



edunet
foundation

2022

Code Unnati

Powered by SAP

Implemented by EDUNET FOUNDATION

Table of Contents

Table of Contents.....	2
Learning Outcome.....	8
Module I Founndational Skills for IR 4.0	9
Unit 1: Python Fundamentals.....	10
1.1 Python Variable	10
1.2 Standard Data types.....	11
1.3 Data type Conversion.....	20
1.4 LAB : Install Anaconda on Windows OS	21
1.5 LAB : Install Anaconda on Linix Kernal	24
1.6 LAB : Familiar with Jupyter Notebook	27
1.7 LAB : Python Libraries.....	31
1.8 LAB : Python Functions and Methods	33
1.9 LAB : Conditional Statement n Python	38
1.10 LAB :Python Loops.....	40
1.11 LAB :Python Datatypes in Python	44
Unit 2 : Data Analysis using NumPy.....	50
2.1 NumPy Library.....	50
2.2 Numpy Stastical Function.....	53
2.3 LAB : Random number Generation using NumPy	54
2.4 LAB : Creating Scalars in NumPy.....	56
2.5 LAB : Creating Vector in NumPy	57
2.6 LAB : Create Matrix in NumPy	58
2.7 LAB : Matrix Multipication in NumPy	58
2.8 LAB : Measure of Central Tendency Using NumPy.....	59
2.9 LAB : Percentile and Interquartile in NumPy	61
2.10 LAB : Array Broadcasting in NumPy.....	63
Unit 3: Data Visualization	66
3.1 What is Data Visualization?.....	66
3.2 Graph Plotting in Python using matplotlib.....	69
3.3 Figures and subplots	71
3.4 Colors, Markers and line styles	73

3.5 Matplotlib Configuration.....	77
3.6 LAB : Line Plot, Bar Plot, Scatter Plot	78
3.7 LAB : Ticks, Labels and Legends, subplots.....	82
3.8 LAB : Histograms and Binning	86
3.9 LAB : Text and Annotations.....	89
3.10 LAB : Three-Dimensional Plotting in Matplotlib	91
Unit 4: Linux fundamentals	94
4.1 Introduction to Linux	94
4.2 LAB : Installation of Ubuntu.....	102
4.3 LAB : Linux commands	111
Unit 5: Database Management	119
5.1 MongoDB Overview	119
5.2 Datatypes Supported by MongoDB	122
5.3 LAB : Import and Export Data	123
5.4 LAB : CRUD Operation	127
Unit 6: Git and GitHub.....	138
6.1 About GitHub.....	138
6.2 Concept of Git	138
6.3 Git Staging	139
6.4 GitHub Commands.....	141
6.5 LAB : Working With GitHub Environment	143
Module II Internet of Things	148
Unit 1: IoT Fundamentals	147
1.1 Internet Usage and Population Stastics	149
1.2 What is Intenet of Things?.....	150
1.3 Why IoT?	151
1.4 IoT Architecture	153
1.5 What is Industrial IoT?.....	154
1.6 IoT Application by Industries	155
1.7 The Future of IoT.....	157
Unit 2: Electronics Concepts	158
2.1 Electronics Components	158
2.2 Logic Gates	166
2.3 Electronics Signals	170
2.4 Pulse Width Modulation.....	171

2.5 What is ADC?	173
2.6 LAB : Use of the Digital Multimeter to Measure various electrical Parameter ..	174
Unit 3: Sensors and Actuators	180
3.1 Sensors Overview and Types of sensors	180
3.2 Introduction to Actuators	188
Unit 4: Networking for IoT	195
4.1 Introduction to Networking Devices	195
4.2 LAN and PAN	196
4.3 IoT WAN	199
4.4 IoT Node	199
4.5 IoT Gateway	200
4.6 IPv4 Vs IPv6	201
4.7 Multi Homing	203
4.8 IoT Protocol stack	204
Unit 5: Raspberry Pi	208
5.1 Introduction to Raspberry Pi	208
5.2 LAB : Install Raspbian OS in Raspberry Pi 4B	211
5.3 LAB : Configure GrovePi+ Kit	213
Unit 6: LAB : Interface GrovePi+ Sensors to Raspberry Pi	224
6.1 LAB : Interface Grove Button with Raspberry Pi	224
6.2 LAB : Interface Grove Rotary angle sensor with Raspberry Pi	226
6.3 LAB : Interface Grove Temperature & Humidity with Raspberry Pi	228
6.4 LAB : Interface Grove Ultrasonic Sensor with Raspberry Pi	230
6.5 LAB : Interface Grove relay Switch with Raspberry Pi	233
Module III Machine Learning	236
Unit 1: Data Manipulation with Pandas	237
1.1 Data Manipulation with Pandas	237
1.2 Introducing Pandas Objects	238
1.3 Pandas Series Object	238
1.4 Pandas DataFrame Object	239
1.5 Create Series From Simple Datatypes	241
1.6 Data Storage format in Pandas	243
1.7 Reading data from files	247
1.8 Groupby Methods	256
1.9 Pivot Tables	260

1.10 Pandas Plotting	263
Unit 2: Python GUI.....	270
2.1 Understanding Python GUI	270
2.2 Introduction to Tkinter.....	271
2.3 Tkinter Themes and Styles.....	277
2.4 LAB : Create GUI using Tkinter	278
Unit 3 : Building Machine Learning Models	281
3.1 Machine Learning Basics	281
3.2 Techniques of ML.....	285
3.3 Scikit Learn library overview.....	288
3.4 Regression vs Classification	290
3.5 Least Square Method.....	291
3.6 Logistic Regression	300
3.7 Estimating Probabilities.....	302
3.8 Decision Trees	304
3.9 Gini Impurity or Entropy.....	306
3.10 Linear SVM Classification	308
3.11 Support Vectors and Margins	311
3.12 Different Distance Methods	315
3.13 Geometric Intuition of K-N	318
3.14 Probability Theory	322
3.15 Naïve Bayes Classifier Algorithm	328
3.16 Bag of Words Approach	330
3.17 Unsupervised Learning	335
3.18 Clustering	337
3.19 K-means clustering.....	339
3.20 Silhouette Score	340
3.21 Machine learning Application using GUI.....	342
Module IV Computer Vision and Edge Computing with openVINO	348
Unit 1: Deep Learning	349
1.1 What is Deep Learning.....	349
1.2 Architecture of DNN	349
1.3 Deep Learning Vs Machine Learning	350
1.4 Concept of Neural Network	352
1.5 Multilayer Perceptron	358

Unit 2: Operational Deep Learning	360
2.1 Gradient Descent	360
2.2 Loss Function and Accuracy Matrics.....	364
2.3 Cross Entropy and MSE.....	368
2.4 OverFitting and Regularization	369
Unit 3: Computer Vision	378
3.1 What is Computer Vision.....	378
3.2 Image Fundamentals.....	379
3.3 Computer Vision using OpenCV.....	380
3.4 LAB : Canny Edge Detection.....	386
3.5 Convolutional Neural Network.....	387
Unit 4: Computer Vision with OpenVINO	391
4.1 Introduction to OpenVINO	391
4.2 OpenVINO Toolkit Components	392
4.3 LAB : Install OpenVINO Toolkit	397
4.4 Exploring OpenVINO Toolkit Directories	402
4.5 Working with Model Optimizer.....	405
4.6 OpenVINO Deep Learning Workbench	408
4.7 Utilizing openVINO for inference at the Edge	425
Reference.....	429



edunet
foundation

This course booklet has been designed by Edunet Foundation
for the Code Unnati programme in partnership with SAP

Learning Outcome

After completing this handbook, learner will be able to

- Demonstrate fundamental understanding of the Data Analysis, Database management and Git System
- Apply the basic principles, models, and algorithms of AI to recognize, model, and solve problems in the analysis and design of information systems.
- Analyze the structures and algorithms of a selection of techniques related to machine learning and Artificial Intelligence.
- Able to design and implement various machine learning algorithms in a range of real-world applications.
- Appreciate the underlying mathematical relationships within and across Machine Learning algorithms and the paradigms of supervised and unsupervised learning.
- Be able to identify new application requirements in the field of computer vision using Deep Learning.
- Demonstrate Fundamental Understand of Internet of things, basic principle of electronics, IoT networking Concept, Use of different types of sensors.
- Able Design solution and create prototype using hardware and software by applying IoT concept.
- Demonstrate fundamental Understanding of the different SAP technical Tools

Module – I

Foundational Skills

for IR 4.0

Unit 1: Python Fundamental

Learning Outcomes:

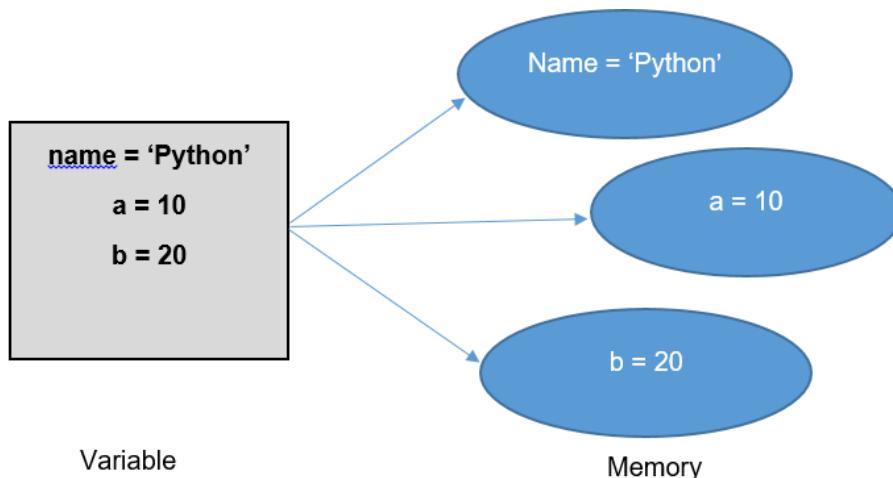
- Understand the python variables and data types used in Python
- Understand converting one data type to another in Python
- Understanding of Local & Global variables

1.1 Python Variables

A variable is a container for a value. It can be assigned a name; you can use it to refer to it later in the program. Based on the data type of the variable, the interpreter allocates memory and decides what can be stored in the reserved memory.

Rules for Variable Names

- Variable name cannot start with a number
- It cannot contain spaces, use _ instead
- Names can not contain any of these symbols (:",<>/?|!@#%^&*~-+)
- Avoid using Python built-in keywords
- Variables names are case-sensitive



Assigning values to Variables

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

```
In [1]: count = 100           #Integer Assignment
kilometers = 1000.0         # Float Assignment
name = 'Rahul'              # String Assignment
```

```
In [2]: print(count,kilometers,name)
```

100 1000.0 Rahul

Multiple Assignment

Python allows you to assign a single value to several variables simultaneously. For example –

```
In [4]: a,b,c=1,2,"john"
print(a,b,c,sep=',')
```

1,2,john

Here, two integer objects with values 1 and 2 are assigned to variables a and b respectively, and one string object with the value "john" is assigned to the variable c.

```
In [5]: a = b = c = 1
```

```
In [6]: print(a,b,c)
```

1 1 1

Here, an integer object is created with the value 1, and all three variables are assigned to the same memory location.

1.2 Standard Data Types

The data stored in memory can be of many types.

For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types –

- Numbers
- String
- List
- Tuple
- Dictionary

1.2.1 Python Numbers

Number data types store numeric values. Number objects are created when you assign a value to them.

```
In [7]: a = 10
```

```
In [8]: a
```

```
Out[8]: 10
```

```
In [12]: # Multiple Assignment
```

```
b,c,d = 11,12,13
```

```
In [13]: print(a,b,c,d)
```

```
10 11 12 13
```

You can also delete the reference to a number object by using the del statement. The syntax of the del statement is –

```
In [14]: #You can also delete the reference to a number object by using the del statement.  
del d
```

```
In [15]: d
```

```
NameError: name 'd' is not defined
```

```
Traceback (most recent call last)  
<ipython-input-15-e983f374794d> in <module>  
----> 1 d
```

```
NameError: name 'd' is not defined
```

You can also delete multiple objects by using the del statement. For example –

```
In [16]: del b,c
```

```
In [17]: b
```

```
NameError Traceback (most recent call last)
<ipython-input-17-89e6c98d9288> in <module>
      1 b
NameError: name 'b' is not defined
```

Python supports three different numerical types –

1. int (signed integers)
2. float (floating point real values)
3. complex (complex numbers)

int	float	complex
10	0	3.14j
100	15.2	45.j
-786	-21.9	9.322e-36j
80	32.3+e18	.876j
-490	-90	-.6545+0J
-0x260	-3.25E+101	3e+26J
0x69	70.2-E12	4.53e-7j

A complex number consists of an ordered pair of real floating-point numbers denoted by $x + yj$, where x and y are the real numbers and j is the imaginary unit.

```
In [18]: a=32.3e18
type(a)
```

```
Out[18]: float
```

```
In [19]: b=3.14j
type(b)
```

```
Out[19]: complex
```

```
In [20]: print(a,b,sep='\n')
```

```
3.23e+19
3.14j
```

1.2.2 Python Strings

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows either pair of single or double quotes.

Subsets of strings can be taken using the slice operator ([] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 to the end.

The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator.

For example –

```
In [21]: str1 = 'Python is a programming language'
```

```
In [22]: print(str1)
```

```
Python is a programming language
```

```
In [23]: print(str1[0])
```

```
P
```

```
In [24]: print(str1[2:10])
```

```
thon is
```

```
In [25]: print(str1[6:18])
```

```
is a progra
```

```
In [27]: print(str1[:])
```

```
Python is a programming language
```

Deleting/Updating from a String

In Python, Updating or deletion of characters from a String is not allowed. This will cause an error because item assignment or item deletion from a String is not supported.

Although deletion of entire String is possible with the use of a built-in del keyword. This is because Strings are immutable, hence elements of a String cannot be changed once it has been assigned. Only new strings can be reassigned to the same name.

```
In [32]: #Updation of a character:  
str1[0]='A'
```

```
-----  
TypeError Traceback (most recent call last)  
<ipython-input-32-6f42197499b8> in <module>  
      1 #Updation of a character:  
----> 2 str1[0]='A'  
  
TypeError: 'str' object does not support item assignment
```

```
In [33]: #Deletion of a character:  
del str1[1]
```

```
-----  
TypeError Traceback (most recent call last)  
<ipython-input-33-b0bad0c6f509> in <module>  
      1 #Deletion of a character:  
----> 2 del str1[1]  
  
TypeError: 'str' object doesn't support item deletion
```

```
In [34]: #Updating Entire String:  
str1="New String"  
print(str1)
```

```
New String
```

```
In [36]: # Deletion of Entire String  
del str1
```

String Formatting

Strings in Python can be formatted with the use of `format()` method which is very versatile and powerful tool for formatting of Strings.

Format method in String contains curly braces {} as placeholders which can hold arguments according to position or keyword to specify the order.

- A string can be left (<), right (>) or center(^) justified with the use of format specifiers, separated by colon (:). Integers such as Binary, hexadecimal, etc. and floats can be rounded or displayed in the exponent form with the use of format specifiers.

```
In [40]: # Default order
String1 = "{} {} {}".format('Python', 'Programming', 'Language')
print("Print String in default order: ")
print(String1)
```

```
Print String in default order:
Python Programming Language
```

```
In [41]: # Positional Formatting
String1 = "{1} {0} {2}".format('Python', 'Programming', 'Language')
print("\nPrint String in Positional order: ")
print(String1)
```

```
Print String in Positional order:
Programming Python Language
```

```
In [42]: # Keyword Formatting
String1 = "{a} {e} {c}".format(a='Python', e='Programming', c='Language')
print("\nPrint String in order of Keywords: ")
print(String1)
```

```
Print String in order of Keywords:
Python Programming Language
```

1.2.3 Python List

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

The values stored in a list can be accessed using the slice operator ([] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator. For example –

```
In [43]: L=[1,2,3,"abes",12.5]
```

```
In [44]: L
```

```
Out[44]: [1, 2, 3, 'abes', 12.5]
```

```
In [45]: L[0:3]
```

```
Out[45]: [1, 2, 3]
```

```
In [46]: L[0]=10
```

```
In [47]: L
```

```
Out[47]: [10, 2, 3, 'abes', 12.5]
```

```
In [48]: #Write a program to take input a sentence and split it into word  
str=input("Enter any sequence")
```

```
Enter any sequence12 34 45 56 67
```

```
In [49]: l=str.split()
```

```
In [50]: l
```

```
Out[50]: ['12', '34', '45', '56', '67']
```

```
In [51]: num=int(input("Enter a number"))  
print(num,"",sep='.' )
```

```
Enter a number12  
12.
```

```
In [52]: list1 = [ 'abcd', 786 , 2.23, 'john', 70.2 ]  
small = [123, 'john']
```

```
In [54]: print(small * 2)      # Prints list two times  
print(list1 + small)      # Prints concatenated lists
```

```
[123, 'john', 123, 'john']  
['abcd', 786, 2.23, 'john', 70.2, 123, 'john']
```

1.2.4 Python Tuples

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

The main differences between lists and tuples are: Lists are enclosed in brackets ([]), and their elements and size can be changed, while tuples are enclosed in parentheses.

(()) and cannot be updated. Tuples can be thought of as read-only lists. For example –

```
In [55]: tuple1 = ('abcd', 786, 2.23, 'john', 70.2 )  
tuple2 = (123, 'john')
```

```
In [58]: print(tuple1)          # Prints the complete tuple  
('abcd', 786, 2.23, 'john', 70.2)
```

```
In [59]: print(tuple1[0])      # Prints first element of the tuple  
abcd
```

```
In [60]: print(tuple1[1:3])    # Prints elements of the tuple starting from 2nd till 3rd  
(786, 2.23)
```

```
In [61]: print(tuple1[2:])     # Prints elements of the tuple starting from 3rd element  
(2.23, 'john', 70.2)
```

```
In [62]: print(tuple2 * 2)      # Prints the contents of the tuple twice  
(123, 'john', 123, 'john')
```

```
In [64]: print(tuple1 + tuple2) # Prints concatenated tuples  
('abcd', 786, 2.23, 'john', 70.2, 123, 'john')
```

The following code is invalid with tuple, because we attempted to update a tuple, which is not allowed. Similar case is possible with lists –

```
In [66]: tuple1 = ('pqrs', 123, 2.14, 'python', 98.6 )  
list1 = [ 'pqrs', 123, 2.14, 'python', 98.6 ]
```

```
In [67]: tuple1[2] = 1000      # Invalid syntax with tuple  
list1[2] = 1000            # Valid syntax with list
```

```
-----  
TypeError                                 Traceback (most recent call last)  
<ipython-input-67-ea927ce875cf> in <module>  
----> 1 tuple1[2] = 1000      # Invalid syntax with tuple  
      2 list1[2] = 1000        # Valid syntax with list  
  
TypeError: 'tuple' object does not support item assignment
```

1.2.5 Python Dictionary

Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

Dictionaries are enclosed by curly braces ({}) and values can be assigned and accessed using square braces ([]). For example –

```
In [71]: dict1 = {}  
dict1['one'] = "This is Python"  
dict1[2]     = "This is Java"  
  
dict2 = {'name': 'rohit', 'id':1234, 'dept': 'technical'}
```

```
In [72]: print(dict1['one'])      # Prints value for 'one' key
```

This is Python

```
In [73]: print(dict1[2])      # Prints value for 2 key
```

This is Java

```
In [74]: print(dict2)      # Prints complete dictionary
```

{'name': 'rohit', 'id': 1234, 'dept': 'technical'}

```
In [75]: print(dict2.keys())  # Prints all the keys
```

dict_keys(['name', 'id', 'dept'])

```
In [76]: print(dict2.values()) # Prints all the values
```

dict_values(['rohit', 1234, 'technical'])

1.3 Data Type Conversion

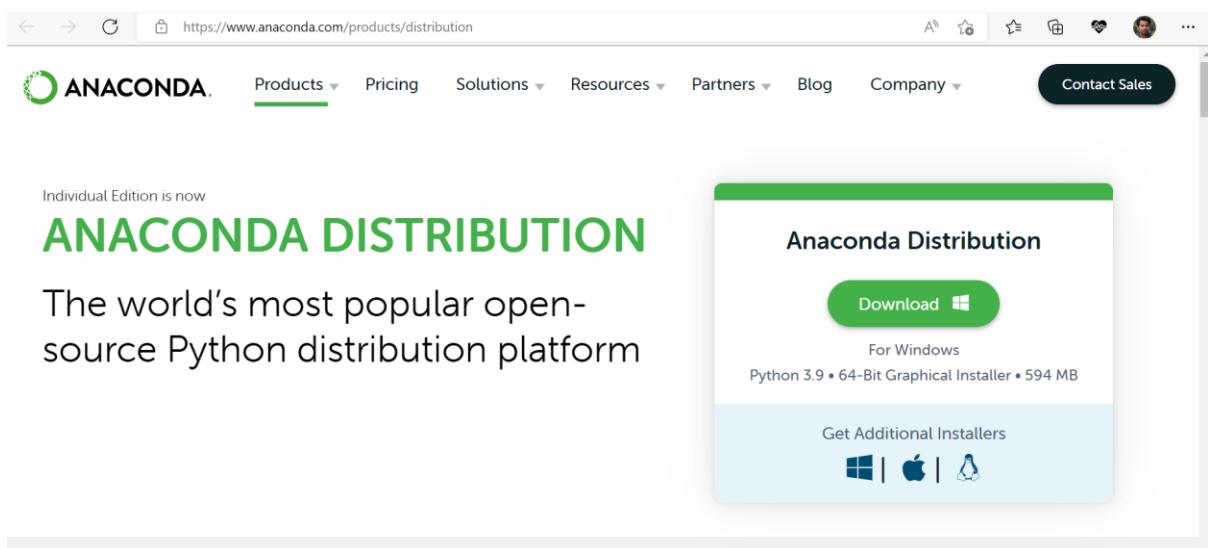
Sr.No.	Function & Description
1	int(x [,base]) Converts x to an integer. base specifies the base if x is a string.
2	long(x [,base]) Converts x to a long integer. base specifies the base if x is a string.
3	float(x) Converts x to a floating-point number.
4	complex(real [,imag]) Creates a complex number.
5	str(x) Converts object x to a string representation.
6	repr(x) Converts object x to an expression string.
7	eval(str) Evaluates a string and returns an object.
8	tuple(s) Converts s to a tuple.
9	list(s) Converts s to a list.
10	set(s) Converts s to a set.
11	dict(d) Creates a dictionary. d must be a sequence of (key,value) tuples.
12	frozenset(s) Converts s to a frozen set.
13	chr(x)

14	unichr(x)	Converts an integer to a Unicode character.
15	ord(x)	Converts a single character to its integer value.
16	hex(x)	Converts an integer to a hexadecimal string.
17	oct(x)	Converts an integer to an octal string.

1.4 PRACTICAL: Installing Anaconda on Windows OS

Anaconda is an open-source software that contains Jupyter, spyder, etc that are used for large data processing, data analytics, heavy scientific computing. Anaconda works for R and python programming language.

1. At first, visit the following link: <https://www.anaconda.com/distribution/> and the page will pop up like this.



The screenshot shows the Anaconda Distribution page. At the top, there's a navigation bar with links for Products, Pricing, Solutions, Resources, Partners, Blog, and Company. A 'Contact Sales' button is also visible. Below the navigation, it says 'Individual Edition is now' followed by 'ANACONDA DISTRIBUTION' in large green letters. Underneath, it says 'The world's most popular open-source Python distribution platform'. On the right side, there's a prominent 'Anaconda Distribution' section with a 'Download' button for Windows, which is described as 'For Windows Python 3.9 • 64-Bit Graphical Installer • 594 MB'. Below this, there's a 'Get Additional Installers' section with icons for Windows, macOS, and Linux.

2. Scroll down the page and select windows.



Windows



macOS



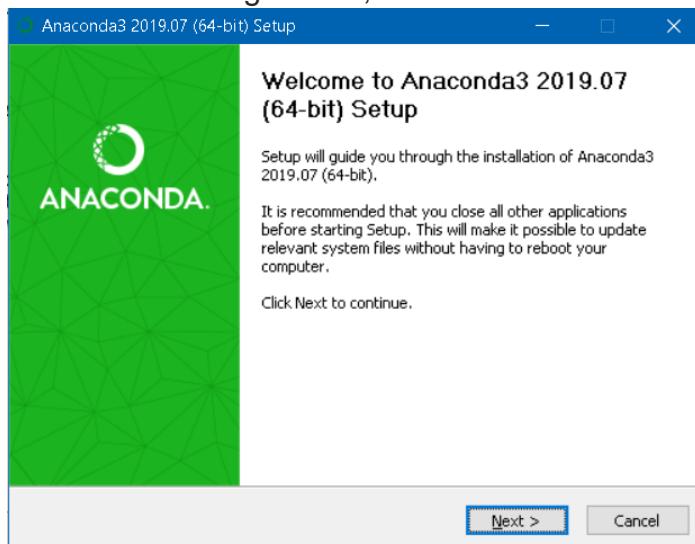
Linux

3. Download Python 3.7 or Above Version (Recommended) as Python version 2 will have no more support by the community at the end of 2019. Depending on your computer system, choose either 32-bit or 64-bit installer to download the .exe file.

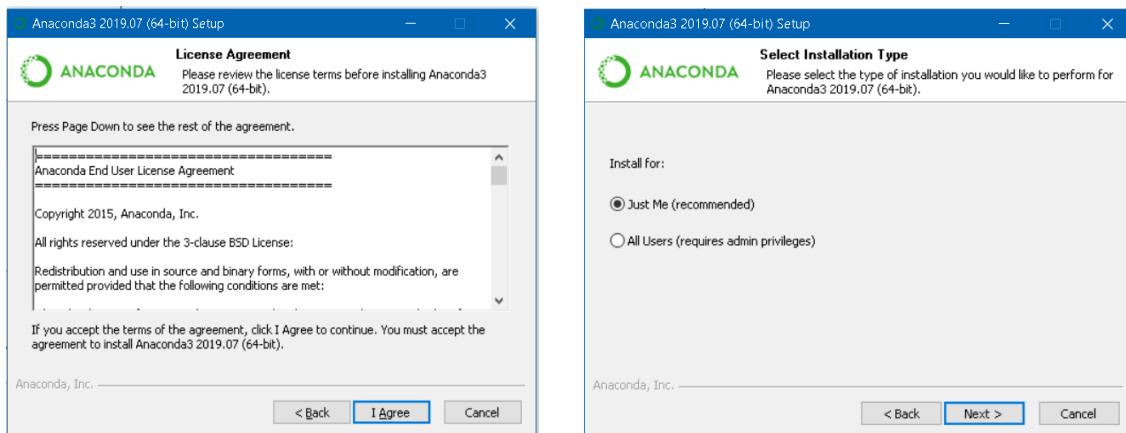
Anaconda Installers

Windows	MacOS	Linux
Python 3.9 64-Bit Graphical Installer (594 MB) 32-Bit Graphical Installer (488 MB)	Python 3.9 64-Bit Graphical Installer (591 MB) 64-Bit Command Line Installer (584 MB) 64-Bit (M1) Graphical Installer (428 MB) 64-Bit (M1) Command Line Installer (420 MB)	Python 3.9 64-Bit (x86) Installer (659 MB) 64-Bit (Power8 and Power9) Installer (367 MB) 64-Bit (AWS Graviton2 / ARM64) Installer (568 MB) 64-bit (Linux on IBM Z & LinuxONE) Installer (280 MB)

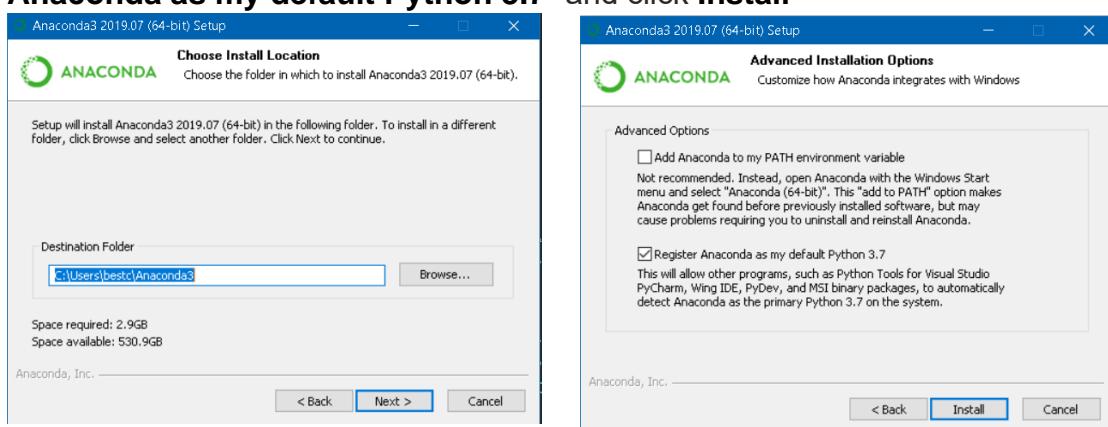
4. After downloading the file, run the file. The file will open, Click **Next**



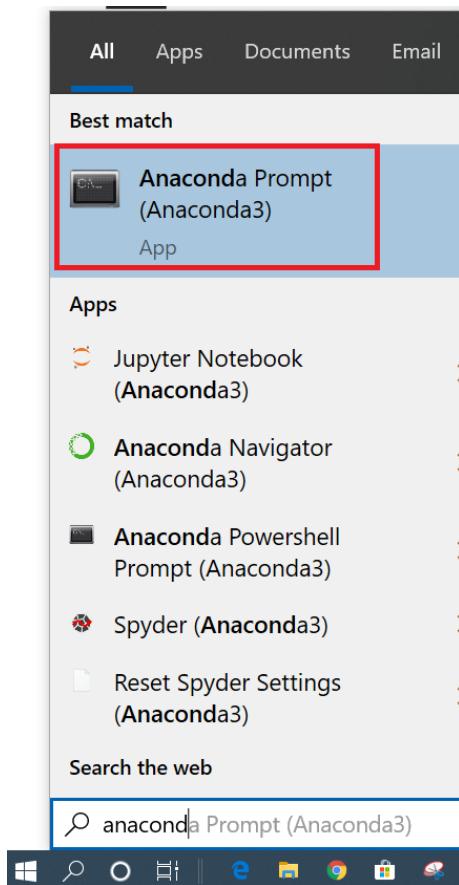
5. And click **I Agree** to the license.
6. Choose **Just Me** and click **Next**



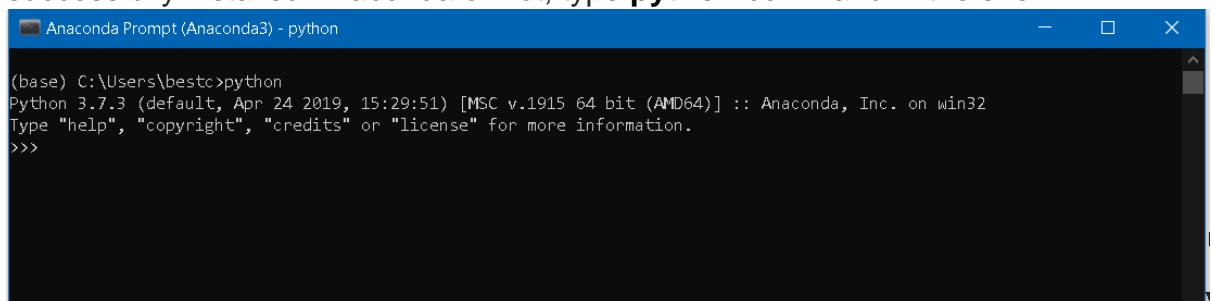
7. Choose the installation location by clicking **Browse** or leave it as it is (default location) and continue to click **Next**.
8. Here, it is highly recommended to choose the second one “**Register Anaconda as my default Python 3.7**” and click **Install**



9. Once the installation is done, open the **Anaconda Prompt** from Windows start menu bar.



10. Anaconda Prompt is shell similar to Windows Command Prompt (Windows Terminal) powered by Anaconda distribution. To check whether we have successfully installed Anaconda or not, type **python** command in the shell.

A screenshot of the Anaconda Prompt window. The title bar says 'Anaconda Prompt (Anaconda3) - python'. The main area of the window shows the Python command-line interface. The output is:
(base) C:\Users\bestc>python
Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>

1.5 PRACTICAL: Installing Anaconda on Linux Kernel

1. At first, visit the following link: <https://www.anaconda.com/distribution/> and the page will pop up like this.
2. Scroll down the page and select windows.

Anaconda Installers

Windows 	MacOS 	Linux 
Python 3.9	Python 3.9	Python 3.9
64-Bit Graphical Installer (594 MB)	64-Bit Graphical Installer (591 MB)	64-Bit (x86) Installer (659 MB)
32-Bit Graphical Installer (488 MB)	64-Bit Command Line Installer (584 MB)	64-Bit (Power8 and Power9) Installer (367 MB)
	64-Bit (M1) Graphical Installer (428 MB)	64-Bit (AWS Graviton2 / ARM64) Installer (568 MB)
	64-Bit (M1) Command Line Installer (420 MB)	64-bit (Linux on IBM Z & LinuxONE) Installer (280 MB)

- Once it is downloaded go to Download folder and run .sh file

```
karthick@LinuxShellTips:~/Downloads$ ls -l
total 557480
-rw-rw-r-- 1 karthick karthick 570853747 Jun 22 18:39 Anaconda3-2021.05-Linux-x86_64.sh
karthick@LinuxShellTips:~/Downloads$ sha256sum Anaconda3-2021.05-Linux-x86_64.sh
2751ab3d678ff0277ae80f9e8a74f218cf70fe9a9cdc7bb1c137d7e47e33d53 Anaconda3-2021.05-Linux-x86_64.sh
```

- Now run the downloaded `.sh` file to install anaconda. As a first step, it will ask you to read the license agreement once you press enter

```
$ bash Anaconda3-2021.05-Linux-x86_64.sh
```

```
karthick@LinuxShellTips:~/Downloads$ bash Anaconda3-2021.05-Linux-x86_64.sh

Welcome to Anaconda3 2021.05

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>>
```

- In the next step, it will ask you to choose a location where the anaconda will be installed. It defaults to your home directory.

```
Anaconda3 will now be installed into this location:  
/home/karthick/anaconda3
```

- Press ENTER to confirm the location
- Press CTRL-C to abort the installation
- Or specify a different location below

```
[/home/karthick/anaconda3] >>> █
```

6. Packages will be installed and once the installation is completed it will ask to initialize **Anaconda3** by running **conda init**. It defaults to **No**. You can choose **Yes** or **No** depending upon how you need it.

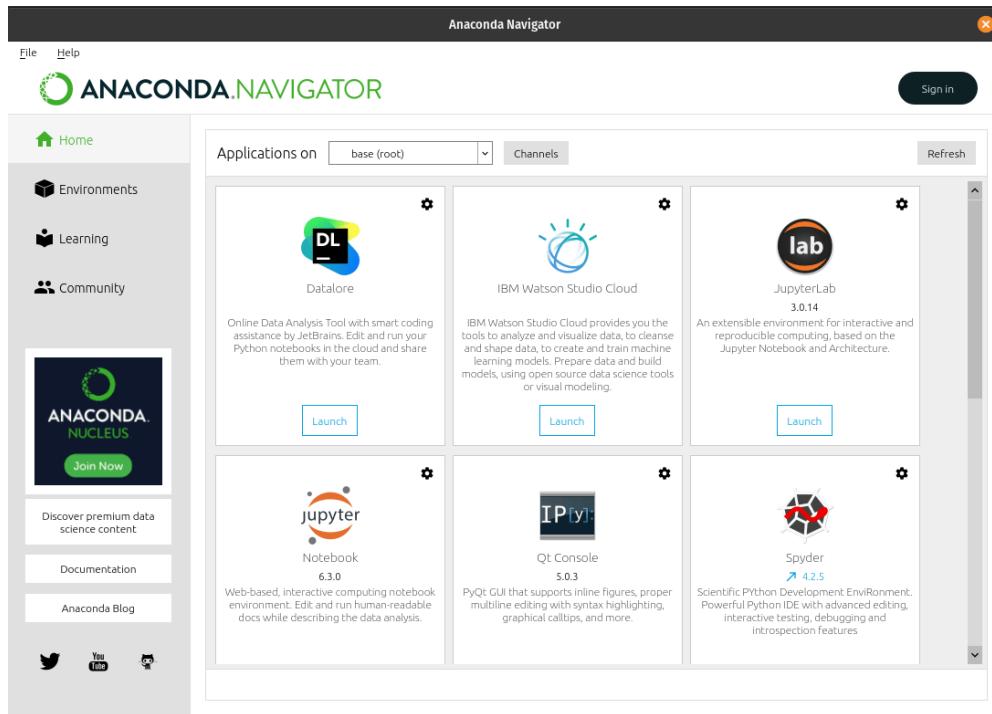
```
Preparing transaction: done  
Executing transaction: done  
installation finished.  
Do you wish the installer to initialize Anaconda3  
by running conda init? [yes|no]  
[no] >>>  
  
You have chosen to not have conda modify your shell scripts at all.  
To activate conda's base environment in your current shell session:  
  
eval "$(/home/karthick/anaconda3/bin/conda shell.YOUR_SHELL_NAME hook)"  
  
To install conda's shell functions for easier access, first activate, then:  
  
conda init  
  
If you'd prefer that conda's base environment not be activated on startup,  
set the auto_activate_base parameter to false:  
  
conda config --set auto_activate_base false  
  
Thank you for installing Anaconda3!  
=====
```

Working with Python and Jupyter notebooks is a breeze with PyCharm Pro, designed to be used with Anaconda. Download now and have the best data tools at your fingertips.

PyCharm Pro for Anaconda is available at: <https://www.anaconda.com/pycharm>

7. Go to the directory where **anaconda** is installed and under the **bin** directory, there is a binary called "**anaconda-navigator**". This will launch the GUI program for anaconda from where you can launch your tools.

```
$ /home/karthick/anaconda3/bin/anaconda-navigator
```



1.6 PRACTICAL: Familiar with jupyter notebook

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at [Project Jupyter](#).

IF you already installed Anaconda in your machine then its very easy to use Jupyter notebook

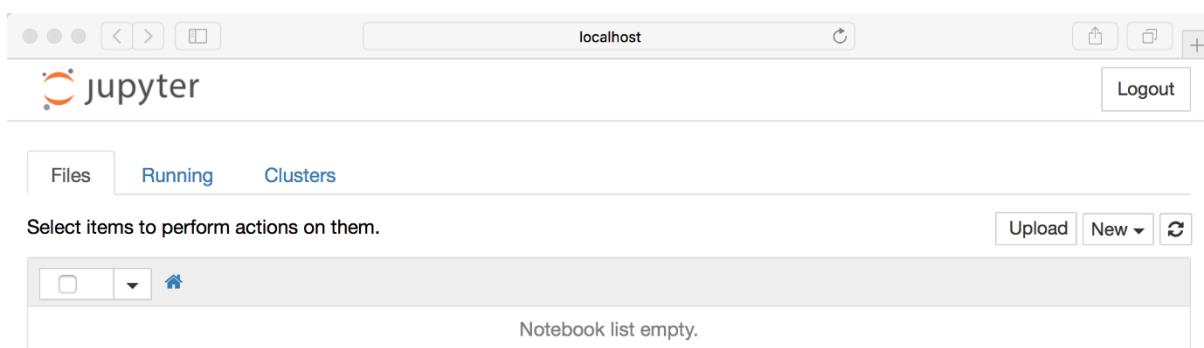
Just Run command from Anaconda to open Jupyter notebook

```
Anaconda Prompt (Anaconda3) - Jupyter notebook
(base) C:\Users\PRAVIN>jupyter notebook
[I 14:30:04.695 NotebookApp] JupyterLab extension loaded from C:\ProgramData\Anaconda3\lib\site-packages\jupyterlab
[I 14:30:04.695 NotebookApp] JupyterLab application directory is C:\ProgramData\Anaconda3\share\jupyter\lab
[I 14:30:04.698 NotebookApp] Serving notebooks from local directory: C:\Users\PRAVIN
[I 14:30:04.698 NotebookApp] Jupyter Notebook 6.1.4 is running at:
[I 14:30:04.698 NotebookApp] http://localhost:8888/?token=fce81d78fb022669006757133ffae92129775d35581a8513
[I 14:30:04.699 NotebookApp] or http://127.0.0.1:8888/?token=fce81d78fb022669006757133ffae92129775d35581a8513
[I 14:30:04.699 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 14:30:04.796 NotebookApp]

To access the notebook, open this file in a browser:
file:///C:/Users/PRAVIN/AppData/Roaming/jupyter/runtime/nserver-11204-open.html
Or copy and paste one of these URLs:
http://localhost:8888/?token=fce81d78fb022669006757133ffae92129775d35581a8513
or http://127.0.0.1:8888/?token=fce81d78fb022669006757133ffae92129775d35581a8513
```

Jupyter notebook will open in your default browser, should start (or open a new tab) to the following URL: <http://localhost:8888/tree>

Your browser should now look something like this:

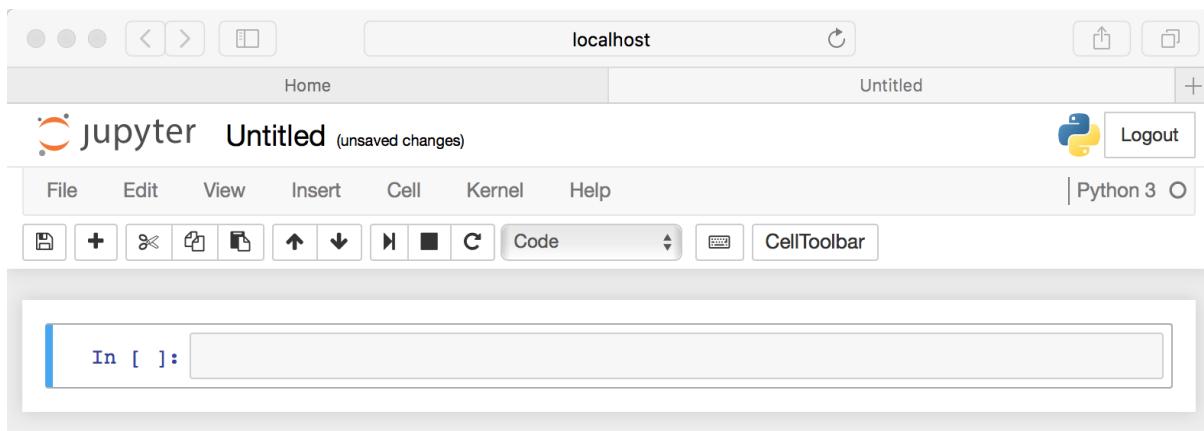


Creating a Notebook

Now that you know how to start a Notebook server, you should probably learn how to create an actual Notebook document.

All you need to do is click on the *New* button (upper right), and it will open up a list of choices. On my machine, I happen to have Python 2 and Python 3 installed, so I can create a Notebook that uses either of these. For simplicity's sake, let's choose Python 3.

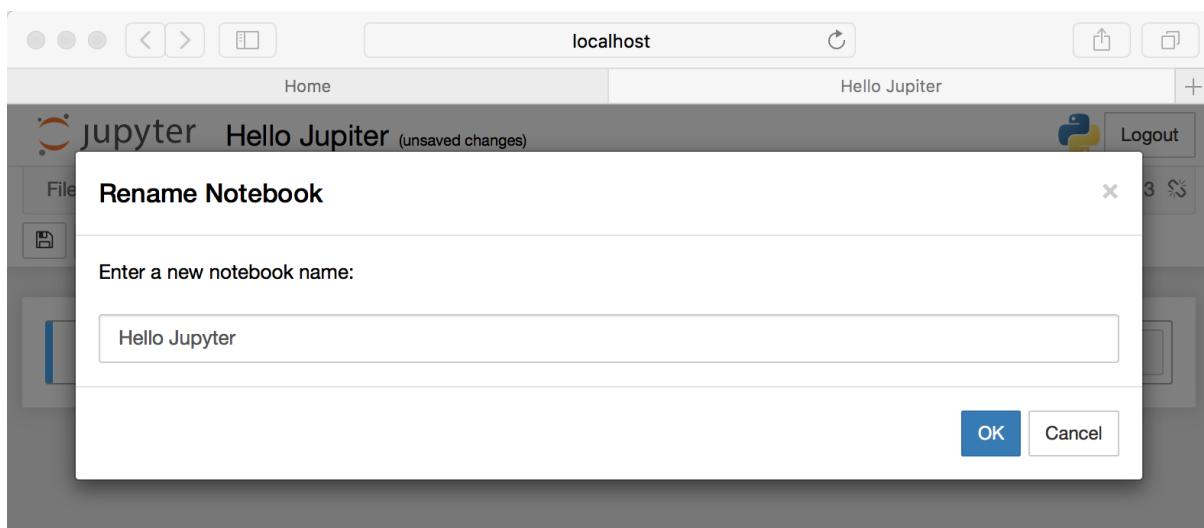
Your web page should now look like this:



Naming

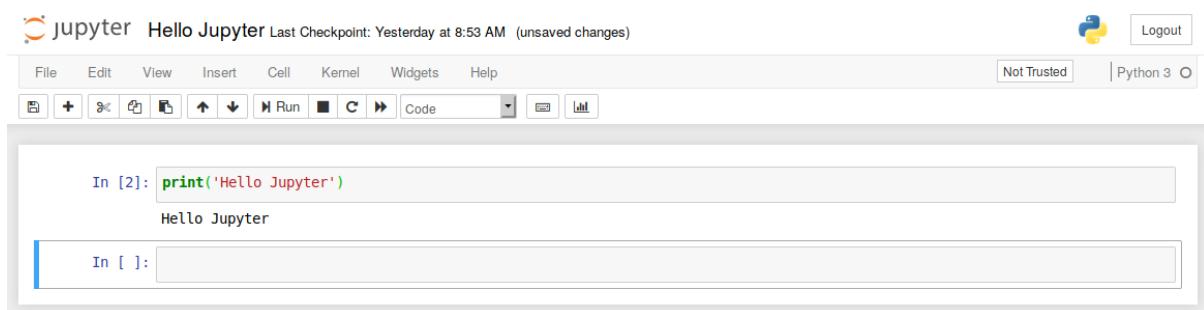
You will notice that at the top of the page is the word *Untitled*. This is the title for the page and the name of your Notebook. Since that isn't a very descriptive name, let's change it!

Just move your mouse over the word *Untitled* and click on the text. You should now see an in-browser dialog titled *Rename Notebook*. Let's rename this one to *Hello Jupyter*.



Running Cells

Running a cell means that you will execute the cell's contents. To execute a cell, you can just select the cell and click the *Run* button that is in the row of buttons along the top. It's towards the middle. If you prefer using your keyboard, you can just press **Shift + Enter**.



If you have multiple cells in your Notebook, and you run the cells in order, you can share your variables and imports across cells. This makes it easy to separate out your code into logical chunks without needing to reimport libraries or recreate variables or functions in every cell.

The Menus

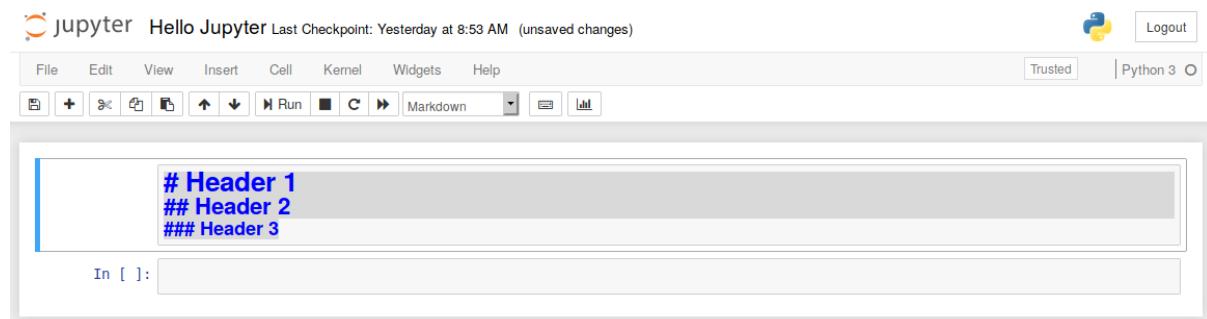
The Jupyter Notebook has several menus that you can use to interact with your Notebook. The menu runs along the top of the Notebook just like menus do in other applications. Here is a list of the current menus:

- File
- Edit
- View
- Insert
- Cell
- Kernel
- Widgets
- Help

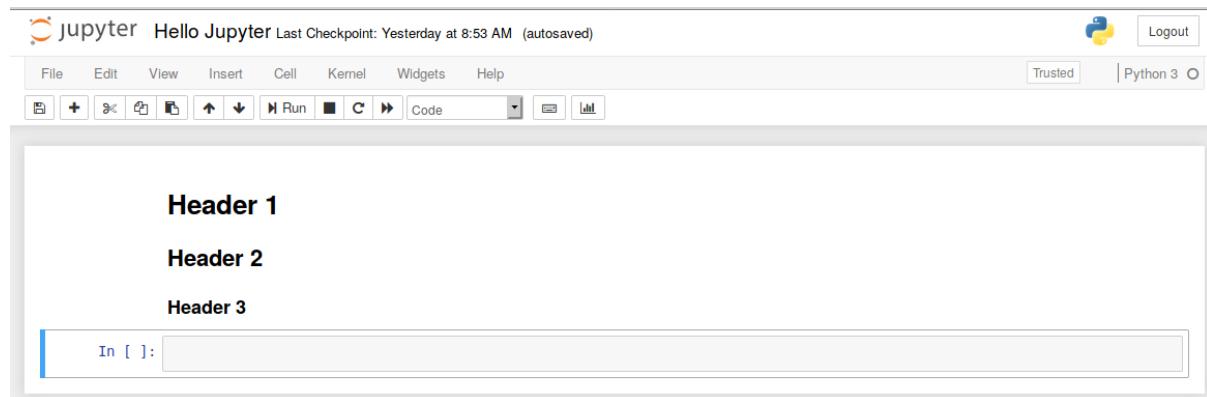
Explore all the Menu one by one to Use of jupyter notebook.

Headers

Creating headers in Markdown is also quite simple. You just have to use the humble pound sign. The more pound signs you use, the smaller the header. Jupyter Notebook even kind of previews it for you:



Then when you run the cell, you will end up with a nicely formatted header:



Exporting Notebooks

When you are working with Jupyter Notebooks, you will find that you need to share your results with non-technical people. When that happens, you can use the nbconvert tool which comes with Jupyter Notebook to convert or export your Notebook into one of the following formats:

- HTML
- LaTeX
- PDF
- RevealJS
- Markdown
- ReStructured Text
- Executable script

The nbconvert tool uses Jinja templates under the covers to convert your Notebook files (.ipynb) into these other formats.

Notebook Extensions

While Jupyter Notebooks have lots of functionality built in, you can add new functionality through extensions. Jupyter actually supports four types of extensions:

- Kernel
- IPython kernel
- Notebook
- Notebook server

Conclusion

The Jupyter Notebook is quite useful not only for learning and teaching a programming language such as Python but also for sharing your data.

You can turn your Notebook into a slideshow or share it online with GitHub. If you want to share a Notebook without requiring your users to install anything, you can use [binder](#) for that.

1.7 PRACTICAL: Python Libraries

Python Libraries are a set of useful functions that eliminate the need for writing codes from scratch.

There are over 137,000 python libraries present today.

Python libraries play a vital role in developing machine learning, data science, data visualization, image and data manipulation applications and more.

What is a Library?

A library is a collection of pre-combined codes that can be used iteratively to reduce the time required to code. They are particularly useful for accessing the pre-written frequently used codes, instead of writing them from scratch every single time. Similar to the physical libraries, these are a collection of reusable resources, which means every library has a root source. This is the foundation behind the numerous open-source libraries available in Python.

Some of the Popular Python Libraries;

1. Numpy
2. Pandas
3. Seaborn
4. Keras
5. Matplotlib
6. Scikit Learn
7. Tensorflow

User Defined Libraries

1. Creating a user defined module.

To create a module just save the code you want in a file with the file extension .py:

Example;

Save this code in a file named mymodule.py

```
def greeting(name):
    print("Hello, " + name)
```

2. Using the Module

Now we can use the module we just created, by using the import statement:

Example

Import the module named mymodule, and call the greeting function:

```
import mymodule
mymodule.greeting("Jonathan")
```

Installing Packages with pip

pip is the standard package manager for Python. It allows you to install and manage additional packages that are not part of the Python standard library.

Check, whether you have an installed version

```
py --version
pip --version
```



```
Command Prompt
C:\Users\User>py --version
Python 3.9.2

C:\Users\User>python --version
Python 3.9.2

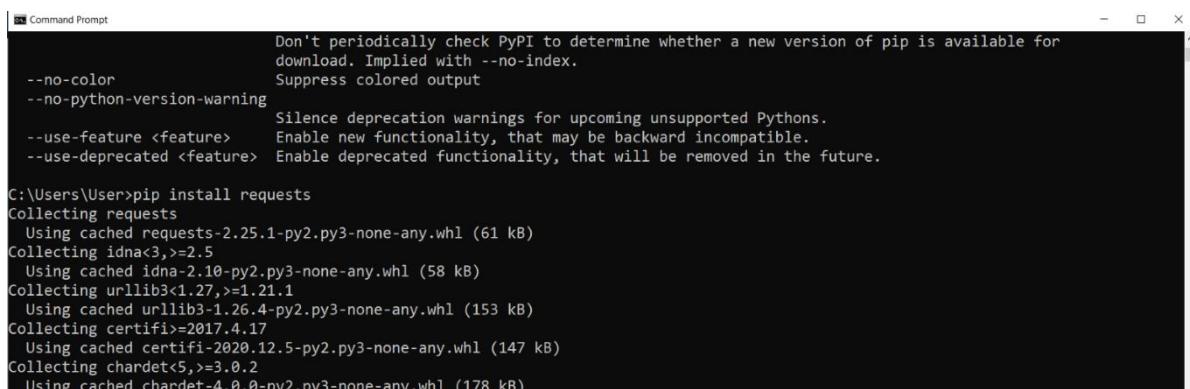
C:\Users\User>pip --version
pip 20.2.3 from c:\python39\lib\site-packages\pip (python 3.9)

C:\Users\User>
```

If you need help, please try help

```
pip help
```

Install the required package using Pip



```
Command Prompt
Don't periodically check PyPI to determine whether a new version of pip is available for
download. Implied with --no-index.

--no-color Suppress colored output
--no-python-version-warning Silence deprecation warnings for upcoming unsupported Pythons.
--use-feature <feature> Enable new functionality, that may be backward incompatible.
--use-deprecated <feature> Enable deprecated functionality, that will be removed in the future.

C:\Users\User>pip install requests
Collecting requests
  Using cached requests-2.25.1-py2.py3-none-any.whl (61 kB)
Collecting idna<3,>=2.5
  Using cached idna-2.10-py2.py3-none-any.whl (58 kB)
Collecting urllib3<1.27,>=1.21.1
  Using cached urllib3-1.26.4-py2.py3-none-any.whl (153 kB)
Collecting certifi>=2017.4.17
  Using cached certifi-2020.12.5-py2.py3-none-any.whl (147 kB)
Collecting chardet<5,>=3.0.2
  Using cached chardet-4.0.0-py2.py3-none-any.whl (178 kB)
```

1.8 PRACTICAL: Python Functions & Methods

In Python, a function is a group of related statements that perform a specific task

Functions help break our program into smaller and modular chunks. As our program grows larger and larger, functions make it more organized and manageable.

Syntax of Function

```
def function_name(parameters):
    """docstring"""
    statement(s)
```

This is the function definition; the components can be described below -

1. Keyword def that marks the start of the function header.
2. A function name to uniquely identify the function. Function naming follows the same rules of writing identifiers in Python.
3. Parameters (arguments) through which we pass values to a function. They are optional.
4. A colon (:) to mark the end of the function header.
5. Optional documentation string (docstring) to describe what the function does.
6. One or more valid python statements that make up the function body. Statements must have the same indentation level (usually 4 spaces).
7. An optional return statement to return a value from the function.

Example of a Function

```
In [ ]: def greet(name):
    """
        This function greets to
        the person passed in as
        a parameter
    """
    print("Hello, " + name + ". Good morning!")
```

How to define & call a function in Python

Function in Python is defined by the "def " statement followed by the function name and parentheses ().

To call a function we simply type the function name with appropriate parameters.

```
In [5]: # Example to Illustrate Function
# Function Definition

def show(n):
    print("Value is",n)
a=10
show(a)           # Calling a Function

def greetings(s):
    print(f"Hello Mr. {s},Good Morning!")
greetings("S.C.Bose")  # Calling a Function

Value is 10
Hello Mr. S.C.Bose,Good Morning!
```

Note: In python, the function definition should always be present before the function call. Otherwise, we will get an error.

Docstrings

The first string after the function header is called the docstring and is short for documentation string. It is briefly used to explain what a function does.

Although optional, documentation is a good programming practice. Unless you can remember what, you had for dinner last week, always document your code.

Return Statement

The return statement is used to exit a function and go back to the place from where it was called.

Syntax of Return Statement

```
return [expression_list]
```

This statement can contain an expression that gets evaluated and the value is returned. If there is no expression in the statement or the return statement itself is not present inside a function, then the function will return the None object.

Example

```
In [5]: def absolute_value(num):
    if num>=0:
        return num
    else:
        return -num -(-4)
a=2
b=8

print(absolute_value(a))
print(absolute_value(b))
```

```
2
8
```

Scope and Lifetime of Variables

Scope of a variable is the portion of a program where the variable is recognized. Parameters and variables defined inside a function are not visible from outside the function. Hence, they have a local scope.

The lifetime of a variable is the period throughout which the variable exists in the memory. The lifetime of variables inside a function is as long as the function executes.

```
In [3]: # Example 1
def test1():
    a=10           # a is Local to function test1      (Local variable)
    print("Value of a inside function",a)

# Outside Function
a=20
test1()
print("Value of a Outside Function",a)
```

Global Variables

A variable declared outside of the function or in global scope is known as global variable.

```
In [3]: # Example 1
def test1():
    a=10           # a is Local to function test1      (Local variable)
    print("Value of a inside function",a)

# Outside Function
a=20
test1()
print("Value of a Outside Function",a)
```

What Happen when we change value of "a" inside a function?

```
PRINTING VALUE OF A OUTSIDE FUNCTION ,A)
```

Python Function Arguments

1. Variable Function Arguments

Functions had a fixed number of arguments. In Python, there are other ways to define a function that can take variable number of arguments.

In Python there are other ways to define a function which can take variable number of arguments.

2. Python Default Arguments

Function arguments can have default values in Python.

We can provide a default value to an argument by using the assignment operator (=). Here is an example.

```
In [11]: def greet(msg,name):
          print('Hello',name+' ,'+msg)

greet('Bhagat Singh','HI')
greet('Sardar Patel','how do you do !')
```

3. Python Keyword Arguments

When we call a function with some values, these values get assigned to the arguments according to their position.

Python allows functions to be called using keyword arguments. When we call functions in this way, the order (position) of the arguments can be changed. Following calls to the above function are all valid and produce the same result.

```
In [20]: greet(name='Priyanshu',msg="How do you do !")
```

```
Hello Priyanshu , How do you do !
```

```
In [21]: greet(msg='How do you do',name='Kartik')
```

```
Hello Kartik , How do you do
```

4. Python Arbitrary Arguments

Sometimes, we do not know in advance the number of arguments that will be passed into a function. Python allows us to handle this kind of situation through function calls with an arbitrary number of arguments.

In the function definition, we use an asterisk (*) before the parameter name to denote this kind of argument. Here is an example.

```
In [22]: def greet(*names):
    for name in names:
        print('Hello',name)

greet('Subhash','Krishna','Ram','Bhagat','Vidit','Mayank','Avnish')
```

Lambda Expression

Python Lambda Functions are anonymous function means that the function is without a name. As we already know that the def keyword is used to define a normal function in Python. Similarly, the lambda keyword is used to define an anonymous function in Python.

Syntax

```
lambda arguments: expression
```

- This function can have any number of arguments but only one expression, which is evaluated and returned.
- One is free to use lambda functions wherever function objects are required.
- You need to keep in your knowledge that lambda functions are syntactically restricted to a single expression.
- It has various uses in particular fields of programming besides other types of expressions in functions.

Example

```
In [4]: # Example 2
n=int(input('Enter the Year : '))
leap=lambda y:(y%4==0 and y%100!=0) or y%400==0
if leap(n):
    print("Leap year")
else:
    print("Not leap Year")
```

1.9 PRACTICAL: Conditional Statement in Python

1. If Statement:

The If statement is the most fundamental decision-making statement, in which the code is executed based on whether it meets the specified condition. It has a code body that only executes if the condition in the if statement is true. The statement can be a single line or a block of code.

Syntax

```
if expression
    Statement
```

Example

```
In [1]: num = 5
if num > 0:
    print(num, "is a positive number.")
print("This statement is true.")
```

2. If-Else Statement

This statement is used when both the true and false parts of a given condition are specified to be executed. When the condition is true, the statement inside the if block is executed; if the condition is false, the statement outside the if block is executed.

Syntax

```
if condition :
    #Will executes this block if the condition is true
else :
    #Will executes this block if the condition is false
```

Example

```
In [2]: num = 5
if num >= 0:
    print("Positive or Zero")
else:
    print("Negative number")
```

3. If-Elif-Else Statement

In this case, the If condition is evaluated first. If it is false, the Elif statement will be executed; if it also comes false, the Else statement will be executed.

Syntax

```
if condition :
    Body of if
elif condition :
    Body of elif
else:
    Body of else
```

Example

```
In [3]: num = 7
if num > 0:
    print("Positive number")
elif num == 0:
    print("Zero")
else:
    print("Negative number")
```

4. Nested If Statement

A Nested IF statement is one in which an If statement is nestled inside another If statement. This is used when a variable must be processed more than once. If, If-else, and If...elif...else statements can be used in the program. In Nested If statements, the

indentation (whitespace at the beginning) to determine the scope of each statement should take precedence.

Syntax

```
if (condition1):
    #Executes if condition 1 is true
    if (condition 2):
        #Executes if condition 2 is true
        #Condition 2 ends here
    #Condition 1 ends here
```

Example

```
In [4]: num = 8
if num >= 0:
    if num == 0:
        print("zero")
    else:
        print("Positive number")
else:
    print("Negative number")
```

1.10 PRACTICAL Python Loops

1. While Loop

A while loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.

Syntax

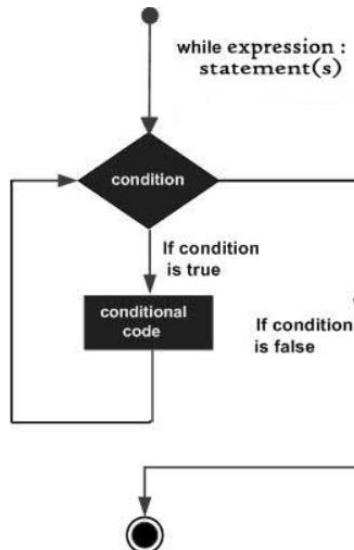
```
while test_expression:
    Body of while
```

Here, statement(s) may be a single statement or a block of statements. The condition may be any expression, and true is any non-zero value. The loop iterates while the condition is true.

When the condition becomes false, program control passes to the line immediately following the loop.

In Python, all the statements indented by the same number of character spaces after a programming construct are considered to be part of a single block of code. Python uses indentation as its method of grouping statements.

Flow Diagram



Here, key point of the while loop is that the loop might not ever run. When the condition is tested and the result is false, the loop body will be skipped and the first statement after the while loop will be executed.

Example

```
In [22]: 1 # Example 1
           2 #initialize counter
           3 i = 1
           4 while i <= 10:
           5     i = i+1    # update counter
           6     print(i,end=' ')
```

2. While loop with else

Python supports to have an else statement associated with a loop statement.

- If the else statement is used with a while loop, the else statement is executed when the condition becomes false.

Example

```
In [4]: 1 # Example to illustrate
2 # the use of else statement and break statement
3 # with the while Loop
4
5 counter = 0
6
7 while counter < 3:
8     if counter == 1:
9         break
10    print("Inside loop")
11    counter = counter + 1
12 else:
13     print("Inside else")
```

3. For Loop

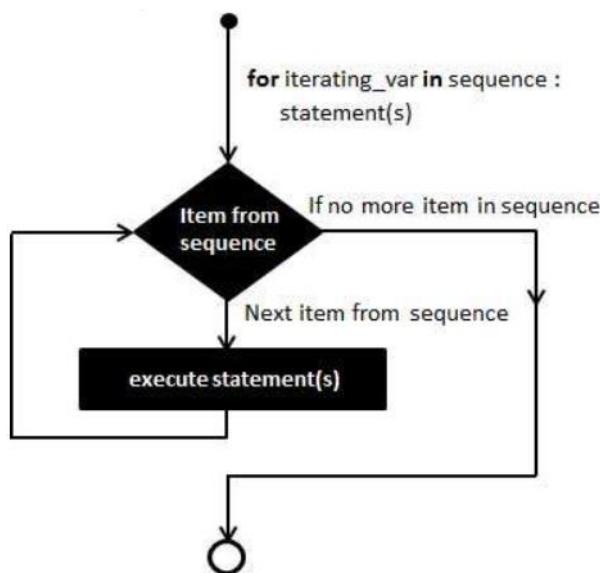
It has the ability to iterate over the items of any sequence, such as a list or a string.

Syntax

for val in sequence:
Body of for

If a sequence contains an expression list, it is evaluated first. Then, the first item in the sequence is assigned to the iterating variable val. Next, the statements block is executed. Each item in the list is assigned to val, and the statement(s) block is executed until the entire sequence is exhausted.

Flow Diagram



Example

```
In [2]: #Example 1
for i in range(1,10):
    print(i,end= " ")
```

```
1 2 3 4 5 6 7 8 9
```

```
In [3]: #Example-2
for i in 'python':
    print(i,end= " ")
```

```
p y t h o n
```

4. Nested Loops

Python programming language allows to use one loop inside another loop.

Syntax (For Loop)

```
for iterating_var in sequence:
    for iterating_var in sequence:
        statements(s)
    statements(s)
```

Syntax (While Loop)

```
while expression:
    while expression:
        statement(s)
    statement(s)
```

Example

In [16]:

```
1 #Example Print floyd Trinangle using for loop
2 n=5
3 i=1
4 for row in range(1,n+1):
5     for col in range(1,row+1):
6         print(i,end=" ")
7         i+=1
8     print()
```

1.11 PRACTICAL: Advanced Data Types in Python

1. Python List

List is an ordered sequence of items. It is one of the most used data types in Python and is very flexible. All the items in a list do not need to be of the same type.

Declaring a list is straight forward. Items separated by commas are enclosed within brackets [].

We can use the slicing operator [] to extract an item or a range of items from a list. The index starts from 0 in Python.

Example

In [5]:

```
a = [5,10,15,20,25,30,35,40]

# a[2] = 15
print("a[2] = ", a[2])

# a[0:3] = [5, 10, 15]
print("a[0:3] = ", a[0:3])

# a[5:] = [30, 35, 40]
print("a[5:] = ", a[5:])
```

2. Python Tuples

Tuple is an ordered sequence of items same as a list. The only difference is that tuples are immutable. Tuples once created cannot be modified.

Tuples are used to write-protect data and are usually faster than lists as they cannot change dynamically.

It is defined within parentheses () where items are separated by commas.

We can use the slicing operator [] to extract items but we cannot change its value.

Example

```
In [6]: t = (5, 'program', 1+3j)

# t[1] = 'program'
print("t[1] = ", t[1])

# t[0:3] = (5, 'program', (1+3j))
print("t[0:3] = ", t[0:3])

# Generates error
# Tuples are immutable
t[0] = 10
```

3. Python Set

Set is an unordered collection of unique items. Set is defined by values separated by comma inside braces {}. Items in a set are not ordered.

Accessing elements of Sets

Set items cannot be accessed by referring to an index, since sets are unordered the items has no index. But you can loop through the set items using a for loop, or ask if a specified value is present in a set, by using the in keyword.

Example

```
In [7]: a = {5,2,3,1,4}

# printing set variable
print("a = ", a)

# data type of variable a
print(type(a))
```

4. Python Dictionary

Dictionary in Python is an unordered collection of data values, used to store data values like a map, which unlike other Data Types that hold only single value as an element, Dictionary holds key: value pair. Key-value is provided in the dictionary to make it more optimized. Each key-value pair in a Dictionary is separated by a colon: whereas each key is separated by a ‘comma’.

Creating Dictionary

In Python, a Dictionary can be created by placing a sequence of elements within curly {} braces, separated by ‘comma’. Values in a dictionary can be of any datatype and can be duplicated, whereas keys can’t be repeated and must be immutable. Dictionary can also be created by the built-in function dict (). An empty dictionary can be created by just placing it to curly braces {}.

Note – Dictionary keys are case sensitive, same name but different cases of Key will be treated distinctly.

Example

```
In [8]: d = {1:'value','key':2}
print(type(d))

print("d[1] = ", d[1])

print("d['key'] = ", d['key'])

# Generates error
print("d[2] = ", d[2])
```

Accessing elements of Dictionary

In order to access the items of a dictionary refer to its key name. Key can be used inside square brackets. There is also a method called get () that will also help in accessing the element from a dictionary.

5. Python Strings

In Python, Strings are arrays of bytes representing Unicode characters. A string is a collection of one or more characters put in a single quote, double-quote or triple quote. In python there is no character data type, a character is a string of length one. It is represented by str class.

String Handling

- Strings are enclosed in single quote, double quote or triple quote
 - s1='Hello'
 - s2="Hello"
 - s3="""Hello"""
 - s4="""Hello"""

- Every string is an object of str class
- Use dir() function to see list of all possible functions on strings
- Use help() function to see syntax and usage of given function

How string manages data?

- Every string is made of certain characters
- Each character has an index number from 0 from front-side
- Each character has an index number from -1 from back-side
- Use len() function to get length of the string

Example

WAP to input a string and show the count of alphabets, digits and special characters using **for loop**.

```
In [12]: st=input()
a,d,s,i=0,0,0,0
for ch in st:
    if (ch>='a' and ch<='z') or (ch>='A' and ch<='Z'):
        a=a+1
    elif ch>='0' and ch<='9':
        d=d+1
    else:
        s=s+1
    i=i+1
print("Alphabets %d Digits %d Special Characters %d" % (a,d,s))
```

```
Flat No 1561, Tower - 3B
Alphabets 12 Digits 5 Special Characters 7
```

String Handling

Basic string functions

- **upper()** convert to upper case
- **lower()** convert to lower case
- **title()** convert first letter of each word to caps
- **capitalize()** convert first letter to caps
- **swapcase()** toggle case
- **isalpha()** returns True if alphabet
- **isdigit()** returns True if digit
- **isupper()** returns True if upper case string
- **islower()** returns True if lower case string
- **istitle()** returns True if string in title case
- **strip()** remove blank space from both sides
- **split()** break the string into list of words
- **join()** join two strings with given separator

Usage of Basic String Functions

```
In [22]: s="I love my India "
print("Actual String : ",s)
print("Length is : ",len(s))
s=s.strip()
print("Stripped String : ",s)
print("Length is : ",len(s))
print("Upper Case : ",s.upper())
print("Lower Case : ",s.lower())
print("Title Case : ",s.title())
print("Capitalize Case : ",s.capitalize())
t=s.split()
print(t)
k=" ".join(t)
print(k)
```

```
Actual String : I love my India
Length is : 17
Stripped String : I love my India
Length is : 15
Upper Case : I LOVE MY INDIA
Lower Case : i love my india
Title Case : I Love My India
Capitalize Case : I love my india
['I', 'love', 'my', 'India']
I love my India
```

String Formatting

Strings in Python can be formatted with the use of format () method which is very versatile and powerful tool for formatting of Strings.

Format method in String contains curly braces {} as placeholders which can hold arguments according to position or keyword to specify the order.

A string can be left (<), right (>) or centre (^) justified with the use of format specifiers, separated by colon (:). Integers such as Binary, hexadecimal, etc. and floats can be rounded or displayed in the exponent form with the use of format specifiers.

```
In [40]: # Default order
String1 = "{} {} {}".format('Python', 'Programming', 'Language')
print("Print String in default order: ")
print(String1)

Print String in default order:
Python Programming Language
```

```
In [41]: # Positional Formatting
String1 = "{1} {0} {2}".format('Python', 'Programming', 'Language')
print("\nPrint String in Positional order: ")
print(String1)
```

```
Print String in Positional order:
Programming Python Language
```

```
In [42]: # Keyword Formatting
String1 = "{a} {e} {c}".format(a='Python', e='Programming', c='Language')
print("\nPrint String in order of Keywords: ")
print(String1)
```

```
Print String in order of Keywords:
Python Programming Language
```

Unit 2: Data Analysis using Numpy

Learning Outcomes:

- Understand the statistical concept using Numpy Library
- Able to implement machine learning algorithm using NumPy
- Solve Numerical Problems using Numpy Library

Let us now get started with the Numpy library to do data analysis.

2.1 Numpy Library

NumPy is an open-source Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices. NumPy stands for Numerical Python.

Why NumPy?

In Python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called `ndarray`, it provides a lot of supporting functions that make working with `ndarray` very easy. Arrays are very frequently used in data science, where speed and resources are very important.

Numpy arrays are stored in a single contiguous (continuous) block of memory. There are two key concepts relating to memory: dimensions and **strides**.

Firstly, many Numpy functions use strides to make things fast. Examples include integer slicing (e.g. `X[1,0:2]`) and broadcasting. Understanding strides helps us better understand how Numpy operates.

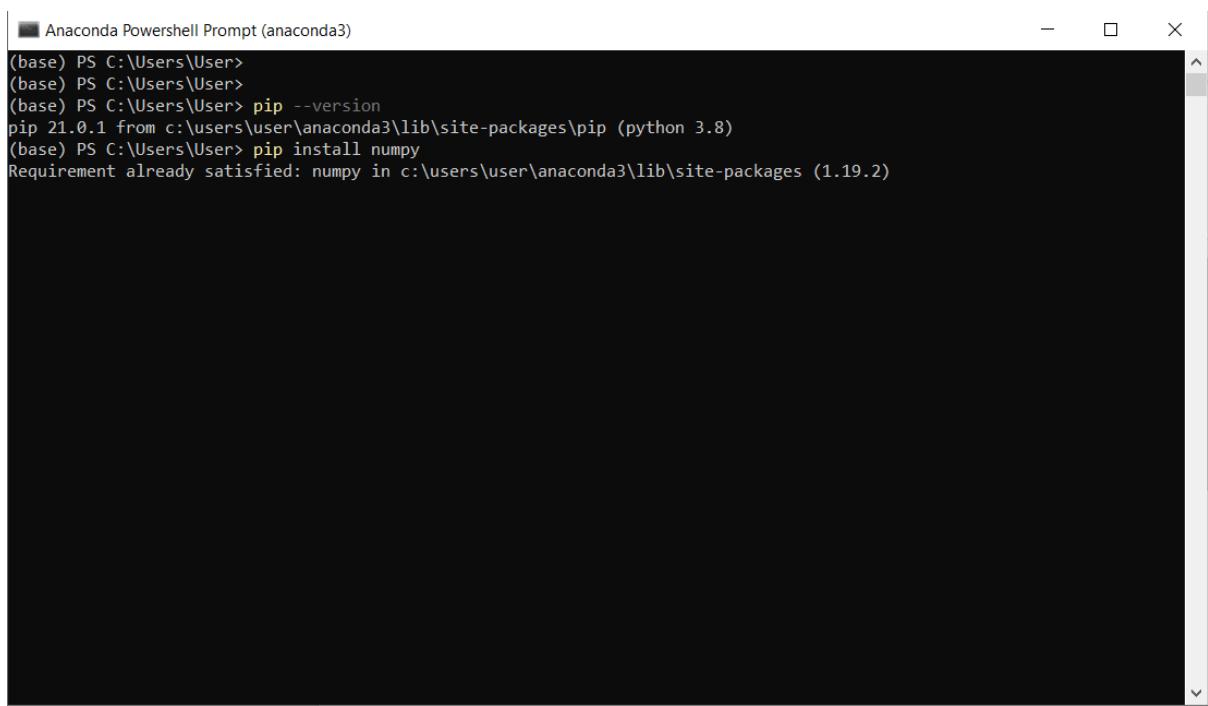
Secondly, we can directly use strides to make our own code faster. This can be particularly useful for data pre-processing in machine learning.

NumPy is a Python library and is written partially in Python, but most of the parts that require fast computation are written in C or C++.

Installing Numpy Module

1. To install NumPy library,

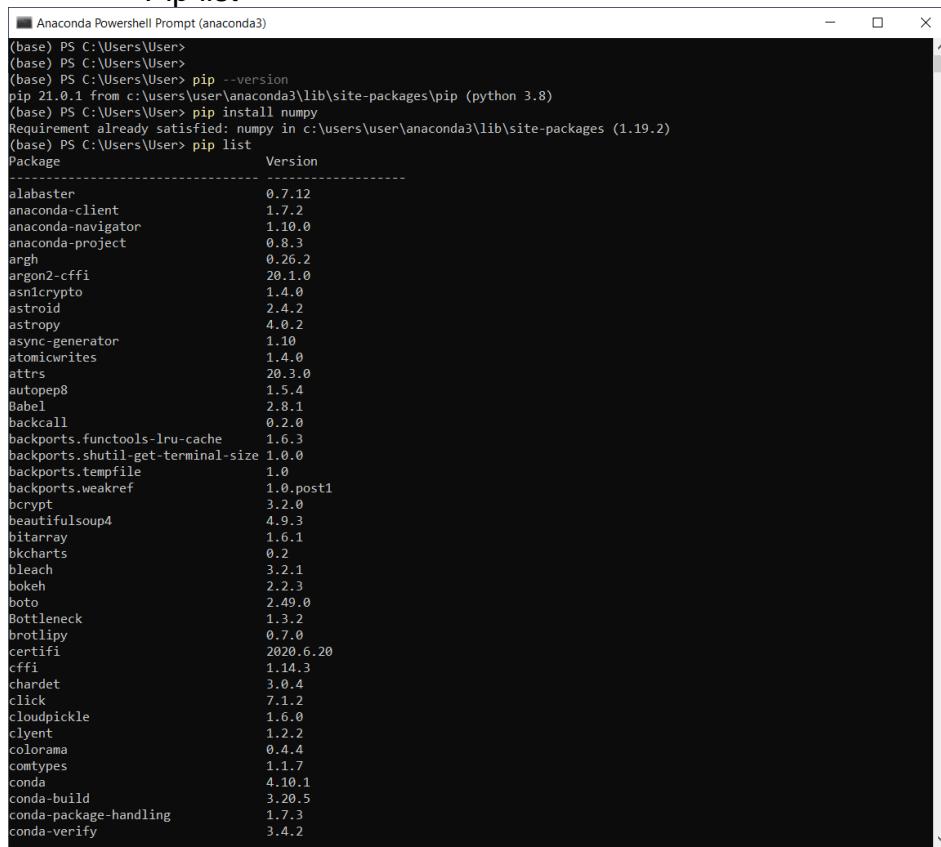
```
pip install NumPy
```



```
[■] Anaconda Powershell Prompt (anaconda3)
(base) PS C:\Users\User>
(base) PS C:\Users\User>
(base) PS C:\Users\User> pip --version
pip 21.0.1 from c:\users\user\anaconda3\lib\site-packages\pip (python 3.8)
(base) PS C:\Users\User> pip install numpy
Requirement already satisfied: numpy in c:\users\user\anaconda3\lib\site-packages (1.19.2)
```

2. To verify the libraries already installed,

Pip list



```
[■] Anaconda Powershell Prompt (anaconda3)
(base) PS C:\Users\User>
(base) PS C:\Users\User>
(base) PS C:\Users\User> pip --version
pip 21.0.1 from c:\users\user\anaconda3\lib\site-packages\pip (python 3.8)
(base) PS C:\Users\User> pip install numpy
Requirement already satisfied: numpy in c:\users\user\anaconda3\lib\site-packages (1.19.2)
(base) PS C:\Users\User> pip list
Package           Version
alabaster        0.7.12
anaconda-client   1.7.2
anaconda-navigator 1.10.0
anaconda-project  0.8.3
argh              0.26.2
argon2-cffi       20.1.0
asn1crypto         1.4.0
astroid            2.4.2
astropy            4.0.2
async-generator    1.10
atomicwrites       1.4.0
attrs              20.3.0
autotep8          1.5.4
Babel              2.8.1
backcall           0.2.0
backports.functools-lru-cache 1.6.3
backports.shutil-get-terminal-size 1.0.0
backports.tempfile  1.0
backports.weakref   1.0.post1
bcrypt             3.2.0
beautifulsoup4     4.9.3
bitarray           1.6.1
bkcharts            0.2
bleach              3.2.1
bokeh              2.2.3
boto                2.49.0
Bottleneck         1.3.2
brotliipy          0.7.0
certifi            2020.6.20
cffi               1.14.3
chardet            3.0.4
click               7.1.2
cloudpickle        1.6.0
clyent              1.2.2
colorama            0.4.4
comtypes            1.1.7
conda              4.10.1
conda-build         3.20.5
conda-package-handling 1.7.3
conda-verify        3.4.2
```

```
■ Anaconda Powershell Prompt (anaconda3)
json5          0.9.5
jsonschema    3.2.0
jupyter       1.0.0
jupyter-client 6.1.7
jupyter-console 6.2.0
jupyter-core   4.6.3
jupyterlab    2.2.6
jupyterlab-pygments 0.1.2
jupyterlab-server 1.2.0
keyring        21.4.0
kiwisolver    1.3.0
lazy-object-proxy 1.4.3
libarchive-c   2.9
llvmlite       0.34.0
locket         0.2.0
lxml           4.6.1
MarkupSafe     1.1.1
matplotlib    3.3.2
mccabe         0.6.1
menuinst       1.4.16
mistune        0.8.4
mkl-fft        1.2.0
mkl-random    1.1.1
mkl-service   2.3.0
mock            4.0.2
more-itertools 8.6.0
mpmath          1.1.0
msgpack         1.0.0
multipledispatch 0.6.0
navigator-updater 0.2.1
nbclient        0.5.1
nbconvert       6.0.7
nbformat        5.0.8
nest-asyncio    1.4.2
networkx       2.5
nltk            3.5
nose            1.3.7
notebook        6.1.4
numba           0.51.2
numexpr         2.7.1
numpy           1.19.2
numpydoc        1.1.0
olefile         0.46
openpyxl        3.0.5
packaging      20.4
pandas          1.1.3
pandocfilters  1.4.3
paramiko        2.7.2
parso           0.7.0
```

Create a NumPy ndarray Object

NumPy is used to work with arrays. The array object in NumPy is called `ndarray`. We can create a NumPy `ndarray` object by using the `array()` function.

1. Create a NumPy ndarray Object

NumPy is used to work with arrays. The array object in NumPy is called `ndarray`. We can create a NumPy `ndarray` object by using the `array()` function.

```
In [3]: import numpy as np

arr = np.array([1, 2, 3, 4, 5])

print(arr)

print(type(arr))

[1 2 3 4 5]
<class 'numpy.ndarray'>
```

Min, Max & Everything in between

```
In [4]: # import numpy library
import numpy

# creating a numpy array of integers
arr = numpy.array([1, 5, 4, 8, 3, 7])

# finding the maximum and
# minimum element in the array
max_element = numpy.max(arr)
min_element = numpy.min(arr)

# printing the result
print('maximum element in the array is: ',
      max_element)
print('minimum element in the array is: ',
      min_element)
```

```
maximum element in the array is: 8
minimum element in the array is: 1
```

2.2 NumPy Statistical Functions

Statistics is concerned with collecting and then analyzing that data. It includes methods for collecting the samples, describing the data, and then concluding that data. NumPy is the fundamental package for scientific calculations and hence goes hand-in-hand for NumPy statistical Functions.

NumPy contains various statistical functions that are used to perform statistical data analysis. These statistical functions are useful when finding a maximum or minimum

of elements. It is also used to find basic statistical concepts like standard deviation, variance, etc.

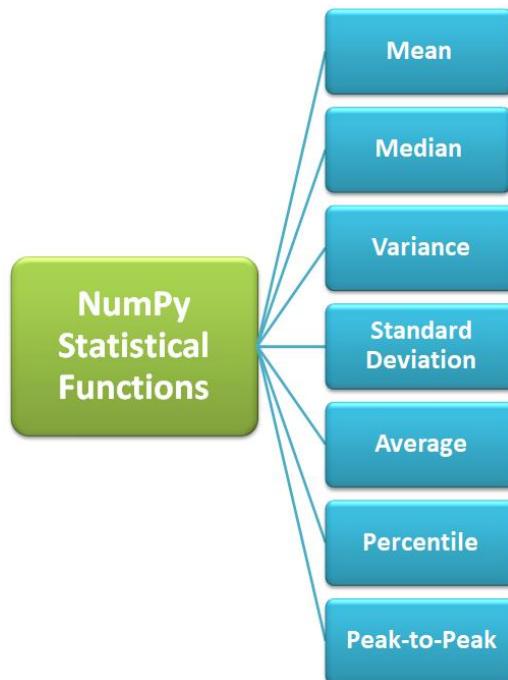


Fig: Numpy Statistical Functions

NumPy Statistical Functions

NumPy is equipped with the following statistical functions, Let us see one by one:

1. np.amin()- This function determines the minimum value of the element along a specified axis.
2. np.amax()- This function determines the maximum value of the element along a specified axis.
3. np.mean()- It determines the mean value of the data set.
4. np.median()- It determines the median value of the data set.
5. np.std()- It determines the standard deviation
6. np.var() – It determines the variance.
7. np.ptp()- It returns a range of values along an axis.
8. np.average()- It determines the weighted average
9. np.percentile()- It determines the nth percentile of data along the specified axis.

2.3 PRACTICAL: Random Number Generator

1. Generate Random Number

NumPy offers the random module to work with random numbers.

Ex: Generate a random integer from 0 to 100 :

```
In [5]: from numpy import random  
  
x = random.randint(100)  
  
print(x)
```

89

```
In [6]: from numpy import random  
  
x = random.randint(100)  
  
print(x)
```

45

2. Generate Random Float

The random module's `rand()` method returns a random float between 0 and 1.

Ex: Generate a random float from 0 to 1 :

```
In [7]: from numpy import random  
  
x = random.rand()  
  
print(x)
```

0.43203375825754287

3. Generate Random Array

In NumPy we work with arrays, and you can use the two methods from the above examples to make random arrays.

Integers

The `randint()` method takes a size parameter where you can specify the shape of an array.

Ex: Generate a 1-D array containing 5 random integers from 0 to 100

```
In [8]: from numpy import random  
  
x=random.randint(100, size=(5))  
  
print(x)  
  
[71 53 93 89 47]
```

Ex: Generate a 2-D array with 3 rows, each row containing 5 random integers from 0 to 100

```
In [9]: from numpy import random  
  
x = random.randint(100, size=(3, 5))  
  
print(x)  
  
[[75  8 37 45  4]  
 [39 17 51 75 29]  
 [66 61 48 52 83]]
```

2.4 PRACTICAL: Creating scalars in Numpy

```
In [13]: # Python program explaining  
# numpy.asscalar() function  
  
import numpy as edunet  
# creating a array of size 1  
in_arr = edunet.array([ 8 ])  
  
print ("Input array : ", in_arr)  
  
→  
out_scalar = edunet.asscalar(in_arr)  
print ("output scalar from input array : ", out_scalar)
```

```
Input array : [8]  
output scalar from input array : 8
```

2.5 PRACTICAL: Creating Vector in Numpy

```
In [12]: # importing numpy
import numpy as np

# creating a 1-D list (Horizontal)
list1 = [1, 2, 3]

# creating a 1-D list (Vertical)
list2 = [[10],
          [20],
          [30]]

# creating a vector1
# vector as row
vector1 = np.array(list1)

# creating a vector 2
# vector as column
vector2 = np.array(list2)

# showing horizontal vector
print("Horizontal Vector")
print(vector1)

print("-----")

# showing vertical vector
print("Vertical Vector")
print(vector2)
```

```
Horizontal Vector
[1 2 3]
-----
Vertical Vector
[[10]
 [20]
 [30]]
```

2.6 PRACTICAL Creating Matrix in Numpy

```
# Python Program illustrating
# numpy.matrix class

import numpy as edunet

# string input
a = edunet.matrix('1 2; 3 4')
print("Via string input : \n", a, "\n\n")

# array-like input
b = edunet.matrix([[5, 6, 7], [4, 6]])
print("Via array-like input : \n", b)
```

Via string input :
[[1 2]
[3 4]]

Via array-like input :
[[list([5, 6, 7]) list([4, 6])]]

2.7 PRACTICAL: Matrix Multiplication in Numpy

Matrix multiplication is an operation that takes two matrices as input and produces single matrix by multiplying rows of the first matrix to the column of the second matrix. In matrix multiplication make sure that the number of rows of the first matrix should be equal to the number of columns of the second matrix.

Ex;

```
Input:matrix1 = ([1, 2, 3],
                 [3, 4, 5],
                 [7, 6, 4])
matrix2 = ([5, 2, 6],
           [5, 6, 7],
           [7, 6, 4])
```

```
Output : [[36 32 32]
          [70 60 66]
          [93 74 100]]
```

```
In [14]: import numpy as np

# input two matrices
mat1 = ([1, 6, 5],[3,4, 8],[2, 12, 3])
mat2 = ([3, 4, 6],[5, 6, 7],[6,56, 7])

# This will return dot product
res = np.dot(mat1,mat2)

# print resulted matrix
print(res)
```

```
[[ 63 320 83]
 [ 77 484 102]
 [ 84 248 117]]
```

2.8 PRACTICAL: Measure of Central Tendency Using Numpy

1. Finding maximum and minimum of array in NumPy
NumPy `np.amin()`and `np.amax()`functions are useful to determine the minimum and maximum value of array elements along a specified axis.

```
In [17]: import numpy as np
arr= np.array([[1,23,78],[98,60,75],[79,25,48]])
print(arr)
#Minimum Function
print(np.amin(arr))
#Maximum Function
print(np.amax(arr))
```

```
[[ 1 23 78]
 [98 60 75]
 [79 25 48]]
1
98
```

2. Finding Mean, Median, Standard Deviation and Variance in NumPy
Mean

Mean is the sum of the elements divided by the number of elements and is given by the following formula:

$$\bar{x} = \frac{1}{n}(x_1 + x_2 + \dots + x_n)$$

It calculates the mean by adding all the items of the arrays and then divides it by the number of elements. We can also mention the axis along which the mean can be calculated.

```
In [18]: import numpy as np
a = np.array([5,6,7])
print(a)
print(np.mean(a))
```

```
[5 6 7]
6.0
```

Median

Median is the middle element of the array. The formula differs for odd and even sets. The median for a given set of data with n elements (observations) is given by

$$\text{Odd data} = \left(\frac{n+1}{2}\right)^{\text{th}} \text{ observation}, \quad \text{Even data} = \frac{\frac{n^{\text{th}}}{2} \text{ obs.} + \left(\frac{n}{2}+1\right)^{\text{th}} \text{ obs.}}{2}$$

It can calculate the median for both one-dimensional and multi-dimensional arrays. Median separates the higher and lower range of data values.

```
In [19]: import numpy as np
a = np.array([5,6,7])
print(a)
print(np.median(a))
```

```
[5 6 7]
6.0
```

Standard Deviation

Standard deviation is the square root of the average of square deviations from mean. The formula for standard deviation is:

$$SD = \sqrt{\frac{\sum |x - \bar{x}|^2}{n}}$$

In [20]:

```
import numpy as np
a = np.array([5,6,7])
print(a)
print(np.std(a))
```

```
[5 6 7]
0.816496580927726
```

Variance

Variance is the average of the squared differences from the mean. Following is the formula for the same:

$$\sigma^2 = \frac{\sum (x_r - \bar{x})^2}{n}$$

In [21]:

```
import numpy as np
a = np.array([5,6,7])
print(a)
print(np.var(a))
```

```
[5 6 7]
0.6666666666666666
```

2.9 PRACTICAL: Percentile & Interquartile in Numpy

Quartiles:

A quartile is a type of quantile. The first quartile (Q1), is defined as the middle number between the smallest number and the median of the data set, the second quartile (Q2) – median of the given data set while the third quartile (Q3), is the middle number between the median and the largest value of the data set.

Algorithm to find Quartiles:

Quartiles are calculated by the help of the median. If the number of entries is an even number i.e. of the form $2n$, then, first quartile (Q1) is equal to the median of the n smallest entries and the third quartile (Q3) is equal to the median of the n largest entries.

If the number of entries is an odd number i.e. of the form $(2n + 1)$, then

- the first quartile (Q1) is equal to the median of the n smallest entries
- the third quartile (Q3) is equal to the median of the n largest entries

- the second quartile(Q2) is the same as the ordinary median.

Range: It is the difference between the largest value and the smallest value in the given data set.

Interquartile Range : The interquartile range (IQR), also called as midspread or middle 50%, or technically H-spread is the difference between the third quartile (Q3) and the first quartile (Q1). It covers the center of the distribution and contains 50% of the observations. $IQR = Q3 - Q1$

Uses;

- The interquartile range has a breakdown point of 25% due to which it is often preferred over the total range.
- The IQR is used to build box plots, simple graphical representations of a probability distribution.
- The IQR can also be used to identify the outliers in the given data set.
- The IQR gives the central tendency of the data.

Decision Making

- The data set having a higher value of interquartile range (IQR) has more variability.
- The data set having a lower value of interquartile range (IQR) is preferable.

Interquartile range using numpy.median

```
In [22]: import numpy as np

data = [32, 36, 46, 47, 56, 69, 75, 79, 79, 88, 89, 91, 92, 93, 96, 97,
       101, 105, 112, 116]

# First quartile (Q1)
Q1 = np.median(data[:10])

# Third quartile (Q3)
Q3 = np.median(data[10:])

# Interquartile range (IQR)
IQR = Q3 - Q1

print(IQR)
```

34.0

Interquartile range using numpy.percentile

```
In [23]: # Import numpy library
import numpy as np

data = [32, 36, 46, 47, 56, 69, 75, 79, 79, 88, 89, 91, 92, 93, 96, 97,
        101, 105, 112, 116]

# First quartile (Q1)
Q1 = np.percentile(data, 25, interpolation = 'midpoint')

# Third quartile (Q3)
Q3 = np.percentile(data, 75, interpolation = 'midpoint')

# Interquartile range (IQR)
IQR = Q3 - Q1

print(IQR)
```

34.0

After knowing the various statistical functions in NumPy which are used in descriptive analytics, let us see some other interesting functionalities of NumPy.

