

A

Major Project Report

On

**HUMAN HEALTH PREDICTION USING MACHINE LEARNING**

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

**DIPLOMA**

IN

**COMPUTER SCIENCE AND ENGINEERING**

By

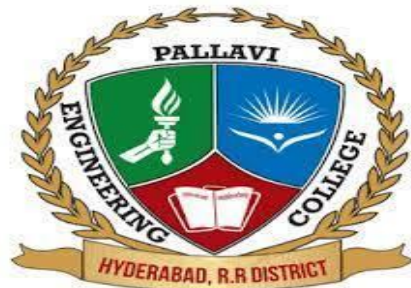
**P.SHIVA KUMAR REDDY**

**20502-CM-042**

Under the supervision of

**Mr.E.RAVI**

**Assistant Professor**



**Department of Computer Science and Engineering**  
**PALLAVI ENGINEERING COLLEGE**

Affiliated to JNTU Hyderabad, Approved by AICTE

Kuntloor, Hayathnagar (M).

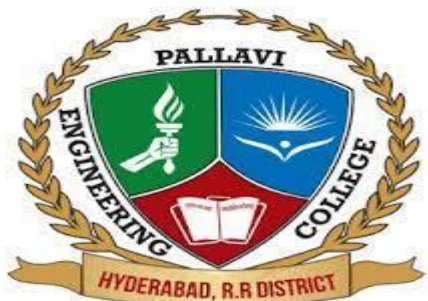
Ranga Reddy Dist. - 501505

TELANGANA STATE

**2022**

# **PALLAVI ENGINEERING COLLEGE**

Approved by AICTE, New Delhi, Affiliated to JNTU, Hyderabad  
Kuntloor, Hayathnagar (M), Hyderabad, R. R. Dist. – 501505, T.S



Department of Computer Science and Engineering

## **CERTIFICATE**

This is to Certify that the Project report entitled “ **HUMAN HEALTH PREDICTION USING MACHINE LEARNING** ” being submitted by **P.SHIVA KUMAR REDDY**, bearing Roll Number: **20502-CM-042** , in a partial fulfillment of the requirements for the award of the degree of **Diploma in COMPUTER SCIENCE AND ENGINEERING**, State Board Of Technical Education And Training, is a record of bonafide work carried out by them. The results presented in this have been verified and are found to be satisfactory. The results embodied in this have not been submitted to any other University for the award of any other degree or diploma.

**Internal Guide**  
**E.RAVI**

**B.Tech,M.Tech(Assistant Professor)**

**Head of the Department**  
**Mr. Erugu Krishna**

**B. Tech, M. Tech, (Ph. D)**

**Submitted for the viva voice examination held on** \_\_\_\_\_

**Internal Examiner**

**External Examiner**

## DECLARATION

Me **P.SHIVA KUMAR REDDY**, bearing **RollNo: 20502-CM-042**, hereby declare that the project report entitled “ **HUMAN HEALTH PREDICTION USING MACHINE LEARNING** ” is bonafide work done by us under the guidance of **Mr.E.Ravi Assistant Professor , Department of Computer Science And Engineering, Pallavi Engineering College**, during the year 2022-23. This Project work is submitted to SBTET in partial fulfillment of the requirements for the award of the degree of Diploma in **Computer Science and Engineering**. **This work has not been submitted for any other institute for any degree, diploma previously.**

**Date:**

**Place: Kuntloor**

**P.SHIVA KUMAR REDDY**

**20502-CM-042**

## ACKNOWLEDGEMENT

As endeavor of a long period can be successful only with the advice of our Parents and well-wishers. We now take this opportunity to express our deep gratitude and appreciation to all those who encouraged us for successful completion of the project work.

We are very much thankful to **Sri M. Komaraiah, Hon'ble Chairman, Pallavi Group of Institutions** for their help in providing good facilities in our college.

We wish to express our sincere gratitude to **E.RAVI Assistant Professor**. His consistent help and encouragement to complete the project work.

Our special thanks to **Mr. Erugu Krishna, Assistant Prof. & Head, Dept. of CSE, PALLAVI ENGINEERING COLLEGE** during the process of project work for his timely suggestions and help inspite of his busy schedule.

We are thankful to our guide **E.RAVI Assistant Professor, Dept. of CSE, PALLAVI ENGINEERING COLLEGE** for this valuable guidance and suggestions in analyzing and testing throughout the period, till end of project work completion.

We also extend our thanks to the entire Teaching and Non-Teaching faculty of PALLAVI ENGINEERING COLLEGE, who have encouraged us throughout the course of Master's degree. Last but not least, we thank our family and all those helped us directly or indirectly for the completion of the project.

**P.SHIVA KUMAR REDDY**  
**20502-CM-042**

# INDEX

## TABLE OF CONTENTS

<b>1. Abstract</b>	<b>i</b>
<b>2. Acknowledgment</b>	<b>iv</b>
<b>3. Introduction</b>	<b>ii</b>
<b>4. Python Introduction</b>	<b>ii-iv</b>
<b>5. Existing System</b>	<b>v</b>
<b>6. Proposed System</b>	<b>vi</b>
<b>7. Methodology</b>	<b>vii</b>
<b>8. Algorithm Technics</b>	<b>viii-ix</b>
<b>9. Implementation</b>	<b>x-xxviii</b>
<b>10. Results</b>	<b>xxix</b>
<b>11. Conlusion</b>	<b>xxx</b>

## ABSTRACT

Disease Prediction using Machine Learning is the system that is used to predict the diseases from the symptoms which are given by the patients or any user. The system processes the symptoms provided by the user as input and gives the output as the probability of the disease. Nave Bayes classifier is used in the prediction of the disease which is a supervised machine learning algorithm. The probability of the disease is calculated by the Nave Bayes algorithm. With an increase in biomedical and healthcare data, accurate analysis of medical data benefits early disease detection and patient care. By using linear regression and decision tree we are predicting diseases like Diabetes, Malaria, Jaundice, Dengue, and Tuberculosis.

**Keywords:** Disease Prediction, Machine learning, Naive Bayes algorithm

# I. INTRODUCTION

Machine Learning is the domain that uses past data for predicting. Machine Learning is the understanding of computer system under which the Machine Learning model learn from data and experience. The Machine Learning algorithm has two phases: 1) Training & 2) Testing. To predict the disease from a patient's symptoms and from the history of the patient, machine learning technology is struggling from past decades. Healthcare issues can be solved efficiently by using Machine Learning Technology. We are applying complete machine learning concepts to keep the track of patient's health.

ML model allows us to build models to get quickly cleaned and processed data and deliver results faster. By using this system doctors will make good decisions related to patient diagnoses and according to that, good treatment will be given to the patient, which increases improvement in patient healthcare services. To introduce machine learning in the medical field, healthcare is the prime example. To improve the accuracy of large data, the existing work will be done on unstructured or textual data. For the prediction of diseases, the existing will be done on linear, KNN, Decision Tree algorithm.

## 1.1 Python introduction:

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python Web site, <https://www.python.org/>, and may be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation.

The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

This tutorial introduces the reader informally to the basic concepts and features of the Python language and system. It helps to have a Python interpreter handy for hands-on experience, but all examples are self-contained, so the tutorial can be read off-line as well.

For a description of standard objects and modules, see `library-index`. `reference-index` gives a more formal definition of the language. To write extensions in C or C++, read `extending-index` and `c-api-index`. There are also several books covering Python in depth.

This tutorial does not attempt to be comprehensive and cover every single feature, or even every commonly used feature. Instead, it introduces many of Python's most noteworthy features, and will give you a good idea of the language's flavor and style. After reading it, you will be able to

read and write Python modules and programs, and you will be ready to learn more about the various Python library modules described in `library-index`.

