AISSMS
INSTITUTE OF INFORMATION TECHNOLOGY
ADDING VALUE TO ENGINEERING

Department of Computer Engineering

A **MACHINE LEARNING**

PROJECT REPORT ON

PREDICTION OF STOCK PRICES FOR INDIAN STOCK MARKET

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING AISSMS IOIT

BE Computer Engineering

SUBMITTED BY

STUDENT NAME	ERP No:
Onasvee Banarse	**09**
Kaustubh Kabra	**37**
Akash Mete	**50**
Harsh Shah	**65**

2022 -2023

Department of Computer Engineering

CERTIFICATE

This is to certify that the project report

"PREDICTION OF STOCK PRICES FOR INDIAN STOCK MARKET"

Submitted by

STUDENT NAME	ERP No:
Onasvee Banarse	09
Kaustubh Kabra	37
Akash Mete	50
Harsh Shah	65

is a bonafide students at this institute and the work has been carried out by them under the supervision of **Prof. Prashant Sadaphule** and it is approved for the partial fulfillment of the Department of Computer Engineering AISSMS IOIT.

(**Prof. Prashant Sadaphule**) (**Dr. S.N.Zaware**)

Mini-Project Guide Head of Computer Department

Place: Pune Date:

Abstract

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 project focuses on the use of Regression and LSTM based Machine learning to predict stock values. Factors considered are open, close, low, high and volume.

In this project we will consider a stock listed on the Indian stock market to predict its future trends. We will analyze the current trends, work with the data, refine it and then predict the future trends of the stock.

Contents

1. Introduction

A correct prediction of stocks can lead to huge profits for the seller and the broker. Frequently, it is brought out that prediction is chaotic rather than random, which means it can be predicted by carefully analyzing the history of the respective stock market. Machine learning is an efficient way to represent such processes. It predicts a market value close to the tangible value, thereby increasing the accuracy. Introduction of machine learning to the area of stock prediction has appealed to many researchers because of its efficient and accurate measurements.

The vital part of machine learning is the dataset used. The dataset should be as concrete as possible because a little change in the data can perpetuate massive changes in the outcome. In this project, supervised machine learning is employed on a dataset obtained from Yahoo Finance. This dataset comprises the following five variables: open, close, low, high and volume. Open, close, low and high are different bid prices for the stock at separate times with nearly direct names. The volume is the number of shares that passed from one owner to another during the time period. The model is then tested on the test data.

Regression and LSTM models are engaged for this conjecture separately. Regression involves minimizing error and LSTM contributes to remembering the data and results for the long run. Finally, the graphs for the fluctuation of prices with the dates (in case of Regression based model) and between actual and predicted price (for the LSTM based model) are plotted.

2. Problem Statement

Use the following dataset to analyze ups and downs in the market and predict future stock price returns based on Indian Market data from 2000 to 2020.

In this Project we will focus on following questions:

- Can machine learning provide an efficient method for prediction of stock prices?
- If yes, then what are the key aspects and roles it will play in this field?
- The different methods that machine learning provides for the prediction of stock prices.
- Preprocessing of dataset so that we get the most accurate result.
- How the current trends of a stock can be used to predict the future results.

3. Software and Hardware Requirement Specification

Software Used:

- **Jupyter Notebook**
  - Jupyter Notebook is an open-source, web-based interactive environment, which allows you to create and share documents that contain live code, mathematical equations, graphics, maps, plots, visualizations, and narrative text. It integrates with many programming languages like Python, PHP, R, C#, etc.

Hardware Used:

The detailed hardware used for the project are:

Item	Description
System	Asus TUF A15
Processor	AMD Ryzen 5 4600H
RAM	8 GB
System Type	64-bit operating system, x64-based processor
SSD	512 GB Solid State Drive
HDD	1 TB Hard Disk Drive
Graphics	NVIDIA 4 GB Graphic Card
Operating System	Windows 11 Operating System

5. Theory

1. Introduction

Stock market prediction seems a complex problem because there are many factors that have yet to be addressed and it doesn't seem statistical at first. But by proper use of machine learning techniques, one can relate previous data to the current data and train the machine to learn from it and make appropriate assumptions. Machine learning as such has many models but this paper focuses on two most important of them and makes predictions using them.

