ABSTRACT

In recent times, Stock prediction is one of the most complicated tasks. 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 Deep learning which makes predictions based on the values of current stock market indices by training on their previous values. This project focuses on the use of Recurrent Neutral Network based Deep learning. Factors considered are open, close, low, high and volume. The programming language is used to predict the stock market using Deep learning.

Stock market price prediction is a difficult undertaking that generally requires a lot of human-computer interaction. The stock market process is fraught with risk and is influenced by a variety of factors. Of all the market sectors, it is one of the most volatile and active. When buying and selling stocks from various corporations and businesses, more caution is required. As a result, stock market forecasting is an important endeavor in business and finance. This study analyzes one of the explicit forecasting tactics based on Machine Learning architectures and predictive algorithms and gives an independent model-based strategy for predicting stock prices. The predictor model is based on the Recurrent Neural Networks' LSTM (Long Short-Term Memory) architecture, which specializes in time series data classification and prediction.

TABLE OF CONTENTS

1. Introduction

The stock market refers to the collection of markets and exchanges where regular activities of buying, selling, and issuance of shares of publicly-held companies take place. When you buy a company's stock, you're purchasing a small piece of that company, called a share.Investors purchase stocks in companies they think will go up in value. If that happens, the company's stock increases in value as well. The stock can then be sold for a profit.When you own stock in a company, you are called a shareholder because you share in the company's profits.

Time-series prediction is a common technique widely used in many real-world applications such as weather forecasting and financial market prediction. The most common algorithms now are based on Recurrent Neural Networks (RNN), as well as its special type - Long-short Term Memory (LSTM).Short-term forecasting, medium-term forecasting, and long-term forecasting are the three types of stock price forecasting. Forecasting for a few seconds, minutes, days, weeks, or months is referred to as short-term forecasting. Forecasting for one or two years is referred to as medium-term predicting, while forecasting for more than two years is referred to as long-term forecasting. LSTMs are very powerful in sequence prediction problems because they're able to store past information. This is important in our case because the previous price of a stock is crucial in predicting its future price. 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.

1.1. Concept an Overview

The main motivation of doing this research is to present a stock prediction model for the prediction of occurrence of stocks. Stock market prediction is basically defined as trying to determine the stock value and offer a robust idea for the people to know and predict the market and the stock prices. It is generally presented using the quarterly financial ratio using the dataset. Thus, relying on a single dataset may not be sufficient for the prediction and can give a result which is inaccurate. Hence, we are contemplating towards the study of machine learning with various datasets integration to predict the market and the stock trends.

2.SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

The existing system fails when there are rare outcomes or predictors, as the algorithm is based on bootstrap sampling.

- The previous results indicate that the stock price is unpredictable when the traditional classifier is used.
- The existence system reported highly predictive values, by selecting an appropriate time period for their experiment to obtain highly predictive scores.
- The existing system does not perform well when there is a change in the operating environment.
- It doesn't focus on external events in the environment, like news events or social media.
- The existing system needs some form of input interpretation, thus need of scaling.
- It doesn't exploit data pre-processing techniques to remove inconsistency and incompleteness of the data.

2.2.PROPOSED SYSTEM

• In this proposed system, This project focus on predicting the stock values using machine learning algorithms. the system “Stock market prediction” have predicted the stock market price using the Recurrent Neutral Network based Machine learning.

• The first one was numpy, which was used to clean and manipulate the data, and getting it into a form ready for analysis. The other was scikit, which was used for real analysis and prediction. The data set we used was from the previous years stock markets collected from the public database available online, 80 % of data was used to train the machine and the rest 20 % to test the data.

• The basic approach of the supervised learning model is to learn the patterns and relationships in the data from the training set and then reproduce them for the test data. We used the python pandas library for data processing which combined different datasets into a data frame. The tuned up dataframe allowed us to prepare the data for feature extraction. The dataframe features were date and the closing price for a particular day. predicted the object variable, which is the price for a given day. We also quantified the accuracy by using the predictions for the test set and the actual values. The proposed system touches different areas of research including data pre-processing.

2. METHODOLOGY

2.1. PROBLEM DEFINITION

The stock market appears in the news every day. You hear about it every time it reaches a new high or a new low. The rate of investment and business opportunities in the Stock market can increase if an efficient algorithm could be devised to predict the short term price of an individual stock.

Previous methods of stock predictions involve the use of Artificial Neural Networks and Convolution Neural Networks which has an error loss at an average of 20%.

In this report, we will see if there is a possibility of devising a model using Recurrent Neural Network which will predict stock price with a less percentage of error. And if the answer turns to be YES, we will also see how reliable and efficient will this model be.

2.2. OBJECTIVE OF THE PROJECT

In the past decades, there is an increasing interest in predicting markets among economists, policymakers, academics and market makers. The objective of the proposed work is to study and improve the supervised learning algorithms to predict the stock price.

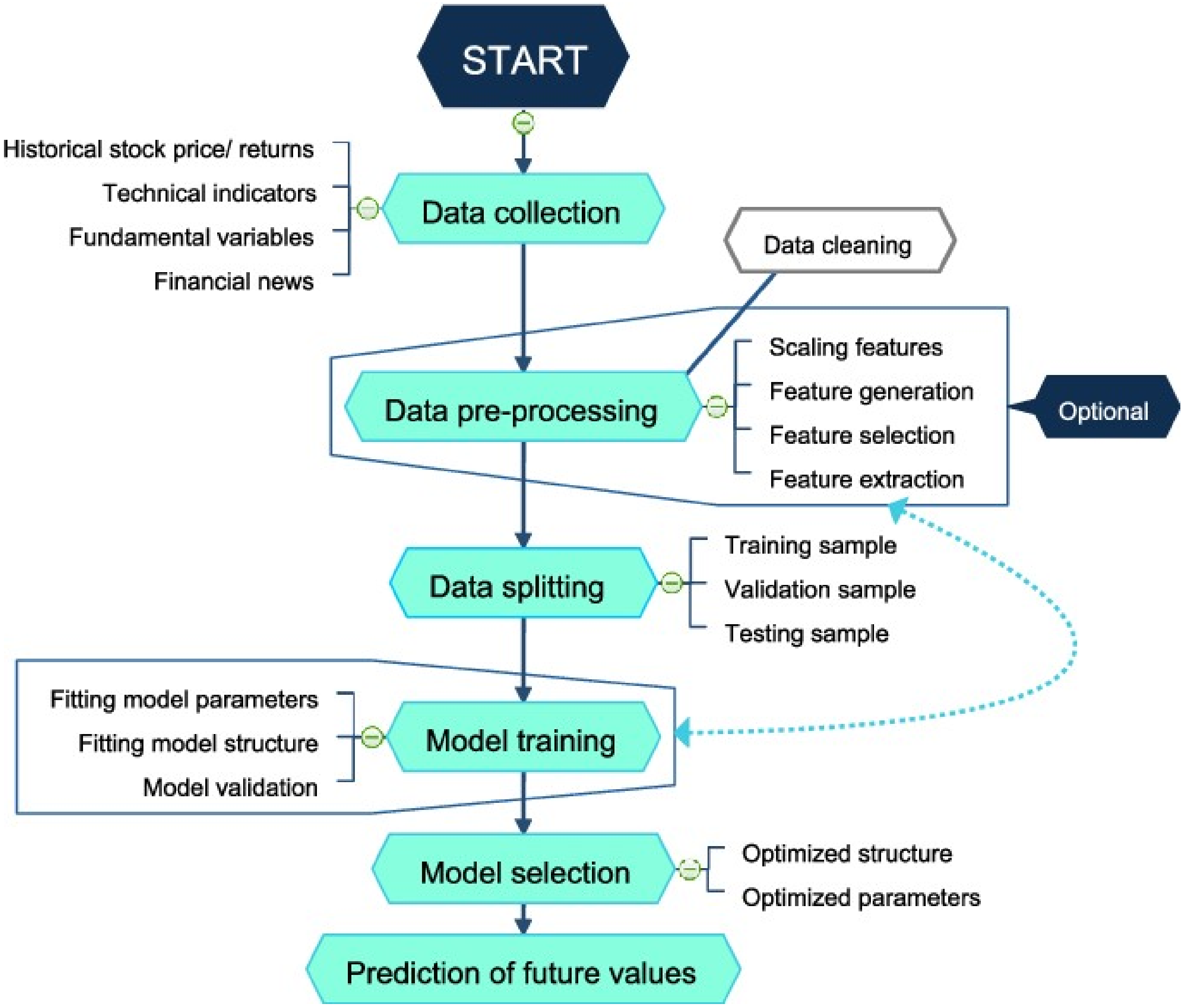
Technical Objective

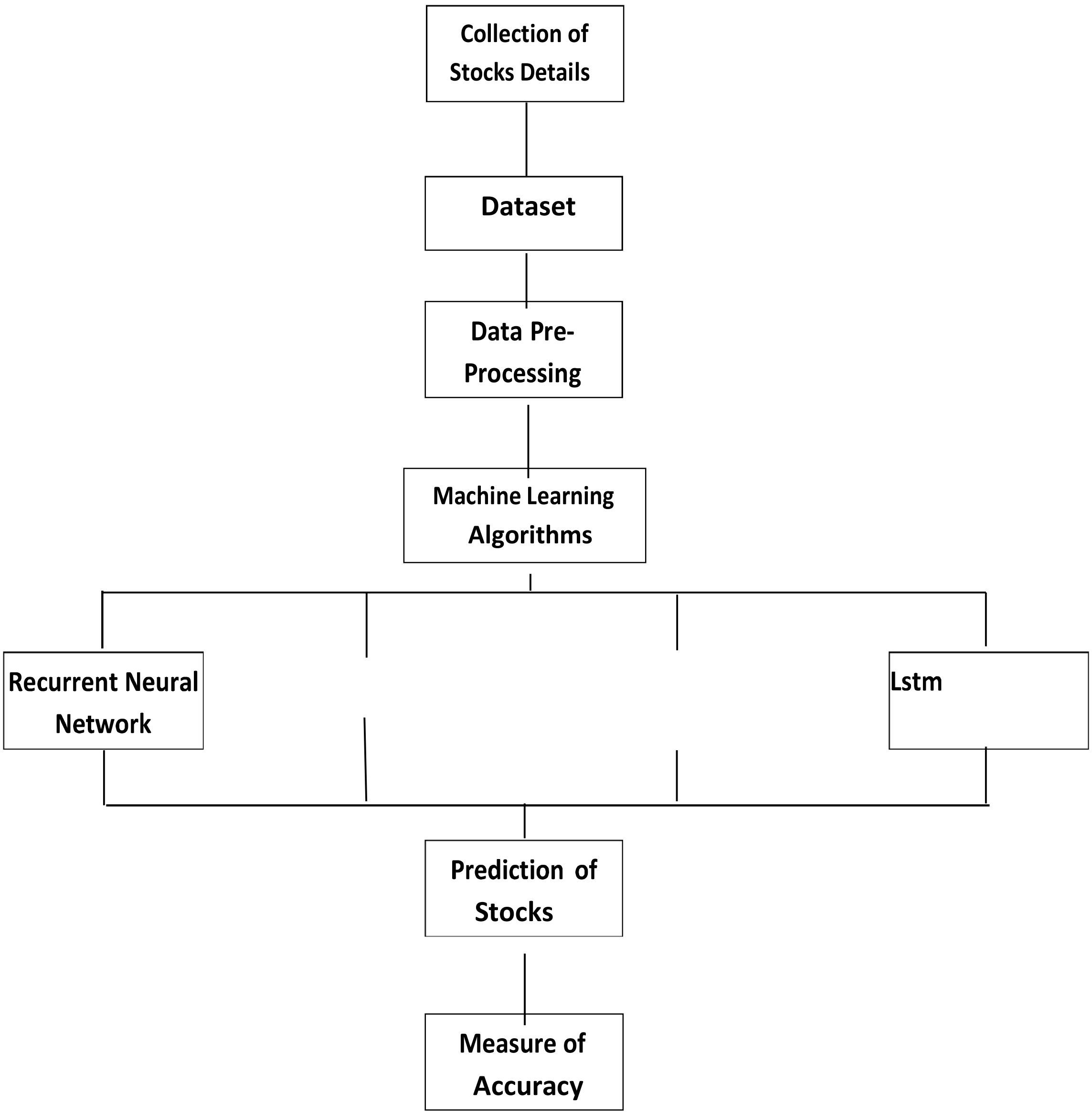
The technical objectives will be implemented in Python. The system must be able to access a list of historical prices. It must calculate the estimated price of stock based on the historical data. It must also provide an instantaneous visualization of the market index.