Python is just the language for you.

You could write a Unix shell script or Windows batch files for some of these tasks, but shell scripts are best at moving around files and changing text data, not well-suited for GUI applications or games. You could write a C/C++/Java program, but it can take a lot of development time to get even a first-draft program. Python is simpler to use, available on Windows, Mac OS X, and Unix operating systems, and will help you get the job done more quickly.

Python is simple to use, but it is a real programming language, offering much more structure and support for large programs than shell scripts or batch files can offer. On the other hand, Python also offers much more error checking than C, and, being a *very-high-level language*, it has high-level data types built in, such as flexible arrays and dictionaries. Because of its more general data types Python is applicable to a much larger problem domain than Awk or even Perl, yet many things are at least as easy in Python as in those languages.

Python allows you to split your program into modules that can be reused in other Python programs. It comes with a large collection of standard modules that you can use as the basis of your programs — or as examples to start learning to program in Python. Some of these modules provide things like file I/O, system calls, sockets, and even interfaces to graphical user interface toolkits like Tk.

Python is an interpreted language, which can save you considerable time during program development because no compilation and linking is necessary. The interpreter can be used interactively, which makes it easy to experiment with features of the language, to write throw-away programs, or to test functions during bottom-up program development. It is also a handy desk calculator.

Python enables programs to be written compactly and readably. Programs written in Python are typically much shorter than equivalent C, C++, or Java programs, for several reasons:

- the high-level data types allow you to express complex operations in a single statement;
- statement grouping is done by indentation instead of beginning and ending brackets;
- no variable or argument declarations are necessary.

Python is *extensible*: if you know how to program in C it is easy to add a new built-in function or module to the interpreter, either to perform critical operations at maximum speed, or to link Python programs to libraries that may only be available in binary form (such as a vendor-specific graphics library). Once you are really hooked, you can link the Python interpreter into an application written in C and use it as an extension or command language for that application.

By the way, the language is named after the BBC show “Monty Python’s Flying Circus” and has nothing to do with reptiles. Making references to Monty Python skits in documentation is not only allowed, it is encouraged!



Now that you are all excited about Python, you'll want to examine it in some more detail. Since the best way to learn a language is to use it, the tutorial invites you to play with the Python interpreter as you read.

In the next chapter, the mechanics of using the interpreter are explained. This is rather mundane information, but essential for trying out the examples shown later.

The rest of the tutorial introduces various features of the Python language and system through examples, beginning with simple expressions, statements and data types, through functions and modules, and finally touching upon advanced concepts like exceptions and user-defined

## II. EXISTING SYSTEM

The existing system predicts the chronic diseases which are for a particular region and for the particular community. Only particular diseases are predicted by this system. In this System, Big Data & CNN Algorithm is used for Disease risk prediction. For S type data, the system is using Machine Learning algorithm i.e Nearest Neighbors, Decision Tree, Nave Bayesian. The accuracy of the existing System is up to 94.8%. In the existing paper, they streamline machine learning algorithms for the effective prediction of chronic disease outbreak in disease-frequent communities. They experiment with the modified prediction models over real life hospital data collected from central China. They propose a convolution neural network-based multi modal disease risk prediction(CNN-MDRP) algorithm using structured and unstructured data from the hospital.

### **III.PROPOSED SYSTEM**

Most of the chronic diseases are predicted by our system. It accepts the structured type of data as input to the machine learning model. This system is used by end-users i.e. patients/any user. In this system, the user will enter all the symptoms from which he or she is suffering. These symptoms then will be given to the machine learning model to predict the disease. Algorithms are then applied to which gives the best accuracy. Then System will predict disease on the basis of symptoms. This system uses Machine Learning Technology. Nave Bayes algorithm is used for predicting the disease by using symptoms, for classification KNN algorithm is used, Logistic regression is used for extracting features which are having most impact value, the Decision tree is used to divide the big dataset into smaller parts. The final output of this system will be the disease predicted by the model

## IV. METHODOLOGY

To calculate performance evaluation in the experiment, first, we denote TP, TN, Fp and FNias true positive(the number of results correctly predicted as required), true negative (the number of results not required), false positive (the number of results incorrectly predicted as required), false negative(the number of results incorrectly predicted as not required)respectively. We can obtain four measurements: recall, precision, accuracy, and F1 measures as follows:

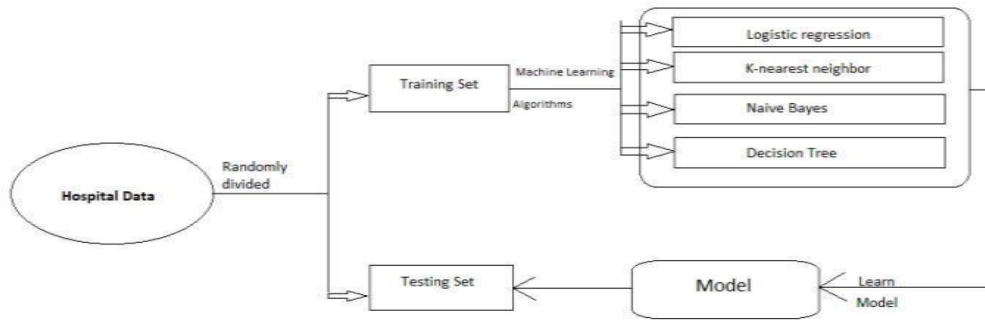
**Accuracy-:**

$$\text{Accuracy} = \frac{\text{TruePositive} + \text{TrueNegative}}{\text{TruePositive} + \text{TrueNegative} + \text{FalsePositive} + \text{FalseNegative}}$$

$$\text{Precision} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalsePositive}}$$

$$\text{Recall} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalseNegative}}$$

$$\text{F1-Measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$



**Fig -1: System Architecture**

## V. ALGORITHM TECHNIQUES

KNN K Nearest Neighbour (KNN) could be terribly easy, simple to grasp, versatile and one amongst the uppermost machine learning algorithms. In the Healthcare System, the user will predict the disease. In this system, the user can predict whether the disease will detect or not. In the proposed system, classifying disease in various classes that shows which disease will happen on the basis of symptoms. KNN rule used for each classification and regression issue. KNN algorithm is based on feature similarity approach. It is the best choice for addressing some of the classification related tasks. K-nearest neighbor classifier algorithm is to predict the target label of a new instance by defining the nearest neighbor class. The closest class will be identified using distance measures like Euclidean distance. If  $K = 1$ , then the case is just assigned to the category of its nearest neighbor.

The value of 'k' has to be specified by the user and the best choice depends on the data. The larger value of 'k' reduces the noise on the classification. If the new feature i.e in our case symptom has to classify, then the distance is calculated and then the class of feature is selected which is nearest to the newer instance. In the instance of categorical variables, the Hamming distance must be used. It conjointly brings up the difficulty of standardization of the numerical variables between zero and one once there's a combination of numerical and categorical variables within the dataset

### ***NAIVE BAYES:***

Naive Bayes is an easy however amazingly powerful rule for prognosticative modeling. The independence assumption that allows decomposing joint likelihood into a product of marginal likelihoods is called as 'naive'. This simplified Bayesian classifier is called as naive Bayes. The Naive Bayes classifier assumes the presence of a particular feature in a class is unrelated to the presence of any other feature. It is very easy to build and useful for large datasets. Naive Bayes is a supervised learning model. Bayes theorem provides some way of calculative posterior chance  $P(b|a)$  from  $P(b)$ ,  $P(a)$  and  $P(a|b)$ . Look at the equation below:

$$P(b \vee a) = P(a \vee b)P(b)/P(a)$$

Above,

- $P(b|a)$  is that the posterior chance of class (b,target) given predictor (a, attributes).  $\cdot P(b)$  is the prior probability of class.
  - $P(a|b)$  is that chance that is that the chance of predictor given class.
  - $P(a)$  is the prior probability of predictor. In our system, Naïve Bayes decides which symptom is to put in classifier and which is not.
- ### 8.3 LOGISTIC REGRESSION
- Logistic regression could be a supervised learning classification algorithm accustomed to predict the chance of a target variable that is Disease.

