8173.70	8189.95	8189.55	NaN
2017-01-02 09:19:00	8174.40	8177.55	8173.70	8189.95	8189.55
...	...	...	...	...	...
2021-01-01 15:25:00	14019.00	14022.70	14018.20	14018.80	14007.30

Fig 3.2.1 Features created Gradient Boosting Regressor

X features consist of close t_1, close t_2, closet_3, and close t_4 and Y column is target class consisting of close value of stock. This becomes a regular regression problem and we are now using Gradient Boosting Regressor to predict the trend of close value of the stock.

GridSearch CV is used to fine tune the Hyper Parameters. The best parameters are now used to build model to solve the regression problem

```
from sklearn.ensemble import GradientBoostingRegressor

gbr = GradientBoostingRegressor()

param_grids={
    'loss' : ['squared_error', 'absolute_error'],
    'n_estimators' : [50,100],
    'criterion' : ['friedman_mse','squared_error']
}

from sklearn.model_selection import GridSearchCV

gscv_gbr=GridSearchCV(estimator=gbr,param_grid=param_grids,cv = 2,verbose = 2)

gscv_gbr.fit(X1_train,Y1_train)
```

Fig 3.2.2 GridSearch CV for Gradient Boosting Regressor

3.3. Random Forest Regressor (model 3)

Similar to the model 1 we have created we also have build model 3 with different features. The model 3 has X features close value of stock shifted by 374 min and Y column is target class
GridSearch CV is used to fine tune Hyper Parameters and using these best parameters we again fit model to get model to predict the trend of close value of stock

```
[160]  1 df_1['close t_1'] = df_1['close'].shift(374)

[161]  1 df_1
```

timestamp	close	close t_1
2020-06-24 14:24:00	10410.25	NaN
2020-06-24 14:25:00	10416.45	NaN
2020-06-24 14:26:00	10408.75	NaN
2020-06-24 14:27:00	10404.50	NaN
2020-06-24 14:28:00	10407.30	NaN
...	...	...
2021-01-01 15:24:00	14022.70	13975.15
2021-01-01 15:25:00	14019.00	13978.15
2021-01-01 15:26:00	14019.10	13978.45
2021-01-01 15:27:00	14018.15	13967.40

Fig3.3.1 Features created for Random Forest Regressor Model 3

4. Gradient Boosting Regressor (model 4)

Similar to the model 2 of Gradient Boosting Regressor we have build this model 4 but the X features in this model has less features . X columns consisting of close value of stock shifted by 374 min

```
[160]  1 df_1['close t_1'] = df_1['close'].shift(374)

[161]  1 df_1
```

timestamp	close	close t_1
2020-06-24 14:24:00	10410.25	NaN
2020-06-24 14:25:00	10416.45	NaN
2020-06-24 14:26:00	10408.75	NaN
2020-06-24 14:27:00	10404.50	NaN
2020-06-24 14:28:00	10407.30	NaN
...	...	...
2021-01-01 15:24:00	14022.70	13975.15
2021-01-01 15:25:00	14019.00	13978.15
2021-01-01 15:26:00	14019.10	13978.45
2021-01-01 15:27:00	14018.15	13967.40

Fig 4.1.1 Feature created for Gradient Boosting Regressor Model 4

All other process is same as model 2, use GridSearch CV to find best parameters and build model to predict close value trend

