**Stock Price & Sentiment Analysis using AI/ML/DL**

A Project Work-II Report

Submitted in partial fulfillment of requirement of the

Degree of

**BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE & ENGINEERING**

BY

**Akshat Gangrade EN18CS301017**

Under the Guidance of

**Mr. Ashish Kumawat**



**Department of Computer Science & Engineering**

**Faculty of Engineering**

**MEDI-CAPS UNIVERSITY, INDORE- 453331**

**Jan-Jun 2022**

# Report Approval

The project work **“Stock price, Sentiment analysis using AI/ML/DL”** is hereby approved as a creditable study of an engineering/computer application subject carried out and presented in a manner satisfactory to warrant its acceptance as prerequisite for the Degree for which it has been submitted.

It is to be understood that by this approval the undersigned do not endorse or approved any statement made, opinion expressed, or conclusion drawn there in; but approve the “Project Report” only for the purpose for which it has been submitted.

Internal Examiner Name:

Designation

Affiliation

External Examiner Name:

Designation

Affiliation

# Declaration

I hereby declare that the project entitled **“Stock price, Sentiment Analysis using AI/ML/DL”** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in ‘Computer Science & Engineering’ completed under the supervision of Mr. Ashish Kumawat of  **Computer Science & Engineering,** Faculty of Engineering, Medi-Caps University Indore is an authentic work.

Further, I/we declare that the content of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for the award of any degree or diploma.

## 

## Akshat Gangrade (EN18CS301017)

Date: 10 May 2022



**Certificate**

I, **Mr. Ashish Kumawat** certify that the project entitled **“Stock Price, Sentiment Analysis using AI/ML/DL”** submittedin partial fulfillment for the award of the degree of Bachelor of Technology by **Akshat Gangrade** istherecordcarried out by them under my guidance and that the work has not formed the basis of award of any other degree elsewhere.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr. Vimal Bhatia

Assistant Professor

Department of Electrical Engineering

Indian Institute of Technology, Indore

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Mr. Ashish Kumawat

Computer Science & Engineering

Medi-caps University,Indore

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

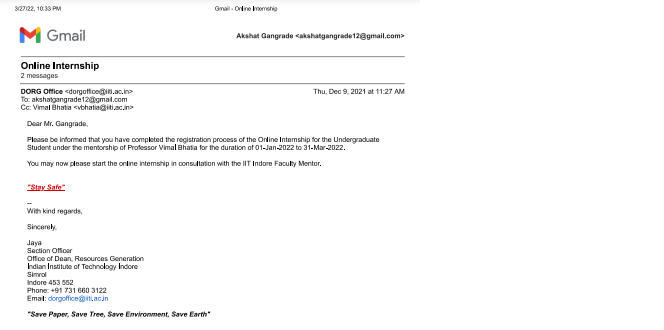
Dr. Pramod S. Nair

Head of the Department

Computer Science & Engineering

Medi-caps University, Indore

**Offer Letter of Internship**

****

**Completion Certificate**

****

# Acknowledgements

I would like to express my deepest gratitude to Honorable Chancellor, **Shri R C Mittal,**who has provided me with every facility to successfully carry out this project, and my profound indebtedness to **Prof. (Dr.) Dilip K Patnaik,** Vice Chancellor, Medi-Caps University, whose unfailing support and enthusiasm has always boosted up my morale. I also thank **Prof. (Dr.) D K Panda,** Pro-Vice Chancellor, Medi-Caps University, **Dr. Suresh K Jain,** Dean, Faculty of Engineering, Medi-Caps University, for giving me a chance to work on this project. I would also like to thank my Head of the Department **Dr. Pramod S. Nair**for his/her continuous encouragement for betterment of the project.

I express my heartfelt gratitude to my Internal Guide, Mr. Ashish Kumawat Department of Computer Science & Engineering, MU, without whose continuous help and support, this project would ever have reached to the completion.

It is their help and support, due to which we became able to complete the design and technical report.

Without their support this report would not have been possible.

## Akshat Gangrade

B.Tech. IV Year

Department of Computer Science & Engineering

Faculty of Engineering

Medi-Caps University, Indore

**Abstract**

Artificial intelligence, Machine learning and Deep learning are the modern days technologies which is creating a revolution in the industry and by leveraging these technologies we are able to solve many problem and during the coarse of the internship. I have learned various tools and library involved and also worked on few test cases.

In the first experiment we make uses Deep Learning techniques to analyze and predict stock price. We have used the data of a stock (NSE TATAGLOBAL) and the various parameters which includes high, low, close, total traded quantity for a day are being used as an input. This data has been used for training deep neural networks (LSTM) for making predictions. All the experiments have been performed in Python Programming Language. Several architectures have been trained and amongst them, some have performed well.. Many other approaches have also been discussed further in the report.

Initial experiments show that LSTM is able to perform well for making predictions and the evaluation has been done on the basis of Root Mean Squared Error (RMSE). All the experiments have obtained RMSE in the range of 76-100.

The second experiment involves sentiment analysis of top news headlines to predict whether the stock price will rise or fall for which various deep learning library is being used for the data wrangling and after cleaning up of data the model is trained using random classifier, countVectorizer methods

**Keywords:**

* Python
* GoogleColab
* Pytorch
* Neural Network
* Deep Learning

**Table Of Contents**

|  |  |  |
| --- | --- | --- |
|  |  | Page No. |
|  | Report Approval | ii |
|  | Declaration | iii |
|  | Certificate | iv |
|  | Offer Letter of the internship | v |
|  | Completion Letter | vi |
|  | Acknowledgement | vii |
|  | Abstract | viii |
|  | Table of Contents | ix |
|  | List of figures | x |
|  | Abbreviations | xii |
|  |  |  |
| Chapter-1 | 1. Introduction | 11-16 |
|  | 1.1 Machine Learning | 12 |
|  | 1.2 Deep Learning | 14 |
|  | 1.3 Data Preprocessing | 15 |
|  | 1.4 Problem Statement | 16 |
|  |  |  |
| Chapter-2 | 2 Libraries and tools used | 16-26 |
|  | 2.1 Introduction | 16 |
|  | 2.2 Python for AI/ML | 17 |
|  | 2.3 Data-Wrangling Tools | 21 |
|  | 2.4 Data Visualization Libraries | 22 |
|  | 2.5 Tensor Flow for deep learning | 26 |
|  |  |  |
| Chapter-3 | 3. Implementation | 27- |
|  | 3.1 Introduction | 27 |
|  | 3.2 Learning Algorithms Used | 28 |
|  | 3.3 Problem-1 | 33 |
|  | 3.4 Probleum-2 | 36 |
|  | 3.5 Probleum-3 | 39 |
|  |  |  |
| Chapter-4 | Limitation | 41 |
| Chapter-5 | Future Scope | 42 |
| Chapter-6 | Conclusion | 43 |
| Chapter-7 | Bibliography and references | 44 |

**List Of Figures**

|  |  |  |
| --- | --- | --- |
| **Serial No.** | **Figure Name** | **Page No.** |
| 1.1 | How Machine Learning works? | 12 |
| 1.2 | Supervised Learning | 14 |
| 1.3 | Artificial Neural Network | 15 |
| 1.4 | Data Preprocessing | 16 |
| 2.1 | Python Libraries | 18 |
| 2.2 | Matplotlib | 22 |
| 2.3 | Plotly | 23 |
| 2.4 | Seaborn | 24 |
| 2.5 | Altair | 25 |
| 2.6 | TensorFlow working | 26 |
| 3.1 | Gradient Descent | 28 |
| 3.2 | Newton’s Method | 30 |
| 3.3 | Performance Comparision | 33 |
| 3.4 | Library Import | 34 |
| 3.5 | Train\_test\_split | 34 |
| 3.6 | Choosing Model | 35 |
| 3.7 | Training Model | 35 |
| 3.8 | Output Prediction | 36 |

**Abbreviations**

* **CNN:** Convolution Neural Network
* **AI:** Artificial Intelligence
* **ML:** Machine Learning
* **GNN:** Graph Neural Network
* **GCN:** Graph Convolution Network
* **GPU:** Graphical Processing Unit
* **MLP:** Multilayer Perceptron
* **MNIST:** Modified National Institute Of Standards And Technology
* **reLU:** Rectified Linear Activation Unit
* **MSV:** Mean Square Value
* **RMSE:** Root Mean Square Value
* **LSTM:** Long-short-term-memory

**Chapter-1**

**Introduction**

* 1. **Machine Learning**

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

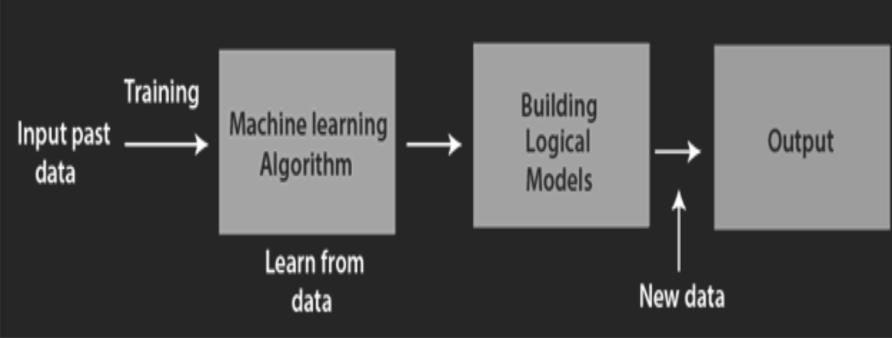
Machine learning is an important component of the growing field of data science. Through the use of statistical methods, algorithms are trained to make classifications or predictions, uncovering key insights within data mining projects. These insights subsequently drive decision making within applications and businesses, ideally impacting key growth metrics.

**How machine learning works:**

1. **A Decision Process:** In general, machine learning algorithms are used to make a prediction or classification. Based on some input data, which can be labeled or unlabeled, your algorithm will produce an estimate of a pattern in the data.

2. **An Error Function:** An error function serves to evaluate the prediction of the model. If there are known examples, an error function can make a comparison to assess the accuracy of the model.

3. **A Model Optimization Process:** If the model can fit better to the data points in the training set, then weights are adjusted to reduce the discrepancy between the known example and the model estimate. The algorithm will repeat this evaluation and optimize the process, updating weights autonomously until a threshold of accuracy has been met.



**Fig 1.1:** How machine learning works

**Real-world machine learning use cases:**

1. **Speech recognition**: It is also known as automatic speech recognition (ASR), computer speech recognition, or speech-to-text, and it is a capability that uses natural language processing (NLP) to process human speech into a written format. For example, Google Assistant and Apple Siri.

2. **Customer service**: Online chatbots are replacing human agents along the customer journey. They answer frequently asked questions (FAQs) around topics, like shipping, or provide personalized advice, cross-selling products or suggesting sizes for users, changing the way we think about customer engagement across websites and social media platforms. For example, the Zomato virtual assistant, Zia, on the Zomato food delivery service's mobile application assists you in tracking your food order and addresses any issues with food quality, quantity or delivery related issues.

3. **Automated stock trading**: Designed to optimize stock portfolios, AI-driven high-frequency trading platforms make thousands or even millions of trades per day without human intervention.

4. **Computer vision**: This AI technology enables computers and systems to derive meaningful information from digital images, videos and other visual inputs, and based on those inputs, it can take action. This ability to provide recommendations distinguishes it from image recognition tasks. Powered by convolutional neural networks, computer vision has applications in photo tagging in social media, radiology imaging in healthcare, and self-driving cars within the automotive industry.

**Types of Machine Learning Methods**

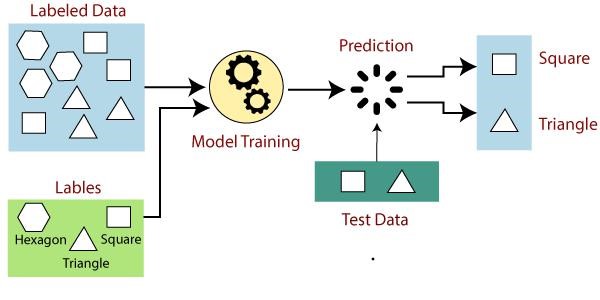
Many machine learning models are defined by the presence or absence of human influence on raw data — whether a reward is offered, specific feedback is given or labels are used. Following are the major machine learning methods:

1. **Supervisedlearning**: The dataset being used has been pre-labelled and classified by users to allow the algorithm to see how accurate its performance is.

2. **Unsupervisedlearning**: The raw dataset being used is unlabeled and an algorithm identifies patterns and relationships within the data without help from users.

3. **Semi**-**supervisedlearning**: The dataset contains structured and unstructured data, which guide the algorithm on its way to making independent conclusions. The combination of the two data types in one training dataset allows machine learning algorithms to learn to label unlabeled data.