### ***DECISION TREE***

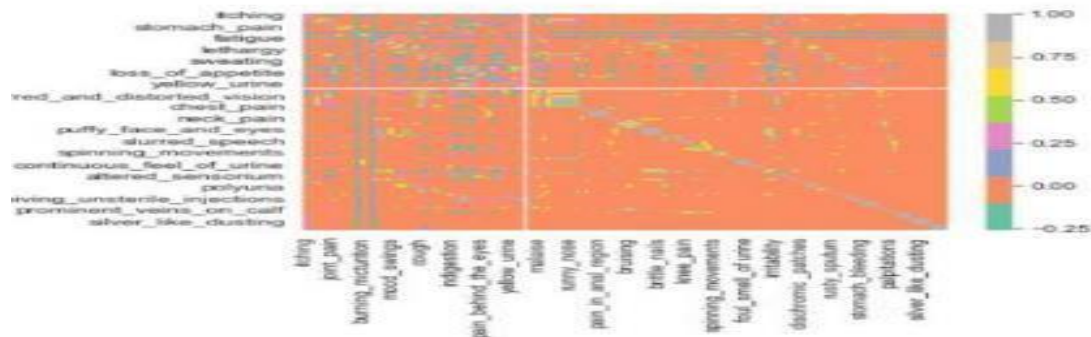
A decision tree is a structure that can be used to divide up a large collection of records into successfully smaller sets of records by applying a sequence of simple decision tree. With each

successive division, the members of the resulting sets become more and more similar to each other. A decision tree model consists of a set of rules for dividing a large heterogeneous population into smaller, more homogeneous (mutually exclusive) groups with respect to a particular target. The target variable is usually categorical and the decision tree is used either to:

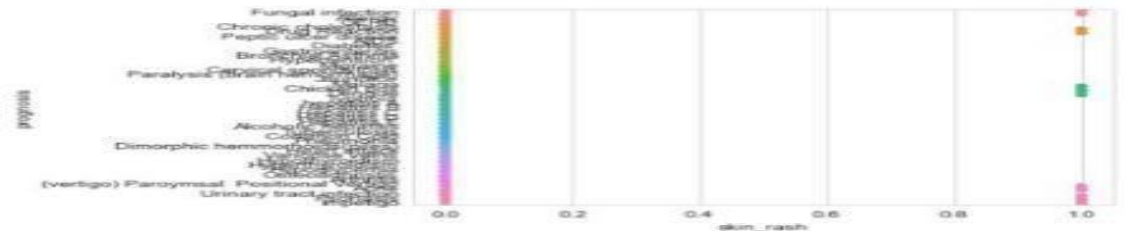
- Calculate the probability that a given record belong to each of the category and,
- To classify the record by assigning it to the most likely class (or category). In this disease prediction system, decision tree divides the symptoms as per its category and reduces the dataset difficulty

Graphs:

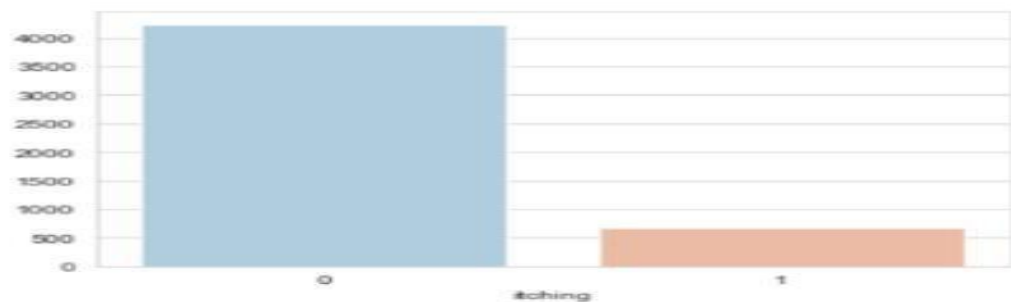
### PREDICTION GRAPH



### Heat-Map



### Swarm Plot



### Count Plot

Fig 2: Graph (Prediction,Swarm,Count)

## VI. IMPLEMENTATION

### Python Programming with Google Colab



In this article, we will learn to practice Python programming using Google colab. We will discuss collaborative programming, automatic setting-up, getting help effectively. Google Colab is a suitable tool for Python beginners.

#### Introduction

Google Colab is the best project from Google Research. It is an open-source, **Jupyter based environment**. It helps us write and execute Python based code, other Python-based third-party tools and machine learning frameworks such as [Python](#), [PyTorch](#), [Tensorflow](#), [Keras](#), [OpenCV](#) and many others. It runs on the web-browser.

Google Colab is an open-source platform where we can write and Python execute code. It requires any source on the internet. We can also mount it into [Google Drive](#). Google Colab requires no configuration to get started and provides free access to GPUs. It facilitates us to share live code, mathematical equations, data visualization, data cleaning and transformation, machine learning models, numerical simulations, and many others.

#### Advantages of Using Google Colab

Google Colab provides many features. These features are given below.

- It comes with pre-installed packages.
- We don't need to install Python or its package.
- It runs on the web-browser.
- It allows us to work with web-browser Jupyter notebooks.
- It is an open-source Google platform, which means anyone can use it free of cost.
- It offers GPU and TPU power.
- It supports both Python versions 2 and 3.
- It provides two hardware accelerations - GPU (Graphical Processing Unit) and TPU (Tensor Processing Unit).
- It provides a free Jupyter notebook environment.

#### What are GPUs and TPUs in Google Colab?

The reason for the popularity of Google Colab is that it provides free GPUs and TPUs. [Machine learning and deep learning](#) training model takes numerous hours on a CPU. We have faced all these issues in our local machine. With GPUs and TPUs, we can train models in a matter of minutes or seconds.

Machine learning enthusiasts always prefer [GPU](#) over [CPU](#) because of the absolute power and speed of execution. But GPUs are much expensive; not everyone can afford them. That's why Google Colab comes into the scenario. Google Colab is completely free and can continuously run for 12 hours. It is sufficient to meet all requirements of the [machine learning](#) and [deep learning](#) projects.

### Difference between GPU and CPU

A few important features are given below.

GPU	CPU
It has hundreds of simpler cores.	It has very complex cores.
It provides thousands of concurrent hardware threads.	It provides single-thread performance optimization.
It exploits floating-point throughput.	It has a transistor space dedicated to complex ILP.
Up to Tesla K80 with 12 GB of GDDR5 VRAM, Intel Xeon Processor with two cores @ 2.20 GHz and 13 GB RAM	Intel Xeon Processor with two cores @ 2.30 GHz and 13 GB RAM.
It provides most die surface for integer and fp units.	It is available with a few die surfaces for integer and fp units.

### Which GPU is used today?

Let's take two scenarios before using the GPU for deep learning.

#### First Scenario:

First, we need to determine our requirements regarding what kind of resources we need to accomplish our tasks. If our machine learning task is small or can be fit in complex sequential processing, we don't need a big system to work on GPU. It means if we are working on the other ML area/algorithm. We don't require using GPU.

#### Second Scenario:

If our task is much extensive and has handle-able data then we can use the GPU hardware acceleration. It enhances the execution speed and provides output within seconds or minutes.

### Start Working with Google Colab

To start working with Google Colab, we need to type the URL on the web browser below. It will launch the following window opens with a popup offering many features.