2. Methods

- **Regression based model:**

Regression is used for predicting continuous values through some given independent values. The project is based upon the use of a linear regression algorithm for predicting correct values by minimizing the error function as given in Figure1. This operation is called gradient descent. Regression uses a given linear function for predicting continuous values:

$$V = a + bK + error$$

Where, V is a continuous value; K represents known independent values; and, a, b are coefficients.

Work was carried out on csv format of data through panda library and calculated the parameter which is to be predicted, the price of the stocks with respect to time. The data is divided into different train sets for cross validation to avoid over fitting. The test set is generally kept 20% of the whole dataset. Linear regression

as given by the above equation is performed on the data and then predictions are made, which are plotted to show the results of the stock market prices vs time.

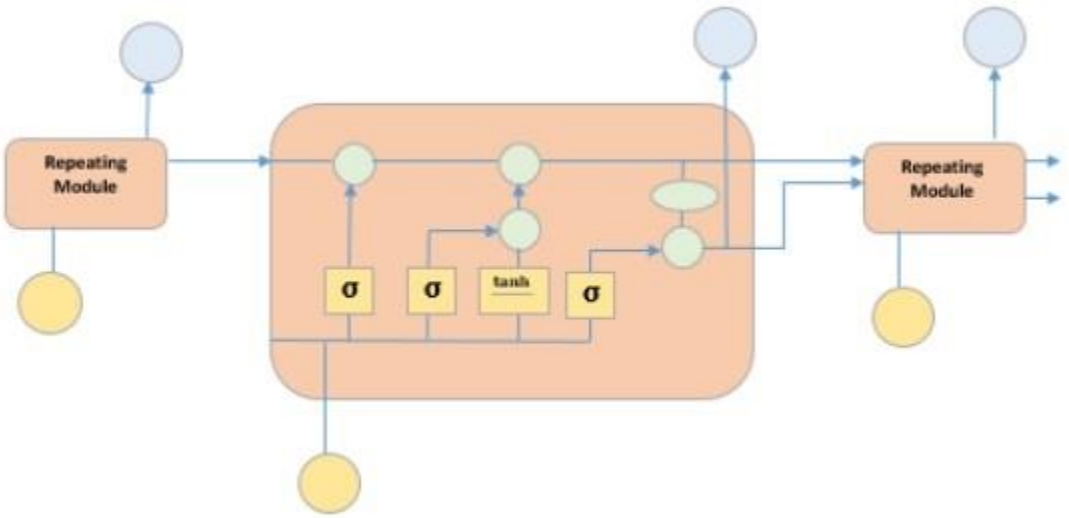
- **Long Short Term Memory (LSTM) Network Based Model**

LSTM is the advanced version of Recurrent-Neural-Networks (RNN) where the information belonging to previous state persists. These are different from RNNs as they involve long term dependencies and RNNs works on finding the relationship between the recent and the current information. This indicates that the interval of information is relatively smaller than that to LSTM. The main purpose behind using this model in stock market prediction is that the predictions depends on large amounts of data and are generally dependent on the long term history of the market. So LSTM regulates error by giving an aid to the RNNs through retaining information for older stages making the prediction more accurate.

Since stock market involves processing of huge data, the gradients with respect to the weight matrix may become very small and may degrade the learning rate of the system. This corresponds to the problem of Vanishing Gradient. LSTM prevents this from happening. The LSTM consists of a remembering cell, input gate, output gate and a forget gate. The cell remembers the value for long term propagation and the gates regulate them.

3. Working of LSTM model

It is special kind of recurrent neural network that is capable of learning long term dependencies in data. This is achieved because the recurring module of the model has a combination of four layers interacting with each other.



The picture above depicts four neural network layers in yellow boxes, point wise operators in green circles, input in yellow circles and cell state in blue circles. An LSTM module has a cell state and three gates which provides them with the power to selectively learn, unlearn or retain information from each of the units. The cell state in LSTM helps the information to flow through the units without being altered by allowing only a few linear interactions. Each unit has an input, output and a forget gate which can add or remove the information to the cell state. The forget gate decides which information from the previous cell state should be forgotten for which it uses a sigmoid function. The input gate controls the information flow to the current cell state using a point-wise multiplication operation of 'sigmoid' and 'tanh' respectively. Finally, the output gate decides which information should be passed on to the next hidden state.

