A Project Report

on

Stock market prediction using Deep learning

Submitted in partial fulfillment of the
requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

CSE

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of :

Ravinder Ahuja

Submitted By :

PAWAN KUMAR
18SCSE1010656

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING / DEPARTMENT OF COMPUTERAPPLICATION
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
DECEMBER, 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the project , entitled " Stock market prediction using Deep learning" in partial fulfillment of the requirements for the award of the Bachelor of Technology submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of July, 2021 to December and 2021, under the supervision of Ravinder Ahuja , Department of Computer Science and Engineering/Computer Application and Information and Science, of School of Computing Science and Engineering , Galgotias University, Greater Noida .

The matter presented in the project has not been submitted by me for the award of any other degree of this or any other places.

Pawan Kumar

18SCSE1010656

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Ravinder Ahuja

CERTIFICATE

The Final Project Viva-Voce examination of Pawan Kumar 18scse1010656 has been held on _____ and his work is recommended for the award of Bachelor of Technology , Computer Science and Engineering .

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Dean

Date: December, 2021

Place: Greater Noida

Table of Contents

ABSTRACT

A stock (also known as equity) is a security that represents the ownership of a fraction of a corporation and are the backbone of any investment portfolio. Advances in trading technology have opened up markets so that nowadays almost everyone hasstocks. From in the last few decades, there has been a dramatic increase in the number of descendants of the average person stock market. In a volatile financial market, such as the stock market, it is important that have the most accurate prediction of future practice. Due to the financial of the record, it is imperative that there be a secure prediction of stock market. In this research paper we will focus on Long-Short-Term Memory(LSTM) Recurrent Neural Network belongs to the family of deep learning algorithms.

While predicting the actual price of a stock is an uphill climb, we can build a model that will predict whether the price will go up or down. It's important to note that there are always other factors that affect the prices of stocks, such as the political atmosphere and the market.

In Stock Market Prediction, the aim is to predict the future value of the financial stocks of a company. The recent trend in stock market prediction technologies is the use of machine learning which makes predictions based on the values of current stock market indices by training on their previous values.

Machine learning itself employs different models to make prediction easier and authentic. The paper focuses on the use of Regression and LSTM based Machine learning to predict stock values. Factors considered are open, close, low, high and volume.

Keywords: random forest, prediction, time series analysis

CHAPTER-1

1. Introduction

Stock market is one of the major fields that investors are dedicated to, thus stock market price trend prediction is always a hot topic for researchers from both financial and technical domains. In this research, our objective is to build a state-of-art prediction model for price trend prediction, which focuses on short-term price trend prediction.

Predicting how the stock market will perform is one of the most difficult things to do. There are so many factors involved in the prediction – physical factors vs. psychological, rational and irrational behavior, etc. All these aspects combine to make share prices volatile and very difficult to predict with a high degree of accuracy.

Using features like the latest announcements about an organization, their quarterly revenue results, etc., machine learning techniques have the potential to unearth patterns and insights we didn't see before, and these can be used to make unerringly accurate predictions.

Prediction and analysis of the stock market are some of the most complicated tasks to do. There are several reasons for this, such as the market volatility and so many other dependent and independent factors for deciding the value of a particular stock in the market. These factors make it very difficult for any stock market analyst to predict the rise and fall with high accuracy degrees.

However, with the advent of Machine Learning and its robust algorithms, the latest market analysis and Stock Market Prediction developments have started incorporating such techniques in understanding the stock market data.

In short, Machine Learning Algorithms are being used widely by many organisations in analysing and predicting stock values. This article shall go through a simple Implementation of analysing and predicting a Popular Worldwide Online Retail Store's stock values using several Machine Learning Algorithms in Python.

1.2 Formulation of Problem

Before we get into the program's implementation to predict the stock market values, let us visualise the data on which we will be working. Here, we will be analysing the stock value of Zomato Inc of India from the Bombay Stock Exchange located in Mumbai , India. The stock value data will be presented in the form of a Comma Separated File (.csv), which can be opened and viewed using Excel or a Spreadsheet.

Zomato has its stocks registered in BSE & NSE and has its values updated during every working day of the stock market. Note that the market doesn't allow trading to happen on Saturdays and Sundays , hence there is a gap between the two dates. For each date, the Opening Value of the stock, Highest and Lowest values of that stock on the same days are noted, along with the Closing Value at the end of the day.

The Adjusted Close Value shows the stock's value after dividends are posted . Additionally, the total volume of the stocks in the market are also given, With these data, it is up to the work of a Machine Learning/Data Scientist to study the data and implement several algorithms that can extract patterns from the Zomato Inc's historical data.

1.2.1 Technology Used

1. Programming Language : Python , DBMS
2. Tools : Heroku , Git , Visual Studio Code
3. Computer with minimum 4GB ram And Good graphic power

LITERATURE REVIEW

For any project planning activity, a good reading of existing projects, ideas and technology is needed. The following subsections provide much needed excerpts from important research papers and literary documents related to Stocks, Artificial Neural Network, Auto Regressive Moving Average and Natural Language Processing.

Factors of change for Stock Markets :

1. Stock Prices: Stock Prices or share prices change every day depending on the market, economy and company's financial performance and outlook.

2. Sentiment: The news headlines, important statements made by key people of a company, announcements all contribute to stock values in a positive, negative or neutral way.

3. Market and Environmental Factors: Market and Environmental factors like Economic boom, depression, political scenario, natural disasters etc. contribute to the performance of stock market in general. Before deciding on the project, lot of research papers of the domain and other articles related to stocks, share markets and machine learning. Following inferences were made from these papers.

4. Formula for calculation of Stock Prices: The methodology used in this study considered the short-term historical stock prices as well as the day of week as inputs.

The overall procedure is governed by the following
equation:

$$y(k) = f\,(y(k-1), y(k-2), y(k-3), ..., y(k-n), D(k))$$

where y(k) is the stock price at time k, n is the number of historical days, and D(k) is the day of week. Selection of Algorithm: -Three algorithms were tested with input datasets of Google Inc. for the Nov to Dec 15. Tested models were Random Forest, Logical Regression and Multi-Layer Perceptron.

1. Random forest[7]: - Random forest is a faster calculation method but is the least accurate when compared with the other two algorithms. Random forest explores all the possible nodes of a tree and selects the
maxima. Accuracy:50-60%. Good for short term predictions and gains .

2. Logical Regression[7]: -The logical regression method makes a graph of the dataset present. It converts the graph to an equation form and substitutes values. Accuracy: 60-64%. Good for midrange predictions.

3. Artificial Neural Network using Multi-Layer Perceptron Classifier[8]:-The Neural network (Multilayer Perceptron Model) works in layers and data is processed in each layer. The final layer produces the output. This is the best

algorithm for long term predictions ranging from 300 days and above. Accuracy: 75% and above.