ExamplesRecentGoogle DriveGitHubUpload

Filter notebooks

Title

First opened

Last opened

Welcome To Colaboratory
 

Jul 16, 20200 minutes ago

Untitled0.ipynb
 

Jul 16, 2020Jul 31, 2020

Welcome To Colaboratory
 

Sep 28, 2019Oct 7, 2019

NEW NOTEBOOKCANCEL

It provides few options to create notebook as well as upload and select from various sources such as - Google Drive, Local computer, GitHub.

Click on the NEW NOTEBOOK button to create new a Colab notebook.

ExamplesRecentGoogle DriveGitHubUpload

Filter notebooks

Title

First opened

Last opened

Welcome To Colaboratory
 

Jul 16, 20200 minutes ago

Untitled0.ipynb
 

Jul 16, 2020Jul 31, 2020

Welcome To Colaboratory
 

Sep 28, 2019Oct 7, 2019

NEW NOTEBOOKCANCEL

GPU is the best choice. GPUs are much suitable for intensive task.

## Uploading File using Goggle Drive

We can upload **GitHub's** project and search it to get the code.

Similarly, we can upload the code directly from Google drive. We can also filter saved notebooks by name, date, owner, or modified date.

It will automatically upload the code in Jupyter Colab.

ExamplesRecentGoogle DriveGitHubUpload

Filter notebooks

Title

Owner

Last modified

Last opened

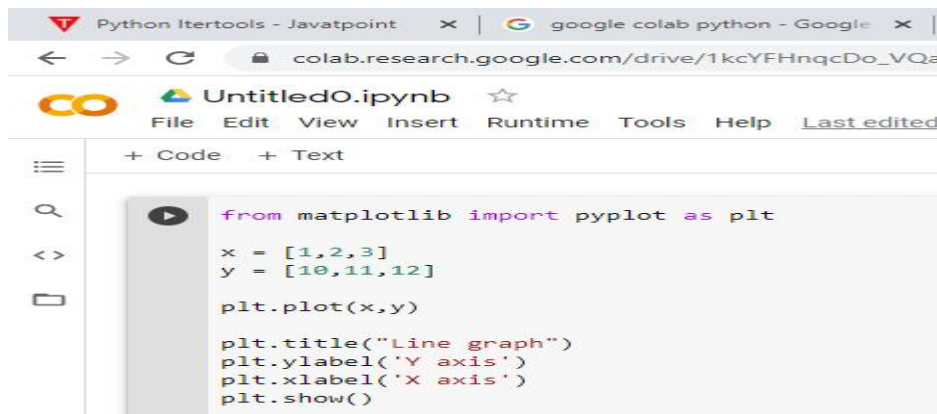
Untitled0.ipynb
 

DEVANSH SHARMA

Jul 31, 2020Jul 31, 2020

NEW NOTEBOOKCANCEL

xii



```
from matplotlib import pyplot as plt

x = [1,2,3]
y = [10,11,12]

plt.plot(x,y)

plt.title("Line graph")
plt.ylabel('Y axis')
plt.xlabel('X axis')
plt.show()
```

## Cloning Repositories in Google Colab

We can also create a [Git repository](#) inside the Google Colab. We just need to visit any [GitHub](#) repository and copy the clone link of the repository.

**Step - 1:** Find the [Github](#) repo and get the "[Git](#)" link.

For example - We are using <https://github.com/rajk9200/construction.git>

**Clone or download > Copy the link**

**Step - 2:** Simply, run the following command.

1. `!git clone https://github.com/rajk9200/construction.git`

After hitting enter, it will start cloning.

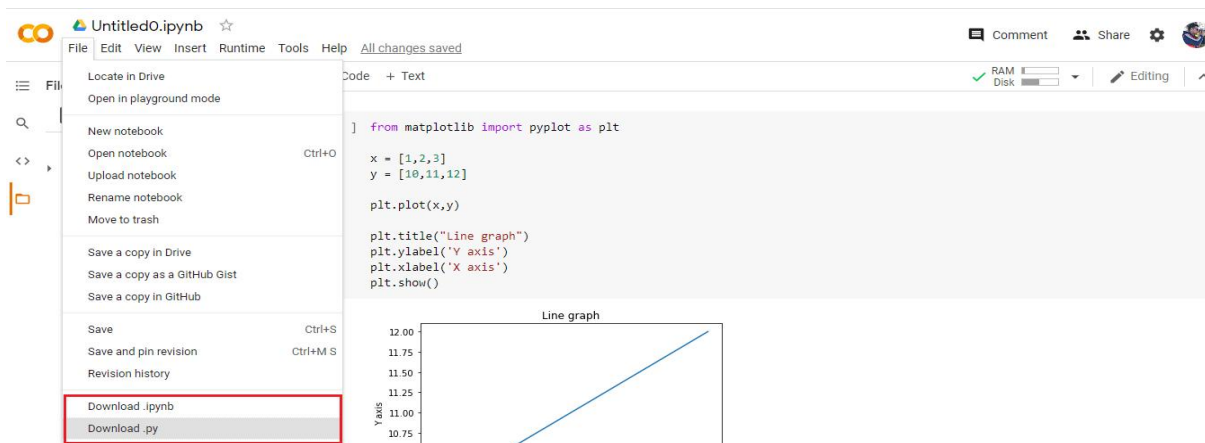
1. Cloning into 'construction'...
2. remote: Enumerating objects: 86, done.
3. remote: Counting objects: 100% (86/86), done.
4. remote: Compressing objects: 100% (70/70), done.
5. remote: Total 86 (delta 7), reused 86 (delta 7), pack-reused 0
6. Unpacking objects: 100% (86/86), done.

**Step - 3:** Open Folder in Google Drive.

**Step - 4:** Open a Notebook and run the Github repo in Google Colab.

## Saving Colab Notebook

All the notebooks are saved in the Google Drive automatically after a certain period. Hence we don't lose our working process. However, we can save our notebook \*.py and \*.ipynb formats explicitly.



## Setting up Hardware Accelerator GPU for Runtime

Google Colab offers a free cloud service with a GPU hardware accelerator. For the machine learning and deep learning, it requires very expensive GPU machine. GPU is most essential for doing multiple computations.

### Why GPU is important for machine learning

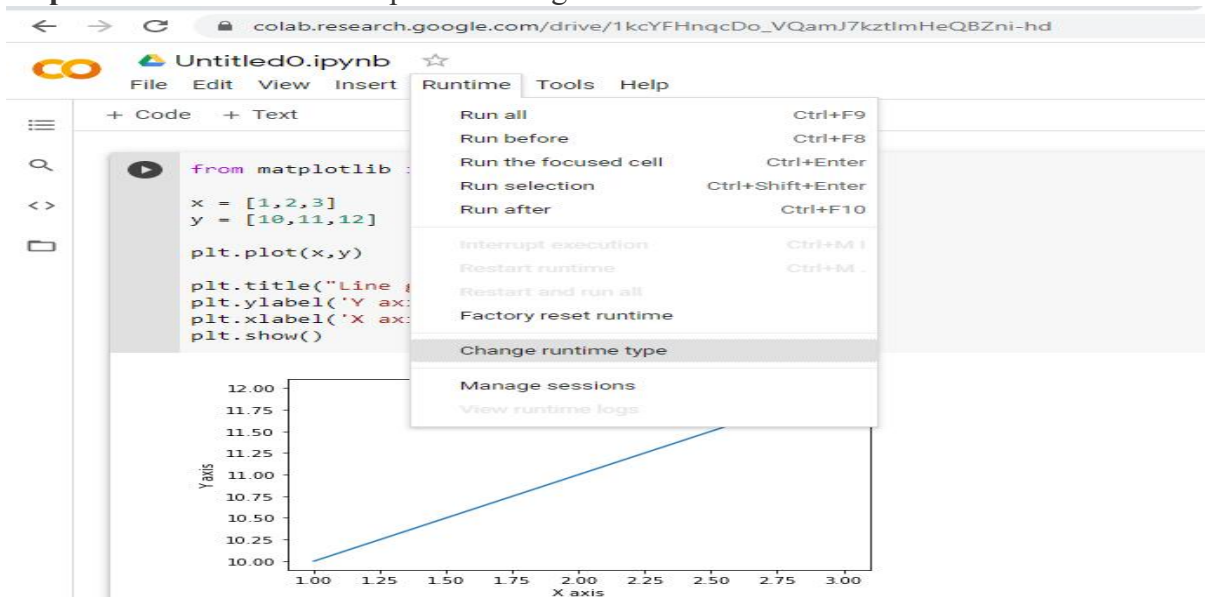
Nowadays, GPU has become the most dominant part of machine learning and deep learning. It provides the optimized capability of more compute-intensive workloads and streaming memory loads.