4. **Reinforcementlearning**: The dataset uses a “rewards/punishments” system, offering feedback to the algorithm to learn from its own experiences by trial and error.



**Fig 1.2:** Supervised Learning

* 1. **Deep Learning:**

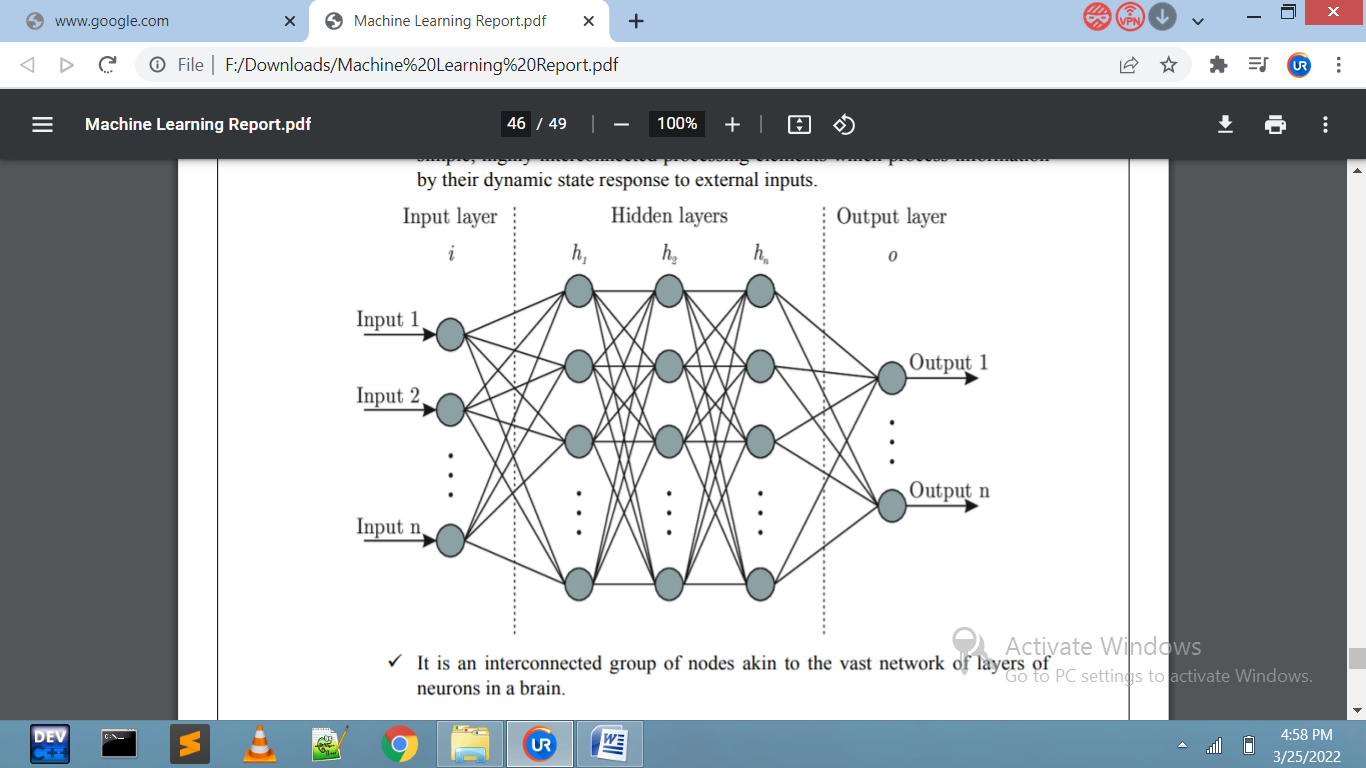
Deep Learning is a specialized form of Machine Learning that uses supervised, unsupervised, or semi-supervised learning to learn data representations.

Why deep learning?

1. The vast availability of Big Data enables machines to be trained.
2. Experts have discovered multi-layered learning networks that can be leveraged for deep learning as they learn in layers.
3. Scientists have figured out that high-performing graphics processing units (GPU) can be used for deep learning.

Artificial Neural Networks:-

1. Deep learning relies on multiple layers of training.
2. Artificial Neural Network is a computing system made up of a number of simple, highly interconnected processing elements which process information by their dynamic state response to external inputs
3. It is an interconnected group of nodes akin to the vast network of layers of neurons in a brain.

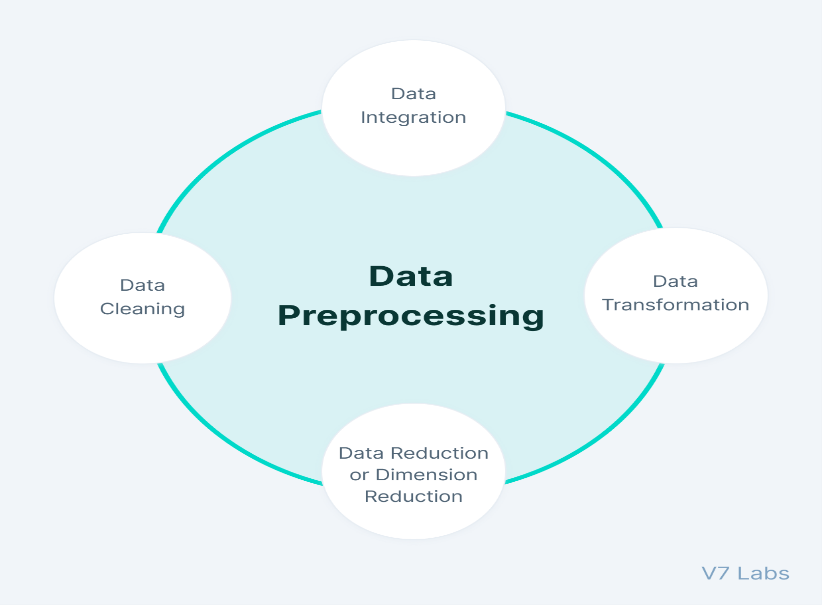


**Fig 1.3:**Artificial neural network

**1.3 Data pre-processing:**

1. Data Preparation:-Machine Learning depends largely on test data. Data preparation involves data selection, filtering, transformation, etc. Data preparation is a crucial step to make it suitable for ML. A large amount of data is generally required for the most common forms of ML.
2. Feature Engineering:-The transformation stage in the data preparation process includes an important step known as Feature Engineering. Feature Engineering refers to selecting and extracting right features from the data that are relevant to the task and model in consideration.
3. Feature Scaling :- Feature scaling is an important step in the data transformation stage of data preparation process. Feature Scaling is a method used in Machine Learning for standardization of independent variables of data features.
4. Datasets:- Machine Learning problems often need training or testing datasets. A dataset is a large repository of structured data. In many cases, it has input and output labels that assist in Supervised Learning.
5. Dimensionality Reduction with Principal Component Analysis:-Dimensionality reduction involves transformation of data to new dimensions in a way that facilitates discarding of some dimensions without losing any key information.

Principal component analysis (PCA) is a technique for dimensionality reduction that helps in arriving at better visualization models.



**Fig 1.4:** Data Pre-Processing

### 1.4 ProblemStatement

1. Designing a neural network model for given set of data and the learning algorithm to be used is Levenberg-Marquardt
2. Designing a machine learning model for the stock price prediction using the previous data of that stock.
3. Sentimental analysis of Top headlines to predict whether the price of stock will rise or fall.

The objective of the above problem statement is to come up with the most effective solution using the Ai / Ml or deep learning technique.

**Chapter-2**

**Library and Tools Used**

**2.1 Introduction**

To leverage the power of the technologies we can make use of the already available tools to perform our tasks. Which Includes python. Python is the language which is widely used because it is easy to use and also has huge set of library support, to work with dataset .

For working with data-sets there are various library available like numpy, pandas etc which makes it easier to work with dataset and also the library can further be used for viewing and further processing of data i.e data wrangling(Cleaning).

As we are very much comfortable in viewing a data in some form of graph, pictorial or some form of graphical representations for which we have various data visualization library which we can make use of to visualize the data in graphical Format For Example MatplotLib,

Now for training or building model we have a deep-learning library with which we can build artificial neural network models .

**2.2 Python for AI/ML:-**

While AI and ML are reshaping the world with its wonderful contributions to the technological world, it is important for developers and programmers to use the right programming language to make the most of AI and ML.

Python offers some great reasons to pick it as the programming language for AI and ML and also why businesses prefer to hire a python developer  
for their project

**Reasons for choosing python over other language:-**

1. A Great Library Ecosystem:-Python is the most in-demand programming language used for AI as it offers a significant choice in libraries.

A library is a module or a group of modules that are published by various sources such as PyPi which comes with a pre-written portion of code that helps users to attain some functionality and carry out different actions.

Python libraries offer base-level items. This saves developers’ time as they do not have to code them from the start every time.

ML needs regular data processing and Python For Machine Learning libraries allow developers to access, manage, and transform data.

*Here are some common libraries that you can use for AI and ML:*

* **Scikit-learn** is used to manage fundamental ML algorithms such as clustering, linear and logistic regressions, classification, and regression among others.
* **Pandas** is used for high-level data structures and analysis. With it, developers can merge and filter data and also collect data from other external sources like Excel.
* **Keras** is used for deep learning. As it accesses the GPU in addition to the CPU of the computer, it allows you to make fast calculations and prototyping.
* **TensorFlow** is used to work with deep learning by arranging, instructing, and utilising artificial neural networks with enormous datasets.
* **Matplotib** is used to develop 2D plots, charts, histograms, and other formats of data visualisation.
* **NLTK** is used for working with processing, computational linguistics, and natural language recognition.
* **Scikit-image** is used for image processing.
* **PyBrain** is used for neural networks, unsupervised, and reinforcement learning.
* **Caffe** is used for deep learning that lets you switch between the CPU and the GPU and processes 60+ mln images a day using a single NVIDIA K40 GPU.
* **StatsModels** is used for statistical algorithms and data exploration.



**Fig 2.1:** Python Libraries

1. Flexibility:-Python’s flexibility makes it a prominent choice for machine learning.

Developers get a choice between using OOPs or scripting.

You do not have to recompile the source code as you can apply any changes and quickly go through the results.

To achieve the set goals, you can integrate Python with other languages.

Also, with this flexibility, the developers get to pick the programming styles with which they are comfortable or they can even merge these styles to come with effective solutions for various types of problems.

The imperative style includes commands that detail how a computer should execute these commands. This style allows you to define the order of computations that occur such as a change of the program state

.

The functional style is also known as declarative as it declares what operations need to be executed.

The object-oriented style is established on two concepts: class and object, where similar objects form classes. Python does not completely back up this style but developers can still implement this style to a certain limit.

The procedural style proceeds tasks in a bit-by-bit style, hence, it happens to be very common among beginners. This style is usually used in sequencing, iteration, modularization, and selection.

With flexibility, programmers get to handle the situation better and operate in a safe environment. This reduces the chances of mistakes

1. Platform Independence:- Python’s adaptability adds to its features of being simple to learn and easy to use. When you use Python for machine learning development, you can run it on any operating system such as Windows, Linux, Unix, and macOS among others.

If the software developer wants to shift the process from one platform to another, they have to execute many small changes and alter a few lines of code so that this code can be executed on the next platform.

Packages like

PyInstaller come in handy when the developer is making ready the code to run on different platforms.

This helps developers not only save time but also money when it comes to testing on different platforms, making the complete process simpler and convenient.

1. Readability:- As Python is simple to read and make out, Python developers do not face troubles

understanding, changing, copying, and pasting their peers’ code.

When using Python, there is no confusion, mistakes, or inconsistent paradigms. This makes the exchange of algorithms, tools, and ideas between AI and ML professionals productive.

Tools like IPython, an interactive shell, offer additional features like debugging, testing, and tab-completion among others, and eases the work process.  
  
That’s why the Python Machine Learning combo is going to be the future of programming.

1. Good Visualization Options:- Python comes with various libraries and few of them happen to be great

visualisation tools.

For developers, the right representation of data in a creative and understandable format happens to be important in artificial intelligence, deep learning, and machine learning.

Libraries like Matplotib helps data scientists make charts, histograms, and plots for effective presentation, visualization, and data understanding.

With various application programming interfaces, the process of making clear reports and visualization also becomes simpler.

1. Community Support:- Having strong community support with any programming language is a win-win

situation for developers and programmers.

As Python is an open-source language, programmers at all levels get to access many resources. Also, the language is free and comes with numerous useful libraries and tools.

Significant Python documentation is accessible online and also in Python communities and forums.

Here programmers and machine learning developers have conversations about errors, how to solve problems, and they help each other out.