6. Code and Output

Firstly, we import the necessary packages using the import command.

Stock Market Prediction using CNN-LSTM model

This project is about analysis of Stock Market and providing predictions to the stockholders. For this, we used CNN-LSTM approach to create a blank model, then use it to train on stock market data. Further implementation is discussed below...

In [40]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
```

Data Preprocessing and Analysis

In [41]:
```
import math
import seaborn as sns
import datetime as dt
from datetime import datetime
sns.set_style("whitegrid")
from pandas.plotting import autocorrelation_plot
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use("ggplot")
```

Then we load the dataset and display it. We check the datatype of each column.

In [42]:
```
data = pd.read_csv('C:/Users/hp/Untitled Folder/archive/COALINDIA.csv')
data.head()
```

Out[42]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Turnover	Trades	Deliverable Volume	%Deliverble
0	2010-11-04	COALINDIA	EQ	245.00	291.00	344.9	291.00	342.0	342.55	327.29	479716245	1.570040e+16	NaN	187584905	0.3910
1	2010-11-05	COALINDIA	EQ	342.55	343.00	356.5	343.00	348.3	349.85	349.78	31927173	1.116747e+15	NaN	10894509	0.3412
2	2010-11-08	COALINDIA	EQ	349.85	351.80	355.9	329.50	331.4	330.75	335.19	46932779	1.573118e+15	NaN	16651623	0.3548
3	2010-11-09	COALINDIA	EQ	330.75	330.15	333.4	325.00	325.4	326.05	327.75	23741956	7.781383e+14	NaN	12977359	0.5466
4	2010-11-10	COALINDIA	EQ	326.05	325.40	327.8	320.05	321.3	322.80	323.78	21057129	6.817982e+14	NaN	6280335	0.2983

In [43]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2598 entries, 0 to 2597
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Date                2598 non-null   object
 1   Symbol              2598 non-null   object
 2   Series              2598 non-null   object
 3   Prev Close          2598 non-null   float64
 4   Open                2598 non-null   float64
 5   High                2598 non-null   float64
 6   Low                 2598 non-null   float64
 7   Last                2598 non-null   float64
 8   Close               2598 non-null   float64
 9   VWAP                2598 non-null   float64
 10  Volume              2598 non-null   int64
 11  Turnover            2598 non-null   float64
 12  Trades              2456 non-null   float64
 13  Deliverable Volume  2598 non-null   int64
```

Using the describe function, we can analyze the statistical data of each and every column present.

```
In [44]: data.describe()
```

Out[44]:

	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Turnover	Trades	Deliverable Volume
count	2598.000000	2598.000000	2598.000000	2598.000000	2598.000000	2598.000000	2598.000000	2.598000e+03	2.598000e+03	2456.000000	2.598000e+03
mean	287.996555	288.234007	291.891744	284.240878	287.950943	287.953464	288.095708	5.617118e+06	1.405398e+14	53493.838355	2.783658e+06
std	74.086926	74.132769	75.064219	73.238459	74.098662	74.144478	74.181016	1.105497e+07	3.292618e+14	30508.369833	4.497473e+06
min	110.550000	110.850000	112.450000	109.550000	110.600000	110.550000	110.500000	2.143700e+04	7.734786e+11	612.000000	1.078600e+04
25%	254.087500	254.737500	257.012500	251.187500	254.325000	254.087500	254.312500	2.236566e+06	6.921541e+13	33748.750000	1.240880e+06
50%	299.950000	300.100000	303.925000	295.950000	300.000000	299.950000	300.290000	3.471441e+06	1.032340e+14	46755.000000	1.975260e+06
75%	341.637500	341.200000	345.462500	336.137500	341.825000	341.637500	340.667500	6.321588e+06	1.596711e+14	64106.500000	3.233135e+06
max	443.400000	445.000000	447.100000	437.000000	443.900000	443.400000	441.930000	4.797162e+08	1.570040e+16	351215.000000	1.875849e+08