Another reason of its popularity, it can launch millions of threads in one call. They function unusually better than CPUs although it has a lower clock speed and it also lacks of many core management features compared to a CPU.

### Setup Hardware Accelerator GPU in Colab

Let's follow the given step to setup GPU.

**Step - 1:** Go to the Runtime option in Google Colab.



**Step - 2:** It will open the following popup screen change **None** to **GPU**. We can also select TPU according to our requirements by following the same process.

## Notebook settings

Hardware accelerator

GPU ?

To get the most out of Colab, avoid using a GPU unless you need one. [Learn more](#)

☐ Omit code cell output when saving this notebook

CANCEL SAVE

**Step - 3:** Now, we will check the details about the GPU in Colab. Type the following code to import the important packages.

1. `import tensorflow as tf`
2. `from tensorflow.python.client import device_lib`

Check the GPU accelerator

1. `tf.test.gpu_device_name()`

### Output:

```
/device:GPU:0
```

Now, we will check the hardware used for GPU.

1. `device_lib.list_local_devices()`

### Output:

```
[name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality {
}
incarnation: 11369748053613106705, name: "/device:XLA_CPU:0"
device_type: "XLA_CPU"
memory_limit: 17179869184
locality {
}
incarnation: 514620808292544972
physical_device_desc: "device: XLA_CPU device", name: "/device:XLA_GPU:0"
device_type: "XLA_GPU"
memory_limit: 17179869184
locality {
}
incarnation: 15275652847823456943
physical_device_desc: "device: XLA_GPU device", name: "/device:GPU:0"
device_type: "GPU"
memory_limit: 14640891840
locality {
  bus_id: 1
  links {
```

```
}  
}  
incarnation: 1942537478599293460  
physical_device_desc: "device: 0, name: Tesla T4, pci bus id: 0000:00:04.0, compute capability: 7.5"]
```

## Using Terminal Commands on Google Colab

We can also use Colab cell for running terminal commands. In the Google Colab, Python libraries such as [Pandas](#), [Numpy](#), scikit-learn.

If we want to install another library of Python using the following command.

1. `!pip install library_name`

It is quite easy to run the command. Everything is similar to the regular terminal. We just need to put an exclamation (!) before writing each command as below.

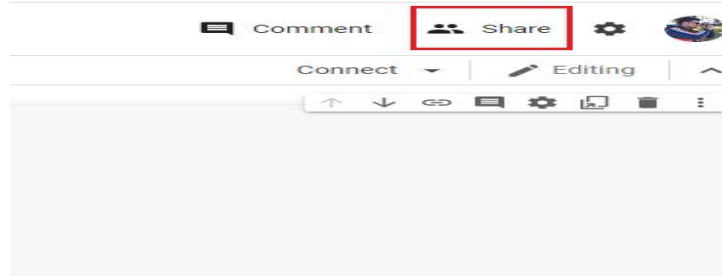
1. `!ls`

Or

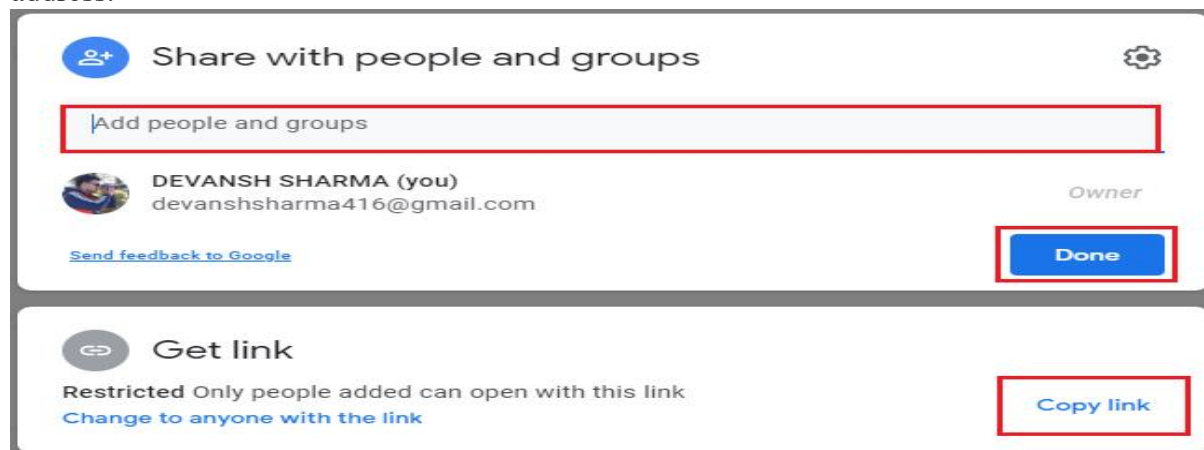
1. `!pwd`

## Sharing Our Notebook

We can share our Colab notebook with others. It is the best way to interact with other data science expert. It helps to share our code same as sharing a Google Doc or Google sheet.



We need to click on the share button. It will show the option of creating a shareable link that we can share through any platform. There is also an option to invite the people through the email address.



It is one of the outstanding features of Google Colab.

## Few Important Colab Commands

Colab provides some amazing commands. It offers various commands that facilitate us to perform operations in short. These commands are used with a **%** prefix. List of all magic commands are given below.

### 1. %lsmagic

#### Output:

Available line magics:

```
%alias %alias_magic %autocall %automagic %autosave %bookmark %cat %cd %clear %colors %config %connect_info %cp %debug %dhist %dirs
```

```
%doctest_mode %ed %edit %env %gui %hist %history %killbgscripts %ldir %less %lf %lk %ll %load %load_ext %loadpy %logoff %logon %logstart %logstate %logstop %ls %lsmagic %lx %macro %magic %man %matplotlib %mkdir %more %mv %notebook %page %pastebin %pdb %pdef %pdoc %pfile %pinfo %pinfo2 %pip %popd %pprint %precision %profile %prun %psearch %psource %pushd %pwd %pycat %pylab %qtconsole %quickref %recall %rehashx %reload_ext %rep %rerun %reset
```

```
%reset_selective %rm %rmdir %run %save %sc %set_env %shell %store %sx %system %tb %tensorflow_version %time %timeit %unalias %unload_ext %who %who_ls %whos %xdel %xmode
```

Available cell magics:

```
%%! %%HTML %%SVG %%bash %%bigquery %%capture %%debug %%file %%html %%javascript %%js %%latex %%perl %%prun %%pypy %%python %%python2 %%python3 %%ruby %%script %%sh %%shell %%svg %%sx %%system %%time %%timeit %%writefile
```

Automagic is ON, % prefix IS NOT needed for line magics.