2.10 PRACTICAL: Array Broadcasting in Numpy

The term broadcasting describes how numpy treats arrays with different shapes during arithmetic operations. Subject to certain constraints, the smaller array is “broadcast” across the larger array so that they have compatible shapes. [4.9]

```
In [27]: a = np.array([1.0, 2.0, 3.0])
b = np.array([2.0, 2.0, 2.0])
a * b
```

Out[27]: array([2., 4., 6.])

```
In [28]: from numpy import array
a = array([1.0, 2.0, 3.0])
b = array([2.0, 2.0, 2.0])
a * b
```

Out[28]: array([2., 4., 6.])

In [29]:

```
from numpy import array
a = array([1.0,2.0,3.0])
b = 2.0
a * b
```

Out[29]: array([2., 4., 6.])

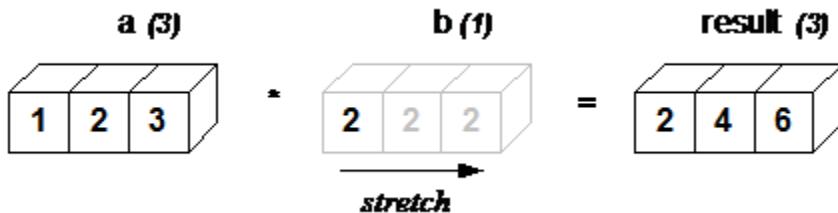


Figure 1: Broadcast Array

Ref: <https://numpy.org/devdocs/user/theory.broadcasting.html#figure-1>

In the simplest example of broadcasting, the scalar ``b`` is stretched to become an array of same shape as ``a`` so the shapes are compatible for element-by-element multiplication.

The result is equivalent to the previous example where b was an array. We can think of the scalar b being stretched during the arithmetic operation into an array with the same shape as a. The new elements in b, as shown in Figure 1, are simply copies of the original scalar.

Sorting Arrays

`numpy.sort()` : This function returns a sorted copy of an array.

Parameters;

`arr` : Array to be sorted.

`axis` : Axis along which we need array to be started.

`order` : This argument specifies which fields to compare first.

`kind` : ['quicksort'{default}, 'mergesort', 'heapsort']Sorting algorithm.

```
In [30]: import numpy as np

# sort along the first axis
a = np.array([[12, 15], [10, 1]])
arr1 = np.sort(a, axis = 0)
print ("Along first axis : \n", arr1)

# sort along the last axis
a = np.array([[10, 15], [12, 1]])
arr2 = np.sort(a, axis = -1)
print ("\nAlong first axis : \n", arr2)

a = np.array([[12, 15], [10, 1]])
arr1 = np.sort(a, axis = None)
print ("\nAlong none axis : \n", arr1)
```

```
Along first axis :
[[10  1]
 [12 15]]
```

```
Along first axis :
[[10 15]
 [ 1 12]]
```

```
Along none axis :
[ 1 10 12 15]
```

Unit 3: Data Visualization

Learning Outcomes:

- Use Matplotlib library and its various functions to visualize the data
- Able to create visualization from any data
- Understand and take decision based on data visualization

3.1 What is Data Visualization?

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data. In the world of Big Data, data visualization tools and technologies are essential to analyze massive amounts of information and make data-driven decisions.



Image: sample Data visualization

Reference: https://cdnl.tblsft.com/sites/default/files/pages/_data_visualization_definition.gif

Our eyes are drawn to colors and patterns. We can quickly identify red from blue, square from circle. Our culture is visual, including everything from art and advertisements to TV and movies.

Data visualization is another form of visual art that grabs our interest and keeps our eyes on the message. When we see a chart, we quickly see trends and outliers. If we can see something, we internalize it quickly. It's storytelling with a purpose. If you've ever stared at a massive spreadsheet of data and couldn't see a trend, you know how much more effective a visualization can be.

Data visualization helps transform your data into an engaging story with details and patterns.

- Better Analysis
- Speed up decision making process
- Quick action
- Identifying patterns
- Story telling is more engaging
- Grasping the latest trends
- Finding errors

Families of Visualizations

Data Visualization's Family speaks to its nature. This allows for hierarchical classification between major groups: Charts, Geospatial Visualizations, and Tables.

Charts

A chart is a representation of data in the form of a graph, diagram, map, or tabular format. This could make the other two families, Geospatial and Tables, subfamilies of it.

Later, we will discuss about different types of charts in detail.

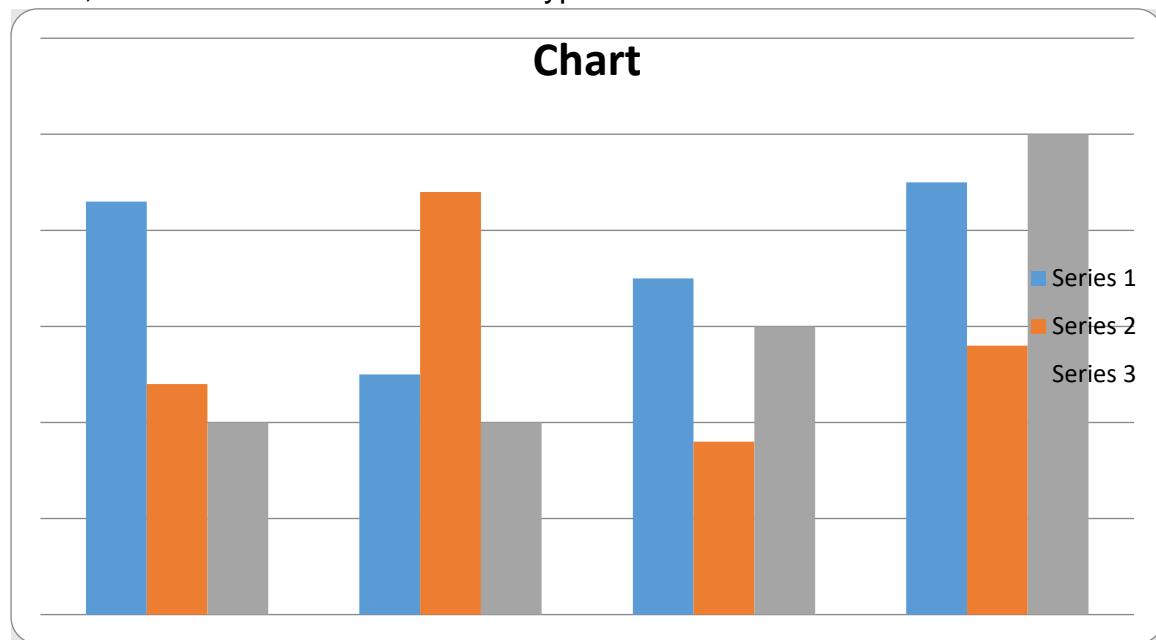


Image: Sample Chart

Geospatial visualization or Geovisualization

These visualizations focus on the relationship between data and its location to create insight. Any positional data works for spatial analysis.



Image: sample Geospatial visualization

Reference: <https://www.tableau.com/learn/articles/data-visualization/glossary>

Tables

Tables, also known as “crosstabs” or “spreadsheets”, are one of the most common ways to present data. They may not be visualizations in themselves, but they can be a powerful tool for visual analysis

1234	678
368	8034
2620	2559
971	322

Image: sample tables visualization

Reference: <https://www.tableau.com/learn/articles/data-visualization/glossary>

3.2 Plotting and Visualization

Plotting is a chart or map showing the movements or progress of an object.

A plot is a graphical technique for representing a data set, usually as a graph showing the relationship between two or more variables.

Python offers multiple great graphing libraries that come packed with lots of different features.

To get a little overview here are a few popular plotting libraries:

- [Matplotlib](#): low level, provides lots of freedom
- [Pandas Visualization](#): easy to use interface, built on Matplotlib
- [Seaborn](#): high-level interface, great default styles
- [ggplot](#): based on R's ggplot2, uses Grammar of Graphics
- [Plotly](#): can create interactive plots

Let us explore the Matplotlib library.

Matplotlib

Here we will learn how to create basic plots using Matplotlib.

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

How to install matplotlib in python?

```
pip install matplotlib  
or  
conda install matplotlib
```

```
import matplotlib.pyplot as plt
```

[matplotlib.pyplot](#) is a collection of command style functions that make matplotlib work like MATLAB.

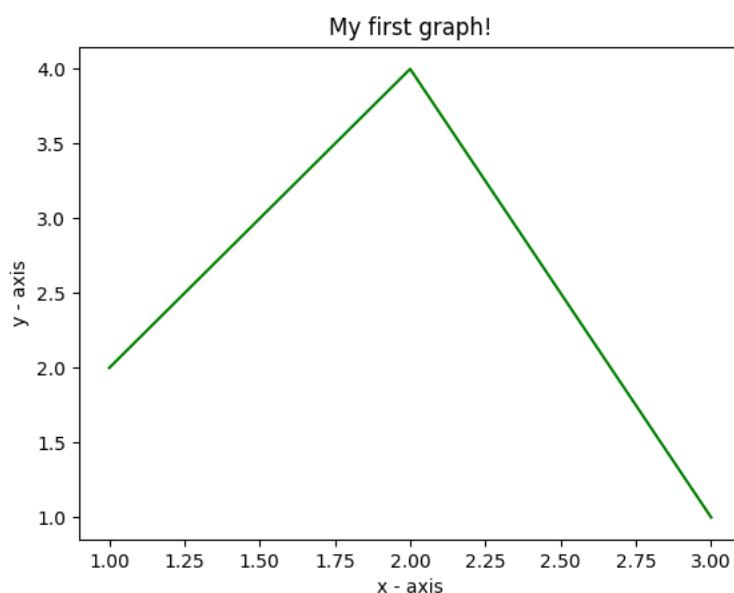
3.2.1 Graph Plotting in Python using matplotlib

In [1]:

```
# importing the required module
import matplotlib.pyplot as plt
# x axis values
x = [1,2,3]
# corresponding y axis values
y = [2,4,1]
# plotting the points
plt.plot(x, y)
# naming the x axis
plt.xlabel('x - axis')
# naming the y axis
plt.ylabel('y - axis')
# giving a title to my graph
plt.title('My first graph!')
# function to show the plot
plt.show()
```

Code: [Graph Plotting in Python using matplotlib.ipynb](#)

Output:



The code seems self explanatory. Following steps were followed:

- Define the x-axis and corresponding y-axis values as lists.
- Plot them on canvas using `.plot()` function.
- Give a name to x-axis and y-axis using `.xlabel()` and `.ylabel()` functions.
- Give a title to your plot using `.title()` function.
- Finally, to view your plot, we use `.show()` function.

3.3 Figures and subplots

Figures in matplotlib

The **Figure** is the overall window or page on which everything is drawn on. The `matplotlib.figure` module contains the Figure class. It is a top-level container for all plot elements. The Figure object is instantiated by calling the `figure()` function from the `pyplot` module –

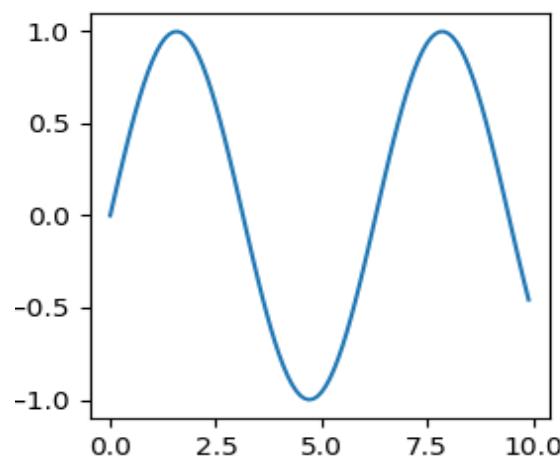
Syntax : `fig = plt.figure()`

```
In [1]: import matplotlib.pyplot as plt
import numpy as np
x = np.arange(0, 10, 0.1)
y = np.sin(x)

# figures in matplotlib using figure()
plt.figure(figsize=(3, 3))
plt.plot(x, y)
plt.show()
```

Code: [Graph Figures in matplotlib.ipynb](#)

Output:



Subplots in matplotlib

The **Matplotlib subplot()** function can be called to plot two or more plots in one figure. Matplotlib supports all kind of subplots including 2x1 vertical, 2x1 horizontal or a 2x2 grid.

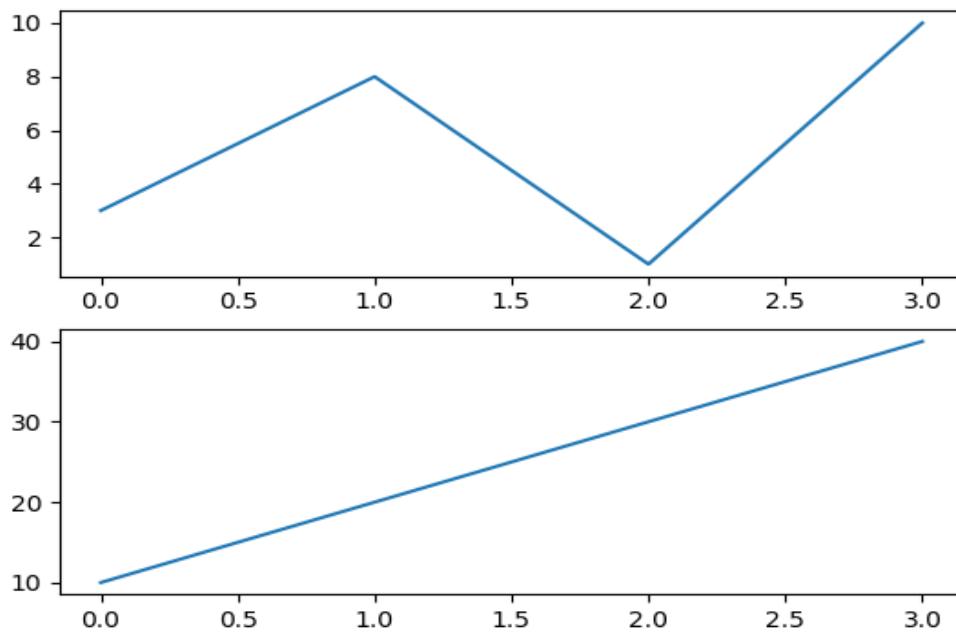
The subplot() function takes three arguments that describes the layout of the figure. The layout is organized in rows and columns, which are represented by the *first* and *second* argument. The third argument represents the index of the current plot.

```
plt.subplot(1, 2, 1)  
#the figure has 1 row, 2 columns, and this plot is the first plot.  
  
plt.subplot(1, 2, 2)  
#the figure has 1 row, 2 columns, and this plot is the second plot.
```

```
In [1]: import matplotlib.pyplot as plt  
import numpy as np  
#plot 1:  
x = np.array([0, 1, 2, 3])  
y = np.array([3, 8, 1, 10])  
plt.subplot(2, 1, 1)  
plt.plot(x,y)  
  
#plot 2:  
x = np.array([0, 1, 2, 3])  
y = np.array([10, 20, 30, 40])  
  
plt.subplot(2, 1, 2)  
plt.plot(x,y)
```

Code: [subplot theory.ipynb](#)

Output



You can draw as many plots you like on one figure, just describe the number of rows, columns, and the index of the plot.

3.4 Colors, Markers and line styles

Matplotlib Markers

You can use the keyword argument marker to emphasize each point with a specified marker:

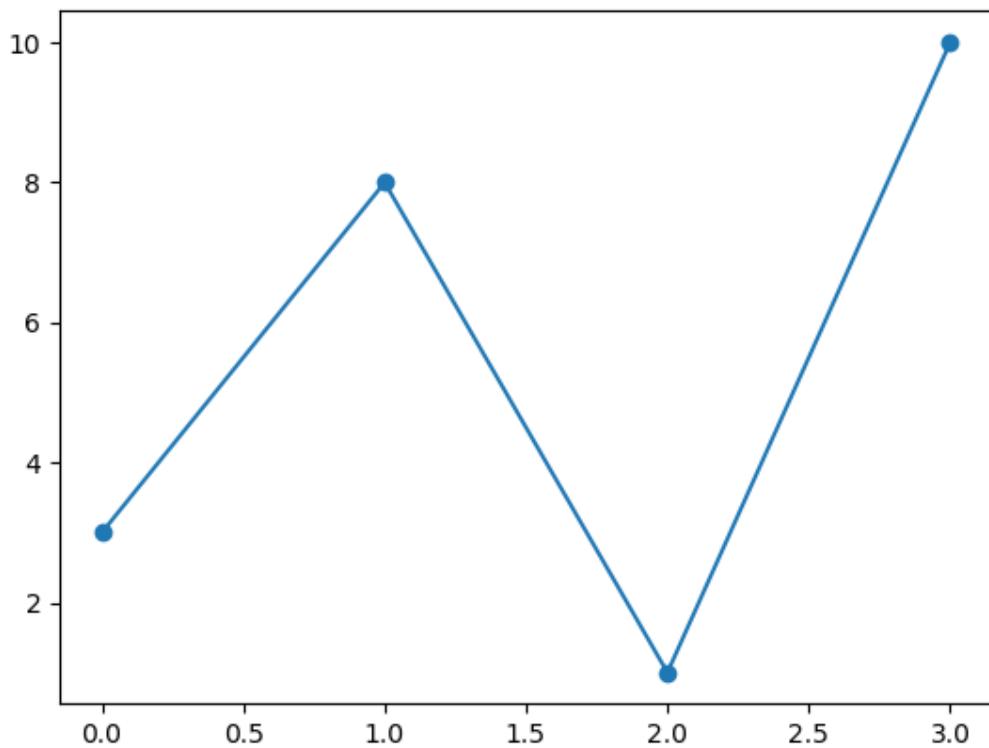
Example:

Mark each point with a circle

```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = 'o')
plt.show()
```



Matplotlib Line:

LineStyle is keyword used to change the style of the plotted line:

Shorter form of `linestyle` is `ls`.

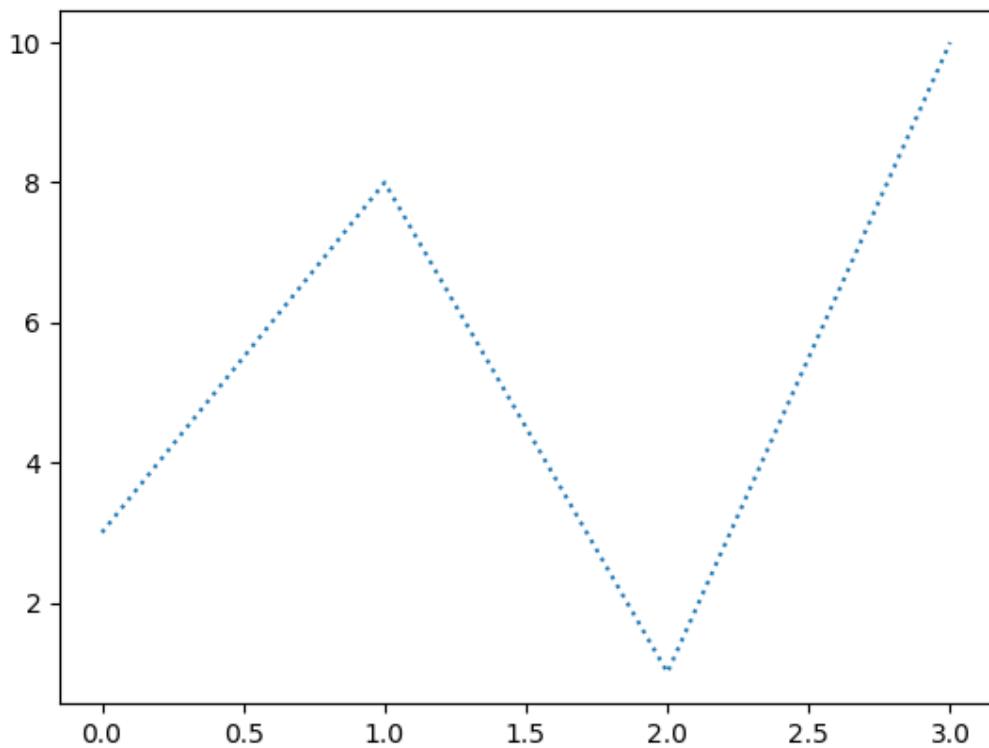
Example:

use dotted line for plot the graph.

```
import matplotlib.pyplot as plt
import numpy as np

y whole points = np.array([3, 8, 1, 10])

plt.plot(y whole points, linestyle = 'dotted')
plt.show()
```



Line Color:

You can use the keyword argument `color` or the shorter `c` to set the color of the line:

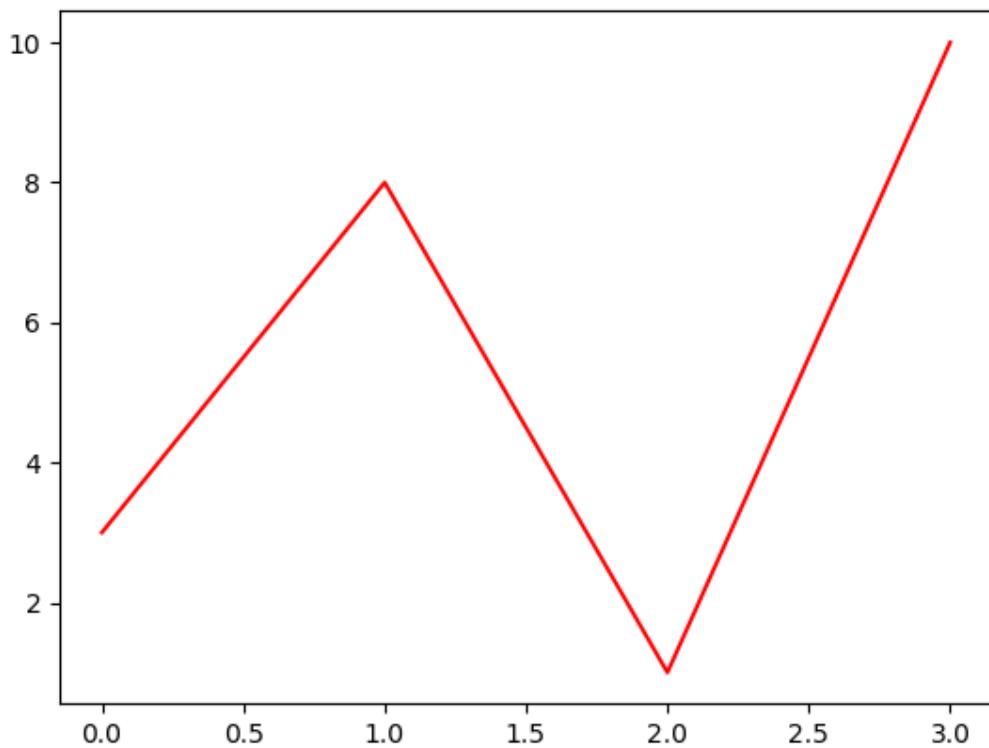
Example:

set the line color red:

```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, color = 'r')
plt.show()
```



Marker Color

You can use the keyword argument `markeredgecolor` or the shorter `mec` to set the color of the edge of the markers:

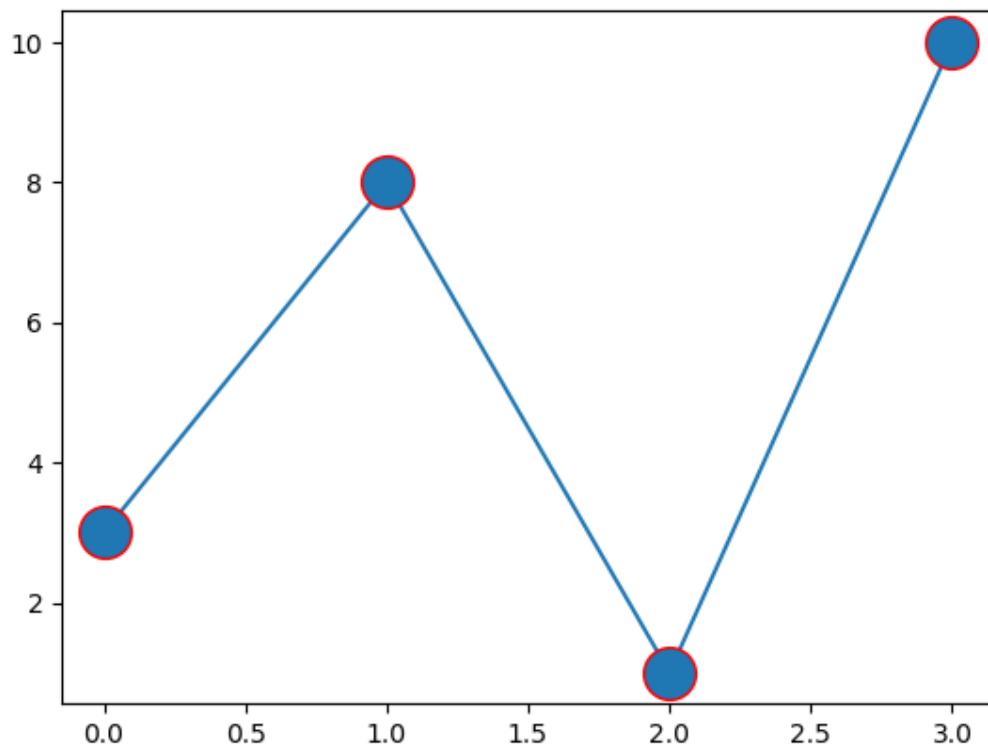
Example:

set the EDGE color to red:

```
import matplotlib.pyplot as plt
import numpy as np

y = np.array([3, 8, 1, 10])

plt.plot(y, marker = 'o', ms = 20, mec = 'r')
plt.show()
```



3.5 Matplotlib Configuration

Installation process of matplotlib

Install Matplotlib with pip

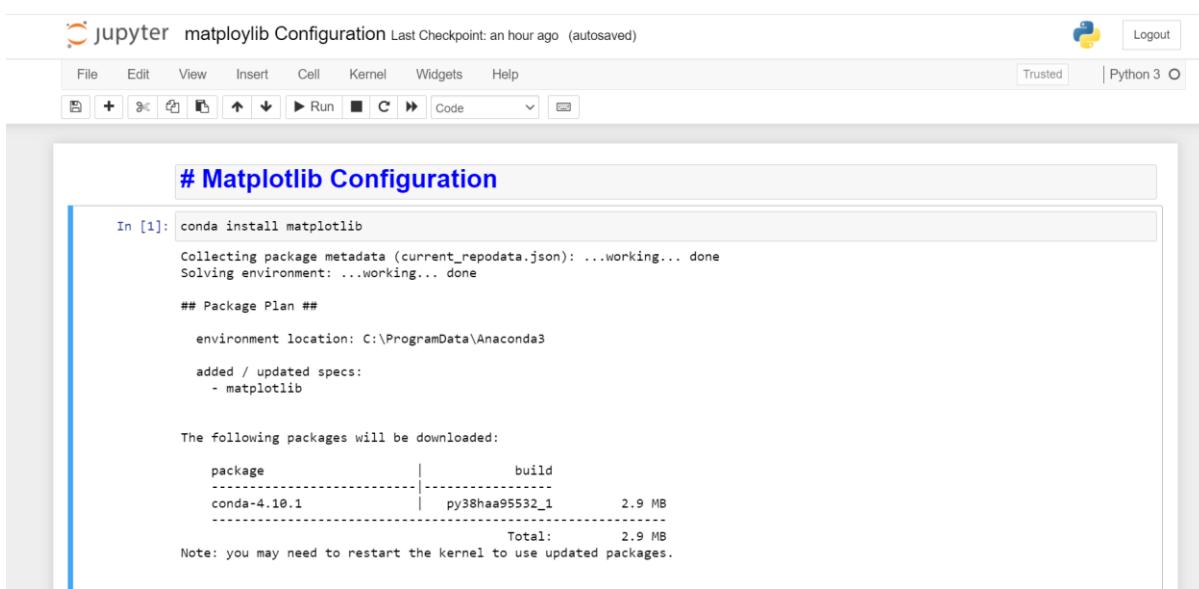
Matplotlib can also be installed using the Python package manager, pip. To install Matplotlib with pip, open a terminal window and type:

```
pip install matplotlib
```

Install Matplotlib with the **Anaconda Prompt**

Matplotlib can be installed using with the Anaconda Prompt. If the Anaconda Prompt is available on your machine, it can usually be seen in the Windows Start Menu. To install Matplotlib, open the Anaconda Prompt and type:

```
conda install matplotlib
```



The screenshot shows a Jupyter Notebook interface with the title "# Matplotlib Configuration". In cell In [1], the command `conda install matplotlib` is run, resulting in the following output:

```
In [1]: conda install matplotlib
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... done

## Package Plan ##

environment location: C:\ProgramData\Anaconda3

added / updated specs:
- matplotlib

The following packages will be downloaded:

  package          |      build
  -----|-----
  conda-4.10.1    | py38haa95532_1      2.9 MB
  -----|-----
                           Total:   2.9 MB

Note: you may need to restart the kernel to use updated packages.
```

After the installation is completed. Let's start using Matplotlib with **Jupyter Notebook**.

We will be plotting various graphs in the **Jupyter Notebook** using **Matplotlib**.

3.6 PRACTICAL: Line Plot, Bar Plot, Scatter Plot

Line Plot

Write below code in **jupyter notebook** for **Line chart** and click on **Run**

```
#Line plot using matplotlib library, here we use pyplot function of matplotlib
# importing matplotlib module
from matplotlib import pyplot as plt

# x-axis values
x = [5, 2, 9, 4, 7]

# Y-axis values
y = [10, 5, 8, 4, 2]

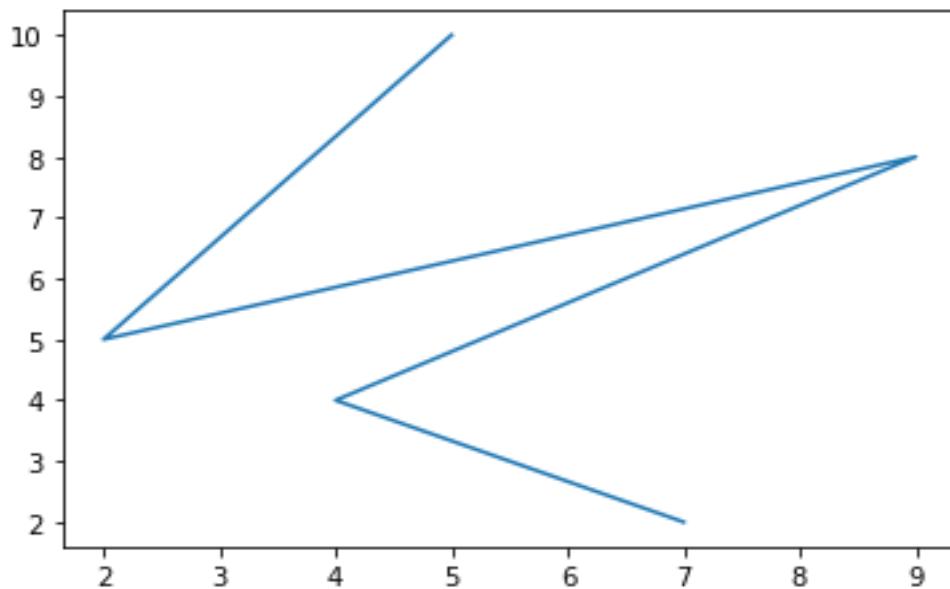
# Function to plot
plt.plot(x, y)

# function to show the plot
plt.show()
```

Code: [Line Plot.ipynb](#)

Result:

When you run the above code in jupyter notebook you will get the output shown below.



Bar Plot

Write the code given below in **jupyter notebook** for **Bar chart** and click on **Run**



```
In [ ]: # Bar Plot

# importing matplotlib module
from matplotlib import pyplot as plt

# x-axis values
x = [5, 2, 9, 4, 7]

# Y-axis values
y = [10, 5, 8, 4, 2]

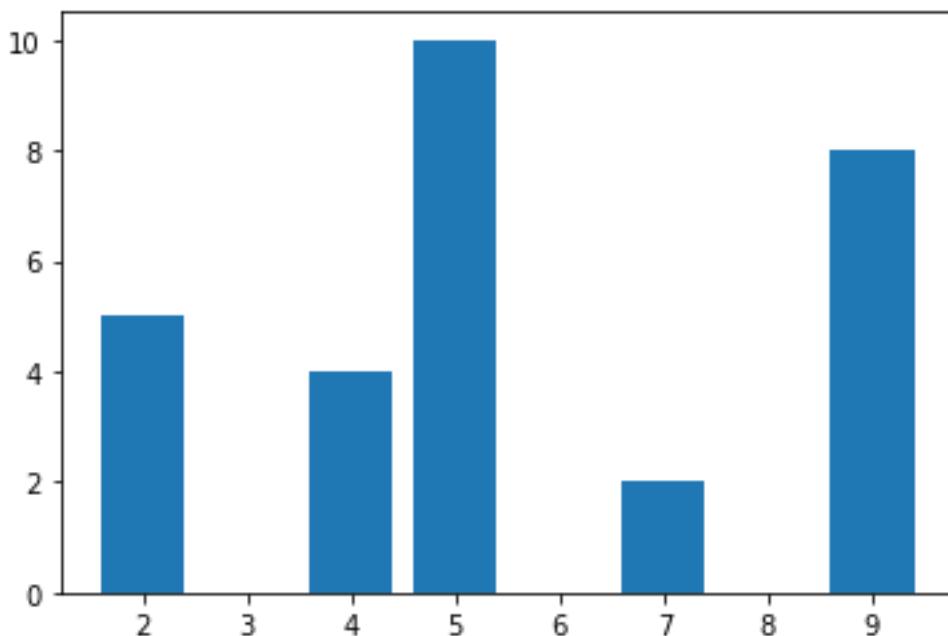
# Function to plot
plt.bar(x, y)

# function to show the plot
plt.show()
```

Code: [Bar Plot.ipynb](#)

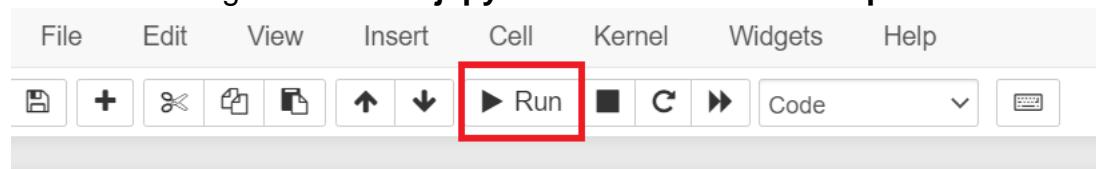
Output:

When you run the above code in jupyter notebook you will get the output given below.



Scatter Plot

Write the code given below in **jupyter notebook** for **Scatter plot** and click on **Run**



```
In [2]: #Scatter Plot
# importing matplotlib module
from matplotlib import pyplot as plt

# x-axis values
x = [5, 2, 9, 4, 7, 3, 8, 10, 1]

# Y-axis values
y = [10, 5, 8, 4, 2, 1, 6, 3, 9]

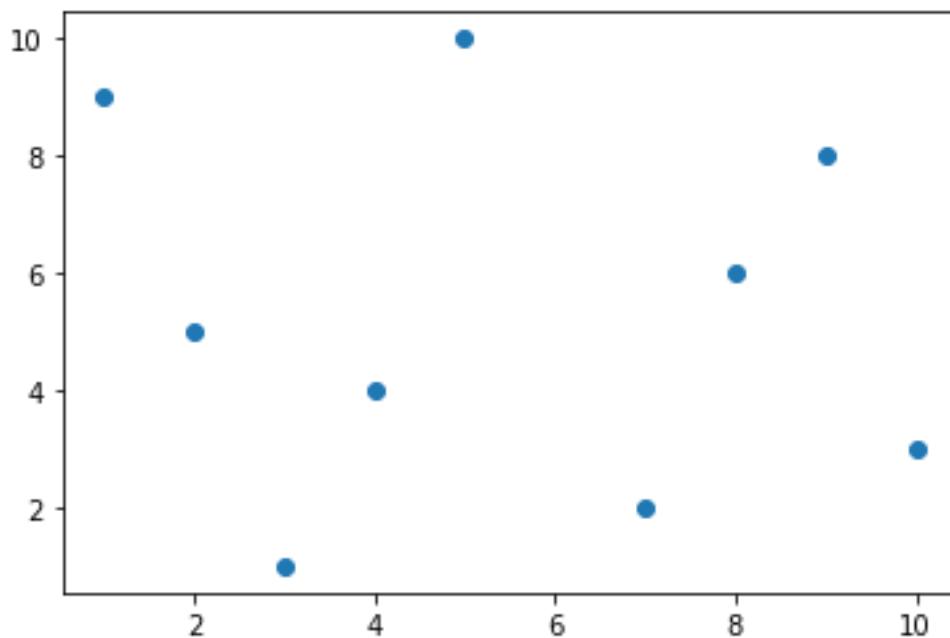
# Function to plot scatter
plt.scatter(x, y)

# function to show the plot
plt.show()
```

Code: [Scatter Plot.ipynb](#)

Result:

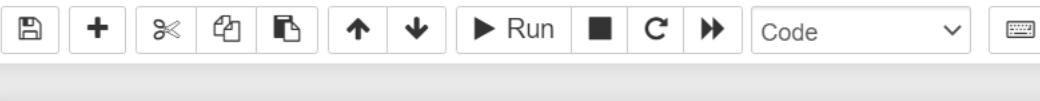
When you run the above code in jupyter notebook you will get the following output.



Pie chart in Python using Matplotlib

Matplotlib API has `pie()` function in its `pyplot` module which creates a pie chart representing the data in an array.

Let's create a simple pie chart using the `pie()` function:



```
In [1]: # Import Libraries
from matplotlib import pyplot as plt
import numpy as np

# Creating dataset
cars = ['AUDI', 'BMW', 'FORD',
        'TESLA', 'JAGUAR', 'MERCEDES']

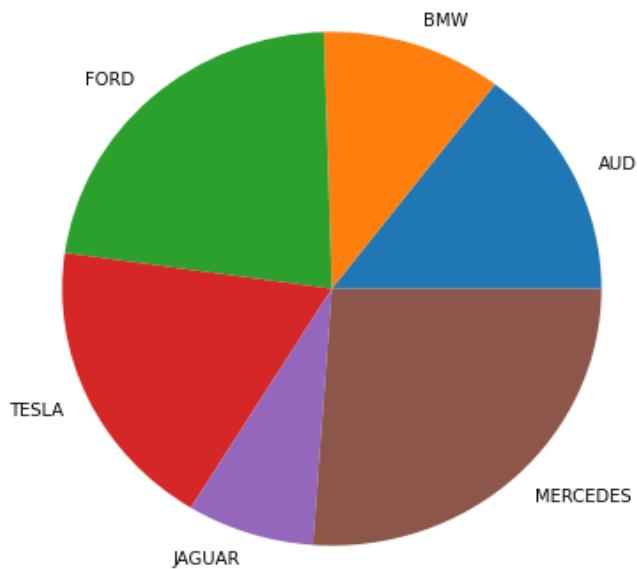
data = [23, 17, 35, 29, 12, 41]

# Creating plot
fig = plt.figure(figsize =(10, 7))
plt.pie(data, labels = cars)

# show plot
plt.show()
```

Code: [pie chart.ipynb](#)

Output:



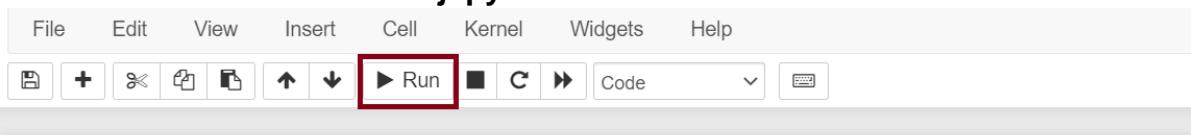
3.7 PRACTICAL: Ticks, Labels and Legends, subplots

Ticks and Labels

Ticks are the values used to show specific points on the coordinate axis. It can be a number or a string. Whenever we plot a graph, the axes adjust and take the default ticks. Matplotlib's default ticks are generally sufficient in common situations but are in no way optimal for every plot.

Here, we will see how to customize these ticks as per our need.

Write the code shown below in **jupyter notebook** and click on **Run**.



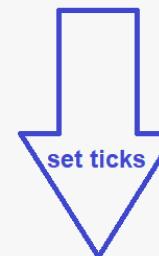
```
In [2]: # Ticks are the markers denoting data points on axes.
# importing libraries
import matplotlib.pyplot as plt
import numpy as np

# values of x and y axes
x = [5, 10, 15, 20, 25, 30, 35, 40, 45, 50]
y = [1, 4, 3, 2, 7, 6, 9, 8, 10, 5]

plt.plot(x, y, 'b')
plt.xlabel('x')
plt.ylabel('y')
# here we set the size for ticks, rotation and color value

plt.tick_params(axis="x", labelsize=18, labelrotation=-60, labelcolor="turquoise")
plt.tick_params(axis="y", labelsize=12, labelrotation=20, labelcolor="orange")

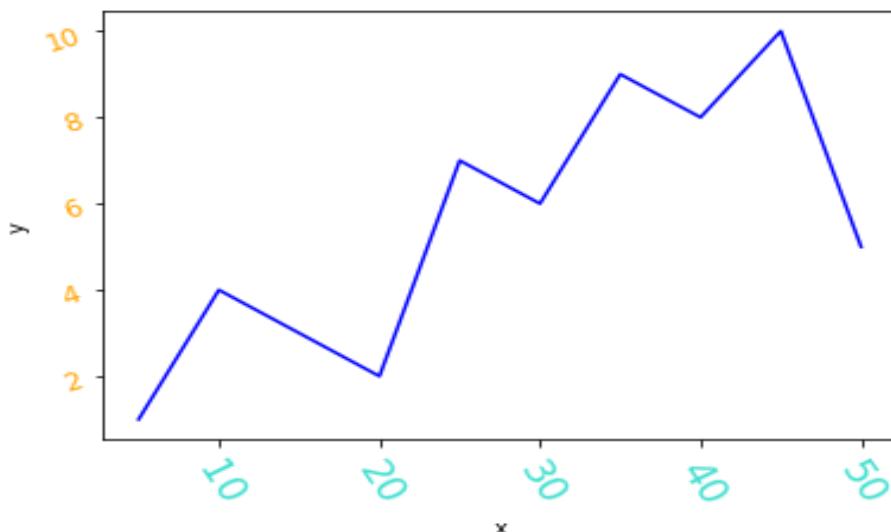
plt.show()
```



Code: [Ticks.ipynb](#)

Output:

When you run the above code in jupyter notebook you will get the following output.



Adding title and Labelling the Axes in the graph

Write the code given below in **jupyter notebook** to add title and label to the graph and click on **Run**.



In [1]: *#Adding title and Labelling the Axes in the graph*

```
# importing matplotlib module
from matplotlib import pyplot as plt

# x-axis values
x = [5, 2, 9, 4, 7]

# Y-axis values
y = [10, 5, 8, 4, 2]

# Function to plot
plt.bar(x, y)
```

Title → `plt.title("Bar graph ")`

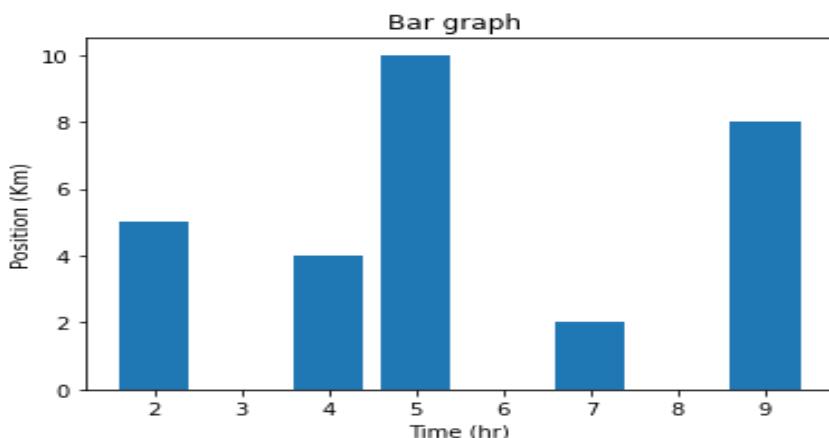
Labeling → `plt.xlabel("Time (hr)")`
`plt.ylabel("Position (Km)")`

```
# function to show the plot
plt.show()
```

Code: [Labelling.ipynb](#)

Result:

When you run the above code in jupyter notebook you will get the following result.



Adding Legend in the graph

A legend is an area describing the elements of the graph. In the matplotlib library, there's a function called **legend()** which is used to place a legend on the axes.

Matplotlib.pyplot.legend()

Write the code given below in **jupyter notebook** to add Legend to the graph and click on **Run**.



```
In [1]: #Adding Legend in the graph
# importing modules
import numpy as np
import matplotlib.pyplot as plt

# Y-axis values
y1 = [2, 3, 4.5]

# Y-axis values
y2 = [1, 1.5, 5]

# Function to plot
plt.plot(y1)
plt.plot(y2)

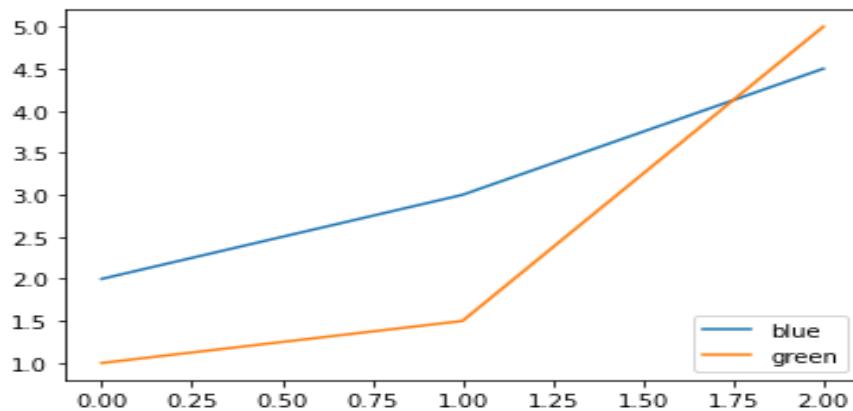
# Function add a Legend
plt.legend(["blue", "green"], loc ="lower right")

# function to show the plot
plt.show()
```

Legend

Code: [Legends.ipynb](#)

Result:



Display Multiple Plots using subplot() function

With the **subplots()** function you can draw multiple plots in one figure



```
In [1]: #subplots
import matplotlib.pyplot as plt
import numpy as np

#plot 1:
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(1, 2, 1)
plt.plot(x,y)

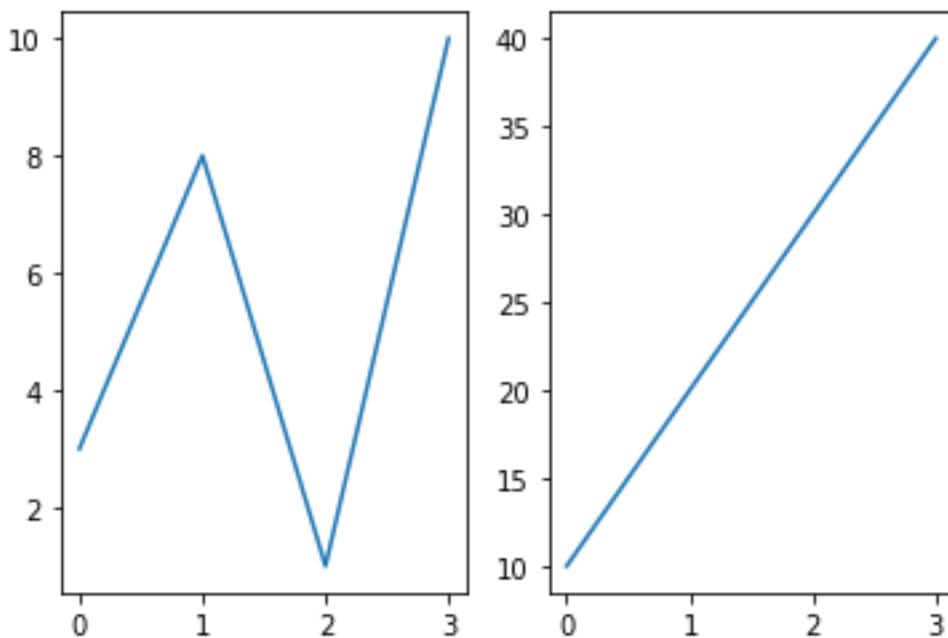
#plot 2:
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(1, 2, 2)
plt.plot(x,y)

plt.show()
```

Code: subplot.ipynb

Result:



3.8 PRACTICAL: Histograms and Binning

A histogram is an accurate graphical representation of the distribution of numerical data. It is an estimate of the probability distribution of a continuous variable (quantitative variable) and was first introduced by Karl Pearson. It is a kind of bar graph.

To construct a histogram, the first step is to "bin" the range of values — that is, divide the entire range of values into a series of intervals — and then count how many values fall into each interval. The bins are usually specified as consecutive, non-overlapping intervals of a variable. The bins (intervals) must be adjacent, and are often (but are not required to be) of equal size.

Bins are also sometimes called "intervals", "classes", or "buckets".

The heights of the bars tell us how many data points are in each bin.

For example, this histogram says that there are 8 pumpkins whose mass is between 6 and 9 kilograms.

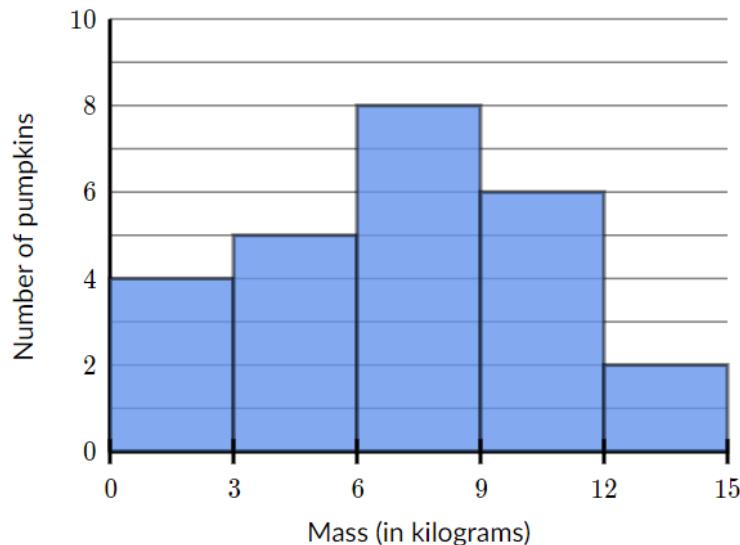


Image: Histogram

Reference: <https://www.khanacademy.org/math/statistics-probability/displaying-describing-data/quantitative-data-graphs/a/histograms-review>

Exercise:



Observe the histogram and find out how many Houses are there in Prince Street?

Answer: _____

Create Histogram in Python using matplotlib

In Matplotlib, we use the `hist()` function to create histograms. The `hist()` function will use an array of numbers to create a histogram, the array is sent into the function as an argument.

The `hist()` function will read the array and produce a histogram:

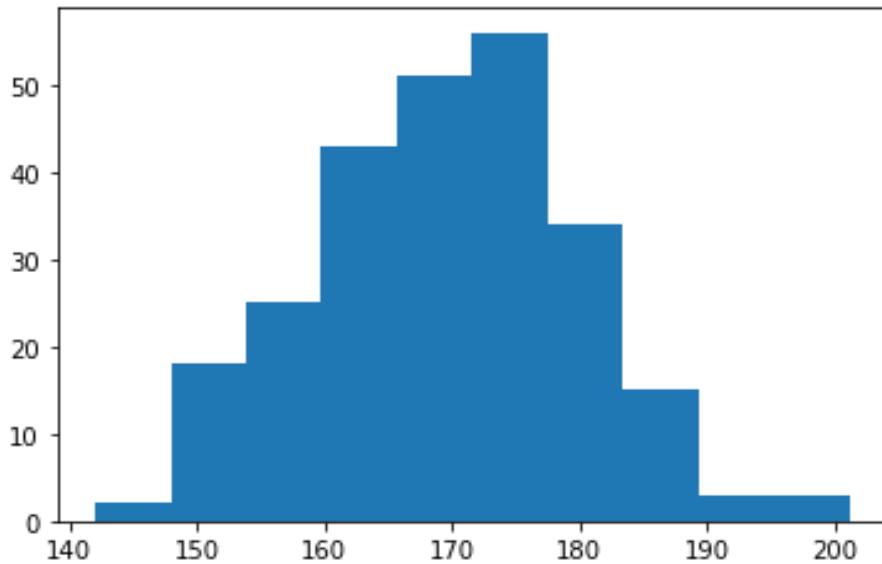
File Edit View Insert Cell Kernel Widgets Help

File + × ⌂ ⌄ ⌅ ⌆ ⌇ Run Cell Code

```
In [1]: #Histogram
import matplotlib.pyplot as plt
import numpy as np

x = np.random.normal(170, 10, 250)
#The hist() function will read the array and produce a histogram:
plt.hist(x)
plt.show()
```

Output:



Binning

Set the **bins** value to the above histogram

File Edit View Insert Cell Kernel Widgets Help

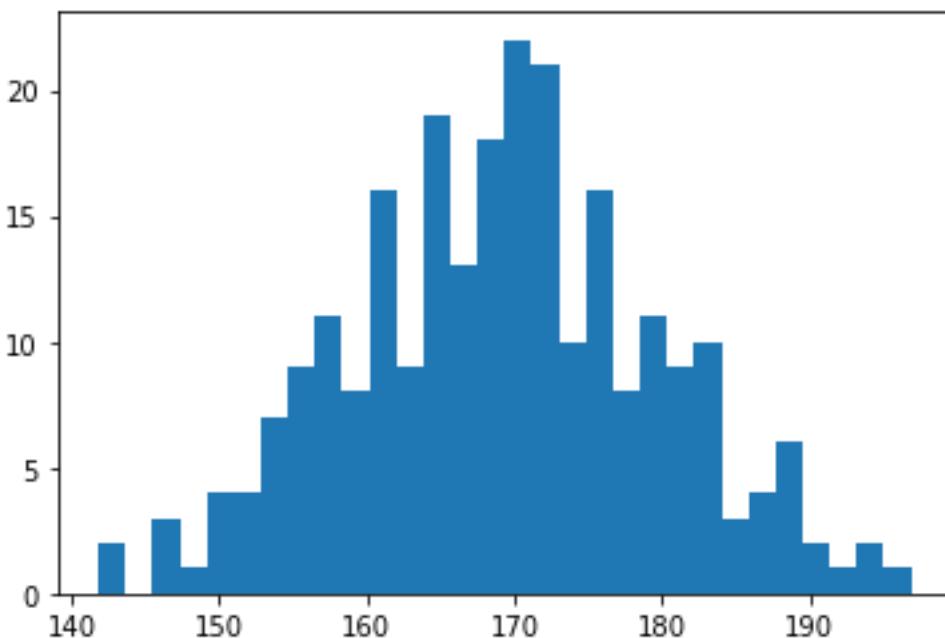
File + × ⌂ ⌄ ⌅ ⌆ ⌇ Run Cell Code

```
In [1]: #Histogram
import matplotlib.pyplot as plt
import numpy as np

data = np.random.normal(170, 10, 250)
#The hist() function will read the array and produce a histogram:
#set the bin value 30
plt.hist(data, bins=30);
plt.show()
```

Code: [hist_bin.ipynb](#)

Output:



3.9 PRACTICAL: Text and Annotations

Create an annotation: a piece of text referring to a data point.

Creating a good visualization involves guiding the reader so that the figure tells a story. In some cases, this story can be told in an entirely visual manner, without the need for added text, but in others, small textual cues and labels are necessary. Perhaps the most basic types of annotations you will use are axes labels and titles, but the options go beyond this. Let's take a look at some data and how we might visualize and annotate it to help convey interesting information.

Let's try to understand it by one example.

The **annotate()** function in pyplot module of matplotlib library is used to annotate the point xy with texts

File Edit View Insert Cell Kernel Widgets Help

File New Open Save Run Cell Kernel Help

```
In [2]: #import matplotlib.pyplot
import matplotlib.pyplot as plt
import numpy as np

#draw plot as fig
fig, ax = plt.subplots()

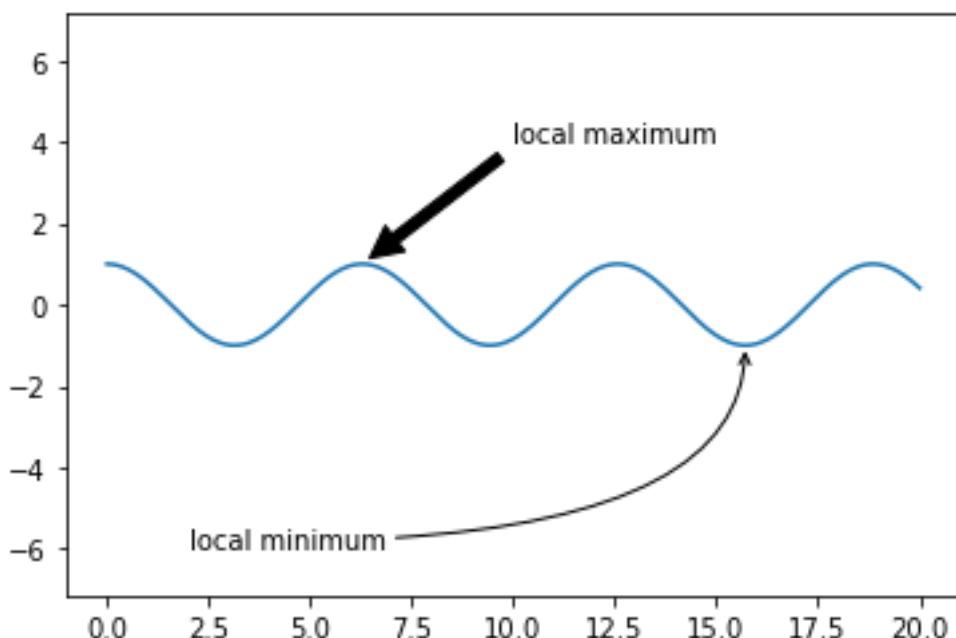
x = np.linspace(0, 20, 1000)
ax.plot(x, np.cos(x))
ax.axis('equal')
#text annotation to the plot where it indicate maximum value of the curve
ax.annotate('local maximum', xy=(6.28, 1), xytext=(10, 4),
            arrowprops=dict(facecolor='black', shrink=0.05))
#text annotation to the plot where it indicate minimum value of the curve
ax.annotate('local minimum', xy=(5 * np.pi, -1), xytext=(2, -6),
            arrowprops=dict(arrowstyle="->",
                           connectionstyle="angle3,angleA=0,angleB=-90"));


```

Code: [text and annotate.ipynb](#)

Here annotation mark is the simple arrow.

Output



In above example you can observe text annotate local maximum and local minimum by arrow

Example 2

File Edit View Insert Cell Kernel Widgets Help

File + × ↻ ↺ ⌂ Run ▶ C ▶ Code

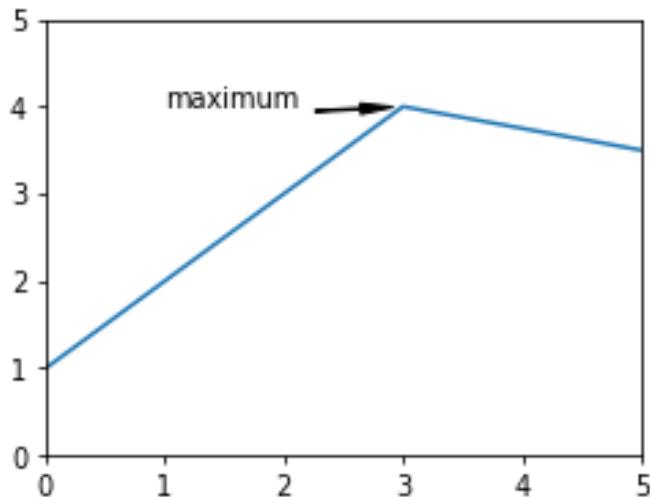
```
In [3]: import matplotlib.pyplot as plt
w = 4
h = 3
d = 70
plt.figure(figsize=(w, h), dpi=d)
plt.axis([0, 5, 0, 5])
x = [0, 3, 5]
y = [1, 4, 3.5]
label_x = 1
label_y = 4
arrow_x = 3
arrow_y = 4

arrow_properties = dict(
    facecolor="black", width=0.5,
    headwidth=4, shrink=0.1)

plt.annotate(
    "maximum", xy=(arrow_x, arrow_y),
    xytext=(label_x, label_y),
    arrowprops=arrow_properties)
plt.plot(x, y)
```

Code: [text and annotate.ipynb](#)

Output:



3.10 PRACTICAL: Three-Dimensional Plotting in Matplotlib

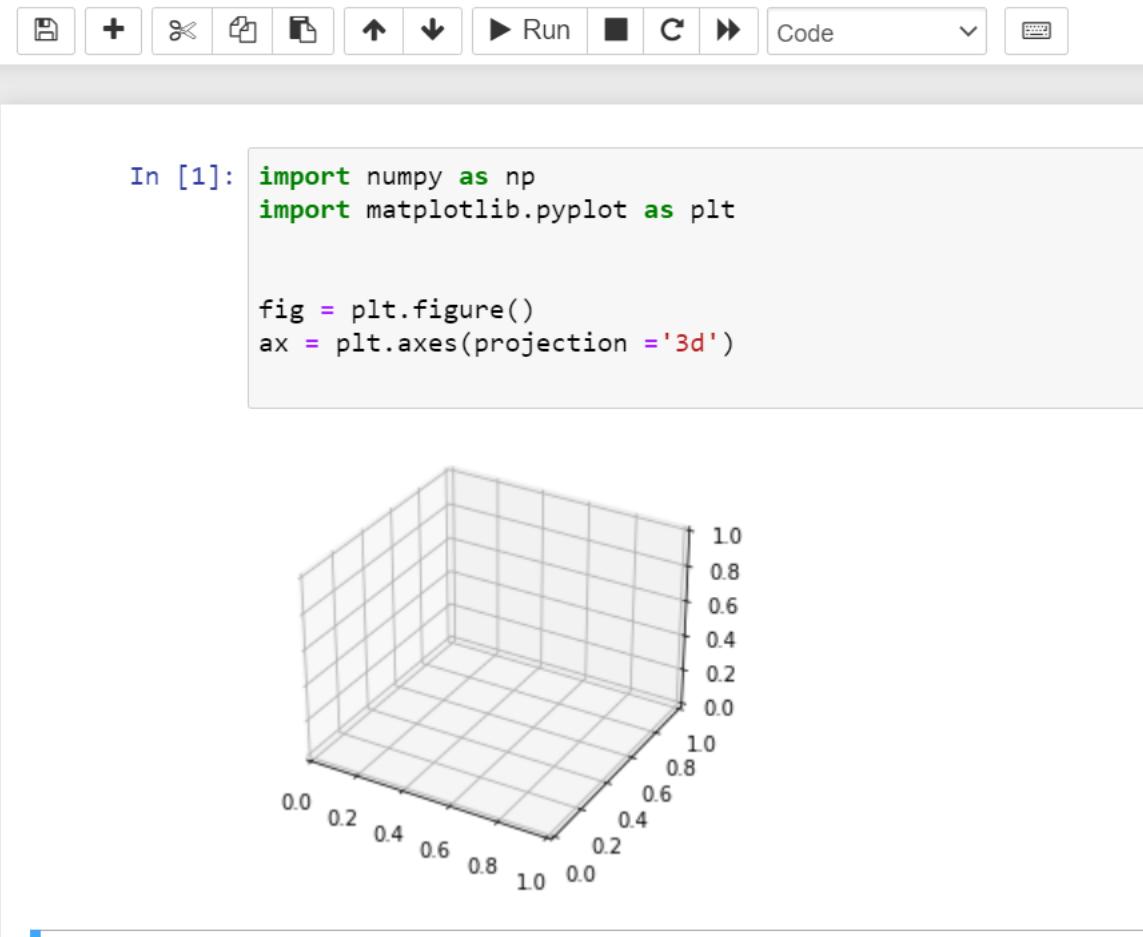
Three-dimensional plots are enabled by importing the mplot3d toolkit, included with the main Matplotlib installation:

Step 1:

```
from mpl_toolkits import mplot3d
```

Once this submodule is imported, three-dimensional axes can be created by passing the keyword `projection='3d'` to any of the normal axes creation routines:

Step 2: create simple 3D graph



The screenshot shows a Jupyter Notebook interface. At the top, there is a toolbar with various icons: a file icon, a plus sign, a crossed-out icon, a refresh icon, a file icon, a double arrow icon, an up arrow icon, a down arrow icon, a run button labeled "Run", a stop button, a clear button, a next button, a "Code" dropdown menu, and a cell type dropdown menu. Below the toolbar is a grey header bar. The main area contains a code cell with the following content:

```
In [1]: import numpy as np
import matplotlib.pyplot as plt

fig = plt.figure()
ax = plt.axes(projection ='3d')
```

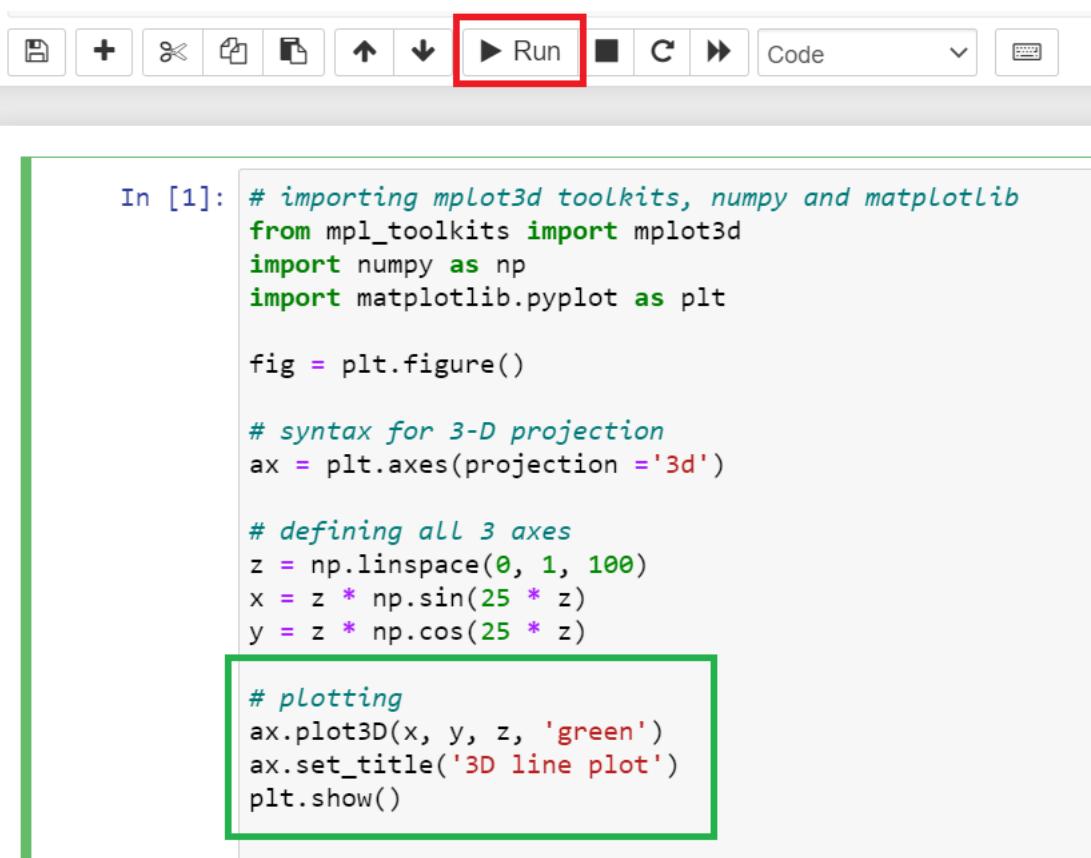
Below the code cell is a 3D plot of a unit cube. The x-axis ranges from 0.0 to 1.0, the y-axis from 0.0 to 1.0, and the z-axis from 0.0 to 1.0. The plot is a wireframe cube centered at (0.5, 0.5, 0.5).

At the bottom of the notebook interface, there is a horizontal line with a blue square icon on the left and the text "Code: 3D plotting matplotlib.ipynb" on the right.

Plotting 3-D Lines and Points

Graph with lines and point are the simplest 3 dimensional graph. `ax.plot3d` and `ax.scatter` are the function to plot line and point graph respectively.

Write down the code given below in jupyter notebook and click on Run



The screenshot shows a Jupyter Notebook interface. At the top, there's a toolbar with various icons: a file icon, a plus sign, a delete icon, a copy icon, a paste icon, up and down arrows, a red 'Run' button (which is highlighted with a red box), a stop button, a clear button, a forward button, a dropdown menu labeled 'Code', and a terminal icon.

In [1]:

```
# importing mplot3d toolkits, numpy and matplotlib
from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt

fig = plt.figure()

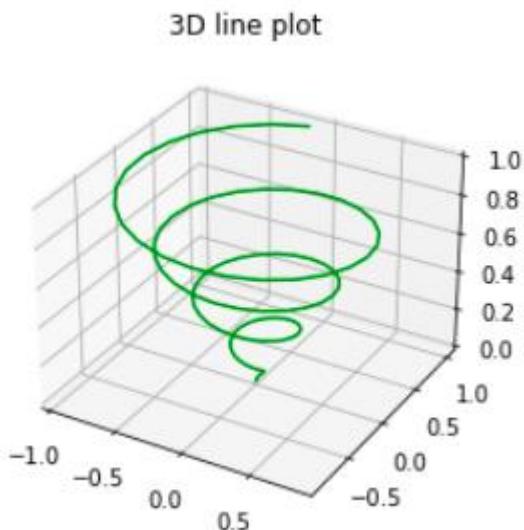
# syntax for 3-D projection
ax = plt.axes(projection ='3d')

# defining all 3 axes
z = np.linspace(0, 1, 100)
x = z * np.sin(25 * z)
y = z * np.cos(25 * z)

# plotting
ax.plot3D(x, y, z, 'green')
ax.set_title('3D line plot')
plt.show()
```

Code: Plotting 3-D Lines and Points.ipynb

Output:



Unit 4: Linux Fundamentals

Learning Outcomes:

- Familiar with Linux Environment
- Understand Linux Kernel, commands and functionalities

4.1 Introduction to Linux

Linux is a Unix-like computer operating system assembled under the model of free and open-source software development and distribution. The defining component of Linux is the Linux kernel, an operating system kernel first released 5 October 1991 by Linus Torvalds.

Linux was originally developed as a free operating system for Intel x86-based personal computers. It has since been ported to more computer hardware platforms than any other operating system. It is a leading operating system on servers and other big iron systems such as mainframe computers and supercomputers more than 90% of today's 500 fastest supercomputers run some variant of Linux, including the 10 fastest. Linux also runs on embedded systems (devices where the operating system is typically built into the firmware and highly tailored to the system) such as mobile phones, tablet computers, network routers, televisions and video game consoles; the Android system in wide use on mobile devices is built on the Linux kernel.

Linux is one of popular version of UNIX operating System. It is open source as its source code is freely available. It is free to use. Linux was designed considering UNIX compatibility. Its functionality list is quite similar to that of UNIX.

Components of Linux System

Linux Operating System has primarily three components

- **Kernel** – Kernel is the core part of Linux. It is responsible for all major activities of this operating system. It consists of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.
- **System Library** – System libraries are special functions or programs using which application programs or system utilities accesses Kernel's features.

These libraries implement most of the functionalities of the operating system and do not require kernel module's code access rights.

- **System Utility** – System Utility programs are responsible to do specialized, individual level tasks.

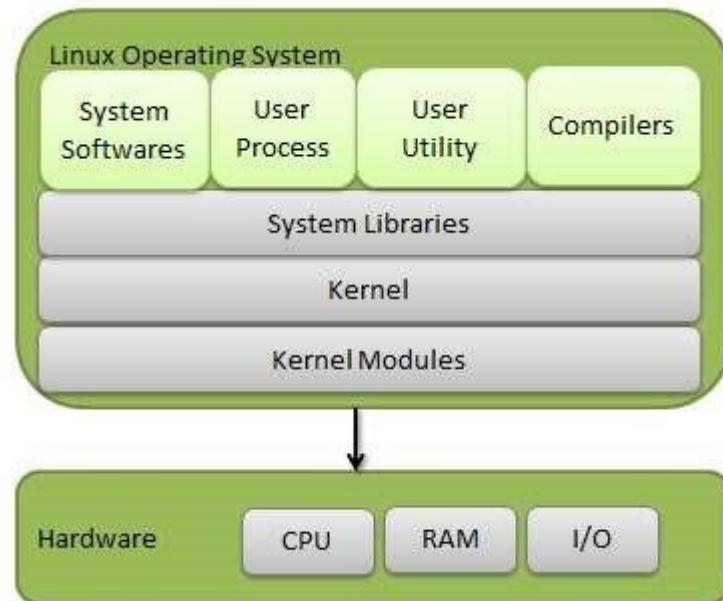


Image: LINUX Operating System [1]

Reference: https://www.tutorialspoint.com/operating_system/os_linux.htm

Kernel Mode vs User Mode

Kernel component code executes in a special privileged mode called **kernel mode** with full access to all resources of the computer. This code represents a single process, executes in single address space and do not require any context switch and hence is very efficient and fast. Kernel runs each processes and provides system services to processes, provides protected access to hardware to processes.

Support code which is not required to run in kernel mode is in System Library. User programs and other system programs works in **User Mode** which has no access to system hardware and kernel code. User programs/ utilities use System libraries to access Kernel functions to get system's low-level tasks.

Basic Features

Following are some of the important features of Linux Operating System.

Portable - Portability means software can work on different types of hardware in same way. Linux kernel and application programs support their installation on any kind of hardware platform.

Open Source - Linux source code is freely available, and it is community-based development project. Multiple Teams works in collaboration to enhance the capability of Linux operating system and it is continuously evolving.

Multi-User - Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.

Multiprogramming - Linux is a multiprogramming system means multiple applications can run at same time.

Hierarchical File System - Linux provides a standard file structure in which system files/ user files are arranged.

Shell - Linux provides a special interpreter program which can be used to execute commands of the operating system. It can be used to do various types of operations, call application programs etc.

Security - Linux provides user security using authentication features like password protection/ controlled access to specific files/ encryption of data.

Linux Advantages

1. **Low cost:** You don't need to spend time and money to obtain licenses since Linux and much of its software come with the GNU General Public License. You can start to work immediately without worrying that your software may stop working anytime because the free trial version expires. Additionally, there are large repositories from which you can freely download high quality software for almost any task you can think of.

2. **Stability:** Linux doesn't need to be rebooted periodically to maintain performance levels. It doesn't freeze up or slow down over time due to memory leaks and such. Continuous up-times of hundreds of days (up to a year or more) are not uncommon.

3. **Performance:** Linux provides persistent high performance on workstations and on networks. It can handle unusually large numbers of users simultaneously and can make old computers sufficiently responsive to be useful again.

4. **Network friendliness:** Linux was developed by a group of programmers over the Internet and has therefore strong support for network functionality; client and server systems can be easily set up on any computer running Linux. It can perform tasks such as network backups faster and more reliably than alternative systems.

5. **Flexibility:** Linux can be used for high performance server applications, desktop applications, and embedded systems. You can save disk space by only installing the components needed for a particular use. You can restrict the use of specific computers by installing for example only selected office applications instead of the whole suite.

6. **Compatibility:** It runs all common UNIX software packages and can process all common file formats.

7. **Choice:** The large number of Linux distributions gives you a choice. Each distribution is developed and supported by a different organization. You can pick the one you like best; the core functionalities are the same; most software runs on most distributions.

8. Fast and easy installation: Most Linux distributions come with user-friendly installation and setup programs. Popular Linux distributions come with tools that make installation of additional software very user friendly as well.

9. Full use of hard disk: Linux continues work well even when the hard disk is almost full.

10. Multi-tasking: Linux is designed to do many things at the same time; e.g., a large printing job in the background won't slow down your other work.

11. Security: Linux is one of the most secure operating systems. —Wall\$! and flexible file access permission systems prevent access by unwanted visitors or viruses. Linux users have the option to select and safely download software, free of charge, from online repositories containing thousands of high-quality packages. No purchase transactions requiring credit card numbers or other sensitive personal information are necessary.

12. Open Source: If you develop software that requires knowledge or modification of the operating system code, LINUX's source code is at your fingertips. Most Linux applications are Open Source as well.

Linux Distribution (Operating System) Names

A few popular names:

1. Redhat Enterprise Linux
2. Fedora Linux
3. Debian Linux
4. Suse Enterprise Linux
5. Ubuntu Linux

Common things between Linux & UNIX

Both share many common applications such as:

1. GUI, file, and window managers (KDE, Gnome)
2. Shells (ksh, csh, bash)
3. Various office applications such as OpenOffice.org
4. Development tools (perl, php, python, GNU c/c++ compilers)
5. Posix interface

Layered Architecture:

Linux System Architecture is consisting of following layers

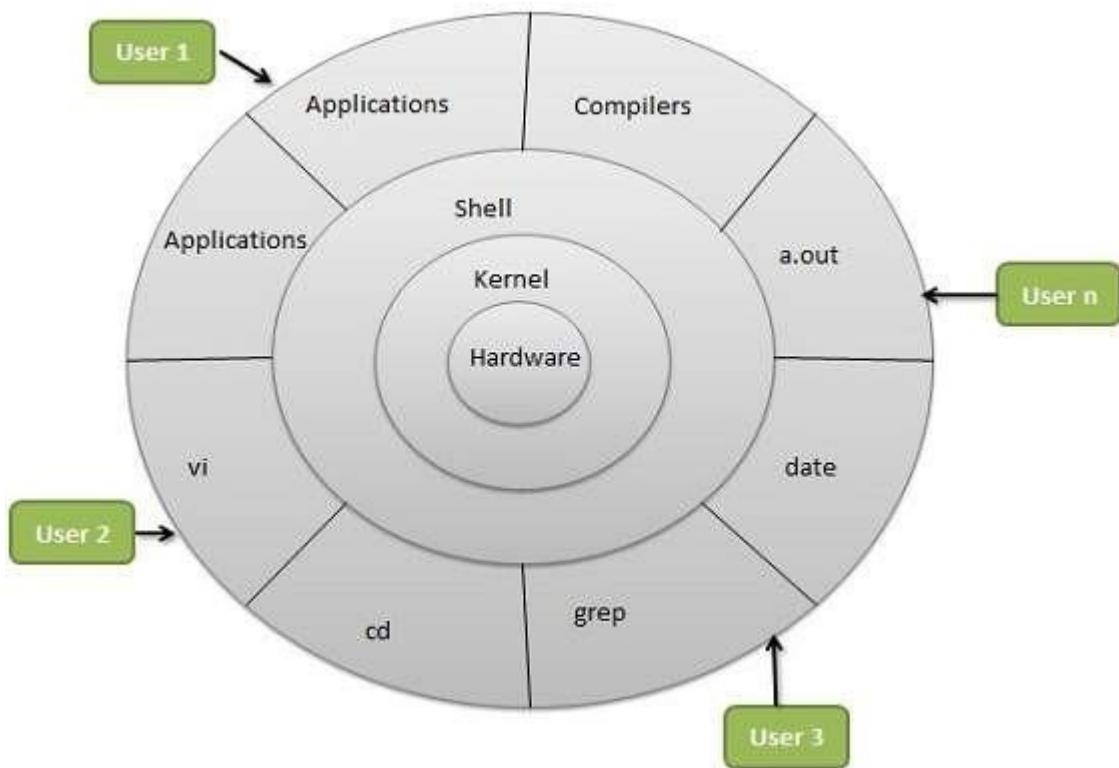


Image: Basic Block diagram of Layered Architecture of LINUX

Reference: https://www.tutorialspoint.com/operating_system/os_linux.html

Hardware layer - Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc).

Kernel - Core component of Operating System, interacts directly with hardware, provides low level services to upper layer components.

Shell - An interface to kernel, hiding complexity of kernel's functions from users. Takes commands from user and executes kernel's functions.

Utilities - Utility programs giving user most of the functionalities of an operating systems.

LINUX File system

Linux file structure files are grouped according to purpose. Ex: commands, data files, documentation. Parts of a Unix directory tree are listed below. All directories are grouped under the root entry "/". That part of the directory tree is left out of the below diagram.

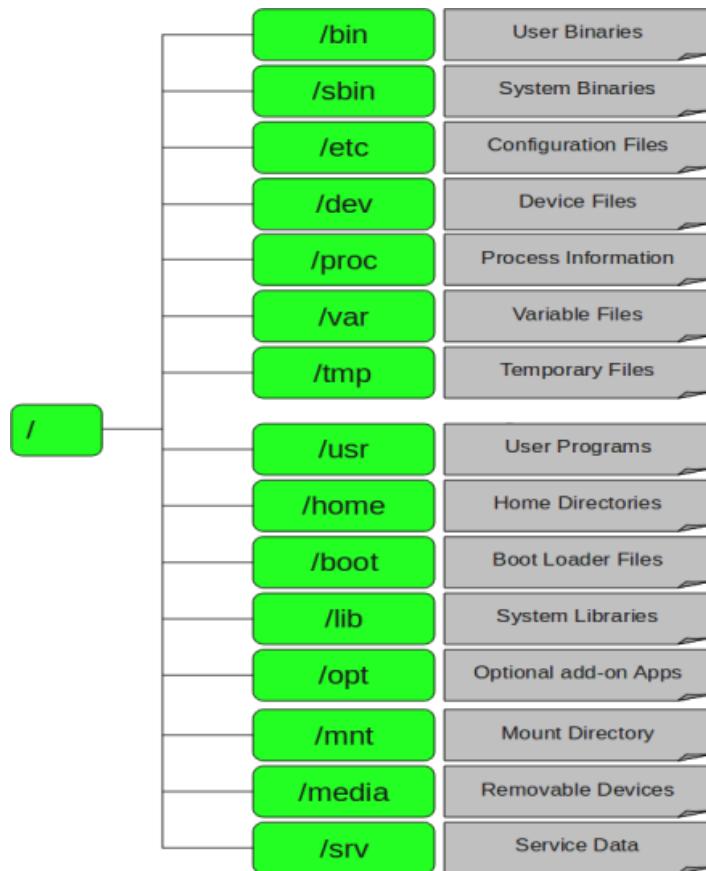


Image: LINUX File Structure

Reference: https://www.tutorialspoint.com/operating_system/os_linux.html

1. / – Root

- Every single file and directory starts from the root directory.
- Only root user has write privilege under this directory.
- Please note that /root is root user's home directory, which is not same as /.

2. /bin – User Binaries

- Contains binary executables.
- Common linux commands you need to use in single-user modes are located under this directory.
- Commands used by all the users of the system are located here.
- For example: ps, ls, ping, grep, cp.

3. /sbin – System Binaries

- Just like /bin, /sbin also contains binary executables.

- But, the linux commands located under this directory are used typically by system administrator, for system maintenance purpose.
- For example: iptables, reboot, fdisk, ifconfig, swapon

4. /etc – Configuration Files

- Contains configuration files required by all programs.
- This also contains startup and shutdown shell scripts used to start/stop individual programs.
- For example: /etc/resolv.conf, /etc/logrotate.conf

5. /dev – Device Files

- Contains device files.
- These include terminal devices, usb, or any device attached to the system.
- For example: /dev/tty1, /dev/usbmon0

6. /proc – Process Information

- Contains information about system process.
- This is a pseudo filesystem contains information about running process. For example: /proc/{pid} directory contains information about the process with that particular pid.
- This is a virtual filesystem with text information about system resources. For example:
`/proc/uptime`

7. /var – Variable Files

- var stands for variable files.
- Content of the files that are expected to grow can be found under this directory.
- This includes — system log files (/var/log); packages and database files (/var/lib); emails (/var/mail); print queues (/var/spool); lock files (/var/lock); temp files needed across reboots (/var/tmp);

8. /tmp – Temporary Files

- Directory that contains temporary files created by system and users.
- Files under this directory are deleted when system is rebooted.

9. /usr – User Programs

- Contains binaries, libraries, documentation, and source-code for second level programs.
- /usr/bin contains binary files for user programs. If you can't find a user binary under
 - /bin, look under /usr/bin. For example: at, awk, cc, less, scp
 - /usr/sbin contains binary files for system administrators. If you can't find a system binary under /sbin, look under /usr/sbin. For example: atd, cron, sshd, useradd, userdel
 - /usr/lib contains libraries for /usr/bin and /usr/sbin
 - /usr/local contains users programs that you install from source. For example, when you
 - install apache from source, it goes under /usr/local/apache2

10. /home – Home Directories

- Home directories for all users to store their personal files.
- For example: /home/john, /home/nikita

11. /boot – Boot Loader Files

- Contains boot loader related files.
- Kernel initrd, vmlinuz, grub files are located under /boot
- For example: initrd.img-2.6.32-24-generic, vmlinuz-2.6.32-24-generic

12. /lib – System Libraries

- Contains library files that supports the binaries located under /bin and /sbin
- Library filenames are either lib* or lib*.so.*
- For example: lib-2.11.1.so, libncurses.so.5.7

13. /opt – Optional add-on Applications

- opt stands for optional.
- Contains add-on applications from individual vendors.
- add-on applications should be installed under either /opt/ or /opt/ sub-directory.

14. /mnt – Mount Directory

- Temporary mount directory where sysadmins can mount filesystems.

15. /media – Removable Media Devices

- Temporary mount directory for removable devices.
- For examples, /media/cdrom for CD-ROM; /media/floppy for floppy drives;
- /media/cdrecorder for CD writer

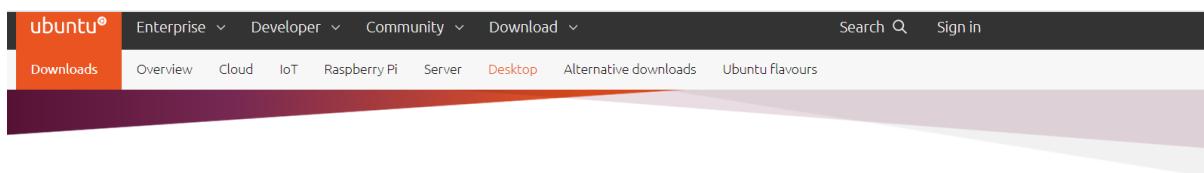
16. /srv – Service Data

- srv stands for service.
- Contains server specific services related data.
- For example, /srv/cvs contains CVS related data.

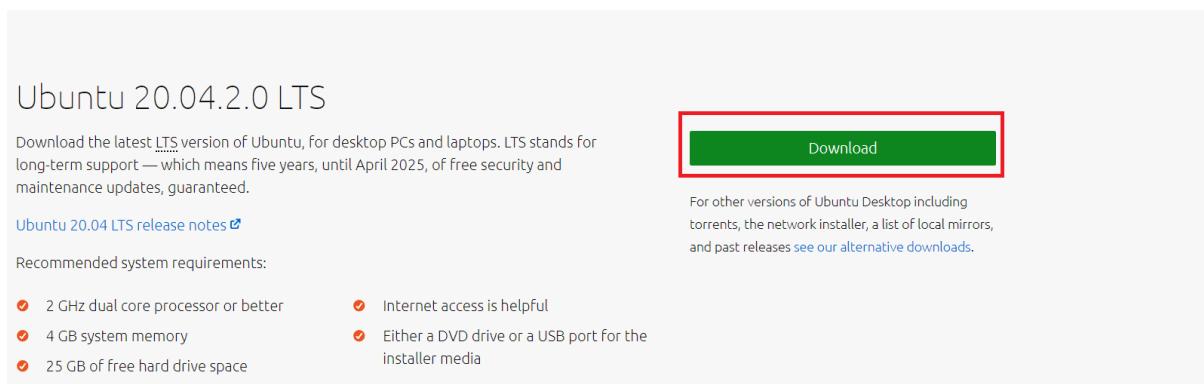
4.2 PRACTICAL: Installation of Ubuntu

Downloading

Visit this link to <http://www.ubuntu.com/download/desktop> download Ubuntu.



Download Ubuntu Desktop

A screenshot of the Ubuntu 20.04.2.0 LTS download page. The page title is "Ubuntu 20.04.2.0 LTS". Below the title, there is a brief description of what LTS means: "Download the latest LTS version of Ubuntu, for desktop PCs and laptops. LTS stands for long-term support — which means five years, until April 2025, of free security and maintenance updates, guaranteed." To the right of this text is a prominent green "Download" button, which is outlined with a red border. Further down the page, there are sections for "Ubuntu 20.04 LTS release notes" and "Recommended system requirements". The system requirements list includes:

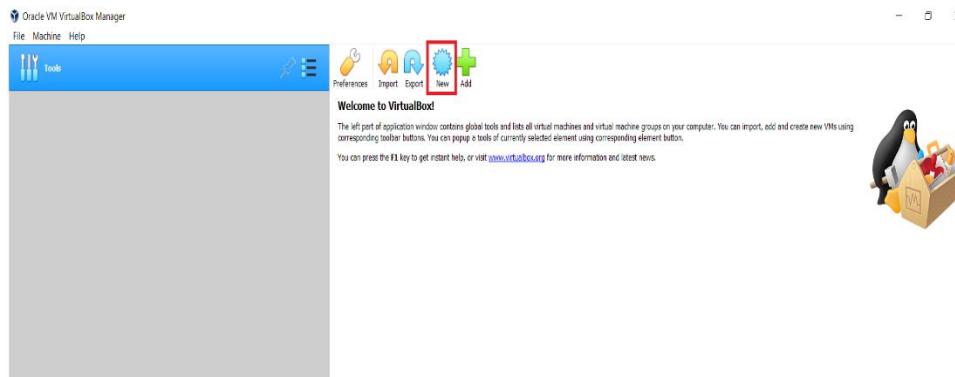
- 2 GHz dual core processor or better
- 4 GB system memory
- 25 GB of free hard drive space
- Internet access is helpful
- Either a DVD drive or a USB port for the installer media

Below the requirements, there is a note about other download options: "For other versions of Ubuntu Desktop including torrents, the network installer, a list of local mirrors, and past releases see our [alternative downloads](#)".

You can select 32/64-bit versions as per your choice.

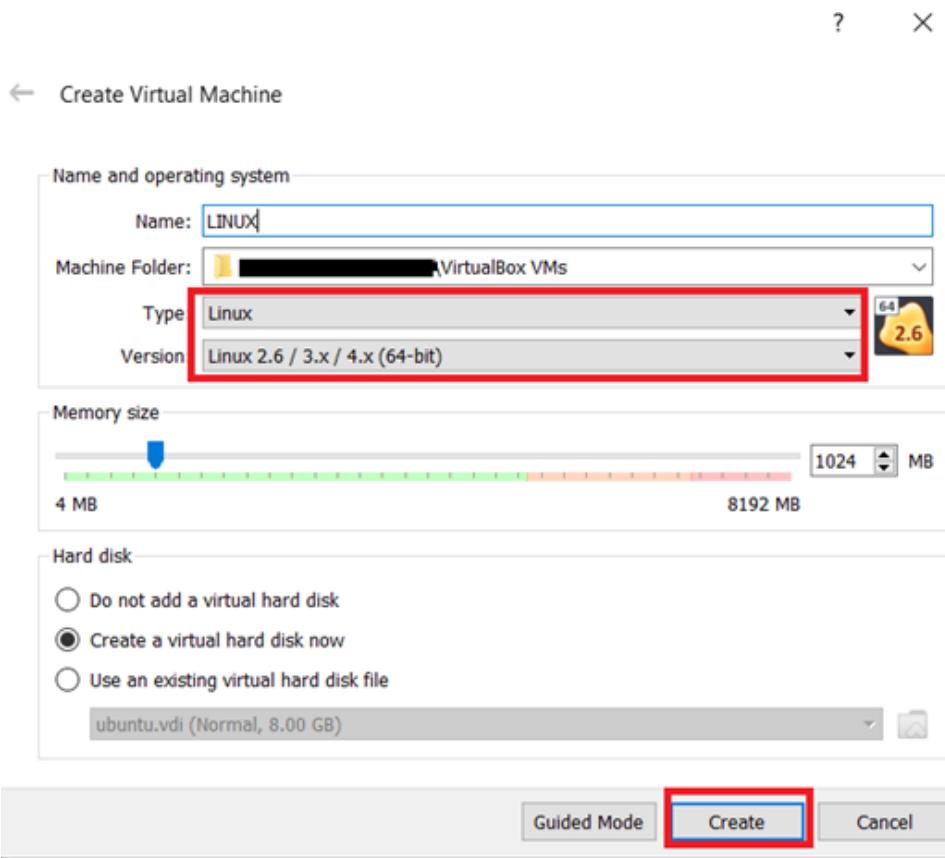
Create a Machine in Virtual Box

Step-1) Open Virtual box and click on new button



Step-2) In next window, give the name of your OS which you are installing in virtual box. And select OS like Linux and version as Ubuntu 32 bit. And click on next

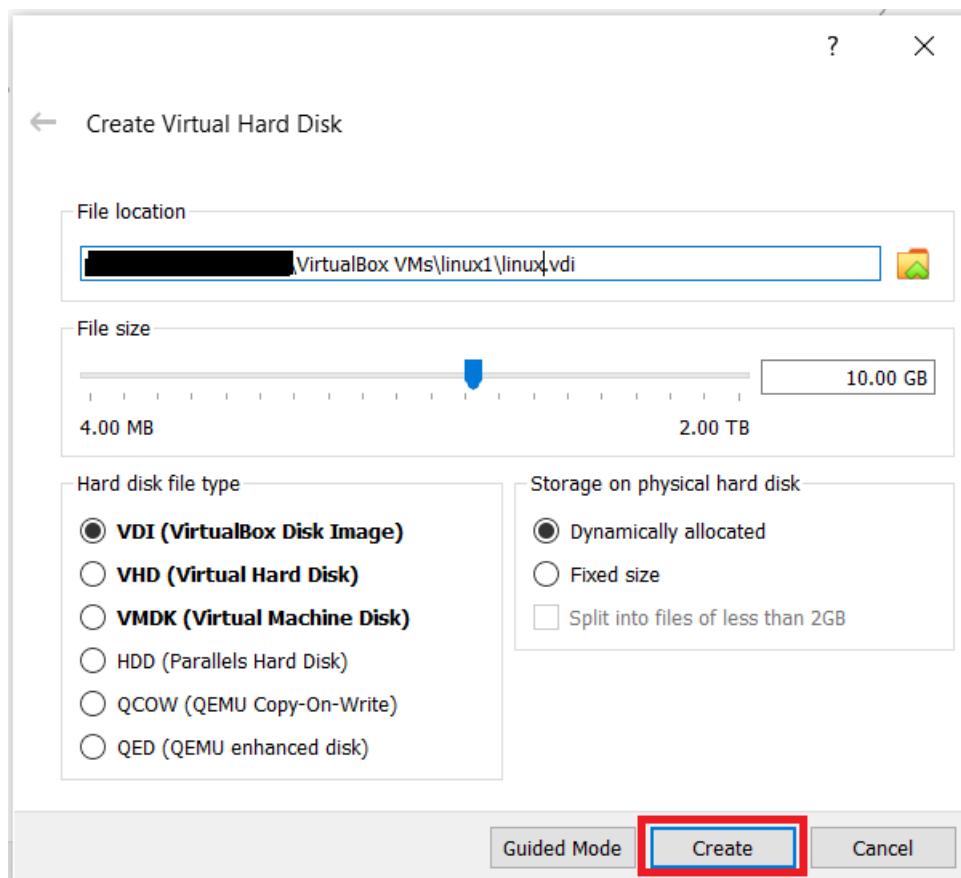
Step-3) Now Allocate Ram Size To your Virtual OS. I recommended keeping 1024mb (1 GB) ram to run Ubuntu better. And click on next.