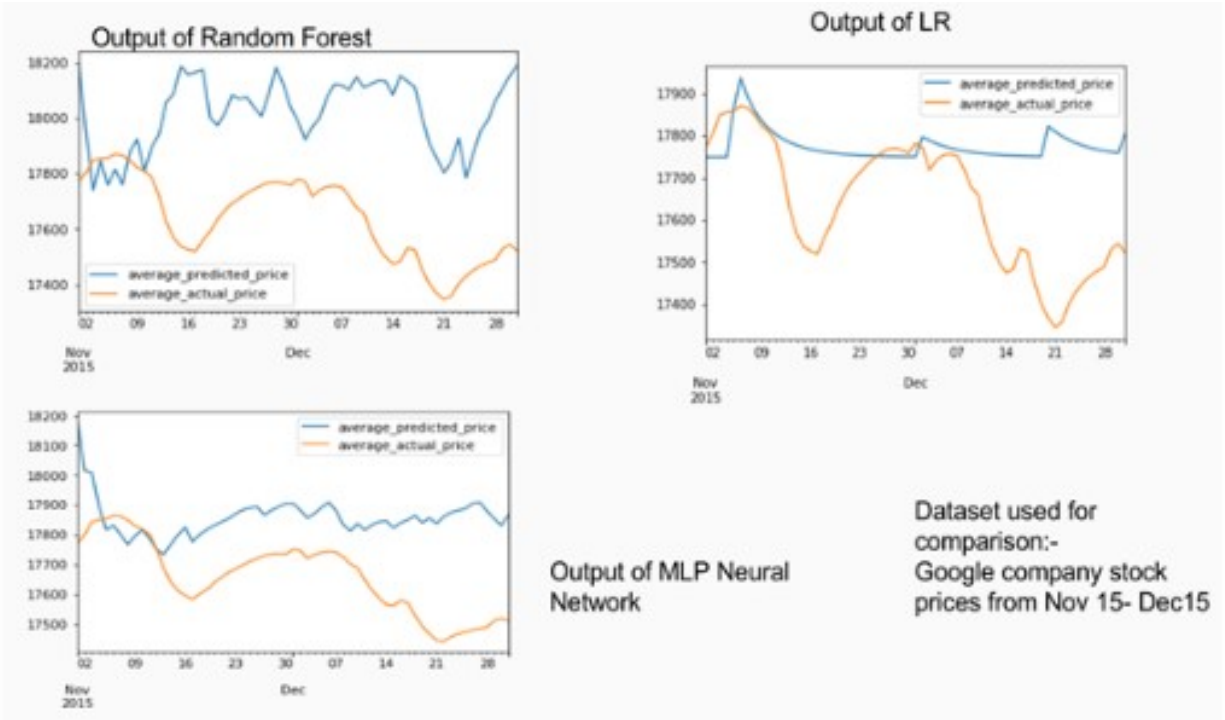


Fig-1 Different Prediction Algorithms and their accuracy

METHODOLOGY

The implementation of this paper begins with preprocessing the data collected from stock market pickled data set. This preprocessed data is classified using popular machine learning algorithm to calculate the polarity score. In order to prepare the data ready to apply Random forest algorithm, noise in the data is removed by smoothing. The working of random forest algorithm is presented below:

i. Randomly select “k” features from total “m” features, where k << m

ii. Among the “k” features, calculate the node “d” using the best split point.

iii. Split the node into daughter nodes using the best split.

iv. Repeat 1 to 3 steps until the “l” number of nodes has been reached.

v. Build forest by repeating steps 1 to 4 for “n” number times to create “n” number of trees.

Random forest algorithm starts by randomly selecting k features from m available features. Over the k selected features a point d has to be selected in order to split the features. This process would be executed iteratively to obtain the tree structure with a root node and leaf nodes as the target features to be processed further. This results in n number of trees in the generated

forest. The algorithm is now tested for its efficiency by measuring the accuracy of predicting the stock price and also by calculating the variance score generated by the algorithm, finally ending up the process by comparing random forest with logistic regression. The experimental results obtained prove that Random Forest algorithm is efficient in predicting the stock price through achieving better score of the regression metrics over logistic regression.

The results obtained are plotted in the form of a graph as presented below:

All the calculations are done based upon the four regression values variance score, mean
absolute error, mean squared error, mean squared log error .

PROJECT DESIGN

As studied in Literature review we can say that stock market prediction is very difficult and most existing systems only tend to use a single model to predict stock prices, the intention is to use Auto regressive Moving Average and Neural Network using Multi-Layer Perceptron together to obtain a more accurate prediction. Along with that for investor to obtain a better understanding of companies' impression on peoples mind the system also should have a sentiment analyser.