1. Ease of Learning:- Python is effortless and so is its syntax. You can use it to apply simple computations

such as the sum of two strings to complicated processes such as building a Machine Learning model.

1. Simple and Consistent:- Python is effortless and so is its syntax. You can use it to apply simple

computations such as the sum of two strings to complicated processes such as building a Machine Learning model.

1. Speed of Execution:- Python is readable and one can also execute its formulas in shorter durations.

Machine Learning, especially Deep Learning, a subset of Machine Learning using Deep Neural Nets is popular for its long model training sessions. These sessions can last from hours to days.

1. Gentle Learning Curve:- Python is readable and one can also execute its formulas in shorter durations.

Machine Learning, especially

Deep Learning, a subset of Machine Learning using Deep Neural Nets is popular for its long model training sessions. These sessions can last from hours to days.

**2.3 Data-wrangling tools:-**

1. NumPy:-

NumPy is a fast and easy-to-use open-source scientific computing Python library. It’s also a fundamental library for the data science ecosystem because many of the most popular Python libraries like Pandas and Matplotlib are built on top of NumPy.

In addition to serving as the foundation for other powerful libraries, NumPy has a number of qualities that make it indispensable for [Python for data analysis](https://www.dataquest.io/path/data-analyst/). Thanks to its speed and versatility, NumPy’s vectorization, indexing, and broadcasting concepts represent the de facto standard for array computing; however, NumPy really shines when working with multi-dimensional arrays. It also offers a comprehensive toolbox of numerical computing tools like linear algebra routines, Fourier transforms, and more.

NumPy can do a lot for many people. Its high-level syntax allows programmers from any background or experience level to use its powerful data processing capabilities. For example, NumPy enabled the Event Horizon Space Telescope to produce the first-ever image of black holes. It also confirmed the existence of gravitational waves, and it’s currently accelerating a variety of scientific studies and sports analytics.

Is it a surprise that a program that covers everything from sports to space can also help you manage and clean your data?

1. Pandas:-

Pandas is one of the libraries powered by NumPy. It’s the #1 most widely used data analysis and manipulation library for Python, and it’s not hard to see why.

Pandas is fast and easy to use, and its syntax is very user-friendly, which, combined with its incredible flexibility for manipulating DataFrames, makes it an indispensable tool for analyzing, manipulating, and cleaning data.

This powerful Python library not only handles numerical data, it also handles text data and dates. It allows you to join, merge, concatenate, or duplicate DataFrames and easily add or remove columns or rows using its drop() function.

In short, pandas combines speed, ease of use, and flexible functionality to create an incredibly powerful tool that makes data manipulation and analysis fast and easy

1. SciPy:-

Unlike the other mentions on this list, SciPy is not just a library; it’s an entire data science ecosystem offering a collection of open source libraries already mentioned on this list, including NumPy, Matplotlib, and pandas.

Additionally, SciPy also makes available a number of specialized tools, one of which is Scikit-learn, whose “Preprocessing” package you can leverage for data cleaning and standardizing of datasets.

1. Dabl:-

One of scikit-learn’s core engineers developed Dabl as a data analysis library to simplify [data exploration](https://www.dataquest.io/course/data-exploration/) and preprocessing.

Dabl has an integral process to detect certain data types and quality problems within a dataset and automatically apply the proper pre-processing procedures.

It can handle missing values, convert categorical variables into numerical values, and it even has built-in visualization options to facilitate quick data exploration.

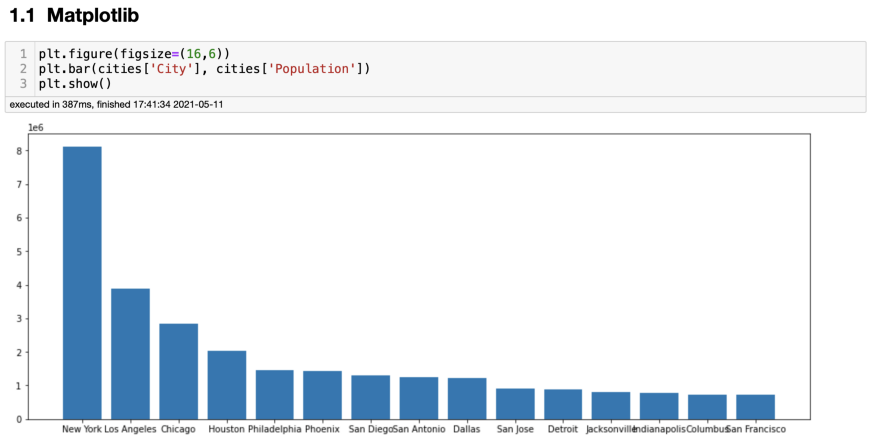
**2.4 Data Visualization library’s:-**

[1. Matplotlib](https://matplotlib.org/)

Matplotlib is a data visualization library and 2-D plotting library of Python It was initially released in 2003

and it is the most popular and widely-used plotting library in the Python community.

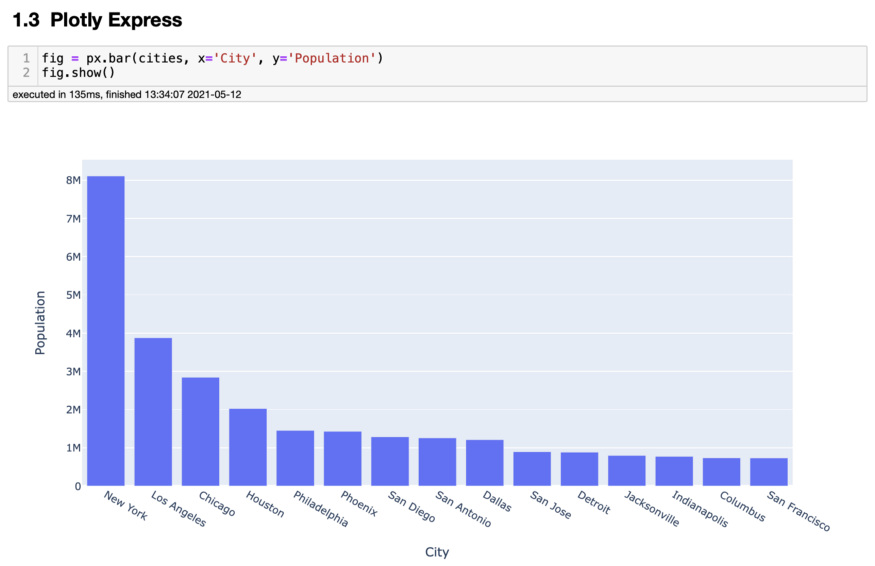
It comes with an interactive environment across multiple platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, etc. It can be used to embed plots into applications using various  GUI toolkits like Tkinter, GTK+, wxPython, Qt, etc. So you can use Matplotlib to create plots, bar charts, pie charts, histograms, scatterplots, error charts, power spectra, stemplots, and whatever other visualization charts you want! The Pyplot module also provides a MATLAB-like interface that is just as versatile and useful as MATLAB while being free and open source.



**Fig 2.2:** MatPlotLib

[2. Plotly](https://plotly.com/)

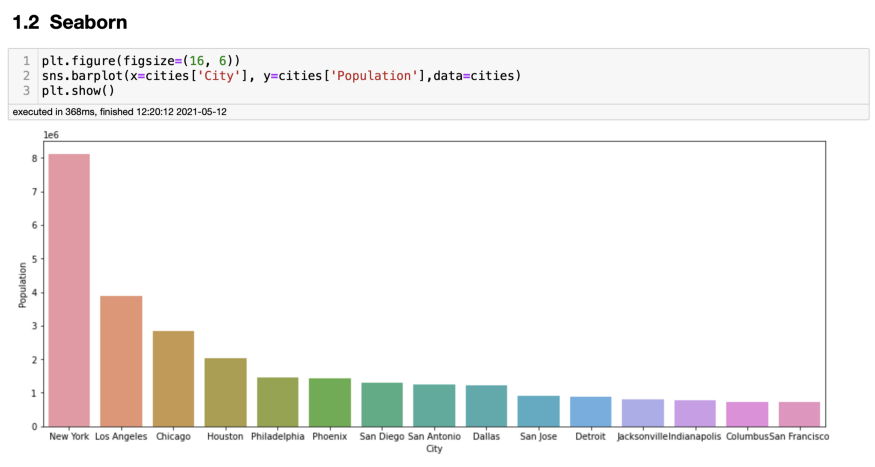
Plotly is a free open-source graphing library that can be used to form data visualizations. Plotly (plotly.py) is built on top of the Plotly JavaScript library (plotly.js) and can be used to create web-based data visualizations that can be displayed in Jupyter notebooks or web applications using Dash or saved as individual [HTML](https://www.geeksforgeeks.org/html-tutorials/) files. Plotly provides more than 40 unique chart types like scatter plots, histograms, line charts, bar charts, pie charts, error bars, box plots, multiple axes, sparklines, dendrograms, 3-D charts, etc. Plotly also provides contour plots, which are not that common in other data visualization libraries. In addition to all this, Plotly can be used offline with no internet connection.



**Fig 2.3:** Plotly

[3. Seaborn](https://seaborn.pydata.org/)

Seaborn is a Python data visualization library that is based on Matplotlib and closely integrated with the NumPy and pandas data structures. Seaborn has various dataset-oriented plotting functions that operate on data frames and arrays that have whole datasets within them. Then it internally performs the necessary statistical aggregation and mapping functions to create informative plots that the user desires. It is a high-level interface for creating beautiful and informative statistical graphics that are integral to exploring and understanding data. The Seaborn data graphics can include bar charts, pie charts, histograms, scatterplots, error charts, etc. Seaborn also has various tools for choosing color palettes that can reveal patterns in the data.



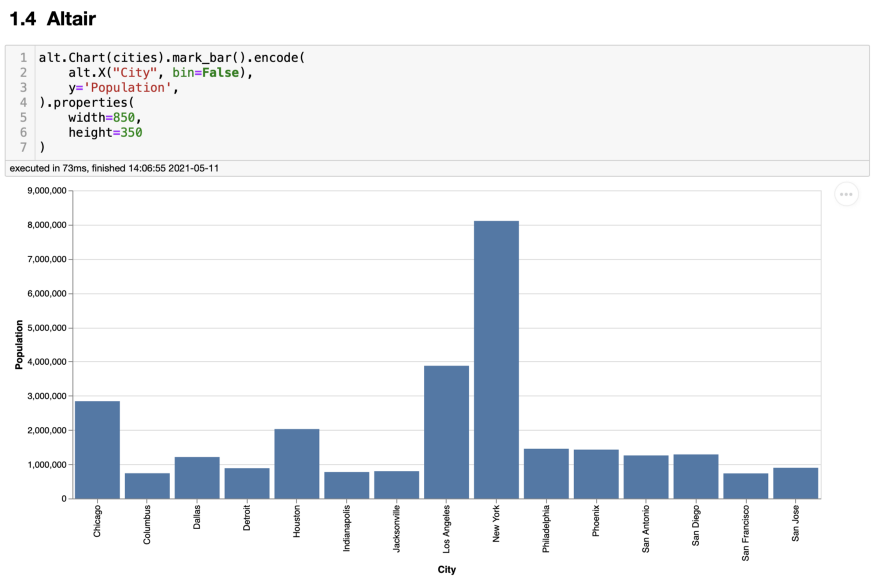
**Fig 2.4:** Seaborn

[4. GGplot](http://ggplot.yhathq.com/)

Ggplot is a Python data visualization library that is based on the implementation of ggplot2 which is created for the programming language R. Ggplot can create data visualizations such as bar charts, pie charts, histograms, scatterplots, error charts, etc. using high-level API. It also allows you to add different types of data visualization components or layers in a single visualization. Once ggplot has been told which variables to map to which aesthetics in the plot, it does the rest of the work so that the user can focus on interpreting the visualizations and take less time in creating them. But this also means that it is not possible to create highly customized graphics in ggplot. Ggplot is also deeply connected with pandas so it is best to keep the data in DataFrames.