Experimental Objective

The experimental objective will be to compare the forecasting ability of Recurrent Neural Network. We will test and evaluate both the systems with same test data to find their prediction accuracy

2.3. BLOCK DIAGRAM OF THE PROJECT

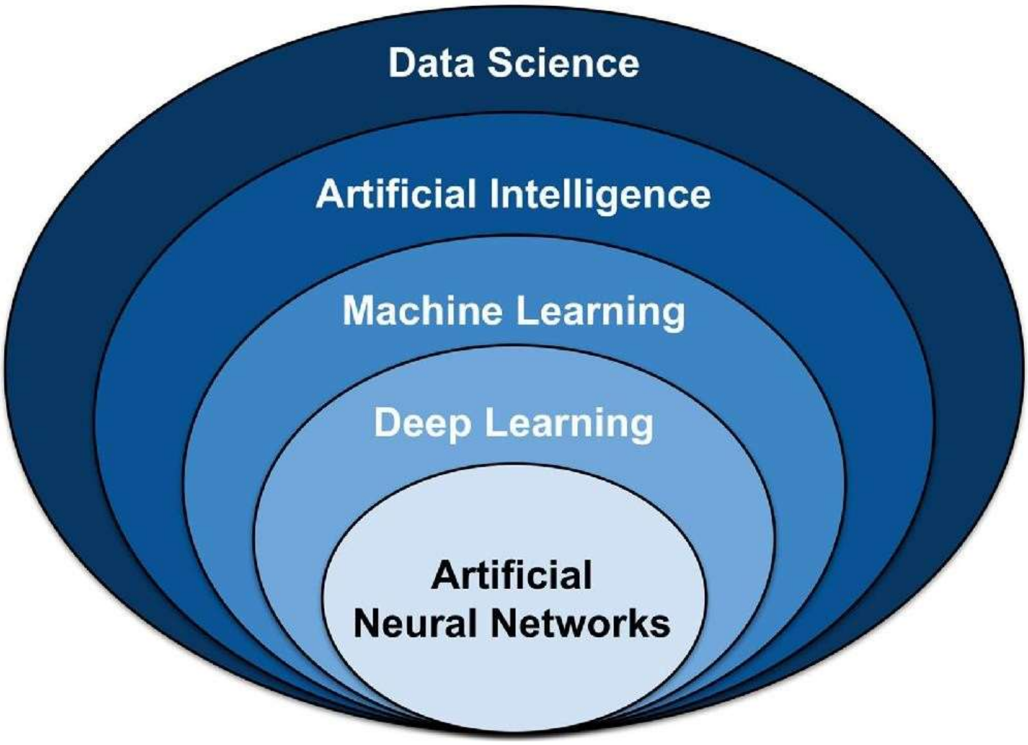



Collection of Stocks Details
Dataset
Data Pre-Processing
Machine Learning Algorithms
Recurrent Neural Network
Lstm
Prediction of Stocks
Measure of Accuracy

2.4. MODULE REQUIREMENTS

- COLLECTION OF DATASET
- SELECTION OF ATTRIBUTES
- DATA PRE-PROCESSING
- DATA VISUALIZATION
- FEATURE SCALING
- X train Y train
- PREDICT

2.5. MACHINE LEARNING



Machine learning, classification refers to a predictive modelling problem where a class label is predicted for a given example of input data. 10 Supervised Learning Supervised learning is the type of machine learning in which machines are trained using well "labelled"training data, and on the basis of that data, machines predict the output. The labelled data means some input data is already tagged with the correct output. In supervised learning, the training data provided to the machines work as the supervisor that teaches the machinesto predict the output correctly. It applies the same concept as a student learns in the supervision of the teacher.

SUPERVISED LEARNING:

Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to find a mapping function to map the input variable(x) with the output variable(y). Unsupervised learning unsupervised learning cannot be directly applied to a regression or classification problem because unlike supervised learning, we have the input data but no correspondingoutput data. The goal of unsupervised learning is to find the underlying structure of dataset,group that data according to similarities, and represent that dataset in a compressed format.Unsupervised learning is helpful for finding useful insights from the data.

UNSUPERVISED LEARNING

Unsupervised learning is much similar to how a human learns to think by their own experiences, which makes it closer to the real AI. Unsupervised learning works on unlabelled and uncategorized data which make unsupervised learning more important. Inreal-world, we do not always have input data with the corresponding output so to solve such cases, we need unsupervised learning. Reinforcement learning.

REINFORCEMENT LEARNING

Reinforcement learning is an area of Machine Learning. It is about taking suitable action to maximize reward in a particular situation. It is employed by various software and machines to find the best possible behaviour or path it should take in a specific situation. Reinforcement learning differs from supervised learning in a way that in supervised learning the training data has the answer key with it so the model is 11 trained with the correct answer itself whereas in reinforcement learning, there is no answer but thereinforcement agent decides what to do to perform the given task. In the absence of a training dataset, it is bound to learn from its experience.

DEEP LEARNING:

Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers.Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks.deep learning works with artificial neural networks, which are designed to imitate how humans think and learn.

2.6. ALGORITHM

Neural Network

A Neural Network consists of different layers connected to each other, working on the structure and function of a human brain. It learns from huge volumes of data and uses complex algorithms to train a neural net.

Here is an example of how neural networks can identify a dog's breed based on their features.

- The image pixels of two different breeds of dogs are fed to the input layer of the neural network.
- The image pixels are then processed in the hidden layers for feature extraction.
- The output layer produces the result to identify if it's a German Shepherd or a Labrador.
- Such networks do not require memorizing the past output.

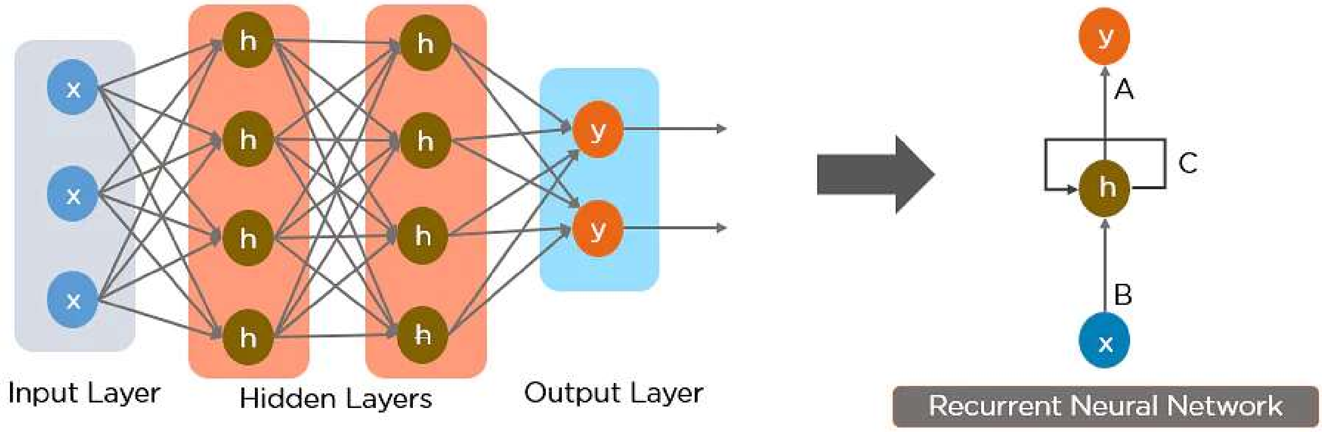
Several neural networks can help solve different business problems. Let's look at a few of them.

- Feed-Forward Neural Network: Used for general Regression and Classification problems.
- Convolutional Neural Network: Used for object detection and image classification.
- Deep Belief Network: Used in healthcare sectors for cancer detection.
- RNN: Used for speech recognition, voice recognition, time series prediction, and natural language processing.

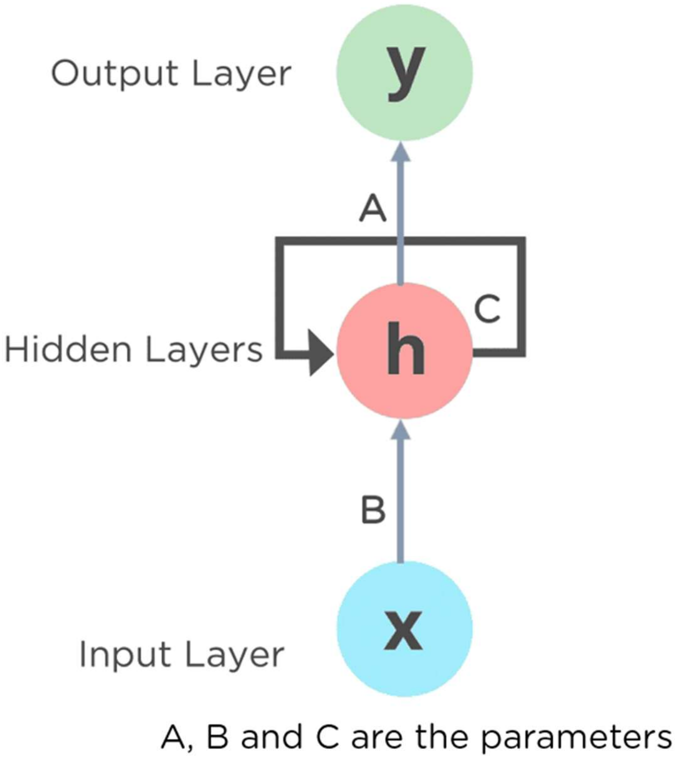
Recurrent Neural Network (RNN)

RNN works on the principle of saving the output of a particular layer and feeding this back to the input in order to predict the output of the layer.

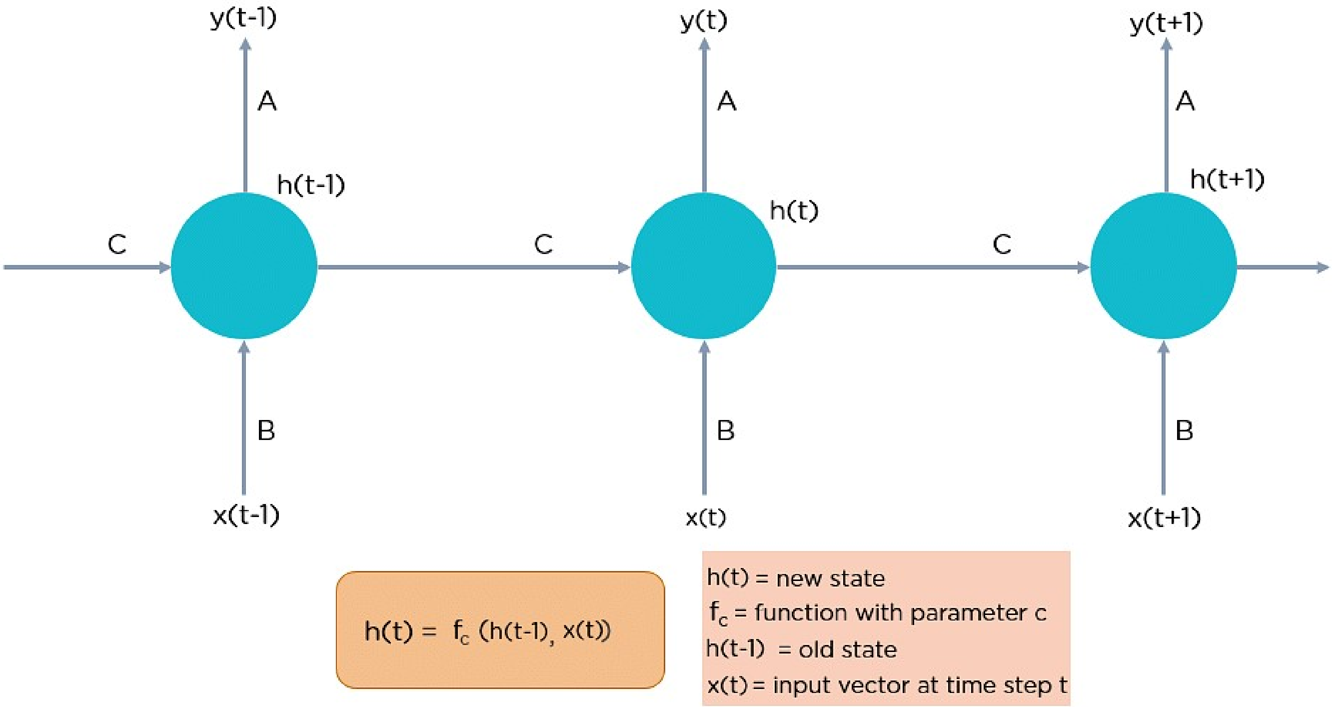
Below is how you can convert a Feed-Forward Neural Network into a Recurrent Neural Network:



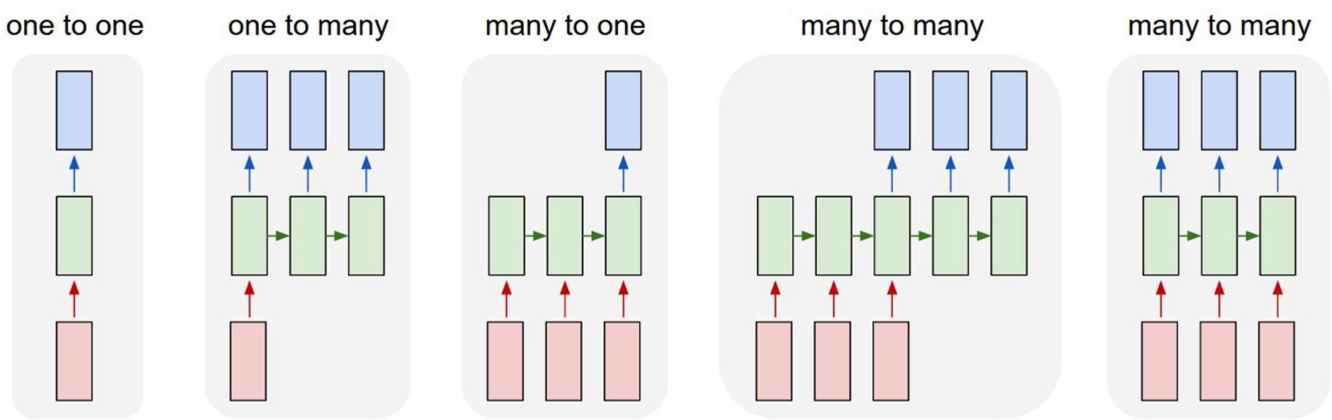
The nodes in different layers of the neural network are compressed to form a single layer of recurrent neural networks. A, B, and C are the parameters of the network.



Here, “x” is the input layer, “h” is the hidden layer, and “y” is the output layer. A, B, and C are the network parameters used to improve the output of the model. At any given time t, the current input is a combination of input at x(t) and x(t-1). The output at any given time is fetched back to the network to improve on the output.



Different types of RNN's



Why Recurrent Neural Networks?