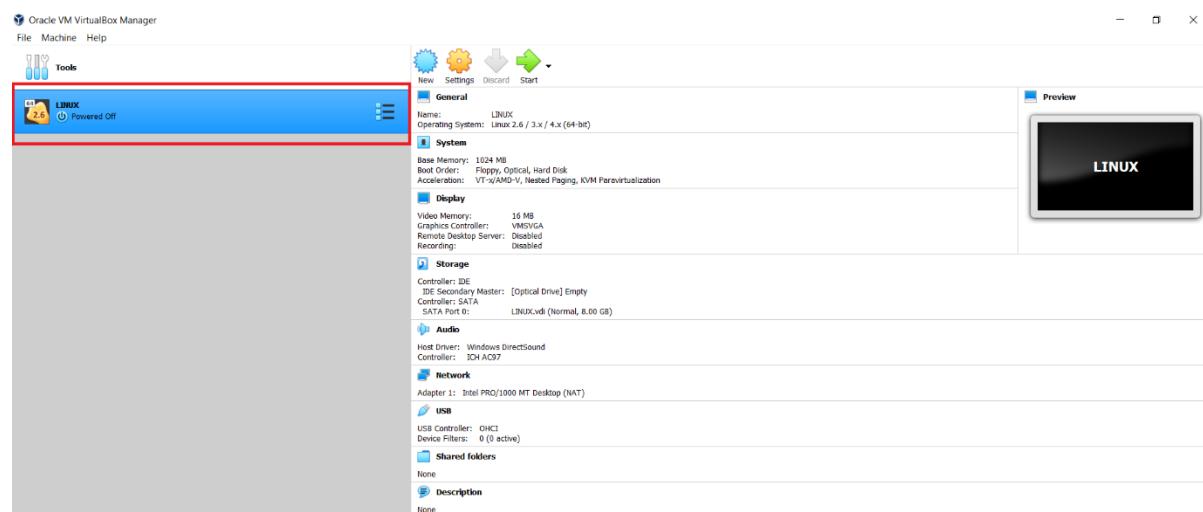
Step-4) Allocate memory to your virtual hard drive .10 GB recommended.

Step-5) select VDI (virtual hard disk) option

Step-6) Click on dynamic allocated. This means that the size of the disk will increase dynamically as per requirement. and click on create button.



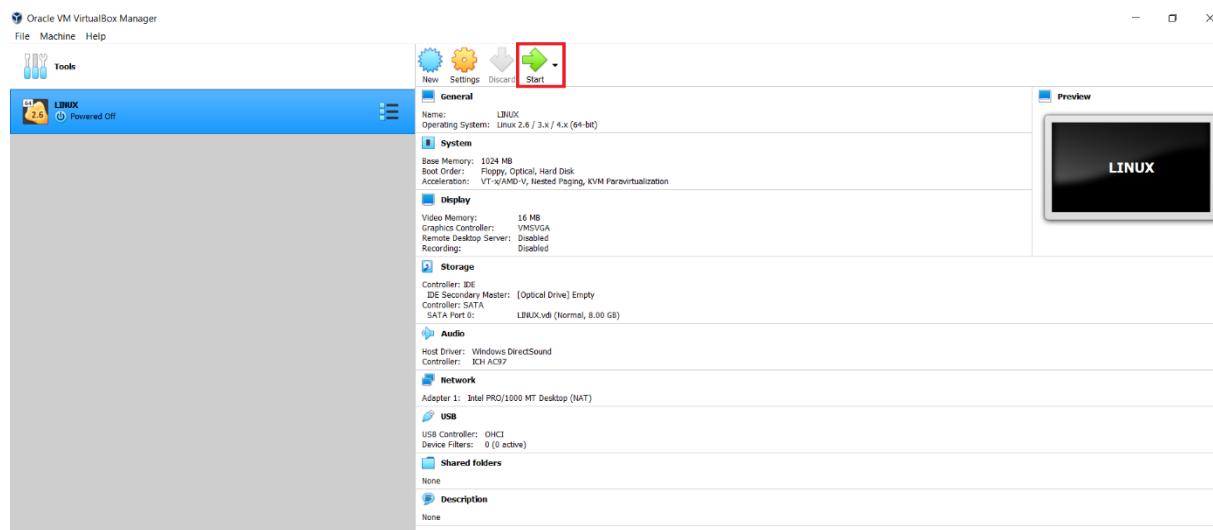
Step-8) Now you can see the machine name in left panel



So a Machine (PC) with 8GB Harddisk, 1GB RAM is ready.

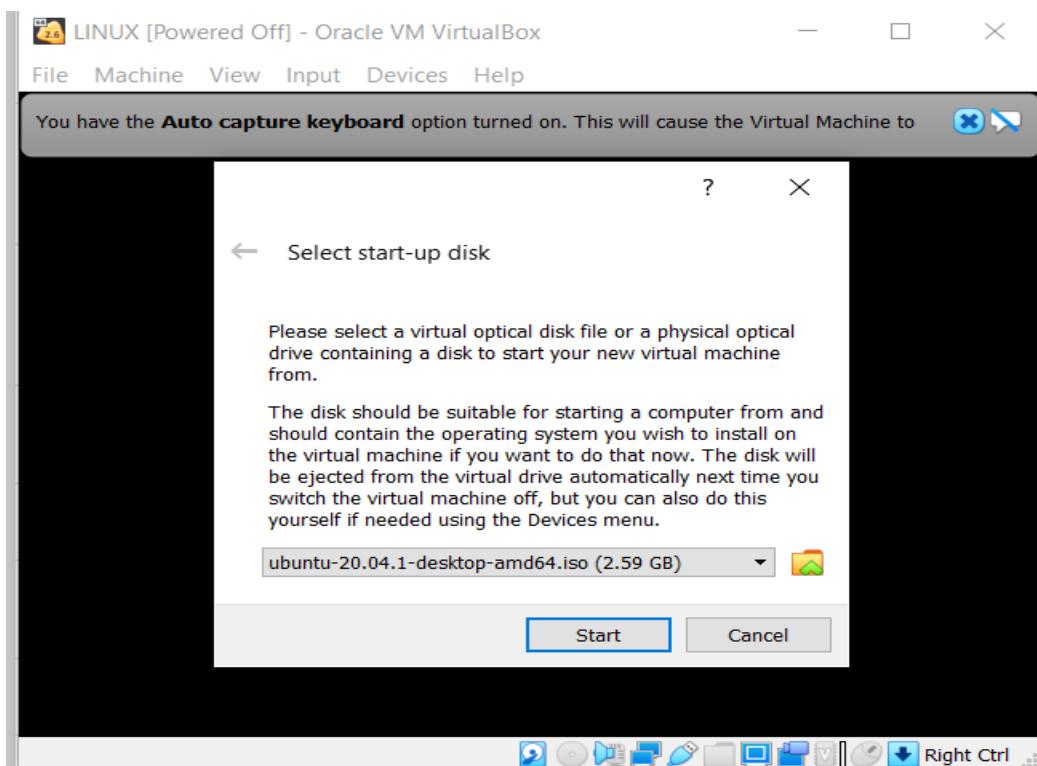
How to Install Ubuntu

Step 1) Select the Machine and Click on Start

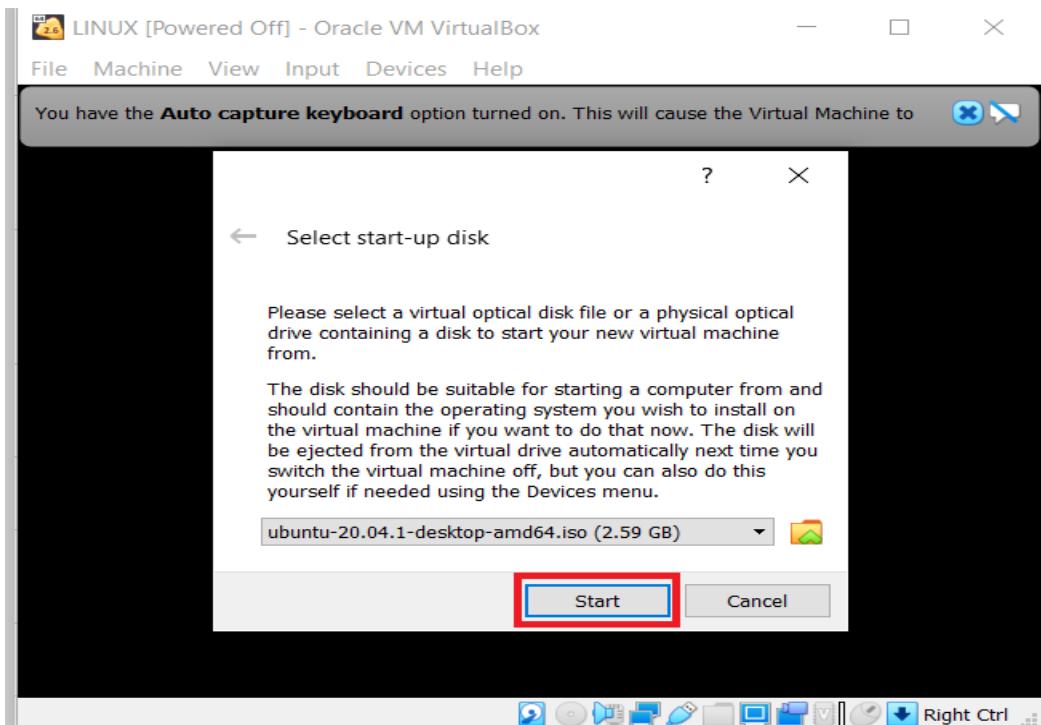


Step 2) Select the Folder Option

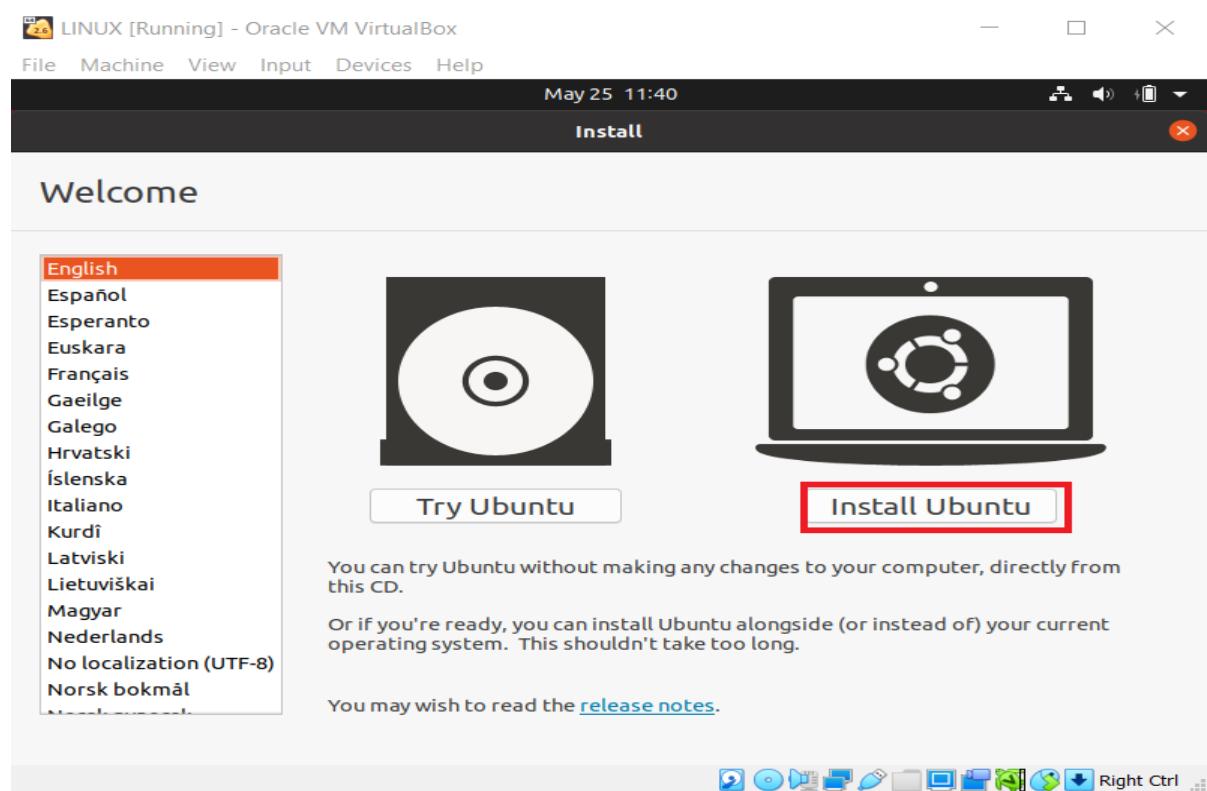
Step 3) Select the Ubuntu iso file



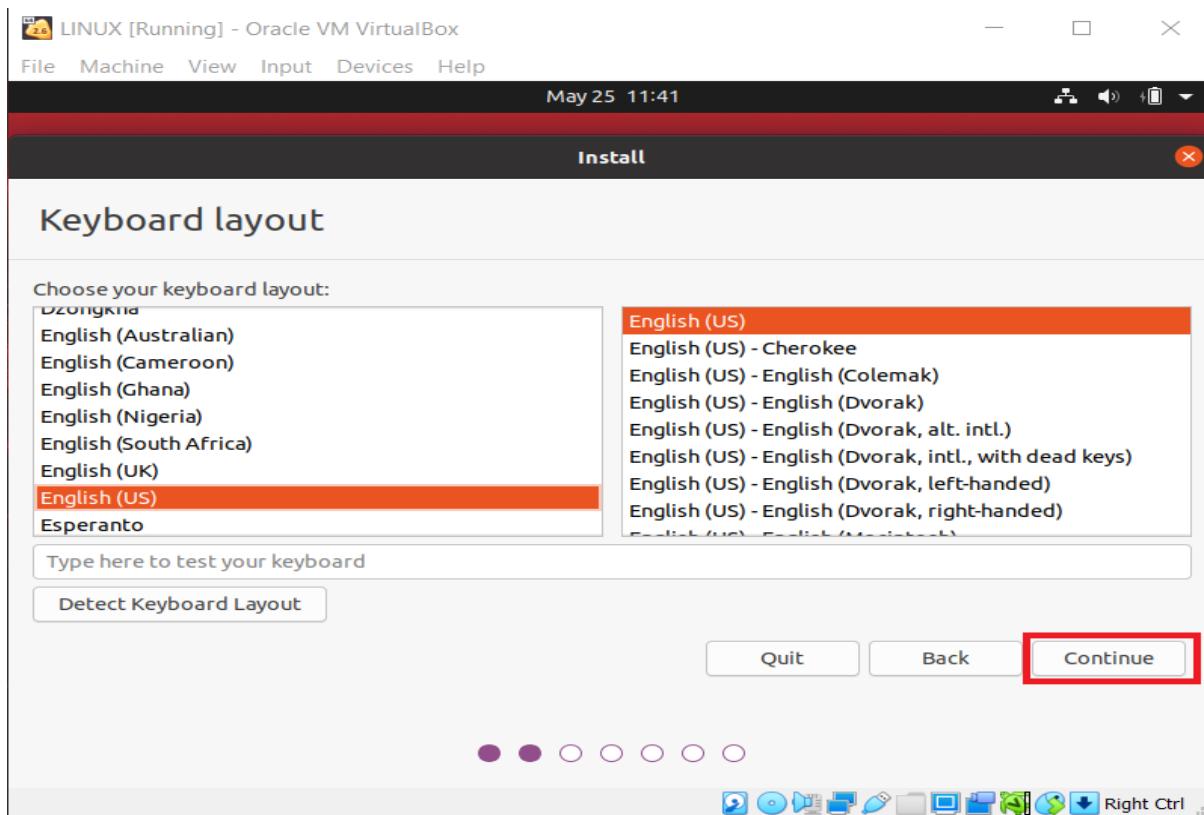
Step 4) Click Start



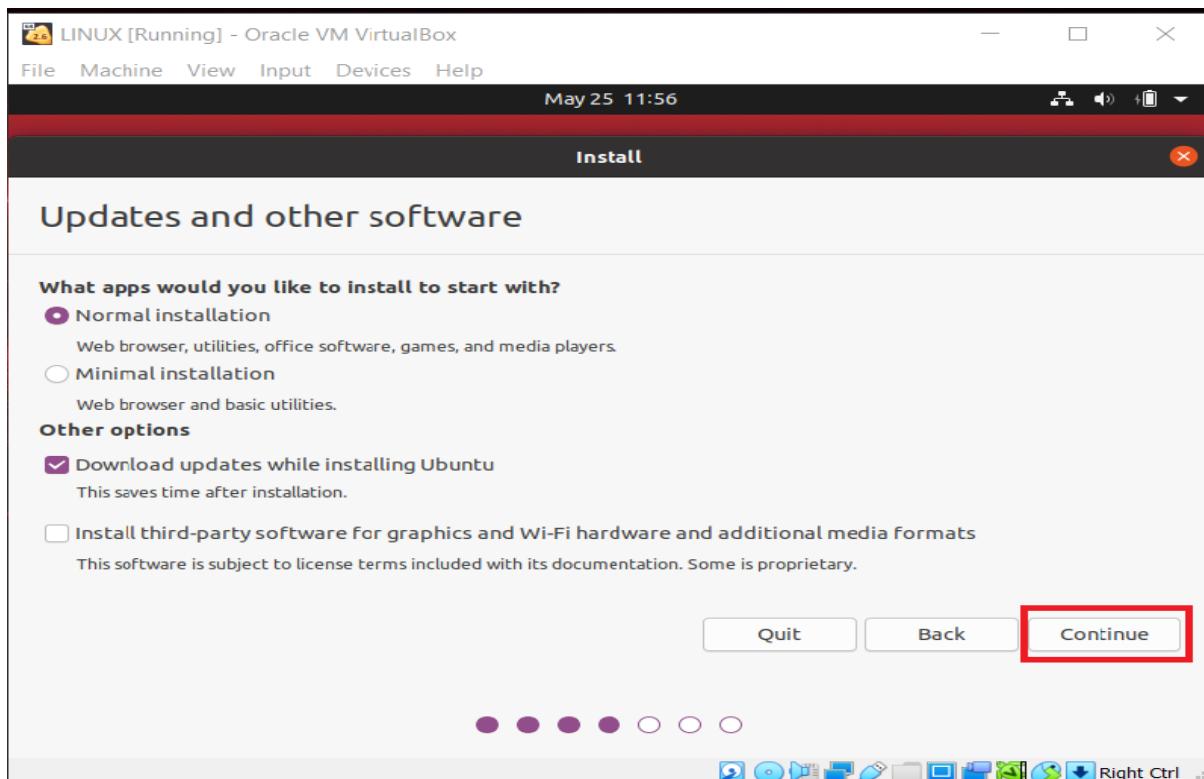
Step-5) You have an option to Run Ubuntu WITHOUT installing. In this tutorial will install Ubuntu



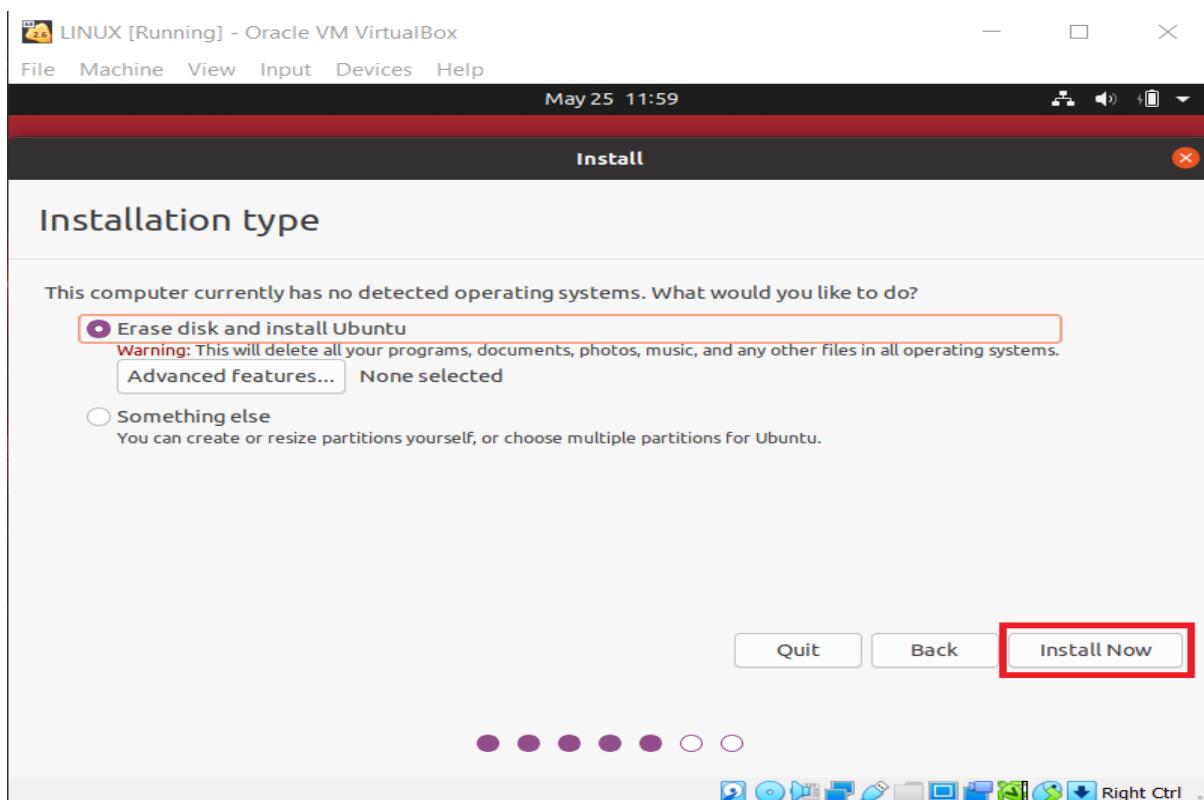
Step-6) Select your keyboard layout, by default English (US) is selected but if you want to change then, you can select in the list. And click on continue



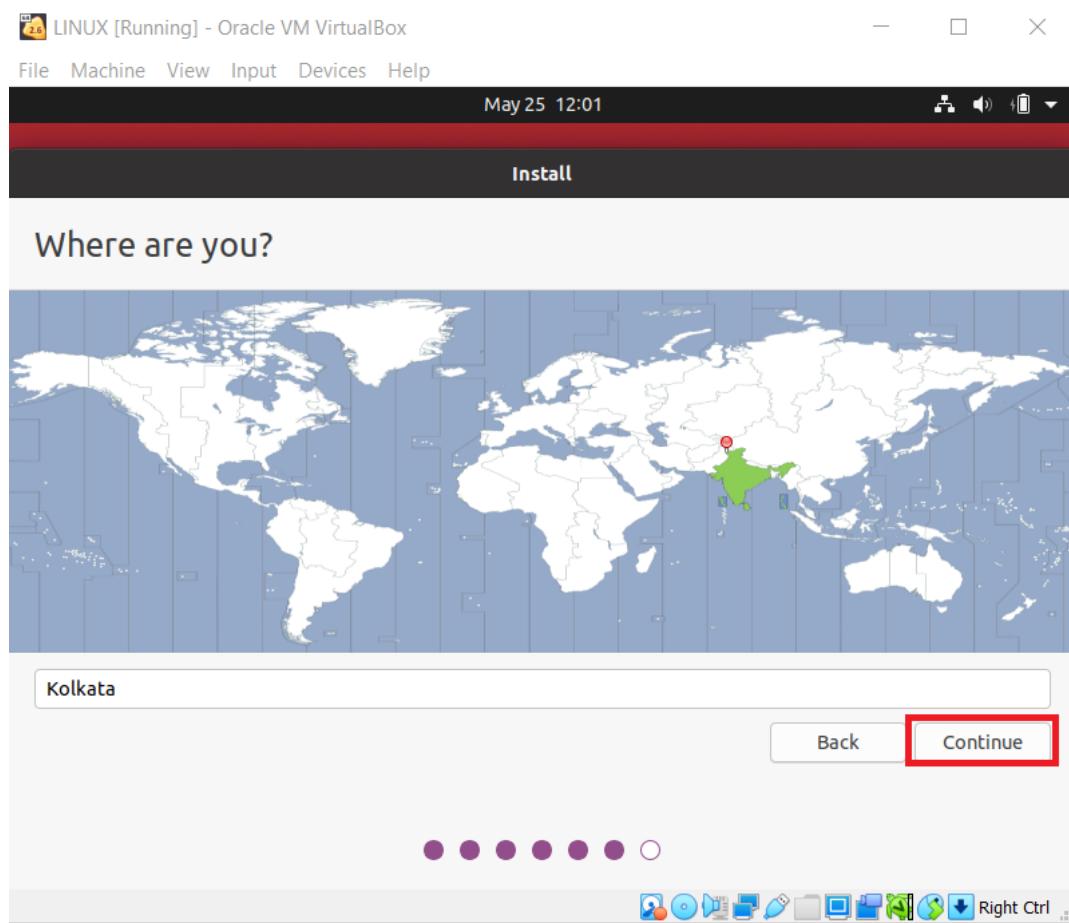
Step-7) Click continue.

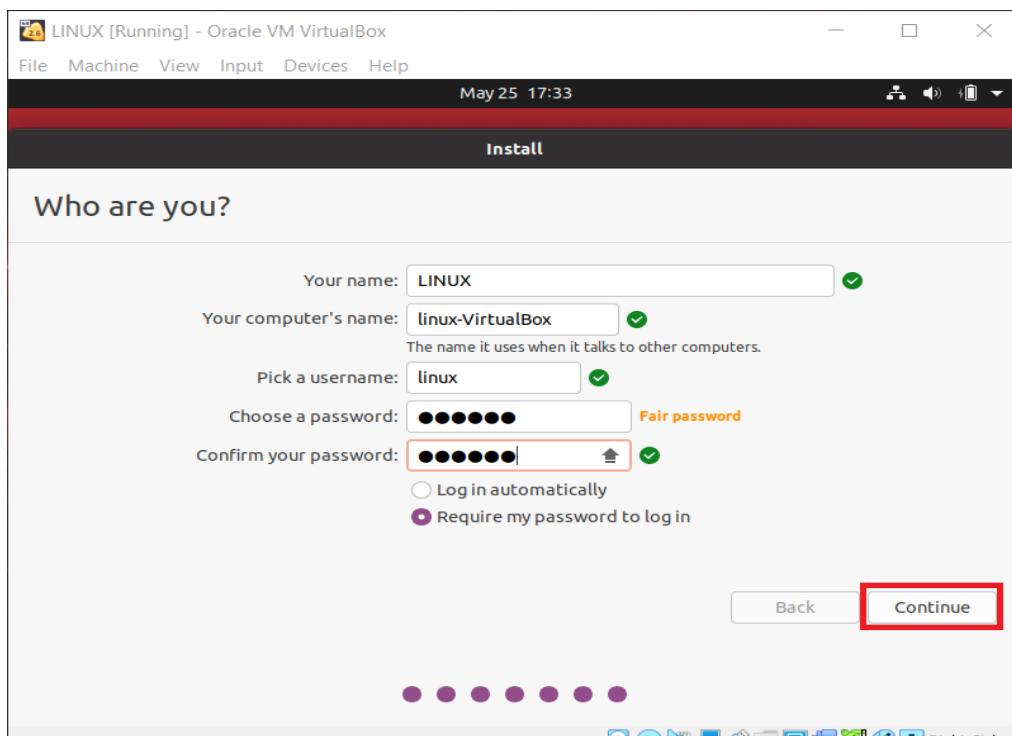


Step-8) Select option to erase the disk and install Ubuntu and click on install now. This option installs Ubuntu into our virtual hard drive which is we made earlier. It will not harm your PC or Windows installation



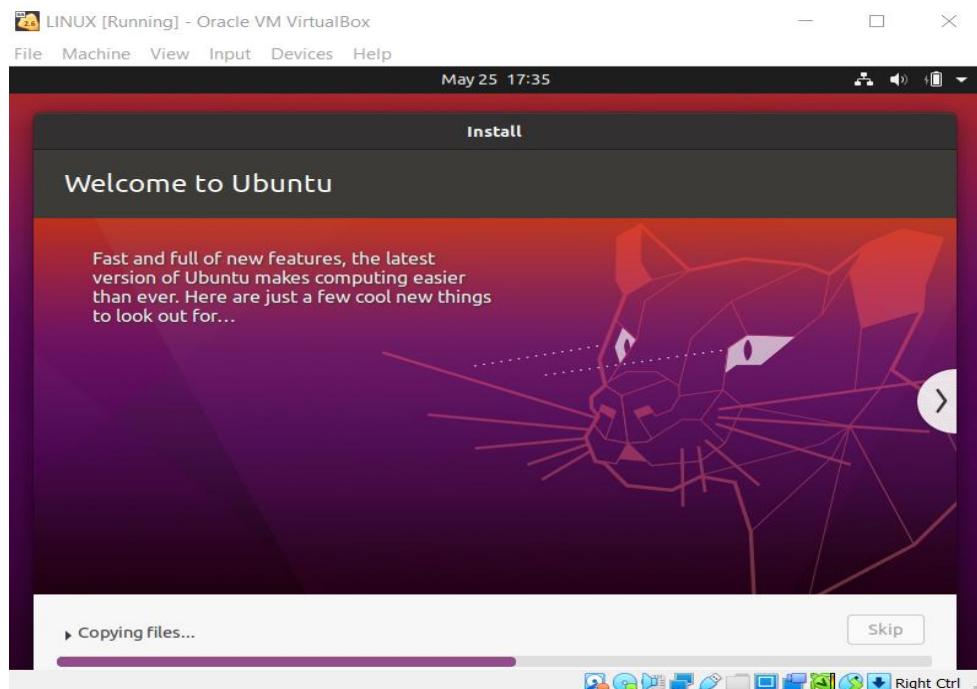
Step-9) Select your location for setting up time zone, and click on continue



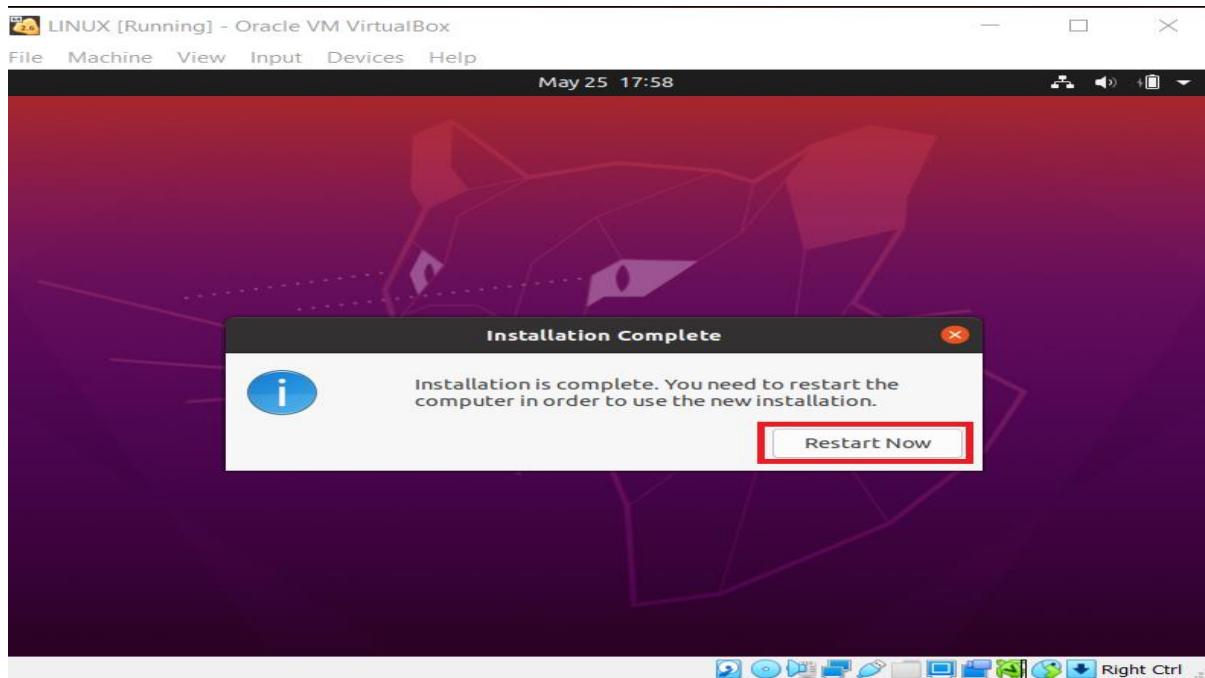


Step-10) Select your username and password for your Ubuntu admin account. This information has been needed for installing any software package into Ubuntu and also for login to your OS. Fill up your details and tick on login automatically to ignore login attempt and click on continue

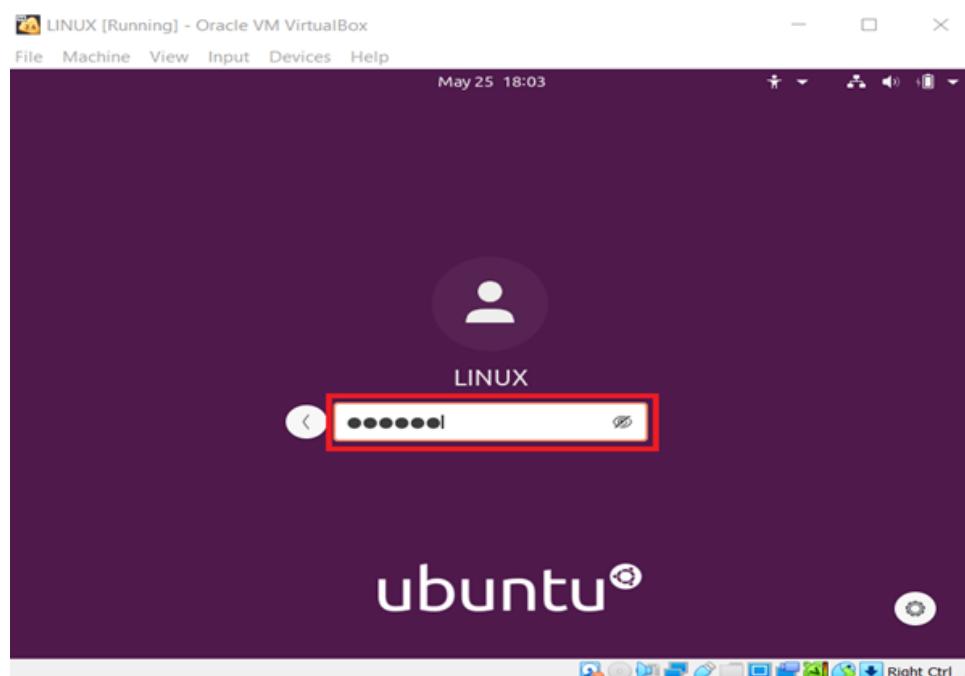
Step-11) Installation process starts. May take up to 30 minutes. Please wait until installation process completes.



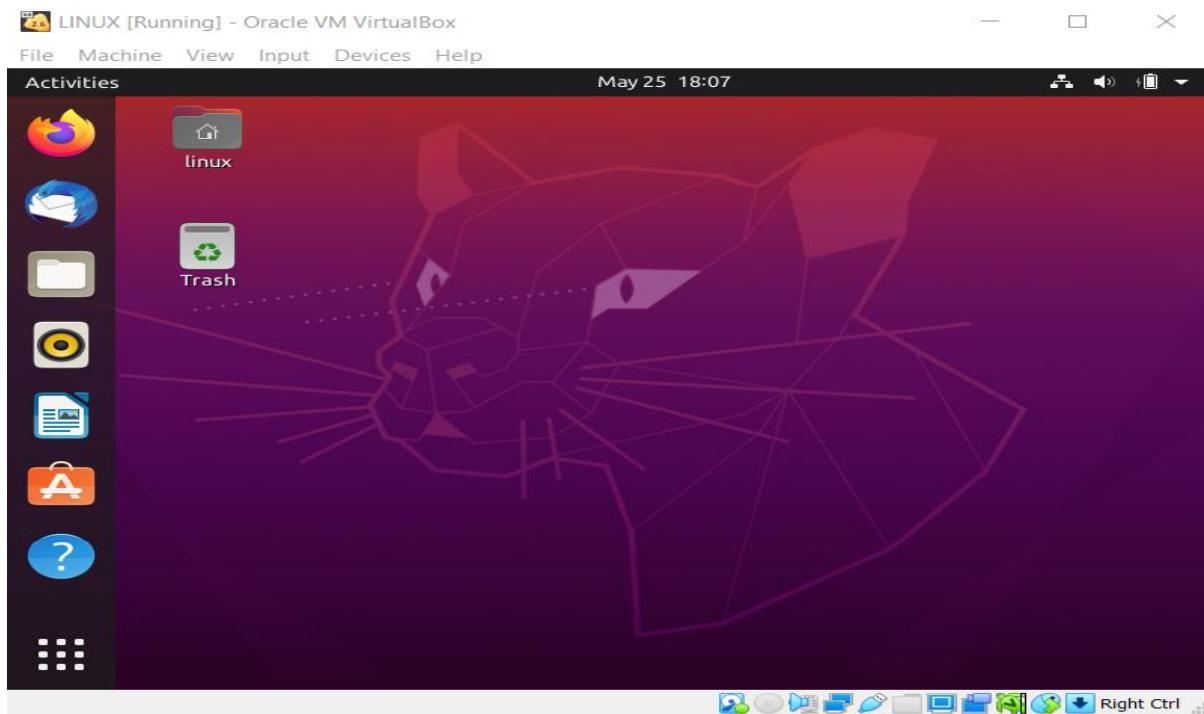
Step -12) Click on Restart Now



Step-13) After finishing the Restarting, you will see Ubuntu Desktop. Enter Password



Step-14) ubuntu Desktop page



After installing OS, let's get started with some basic commands used in Linux.

4.3 PRACTICAL: Linux Commands

Here is a list of basic Linux commands:

1. *pwd command*

Use the `pwd` command to find out the path of the current working directory (folder) you're in. The command will return an absolute (full) path, which is basically a path of all the directories that starts with a forward slash (/). An example of an absolute path is `/home/username`.

2. *cd command*

To navigate through the Linux files and directories, use the `cd` command. It requires either the full path or the name of the directory, depending on the current working directory that you're in.

Let's say you're in `/home/username/Documents` and you want to go to `Photos`, a subdirectory of `Documents`. To do so, simply type the following command: `cd Photos`.

Another scenario is if you want to switch to a completely new directory, for example,/home/username/Movies. In this case, you have to type cd followed by the directory's absolute path: cd /home/username/Movies.

There are some shortcuts to help you navigate quickly:

cd .. (with two dots) to move one directory up

cd to go straight to the home folder

cd- (with a hyphen) to move to your previous directory

On a side note, Linux's shell is case sensitive. So, you have to type the name's directory exactly as it is.

3. ls command

The ls command is used to view the contents of a directory. By default, this command will display the contents of your current working directory.

If you want to see the content of other directories, type ls and then the directory's path. For example, enter ls /home/username/Documents to view the content of Documents.

There are variations you can use with the ls command:

ls -R will list all the files in the sub-directories as well

ls -a will show the hidden files

ls -al will list the files and directories with detailed information like the permissions, size, owner, etc.

4. cat command

cat (short for concatenate) is one of the most frequently used commands in Linux. It is used to list the contents of a file on the standard output (stdout). To run this command, type cat followed by the file's name and its extension. For instance: cat file.txt.

Here are other ways to use the cat command:

cat > filename creates a new file

cat filename1 filename2>filename3 joins two files (1 and 2) and stores the output of them in a new file (3)

to convert a file to upper- or lower-case use, cat filename | tr a-z A-Z >output.txt

5. cp command

Use the cp command to copy files from the current directory to a different directory. For instance, the command cp scenery.jpg /home/username/Pictures would create a copy of scenery.jpg (from your current directory) into the Pictures directory.

6. mv command

The primary use of the mv command is to move files, although it can also be used to rename files.

The arguments in mv are similar to the cp command. You need to type mv, the file's name, and the destination's directory. For example: mv file.txt /home/username/Documents.

To rename files, the Linux command is mv oldname.ext newname.ext

7. mkdir command

Use mkdir command to make a new directory — if you type mkdir Music it will create a directory called Music.

There are extra mkdir commands as well:

To generate a new directory inside another directory, use this Linux basic command mkdir Music/Newfile

use the p (parents) option to create a directory in between two existing directories. For example, mkdir -p Music/2020/Newfile will create the new “2020” file.

8. rmdir command

If you need to delete a directory, use the rmdir command. However, rmdir only allows you to delete empty directories.

9. rm command

The rm command is used to delete directories and the contents within them. If you only want to delete the directory — as an alternative to rmdir — use rm -r.

Note: Be very careful with this command and double-check which directory you are in. This will delete everything and there is no undo.

10. touch command

The touch command allows you to create a blank new file through the Linux command line. As an example, enter touch /home/username/Documents/Web.html to create an HTML file entitled Web under the Documents directory.

11. locate command

You can use this command to locate a file, just like the search command in Windows. What's more, using the -i argument along with this command will make it case-insensitive, so you can search for a file even if you don't remember its exact name.

To search for a file that contains two or more words, use an asterisk (*). For example, locate -i school*note command will search for any file that contains the word "school" and "note", whether it is uppercase or lowercase.

12. find command

Similar to the locate command, using find also searches for files and directories. The difference is, you use the find command to locate files within a given directory.

As an example, find /home/ -name notes.txt command will search for a file called notes.txt within the home directory and its subdirectories.

Other variations when using the find are:

To find files in the current directory use, find . -name notes.txt

To look for directories use, / -type d -name notes. txt

13. grep command

Another basic Linux command that is undoubtedly helpful for everyday use is grep. It lets you search through all the text in a given file.

To illustrate, grep blue notepad.txt will search for the word blue in the notepad file. Lines that contain the searched word will be displayed fully.

14. sudo command

Short for "SuperUser Do", this command enables you to perform tasks that require administrative or root permissions. However, it is not advisable to use this command for daily use because it might be easy for an error to occur if you did something wrong.

15. df command

Use df command to get a report on the system's disk space usage, shown in percentage and KBs. If you want to see the report in megabytes, type df -m.

16. du command

If you want to check how much space a file or a directory takes, the du (Disk Usage) command is the answer. However, the disk usage summary will show disk block

numbers instead of the usual size format. If you want to see it in bytes, kilobytes, and megabytes, add the -h argument to the command line.

17. head command

The head command is used to view the first lines of any text file. By default, it will show the first ten lines, but you can change this number to your liking. For example, if you only want to show the first five lines, type head -n 5 filename.ext.

18. tail command

This one has a similar function to the head command, but instead of showing the first lines, the tail command will display the last ten lines of a text file. For example, tail -n filename.ext.

19. diff command

Short for difference, the diff command compares the contents of two files line by line. After analyzing the files, it will output the lines that do not match. Programmers often use this command when they need to make program alterations instead of rewriting the entire source code.

The simplest form of this command is diff file1.ext file2.ext

20. tar command

The tar command is the most used command to archive multiple files into a tarball — a common Linux file format that is similar to zip format, with compression being optional.

This command is quite complex with a long list of functions such as adding new files into an existing archive, listing the content of an archive, extracting the content from an archive, and many more.

21. chmod command

chmod is another Linux command, used to change the read, write, and execute permissions of files and directories. As this command is rather complicated

22. chown command

In Linux, all files are owned by a specific user. The chown command enables you to change or transfer the ownership of a file to the specified username. For instance, chown linuxuser2 file.ext will make linuxuser2 as the owner of the file.ext.

23. jobs command

jobs command will display all current jobs along with their statuses. A job is basically a process that is started by the shell.

24. kill command

If you have an unresponsive program, you can terminate it manually by using the kill command. It will send a certain signal to the misbehaving app and instructs the app to terminate itself.

There is a total of **sixty-four signals** that you can use, but people usually only use two signals:

SIGTERM (15) — requests a program to stop running and gives it some time to save all of its progress. If you don't specify the signal when entering the kill command, this signal will be used.

SIGKILL (9) — forces programs to stop immediately. Unsaved progress will be lost.

Besides knowing the signals, you also need to know the process identification number (PID) of the program you want to kill. If you don't know the PID, simply run the command ps ux.

After knowing what signal you want to use and the PID of the program, enter the following syntax:

`kill [signal option] PID.`

25. ping command

Use the ping command to check your connectivity status to a server. For example, by simply entering `ping google.com`, the command will check whether you're able to connect to Google and also measure the response time.

26. wget command

The Linux command line is super useful — you can even download files from the internet with the help of the wget command. To do so, simply type `wget` followed by the download link.

27. uname command

The uname command, short for Unix Name, will print detailed information about your Linux system like the machine name, operating system, kernel, and so on.

28. top command

As a terminal equivalent to Task Manager in Windows, the top command will display a list of running processes and how much CPU each process uses. It's very useful to monitor system resource usage, especially knowing which process needs to be terminated because it consumes too many resources.

29. history command

When you've been using Linux for a certain period of time, you'll quickly notice that you can run hundreds of commands every day. As such, running history command is particularly useful if you want to review the commands you've entered before.

30. man command

Confused about the function of certain Linux commands? Don't worry, you can easily learn how to use them right from Linux's shell by using the man command. For instance, entering man tail will show the manual instruction of the tail command.

31. echo command

This command is used to move some data into a file. For example, if you want to add the text, "Hello, my name is John" into a file called name.txt, you would type echo Hello, my name is John >> name.txt

32. zip, unzip command

Use the zip command to compress your files into a zip archive, and use the unzip command to extract the zipped files from a zip archive.

33. hostname command

If you want to know the name of your host/network simply type hostname. Adding a -I to the end will display the IP address of your network.

34. useradd, userdel command

Since Linux is a multi-user system, this means more than one person can interact with the same system at the same time. useradd is used to create a new user, while passwd is adding a password to that user's account. To add a new person named John type, useradd John and then to add his password type, passwd 123456789.

To remove a user is very similar to adding a new user. To delete the users account type, userdel UserName [3]

Command	Description
ls	Lists all files and directories in the present working directory
ls - R	Lists files in sub-directories as well
ls - a	Lists hidden files as well
ls - al	Lists files and directories with detailed information like permissions, size, owner, etc.
cat > filename	Creates a new file

cat filename	Displays the file content
cat file1 file2 > file3	Joins two files (file1, file2) and stores the output in a new file (file3)
mv file "new file path"	Moves the files to the new location
mv filename new_file_name	Renames the file to a new filename
sudo	Allows regular users to run programs with the security privileges of the superuser or root
rm filename	Deletes a file
man	Gives help information on a command
history	Gives a list of all past basic Linux commands list typed in the current terminal session
clear	Clears the terminal
mkdir directoryname	Creates a new directory in the present working directory or a at the specified path
Rmdir	Deletes a directory
Mv	Renames a directory
pr -x	Divides the file into x columns
pr -h	Assigns a header to the file
pr -n	Denotes the file with Line Numbers
lp -nc lpr c	Prints "c" copies of the File
lp -d lpr -P	Specifies name of the printer
apt-get	Command used to install and update packages
mail -s 'subject' -c 'cc-address' -b 'bcc-address' 'to-address'	Command to send email
mail -s "Subject" to-address < Filename	Command to send email with attachment

Unit 5: Database Management

Learning Outcomes:

- Understand the Fundamental Concept of MongoDB
- Manage Large Database using MongoDB
- Perform CRUD operation

5.1 MongoDB - Overview

MongoDB is a free and open-source cross-platform document-oriented database. Classified as a NoSQL database, MongoDB avoids the traditional table-based relational database structure in favor of JSON-like documents with dynamic schemas, making the integration of data in certain types of applications easier and faster.

Database

Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

Collection

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

Document

A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

The following table shows the relationship of RDBMS terminology with MongoDB.

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key <code>_id</code> provided by MongoDB itself)

Create Database

The use Command

MongoDB use DATABASE_NAME is used to create database. The command will create a new database if it doesn't exist, otherwise it will return the existing database.

Syntax-

use DATABASE_NAME

Drop Database

The dropDatabase() Method

MongoDB db.dropDatabase() command is used to drop a existing database.

Syntax-

db.dropDatabase()

This will delete the selected database. If you have not selected any database, then it will delete default 'test' database.

Create Collection

The createCollection() Method

MongoDB db.createCollection(name, options) is used to create collection.

Syntax-

db.createCollection(name, options)

In the command, name is name of collection to be created. Options is a document and is used to specify configuration of collection.

Parameters	type	Description
Name	String	Name of the collection to be created
Options	Document	(Optional) Specify options about memory size and indexing

Drop Collection

The drop() Method

MongoDB's db.collection.drop() is used to drop a collection from the database.

Syntax-

db.COLLECTION_NAME.drop()

Insert Document

The insert() Method

To insert data into MongoDB collection, you need to use MongoDB's insert() or save() method.

Syntax-

>db.COLLECTION_NAME.insert(document)

Query Document

The find() Method

To query data from MongoDB collection, you need to use MongoDB's find() method.

Syntax-

```
db.COLLECTION_NAME.find()
```

find() method will display all the documents in a non-structured way.

Update Document

MongoDB's update() and save() methods are used to update document into a collection. The update() method updates the values in the existing document while the save() method replaces the existing document with the document passed in save() method.

MongoDB Update() Method

The update() method updates the values in the existing document.

Syntax

```
>db.COLLECTION_NAME.update(SELECTION_CRITERIA, UPDATED_DATA)
```

Delete Document

The remove() Method

MongoDB's remove() method is used to remove a document from the collection. remove() method accepts two parameters. One is deletion criteria and second is justOne flag.

- deletion criteria – (Optional) deletion criteria according to documents will be removed.
- justOne – (Optional) if set to true or 1, then remove only one document.

Syntax

```
>db.COLLECTION_NAME.remove(DELLETION_CRITTERIA)
```

Projection

In MongoDB, projection means selecting only the necessary data rather than selecting whole of the data of a document. If a document has 5 fields and you need to show only 3, then select only 3 fields from them.

The find() Method-

MongoDB's find() method, explained in MongoDB Query Document accepts second optional parameter that is list of fields that you want to retrieve. In MongoDB, when you execute find() method, then it displays all fields of a document. To limit this, you need to set a list of fields with value 1 or 0. 1 is used to show the field while 0 is used to hide the fields.

Syntax

```
>db.COLLECTION_NAME.find({}, {KEY:1})
```

Aggregation

Aggregations operations process data records and return computed results. Aggregation operations group values from multiple documents together, and can perform a variety of operations on the grouped data to return a single result. In SQL count(*) and with group by is an equivalent of MongoDB aggregation.

The aggregate() Method

For the aggregation in MongoDB, you should use aggregate() method.

Syntax

```
>db.COLLECTION_NAME.aggregate(AGGREGATE_OPERATION)
```

5.2 Datatypes supported by MongoDB

MongoDB supports many datatypes. Some of them are –

- **String** – This is the most commonly used datatype to store the data. String in MongoDB must be UTF-8 valid.
- **Integer** – This type is used to store a numerical value. Integer can be 32 bit or 64 bit depending upon your server.
- **Boolean** – This type is used to store a boolean (true/ false) value.
- **Double** – This type is used to store floating point values.
- **Min/ Max keys** – This type is used to compare a value against the lowest and highest BSON elements.
- **Arrays** – This type is used to store arrays or list or multiple values into one key.
- **Timestamp** – ctimestamp. This can be handy for recording when a document has been modified or added.
- **Object** – This datatype is used for embedded documents.
- **Null** – This type is used to store a Null value.
- **Symbol** – This datatype is used identically to a string; however, it's generally reserved for languages that use a specific symbol type.
- **Date** – This datatype is used to store the current date or time in UNIX time format. You can specify your own date time by creating object of Date and passing day, month, year into it.
- **Object ID** – This datatype is used to store the document's ID.
- **Binary data** – This datatype is used to store binary data.
- **Code** – This datatype is used to store JavaScript code into the document.
- **Regular expression** – This datatype is used to store regular expression.

5.3 PRACTICAL: Import and Export Data

You can use MongoDB Compass to import and export data to and from collections. Compass supports import and export for both JSON and CSV files. To import or export data to or from a collection, navigate to the detailed collection view by either selecting the collection from the Databases tab or clicking the collection in the left-side navigation.

Import Data into a Collection

MongoDB Compass can import data into a collection from either a **JSON** or **CSV** file.

Limitations

- Importing data into a collection is not permitted in **MongoDB Compass Readonly Edition**.
- Importing data is not available if you are connected to a [Data Lake](#).

Format Your Data

Before you can import your data into MongoDB Compass you must first ensure that it is formatted correctly.

When importing data from a JSON file, you can format your data as:

- Newline-delimited documents, or
- Comma-separated documents in an array

The following newline-delimited .json file is formatted correctly:

```
{ "type": "home", "number": "212-555-1234" }
{ "type": "cell", "number": "646-555-4567" }
{ "type": "office", "number": "202-555-0182" }
```

The following comma-separated .json array file is also formatted correctly:

```
[{ "type": "home", "number": "212-555-1234" }, { "type": "cell", "number": "646-555-4567" }, { "type": "office", "number": "202-555-0182" }]
```

Procedure

To import your formatted data into a collection:

1. Connect to the deployment containing the collection you wish to import data into.
2. Navigate to your desired collection.
3. Click the Add Data dropdown and select Import File.

test.flightStats

DOCUMENTS 3 TOTAL SIZE 1.4KB AVG. SIZE 492B | INDEXES 1 TOTAL SIZE 4.0KB AVG. SIZE 4.0KB

Documents Aggregations Schema Explain Plan Indexes Validation

The screenshot shows the MongoDB Compass interface for the 'test.flightStats' database. At the top, there are tabs for 'Documents', 'Aggregations', 'Schema', 'Explain Plan', 'Indexes', and 'Validation'. Below the tabs, there's a toolbar with a 'FILTER' button, 'VIEW' dropdown, and other search/refresh buttons. A red box highlights the 'ADD DATA' button, which has a dropdown menu showing 'Import File' and 'Insert Document'. The main area displays a document with fields like '_id', 'sCode', and 'cancelled'. Below the document list, it says 'Displaying documents 1 - 3 of 3'.

Reference: <https://www.mongodb.com/docs/compass/current/import-export/>

4. Select the location of the source data file under select File
5. Choose the appropriate file type
6. Under **Select Input File Type**, select either JSON or CSV

If you are importing a CSV file, you may specify fields to import and the types of those fields under **Specify Fields and Types**. The default data type for all fields is string.

The screenshot shows the 'Import To Collection test.people' dialog box. It includes sections for 'Select File' (set to '/Users/zach.carr/Desktop/people.csv'), 'Select Input File Type' (with 'CSV' selected), 'Options' (including 'Select delimiter' set to 'COMMA' and checkboxes for 'Ignore empty strings' and 'Stop on errors'), and 'Specify Fields and Types' (a table with 10 rows of data, each row having a checked checkbox and a dropdown menu for the 'String' field type). At the bottom are 'CANCEL' and 'IMPORT' buttons.

	<input checked="" type="checkbox"/> String			
1	5ca4bbcea2dd94ee58162a68	fmiller	Elizabeth Ray	9286 Bethany Glens
2	5ca4bbcea2dd94ee58162a69	valenciacjennifer	Lindsay Cowan	Unit 1047 Box 4089
3	5ca4bbcea2dd94ee58162a6a	hillrachel	Katherine David	55711 Janet Plaza A
4	5ca4bbcea2dd94ee58162a6b	serranobrian	Leslie Martinez	Unit 2676 Box 9352
5	5ca4bbcea2dd94ee58162a6c	charleshudson	Brad Cardenas	2765 Powers Meadow
6	5ca4bbcea2dd94ee58162a6d	gregoryharrison	Natalie Ford	17677 Mark Crest Wa
7	5ca4bbcea2dd94ee58162a6e	hmyers	Dana Clarke	50047 Smith Point S
8	5ca4bbcea2dd94ee58162a6f	andrewhamilton	Gary Nichols	633 Miller Turnpike
9	5ca4bbcea2dd94ee58162a70	matthewray	John Parks	38456 Rachael Cause
10	5ca4bbcea2dd94ee58162a71	glopez	Jennifer Lawrence	4140 Pamela Hollow

Reference: <https://www.mongodb.com/docs/compass/current/import-export/>

To exclude a field from a CSV file you are importing, uncheck the checkbox next to that field name. To select a type for a field, use the dropdown menu below that field name.

- Configure import options.

Under Options, configure the import options for your use case.

If you are importing a CSV file, you may select how your data is delimited

For both JSON and CSV file imports, you can toggle Ignore empty strings and Stop on errors:

- if checked, Ignore empty strings drops fields with empty string values from your imported documents. The document is still imported with all other fields.
- If checked, Stop on errors prevents any data from being imported in the event of an error. If unchecked, data is inserted until an error is encountered and successful inserts are not rolled back. The import operation will not continue after encountering an error in either case.
- Click Import.

Export Data from a Collection

MongoDB Compass can export data from a collection as either a **JSON** or **CSV** file. If you specify a [filter](#), Compass only exports documents which match the specified query.

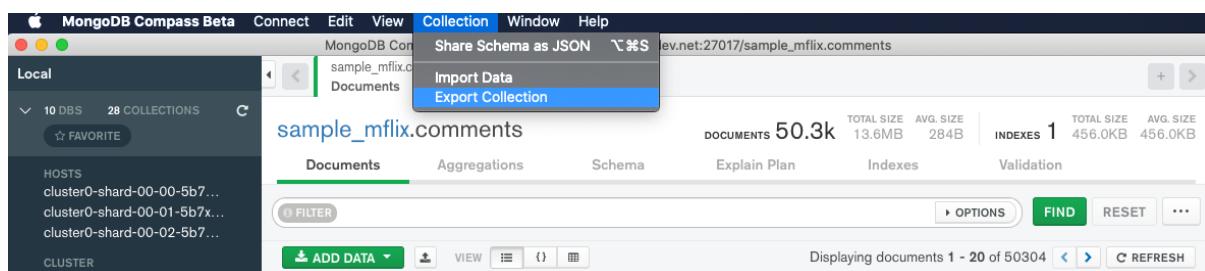
Behaviour

While it is possible to exclude documents by using a query filter, it is not possible to re-shape exported documents with a project document. Even when you specify a project option in the query, Compass still exports the entire document.

Procedure

To export an entire collection to a file:

1. Connect to the deployment containing the collection you wish to export data from.
2. Navigate to your desired collection.
3. Click Collection in the top-level menu and select Export Collection.

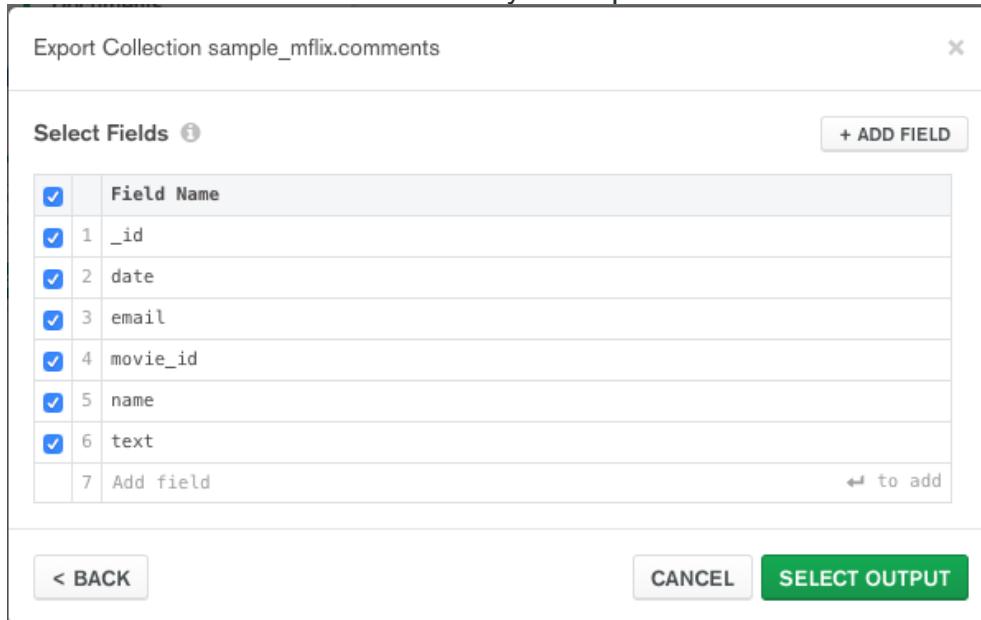


Reference: <https://www.mongodb.com/docs/compass/current/import-export/>

The export dialog initially displays the query entered in the query bar prior to export, if applicable. If no query was specified, this section displays a find operation with no parameters, which returns all documents in the collection.

To ignore the query filter and export your entire collection, select Export Full Collection and click Select Fields.

4. Select document fields to include in your exported file.



Export Collection sample_mflix.comments

Select Fields i

	Field Name
<input checked="" type="checkbox"/>	_id
<input checked="" type="checkbox"/>	date
<input checked="" type="checkbox"/>	email
<input checked="" type="checkbox"/>	movie_id
<input checked="" type="checkbox"/>	name
<input checked="" type="checkbox"/>	text
7	Add field <small>↔ to add</small>

< BACK CANCEL SELECT OUTPUT

Reference: <https://www.mongodb.com/docs/compass/current/import-export/>

5. Choose a file type and export location.



Export Collection sample_mflix.comments

Select Export File Type

JSON

CSV

Output BROWSE

< BACK CANCEL EXPORT

Reference: <https://www.mongodb.com/docs/compass/current/import-export/>

6. Click Export.

5.4 PRACTICAL: CRUD Operations

CRUD operations create, read, update, and delete documents.

Create Operations-

Create or insert operations add new documents to a collection. If the collection does not currently exist, insert operations will create the collection.

MongoDB provides the following methods to insert documents into a collection:

- db.collection.insertOne() New in version 3.2
- db.collection.insertMany() New in version 3.2

Read Operations-

Read operations retrieve documents from a collection; i.e. query a collection for documents. MongoDB provides the following methods to read documents from a collection:

- db.collection.find()

Update Operations-

Update operations modify existing documents in a collection. MongoDB provides the following methods to update documents of a collection:

- db.collection.updateOne() New in version 3.2
- db.collection.updateMany() New in version 3.2
- db.collection.replaceOne() New in version 3.2

In MongoDB, update operations target a single collection. All write operations in MongoDB are atomic on the level of a single document.

Delete Operations-

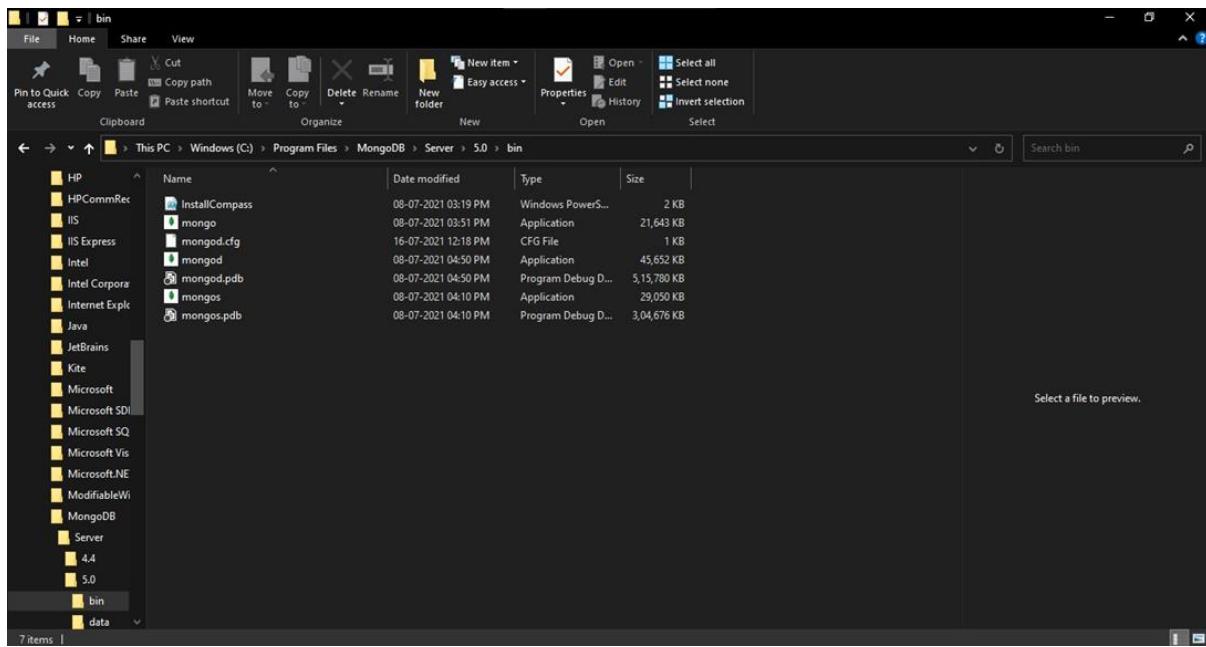
Delete operations remove documents from a collection. MongoDB provides the following methods to delete documents of a collection:

- db.collection.deleteOne() New in version 3.2
- db.collection.deleteMany() New in version 3.2

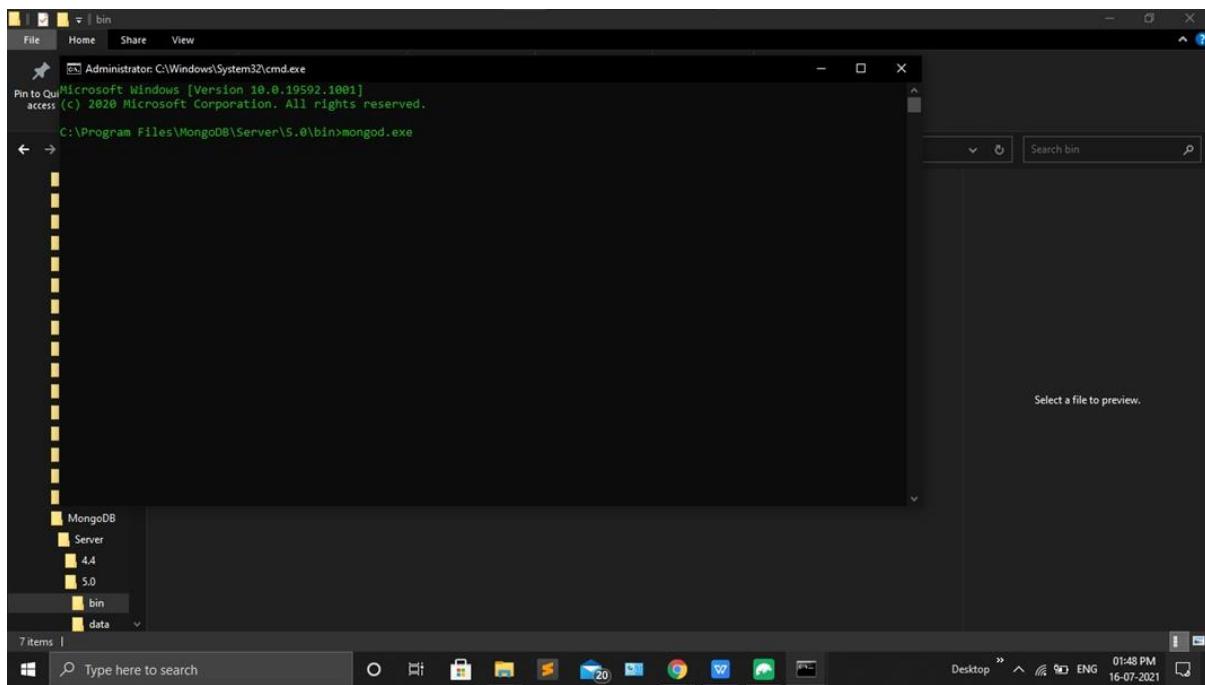
In MongoDB, delete operations target a single collection. All write operations in MongoDB are atomic on the level of a single document.

Procedure

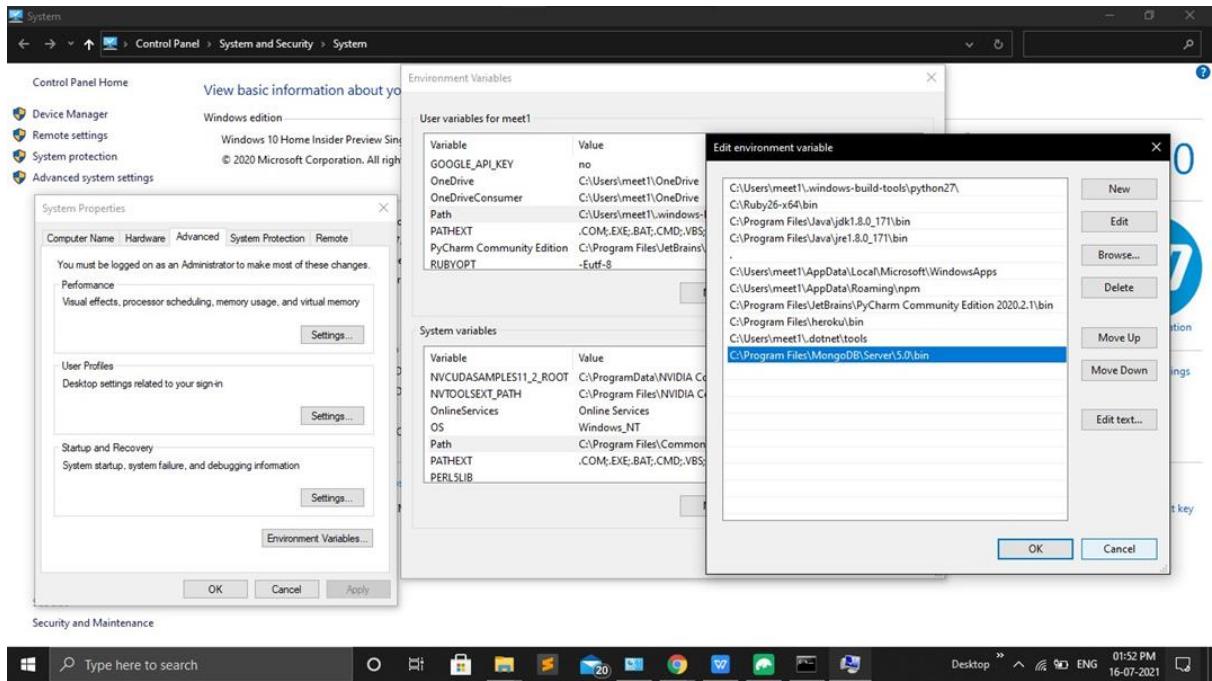
1. Go to Program files> MongoDB > Servers > 5.0 > bin



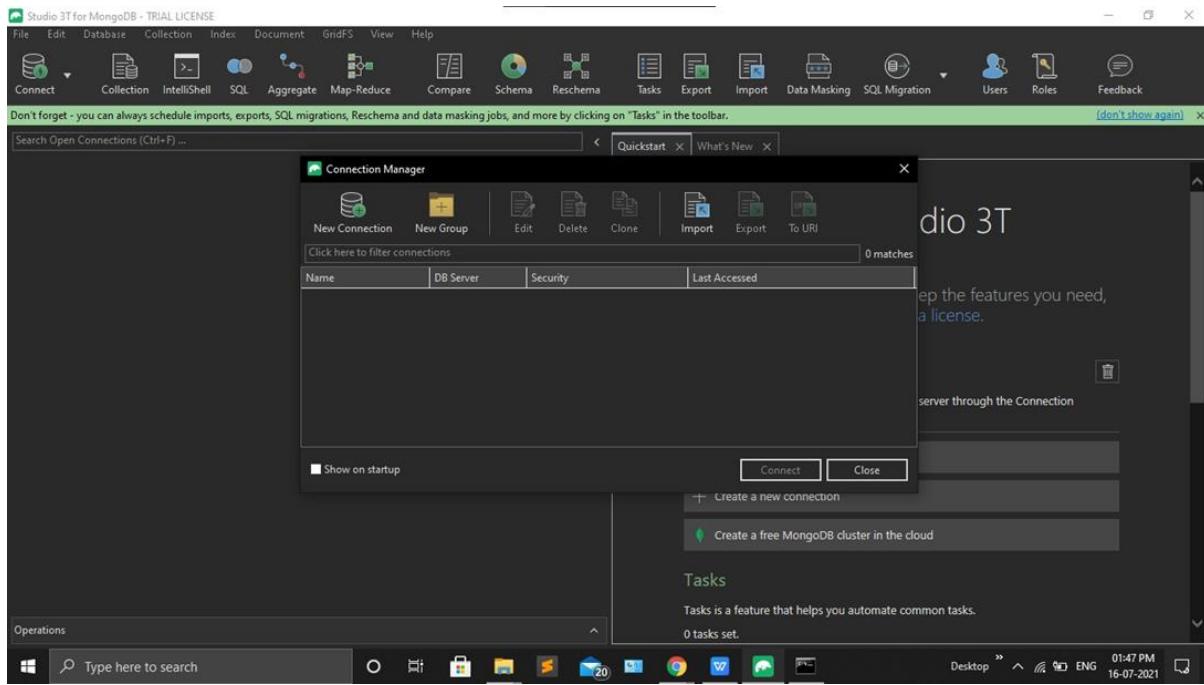
- Open the cmd at the above location, and type “mongod.exe” so the daemon will wait for the connection.



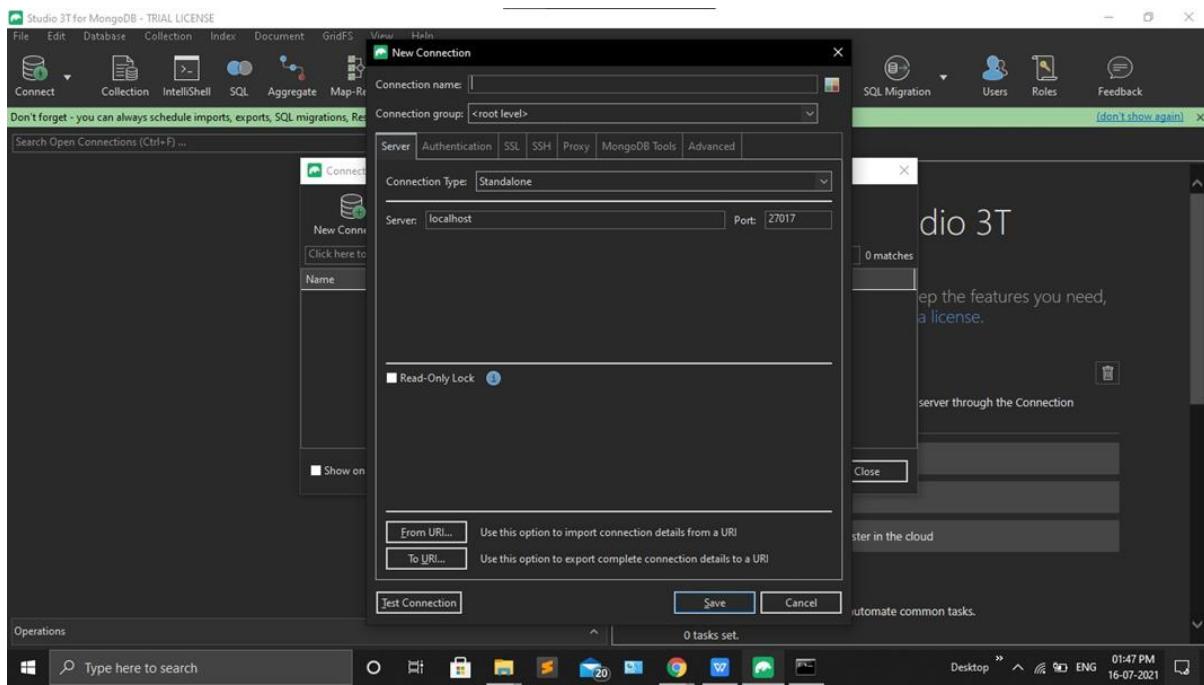
- Now go to my computer, then go to the properties of my computer. In properties, Advanced System Settings > Environment Variables > in path variable for user or system Add new path “C:\Program Files\MongoDB\Servers\5.0\bin” according to your location.



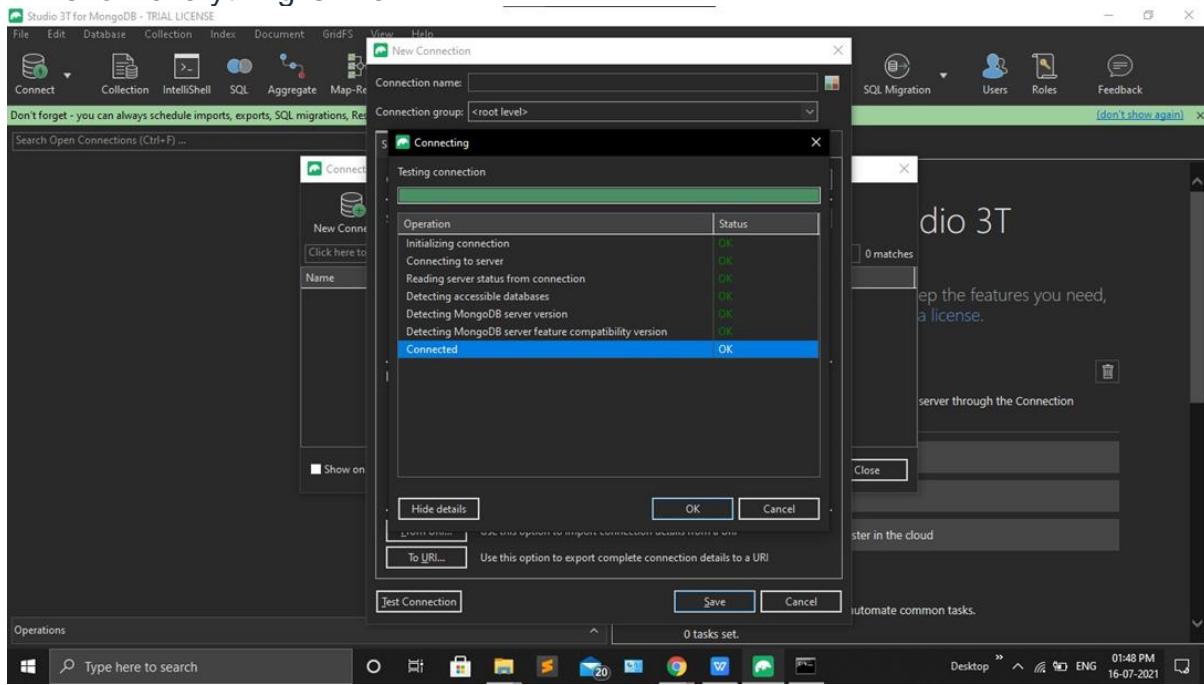
- Now come back to studio 3t, click on “new connection” option it will pop this window out.



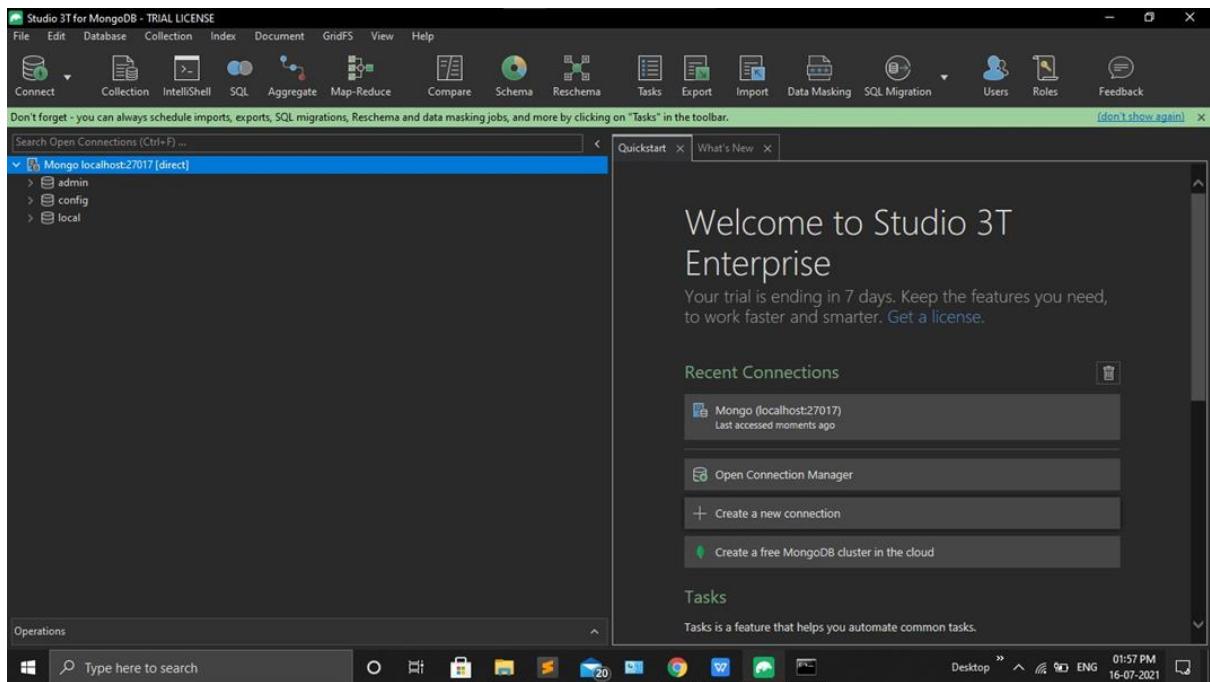
- Then again, click on new connection option it will pop this second window out. which shows root level connection group with localhost on port 27017



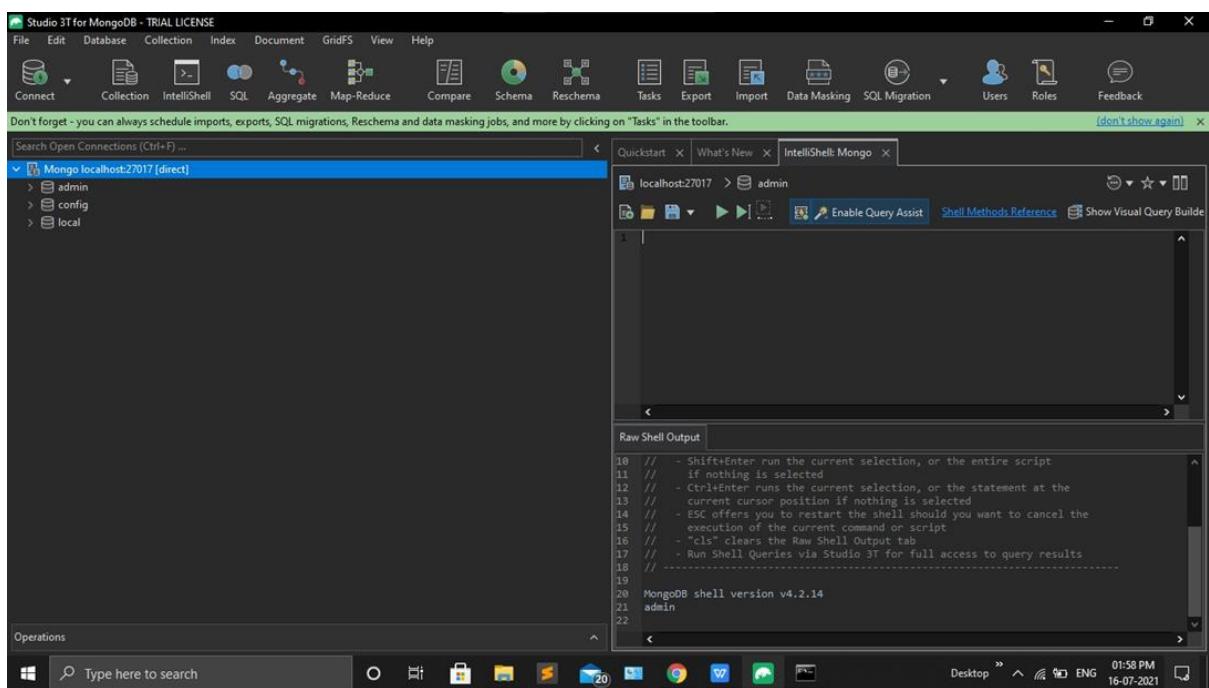
6. Before clicking on saving the connection lets test the connection, all green ticks show everything is fine.



7. Now just save this connection and then u can see this window with admin config and local/



- Click on the IntelliShell and then one window will pop on your left-hand side of screen as you can see below.



- Now Start CRUD operation.

Create User

```
db.createUser({
```

```
    user: "mrugank", pwd: "abc",
    roles:[{role:"userAdminAnyDatabase",db:"admin"}]
})
```

```
1 db.createUser({
2   user: "mrugank",
3   pwd: "abc",
4   roles:[{role:"userAdminAnyDatabase",db:"admin"}]
5 })
```

_id	role	db
563e0a2f4a5c4a0010000001	userAdminAnyD... userAdminAnyD...	admin

Create Collection

```
db.createCollection("emp")
```

```
1 db.createCollection("emp")
```

_id	ok
	1.0

Insert Data into Database

```
db.emp.insert({"empid": 1, "empname" : "abc"})
db.emp.insert({"empid": 2, "empname" : "def"})
db.emp.insert({"empid": 3, "empname" : "ghi"})
db.emp.insert({"empid": 4, "empname" : "jkl"})
db.emp.insert({"empid": 5, "empname" : "mno"})
db.emp.insert({"empid": 6, "empname" : "pqr"})
db.emp.insert({"empid": 7, "empname" : "stu"})
db.emp.insert({"empid": 8, "empname" : "vwx"})
db.emp.insert({"empid": 9, "empname" : "yza"})
db.emp.insert({"empid": 10, "empname" : "bcd"})
```

```

1 db.emp.insert({"empid": 1, "empname": "abc"})
2 db.emp.insert({"empid": 2, "empname": "def"})
3 db.emp.insert({"empid": 3, "empname": "ghi"})
4 db.emp.insert({"empid": 4, "empname": "jkl"})
5 db.emp.insert({"empid": 5, "empname": "mno"})
6 db.emp.insert({"empid": 6, "empname": "pqr"})
7 db.emp.insert({"empid": 7, "empname": "stu"})
8 db.emp.insert({"empid": 8, "empname": "vwx"})
9 db.emp.insert({"empid": 9, "empname": "yza"})
10 db.emp.insert({"empid": 10, "empname": "bcd"})

```

```

var myemp=[ {"empid": 14, "empname": "smith"}, {"empid": 15,"empname": "john"}]
]; db.emp.insert(myemp)

```

```

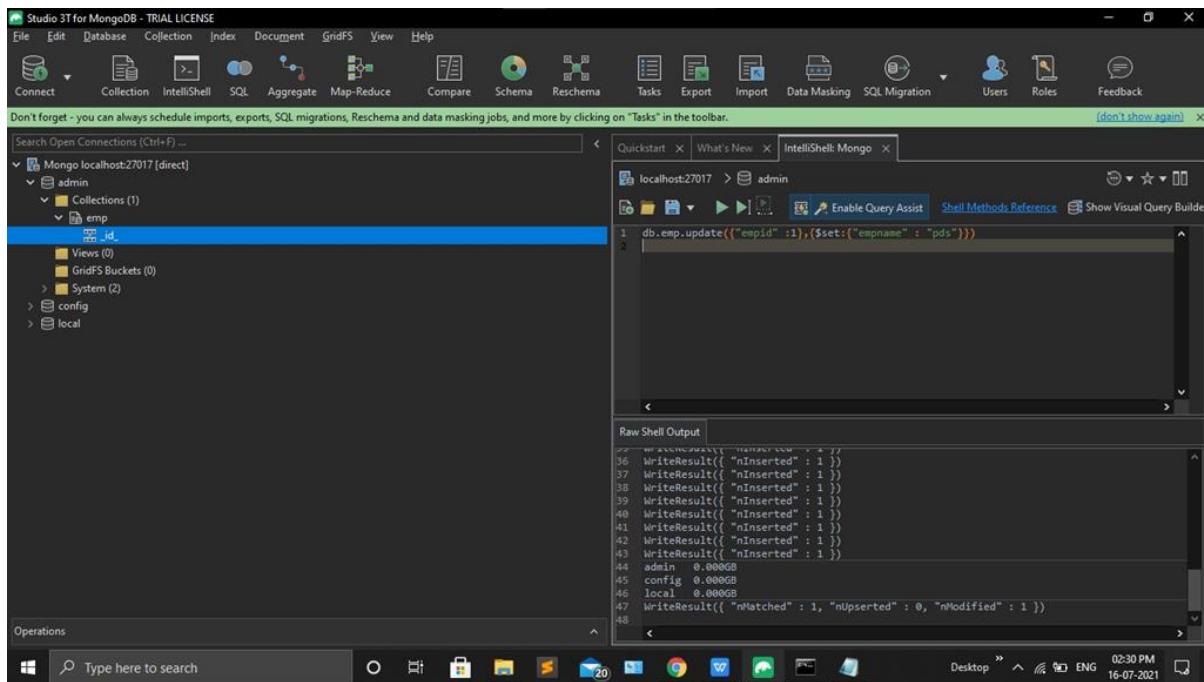
1 var myemp[ {"empid": 14, "empname": "smith"}, {"empid": 15,"empname": "john"} ]
2 db.emp.insert(myemp)

56 WriteResult({ "nInserted": 1 })
57 WriteResult({ "nInserted": 1 })
58 BulkWriteResult({
59   "writeErrors": [ ],
60   "writeConcernErrors": [ ],
61   "nInserted": 2,
62   "nUpended": 0,
63   "nMatched": 0,
64   "nModified": 0,
65   "nRemoved": 0,
66   "nUpserted": [ ]
67 })
68

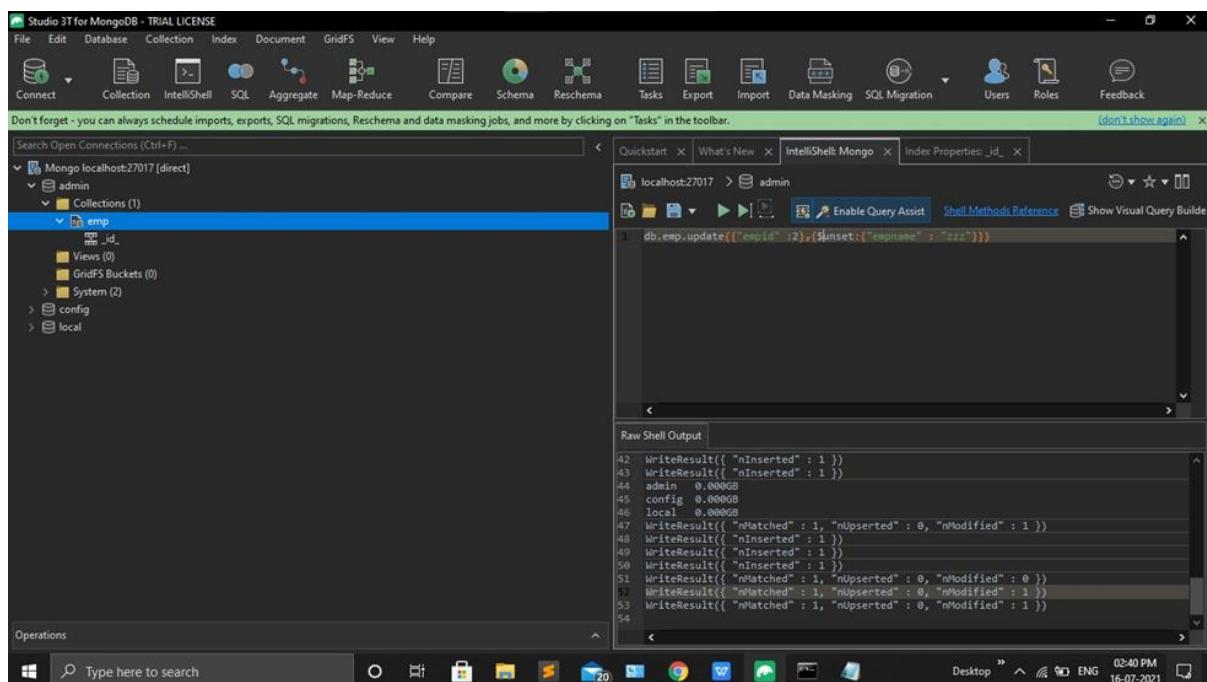
```

UPDATE AND DELETE DATA

```
db.emp.update({"empid": 1},{$set:{"empname": "pds"})}
```



Remove particular field: db.emp.update({"empid":2}, {\$unset:{"empname" : "zzz"}})



```
update value of age by 5:  
db.emp.update({ "empid": 14 }, { $inc: { "age": 5 } })
```

```
db.emp.update({empid: 14}, {$inc: {age: 5}})
```

```
65 "nRemoved" : 0,
66 "upserted" : [ ]
67 }
68 2021-07-16T14:50:27.642+0530 E QUERY [js] Error: need an update object or pipeline :
69 DBCollection.prototype._parseUpdate@src/mongo/shell/collection.js:446:15
70 DBCollection.prototype.update@src/mongo/shell/collection.js:492:18
71 @shell:1:1
72 WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
73 WriteResult({ "nRemoved" : 1 })
```

Removing whole document
`db.emp.remove({empid:7})`

```
db.emp.remove({empid:7})
```

```
66 "upserted" : [ ]
67 }
68 2021-07-16T14:50:27.642+0530 E QUERY [js] Error: need an update object or pipeline :
69 DBCollection.prototype._parseUpdate@src/mongo/shell/collection.js:446:15
70 DBCollection.prototype.update@src/mongo/shell/collection.js:492:18
71 @shell:1:1
72 WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
73 WriteResult({ "nRemoved" : 1 })
```

If no document matching with particular condition exists, then if we set upsert "true", then initially it will also create it.