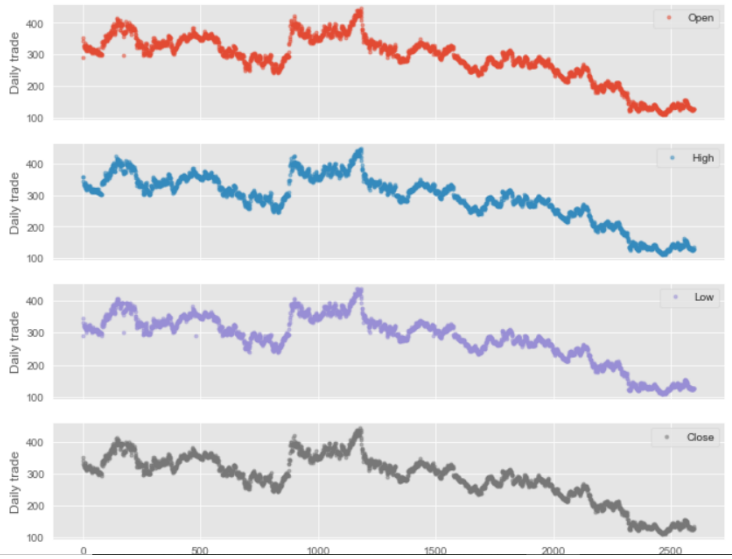
```
In [52]: data=data[['Date','Open','High','Low','Close','Volume']]

In [53]: data.isnull().sum()
```

Out[53]:

```
Date      0
Open      0
High      0
Low       0
Close     0
Volume    0
dtype: int64
```

Now, we visualize the trend of data. We can see below the variation in value of stock with time.

```
In [56]: cols_plot = ['Open', 'High', 'Low','Close']
axes = data[cols_plot].plot(marker='.', alpha=0.5, linestyle='None', figsize=(11, 9), subplots=True)
for ax in axes:
    ax.set_ylabel('Daily trade')
```



Using sklearn library, we split the data into training and testing data.

In [58]:
```
from sklearn.model_selection import train_test_split

X = []
Y = []
window_size=100
for i in range(1 , len(df) - window_size -1 , 1):
    first = df.iloc[i,2]
    temp = []
    temp2 = []
    for j in range(window_size):
        temp.append((df.iloc[i + j, 2] - first) / first)
    temp2.append((df.iloc[i + window_size, 2] - first) / first)
    X.append(np.array(temp).reshape(100, 1))
    Y.append(np.array(temp2).reshape(1, 1))

x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, shuffle=True)

train_X = np.array(x_train)
test_X = np.array(x_test)
train_Y = np.array(y_train)
test_Y = np.array(y_test)

train_X = train_X.reshape(train_X.shape[0],1,100,1)
test_X = test_X.reshape(test_X.shape[0],1,100,1)

print(len(train_X))
print(len(test_X))
```

```
1996
500
```

Using tenserflow, we build our model. We can add as many layers as we want to our model. Then we deploy our model.

In [59]:
```
# For creating model and training
import tensorflow as tf
from tensorflow.keras.layers import Conv1D, LSTM, Dense, Dropout, Bidirectional, TimeDistributed
from tensorflow.keras.layers import MaxPooling1D, Flatten
from tensorflow.keras.regularizers import L1, L2
from tensorflow.keras.metrics import Accuracy
from tensorflow.keras.metrics import RootMeanSquaredError

model = tf.keras.Sequential()

# Creating the Neural Network model here...
# CNN layers
model.add(TimeDistributed(Conv1D(64, kernel_size=3, activation='relu', input_shape=(None, 100, 1))))
model.add(TimeDistributed(MaxPooling1D(2)))
model.add(TimeDistributed(Conv1D(128, kernel_size=3, activation='relu')))
model.add(TimeDistributed(MaxPooling1D(2)))
model.add(TimeDistributed(Conv1D(64, kernel_size=3, activation='relu')))
model.add(TimeDistributed(MaxPooling1D(2)))
model.add(TimeDistributed(Flatten()))
# model.add(Dense(5, kernel_regularizer=L2(0.01)))

# LSTM layers
model.add(Bidirectional(LSTM(100, return_sequences=True)))
model.add(Dropout(0.5))
model.add(Bidirectional(LSTM(100, return_sequences=False)))
model.add(Dropout(0.5))

#Final layers
model.add(Dense(1, activation='linear'))
model.compile(optimizer='adam', loss='mse', metrics=['mse', 'mae'])

history = model.fit(train_X, train_Y, validation_data=(test_X,test_Y), epochs=40,batch_size=40, verbose=1, shuffle =True)
```