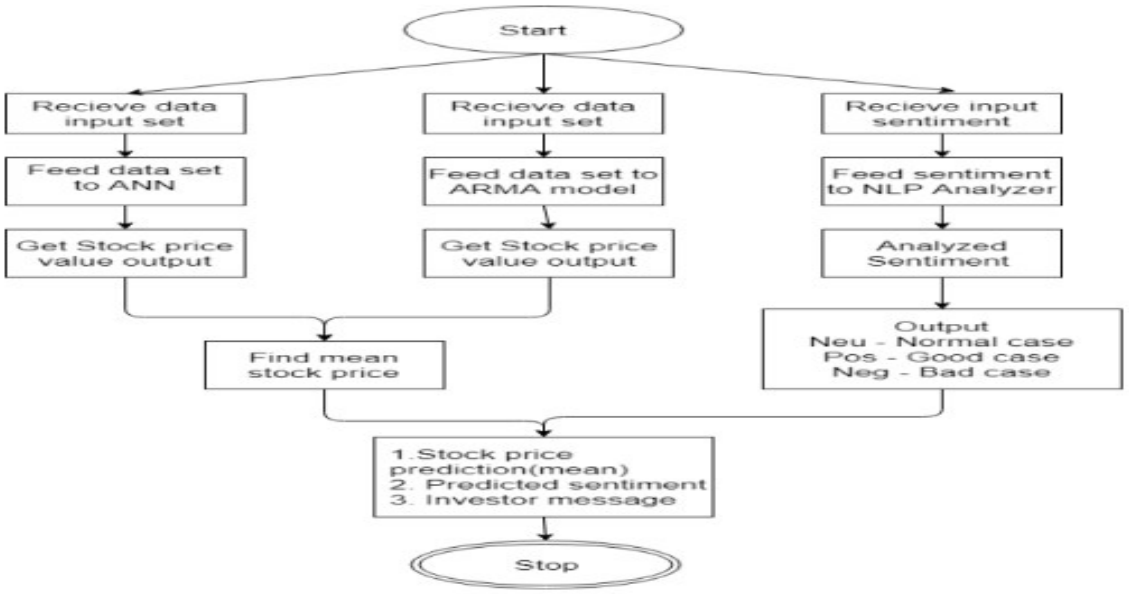


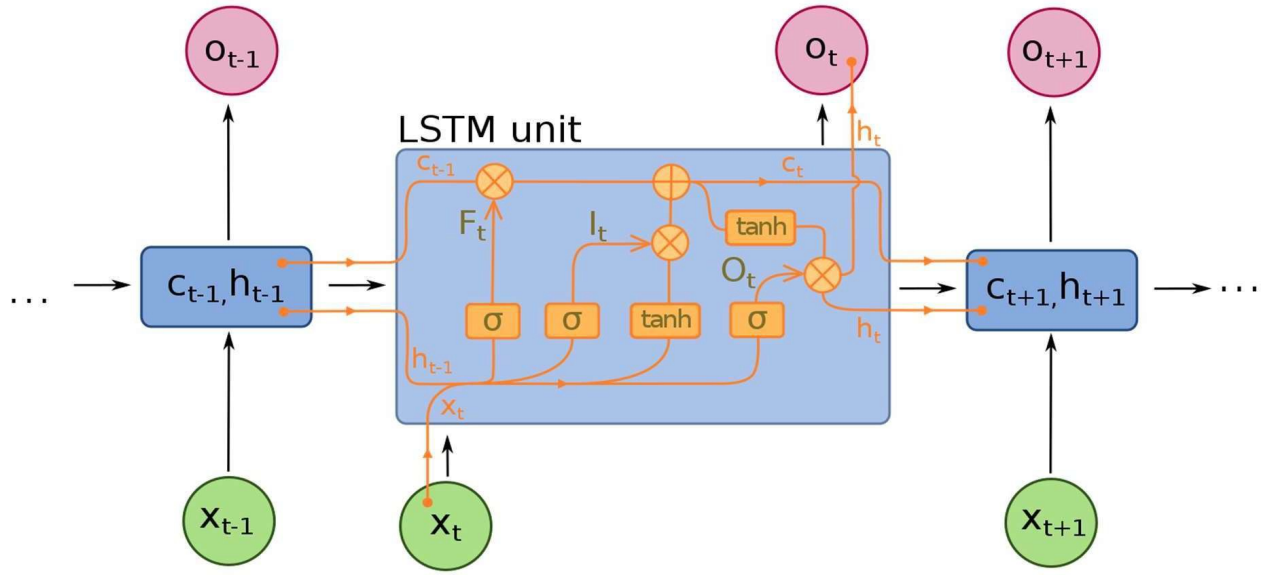
Fig-2: Data Flow diagram of system

Figure 4.1: Use Case Diagram for the system

Use case index

Use case ID	Use case name	Primary actor	scope	complexity	priority
1	Collect data	admin	in	high	1
2	Compute result and prepare	admin	in	high	1
3	System update	admin	in	high	1
4	View trade exchange	user	in	medium	2
5	Company stock	user	in	medium	2
6	View predicted outcome	user	in	high	1

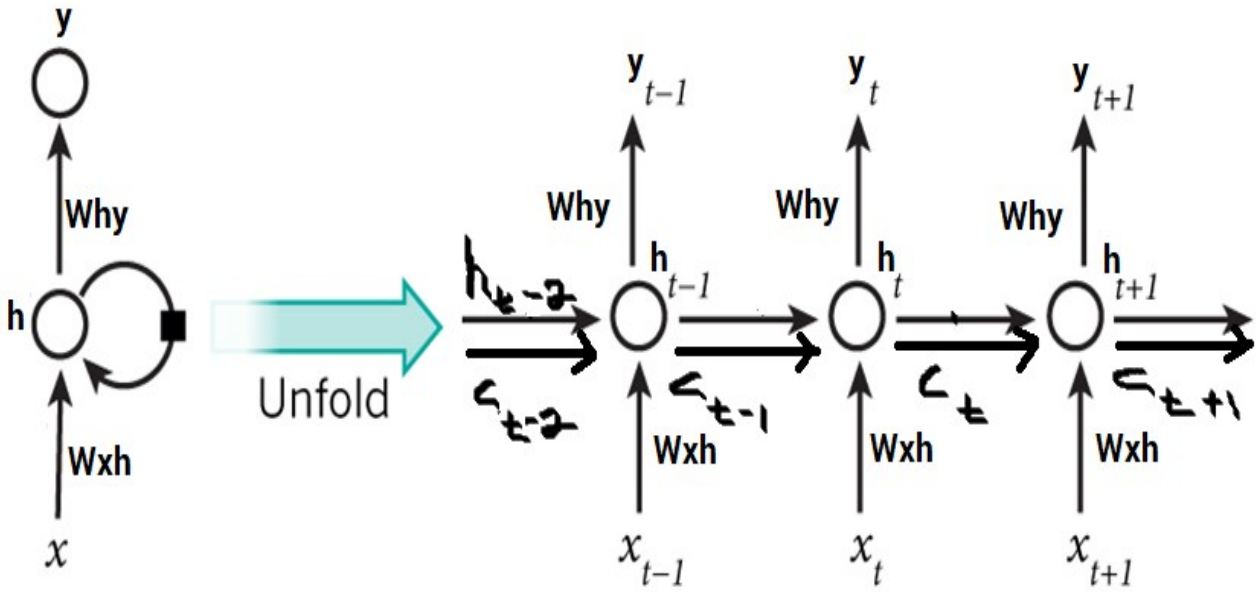
We propose to use LSTM (Long Short Term Memory) algorithm to provide efficientstock price prediction .