[5. Altair](https://altair-viz.github.io/)

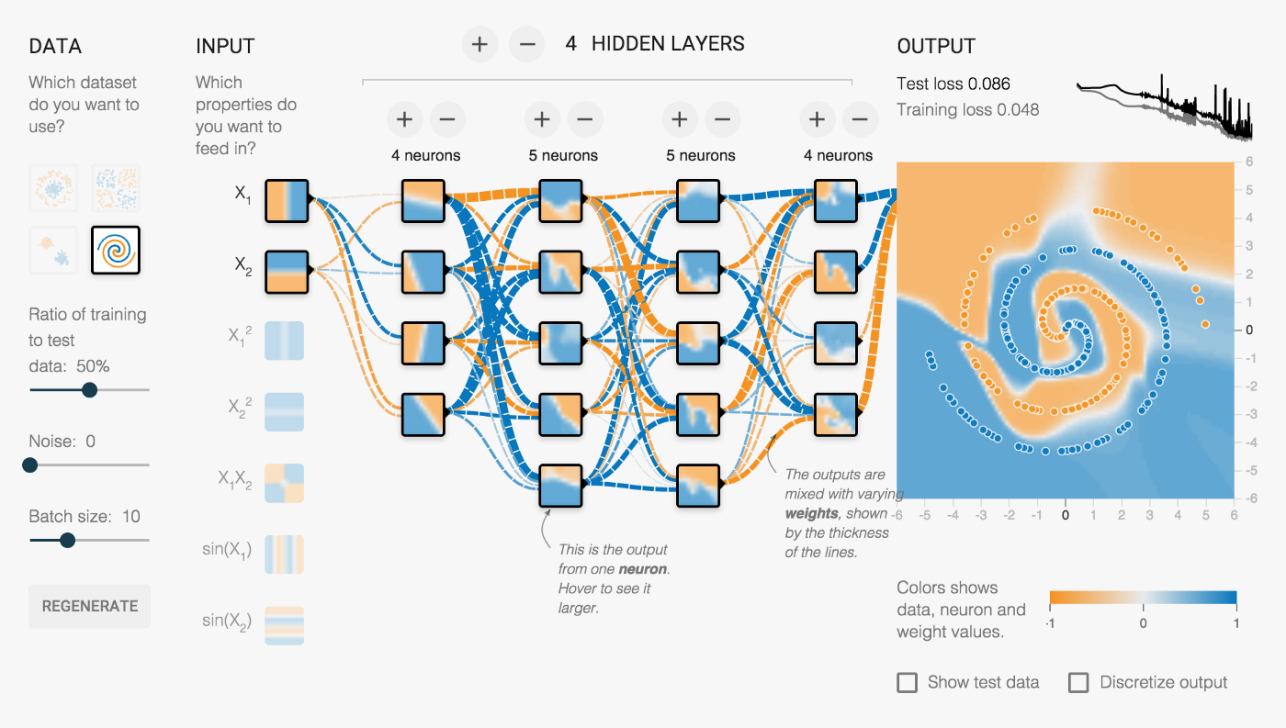
Altair is a statistical data visualization library in Python. It is based on Vega and Vega-Lite which are a sort of declarative language for creating, saving, and sharing data visualization designs that are also interactive. Altair can be used to create beautiful data visualizations of plots such as bar charts, pie charts, histograms, scatterplots, error charts, power spectra, stemplots, etc. using a minimal amount of coding. Altair has dependencies which include python 3.6, entrypoints, jsonschema, NumPy, Pandas, and Toolz which are automatically installed with the Altair installation commands. You can open Jupyter Notebook or JupyterLab and execute any of the code to obtain that data visualizations in Altair. Currently, the source for Altair is available on GitHub



**Fig 2.5:** Altair

**2.5 TensorFlow for Deep-Lerning:-**

* TensorFlow is a framework created by Google for creating Deep Learning models. Deep Learning is a category of machine learning models (=algorithms) that use multi-layer neural networks.
* Machine Learning has enabled us to build complex applications with great accuracy. Whether it has to do with images, videos, text or even audio, Machine Learning can solve problems from a wide range.
* Tensorflow can be used to achieve all of these applications.
* The reason for its popularity is the ease with which developers can build and deploy applications. The GitHub projects which we’ll look closer at due the next parts are very powerful but also so easy to work with.
* Moreover, Tensorflow was created with processing power limitations in mind. The library can be ran on computers of all kinds, even on smartphones (yes, even on that overpriced thing with half an apple on it). can guarantee you, working on a Intel Core i3 with 8 GB of RAM, you won’t have performance issues.

****

**Fig 2.6** TesorFlow Working

**Chapter-3**

**IMPLEMENTATION**

**3.1 Introduction**

For the implementation of the problem statement different methods being used for different problem objective is to come up with the most effective solution which best fits with the data available to us.

The steps involved in all the problem statement are same which includes:-

* Data Wrangling :- The cleaning of dataset we have as the data we may not be directly given to our machine learning or neural network model as in our data there might be some missing values, format of the dataset may not be aligned with the input required for the learning model. Therefore, before feeding the data to machine learning or neural network model data is cleaned for which we have various library available which includes numpy , pandas which is widely used
* Data Visualization:- Visualization of the data helps us to understand more about the dataset which helps us to further decide up the learning algorithm or model which can Learning algorithms plays an important role in the accuracy of our model as there are wide range of learning algorithms available which works differently on the basis of the dataset we provide. So it is very important to choose the learning algorithm which best fits our data to get good accuracy out of our neural network model
* Some of the widely used learning algorithms available for training our neural network model are:-

use for further processing of data

* Splitting up of Dataset:- The dataset is splitted into two parts that is training and test dataset the training dataset is used to train the data and the test data is to test how the model is performing and to calculate score , efficiency and accuracy of our model
* Model Building :- As there is no fixed method available we need to build a model using hit and trail and choose the model which best suits our data.
* Learning Algorithms to be used:- There are wide range of learning algorithms available to train our model we need to choose the one which best fits our data.
* Training Model using dataset
* Predicting output

**3.2 Learning Algorithms used:-**

1. **Gradient descent**:-

Gradient descent is the most straightforward training algorithm. It requires information from the gradient vector, and hence it is a first-order method.

Let denote f(w(i))=f(i)f(w(i))=f(i) and ∇f(w(i))=g(i)∇f(w(i))=g(i). The method begins at a point w(0)w(0) and, until a stopping criterion is satisfied, moves from w(i)w(i) to w(i+1)w(i+1) in the training direction d(i)=−g(i)d(i)=−g(i). Therefore, the gradient descent method iterates in the following way:

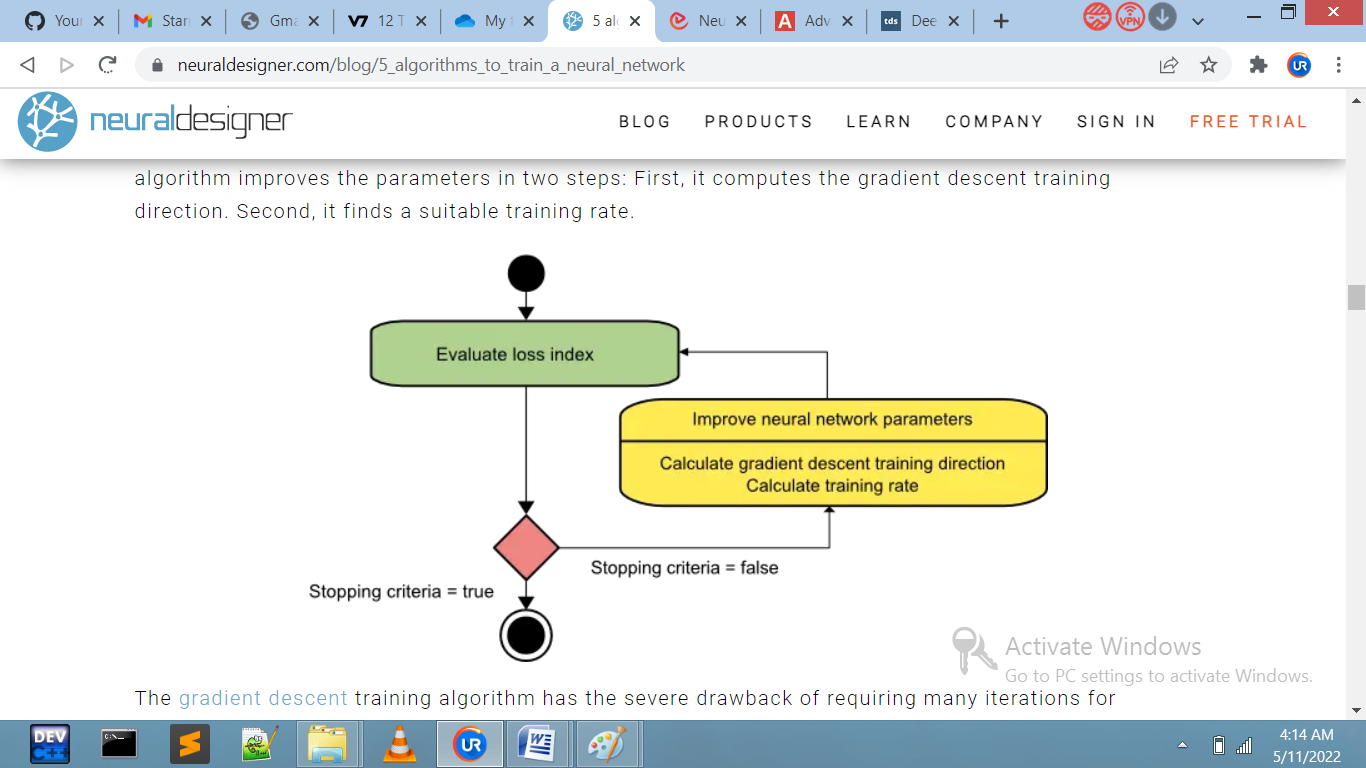
w(i+1)=w(i)−g(i)η(i),w(i+1)=w(i)−g(i)η(i),

for i=0,1,…i=0,1,….

The parameter ηη is the training rate. This value can either be set to a fixed value or found by one-dimensional optimization along the training direction at each step. An optimal value for the training rate obtained by line minimization at each successive step is generally preferable. However, there are still many software tools that only use a fixed value for the training rate.

The next picture is an activity diagram of the training process with gradient descent. As we can see, the algorithm improves the parameters in two steps: First, it computes the gradient descent training direction. Second, it finds a suitable training rate.

The gradient descent training algorithm has the severe drawback of requiring many iterations for functions that have long, narrow valley structures. Indeed, the downhill gradient is the direction in which the loss function decreases the most rapidly, but this does not necessarily produce the fastest convergence.



**Fig 3.1:** Gradient Descent

The following picture illustrates this issue.

Gradient descent is the recommended algorithm when we have massive neural networks, with many thousand parameters. The reason is that this method only stores the gradient vector (size nn), and it does not store the Hessian matrix (size n2n2).

1. **Newton's method :-**

Newton's method is a second-order algorithm because it uses the Hessian matrix. This method's objective is to find better training directions by using the second derivatives of the [loss](https://www.neuraldesigner.com/learning/tutorials/training-strategy#LossIndex) function.

Let denote f(w(i))=f(i)f(w(i))=f(i), ∇f(w(i))=g(i)∇f(w(i))=g(i) and Hf(w(i))=H(i)Hf(w(i))=H(i).

Consider the quadratic approximation of ff at w(0)w(0) using the Taylor's series expansion

f=f(0)+g(0)⋅(w−w(0))+0.5⋅(w−w(0))2⋅H(0)f=f(0)+g(0)⋅(w−w(0))+0.5⋅(w−w(0))2⋅H(0)

H(0)H(0) is the Hessian matrix of ff evaluated at the point w(0)w(0). By setting gg equal to 00 for the minimum of f(w)f(w), we obtain the next equation

g=g(0)+H(0)⋅(w−w(0))=0g=g(0)+H(0)⋅(w−w(0))=0

Therefore, starting from a parameter vector w(0)w(0), Newton's method iterates as follows

w(i+1)=w(i)−H(i)−1⋅g(i)w(i+1)=w(i)−H(i)−1⋅g(i)