RNN were created because there were a few issues in the feed-forward neural network:

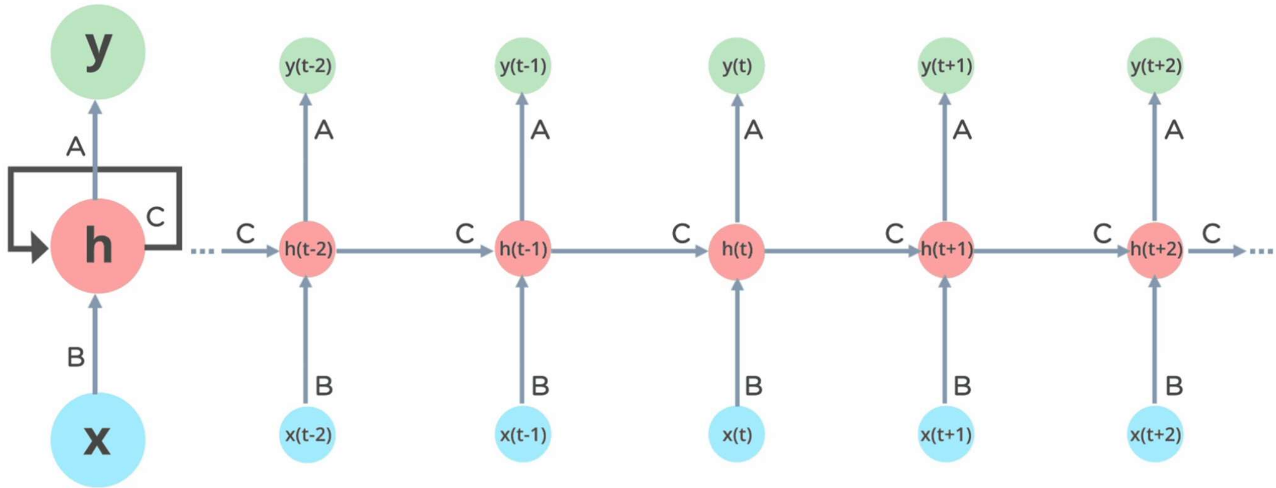
- Cannot handle sequential data

- Considers only the current input
- Cannot memorize previous inputs

The solution to these issues is the RNN. An RNN can handle sequential data, accepting the current input data, and previously received inputs. RNNs can memorize previous inputs due to their internal memory.

How Does Recurrent Neural Networks Work?

In Recurrent Neural networks, the information cycles through a loop to the middle hidden layer.



Working of Recurrent Neural Network

The input layer 'x' takes in the input to the neural network and processes it and passes it onto the middle layer.

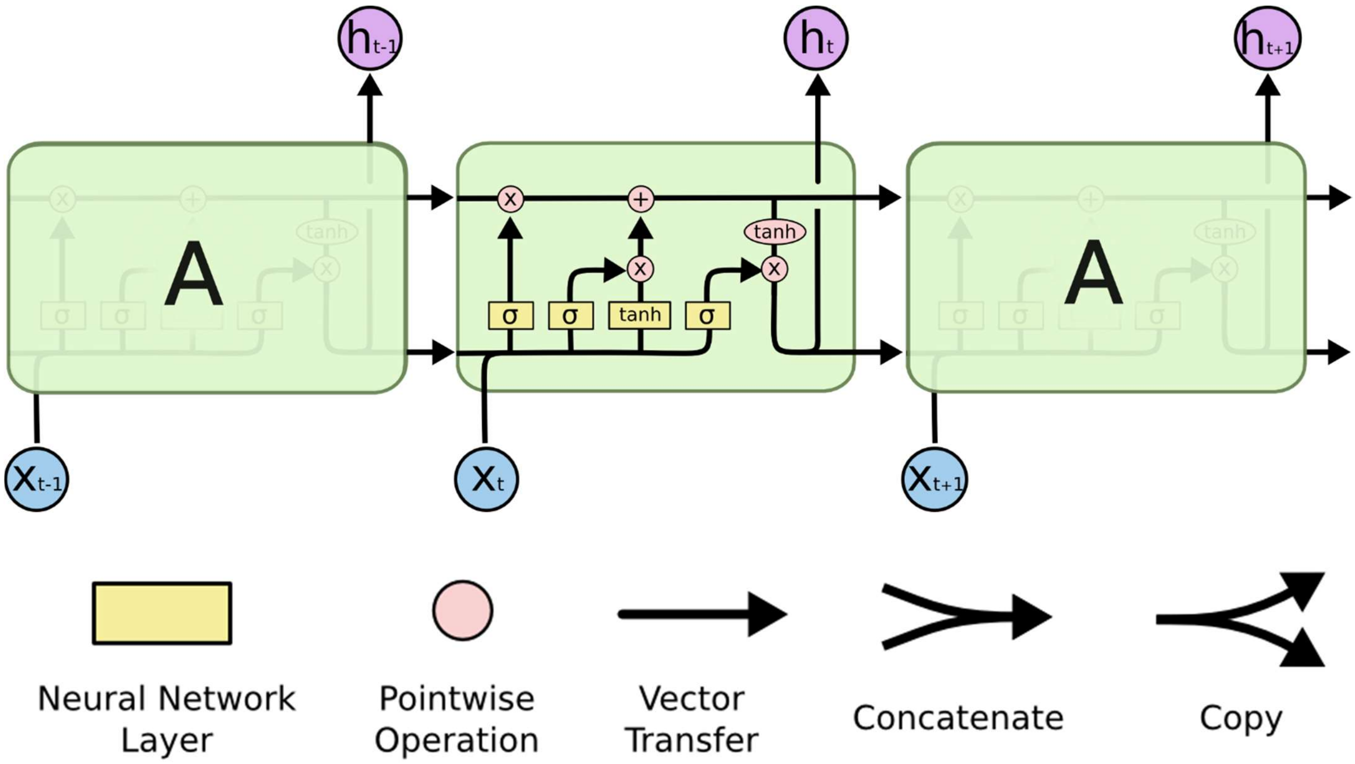
The middle layer 'h' can consist of multiple hidden layers, each with its own activation functions and weights and biases. If you have a neural network where the various parameters of different hidden layers are not affected by the previous layer, ie: the neural network does not have memory, then you can use a recurrent neural network.

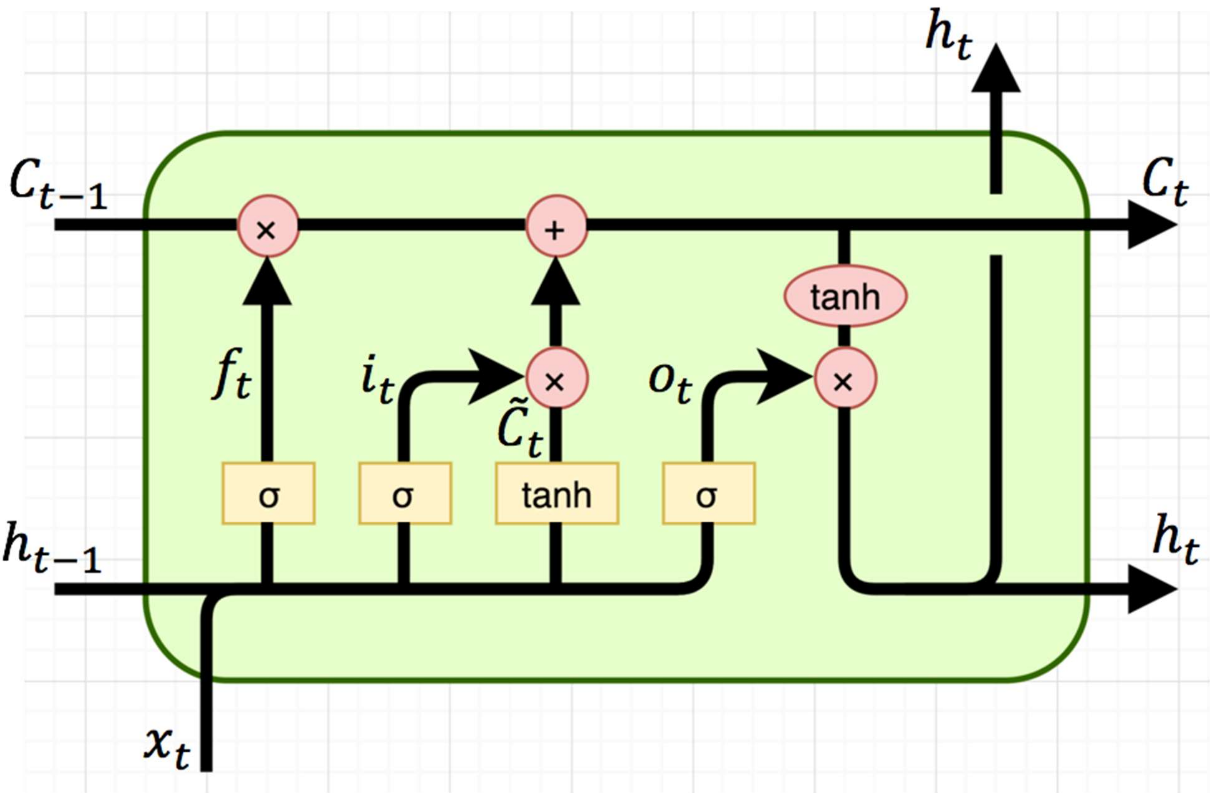
The Recurrent Neural Network will standardize the different activation functions and weights and biases so that each hidden layer has the same parameters. Then, instead of creating multiple hidden layers, it will create one and loop over it as many times as required.

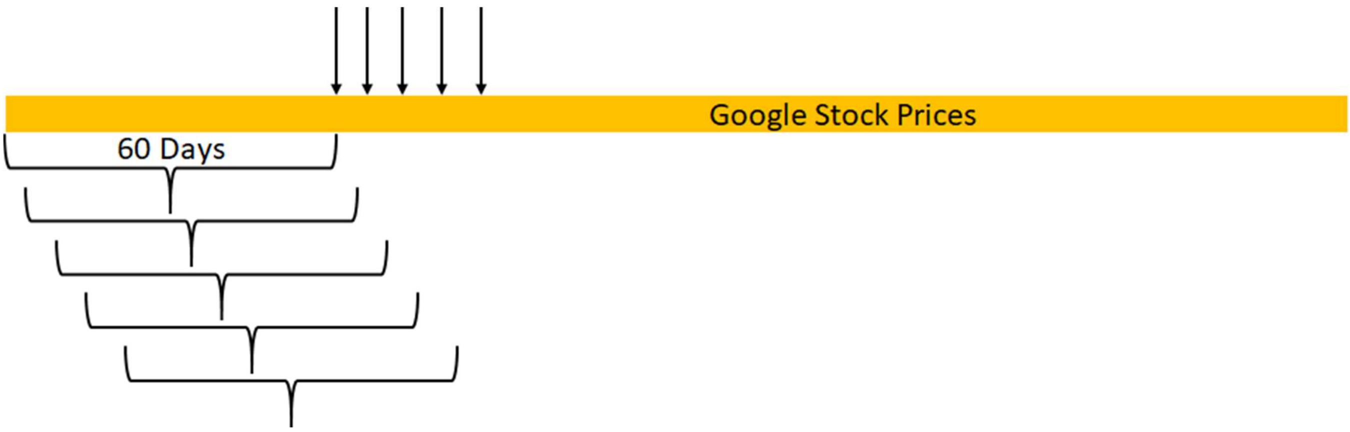
Long Short Term Memory (LSTM) Networks

Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies.

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn




h_t
C_{t-1}
C_t
×
+
tanh
f_t
i_t
×
$\tilde{C}_t$
o_t
×
σ
σ
tanh
σ
h_{t-1}
h_t
x_t


Google Stock Prices
60 Days

3.LANGUAGES/TOOLS, DATASET

3.2. LANGUAGES / TOOLS DESCRIPTION

HARDWARE REQUIREMENTS

Operating System	:	Windows 10
RAM	:	8.00GB
System Type	:	64-bit OS
Processor	:	Intel(R) core i5

SOFTWARE REQUIREMENTS

Front-End	:	Python
Scripts	:	Python3.7.4 Shell
Tools	:	Google Colab

PYTHON

In technical terms, Python is an object-oriented, high-level programming language with integrated dynamic semantics primarily for web and app development. It is extremely attractive in the field of Rapid ApplicationDevelopment because it offers dynamic typing and dynamic binding options. Python is relatively simple, so it's easy to learn since it requires a unique syntax that focuses on readability. Developers can read and translate Python code much easier than other languages. In turn, this reduces the cost of program maintenance and development because it allows teams to work collaborativelywithout significant language and experience barriers. Additionally, Python supports the use of modules and packages, which means that programs can be designed in a modular style and code can be reused across a variety of projects.Once you've developed a module or package you need, it can be scaled for usein other projects, and it's easy to import or export these modules.Python is an open sourcecommunity language.

PYTHON FEATURES

Easy-to-learn – Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

Easy-to-read − Python code is more clearly defined and visible to the eyes.

Easy-to-maintain − Python's source code is fairly easy-to-maintain.

A broad standard library − Python's bulk of the library is very portable and cross- platform compatible on UNIX, Windows, and Macintosh.

Interactive Mode − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

Portable − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

Extendable −You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

Databases − Python provides interfaces to all major commercial databases.

GUI Programming − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

Scalable − Python provides a better structure and support for large programs than shell scripting.

Uses of Python

Python is a general-purpose programming language, which is another way to say that it can be used for nearly everything. Most importantly, it is an interpreted languages

Which means that the written code is not actually translated to a computer- readable format at runtime. Whereas, most programming languages do this conversion before the program is even run. This type of language is also referred to as a "scripting language" because it was initially meant to be used for trivial projects.

Import Important Packages

Sklearn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

TensorFlow

TensorFlow is an open source machine learning framework for all developers. It is used for implementing machine learning and deep learning applications. To develop and research on fascinating ideas on artificial intelligence, Google team created TensorFlow. TensorFlow is designed in Python programming language, hence it is considered an easy to understand framework.

3.3. PACKAGES, FUNCTIONS

Numpy

NumPy is the fundamental package for scientific computing in Python. It is a Pythonlibrary that provides a multidimensional array object, various derived objects (suchas masked arrays and matrices), and an assortment of routines for fast operations onarrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O,discrete Fourier transforms, basic linear algebra, basic statistical operations, randomsimulation and much more.

At the core of the NumPy package, is the *ndarray* object. This encapsulates *n*- dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

- NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an *ndarray* will create a new array and delete the original.
- The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arraysof (Python, including NumPy) objects, thereby allowing for arrays of differentsized elements.
- NumPy arrays facilitate advanced mathematical and other types of operationson large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences.
- A growing plethora of scientific and mathematical Python-based packages areusing NumPy arrays; though these typically support Python-sequence input,

they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today's scientific/mathematical Python-basedsoftware, just knowing how to use Python's built-in sequence types is insufficient - one also needs to know how to use NumPy arrays.