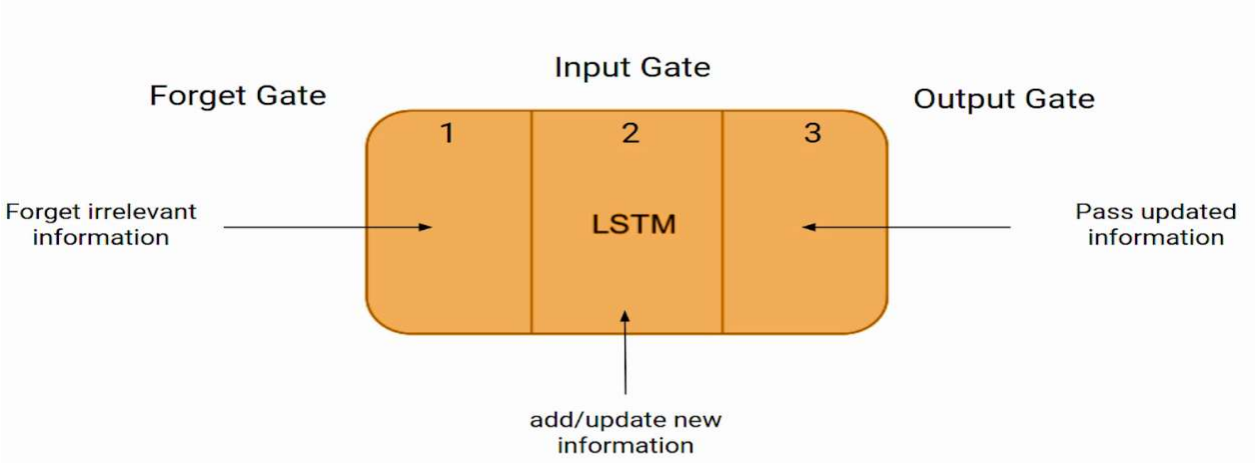


LSTM networks are an extension of recurrent neural networks (RNNs) mainly introduced to handle situations where RNNs fail. Talking about RNN, it is a network that works on the present input by taking into consideration the previous output (feedback) and storing in its memory for a short period of time (short-term memory). Out of its various applications, the most popular ones are in the fields of speech processing, non-Markovian control, and music composition. Nevertheless, there are drawbacks to RNNs. First, it fails to store information for a longer period of time. At times, a reference to certain information stored quite a long time ago is required to predict the current output. But RNNs are absolutely incapable of handling such "long-term dependencies". Second, there is no finer control over which part of the context needs to be carried forward and how much of the past needs to be 'forgotten'. Other issues with RNNs are exploding and vanishing gradients (explained later) which occur during the training process of a network through backtracking. Thus, Long Short-Term Memory (LSTM) was brought into the picture. It has been so designed that the vanishing gradient problem is almost completely removed, while the training model is left unaltered. Long time lags in certain problems are bridged using LSTMs where they also handle noise, distributed representations, and continuous values. With LSTMs, there is no need to keep a finite number of states from beforehand as required in the hidden Markov model (HMM). LSTMs provide us with a large range of parameters such as learning rates, and input and output biases. Hence, no need for fine adjustments. The complexity to update each weight is reduced to O(1) with LSTMs, similar to that of Back Propagation Through Time (BPTT), which is an advantage.

Architecture:

The basic difference between the architectures of RNNs and LSTMs is that the hidden layer of LSTM is a gated unit or gated cell. It consists of four layers that interact with one another in a way to produce the output of that cell along with the cell state. These two things are then passed onto the next hidden layer. Unlike RNNs which have got the only single neural net layer of tanh, LSTMs comprises of three logistic sigmoid gates and one tanh layer. Gates have been introduced in order to limit the information that is passed through the cell. They determine which part of the information will be needed by the next cell and which part is to be discarded. The output is usually in the range of 0-1 where '0' means 'reject all' and '1' means 'include all'.





Let's take an example to understand how LSTM works. Here we have two sentences separated by a full stop. The first sentence is "Bob is a nice person" and the second sentence is "Dan, on the Other hand, is evil". It is very clear, in the first sentence we are talking about Bob and as soon as we encounter the full stop(.) we started talking about Dan.

As we move from the first sentence to the second sentence, our network should realize that we are no more talking about Bob. Now our subject is Dan. Here, the Forget gate of the network allows it to forget about it. Let's understand the roles played by these gates in LSTM architecture.

WORKING OF PROJECT

We shall move on to the part where we put the LSTM into use in predicting the stock value using Machine Learning in Python.

Importing the Libraries

As we all know, the first step is to import libraries that are necessary to reprocesses the stock data of Microsoft Corporation and the other required libraries for building and visualising the outputs of the LSTM model. For this, we will use the Keras library under the TensorFlow framework. The required modules are imported from the Keras library individually.

#Importing the Libraries

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 mandates

from sklearn. Preprocessing import MinMaxScaler

from sklearn import linear_model

from Keras. Models import Sequential

from Keras. Layers import Dense

import Keras. Backend as K

from Keras. Callbacks import EarlyStopping

from Keras. Optimisers import Adam

from Keras. Models import load_model

from Keras. Layers import LSTM

from Keras. utils.vis_utils import plot_model
```

The dataset

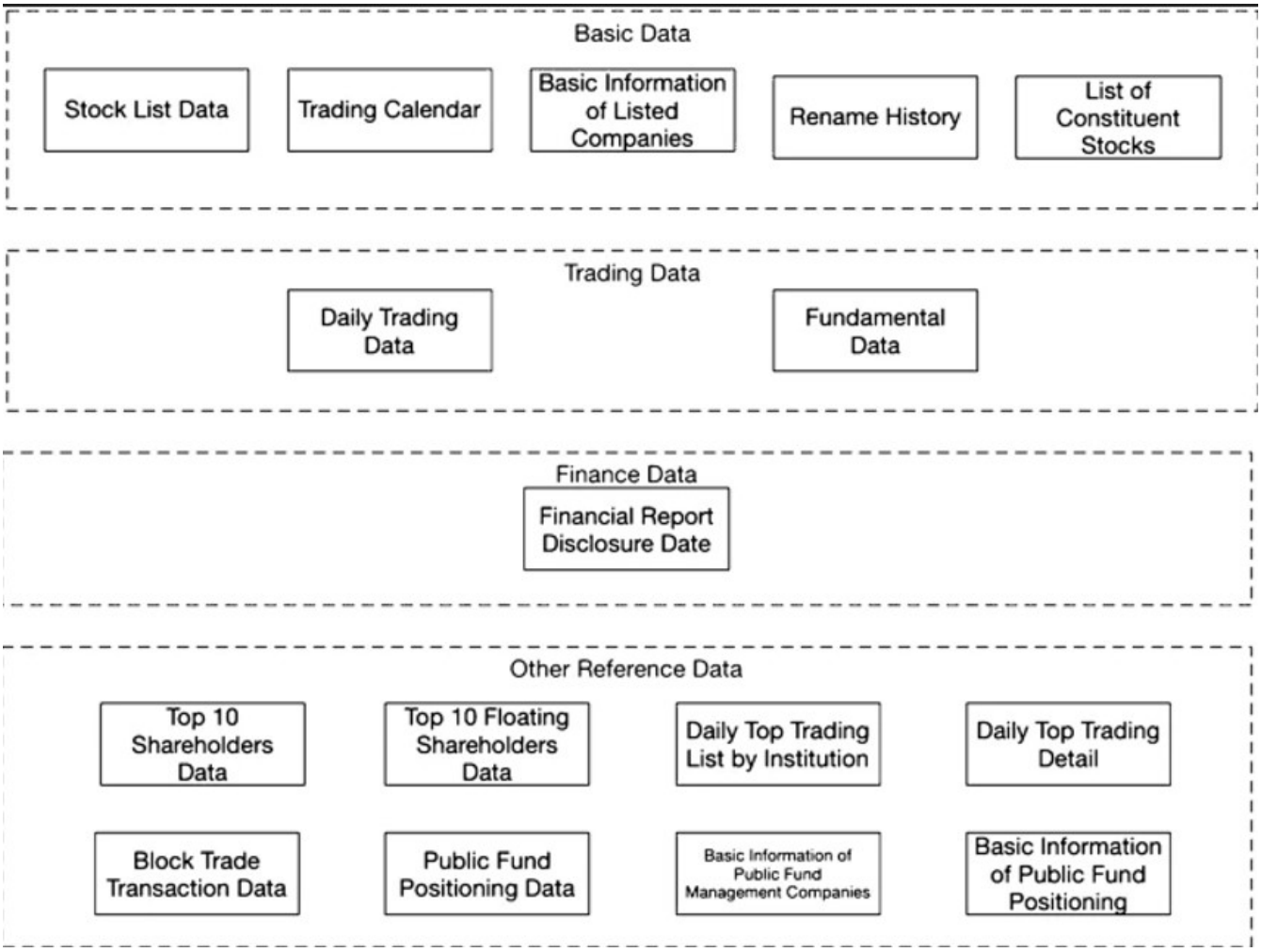
This section details the data that was extracted from the public data sources, and the final dataset that was prepared. Stock market-related data are diverse, so we first compared the related works from the survey of financial research works in stock market data analysis to specify the data collection directions. After collecting the data, we defined a data structure of the dataset. Given below, we describe the dataset in detail, including the data structure, and data tables in each category of data with the segment definitions.

Description of our dataset

In this section, we will describe the dataset in detail. This dataset consists of 3558 stocks from the Chinese stock market. Besides the daily price data, daily fundamental data of each stock ID, we also collected the suspending and resuming history, top 10 shareholders, etc. We list two reasons that we choose 2 years as the time span of this dataset: (1) most of the investors perform stock market price trend analysis using the data within the latest 2 years, (2) using more recent data would benefit the analysis result. We collected data through the open-sourced API, namely Tushare [43], mean-while we also leveraged a web-scraping technique to collect data from Sina Finance web pages, SWS Research website.

Data structure

Figure 1 illustrates all the data tables in the dataset. We collected four categories of data in this dataset: (1) basic data, (2) trading data, (3) finance data, and (4) other reference data. All the data tables can be linked to each other by a common field called “Stock ID” It is a unique stock identifier registered in the Chinese Stock market. Table 1 shows an overview of the dataset.



Loading the Dataset

The next step is to load in our training dataset and select the Open and Highcolumnsthat we'll use in our modeling.