`db.emp.update({empid:8},{empid:9,empname:"heena"},{upsert:true});`

```
db.emp.update({empid:4},{$rename:{"empname":"nameemp"}});
```

Renaming any particular field

```
db.emp.update({empid:4},{$rename:{"empname":"nameemp"}});
```

```
db.emp.update({empid:4},{$rename:{"empname":"nameemp"}});
```

RETRIEVAL OF DATA

```
db.emp.find().forEach(printjson)
```

```
db.emp.find().forEach(printjson)
```

```
[{"_id": ObjectId("60f148135d3ed97126f4c30"), "empid": 1.0, "empname": "pds"}, {"_id": ObjectId("60f148285d3ed97126f4c3e"), "empid": 1.0, "empname": "martin"}, {"_id": ObjectId("60f1487f5d3ed97126f4c3f"), "empid": 1.0, "empname": "abc"}, {"_id": ObjectId("60f14875d3ed97126f4c40"), "empid": 2.0}, {"_id": ObjectId("60f1487fsd3ed97126f4c41")}]
```

Run the following command and check the output

```
db.emp.find().pretty()
```

```
db.emp.find( { } )
db.emp.find({empname : "smith"} ).forEach(printjson)
db.emp.find({empid : {$gt:2}})

var myemp=db.emp.find( { empid : {$gt:5}}) while(myemp.hasNext()){
print(tojson(myemp.next()));}
```

The screenshot shows the Studio 3T interface for MongoDB. The top menu bar includes File, Edit, Database, Collection, Index, Document, GridFS, View, and Help. The toolbar below has icons for Connect, Collection, IntelliShell, SQL, Aggregate, Map-Reduce, Compare, Schema, Reschema, Tasks, Export, Import, Data Masking, SQL Migration, Users, Roles, and Feedback.

The left sidebar shows connections: admin, config, and local. The main window has tabs for Quickstart and IntelliShell: Mongo. The IntelliShell tab contains the following code:

```
1 var myemp=db.emp.find( { empid : {$gt:5}});
2 while(myemp.hasNext()){ print(tojson(myemp.next()));}
3 }
```

The Result tab displays the output of the query. It shows a table with columns: _id, empid, empname, and age. The data is as follows:

_id	empid	empname	age
60f14875d3eed...	6.0	pqr	
60f14875d3eed...	9.0	heena	
60f14875d3eed...	9.0	yza	
60f14875d3eed...	10.0	bcd	
60f14e335d3eed...	12.0	martin	
60f14e335d3eed...	13.0	martin	
60f14e335d3eed...	14.0	neha	28.0
60f14e655d3eed...	14.0	smith	
60f14e655d3eed...	15.0	john	

At the bottom, it says "1 document selected". The system tray at the bottom right shows the date and time as 16-07-2021 03:12 PM.

Unit 6: Git & GitHub

Learning Outcomes:

- Understand the concept of version control system
- Able create repository, branch, pull request in GitHub
- Understand the use and application of Git and GitHub

There are many ways you can manage and store your writing projects. Some people prefer cloud storage services (like Dropbox) or online editors (like Google Docs), while others use desktop applications (like Microsoft Word). We are going to use something called GitHub.



6.1 About GitHub

If you've never heard of GitHub, it's the world's most popular destination to store and maintain open-source code. That might sound like a crazy place to host your writing, but it's not! After all, code is just lines and lines of text, like your article, story, or dissertation.

Around 2013, GitHub started encouraging people to create repositories for all kinds of information, not just code. GitHub never really left its coding roots, but some people still use it to store writing and other non-coding projects. For example, one person used Git and GitHub to write an instructional book, while another wrote a novel.

6.2 Concept of Git

Git is an open-source program created by Linus Torvalds, of Linux fame. Git tracks changes to documents and makes it easier for multiple people to work on the same document remotely. In tech-speak, it's called a distributed version control system (or distributed VCS). Git doesn't arbitrarily save versions of your documents at set intervals. Instead, it stores changes to your documents only when you tell it to.

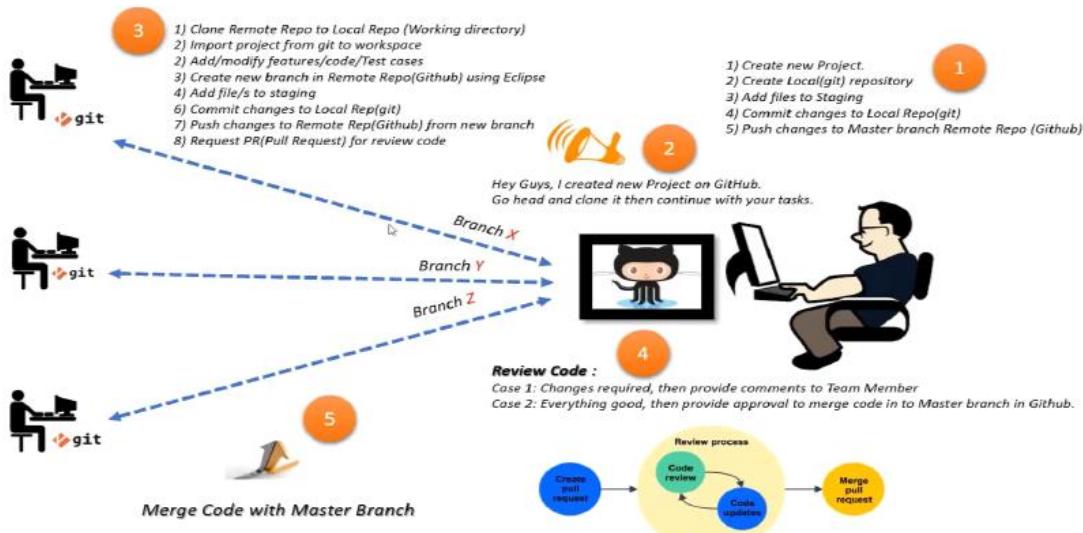


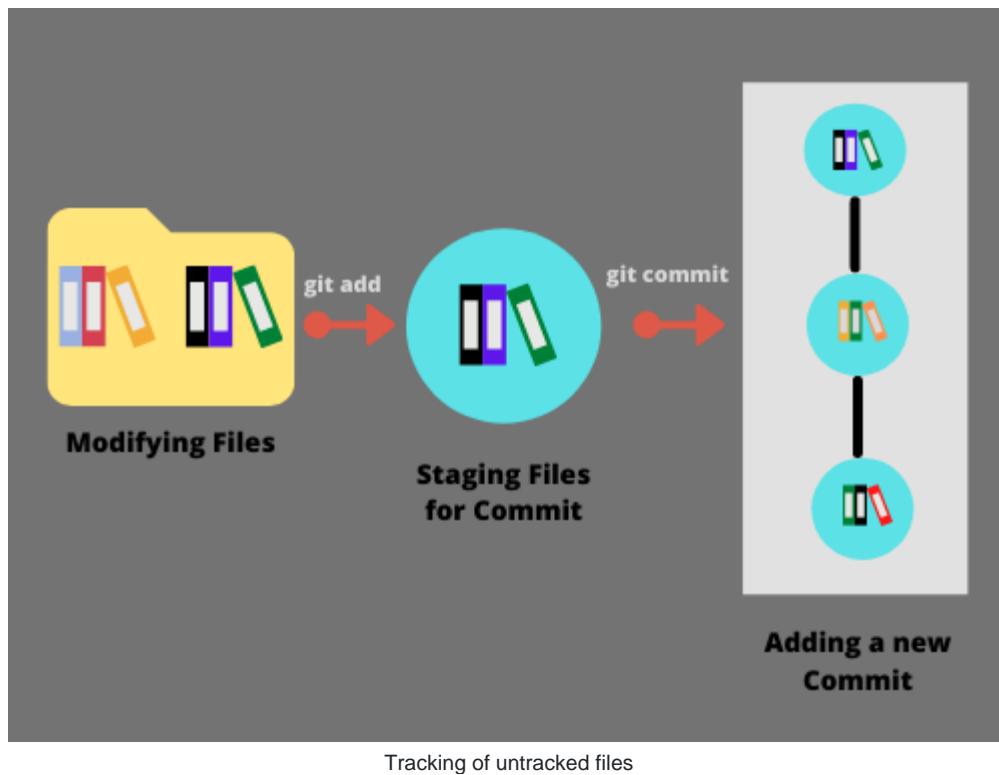
Image: Version Control system

6.3 Git Staging

Staging is the process of organizing and preparing our project files for a commit. It is the intermediate step between modifying our files and storing them permanently in the repository. In this tutorial, we will be looking at the Staging Area and try to understand its role.

What is the Staging Area?

- The staging area is an **intermediate step** between making changes to files and capturing the snapshots of these updates. It is sometimes also known as **Git Index**.
- We reach the staging area when we have completed making changes to our files and are ready to commit these changes permanently.
- Files in the working directory are not tracked by Git. Git will only start tracking changes of those files which are added to the staging area. Whenever we try to commit, only the snapshots of those files are captured which were added to the staging area and are stored permanently in the repository.

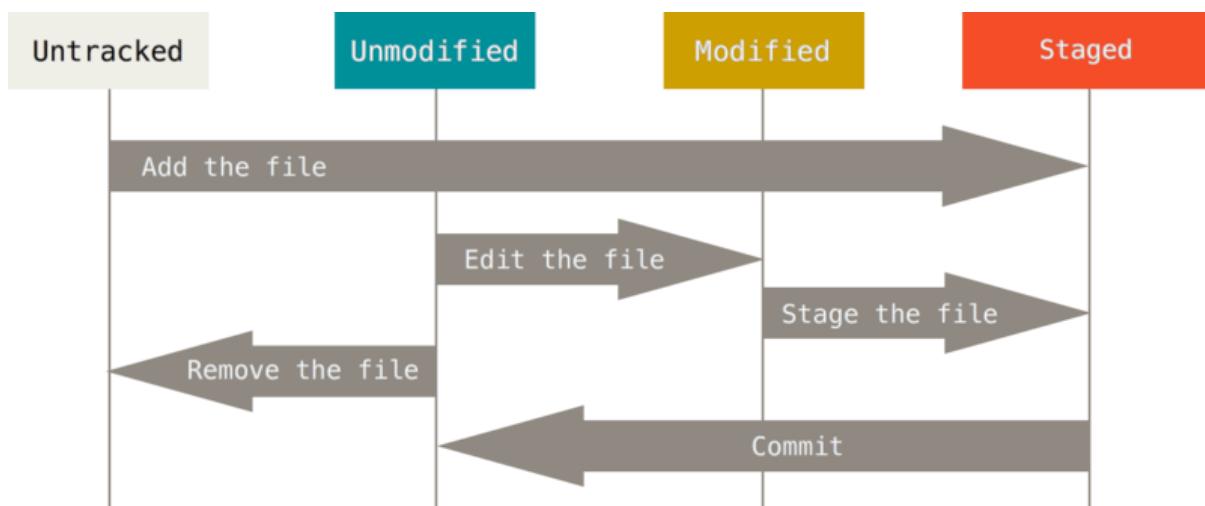


Tracking of untracked files

Why is the Staging Area needed?

Staging helps in keeping the commits **atomic** which makes it easier to understand the project. We can modify multiple files in the working directory but only add some of them to the staging area and then commit thus making it easier to roll back just a part of the project that involves those files instead of reverting the entire project to a previous version. In this way, staging gives developers more control over how they want to achieve version control.

Tracked & Untracked Stages of Github



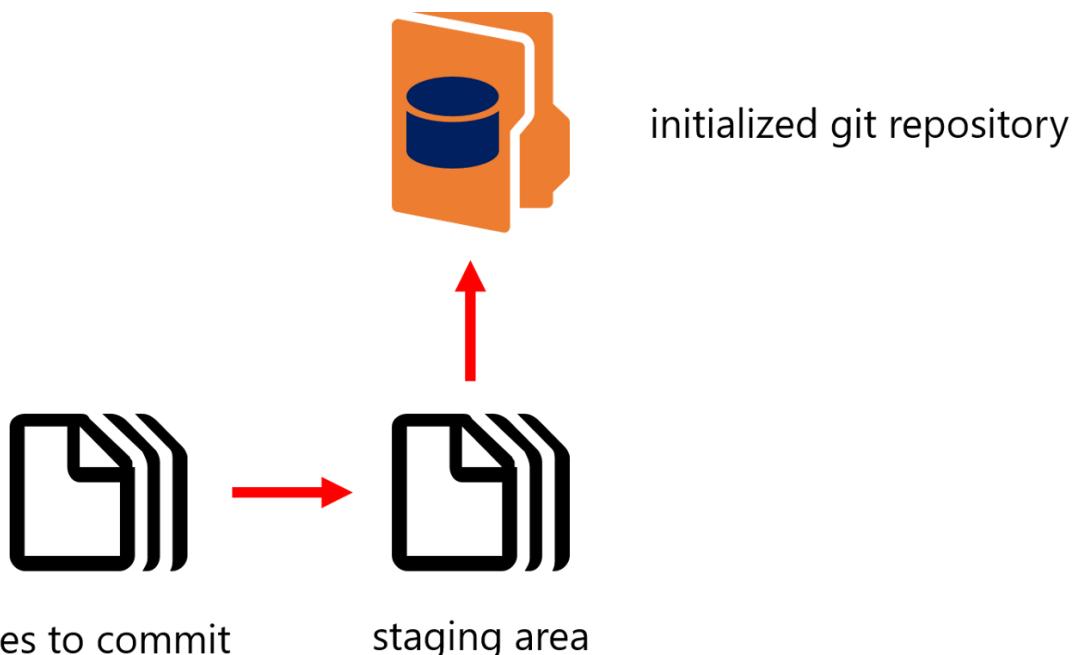
Intermediate Snapshots of Staging

Remember that each file in your working directory can be in one of two states: tracked or untracked. In Github terms “stage” and “snapshot” are analogous to use. Tracked files are files that were in the last stages (any other than tracked stage); they can be unmodified, modified, or staged. Untracked files are everything else – any files in your working directory that were not in your last snapshot and are not in your staging area. When you first clone a repository, all of your files will be tracked and unmodified because you just checked them out and haven’t edited anything.

As you edit files, Git sees them as modified, because you’ve changed them since your last commit. You stage these modified files and then commit all your staged changes, and the cycle repeats.

6.4 GitHub Commands

With Git, you record local changes to your code using a *command-line* tool, called the “Git Shell” (you can use Git in other command-line tools — Refer to Git Shell through the following sections). Command-line lets you enter commands to view, change, and manage files and folders in a simple terminal, instead of using a graphical user interface (GUI). If you have not used command-line before, don’t worry, once you get started, it is incredibly straightforward.



Essentially, when using Git, you make changes to your code files as you normally would during the development process. When you have completed a coding milestone, or want to snapshot certain changes, you add the files you changed to a staging area and then commit them to the version history of your project (repository) using Git. Below, you’ll learn about the Git commands you use for those steps.

Terminal Commands

While using Git on the command line, chances are you will also use some basic terminal commands while going through your project and system files / folders, including:

- `pwd` - check where you are in the current file system
- `ls` - list files in the current directory (folder)
- `cd [directory-name]` - moves to the given directory name or path
- `mkdir [directory-name]` - makes a new directory with the given name

Creating Repositories

When you wish to utilize Git for a project, the first command you must do is `git init`, with the name of your project:

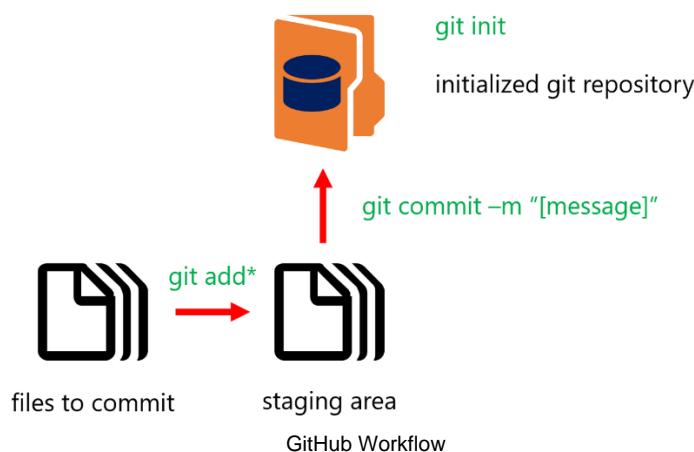
`git init [project-name]`

You run this command on the Git Shell command-line in the main *directory* (folder) of your project, which you can navigate to in the Shell using the commands listed above. Once you run this command, Git creates a hidden `.git` file inside the main directory of your project. This file tracks the version history of your project and is what turns the project into a Git *repository*, enabling you to run Git commands on it.

Making Changes

- `git add [file]` or `git add *`
Once you make changes to your files and choose to snapshot them to your project's version history, you have to add them to the staging area with `git add`, by file name, or by including all of the files in your current folder using `git add *`.
- `git commit -m “[message]”`
To finally commit the changes you made to your files from the staging area to your repository's version history, you need to run `git commit` with a descriptive message of what changes you made.
- `git status`
If at any point, you wish to view a summary of the files you have changed and not yet committed, simply run `git status` in your project's repository on the Git Shell command-line.

How It Works

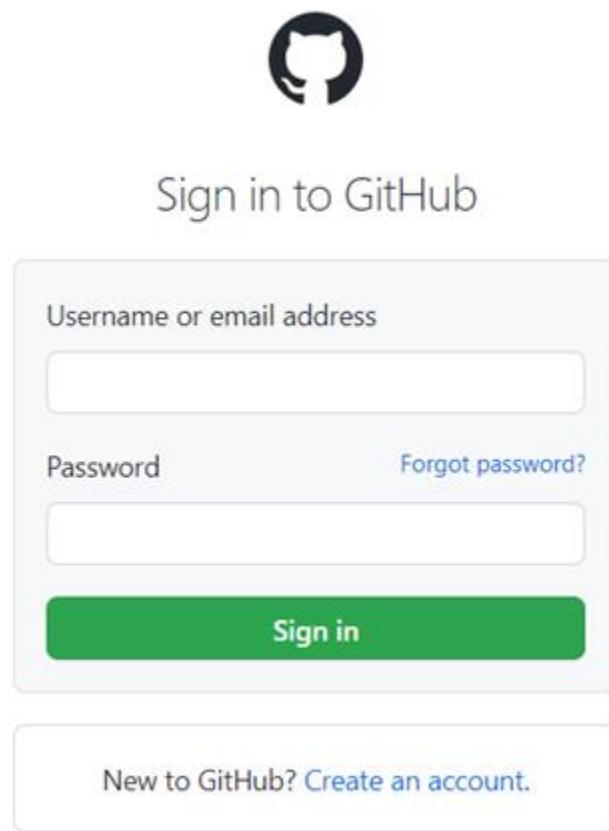


Now, with the basic Git commands in place, you can utilize Git to snapshot the version history of your project. Simply initialize a new repository by running `git init` in your project's main directory. Using `git add *`, or `git add` with specific file names, you add your changes to the staging area. Finally, using `git commit`, you can add your changes to the repository's version history.

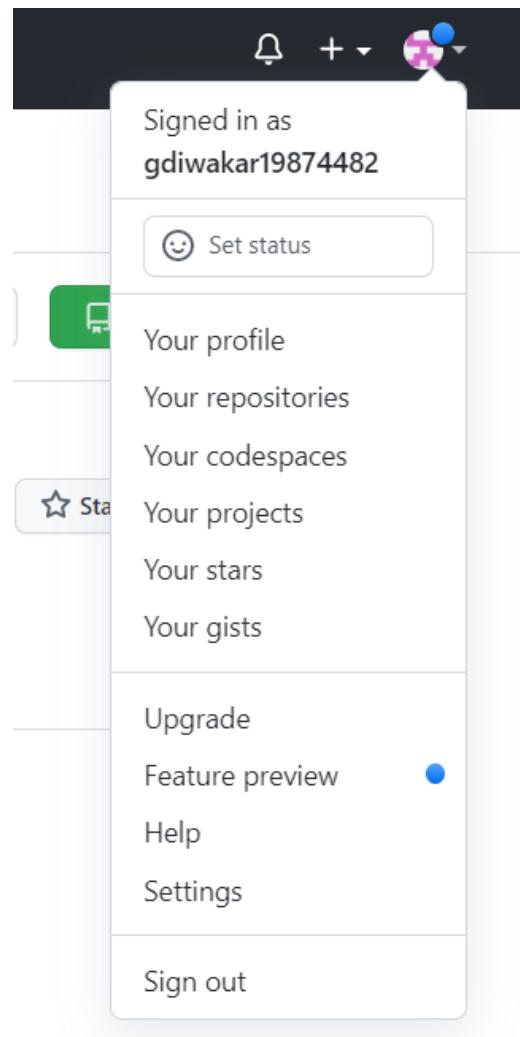
6.5 Practical: Working with GitHub Environment

1. Register / Signin yourself on Github.

[GitHub: Where the world builds software · GitHub](#)



2. Switch to Repositories tab on your profile



3. Create a new repository

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * Repository name *

 gdiwakar19874482 / test2 

Great repository names are short and memorable. Need inspiration? How about [fuzzy-funicular](#)?

Description (optional)

 **Public**
Anyone on the internet can see this repository. You choose who can commit.

 **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

Add a README file
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

4. Open Anaconda Command Prompt and install git library

```
pip install git
```

5. Create a directory with two random .txt files. Place any piece of text in those files.

Configure the access of github profile on local github library

```
git config --global user.name username
```

```
git config --global user.email useremail
```

** Place your github login username and email information.

6. Change the folder to as current working directory. And run git init code

```
(keras-gpu) C:\Git_hub_test>git init
Initialized empty Git repository in C:/Git_hub_test/.git/
```

7. Git status shows files are untracked

```
(keras-gpu) C:\Git_hub_test>git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    file1.txt
    file2.txt

nothing added to commit but untracked files present (use "git add" to track)
```

8. Add file1.txt to stage and check git status

```
(keras-gpu) C:\Git_hub_test>git add file1.txt

(keras-gpu) C:\Git_hub_test>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   file1.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    file2.txt
```

9. Now add another file2.txt to staging stage and check git status

```
keras-gpu) C:\Git_hub_test>git add file2.txt

keras-gpu) C:\Git_hub_test>git status
on branch master

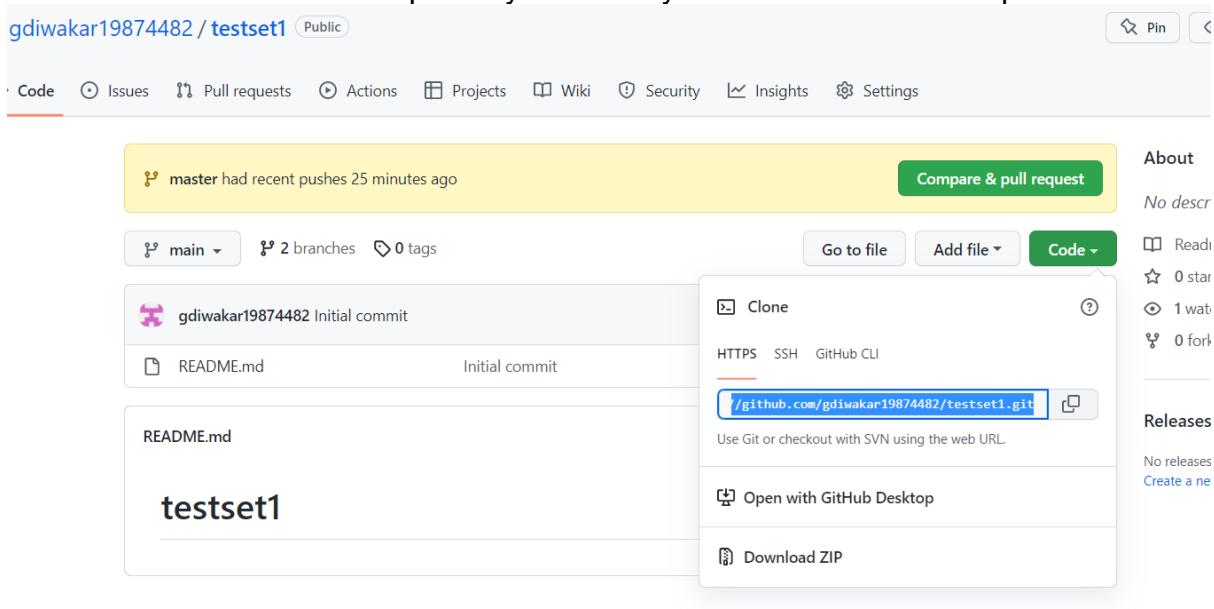
 0 commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   file1.txt
    new file:   file2.txt
```

10. Commit the traced file using git commit command. You can also mention the commit message.

```
(keras-gpu) C:\Git_hub_test>git commit -m "Test_Github"
[master (root-commit) 46b0a33] Test_Github
 2 files changed, 2 insertions(+)
 create mode 100644 file1.txt
 create mode 100644 file2.txt
```

11. Add the address of repository to where you test files are to be uploaded.



The screenshot shows a GitHub repository page for 'testset1'. The 'Code' tab is selected. A dropdown menu is open over the 'Code' tab, showing options like 'Clone' with a URL 'https://github.com/gdiwakar19874482/testset1.git', 'Add file', 'GitHub Desktop', and 'Download ZIP'.

```
(keras-gpu) C:\Git_hub_test>git remote add origin https://github.com/gdiwakar19874482/testset1.git
```

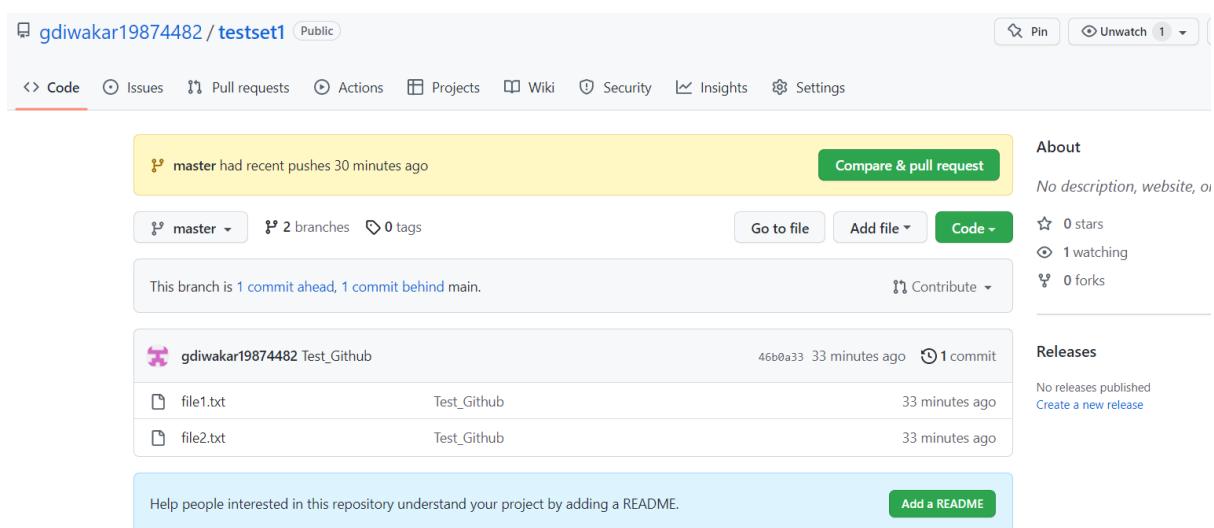
12. Upload your local repository to github using git push command

```
(keras-gpu) C:\Git_hub_test>git push origin master
info: please complete authentication in your browser...
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 285 bytes | 285.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:     https://github.com/gdiwakar19874482/testset1/pull/new/master
remote:
To https://github.com/gdiwakar19874482/testset1.git
 * [new branch]      master -> master

(keras-gpu) C:\Git_hub_test>git remote -v
origin  https://github.com/gdiwakar19874482/testset1.git (fetch)
origin  https://github.com/gdiwakar19874482/testset1.git (push)
```

**Note: GUI window will request for read/ write access.

13. You will observe and conclude that your local repository gets uploaded to github repository



The screenshot shows a GitHub repository page for 'gdiwakar19874482 / testset1'. The 'Code' tab is selected. A yellow banner at the top indicates 'master had recent pushes 30 minutes ago'. Below this, there are buttons for 'Go to file', 'Add file', and 'Code'. The main content area shows a message 'This branch is 1 commit ahead, 1 commit behind main.' followed by a list of commits:

Author	Commit Message	Date	Actions	
gdiwakar19874482	Test_Github	46b0a33 33 minutes ago	1 commit	
	file1.txt	Test_Github	33 minutes ago	
	file2.txt	Test_Github	33 minutes ago	

At the bottom, there's a note to 'Help people interested in this repository understand your project by adding a README.' and a 'Add a README' button.

Module - II

Internet of Things

Unit 1: IoT Fundamentals

Learning Outcomes:

- Understand the basic concept of IoT
- Understand the use and application of IoT

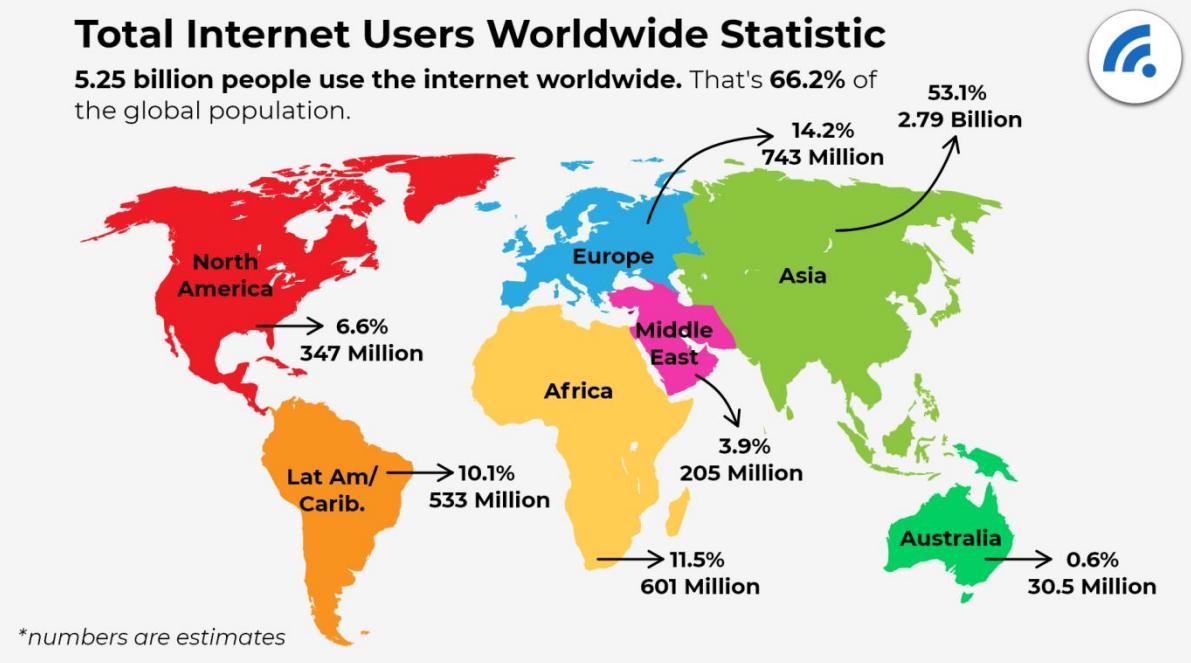
1.1 Internet Usage and Population Statistics

A total of 5 billion people around the world use the internet today – equivalent to 63 percent of the world's total population.

Internet users continue to grow too, with the latest data indicating that the world's connected population grew by almost 200 million in the 12 months to April 2022.

There are now fewer than 3 billion people who remain “unconnected” to the internet, with the majority of these people located in Southern and Eastern Asia, and in Africa.

That means that there's still plenty more work to do before the world reaches the goal of “universal access”, and the quality of people's internet access is also an important consideration.



However, internet users continue to grow at an annual rate of more than 4 percent, and current trends suggest that two-thirds of the world's population should be online by the middle of 2023.

What's more, the ongoing coronavirus pandemic continues to have a meaningful impact on internet user research, so actual user figures and growth rates may be higher than current data suggests.

The vast majority of the world's internet users – 92.4 percent – use a mobile phone to go online at least some of the time, and mobile phones now account for more than half of our online time, and more than half of the world's web traffic.

However, roughly two-thirds of internet users in the world's larger economies still use laptops and desktops for at least some of their online activities.

1.2 What is Internet of Things?

Introduction

Internet of Things (IoT) comprises things that have unique identities and are connected to the internet.

Existing devices, such as networked computers or 4G enabled mobile phones already have some form of unique identities and are also connected to the internet, the focus on IoT is in the configuration, control and networking via the internet of devices or things, that are traditionally not associated with the Internet. These include devices such as thermostats, utility meters, a blue tooth-connected headset, irrigation pumps and sensor or control circuits for an electric car's engine.

Experts forecast that by the year 2020 there will be a total of 50 billion devices/ things connected to the internet.

The scope of IoT is not limited to just connected things (Devices, appliance, machines) to the Internet. Applications on IoT networks extract and create information from lower-level data by filtering, processing, categorizing, condensing and contextualizing the data. The information obtained is then organized and structured to infer knowledge about the system and/or its user, its environment and its operations and progress towards its objectives, allowing a smarter performance.

Definition and characteristics of IoT:

Definition

A dynamic global network infrastructure with self – configuring based on standard and interoperable communication protocols where physical and virtual “things” have identified, physical attributes, and virtual personalities and use intelligent interfaces, often communicate data associated with users and their environment Characteristics

Dynamic and self-Adapting:

IoT devices and systems may have the capability to dynamically adapt with the changing contexts and take actions based on their operating condition.Ex: Surveillance cameras can adapt their modes based on whether it is day or night.

Self – Configuring:

IoT devices may have self-Configuring capability allowing a large number of devices to work together to provide certain functionality.

Interoperable communication protocols:

IoT Devices may support a number of interoperable communication protocols and can communicate with other devices and also with the infrastructure.

Unique Identity:

Each IoT devices has a unique identity and a unique identifier.(IPaddress, URI).IoT systems may have intelligent interfaces which adapt based on the context, allow communication with users and the environment contexts.

Integrated into information network:

IoT devices are usually integrated into the information network that allows them to communicate and exchange data with other devices and systems.

1.3 Why IoT?

When something is connected to the internet, that means that it can send information or receive information, or both. This ability to send and/or receive information makes things “smart.”

Let's use smartphones again as an example. Right now you can listen to just about any song in the world, but it's not because your phone actually has every song in the world stored on it. It's because every song in the world is stored somewhere else, but your phone can send information (asking for that song) and then receive information (streaming that song on your phone).

To be smart, a thing doesn't need to have super storage or a super computer inside of it - it just needs access to it. All a thing has to do is connect to super storage or to a

super computer. In the Internet of Things, all the things that are being connected to the internet can be put into three categories:

Things that collect information and then send it.

Things that receive information and then act on it.

Things that do both.

And all three of these have enormous benefits that compound on each other.

1. Collecting and Sending Information

Sensors could be temperature sensors, motion sensors, moisture sensors, air quality sensors, light sensors, you name it. These sensors, along with a connection, allow us to automatically collect information from the environment which, in turn, allows us to make more intelligent decisions.

On a farm, automatically getting information about the soil moisture can tell farmers exactly when their crops need to be watered. Instead of watering too much (which can be an expensive over-use of irrigation systems) or watering too little (which can be an expensive loss of crops), the farmer can ensure that crops get exactly the right amount of water. This enables farmers to increase their crop yield while decreasing their associated expenses.

Just as our sight, hearing, smell, touch, and taste allow us, humans, to make sense of the world, sensors allow machines (and the humans monitoring the machines) to make sense of the world.

2. Receiving and Acting on Information

We're all very familiar with machines getting information and then acting. Your printer receives a document and it prints it. Your car receives a signal from your car keys and the doors open. The examples are endless.

Whether it's a simple as sending the command "turn on" or as complex as sending a 3D model to a 3D printer, we know that we can tell machines what to do from far away. So what?

The real power of the Internet of Things arises when things can do both of the above. Things that collect information and send it, but also receive information and act on it.

3. Doing Both: The Goal of an IoT System

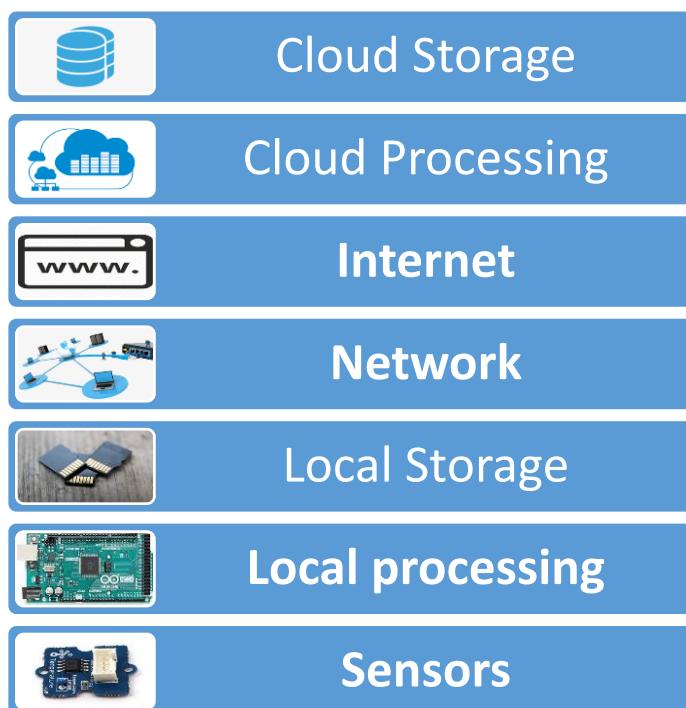
Let's quickly go back to the farming example. The sensors can collect information about the soil moisture to tell the farmer how much to water the crops, but you don't actually need the farmer. Instead, the irrigation system can automatically turn on as needed, based on how much moisture is in the soil.

You can take it a step further too. If the irrigation system receives information about the weather from its internet connection, it can also know when it's going to rain and decide not to water the crops today because they'll be watered by the rain anyways.

And it doesn't stop there! All this information about the soil moisture, how much the irrigation system is watering the crops, and how well the crops actually grow can be collected and sent to supercomputers that run amazing algorithms that can make sense of all this information.

And that's just one kind of sensor. Add in other sensors like light, air quality, and temperature, and these algorithms can learn much, much more. With dozens, hundreds, thousands of farms all collecting this information, these algorithms can create incredible insights into how to make crops grow the best, helping to feed the world.

1.4 IOT Architecture



Sensors

They transform analog data given from scanning the environment to digital data, but they merely do any processing. On the bright side, they don't consume much power and can live on batteries for a long time. Sensors are present in everyday life more than you would expect. They improve industry, agriculture, homes, transportation or

smart phones for example. They are tools which help monitoring the environment, collecting data about it and, with the help of computers, acting accordingly.

Local processing and storage devices

Local processing devices are the second level and third in IoT. At this point, data is locally stored and processed, ideally not sent forwards unless relevant. This part is explained in detail in the hardware section, as said devices are nothing more than microcontrollers and embeded boards, which handle the data they receive from the sensors.

Network and Internet

There is hardware which connects to the previously described devices, pulls out data and sends it to the cloud to be stored. There are 4 protocols used at this level: CoAP, MQTT (less secure and designed for machine to machine communication), HTTP (web protocol) and XMPP which functions as a chat.

Cloud

In the cloud, which comes next, data is collected and the main goal is for it to reach the point of making predictions based on the stored information. The cloud however, even though it represents one of the most useful features of the internet, is not used properly. Data sent to the cloud didn't reach the level of being formerly processed. Which means there is no preselected data. The cloud is constantly loaded with irrelevant information and thus losing it's property of being practical.

1.5 What is industrial IoT?

Industrial IoT (IIoT) refers to the application of IoT technology in industrial settings, especially with respect to instrumentation and control of sensors and devices that engage cloud technologies. Recently, industries have used machine-to-machine communication (M2M) to achieve wireless automation and control. But with the emergence of cloud and allied technologies (such as analytics and machine learning), industries can achieve a new automation layer and with it create new revenue and business models. IIoT is sometimes called the fourth wave of the industrial revolution, or Industry 4.0. The following are some common uses for IIoT:

- Smart manufacturing
- Connected assets and preventive and predictive maintenance
- Smart power grids
- Smart cities
- Connected logistics
- Smart digital supply chains

1.6 IoT Applications by Industries

The ubiquity of the Internet of Things is a fact of life thanks to its adoption by a wide range of industries. IoT's versatility makes it an attractive option for so many businesses, organizations, and government branches, that it doesn't make sense to ignore it. Let us learn about IoT applications across industries below:

IoT Applications in Agriculture

For indoor planting, IoT makes monitoring and management of micro-climate conditions a reality, which in turn increases production. For outside planting, devices using IoT technology can sense soil moisture and nutrients, in conjunction with weather data, better control smart irrigation and fertilizer systems. If the sprinkler systems dispense water only when needed, for example, this prevents wasting a precious resource.

IoT Applications in Consumer Use

For the private citizen, IoT devices in the form of wearables and smart homes make life easier. Wearables cover accessories such as Fitbit, smartphones, Apple watches, health monitors, to name a few. These devices improve entertainment, network connectivity, health, and fitness.

Smart homes take care of things like activating environmental controls so that your house is at peak comfort when you come home. Dinner that requires either an oven or a crockpot can be started remotely, so the food is ready when you arrive. Security is made more accessible as well, with the consumer having the ability to control appliances and lights remotely, as well as activating a smart lock to allow the appropriate people to enter the house even if they don't have a key.

IoT Applications in Healthcare

First and foremost, wearable IoT devices let hospitals monitor their patients' health at home, thereby reducing hospital stays while still providing up to the minute real-time information that could save lives. In hospitals, smart beds keep the staff informed as to the availability, thereby cutting wait time for free space. Putting IoT sensors on critical equipment means fewer breakdowns and increased reliability, which can mean the difference between life and death.

Elderly care becomes significantly more comfortable with IoT. In addition to the above-mentioned real-time home monitoring, sensors can also determine if a patient has fallen or is suffering a heart attack.

IoT Applications in Insurance

Even the insurance industry can benefit from the IoT revolution. Insurance companies can offer their policyholders discounts for IoT wearables such as Fitbit. By employing

fitness tracking, the insurer can offer customized policies and encourage healthier habits, which in the long run, benefits everyone, insurer, and customer alike.

IoT Applications in Manufacturing

The world of manufacturing and industrial automation is another big winner in the IoT sweepstakes. RFID and GPS technology can help a manufacturer track a product from its start on the factory floor to its placement in the destination store, the whole supply chain from start to finish. These sensors can gather information on travel time, product condition, and environmental conditions that the product was subjected to.

Sensors attached to factory equipment can help identify bottlenecks in the production line, thereby reducing lost time and waste. Other sensors mounted on those same machines can also track the performance of the machine, predicting when the unit will require maintenance, thereby preventing costly breakdowns.

IoT Applications in Retail

IoT technology has a lot to offer the world of retail. Online and in-store shopping sales figures can control warehouse automation and robotics, information gleaned from IoT sensors. Much of this relies on RFIDs, which are already in heavy use worldwide.

Mall locations are iffy things; business tends to fluctuate, and the advent of online shopping has driven down the demand for brick and mortar establishments. However, IoT can help analyze mall traffic so that stores located in malls can make the necessary adjustments that enhance the customer's shopping experience while reducing overhead.

Speaking of customer engagement, IoT helps retailers target customers based on past purchases. Equipped with the information provided through IoT, a retailer could craft a personalized promotion for their loyal customers, thereby eliminating the need for costly mass-marketing promotions that don't stand as much of a chance of success. Much of these promotions can be conducted through the customers' smartphones, especially if they have an app for the appropriate store.

IoT Applications in Transportation

By this time, most people have heard about the progress being made with self-driving cars. But that's just one bit of the vast potential in the field of transportation. The GPS, which, if you think of it, is another example of IoT, is being utilized to help transportation companies plot faster and more efficient routes for trucks hauling freight, thereby speeding up delivery times.

There's already significant progress made in navigation, once again alluding to a phone or car's GPS. But city planners can also use that data to help determine traffic patterns, parking space demand, and road construction and maintenance.

There's even a possibility that apps can be made that can prevent a car from starting if the driver is inebriated!

IoT Applications in Utilities/Energy

IoT sensors can be employed to monitor environmental conditions such as humidity, temperature, and lighting. The information provided by IoT sensors can aid in the creation of algorithms that regulate energy usage and make the appropriate adjustments, eliminating the human equation (and let's face it, who of us hasn't forgotten to switch off lights in a room or turn down the thermostat?).

With IoT-driven environmental control, businesses and private residences can experience significant energy savings, which in the long run, benefits everyone, including the environment!

On a larger scale, data gathered by the Internet of Things can be used to help run municipal power grids more efficiently, analyzing factors such as usage. Also, the sensors can help pinpoint outages faster, thereby increasing the response time of repair crews and decreasing blackout times.

1.7 The Future of the Internet of Things

So, considering the above, just what does the future have in store for the Internet of Things?

A Gartner report predicts that connected devices across all manner of technologies will hit 20.6 billion. That number could hit 1 trillion by 2025, according to HP, and that's just a staggering figure. According to a Cisco report, the next decade will see IoT devices creating \$14.4 trillion worth of value across several industries like the ones mentioned above.

In other words, the Internet of Things is poised to create life-changing conditions in our lives, both in a professional and personal capacity. Many of the innovations mentioned are already in place to one extent or another. One thing's for sure: there's no going back. The IoT offers an unprecedented degree of control and efficiency that no industry can ignore.

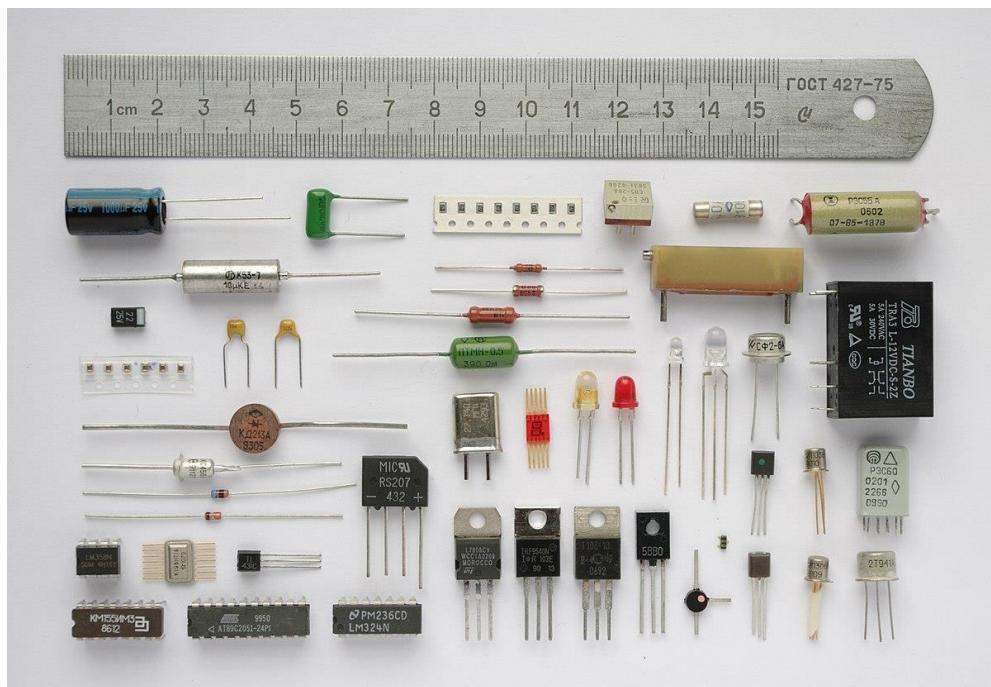
Unit 2: Electronics Concept

Learning Outcomes:

- Understand the Concept of Basic electronics
 - Able to create simple electronics circuit
 - Understand the use of electronics components and its applications

2.1 Electronics Components

An electronic component is a physical entity in an electronic system used to affect movement of electrons. Electronic components have a number of electronic terminals or leads which connect to other electronic components over wire to create an electronic circuit.

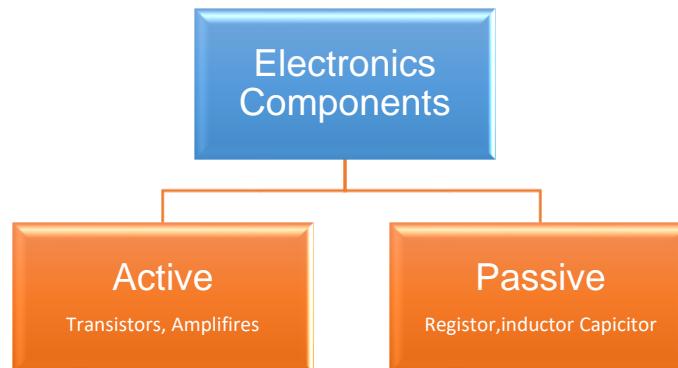


Source: <https://en.wikipedia.org/wiki/File:Componentes.JPG>

Classification of Electronics Components

They can be classified into two types i.e., Active Components and Passive Components. Active elements are those which possess gain. They can give energy to the circuit. On the contrary passive elements do not possess gain and they cannot give energy continuously to the circuit.

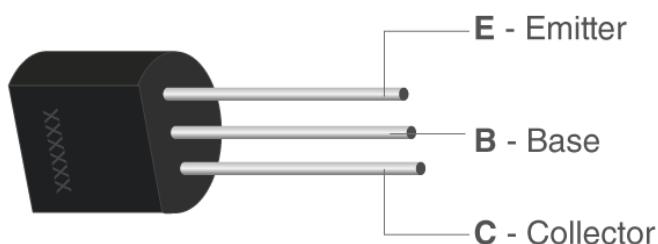
Passive components cannot amplify or energize the energy of the signal associated with them, they can only attenuate it, while active components can energize or amplify the signal.



1. Transistor

A transistor is a semiconductor device used to amplify and switch electronic signals and electrical power.

Transistor is formed from two words “Transfer” and “Resistor”. Thus, it transfers resistance from one part of circuit to other. If at input side the resistance is high then the resistance at the output side will be low. Transistor is a three terminal device which can act as a switch or an amplifier. It can be controlled either by voltage or current thus it is called voltage-controlled device or a current controlled device.



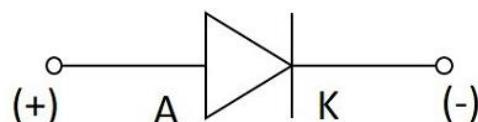
Transistor applications

- Transistor have application in both Analog and Digital Circuit
- Power Regulator Circuits
- Microprocessor ICs
- Mobile Phone charger
- AC to DC adaptors

- TV Power supply section
- SMPS
- Electronics Switching devices etc.

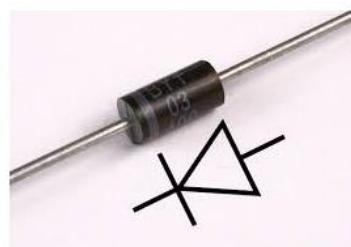
2. Diode

A semiconductor diode is a two terminal electronic component with a PN junction. This is also called as a Rectifier.



Symbol of a Diode

The anode which is the positive terminal of a diode is represented with A and the cathode, which is the negative terminal is represented with K. To know the anode and cathode of a practical diode, a fine line is drawn on the diode which means cathode, while the other end represents anode.



Representing anode and cathode of a practical diode through its symbol

Biasing of a Diode

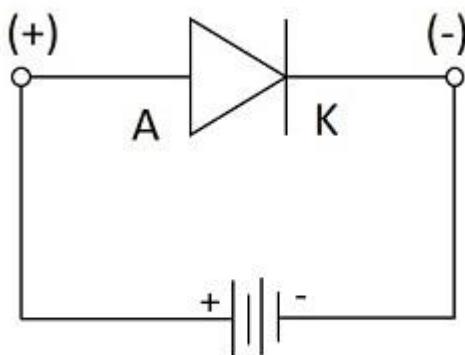
When a diode or any two-terminal component is connected in a circuit, it has two biased conditions with the given supply. They are Forward biased condition and Reverse biased condition.

Forward Biased Condition

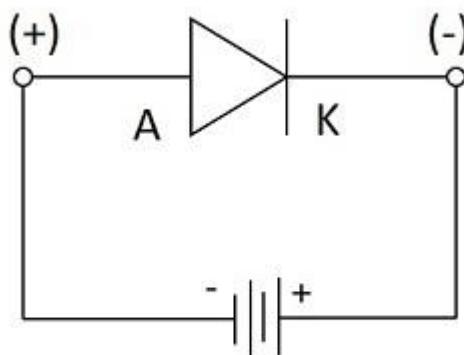
When a diode is connected in a circuit, with its **anode to the positive** terminal and **cathode to the negative** terminal of the supply, then such a connection is said to be **forward biased** condition.

Reverse Biased Condition

When a diode is connected in a circuit, with its **anode to the negative** terminal and **cathode to the positive** terminal of the supply, then such a connection is said to be **Reverse biased** condition.



Forward biased Connection

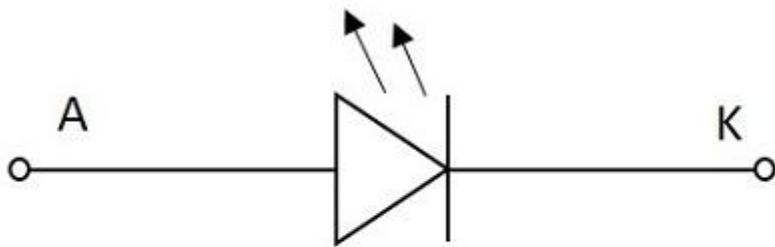


Reverse biased Connection

3. LED – Light Emitting Diodes

This one is the most popular diodes used in our daily life. This is also a normal PN junction diode except that instead of silicon and germanium, the materials like gallium arsenide, gallium arsenide phosphide are used in its construction.

The figure below shows the symbol of a Light emitting diode.



Symbol of LED

Like a normal PN junction diode, this is connected in forward bias condition so that the diode conducts. The conduction takes place in a LED when the free electrons in the conduction band combine with the holes in the valence band. This process of recombination **emits light**. This process is called as **Electroluminescence**. The color of the light emitted depends upon the gap between the energy bands.

The LEDs for non-visible Infrared light are used mostly in remote controls.



Source: https://www.tutorialspoint.com/basic_electronics/basic_electronics_optoelectric_diodes.htm

Applications of LED

There are many applications for LED such as –

In Displays

- Especially used for seven segment display
- Digital clocks
- Microwave ovens
- Traffic signalling
- Display boards in railways and public places
- Toys

In Electronic Appliances

- Stereo tuners
- Calculators
- DC power supplies
- On/Off indicators in amplifiers
- Power indicators

Commercial Use

- Infrared readable machines
- Barcode readers
- Solid state video displays

Optical Communications

- In Optical switching applications
- For Optical coupling where manual help is unavailable
- Information transfer through FOC
- Image sensing circuits
- Burglar alarms

- In Railway signalling techniques
- Door and other security control systems

4. Resistor

Resistor is an electrical component that reduces the electric current. The resistor's ability to reduce the current is called resistance and is measured in units of ohms (symbol: Ω). The resistor's resistance limits the flow of electrons through a circuit. Resistors come in a variety of shapes and sizes.

Resistor Color Code Chart

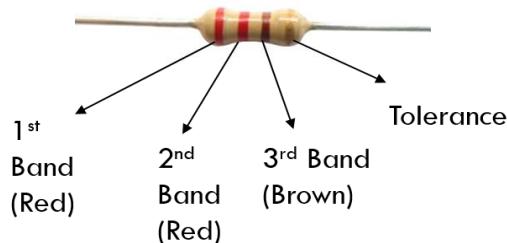
The following color code chart is used to calculate the value of resistance of a resistor. All Resistors follow a colour coded pattern as show below.

Color	Color	1st Band	2nd Band	3rd Band Multiplier	4th Band Tolerance
Black		0	0	$\times 1\Omega$	
Brown		1	1	$\times 10\Omega$	$\pm 1\%$
Red		2	2	$\times 100\Omega$	$\pm 2\%$
Orange		3	3	$\times 1k\Omega$	
Yellow		4	4	$\times 10k\Omega$	
Green		5	5	$\times 100k\Omega$	$\pm 0.5\%$
Blue		6	6	$\times 1M\Omega$	$\pm 0.25\%$
Violet		7	7	$\times 10M\Omega$	$\pm 0.10\%$
Grey		8	8	$\times 100M\Omega$	$\pm 0.05\%$
White		9	9	$\times 1G\Omega$	
Gold				$\times 0.1\Omega$	$\pm 5\%$
Silver				$\times 0.01\Omega$	$\pm 10\%$

Source: <https://sites.google.com/view/trainingday-1/basics-of-electronics/components?authuser=0>

How to Calculate Resistance of a Resistor?

3 Band Resistor Resistance Calculation



Source: <https://sites.google.com/view/trainingday-1/basics-of-electronics/components?authuser=0>

Resistance= 1st band value (first digit) 2nd band value (second digit) multiplied by multiplier value of 3rd band.

$$= 1^{\text{st}} \text{ band } 2^{\text{nd}} \text{ band} \times 3^{\text{rd}} \text{ band} \text{ (Multiplier)}$$

$$= 22 \times 10 \Omega \text{ (ohms)}$$

$$= 220 \Omega \text{ (ohms)}$$

Ohm's Law

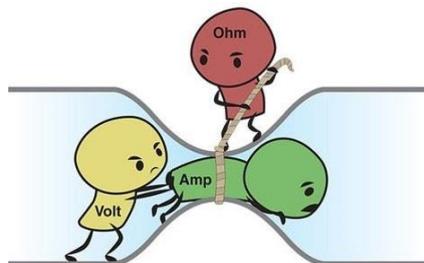
Ohm's law states that the current through a conductor between two points is directly proportional to the voltage or potential difference between the two points provided the temperature is constant for a constant length and area.

Ohm's Law Formula

$$\text{Voltage} = \text{Current} \times \text{Resistance}$$

$$V = I \times R$$

Where, V= voltage (Unit: volts or V), I= current (Unit: Amperes or A) and R= resistance (Unit: ohms or Ω)



Source: <https://sites.google.com/view/trainingday-1/basics-of-electronics/components?authuser=0>

5. Buzzer



Source: <https://www.instructables.com/How-to-use-a-Buzzer-Arduino-Tutorial/>

There are many ways to communicate between the user and a product. One of the best ways is audio communication using a buzzer

What is a Buzzer?

An audio signalling device like a beeper or buzzer may be electromechanical or piezoelectric or mechanical type. The main function of this is to convert the signal from audio to sound. Generally, it is powered through DC voltage and used in timers, alarm devices, printers, alarms, computers, etc. Based on the various designs, it can generate different sounds like alarm, music, bell & siren.

The **pin configuration of the buzzer** is shown below. It includes two pins namely positive and negative. The positive terminal of this is represented with the '+' symbol or a longer terminal. This terminal is powered through 6Volts whereas the negative terminal is represented with the '-' symbol or short terminal and it is connected to the GND terminal.



How to use a Buzzer?

A buzzer is an efficient component to include the features of sound in our system or project. It is an extremely small & solid two-pin device thus it can be simply utilized on breadboard or PCB. So, in most applications, this component is widely used.

There are two kinds of buzzers commonly available like simple and readymade. Once a simple type is power-driven then it will generate a beep sound continuously. A readymade type looks heavier & generates a Beep. Beep. Beep. This sound is because of the internal oscillating circuit within it.

This buzzer uses a DC power supply that ranges from 4V – 9V. To operate this, a 9V battery is used but it is suggested to utilize a regulated +5V/+6V DC supply. Generally, it is connected through a switching circuit to switch ON/OFF the buzzer at the necessary time interval.

2.2 Logic Gates

A Logic gate is a kind of the basic building block of a digital circuit having two inputs and one output. The input and output relationship are based on a certain logic. These gates are implemented using electronic switches such as diodes, transistors. But, in practice, the basic logic gates are built using CMOS technology, MOSFET (Metal Oxide Semiconductor FET), FETS. Logic gates are used in microcontrollers, microprocessors, electronic and electrical project circuits, and embedded system applications. The basic logic gates are categorized into seven types as AND, OR, XOR, NAND, NOR, XNOR, and NOT.

Logic Gates Operation



These are the important digital devices, mainly based on the Boolean function. Logic gates are used to carry out the logical operations on single or multiple binary inputs and result in one binary output. In simple words, logic gates are the electronic circuits in a digital system.

Types of Basic Logic Gates

There are various basic logic gates used to perform operations in digital systems. The common ones are given below.

- OR Gate
- AND Gate
- NOT Gate
- XOR Gate

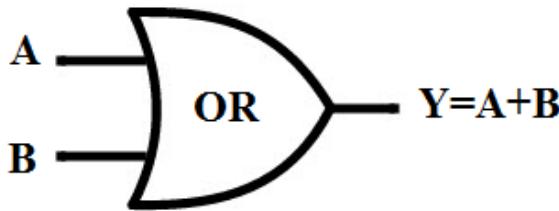
Also, these gates can be found in a combination of one or two. Therefore, we get other gates like NAND, NOR, EXOR, and EXNOR Gates.

1. OR GATE

The OR gate is a digital logic gate with 'n' i/p's and one o/p, that performs logical conjunction based on the combinations of its inputs.

The output of the OR gate is true only when one or more inputs are true. If all the i/p's of the gate are false, then only the output of the OR gate is false.

The symbol and truth table of an OR gate with two inputs is shown below.



Inputs		Output
A	B	$Y=A+B$
0	0	0
0	1	1
1	0	1
1	1	1

Boolean expression of OR gate can be given by,

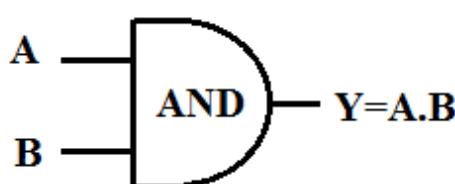
$$\underline{Y = A + B},$$

which reads as Y equals A 'OR' B.

2. AND GATE

The AND gate is a digital logic gate with 'n' i/p/s one o/p, which performs logical conjunction based on the combinations of its inputs.

The output of this gate is true only when all the inputs are true. When one or more inputs of the AND gate's i/p/s are false, then only the output of the AND gate is false. The symbol and truth table of an AND gate with two inputs is shown below.



Inputs		Output
A	B	$Y=A.B$
0	0	0
0	1	0
1	0	0
1	1	1

Boolean expression of AND gate can be given by,

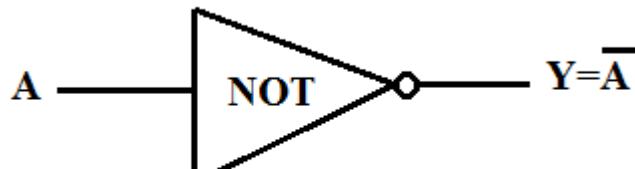
$$Y = A \cdot B$$

which reads as Y equals A 'AND' B.

3. NOT GATE

The NOT gate is a digital logic gate with one input and one output that operates an inverter operation of the input. The output of the NOT gate is the reverse of the input. When the input of the NOT gate is true then the output will be false and vice versa.

The symbol and truth table of a NOT gate with one input is shown below. By using this gate, we can implement NOR and NAND gates

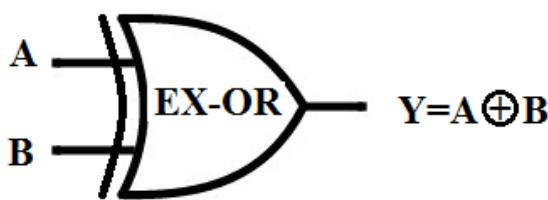


Input	Output
A	$Y = \bar{A}$
0	1
1	0

4. XOR GATE

The Exclusive-OR gate is a digital logic gate with two inputs and one output. The short form of this gate is Ex-OR. It performs based on the operation of the OR gate.

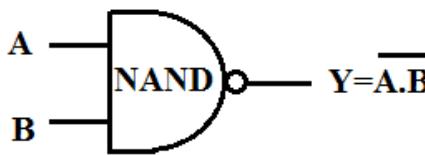
If any one of the inputs of this gate is high, then the output of the EX-OR gate will be high. The symbol and truth table of the EX-OR are shown below.



Inputs		Output
A	B	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

5. NAND GATE

The NAND gate is a digital logic gate with 'n' i/p's and one o/p, that performs the operation of the AND gate followed by the operation of the NOT gate. NAND gate is designed by combining the AND and NOT gates. If the input of the NAND gate high, then the output of the gate will be low. The symbol and truth table of the NAND gate with two inputs is shown below.



Inputs		Output
A	B	$Y = \overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

Summary

- For AND Gate – If both the inputs are high then the output is also high
- For OR Gate – If a minimum of one input is high then the output is High
- For XOR Gate – If the minimum one input is high then only the output is high
- NAND Gate – If the minimum one input is low then the output is high
- NOR Gate – If both the inputs are low then the output is high.

2.3 Electronic Signal

A Signal can be understood as "a representation that gives some information about the data present at the source from which it is produced." This is usually time varying. Hence, a signal can be a source of energy which transmits some information. This can easily be represented on a graph.

A signal can be of any type that conveys some information. This signal produced from an electronic equipment, is called as Electronic Signal or Electrical Signal. These are generally time variants.

Types of Signals

Signals can be classified either as Analog or Digital, depending upon their characteristics.

1. Analog Signal
2. Digital Signal

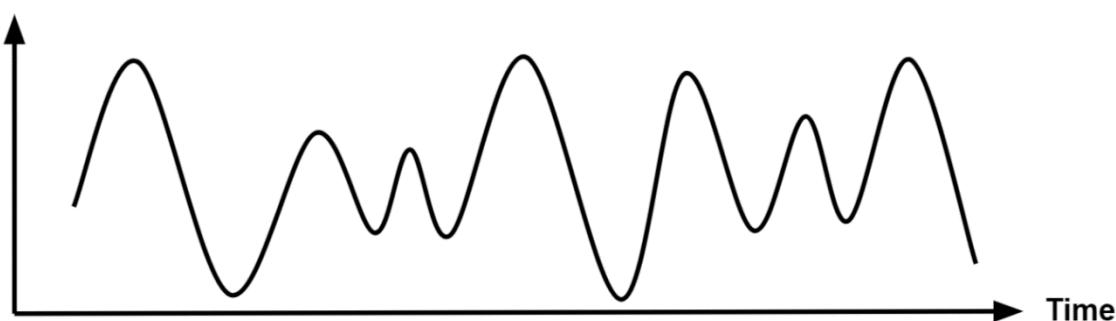
Analog Signal

An analog signal is time-varying and generally bound to a range (e.g. +12V to -12V), but there is an infinite number of values within that continuous range. An analog signal uses a given property of the medium to convey the signal's information, such as electricity moving through a wire. In an electrical signal, the voltage, current, or frequency of the signal may be varied to represent the information.

Analog signals are often calculated responses to changes in light, sound, temperature, position, pressure, or other physical phenomena.

When plotted on a voltage vs. time graph, an analog signal should produce a smooth and continuous curve. There should not be any discrete value changes

Amplitude



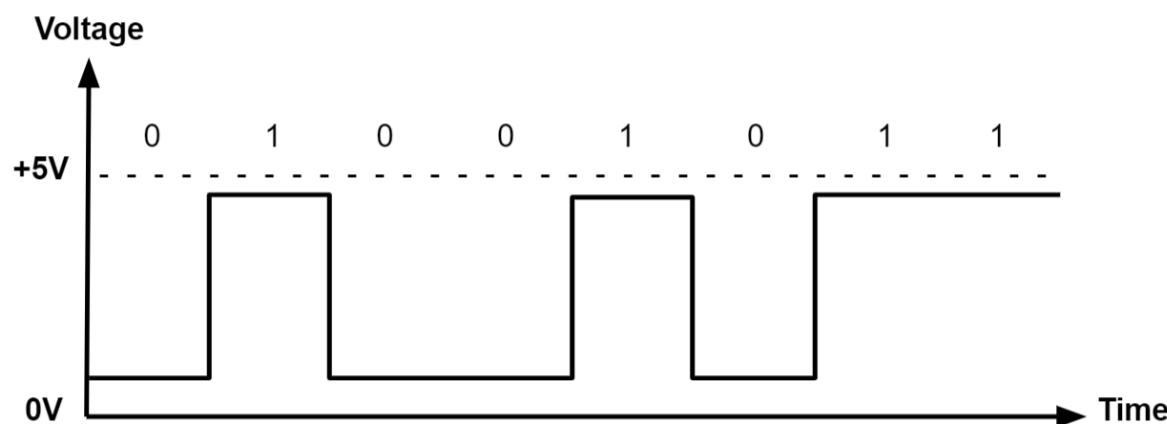
Source: <https://www.monolithicpower.com/en/analog-vs-digital-signal>

Digital Signal

A digital signal is a signal that represents data as a sequence of discrete values. A digital signal can only take on one value from a finite set of possible values at a given time. With digital signals, the physical quantity representing the information can be many things:

- Variable electric current or voltage
- Phase or polarization of an electromagnetic field
- Acoustic pressure
- The magnetization of a magnetic storage media

Digital signals are used in all digital electronics, including computing equipment and data transmission devices. When plotted on a voltage vs. time graph, digital signals are one of two values, and are usually between 0V and VCC (usually 1.8V, 3.3V, or 5V)



Source: <https://www.monolithicpower.com/en/analog-vs-digital-signal>

2.4 PWM – Pulse Width Modulation

In power electronics, pulse width modulation is a proven effective technique that is used to control semiconductor devices. Pulse width modulation or PWM is a commonly used control technique that generates analog signals from digital devices such as microcontrollers. The signal thus produced will have a train of pulses, and these pulses will be in the form of square waves.

What is Pulse Width Modulation?

Pulse width modulation reduces the average power delivered by an electrical signal by converting the signal into discrete parts. In the PWM technique, the signal's energy

is distributed through a series of pulses rather than a continuously varying (analog) signal.

Important Parameters associated with PMW signal

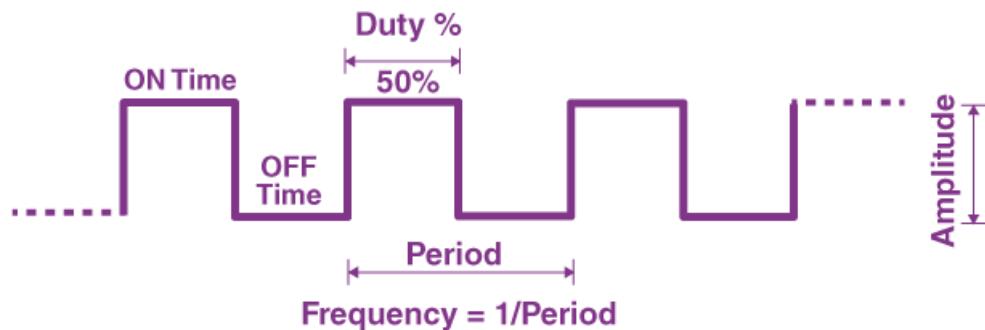
Duty Cycle of PMW

As we know, a PWM signal stays “ON” for a given time and stays “OFF” for a certain time. The percentage of time for which the signal remains “ON” is known as the duty cycle. If the signal is always “ON,” then the signal must have a 100 % duty cycle. The formula to calculate the duty cycle is given as follows:

$$\text{Duty Cycle} = \frac{\text{Turn ON time}}{\text{Turn ON time} + \text{Turn OFF time}}$$

The average value of the voltage depends on the duty cycle. As a result, the average value can be varied by controlling the width of the “ON” of a pulse.

Frequency of PMW



The frequency of PMW determines how fast a PMW completes a period. The frequency of a pulse is shown in the figure above.

The frequency of PMW can be calculated as follows:

$$\text{Frequency} = 1/\text{Time Period}$$

$$\text{Time Period} = \text{ON time} + \text{OFF time}$$

Output Voltage of PWM signal

The output voltage of the PWM signal will be the percentage of the duty cycle. For example, for a 100% duty cycle, if the operating voltage is 5 V then the output voltage will also be 5 V. If the duty cycle is 50%, then the output voltage will be 2.5v.

Applications of Pulse Width Modulation

Due to the high efficiency, low power loss, and the PWM technique's ability to precisely control the power, the technique is used in a variety of power applications. Some of the applications of PWM are as follows:

- The pulse width modulation technique is used in telecommunication for encoding purposes.

- The PWM helps in voltage regulation and therefore is used to control the speed of motors.
- The PWM technique controls the fan inside a CPU of the computer, thereby successfully dissipating the heat.
- PWM is used in Audio/Video Amplifiers.

2.5 ADC – Analog to Digital Converter

Analog-to-Digital converters (ADC) translate analog signals, real world signals like temperature, pressure, voltage, current, distance, or light intensity, into a digital representation of that signal. This digital representation can then be processed, manipulated, computed, transmitted or stored.

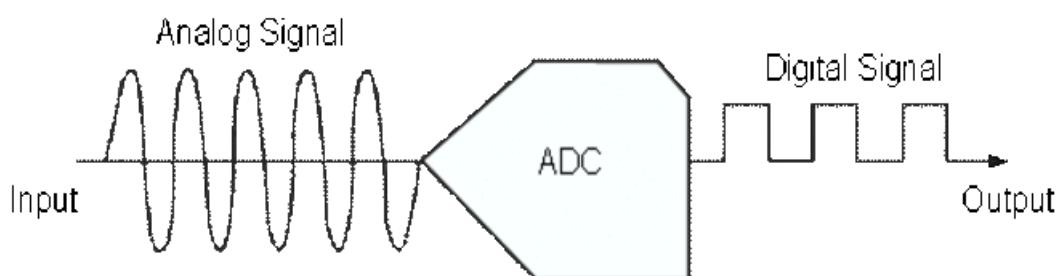


Image: Analog to Digital Conversion

Reference: <https://wiki.analog.com/university/courses/electronics/text/chapter-20>

Digital signals are represented by a sequence of discrete values where the signal is broken down into sequences that depend on the time series or sampling rate. The easiest way to explain this is through a visual! Figure shows a great example of what analog and digital signals look like.

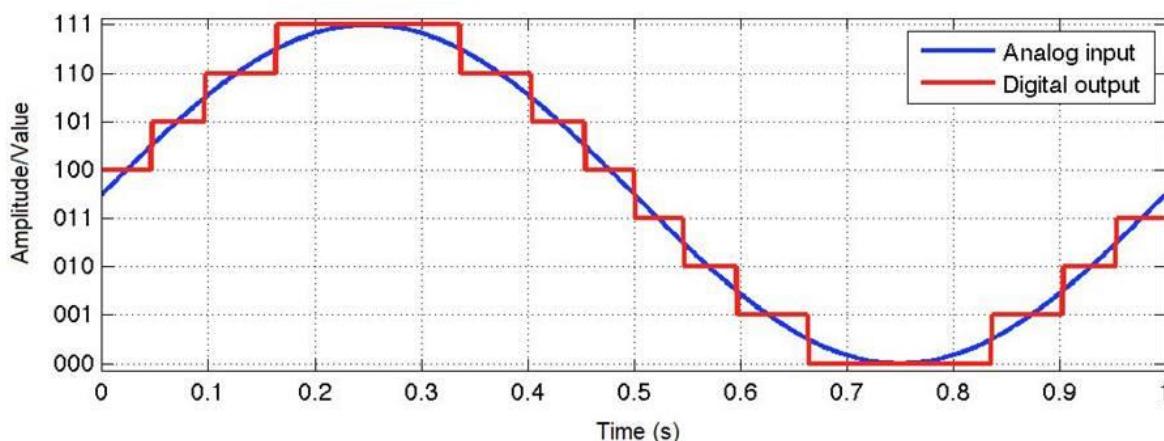


Image: continuous signal (analog) turning into a digital signal.

Reference: <https://www.arrow.com/en/research-and-events/articles/engineering-resource-basics-of-analog-to-digital-converters#:~:text=ADCs%20follow%20a%20sequence%20when,its%20sampling%20rate%20and%20resolution.>

Microcontrollers can't read values unless it's digital data. This is because microcontrollers can only see "levels" of the voltage, which depends on the resolution of the ADC and the system voltage.

ADCs follow a sequence when converting analog signals to digital. They first sample the signal, then quantify it to determine the resolution of the signal, and finally set binary values and send it to the system to read the digital signal. Two important aspects of the ADC are its sampling rate and resolution.

Applications of Analog to Digital Converter

The applications of ADC include the following.

- AC (air conditioner) includes temperature sensors to maintain the temperature within the room. So this conversion of temperature can be done from analog to digital with the help of ADC.
- It is also used in a digital oscilloscope to convert the signal from analog to digital to display.
- ADC is used to convert the analog voice signal to digital in mobile phones because mobile phones use digital voice signals but actually, the voice signal is in the form of analog. So ADC is used to convert the signal before sending the signal toward the transmitter of the cell phone.
- ADC is used in medical devices like MRI and X-Ray to convert the images from analog to digital before alteration.
- The camera in the mobile mainly used for capturing images as well as videos. These are stored in the digital device, so these are converted to digital form using ADC.
- The cassette music can also be changed into a digital like CDS & thumb drives use ADC.
- At present ADC is used in every device because almost all devices available in the market are in digital version. So these devices use ADC.

2.6 PRACTICAL: Use of Digital Multimeter to Measure various Electronics Parameter

Requirement: One DMM, LED, Wire

What is Digital Multimeter?

Multimeter are widely used by professionals in several fields including industrial maintenance and testing, research, appliance repair and electrical installation. However, a digital Multimeter or DMM is also an invaluable test instrument for home and DIY use.

Multimeter can used for measuring voltage, current and resistance and can check:

- Battery voltages
- Continuity of cables and power cords
- Home appliances and electronic devices
- Fuses

What Does a Multimeter Measure?

A basic Multimeter allows you to measure the following:

- DC voltage
- DC current
- AC voltage
- AC current (not all basic meters have this function)
- Resistance
- Continuity - indicated by a buzzer or tone

In addition, meters may have the following functions:

- Capacitance measurement
- Transistor HFE or DC current gain
- Temperature measurement with an additional probe
- Diode test
- Frequency measurement

The value measured by the instrument is indicated on an LCD display or scale.

Parts of a Meter

- **The Display.** This is usually a multidigit, 7 segment LCD display. Some laboratory instruments however have LED displays which are easier to read under certain lighting conditions.
- **Rotary Range Selector Dial.** This allows you to select the function which you will be using on the meter. On a non-autoranging meter, it also selects the range.
- **Connection Ports.** These are 4mm diameter female sockets into which 4 mm probe leads are plugged.
- **Probes.** These have a pointed tip on one end for touching against the point of measurement and a plug on the other end for insertion into a connection socket.



Image: Digital Multimeter

Reference: <https://pixabay.com/vectors/device-electric-electronics-measure-1296017/>



Image: Digital Multimeter Probe

Reference: <https://dengarden.com/home-improvement/Using-a-Multimeter>

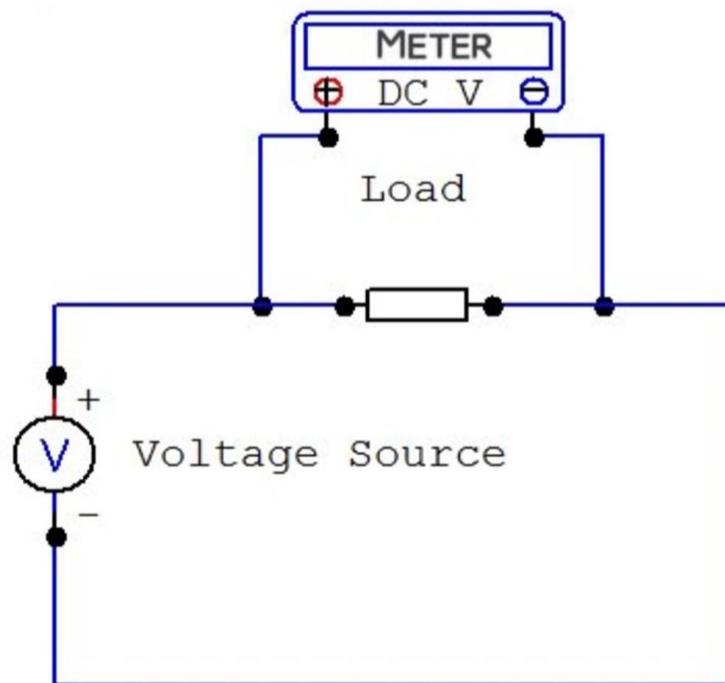
Connection Sockets on a Meter

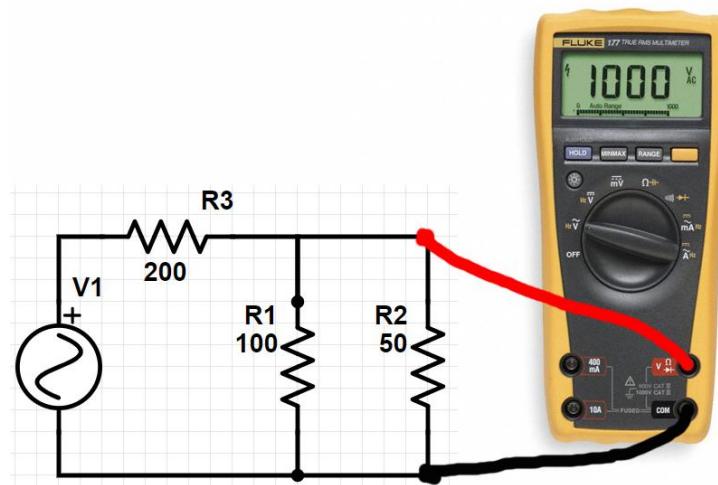
The arrangement is non-standard and depends on the brand/model of meter, so it's important to understand the function of each socket to avoid damage to the meter:

- **Com** is the common socket into which the black probe lead is plugged. This is standard on all meters.
 - **VΩmA** marked on a socket indicates that the red probe lead is plugged into it for measuring voltage, resistance or low current ("mA" means "milliamps"). If there is no mention of "mA" on this socket, there will be one or more separate sockets for connecting the probe lead to measure current. These sockets will be marked "A" or "mA" with the max current range (e.g. 10A for high current readings and 400 mA for lower current readings).



Measuring Voltage - Meter in Parallel with Load or Voltage Source

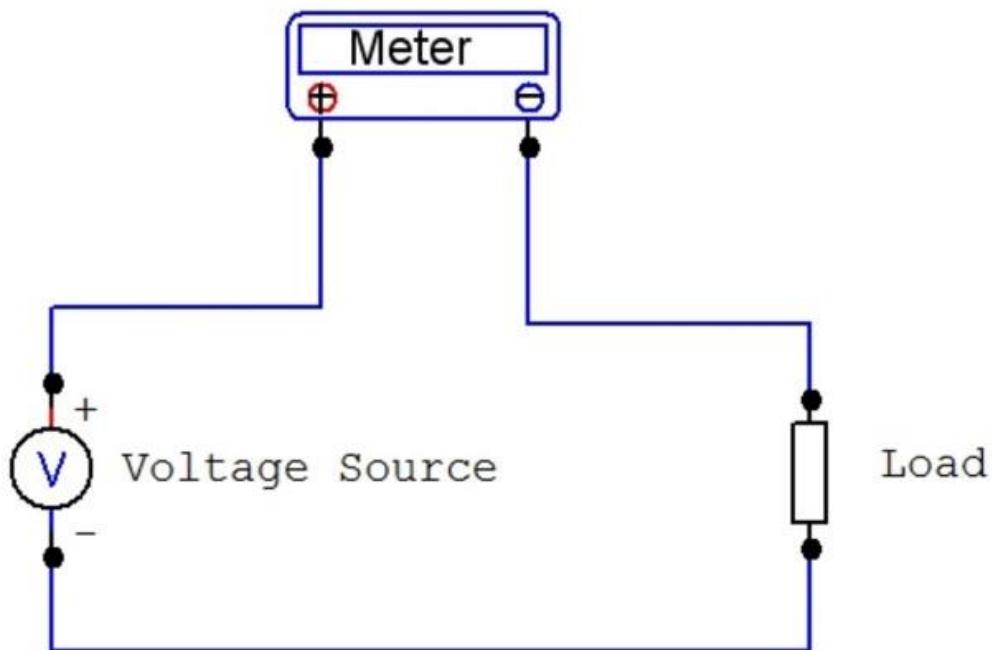


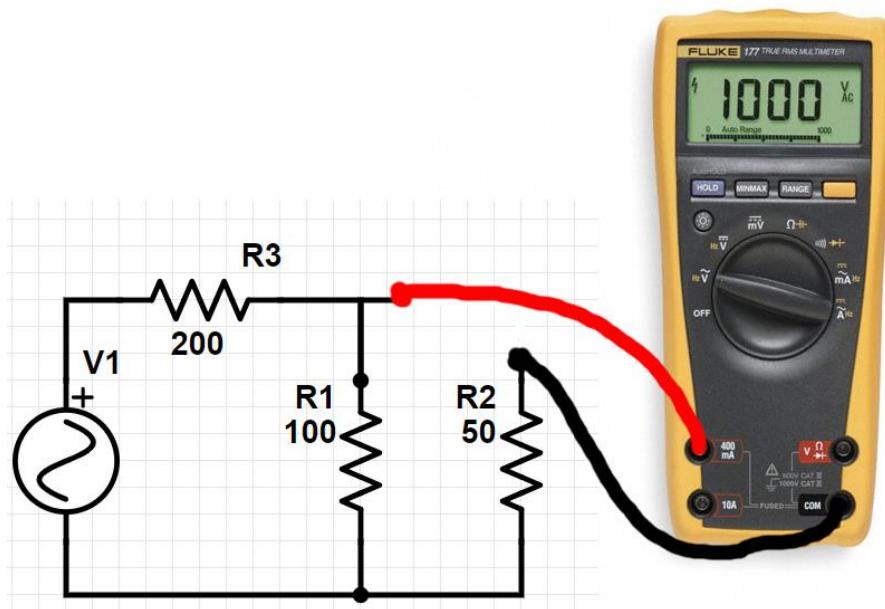


Reference: <https://dengarden.com/home-improvement/Using-a-Multimeter>

- First setup one simple circuit to measure Voltage across the Load.
- Once circuit setup, take Multimeter and setup knob on DC V and connect lead of the probs across the LOAD.
- Now you can measure Voltage across the Load, Voltage value appear on Multimeter LCD screen.

Measuring Current - Meter in Series





Series Connection for Measuring Current

Reference: <https://practicalee.com/multimeter/>

- Make connection as shown in above figure to measure the current passing through the circuit
- Make sure Red Wire connected to mA Probe of Multimeter
- Read the output result in LCD screen of Multimeter

What Do the Symbols on the Range Dial Mean?

Symbols and abbreviations used on a meter

V	volts
A	amps
mA	milliAmps
~	AC
—	DC
Ω	Ohms (Resistance)
— —	Capacitance
Hz	Hertz (frequency)
→+—	Diode test
)))	Continuity test buzzer



Image: Symbols used on an auto ranging DMM

Reference: <https://dengarden.com/home-improvement/Using-a-Multimeter>

Unit 3: Sensors and Actuators

Learning Outcomes:

- Understand the Concept of Sensors and Actuators
- Able to identify various sensors and actuators
- Understand the use and applications of Various Sensors and Actuators

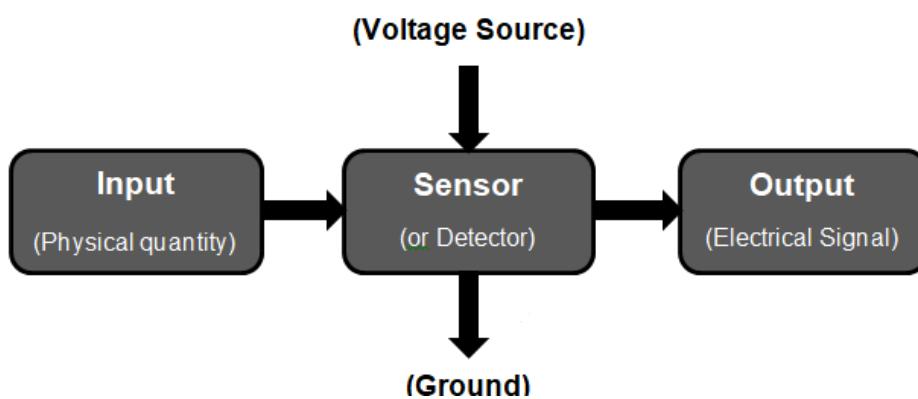
Let's Start Explore Sensors

3.1 Sensors

The era of automation has begun already. Most of the things that we use now can be automated. To design automated devices first we need to know about the sensors, these are the modules/devices which are helpful in making things done without human intervention. Even the mobiles or smartphones which we daily use will have some sensors like hall sensor, proximity sensor, accelerometer, touch screen, microphone etc. These sensor acts as eyes, ears, nose of any electrical equipment which senses the parameters in outside world and give readings to devices or Microcontroller.

What is Sensor?

The sensor can be defined as a device which can be used to sense/detect the physical quantity like force, pressure, strain, light etc and then convert it into desired output like the electrical signal to measure the applied physical quantity. In few cases, a sensor alone may not be sufficient to analyze the obtained signal. In those cases, a **signal conditioning unit** is used in order to maintain sensor's output voltage levels in the desired range with respect to the end device that we use.



Block Diagram of Sensor

In **signal conditioning unit**, the output of the sensor may be amplified, filtered or modified to the desired output voltage. For example, if we consider a microphone, it detects the audio signal and converts to the output voltage (is in terms of millivolts) which becomes hard to drive an output circuit. So, a signal conditioning unit (an

amplifier) is used to increase the signal strength. But the signal conditioning may not be necessary for all the sensors like photodiode, LDR etc.

Most of the sensors can't work independently. So, sufficient input voltage should be applied to it. Various sensors have different operating ranges which should be considered while working with it else the sensor may get damaged permanently.

Characteristics of Sensors

A good sensor should have the following characteristics

1. High Sensitivity: Sensitivity indicates how much the output of the device changes with unit change in input (quantity to be measured). For example, the voltage of a temperature sensor changes by 1mV for every 1°C change in temperature than the sensitivity of the sensor is said to be 1mV/°C.
2. Linearity: The output should change linearly with the input.
3. High Resolution: Resolution is the smallest change in the input that the device can detect.
4. Less Noise and Disturbance.
5. Less power consumption.

Types sensors

Sensors are split up into four main categories. Such as,

- Analog Sensor
- Digital Sensor
- Active Sensor
- Passive Sensor

Analog Sensors

The sensor that produces continuous signal with respect to time with analog output is called as Analog sensors. The analog output generated is proportional to the measured or the input given to the system. Generally, analog voltage in the range of 0 to 5 V or current is produced as the output. The various physical parameters like temperature, stress, pressure, displacement, etc. are examples for continuous signals. Examples: accelerometers, speed sensors, pressure sensors, light sensors, temperature sensors.

Digital Sensors

When data is converted and transmitted digitally, it is called as Digital sensors. Digital sensors are the one, which produces discrete output signals. Discrete signals will be non-continuous with time and it can be represented in "bits" for serial transmission and in "bytes" for parallel transmission. The measuring quantity will be represented in digital format. Digital output can be in form of Logic 1 or logic 0 (ON or OFF). A digital sensor consists of sensor, cable and a transmitter. The measured signal is converted into a digital signal inside sensor itself without any external component. Cable is used for long distance transmission.

Active Sensor

Based on power requirement sensors can be classified as active and passive

Active sensors are those which do not require external power source for their functioning. They generate power within themselves to operate and hence called as self-generating type. The energy for functioning is derived from the quantity being measured. For example, piezoelectric crystal generates electrical output (charge) when subjected to acceleration.

Passive sensors

Passive sensors require external power source for their functioning. Most of the resistive, inductive and capacitive sensors are passive (just as resistors, inductors and capacitors are called passive devices).

Applications of Sensors

- Automobile
- Manufacturing
- Agriculture
- Aviation
- Medical & Healthcare sector etc.

Let's discuss different types of sensors

1. Rotary Angle Sensor or Potentiometer

The rotary angle sensor produces analog output between 0 and Vcc (5V DC with Seeeduino) on its D1 connector. The D2 connector is not used. The angular range is 300 degrees with a linear change in value. The resistance value is 10k ohms, perfect for Arduino use. This may also be known as a "potentiometer".



2. Sound Sensor

- The sound sensor is one type of module used to notice the sound. Generally, this module is used to detect the intensity of sound. The applications of this module mainly include switch, security, as well as monitoring. The accuracy of this sensor can be changed for the ease of usage.

- This sensor employs a microphone to provide input to buffer, peak detector and an amplifier. This sensor notices a sound, & processes an o/p voltage signal to a microcontroller. After that, it executes required processing.
- This sensor is capable to determine noise levels within DB's or decibels at 3 kHz 6 kHz frequencies approximately wherever the human ear is sensitive. In smartphones, there is an android application namely decibel meter used to measure the sound level.