#### Command for List of Local Directories

### 1. %ldir

#### Output:

```
drwxr-xr-x 1 root 4096 Dec 2 22:04 sample_data/
```

#### Command for getting Notebook History

### 1. %history

#### Output:

```
%lsmagic
```

```
%ldir
```

```
%history
```

#### Getting CPU Time

### 1. %time

#### Output:

```
CPU times: user 4 µs, sys: 0 ns, total: 4 µs
```

```
Wall time: 7.15 µs
```

#### Getting the how long has the system has been running

1. %uptime

### Output:

```
08:44:32 up 12 min, 0 users, load average: 0.00, 0.03, 0.03
```

### Display available and used memory

1. !free -hprint("-"\*100)

### Output:

```
/bin/bash: -c: line 0: syntax error near unexpected token `('
/bin/bash: -c: line 0: `free -hprint("-"*100)'
```

### Display the CPU specification

1. !lscpu  
2. print("-"\*70)

### Output:

```
Architecture:      x86_64
CPU op-mode(s):    32-bit, 64-bit
Byte Order:        Little Endian
CPU(s):            2
On-line CPU(s) list: 0,1
Thread(s) per core: 2
Core(s) per socket: 1
Socket(s):         1
NUMA node(s):      1
Vendor ID:         GenuineIntel
CPU family:        6
Model:             63
Model name:        Intel(R) Xeon(R) CPU @ 2.30GHz
Stepping:          0
CPU MHz:           2300.000
BogoMIPS:          4600.00
Hypervisor vendor: KVM
Virtualization type: full
L1d cache:         32K
L1i cache:         32K
L2 cache:          256K
L3 cache:          46080K
NUMA node0 CPU(s): 0,1
Flags:             fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2
ss ht syscall nx pdpe1gb rdtscp lm constant_tsc rep_good nopl xtopology nonstop_tsc cpuid tsc_known_freq pni
pclmulqdq ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt aes xsave avx f16c rdrand hypervisor lahf_lm
abm invpcid_single ssbd ibrs ibpb stibp fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid xsaveopt arat
md clear arch capabilities
```

### Getting the List of All running VM processes

1. %%shecho "List all running VM processes."  
2. ps -ef  
3. echo "Done"

### Output:

```
List all running VM processes.
```

```
Done
```

```
error: garbage option
```

```
Usage:
```

```
ps [options]
```

```
Try 'ps --help <simple|list|output|threads|misc|all>'
```

```
or 'ps --help <s|l|o|m|a>'
```

```
for additional help text.
```

```
For more details see ps(1).
```

## How to Design Form in Google Colab

Google Colab provides the facility to design forms. Let's see the following code to understand how to design form in Google Colab. In this section, we will design a form takes the student information.

```
1.      # @ title student details
2.      # @ mark down information
3.
4.      Name = "" #@param {type:"string"}
5.      Age = 0#@param {type:"number"}
6.      Course = "" #@param {type:"string"}
7.
8.      Gender = 'Male' #@param ["Male", "Female", "Others"]
9.
10.     Date_of_Birth = '2000-03-22' #@param {type:"date"}
11.
12.     #@markdown ---print("Submitting the form")
13.
14.     print("Submitted")
```

Output:



## Graph Plotting

We can plot various graphs in Colab for the data visualization, as well. In the following example, the graph will show a plot containing more than one polynomial,  $Y = x^3 + x^2 + x[3]$ . Let's see the following code.

**Example -**

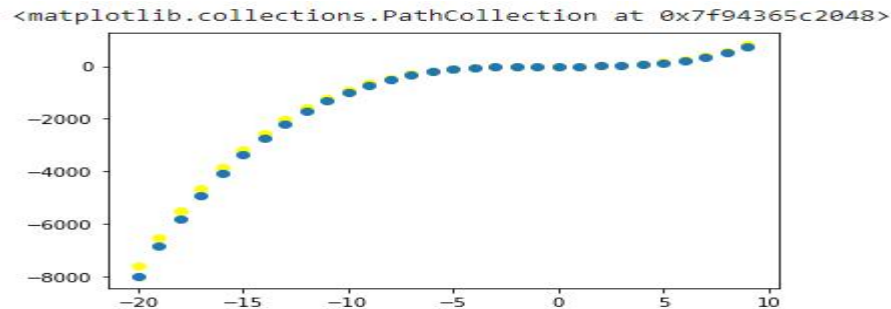


```

1. import numpy as np
2.
3. import matplotlib.pyplot as plt
4. x = np.arange(-20,10)
5. y = np.power(x,3)
6. y1 = np.power(x,3) + np.power(x,2) + x
7. plt.scatter(x,y1,c="yellow")
8. plt.scatter(x,y)

```

**Output:**



The below code will draw the heat map.

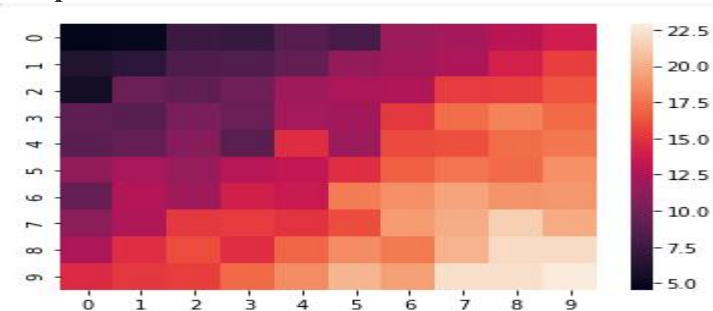
**Example -**

```

1. import matplotlib.pyplot as plt
2. import numpy as np
3. import seaborn as sns
4. length = 10
5. data = 5 + np.random.randn(length, length)
6. data += np.arange(length)
7. data += np.reshape(np.arange(length), (length, 1))
8. sns.heatmap(data)
9. plt.show()

```

**Output:**



**TPU (Tensor Processing Unit) in Google Colab**

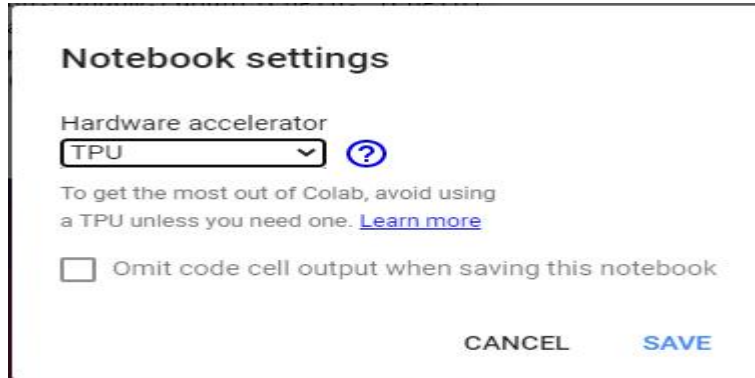
The TPU (Tensor Processing Unit) is used to accelerate on a **Tensorflow** graphs. They are based on the [AI](#) acceleration application-specification integrated circuit which is specially designed for the [neural network](#) machine. It is developed by Google.

TPU consists of an excellent configuration of teraflops, floating-point performance, and others. There are 180 teraflops of floating-point performance in each TPU. Generally, a teraflop is the measurement of the computer's speed. Its speed can be a trillion floating-point operations per second.

### How to set up a TPU in Google Colab?

The steps are the same as the setup of a GPU.

- Runtime -----> Change runtime



### Check Running on TPU Hardware Accelerator

To check the TPU hardware acceleration, it requires the Tensorflow package. Consider the below code.