Pandas

Pandas is an open source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another packagenamed Numpy, which provides support for multi-dimensional arrays. As one of themost popular data wrangling packages, Pandas works well with many other data science modules inside the Python ecosystem, and is typically included in every Python distribution, from those that come with your operating system to commercialvendor distributions like Active State's Active Python.

Pandas makes it simple to do many of the time consuming, repetitive tasks associatedwith working with data, including:

- Data cleansing
- Data fill
- Data normalization
- Merges and joins
- Data visualization
- Statistical analysis
- Data inspection

3.4. DATASET

- https://finance.yahoo.com/quote/GOOG/history/

- This data set has 7 columns with all the necessary values such as opening price of the stock, the closing price of it, its highest in the day and much more.

	A	B	C	D	E	F	G
1	Date	Open	High	Low	Close	Adj Close	Volume
2	19-08-2004	50.05005	52.08208	48.02803	50.22022	50.22022	44659000
3	20-08-2004	50.55556	54.59459	50.3003	54.20921	54.20921	22834300
4	23-08-2004	55.43043	56.7968	54.57958	54.75475	54.75475	18256100
5	24-08-2004	55.67567	55.85585	51.83684	52.48749	52.48749	15247300
6	25-08-2004	52.53253	54.05405	51.99199	53.05306	53.05306	9188600
7	26-08-2004	52.52753	54.02903	52.38238	54.00901	54.00901	7094800
8	27-08-2004	54.1041	54.36437	52.8979	53.12813	53.12813	6211700
9	30-08-2004	52.69269	52.7978	51.05606	51.05606	51.05606	5196700
10	31-08-2004	51.2012	51.90691	51.13113	51.23624	51.23624	4917800
11	01-09-2004	51.4014	51.53654	49.88488	50.17518	50.17518	9138200
12	02-09-2004	49.64465	51.23624	49.51952	50.80581	50.80581	15118600
13	03-09-2004	50.52552	50.92092	49.70971	50.05505	50.05505	5152400
14	07-09-2004	50.55556	51.05105	49.85486	50.84084	50.84084	5847500
15	08-09-2004	50.42042	51.56657	50.3003	51.2012	51.2012	4985600
16	09-09-2004	51.31632	51.40641	50.55055	51.20621	51.20621	4061700
17	10-09-2004	50.85085	53.33333	50.7007	52.71772	52.71772	8698800
18	13-09-2004	53.36837	54.25926	53.28328	53.8038	53.8038	7844100
19	14-09-2004	53.77878	56.05606	53.44845	55.8008	55.8008	10828900
20	15-09-2004	55.33534	57.17217	55.15516	56.05606	56.05606	10713000
21	16-09-2004	56.22623	57.95796	55.88088	57.04204	57.04204	9266300

Input dataset attributes

- Date
- Open
- High
- Low
- Close
- Adj Close
- Volume

3.5. DATA STORAGE

DATA STORAGE DEFINITION

There are two types of digital information: input and output data. Users provide the input data. Computers provide output data. But a computer's CPU can'tcompute anything or produce output data without the user's input.

Users can enter the input data directly into a computer. However, they have found early on in the computer-era that continually entering data manually is time- and energy-prohibitive. One short-term solution is computer memory, also known as random access memory (RAM). But its storage capacity and memory retention are limited. Read-only memory (ROM) is, as the name suggests, the data can only be read but not necessarily edited. They control a computer's basic functionality.

Although advances have been made in computer memory with dynamic RAM (DRAM) and synchronous DRAM (SDRAM), they are still limited by cost, space and memory retention. When a computer powers down, so does the RAM's ability to retain data.

With data storage space, users can save data onto a device. And should the computerpower down, the data is retained. And instead of manually entering data into a computer, users can instruct the computer to pull data from storage devices. Computers can read input data from various sources as needed, and it can then createand save the output to the same sources or other storage locations. Users can also share data storage with others.

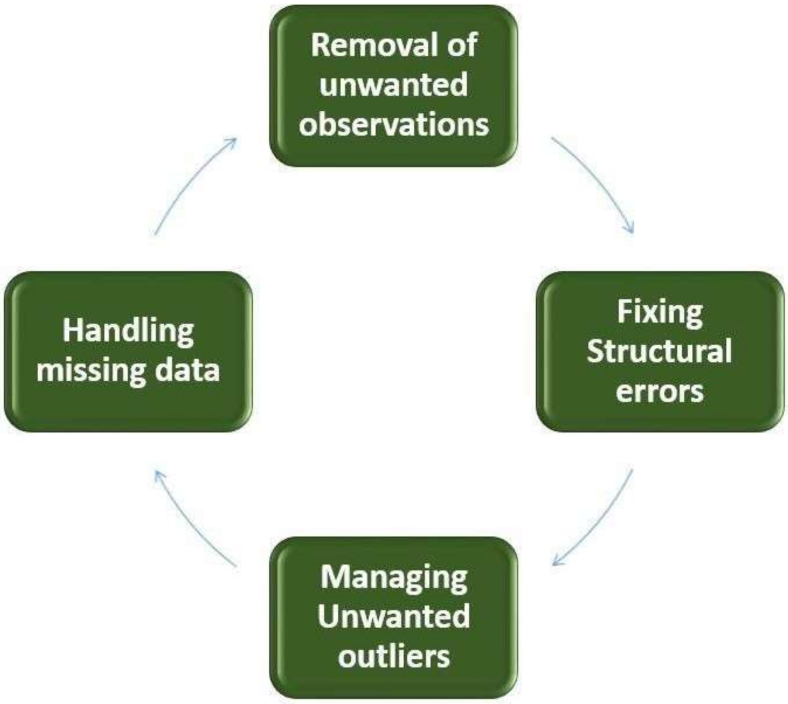
Today, organizations and users require data storage to meet today's high-level computational needs like big data projects, artificial intelligence (AI), machine learning and the internet of things (IoT).

3.6. DATA CLEANING

Data cleaning is one of the important parts of machine learning. It plays asignificant part in building a model. It surely isn't the fanciest part of machinelearning and at the same time, there aren't any hidden tricks or secrets to uncover.However, the success or failure of a project relies on proper data cleaning.Professional data scientists usually invest a very large portion of their time in thisstep because of the belief that **"Better data beats fancier algorithms".**

If we have a well-cleaned dataset, there are chances that we can get achieve good results with simple algorithms also, which can prove very beneficial at times especially in terms of computation when the dataset size is large. Obviously, different types of data will require different types of cleaning. However, thissystematic approach can always serve as a good starting point.

Steps involved in Data Cleaning:



4. MODULE DESCRIPTION

4.1 COLLECTION OF DATASET

Initially, we collect a dataset for our Stock market prediction. After the collection of the dataset, we split the dataset into training data and testing data. Thetraining dataset is used for prediction model learning and testing data is used for evaluating the prediction model. For this project, 70% of training data is used and 30% of data is used for testing. The dataset used for this project is Stock Price.

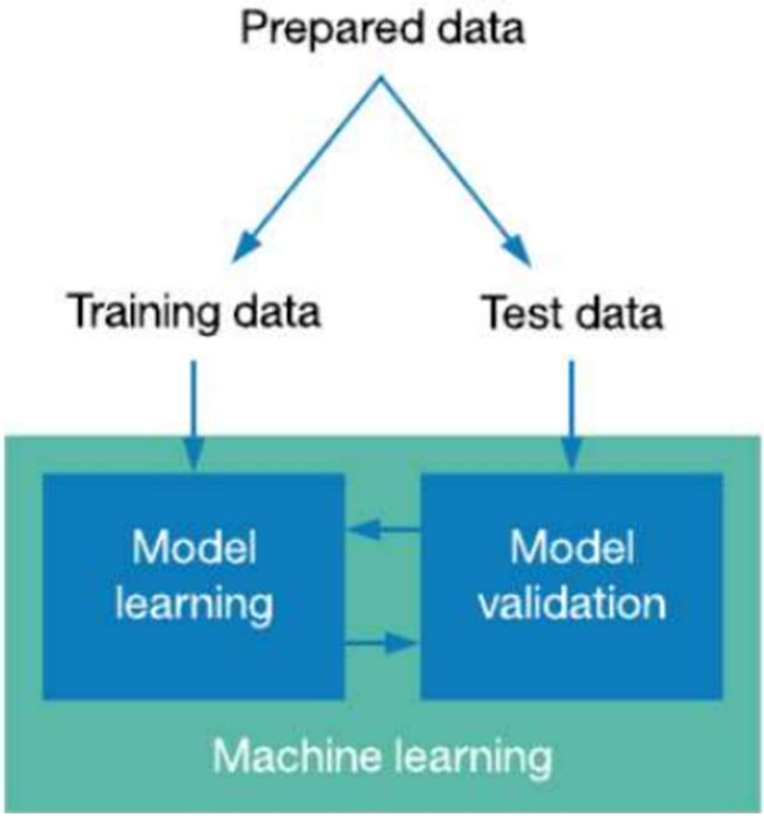
4.2 SELECTION OF ATTRIBUTES

Attribute or Feature selection includes the selection of appropriate attributes for the prediction system. This is used to increase the efficiency of the system.

4.3 PRE-PROCESSING OF DATA

Data pre-processing is an important step for the creation of a machine learning model. Initially, data may not be clean or in the required format for the model whichcan cause misleading outcomes. In pre-processing of data, we transform data into our required format. It is used to deal with noises, duplicates, and missing values ofthe dataset. Data pre-processing has the activities like importing datasets, splitting datasets, attribute scaling, etc. Preprocessing of data is required for improving the accuracy of the model.

Data Preparation



BALANCING OF DATA

Imbalanced datasets can be balanced in two ways. They are

Under Sampling and Over Sampling

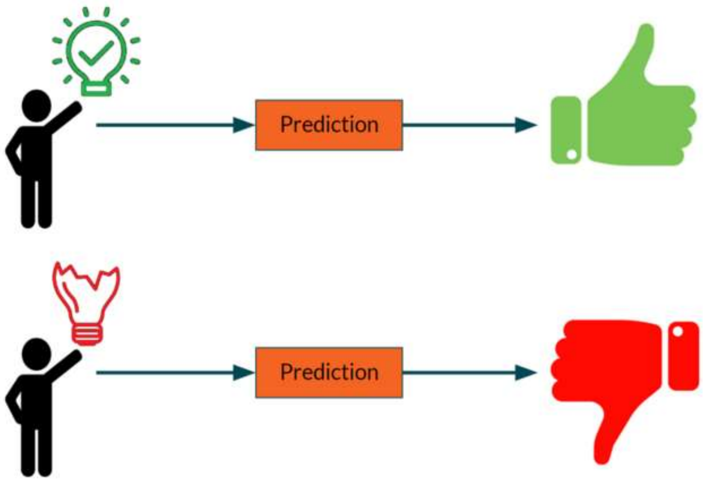
(a) Under Sampling:

In Under Sampling, dataset balance is done by the reduction of the size of the ample class. This process is considered when the amount of data is adequate.

(b) Over Sampling:

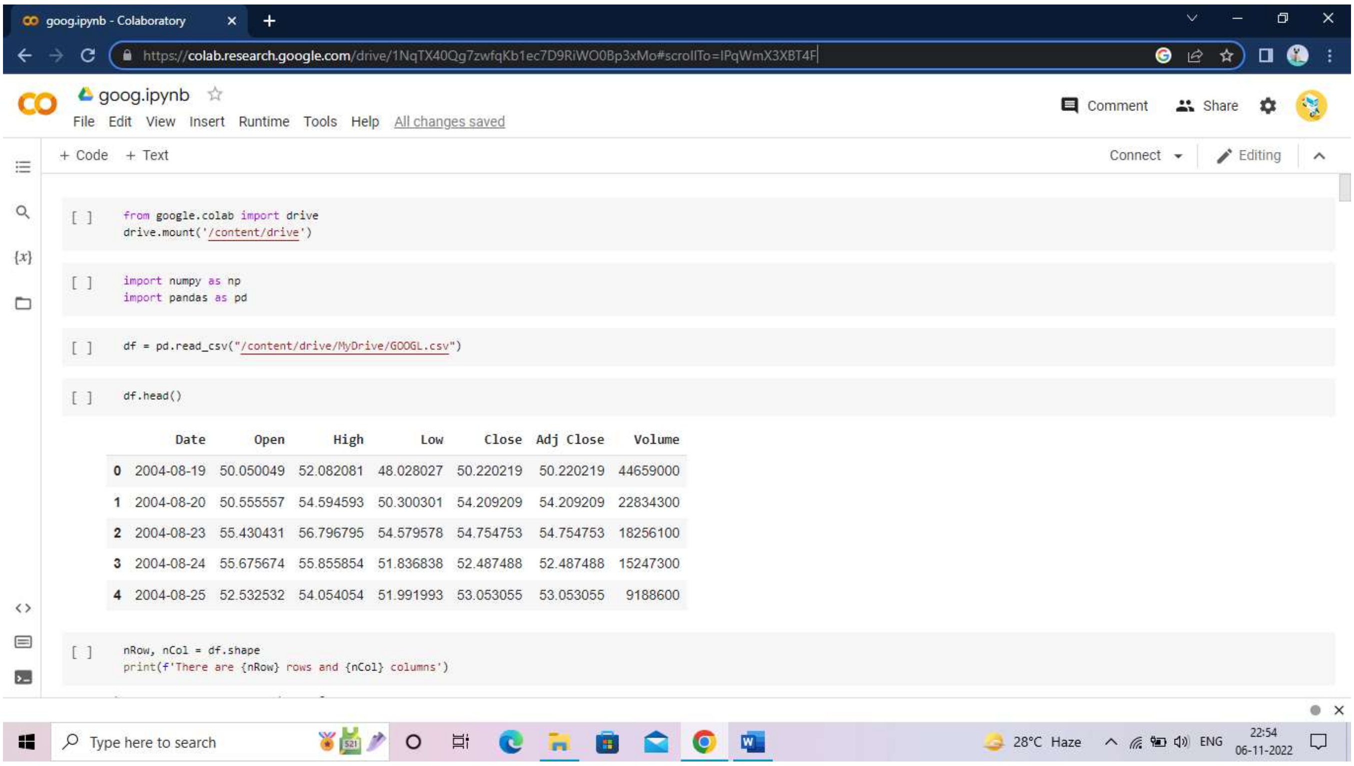
In Over Sampling, dataset balance is done by increasing the size of the scarce samples. This process is considered when the amount of data is inadequate.