```
dataset_train = pd.read_csv('NSE-TATAGLOBAL.csv')
training_set = dataset_train.iloc[:, 1:2].values
```

Getting Visualising the Data

Using the Pandas Data reader library, we shall upload the local system's stock data as a Comma Separated Value (.csv) file and store it to a pandas DataFrame. Finally, we shall also view the data.

```
#Get the Dataset

df = pd.read_csv("MicrosoftStockData.csv",na_values=['null'],index_col='Date', parse_dates=True,infer_datetime_format=True)

df.head()
```

Feature Scaling

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

Creating Data with Timesteps

LSTMs expect our data to be in a specific format, usually a 3D array. We start by creating data in 60 timesteps and converting it into an array using NumPy. Next, we convert the data into a 3D dimension array with X_train samples, 60 timestamps, andone feature at each step.

```
X_train = []
y_train = []
for i in range(60, 2035):
    X_train.append(training_set_scaled[i-60:i, 0])
    y_train.append(training_set_scaled[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))
```

Building the LSTM

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

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 Future Stock using the Test Set

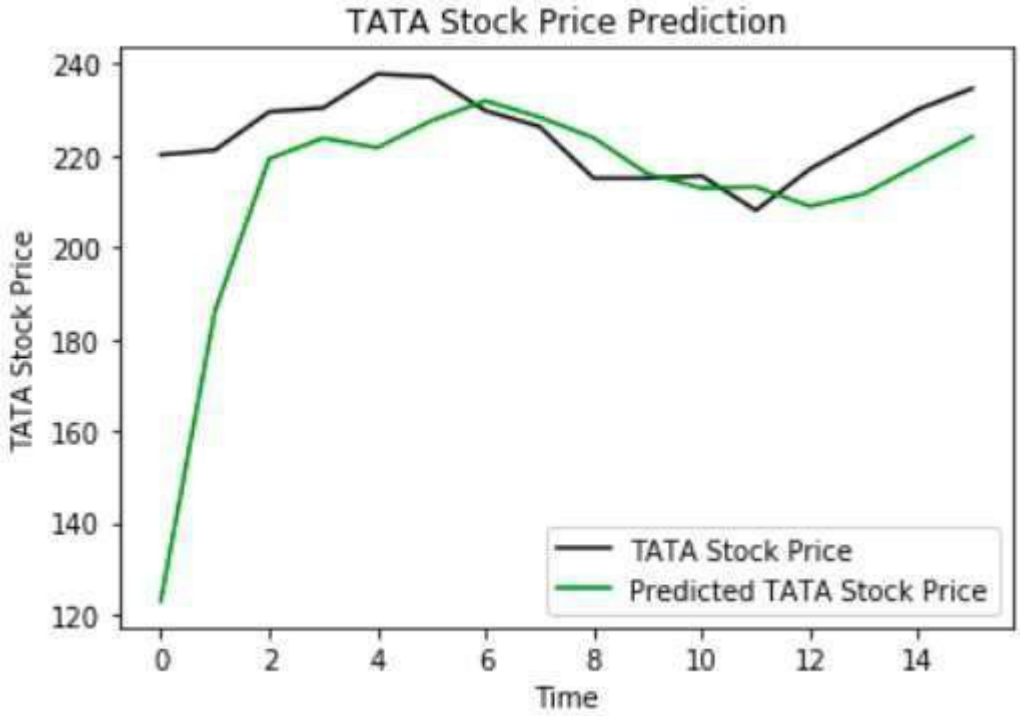
```
dataset_test = pd.read_csv('tatatest.csv')
real_stock_price = dataset_test.iloc[:,
1:2].values


dataset_total = pd.concat((dataset_train['Open'], dataset_test['Open']),
axis = 0)inputs = dataset_total[len(dataset_total) - len(dataset_test) -
60:].values
inputs = inputs.reshape(-
1,1) inputs =
sc.transform(inputs)
X_test = []
for i in range(60, 76):
  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))
predicted_stock_price = regressor.predict(X_test)
predicted_stock_price=sc.inverse_transform(predicted_stoc
k_price)
```

Plotting the Results

plt.plot(real_stock_price, color = 'black', label = 'TATA Stock Price')
plt.plot(predicted_stock_price, color = 'green', label = 'Predicted TATA Stock Price')

plt.title('TATA Stock Price Prediction')plt.xlabel('Time')
plt.ylabel('TATA Stock Price')plt.legend()
plt.show()



We will first sort the dataset in ascending order and then create a separate dataset so that any new feature created does not affect the original data.