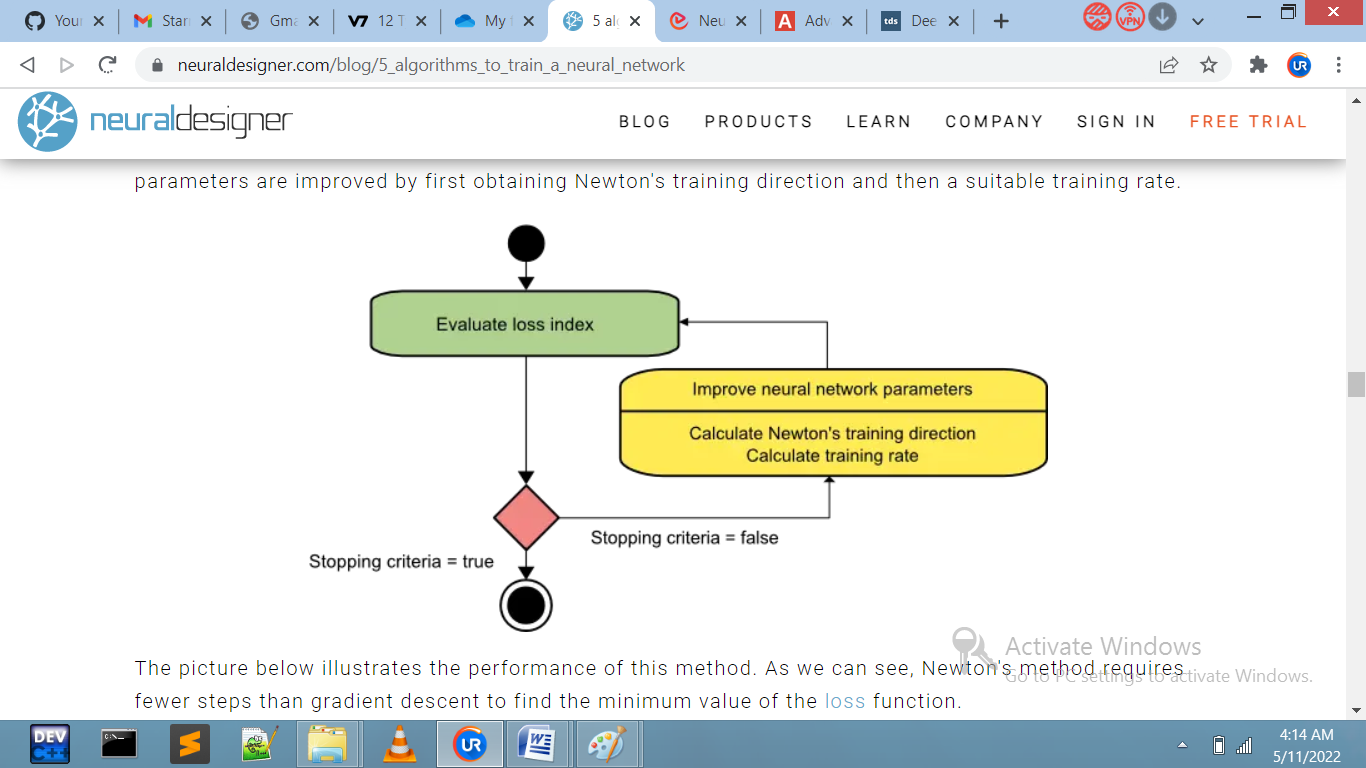
for i=0,1,…i=0,1,….

The vector H(i)−1⋅g(i)H(i)−1⋅g(i) is known as Newton's step. Note that this parameter change may move towards a maximum rather than a minimum. This occurs if the Hessian matrix is not positive definite. Thus, Newton's method does not guarantee to reduce the loss index at each iteration. To prevent that, Newton's method equation is usually modified as:

w(i+1)=w(i)−(H(i)−1⋅g(i))ηw(i+1)=w(i)−(H(i)−1⋅g(i))η

for i=0,1,…i=0,1,….

The training rate, ηη, can either be set to a fixed value or found by line minimization. The vector d(i)=H(i)−1⋅g(i)d(i)=H(i)−1⋅g(i) is now called Newton's training direction.



**Fig 3.2:** Newton’s Method

The following figure depicts the state diagram for the training process with Newton's method. The parameters are improved by first obtaining Newton's training direction and then a suitable training rate.

1. **Conjugate gradient**

The conjugate gradient method can be regarded as something intermediate between gradient descent and Newton's method. It is motivated to accelerate the typically slow convergence associated with gradient descent. This method also avoids the information requirements associated with the Hessian matrix's storage, evaluation, and inversion, as Newton's method requires.

In the conjugate gradient training algorithm, the search is performed along with conjugate directions.

They generally produce faster convergence than gradient descent directions. These training directions are conjugated concerning the Hessian matrix.

Let denote d the training direction vector. Then, starting with an initial parameter vector w(0)w(0) and an initial training direction vector d(0)=−g(0)d(0)=−g(0), the conjugate gradient method constructs a sequence of training directions as:

d(i+1)=g(i+1)+d(i)⋅γ(i),d(i+1)=g(i+1)+d(i)⋅γ(i),

for i=0,1,…i=0,1,….

Here γγ is called the conjugate parameter, and there are different ways to calculate it. Two of the most used are due to Fletcher and Reeves and Polak and Ribiere. For all conjugate gradient algorithms, the training direction is periodically reset to the negative of the gradient.

The parameters are then improved according to the following expression.

w(i+1)=w(i)+d(i)⋅η(i)w(i+1)=w(i)+d(i)⋅η(i)

for i=0,1,…i=0,1,….

The training rate, ηη, is usually found by line minimization.

The picture below depicts an activity diagram for the training process with the conjugate gradient. Here

improvement of the parameters is done in two steps.

First, the algorithm computes the conjugate gradient training direction.

Second, it finds a suitable training rate in that direction.

This method has proved to be more effective than gradient descent in training neural networks. Since it does not require the Hessian matrix, the conjugate gradient also performs well with vast neural networks

1. **Quasi-Newton method**

The application of Newton's method is computationally expensive. Indeed, it requires many operations to evaluate the Hessian matrix and compute its inverse. Alternative approaches, known as quasi-Newton, are developed to solve that drawback. These methods do not calculate the Hessian directly and then evaluate its inverse. Instead, they build up an approximation to the inverse Hessian. This approximation is computed using only information on the first derivatives of the loss function.

The Hessian matrix is composed of the second partial derivatives of the loss function. Thus, the main idea behind the quasi-Newton method is approximating the inverse Hessian by another matrix GG, using only the first partial derivatives of the loss function. Then, the quasi-Newton formula is expressed as

w(i+1)=w(i)−(G(i)⋅g(i))⋅η(i)w(i+1)=w(i)−(G(i)⋅g(i))⋅η(i)

for i=0,1,…i=0,1,….

The training rate ηη can either be set to a fixed value or found by line minimization. The inverse Hessian approximation GG has different flavors. Two of the most used ones are the Davidon–Fletcher–Powell (DFP) and the Broyden–Fletcher–Goldfarb–Shanno (BFGS) formulas.

The activity diagram of the quasi-Newton training process is illustrated below. Improvement of the parameters is performed in two steps.

First, the algorithms obtains the quasi-Newton training direction.

Second, it finds a satisfactory training rate.

This is the default method to use in most cases: It is faster than gradient descent and conjugate gradient, and the exact Hessian does not need to be computed and inverted.

1. **Levenberg-Marquardt algorithm**

The Levenberg-Marquardt algorithm is designed to work specifically with loss functions which take the form of a sum of squared errors. It works without computing the exact Hessian matrix. Instead, it works with the gradient vector and the Jacobian matrix.

Consider a loss function which takes the form of a sum of squared errors,

f=m∑i=1e2if=∑i=1mei2

Here mm is the number of training samples.

We can define the Jacobian matrix of the loss function as that containing the derivatives of the errors concerning the parameters,

Ji,j=∂ei∂wj,Ji,j=∂ei∂wj,

for i=1,…,mi=1,…,m and j=1,…,nj=1,…,n.

Where mm is the number of samples in the data set, and nn is the number of parameters in the neural network.

Note that the size of the Jacobian matrix is m⋅nm⋅n.

We can compute the gradient vector of the loss function as

∇f=2JT⋅e∇f=2JT⋅e

Here ee is the vector of all error terms.

\

Finally, we can approximate the Hessian matrix with the following expression.

Hf≈2JT⋅J+λIHf≈2JT⋅J+λI

Where λλ is a damping factor that ensures the positiveness of the Hessian and II is the identity matrix.

The next expression defines the parameters improvement process with the Levenberg-Marquardt algorithm

w(i+1)=w(i)−(J(i)T⋅J(i)+λ(i)I)−1⋅(2J(i)T⋅e(i)),w(i+1)=w(i)−(J(i)T⋅J(i)+λ(i)I)−1⋅(2J(i)T⋅e(i)),

for i=0,1,…i=0,1,….

When the damping parameter λλ is zero, this is just Newton's method, using the approximate Hessian matrix. On the other hand, when λλ is large, this becomes gradient descent with a small training rate.

The parameter λλ is initialized to be large so that the first updates are small steps in the gradient descent direction. If any iteration results in a fail, then λλ is increased by some factor. Otherwise, as the loss decreases, λλ is reduced, so the Levenberg-Marquardt algorithm approaches the Newton method. This process typically accelerates the convergence to the minimum.

The picture below represents a state diagram for the training process of a neural network with the

Levenberg-Marquardt algorithm. The first step is to calculate the loss, the gradient, and the Hessian approximation. Then the damping parameter is adjusted to reduce the loss at each iteration

As we have seen, the Levenberg-Marquardt algorithm is a method tailored for functions of the type sum-of-squared-error. That makes it to be very fast when training neural networks measured on that kind of errors.

However, this algorithm has some drawbacks. First, it cannot minimize functions such as the root mean squared error or the cross-entropy error. Also, the Jacobian matrix becomes enormous for big data sets and neural networks, requiring much memory. Therefore, the Levenberg-Marquardt algorithm is not recommended when we have big data sets or neural networks.

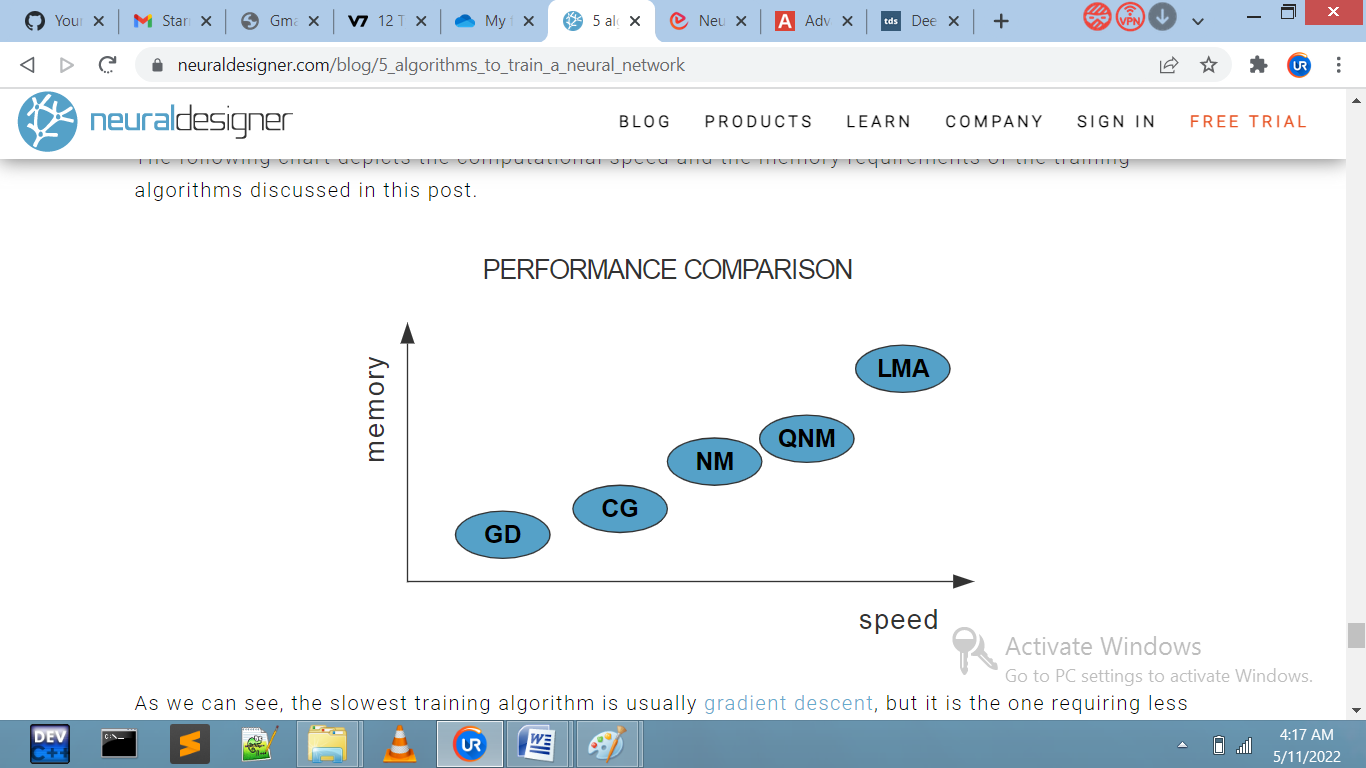
**Performance comparison**

The following chart depicts the computational speed and the memory requirements of the training algorithms discussed in this post.

As we can see, the slowest training algorithm is usually gradient descent, but it is the one requiring less memory.

On the contrary, the fastest one might be the Levenberg-Marquardt algorithm, but it usually requires much memory.

A good compromise might be the quasi-Newton method.