1. `import tensorflow as tf`
- 2.
3. `tpu = tf.distribute.cluster_resolver.TPUClusterResolver()`
4. `print('Running on TPU ', tpu.cluster_spec().as_dict()['worker'])`

```
Running on TPU ['10.43.45.130:8470']
```

### Conclusion

Google Colab is a [Jupyter notebook](#) developed by Google Research. It is used to execute the Python-based code to build a machine learning or deep learning model. It provides the GPU and TPU hardware acceleration and available for free (unless we would like to go pro).

It is quite easy to use and share due to the zero-configuration features requirement.

We can combine the executable code with and [HTML](#), images, [Latex](#), and others. One of the best thing, it has included a vital machine learning library like **TensorFlow**, already installed. It is a perfect tool for machine learning and deep learning model building.

Colab is outstanding for developing neural networks. The parallelism and execution of multiple threads can be achieved by the CPU based hardware accelerator.

If we use the [IDE](#) for machine learning and deep learning model, it will take Nemours hour to execute code. But with the help of the Google Colab, it can be done within few seconds or minutes. It provides the facility to import data from Github and Google drive.

### LIBRARIES USED :

**NUMPY:**

Numpy stands for numerical python. As the name gave it away, it's an open source library for the Python programming language.

I hear you thinking: "another library..." but no such thing is true! Numpy is one of the most useful libraries especially if you're crunching numbers .

### **Purpose of numpy:**

Numpy adds support for large, multi-dimensional matrices and arrays, along with a gigantic collection of top-end mathematical functions to operate on these arrays and matrices

It's objective is to make it easier for you to transform difficult functions or calculate some data analysis Numpy's biggest advantage is its fastness. It's so much faster than using the built-in Python's functions.

For example, it lets you simply calculate the mean and median of a dataframe with a plain line of code for each:

### **How to install by using Anaconda Prompt**

First off you need to install Numpy but only if you're not using Anaconda. To do so

- Pip install numpy
- Conda install numpy

### **How to import numpy**

- Import numpy
- Import numpy as np
- from numpy import \*
- from numpy import add, subtract

### **NumPy Array Slicing:**

Slicing in python means taking elements from one given index to another given index. We pass slice instead of index like this: `[start:end]`. We can also define the step, like this: `[start:end:step]`. If we don't pass start its considered 0. If we don't pass end its considered length of array in that dimension. If we don't pass step its considered 1. Negative Slicing: Use the minus operator to refer to an index from the end: Use the step value to determine the step of the slicing.

### **NumPy Data Types:**

By default Python have these data types:

- strings - used to represent text data, the text is given under quote marks. e.g. "ABCD"
- integer - used to represent integer numbers. e.g. -1, -2, -3
- float - used to represent real numbers. e.g. 1.2, 42.42
- boolean - used to represent True or False.
- complex - used to represent complex numbers. e.g.  $1.0 + 2.0j$ ,  $1.5 + 2.5j$

### **Data Types in NumPy:**

NumPy has some extra data types, and refer to data types with one character, like i for integers, u for unsigned integers etc. Below is a list of all data types in NumPy and the characters used to represent them.

- i - integer
- b - boolean
- u - unsigned integer
- f - float
- c - complex float
- m - timedelta
- M - datetime
- O - object
- S - string
- U - unicode string
- V - fixed chunk of memory for other type ( void )

### **Checking the Data Type of an Array:**

The NumPy array object has a property called dtype that returns the data type of the array.

### **Creating Arrays With a Defined Data Type:**

We use the array() function to create arrays, this function can take an optional argument: dtype that allows us to define the expected data type of the array elements:

### **What if a Value Can Not Be Converted:**

If a type is given in which elements can't be casted then NumPy will raise a ValueError.

### **Converting Data Type on Existing Arrays:**

The best way to change the data type of an existing array, is to make a copy of the array with the astype() method. The astype() function creates a copy of the array, and allows you to specify the data type as a parameter. The data type can be specified using a string, like 'f' for float, 'i' for integer etc. or you can use the data type directly like float for float and int for integer.

### **NumPy Array Copy vs View:**

The main difference between a copy and a view of an array is that the copy is a new array, and the view is just a view of the original array. The copy *owns* the data and any changes made to the copy will not affect original array, and any changes made to the original array will not affect the copy. The view *does not own* the data and any changes made to the view will affect the original array, and any changes made to the original array will affect the view.

### **Check if Array Owns its Data:**

As mentioned above, copies *owns* the data, and views *does not own* the data, but how can we check this? Every NumPy array has the attribute base that returns None if the array owns the data. Otherwise, the base attribute refers to the original object.

### **NumPy Array Shape:**

The shape of an array is the number of elements in each dimension. Get the Shape of an Array: NumPy arrays have an attribute called shape that returns a tuple with each index having the number of corresponding elements.

### **What does the shape tuple represent:**

Integers at every index tells about the number of elements the corresponding dimension has.

### **NumPy Array Reshaping:**

Reshaping means changing the shape of an array. The shape of an array is the number of elements in each dimension. By reshaping we can add or remove dimensions or change number of elements in each dimension.

### **Can We Reshape Into any Shape:**

Yes, as long as the elements required for reshaping are equal in both shapes. We can reshape an 8 elements 1D array into 4 elements in 2 rows 2D array but we cannot reshape it into a 3 elements 3 rows 2D array as that would require  $3 \times 3 = 9$  elements.

### **Unknown Dimension:**

You are allowed to have one "unknown" dimension. Meaning that you do not have to specify an exact number for one of the dimensions in the reshape method. Pass -1 as the value, and NumPy will calculate this number for you.

### **Flattening the arrays:**

Flattening array means converting a multidimensional array into a 1D array. We can use `reshape(-1)` to do this.

### **NumPy Array Iterating:**

Iterating means going through elements one by one. As we deal with multi-dimensional arrays in numpy, we can do this using basic for loop of python. If we iterate on a 1-D array it will go through each element one by one. In a 2-D array it will go through all the rows.

### **Iterating Arrays Using `nditer()`:**

The function `nditer()` is a helping function that can be used from very basic to very advanced iterations. It solves some basic issues which we face in iteration, let's go through it with examples.

### **Iterating on Each Scalar Element:**

In basic for loops, iterating through each scalar of an array we need to use  $n$  for loops which can be difficult to write for arrays with very high dimensionality.

### **Iterating Array With Different Data Types:**

We can use `op_dtypes` argument and pass it the expected datatype to change the datatype of elements while iterating. NumPy does not change the data type of the element in-place (where the element is in array) so it needs some other space to perform this action, that extra space is called buffer, and in order to enable it in `nditer()` we pass `flags=['buffered']`.

### **Iterating With Different Step Size:**

We can use filtering and followed by iteration.

### **Enumerated Iteration Using `ndenumerate()`:**

Enumeration means mentioning sequence number of somethings one by one. Sometimes we require corresponding index of the element while iterating, the `ndenumerate()` method can be used for those use cases.

### **NumPy Joining Array:**

Joining means putting contents of two or more arrays in a single array. In SQL we join tables based on a key, whereas in NumPy we join arrays by axes. We pass a sequence of arrays that we want to join to the `concatenate()` function, along with the axis. If axis is not explicitly passed, it is taken as 0.

### **Joining Arrays Using Stack Functions:**

Stacking is same as concatenation, the only difference is that stacking is done along a new axis. We can concatenate two 1-D arrays along the second axis which would result in putting them one over the other, i.e. stacking. We pass a sequence of arrays that we want to join to the `stack()` method along with the axis. If axis is not explicitly passed it is taken as 0.

### **Stacking Along Rows Searching Arrays:**

You can search an array for a certain value, and return the indexes that get a match. To search an array, use the `where()` method.

### **NumPy provides a helper function:**

`hstack()` to stack along rows.

### **Stacking Along Columns:**

NumPy provides a helper function:

`vstack()` to stack along columns.