```
#setting index as date values
df['Date'] = pd.to_datetime(df.Date,format='%Y-%m-%d')
df.index = df['Date']

#sorting
data = df.sort_index(ascending=True, axis=0)

#creating a separate dataset
new_data = pd.DataFrame(index=range(0,len(df)),columns=['Date', 'Close'])

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

#create features
from fastai.structured import  add_datepart
add_datepart(new_data, 'Date')
new_data.drop('Elapsed', axis=1, inplace=True)  #elapsed will be the time
stamp
```

```
new_data['mon_fri'] = 0
for i in range(0,len(new_data)):
    if (new_data['Dayofweek'][i] == 0 or new_data['Dayofweek'][i] == 4):
        new_data['mon_fri'][i] = 1
    else:
        new_data['mon_fri'][i] = 0
```

If the day of week is equal to 0 or 4, the column value will be 1, otherwise 0. Similarly, you can create multiple features. If you have some ideas for features that can be helpful in predicting stock price, please share in the comment section.

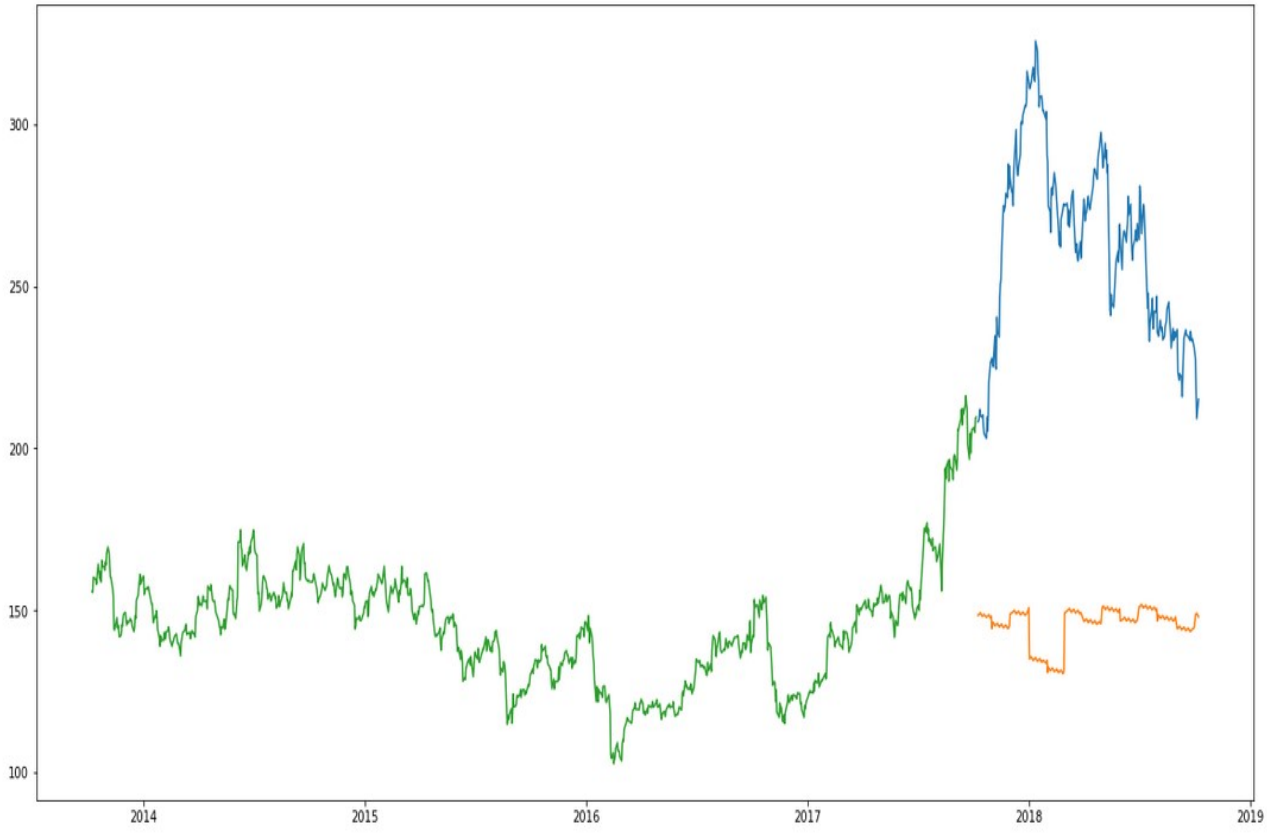
We will now split the data into train and validation sets to check the performance of the model.

```
#split into train and validation
train = new_data[:987]
valid = new_data[987:]

x_train = train.drop('Close', axis=1)
y_train = train['Close']
x_valid = valid.drop('Close', axis=1)
y_valid = valid['Close']

#implement linear regression
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train,y_train)
```