**Fig 3.3:** Performance Comparision

**Conclusions**

To conclude, if our neural network has many thousands of parameters, we can use gradient descent or conjugate gradient, to save memory.

If we have many neural networks to train with just a few thousand samples and a few hundred parameters, the best choice might be the Levenberg-Marquardt algorithm.

In the rest of situations, the quasi-Newton method will work well.

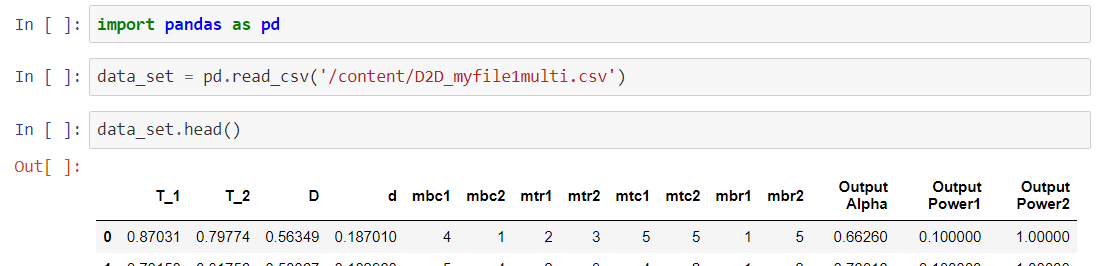
Neural Designer implements all that optimization algorithms. You can download the free trial to see how they work in practice.

**3.3 Problem 1 -** Designing a neural network model for given set of data and the learning algorithm to be used is Levenberg-Marquardt

Steps Involved:-

Importing of Libraries:-

In the first step all the libraries that are involved or that will be used later is imported



**Fig 3.3:** Library Import

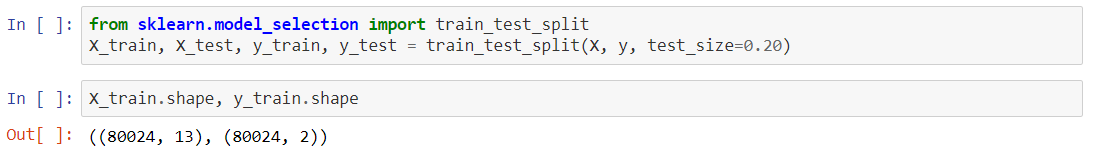
Loading Dataset:-

Now, the dataset required for the model to be trained is imported on which the further operation will be done.

Splitting data into training and testing data:-

In a supervised –learning model the learning is done under supervision i.e. we have the output corresponding to particular input

Therefore, at the time of training the dataset is split into two parts training data and testing data.

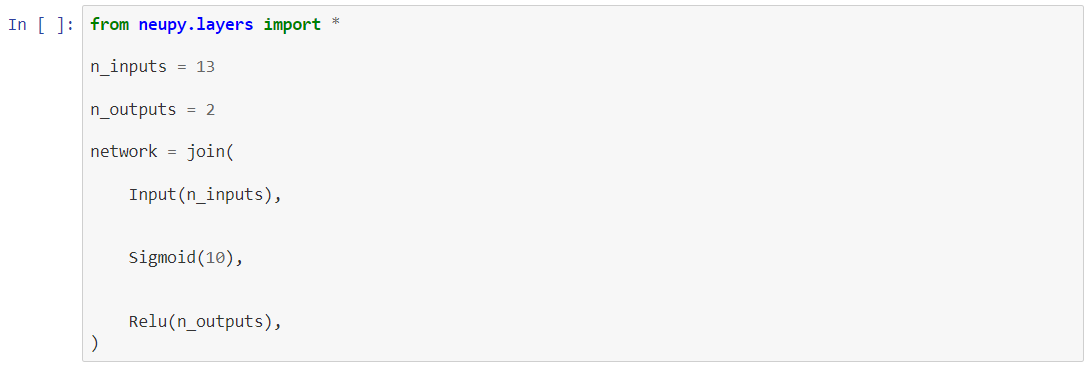


**Fig 3.4:** Train\_Test\_Split

Now, entire dataset is divided into training and testing set so that prediction does not over fit or under fit and correct values are obtained. train\_test\_split() is inbuilt function from scikit learn for splitting x and y variables data. “test size” parameter is used to divide of entire dataset (20%) into test data and remaining as training data. Setting random state as null would not allow random values to be taken from dataset.

Choosing the model:-

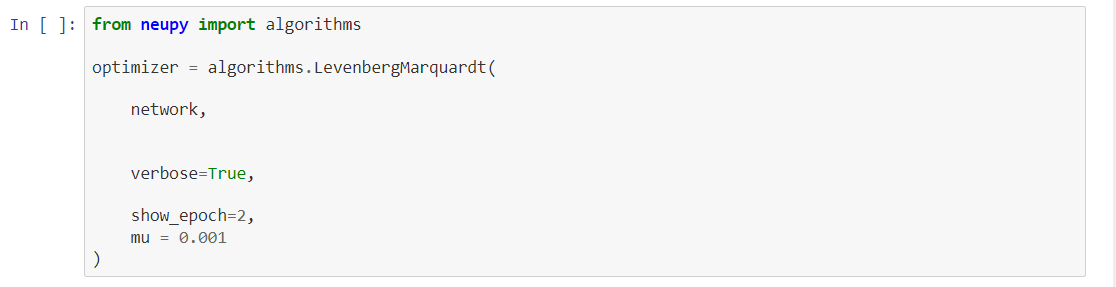
In the step model is chosen which best suit our data here for the problem deep neural network model is used with 13 input layer and 2 output layer with also sigmoid function is used as an activation function. The library used for the purpose is neupy library.

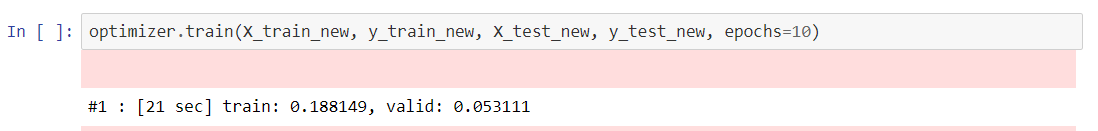


**Fig 3.5:** Choosing Model

Fitting and training :-

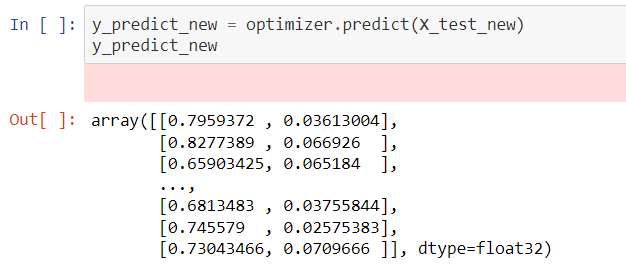
Now the training of the neural network is done and the learning algorithm used is levenberg-marquardt . Levenberg-Marqudart is the method tailored for functions of the type sum-of-squared-errors





**Fig 3.6:** Training Model

Predict output:-



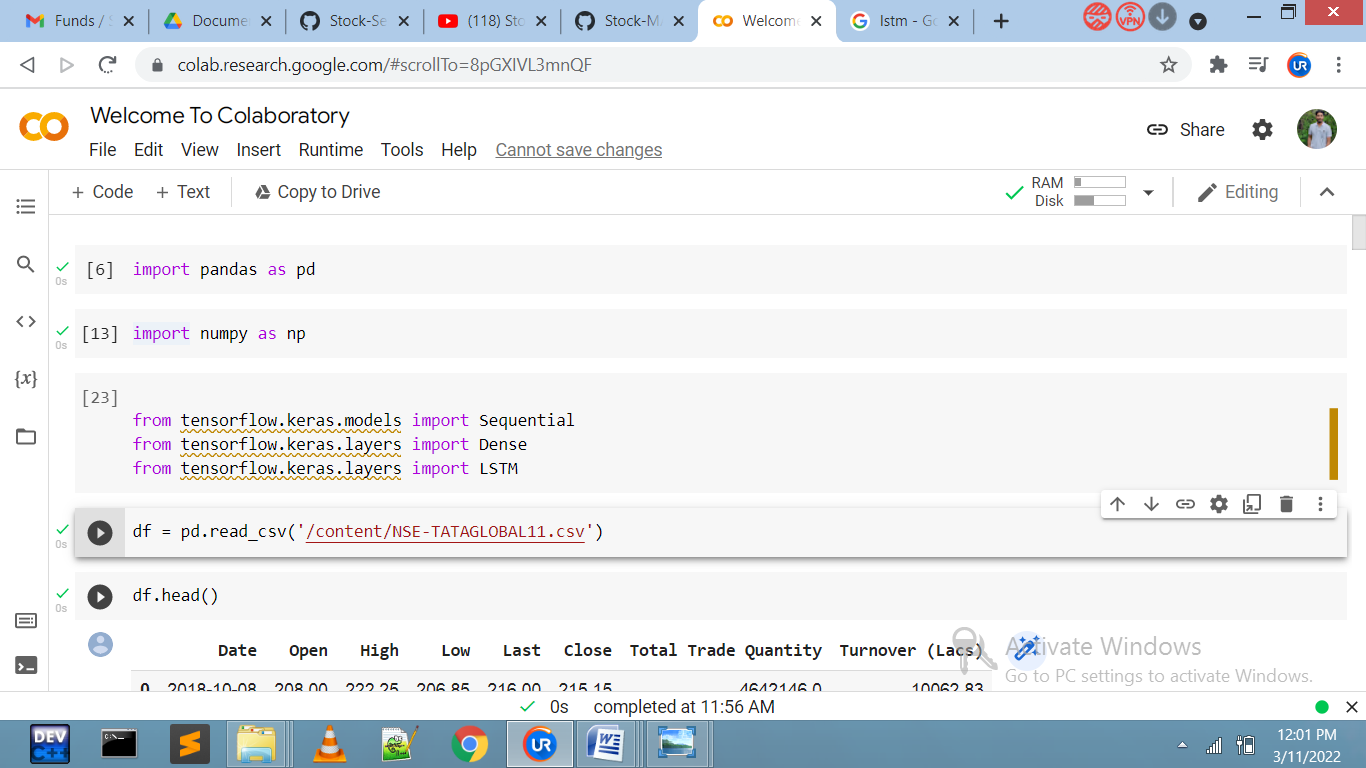
**Fig 3.7:** Predicting Output

**3.4 Problem 2:-** Designing a machine learning model for the stock price prediction using the previous data of that stock.

Steps Involved:-

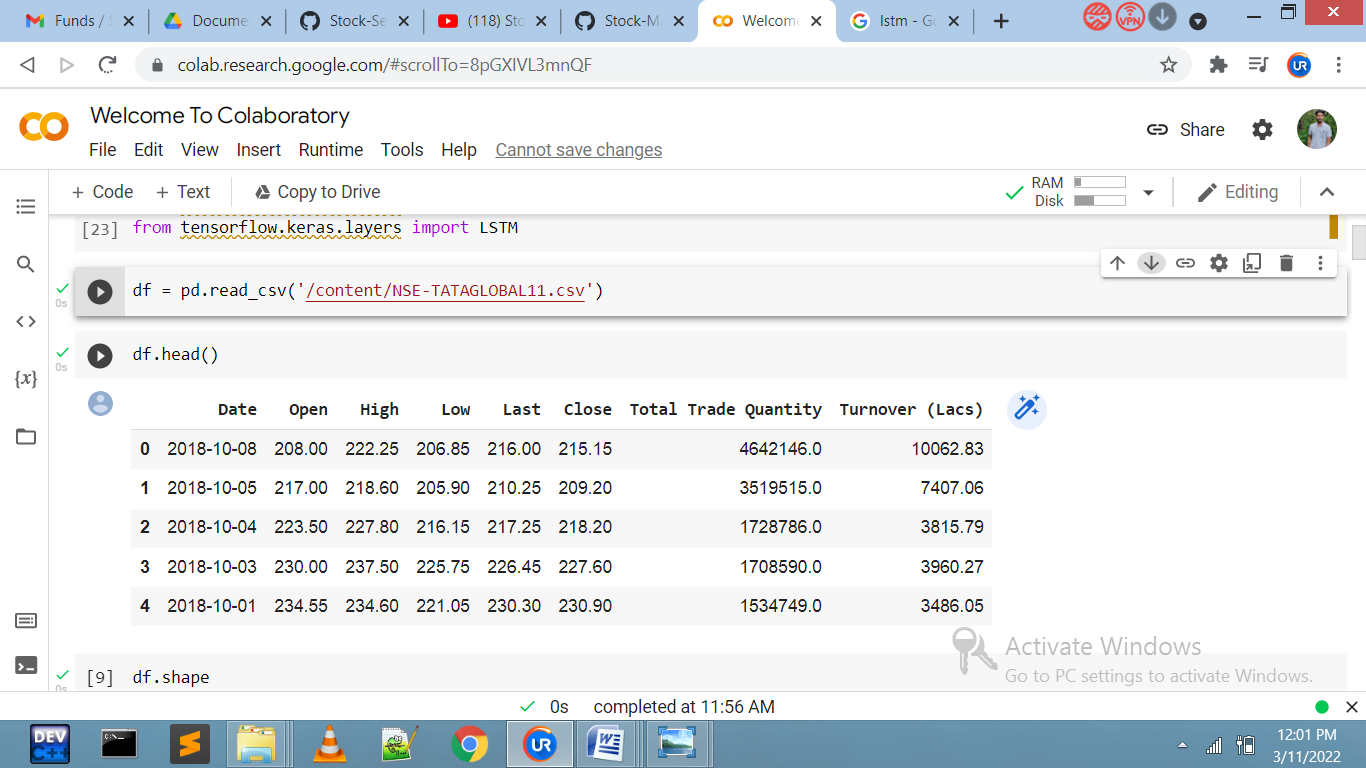
Importing of Libraries:-

In the first step all the libraries that are involved or that will be used later is imported



Loading Dataset:-

Now, the dataset required for the model to be trained is imported on which the further operation will be done.