Sound Sensor Pin Configuration

This sensor includes three pins which include the following.



sound-sensor-module

- Pin1 (VCC): 3.3V DC to 5V DC
- Pin2 (GND): This is a ground pin
- Pin3 (DO): This is an output pin

Working Principle

The working principle of this sensor is related to human ears. Because human eye includes a diaphragm and the main function of this diaphragm is, it uses the vibrations and changes into signals. Whereas in this sensor, it uses a microphone and the main function of this is, it uses the vibrations and changes into current otherwise voltage. Generally, it includes a diaphragm which is designed with magnets that are twisted with metal wire. When sound signals hit the diaphragm, then magnets within the sensor vibrates & simultaneously current can be stimulated from the coils.

Features

The features of the sound sensor include the following

- These sensors are very simple to use
- It gives analog o/p signal
- Simply incorporates using logic modules on the input area

Application

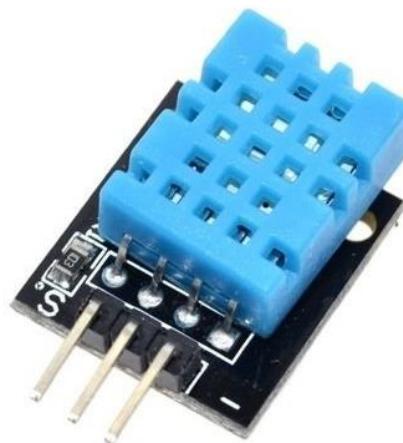
- Security system for Office or Home
- Spy Circuit
- Home Automation
- Robotics
- Smart Phones

- Ambient sound recognition
- Audio amplifier
- Sound level recognition (not capable to obtain precise dB value)

3. Temperature and humidity sensor

Temperature and humidity sensor (or rh temp sensor) is devices that can convert temperature and humidity into electrical signals that can easily measure temperature and humidity. Temperature humidity transmitters on the market generally measure the amount of temperature and relative humidity in the air, and convert it into electrical signals or other signal forms according to certain rules and output the device to the instrument or software to meet the environmental monitoring needs of users.

- DHT11 is a Humidity and Temperature Sensor, which generates calibrated digital output. DHT11 can be interface with any microcontroller like Arduino, Raspberry Pi, etc. and get instantaneous results. DHT11 is a low-cost humidity and temperature sensor which provides high reliability and long-term stability.
- It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and outputs a digital signal on the data pin (no analog input pins needed). It's very simple to use, and libraries and sample codes are available for Arduino and Raspberry Pi.
- This module makes it easy to connect the DHT11 sensor to an Arduino or microcontroller as includes the pull up resistor required to use the sensor. Only three connections are required to be made to use the sensor - Vcc, Gnd and Output.



- It has high reliability and excellent long-term stability, thanks to the exclusive digital signal acquisition technique and temperature & humidity sensing technology.

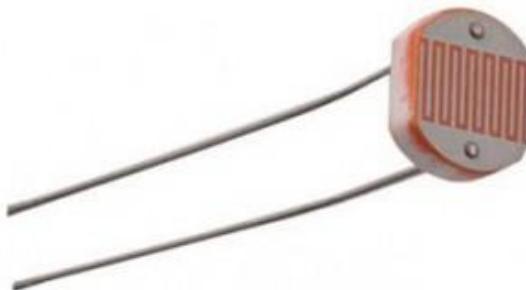
4. Light Sensor

The light sensor is used to discover the light & generates a voltage difference. The light sensors used in robots are two types photovoltaic cells & photoresistors. Photovoltaic cells are used to change the solar radiation energy to electrical and these sensors are used in solar robot manufacturing. Photoresistors are used to alter their

resistance by modifying light intensities. When light is more on it then resistance will be less. These light sensors are not expensive, so used easily in robots.

LDR is one the sensor belongs to light sensor family. An LDR is a component that has a (variable) resistance that changes with the light intensity that falls upon it. This allows them to be used in light sensing circuits.

As its name implies, the **Light Dependent Resistor** (LDR) is made from a piece of exposed semiconductor material such as cadmium sulphide that changes its electrical resistance from several thousand Ohms in the dark to only a few hundred Ohms when light falls upon it by creating hole-electron pairs in the material.



Light Sensor

The most common type of LDR has a resistance that falls with an increase in the light intensity falling upon the device (as shown in the image above). The resistance of an LDR may typically have the following resistances:

Daylight = 5000Ω

Dark = 20000000Ω

You can therefore see that there is a large variation between these figures. If you plotted this variation on a graph, you would get something similar to that shown by the graph shown above.

Applications of LDRs

There are many applications for Light Dependent Resistors. These include:

- *Lighting switch*

The most obvious application for an LDR is to automatically turn on a light at a certain light level. An example of this could be a street light or a garden light.

- *Camera shutter control*

LDRs can be used to control the shutter speed on a camera. The LDR would be used to measure the light intensity which then adjusts the camera shutter speed to the appropriate level.

5. Ultrasonic Sensor

An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal. Ultrasonic waves travel faster than the speed of audible sound (i.e., the sound that humans can hear). Ultrasonic sensors have two main components: the transmitter (which emits the sound using piezoelectric crystals) and the receiver (which encounters the sound after it has travelled to and from the target).

Ultrasonic sensors work by emitting sound waves at a frequency which is too high for humans to hear.



Image: Ultrasonic Sensor

Reference: <https://robu.in/ultrasonic-sensor-working-principle/>

An above image shows the HC-SR-04 ultrasonic sensor which has a transmitter, receiver. The pin configuration is,

- VCC - +5 V supply
- TRIG – Trigger input of the sensor. Microcontroller applies 10 us trigger pulse to the HC-SR04 ultrasonic module.
- ECHO–Echo output of the sensor. Microcontroller reads/monitors this pin to detect the obstacle or to find the distance.
- GND – Ground

Sound is a mechanical wave travelling through the mediums, which may be a solid, or liquid or gas. Sound waves can travel through the mediums with specific velocity depends on the medium of propagation. The sound waves which are having high frequency reflect from boundaries and produce distinctive echo patterns.

Features of an Ultrasonic Sensor

- Supply voltage: 5V (DC).
- Supply current: 15mA.
- Modulation frequency: 40Hz.

- Output: 0 – 5V (Output high when obstacle detected in range).
- Beam Angle: Max 15 degrees.
- Distance: 2 cm – 400 cm.
- Accuracy: 0.3cm.
- Communication: Positive TTL pulse.

Ultrasonic Sensor Working Principle

Ultrasonic sensors emit short, high-frequency sound pulses at regular intervals. These propagate in the air at the velocity of sound. If they strike an object, then they reflected back as an echo signal to the sensor, which itself computes the distance to the target based on the time-span between emitting the signal and receiving the echo.

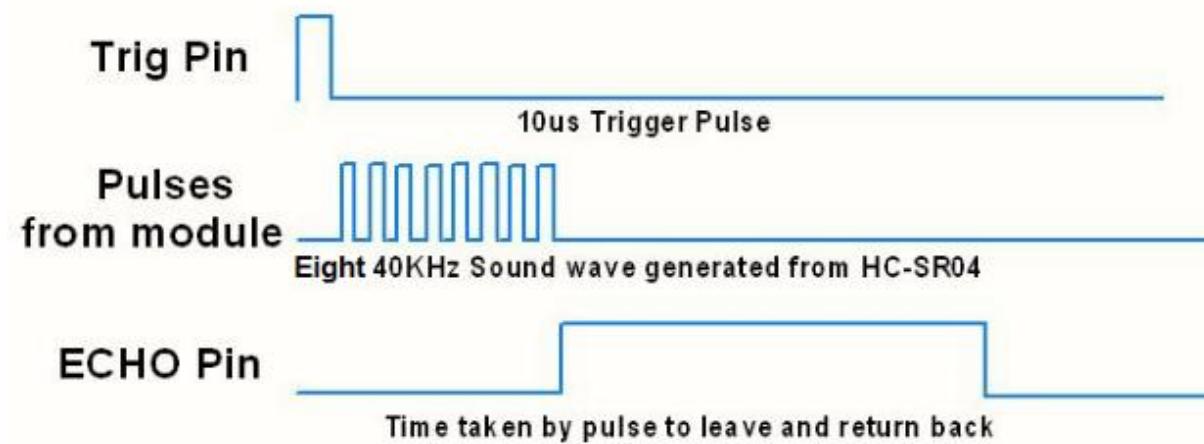


Reference: <https://robu.in/ultrasonic-sensor-working-principle/>

Ultrasonic **sensors** are excellent at suppressing background interference. Virtually all materials which reflect sound can be detected, regardless of their colour. Even transparent materials or thin foils represent no problem for an ultrasonic sensor.

microsonic ultrasonic sensors are suitable for target distances from 20 mm to 10 m and as they measure the time of flight, they can ascertain a measurement with pinpoint accuracy. Some of our sensors can even resolve the signal to an accuracy of 0.025 mm. Ultrasonic sensors can see through dust-laden air and ink mists. Even thin deposits on the sensor membrane do not impair its function.

Timing Diagram of Ultrasonic Sensor



Reference: <https://robu.in/ultrasonic-sensor-working-principle/>

1. First, need to transmit trigger pulse of at least 10 us to the HC-SR04 Trig Pin.
2. Then the HC-SR04 automatically sends Eight 40 kHz sound wave and wait for rising edge output at Echo pin.
3. When the rising edge capture occurs at Echo pin, start the Timer and wait for a falling edge on Echo pin.
4. As soon as the falling edge captures at the Echo pin, read the count of the Timer. This time count is the time required by the sensor to detect an object and return back from an object.

How to calculate Distance?

If you need to measure the specific distance from your sensor, this can be calculated based on this formula:

We know that, **Distance = Speed * Time**. The speed of sound waves is 343 m/s. So,

$$\text{Total Distance} = (343 * \text{Time of hight (Echo) pulse})/2$$

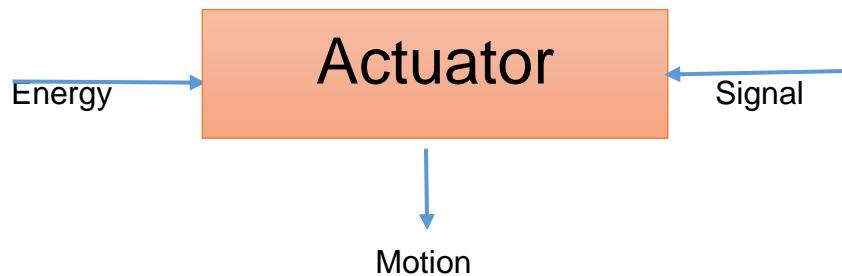
Total distance is divided by 2 because the signal travels from HC-SR04 to object and returns to the module HC-SR-04.

3.2 Introduction to Actuators

An actuator is a device that produces a motion by converting energy and signals going into the system. The motion it produces can be either rotary or linear.

An actuator is a device that produces a motion by converting energy and signals going into the system. The motion it produces can be either rotary or linear. Linear actuators, as the name implies, produce linear motion. This means that linear actuators can move forward or backwards on a set linear plane – a set distance they can travel in either direction before they must stop. Rotary actuators on the other hand produce rotary motion, meaning that the actuator revolves on a circular plane. Unlike the linear

actuator, the rotary actuator is not limited by a set path, which means it can keep rotating in the same direction for as long as necessary.



Linear or rotary actuators are available in various forms depending on the power-supply source. The actuator could be electrical, pneumatic or hydraulic.

Types of Actuators:

1. Hydraulic Actuators

A hydraulic actuator uses hydraulic power to perform a mechanical operation. They are actuated by a cylinder or fluid motor. The mechanical motion is converted to rotary, linear, or oscillatory motion, according to the need of the IoT device. Construction equipment uses hydraulic actuators because hydraulic actuators can generate a large amount of force.

Advantages :

- Hydraulic actuators can produce a large magnitude of force and high speed.
- Used in welding, clamping, etc.
- Used for lowering or raising the vehicles in car transport carriers.

2.Pneumatic Actuators –

A pneumatic actuator uses energy formed by vacuum or compressed air at high pressure to convert into either linear or rotary motion. Example- Used in robotics, use sensors that work like human fingers by using compressed air.

Advantages :

- They are a low-cost option and are used at extreme temperatures where using air is a safer option than chemicals.
- They need low maintenance, are durable, and have a long operational life.
- It is very quick in starting and stopping the motion.

3.Electrical Actuators

An electric actuator uses electrical energy, is usually actuated by a motor that converts electrical energy into mechanical torque. An example of an electric actuator is a solenoid based electric bell.

Advantages :

- It has many applications in various industries as it can automate industrial valves.
- It produces less noise and is safe to use since there are no fluid leakages.

- It can be re-programmed and it provides the highest control precision positioning.

1. LCD

In LCD 16x2, the term LCD stands for Liquid Crystal Display that uses a plane panel display technology, used in screens of computer monitors & TVs, smartphones, tablets, mobile devices, etc. Both the displays like LCD & CRTs look the same but their operation is different. Instead of electrons diffraction at a glass display, a liquid crystal display has a backlight that provides light to each pixel that is arranged in a rectangular network.

Every pixel includes a blue, red, green sub-pixel that can be switched ON/OFF. Once all these pixels are deactivated, then it will appear black and when all the sub-pixels are activated then it will appear white. By changing the levels of each light, different color combinations are achievable. This article discusses an overview of LCD 16X2 & its working with applications.

What is LCD 16X2?

An electronic device that is used to display data and the message is known as LCD 16x2. As the name suggests, it includes 16 Columns & 2 Rows so it can display 32 characters ($16 \times 2 = 32$) in total & every character will be made with 5×8 (40) Pixel Dots. So the total pixels within this LCD can be calculated as 32×40 otherwise 1280 pixels.



Reference: <https://www.watelectronics.com/lcd-16x2/>

16 X2 displays mostly depend on multi-segment LEDs. There are different types of displays available in the market with different combinations such as 8x2, 8x1, 16x1, and 10x2, however, the LCD 16x2 is broadly used in devices, DIY circuits, electronic projects due to less cost, programmable friendly & simple to access.

Specifications of LCD 16X2

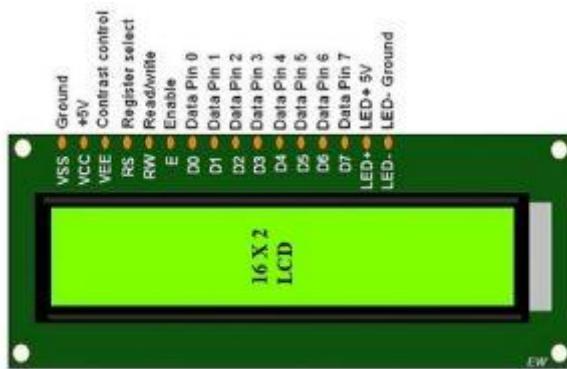
The **specifications of LCD 16X2** are discussed below.

- The operating voltage of this display ranges from 4.7V to 5.3V
- The display bezel is 72 x 25mm

- The operating current is 1mA without a backlight
- PCB size of the module is 80L x 36W x 10H mm
- HD47780 controller
- LED color for backlight is green or blue
- Number of columns – 16
- Number of rows – 2
- Number of LCD pins – 16
- Characters – 32
- It works in 4-bit and 8-bit modes
- Pixel box of each character is 5x8 pixel
- Font size of character is 0.125Width x 0.200height

LCD 16X2 Pin Configuration

The **pin configuration of LCD 16 X 2** is discussed below so that LCD 16x2 connection can be done easily with external devices.



16X2 LCD Pin Diagram

- Pin1 (Ground): This pin connects the ground terminal.
- Pin2 (+5 Volt): This pin provides a +5V supply to the LCD
- Pin3 (VE): This pin selects the contrast of the LCD.
- Pin4 (Register Select): This pin is used to connect a data pin of an MCU & gets either 1 or 0. Here, data mode = 0 and command mode =1.
- Pin5 (Read & Write): This pin is used to read/write data.
- Pin6 (Enable): This enables the pin must be high to perform the Read/Write procedure. This pin is connected to the data pin of the microcontroller to be held high constantly.
- Pin7 (Data Pin): The data pins are from 0-7 which are connected through the microcontroller for data transmission. The LCD module can also work on the 4-bit mode through working on pins 1, 2, 3 & other pins are free.
- Pin8 – Data Pin 1
- Pin9 – Data Pin 2
- Pin10 – Data Pin 3
- Pin11 – Data Pin 4
- Pin12 – Data Pin 5
- Pin13 – Data Pin 6
- Pin14 – Data Pin 7
- Pin15 (LED Positive): This is a +Ve terminal of the backlight **LED** of the display & it is connected to +5V to activate the LED backlight.

- Pin16 (LED Negative): This is a -Ve terminal of a backlight LED of the display & it is connected to the GND terminal to activate the LED backlight.

2. Stepper Motor

A **stepper motor** is a brushless, synchronous electric motor that converts digital pulses into mechanical shaft rotation. Every revolution of the stepper motor is divided into a discrete number of steps, in many cases 200 steps, and the motor must be sent a separate pulse for each step. The stepper motor can only take one step at a time and each step is the same size. Since each pulse causes the motor to rotate a precise angle, typically 1.8° , the motor's position can be controlled without any feedback mechanism. As the digital pulses increase in frequency, the step movement changes into continuous rotation, with the speed of rotation directly proportional to the frequency of the pulses. Step motors are used every day in both industrial and commercial applications because of their low cost, high reliability, high torque at low speeds and a simple, rugged construction that operates in almost any environment.

Stepper Motor Application

Stepper motors are diverse in their uses, but some of the most common include:

- 3D printing equipment
- Textile machines
- Printing presses
- Gaming machines
- Medical imaging machinery
- Small robotics
- CNC milling machines
- Welding equipment

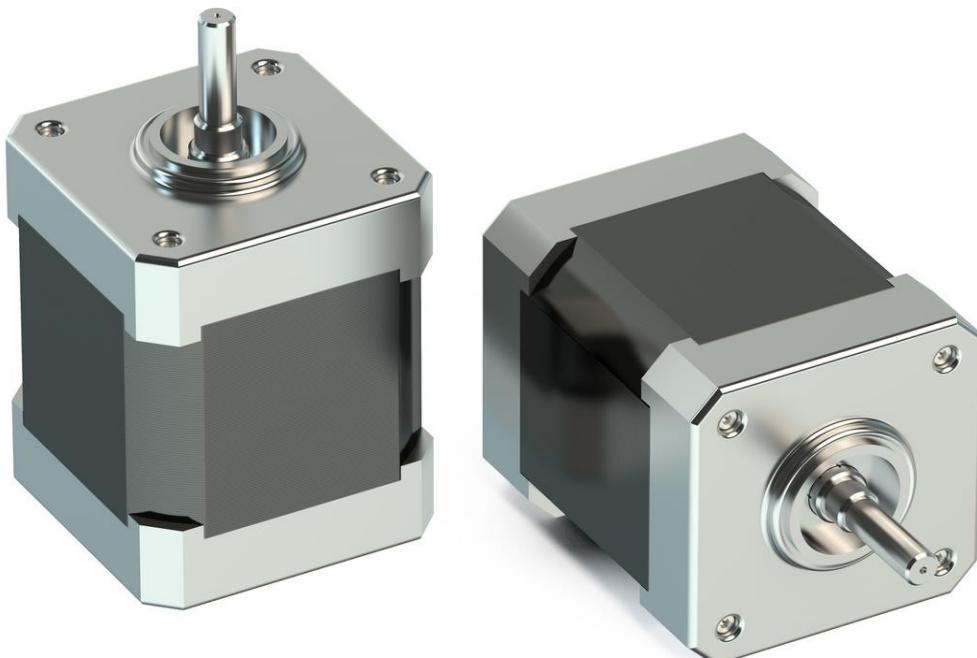


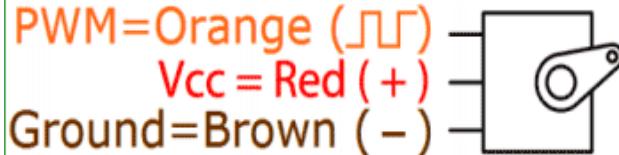
Image: Stepper Motor

Reference: <https://www.automate.org/blogs/what-kinds-of-applications-are-best-for-stepper-motors>

3. What is a Servo Motor?

A **servo motor** is a type of motor that can rotate with great precision. Normally this type of motor consists of a control circuit that provides feedback on the current position of the motor shaft, this feedback allows the servo motors to rotate with great precision. If you want to rotate an object at some specific angles or distance, then you use a servo motor. It is just made up of a simple motor which runs through a **servo mechanism**. If motor is powered by a DC power supply then it is called DC servo motor, and if it is AC-powered motor then it is called AC servo motor. For this tutorial, we will be discussing only about the **DC servo motor working**. Apart from these major classifications, there are many other types of servo motors based on the type of gear arrangement and operating characteristics. A servo motor usually comes with a gear arrangement that allows us to get a very high torque servo motor in small and lightweight packages. Due to these features, they are being used in many applications like toy car, RC helicopters and planes, Robotics, etc.

Servo motors are rated in kg/cm (kilogram per centimeter) most hobby servo motors are rated at 3kg/cm or 6kg/cm or 12kg/cm. This kg/cm tells you how much weight your servo motor can lift at a particular distance. For example: A 6kg/cm Servo motor should be able to lift 6kg if the load is suspended 1cm away from the motors shaft, the greater the distance the lesser the weight carrying capacity. The position of a servo motor is decided by electrical pulse and its circuitry is placed beside the motor.



Reference [https://circuitdigest.com/article/servo-motor-working-and-basics#:~:text=Servo%20motor%20works%20on%20PWM,\(potentiometer\)%20and%20some%20gears.](https://circuitdigest.com/article/servo-motor-working-and-basics#:~:text=Servo%20motor%20works%20on%20PWM,(potentiometer)%20and%20some%20gears.)

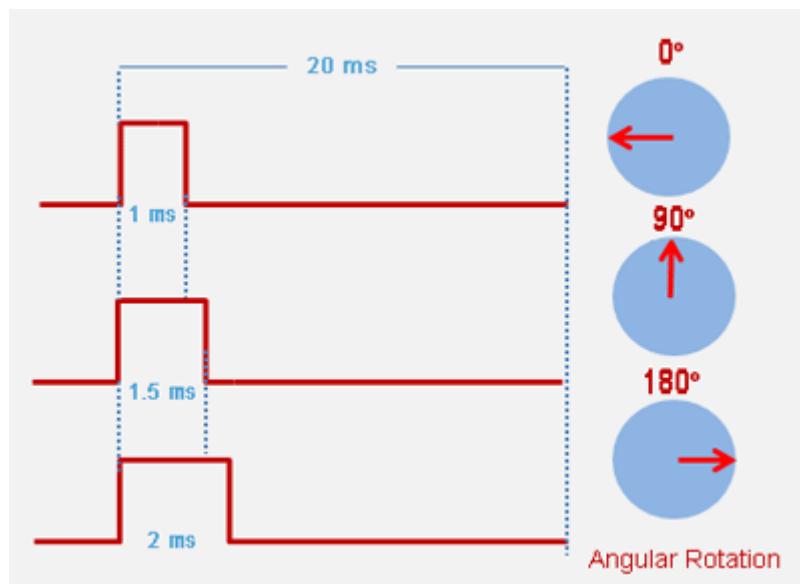
Controlling Servo Motor:

All motors have three wires coming out of them. Out of which two will be used for Supply (positive and negative) and one will be used for the signal that is to be sent from the MCU.

Servo motor is controlled by PWM (Pulse with Modulation) which is provided by the control wires. There is a minimum pulse, a maximum pulse and a repetition rate. Servo

motor can turn 90 degree from either direction from its neutral position. The servo motor expects to see a pulse every 20 milliseconds (ms) and the length of the pulse will determine how far the motor turns. For example, a 1.5ms pulse will make the motor turn to the 90° position, such as if pulse is shorter than 1.5ms shaft moves to 0° and if it is longer than 1.5ms than it will turn the servo to 180°.

Servo motor works on **PWM (Pulse width modulation)** principle, means its angle of rotation is controlled by the duration of applied pulse to its Control PIN. Basically servo motor is made up of **DC motor which is controlled by a variable resistor (potentiometer) and some gears**. High speed force of DC motor is converted into torque by Gears. We know that $WORK = FORCE \times DISTANCE$, in DC motor Force is less and distance (speed) is high and in Servo, force is High and distance is less. The potentiometer is connected to the output shaft of the Servo, to calculate the angle and stop the DC motor on the required angle.



Servo motor can be rotated from 0 to 180 degrees, but it can go up to 210 degrees, depending on the manufacturing. This degree of rotation can be controlled by applying the **Electrical Pulse** of proper width, to its Control pin. Servo checks the pulse in every 20 milliseconds. The pulse of 1 ms (1 millisecond) width can rotate the servo to 0 degrees, 1.5ms can rotate to 90 degrees (neutral position) and 2 ms pulse can rotate it to 180 degree.

All servo motors work directly with your +5V supply rails but we have to be careful about the amount of current the motor would consume if you are planning to use more than two servo motors a proper servo shield should be designed.

Exercise:

Explore Various types of Actuators and Interface with Raspberry Pi.

Unit 4: Networking for IOT

Learning Outcomes:

- Understand different types of IoT Protocols
- Able to configure Device for IoT application
- Understand the Concept of Different Networking Devices

4.1 Networking devices

What are network devices?

Network devices, or networking hardware, are physical devices that are required for communication and interaction between hardware on a computer network.

Types of network devices

Here is the common network device list:

- Hub
- Switch
- Router
- Bridge
- Gateway
- Modem
- Repeater
- Access Point

1. *Repeater –*

A repeater operates at the physical layer. Its job is to regenerate the signal over the same network before the signal becomes too weak or corrupted so as to extend the length to which the signal can be transmitted over the same network. An important point to be noted about repeaters is that they do not amplify the signal. When the signal becomes weak, they copy the signal bit by bit and regenerate it at the original strength. It is a 2-port device.

2. *Hub*

A hub is basically a multiport repeater. A hub connects multiple wires coming from different branches, for example, the connector in star topology which connects different stations. Hubs cannot filter data, so data packets are sent to all connected devices. In other words, the collision domain of all hosts connected through Hub remains one. Also, they do not have the intelligence to

find out the best path for data packets which leads to inefficiencies and wastage.

Types of Hubs

Active Hub: - These are the hubs that have their own power supply and can clean, boost, and relay the signal along with the network. It serves both as a repeater as well as a wiring center. These are used to extend the maximum distance between nodes.

Passive Hub: - These are the hubs that collect wiring from nodes and power supply from the active hub. These hubs relay signals onto the network without cleaning and boosting them and can't be used to extend the distance between nodes.

Intelligent Hub: - It works like active hubs and includes remote management capabilities. They also provide flexible data rates to network devices. It also enables an administrator to monitor the traffic passing through the hub and to configure each port in the hub.

3. Bridge

A bridge operates at the data link layer. A bridge is a repeater, with add on the functionality of filtering content by reading the MAC addresses of source and destination. It is also used for interconnecting two LANs working on the same protocol. It has a single input and single output port, thus making it a 2 port device.

Types of Bridges

Transparent Bridges: - These are the bridge in which the stations are completely unaware of the bridge's existence i.e. whether or not a bridge is added or deleted from the network, reconfiguration of the stations is unnecessary. These bridges make use of two processes i.e. bridge forwarding and bridge learning.

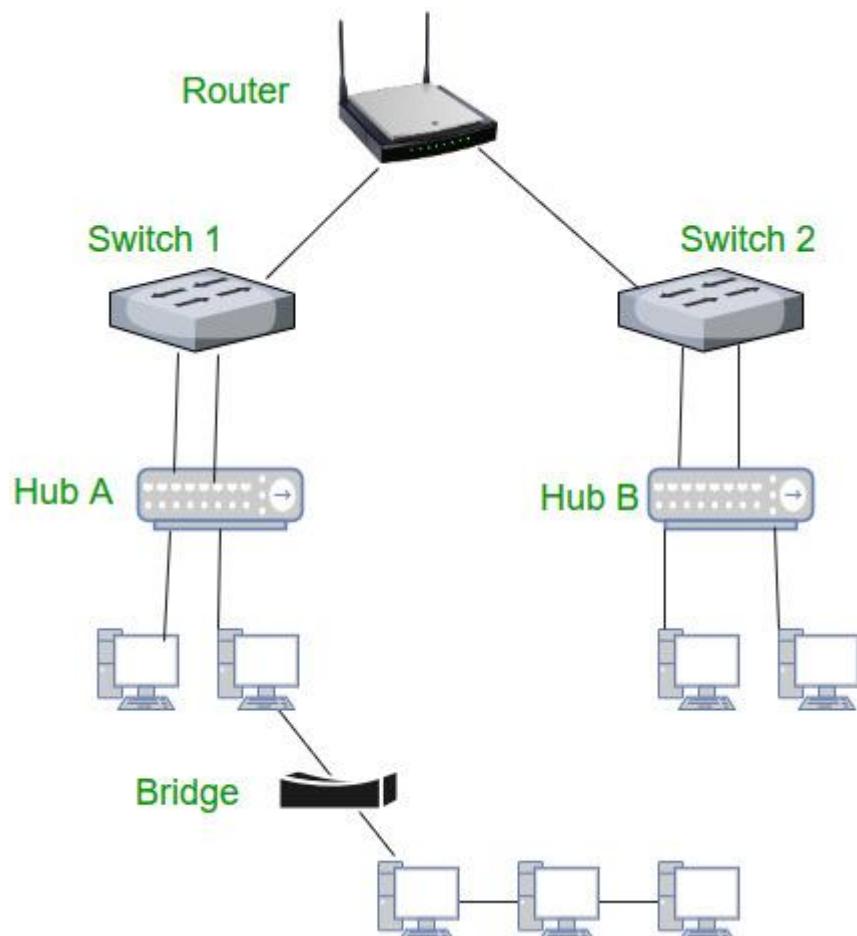
Source Routing Bridges: - In these bridges, routing operation is performed by the source station and the frame specifies which route to follow. The host can discover the frame by sending a special frame called the discovery frame, which spreads through the entire network using all possible paths to the destination.

4. Switch

A switch is a multiport bridge with a buffer and a design that can boost its efficiency (a large number of ports imply less traffic) and performance. A switch is a data link layer device. The switch can perform error checking before forwarding data, which makes it very efficient as it does not forward packets that have errors and forward good packets selectively to the correct port only. In other words, the switch divides the collision domain of hosts, but broadcast domain remains the same.

5. Routers

A router is a device like a switch that routes data packets based on their IP addresses. The router is mainly a Network Layer device. Routers normally connect LANs and WANs together and have a dynamically updating routing table based on which they make decisions on routing the data packets. Router divide broadcast domains of hosts connected through it.



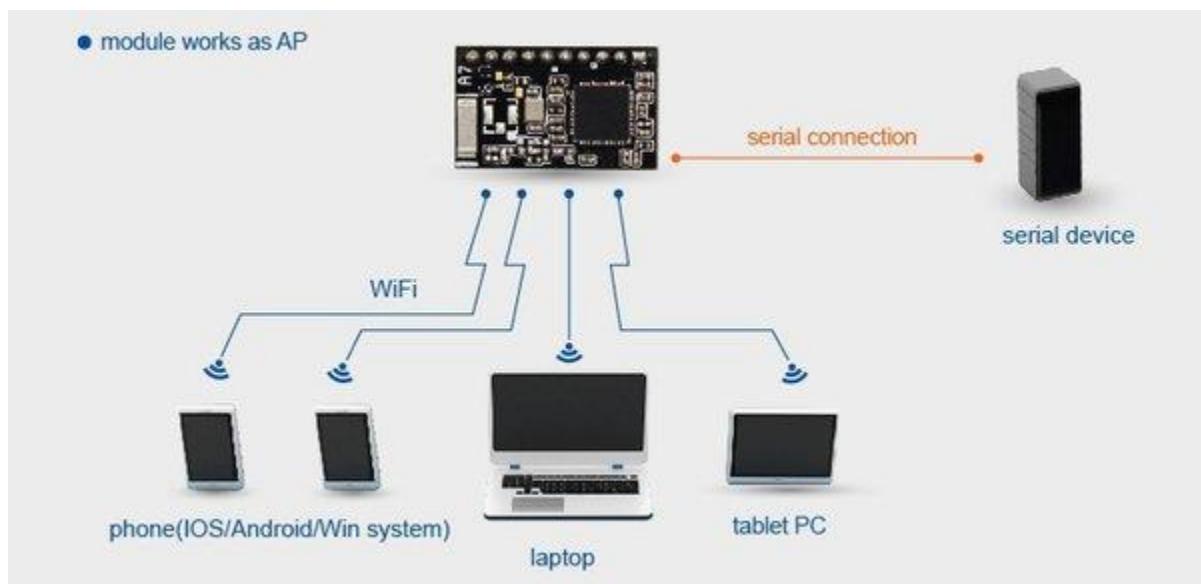
Reference: <https://www.geeksforgeeks.org/network-devices-hub-repeater-bridge-switch-router-gateways/>

6. Gateway

A gateway, as the name suggests, is a passage to connect two networks together that may work upon different networking models. They basically work as the messenger agents that take data from one system, interpret it, and transfer it to another system. Gateways are also called protocol converters and can operate at any network layer. Gateways are generally more complex than switches or routers. Gateway is also called a protocol converter.

4.2 Local and Personal Area Networks (LAN/PAN) for IOT

Networks that cover fairly short distances are called personal area networks (PAN) and local area networks (LAN). PAN and LAN networks are considered to be fairly cost-effective, but the transfer of data can sometimes be unreliable.



Reference: <https://www.quora.com/How-does-the-Internet-of-Things-work-in-a-LAN-network>

Wireless personal and local area network technologies that are commonly incorporated into IoT connectivity solutions are WiFi and Bluetooth. WiFi can be used for applications that run in a local environment, or in a distributed setting if there are multiple access points integrated into a larger network. One downside to WiFi is that it works only if the signal is strong and you're close to the access point. Also, WiFi is generally more power-hungry than people think, but it is possible to operate it in a way that's a little more power-efficient (for example, your device only connects periodically to send data, then goes back to sleep).

Bluetooth Low Energy (BLE) is a more energy-efficient wireless network protocol—if you're not receiving data constantly, a single battery running BLE could last up to five years. However, compared to WiFi it is slower to transmit and is more limited in the amount of data it is capable of sending.

Both WiFi and Bluetooth are easy to connect in most cases, although WiFi does have some security challenges that may be difficult to overcome.

4.3 IOT WAN

A wide area network (also known as WAN), is a large network of information that is not tied to a single location. WANs can facilitate communication, the sharing of information and much more between devices from around the world through a WAN provider.

WANs can be vital for international businesses, but they are also essential for everyday use, as the internet is considered the largest WAN in the world.

An Internet of Things (IoT) gateway is a device which serves as the connection point between IoT devices and the cloud. This gateway can be a hardware appliance or virtual.

4.4 IOT NODE



Reference: [https://advdownload.advantech.com/productfile/PIS/WISE_1020/Product%20-%20Photo\(Main\)/WISE-1020-OS01E_3D_B20151008140951.jpg](https://advdownload.advantech.com/productfile/PIS/WISE_1020/Product%20-%20Photo(Main)/WISE-1020-OS01E_3D_B20151008140951.jpg)

The most numerous type of device in the IoT can be referred to as the node. These are all the exciting devices that are providing sensor data, or devices that are being controlled from the cloud. This means things like door locks, security sensors, temperature sensors, and more.

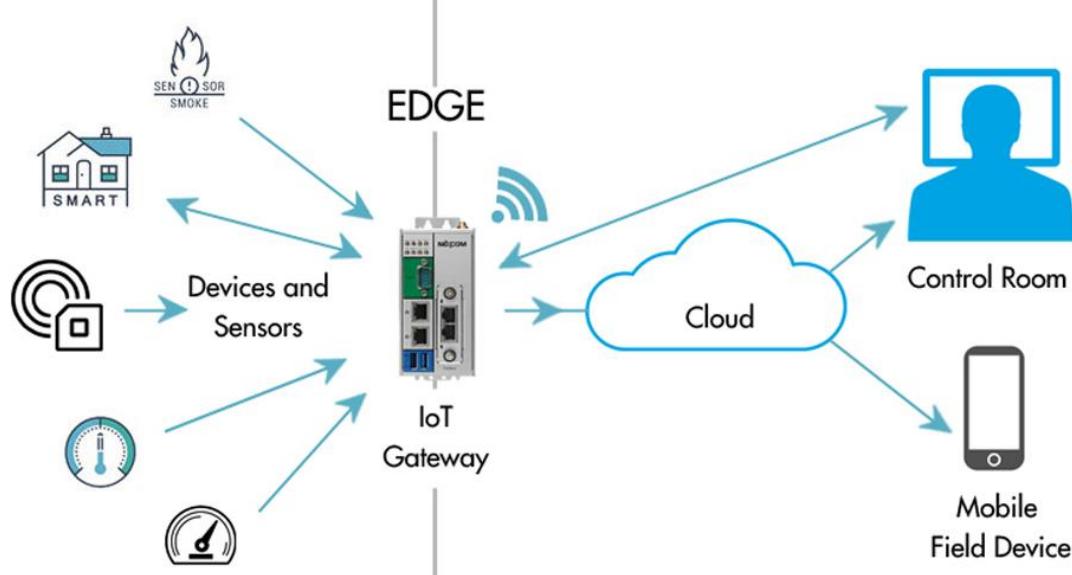
Put simply, the node is the “thing” in Internet of Things, and until recently they were a practical impossibility. Nodes tend to be either lightweight sensor devices, which primarily gather status information over a pre-programmed interval, or middleweight devices which also offer controllable functions (like a door lock which can be toggled,

traffic lights whose patterns can be adjusted, or industrial equipment which can be disabled if a fault is triggered.

The IoT node as we know it today, in its most minimal use case, can be a sensor embedded in an object that is never serviced again across the life of the device. They can be wireless and operated on a coin cell battery for years. What seemed impossible just a few years ago is now quickly becoming standard. And that's thanks to incredible innovations in low-power operation of wireless modules.

4.5 IOT Gateway

An IoT gateway works by receiving data from IoT sensors, which it can then send onwards to the cloud; it also receives information from the cloud which then goes to the device itself to help it perform necessary functions, such as regulating environmental changes and detecting possible issues with functioning. All information moving from an IoT device to the cloud, or vice versa, goes through the connected IoT gateway. By managing this connection, the gateway can perform security tasks, help manage devices and translate protocols.



https://www.ezenroute.com/assets/images/gif/iot_gateway_img_1.jpg

One benefit of an IoT gateway is added security for the IoT network and data. Because the gateway protects information moving in both directions, it protects data moving to the cloud from leaks, as well as prevents unauthorized control of IoT devices from outside parties.

Traditional IoT gateways are non-intelligent and perform basic gateway functionalities. However, non-intelligent gateways have recently been pushed out by "smart" IoT gateways, which are able to perform edge analytics on data produced by IoT devices before it is sent to the cloud. This makes analytics much faster and cuts down on

storage for the vast amount of data produced by IoT products. However, performing edge analytics instead of keeping all IoT data may not be an effective solution in some situations, as this process causes the loss of a lot of raw data.

IoT gateways can also be used to convert non-cloud connected legacy devices to the internet for brownfield development. By connecting a gateway to a device's sensors, the data can be analyzed or transported directly by the gateway, even though the device itself would be unable to do so.

4.6 IPv4 vs IPv6

What is IP?

An IP (Internet Protocol) address is a numerical label assigned to each device connected to a computer network that uses the IP protocol for communication. An IP address acts as an identifier for a specific device on a particular network. The IP address is also called an IP number or Internet address.

IP address specifies the technical format of the addressing and packets scheme. Most networks combine IP with a TCP (Transmission Control Protocol). It also allows developing a virtual connection between a destination and a source.

Now in this IPv4 and IPv6 difference tutorial, we will learn What is IPv4 and IPv6?



Example: 127.255.255.255

Example:

2001:0db8:85a3:0000:0000:8a2e:0370:7334

<https://www.guru99.com/difference-ipv4-vs-ipv6.html>

What is IPv4?

IPv4 is an IP version widely used to identify devices on a network using an addressing system. It was the first version of IP deployed for production in the ARPANET in 1983. It uses a 32-bit address scheme to store 2^{32} addresses which is more than 4 billion addresses. It is considered the primary Internet Protocol and carries 94% of Internet traffic.

What is IPv6?

IPv6 is the most recent version of the Internet Protocol. This new IP address version is being deployed to fulfill the need for more Internet addresses. It was aimed to resolve issues that are associated with IPv4. With 128-bit address space, it allows 340 undecillion unique address space. IPv6 is also called IPng (Internet Protocol next generation).

Internet Engineer Taskforce initiated it in early 1994. The design and development of that suite are now called IPv6.

KEY DIFFERENCE

- IPv4 is 32-Bit IP address whereas IPv6 is a 128-Bit IP address.
- IPv4 is a numeric addressing method whereas IPv6 is an alphanumeric addressing method.
- IPv4 binary bits are separated by a dot(.) whereas IPv6 binary bits are separated by a colon(:).
- IPv4 offers 12 header fields whereas IPv6 offers 8 header fields.
- IPv4 supports broadcast whereas IPv6 doesn't support broadcast.
- IPv4 has checksum fields while IPv6 doesn't have checksum fields
- When we compare IPv4 and IPv6, IPv4 supports VLSM (Variable Length Subnet Mask) whereas IPv6 doesn't support VLSM.
- IPv4 uses ARP (Address Resolution Protocol) to map to MAC address whereas IPv6 uses NDP (Neighbour Discovery Protocol) to map to MAC address.

IPv4	IPv6
IPv4 has a 32-bit address length	IPv6 has a 128-bit address length
It Supports Manual and DHCP address configuration	It supports Auto and renumbering address configuration
In IPv4 end to end, connection integrity is Unachievable	In IPv6 end to end, connection integrity is Achievable
It can generate 4.29×10^9 address space	Address space of IPv6 is quite large it can produce 3.4×10^{38} address space
The Security feature is dependent on application	IPSEC is an inbuilt security feature in the IPv6 protocol
Address representation of IPv4 is in decimal	Address Representation of IPv6 is in hexadecimal
Fragmentation performed by Sender and forwarding routers	In IPv6 fragmentation performed only by the sender
In IPv4 Packet flow identification is not available	In IPv6 packet flow identification are Available and uses the flow label field in the header

IPv4	IPv6
In IPv4 checksum field is available	In IPv6 checksum field is not available
It has broadcast Message Transmission Scheme	In IPv6 multicast and anycast message transmission scheme is available
In IPv4 Encryption and Authentication facility not provided	In IPv6 Encryption and Authentication are provided
IPv4 has a header of 20-60 bytes.	IPv6 has header of 40 bytes fixed
IPv4 consist of 4 fields which are separated by dot (.)	IPv6 consist of 8 fields, which are separated by colon (:)
Example of IPv4 – 66.94.29.13	Example of IPv6 – 2001:0000:3238:DFE1:0063:0000:0000:FEFB

4.7 Multi-homing

Multi-homing is a method of configuring one computer, called the host, with more than one network connection and IP address. The multi-homed method provides enhanced and reliable Internet connectivity without compromising efficient performance.

Why Is Multi-Homing Important?

With more and more Internet-connected devices, an organization's workforce is no longer sequestered to a single location. Instead, an organization may have employees connecting to their internal network and accessing sensitive data from across the globe. Because of this, old access security measures are no longer enough and must be replaced with safeguards that allow employees and other verified users safe and secure access from anywhere, at any time, from any device.

Using a multi-homed approach can:

- Help load balancing and let a network work with less downtime
- Give added safeguards against system failure
- Help maintain the system during disaster and recovery

Why multi-homing in IOT?

In IoT particular Node or an IoT device or the sub network, IoT sub network can be connected with multiple networks for improving the reliability. So, basically multi-homing is a concept that is used for improving the overall liability of the network in that way. So, in the same state if some component of the network or maybe a Node has gone down, there is another network that can take over.

4.8 IoT Protocol Stack

The Open Systems Interconnection (OSI) and Transmission Control Protocol/Internet Protocol (TCP/IP) networking models are the most common frameworks to encapsulate networking tasks in the form of multiple layers. Widely adopted IoT protocols can be mapped to these two models as outlined in the below table. For an IoT network to function effectively, protocols at different layers must be interoperable with each other.

OSI Model	TCP/IP Model	Layer functions	Protocols
Application	Application	Directly interact with and support user's applications	MQTT, HTTPS, AMQP, CoAP
Presentation			
Session			
Transport	Transport	Handle reliability, flow control, congestion avoidance, and error correction	TCP, UDP
Network	Internet	Involve logical addressing and define how data is routed from sources to final destination hosts identified by IP addresses (traffic directing)	IP (e.g. IPv6, 6LoWPAN...)
Data Link	Network access & physical	Define the physical connection of end devices to the network	LPWAN, WiFi, LTE, BLE, Zigbee
Physical			

Link Layer Protocols:

Link Layer protocols determine how the data is physically sent over the networks physical layer or medium(example copper wire, electrical cable, or radio wave). The Scope of The Link Layer is the Last Local Network connections to which host is attached. Host on the same link exchange data packets over the link layer using the link layer protocol. Link layer determines how the packets are coded and signaled by the hardware device over the medium to which the host is attached.

802.3 Ethernet:

802.3 is a collections of wired Ethernet standards for the link layer. For example 802.3 10BASE5 Ethernet that uses coaxial cable as a shared medium, 802.3.i is standard for 10 BASET Ethernet over copper twisted pair connection, Standards provide data rates from 10 Mb/s to 40 gigabits per second and the higher. The shared medium in Ethernet can be a coaxial cable , twisted pair wire or and Optical fiber. Shared medium carries the communication for all the devices on the network.

802.1- WI-FI:

IEEE 802.3 is a collections of wireless Local area network.(WLAN) communication standards, including extensive descriptions of the link layer. For example 802.11a operate in the 5 GHz band, 802.11b and 802.11g operate in the 2.4 GHz band. 802.11ac operate in the 5G hertz band.

802.16 wiMAX:

IEEE 802.16 is a collection of wireless broadband standards, including extensive descriptions for the link layer also called WiMAX. The WiMAX standard provides data rates from 1.5 Mb/s to 1Gb/s. The recent update provides data rates of hundred megabits per second for mobile stations.

802.15.4 LR-WPAN:

IEEE 802.15.4 is a collection of standards for low rate wireless personal area networks (LRWPAN). These standards form the basis of specifications for high level communication like Zigbee. LR-WPAN standards provide data rates from 40 k b/s. These standards provide low cost and low speed communications for power constrained devices.

2G / 3G / 4G mobile communications:

These are the different generations of mobile communication standards including second generation (2G including GSM and CDMA), 3rd Generation (3G including UMTS and CDMA2000) and 4th generation (4G including LTE).

Network / internet layer Protocols:

The network layer is responsible for sending of IP datagrams from the source network to the destination network. This layer performs the host addressing and packet routing. The datagrams contain a source and destination address which are used to route them from the source to the destination across multiple networks. Host identification is done using the hierarchy IP addressing schemes such as IPv4 or IPv6.

IPv4:

Internet protocol version for open parents close (IPv4) is the most deployed internet protocol that is used to identify the device on a network using a hierarchy address scheme. It uses 32-bit addresses that allow a total of 2³² addresses. As more and more devices got connected to the internet, the IPv4 has succeeded by IPv6.

IPv6:

It is the newest version of internet protocol and successor to IPv4. IPv6 uses 128-bit address schemes that allow a total of 2¹²⁸ addresses.

6LoWPAN:

IPv6 over low power wireless personal area networks brings IP protocol to the low power device which has limited processing capability. It operates in the 2.4 GHz frequency range and provides a data transfer rate of up to 50 kb/s.

Transport layer protocols:

The Transport layer protocols provide end-to-end message transfer capability independent of the underlying network. The message transfer capability can be set up on connections, either using handshake or without handshake acknowledgements. Provides functions such as error control, segmentation, flow control and congestion control.

TCP:

Transmission control protocol is the most widely used to transport layer protocol that is used by the web browsers along with HTTP , HTTPS application layer protocols email program (SMTP application layer protocol) and file transfer protocol. TCP is a connection Oriented and stateful protocol while IP protocol deals with sending packets,TCP ensures reliable transmissions of packets in order. TCP also provide error deduction capability so that duplicate packets can be discarded and low packets are retransmitted The flow control capability ensures that the rate at which the sender since the data is now to too to high for the receiver to process.

UDP:

Unlike TCP, which requires carrying out an initial setup procedure, UDP is a connection less protocol. UDP is useful for time sensitive application they have very small data units to exchange and do not want the overhead of connection setup. UDP is a transactions oriented and stateless protocol. UDP does not provide guaranteed delivery, ordering of messages and duplicate eliminations.

Application layer protocols:

Application layer protocol define how the application interfaces with the lower layer protocols to send the data over the network. Data are typically in files, is encoded by the application layer protocol and encapsulated in the transport layer protocol. Application layer protocol enable process-to-process connection using ports.

Http:

Hypertext transfer protocol is the application layer protocol that forms the foundations of world wide web http includes, ,commands such as GET, PUT,POST, DELETE, HEAD, TRACE, OPTIONS etc. The protocol follows a request response model where client sends request to server using the http, commands. Http is a stateless protocol and each http request is independent father request and http client can be a browser or an application running on the client example and application running on an IoT device ,mobile mobile applications or other software.

CoAP:

Constrained application protocol is an application layer protocol for machine to machine application M2M meant for constrained environment with constrained devices and constrained networks. Like http CoAP is a web transfer protocol and uses a request- response model, however it runs on the top of the UDP instead of TC CoAP uses a client –server architecture where client communicate with server using connectionless datagrams.It is designed to easily interface with http like http,CoAP supports method such as GET, PUT, DELETE .

Websocket:

Websocket protocol allows full duplex communication over a single socket connection for sending message between client and server. Websocket is based on TCP and Allows streams of messages to be sent back and forth between the client and server while keeping the TCP connection open. The client can be a browser, a mobile application and IoT device

MQTT :

Message Queue Telemetry Transport it is a lightweight message protocol based on public -subscribe model MQTT uses a client server Architecture by the clients such as an IoT device connect to the server also called the MQTT broker and publishers' message to topic on the server. The broker forwards the message to the clients subscribed to topic MQTT is well suited for constrained and environments.

AMQP:

Advanced Message Queuing protocols. it is an open application layer protocol for business messaging. AMQP support point to point and publish - subscribe model routing and queuing. AMQP broker receive message from publishers example devices or applications that generate data and about them over connections to consumers publishers publish the message to exchange which then distribute message copies to queues.

Unit 5: Raspberry Pi

Learning Outcomes:

- Understand the Hardware structure of Raspberry Pi
- Able to install operating system into raspberry pi
- Able to interface various sensors and actuators

5.1 Introduction to Raspberry Pi

Raspberry Pi, developed by Raspberry Pi Foundation in association with Broadcom, is a series of small single-board computers and perhaps the most inspiring computer available today.

Raspberry Pi is a small single board computer. By connecting peripherals like Keyboard, mouse, display to the Raspberry Pi, it will act as a mini personal computer. Raspberry Pi is popularly used for real time Image/Video Processing, IoT based applications and Robotics applications. Raspberry Pi is slower than laptop or desktop but is still a computer which can provide all the expected features or abilities, at a low power consumption.

Raspberry Pi Foundation officially provides Debian based Raspbian OS. Also, they provide NOOBS OS for Raspberry Pi. We can install several Third-Party versions of OS like Ubuntu, Archlinux, RISC OS, Windows 10 IOT Core, etc.

Raspbian OS is official Operating System available for free to use. This OS is efficiently optimized to use with Raspberry Pi. Raspbian have GUI which includes tools for Browsing, Python programming, office, games, etc.

We should use SD card (minimum 8 GB recommended) to store the OS (operating System). Raspberry Pi is more than computer as it provides access to the on-chip hardware i.e. GPIOs for developing an application. By accessing GPIO, we can connect devices like LED, motors, sensors, etc and can control them too.

It has ARM based Broadcom Processor SoC along with on-chip GPU (Graphics Processing Unit).

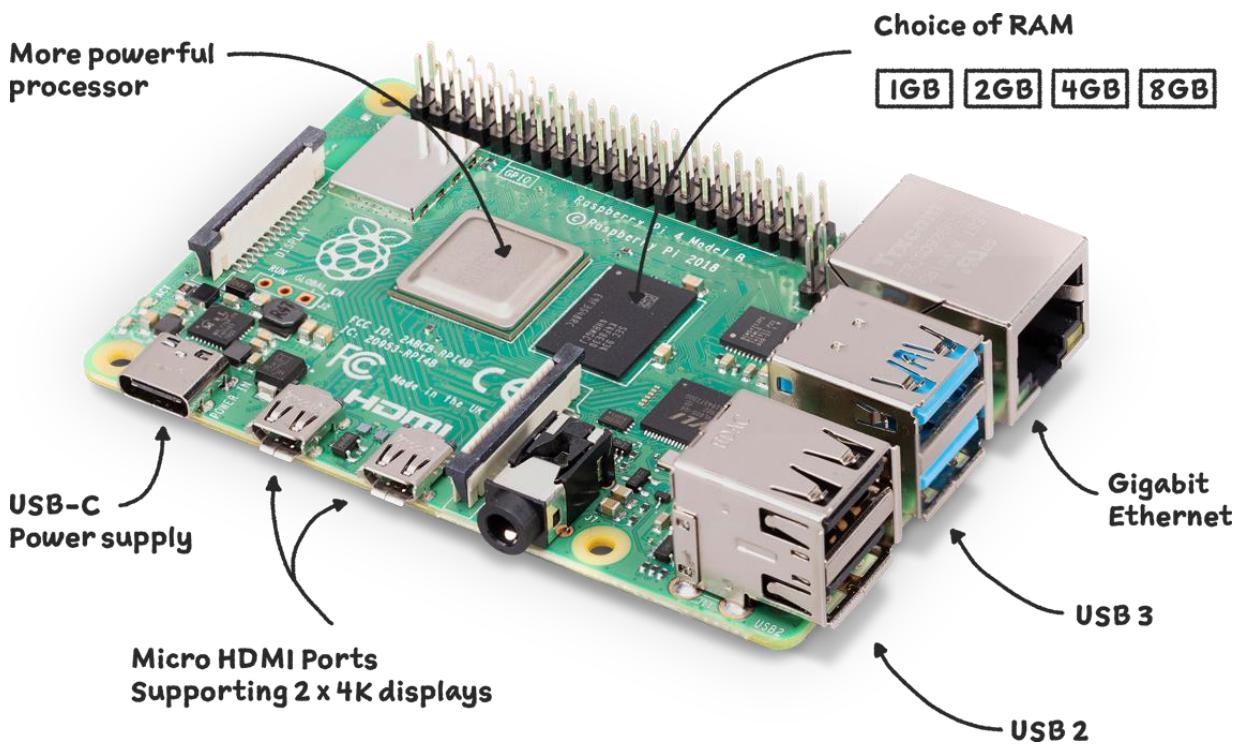
The CPU speed of Raspberry Pi varies from 700 MHz to 1.2 GHz. Also, it has on-board SDRAM that ranges from 256 MB to 1 GB.

Raspberry Pi also provides on-chip SPI, I2C, I2S and UART modules.

There are different versions of raspberry pi available as listed below:

1. Raspberry Pi 1 Model A
2. Raspberry Pi 1 Model A+
3. Raspberry Pi 1 Model B
4. Raspberry Pi 1 Model B+
5. Raspberry Pi 2 Model B
6. Raspberry Pi 3 Model B
7. Raspberry Pi Zero

Raspberry Pi 4 – model B



Raspberry Pi 4 Model B is the latest product in the popular Raspberry Pi range of computers. It offers ground-breaking increases in processor speed, multimedia performance, memory, and connectivity compared to the prior-generation Raspberry Pi 3 Model B+, while retaining backwards compatibility and similar power consumption. For the end user, Raspberry Pi 4 Model B provides desktop performance comparable to entry-level x86 PC systems.

This product's key features include a high-performance 64-bit quad-core processor, dual-display support at resolutions up to 4K via a pair of micro-HDMI ports, hardware video decode at up to 4Kp60, up to 4GB of RAM, dual-band 2.4/5.0 GHz wireless LAN, Bluetooth 5.0, Gigabit Ethernet, USB 3.0, and PoE capability (via a separate PoE HAT add-on).

The dual-band wireless LAN and Bluetooth have modular compliance certification, allowing the board to be designed into end products with significantly reduced compliance testing, improving both cost and time to market.

Specification

Processor	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
Memory	1GB, 2GB, 4GB or 8GB LPDDR4 (depending on model) with on-die ECC
Connectivity	2.4 GHz and 5.0 GHz IEEE 802.11b/g/n/ac wireless LAN, Bluetooth 5.0, BLE Gigabit Ethernet 2 × USB 3.0 ports 2 × USB 2.0 ports.
GPIO	Standard 40-pin GPIO header (fully backwards-compatible with previous boards)
Video & Sound	2 × micro HDMI ports (up to 4Kp60 supported) 2-lane MIPI DSI display port 2-lane MIPI CSI camera port 4-pole stereo audio and composite video port
Multimedia	H.265 (4Kp60 decode); H.264 (1080p60 decode, 1080p30 encode); OpenGL ES, 3.0 graphics
SD Card Support	Micro SD card slot for loading operating system and data storage
Input Power	5V DC via USB-C connector (minimum 3A1) 5V DC via GPIO header (minimum 3A1) Power over Ethernet (PoE)-enabled (requires separate PoE HAT)
Environment	Operating temperature 0–50°C

Uses

Like a desktop computer, you can do almost anything with the Raspberry Pi. You can start and manage programs with its graphical windows desktop. It also has the shell for accepting text commands.

We can use the Raspberry Pi computer for the following –

- Playing games
- Browsing the internet
- Word processing
- Spreadsheets
- Editing photos
- Paying bills online
- Managing your accounts.

The best use of Raspberry Pi is to learn how a computer works. You can also learn how to make electronic projects or programs with it.

It comes with two programming languages, **Scratch** and **Python**. Through GPIO (general-purpose input output) pins, Raspberry Pi can be connected to other circuits, so that you can control the other devices of your choice.

5.2 Install Raspbian OS in Raspberry Pi 4 B

Raspberry Pi recommends the use of Raspberry Pi Imager to install an operating system on to your SD card. You will need another computer with an SD card reader to install the image. Raspberry Pi Imager can be run on another Raspberry Pi, but also works on Microsoft Windows, Apple macOS, and Linux.

Using Raspberry Pi Imager

Raspberry Pi have developed a graphical SD card writing tool that works on Mac OS, Ubuntu 18.04, and Windows called Raspberry Pi Imager; this is the easiest option for most users since it will download the image automatically and install it to the SD card.

Download the latest version of [Raspberry Pi Imager](#) and install it. If you want to use Raspberry Pi Imager from a second Raspberry Pi, you can install it from a terminal using ***sudo apt install rpi-imager***. Then:

- Connect an SD card reader with the SD card inside.
- Open Raspberry Pi Imager and choose the required OS from the list presented.
- Choose the SD card you wish to write your image to.
- Review your selections and click on the Write button to begin writing data to the SD Card.

Note:

If using Raspberry Pi Imager on Windows 10 with controlled folder access enabled, you will need to explicitly allow Raspberry Pi Imager permission to write the SD card. If this is not done, the imaging process will fail with a "failed to write"

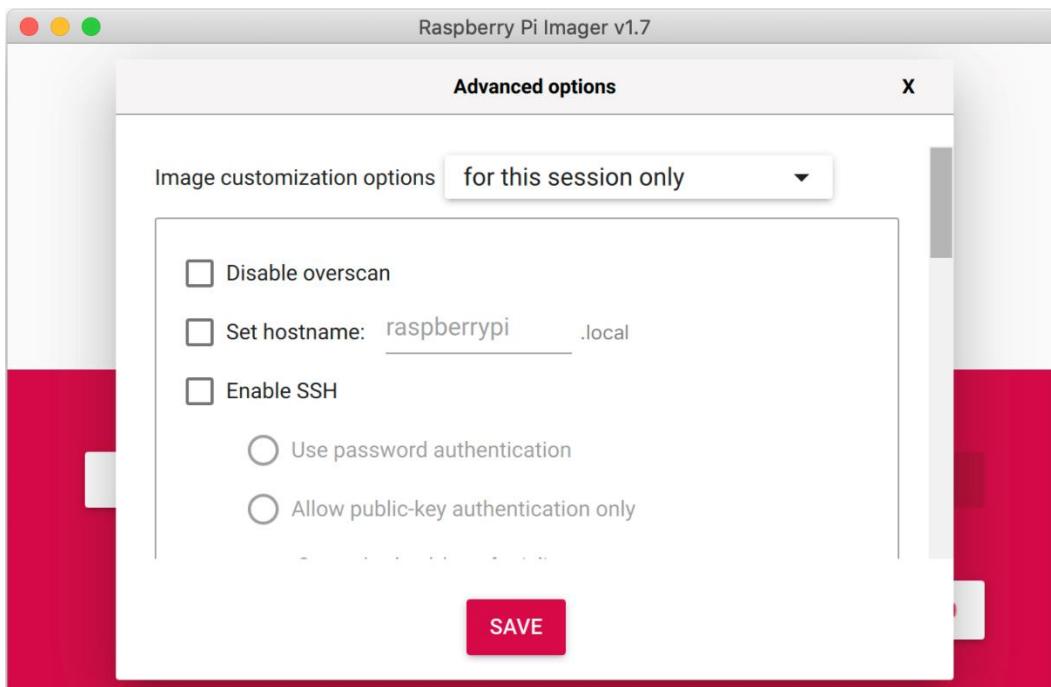
You can now insert the SD card into the Raspberry Pi and power it up. When your Raspberry Pi boots for the first time a configuration wizard will run that allows you to set up your Raspberry Pi.

Advanced Options

When you have the Raspberry Pi Imager open, and after you have selected the operating system to install, a cog wheel will appear allowing you to open an "Advanced Options" menu if it is supported by the operating system. This menu lets you carry out tasks like enabling SSH, or setting your Raspberry Pi's hostname, and configuring the default user before first boot.



Amongst other things the Advanced Options menu is useful for when you want to configure a headless Raspberry Pi.



If you are installing Raspberry Pi OS Lite and intend to run it headless, you will still need to create a new user account. Since you will not be able to create the user

account on first boot, you **MUST** configure the operating system using the Advanced Menu.

Downloading an Image

If you are using a different tool than Raspberry Pi Imager to write to your SD Card, most require you to download the image first, then use the tool to write it to the card. Official images for recommended operating systems are available to download from the Raspberry Pi website downloads page. Alternative operating systems for Raspberry Pi computers are also available from some third-party vendors.

You may need to unzip the downloaded file (.zip) to get the image file (.img) you need to write to the card.

5.3 Configure GrovePi+ Kit

What is GrovePi+

GrovePi+ is add-on board with 15 Grove 4-pin interfaces that brings Grove sensors to the Raspberry Pi. It is the newest version compatible with Raspberry Pi model B/B+/A+/2/3/4 perfectly.

GrovePi+ is an easy-to-use and modular system for hardware hacking with the Raspberry Pi, no need for soldering or breadboards: plug in your Grove sensors and start programming directly. Grove is an easy to use collection of more than 100 inexpensive plug-and-play modules that sense and control the physical world. By connecting Grove Sensors to Raspberry Pi, it empowers your Pi in the physical world. With hundreds of sensors to choose from Grove families, the possibilities for interaction are endless.

- Compatible with Raspberry Pi model B/B+/A+/2/3/4
- Faster SPI and higher reliability UART connections
- Easier to assemble camera cables and LCD cables
- Simplified procedures of firmware update

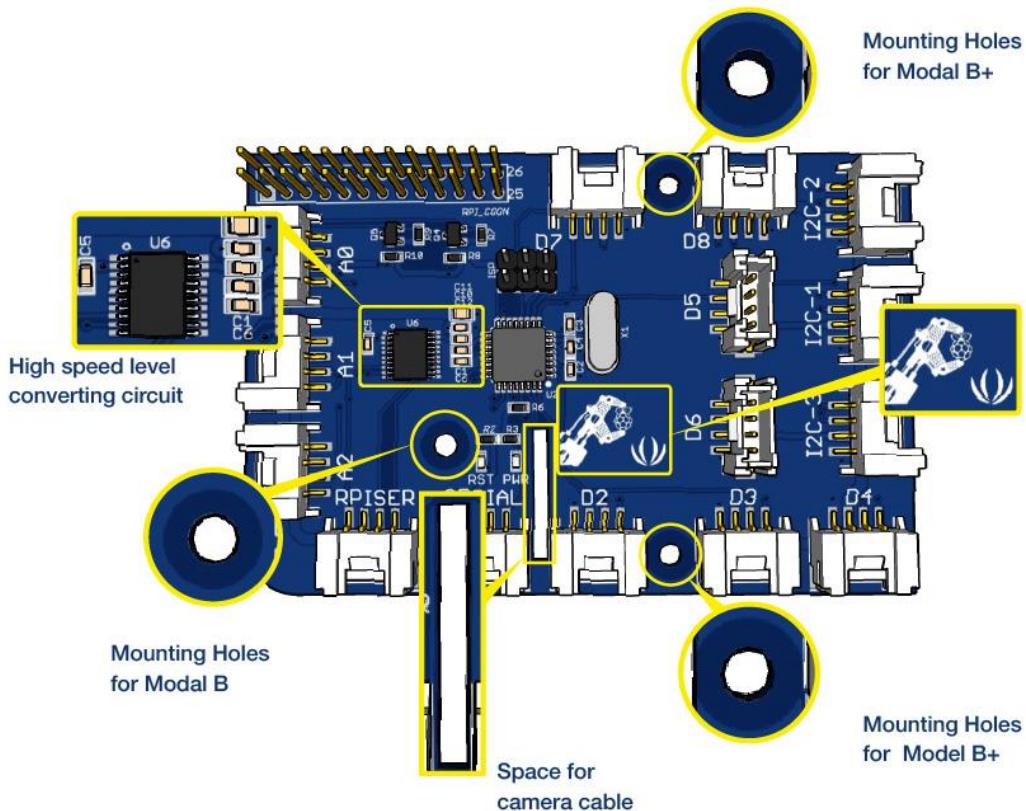


Image: GrovePi+ Board
Reference: [GrovePi+ add-on for Raspberry Pi – Seeed Studio](#)

Set-up in 4 simple steps

- Step 1 - Slip the GrovePi+ board over your Raspberry Pi
 - Step 2 - Connect the Grove modules to the GrovePi+ board
 - Step 3 - Upload your program to Raspberry Pi
 - Step 4 - Begin taking in the world data

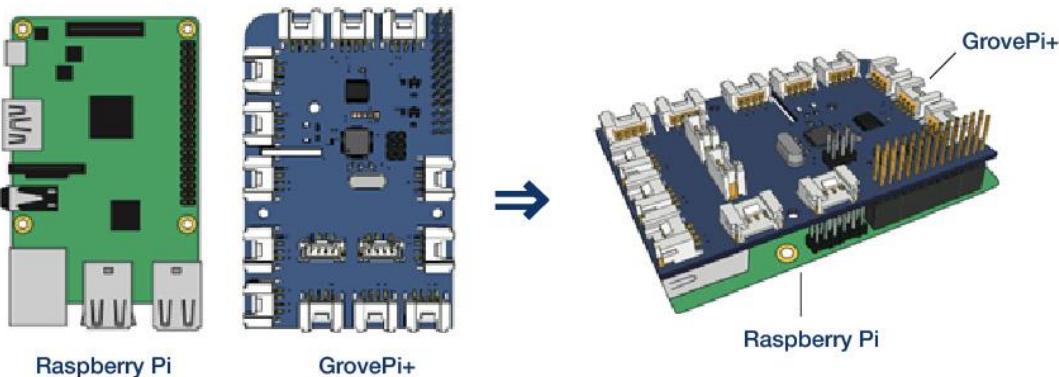


Image: GrovePi+ Board and Raspberry Pi interface
Reference: [GrovePi+ add-on for Raspberry Pi – Seeed Studio](#)

Features:

- 7 digital Ports
- 3 analog Ports
- 3 I2C ports
- 1 Serial port connect to GrovePi
- 1 Serial port connect to Raspberry Pi
- Grove header Vcc output Voltage: 5Vdc

The GrovePi Starter Kit gets you up and running with the GrovePi quickly. The starter kit bundles the most popular sensors for education and hobbyists, and lets you start playing and prototyping hardware with Raspberry Pi. No soldering required!

The GrovePi Starter Kit package includes:

- GrovePi+ Board
- 12 different Grove sensors and modules
- Grove cables for connecting the sensors to the GrovePi board.

GrovePi Kit Includes



Grove pi+ × 1



Grove - sound Sensor × 1



Grove - Temperature&Humidity × 1



Grove - light sensor × 1



Grove - Relay × 1



Grove - Button × 1



Grove - Ultrasonic Ranger × 1



Grove - Rotary Angle Sensor × 1



Grove-LCD RGB Backlight × 1



Grove - red led × 1



Grove Buzzer × 1



Grove blue led × 1



Grovepi+Guidebook × 1



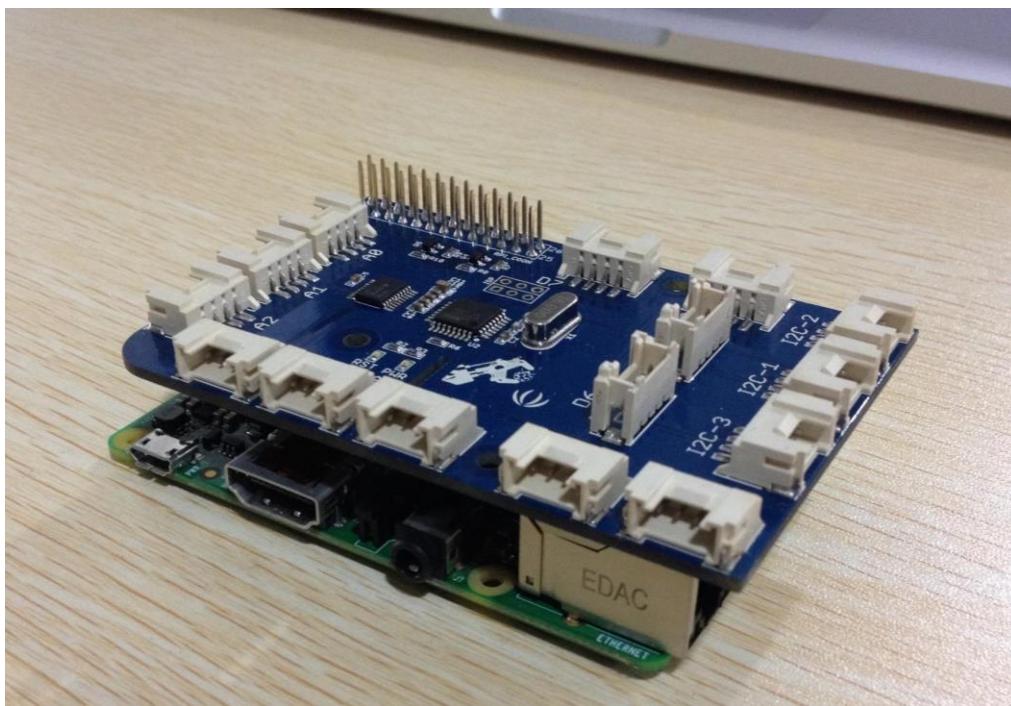
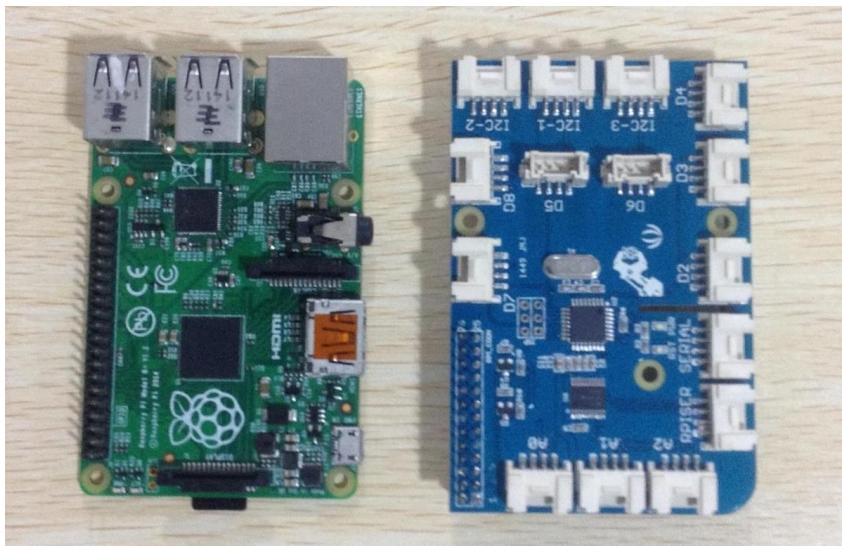
Grove green led × 1

Cables × 10

Image: Grove Pi Kit Sensors
Reference: [starter-Kit-content.jpg \(758x647\) \(dexterindustries.com\)](#)

Hardware connection for GrovePi+ to raspberry Pi

First, mount your GrovePi on the Raspberry Pi. The GrovePi slides over top of the Raspberry Pi as shown in the picture below.



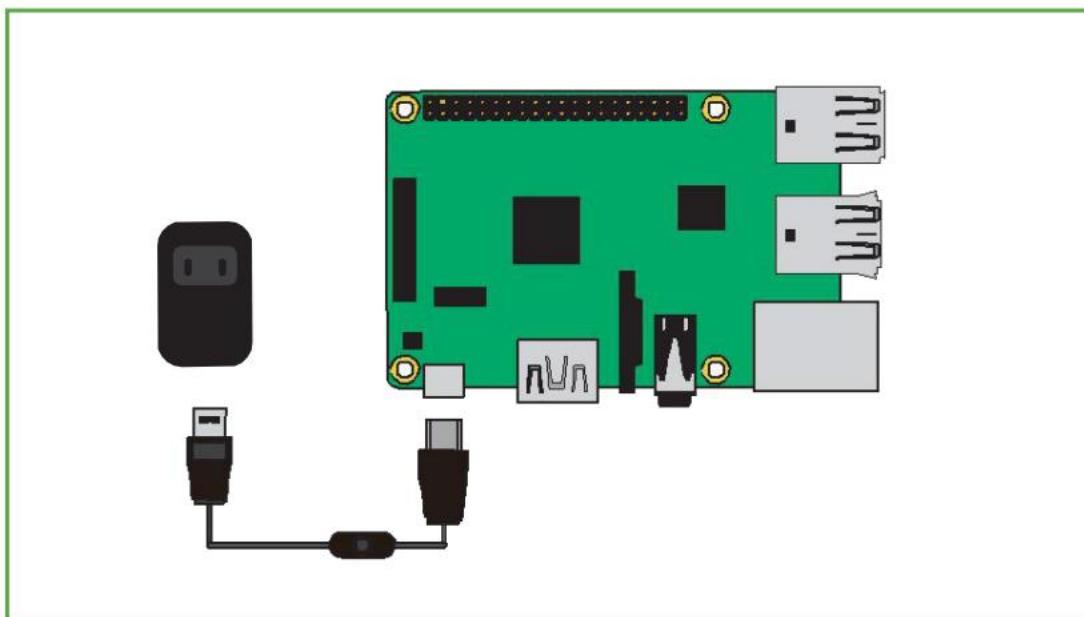
Ensure that the pins are properly aligned when stacking the GrovePi.

Powering up the Raspberry Pi

To power the GrovePi+ and the Raspberry Pi, you can use the micro-USB power port on the Raspberry Pi.

Remember to use a good power adapter capable of supplying 1A at 5V and you should be fine with the power.

If you want to run the GrovePi+ in a standalone configuration, then you should use a USB power bank



Setup the Software on the Raspberry Pi

Next we will install the software on the Raspberry Pi. There are two options for installation:

- You can use our BrickPi Image.
- Use your own image. If you already have your own flavor of linux running on the Raspberry Pi, you can use our bash script to setup for the GrovePi.

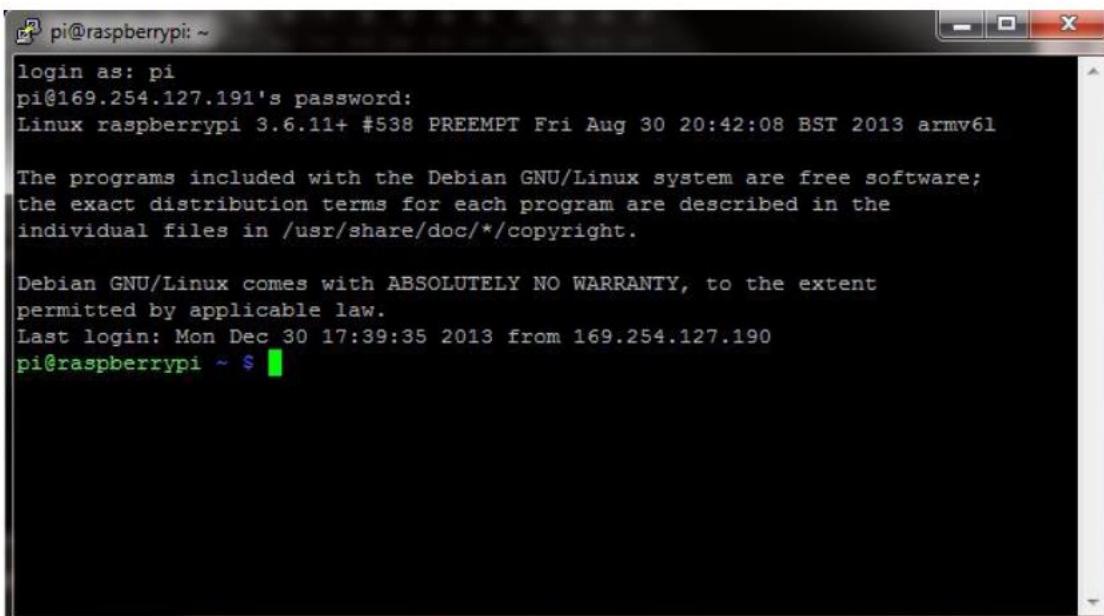
Using the BrickPi Image

- Download the Brick Pi Image and install the image on your SD card. [Here is the link to the BrickPi Page with steps to configure the SD card](#). You will need a minimum of 4GB SD Card for this installation.
- Clone the Github repository at an appropriate location in Raspbian
`git clone https://github.com/DexterInd/GrovePi.git`
- Run the bash script in the Scripts folder to configure the Raspbian. [Here is the tutorial for setting up with the Script.](#)
- Restart your Raspberry Pi.

Configuring your own image

- Clone the Github repository at an appropriate location
- `git clone https://github.com/DexterInd/GrovePi.git`
- Run the bash script in the Scripts folder to configure the Raspbian. [here](#) is the tutorial for setting up with the Script.
 - Restart the Raspberry Pi and start using the Grove Pi.

1. Power on the Raspberry Pi, without the GrovePi attached, and open a terminal (we'll be doing it on SSH, but it the same when using a standard Raspberry Pi setup with a monitor).



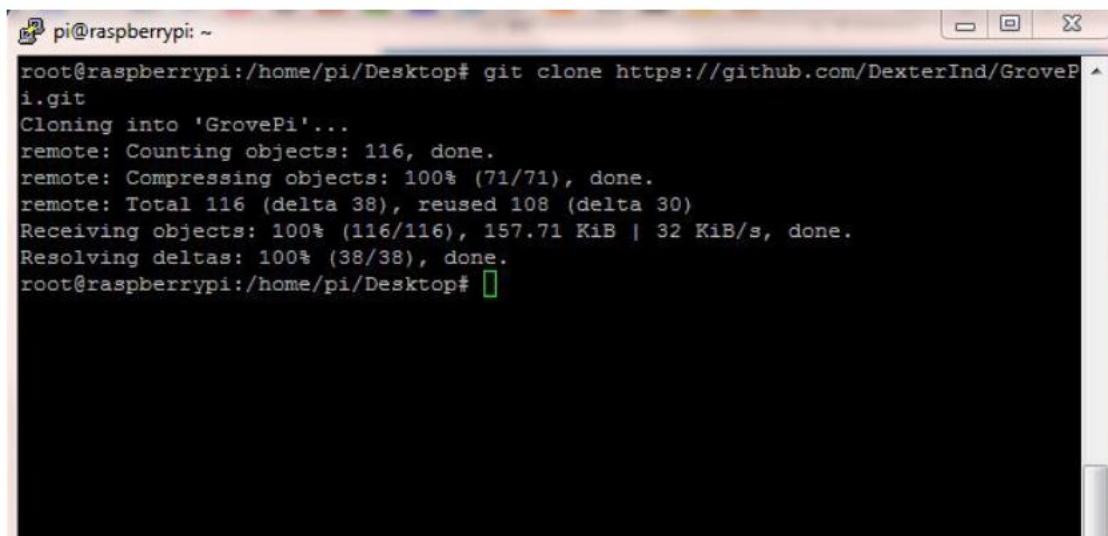
```
pi@raspberrypi: ~
login as: pi
pi@169.254.127.191's password:
Linux raspberrypi 3.6.11+ #538 PREEMPT Fri Aug 30 20:42:08 BST 2013 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Dec 30 17:39:35 2013 from 169.254.127.190
pi@raspberrypi ~ $
```

2. Change directories go to an appropriate location on your Pi where you want the GrovePi files to be stored (We recommend that you do it on the Desktop because it is easy to access and compatible with all our examples too). Clone the GrovePi git.

`git clone https://github.com/DexterInd/GrovePI`

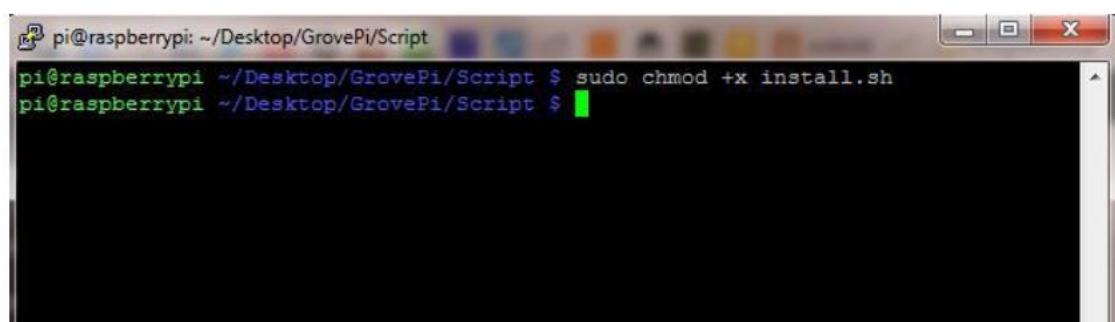


```
pi@raspberrypi: ~
root@raspberrypi:/home/pi/Desktop# git clone https://github.com/DexterInd/GrovePi.git
Cloning into 'GrovePi'...
remote: Counting objects: 116, done.
remote: Compressing objects: 100% (71/71), done.
remote: Total 116 (delta 38), reused 108 (delta 30)
Receiving objects: 100% (116/116), 157.71 KiB | 32 KiB/s, done.
Resolving deltas: 100% (38/38), done.
root@raspberrypi:/home/pi/Desktop#
```

When the repository is done downloading, there should be a new folder called "GrovePi"

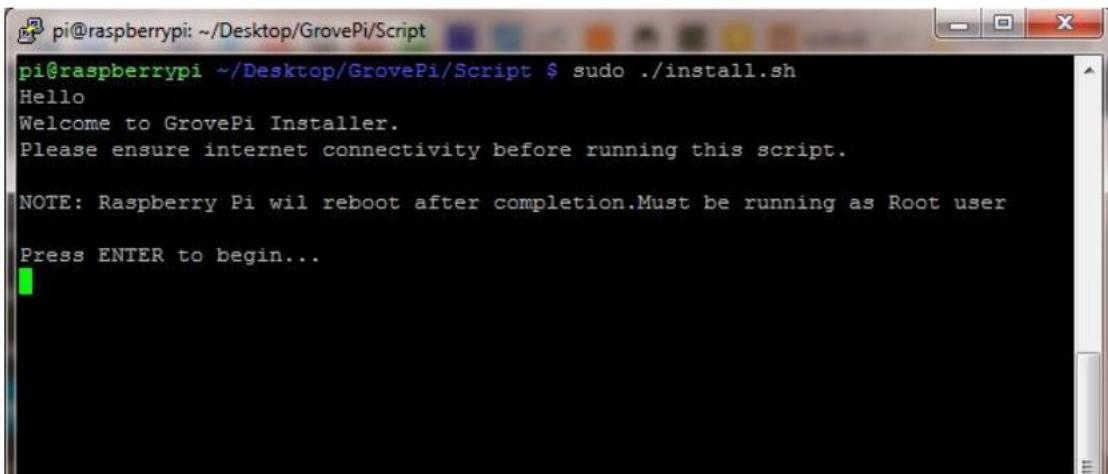
3. Go to the Scripts folder in the GrovePi folder
`cd GrovePi/Script`
4. Make the install.sh bash script as executable. We do this by modifying the permissions of the script:

```
sudo chmod +x install.sh
```



```
pi@raspberrypi: ~/Desktop/GrovePi/Script
pi@raspberrypi ~/Desktop/GrovePi/Script $ sudo chmod +x install.sh
pi@raspberrypi ~/Desktop/GrovePi/Script $
```

5. Start the script. You must be the root user, so be sure to
`sudo ./install.sh`



```
pi@raspberrypi: ~/Desktop/GrovePi/Script
pi@raspberrypi ~/Desktop/GrovePi/Script $ sudo ./install.sh
Hello
Welcome to GrovePi Installer.
Please ensure internet connectivity before running this script.

NOTE: Raspberry Pi will reboot after completion. Must be running as Root user
Press ENTER to begin...
```

Press “Enter” to start when you are prompted.

6. The script will download packages from the internet which are used by the GrovePi. Press “y” when the terminal prompts and asks for permission to start the download.



```
pi@raspberrypi: ~
root@raspberrypi:/home/pi/Desktop/GrovePi/Script# chmod +x install.sh
root@raspberrypi:/home/pi/Desktop/GrovePi/Script# sudo ./install.sh
Hello
Welcome to GrovePi Installer.
Please ensure internet connectivity before running this script.

NOTE: Raspberry Pi will reboot after completion. Must be running as Root user
Press ENTER to begin...

Check for internet connectivity...
=====
Connected

Installing Dependencies
=====
Reading package lists... Done
Building dependency tree
Reading state information... Done
git is already the newest version.
git set to manually installed.
python-rpi.gpio is already the newest version.
The following extra packages will be installed:
  arduino-core avr-libc avrdude binutils-avr extra-xdg-menus gcc-avr libftdi1
    libjna-java librxtx-java lrzsz python-pkg-resources python-setuptools
    python2.6 python2.6-minimal
Suggested packages:
  arduino-mk avrdude-doc task-c-devel gcc-doc gcc-4.2 libjna-java-doc
  python-distribute python-distribute-doc python-wxgtk2.8 python-wxgtk2.6
  python-wxgtk python2.6-doc binfmt-support
Recommended packages:
  python-dev-all
The following NEW packages will be installed:
  arduino arduino-core avr-libc avrdude binutils-avr extra-xdg-menus gcc-avr
  i2c-tools libftdi1 libi2c-dev libjna-java librxtx-java lrzsz minicom
  python-pip python-pkg-resources python-serial python-setuptools python-smbus
  python2.6 python2.6-minimal
0 upgraded, 21 newly installed, 0 to remove and 0 not upgraded.
Need to get 29.8 MB of archives.
After this operation, 98.1 MB of additional disk space will be used.
Do you want to continue [Y/n]? 
```

7. . The Raspberry Pi will automatically restart when the installation is

```
(Reading database ... 69573 files and directories currently installed.)
Preparing to replace avrdude 5.11.1-1 (using avrdude_5.10-4_armhf.deb) ...
Unpacking replacement avrdude ...
Setting up avrdude (1:5.10-4) ...
Installing new version of config file /etc/avrdude.conf ...
Processing triggers for man-db ...
--2013-12-30 17:32:19--  http://project-downloads.drogon.net/gertboard/setup.sh
Resolving project-downloads.drogon.net (project-downloads.drogon.net)... 195.10.
226.169, 2a00:ce0:2:feed:beef:cafe:0:4
Connecting to project-downloads.drogon.net (project-downloads.drogon.net)|195.10.
.226.169|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1870 (1.8K) [application/x-sh]
Saving to: 'setup.sh'

100%[=====] 1,870      --.-K/s   in 0.003s

2013-12-30 17:32:20 (703 KB/s) - 'setup.sh' saved [1870/1870]

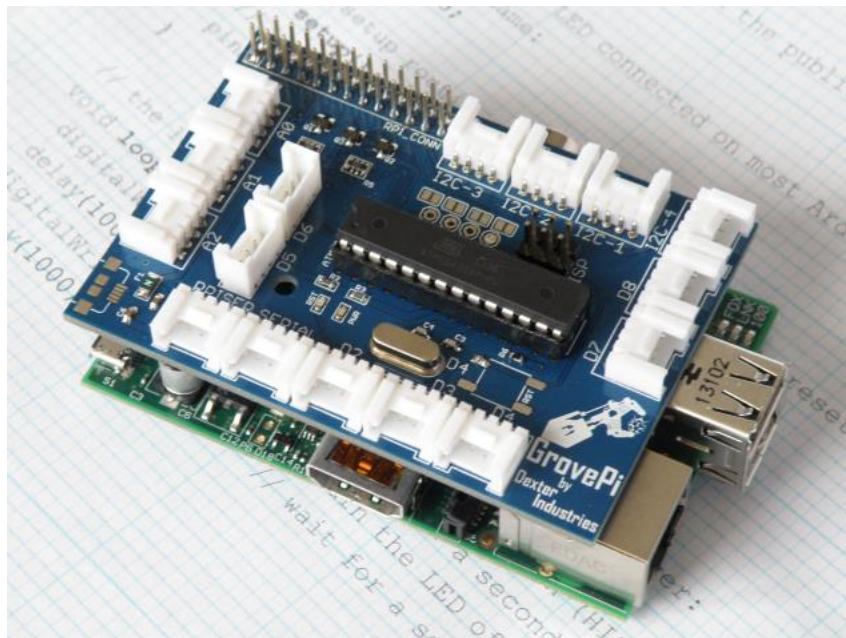
Setting up Raspberry Pi to make it work with the Gertboard
and the ATmega chip on-board with the Arduino IDE.

Checking ...
  Avrdude: OK
  Arduino IDE: OK
Fetching files:
  boards.txt
  programmers.txt
  avrsetup
Replacing/updating files:
  inittab: OK
  cmdline.txt: OK
  boards.txt: OK
  programmers.txt: OK
All Done.
Check and reboot now to apply changes.

Restarting
3
2
1

Broadcast message from root@raspberrypi (pts/0) (Mon Dec 30 17:32:28 2013):
The system is going down for reboot NOW!
root@raspberrypi:/home/pi/Desktop/GrovePi/Script#
```

8. Now when the Raspberry pi is powered down, stack the Grove Pi on top of the Raspberry Pi and power on the Raspberry Pi. A green light should power up on the Grove Pi. (Ensure that the pins are properly connected before powering the Raspberry Pi)

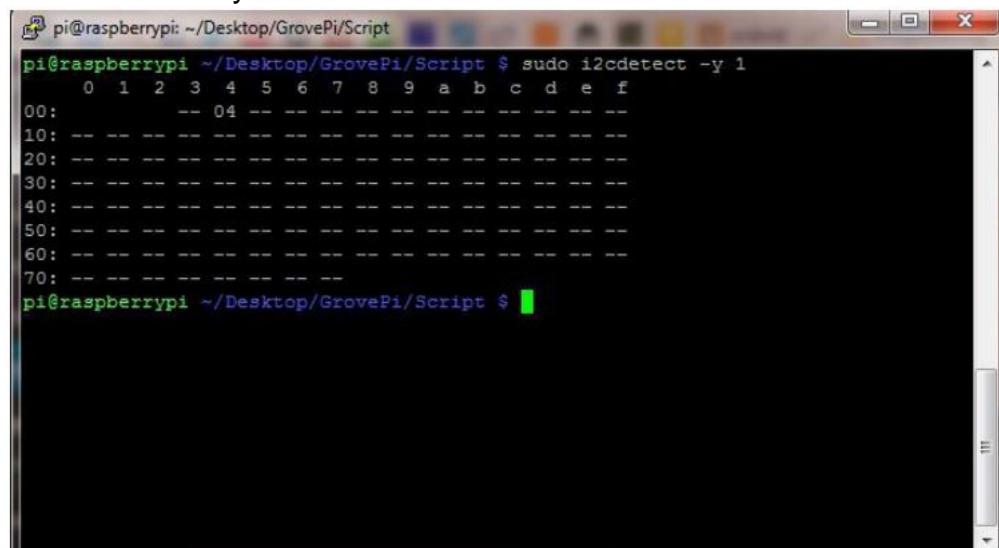


9. Now to check that the script was correctly installed. We will check that the Raspberry Pi is able to detect the Grove pi: run i2cdetect

```
sudo i2cdetect -y
```

If you have an Original Raspberry Pi (Sold before October 2012) – the I2C is port 0

```
sudo i2cdetect -y
```



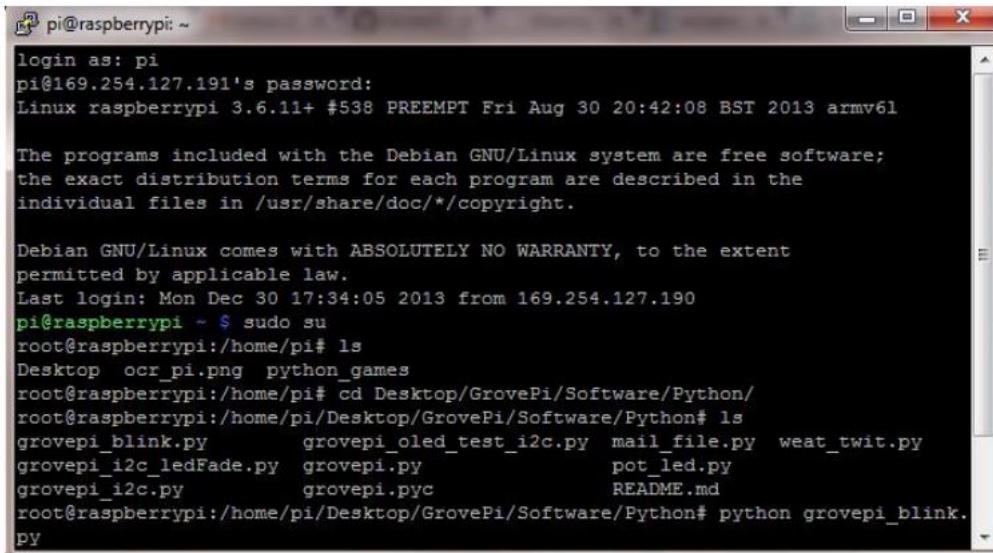
```
pi@raspberrypi: ~/Desktop/GrovePi/Script $ sudo i2cdetect -y 1
      0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: -- 04 -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- -- --
pi@raspberrypi: ~/Desktop/GrovePi/Script $
```

If you can see a “04” in the output, this means the Raspberry Pi is able to detect the GrovePi

10. To test the Grove Pi, connect a Grove LED to port D4 and run the blink example

```
cd GrovePi/Software/Python
```

python grovepi_blink.py



```
pi@raspberrypi: ~
login as: pi
pi@169.254.127.191's password:
Linux raspberrypi 3.6.11+ #538 PREEMPT Fri Aug 30 20:42:08 BST 2013 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Dec 30 17:34:05 2013 from 169.254.127.190
pi@raspberrypi ~ $ sudo su
root@raspberrypi:/home/pi# ls
Desktop ocr_pi.png python_games
root@raspberrypi:/home/pi# cd Desktop/GrovePi/Software/Python/
root@raspberrypi:/home/pi/Desktop/GrovePi/Software/Python# ls
grovepi_blink.py      grovepi_oled_test_i2c.py  mail_file.py  weat_twit.py
grovepi_i2c_ledFade.py grovepi.py                pot_led.py
grovepi_i2c.py         grovepi.pyc              README.md
root@raspberrypi:/home/pi/Desktop/GrovePi/Software/Python# python grovepi_blink.py
```

If everything is installed correctly, the LED should start

Unit 6: Raspberry Pi and GrovePi+

Learning Outcomes:

- Interface grovePi+ with Raspberry Pi
- Able to install grovePi+ sensors to Raspberry Pi
- Able to interface various sensors and actuators

6.1 PRACTICAL: Interface Grove Button with Raspberry Pi

Hardware

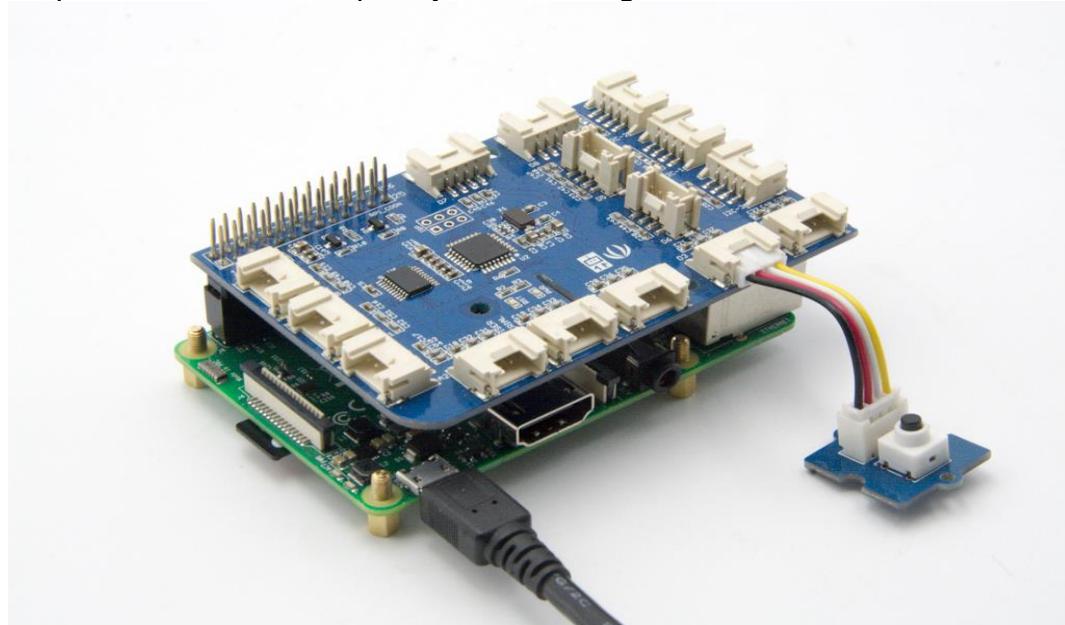
Step 1. Prepare the below stuffs:



Step 2. Plug the GrovePi_Plus into Raspberry.