4.4 PREDICTION OF STOCKS



5. RESULTS AND DISCUSSION

5.1. INPUT AND OUTPUT SCREENS



```
from google.colab import drive
drive.mount('/content/drive')
```

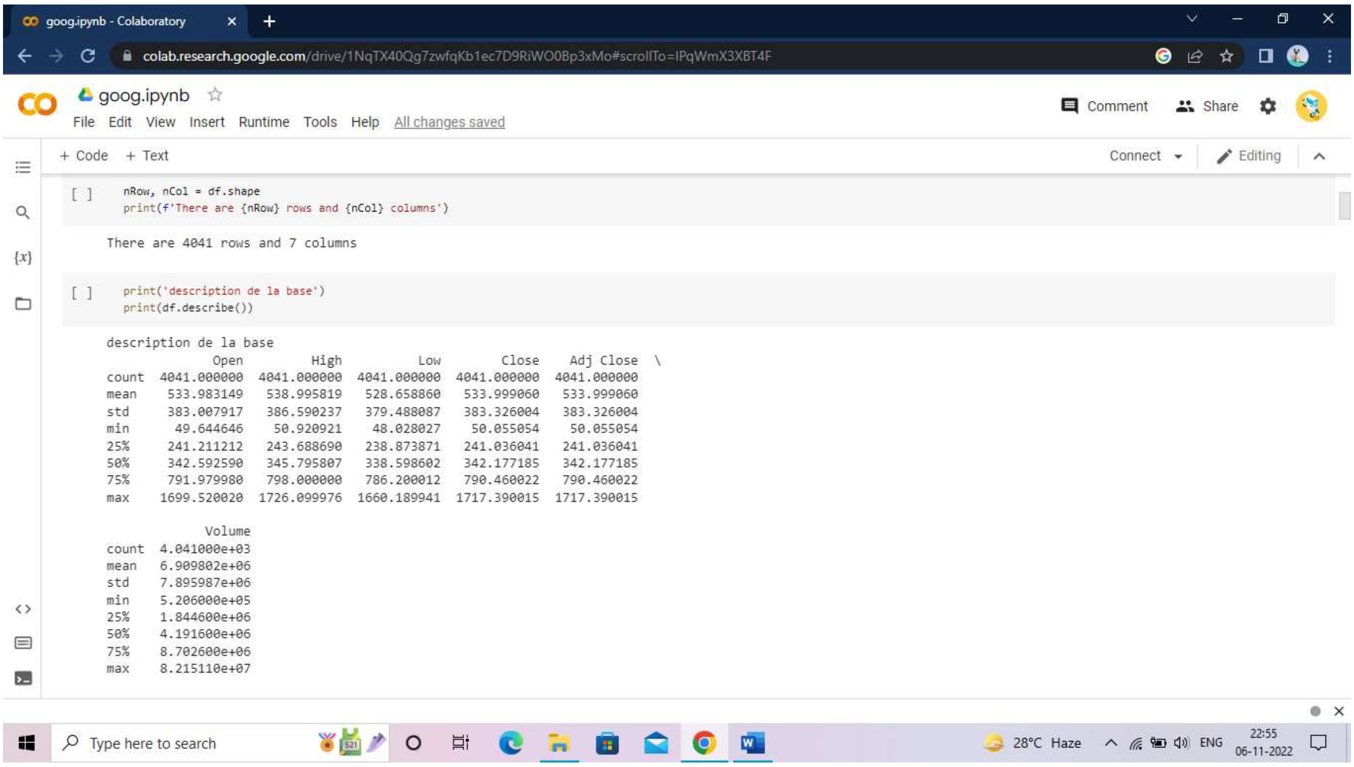
```
import numpy as np
import pandas as pd
```

```
df = pd.read_csv("/content/drive/MyDrive/GOOGL.csv")
```

```
df.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2004-08-19	50.050049	52.082081	48.028027	50.220219	50.220219	44659000
1	2004-08-20	50.555557	54.594593	50.300301	54.209209	54.209209	22834300
2	2004-08-23	55.430431	56.796795	54.579578	54.754753	54.754753	18256100
3	2004-08-24	55.675674	55.855854	51.836838	52.487488	52.487488	15247300
4	2004-08-25	52.532532	54.054054	51.991993	53.053055	53.053055	9188600

```
nRow, nCol = df.shape
print(f'There are {nRow} rows and {nCol} columns')
```

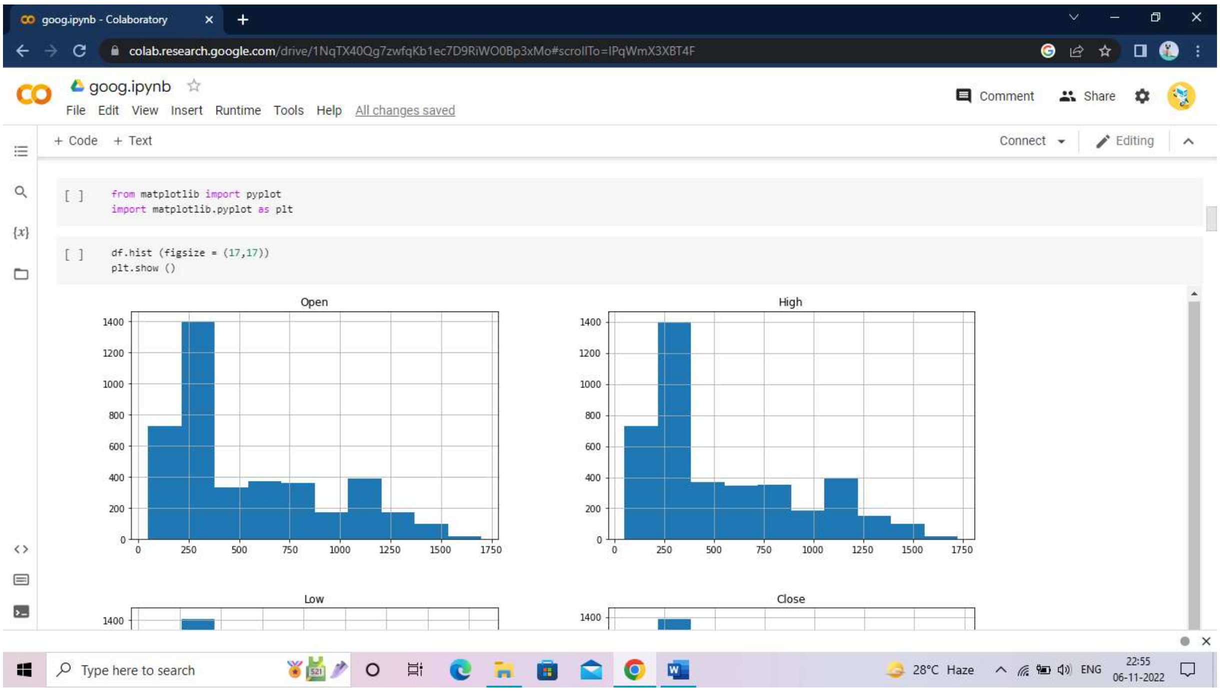


```
nRow, nCol = df.shape
print(f'There are {nRow} rows and {nCol} columns')
```

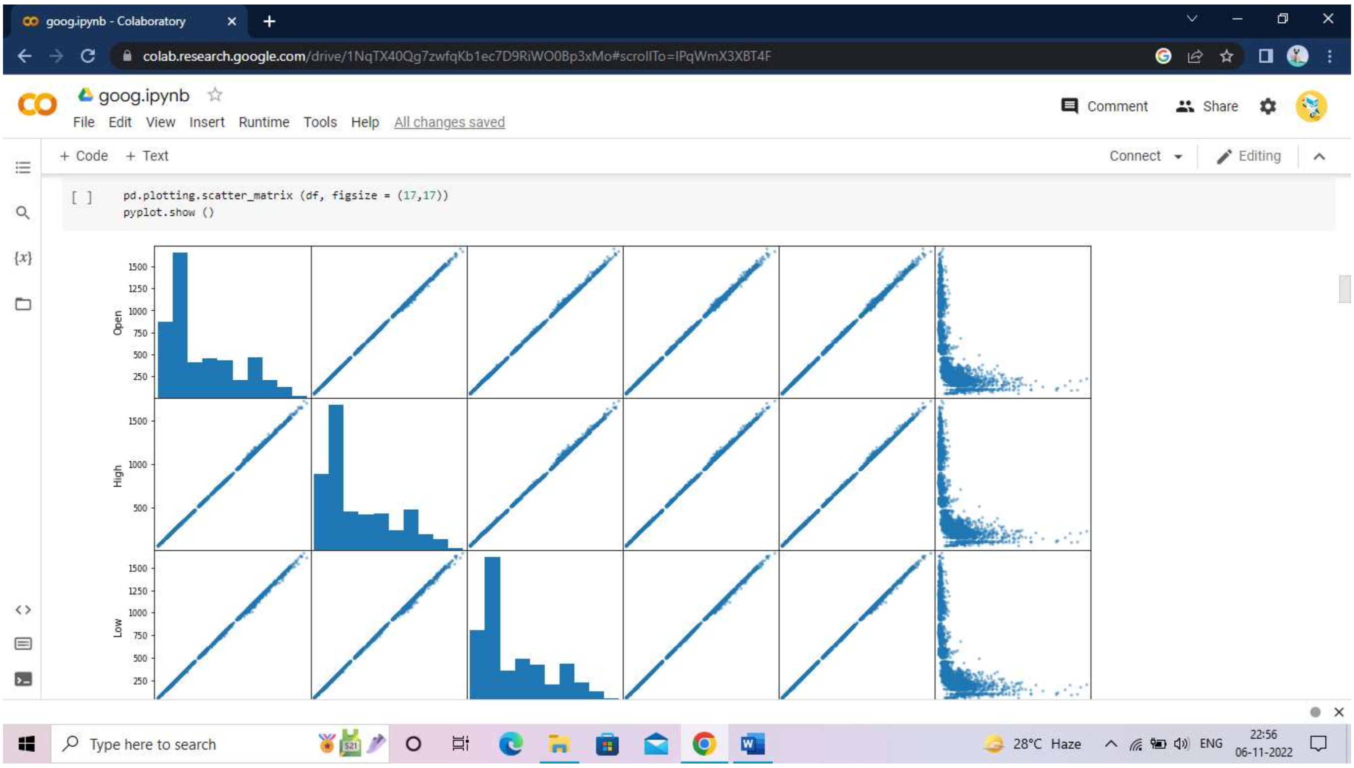
```
There are 4041 rows and 7 columns
```

```
print('description de la base')
print(df.describe())
```

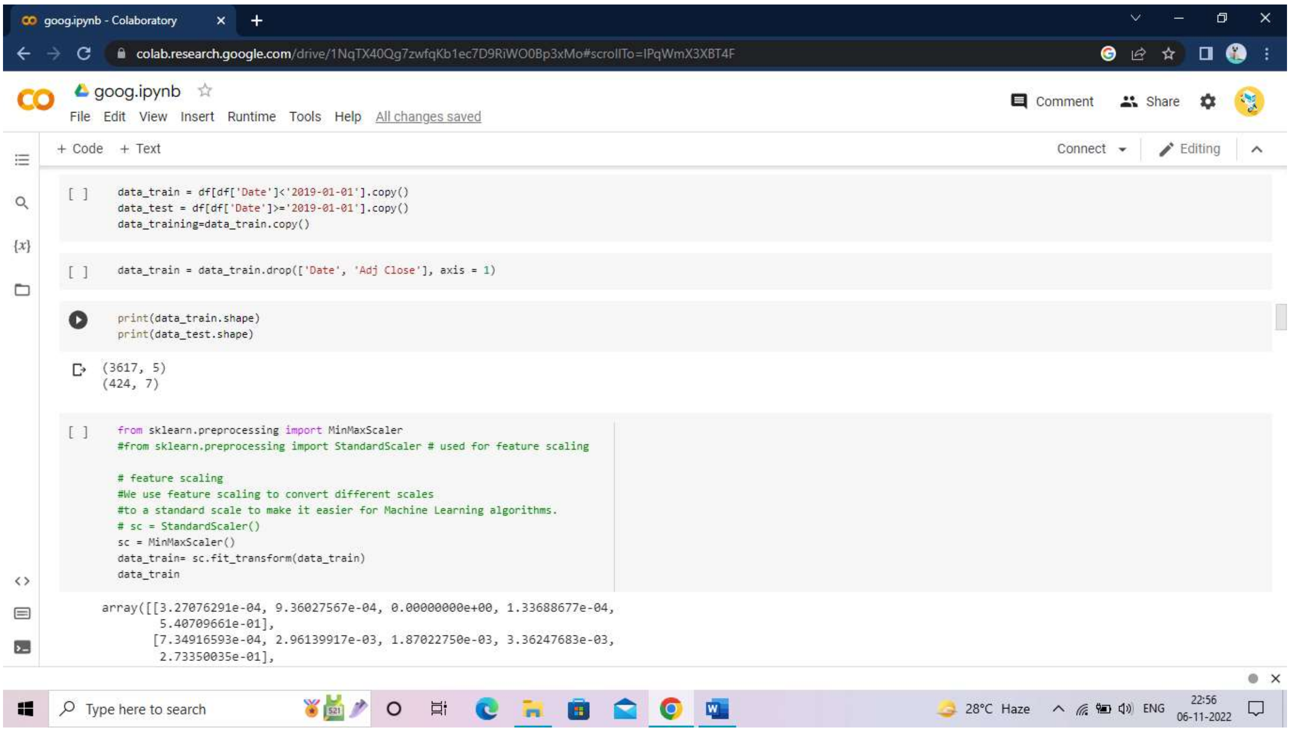
```
description de la base
              Open         High          Low        Close    Adj Close  \
count  4041.000000  4041.000000  4041.000000  4041.000000  4041.000000
mean    533.983149   538.995819   528.658860   533.999060   533.999060
std     383.007917   386.590237   379.488087   383.326004   383.326004
min      49.644646    50.920921    48.028027    50.055054    50.055054
25%     241.211212   243.688690   238.873871   241.036041   241.036041
50%     342.592590   345.795807   338.598602   342.177185   342.177185
75%     791.979980   798.000000   786.200012   790.460022   790.460022
max    1699.520020  1726.099976  1660.189941  1717.390015  1717.390015