```
#make predictions and find the rmse
preds = model.predict(x_valid)
rms=np.sqrt(np.mean(np.power((np.array(y_valid)-np.array(preds)),2)))
rms
121.16291596523156
```



Algorithm elaboration

This section provides comprehensive details on the algorithms we built while utilizing and customizing different existing techniques. Details about the terminologies, parameters, as well as optimizers. From the legend on the right side of Fig. 3, we note the algorithm steps as octagons, all of them can be found in this "Algorithm elaboration" section.

Before dive deep into the algorithm steps, here is the brief introduction of data pre-processing: since we will go through the supervised learning algorithms, we also need to program the ground truth. The ground truth of this research is programmed by comparing the closing price of the current trading date with the closing price of the previous trading date the users want to compare with. Label the price increase as 1, else the ground truth will be labeled as 0. Because this research work is not only focused on predicting the price trend of a specific period of time but short-term in general, the ground truth processing is according to a range of trading days. While the algorithms will not change with the prediction term length, we can regard the term length as a parameter.

The algorithmic detail is elaborated, respectively, the first algorithm is the hybrid feature engineering part for preparing high-quality training and testing data. It corresponds to the Feature extension, RFE, and PCA

blocks in Fig. 3. The second algorithm is the LSTM procedure block, including time-series data pre-processing, NN constructing, training, and testing.

Algorithm 1: Short-term stock market price trend prediction—applying feature engineering using FE + RFE + PCA

The function FE is corresponding to the feature extension block. For the feature extension procedure, we apply three different processing methods to translate the findings from the financial domain to a technical module in our system design. While not all the indices are applicable for expanding, we only choose the proper method(s) for certain features to perform the feature extension (FE), according to Table 2.

Normalize method preserves the relative frequencies of the terms, and transform the technical indices into the range of [0, 1]. Polarize is a well-known method often used by real-world investors, sometimes they prefer to consider if the technical index value is above or below zero, we program some of the features using polarize method and prepare for RFE. Max-min (or min-max) [35] scaling is a transformation method often used as an alternative to zero mean and unit variance scaling. Another well-known method used is fluctuation percentage, and we transform the technical indices fluctuation percentage into the range of $[-1, 1]$.

The function RFE () in the first algorithm refers to recursive feature elimination. Before we perform the training data scale reduction, we will have to make sure that the features we selected are effective. Ineffective features will not only drag down the classification precision but also add more computational complexity. For the feature selection part, we choose recursive feature elimination (RFE). As [45] explained, the process of recursive feature elimination can be split into the ranking algorithm, resampling, and external validation.

For the ranking algorithm, it fits the model to the features and ranks by the importance to the model. We set the parameter to retain *i* numbers of features, and at each iteration of feature selection retains *Si* top-ranked features, then refit the model and assess the performance again to begin another iteration. The ranking algorithm will eventually determine the top *Si* features.

The RFE algorithm is known to have suffered from the over-fitting problem. To eliminate the over-fitting issue, we will run the RFE algorithm multiple times on randomly selected stocks as the training set and ensure all the features we select are high-weighted. This procedure is called data resampling. Resampling can be built as an optimization step as an outer layer of the RFE algorithm.

The last part of our hybrid feature engineering algorithm is for optimization purposes. For the training data matrix scale reduction, we

apply Randomized principal component analysis (PCA) [31], before we decide the features of the classification model.

Financial ratios of a listed company are used to present the growth ability, earning ability, solvency ability, etc. Each financial ratio consists of a set of technical indices, each time we add a technical index (or feature) will add another column of data into the data matrix and will result in low training efficiency and redundancy. If non-relevant or less relevant features are included in training data, it will also decrease the precision of classification.

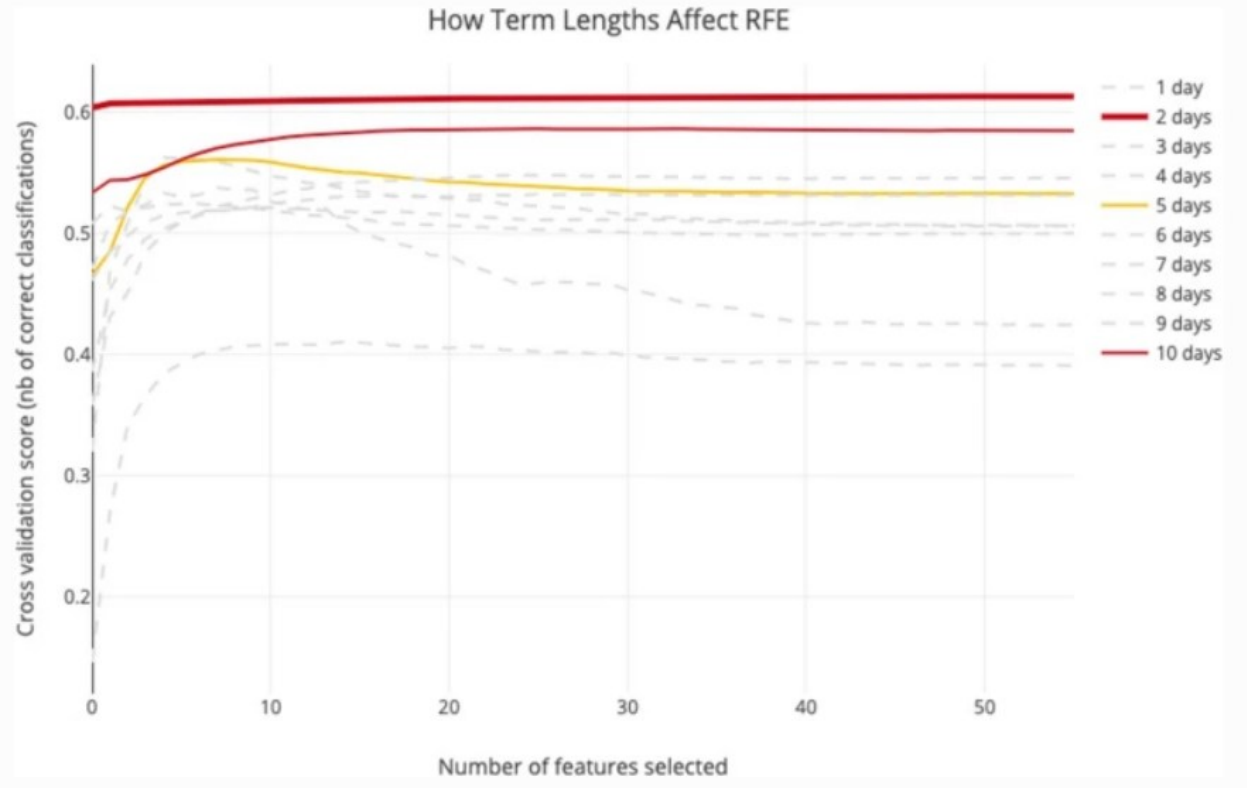
RESULT

Some procedures impact the efficiency but do not affect the accuracy or precision and vice versa, while other procedures may affect both efficiency and prediction result. To fully evaluate our algorithm design, we structure the evaluation part by main procedures and evaluate how each procedure affects the algorithm performance. First, we evaluated our solution on a machine with 2.2 GHz i7 processor, with 16 GB of RAM. Furthermore, we also evaluated our solution on Amazon EC2 instance, 3.1 GHz Processor with 16 vCPUs, and 64 GB RAM.

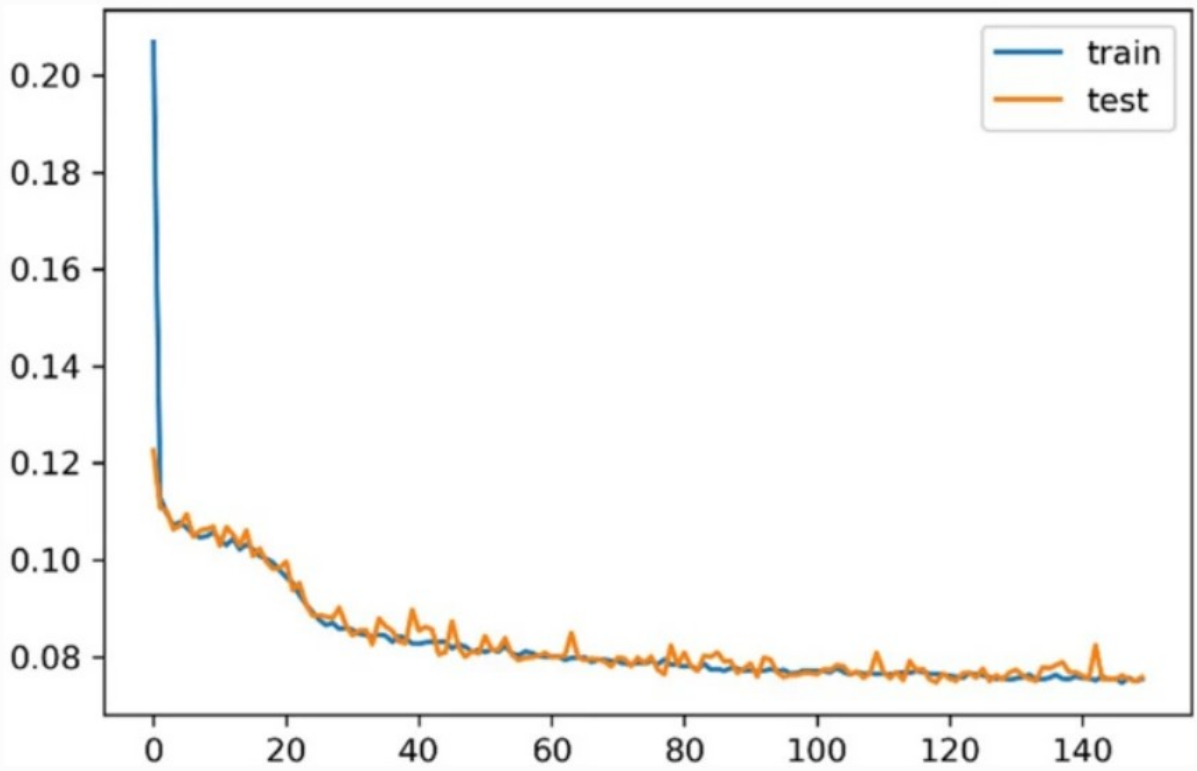
In the implementation part, we expanded 20 features into 54 features, while we retain 30 features that are the most effective. In this section, we discuss the evaluation of feature selection. The dataset was divided into two different subsets, i.e., training and testing datasets. Test procedure included two parts, one testing dataset is for feature selection, and another one is for model testing. We note the feature selection dataset and model testing dataset as DS_test_f and DS_test_m, respectively.

We randomly selected two-thirds of the stock data by stock ID for RFE training and note the dataset as DS_train_f; all the data consist of full technical indices and expanded features throughout 2018. The estimator of the RFE algorithm is SVR with linear kernels. We rank the 54