5. Facebook Prophet (model 5)

Facebook Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is robust to missing data and shifts in the trend, and typically handles outliers well.
We have used this Library to build model which predict values for future data frame. Library is very easy to use and takes cares of everything that we did manually in Random Forest and Gradient Boosting Regressor

```
[208]  1 forecast = model.predict(future)
       2 forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

	ds	yhat	yhat_lower	yhat_upper
50008	2021-01-12 15:26:00	14271.801612	13619.086453	14863.996480
50009	2021-01-13 15:26:00	14307.776869	13525.622111	15007.957450
50010	2021-01-14 15:26:00	14340.579687	13513.359097	15104.088972
50011	2021-01-15 15:26:00	14368.398894	13411.403165	15250.676682
50012	2021-01-16 15:26:00	14631.288007	13574.345512	15654.216179

Fig 5.1 Facebook prophet screenshot

Various graphs are plotted using library such as follows

Fig 5.2 Plot in library

6. LSTM

Long Short Term Memory Network is an advanced RNN, a sequential network, that allows information to persist. To solve the problem of Vanishing and Exploding Gradients in a Deep Recurrent Neural Network, many variations were developed. One of the most famous of them is the Long Short Term Memory Network (LSTM). In concept, an LSTM recurrent unit tries to “remember” all the past knowledge that the network is seen so far and to “forget” irrelevant data. This is done by introducing different activation function layers called “gates” for different purposes. A recurrent neural network is also known as RNN is used for persistent memory.LSTM is very sensitive to the scale of the data, Here the scale of the Close value is in a kind of scale, we should always try to transform the value.
Here we will use min-max scalar to transform the values from 0 to 1.
We should reshape so that we can use fit transform. A LSTM Memory Network consists of four different gates : Forget Gate(f), Input Gate(i), Input Modulation Gate(g) & Output Gate(o).

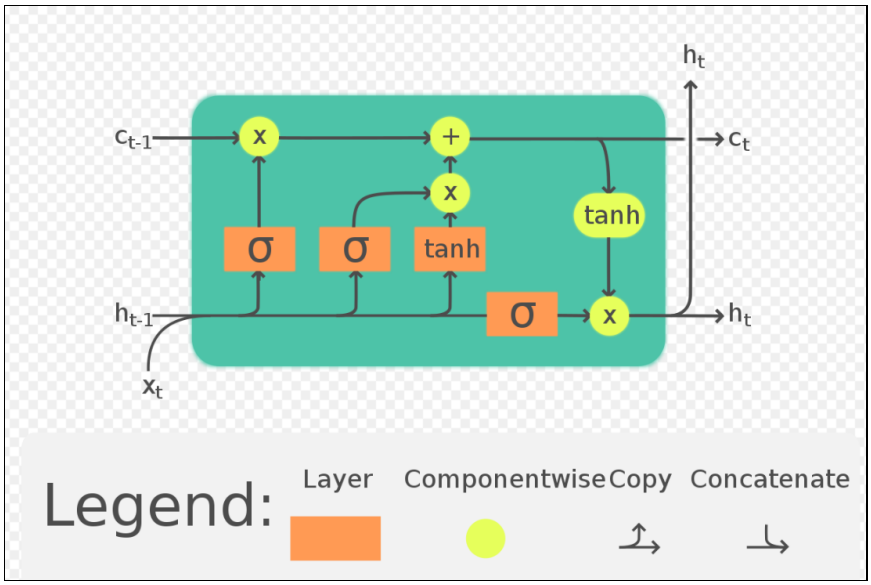


Fig 6.1 LSTM Diagram

Here in our Project we have created 2 LSTM models:
1st with 1 Neural Node

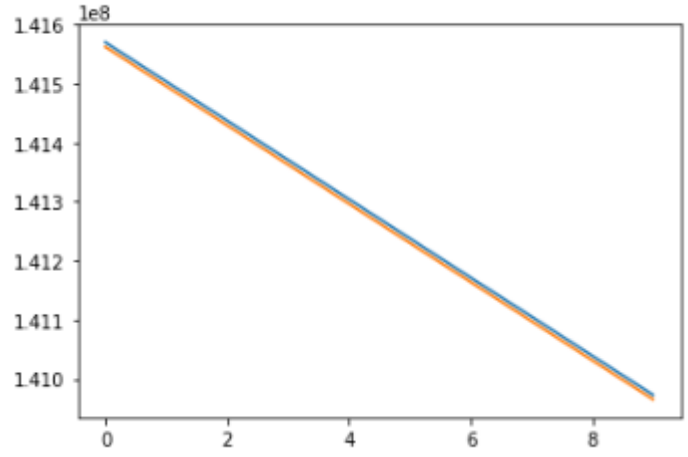
```
model = keras.Sequential()

# Adding LSTM layers
model.add(keras.layers.LSTM(units=1,return_sequences = False,
                            input_shape= (X_train.shape[1],X_train.shape[2])))
# Adding the output layer
model.add( keras.layers.Dense(1, activation='linear'))

# Compiling the LSTM
model.compile(optimizer = 'adam', loss = 'mean_squared_error')
```

It's Loss & Validation Loss curves are as Follows:

```
[<matplotlib.lines.Line2D at 0x7ffb227b4b90>]
```



2nd with 10 Neural Nodes:

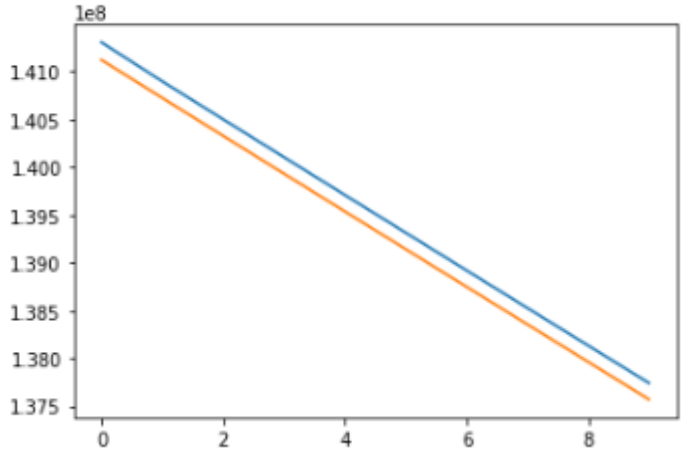
```
model = keras.Sequential()