Step 3. Connect Grove-Button to D3 port of GrovePi_Plus.

Step 4. Connect the Raspberry to PC through USB cable.



Software

Step 1. Follow [Setting Software](#) to configure the development environment.

Step 2. Git clone the Github repository.

cd ~

git clone https://github.com/DexterInd/GrovePi.git

Step 3. Execute below commands.

cd ~/GrovePi/Software/Python

python3 grove_button.py

Here is the grove_button.py code.

```
import time
import grovepi
```

```
# Connect the Grove Button to digital port D3
```

```
# SIG,NC,VCC,GND
```

```
button = 3
```

```
grovepi.pinMode(button,"INPUT")
```

```
while True:
```

```
    try:
```

```
        print(grovepi.digitalRead(button))
```

```
        time.sleep(.5)
```

```
    except IOError:
```

```
        print ("Error")
```

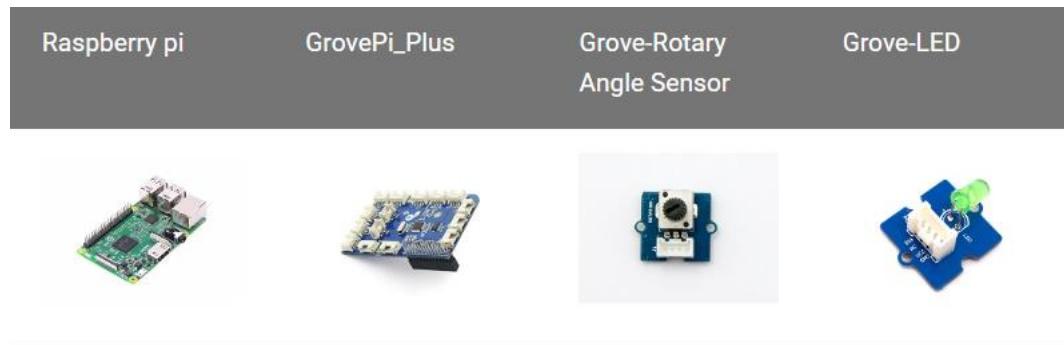
Step 4. We will see the button on and off.

```
pi@raspberrypi:~/GrovePi/Software/Python $ python3 grove_button.py
0
1
1
1
0
0
```

6.2 PRACTICAL: Interface Grove Rotary Angle sensor with Raspberry Pi

Hardware setup

Step 1: Prepare below stuff

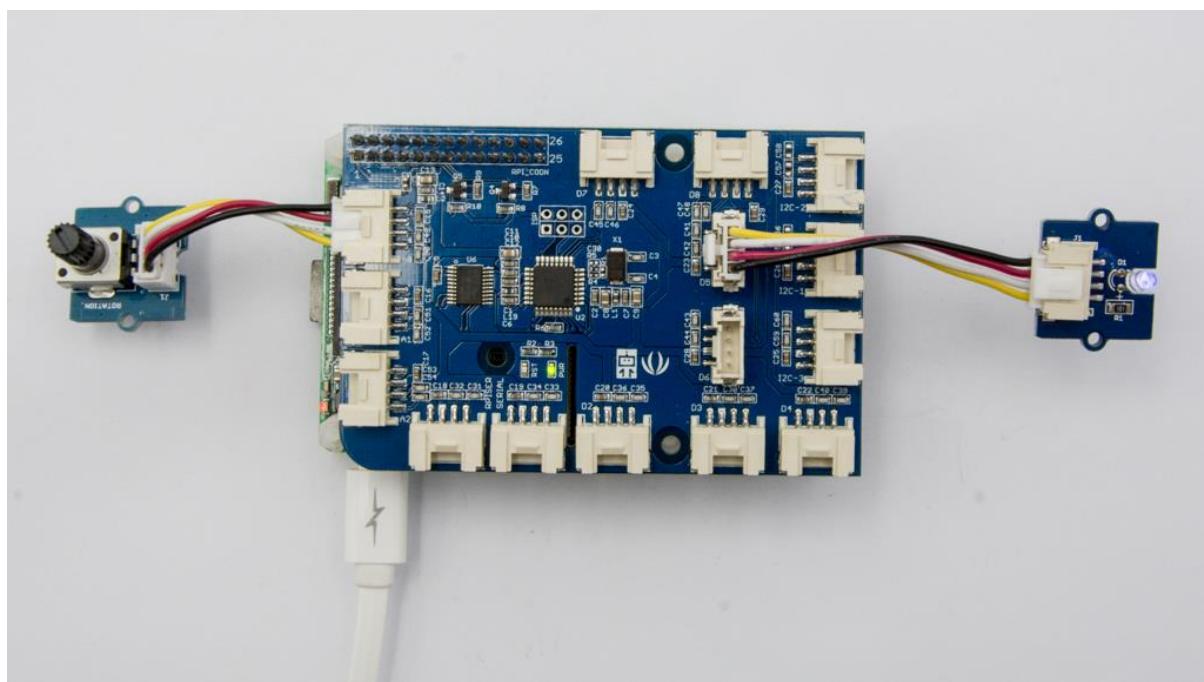


Step 2. Plug the GrovePi_Plus into Raspberry.

Step 3. Connect Grove-Rotary Angle Sensor to A0 port of GrovePi_Plus.

Step 4. Connect Grove-LED to D5 port of GrovePi_Plus.

Step 5. Connect the Raspberry to PC through USB cable.



software setup

Step 1. Follow Setting Software to configure the development environment.

Step 2. Git clone the Github repository.

cd ~

git clone https://github.com/DexterInd/GrovePi.git

Step3: Execute below commands to monitor the loudness.

cd ~/GrovePi/Software/Python

python3 grove_rotary_angle_sensor.py

Here is the grove_rotary_angle_sensor.py code.

Code:

```
import time
import grovepi
# Connect the Grove Rotary Angle Sensor to analog port A0
# SIG,NC,VCC,GND
potentiometer = 0
# Connect the LED to digital port D5
# SIG,NC,VCC,GND
led = 5

grovepi.pinMode(potentiometer,"INPUT")
grovepi.pinMode(led,"OUTPUT")
time.sleep(1)
# Reference voltage of ADC is 5v
adc_ref = 5
# Vcc of the grove interface is normally 5v
grove_vcc = 5
# Full value of the rotary angle is 300 degrees, as per it's specs (0 to 300)
full_angle = 300

while True:
    try:
        # Read sensor value from potentiometer
        sensor_value = grovepi.analogRead(potentiometer)
        # Calculate voltage
        voltage = round((float)(sensor_value) * adc_ref / 1023, 2)
        # Calculate rotation in degrees (0 to 300)
        degrees = round((voltage * full_angle) / grove_vcc, 2)
        # Calculate LED brightness (0 to 255) from degrees (0 to 300)
        brightness = int(degrees / full_angle * 255)
        # Give PWM output to LED
        grovepi.analogWrite(led,brightness)

        print("sensor_value = %d voltage = %.2f degrees = %.1f brightness = %d"
        %(sensor_value, voltage, degrees, brightness))
    except KeyboardInterrupt:
        grovepi.analogWrite(led,0)
        break
```

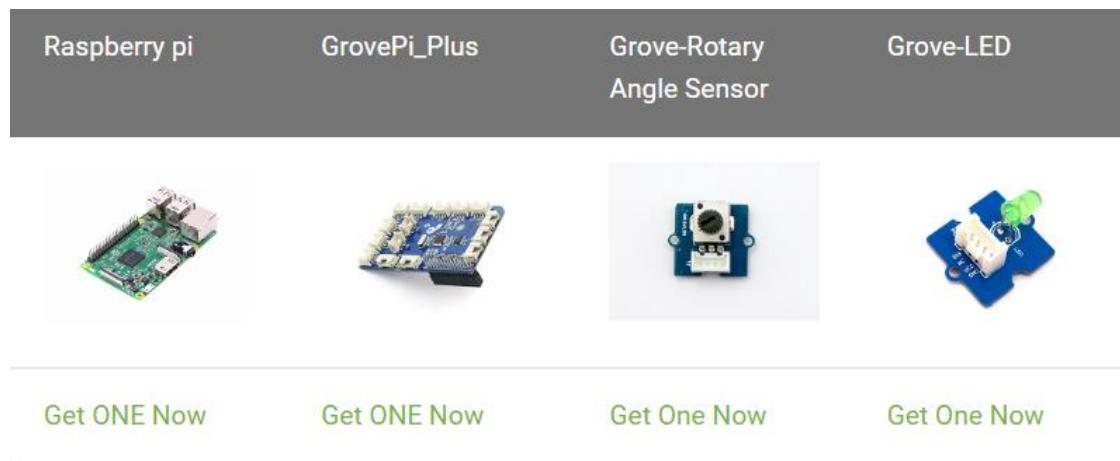
```
except IOError:  
    print ("Error")
```

Step 4: Adjust Grove-Rotary Angle Sensor and we will see the Grove-LED changes the brightness.

6.3 PRACTICAL: Interface Grove Temperature and humidity sensor (DHT22) with Raspberry Pi

Hardware

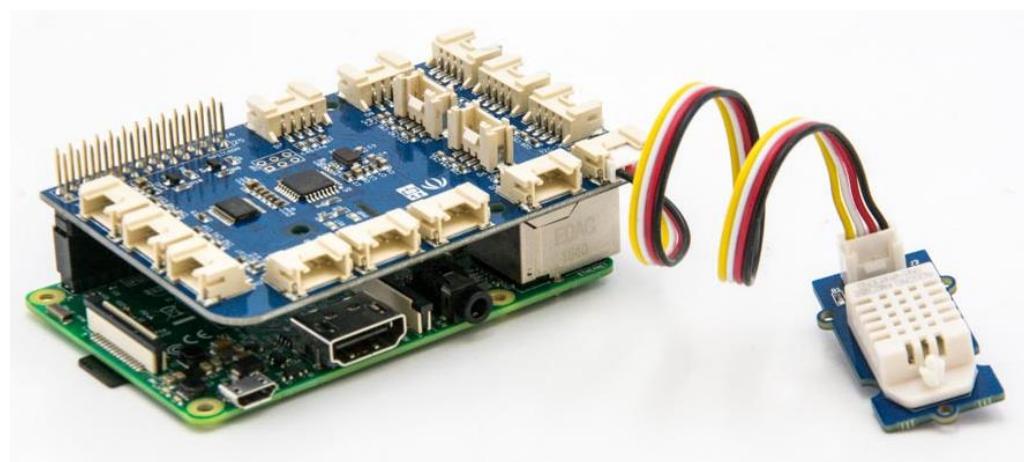
Step 1: Prepare below stuff



Step 2. Plug the GrovePi_Plus into Raspberry.

Step 3: Connect Grove - Temperature & Humidity Sensor Pro to D4 port of GrovePi+

Step 4. Connect the Raspberry to PC via USB cable.



Software

If this is the first time you use GrovePi, please do this part step by step. If you are an old friend with GrovePi, you can skip **Step1** and **Step2**.

Step 1. Setting Up The Software. In the command line, type the following commands:

```
sudo curl -kL dexterindustries.com/update_grovepi | bash
sudo reboot
cd /home/pi/Desktop
git clone https://github.com/DexterInd/GrovePi.git
```

Step 2. Follow Updating the Firmware to update the newest firmware of GrovePi.

Step 3. Configure Parameter

```
cd /home/pi/Desktop/GrovePi/Software/Python/
sudo nano grove_dht_pro.py
```

Change the default parameter [temp,humidity] = grovepi.dht(sensor,blue) into [temp,humidity] = grovepi.dht(sensor,white). Then the code should be like:

```
import grovepi
import math
# Connect the Grove Temperature & Humidity Sensor Pro to digital port D4
# This example uses the blue colored sensor.
# SIG,NC,VCC,GND
sensor = 4 # The Sensor goes on digital port 4.

# temp_humidity_sensor_type
# Grove Base Kit comes with the blue sensor.
blue = 0 # The Blue colored sensor.
white = 1 # The White colored sensor.

while True:
    try:
        # This example uses the blue colored sensor.
        # The first parameter is the port, the second parameter is the type of sensor.
        [temp,humidity] = grovepi.dht(sensor,white)
        if math.isnan(temp) == False and math.isnan(humidity) == False:
            print("temp = %.02f C humidity =%.02f%%"(temp, humidity))

    except IOError:
        print ("Error")
```

Then tap **Ctrl + X** to quit nano. Tap **Y** to save the change.

Step 4. Run the following command to get the result.

```
sudo python3 grove_dht_pro.py
```

Result

```
pi@raspberrypi:~/GrovePi/Software/Python $ sudo python3 grove_dht_pro.py
temp = 22.90 C humidity =42.30%
```

6.4 PRACTICAL: Interface Grove Ultrasonic sensor with Raspberry Pi



This Grove - Ultrasonic ranger is a non-contact distance measurement module which works at 40KHz. When we provide a pulse trigger signal with more than 10uS through signal pin, the Grove_Ultrasonic_Ranger will issue 8 cycles of 40kHz cycle level and detect the echo. The pulse width of the echo signal is proportional to the measured distance. Here is the formula: Distance = echo signal high time * Sound speed (340M/S)/2. Grove_Ultrasonic_Ranger's trig and echo signal share 1 SIG pin.

Hardware

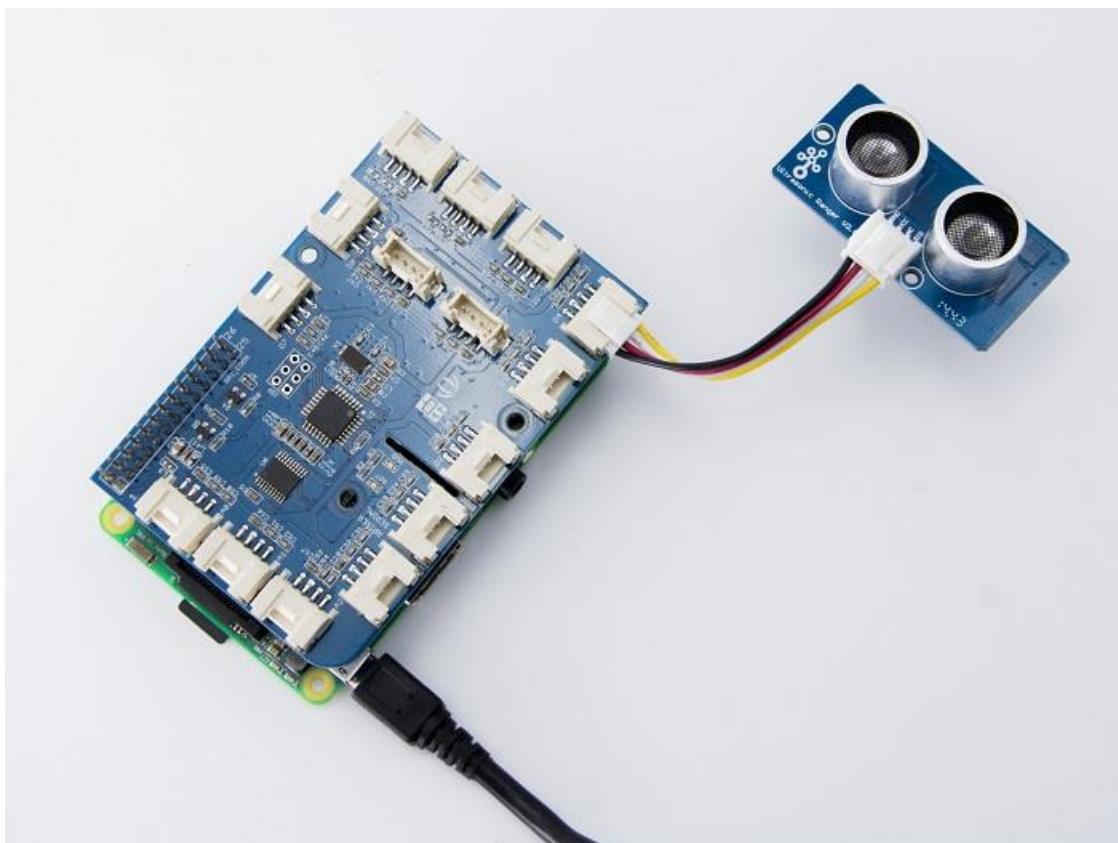
Step 1: Prepare the below Hardware



Step 2: Plug the GrovePi_Plus into Raspberry.

Step 3: Connect Grove-Ultrasonic ranger to **D4** port of GrovePi_Plus.

Step 4: Connect the Raspberry to PC through USB cable.



Software

Step 1: Follow [Setting Software](#) to configure the development environment.

Step 2: Git clone the Github repository.

cd ~

git clone https://github.com/DexterInd/GrovePi.git

Step 3: Execute below commands to use the ultrasonic_ranger to measure the distance.

```
cd ~/GrovePi/Software/Python  
python3 grove_ultrasonic.py
```

Here is the grove_ultrasonic.py code.

GrovePi + Grove Ultrasonic Ranger

```
from grovepi import *
```

```
# Connect the Grove Ultrasonic Ranger to digital port D4  
# SIG,NC,VCC,GND
```

ultrasonic_ranger = 4

while True:

try:

```
# Read distance value from Ultrasonic  
print ultrasonicRead(ultrasonic_ranger)
```

```
except TypeError:
```

```
print "Error"
```

```
except IOError:
```

```
print "Error"
```

Step 4: We will see the distance display on terminal as below.

6.5 PRACTICAL: Interface Grove Relay with Raspberry Pi



The Grove-Relay module is a digital normally-open switch. Through it, you can control circuit of high voltage with low voltage, say 5V on the controller. There is an indicator LED on the board, which will light up when the controlled terminals get closed.

Hardware

Step 1: Prepare the below Hardware

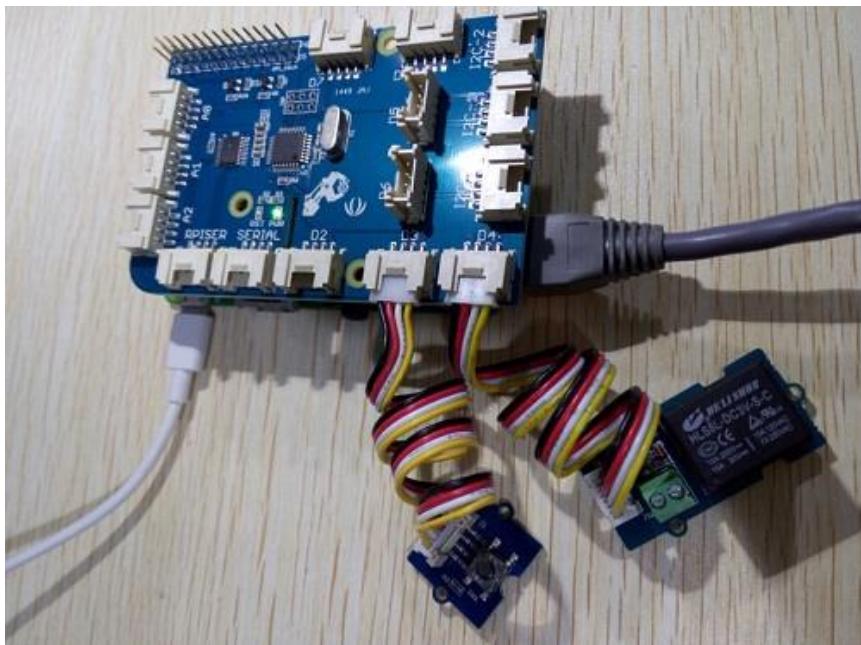


Step 2. Plug the GrovePi_Plus into Raspberry.

Step 3. Connect the Grove-Relay to D4 port of GrovePi_Plus.

Step 4. Connect the Grove-Button to D3 port of GrovePi_Plus.

Step 5. Connect the Raspberry to PC via USB cable.



Software

Step 1. Setting Up The Software. In the command line, type the following commands:

```
sudo curl -kL dexterindustries.com/update_grovepi | bash
sudo reboot
cd /home/pi/Desktop
git clone https://github.com/DexterInd/GrovePi.git
```

Step 2. Follow Updating the Firmware to update the latest firmware of GrovePi.

Step 3. Run the following command to get the result.

```
cd /home/pi/Desktop/GrovePi/Software/Python/
sudo python3 grove_switch_relay.py
```

If you want to check the code, you can use the following command:

```
sudo nano grove_switch_relay.py
```

Code

```
# Raspberry Pi + Grove Switch + Grove Relay

import time
import grovepi
# Connect the Grove Switch to digital port D3
# SIG,NC,VCC,GND
```

```
switch = 3
# Connect the Grove Relay to digital port D4
# SIG,NC,VCC,GND

relay = 4
grovepi.pinMode(switch,"INPUT")
grovepi.pinMode(relay,"OUTPUT")
while True:
    try:
        if grovepi.digitalRead(switch):
            grovepi.digitalWrite(relay,1)
        else:
            grovepi.digitalWrite(relay,0)
            time.sleep(.05)
    except KeyboardInterrupt:
        grovepi.digitalWrite(relay,0)
        break
    except IOError:
        print "Error"
```

Module – III

Machine Learning

Unit 1: Data Manipulation with Pandas

Learning Outcomes:

- Understand the concept and features of Pandas library
- Able manipulate data using Pandas library
- Able to implement data analysis solution using Pandas library

1.1 Data Manipulation with Pandas

Pandas is a fast, powerful, flexible and easy to use open-source data analysis and manipulation tool. Pandas is a newer package built on top of NumPy, and provides an efficient implementation of a **DataFrame**.

DataFrames are essentially multidimensional arrays with attached row and column labels, and often with heterogeneous types and/or missing data. As well as offering a convenient storage interface for labeled data, Pandas implements a number of powerful data operations familiar to users of both database frameworks and spreadsheet programs.

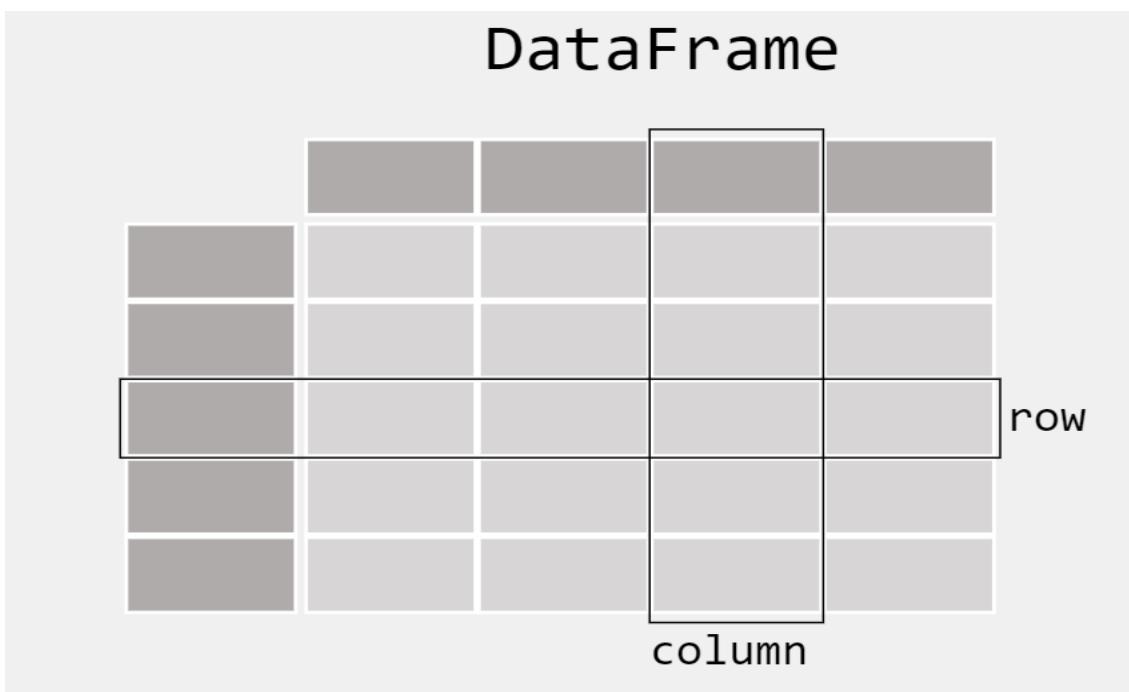


Image: dataframe

Reference: https://pandas.pydata.org/docs/getting_started/index.html

1.2 Introducing Pandas Objects

Pandas objects can be thought of as enhanced versions of NumPy structured arrays in which the rows and columns are identified with labels rather than simple integer indices

There are three fundamental Pandas data structures:

- Series,
- DataFrame
- Index.

1.3 The Pandas Series Object

A Pandas Series is a one-dimensional array of indexed data. It can be created from a list or array as follows:

```
In [1]: #Pandas Series Object
import numpy as np
import pandas as pd
data = pd.Series([0.25, 0.5, 0.75, 1.0])
data

Out[1]: 0    0.25
        1    0.50
        2    0.75
        3    1.00
       dtype: float64
```

In []:

As we see in the output, the Series wraps both a sequence of values and a sequence of indices, which we can access with the values and index attributes. The values are simply a familiar NumPy array:

```
In [2]: data.values

Out[2]: array([0.25, 0.5 , 0.75, 1. ])
```

Like with a NumPy array, data can be accessed by the associated index via the familiar Python square-bracket notation:

```
In [3]: data[1]  
  
Out[3]: 0.5  
  
In [4]: data[1:3]  
  
Out[4]: 1    0.50  
2    0.75  
dtype: float64
```

1.4 The Pandas DataFrame Object

The next fundamental structure in Pandas is the DataFrame. Like the Series object discussed in the previous section, the DataFrame can be thought of either as a generalization of a NumPy array, or as a specialization of a Python dictionary.

DataFrame as a generalized NumPy array

If a Series is an analog of a one-dimensional array with flexible indices, a DataFrame is an analog of a two-dimensional array with both flexible row indices and flexible column names.

Just as you might think of a two-dimensional array as an ordered sequence of aligned one-dimensional columns, you can think of a DataFrame as a sequence of aligned Series objects. Here, by "aligned" we mean that they share the same index.

To demonstrate this, let's first construct two new Series listing of the area and the population of say five states of USA.:

```
In [5]: area_dict = {'California': 423967, 'Texas': 695662, 'New York': 141297,  
                  'Florida': 170312, 'Illinois': 149995}  
area = pd.Series(area_dict)  
area  
  
Out[5]: California    423967  
        Texas      695662  
        New York   141297  
        Florida    170312  
        Illinois   149995  
       dtype: int64  
  
In [7]: population_dict = {'California': 38332521, 'Texas': 26448193, 'New York': 19651127,  
                           'Florida': 19552860, 'Illinois': 12882135}  
population = pd.Series(population_dict)  
population  
  
Out[7]: California    38332521  
        Texas      26448193  
        New York   19651127  
        Florida    19552860  
        Illinois   12882135  
       dtype: int64
```

Now that we have the area and the population Series from before, we can use a dictionary to construct a single two-dimensional object containing this information:

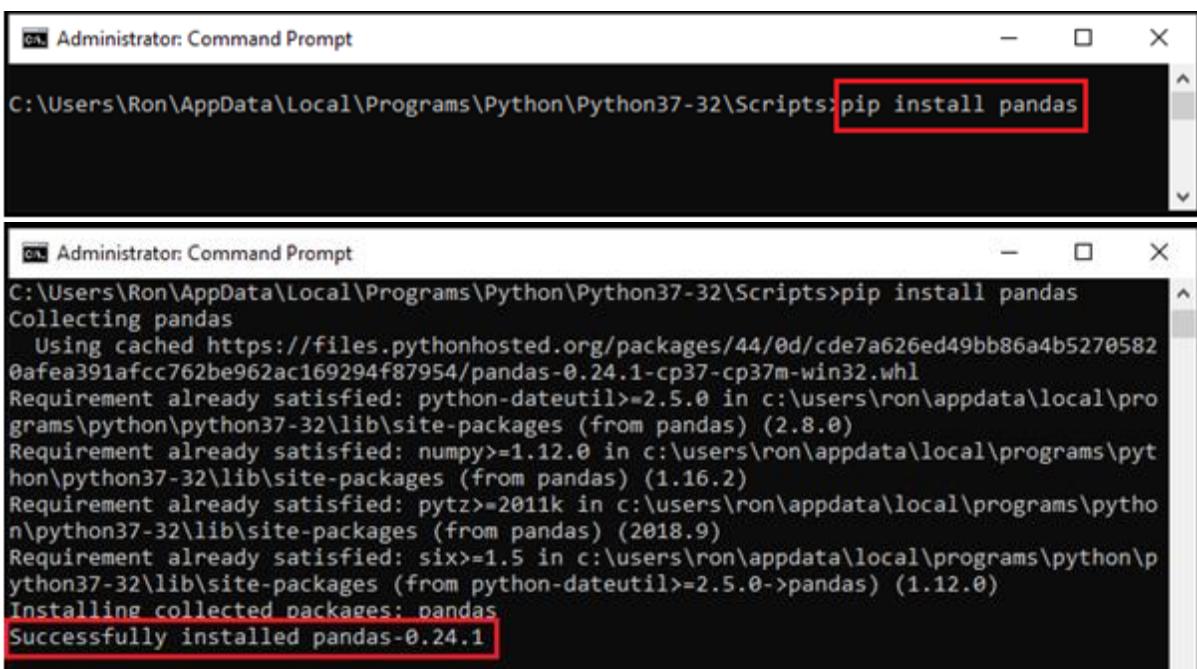
```
In [8]: states = pd.DataFrame({'population': population,
                                'area': area})
states
```

Out[8]:

	population	area
California	38332521	423967
Texas	26448193	695662
New York	19651127	141297
Florida	19552860	170312
Illinois	12882135	149995

Installing Pandas from pip

`pip install pandas`



The image shows two screenshots of the Windows Command Prompt. Both windows are titled "Administrator: Command Prompt". The top window shows the command `C:\Users\Ron\AppData\Local\Programs\Python\Python37-32\Scripts>pip install pandas` being typed. The bottom window shows the output of the command, which includes the message "Successfully installed pandas-0.24.1".

```
C:\Users\Ron\AppData\Local\Programs\Python\Python37-32\Scripts>pip install pandas
Collecting pandas
  Using cached https://files.pythonhosted.org/packages/44/0d/cde7a626ed49bb86a4b5270582
  0afea391afcc762be962ac169294f87954/pandas-0.24.1-cp37-cp37m-win32.whl
Requirement already satisfied: python-dateutil>=2.5.0 in c:\users\ron\appdata\local\pro
grams\python\python37-32\lib\site-packages (from pandas) (2.8.0)
Requirement already satisfied: numpy>=1.12.0 in c:\users\ron\appdata\local\programs\pyt
hon\python37-32\lib\site-packages (from pandas) (1.16.2)
Requirement already satisfied: pytz>=2011k in c:\users\ron\appdata\local\programs\pytho
n\python37-32\lib\site-packages (from pandas) (2018.9)
Requirement already satisfied: six>=1.5 in c:\users\ron\appdata\local\programs\pytho
n\python37-32\lib\site-packages (from python-dateutil>=2.5.0->pandas) (1.12.0)
Installing collected packages: pandas
Successfully installed pandas-0.24.1
```

You can quickly check if the package was successfully installed in Python, by opening the Python IDLE and then running the command “import pandas”

If no errors appear, then the package was successfully installed.

For more details on Python packages you can refer <https://packaging.python.org/>

1.5 Creating Series from simple datatypes

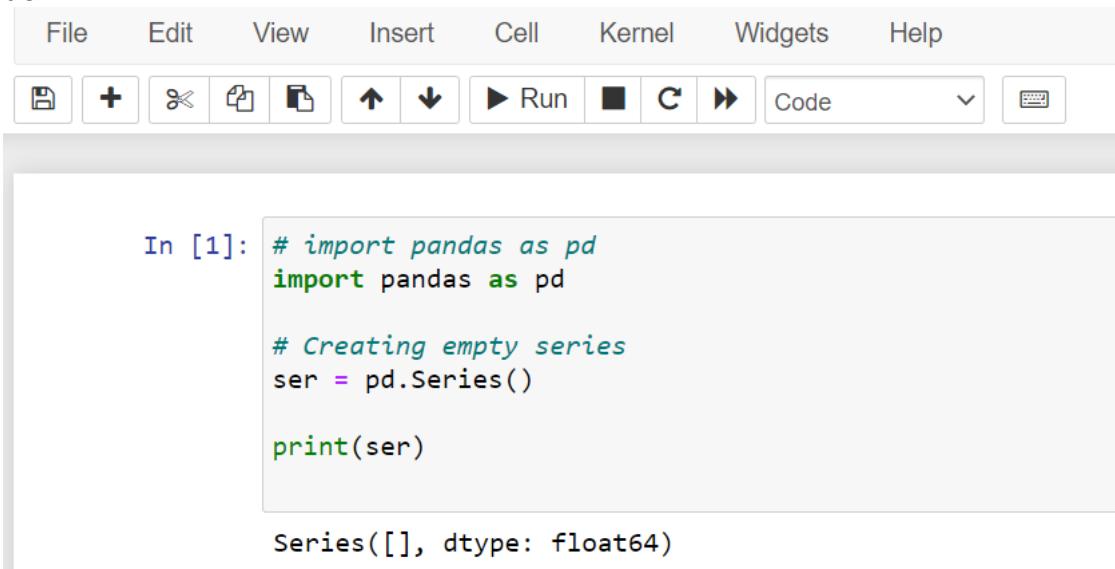
Creating a Pandas Series

The Pandas Series can be defined as a one-dimensional array that is capable of storing various data types. We can easily convert the list, tuple, and dictionary into series using "series' method. The row labels of series are called the index. A Series cannot contain multiple columns. It has the following parameter:

- data: It can be any list, dictionary, or scalar value.
- index: The value of the index should be unique and hashable. It must be of the same length as data. If we do not pass any index, default np.arange(n) will be used.
- dtype: It refers to the data type of series.
- copy: It is used for copying the data.

Create an Empty Series:

We can easily create an empty series in Pandas which means it will not have any value.



In [1]: `# import pandas as pd
import pandas as pd

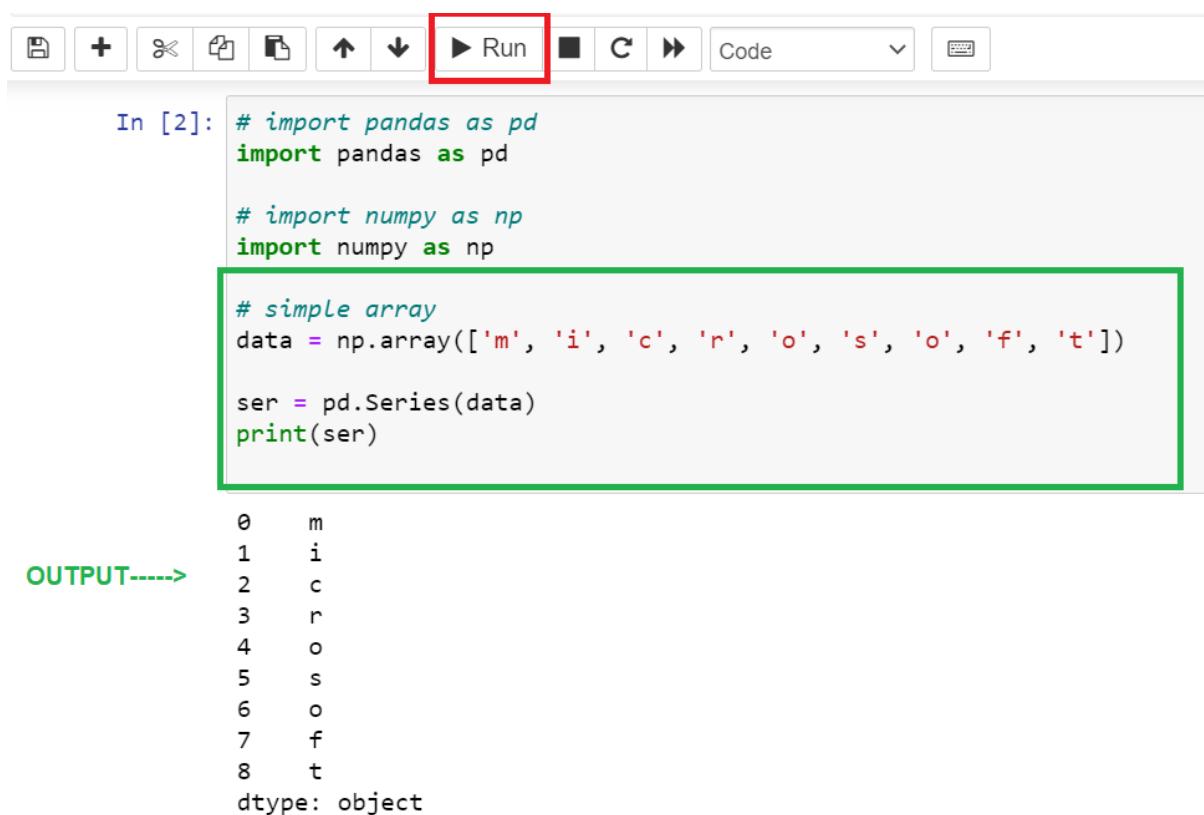
Creating empty series
ser = pd.Series()

print(ser)`

Ser[[], dtype: float64]

Creating a series from array:

In order to create a series from array, we have to import a numpy module and have to use array() function.



The screenshot shows a Jupyter Notebook interface. At the top, there's a toolbar with various icons: file, new, cell, run, cell, up, down, etc. The 'Run' button is highlighted with a red box. Below the toolbar, the code cell contains the following Python code:

```
In [2]: # import pandas as pd
import pandas as pd

# import numpy as np
import numpy as np

# simple array
data = np.array(['m', 'i', 'c', 'r', 'o', 's', 'o', 'f', 't'])

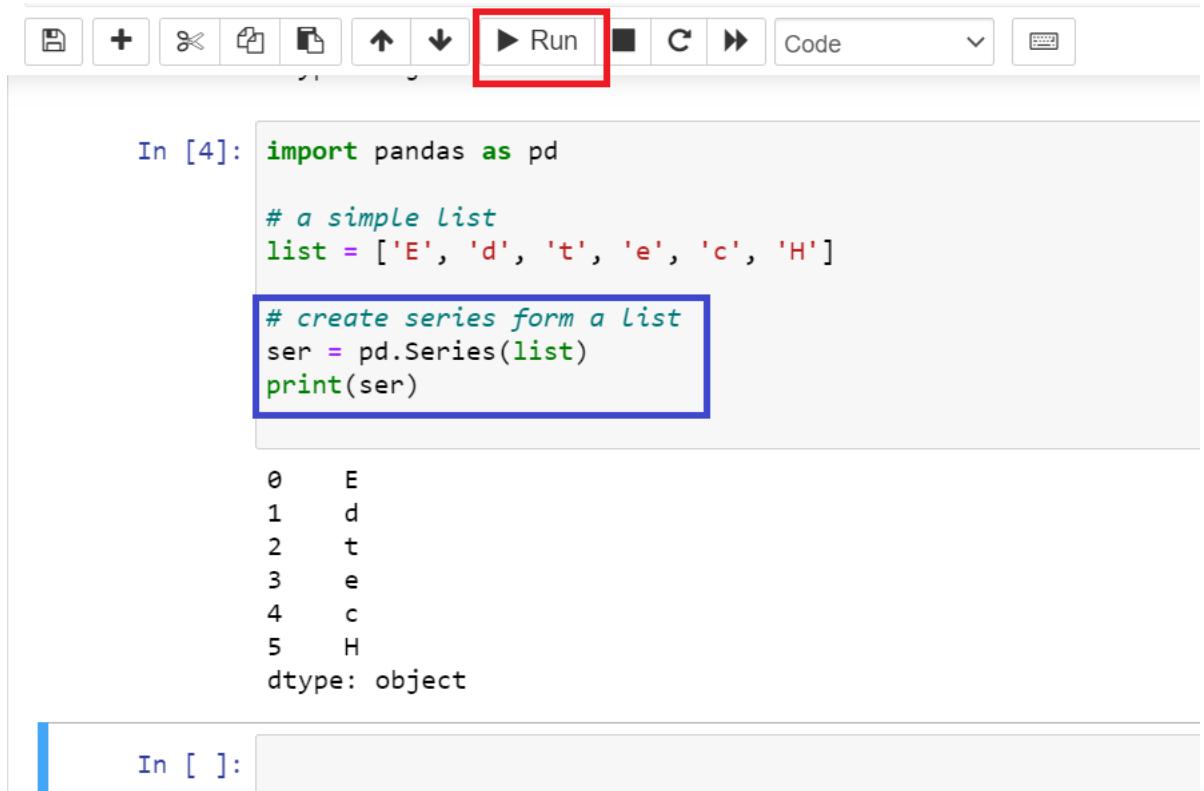
ser = pd.Series(data)
print(ser)
```

The output cell, labeled 'OUTPUT---->', displays the resulting Series object:

```
0    m
1    i
2    c
3    r
4    o
5    s
6    o
7    f
8    t
dtype: object
```

Creating a series from Lists:

In order to create a series from list, we have to first create a list after that we can create a series from list.



The screenshot shows a Jupyter Notebook interface. At the top, there's a toolbar with various icons: file, new, cell, run, cell, up, down, etc. The 'Run' button is highlighted with a red box. Below the toolbar, the code cell contains the following Python code:

```
In [4]: import pandas as pd

# a simple list
list = ['E', 'd', 't', 'e', 'c', 'H']

# create series form a List
ser = pd.Series(list)
print(ser)
```

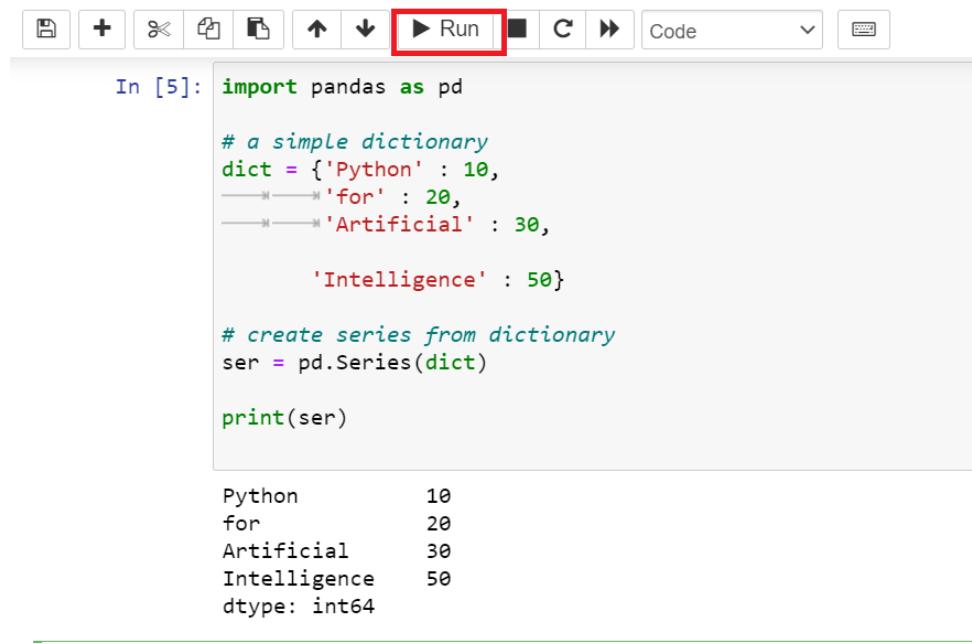
The output cell displays the resulting Series object:

```
0    E
1    d
2    t
3    e
4    c
5    H
dtype: object
```

Below the output, there's a partially visible input cell starting with 'In []:'.

Creating a series from Dictionary:

In order to create a series from dictionary, we have to first create a dictionary after that we can make a series using dictionary. Dictionary key are used to construct a index.



The screenshot shows a Jupyter Notebook interface. At the top, there's a toolbar with various icons: file, new, cell, run, etc. The 'Run' button is highlighted with a red box. Below the toolbar, the code cell content is:

```
In [5]: import pandas as pd

# a simple dictionary
dict = {'Python' : 10,
         'for' : 20,
         'Artificial' : 30,
         'Intelligence' : 50}

# create series from dictionary
ser = pd.Series(dict)

print(ser)
```

After running the cell, the output is displayed below:

```
Python      10
for        20
Artificial    30
Intelligence  50
dtype: int64
```

1.6 Data Storage Formats in Pandas

The different data storage formats available to be manipulated by Pandas library are text, binary and SQL. Below is a table containing available 'readers' and 'writers' functions of the pandas I/O API set with data format and description. [6.5]

Format Type	Data Description	Reader	Writer
text	CSV	<code>read_csv</code>	<code>to_csv</code>
text	Fixed-Width Text File	<code>read_fwf</code>	
text	JSON	<code>read_json</code>	<code>to_json</code>
text	HTML	<code>read_html</code>	<code>to_html</code>
text	Local clipboard	<code>read_clipboard</code>	<code>to_clipboard</code>
binary	MS Excel	<code>read_excel</code>	<code>to_excel</code>
binary	OpenDocument	<code>read_excel</code>	
binary	HDF5 Format	<code>read_hdf</code>	<code>to_hdf</code>
binary	Feather Format	<code>read_feather</code>	<code>to_feather</code>
binary	Parquet Format	<code>read_parquet</code>	<code>to_parquet</code>
binary	ORC Format	<code>read_orc</code>	
binary	Msgpack	<code>read_msgpack</code>	<code>to_msgpack</code>
binary	Stata	<code>read_stata</code>	<code>to_stata</code>
binary	SAS	<code>read_sas</code>	
binary	SPSS	<code>read_spss</code>	

binary	Python Pickle Format	<code>read_pickle</code>	<code>to_pickle</code>
SQL	SQL	<code>read_sql</code>	<code>to_sql</code>
SQL	Google BigQuery	<code>read_gbq</code>	<code>to_gbq</code>

Figure: “readers” and “writers” functions in pandas
Reference: https://pandas.pydata.org/pandas-docs/stable/user_guide/io.html

CSV file and JSON file

What is CSV file?

A CSV is a comma-separated values file, which allows data to be saved in a tabular format. CSV files can be used with many spreadsheets program, such as Microsoft Excel or Google Spreadsheets. They differ from other spreadsheet file types because you can only have a single sheet in a file, they cannot save cell, column, or row. Also, you cannot save formulas in this format.

Why are .CSV files used?

These files serve a number of different business purposes. Take, for instance, they help companies export a high volume of data to a more concentrated database.

They also serve two other primary business functions:

- CSV files are plain-text files, making them easier for the website developer to create
- Since they're plain text, they're easier to import into a spreadsheet or another storage database, regardless of the specific software you're using.
- To better organize large amounts of data.

How do I save CSV files?

Saving CSV files is relatively easy, you just need to know where to change the file type. Under the "File name" section in the "Save As" tab, you can select "Save as type" and change it to "CSV (Comma delimited) (*.csv)". Once that option is selected, you are on your way to quicker and easier data organization. This should be the same for both Apple and Microsoft operating systems.

What is a JSON file?

A JSON file is a file that stores simple data structures and objects in JavaScript Object Notation (JSON) format, which is a standard data interchange format. It is primarily used for transmitting data between a web application and a server. JSON files are lightweight, text-based, human-readable, and can be edited using a text editor.

How do I open a JSON file?

Because JSON files are plain text files, you can open them in any text editor, including:

- Microsoft Notepad (Windows)
- AppleTextEdit (Mac)
- Vim (Linux)
- GitHub Atom (cross-platform)

You can also open a JSON file in the Google Chrome and Mozilla Firefox web browsers by dragging and dropping the file into your browser window.

Structures of JSON

JSON supports two widely used (amongst programming languages) data structures.

- A collection of name/value pairs. Different programming languages support this data structure in different names. Like object, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In various programming languages, it is called as array, vector, list, or sequence.

Since data structure supported by JSON is also supported by most of the modern programming languages, it makes JSON a very useful data-interchange format.

Syntax:

```
{ string : value, .....}
```

Explanation of Syntax

An object starts and ends with '{' and '}'. Between them, a number of string value pairs can reside. String and value is separated by a ':' and if there are more than one string value pairs, they are separated by ','.

Example

```
{
  "firstName": "John",
  "lastName": "Maxwell",
  "age": 40,
  "email": "john@example.com"
}
```

In JSON, objects can nest arrays (starts and ends with '[' and ']') within it. The following example shows that.

```
{
  "Students": [
    {
      "Name": "Amit Goenka",
      "Major": "Physics"
    },
    {
      "Name": "Smita Pallod",
      "Major": "Chemistry"
    },
    {
      "Name": "Rajeev Sen",
      "Major": "Mathematics"
    }
  ]
}
```

Array:

Syntax:

[value,]

Explanation of Syntax:

An Array starts and ends with '[' and ']'. Between them, a number of values can reside. If there are more than one values, they are separated by ','.

Example

[100, 200, 300, 400]

If the JSON data describes an array, and each element of that array is an object.

```
[  
  {  
    "name": "John Maxwell",  
    "email": "john@example.com"  
  },  
  {  
    "name": "Dale Carnegie",  
    "email": "dale@example.com"  
  }  
]
```

Remember that even arrays can also be nested within an object. The following shows that.

```
{  
  "firstName": "John",  
  "lastName": "Maxwell",  
  "age": 40,  
  "address":  
    {  
      "streetAddress": "144 J B Queens Road",  
      "city": "Dallas",  
      "state": "Washington",  
      "postalCode": "75001"  
    },  
  "phoneNumber":  
    [  
      {  
        "type": "personal",  
        "number": "(214)5096995"  
      },  
      {  
        "type": "fax",  
        "number": "13235551234"  
      }  
    ]  
}
```

}]

Value

Syntax:

String || Number || Object || Array || TRUE || FALSE || NULL

A value can be a string, a number, an object, an Array, a Boolean value (i.e. true or false) or Null. This structure can be nested.

String

A string is a sequence of zero or more Unicode characters, enclosed by double quotes, using backslash escapes. A character is represented as a single character string, similar to a C or Java string.

The following table shows supported string types.

String Types	Description
"	A double quotation mark.
\	Reverse Solidus
/	Solidus
b	Backspace
f	form feed
n	newline
r	Carriage return
t	Horizontal tab
u	Four hexadecimal digits

Number

The following table shows supported number types.

Number Types	Description
Integer	Positive or negative Digits.1-9 and 0.
Fraction	Fractions like .8.
Exponent	e, e+, e-, E, E+, E-

Whitespace

Whitespace can be placed between any pair of supported data-types.

1.7 Reading data from files

Load CSV files to Python Pandas

The basic process of loading data from a CSV file into a Pandas DataFrame is achieved using the “read_csv” function in Pandas:

```
# Load the Pandas libraries with alias 'pd'
```

```
import pandas as pd
```

```
# Read data from file 'filename.csv'  
  
# (in the same directory that your python process is based)  
  
# Control delimiters, rows, column names with read_csv (see later)  
  
data = pd.read_csv("filename.csv")  
  
  
# Preview the first 5 lines of the loaded data  
  
data.head();
```

While this code seems simple, an understanding of three fundamental concepts is required to fully grasp and debug the operation of the data loading procedure if you run into issues:

File Extensions and File Types

The first step to working with comma-separated-value (CSV) files is understanding the concept of file types and file extensions.

- Data is stored on your computer in individual “files”, or containers, each with a different name.
- Each file contains data of different types – the internals of a Word document is quite different from the internals of an image.
- Computers determine how to read files using the “file extension”, that is the code that follows the dot (“.”) in the filename.
- So, a filename is typically in the form “<random name>.<file extension>”. Examples:
 - project1.DOCX – a Microsoft Word file called project1.
 - shanes_file.TXT – a simple text file called shanes_file
 - IMG_5673.JPG – An image file called IMG_5673.
 - Other well known file types and extensions include: XLSX: Excel, PDF: Portable Document Format, PNG – images, ZIP – compressed file format, GIF – animation, MPEG – video, MP3 – music etc. See a complete list of extensions here.
 1. A CSV file is a file with a “.csv” file extension, e.g. “data.csv”, “super_information.csv”. The “CSV” in this case lets the computer know that the data contained in the file is in “comma separated value” format.

Data Representation in CSV files

A “CSV” file, that is, a file with a “csv” filetype, is a basic text file. Any text editor such as NotePad on windows orTextEdit on Mac, can open a CSV file and show the contents. Sublime Text is a wonderful and multi-functional text editor option for any platform.

CSV is a standard form for storing tabular data in text format, where commas are used to separate the different columns, and newlines (carriage return / press enter) are used to separate rows. Typically, the first row in a CSV file contains the names of the columns for the data.

An example of a table data set and the corresponding CSV-format data is shown in the diagram below.

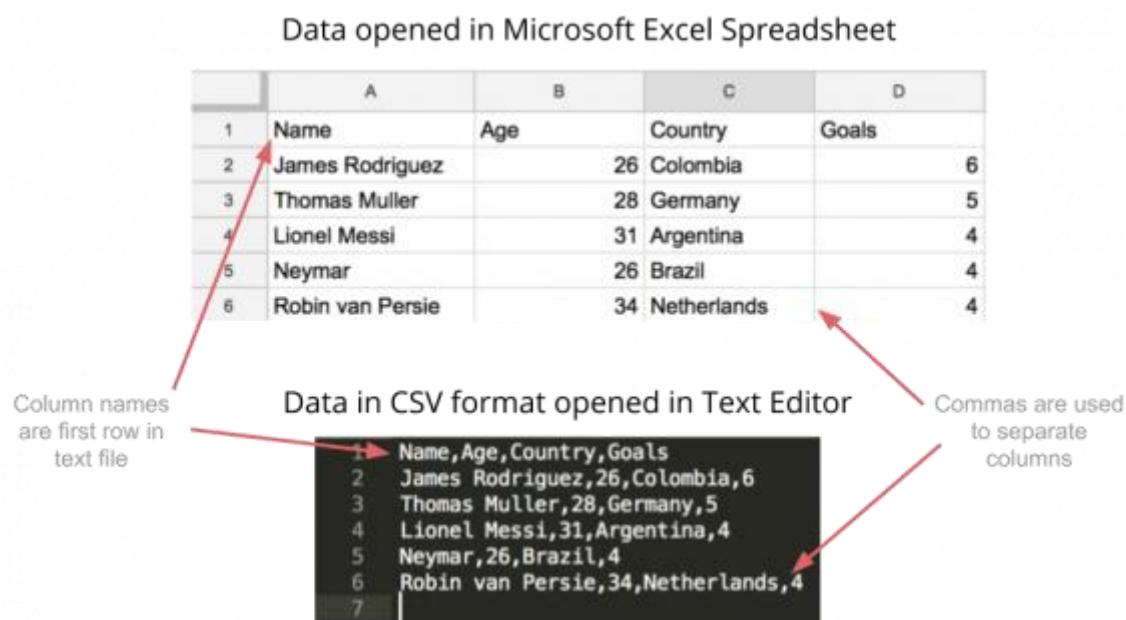


Figure: Comma-separated value files, or CSV files, are simple text files where commas and newlines are used to define tabular data in a structured way.

Reference: <https://www.shanelynn.ie/python-pandas-read-csv-load-data-from-csv-files>

Note that almost any tabular data can be stored in CSV format – the format is popular because of its simplicity and flexibility. You can create a text file in a text editor, save it with a .csv extension, and open that file in Excel or Google Sheets to see the table form.

Other Delimiters / Separators – TSV files

The comma separation scheme is by far the most popular method of storing tabular data in text files.

The choice of the ‘,’ comma character to delimit columns, however, is arbitrary, and can be substituted where needed. Popular alternatives include tab (“\t”) and semi-colon (“;”). Tab-separate files are known as TSV (Tab-Separated Value) files.

When loading data with Pandas, the `read_csv` function is used for reading any delimited text file, and by changing the delimiter using the `sep` parameter.

Delimiters in Text Fields – Quoteflag

One complication in creating CSV files is if you have commas, semicolons, or tabs actually in one of the text fields that you want to store. In this case, it's important to use a "quote character" in the CSV file to create these fields.

The quote character can be specified in Pandas.read_csv using the quotechar argument. By default (as with many systems), it's set as the standard quotation marks (""). Any commas (or other delimiters as demonstrated below) that occur between two quote characters will be ignored as column separators.

In the example shown, a semicolon-delimited file, with quotation marks as a quotechar is loaded into Pandas, and shown in Excel. The use of the quotechar allows the "NickName" column to contain semicolons without being split into more columns.

Semi-colon separated data in text file

```
CustomerId; CustomerName; Address; Age; NickNames
1;Shane Lynn;Dublin, Ireland; 30;"Shaneo;Lynno;Slynn"
2;Johnny Ives;London, United Kingdom;40;"Johnson;Big John;Ivy"
3;Simon Smith;Rue de Rue, Paris, France;50;"Frenchy;Smitho;Hammer"
4;Ronald Mc Donald;The big Farm, McDonalds Farm; 60;"Ronnie;Maccie;Donnie"
5;Jonathan Swift;Celbridge Abbey, Celbridge, Ireland;70;"Jonno;Speedy;Swifter"
```

Semicolons (;) are used here to separate columns

Semi-colon separated data loaded into Excel

A	B	C	D	E
1	CustomerId	CustomerName	Address	Age NickNames
2	1	Shane Lynn	Dublin, Ireland	30 Shaneo;Lynno;Slynn
3	2	Johnny Ives	London, United Kingdom	40 Johnson;Big John;Ivy
4	3	Simon Smith	Rue de Rue, Paris, France	50 Frenchy;Smitho;Hammer
5	4	Ronald Mc Donald	The big Farm, McDonalds Farm	60 Ronnie;Maccie;Donnie
6	5	Jonathan Swift	Celbridge Abbey, Celbridge, Ireland	70 Jonno;Speedy;Swifter

The data in the last column contains semicolons, so the quotation character is used to 'quote' the values

Semi-colon separated data loaded to Pandas

```
pd.read_csv('test_delimited.csv', sep=';', quotechar='"', encoding='utf8')
```

The 'sep' argument tells Pandas how to break up data into columns

Specify the quotechar if necessary - the default is "".

Customerid	CustomerName	Address	Age	NickNames
0	1	Shane Lynn	Dublin, Ireland	30 Shaneo;Lynno;Slynn
1	2	Johnny Ives	London, United Kingdom	40 Johnson;Big John;Ivy
2	3	Simon Smith	Rue de Rue, Paris, France	50 Frenchy;Smitho;Hammer
3	4	Ronald Mc Donald	The big Farm, McDonalds Farm	60 Ronnie;Maccie;Donnie
4	5	Jonathan Swift	Celbridge Abbey, Celbridge, Ireland	70 Jonno;Speedy;Swifter

Reference: <https://www.shanelynn.ie/python-pandas-read-csv-load-data-from-csv-files>

Python – Paths, Folders, Files

When you specify a filename to Pandas.read_csv, Python will look in your "current working directory". Your working directory is typically the directory that you started from your Python process or Jupyter notebook.

```
In [26]: pd.read_csv('file_not_in_right_place.csv')

FileNotFoundError                         Traceback (most recent call last)
<ipython-input-26-f3609a36b9ff> in <module>()
----> 1 pd.read_csv('file_not_in_right_place.csv')

~/Envs/analysis/lib/python3.6/site-packages/pandas/io/parsers.py in parser_f(filepath_or_buffer, sep, delimiter, header, names, index_col, usecols, squeeze, prefix, mangle_dupe_cols, dtype, engine, converters, true_values, false_value, skipinitialspace, skiprows, na_values, keep_default_na, na_filter, verbose, skip_blank_lines, parse_dates, infer_datetime_format, keep_date_col, date_parser, dayfirst, iterator, chunksize, compression, thousands, decimal, lineterminator, quotechar, quoting, escapechar, comment, encoding, dialect, tupleize_cols, error_bad_lines, warn_bad_lines, skipfooter, doublequote, delim_whitespace, low_memory, memory_map, float_precision)
    676     skip_blank_lines=skip_blank_lines)
    677 
--> 678     return _read(filepath_or_buffer, kwds)
    679 
    680     parser_f._name_ = name

~/Envs/analysis/lib/python3.6/site-packages/pandas/io/parsers.py in _read(filepath_or_buffer, kwds)
    438 
    439     # Create the parser.
--> 440     parser = TextFileReader(filepath_or_buffer, **kwds)
    441 
    442     if chunksize or iterator:

~/Envs/analysis/lib/python3.6/site-packages/pandas/io/parsers.py in __init__(self, f, engine, **kwds)
    785         self.options['has_index_names'] = kwds['has_index_names']
    786 
--> 787         self._make_engine(self.engine)
    788 
    789     def close(self):
```

Reference: <https://www.shanelynn.ie/python-pandas-read-csv-load-data-from-csv-files/>

Finding your Python Path

Your Python path can be displayed using the built-in “os” module. The OS module is for operating system dependent functionality into Python programs and scripts.

To find your current working directory, the function required is `os.getcwd()`. The `os.listdir()` function can be used to display all files in a directory, which is a good check to see if the CSV file you are loading is in the directory as expected.

```
# Find out your current working directory
```

```
import os
print(os.getcwd())
```

```
# Out: /Users/shane/Documents/blog
```

```
# Display all of the files found in your current working directory
```

```
print(os.listdir(os.getcwd()))
```

```
# Out: ['test_delimited.ssv', 'CSV Blog.ipynb', 'test_data.csv']
```

In the example above, my current working directory is in the ‘/Users/Shane/Document/blog’ directory. Any files that are places in this directory will be immediately available to the Python `file_open()` function or the Pandas `read_csv()` function.

Instead of moving the required data files to your working directory, you can also change your current working directory to the directory where the files reside using `os.chdir()`.

File Loading: Absolute and Relative Paths

When specifying file names to the `read_csv` function, you can supply both absolute or relative file paths.

- A relative path is the path to the file if you start from your current working directory. In relative paths, typically the file will be in a subdirectory of the working directory and the path will not start with a drive specifier, e.g.

(data/test_file.csv). The characters ‘..’ are used to move to a parent directory in a relative path.

- An absolute path is the complete path from the base of your file system to the file that you want to load, e.g. c:/Documents/Shane/data/test_file.csv. Absolute paths will start with a drive specifier (c:/ or d:/ in Windows, or ‘/’ in Mac or Linux)

It's recommended and preferred to use relative paths where possible in applications, because absolute paths are unlikely to work on different computers due to different directory structures.

```
In [32]: # Show Current working directory
import pandas as pd
import os
os.getcwd()

Out[32]: '/Users/shane/Documents/Blog/read_csv'

In [33]: os.listdir(os.getcwd())
Out[33]: ['.ipynb_checkpoints', 'test_delimited.ssv', 'CSV Blog.ipynb', 'test_data.csv']

In [34]: # load file using relative path
pd.read_csv('test_data.csv')

Out[34]:
   id first_name second_name
0   1     Shane      Lynn
1   2    Jimmy      Jacobs
2   3     Mark  Christos
3   4   Seamus   O'Higgins
4   5    Juän      Encoding

In [35]: # load file using absolute path (same result)
pd.read_csv('/Users/shane/Documents/Blog/read_csv/test_data.csv')

Out[35]:
   id first_name second_name
0   1     Shane      Lynn
1   2    Jimmy      Jacobs
2   3     Mark  Christos
3   4   Seamus   O'Higgins
4   5    Juän      Encoding
```

Figure: Loading the same file with Pandas read_csv using relative and absolute paths. Relative paths are directions to the file starting at your current working directory, where absolute paths always start at the base of your file system.

Reference: <https://www.shanelynn.ie/python-pandas-read-csv-load-data-from-csv-files>

Pandas CSV File Loading Errors

The most common error's you'll get while loading data from CSV files into Pandas will be:

1. FileNotFoundError: File b'filename.csv' does not exist

A File Not Found error is typically an issue with path setup, current directory, or file name confusion (file extension can play a part here!)

2. `UnicodeDecodeError`: 'utf-8' codec can't decode byte in position : invalid continuation byte

A Unicode Decode Error is typically caused by not specifying the encoding of the file, and happens when you have a file with non-standard characters. For a quick fix, try opening the file in Sublime Text, and re-saving with encoding 'UTF-8'.

3. `pandas.parser.CParserError`: Error tokenizing data.

Parse Errors can be caused in unusual circumstances to do with your data format – try to add the parameter “`engine='python'`” to the `read_csv` function call; this changes the data reading function internally to a slower but more stable method.

Advanced Read CSV Files

There are some additional flexible parameters in the Pandas `read_csv()` function that are useful to have in your arsenal of data science techniques:

- Specifying Data Types

As mentioned before, CSV files do not contain any type information for data. Data types are inferred through examination of the top rows of the file, which can lead to errors. To manually specify the data types for different columns, the `dtype` parameter can be used with a dictionary of column names and data types to be applied, for example: `dtype={"name": str, "age": np.int32}`

Note that for dates and date times, the format, columns, and other behaviour can be adjusted using `parse_dates`, `date_parser`, `dayfirst`, `keep_date` parameters.

- Skipping and Picking Rows and Columns From File

The `nrows` parameter specifies how many rows from the top of CSV file to read, which is useful to take a sample of a large file without loading completely. Similarly the `skiprows` parameter allows you to specify rows to leave out, either at the start of the file (provide an int), or throughout the file (provide a list of row indices). Similarly, the `usecols` parameter can be used to specify which columns in the data to load.

- Custom Missing Value Symbols

When data is exported to CSV from different systems, missing values can be specified with different tokens. The `na_values` parameter allows you to customise the characters that are recognised as missing values. The default values interpreted as NA/NaN are: "", '#N/A', '#N/A N/A', '#NA', '-1.#IND', '-1.#QNAN', '-NaN', '-nan', '1.#IND', '1.#QNAN', 'N/A', 'NA', 'NULL', 'NaN', 'n/a', 'nan', 'null'.

```
# Advanced CSV loading example
data = pd.read_csv(
    "data/files/complex_data_example.tsv",      # relative python path to subdirectory
    sep='\t'          # Tab-separated value file.
    quotechar="",    # single quote allowed as quote character
    dtype={"salary": int},           # Parse the salary column as an integer
    usecols=['name', 'birth_date', 'salary']. # Only load the three columns specified.
    parse_dates=['birth_date'],       # Interpret the birth_date column as a date
    skiprows=10,        # Skip the first 10 rows of the file
    na_values=['.', '?'])     # Take any '.' or '??' values as NA
) [6. 9]
```

PRACTICAL: Load JSON files to Python Pandas

Below are the steps to load JSON String into Pandas DataFrame

Step 1: Prepare the JSON String

To start with a simple example, let's say that you have the following data about different products and their prices:

Product	Price
Desktop Computer	700
Tablet	250
iPhone	800
Laptop	1200

This data can be captured as a JSON string:

```
{"Product":{"0":"Desktop Computer","1":"Tablet","2":"iPhone","3":"Laptop"},"Price":{"0":700,"1":250,"2":800,"3":1200}}
```

Step 2: Create the JSON File

Once you have your JSON string ready, save it within a JSON file.

Alternatively, you can copy the JSON string into Notepad, and then save that file with a .json file extension.

For example, open Notepad, and then copy the JSON string into it:

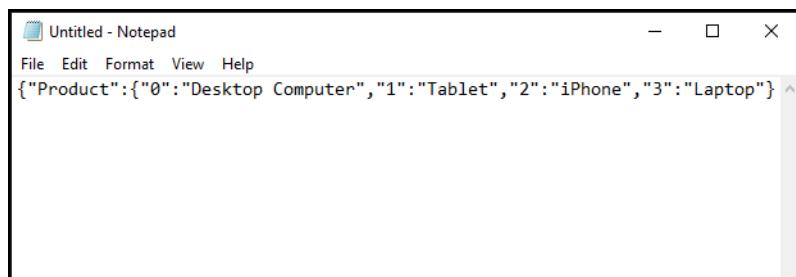


Figure: Copy the JSON string into the notepad.

Reference: <https://datatofish.com/load-json-pandas-dataframe>

Then, save the notepad with your desired file name and add the .json extension at the end of the file name. Here, I named the file as data.json:

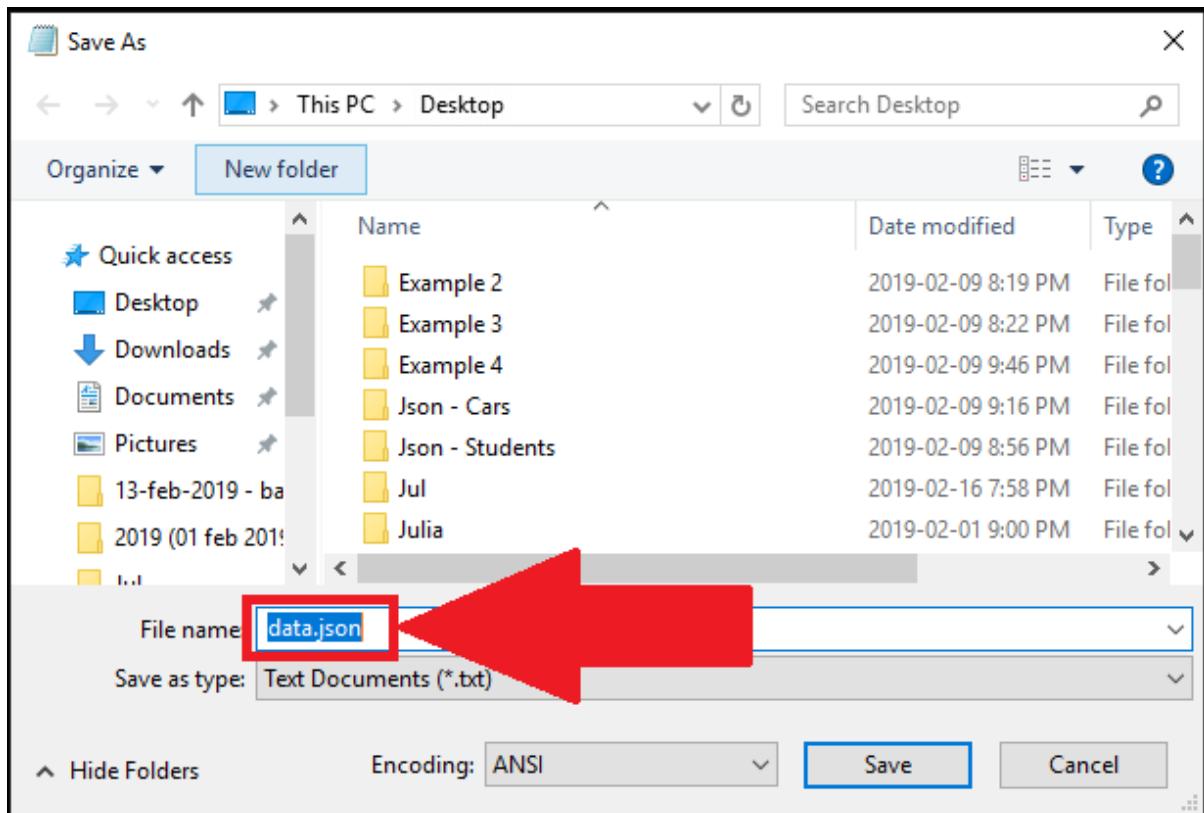


Figure: Saving the file with .json extension.

Reference: <https://datatofish.com/load-json-pandas-dataframe>

Step 3: Load the JSON File into Pandas DataFrame

Finally, load your JSON file into Pandas DataFrame.

```
import pandas as pd
```

```
pd.read_json (r'Path where you saved the JSON file\file Name.json')
```

In this case, The JSON file is stored on the Desktop, under this path:

C:\Users\Ron\Desktop\data.json

So this is the code that is used to load the JSON file into the DataFrame:

```
import pandas as pd
df = pd.read_json (r'C:\Users\Ron\Desktop\data.json')
print (df)
```

Run the code in Python (adjusted to your path), and you'll get the following DataFrame:

	Product	Price
0	Desktop Computer	700
1	Tablet	250
2	iPhone	800
3	Laptop	1200

1. different JSON strings

Below are 3 different ways that you could capture the data as JSON strings.

Each of those strings would generate a DataFrame with a different orientation when loading the files into Python.

1. Index orientation

2.

```
{"0":{"Product":"Desktop Computer","Price":700}, "1":{"Product":"Tablet","Price":250}, "2":{"Product":"iPhone", "Price":800}, "3":{"Product":"Laptop", "Price":1200}}
```

	0	1	2	3
Price	700	250	800	1200
Product	Desktop Computer	Tablet	iPhone	Laptop

3. Values orientation

```
[["Desktop Computer",700],["Tablet",250],["iPhone",800],["Laptop",1200]]
```

	0	1
Product	Desktop Computer	700
	Tablet	250
	iPhone	800
	Laptop	1200

4. Columns orientation

```
{"Product":{"0":"Desktop Computer","1":"Tablet","2":"iPhone","3":"Laptop"}, "Price":{"0":700,"1":250,"2":800,"3":1200}}
```

	Product	Price
0	Desktop Computer	700
1	Tablet	250
2	iPhone	800
3	Laptop	1200

1.8 PRACTICAL: Group by Methods

Before applying groupby function to the dataset, let's go over a visual example. Assume we have two features. One is color which is a categorical feature and the other one is a numerical feature, values. We want to group values by color and calculate the mean (or any other aggregation) of values for different colors. Then finally sort the colors based on average values. The following figure shows the steps of this process.

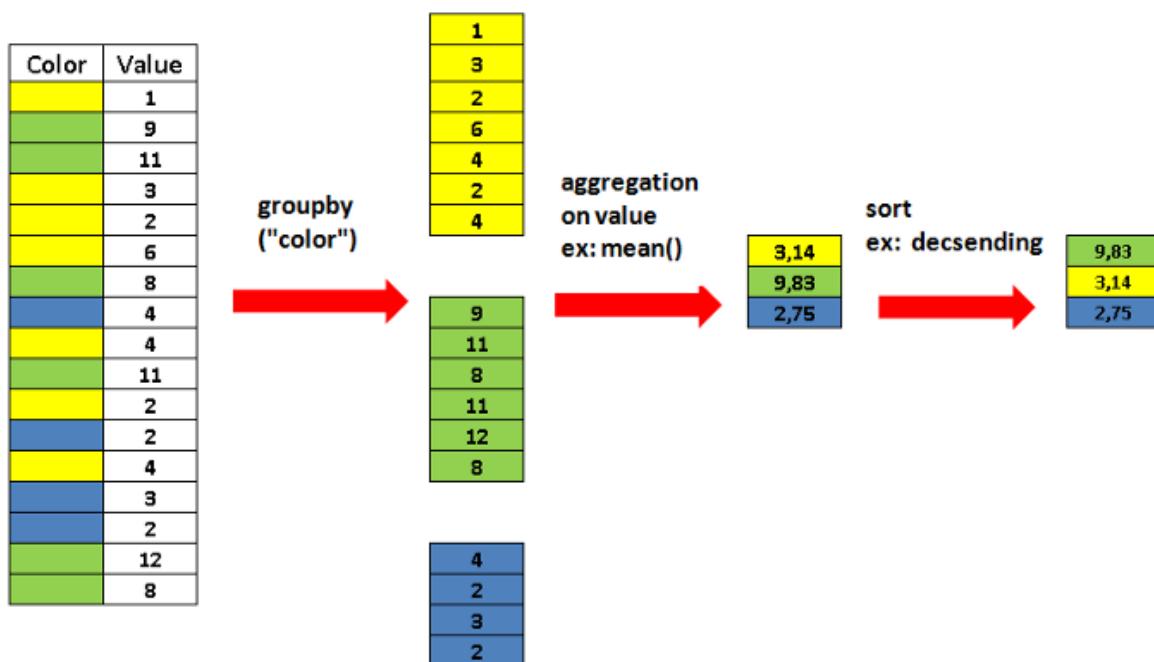


Figure: groupby in Pandas

Reference:<https://towardsdatascience.com/pandas-groupby-explained-453692519d0>

A Sample DataFrame

In order to demonstrate the effectiveness and simplicity of the grouping commands, we will need some data. The dataset contains 830 entries from my mobile phone log spanning a total time of 5 months. The CSV file can be loaded into a pandas

DataFrame using the `pandas.DataFrame.from_csv()` function, and looks like this:

	date	duration	item	month	network	network_type
0	15/10/14 06:58	34.429	data	2014-11	data	data
1	15/10/14 06:58	13.000	call	2014-11	Vodafone	mobile
2	15/10/14 14:46	23.000	call	2014-11	Meteor	mobile
3	15/10/14 14:48	4.000	call	2014-11	Tesco	mobile
4	15/10/14 17:27	4.000	call	2014-11	Tesco	mobile
5	15/10/14 18:55	4.000	call	2014-11	Tesco	mobile
6	16/10/14 06:58	34.429	data	2014-11	data	data
7	16/10/14 15:01	602.000	call	2014-11	Three	mobile
8	16/10/14 15:12	1050.000	call	2014-11	Three	mobile

9	16/10/14 15:30	19.000	call	2014-11	voicemail	voicemail
10	16/10/14 16:21	1183.000	call	2014-11	Three	mobile
11	16/10/14 22:18	1.000	sms	2014-11	Meteor	mobile
...

The main columns in the file are:

- date: The date and time of the entry
- duration: The duration (in seconds) for each call, the amount of data (in MB) for each data entry, and the number of texts sent (usually 1) for each sms entry.
- item: A description of the event occurring – can be one of call, sms, or data.
- month: The billing month that each entry belongs to – of form 'YYYY-MM'.
- network: The mobile network that was called/texted for each entry.
- network_type: Whether the number being called was a mobile, international ('world'), voicemail, landline, or other ('special') number.

The date column can be parsed using the **dateutil** library.

```
import pandas as pd
import dateutil

# Load data from csv file
data = pd.DataFrame.from_csv('phone_data.csv')
# Convert date from string to date times
data['date'] = data['date'].apply(dateutil.parser.parse, dayfirst=True)
```

Summarizing the DataFrame

Once the data has been loaded into Python, Pandas makes the calculation of different statistics very simple. For example, mean, max, min, standard deviations and more for columns are easily calculable:

```
# How many rows the dataset
data['item'].count()
Out[38]: 830

# What was the longest phone call / data entry?
data['duration'].max()
Out[39]: 10528.0

# How many seconds of phone calls are recorded in total?
data['duration'][data['item'] == 'call'].sum()
Out[40]: 92321.0

# How many entries are there for each month?
data['month'].value_counts()
Out[41]:
```

```
2014-11 230
2015-01 205
2014-12 157
2015-02 137
2015-03 101
dtype: int64
```

```
# Number of non-null unique network entries
data['network'].nunique()
```

```
Out[42]: 9
```

The `.describe()` function is a useful summarisation tool that will quickly display statistics for any variable or group it is applied to. The `describe()` output varies depending on whether you apply it to a numeric or character column.

Groupby output format – Series or DataFrame?

The output from a groupby and aggregation operation varies between Pandas Series and Pandas Dataframes. As a rule of thumb, if you calculate more than one column of results, your result will be a DataFrame. For a single column of results, the `agg` function, by default, will produce a Series.

You can change this by selecting your operation column differently:

```
# produces Pandas Series
data.groupby('month')['duration'].sum()
# Produces Pandas DataFrame
data.groupby('month')[['duration']].sum()
```

The groupby output will have an index or multi-index on rows corresponding to your chosen grouping variables. To avoid setting this index, pass “`as_index=False`” to the groupby operation.

```
data.groupby('month', as_index=False).agg({"duration": "sum"})
```

```
In [35]: data.groupby('month', as_index=False).agg({"duration": "sum"})
Out[35]:
   month    duration
0  2014-11  26639.441
1  2014-12  14641.870
2  2015-01  18223.299
3  2015-02  15522.299
4  2015-03  22750.441
```

Figure: Using the `as_index` parameter while Grouping data in pandas prevents setting a row index on the result.
Reference: <https://www.shanelynn.ie/summarising-aggregation-and-grouping-data-in-python-pandas>

Multiple Statistics per Group

The aggregation functionality provided by the agg() function allows multiple statistics to be calculated per group in one calculation.

Applying a single function to columns in groups

Instructions for aggregation are provided in the form of a python dictionary or list. The dictionary keys are used to specify the columns upon which you'd like to perform operations, and the dictionary values to specify the function to run.

For example:

```
# Group the data frame by month and item and extract a number of stats from each
# group
data.groupby(
    ['month', 'item']
).agg(
{
    'duration':sum, # Sum duration per group
    'network_type': "count", # get the count of networks
    'date': 'first' # get the first date per group
}
)
```

s

1.9 PRATICAL: Pivot Tables

You may be familiar with pivot tables in Excel to generate easy insights into your data. The function is quite similar to the group by function available in Pandas.

How to Build a Pivot Table in Python

It's a table of statistics that helps summarize the data of a larger table by "pivoting" that data. In Pandas, we can construct a pivot table using the following syntax,
`pandas.pivot_table(data, values=None, index=None, columns=None, aggfunc='mean', fill_value=None, margins=False, dropna=True, margins_name='All', observed=False)`

The levels in the pivot table will be stored in MultiIndex objects (hierarchical indexes) on the index and columns of the result DataFrame.

Parameters

Data: DataFrame

values: column to aggregate, optional

index: column, Grouper, array, or list of the previous

If an array is passed, it must be the same length as the data. The list can contain any of the other types (except list). Keys to group by on the pivot table index. If an array is passed, it is being used as the same manner as column values.

Columns: column, Grouper, array, or list of the previous

If an array is passed, it must be the same length as the data. The list can contain any of the other types (except list). Keys to group by on the pivot table column. If an array is passed, it is being used as the same manner as column values.

Aggfunc: function, list of functions, dict, default numpy.mean

If list of functions passed, the resulting pivot table will have hierarchical columns whose top level are the function names (inferred from the function objects themselves) If dict is passed, the key is column to aggregate and value is function or list of functions.

fill_value: scalar, default None

Value to replace missing values with (in the resulting pivot table, after aggregation).

Margins: bool, default False

Add all row / columns (e.g. for subtotal / grand totals).

Dropna: bool, default True

Do not include columns whose entries are all NaN.

margins_name: str, default 'All'

Name of the row / column that will contain the totals when margins is True.

Observed: bool, default False

This only applies if any of the groupers are Categoricals. If True: only show observed values for categorical groupers. If False: show all values for categorical groupers.

Returns: DataFrame

An Excel style pivot table.

We'll use Pandas to import the data into a dataframe called df. We'll also print out the first five rows using the .head() function:

```
import pandas as pd df =  
pd.read_excel('https://github.com/datagy/pivot_table_pandas/raw/master/sample_piv  
ot.xlsx', parse_dates=['Date']) print(df.head())
```

Out:

	Date	Region	Type	Units	Sales
0	2020-07-11	East	Children's Clothing	18	306
1	2020-09-23	North	Children's Clothing	14	448
2	2020-04-02	South	Women's Clothing	17	425
3	2020-02-28	East	Children's Clothing	26	832
4	2020-03-19	West	Women's Clothing	3	33

Creating a Pivot Table in Pandas

We'll begin by aggregating the Sales values by the Region the sale took place in:
sales_by_region = pd.pivot_table(df, index = 'Region', values = 'Sales')
print(sales_by_region)

This returns the following output:

Sales	Region
East	408.182482
North	438.924051
South	432.956204
West	452.029412

This gave us a summary of the Sales field by Region. The default parameter for aggfunc is mean. Because of this, the Sales field in the resulting dataframe is the average of Sales per Region.

If we wanted to change the type of function used, we could use the aggfunc parameter. For example, if we wanted to return the sum of all Sales across a region, we could write:

```
total_by_region = pd.pivot_table(df, index = 'Region', values = 'Sales',  
aggfunc='sum') print(total_by_region)
```

This returns:

Sales	Region
East	167763
North	138700
South	59315
West	61476

Filtering Python Pivot Tables

Let's create a dataframe that generates the mean Sale price by Region:

```
avg_region_price = pd.pivot_table(df, index = 'Region', values = 'Sales')
```

The values in this dataframe are:

Sales	Region
East	408.182482
North	438.924051
South	432.956204
West	452.029412

Now, say we wanted to filter the dataframe to only include Regions where the average sale price was over 450, we could write:

```
avg_region_price[avg_region_price['Sales'] > 450]
```

Sales	Region
West	452.029412

We can also apply multiple conditions, such as filtering to show only sales greater than 450 or less than 430.

```
avg_region_price[(avg_region_price['Sales'] > 450) | (avg_region_price['Sales'] < 430)]
```

We have wrapped each condition in brackets and separated the conditions by a pipe (|) symbol. This returns the following:

Sales	Region
East	408.182482
West	452.029412

Adding Columns to a Pandas Pivot Table

Adding columns to a pivot table in Pandas can add another dimension to the tables. The Columns parameter allows us to add a key to aggregate by. For example, if we wanted to see the number of units sold by Type and by Region, we could write:
`columns_example = pd.pivot_table(df, index = 'Type', columns = 'Region', values = 'Units', aggfunc = 'sum') print(columns_example)`

Region	East	North	South	West
Type				
Children's Clothing	2318.0	1763.0	1017.0	789.0
Men's Clothing	2420.0	0.0	725.0	829.0
Women's Clothing	3372.0	2596.0	1056.0	1006.0

Columns are optional as we indicated above and provide the keys by which to separate the data. The pivot table aggregates the values in the values parameter.

1.10 PRACTICAL: Pandas Plotting

Plot a Scatter Diagram using Pandas

Scatter plots are used to depict a relationship between two variables. Here are the steps to plot a scatter diagram using Pandas.

Step 1: Prepare the data

To start, prepare the data for your scatter diagram.

For example, the following data will be used to create the scatter diagram. This data captures the relationship between two variables related to an economy:

Unemployment_Rate	Stock_Index_Price
6.1	1500
5.8	1520
5.7	1525
5.7	1523
5.8	1515
5.6	1540
5.5	1545
5.3	1560
5.2	1555
5.2	1565

Step 2: Create the DataFrame

Once you have your data ready, you can proceed to create the DataFrame in Python.

For our example, the DataFrame would look like this:

```
import pandas as pd
```

```
data = {'Unemployment_Rate': [6.1,5.8,5.7,5.7,5.8,5.6,5.5,5.3,5.2,5.2],  
        'Stock_Index_Price': [1500,1520,1525,1523,1515,1540,1545,1560,1555,1565]  
       }
```

```
df = pd.DataFrame(data,columns=['Unemployment_Rate','Stock_Index_Price'])  
print (df)
```

Run the code in Python, and you'll get the following DataFrame:

```
   Unemployment_Rate  Stock_Index_Price  
0              6.1          1500  
1              5.8          1520  
2              5.7          1525  
3              5.7          1523  
4              5.8          1515  
5              5.6          1540  
6              5.5          1545  
7              5.3          1560  
8              5.2          1555  
9              5.2          1565
```

Step 3: Plot the DataFrame using Pandas

Finally, you can plot the DataFrame by adding the following syntax:

```
df.plot(x ='Unemployment_Rate', y='Stock_Index_Price', kind = 'scatter')
```

Notice that you can specify the type of chart by setting kind = 'scatter'

You'll also need to add the Matplotlib syntax to show the plot (ensure that the Matplotlib package is install in Python):

- `import matplotlib.pyplot as plt`
- `plt.show()`

Putting everything together:

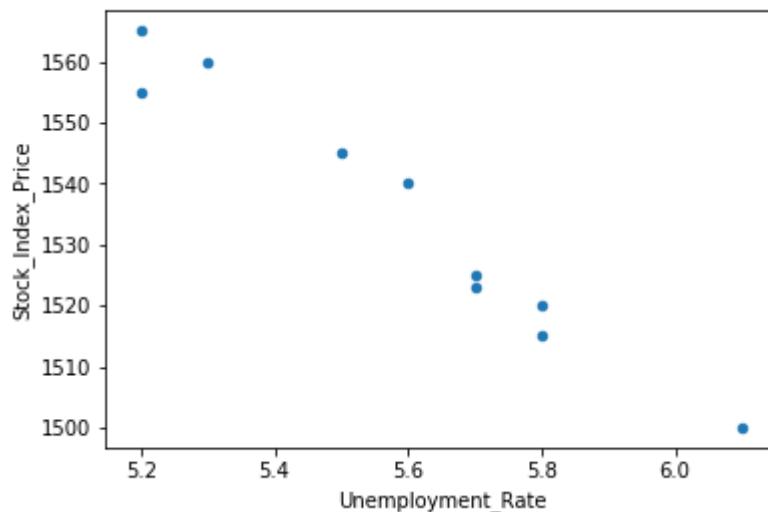
```
import pandas as pd  
import matplotlib.pyplot as plt
```

```
data = {'Unemployment_Rate': [6.1,5.8,5.7,5.7,5.8,5.6,5.5,5.3,5.2,5.2],
```

```
'Stock_Index_Price': [1500, 1520, 1525, 1523, 1515, 1540, 1545, 1560, 1555, 1565]  
}
```

```
df = pd.DataFrame(data,columns=['Unemployment_Rate','Stock_Index_Price'])  
df.plot(x ='Unemployment_Rate', y='Stock_Index_Price', kind = 'scatter')  
plt.show();
```

Once you run the above code, you'll get the following scatter diagram:



Plot a Line Chart using Pandas

Line charts are often used to display trends overtime. Let's now see the steps to plot a line chart using Pandas.