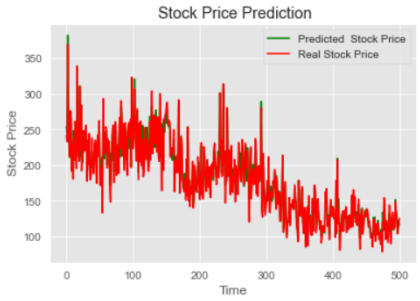
Here, we evaluate our model in depth. We find the error, variance and R2 score of the model.

```
In [64]: model.evaluate(test_X, test_Y)

         16/16 [==============================] - 0s 4ms/step - loss: 0.0023 - mse: 0.0023 - mae: 0.0378

Out[64]: [0.002256642561405897, 0.002256642561405897, 0.037760913372039795]
```

```
In [65]: from sklearn.metrics import explained_variance_score, mean_poisson_deviance, mean_gamma_deviance
         from sklearn.metrics import r2_score
         from sklearn.metrics import max_error

         # predict probabilities for test set
         yhat_probs = model.predict(test_X, verbose=0)
         # reduce to 1d array
         yhat_probs = yhat_probs[:, 0]

         var = explained_variance_score(test_Y.reshape(-1,1), yhat_probs)
         print('Variance: %f' % var)

         r2 = r2_score(test_Y.reshape(-1,1), yhat_probs)
         print('R2 Score: %f' % var)

         var2 = max_error(test_Y.reshape(-1,1), yhat_probs)
         print('Max Error: %f' % var2)

         Variance: 0.904963
         R2 Score: 0.904963
         Max Error: 0.178317
```

Then, we plot the actual result vs the predicted result to see our prediction.

```
In [66]: predicted  = model.predict(test_X)
         test_label = test_Y.reshape(-1,1)
         predicted = np.array(predicted[:,0]).reshape(-1,1)
         len_t = len(train_X)
         for j in range(len_t , len_t + len(test_X)):
             temp = data.iloc[j,3]
             test_label[j - len_t] = test_label[j - len_t] * temp + temp
             predicted[j - len_t] = predicted[j - len_t] * temp + temp
         plt.plot(predicted, color = 'green', label = 'Predicted  Stock Price')
         plt.plot(test_label, color = 'red', label = 'Real Stock Price')
         plt.title(' Stock Price Prediction')
         plt.xlabel('Time')
         plt.ylabel(' Stock Price')
         plt.legend()
         plt.show()
```



7. Conclusion

In this project we have demonstrated the implementation of stock prices prediction for the Indian market using machine learning algorithms.

Overall, the project has managed to answer the questions whether and how Machine Learning can be used in the field of stock market and have provided samples of output for different steps which we have implemented in our mini project.

8. References

[1] M. Usmani, S. H. Adil, K. Raza and S. S. A. Ali, "Stock market prediction using machine learning techniques," 2016 3rd International Conference on Computer and Information Sciences (ICCOINS), Kuala Lumpur, 2016, pp. 322-327.

[2] K. Raza, "Prediction of Stock Market performance by using machine learning techniques," 2017 International Conference on Innovations in Electrical Engineering and Computational Technologies (ICIEECT), Karachi, 2017, pp. 1-1.

[3] H. Gunduz, Z. Cataltepe and Y. Yaslan, "Stock market direction prediction using deep neural networks," 2017 25th Signal Processing and Communications Applications Conference (SIU), Antalya, 2017, pp. 1-4.

[4] M. Billah, S. Waheed and A. Hanifa, "Stock market prediction using an improved training algorithm of neural network," 2016 2nd International Conference on Electrical, Computer & Telecommunication Engineering (ICECTE), Rajshahi, 2016, pp. 1-4.

[5] H. L. Siew and M. J. Nordin, "Regression techniques for the prediction of stock price trend," 2012 International Conference on Statistics in Science, Business and Engineering (ICSSBE), Langkawi, 2012, pp. 1-5.

[6] K. V. Sujatha and S. M. Sundaram, "Stock index prediction using regression and neural network models under non normal conditions," INTERACT-2010, Chennai, 2010, pp. 59-63.

[7] S. Liu, G. Liao and Y. Ding, "Stock transaction prediction modeling and analysis based on LSTM," 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA), Wuhan, 2018, pp. 2787-2790.