# Adding LSTM layers
model.add(keras.layers.LSTM(units=10,return_sequences = True,
                            input_shape= (X_train.shape[1],X_train.shape[2])))

model.add(keras.layers.LSTM(units=5,return_sequences = False))
# Adding the output layer
model.add( keras.layers.Dense(1, activation='linear'))

# Compiling the LSTM
model.compile(optimizer = 'adam', loss = 'mean_squared_error')
```

```
[<matplotlib.lines.Line2D at 0x7ffb203b04d0>]
```



Chapter 4
Specification Requirement

4.1 Hardware Requirement

- Ryzen 5000 series or Intel I5 series Processor
- Online compiling service by Google Colab
- 512 GB of SSD or HDD
- 8 GB RAM
- Basic GPU

4.2 Software Requirement

- Windows OS we have used Windows 10
- Python 3.10 with IDE Jupyter Notebook or Google Colab
- Windows Browser like Microsoft or Edge Google Chrome
- Library
    - Panadas
    - SkLearn
    - Numpy
    - Statsmodel
    - pmdArima
    - TensorFlow or Keras
    - Facebook Prophet

Chapter 5
Future Scope

- Implement Pycaret Library - Pycaret Library is an open-source, low-code machine learning library in Python that automates machine learning workflows. It is an end-to-end machine learning and model management tool that exponentially speeds up the experiment cycle and makes you more productive.

- Create a WebApp using Streamlit - It is an open-source app framework in Python language. It helps us create web apps for data science and machine learning in a short time. It is compatible with major Python libraries such as scikit-learn, Keras, PyTorch, SymPy(latex), NumPy, pandas, Matplotlib etc.

- Deployment on AWS and Heroku

- Create UI which can be used by stock investor to analyse the trend of stock

- User can select Machine Learning or DL model and select date using UI and he will get the predicted close value of stock

- In this we have used Nifty 50 stock and we want to apply this to all different types stocks in stock Market

Chapter 6
Conclusion

We have created 6 models using Nifty 50 stock data having 50000 rows as shown in table below. We have calculated the evaluating parameters like R2, Mean Squared Error (MSE) And Mean Absolute Error (MAE) to compare different models

Model is a good model when following

- R2 values should be High
- MSE and MAE value should be small

Model Name	Algorithm	Data size	X column or features	r2	mse	mae
model 1	Random Forest Regressor	50000	close t_1 to close t_4 lag value 1 min to 4 min	0.999957	41.33685	4.098979
model 2	Gradient Boosting Regressor	50000		0.999919	77.85788	6.222007
model 3	Random Forest Regressor	50000	close t_1 (374 min)	0.986207	13000.9	80.89605
model 4	Gradient Boosting Regressor	50000	close t_1 (374 min)	0.989716	9693.396	73.19261
model 5	Fb Prophet	50000	Done by library	0.986225	12984.58	80.97747
Model 6	LSTM	50000	--	--	--	--

Table 6.1 Model Comparison using evaluating parameters

As seen above the model 1 and model 2 have high R2 value compared to other models and also MSE and MAE is small for these models

Here we can observe High R2 for all models because out stock data is per minute data and there not much difference in close value of stock every minute

When we compare model 1 and model 2 we can observe model 1 using Random Forest Regressor have low MSE and MAE as compare to model 2 using Gradient Boosting Regressor

Model 1 using Random Forest Regressor and X columns having 4 features of lag values from 1 min to 4 min is best model among all model as in table so it works good for analysing trend

Chapter 7
References

- Dataset on Kaggle website - https://www.kaggle.com/datasets/hk7797/stock-market-india
- Krish Naik GitHub - https://github.com/krishnaik06/Live-Time-Series
- Financial Domain concepts - https://corporatefinanceinstitute.com/resources/knowledge/trading-investing/exponentially-weighted-moving-average-ewma/
- Pandas Docs - https://pandas.pydata.org/
- Library Used - Sklearn, Statsmodel, Keras Docs - https://keras.io/
- Facebook Prophet Library - https://facebook.github.io/prophet/