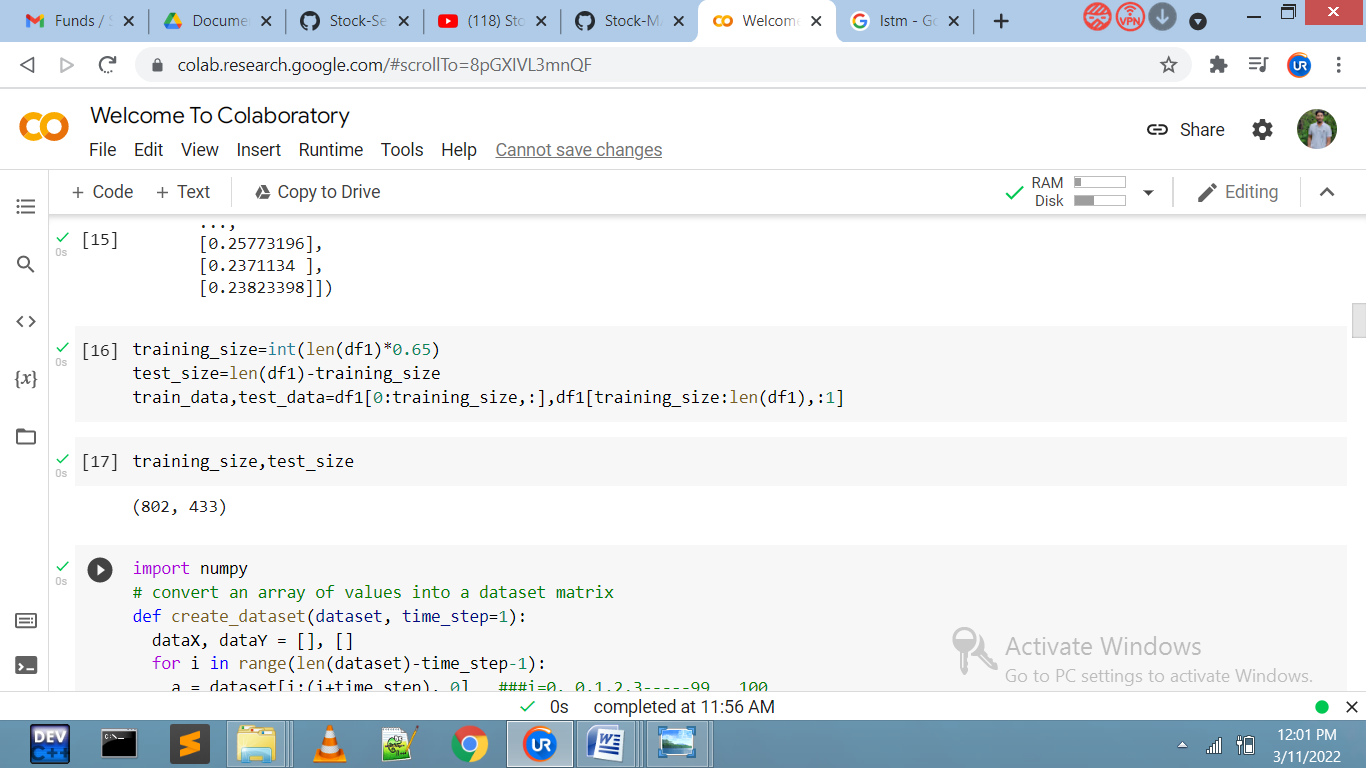


Splitting data into training and testing data:-

In a supervised –learning model the learning is done under supervision i.e. we have the output corresponding to particular input

Therefore, at the time of training the dataset is split into two parts training data and testing data.

Therefore, at the time of training the dataset is split into two parts training data and testing data.

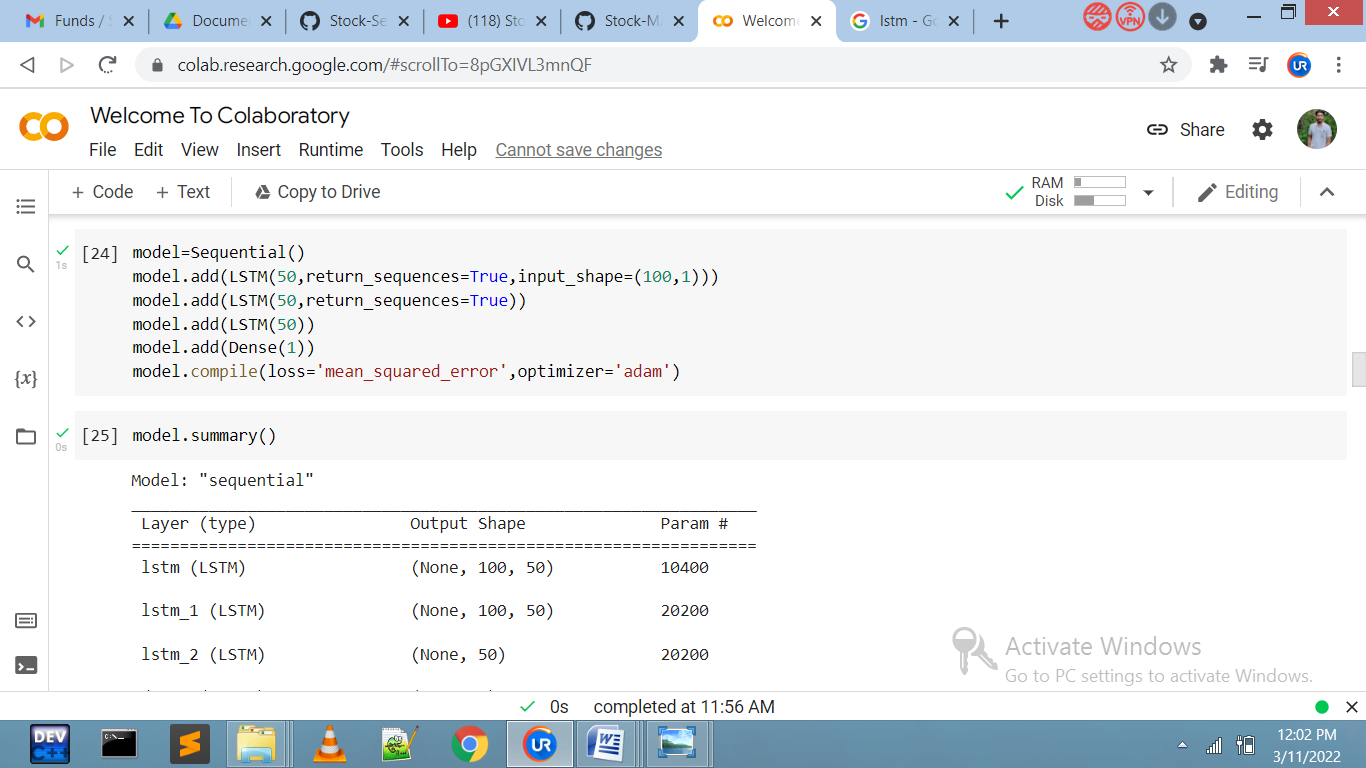


Now, entire dataset is divided into training and testing set so that prediction does not overfit or underfit and correct values are obtained. train\_test\_split() is inbuilt function from scikit learn for splitting x and y variables data. “test size” parameter is used to divide of entire dataset (20%) into test data and

remaining as training data. Setting random state as null would not allow random values to be taken from dataset.

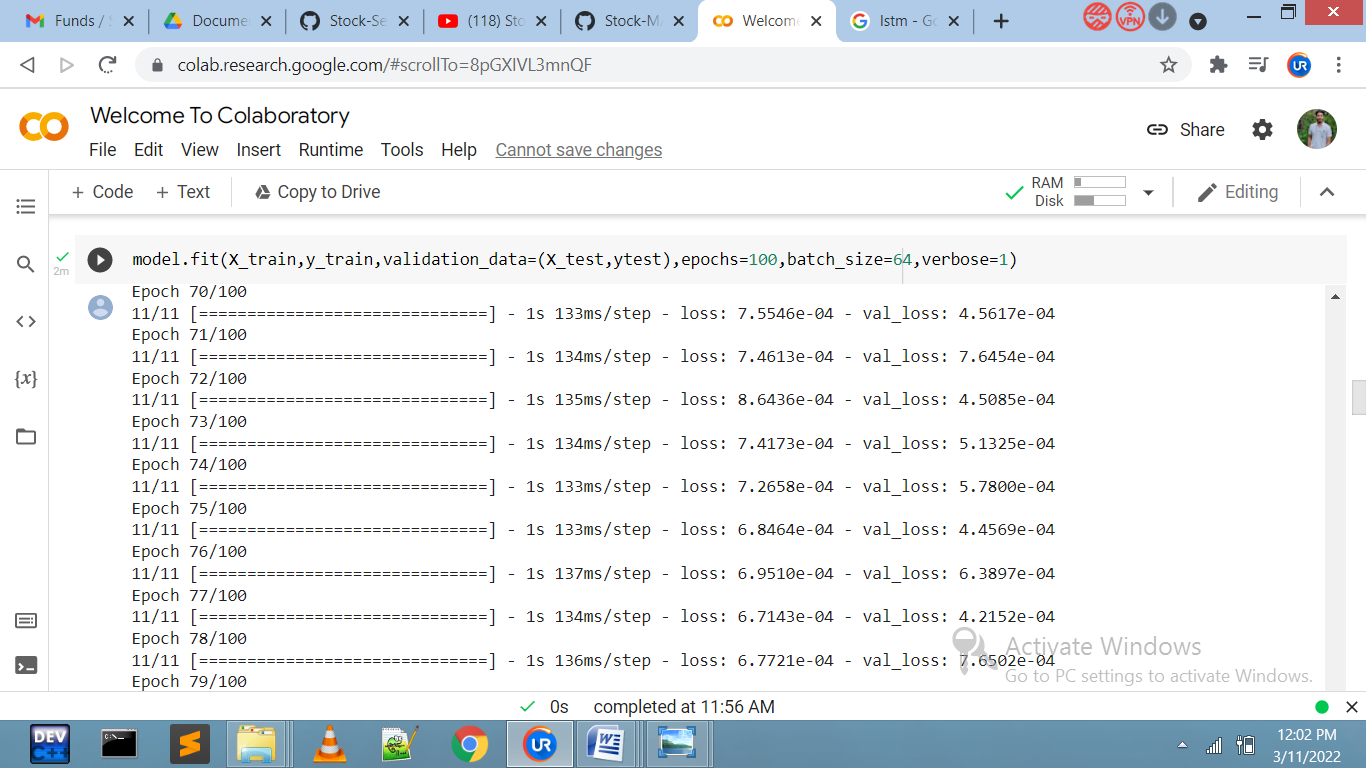
Choosing the model:-

LSTM model is used for the problem as LSTM networks are well-suited to classify, processing and making predictions baded on time series data.