             Volume
count  4.041000e+03
mean   6.909802e+06
std    7.895987e+06
min    5.206000e+05
25%    1.844600e+06
50%    4.191600e+06
75%    8.702600e+06
max    8.215110e+07
```



goog.ipynb - Colaboratory
colab.research.google.com/drive/1NqTX40Qg7zwfqKb1ec7D9RiWO0Bp3xMo#scrollTo=IPqWmX3XBT4F
goog.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Connect Editing
from matplotlib import pyplot
import matplotlib.pyplot as plt
df.hist (figsize = (17,17))
plt.show ()
Open
High
Low
Close
Type here to search
28°C Haze
ENG
22:55
06-11-2022



goog.ipynb - Colaboratory
colab.research.google.com/drive/1NqTX40Qg7zwfqKb1ec7D9RiWO0Bp3xMo#scrollTo=IPqWmX3XBT4F
goog.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Connect Editing
pd.plotting.scatter_matrix (df, figsize = (17,17))
pyplot.show ()
Open
High
Low
Type here to search
28°C Haze
ENG
22:56
06-11-2022



```
data_train = df[df['Date']<'2019-01-01'].copy()
data_test = df[df['Date']>='2019-01-01'].copy()
data_training=data_train.copy()
```

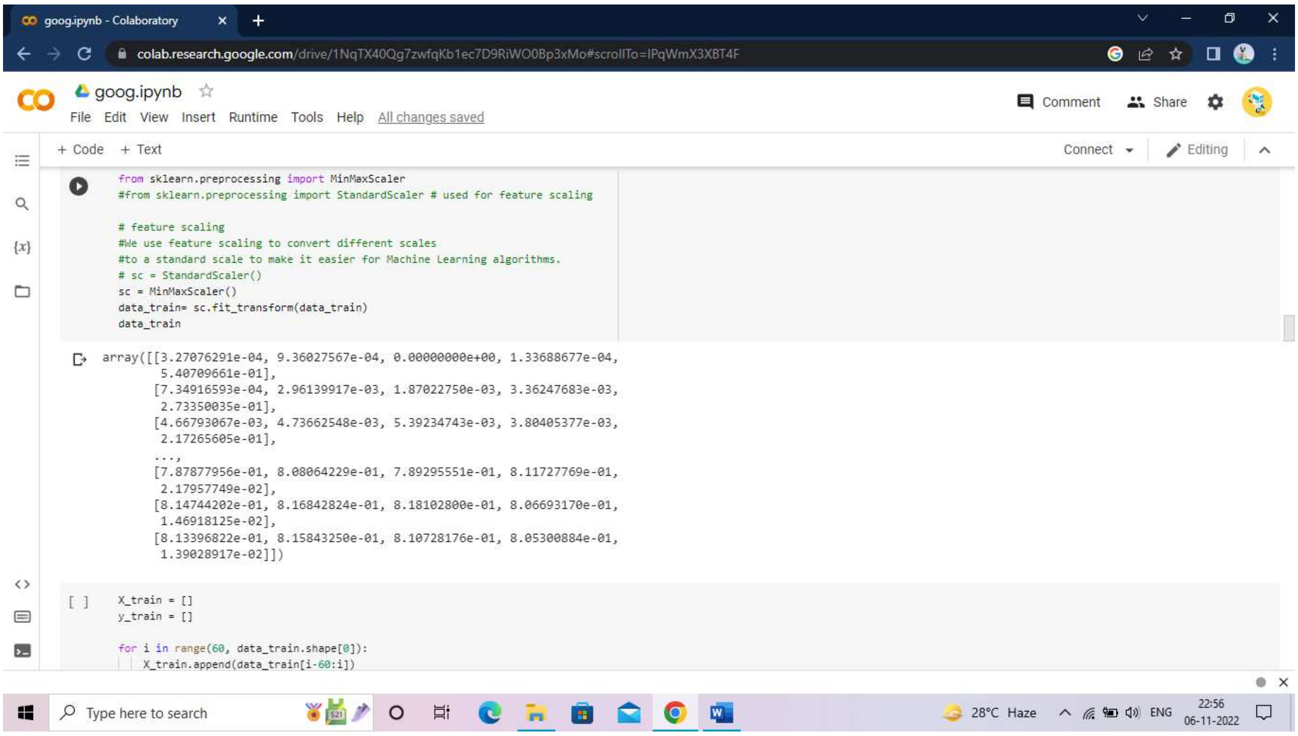
```
data_train = data_train.drop(['Date', 'Adj Close'], axis = 1)
```

```
print(data_train.shape)
print(data_test.shape)
```

```
(3617, 5)
(424, 7)
```

```
from sklearn.preprocessing import MinMaxScaler
#from sklearn.preprocessing import StandardScaler # used for feature scaling

# feature scaling
#We use feature scaling to convert different scales
#to a standard scale to make it easier for Machine Learning algorithms.
# sc = StandardScaler()
sc = MinMaxScaler()
data_train= sc.fit_transform(data_train)
data_train
```

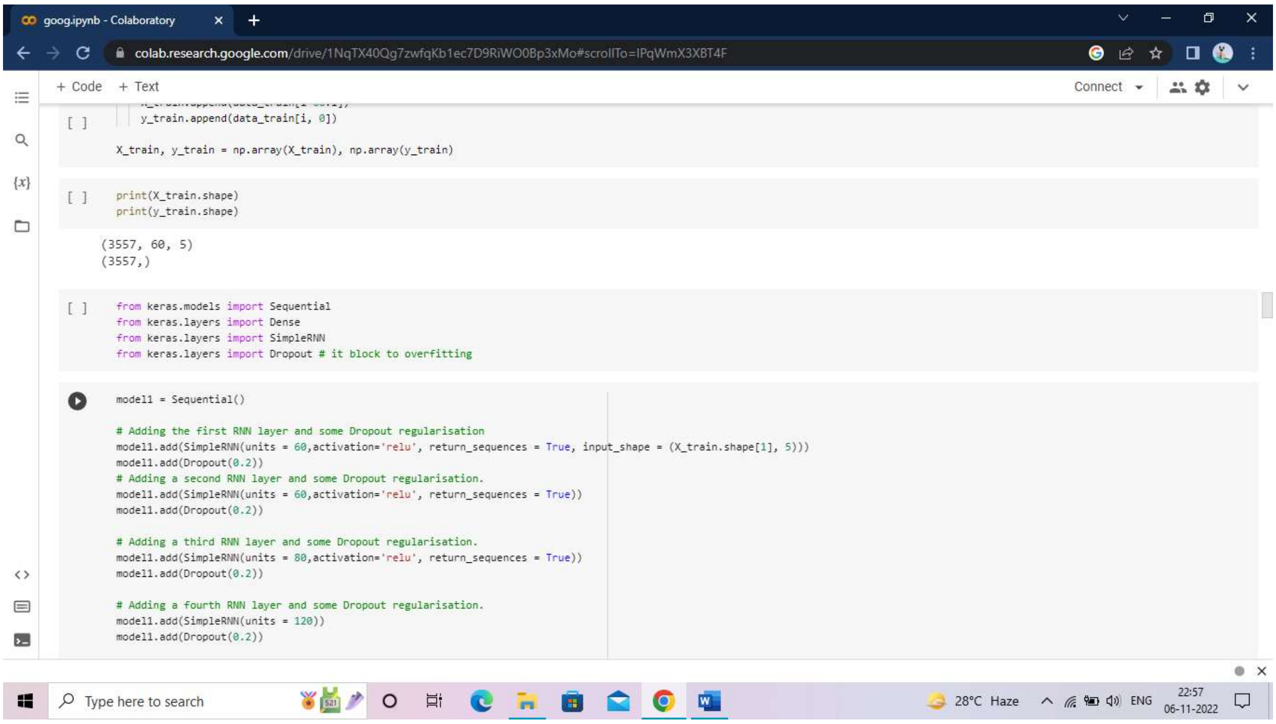
```
array([[3.27076291e-04, 9.36027567e-04, 0.00000000e+00, 1.33688677e-04,
        5.40709661e-01],
       [7.34916593e-04, 2.96139917e-03, 1.87022750e-03, 3.36247683e-03,
        2.73350035e-01],
```

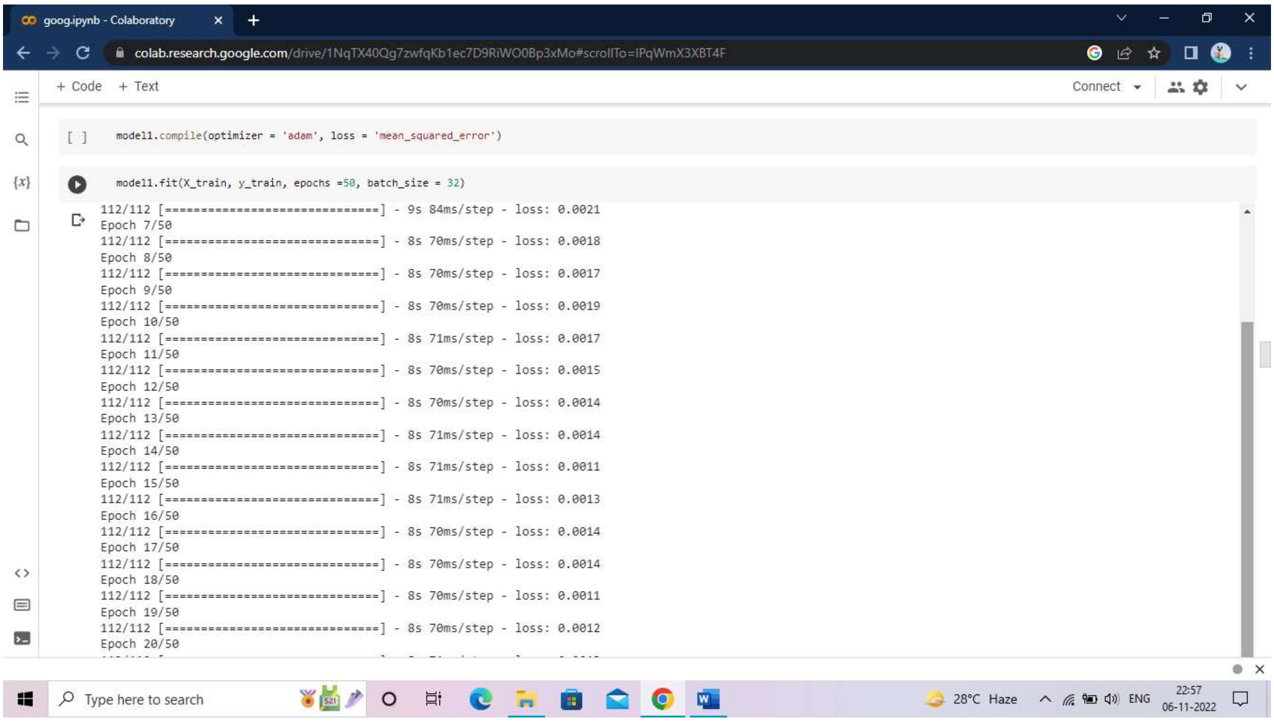


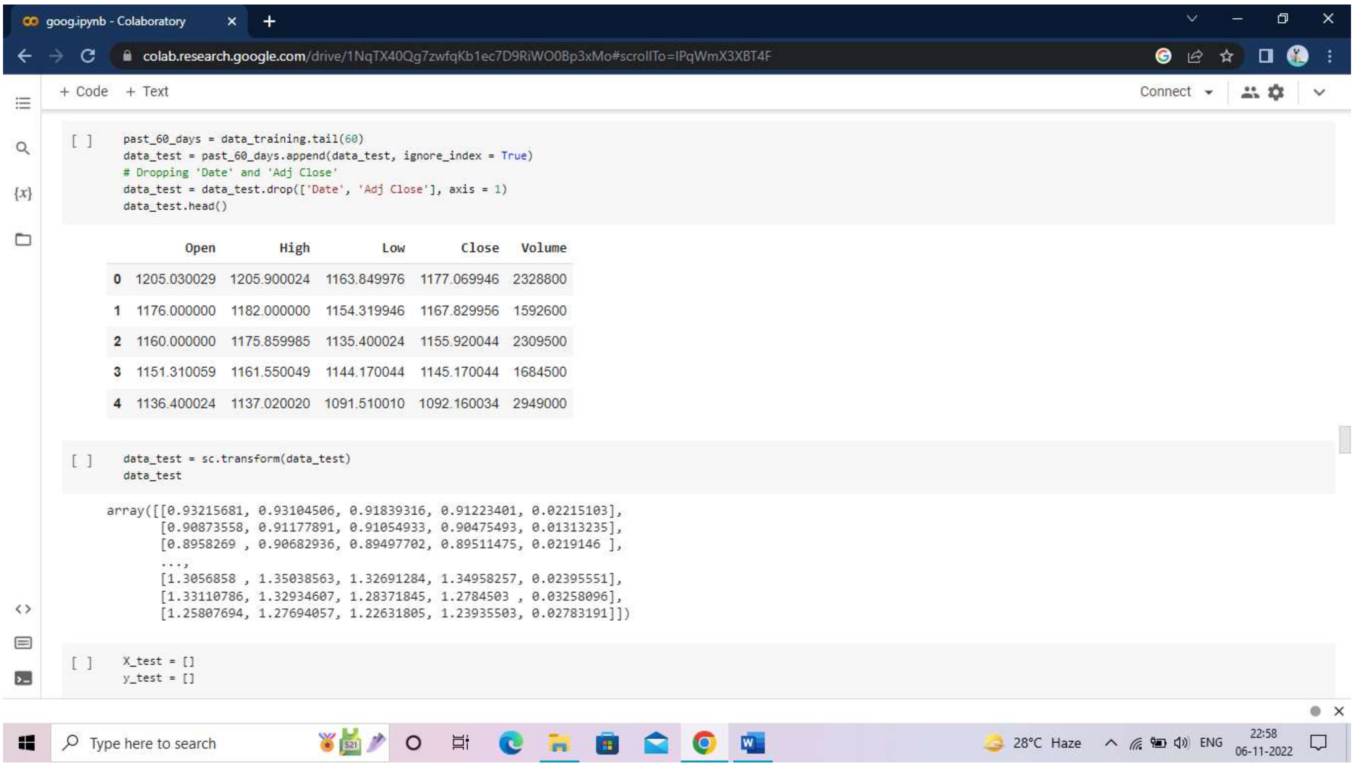
```
from sklearn.preprocessing import MinMaxScaler
#from sklearn.preprocessing import StandardScaler # used for feature scaling