Step 1: Prepare the data

To start, prepare your data for the line chart. Here is an example of a dataset that captures the unemployment rate over time:

Year	Unemployment_Rate
1920	9.8
1930	12
1940	8
1950	7.2
1960	6.9
1970	7
1980	6.5
1990	6.2
2000	5.5
2010	6.3

Step 2: Create the DataFrame

Now create the DataFrame based on the above data:

```
import pandas as pd  
data = {'Year': [1920, 1930, 1940, 1950, 1960, 1970, 1980, 1990, 2000, 2010],
```

```
'Unemployment_Rate': [9.8,12,8,7.2,6.9,7,6.5,6.2,5.5,6.3] }  
df = pd.DataFrame(data,columns=['Year','Unemployment_Rate'])  
print (df);
```

This is how the DataFrame would look like:

Year	Unemployment_Rate
0 1920	9.8
1 1930	12.0
2 1940	8.0
3 1950	7.2
4 1960	6.9
5 1970	7.0
6 1980	6.5
7 1990	6.2
8 2000	5.5
9 2010	6.3

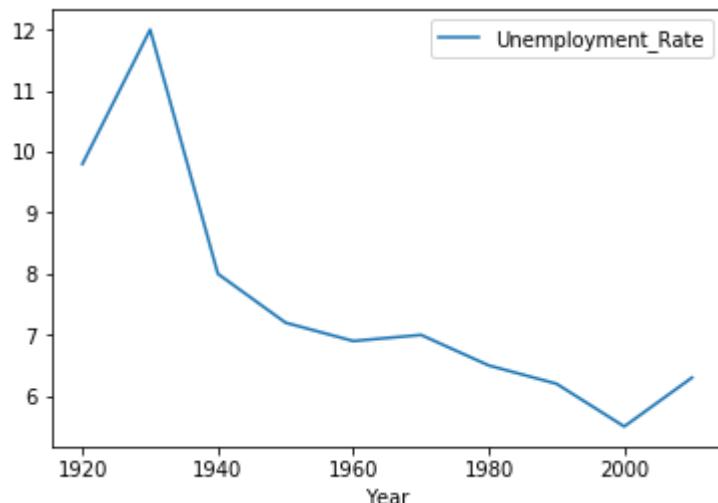
Step 3: Plot the DataFrame using Pandas

Finally, plot the DataFrame by adding the following syntax:

```
df.plot(x ='Year', y='Unemployment_Rate', kind = 'line')
```

You'll notice that the `kind` is now set to '`line`' in order to plot the line chart.

```
import pandas as pd  
import matplotlib.pyplot as plt  
data = {'Year': [1920,1930,1940,1950,1960,1970,1980,1990,2000,2010],  
        'Unemployment_Rate': [9.8,12,8,7.2,6.9,7,6.5,6.2,5.5,6.3]}  
df = pd.DataFrame(data,columns=['Year','Unemployment_Rate'])  
df.plot(x ='Year', y='Unemployment_Rate', kind = 'line')  
plt.show();
```



Plot a Bar Chart using Pandas

Bar charts are used to display categorical data. Let's now see how to plot a bar chart using Pandas.

Step 1: Prepare your data

As before, you'll need to prepare your data. Here, the following dataset will be used to create the bar chart:

Country	GDP_Per_Capita
USA	45000
Canada	42000
Germany	52000
UK	49000
France	47000

Step 2: Create the DataFrame

Create the DataFrame as follows:

```
import pandas as pd
data = {'Country': ['USA','Canada','Germany','UK','France'],
        'GDP_Per_Capita': [45000,42000,52000,49000,47000] }
df = pd.DataFrame(data,columns=['Country','GDP_Per_Capita'])
print (df)
```

You'll then get this DataFrame:

```
   Country  GDP_Per_Capita
0      USA            45000
1    Canada           42000
2  Germany           52000
3       UK            49000
4    France           47000
```

Step 3: Plot the DataFrame using Pandas

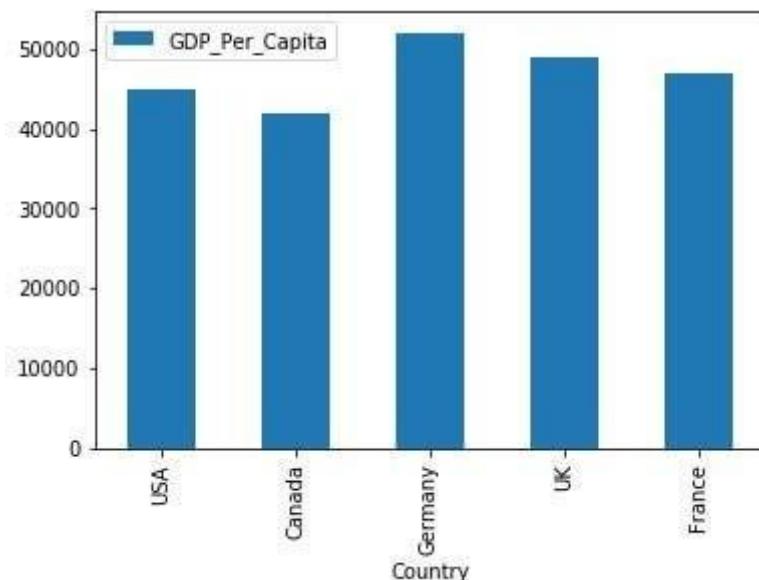
Finally, add the following syntax to the Python code:

```
df.plot(x ='Country', y='GDP_Per_Capita', kind = 'bar')
```

In this case, set the **kind = 'bar'** to plot the bar chart.

```
import pandas as pd
import matplotlib.pyplot as plt
data = {'Country': ['USA','Canada','Germany','UK','France'],
        'GDP_Per_Capita': [45000,42000,52000,49000,47000]
       }
df = pd.DataFrame(data,columns=['Country','GDP_Per_Capita'])
df.plot(x ='Country', y='GDP_Per_Capita', kind = 'bar')
plt.show();
```

Run the code and you'll get this bar chart:



Plot a Pie Chart using Pandas

Step 1: Prepare your data

For demonstration purposes, the following data about the status of tasks was prepared:

Tasks Pending	300
Tasks Ongoing	500
Tasks Completed	700

The goal is to create a pie chart based on the above data.

Step 2: Create the DataFrame

You can then create the DataFrame using this code:

```
import pandas as pd
data = {'Tasks': [300,500,700]}
df = pd.DataFrame(data,columns=['Tasks'],index = ['Tasks Pending','Tasks Ongoing','Tasks Completed'])
print (df);
```

You'll now see this DataFrame:

	Tasks
Tasks Pending	300
Tasks Ongoing	500
Tasks Completed	700

Step 3: Plot the DataFrame using Pandas

Finally, plot the DataFrame by adding the following syntax:

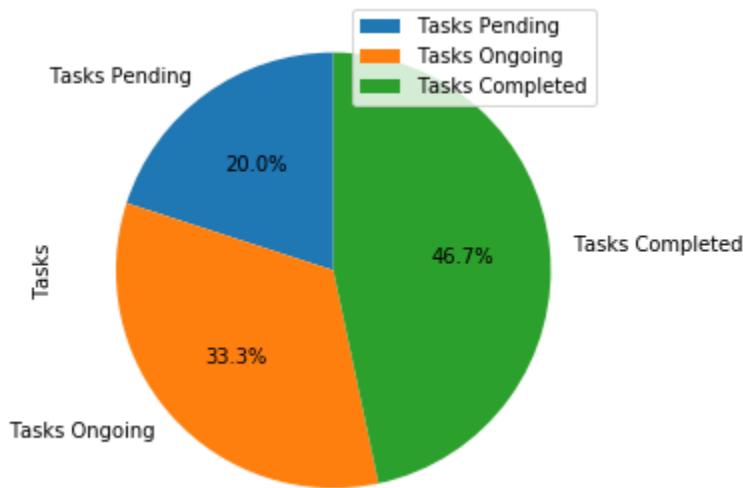
```
df.plot.pie(y='Tasks',figsize=(5, 5),autopct='%.1f%%', startangle=90)
```

The complete code is as follows:

```
import pandas as pd
import matplotlib.pyplot as plt
data = {'Tasks': [300,500,700]}
df = pd.DataFrame(data,columns=['Tasks'],index = ['Tasks Pending','Tasks Ongoing','Tasks Completed'])
```

```
df.plot.pie(y='Tasks', figsize=(5, 5), autopct='%1.1f%%', startangle=90)  
plt.show();
```

Once you run the code, you'll get this pie chart: [6.15]



So, in this chapter we have explored about various important concepts of Data analytics and the libraries which are used in data analysis like NumPy, Pandas and Matplotlib. Using these libraries, we can analyse our data and make sense out of data.

Unit 2: Python GUI - Tkinter

Learning Outcomes:

- Understand the concept and features of Python GUI
- Able to create Graphical user interface using python programming

2.1 Understanding Python GUI

What is a GUI?

A GUI (graphical user interface) is a system of interactive visual components for computer software. A GUI displays objects that convey information and represent actions that can be taken by the user. The objects change color, size, or visibility when the user interacts with them.

Python GUI

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create GUI applications.

2.1.1 Different Python GUI Tools

Developing is a core part and there is no shortage of programming languages, with Python being the trending one. Python is an interactive programming language and getting started with programming a GUI (Graphical User Interface) framework is not much of a difficult task. Python has a diverse range of options for GUI frameworks.

Kivy

Kivy is an OpenGL ES 2 accelerated framework for the creation of new user interfaces. It supports multiple platforms namely Windows, Mac OSX, Linux, Android iOS and Raspberry Pi. It is open source and comes with over 20 widgets in its toolkit.

PyQT

PyQT is one of the favoured cross-platform Python bindings implementing the Qt library for the Qt (owned by Nokia) application development framework. Currently, PyQT is available for Unix/Linux, Windows, Mac OS X and Sharp Zaurus. It combines the best of Python and Qt and it is up to the programmer to decide whether to create a program by coding or using Qt Designer to create visual dialogs.

It is available in both commercial as well as GPL license. Although some features may not be available in the free version, if your application is open source, then you can use it under the free license.

Tkinter

Tkinter is commonly bundled with Python, using Tk and is Python's standard GUI framework. It is popular for its simplicity and graphical user interface. It is open source and available under the Python License.

One of the advantages of choosing Tkinter is that since it comes by default, there is an abundance of resources, both codes and reference books. Also, with the community being old and active, there are many users who can help you out in case of doubts.

WxPython

WxPython is an open-source wrapper for cross-platform GUI library WxWidgets (earlier known as WxWindows) and implemented as a Python extension module. With WxPython you as a developer can create native applications for Windows, Mac OS and Unix.

PyGUI

PyGUI is a graphical application cross-platform framework for Unix, Macintosh and Windows. Compared to some other GUI frameworks, PyGUI is by far the simplest and lightweight of them all, as the API is purely in sync with Python. PyGUI inserts very less code between the GUI platform and Python application, hence the display of the application usually displays the natural GUI of the platform.

PySide

PySide is a free and cross-platform GUI toolkit Qt initiated and sponsored by Nokia, Qt is a UI framework and a cross-platform application. PySide currently supports Linux/X11, Mac OS X, Maemo and Windows and support for Android is in the plans for the near future. PySide provides tools to work with multimedia, XML documents, network, databases and GUI. A key feature of PySide is its API compatibility with PyQt4, so if you wish to migrate to PySide then the process will be hassle-free.

2.2 Introduction to Tkinter

Overview

Tkinter is an open source, portable graphical user interface (GUI) library designed for use in Python scripts.

Tkinter relies on the Tk library, the GUI library used by Tcl/Tk and Perl, which is in turn implemented in C. Therefore, Tkinter can be said to be implemented using multiple layers.

Several competing GUI toolkits are available to use with the Python language, namely

Advantages of Tkinter

Layered approach

The layered approach used in designing Tkinter gives Tkinter all of the advantages of the TK library. Therefore, at the time of creation, Tkinter inherited from the benefits of a GUI toolkit that had been given time to mature. This makes early versions of Tkinter a lot more stable and reliable than if it had been rewritten from scratch. Moreover, the conversion from Tcl/Tk to Tkinter is really trivial, so that Tk programmers can learn to use Tkinter very easily.

Accessibility

Learning Tkinter is very intuitive, and therefore quick and painless. The Tkinter implementation hides the detailed and complicated calls in simple, intuitive methods. This is a continuation of the Python way of thinking, since the language excels at quickly building prototypes. It is therefore expected that its preferred GUI library be implemented using the same approach. For example, here is the code for a typical “Hello world”-like application

Portability

Python scripts that use Tkinter do not require modifications to be ported from one platform to the other. Tkinter is available for any platform that Python is implemented for, namely Microsoft Windows, X Windows, and Macintosh. This gives it a great advantage over most competing libraries, which are often restricted to one or two platforms. Moreover, Tkinter will provide the native look-and-feel of the specific platform it runs on.

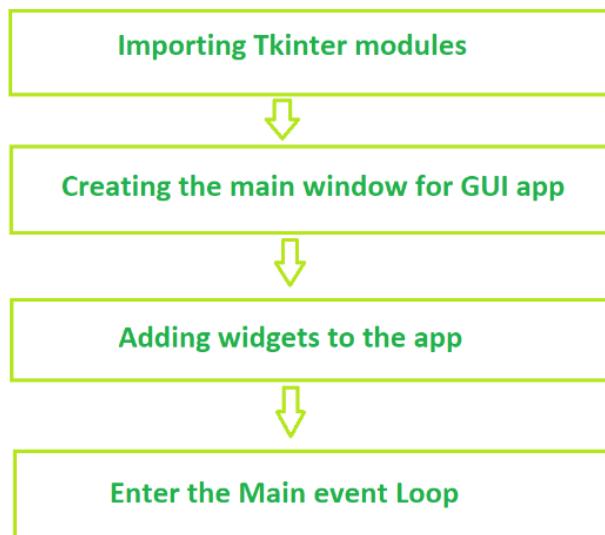
Availability

Tkinter is now included in any Python distribution. Therefore, no supplementary modules are required in order to run scripts using Tkinter.

Tkinter Widgets

Tkinter widgets are a subset of classes in Tkinter. They all inherit from the class Widget. Each one of them, when instantiated, creates a window component that can be styled to respond to the programmer’s need. To control the appearance of a widget, you usually use options rather than method calls. Typical options include text and color, size, command callbacks, etc. (see table below) When widgets are instantiated, they don’t immediately appear on the screen.

Fundamental Structure of Tkinter Program



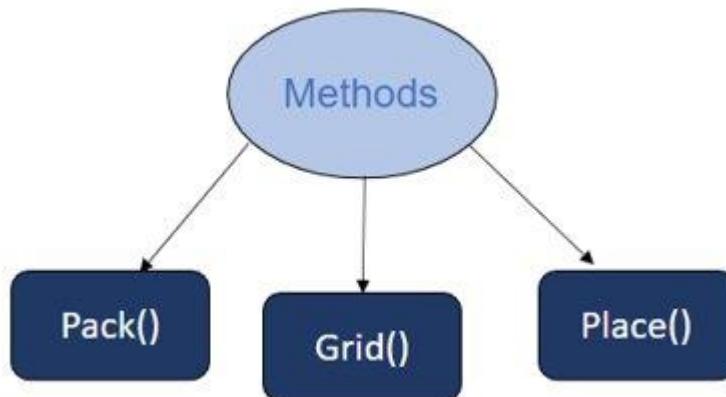
Basic Tkinter Widgets

Widgets	Description
Label	It is used to display text or images on the screen.
Button	It is used to add buttons to your application.
Canvas	It is used to draw pictures and other layouts like texts, graphics etc.
ComboBox	It contains a down arrow to select from a list of available options
CheckButton	It displays a number of options to the user as toggle buttons from which user can select any number of options.
RadioButton	It is used to implement one-of-many selections as it allows only one option to be selected
Entry	It is used to input single line text entry from user
Frame	It is used as container to hold and organize the widgets
Message	It works same as that of label and refers to multi-line and non-editable text
Scale	It is used to provide a graphical slider which allows you to select any value from that scale
Scrollbar	It is used to scroll down the contents. It provides a slide controller.
SpinBox	It allows the user to select from a given set of values
Text	It allows the user to edit multiline text and format the way it has to be displayed
Menu	It is used to create all kinds of menus used by an application

Python Tkinter Geometry Manager

In order to organize or arrange or place all the widgets in the parent window, Tkinter provides us with the geometric configuration of the widgets. The GUI Application Layout is mainly controlled by Geometric Managers of Tkinter.

It is important to note here that each window and Frame in your application is allowed to use only one geometry manager.



1. Tkinter pack () Geometry Manager

The pack () method mainly uses a packing algorithm in order to place widgets in a Frame or window in a specified order.

This method is mainly used to organize the widgets in a block.

Packing Algorithm:

The steps of Packing algorithm are as follows:

- Firstly, this algorithm will compute a rectangular area known as a Parcel which is tall (or wide) enough to hold the widget and then it will fill the remaining width (or height) in the window with blank space.
- It will center the widget until any different location is specified.

This method is powerful but it is difficult to visualize.

Here is the syntax for using pack () function:

```
widget.pack(options)
```

2. Tkinter grid () Geometry Manager

The most used geometry manager is grid() because it provides all the power of pack() function but in an easier and maintainable way.

The grid () geometry manager is mainly used to split either a window or frame into rows and columns.

- You can easily specify the location of a widget just by calling grid () function and passing the row and column indices to the row and column keyword arguments, respectively.
- The index of both the row and column starts from 0, so a row index of 2 and a column index of 2 tells the grid () function to place a widget in the third column of the third row (0 is first, 1 is second and 2 means third).

Here is the syntax of the grid () function:

```
widget.grid(options)
```

3. Trinket place () Geometry Manager

The place () Geometry Manager organizes the widgets to place them in a specific position as directed by the programmer.

This method basically organizes the widget in accordance with its x and y coordinates. Both x and y coordinates are in pixels.

Thus, the origin (where x and y are both 0) is the top-left corner of the Frame or the window.

Thus, the y argument specifies the number of pixels of space from the top of the window, to place the widget, and the x argument specifies the number of pixels from the left of the window.

Here is the syntax of the place () method:

```
widget.place(options)
```

Working with Image in Tkinter

Tkinter relies on the Python Pillow (aka PIL) package for image processing capabilities. Using Pillow, a Tkinter function that displays a text-based message can instead display an image-based message.

How To Display Images with Tkinter's Label Widget

Tkinter's label widget can be used to display either images or text. To display an image requires the use of Image and ImageTk imported from the Python Pillow (aka PIL) package.

A label widget can display either PhotoImage or BitmapImage objects:

- The PhotoImage class is used to display grayscale or true color icons, as well as images in labels. Note that only GIF and PGM/PPM image formats are supported.
- The BitmapImage class is used to display only monochrome (two-color) images in labels.

How To Use Pillow with Tkinter

Tkinter relies on Pillow for working with images. A pillow is a fork of the Python Imaging Library, and can be imported in a Python console as PIL. Pillow has the following characteristics:

- Support for a wide range of image file formats, including PNG, JPEG and GIF.
- Basic image processing and manipulation functionality
- Width and height options that can be used to set the Label size for an image. If a size is not specified, the Label will be just large enough to display its contents.

To check if Pillow is already installed, enter the following command in a Python console:

```
help('PIL')
```

If Pillow is not installed, you can install it with:

```
python3 -m pip install pillow
```

How to manage images with PIL and Tkinter?

To import ImageTk and Image in a Python console, enter:

```
from PIL import ImageTk, Image
```

An image can be opened with the following code snippet:

```
image1 = Image.open("<path/image_name>")
```

The resize () option can be used to set an image's height and width. In the following example, an image's dimensions are set to 50 pixels high and wide:

```
image1 = img.resize((50, 50), Image.ANTIALIAS)
```

2.3 Introduction to Tkinter Themes and Styles

In Tkinter, a theme determines the “look & feel” of all the widgets. It's a collection of styles for all the ttk widgets.

A style specifies the appearance of a widget class e.g., a Button. Each theme comes with a set of styles. It's possible to change the appearance of widgets by:

- Modifying the built-in styles
- or creating new styles

Tkinter allows you to change the current theme to another. When you change the current theme to a new one, Tkinter will apply the styles of that theme to all the ttk widgets.

To get the available themes, you use the theme_names() method of the ttk.Style instance.

First, create a new instance of the ttk.Style class:

```
style = ttk.Style(root)
```

Second, get the available themes by calling the theme_names() method:

```
style = ttk.Style(root)
```

To get the current theme, you use the `theme_use()` method:

```
current_theme = style.theme_use()
```

To change the current theme to a new one, you pass the new theme name to the `theme_use()` method:

```
current_theme = style.theme_use()
```

Summary

- Create an instance of the `ttk.Style` class to access the style database.
- Use the `style.theme_names()` method to get available themes from the Operating System on which the Tkinter application is running.
- Use the `style.theme_use()` method to change the current theme to a new one.

2.4 PRACTICAL: Tkinter

Working with Iris Dataset

Importing Packages

```
In [21]: import tkinter as tk
         from tkinter import ttk
         import pickle
         import joblib
         import numpy as np
         import seaborn as sns
         from sklearn.utils import shuffle
         from sklearn.linear_model import LogisticRegression
```

Basics of tkinter

Creating a Window

```
In [22]: window=tk.Tk()
          tk.mainloop()
```

Creating Label widget in window

```
In [23]: window=tk.Tk()
label1=tk.Label(window,text="Hello")
label1.pack()
tk.mainloop()
```

Creating Entry widget in window

```
In [24]: window=tk.Tk()
x1=tk.StringVar()
tk.Entry(window,textvariable=x1).pack()
tk.mainloop()
print(x1.get())
```

Creating Button widget in window

```
In [25]: window=tk.Tk()
tk.Button(window,text="click",command= lambda : print("Clicked")).pack()
tk.mainloop()

Clicked
```

Working on Iris Dataset

```
In [26]: # Importing Dataset and printing head
iris=sns.load_dataset("iris")
iris.head()
```

Out[26]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
In [27]: # unique species
iris.species.unique()
```

Out[27]: array(['setosa', 'versicolor', 'virginica'], dtype=object)

```
In [29]: iris.species_num.unique()
```

```
Out[29]: array([0, 1, 2], dtype=int64)
```

```
In [30]: iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype  
 ---  --          --          --      
 0   sepal_length  150 non-null   float64 
 1   sepal_width   150 non-null   float64 
 2   petal_length  150 non-null   float64 
 3   petal_width   150 non-null   float64 
 4   species      150 non-null   object  
 5   species_num   150 non-null   int64   
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

```
In [31]: # dropping the species name column
iris.drop('species',axis=1,inplace=True)
iris.head()
```

```
Out[31]:
```

	sepal_length	sepal_width	petal_length	petal_width	species_num
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

```
In [32]: # shuffling data
iris=shuffle(iris)
iris.reset_index(drop=True)
iris.head()
```

```
Out[32]:
```

	sepal_length	sepal_width	petal_length	petal_width	species_num
71	6.1	2.8	4.0	1.3	1
76	6.8	2.8	4.8	1.4	1
129	7.2	3.0	5.8	1.6	2
45	4.8	3.0	1.4	0.3	0
141	6.9	3.1	5.1	2.3	2

Unit 3: Building Machine Learning Models

Learning Outcomes:

- Understand the basics of machine learning, its types and applications
- Create Machine Learning model using various Algorithms
- Demonstrate working of ML model without coding
- Able to differentiate between Supervised and Unsupervised Learning
- Able to identify and apply specific ML algorithm to solve real life problems

3.1 Machine Learning Basics

Machine Learning is undeniably one of the most influential and powerful technologies in today's world. Machine learning is a tool for turning information into knowledge. In the past 50 years, there has been an explosion of data. This mass of data is useless unless we analyze it and find the patterns hidden within. Machine learning techniques are used to automatically find the valuable underlying patterns within complex data that we would otherwise struggle to discover. The hidden patterns and knowledge about a problem can be used to predict future events and perform all kinds of complex decision making. Machine learning is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead.

In laymen's terms, Machine Learning is about making predictions (answering questions) and classifications based on data. The more data you have, the easier it will be to recognize patterns and inferences. The data is used to train a machine learning model, then the model is used to make predictions and answer questions.

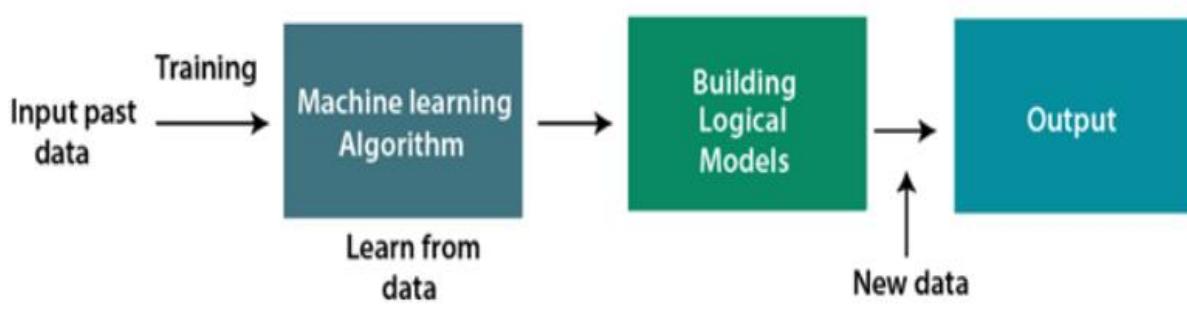


Image: ML working concept

Reference: <https://static.javatpoint.com/tutorial/machine-learning/images/introduction-to-machine-learning2.png>

Terminology of ML

Dataset: A set of data examples, that contain features important to solving the problem.

Features: Important pieces of data that help us understand a problem. These are fed in to a Machine Learning algorithm to help it learn.

Model: The representation (internal model) of a phenomenon that a Machine Learning algorithm has learnt. It learns this from the data it is shown during training. The model is the output you get after training an algorithm. For example, a decision tree algorithm would be trained and produce a decision tree model.

Process

Data Collection: Collect the data that the algorithm will learn from.

Data Preparation: Format and engineer the data into the optimal format, extracting important features and performing dimensionality reduction.

Training: Also known as the fitting stage, this is where the Machine Learning algorithm actually learns by showing it the data that has been collected and prepared.

Evaluation: Test the model to see how well it performs.

Tuning: Fine tune the model to maximize its performance.

Let us see some real life applications of machine learning in our day-to-day life.

Real time Application of Machine Learning

Machine learning is relevant in many fields, industries, and has the capability to grow over time. Here are six real-life examples of how machine learning is being used.



Image: Applications of ML

1. Image recognition

Image recognition is a well-known and widespread example of machine learning in the real world. It can identify an object as a digital image, based on the intensity of the pixels in black and white images or colour images.

- Real-world examples of image recognition:
- Label an x-ray as cancerous or not
- Assign a name to a photographed face (aka “tagging” on social media)
- Recognise handwriting by segmenting a single letter into smaller images
- Machine learning is also frequently used for facial recognition within an image. Using a database of people, the system can identify commonalities and match them to faces. This is often used in law enforcement.

2. Speech recognition

Machine learning can translate speech into text. Certain software applications can convert live voice and recorded speech into a text file. The speech can be segmented by intensities on time-frequency bands as well.

Real-world examples of speech recognition:

- Voice search
- Voice dialling
- Appliance control

Some of the most common uses of speech recognition software are devices like Google Home or Amazon Alexa.

3. Medical diagnosis

Machine learning can help with the diagnosis of diseases. Many physicians use chatbots with speech recognition capabilities to discern patterns in symptoms.

Real-world examples for medical diagnosis:

- Assisting in formulating a diagnosis or recommends a treatment option
- Oncology and pathology use machine learning to recognise cancerous tissue
- Analyse bodily fluids
- In the case of rare diseases, the joint use of facial recognition software and machine learning helps scan patient photos and identify phenotypes that correlate with rare genetic diseases.

4. Statistical arbitrage

Arbitrage is an automated trading strategy that's used in finance to manage a large volume of securities. The strategy uses a trading algorithm to analyze a set of securities using economic variables and correlations.

Real-world examples of statistical arbitrage:

- Algorithmic trading which analyses a market microstructure
- Analyse large data sets
- Identify real-time arbitrage opportunities

Machine learning optimizes the arbitrage strategy to enhance results.

5. Predictive analytics

Machine learning can classify available data into groups, which are then defined by rules set by analysts. When the classification is complete, the analysts can calculate the probability of a fault.

Real-world examples of predictive analytics:

- Predicting whether a transaction is fraudulent or legitimate
- Improve prediction systems to calculate the possibility of fault

Predictive analytics is one of the most promising examples of machine learning. It's applicable for everything; from product development to real estate pricing.

6. Extraction

Machine learning can extract structured information from unstructured data. Organizations amass huge volumes of data from customers. A machine learning algorithm automates the process of annotating datasets for predictive analytics tools.

Real-world examples of extraction:

- Generate a model to predict vocal cord disorders
- Develop methods to prevent, diagnose, and treat the disorders
- Help physicians diagnose and treat problems quickly

Typically, these processes are tedious. But machine learning can track and extract information to obtain billions of data samples.

Let us see the different techniques in machine learning.

3.2 Techniques of ML

The two main categories of machine learning techniques are supervised learning and unsupervised learning.

Supervised Machine Learning

Supervised learning is the types of machine learning in which machines are trained using well "labelled" training data, and on basis of that data, machines predict the output. The labelled data means some input data is already tagged with the correct output. In supervised learning, the training data provided to the machines work as the supervisor that teaches the machines to predict the output correctly. It applies the same concept as a student learns in the supervision of the teacher.

Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to find a mapping function to map the input variable (x) with the output variable (y).

In the real-world, supervised learning can be used for Risk Assessment, Image classification, Fraud Detection, spam filtering, etc.

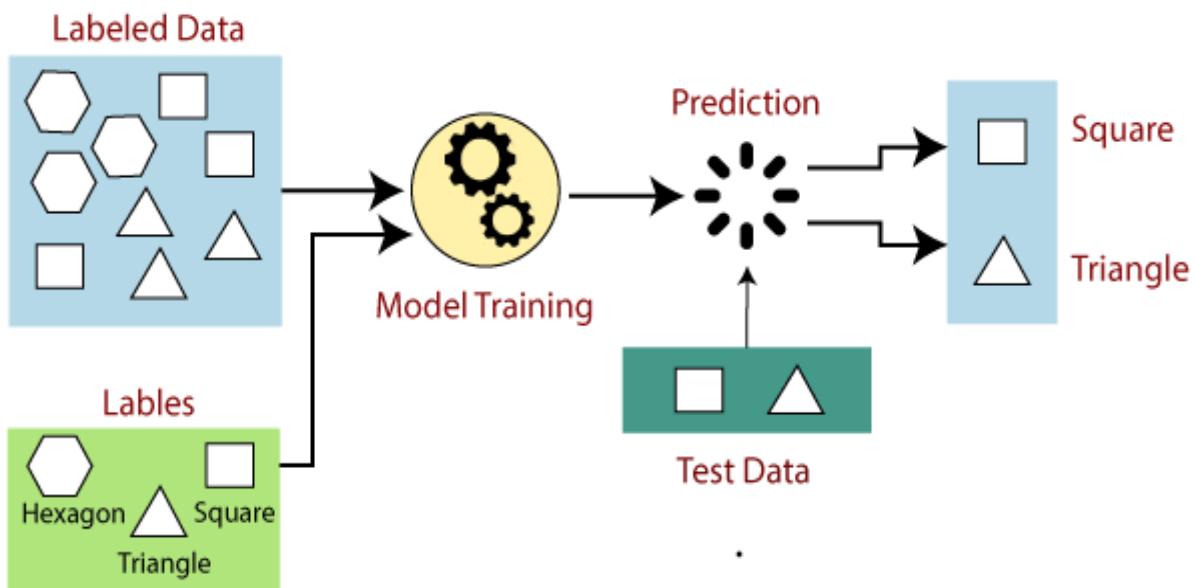


Image: Working of Supervised learning

Reference: <https://www.javatpoint.com/supervised-machine-learning>

Types of supervised Machine learning Algorithms:

Supervised learning can be further divided into two types of problems:

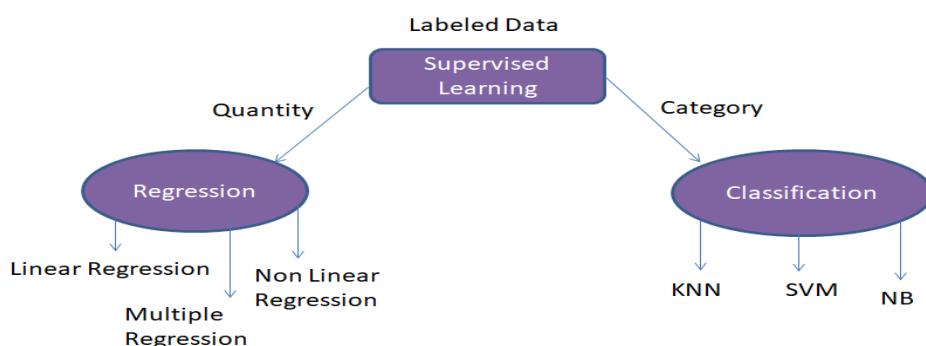


Image: Categories of supervised learning

Unsupervised Machine Learning

Unsupervised learning is a machine learning technique in which models are not supervised using training dataset. Instead, models itself find the hidden patterns and insights from the given data. It can be compared to learning which takes place in the human brain while learning new things.

Unsupervised learning cannot be directly applied to a regression or classification problem because unlike supervised learning, we have the input data but no corresponding output data. The goal of unsupervised learning is to find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format.

Example: Suppose the unsupervised learning algorithm is given an input dataset containing images of different types of cats and dogs. The algorithm is never trained upon the given dataset, which means it does not have any idea about the features of the dataset. The task of the unsupervised learning algorithm is to identify the image features on their own. Unsupervised learning algorithm will perform this task by clustering the image dataset into the groups according to similarities between images.

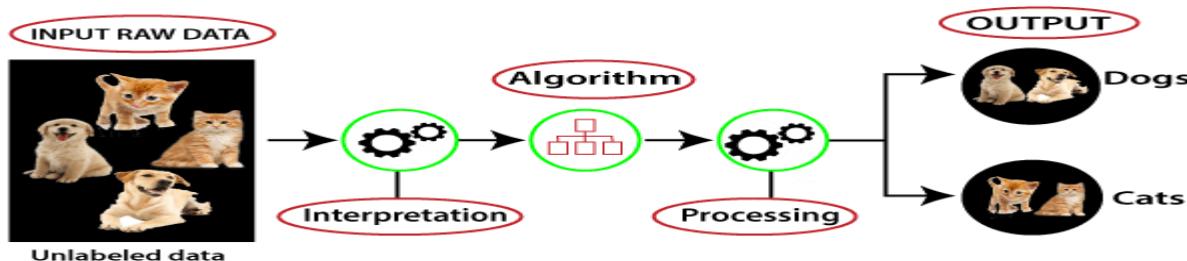


Image: Working of unsupervised learning

Reference: <https://www.javatpoint.com/unsupervised-machine-learning>

Types of Unsupervised Learning Algorithm:

The unsupervised learning algorithm can be further categorized into two types of problems:

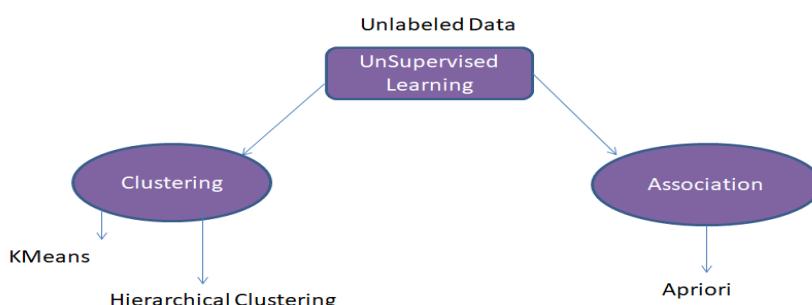


Image: Categories of unsupervised learning

Reinforcement Machine Learning

In this type of machine learning, during the training of algorithm it uses system of rewards and penalty. The learning system, called agent in this context, learns with an interactive environment. The agent selects and performs actions and receives rewards by performing correctly and penalties for performing incorrectly. In reinforcement learning the agent learns by itself, without the intervention from a human. For example, Self-Driving Cars, robots etc.

Reinforcement Learning

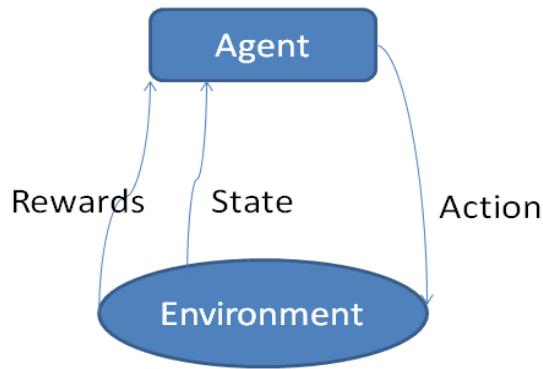


Image: Reinforcement Machine Learning

Now, after understanding the brief overview of machine learning and its types, let us explore a very popular library in machine learning which is used by many machine learning engineers and data scientists to perform various data science and AI projects, named as Scikit-learn. Let us get started with Scikit learn library.

3.3 Scikit Learn library overview

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

Origin of Scikit-Learn

It was originally called scikits.learn and was initially developed by David Cournapeau as a Google summer of code project in 2007. Later, in 2010, Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, and Vincent Michel, from FIRCA (French Institute for Research in Computer Science and Automation), took this project at another level and made the first public release (v0.1 beta) on 1st Feb. 2010.

Version History: -

- April 2021: scikit-learn 0.24.2
- Jan 2021: scikit-learn 0.24.1
- Dec 2020: scikit-learn 0.24.0
- May 2020: scikit-learn 0.23.2
- May 2020: scikit-learn 0.23.1
- May 2020: scikit-learn 0.23.0
- Jan 2020: scikit-learn 0.22.1
- Dec 2019: scikit-learn 0.22.0
- May 2019: scikit-learn 0.21.0

- March 2019: scikit-learn 0.20.3
- December 2018: scikit-learn 0.20.2
- November 2018: scikit-learn 0.20.1
- September 2018: scikit-learn 0.20.0
- July 2018: scikit-learn 0.19.2
- July 2017: scikit-learn 0.19.0
- September 2016: scikit-learn 0.18.0
- November 2015: scikit-learn 0.17.0
- March 2015: scikit-learn 0.16.0
- July 2014: scikit-learn 0.15.0
- August 2013: scikit-learn 0.14

Scikit Learn is built on top of several common data and math Python libraries. Such a design makes it super easy to integrate between them all. You can pass numpy arrays and pandas data frames directly to the ML algorithms of Scikit! It uses the following libraries:

NumPy: For any work with matrices, especially math operations

SciPy: Scientific and technical computing

Matplotlib: Data visualization

IPython: Interactive console for Python

Sympy: Symbolic mathematics

Pandas: Data handling, manipulation, and analysis

Scikit Learn is focused on Machine Learning, e.g data modelling. It is not concerned with the loading, handling, manipulating, and visualizing of data. Thus, it is natural and common practice to use the above libraries, especially NumPy, for those extra steps; they are made for each other!

Scikit's robust set of algorithm offerings include:

- Regression: Fitting linear and non-linear models
- Clustering: Unsupervised classification
- Decision Trees: Tree induction and pruning for both classification and regression tasks
- Neural Networks: End-to-end training for both classification and regression. Layers can be easily defined in a tuple
- SVMs: for learning decision boundaries
- Naive Bayes: Direct probabilistic modelling
- Even beyond that, it has some very convenient and advanced functions not commonly offered by other libraries:
- Ensemble Methods: Boosting, Bagging, Random Forest, Model voting and averaging
- Feature Manipulation: Dimensionality reduction, feature selection, feature analysis
- Outlier Detection: For detecting outliers and rejecting noise

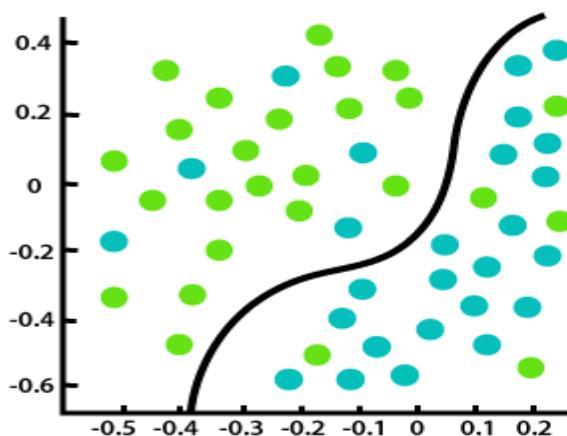
- Model selection and validation: Cross-validation, Hyperparameter tuning, and metrics.

Now we have discussed about the machine learning library `sklearn`, let us start with the basic algorithms in machine learning.

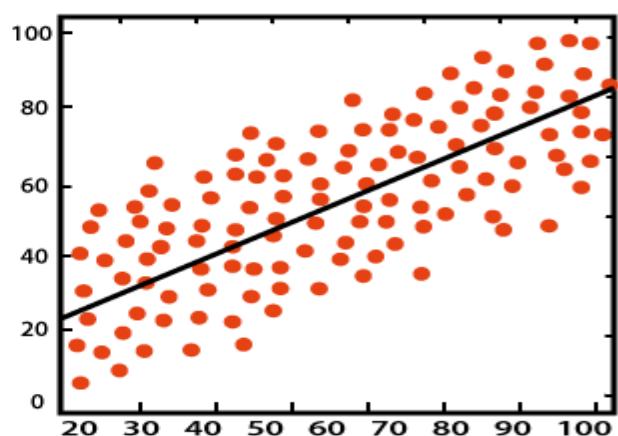
3.4 Regression vs Classification

Regression and Classification algorithms are Supervised Learning algorithms. Both the algorithms are used for prediction in Machine learning and work with the labelled datasets. But the difference between both is how they are used for different machine learning problems.

The main difference between Regression and Classification algorithms is that Regression algorithms are used to predict the continuous values such as price, salary, age, etc. whereas Classification algorithms are used to predict/Classify the discrete values such as Male or Female, True or False, Spam or Not Spam, etc.



Classification



Regression

Image: Classification vs Regression

Reference: <https://www.javatpoint.com/regression-vs-classification-in-machine-learning>

Classification

Classification is a process of finding a function which helps in dividing the dataset into classes based on different parameters. In Classification, a computer program is trained on the training dataset and based on that training, it categorizes the data into different classes.

The task of the classification algorithm is to find the mapping function to map the input (x) to the discrete output (y).

Example: The best example to understand the Classification problem is Email Spam Detection. The model is trained on the basis of millions of emails on different

parameters, and whenever it receives a new email, it identifies whether the email is spam or not. If the email is spam, then it is moved to the Spam folder.

Types of ML Classification Algorithms:

Classification Algorithms can be further divided into the following types:

- i) Logistic Regression
- ii) K-Nearest Neighbours
- iii) Support Vector Machines
- iv) Kernel SVM
- v) Naïve Bayes
- vi) Decision Tree Classification
- vii) Random Forest Classification

Regression:

Regression is a process of finding the correlations between dependent and independent variables. It helps in predicting the continuous variables such as prediction of Market Trends, prediction of House prices, etc.

The task of the Regression algorithm is to find the mapping function to map the input variable (x) to the continuous output variable (y).

Example: Suppose we want to do weather forecasting, so for this, we will use the Regression algorithm. In weather prediction, the model is trained on the past data, and once the training is completed, it can easily predict the weather for future days.

Types of Regression Algorithm:

- i) Simple Linear Regression
- ii) Multiple Linear Regression
- iii) Polynomial Regression
- iv) Support Vector Regression
- v) Decision Tree Regression
- vi) Random Forest Regression

Before starting the linear regression algorithm, let us get started with least square method which is used as a backbone of regression analysis.

3.5 Least Square Method

The least-squares regression method is a technique commonly used in Regression Analysis. It is a mathematical method used to find the best fit line that represents the relationship between an independent and dependent variable.

Line of Best Fit

Line of best fit is drawn to represent the relationship between 2 or more variables. To be more specific, the best fit line is drawn across a scatter plot of data points in order to represent a relationship between those data points.

Regression analysis makes use of mathematical methods such as least squares to obtain a definite relationship between the predictor variable (s) and the target variable. The least-squares method is one of the most effective ways used to draw the line of best fit. It is based on the idea that the square of the errors obtained must be minimized to the most possible extent and hence the name least squares method.

Least Squares Regression Example

Consider an example. Tom who is the owner of a retail shop, found the price of different T-shirts vs the number of T-shirts sold at his shop over a period of one week.

He tabulated this like:

Price of T-shirts in dollars (x)	# of T-shirts sold (y)
2	4
3	5
5	7
7	10
9	15

Let us use the concept of least squares regression to find the line of best fit for the above data.

Step 1: Calculate the slope 'm' by using the following formula, where n is the number of observations:

$$m = \frac{n \sum xy - (\sum x)(\sum y)}{n \sum x^2 - (\sum x)^2}$$

After you substitute the respective values, $m = 1.518$ approximately.

Step 2: Compute the y-intercept value

$$c = \frac{(\sum y \sum x^2) - \sum x \sum xy}{n(\sum x^2) - (\sum x)^2}$$

After you substitute the respective values, $c = 0.305$ approximately.

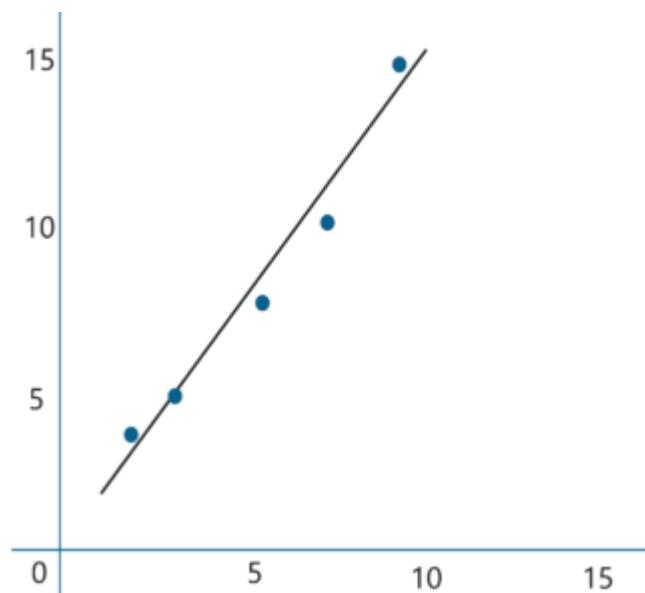
Step 3: Substitute the values in the final equation

$$y = mx + c$$

Once you substitute the values, it should look something like this:

Price of T-shirts in dollars (x)	# of T-shirts sold (y)	$Y = mx + c$	Error ($Y - y$)
2	4	3.341	-0.659
3	5	4.859	-0.141
5	7	7.895	0.895
7	10	10.931	0.931
9	15	13.967	-1.033

Let's construct a graph that represents the $y = mx + c$ line of best fit:



Now, Tom can use the above equation to estimate how many T-shirts of price \$8 can he sell at the retail shop.

$$y = 1.518 \times 8 + 0.305 = 12.45 \text{ T-shirts}$$

The least squares regression method works by minimizing the sum of the square of the errors as small as possible, hence the name least squares. Basically, the distance between the line of best fit and the error must be minimized as much as possible.

A few things to keep in mind before implementing the least squares regression method is:

- The data must be free of outliers because they might lead to a biased and wrongful line of best fit.
- The line of best fit can be drawn iteratively until you get a line with the minimum possible squares of errors.
- This method works well even with non-linear data.
- Technically, the difference between the actual value of 'y' and the predicted value of 'y' is called the Residual (denotes the error).

Linear Regression

Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as sales, salary, age, product price, etc.

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (x) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

The linear regression model provides a sloped straight line representing the relationship between the variables.

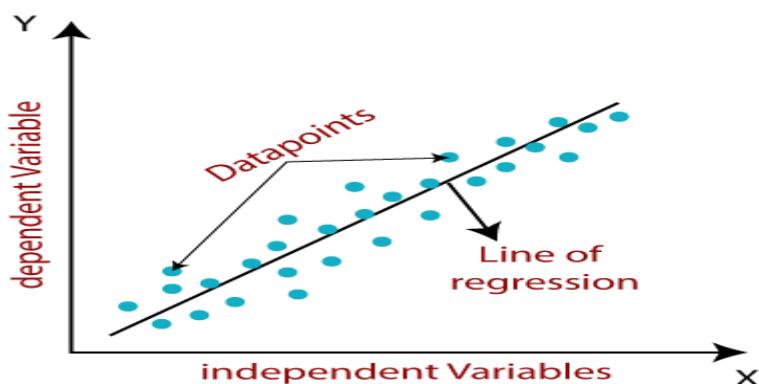


Image: Relationship between the variables in linear regression
 Reference:<https://www.javatpoint.com/linear-regression-in-machine-learning>

Mathematically, we can represent a linear regression as:

$$y = a_0 + a_1 x + \epsilon$$

Here,

y = Dependent Variable (Target Variable)

x = Independent Variable (predictor Variable)

a_0 = intercept of the line (Gives an additional degree of freedom)

a_1 = Linear regression coefficient (scale factor to each input value).

ϵ = random error

The values for x and y variables are training datasets for Linear Regression model representation.

Types of Linear Regression

Linear regression can be further divided into two types of the algorithm:

1. Simple Linear Regression:

If a single independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Simple Linear Regression.

2. Multiple Linear regression:

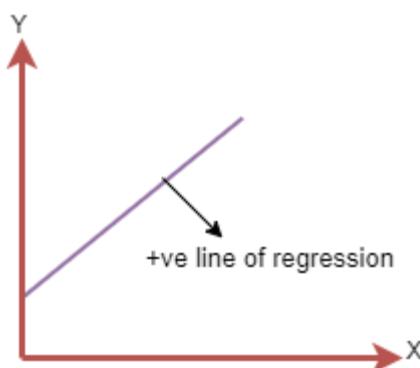
If more than one independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Multiple Linear Regression.

Linear Regression Line

A linear line showing the relationship between the dependent and independent variables is called a regression line. A regression line can show two types of relationship:

- Positive Linear Relationship:

If the dependent variable increases on the Y-axis and independent variable increases on X-axis, then such a relationship is termed as a Positive linear relationship.



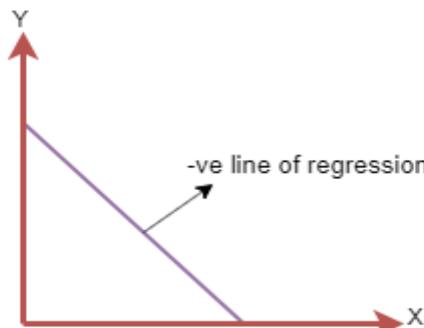
The line equation will be: $Y = a_0 + a_1 x$

Image:Positive linear relationship

Reference: <https://www.javatpoint.com/linear-regression-in-machine-learning>

- Negative Linear Relationship:

If the dependent variable decreases on the Y-axis and independent variable increases on the X-axis, then such a relationship is called a negative linear relationship.



The line of equation will be: $Y = -a_0 + a_1x$

Image:Negative linear relationship

Reference: <https://www.javatpoint.com/linear-regression-in-machine-learning>

Assumptions of Linear Regression

These are some formal checks while building a Linear Regression model, which ensures to get the best possible result from the given dataset.

Linear relationship between the features and target

Linear regression assumes the linear relationship between the dependent and independent variables.

Small or no multicollinearity between the features

Multicollinearity means high-correlation between the independent variables. Due to multicollinearity, it may be difficult to find the true relationship between the predictors and target variables. Or we can say, it is difficult to determine which predictor variable is affecting the target variable and which is not. So, the model assumes either little or no multicollinearity between the features or independent variables.

Homoscedasticity Assumption

Homoscedasticity is a situation when the error term is the same for all the values of independent variables. With homoscedasticity, there should be no clear pattern distribution of data in the scatter plot.

Normal distribution of error terms

Linear regression assumes that the error term should follow the normal distribution pattern. If error terms are not normally distributed, then confidence intervals will become either too wide or too narrow, which may cause difficulties in finding coefficients.

It can be checked using the q-q plot. If the plot shows a straight line without any deviation, which means the error is normally distributed.

No autocorrelations

The linear regression model assumes no autocorrelation in error terms. If there will be any correlation in the error term, then it will drastically reduce the accuracy of the model. Autocorrelation usually occurs if there is a dependency between residual errors.

Mathematical Intuition

When working with linear regression, our main goal is to find the best fit line that means the error between predicted values and actual values should be minimized. The best fit line will have the least error.

The different values for weights or the coefficient of lines (a_0, a_1) gives a different line of regression, so we need to calculate the best values for a_0 and a_1 to find the best fit line, so to calculate this we use cost function.

Cost function

The different values for weights or coefficient of lines (a_0, a_1) gives the different line of regression, and the cost function is used to estimate the values of the coefficient for the best fit line.

Cost function optimizes the regression coefficients or weights. It measures how a linear regression model is performing.

We can use the cost function to find the accuracy of the mapping function, which maps the input variable to the output variable. This mapping function is also known as Hypothesis function.

For Linear Regression, we use the Mean Squared Error (MSE) cost function, which is the average of squared error occurred between the predicted values and actual values. It can be written as:

For the above linear equation, MSE can be calculated as:

$$MSE = \frac{1}{N} \sum_{i=1}^n (y_i - (a_1 x_i + a_0))^2$$

where,

N=Total number of observations

y_i = Actual value

$(a_1 x_i + a_0)$ = Predicted value

Residuals:

The distance between the actual value and predicted values is called residual. If the observed points are far from the regression line, then the residual will be high, and so cost function will be high. If the scatter points are close to the regression line, then the residual will be small and hence the cost function.

Gradient Descent:

- Gradient descent is used to minimize the MSE by calculating the gradient of the cost function.
- A regression model uses gradient descent to update the coefficients of the line by reducing the cost function.
- It is done by a random selection of values of coefficient and then iteratively update the values to reach the minimum cost function.

Model Performance:

The Goodness of fit determines how the line of regression fits the set of observations. The process of finding the best model out of various models is called optimization. It can be achieved by the following method:

R-squared method:

- R-squared is a statistical method that determines the goodness of fit. It measures the strength of the relationship between the dependent and independent variables on a scale of 0-100%.
- The high value of R-square determines the less difference between the predicted values and actual values and hence represents a good model.
- It is also called a coefficient of determination, or coefficient of multiple determination for multiple regression.

It can be calculated from the below formula:

$$R - \text{squared} = \frac{\text{Explained Variation}}{\text{Total Variation}}$$

Ordinary Least Square Method

Ordinary least squares, or linear least squares, estimates the parameters in a regression model by minimizing the sum of the squared residuals. This method draws a line through the data points that minimizes the sum of the squared differences between the observed values and the corresponding fitted values.

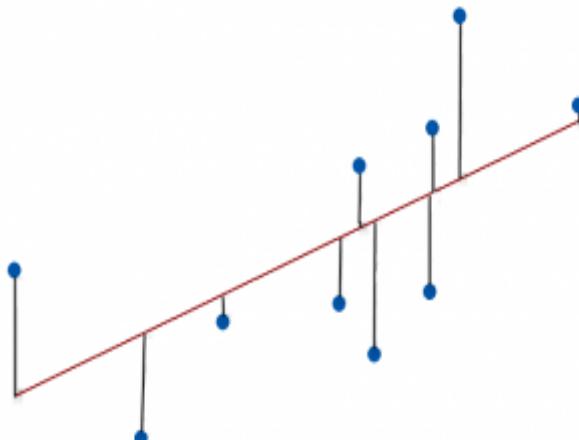


Image: Ordinary least squares line

Reference: <https://statisticsbyjim.com/glossary/ordinary-least-squares/#:~:text=Ordinary%20least%20squares%2C%20or%20linear,an%20the%20corresponding%20fitted%20values>

- The Ordinary Least Squares procedure seeks to minimize the sum of the squared residuals. This means that given a regression line through the data we calculate the distance from each data point to the regression line, square it, and sum all of the squared errors together. This is the quantity that ordinary least squares seeks to minimize.
- This approach treats the data as a matrix and uses linear algebra operations to estimate the optimal values for the coefficients. It means that all of the data must be available and you must have enough memory to fit the data and perform matrix operations.
- Ordinary Least Squares is a form of statistical regression used as a way to predict unknown values from an existing set of data. An example of a scenario in which one may use Ordinary Least Squares, or OLS, is in predicting shoe size from a data set that includes height and shoe size. Given the data, one can use the ordinary least squares formula to create a rate of change and predict shoe size, given a subject's height. In short, OLS takes an input, the independent variable, and produces an output, the dependent variable.

OLS method equation:

$$m = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$
$$b = \bar{y} - m * \bar{x}$$

x = independent variables

\bar{x} = average of independent variables

y = dependent variables

\bar{y} = average of dependent variables

Ordinary Least Squares method works for both univariate dataset which means single independent variables and single dependent variables and multi-variate dataset which contains a single independent variable set and multiple dependent variables sets.

PRACTICAL: Linear Regression in Sklearn

Let us see how to build a simple linear machine learning model for the Diabetes dataset. This dataset consists of Ten baseline variables, age, sex, body mass index, average blood pressure, and six blood serum measurements were obtained for each of $n = 442$ diabetes patients, as well as the response of interest, a quantitative measure of disease progression one year after baseline. We will apply Linear Regression for this considering one input feature and output target.

Link for Project:

<https://github.com/Edunet-Foundation/Tech-Saksham/blob/main/Tech%20Saksham%20AI%20Practical%20-%20Notebooks/Chapter%204/1.Linear%20Regression.ipynb>

We have discussed about regression, ordinary least square method, gradient descent optimization technique and metrics to measure the performance of regression. Now, let's get started with the cloud platform where we can train, test and deploy our machine learning model.

3.6 Logistic Regression

- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
- Logistic regression predicts the output of a categorical dependent variable. Therefore, the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.
- Logistic Regression is much similar to the Linear Regression except how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
- The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.

- Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.
- Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:

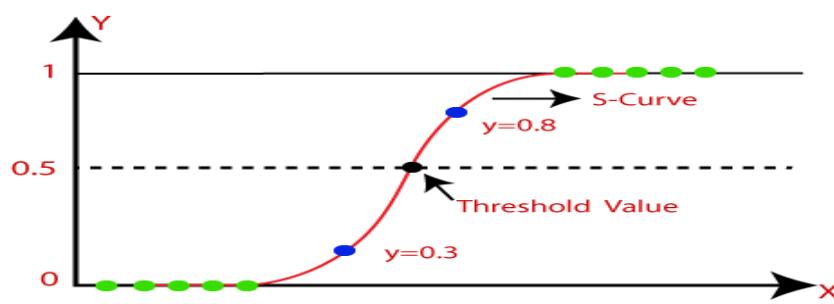


Image: logistic function

Reference:<https://www.javatpoint.com/logistic-regression-in-machine-learning>

Logistic Function (Sigmoid Function)

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1.
- The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.
- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

Assumptions for Logistic Regression

- The dependent variable must be categorical in nature.
- The independent variable should not have multi-collinearity.

Logistic Regression Equation

The Logistic regression equation can be obtained from the Linear Regression equation.

- We know the equation of the straight line can be written as:

$$y = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3 + \dots + b_n x_n$$

- In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by $(1-y)$:

$$\frac{y}{1-y}; \text{ 0 for } y=0, \text{ and infinity for } y=1$$

- But we need range between $-\infty$ to $+\infty$, then take logarithm of the equation it will become:

$$\log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \dots + b_nx_n$$

The above equation is the final equation for Logistic Regression.

Type of Logistic Regression

On the basis of the categories, Logistic Regression can be classified into three types:

Binomial: In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.

Multinomial: In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"

Ordinal: In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

3.7 Estimating probabilities

Probability is one of the foundations of machine learning (along with linear algebra and optimization). Probability is a measure of uncertainty. Probability applies to machine learning because in the real world, we need to make decisions with incomplete information. Hence, we need a mechanism to quantify uncertainty – which Probability provides us. Using probability, we can model elements of uncertainty such as risk in financial transactions and many other business processes. In contrast, in traditional programming, we work with deterministic problems i.e., the solution is not affected by uncertainty.

Probability of an event

Probability quantifies the likelihood or belief that an event will occur. Probability theory has three important concepts: Event - an outcome to which a probability is assigned; The Sample Space which represents the set of possible outcomes for the events and the Probability Function which maps a probability to an event. The probability function indicates the likelihood that the event being a part of the sample space is drawn. The probability distribution represents the shape or distribution of all events in the sample space. The probability of an event can be calculated directly by counting all the

occurrences of the event and dividing them by the total possible outcomes of the event. Probability is a fractional value and has a value in the range between 0 and 1, where 0 indicates no probability and 1 represents full probability.

Applications

It explores how probability can apply to machine learning

1. Sampling - Dealing with non-deterministic processes

Probability forms the basis of sampling. In machine learning, uncertainty can arise in many ways – for example - noise in data. Probability provides a set of tools to model uncertainty. Noise could arise due to variability in the observations, as a measurement error or from other sources. Noise effects both inputs and outputs.

Typically, we are given a dataset i.e., we do not have control on the creation and sampling process of the dataset. To cater for this lack of control over sampling, we split the data into train and test sets or we use resampling techniques. Hence, probability (through sampling) is involved when we have incomplete coverage of the problem domain.

2. Pattern recognition

Pattern recognition is a key part of machine learning. We can approach machine learning as a pattern recognition problem from a Bayesian standpoint. In Pattern Recognition – one takes a Bayesian view and presents approximate inference algorithms for situations where exact answers are not feasible. For the same reasons listed above, Probability theory is a key part of pattern recognition because it helps to cater for noise / uncertainty and for the finite size of the sample and also to apply Bayesian principles to machine learning.

3. Training - use in Maximum likelihood estimation

Many iterative machine learning techniques like Maximum likelihood estimation (MLE) are based on probability theory. MLE is used for training in models like linear regression, logistic regression and artificial neural networks.

4. Developing specific algorithms

Probability forms the basis of specific algorithms like Naive Bayes classifier

5. Hyperparameter optimization

In machine learning models such as neural networks, hyperparameters are tuned through techniques like grid search. Bayesian optimization can be also used for hyperparameter optimization.

6. Model evaluation

In binary classification tasks, we predict a single probability score. Model evaluation techniques require us to summarize the performance of a model based on predicted probabilities. For example – aggregation measures like log loss require the understanding of probability theory.

PRACTICAL: Logistic Regression using Sklearn

Let us consider dataset on our own. We can create linear separable dataset using make_classification class in Sklearn. Then Apply Logistic Regression to build a machine Learning model.

The Link for project is given below:

<https://github.com/Edunet-Foundation/Tech-Saksham/blob/main/Tech%20Saksham%20AI%20Practical%20-%20Notebooks/Chapter%204/2.Logistic%20Regression.ipynb>

After understanding the fundamentals of logistics regression, let us get started to build model using ML studio.

3.8 Decision Trees

- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.
- In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.
- A decision tree simply asks a question and based on the answer (Yes/No), it further split the tree into subtrees.

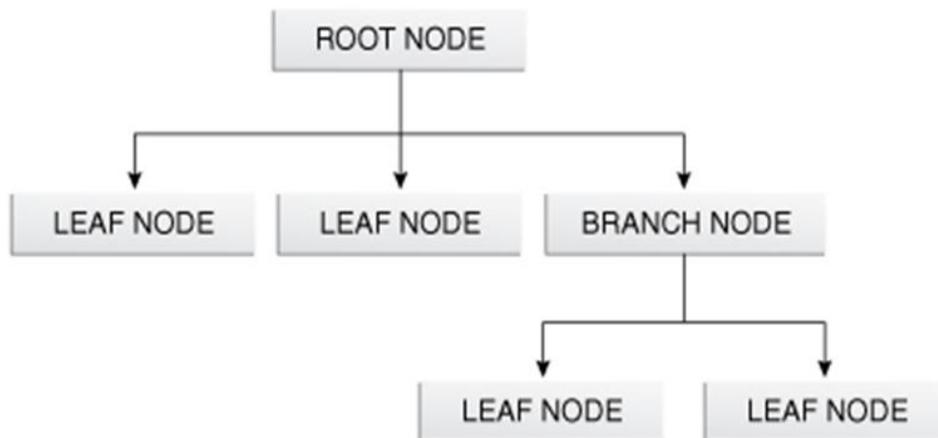


Image: General structure of a decision tree

Why use Decision Trees?

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.

The logic behind the decision tree can be easily understood because it shows a tree-like structure.

Decision Tree Terminologies

1. **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
2. **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
3. **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
4. **Branch/Sub Tree:** A tree formed by splitting the tree.
5. **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
6. **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

How does the Decision Tree algorithm Work?

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).

Step-3: Divide the S into subsets that contains possible values for the best attributes.

Step-4: Generate the decision tree node, which contains the best attribute.

Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Example: Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:

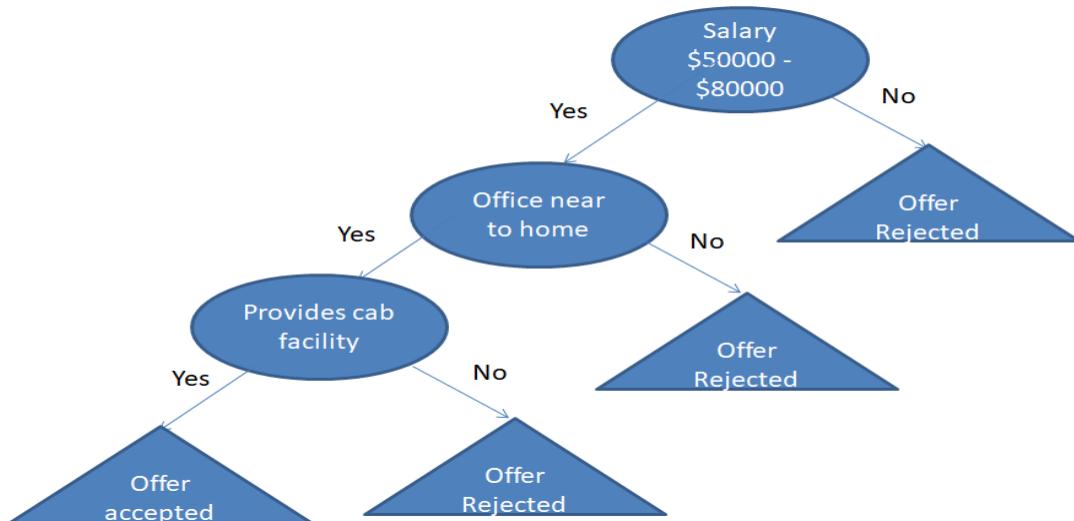


Image: Example

We understood that how to construct a decision tree but the main thing is that how we can select the best node for splitting, right! So, let us explore it.

3.9 Gini Impurity or Entropy

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a

technique which is called as Attribute selection measure or ASM. By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

1. Information Gain
2. Gini Index

Gini index and entropy are the criteria for calculating information gain. Decision tree algorithms use information gain to split a node.

Both gini and entropy are measures of impurity of a node. A node having multiple classes is impure whereas a node having only one class is pure.

Entropy in statistics is analogous to entropy in thermodynamics where it signifies disorder. If there are multiple classes in a node, there is disorder in that node.

$$Gini = 1 - \sum_{i=1}^n p^2(c_i)$$

$$Entropy = \sum_{i=1}^n -p(c_i) \log_2(p(c_i))$$

where $p(c_i)$ is the probability/percentage of class c_i in a node.

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index.
- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
- Information gain is the entropy of parent node minus sum of weighted entropies of child nodes.
- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

Information Gain= Entropy(S)- [(Weighted Avg) *Entropy (each feature)]

Weight of a child node is number of samples in the node/total samples of all child nodes. Similarly, information gain is calculated with gini score.

PRACTICAL: Decision Tree Classifier using Sklearn

The iris dataset is a classic and very easy multi-class classification dataset. The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters. Let us apply Decision Tree Classifier on the same.

The link for the project:

<https://github.com/Edunet-Foundation/Tech-Saksham/blob/main/Tech%20Saksham%20AI%20Practical%20-%20Notebooks/Chapter%204/3.Decision%20Tree.ipynb>

Now after exploring decision tree algorithm let's get started with another very popular classification algorithm called Support Vector Machine.

3.10 Linear SVM Classification

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

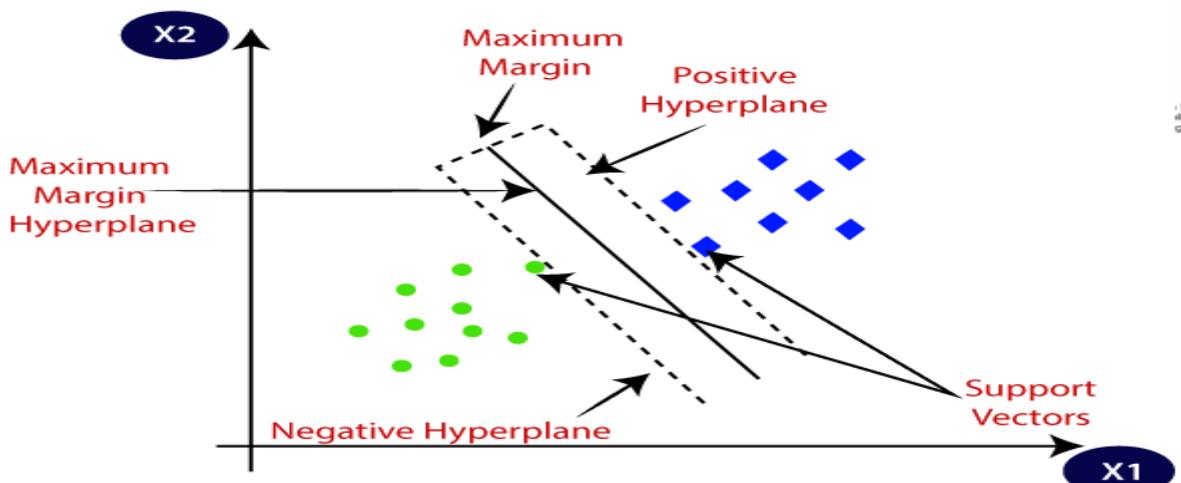


Image: Support Vector Machine

Reference: <https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm>

Example: SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:

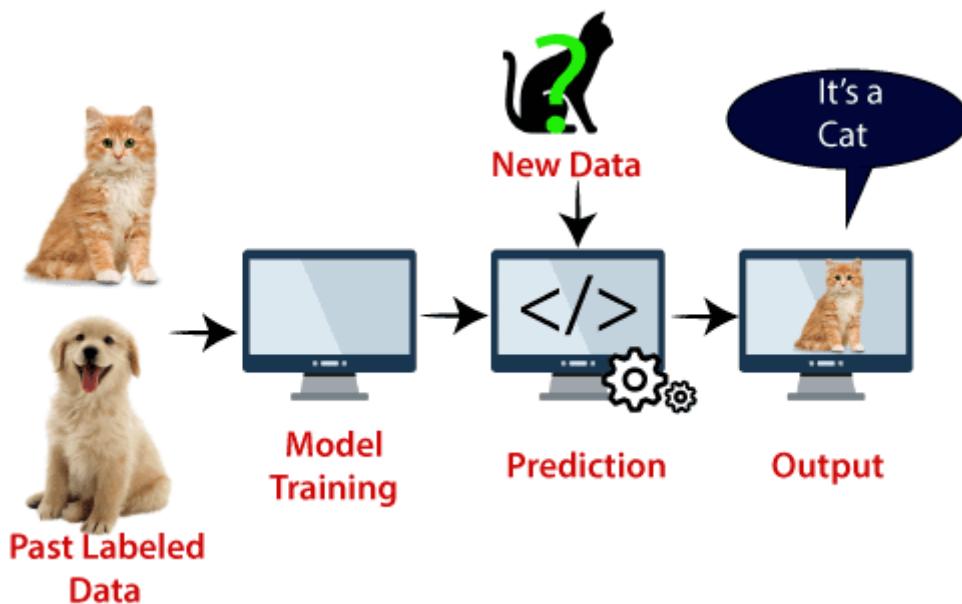


Image: Support Vector Machine Example
Reference: <https://cdn.inblog.in/user/uploads/x7PHukCnjBmaGEAQTDUKbiwszsZvJt.png>

SVM algorithm can be used for Face detection, image classification, text categorization, etc.

Types of SVM

SVM can be of two types:

1. Linear SVM

Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier

2. Non-linear SVM

Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

How does SVM works?

Linear SVM:

The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x_1 and x_2 . We want a classifier that can classify the pair (x_1, x_2) of coordinates in either green or blue. Consider the below image:

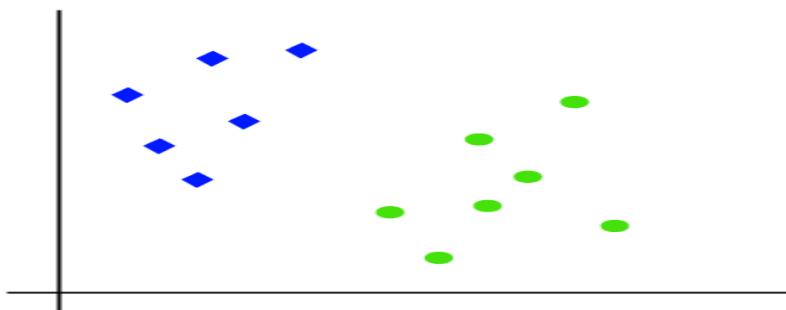


Image: Dataset with Green Blue tags

Reference: <https://static.javatpoint.com/tutorial/machine-learning/images/support-vector-machine-algorithm3.png>

So, as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:

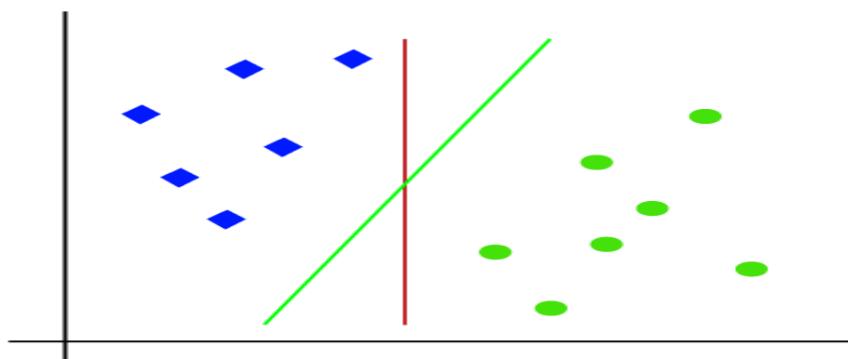


Image: Multiple lines to separate Classes

Reference: <https://static.javatpoint.com/tutorial/machine-learning/images/support-vector-machine-algorithm4.png>

Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a hyperplane. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as margin. And the goal of SVM is to maximize this margin. The hyperplane with maximum margin is called the optimal hyperplane.

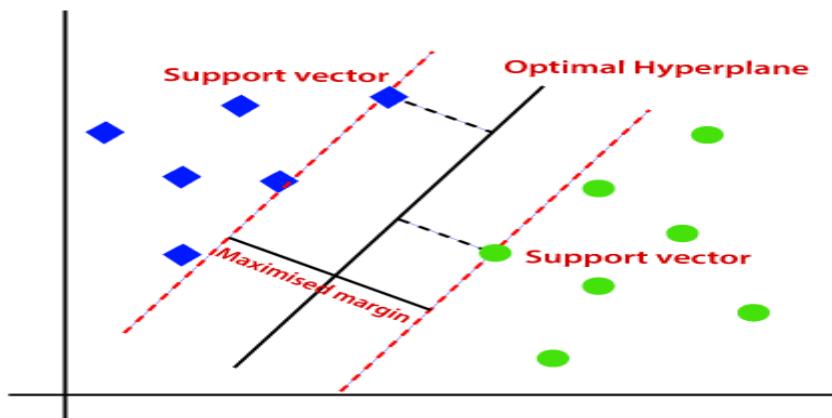


Image: Hyperplane
Reference: <https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm>

3.11 Support Vectors and Margins

Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.

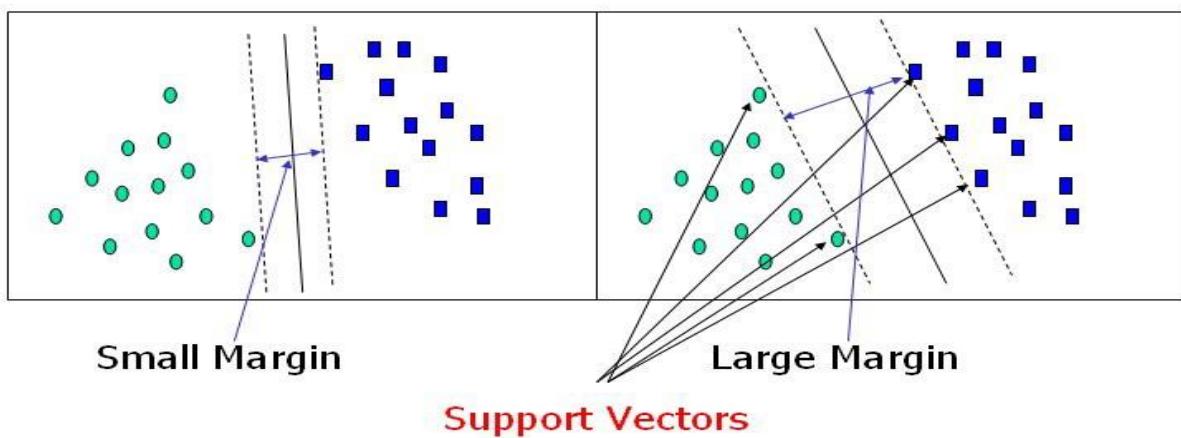


Image: Support Vectors
Reference: https://miro.medium.com/max/700/0*ecA4Ls8kBYSM5nza.jpg

Terminologies used in SVM

The points closest to the hyperplane are called as the support vector points and the distance of the vectors from the hyperplane are called the margins.

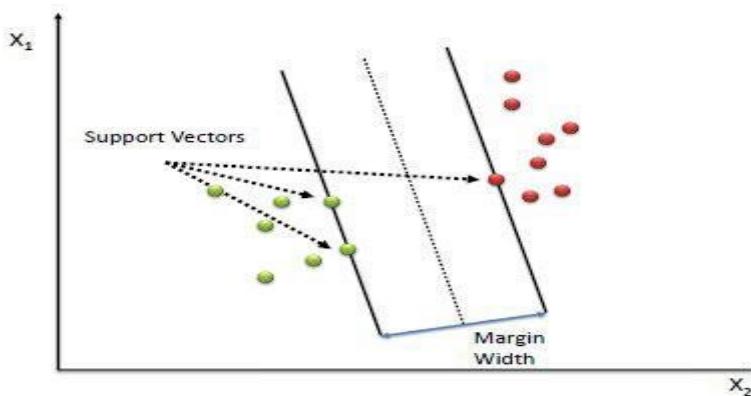


Image: Support vectors and margins

Reference: <https://towardsdatascience.com/support-vector-machines-svm-c9ef22815589>

The basic intuition to develop over here is that more the farther SV points, from the hyperplane, more is the probability of correctly classifying the points in their respective region or classes. SV points are very critical in determining the hyperplane because if the position of the vectors changes the hyperplane's position is altered. Technically this hyperplane can also be called as margin maximizing hyperplane.

Hard margin SVM

Assume 3 hyperplanes namely (π , π^+ , π^-) such that ' π^+ ' is parallel to ' π ' passing through the support vectors on the positive side and ' π^- ' is parallel to ' π ' passing through the support vectors on the negative side.

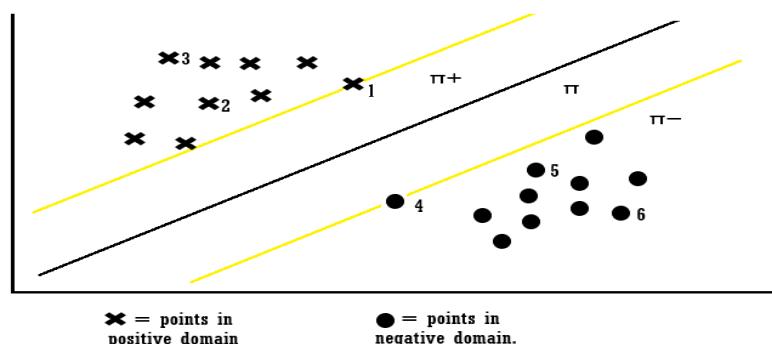


Image: Support vectors and Hyperplanes

Reference: https://miro.medium.com/max/687/1*ppsJ51I8o5kTC1q2opSjwQ.png

the equations of each hyperplane can be considered as:

$$\begin{aligned}\pi &= b + w^T X = 0 \\ \pi^+ &= b + w^T X = 1 \\ \pi^- &= b + w^T X = -1\end{aligned}$$

for the point X1:

$$\begin{aligned} y_1 &= 1 \\ y_1(w^T x_1 + b) &= 1 \end{aligned}$$

Explanation: when the point X1 we can say that point lies on the hyperplane and the equation determines that the product of our actual output and the hyperplane equation is 1 which means the point is correctly classified in the positive domain.

for the point X3:

$$\begin{aligned} y_1 &= 1 \\ y_1(w^T x_1 + b) &> 1 \end{aligned}$$

Explanation: when the point X3 we can say that point lies away from the hyperplane and the equation determines that the product of our actual output and the hyperplane equation is greater 1 which means the point is correctly classified in the positive domain.

for the point X4:

$$\begin{aligned} y_1 &= -1 \\ y_1(w^T x_1 + b) &= 1 \end{aligned}$$

Explanation: when the point X4 we can say that point lies on the hyperplane in the negative region and the equation determines that the product of our actual output and the hyperplane equation is equal to 1 which means the point is correctly classified in the negative domain.

for the point X6:

$$\begin{aligned} y_1 &= -1 \\ y_1(w^T x_1 + b) &> 1 \end{aligned}$$

Explanation: when the point X6 we can say that point lies away from the hyperplane in the negative region and the equation determines that the product of our actual output and the hyperplane equation is greater 1 which means the point is correctly classified in the negative domain.

Let's look into the constraints which are not classified:

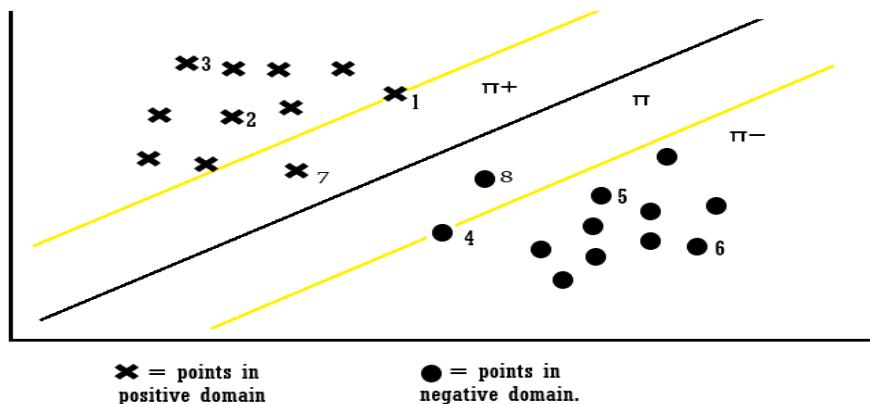


Image: Unclassified Constraints
 Reference: https://miro.medium.com/max/687/1*ppsJ51I8o5kTC1q2opSiwQ.png

for point X7:

$$\begin{aligned} y_1 &= 1 \\ y_1(w^T x_1 + b) & \\ (1)(1) &< \end{aligned}$$

Explanation: When $X_i = 7$ the point is classified incorrectly because for point 7 the $w^T + b$ will be smaller than one and this violates the constraints. So we found the misclassification because of constraint violation. Similarly, we can also say for points $X_i = 8$.

Thus, from the above examples, we can conclude that for any point X_i ,

if $y_i(w^T x_i + b) \geq 1$:

then X_i is correctly classified

else:

X_i is incorrectly classified.

So, we can see that if the points are linearly separable then only our hyperplane is able to distinguish between them and if any outlier is introduced then it is not able to separate them. So, these type of SVM is called as **hard margin SVM** (since we have very strict constraints to correctly classify each and every datapoint).

Soft margin SVM

We basically consider that the data is linearly separable and this might not be the case in real life scenario. We need an update so that our function may skip few outliers and be able to classify almost linearly separable points. For this reason, we introduce a new Slack variable (ξ) which is called X_i .

if we introduce ξ it into our previous equation we can rewrite it as

$$y_i(w^T x_i + b) \geq 1 - \xi_i$$

if $\xi_i = 0$,

the points can be considered as correctly classified.

else:

$\xi_i > 0$, Incorrectly classified points.

so, if $\xi_i > 0$ it means that X_i (variables) lies in incorrect dimension, thus we can think of ξ_i as an error term associated with X_i (variable). The average error can be given as;

$$\frac{1}{n} \sum_{i=1}^n \xi_i$$

thus, our objective, mathematically can be described as;

$$\underset{w,b}{\text{minimize}} \frac{1}{2} \|w\|^2 + \sum_{i=1}^n \zeta_i$$

such that $y_i (w^T \cdot X_i + b) \geq 1 - \zeta_i \quad \text{For all } i = 1, 2, \dots, n$

where $\xi_i = \zeta_i$

PRACTICAL: Linear SVM in Sklearn

Let us consider a Breast Cancer dataset. Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. We will apply SVM with Linear Kernel.

Link for the project:

<https://github.com/Edunet-Foundation/Tech-Saksham/blob/main/Tech%20Saksham%20AI%20Practical%20-%20Notebooks/Chapter%204/4.SVM.ipynb>

After understanding the working of support vector machine algorithm let us explore about how we can calculate the distance between two datapoints and various distance calculating techniques.

3.12 Different distance methods

Distance metrics play an important role in machine learning. They provide a strong foundation for several machine learning algorithms like k-nearest neighbors for supervised learning and k-means clustering for unsupervised learning. Different distance metrics are chosen depending upon the type of the data. So, it is important to know the various distance metrics and the intuitions behind it.

An effective distance metric improves the performance of our machine learning model, whether that's for classification tasks or clustering.

There are several measures of distance that can be used, and it is important to be aware of them while considering the best solution for a given situation to avoid errors and interpretation issues.

Types of Distance Metrics in Machine Learning

1. Euclidean Distance

- Euclidean Distance represents the shortest distance between two points.
- Euclidean distance formula can be used to calculate the distance between two data points in a plane.

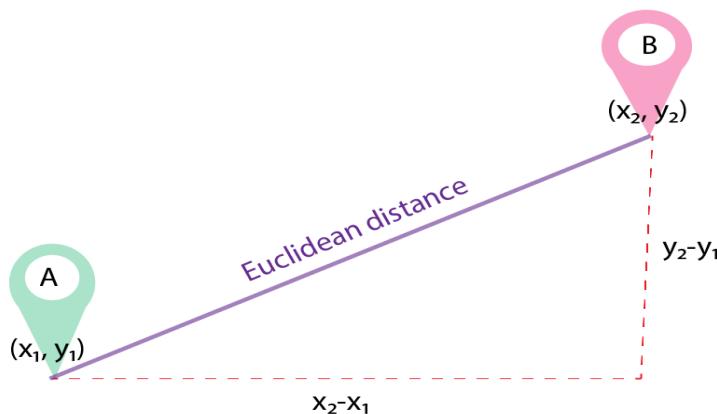


Image: Euclidean Distance

Reference:<https://medium.com/analytics-vidhya/role-of-distance-metrics-in-machine-learning-e43391a6bf2e>

- Euclidean distance is generally used when calculating the distance between two rows of data that have numerical values, such as floating point or integer values.
- If columns have values with differing scales, it should be normalized or standardized before calculating the Euclidean distance. Otherwise, columns that have large values will dominate the distance measure.
- Euclidean distance is calculated as the square root of the sum of the squared differences between the two vectors.

$$D_e = \left[\sum_{i=1}^n (p_i - q_i)^2 \right]^{1/2}$$

where,

n = number of dimensions

p_i, q_i = data points

2. Manhattan Distance

- Manhattan Distance is the sum of absolute differences between points across all the dimensions.
- We use Manhattan distance, also known as city block distance, or taxicab geometry if we need to calculate the distance between two data points in a grid-like path just like a chessboard or city blocks.
- The name taxicab refers to the intuition for what the measure calculates: the shortest path that a taxicab would take between city blocks (coordinates on the grid).

Let's say, we want to calculate the distance, d , between two data points- A and B.

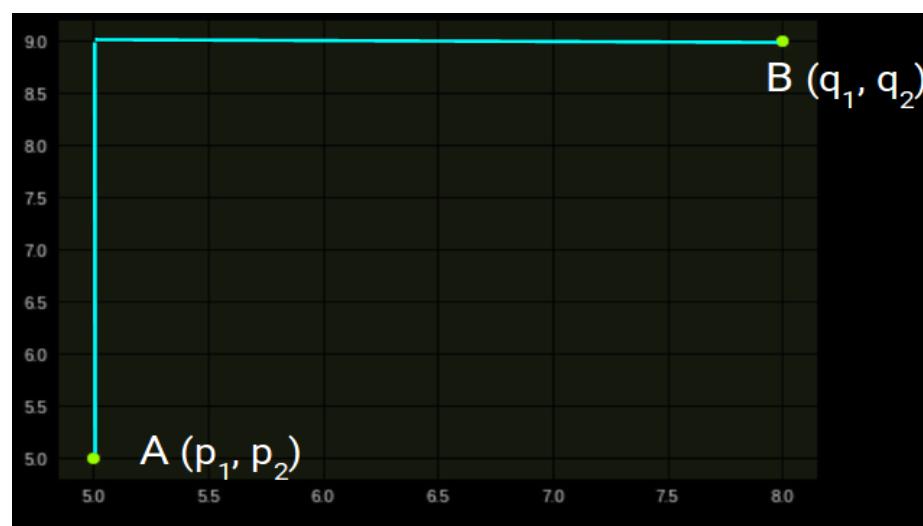


Image: Euclidean Distance between A and B

Reference: <https://medium.com/analytics-vidhya/role-of-distance-metrics-in-machine-learning-e43391a6bf2e>

Distance d will be calculated using an absolute sum of difference between its cartesian co-ordinates as below:

$$d = |p_1 - q_1| + |p_2 - q_2|$$

And the generalized formula for an n-dimensional space is given as:

$$D_m = \sum_{i=1}^n |p_i - q_i|$$

where,

n = number of dimensions

p_i, q_i = data points

The Manhattan Distance is preferred over the Euclidean distance metric as the dimension of the data increases. This occurs due to something known as the ‘curse of dimensionality’.

3. Minkowski Distance

- Minkowski Distance is the generalized form of Euclidean and Manhattan Distance.
- Minkowski Distance calculates the distance between two points.
- It is a generalization of the Euclidean and Manhattan distance measures and adds a parameter, called the “order” or “p”, that allows different distance measures to be calculated.
- The Minkowski distance measure is calculated as follows:

$$D = \left(\sum_{i=1}^n |P_i - Q_i|^p \right)^{1/p}$$

where “p” is the order parameter.

When p is set to 1, the calculation is the same as the Manhattan distance. When p is set to 2, it is the same as the Euclidean distance.

- p=1: Manhattan distance
- p=2: Euclidean distance

Intermediate values provide a controlled balance between the two measures.

- It is common to use Minkowski distance when implementing a machine learning algorithm that uses distance measures as it gives control over the type of distance measure used for real-valued vectors via a hyperparameter “p” that can be tuned.

4. Hamming Distance

Hamming Distance measures the similarity between two strings of the same length. The Hamming Distance between two strings of the same length is the number of positions at which the corresponding characters are different.

$$d = \min \{ d(x, y) : x, y \in C, x \neq y \}$$

So, this is how we can calculate the distance between datapoints, which is the core concept behind our next algorithm called K-Nearest Neighbors.

3.13 Geometric Intuition of K-NN

KNN assumes that all our data points are geometrically close to each other or in other words the neighbourhood points should be close to each other.

K-Nearest Neighbor (K-NN) Algorithm for Machine Learning

- K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears, it can be easily classified into a well suited category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.
- It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Example: Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So, for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs' images and based on the most similar features it will put it in either cat or dog category.



Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x_1 , so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:

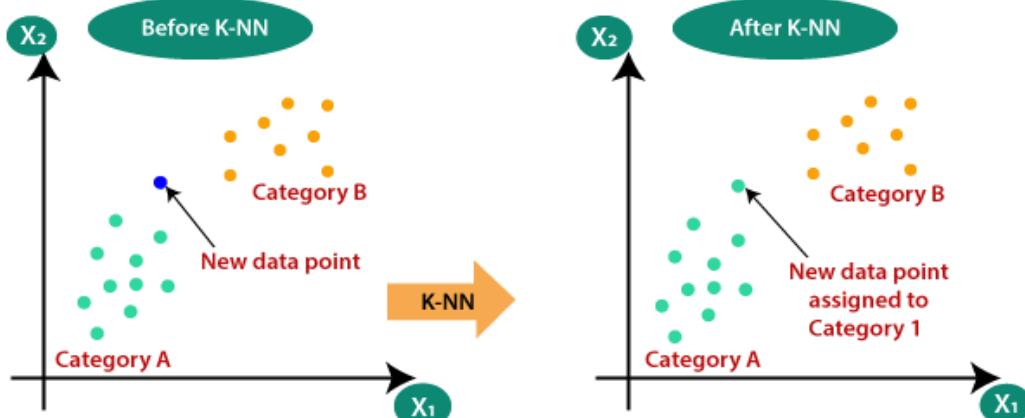


Image: Before and After KNN
Reference :<https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>

How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

Step-1: Select the number K of the neighbors

Step-2: Calculate the Euclidean distance of K number of neighbors

Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.

Step-4: Among these K neighbors, count the number of the data points in each category.

Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.

Step-6: Our model is ready. Suppose we have a new data point and we need to put it in the required category. Consider the image given below:

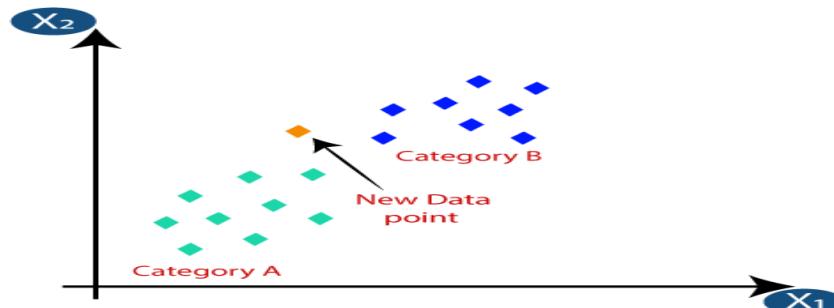


Image: New data point
Reference :<https://static.javatpoint.com/tutorial/machine-learning/images/k-nearest-neighbor-algorithm-for-machine-learning3.png>

- Firstly, we will choose the number of neighbors, so we will choose the k=5.