### **NumPy Splitting Array:**

Splitting is reverse operation of Joining. Joining merges multiple arrays into one and Splitting breaks one array into multiple. We use `array_split()` for splitting arrays, we pass it the array we want to split and the number of splits.

### **Split Into Arrays:**

The return value of the `array_split()` method is an array containing each of the split as an array. If you split an array into 3 arrays, you can access them from the result just like any array element:

### **Splitting 2-D Arrays:**

Use the same syntax when splitting 2-D arrays. Use the `array_split()` method, pass in the array you want to split and the number of splits you want to do.

### **NumPy Searching Arrays:**

There is a method called `searchsorted()` which performs a binary search in the array, and returns the index where the specified value would be inserted to maintain the search order. The `searchsorted()` method is assumed to be used on sorted arrays.

### **Search From the Right Side:**

By default the left most index is returned, but we can give `side='right'` to return the right most index instead.

### **NumPy Sorting Arrays:**

Sorting means putting elements in an *ordered sequence*. *Ordered sequence* is any sequence that has an order corresponding to elements, like numeric or alphabetical, ascending or descending. The NumPy ndarray object has a function called `sort()`, that will sort a specified array.

### **NumPy Filter Array:**

Getting some elements out of an existing array and creating a new array out of them is called *filtering*. In NumPy, you filter an array using a *boolean index list*. A *boolean index list* is a list of booleans corresponding to indexes in the array. If the value at an index is True that element is contained in the filtered array, if the value at that index is False that element is excluded from the filtered array. We can directly substitute the array instead of the iterable variable in our condition and it will work just as we expect it to.

## **Pandas**

Pandas is a Python open source library that gives you a highly useful set of tools to do data analysis

Learning Pandas is a must for stepping up your Machine Learning game. Not only is it used for data analysis but also for data science, Machine Learning

To put it simply: if it uses data, you're gonna need Pandas. It can help you load, prepare, merge, join, reshape, analyze, process and adjust data in a blink of an eye.

### **Purpose**

Pandas is an open-source library that lets you easily use datastructures and data analysis tools for the Python programming language.

Pandas is structured around DataFrame objects.

All of your data comes into one big DataFrame where you can select out some samples or other data manipulation if wanted.

Some other fancy things Pandas lets you do are:

- Reading and writing data between in-memory datastructures and different formats such as CSV, text files, Microsoft Excel files, SQL databases, ...
- High-performance merging and joining of data sets
- Data alignment and integrated handling of missing data

### **How to import**

First off you need to install Numpy but only if you're not using Anaconda. To do so:

- Pip install pandas
- Conda install pandas

### **How to import numpy**

- Import pandas
- Import pandas as pd
- from pandas import \*
- from numpy import Series, DataFrame

You always import Pandas as pd, its just silence agreed on.

## **SEABORN LIBRARY**

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures. Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them. There are several Features in the seaborn library. This uses the matplotlib rcParam system and will affect how all matplotlib plots look, even if you don't make them with seaborn. Beyond the default theme, there are several other options, and you can independently control the style and scaling of the plot to quickly translate your work between presentation contexts (e.g., making a version of your figure that will have readable fonts when projected during a talk). If you like the matplotlib defaults or prefer a different theme, you can skip this step and still use the seaborn plotting functions. Seaborn is an open-source Python library built on top of matplotlib. It is used for data visualization and exploratory data analysis. Seaborn works easily with dataframes and the Pandas library. The graphs created can also be customized easily. Below are a few benefits of Data Visualization.

Graphs can help us find data trends that are useful in any machine learning or forecasting project.

- Graphs make it easier to explain your data to non-technical people.
- Visually attractive graphs can make presentations and reports much more appealing to the reader.

### **Types of Graphs**

- Relational Graphs
- Distribution Graphs
- Categorical Graphs
- Regression Plots
- Matrix Plots
- Multi-Plot grids

### **Matplotlib:**

Matplotlib is a plotting library for the programming language and its numerical mathematics extension numpy. Matplotlib is an amazing visualisation library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualisation library built on NumPy arrays and designed to work with the broader SciPy stack. One of the greatest benefits of visualisation is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

### **Importing matplotlib:**

- `from matplotlib import pyplot as plt`
- `import matplotlib.pyplot as plt`

### **Basic plots in Matplotlib:**

Matplotlib comes with a wide variety of plots. Plots helps to understand trends, patterns, and to make correlations. They're typically instruments for reasoning about quantitative information.

### **What Does A Matplotlib Python Plot Look Like?**

At first sight, it will seem that there are quite some components to consider when you start plotting with this Python data visualisation library. You'll probably agree with me that it's confusing and sometimes even discouraging seeing the amount of code that is necessary for some plots, not knowing where to start yourself and which components you should use. Luckily, this library is very flexible and has a lot of handy, built-in defaults that will help you out tremendously. As such, you don't need much to get started: you need to make the necessary imports, prepare some data, and you can start plotting with the help of the `plot()` function! When you're ready, don't forget to show your plot using the `show()` function.

### **Matplotlib Tutorial: Python Plotting**

This Matplotlib tutorial takes you through the basics Python data visualisation: the anatomy of a plot, pyplot and pylab, and much more. Humans are very visual creatures: we understand things



better when we see things visualised. However, the step to presenting analyses, results or insights can be a bottleneck: you might not even know where to start or you might have already a right format in mind, but then questions like “Is this the right way to visualise the insights that I want to bring to my audience?” will have definitely come across your mind. When you’re working with the Python plotting library Matplotlib, the first step to answering the above questions is by building up knowledge on topics like:

- The anatomy of matplotlib: plot creation, which could raise questions about what module you exactly need to import (pylab or pyplot?),
- how you exactly should go about initialising the figure and the Axes of your plot, how to use matplotlib in Jupyter notebooks, etc.
- Plotting routines, from simple ways to plot your data to more advanced ways of visualising your data, with a focus on plot legends and text, titles, axes labels and plot layout.
- Saving, showing, clearing, ... your plots: show the plot, save one or more figures to, for example, pdf files, clear the axes, clear the figure or close the plot, etc.
- Lastly, you’ll briefly cover two ways in which you can customize Matplotlib: with style sheets and the rc settings.

## VII. RESULTS

Comparison of accuracy of algorithm.

Decision Tree 84.5%

Random Forest 98.95%

Naïve Bayes 89.4%

SVM 96.49%

KNN 71.28% We found that the Support Vector Machine (SVM) algorithm is widely used (in 30 studies) followed by the Naïve Bayes algorithm (in 24 studies). However, the Random Forest algorithm showed relatively high accuracy. In the 40 studies in which it was used, RF showed the highest accuracy of 98.95%.

This was followed by SVM which included 96% of the accuracy considered.

## VIII.CONCLUSION

The main aim of this disease prediction system is to predict the disease on the basis of the symptoms. This system takes the symptoms of the user from which he or she suffers as input and generates final output as a prediction of disease. Average prediction accuracy probability of 100% is obtained. Disease Predictor was successfully implemented using the rails framework. This system gives a user-friendly environment and easy to use.

As the system is based on the web application, the user can use this system from anywhere and at any time. In conclusion, for disease risk modeling, the accuracy of risk prediction depends on the diversity feature of the hospital data.

This systematic review aims to determine the performance, limitations, and future use of Software in health care. Findings may help inform future developers of Disease Predictability Software and promote personalized patient care. The program predicts Patient Diseases. Disease Prediction is done through User Symbols.

In this System Decision tree, Unplanned Forest, the Naïve Bayes Algorithm is used to predict diseases. For the data format, the system uses the Machine Learning algorithm Process Data on Database Data namely, Random Forest, Decision Tree, Naive Bayes. System accuracy reaches 98.3%. machine learning skills are designed to successfully predict outbreaks.