# feature scaling
#We use feature scaling to convert different scales
#to a standard scale to make it easier for Machine Learning algorithms.
# sc = StandardScaler()
sc = MinMaxScaler()
data_train= sc.fit_transform(data_train)
data_train
```

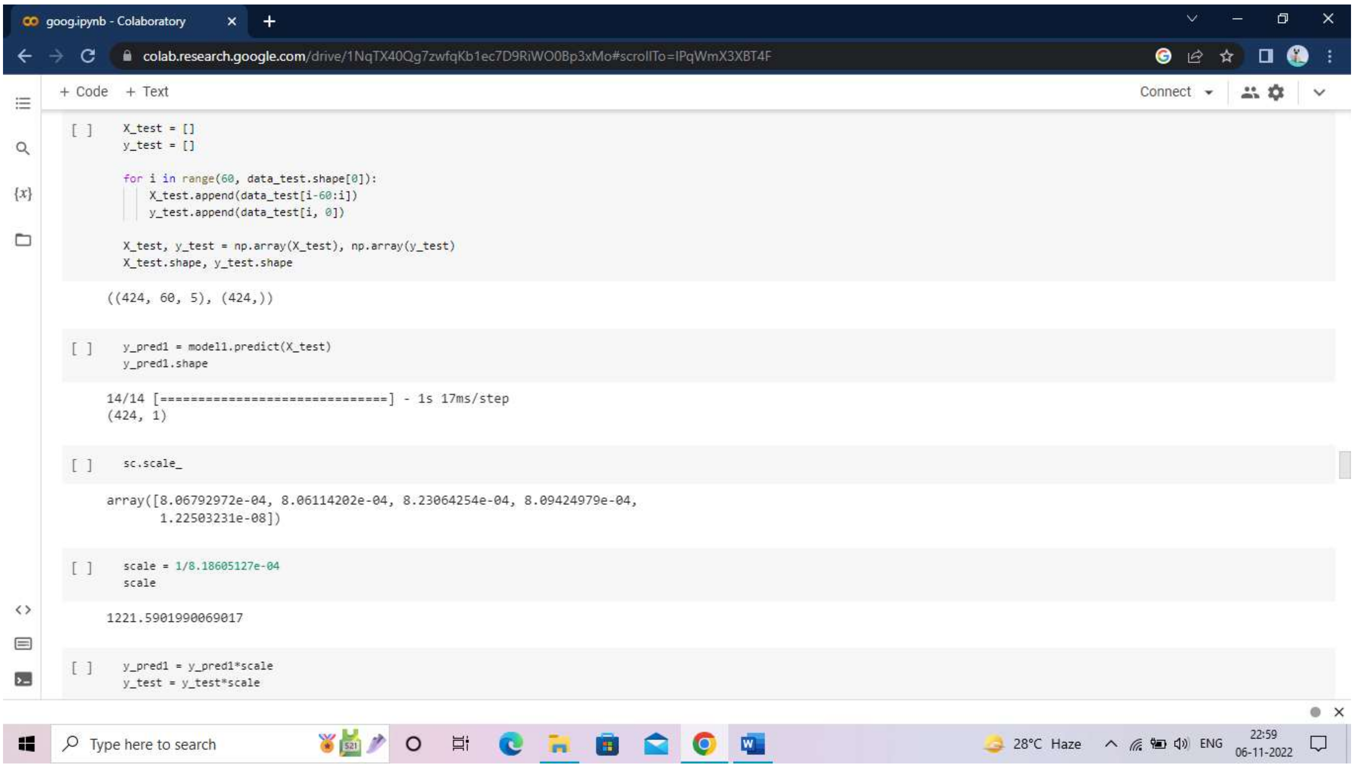
```
array([[3.27076291e-04, 9.36027567e-04, 0.00000000e+00, 1.33688677e-04,
        5.40709661e-01],
       [7.34916593e-04, 2.96139917e-03, 1.87022750e-03, 3.36247683e-03,
        2.73350035e-01],
       [4.66793067e-03, 4.73662548e-03, 5.39234743e-03, 3.80405377e-03,
        2.17265605e-01],
       ...,
       [7.87877956e-01, 8.08064229e-01, 7.89295551e-01, 8.11727769e-01,
        2.17957749e-02],
       [8.14744202e-01, 8.16842824e-01, 8.18102800e-01, 8.06693170e-01,
        1.46918125e-02],
       [8.13396822e-01, 8.15843250e-01, 8.10728176e-01, 8.05300884e-01,
        1.39028917e-02]])
```

```
X_train = []
y_train = []

for i in range(60, data_train.shape[0]):
    X_train.append(data_train[i-60:i])
```


goog.ipynb - Colaboratory
colab.research.google.com/drive/1NqTX40Qg7zwfqKb1ec7D9RiWO0Bp3xMo#scrollTo=lPqWmX3XBT4F
+ Code + Text
Connect
y_train.append(data_train[i, 0])
X_train, y_train = np.array(X_train), np.array(y_train)
print(X_train.shape)
print(y_train.shape)
(3557, 60, 5)
(3557,)
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import SimpleRNN
from keras.layers import Dropout # it block to overfitting
model1 = Sequential()
# Adding the first RNN layer and some Dropout regularisation
model1.add(SimpleRNN(units = 60,activation='relu', return_sequences = True, input_shape = (X_train.shape[1], 5)))
model1.add(Dropout(0.2))
# Adding a second RNN layer and some Dropout regularisation.
model1.add(SimpleRNN(units = 60,activation='relu', return_sequences = True))
model1.add(Dropout(0.2))
# Adding a third RNN layer and some Dropout regularisation.
model1.add(SimpleRNN(units = 80,activation='relu', return_sequences = True))
model1.add(Dropout(0.2))
# Adding a fourth RNN layer and some Dropout regularisation.
model1.add(SimpleRNN(units = 120))
model1.add(Dropout(0.2))
Type here to search
28°C Haze
ENG
22:57
06-11-2022

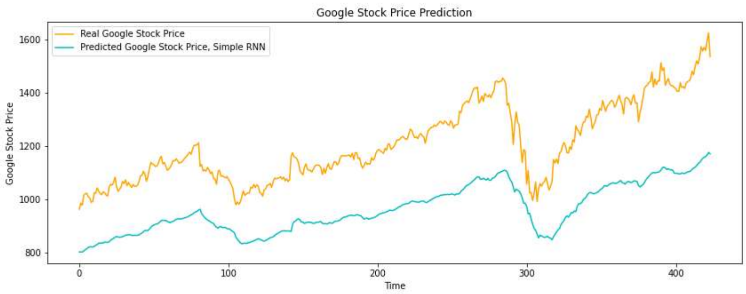

goog.ipynb - Colaboratory
colab.research.google.com/drive/1NqTX40Qg7zwfqKb1ec7D9RiWO0Bp3xMo#scrollTo=lPqWmX3XBT4F
+ Code + Text
Connect
model1.compile(optimizer = 'adam', loss = 'mean_squared_error')
model1.fit(X_train, y_train, epochs =50, batch_size = 32)
112/112 [==============================] - 9s 84ms/step - loss: 0.0021
Epoch 7/50
112/112 [==============================] - 8s 70ms/step - loss: 0.0018
Epoch 8/50
112/112 [==============================] - 8s 70ms/step - loss: 0.0017
Epoch 9/50
112/112 [==============================] - 8s 70ms/step - loss: 0.0019
Epoch 10/50
112/112 [==============================] - 8s 71ms/step - loss: 0.0017
Epoch 11/50
112/112 [==============================] - 8s 70ms/step - loss: 0.0015
Epoch 12/50
112/112 [==============================] - 8s 70ms/step - loss: 0.0014
Epoch 13/50
112/112 [==============================] - 8s 71ms/step - loss: 0.0014
Epoch 14/50
112/112 [==============================] - 8s 71ms/step - loss: 0.0011
Epoch 15/50
112/112 [==============================] - 8s 71ms/step - loss: 0.0013
Epoch 16/50
112/112 [==============================] - 8s 70ms/step - loss: 0.0014
Epoch 17/50
112/112 [==============================] - 8s 70ms/step - loss: 0.0014
Epoch 18/50
112/112 [==============================] - 8s 70ms/step - loss: 0.0011
Epoch 19/50
112/112 [==============================] - 8s 70ms/step - loss: 0.0012
Epoch 20/50
Type here to search
28°C Haze
ENG
22:57
06-11-2022


goog.ipynb - Colaboratory
colab.research.google.com/drive/1NqTX40Qg7zwfqKb1ec7D9RiWO0Bp3xMo#scrollTo=IPqWmX3XBT4F
+ Code + Text
Connect
past_60_days = data_training.tail(60)
data_test = past_60_days.append(data_test, ignore_index = True)
# Dropping 'Date' and 'Adj Close'
data_test = data_test.drop(['Date', 'Adj Close'], axis = 1)
data_test.head()
Open High Low Close Volume
0 1205.030029 1205.900024 1163.849976 1177.069946 2328800
1 1176.000000 1182.000000 1154.319946 1167.829956 1592600
2 1160.000000 1175.859985 1135.400024 1155.920044 2309500
3 1151.310059 1161.550049 1144.170044 1145.170044 1684500
4 1136.400024 1137.020020 1091.510010 1092.160034 2949000
data_test = sc.transform(data_test)
data_test
array([[0.93215681, 0.93104506, 0.91839316, 0.91223401, 0.02215103],
[0.90873558, 0.91177891, 0.91054933, 0.90475493, 0.01313235],
[0.8958269 , 0.90682936, 0.89497702, 0.89511475, 0.0219146],
...,
[1.3056858 , 1.35038563, 1.32691284, 1.34958257, 0.02395551],
[1.33110786, 1.32934607, 1.28371845, 1.2784503 , 0.03258096],
[1.25807694, 1.27694057, 1.22631805, 1.23935503, 0.02783191]])
X_test = []
y_test = []
Type here to search
28°C Haze
ENG
22:58
06-11-2022


goog.ipynb - Colaboratory
colab.research.google.com/drive/1NqTX40Qg7zwfqKb1ec7D9RiWO0Bp3xMo#scrollTo=IPqWmX3XBT4F
+ Code + Text
Connect
X_test = []
y_test = []
for i in range(60, data_test.shape[0]):
X_test.append(data_test[i-60:i])
y_test.append(data_test[i, 0])
X_test, y_test = np.array(X_test), np.array(y_test)
X_test.shape, y_test.shape
((424, 60, 5), (424,))
y_pred1 = model1.predict(X_test)
y_pred1.shape
14/14 [==============================] - 1s 17ms/step
(424, 1)
sc.scale_
array([8.06792972e-04, 8.06114202e-04, 8.23064254e-04, 8.09424979e-04,
1.22503231e-08])
scale = 1/8.18605127e-04
scale
1221.5901990069017
y_pred1 = y_pred1*scale
y_test = y_test*scale
Type here to search
28°C Haze
ENG
22:59
06-11-2022

```
y_pred1 = y_pred1*scale
y_test = y_test*scale
```

```
# Visualising the results
plt.figure(figsize=(14,5))
plt.plot(y_test, color = 'orange', label = 'Real Google Stock Price')
plt.plot(y_pred1, color = "c", label = 'Predicted Google Stock Price, Simple RNN')
plt.title('Google Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('Google Stock Price')
plt.legend()
plt.show()
```



LSTM

```
from tensorflow.keras.layers import LSTM
```

```
#Initializing the RNN
model2 = Sequential()

# Adding the first RNN layer and some Dropout regularisation
model2.add(LSTM(units = 60, activation = 'relu', return_sequences = True, input_shape = (X_train.shape[1], 5)))
model2.add(Dropout(0.2))
# Adding a second RNN layer and some Dropout regularisation.
model2.add(LSTM(units = 60, activation = 'relu', return_sequences = True))
model2.add(Dropout(0.2))
# Adding a third RNN layer and some Dropout regularisation.
model2.add(LSTM(units = 80, activation = 'relu', return_sequences = True))
model2.add(Dropout(0.2))
# Adding a fourth RNN layer and some Dropout regularisation.
model2.add(LSTM(units = 120, activation = 'relu'))
model2.add(Dropout(0.2))
# Adding the output layer
model2.add(Dense(units = 1))
```

```
model2.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 60, 60)            15840

 dropout_4 (Dropout)         (None, 60, 60)            0
```

```
model2.compile(optimizer = 'adam', loss = 'mean_squared_error')
```

```
model2.fit(X_train, y_train, epochs = 50, batch_size = 32)
```

```
Epoch 1/50
112/112 [==============================] - 26s 189ms/step - loss: 0.0130
Epoch 2/50
112/112 [==============================] - 21s 189ms/step - loss: 0.0020
Epoch 3/50
112/112 [==============================] - 21s 188ms/step - loss: 0.0020
Epoch 4/50
112/112 [==============================] - 21s 189ms/step - loss: 0.0018
Epoch 5/50
112/112 [==============================] - 23s 205ms/step - loss: 0.0017
Epoch 6/50
112/112 [==============================] - 21s 188ms/step - loss: 0.0015
Epoch 7/50
112/112 [==============================] - 21s 189ms/step - loss: 0.0016
Epoch 8/50
112/112 [==============================] - 21s 189ms/step - loss: 0.0016
Epoch 9/50
112/112 [==============================] - 21s 187ms/step - loss: 0.0015
Epoch 10/50
112/112 [==============================] - 21s 189ms/step - loss: 0.0015
Epoch 11/50
112/112 [==============================] - 21s 189ms/step - loss: 0.0014
Epoch 12/50
112/112 [==============================] - 21s 188ms/step - loss: 0.0013
Epoch 13/50
112/112 [==============================] - 23s 205ms/step - loss: 0.0011
Epoch 14/50
112/112 [==============================] - 21s 188ms/step - loss: 0.0012
```