features by voting and get 30 effective features then process them using the PCA algorithm to perform dimension reduction and reduce the features into 20 principal components. The rest of the stock data forms the testing dataset DS_test_f to validate the effectiveness of principal components we extracted from selected features. We reformed all the data from 2018 as the training dataset of the data model and noted as DS_train_m. The model testing dataset DS_test_m consists of the first 3 months of data in 2019, which has no overlap with the dataset we utilized in the previous steps. This approach is to prevent the hidden problem caused by overfitting.



In this section, we discuss and compare the results of our proposed model, other approaches, and the real world data .



Learning curve of proposed solution

Conclusion

This work consists of three parts: data extraction and pre-processing of the Indian stock market(NSE & BSE) dataset, carrying out feature engineering, and stock price trend prediction model based on the long short-term memory (LSTM). We collected, cleaned-up, and structured 2 years of Chinese stock market data. We reviewed different techniques often used by real-world investors, developed a new algorithm component, and named it as feature extension, which is proved to be effective. We applied the feature expansion (FE) approaches with recursive feature elimination (RFE), followed by principal component analysis (PCA), to build a feature engineering procedure that is both effective and efficient. The system is customized by assembling the feature engineering procedure with an LSTM prediction model, achieved high prediction accuracy that outperforms the leading models in most related works. We also carried out a comprehensive evaluation of this work. By comparing the most frequently used machine learning models with our proposed LSTM model under the feature engineering part of our proposed system, we conclude many heuristic findings that could be future research questions in both technical and financial research domains.

Our proposed solution is a unique customization as compared to the previous works because rather than just proposing yet another state-of-

the-art LSTM model, we proposed a fine-tuned and customized deep learning prediction system along with utilization of comprehensive feature engineering and combined it with LSTM to perform prediction. By researching into the observations from previous works, we fill in the gaps between investors and researchers by proposing a feature extension algorithm before recursive feature elimination and get a noticeable improvement in the model performance.

Though we have achieved a decent outcome from our proposed solution, this research has more potential towards research in future. During the evaluation procedure, we also found that the RFE algorithm is not sensitive to the term lengths other than 2-day, weekly, biweekly. Getting more in-depth research into what technical indices would influence the irregular term lengths would be a possible future research direction. Moreover, by combining latest sentiment analysis techniques with feature engineering and deep learning model, there is also a high potential to develop a more comprehensive prediction system which is trained by diverse types of information such as tweets, news, and other text-based data.

References

1. Atsalakis GS, Valavanis KP. Forecasting stock market short-term trends using a neuro-fuzzy based methodology. Expert Syst Appl. 2009;36(7):10696–707.

2. Ayo CK. Stock price prediction using the ARIMA model. In: 2014 UKSim-AMSS 16th international conference on computer modelling and simulation. 2014. https://doi.org/10.1109/UKSim.2014.67.

3. Brownlee J. Deep learning for time series forecasting: predict the future with MLPs, CNNs and LSTMs in Python. Machine Learning Mastery. 2018. https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/

4. Eapen J, Bein D, Verma A. Novel deep learning model with CNN and bi-directional LSTM for improved stock market index prediction. In: 2019 IEEE 9th annual computing and communication workshop and conference (CCWC). 2019. pp. 264–70. https://doi.org/10.1109/CCWC.2019.8666592.

5. Fischer T, Krauss C. Deep learning with long short-term memory networks for financial market predictions. Eur J Oper Res. 2018;270(2):654–69. https://doi.org/10.1016/j.ejor.2017.11.054.
6. Guyon I, Weston J, Barnhill S, Vapnik V. Gene selection for cancer classification using support vector machines. Mach Learn 2002;46:389–422.

7. Hafezi R, Shahrabi J, Hadavandi E. A bat-neural network multi-agent system (BNNMAS) for stock price prediction: case study of DAX stock price. Appl Soft Comput J. 2015;29:196–210. https://doi.org/10.1016/j.asoc.2014.12.028.

8. Halko N, Martinsson PG, Tropp JA. Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. SIAM Rev. 2001;53(2):217–88.

9. Hassan MR, Nath B. Stock market forecasting using Hidden Markov Model: a new approach. In: Proceedings—5th international conference on intelligent systems design and

applications 2005, ISDA'05. 2005. pp. 192–6. https://doi.org/10.1109/ISDA.2005.85.

10. Hochreiter S, Schmidhuber J. Long short-term memory. J Neural Comput. 1997;9(8):1735–80. https://doi.org/10.1162/neco.1997.9.8.1735.

11. Hsu CM. A hybrid procedure with feature selection for resolving stock/futures price forecasting problems. Neural Comput Appl. 2013;22(3–4):651–71. https://doi.org/10.1007/s00521-011-0721-4.

12. Huang CF, Chang BR, Cheng DW, Chang CH. Feature selection and parameter optimization of a fuzzy-based stock selection model using genetic algorithms. Int J Fuzzy Syst. 2012;14(1):65–75. https://doi.org/10.1016/J.POLYMER.2016.08.021.

13. Huang CL, Tsai CY. A hybrid SOFM-SVR with a filter-based feature selection for stock market forecasting. Expert Syst Appl. 2009;36(2 PART 1):1529–39. https://doi.org/10.1016/j.eswa.2007.11.062.

14.Idrees SM, Alam MA, Agarwal P. A prediction approach for stock market volatility based on time series data. IEEE Access. 2019;7:17287–98. https://doi.org/10.1109/ACCESS.2019.2895252.

15. Ince H, Trafalis TB. Short term forecasting with support vector machines and application to stock price prediction. Int J Gen Syst. 2008;37:677–87. https://doi.org/10.1080/03081070601068595.