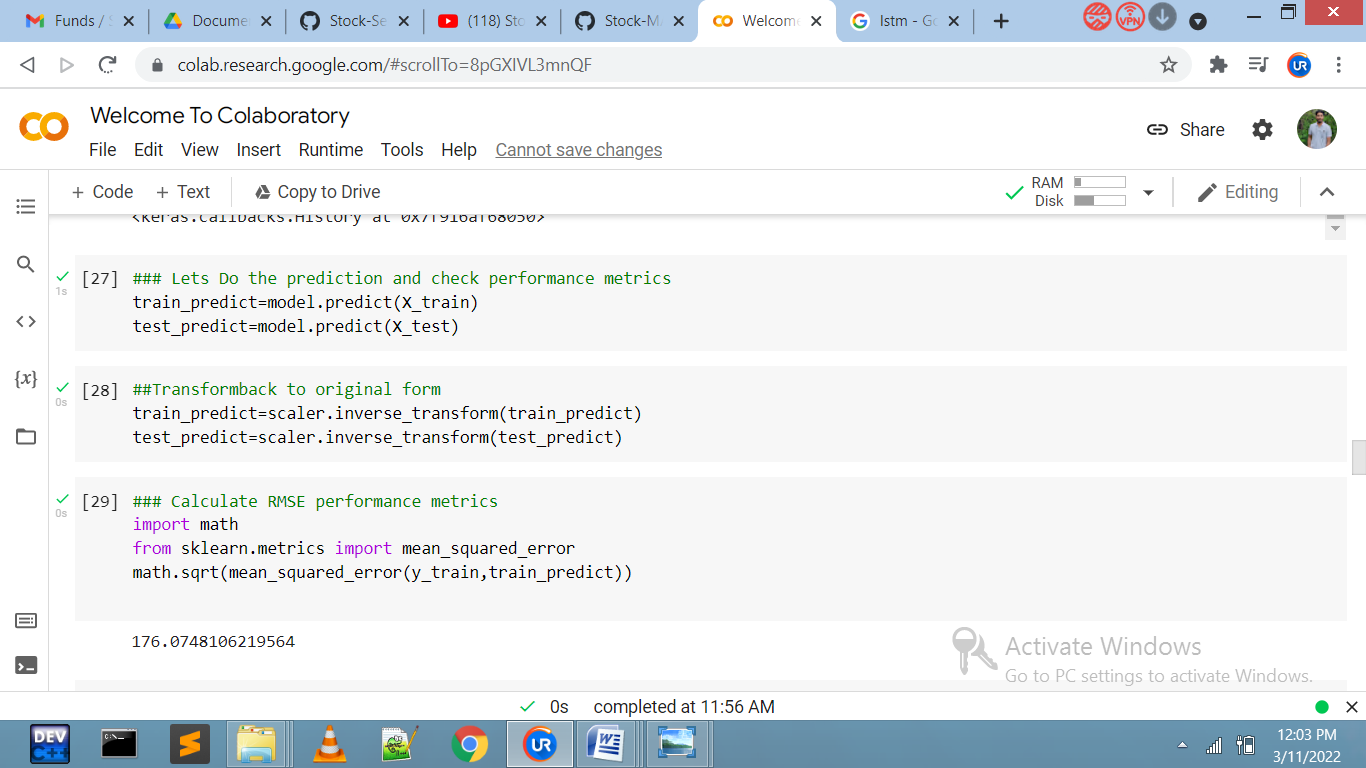


Fitting and training :-

We now fit and train network and the optimizer or the learning algorithm used for the purpose is Adam optimizer.



Predict output and accuracy score calculation:-

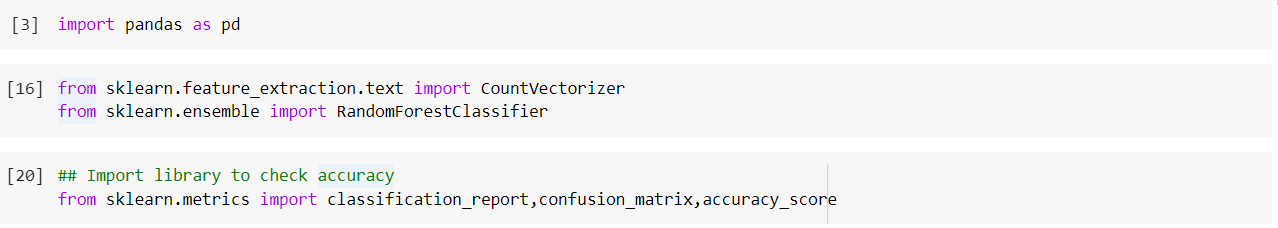


**3.5 Problem 3:-** Sentimental analysis of a Top headlines to predict whether the price of stock will rise or fall.

Steps Involved:-

1. Importing of Libraries:-

In the first step all the libraries that are involved or that will be used lated is imported



Loading Dataset:-

Now, the dataset required for the model to be trained is imported on which the further operation will be done.

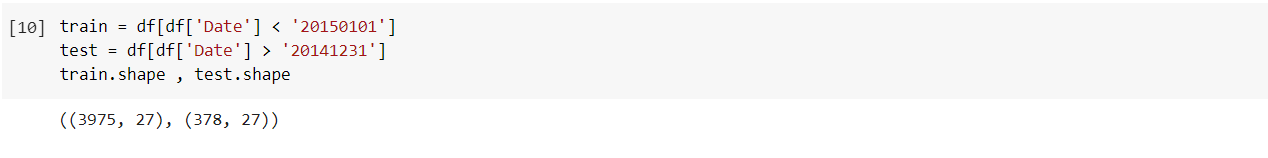


Splitting data into training and testingdata:-

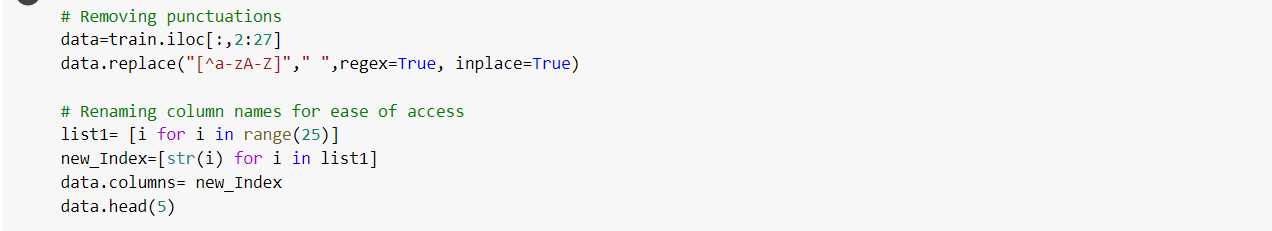
In a supervised –learning model the learning is done under supervision i.e. we have the output corresponding to particular input

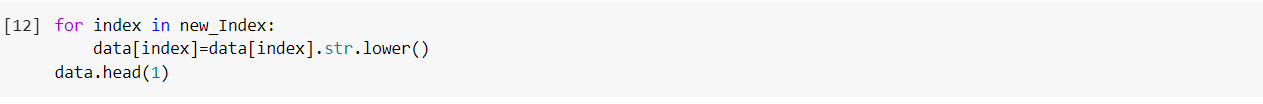
Therefore, at the time of training the dataset is split into two parts training data and testing data.

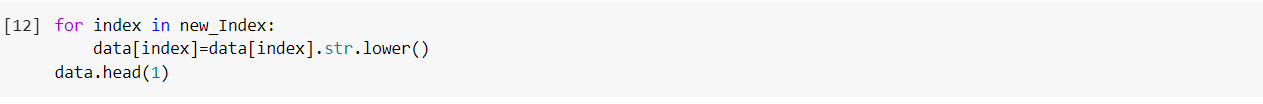
.



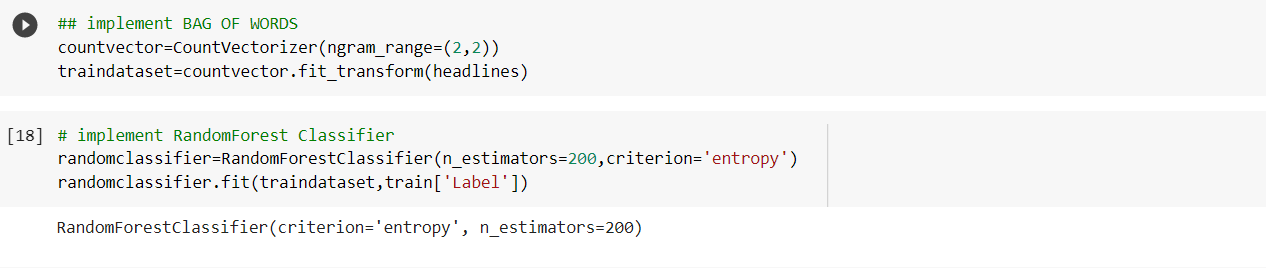
Data Cleaning:-



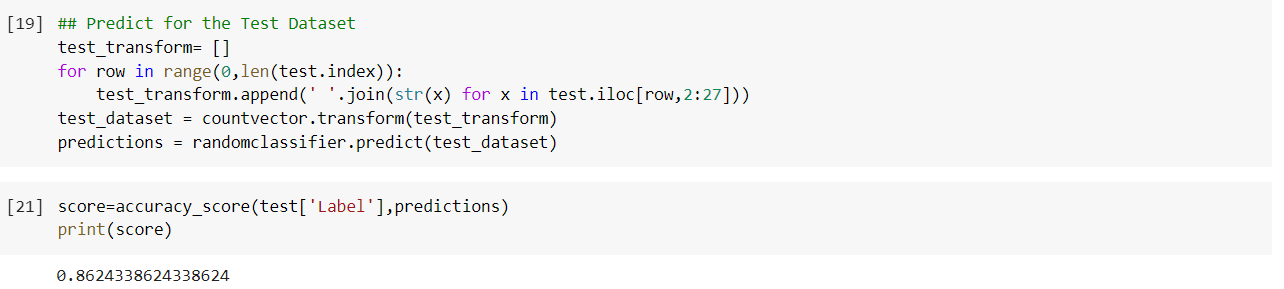
****

****

Fitting and training :-



Score or accuracy calculation:-



**Chapter-4**

**Limitations**

* Hardware Dependence:

1.Artificial Neural Networks require processors with parallel processing power, by their structure.

2. For this reason, the realization of the equipment is dependent.

* Assurance of proper network structure:

1. There is no specific rule for determining the structure of artificial neural networks.4
2. The appropriate network structure is achieved through experience and trial and error.

* The difficulty of showing the problem to the network:

1. ANNs can work with numerical information.
2. Problems have to be translated into numerical values before being introduced to ANN.
3. The display mechanism to be determined will directly influence the performance of the network.
4. This is dependent on the user's ability.

* The duration of the network is unknown:

1. The network is reduced to a certain value of the error on the sample means that the training has been completed.
2. The value does not give us optimum results.

* Time and space Complexity are higher
* Due to less hardware and GPU resources, sometimes we can’t able to perform training on dataset
* Due to less correlation in our dataset the predictions of the output has high loss and output should be wrong.
* Due to over fitting of the model, the predicted output should be wrong

**Chapter-5**

**Future Scope**

* We can try out with different set of neural network model and with different learning algorithms and also we can check results using different activation functions on our datasets which give better results.
* We can further increase the datasets, which helps to increase the accuracy of the model.
* We can apply various optimizations and normalizations to our datasets.
* We can fine tune hyper parameters.
* Effective preprocessing and clustering can improve neural network forecasting accuracy on sequential data.
* A bidirectional LSTM was the most effective method for the observed training data. Static features did not improve accuracy.

**Chapter-6**

**Conclusion**

The experiments in the research presented in this report were done in order to predict stock price for which dense neural network model is being used for the purpose. Several other architectures are also possible which can give better results. In the other experiment we headlines is being used to predict whether the stock price will rise or fall using the random forest classifier

In Deep Learning, especially neural networks, there is no thumb rule or a fixed specified rule through which one can decide how many convolutional, pooling layers are required. Therefore there was a requirement of comparing different layer models. All the models were following the trend of the actual values, and the RMSE obtained was in the range of 75-100 km/s. Other performance metrics have also been included along with each architecture.

Deep Learning has proved to be a game-changer especially in efficiently analyzing the previous data of the stock to predict out the future prices of stock as stock prices depend on further more parameters but with the deep learning and neural network model we can come up with a value which may give us some insights of particular stock prices. Similar approach can further be applied on various other use cases as deep learning can be used with any real-world scenario which can be converted to some numbers and to find out a pattern in it.

**Chapter-7**

**Bibliography and references**

* [**https://expertsystem.com/**](https://expertsystem.com/)
* [**https://www.geeksforgeeks.org/**](https://www.geeksforgeeks.org/)
* [**https://www.wikipedia.org/**](https://www.wikipedia.org/)
* [**https://www.coursera.org/learn/machine-learning**](https://www.coursera.org/learn/machine-learning)
* [**https://machinelearningmastery.com/**](https://machinelearningmastery.com/)
* [**https://towardsdatascience.com/machine-learning/home**](https://towardsdatascience.com/machine-learning/home)