```
X_test = []
y_test = []

for i in range(60, data_test.shape[0]):
    X_test.append(data_test[i-60:i])
    y_test.append(data_test[i, 0])

X_test, y_test = np.array(X_test), np.array(y_test)
X_test.shape, y_test.shape
```

```
((424, 60, 5), (424,))
```

```
#predictions
y_pred2 = model2.predict(X_test)
y_pred2.shape
```

```
14/14 [==============================] - 1s 56ms/step
(424, 1)
```

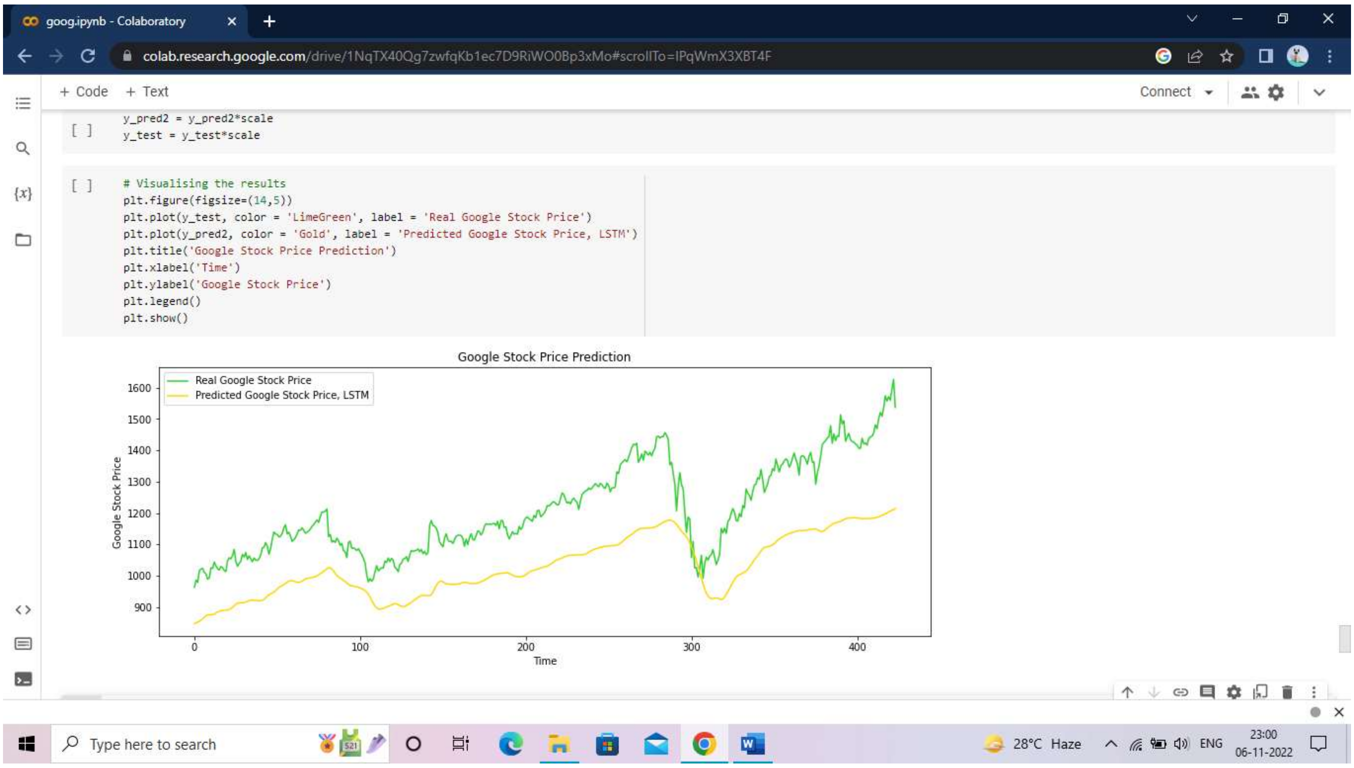
```
sc.scale_
```

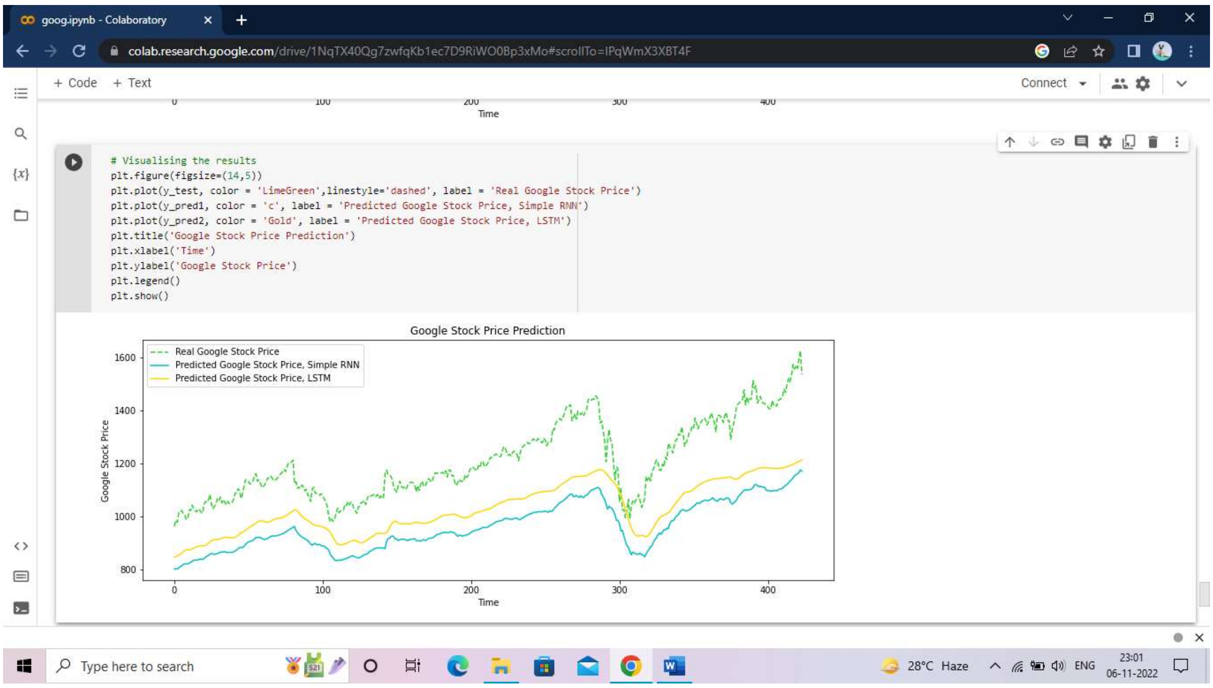
```
array([8.06792972e-04, 8.06114202e-04, 8.23064254e-04, 8.09424979e-04,
       1.22503231e-08])
```

```
scale = 1/8.18605127e-04
scale
```

```
1221.5901990069017
```

```
y_pred2 = y_pred2*scale
```


goog.ipynb - Colaboratory
colab.research.google.com/drive/1NqTX40Qg7zwfqKb1ec7D9RiWO0Bp3xMo#scrollTo=IPqWmX3XBT4F
+ Code + Text
Connect
y_pred2 = y_pred2*scale
y_test = y_test*scale
# Visualising the results
plt.figure(figsize=(14,5))
plt.plot(y_test, color = 'LimeGreen', label = 'Real Google Stock Price')
plt.plot(y_pred2, color = 'Gold', label = 'Predicted Google Stock Price, LSTM')
plt.title('Google Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('Google Stock Price')
plt.legend()
plt.show()
Google Stock Price Prediction
Real Google Stock Price
Predicted Google Stock Price, LSTM
Google Stock Price
Time
Type here to search
28°C Haze
ENG
23:00
06-11-2022


goog.ipynb - Colaboratory
colab.research.google.com/drive/1NqTX40Qg7zwfqKb1ec7D9RiWO0Bp3xMo#scrollTo=IPqWmX3XBT4F
+ Code + Text
Connect
# Visualising the results
plt.figure(figsize=(14,5))
plt.plot(y_test, color = 'LimeGreen',linestyle='dashed', label = 'Real Google Stock Price')
plt.plot(y_pred1, color = 'c', label = 'Predicted Google Stock Price, Simple RNN')
plt.plot(y_pred2, color = 'Gold', label = 'Predicted Google Stock Price, LSTM')
plt.title('Google Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('Google Stock Price')
plt.legend()
plt.show()
Google Stock Price Prediction
Real Google Stock Price
Predicted Google Stock Price, Simple RNN
Predicted Google Stock Price, LSTM
Google Stock Price
Time
Type here to search
28°C Haze
ENG
23:01
06-11-2022

5.2. DATA VISUALIZATION

Data visualization is the representation of data through use of common graphics, such as charts, plots, info graphics, and even animations. These visual displays of information communicate complex data relationships and data-driven insights in a way that is easy to understand.

Data visualization can be utilized for a variety of purposes, and it's important to notethat is not only reserved for use by data teams. Management also leverages it to convey organizational structure and hierarchy while data analysts and data scientistsuse it to discover and explain patterns and trends. Harvard Business Review (link resides outside IBM) categorizes data visualization into four key purposes: idea generation, idea illustration, visual discovery, and everyday dataviz.

TYPES OF DATA VISUALIZATION

- **Tables**: This consists of rows and columns used to compare variables. Tablescan show a great deal of information in a structured way, but they can also overwhelm users that are simply looking for high-level trends.
- **Pie charts and stacked bar charts:** These graphs are divided into sections that represent parts of a whole. They provide a simple way to organize data and compare the size of each component to one other.
- **Line charts and area charts:** These visuals show change in one or more quantities by plotting a series of data points over time and are frequently usedwithin predictive analytics. Line graphs utilize lines to demonstrate these changes while area charts connect data points with line segments, stacking variables on top of one another and using color to distinguish betweenvariables.
- **Scatter plots**: These visuals are beneficial in reveling the relationship between two variables, and they are commonly used within regression data analysis. However, these can sometimes be confused with bubble charts, which are used to visualize three variables via the x-axis, the y-axis, and the size of the bubble.
- **Heat maps:** These graphical representation displays are helpful in visualizingbehavioral data by location. This can be a location on a map, or even a webpage.

6. CONCLUSION AND FUTURE ENHANCEMENT

6.1. CONCLUSION

the most recent market research and Stock Market Prediction advancements have begun to include such approaches in analyzing stock market data. The Opening Value of the stock, the Highest and Lowest values of that stock on the same days, as well as the Closing Value at the end of the day, are all indicated for each date. Furthermore, the total volume of the stocks in the market is provided, With this information, it is up to the job of a Machine LearningData Scientist to look at the data and develop different algorithms that may help in finding appropriate stocks values.

Predicting the stock market was a time-consuming and laborious procedure a few years or even a decade ago. However, with the application of machine learning for stock market forecasts, the procedure has become much simpler. Machine learning not only saves time and resources but also outperforms people in terms of performance. it will always prefer to use a trained computer algorithm since it will advise you based only on facts, numbers, and data and will not factor in emotions or prejudice.

6.2. FUTURE ENHANCEMENT

For our future work we will try to find the best sets for bout data length and number of training epochs that beater suit our assets and maximize our predictions accuracy. Some future works can be studied. One can apply after-market stock data to inputs. After-market stock data is difficult to get but applying the data can significantly enlarge training sample size and reduce noise. More, we only test LSTM model in our thesis, other Recurrent Neural Network structures can also be applied to stock market prediction such as, Echo State Network, Neural Turing Machines and Continuous-time RNN.

7. REFERENCES

7.1. BOOK REFERENCES

- Sreelekshmy Selvin, Vinayakumar R, Gopalakrishnan E.A, Vijay Krishna Menon, Soman K.P, "Stock Price Prediction Using LSTM, RNN And CNN-SLIDING WINDOW MODEL", 2017
- Vinod Mehta at el. , "Stock Price Prediction Using Regression And Aritificial Neural Network", 2017.
- Ryo Akita, Akira Yoshihara, Takashi Matsubara, Kuniaki Uehara, "Deep learning for stock prediction using numerical and textual information", 2016.
- Bhagyashree Nigade at el., "Comparative Study of Stock Prediction System using Regression Techniques", March - April 2017.
- S Abdul Salam Suleiman Polanyi, Adele, Kayode S., Jimoh, R. G, "Stock Trend Prediction Using Regression Analysis – A Data Mining Approach", July 2011.
- Amit B. Suthar, Ms. Hiral R. Patel, Dr. Satyen M. Parikh, "A Comparative Study on Financial Stock Market Prediction Models", 2012.
- S.Prasanna, Dr.D.Ezhilmaran, "An analysis on Stock Market Prediction using Data Mining Techniques", 2013.
- 12. Ruchi Desai, Prof.Snehal Gandhi, "Stock Market Prediction Using Data Mining", 2014.
- G. S. Navale, Nishant Dudhwala, Kunal Jadhav, "Prediction of Stock Market using Data Mining and Artificial Intelligence", 2016.
- Bhagyashree Nigade, Aishwarya Pawar at el., "Stock Trend Prediction Using Regression Analysis – A Data Mining Approach", February 2017.
- Shalini Lotlikar at el, "Stock Prediction Using Clustering And Regression Techniques", May 2017 17. Mr. Pramod Mali, Hemangi Karchalkar at el., "Open Price Prediction of Stock Market using Regression Analysis", May 2017.

7.2. WEB REFERENCES

- https://aditi-mittal.medium.com/understanding-rnn-and-lstm-f7cdf6dfc14e
- https://colah.github.io/posts/2015-08-Understanding-LSTMs/
- https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/
- https://medium.com/analytics-vidhya/share-price-prediction-using-rnn-and-lstm-8776456dea6f
- https://www.jetir.org/papers/JETIRK006164.pdf
- https://www.datacamp.com/tutorial/lstm-python-stock-market
- https://www.analyticsvidhya.com/blog/2021/12/stock-price-prediction-using-lstm/