- Next, we will calculate the Euclidean distance between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:

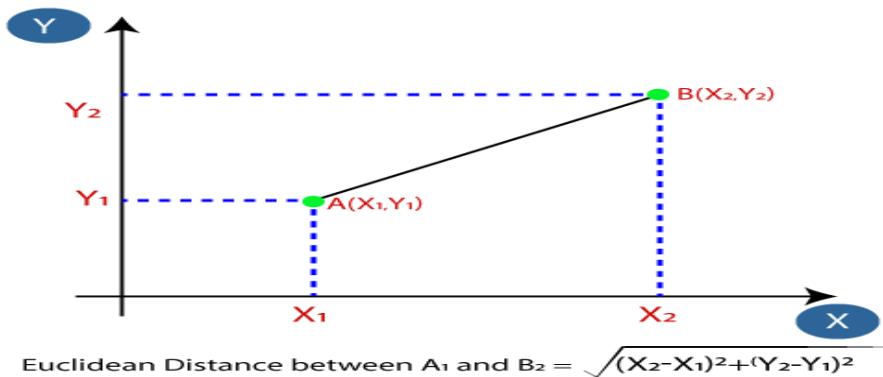


Image: Euclidean distance calculation

Reference: <https://static.javatpoint.com/tutorial/machine-learning/images/k-nearest-neighbor-algorithm-for-machine-learning4.png>

- By calculating the Euclidean distance we get the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the image below:

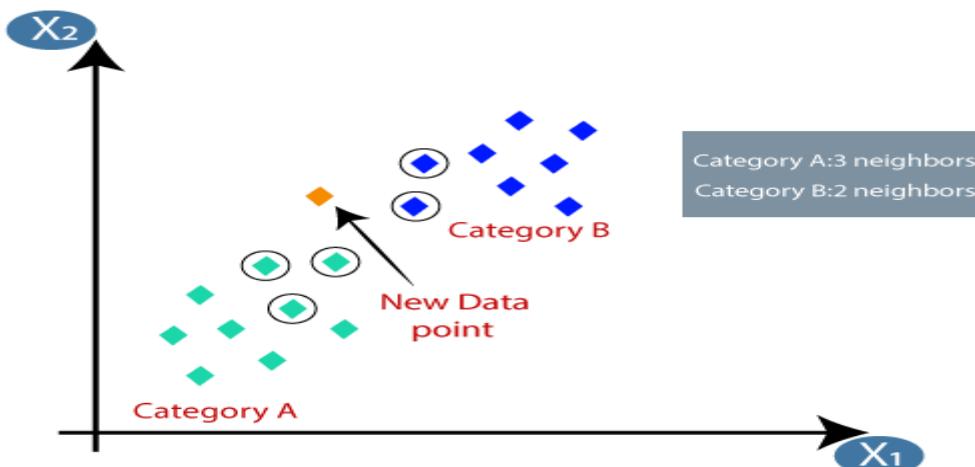


Image: Finding of Nearest neighbors

Reference: <https://static.javatpoint.com/tutorial/machine-learning/images/k-nearest-neighbor-algorithm-for-machine-learning4.png>

- As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

How to select the value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm:

- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.

- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.

Advantages of KNN Algorithm

- It is simple to implement.
- It is robust to the noisy training data.
- It can be more effective if the training data is large.

Disadvantages of KNN Algorithm

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

PRACTICAL: KNN using Sklearn

We will use the same dataset which was used earlier of iris flower and apply K NN Algorithm. Also we will see how to plot a decision boundary with respect to 3 categories using ListedColorMap class from matplotlib colors library. Here we can fine tune the parameters accordingly

<https://github.com/Edunet-Foundation/Tech-Saksham/blob/main/Tech%20Saksham%20AI%20Practical%20%20Notebooks/Chapter%204/6.Decision%20Boundary%20for%20KNN.ipynb>

We have explored the working of KNN algorithm, before starting out next algorithm lets us learn about probability concepts which are the foundations of our next Naive Bayes algorithm.

3.14 Probability Theory

Probability is the measure of the likelihood that an event will occur in a Random Experiment. Probability is quantified as a number between 0 and 1, where, loosely speaking, 0 indicates impossibility and 1 indicates certainty. The higher the probability of an event, the more likely it is that the event will occur.

Example:-

A simple example is the tossing of a fair (unbiased) coin. Since the coin is fair, the two outcomes ("heads" and "tails") are both equally probable; the probability of "heads" equals the probability of "tails"; and since no other outcomes are possible, the probability of either "heads" or "tails" is 1/2 (which could also be written as 0.5 or 50%).

Random Experiment

A random experiment is a physical situation whose outcome cannot be predicted until it is observed.

Sample Space

A sample space, is a set of all possible outcomes of a random experiment.

Example:

Random Experiment: Toss a fair coin once.

Sample Space: $\Omega = \{\text{Head, Tail}\}$

Random Variable

A random variable, is a variable whose possible values are numerical outcomes of a random experiment. There are two types of random variables.

1. Discrete Random Variable is one which may take on only a countable number of distinct values such as 0,1,2,3,4,..... Discrete random variables are usually (but not necessarily) counts.

2. Continuous Random Variable is one which takes an infinite number of possible values. Continuous random variables are usually measurements.

X, is defined as a function from the sample space to the real numbers.

$$X : \Omega \rightarrow \mathbb{R}$$

Example

For above random experiment

$$X = \begin{cases} 1 & \text{if Head} \\ 0 & \text{if Tail} \end{cases}$$

Conditional Probability

Conditional Probability is a measure of the probability of an event given that (by assumption, presumption, assertion or evidence) another event has already occurred. If the event of interest is A and the event B is known or assumed to have occurred, “the conditional probability of A given B”, is usually written as $P(A|B)$.

$$P(A | B) = \frac{P(A \cap B)}{P(B)}$$

Example

Suppose that somebody secretly rolls two fair six-sided dice,

Let D1 be the value rolled on die 1.

Let D2 be the value rolled on die 2.

What is the probability that D1 = 2

		D2					
		1	2	3	4	5	6
D1	1						
	2						
	3						
	4						
	5						
	6						

sample space = 36 outcomes. So D1 = 2 in exactly 6 of the 36 outcomes;

$P(D1=2) = 6/36 = 1/6$.

What is the probability that $D1+D2 \leq 5$

		D2					
		1	2	3	4	5	6
D1	1						
	2						
	3						
	4						
	5						
	6						

$D1+D2 \leq 5$ for exactly 10 of the same 36 outcomes, thus $P(D1+D2 \leq 5) = 10/36$.

What is the probability that $D1 = 2$ given that $D1+D2 \leq 5$

		D2					
		1	2	3	4	5	6
D1	1						
	2						
	3						
	4						
	5						
	6						

For $D1 = 2$, there are 3 out of 10 outcomes,

So the conditional probability $P(D1=2 | D1+D2 \leq 5) = 3/10 = 0.3$

Independence

Two events are said to be independent of each other, if the probability that one event occurs in no way affects the probability of the other event occurring, or in other words if we have observation about one event it doesn't affect the probability of the other. For Independent events A and B below is true

$$P(A, B) = P(A) * P(B) \quad \text{where} \quad P(A) \neq 0 \quad \text{and} \quad P(B) \neq 0$$

$$P(A | B) = P(A) \quad \text{and} \quad P(B | A) = P(A)$$

Example: -

Let's say you rolled a die and flipped a coin. The probability of getting any number face on the die is no way influences the probability of getting a head or a tail on the coin.

Conditional Independence

Two events A and B are conditionally independent given a third event C precisely if the occurrence of A and the occurrence of B are independent events in their conditional probability distribution given C. In other words, A and B are conditionally independent given C if and only if, given knowledge that C already occurred, knowledge of whether A occurs provides no additional information on the likelihood of B occurring, and knowledge of whether B occurs provides no additional information on the likelihood of A occurring.

$$P(A | C, B | C) = P(A | C) * P(B | C) \quad \text{where} \quad P(A | C) \neq 0 \quad \text{and} \quad P(B | C) \neq 0$$

Example: -

A box contains two coins, a regular coin and one fake two-headed coin ($P(H)=1$, $P(T)=0$). I choose a coin at random and toss it twice.

Let

A = First coin toss results in an HH.

B = Second coin toss results in an HH.

C = Coin 1 (regular) has been selected.

If C is already observed i.e., we already know whether a regular coin is selected or not, the event A and B becomes independent as the outcome of 1 doesn't affect the outcome of another event.

Expectation

The expectation of a random variable X is written as $E(X)$. If we observe N random values of X, then the mean of the N values will be approximately equal to $E(X)$ for large N. In more concrete terms, the expectation is what you would expect the outcome of an experiment to be on an average if you repeat the experiment a large number of time.

If X is a continuous random variable with p.d.f. $F_X(x)$

$$E[X] = \int_{-\infty}^{\infty} x F_X(x) dx$$

If X is a discrete random variable with probability function $F_X(x)$.

$$E[X] = \sum_x x F_X(x) = \sum_x x P(X = x)$$

If $f(X)$ is a function of X then the Expected value of $f(X)$

$$E[f(X)] = \sum_x f(x) F_X(x) = \sum_x f(x) P(X = x)$$

Example

Expected value when we roll a fair die (random experiment).

Let X represent the outcome of the experiment.

$X = \{1, 2, 3, 4, 5, 6\}$, Each of these has a probability of $1/6$ of occurring as it's a fair die.

So

$$\begin{aligned} E[X] &= 1 * 1/6 + 2 * 1/6 + 3 * 1/6 + 4 * 1/6 + 5 * 1/6 + 6 * 1/6 \\ &= 7/2 = 3.5 \end{aligned}$$

So the expectation is 3.5 . If you think about it, 3.5 is halfway between the possible values the die can take and so this is what you should have expected.

Variance

The variance of a random variable X is a measure of how concentrated the distribution of a random variable X is around its mean. It's defined as

$$Var[X] = E[X - E[X]^2] = E[X^2] - [E[X]]^2$$

Example

For the above experiment with die the variance is.

Let $f(X) = X^2$, Then using the last definition of Expectation

$$E[X^2] = E[f(X)] = \sum_x f(x)P(X=x)$$

$$E[X^2] = f(1)P(x=1) + f(2)P(x=2) + \dots + f(6)P(x=6) = 15.6$$

$$\begin{aligned} Var[X] &= E[X^2] - (E[X])^2 = 15.6 - 3.5^2 \\ &= 3.35 \end{aligned}$$

Probability Distribution

Is a mathematical function that maps the all possible outcomes of an random experiment with it's associated probability. It depends on the Random Variable X, whether it's discrete or continues.

1. Discrete Probability Distribution: The mathematical definition of a discrete probability function, $p(x)$, is a function that satisfies the following properties. This is referred as Probability Mass Function.

1. The probability that x can take a specific value is $p(x)$, i.e.

$$P[X=x]=p(x)$$

2. $p(x)$ is non-negative for all real x .

3. The sum of $p(x)$ over all possible values of x is 1, i.e.

$$\sum_x P(x) = 1$$

Example: For single coin flip the PMF can be represented in a table as below

X	Head	Tail
P(x)	0.5	0.5

2. Continuous Probability Distribution: The mathematical definition of a continuous probability function, $f(x)$, is a function that satisfies the following properties. This is referred as Probability Density Function.

1. The probability that x is in between two points a and b is P(x)

$$P[a \leq x \leq b] = \int_a^b f(x)dx$$

2. f(x) is non-negative for all real x.

3. The sum of p(x) over all possible values of x is 1, i.e.

$$\int_{-\infty}^{\infty} f(x)dx = 1$$

3.15 Naïve Bayes Classifier Algorithm

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.
- It is mainly used in text classification that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.
- Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

Why is it called Naïve Bayes?

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, which can be described as:

- Naïve: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- Bayes: It is called Bayes because it depends on the principle of Bayes' Theorem.

Bayes' Theorem

Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.

The formula for Bayes' theorem is given as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

where,

- $P(A|B)$ is Posterior probability: Probability of hypothesis A on the observed event B.
- $P(B|A)$ is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.
- $P(A)$ is Prior Probability: Probability of hypothesis before observing the evidence.
- $P(B)$ is Marginal Probability: Probability of Evidence.

Advantages of Naïve Bayes Classifier

- Naïve Bayes is one of the fast and easy ML algorithms to predict a class of datasets.
- It can be used for Binary as well as Multi-class Classifications.
- It performs well in Multi-class predictions as compared to the other Algorithms.
- It is the most popular choice for text classification problems.

Disadvantages of Naïve Bayes Classifier

- Naive Bayes assumes that all features are independent or unrelated, so it cannot learn the relationship between features.

Applications of Naïve Bayes Classifier

- It is used for Credit Scoring.
- It is used in medical data classification.
- It can be used in real-time predictions because Naïve Bayes Classifier is an eager learner.
- It is used in Text classification such as Spam filtering and Sentiment analysis.

Types of Naïve Bayes Model

There are three types of Naive Bayes Model, which are given below:

1. Gaussian: The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.
2. Multinomial: The Multinomial Naïve Bayes classifier is used when the data is multinomial distributed. It is primarily used for document classification problems, it means a particular document belongs to which category such as Sports, Politics, education, etc.

3. Bernoulli: The Bernoulli classifier works similar to the Multinomial classifier, but the predictor variables are the independent Booleans variables. Such as if a particular word is present or not in a document. This model is also famous for document classification tasks.

PRACTICAL: Naïve Bayes using Sklearn

Let us consider a dataset about sports. We have 2 features telling us about weather and humidity. The target what we need to find is whether a person will play or not . Let us use GaussianNB to predict the output.

Link for the project : https://github.com/Edunet-Foundation/Tech-Saksham/blob/main/Tech%20Saksham%20AI%20Practical%20-%20Notebooks/Chapter%204/7.Naive_Bayes.py

3.16 Bag of Words Approach

The bag-of-words model is a way of representing text data when modeling text with machine learning algorithms. The bag-of-words model is simple to understand and implement and has seen great success in problems such as language modeling and document classification.

A bag-of-words model, or BoW for short, is a way of extracting features from text for use in modeling, such as with machine learning algorithms. The approach is very simple and flexible, and can be used in a myriad of ways for extracting features from documents. A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:

- A vocabulary of known words.
- A measure of the presence of known words.

It is called a “*bag*” of words, because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document.

Here's a sample of reviews about a particular horror movie.

- Review 1: This movie is very scary and long
- Review 2: This movie is not scary and is slow
- Review 3: This movie is spooky and good

We can see that there are some contrasting reviews about the movie as well as the length and pace of the movie. Imagine looking at a thousand reviews like these.

Clearly, there is a lot of interesting insights we can draw from them and build upon them to gauge how well the movie performed.

However, as we saw above, we cannot simply give these sentences to a machine learning model and ask it to tell us whether a review was positive or negative. We need to perform certain text preprocessing steps.

We will first build a vocabulary from all the unique words in the above three reviews. The vocabulary consists of these 11 words: 'This', 'movie', 'is', 'very', 'scary', 'and', 'long', 'not', 'slow', 'spooky', 'good'. We can now take each of these words and mark their occurrence in the three movie reviews above with 1s and 0s. This will give us 3 vectors for 3 reviews:

	1 This	2 movie	3 is	4 very	5 scary	6 and	7 long	8 not	9 slow	10 spooky	11 good	Length of the review(in words)
Review 1	1	1	1	1	1	1	1	0	0	0	0	7
Review 2	1	1	2	0	0	1	1	0	1	0	0	8
Review 3	1	1	1	0	0	0	1	0	0	1	1	6

Image: Bag of Words Approach

Reference: <https://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/BoWBag-of-Words-model-2.png>

Vector of Review 1: [1 1 1 1 1 1 1 0 0 0 0]

Vector of Review 2: [1 1 2 0 0 1 1 0 1 0 0]

Vector of Review 3: [1 1 1 0 0 0 1 0 0 1 1]

And that's the core idea behind a Bag of Words (BoW) model.

Term Frequency-Inverse Document Frequency (TF-IDF)

Term frequency-inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.

Term Frequency (TF)

It is a measure of how frequently a term, t, appears in a document, d:

$$tf_{t,d} = \frac{n_{t,d}}{\text{Number of terms in the document}}$$

Here, in the numerator, n is the number of times the term “t” appears in the document “d”. Thus, each document and term would have its own TF value. Take the same vocabulary we had built in the Bag-of-Words model to show how to calculate the TF for Review #2:

Review 2: This movie is not scary and is slow

Here,

- Vocabulary: ‘This’, ‘movie’, ‘is’, ‘very’, ‘scary’, ‘and’, ‘long’, ‘not’, ‘slow’, ‘spooky’, ‘good’
- Number of words in Review 2 = 8
- TF for the word ‘this’ = (number of times ‘this’ appears in review 2)/ (number of terms in review 2) = 1/8

Similarly,

- $\text{TF}(\text{'movie'}) = 1/8$
- $\text{TF}(\text{'is'}) = 2/8 = 1/4$
- $\text{TF}(\text{'very'}) = 0/8 = 0$
- $\text{TF}(\text{'scary'}) = 1/8$
- $\text{TF}(\text{'and'}) = 1/8$
- $\text{TF}(\text{'long'}) = 0/8 = 0$
- $\text{TF}(\text{'not'}) = 1/8$
- $\text{TF}(\text{'slow'}) = 1/8$
- $\text{TF}(\text{'spooky'}) = 0/8 = 0$
- $\text{TF}(\text{'good'}) = 0/8 = 0$

We can calculate the term frequencies for all the terms and all the reviews in this manner:

Term	Review 1	Review 2	Review 3	TF (Review 1)	TF (Review 2)	TF (Review 3)
This	1	1	1	1/7	1/8	1/6
movie	1	1	1	1/7	1/8	1/6
is	1	2	1	1/7	1/4	1/6
very	1	0	0	1/7	0	0
scary	1	1	0	1/7	1/8	0
and	1	1	1	1/7	1/8	1/6
long	1	0	0	1/7	0	0
not	0	1	0	0	1/8	0
slow	0	1	0	0	1/8	0
spooky	0	0	1	0	0	1/6
good	0	0	1	0	0	1/6

Image: Term Frequency (TF)

Reference: <https://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/TF-matrix-1.png>

Inverse Document Frequency (IDF)

IDF is a measure of how important a term is. We need the IDF value because computing just the TF alone is not sufficient to understand the importance of words.

$$idf_t = \log \frac{\text{number of documents}}{\text{number of documents with term 't'}}$$

We can calculate the IDF values for all the words in Review 2:

$\text{IDF}(\text{'this'}) = \log (\text{number of documents}/\text{number of documents containing the word 'this'}) = \log (3/3) = \log (1) = 0$

Similarly,

- $\text{IDF}(\text{'movie'}) = \log (3/3) = 0$
- $\text{IDF}(\text{'is'}) = \log (3/3) = 0$
- $\text{IDF}(\text{'not'}) = \log (3/1) = \log (3) = 0.48$
- $\text{IDF}(\text{'scary'}) = \log (3/2) = 0.18$
- $\text{IDF}(\text{'and'}) = \log (3/3) = 0$
- $\text{IDF}(\text{'slow'}) = \log (3/1) = 0.48$

We can calculate the IDF values for each word like this. Thus, the IDF values for the entire vocabulary would be:

Term	Review 1	Review 2	Review 3	IDF
This	1	1	1	0.00
movie	1	1	1	0.00
is	1	2	1	0.00
very	1	0	0	0.48
scary	1	1	0	0.18
and	1	1	1	0.00
long	1	0	0	0.48
not	0	1	0	0.48
slow	0	1	0	0.48
spooky	0	0	1	0.48
good	0	0	1	0.48

Image: IDF values for each word

Reference: <https://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/IDF-matrix.png>

Hence, we see that words like "is", "this", "and", etc., are reduced to 0 and have little importance; while words like "scary", "long", "good", etc. are words with more importance and thus have a higher value.

We can now compute the TF-IDF score for each word in the corpus. Words with a higher score are more important, and those with a lower score are less important:

$$(tf_idf)_{t,d} = tf_{t,d} * idf_t$$

We can now calculate the TF-IDF score for every word in Review 2:

$$\text{TF-IDF ('this', Review 2)} = \text{TF ('this', Review 2)} * \text{IDF('this')} = 1/8 * 0 = 0$$

Similarly,

- TF-IDF ('movie', Review 2) = 1/8 * 0 = 0
- TF-IDF ('is', Review 2) = 1/4 * 0 = 0
- TF-IDF ('not', Review 2) = 1/8 * 0.48 = 0.06
- TF-IDF ('scary', Review 2) = 1/8 * 0.18 = 0.023
- TF-IDF ('and', Review 2) = 1/8 * 0 = 0
- TF-IDF ('slow', Review 2) = 1/8 * 0.48 = 0.06

Similarly, we can calculate the TF-IDF scores for all the words with respect to all the reviews:

Term	Review 1	Review 2	Review 3	IDF	TF-IDF (Review 1)	TF-IDF (Review 2)	TF-IDF (Review 3)
This	1	1	1	0.00	0.000	0.000	0.000
movie	1	1	1	0.00	0.000	0.000	0.000
is	1	2	1	0.00	0.000	0.000	0.000
very	1	0	0	0.48	0.068	0.000	0.000
scary	1	1	0	0.18	0.025	0.022	0.000
and	1	1	1	0.00	0.000	0.000	0.000
long	1	0	0	0.48	0.068	0.000	0.000
not	0	1	0	0.48	0.000	0.060	0.000
slow	0	1	0	0.48	0.000	0.060	0.000
spooky	0	0	1	0.48	0.000	0.000	0.080
good	0	0	1	0.48	0.000	0.000	0.080

Image: TF-IDF scores

Reference: https://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/TF_IDF-matrix.png

We have now obtained the TF-IDF scores for our vocabulary. TF-IDF also gives larger values for less frequent words and is high when both IDF and TF values are high i.e. the word is rare in all the documents combined but frequent in a single document.

PRACTICAL: TFIDF using Sklearn:

We will work on using the TfidfVectorizer to learn vocabulary and inverse document frequencies across 2 small documents and then encode one of those documents. A vocabulary of multiple words is learned from the documents and each word is assigned a unique integer index in the output vector.

Link:

[https://github.com/Edunet-Foundation/Tech-Saksham/blob/main/Tech%20Saksham%20AI%20Practical%20-%20Notebooks/Chapter%204/9.Bag%20of%20Word%20with%20Vectorization%20Technique\(TFIDF\).ipynb](https://github.com/Edunet-Foundation/Tech-Saksham/blob/main/Tech%20Saksham%20AI%20Practical%20-%20Notebooks/Chapter%204/9.Bag%20of%20Word%20with%20Vectorization%20Technique(TFIDF).ipynb)

PRACTICAL: Spam Ham Using using BOW concept:

Spam Ham classification is one of the classical application of Machine Learning. There are two types of data present in this repository, which is **ham** (non-spam) and **spam** data. Furthermore, in the ham data, there are easy and hard, which means there is some non-spam data that has a very high similarity with spam data. This might pose a difficulty for our system to make a decision. Let us use the concept of Bag of Words to convert Text to Numbers and then apply Classification algorithm on it.

Link for the project:

https://github.com/Edunet-Foundation/Tech-Saksham/blob/main/Tech%20Saksham%20AI%20Practical%20-%20Notebooks/Chapter%204/10.%20Spam_Ham_Demonstration.ipynb

3.17 Unsupervised Learning

Unsupervised transformations of a dataset are algorithms that create a new representation of the data which might be easier for humans or other machine learning algorithms to understand compared to the original representation of the data. A common application of unsupervised transformations is dimensionality reduction, which takes a high-dimensional representation of the data, consisting of many features, and finds a new way to represent this data that summarizes the essential characteristics with fewer features. A common application for dimensionality reduction is reduction to two dimensions for visualization purposes.

Another application for unsupervised transformations is finding the parts or components that “make up” the data. An example of this is topic extraction on collections of text documents. Here, the task is to find the unknown topics that are talked about in each document, and to learn what topics appear in each document. This can be useful for tracking the discussion of themes like elections, gun control, or pop stars on social media.

Clustering algorithms, on the other hand, partition data into distinct groups of similar items. Consider the example of uploading photos to a social media site. To allow you to organize your pictures, the site might want to group together pictures that show the same person. However, the site doesn’t know which pictures show whom, and it doesn’t know how many different people appear in your photo collection. A sensible approach would be to extract all the faces and divide them into groups of faces that

look similar. Hopefully, these correspond to the same person, and the images can be grouped together for you. [11.1]

Challenges in Unsupervised Learning

A major challenge in unsupervised learning is evaluating whether the algorithm learned something useful. Unsupervised learning algorithms are usually applied to data that does not contain any label information, so we don't know what the right output should be. Therefore, it is very hard to say whether a model "did well." For example, our hypothetical clustering algorithm could have grouped together all the pictures that show faces in profile and all the full-face pictures. This would certainly be a possible way to divide a collection of pictures of people's faces, but it's not the one we were looking for. However, there is no way for us to "tell" the algorithm what we are looking for, and often the only way to evaluate the result of an unsupervised algorithm is to inspect it manually. As a consequence, unsupervised algorithms are used often in an exploratory setting, when a data scientist wants to understand the data better, rather than as part of a larger automatic system. Another common application for unsupervised algorithms is as a pre-processing step for supervised algorithms. Learning a new representation of the data can sometimes improve the accuracy of supervised algorithms, or can lead to reduced memory and time consumption.

Why Unsupervised Learning?

Here, are prime reasons for using Unsupervised Learning:

- Unsupervised machine learning finds all kind of unknown patterns in data.
- Unsupervised methods help you to find features which can be useful for categorization.
- It takes place in real time, so all the input data is to be analysed and labelled in the presence of learners.
- It is easier to get unlabelled data from a computer than labelled data, which needs manual intervention.

Unsupervised Learning can be classified into two categories:

- *Parametric Unsupervised Learning*

In this case, we assume a parametric distribution of data. It assumes that sample data comes from a population that follows a probability distribution based on a fixed set of parameters. Theoretically, in a normal family of distributions, all members have the same shape and are *parameterized* by mean and standard deviation. That means if you know the mean and standard deviation, and that the distribution is normal, you know the probability of any future observation. Parametric Unsupervised Learning involves construction of

Gaussian Mixture Models and using Expectation-Maximization algorithm to predict the class of the sample in question. This case is much harder than the standard supervised learning because there are no answer labels available and hence there is no correct measure of accuracy available to check the result.

- ***Non-parametric Unsupervised Learning***

In non-parameterized version of unsupervised learning, the data is grouped into clusters, where each cluster (hopefully) says something about categories and classes present in the data. This method is commonly used to model and analyze data with small sample sizes. Unlike parametric models, nonparametric models do not require the modeler to make any assumptions about the distribution of the population, and so are sometimes referred to as a distribution-free method.

Let us get started with our first type of unsupervised machine learning algorithm called clustering.

3.18 Clustering

It is basically a type of unsupervised learning method. An unsupervised learning method is a method in which we draw references from datasets consisting of input data without labelled responses. Generally, it is used as a process to find meaningful structure, explanatory underlying processes, generative features, and groupings inherent in a set of examples.

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects on the basis of similarity and dissimilarity between them.

For ex– The data points in the graph below clustered together can be classified into one single group. We can distinguish the clusters, and we can identify that there are 3 clusters in the below picture.[11.4]

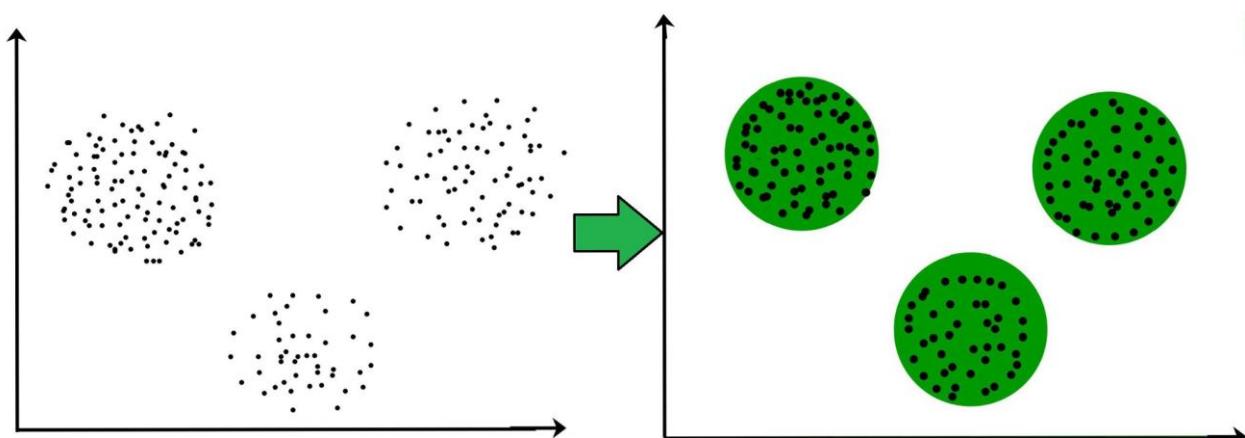


Fig : Clustering

Reference:<https://www.geeksforgeeks.org/clustering-in-machine-learning/#:~:text=Clustering%20is%20the%20task%20of,data%20points%20in%20other%20groups.>

It is not necessary for clusters to be a spherical. Such as :

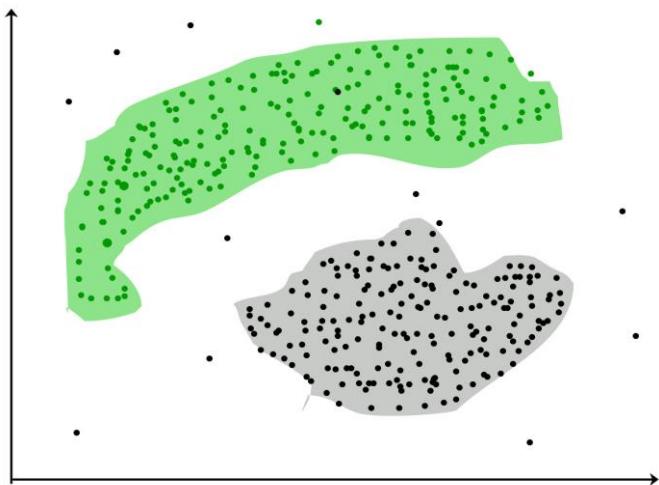


Fig : Unspesical Clustering

Reference : <https://www.geeksforgeeks.org/clustering-in-machine-learning/#:~:text=Clustering%20is%20the%20task%20of,data%20points%20in%20other%20groups.>

Why Clustering?

Clustering is very much important as it determines the intrinsic grouping among the unlabelled data present. There are no criteria for a good clustering. It depends on the user, what is the criteria they may use which satisfy their need. For instance, we could be interested in finding representatives for homogeneous groups (data reduction), in finding “natural clusters” and describe their unknown properties (“natural” data types), in finding useful and suitable groupings (“useful” data classes) or in finding unusual data objects (outlier detection). This algorithm must make some assumptions which constitute the similarity of points and each assumption make different and equally valid clusters.

Clustering Methods:

- **Density-Based Methods:** These methods consider the clusters as the dense region having some similarity and different from the lower dense region of the space. These methods have good accuracy and ability to merge two clusters. Example *DBSCAN (Density-Based Spatial Clustering of Applications with Noise)* , *OPTICS (Ordering Points to Identify Clustering Structure)* etc.
- **Hierarchical Based Methods:** The clusters formed in this method forms a tree-type structure based on the hierarchy. New clusters are formed using the

previously formed one. It is divided into two category Agglomerative (*bottom up approach*) and Divisive (*top down approach*). Some examples: *CURE (Clustering Using Representatives)*, *BIRCH (Balanced Iterative Reducing Clustering and using Hierarchies)* etc.

- **Partitioning Methods:** These methods partition the objects into k clusters and each partition forms one cluster. This method is used to optimize an objective criterion similarity function such as when the distance is a major parameter. For e.g. *K-means*, *CLARANS (Clustering Large Applications based upon Randomized Search)* etc.
- **Grid-based Methods:** In this method the data space is formulated into a finite number of cells that form a grid-like structure. All the clustering operation done on these grids are fast and independent of the number of data objects example *STING (Statistical Information Grid)*, *wave cluster*, *CLIQUE (CLustering In Quest)* etc.

3.19 K-means clustering

K-means clustering is one of the simplest and popular unsupervised machine learning algorithms. Typically, unsupervised algorithms make inferences from datasets using only input vectors without referring to known, or labelled, outcomes.

A cluster refers to a collection of data points aggregated together because of certain similarities. You'll define a target number k , which refers to the number of centroids you need in the dataset. A centroid is the imaginary or real location representing the center of the cluster. Every data point is allocated to each of the clusters through reducing the in-cluster sum of squares. In other words, the K-means algorithm identifies k number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible. The '*means*' in the K-means refers to averaging of the data; that is, finding the centroid.

How the K-means algorithm works

To process the learning data, the K-means algorithm in data mining starts with a first group of randomly selected centroids, which are used as the beginning points for every cluster, and then performs iterative (repetitive) calculations to optimize the positions of the centroids.

It halts creating and optimizing clusters when either:

- The centroids have stabilized — there is no change in their values because the clustering has been successful.
- The defined number of iterations has been achieved.

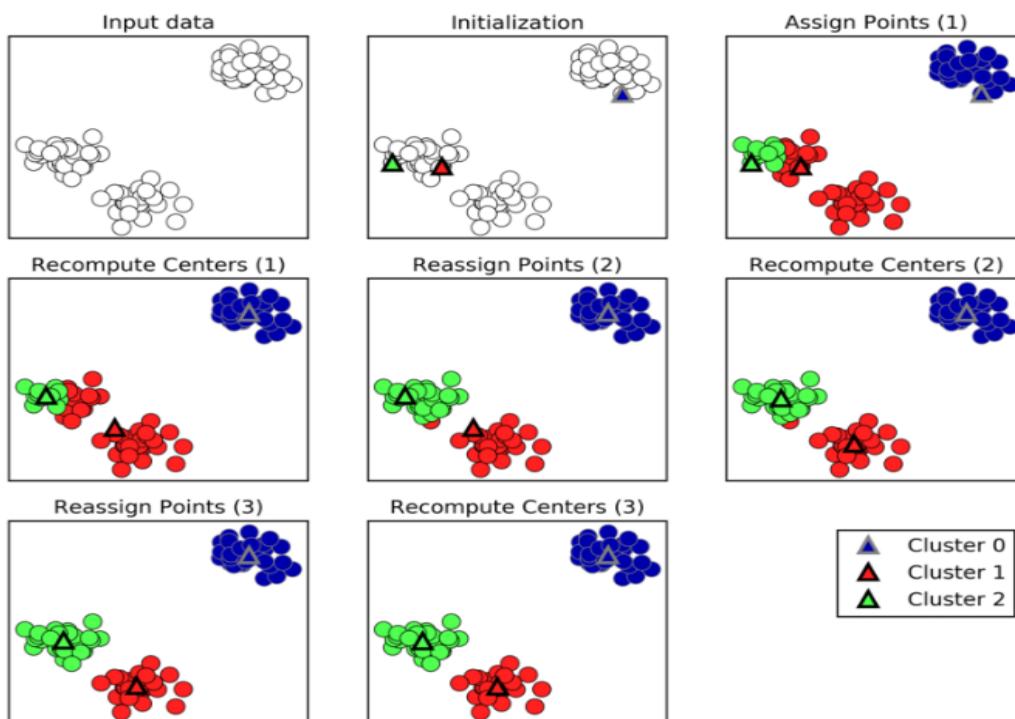


Fig : k-means clustering steps

Reference - Andreas C. Müller and Sarah Guido , Introduction to Machine learning with Python , O'reilly , October 2016.
Please refer to the link below for a nice animated view of k-means clustering.

https://commons.wikimedia.org/wiki/File:K-means_convergence.gif

3.20 Silhouette Score

The Silhouette Score and Silhouette Plot are used to measure the separation distance between clusters. It displays a measure of how close each point in a cluster is to points in the neighbouring clusters. This measure has a range of [-1, 1] and is a great tool to visually inspect the similarities within clusters and differences across clusters.

The Silhouette Score is calculated using the mean intra-cluster distance (i) and the mean nearest-cluster distance (n) for each sample. The Silhouette Coefficient for a sample is

$$SI = (n - i) / \max(i, n).$$

n is the distance between each sample and the nearest cluster that the sample is not a part of while i is the mean distance within each cluster.

Rand Index

Another commonly used metric is the Rand Index. It computes a similarity measure between two clusters by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clusterings.

The formula of the Rand Index is:

$$RI = \frac{\text{Number of Agreeing Pairs}}{\text{Number of Pairs}}$$

The RI can range from zero to 1, a perfect match. The only drawback of Rand Index is that it assumes that we can find the ground-truth clusters labels and use them to compare the performance of our model, so it is much less useful than the Silhouette Score for pure Unsupervised Learning tasks.

PRACTICAL : K-Means Clustering

Using the Iris Dataset, we will work on building an unsupervised learning based on 2 features of the dataset.

Link for the project code:

[https://github.com/Edunet-Foundation/Tech-Saksham/tree/main/Tech%20Saksham%20AI%20Practical%20-%20Notebooks/Chapter%205/1.k means clustering.ipynb](https://github.com/Edunet-Foundation/Tech-Saksham/tree/main/Tech%20Saksham%20AI%20Practical%20-%20Notebooks/Chapter%205/1.k%20means%20clustering.ipynb)

We will also work on another dataset for Customer Segmentation. This will help Managers of Mall to market the things effectively.

Customer segmentation is the practice of dividing a company's customers into groups that reflect similarity among customers in each group. The goal of segmenting customers is to decide how to relate to customers in each segment in order to maximize the value of each customer to the business.

Link for the project

https://github.com/Edunet-Foundation/Tech-Saksham/blob/main/Tech%20Saksham%20AI%20Practical%20-%20Notebooks/Chapter%205/2.Customer%20Segmentation/Pract_Customer_Segmentation.ipynb

So, in this chapter we explored the various algorithms which falls under the category of supervised and unsupervised machine learning. But there are some drawbacks or limitations on these algorithms. So, in the next chapter we are going to discuss the subset of machine learning named deep learning and how it helps to overcome the limitations of machine learning. So let us get started with deep learning.

3.21 Machine learning Application using Python GUI

1.25.1 ML realization using GUI Approach

[Link for the project](#)

Steps of Hands-on

1. Run the aforementioned jupyter notebook. It will open a GUI window

Machine Learning

Machine Learning

File Selection

File Name :

Train Test Split

X y Test Size: 0.25 Random State: None

Linear Regression

Logistic Regression

Decision Tree

Random Forest

2. Mention the name of file in file name and click on Select followed by Show button to view the dataframe

Machine Learning

Machine Learning

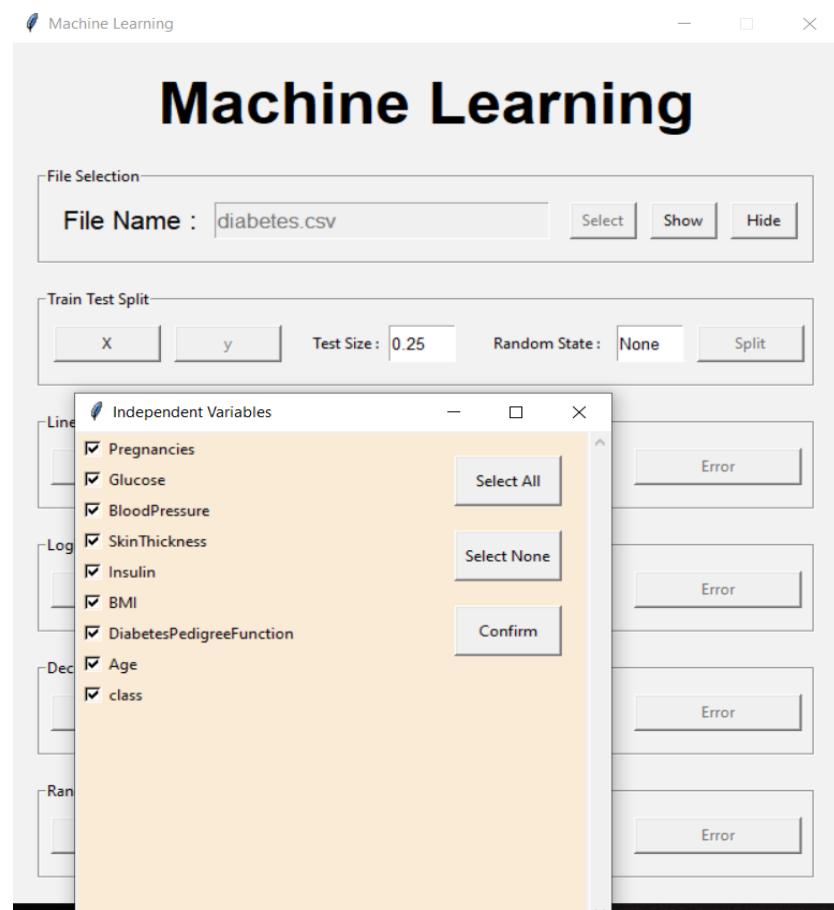
File Selection

File Name : diabetes.csv

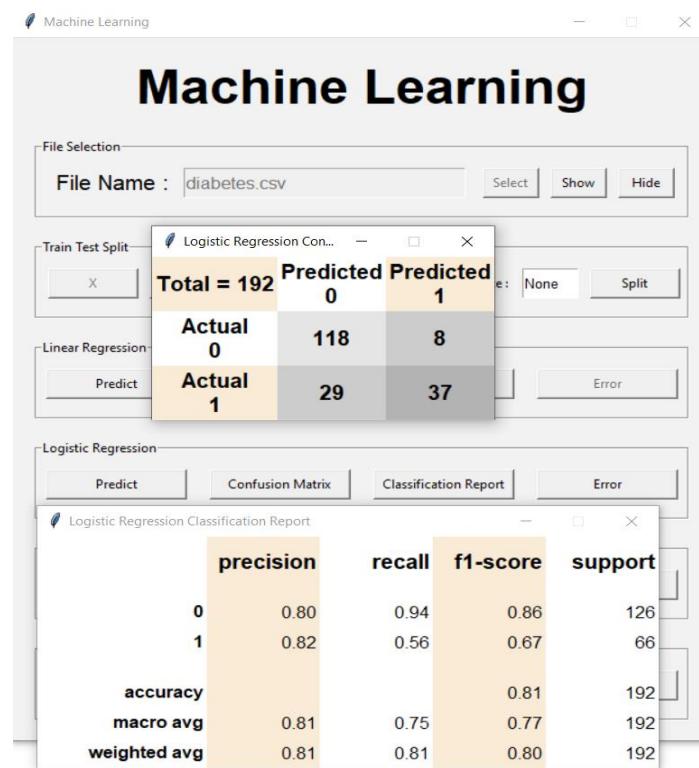
diabetes.csv

#	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabetes
1	6.0	148.0	72.0	35.0	0.0	33.6	
2	1.0	85.0	66.0	29.0	0.0	26.6	
3	8.0	183.0	64.0	0.0	0.0	23.3	
4	1.0	89.0	66.0	23.0	94.0	28.1	
5	0.0	137.0	40.0	35.0	168.0	43.1	
6	5.0	116.0	74.0	0.0	0.0	25.6	
7	3.0	78.0	50.0	32.0	88.0	31.0	
8	10.0	115.0	0.0	0.0	0.0	35.3	
9	2.0	197.0	70.0	45.0	543.0	30.5	
0	8.0	125.0	96.0	0.0	0.0	0.0	
1	4.0	110.0	92.0	0.0	0.0	37.6	
2	10.0	168.0	74.0	0.0	0.0	38.0	
3	10.0	139.0	80.0	0.0	0.0	27.1	
4	1.0	189.0	60.0	23.0	846.0	30.1	
5	5.0	166.0	72.0	19.0	175.0	25.8	
6	7.0	100.0	0.0	0.0	0.0	30.0	
7	0.0	118.0	84.0	47.0	230.0	45.8	
8	7.0	107.0	74.0	0.0	0.0	29.6	
9	1.0	103.0	30.0	38.0	83.0	43.3	
0	1.0	115.0	70.0	30.0	96.0	34.6	
1	3.0	126.0	88.0	41.0	235.0	39.3	
2	8.0	99.0	84.0	0.0	0.0	35.4	
3	7.0	196.0	90.0	0.0	0.0	39.8	
4	9.0	119.0	80.0	35.0	0.0	29.0	
5	11.0	143.0	94.0	33.0	146.0	36.6	

3. Select Train and Test variable from dataset Train_test split X & Y button



4. Select and click on Predict button of suitable Machine Learning Algorithm. Followed by it, make a click on Confusion Matrix and Classification report to view the machine learning outcome



5. Take any other dataset of interest and proceed accordingly to check performance of various ML algorithms on it.

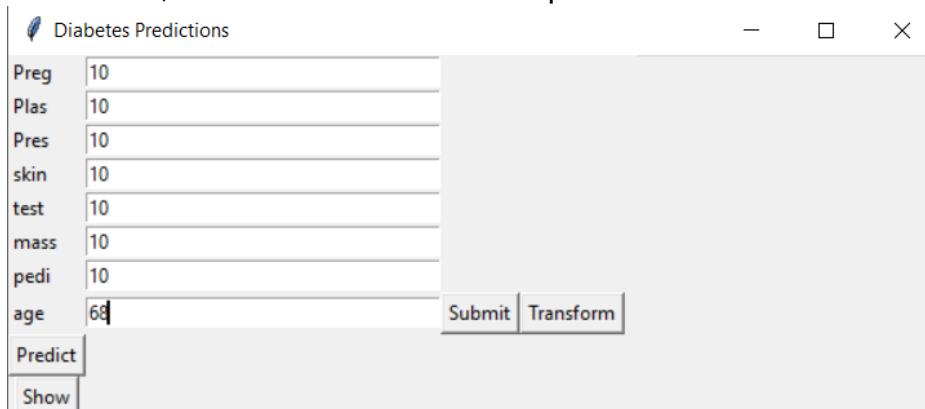
Deploy a ML test window using Tkinter

[Link of Project](#)

This project first model a Diabetes diagnosis classification model using Support Vector Machine. Resultant model is saved as .pkl. As an end user you need to reload saved model and need to create a test window interface to Diagnose diabetes in future.

Steps of Hands-on

1. Run the aforesaid jupyter notebook file to save the model as .pkl file.
2. At test end, need to feed the diabetes parameters to GUI.



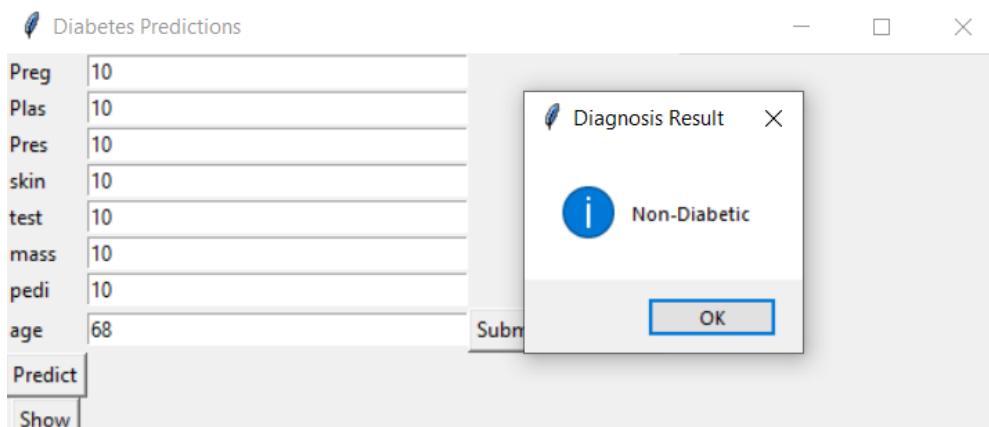
Preg	10
Plas	10
Pres	10
skin	10
test	10
mass	10
pedi	10
age	68

Submit Transform

Predict

Show

3. Click on submit button to save the user input to pandas DataFrame
4. As default data is read as text. It needs to apply text to numeric transformation to convert it to numeric. For that, you need to click on Transform.
5. Now the .pkl file will be loaded to test model performance on user data. For that, you need to click on Predict button.
6. Next, you need to click on Show to open a message window depicting the diagnosis result



7. You can change the dataset, redefine variable heads, ML algorithm and can adopt given template to come up with another similar interesting projects.

Module – IV

Computer Vision &

Edge Computing

with OpenVINO

toolkit

Unit 1: Deep Learning

Learning Outcomes:

- Understand the concept and features of Deep Learning
- Able solve problems using deep learning and neural network
- Understand the concept of forward/backward Propagation

1.1 What is Deep Learning?

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called **artificial neural networks**.

Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behavior of the human brain—albeit far from matching its ability—allowing it to “learn” from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy.

Deep learning drives many artificial intelligence (AI) applications and services that improve automation, performing analytical and physical tasks without human intervention. Deep learning technology lies behind everyday products and services (such as digital assistants, voice-enabled TV remotes, and credit card fraud detection) as well as emerging technologies (such as self-driving cars).

Deep learning neural networks, or artificial neural networks, attempts to mimic the human brain through a combination of data inputs, weights, and bias. These elements work together to accurately recognize, classify, and describe objects within the data.

1.2 Architecture

Deep Neural Networks

It is a neural network that incorporates the complexity of a certain level, which means several numbers of hidden layers are encompassed in between the input and output layers. They are highly proficient on model and process non-linear associations.

Deep Belief Networks

A deep belief network is a class of Deep Neural Network that comprises of multi-

layer belief networks.

Steps to perform DBN:

1. With the help of the Contrastive Divergence algorithm, a layer of features is learned from perceptible units.
2. Next, the formerly trained features are treated as visible units, which perform learning of features.
3. Lastly, when the learning of the final hidden layer is accomplished, then the whole DBN is trained.

Recurrent Neural Networks

It permits parallel as well as sequential computation, and it is exactly similar to that of the human brain (large feedback network of connected neurons). Since they are capable enough to reminisce all of the imperative things related to the input they have received, so they are more precise.

1.3 Deep learning vs. machine learning

Parameter	Machine Learning	Deep Learning
Data Dependency	Although machine learning depends on the huge amount of data, it can work with a smaller amount of data.	Deep Learning algorithms highly depend on a large amount of data, so we need to feed a large amount of data for good performance.
Execution time	Machine learning algorithm takes less time to train the model than deep learning, but it takes a long-time duration to test the model.	Deep Learning takes a long execution time to train the model, but less time to test the model.
Hardware Dependencies	Since machine learning models do not need much amount of data, so they can work on low-end machines.	The deep learning model needs a huge amount of data to work efficiently, so they need GPU's and hence the high-end machine.

Feature Engineering	Machine learning models need a step of feature extraction by the expert, and then it proceeds further.	Deep learning is the enhanced version of machine learning, so it does not need to develop the feature extractor for each problem; instead, it tries to learn high-level features from the data on its own.
Problem-solving approach	To solve a given problem, the traditional ML model breaks the problem in sub-parts, and after solving each part, produces the final result.	The problem-solving approach of a deep learning model is different from the traditional ML model, as it takes input for a given problem, and produce the end result. Hence it follows the end-to-end approach.
Interpretation of result	The interpretation of the result for a given problem is easy. As when we work with machine learning, we can interpret the result easily, it means why this result occur, what was the process.	The interpretation of the result for a given problem is very difficult. As when we work with the deep learning model, we may get a better result for a given problem than the machine learning model, but we cannot find why this particular outcome occurred, and the reasoning.
Type of data	Machine learning models mostly require data in a structured form.	Deep Learning models can work with structured and unstructured data both as they rely on the layers of the Artificial neural network.
Suitable for	Machine learning models are suitable for solving simple or bit-complex problems.	Deep learning models are suitable for solving complex problems.

1.4 Concept of Neural Networks

Artificial neural networks (ANNs), usually simply called neural networks (NNs), are computing systems vaguely inspired by the biological neural networks that constitute animal brains.

An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron that receives a signal then processes it and can signal neurons connected to it. The "signal" at a connection is a real number, and the output of each neuron is computed by some non-linear function of the sum of its inputs. The connections are called edges. Neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Neurons may have a threshold such that a signal is sent only if the aggregate signal crosses that threshold. Typically, neurons are aggregated into layers. Different layers may perform different transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the layers multiple times.

Neural networks learn (or are trained) by processing examples, each of which contains a known "input" and "result," forming probability-weighted associations between the two, which are stored within the data structure of the net itself. The training of a neural network from a given example is usually conducted by determining the difference between the processed output of the network (often a prediction) and a target output. This is the error. The network then adjusts its weighted associations according to a learning rule and using the error value. Successive adjustments will cause the neural network to produce output which is increasingly similar to the target output. After a sufficient number of these adjustments the training can be terminated based upon certain criteria. This is known as supervised learning.

Such systems "learn" to perform tasks by considering examples, generally without being programmed with task-specific rules. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as "cat" or "no cat" and using the results to identify cats in other images. They do this without any prior knowledge of cats, for example,

that the cats have fur, tails, whiskers, and cat-like faces. Instead, they automatically generate identifying characteristics from the examples that they process.

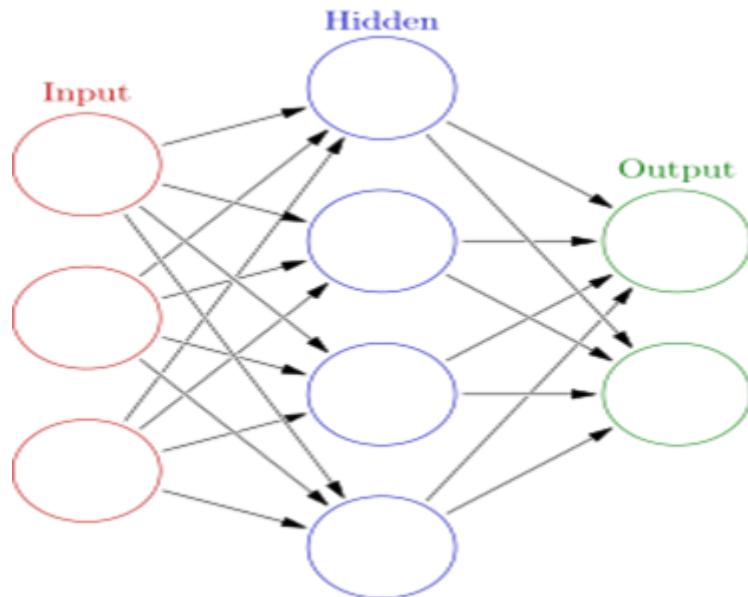


Fig: Neural Network

Reference - Glosser.ca, CC BY-SA 3.0 <<https://creativecommons.org/licenses/by-sa/3.0/>>, via Wikimedia Commons

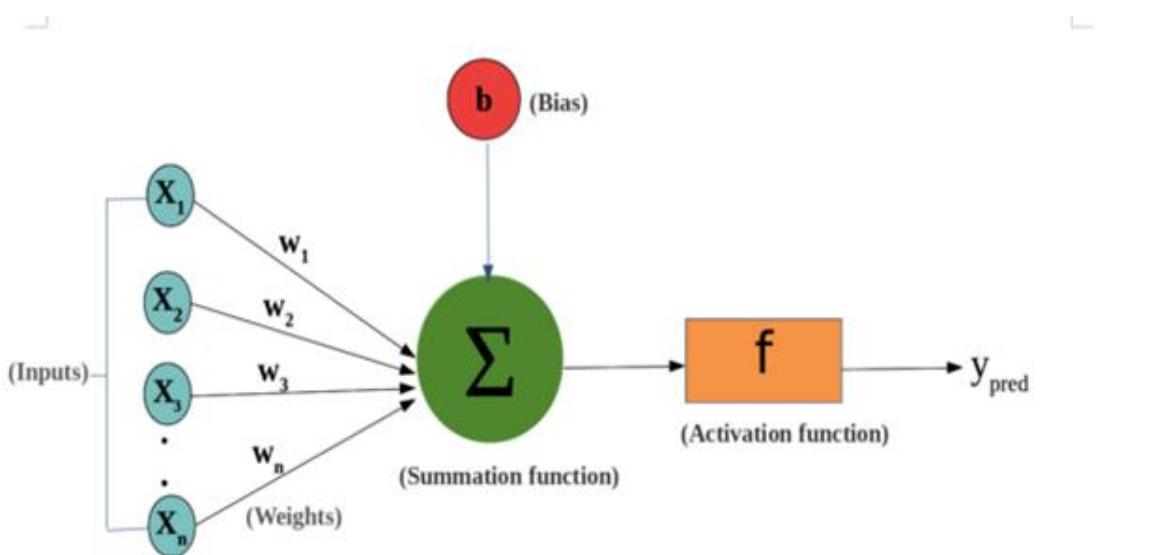


Fig: Basic building blocks of Neural Networks

Reference - <https://towardsdatascience.com/whats-the-role-of-weights-and-bias-in-a-neural-network-4cf7e9888a0f>

Let us understand the core part of artificial neural networks which is Neuron.

Neurons

ANNs are composed of artificial neurons which are conceptually derived from biological neurons. Each artificial neuron has inputs and produces a single output which can be sent to multiple other neurons. The inputs can be the feature values of a sample of external data, such as images or documents, or they can be the outputs of other neurons. The outputs of the final output neurons of the neural net accomplish the task, such as recognizing an object in an image.

To find the output of the neuron, first we take the weighted sum of all the inputs, weighted by the weights of the connections from the inputs to the neuron. We add a bias term to this sum. This weighted sum is sometimes called the activation. This weighted sum is then passed through a (usually nonlinear) activation function to produce the output. The initial inputs are external data, such as images and documents. The ultimate outputs accomplish the task, such as recognizing an object in an image.

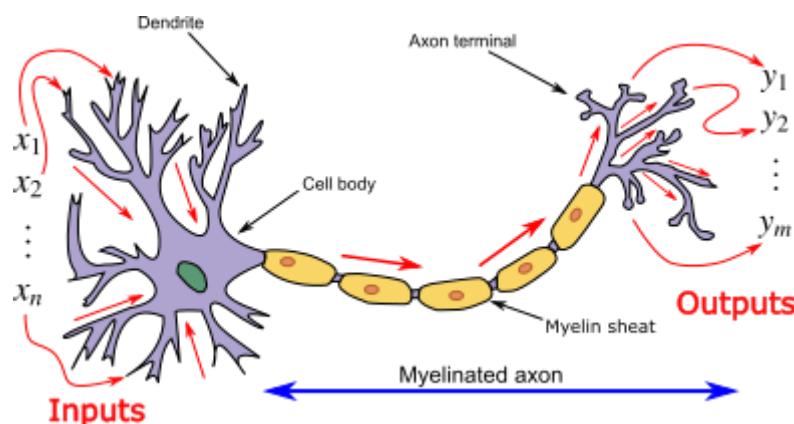


Fig: Neuron

Reference - Egm4313.s12 at English Wikipedia, CC BY-SA 3.0 <<https://creativecommons.org/licenses/by-sa/3.0/>>, via Wikimedia Commons

The activation function of a node defines the output of that node given an input or set of inputs.

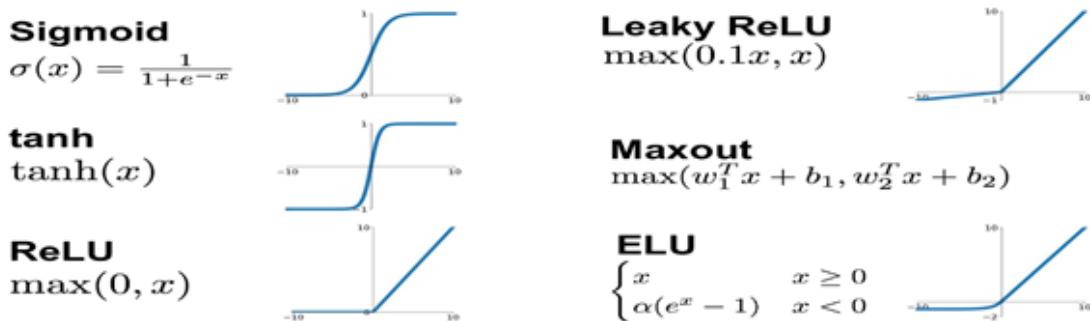


Fig: Different Activation functions

Reference - <https://hsf-training.github.io/hsf-training-ml-webpage/03-nn/index.html>

Weights

Weights are the real values that are associated with each feature which tells the importance of that feature in predicting the final value.

What do the weights in a Neuron convey to us?

- Importance of the feature

Weights associated with each feature, convey the importance of that feature in predicting the output value. Features with weights that are close to zero said to have lesser importance in the prediction process compared to the features with weights having a larger value.

- Tells the relationship between a particular feature in the dataset and the target value.

Bias

Bias is used for shifting the activation function towards left or right, it can be referred to as a y-intercept in the line equation.

Forward Propagation

Forward propagation is how neural networks make predictions. Input data is “forward propagated” through the network layer by layer to the final layer which outputs a prediction.

Forward propagation (or *forward pass*) refers to the calculation and storage of intermediate variables (including outputs) for a neural network in order from the input

layer to the output layer. We now work step-by-step through the mechanics of a neural network with one hidden layer. This may seem tedious but in the eternal words of funk virtuoso James Brown, you must “pay the cost to be the boss”.

For the sake of simplicity, let us assume that the input example is $x \in \mathbb{R}^d$ and that our hidden layer does not include a bias term. Here the intermediate variable is:

$$z = W^{(1)} x,$$

where $W^{(1)} \in \mathbb{R}^{q \times h}$ is the weight parameter of the hidden layer. After running the intermediate variable $z \in \mathbb{R}^h$ through the activation function ϕ we obtain our hidden activation vector of length h ,

$$h = \phi(z)$$

The hidden variable h is also an intermediate variable. Assuming that the parameters of the output layer only possess a weight of $W^{(2)} \in \mathbb{R}^{q \times h}$, we can obtain an output layer variable with a vector of length q :

$$o = W^{(2)} h.$$

Assuming that the loss function is l and the example label is y , we can then calculate the loss term for a single data example,

$$L = l(o, y)$$

According to the definition of L2 regularization, given the hyperparameter λ , the regularization term is

$$s = \frac{\lambda}{2} (\| W^{(1)} \|_F + \| W^{(2)} \|_F)$$

where the Frobenius norm of the matrix is simply the L2 norm applied after flattening the matrix into a vector. Finally, the model’s regularized loss on a given data example is:

$$J = L + s.$$

We refer to J as the objective function in the following discussion.

Backpropagation

In machine learning, backpropagation is a widely used algorithm for training feedforward neural networks. Generalizations of backpropagation exist for other artificial neural networks (ANNs), and for functions generally. These classes of algorithms are all referred to generically as "backpropagation". In fitting a neural network, backpropagation computes the gradient of the loss function with respect to the weights of the network for a single input–output example, and does so efficiently, unlike a naive direct computation of the gradient with respect to each weight individually.

This efficiency makes it feasible to use gradient methods for training multilayer networks, updating weights to minimize loss; gradient descent, or variants such as stochastic gradient descent, are commonly used. The backpropagation algorithm works by computing the gradient of the loss function with respect to each weight by the chain rule, computing the gradient one layer at a time, iterating backward from the last layer to avoid redundant calculations of intermediate terms in the chain rule; this is an example of dynamic programming.

Backpropagation refers to the method of calculating the gradient of neural network parameters. In short, the method traverses the network in reverse order, from the output to the input layer, according to the *chain rule* from calculus.

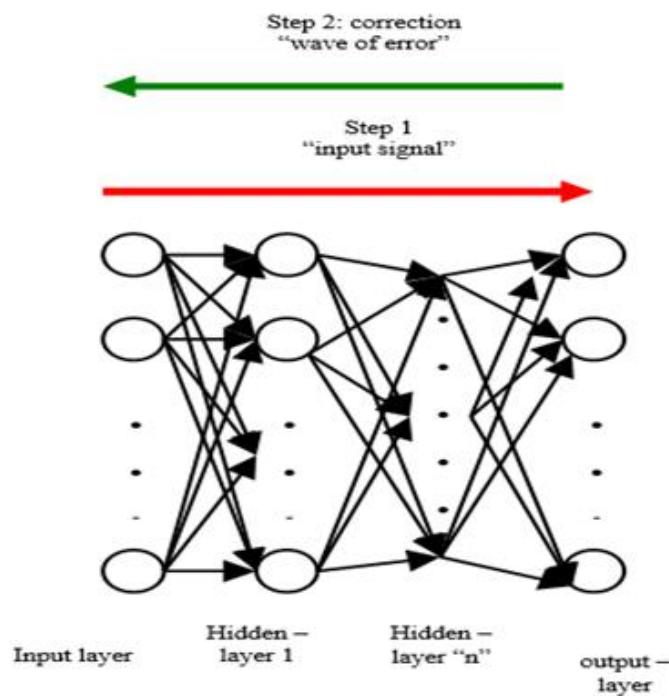
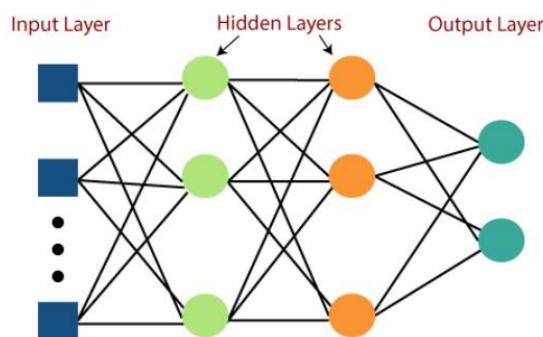


Fig: Forward and backward propagation in Neural Networks
Reference - Jorge Guerra Pires, CC BY-SA 3.0 <<https://creativecommons.org/licenses/by-sa/3.0>>, via Wikimedia Commons

1.5 Multilayer Perceptron

Multilayer Perceptron is commonly used in simple regression problems. However, MLPs are not ideal for processing patterns with sequential and multidimensional data.

Multi-Layer perceptron defines the most complex architecture of artificial neural networks. It is substantially formed from multiple layers of the perceptron. TensorFlow is a very popular deep learning framework released by, and this notebook will guide to build a neural network with



Reference -<https://www.javatpoint.com/multi-layer-perceptron-in-tensorflow>

MLP networks are used for supervised learning format. A typical learning algorithm for MLP networks is also called back propagation's algorithm.

A multilayer perceptron (MLP) is a feed forward artificial neural network that generates a set of outputs from a set of inputs. An MLP is characterized by several layers of input nodes connected as a directed graph between the input nodes connected as a directed graph between the input and output layers. MLP uses backpropagation for training the network. MLP is a deep learning method.

Unit 2: Operational Deep Learning

Learning Outcomes:

- Understanding concepts of Deep Learning like Gradient Descent, Cross Entropy
- Understanding Overfitting and Regularization
- Understanding TensorFlow, Keras API and different layers

2.1 Gradient Descent

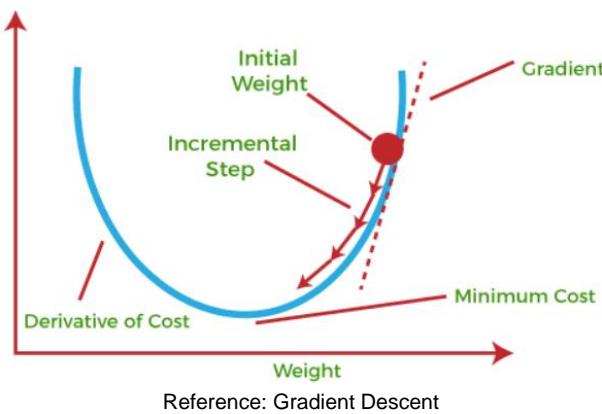
Introduction

Gradient Descent is known as one of the most commonly used optimization algorithms to train machine learning models by means of minimizing errors between actual and expected results. Further, gradient descent is also used to train Neural Networks. In mathematical terminology, Optimization algorithm refers to the task of minimizing/maximizing an objective function $f(x)$ parameterized by x .

Gradient descent was initially discovered by "**Augustin-Louis Cauchy**" in the mid-18th century. **Gradient Descent is defined as one of the most commonly used iterative optimization algorithms of machine learning to train the machine learning and deep learning models. It helps in finding the local minimum of a function.**

The best way to define the local minimum or local maximum of a function using gradient descent is as follows:

- If we move towards a negative gradient or away from the gradient of the function at the current point, it will give the local minimum of that function.
- Whenever we move towards a positive gradient or towards the gradient of the function at the current point, we will get the local maximum of that function.



The main objective of using a gradient descent algorithm is to minimize the cost function using iteration. To achieve this goal, it performs two steps iteratively:

- Calculates the first-order derivative of the function to compute the gradient or slope of that function.
- Move away from the direction of the gradient, which means slope increased from the current point by alpha times, where Alpha is defined as Learning Rate. It is a tuning parameter in the optimization process which helps to decide the length of the steps.

Cost Function

The cost function is defined as the measurement of difference or error between actual values and expected values at the current position and present in the form of a single real number. It helps to increase and improve machine learning efficiency by providing feedback to this model so that it can minimize error and find the local or global minimum.

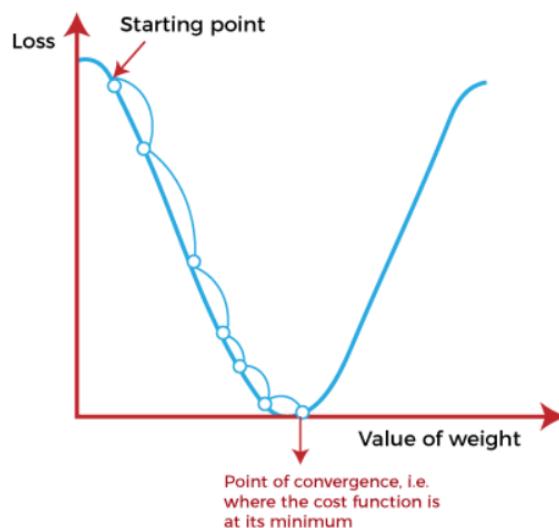
The cost function is calculated after making a hypothesis with initial parameters and modifying these parameters using gradient descent algorithms over known data to reduce the cost function.

How does Gradient Descent work?

Before starting the working principle of gradient descent, we should know some basic concepts to find out the slope of a line from linear regression. The equation for simple linear regression is given as:

$$y = mx + c$$

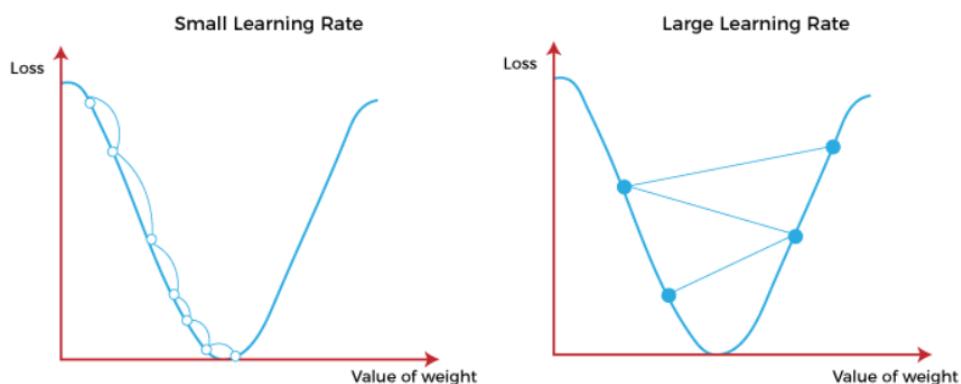
where 'm' represents the slope of the line, and 'c' represents the intercepts on the y-axis.



The starting point (shown in above fig.) is used to evaluate the performance as it is considered just as an arbitrary point. At this starting point, we will derive the first derivative or slope and then use a tangent line to calculate the steepness of this slope. Further, this slope will inform the updates to the parameters (weights and bias).

Learning Rate

It is defined as the step size taken to reach the minimum or lowest point. This is typically a small value that is evaluated and updated based on the behaviour of the cost function. If the learning rate is high, it results in larger steps but also leads to risks of overshooting the minimum. At the same time, a low learning rate shows the small step sizes, which compromises overall efficiency but gives the advantage of more precision.



Types of Gradient Descent

1. **Batch Gradient Descent:** Batch gradient descent (BGD) is used to find the error for each point in the training set and update the model after evaluating all training

examples. This procedure is known as the training epoch. In simple words, it is a greedy approach where we have to sum over all examples for each update.

Advantages of Batch Gradient Descent:

1. It produces less noise in comparison to other gradient descent.
2. It produces stable gradient descent convergence.

2. Stochastic Gradient Descent: Stochastic gradient descent (SGD) is a type of gradient descent that runs one training example per iteration. Or in other words, it processes a training epoch for each example within a dataset and updates each training example's parameters one at a time. As it requires only one training example at a time, hence it is easier to store in allocated memory.

Advantages of Stochastic Gradient Descent

1. It is easier to allocate in desired memory.
 2. It is more efficient for large datasets
- 3. Minibatch Gradient Descent:** Mini Batch gradient descent is the combination of both batch gradient descent and stochastic gradient descent. It divides the training datasets into small batch sizes then performs the updates on those batches separately. Splitting training datasets into smaller batches make a balance to maintain the computational efficiency of batch gradient descent and speed of stochastic gradient descent.

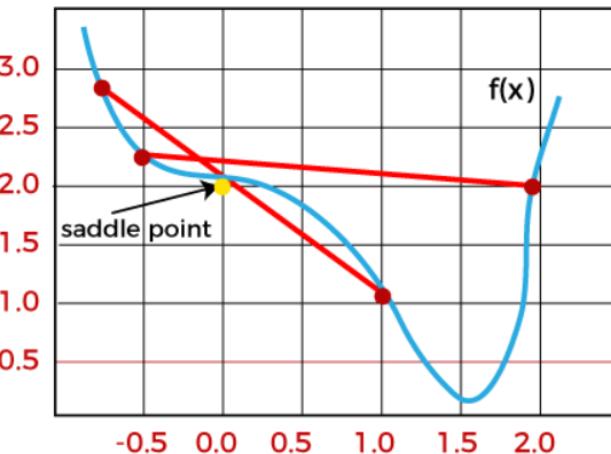
Advantages of MiniBatch Gradient Descent

1. It is computationally efficient.
2. It produces stable gradient descent convergence.

Challenges with Gradient Descent

1. Local Minima & Saddle Point

For convex problems, gradient descent can find the global minimum easily, while for non-convex problems, it is sometimes difficult to find the global minimum, where the machine learning models achieve the best results.



In contrast, with saddle points, the negative gradient only occurs on one side of the point, which reaches a local maximum on one side and a local minimum on the other side. The name of a saddle point is taken by that of a horse's saddle.

The name of local minima is because the value of the loss function is minimum at that point in a local region. In contrast, the name of the global minima is given so because the value of the loss function is minimum there, globally across the entire domain the loss function.

2. Vanishing and Exploding Gradient

Vanishing Gradients

Vanishing Gradient occurs when the gradient is smaller than expected. During backpropagation, this gradient becomes smaller than causing the decrease in the learning rate of earlier layers than the later layer of the network. Once this happens, the weight parameters update until they become insignificant.

Exploding Gradient

Exploding gradient is just opposite to the vanishing gradient as it occurs when the Gradient is too large and creates a stable model. Further, in this scenario, model weight increases, and they will be represented as NaN. This problem can be solved using the dimensionality reduction technique, which helps to minimize complexity within the model.

2.2 Loss Function & Accuracy Metrics

Loss functions play an important role in any statistical model - they define an objective which the performance of the model is evaluated against and the parameters learned by the model are determined by minimizing a chosen loss function. Loss functions

define what a good prediction is and isn't. In short, choosing the right loss function dictates how well your estimator will be.

Regression Metrics

Mean Squared Error: Mean squared error is perhaps the most popular metric used for regression problems. It essentially finds the average of the squared difference between the target value and the value predicted by the regression model.

$$MSE = \frac{1}{N} \sum_{j=1}^N (y_j - \check{y}_j)^2$$

where:

- y_j : ground-truth value
- \check{y}_j : predicted value from the regression model
- N: number of datums

Mean Absolute Error: Mean Absolute Error is the average of the difference between the ground truth and the predicted values. Mathematically, its represented as

$$MAE = \frac{1}{N} \sum_{j=1}^N |y_j - \check{y}_j|$$

where:

- y_j : ground-truth value
- \check{y}_j : predicted value from the regression model
- N: number of datums

Root Mean Squared Error: Root Mean Squared Error corresponds to the square root of the average of the squared difference between the target value and the value predicted by the regression model. Basically, $\sqrt{\text{MSE}}$. Mathematically it can be represented as:

$$RMSE = \sqrt{\frac{1}{N} \sum_{j=1}^N (y_j - \check{y}_j)^2}$$

Few key points related to RMSE:

- It retains the differentiable property of MSE.
- It handles the penalization of smaller errors done by MSE by square rooting it.

R² Coefficient of Determination: R² Coefficient of determination actually works as a post metric, meaning it's a metric that's calculated using other metrics.

The point of even calculating this coefficient is to answer the question "How much (what %) of the total variation in Y(target) is explained by the variation in X (regression line)"

$$\text{coeff}(R^2) = 1 - \frac{SE(\text{line})}{SE(\bar{Y})}$$

This coefficient can be implemented simply using NumPy arrays in Python.

Adjusted R²: The Vanilla R² method suffers from some demons, like misleading the researcher into believing that the model is improving when the score is increasing but in reality, the learning is not happening. This can happen when a model overfits the data, in that case the variance explained will be 100% but the learning hasn't happened. To rectify this, R² is adjusted with the number of independent variables.

$$R_a^2 = 1 - \left[\left(\frac{n-1}{n-k-1} \right) \cdot (1 - R^2) \right]$$

where:

- n = number of observations
- k = number of independent variables
- Ra² = adjusted R²

Classification Metrics

Classification problems are one of the world's most widely researched areas. Use cases are present in almost all production and industrial environments. Speech recognition, face recognition, text classification

So, in order to evaluate Classification models, we'll discuss these metrics in detail

Accuracy: Classification accuracy is perhaps the simplest metric to use and implement and is defined as the number of correct predictions divided by the total number of predictions, multiplied by 100. We can implement this by comparing ground truth and predicted values in a loop or simply utilize the scikit-learn module.

Start by just importing the accuracy_score function from the metrics class.

```
from sklearn.metrics import accuracy_score
```

Then, just by passing the ground truth and predicted values, you can determine the accuracy of your model:

```
print(f'Accuracy Score is {accuracy_score(y_test,y_hat)}')
```

Confusion Matrix: Confusion Matrix is a tabular visualization of the ground-truth labels versus model predictions. Each row of the confusion matrix represents the instances in a predicted class and each column represents the instances in an actual class. Confusion Matrix is not exactly a performance metric but sort of a basis on which other metrics evaluate the results.

		Predicted	
		Has Cancer	Doesn't Have Cancer
Ground Truth	Has Cancer	TP	FP
	Doesn't Have Cancer	FN	TN

- **True Positive (TP)** signifies how many positive class samples your model predicted correctly.
- **True Negative (TN)** signifies how many negative class samples your model predicted correctly.
- **False Positive (FP)** signifies how many negative class samples your model predicted incorrectly. This factor represents Type-I error in statistical nomenclature. This error positioning in the confusion matrix depends on the choice of the null hypothesis.
- **False Negative (FN)** signifies how many positive class samples your model predicted incorrectly. This factor represents Type-II error in statistical nomenclature. This error positioning in the confusion matrix also depends on the choice of the null hypothesis.

Precision: Precision is the ratio of true positives and total positives predicted.

$$P = \frac{TP}{TP+FP}$$

$$0 < P < 1$$

The precision metric focuses on Type-I errors (FP). A Type-I error occurs when we reject a true null Hypothesis(H^0).

Recall/Sensitivity/Hit-Rate: A Recall is essentially the ratio of true positives to all the positives in ground truth.

$$R = \frac{TP}{TP+FN}$$

$$0 < R < 1$$

The recall metric focuses on type-II errors (FN). A type-II error occurs when we accept a false null hypothesis(H^0).

F1-score: The F1-score metric uses a combination of precision and recall. In fact, the F1 score is the harmonic mean of the two. The formula of the two essentially is:

$$F_1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}}$$

Now, a high F1 score symbolizes a high precision as well as high recall. It presents a good balance between precision and recall and gives good results on imbalanced classification problems.

A low F1 score tells you (almost) nothing—it only tells you about performance at a threshold. Low recall means we didn't try to do well on very much of the entire test set. Low precision means that, among the cases we identified as positive cases, we didn't get many of them right.

2.3 Cross Entropy & MSE

Assume we have two distributions of data and need to be compared. Cross entropy employs the concept of entropy which we have seen above. Cross entropy is a measure of the entropy difference between two probability distributions. Assume the first probability distribution is denoted by A and the second probability distribution is denoted by B.

The average number of bits required to send a message from distribution A to distribution B is referred to as cross-entropy. Cross entropy is a concept used in machine learning when algorithms are created to predict from the model. The construction of the model is based on a comparison of actual and expected results.

Mathematically we can represent cross-entropy as below:

$$H(p, q) = - \sum_{x \in \mathcal{X}} p(x) \log q(x)$$

In the above equation, x is the total number of values and p(x) is the probability of distribution in the real world. In the projected distribution B, A is the probability distribution and q(x) are the probability of distribution.

Cross-Entropy as Loss Function

When optimizing classification models, cross-entropy is commonly employed as a loss function. The logistic regression technique and artificial neural network can be utilized for classification problems.

In classification, each case has a known class label with a probability of 1.0 while all other labels have a probability of 0.0. Here model calculates the likelihood that a given example belongs to each class label. The difference between two probability distributions can then be calculated using cross-entropy.

When we are dealing with Two Class probability, the probability is modelled as Bernoulli distribution for the positive class. This means that the mode explicitly predicts the probability for class 1, while the probability for class 0 is given as 1 – projected probability.

- Class 1 = 1 (originally predicted)
- Class 0 = 1 – originally predicted

Mean Squared Error

Mean squared error is perhaps the most popular metric used for regression problems. It essentially finds the average of the squared difference between the target value and the value predicted by the regression model.

$$MSE = \frac{1}{N} \sum_{j=1}^N (y_j - \hat{y}_j)^2$$

where:

- y_j : ground-truth value
- \hat{y}_j : predicted value from the regression model
- N: number of datums

2.4 Overfitting & Regularization

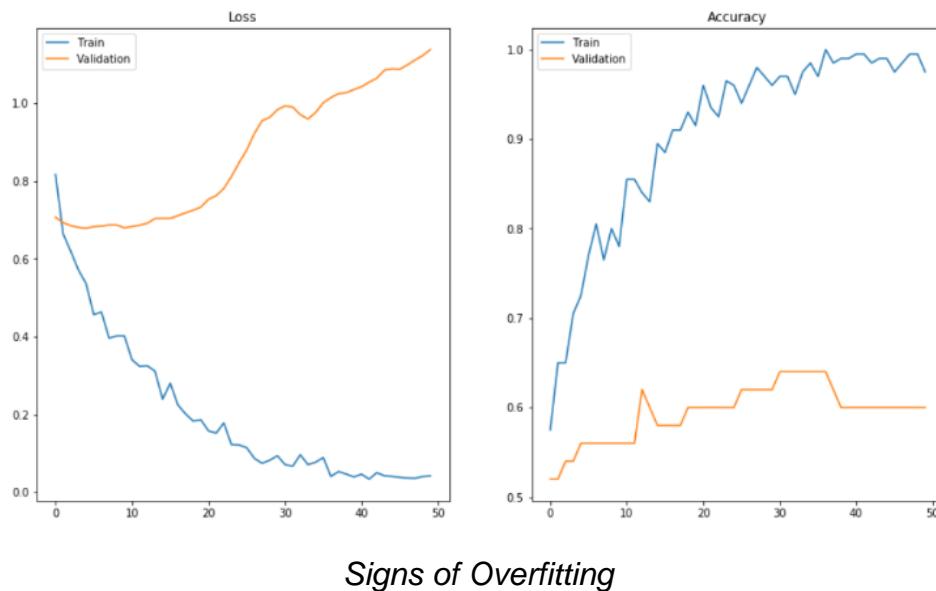
It is a common pitfall in deep learning algorithms in which a model tries to fit the training data entirely and ends up memorizing the data patterns and the noise and random fluctuations.

These models fail to generalize and perform well in the case of unseen data scenarios, defeating the model's purpose.

When can overfitting occurs?

The high variance of the model performance is an indicator of an overfitting problem.

The training time of the model or its architectural complexity may cause the model to overfit. If the model trains for too long on the training data or is too complex, it learns the noise or irrelevant information within the dataset.



Signs of Overfitting

Overfitting: Key Concepts

Bias - Bias measures the difference between the model's prediction and the target value. If the model is oversimplified, then the predicted value would be far from the ground truth resulting in more bias.

Variance - Variance is the measure of the inconsistency of different predictions over varied datasets. If the model's performance is tested on different datasets, the closer the prediction, the lesser the variance. Higher variance is an indication of overfitting in which the model loses the ability to generalize.

Bias-Variance Trade-off - A simple linear model is expected to have a high bias and low variance due to less complexity of the model and fewer trainable parameters. On the other hand, complex non-linear models tend to observe an opposite behaviour. In an ideal scenario, the model would have an optimal balance of bias and variance.

Model generalization - Model generalization means how well the model is trained to extract useful data patterns and classify unseen data samples.

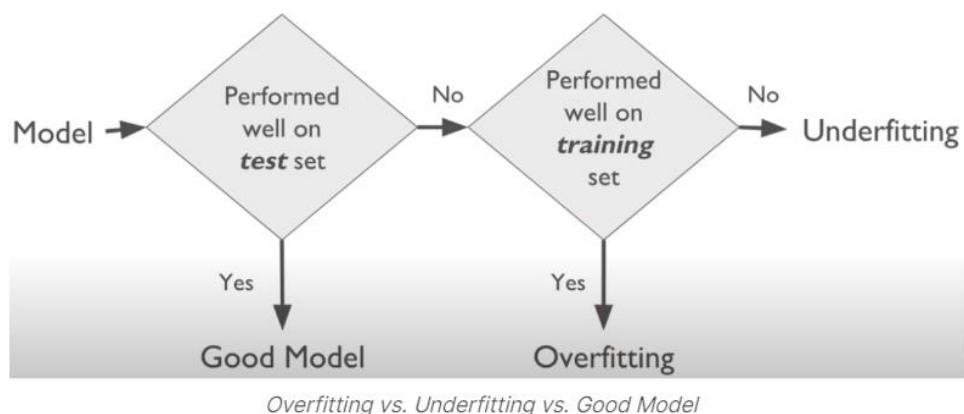
Feature selection - It involves selecting a subset of features from all the extracted features that contribute most towards the model performance. Including all the features unnecessarily increases the model complexity and redundant features can significantly increase the training time.

Overfitting vs. Underfitting

Underfitting occurs when we have a high bias in our data, i.e., we are oversimplifying the problem, and as a result, the model does not work correctly in the training data.

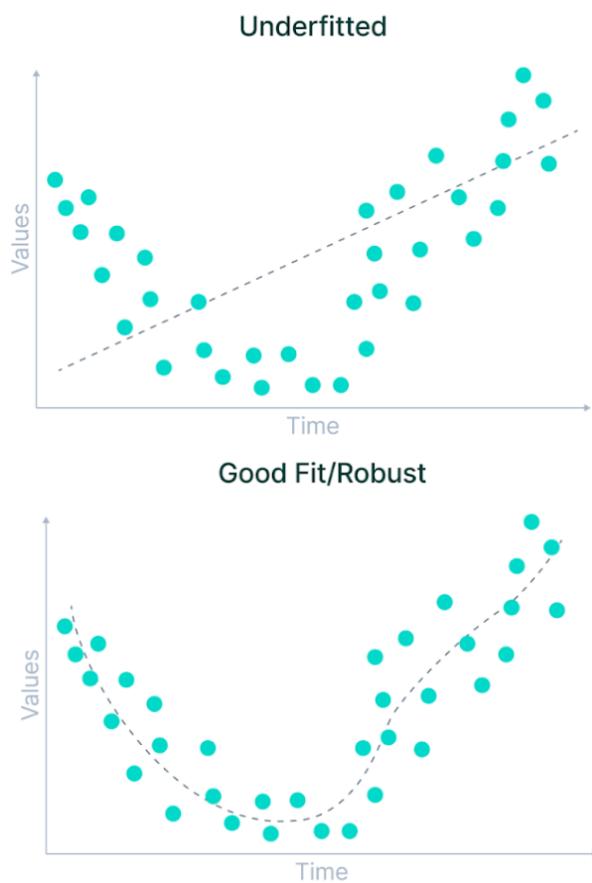
Overfitting occurs when the model has a high variance, i.e., the model performs well on the training data but does not perform accurately in the evaluation set. The model

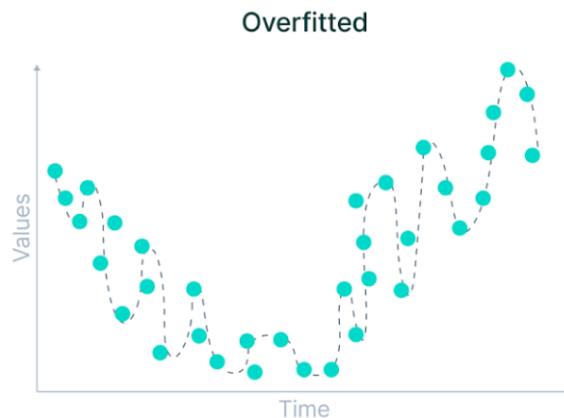
memorizes the data patterns in the training dataset but fails to generalize to unseen examples.



The goal is to find a good fit such that the model picks up the patterns from the training data and does not end up memorizing the finer details.

Visual comparison to get a better understanding of the differences.





Underfitted vs. Fit vs. Overfitted model

Techniques to avoid Overfitting

Train with more data

With the increase in the training data, the crucial features to be extracted become prominent. The model can recognize the relationship between the input attributes and the output variable. The only assumption in this method is that the data to be fed into the model should be clean; otherwise, it would worsen the problem of overfitting.

Data augmentation

An alternative method to training with more data is data augmentation, which is less expensive and safer than the previous method. Data augmentation makes a sample data look slightly different every time the model processes it.

Feature selection

Every model has several parameters or features depending upon the number of layers, number of neurons, etc. The model can detect many redundant features or features determinable from other features leading to unnecessary complexity. We very well know that the more complex the model, the higher the chances of the model to overfit.

Cross-validation

Cross-validation is a robust measure to prevent overfitting. The complete dataset is split into parts. In standard K-fold cross-validation, we need to partition the data into k folds. Then, we iteratively train the algorithm on k-1 folds while using the remaining holdout fold as the test set. This method allows us to tune the hyperparameters of the neural network or machine learning model and test it using completely unseen data.

Regularization

Regularization is one of the most important concepts of machine learning. It is a technique to prevent the model from overfitting by adding extra information to it.

Sometimes the machine learning model performs well with the training data but does not perform well with the test data. It means the model is not able to predict the output when deals with unseen data by introducing noise in the output, and hence the model is called overfitted. This problem can be deal with the help of a regularization technique.

It mainly regularizes or reduces the coefficient of features toward zero. In simple words, "*In regularization technique, we reduce the magnitude of the features by keeping the same number of features.*"

How does Regularization work?

Regularization works by adding a penalty or complexity term to the complex model. Let's consider the simple linear regression equation:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \dots + \beta_n x_n + b$$

In the above equation, Y represents the value to be predicted

X₁, X₂, ...X_n are the features for Y.

$\beta_0, \beta_1, \dots, \beta_n$ are the weights or magnitude attached to the features, respectively. Here represents the bias of the model, and b represents the intercept.

Linear regression models try to optimize the β_0 and b to minimize the cost function. The equation for the cost function for the linear model is given below:

$$\sum_{i=1}^M (y_i - y'_i)^2 = \sum_{i=1}^M (y_i - \sum_{j=0}^n \beta_j * X_{ij})^2$$

Now, we will add a loss function and optimize parameter to make the model that can predict the accurate value of Y. The loss function for the linear regression is called as **RSS or Residual sum of squares**.

Techniques of Regularization

There are mainly two types of regularization techniques, which are given below:

1. Ridge Regression
2. Lasso Regression

Ridge Regression

Ridge regression is one of the types of linear regression in which a small amount of bias is introduced so that we can get better long-term predictions.

Ridge regression is a regularization technique, which is used to reduce the complexity of the model. It is also called as L2 regularization.

In this technique, the cost function is altered by adding the penalty term to it. The amount of bias added to the model is called Ridge Regression penalty. We can

calculate it by multiplying with the lambda to the squared weight of each individual feature.

The equation for the cost function in ridge regression will be:

$$\sum_{i=1}^M (y_i - \hat{y}_i)^2 = \sum_{i=1}^M \left(y_i - \sum_{j=0}^n \beta_j * x_{ij} \right)^2 + \lambda \sum_{j=0}^n \beta_j^2$$

In the above equation, the penalty term regularizes the coefficients of the model, and hence ridge regression reduces the amplitudes of the coefficients that decreases the complexity of the model.

As we can see from the above equation, if the values of λ tend to zero, the equation becomes the cost function of the linear regression model. Hence, for the minimum value of λ , the model will resemble the linear regression model.

Lasso Regression

Lasso regression is another regularization technique to reduce the complexity of the model. It stands for Least Absolute and Selection Operator.

It is similar to the Ridge Regression except that the penalty term contains only the absolute weights instead of a square of weights.

Since it takes absolute values, hence, it can shrink the slope to 0, whereas Ridge Regression can only shrink it near to 0.

It is also called as L1 regularization. The equation for the cost function of Lasso regression will be:

$$\sum_{i=1}^M (y_i - \hat{y}_i)^2 = \sum_{i=1}^M \left(y_i - \sum_{j=0}^n \beta_j * x_{ij} \right)^2 + \lambda \sum_{j=0}^n |\beta_j|$$

Some of the features in this technique are completely neglected for model evaluation.

Hence, the Lasso regression can help us to reduce the overfitting in the model as well as the feature selection.

Key Difference between Ridge Regression and Lasso Regression

Ridge regression is mostly used to reduce the overfitting in the model, and it includes all the features present in the model. It reduces the complexity of the model by shrinking the coefficients.

Lasso regression helps to reduce the overfitting in the model as well as feature selection.

TensorFlow 2.0 and Keras API

TensorFlow is one of the most popular and widely used Deep Learning libraries in the companies these days. It helps you work with complex data and build neural network models to solve business problems.

TensorFlow 2.0

TensorFlow 2.0 is a library that provides a comprehensive ecosystem of tools for developers, researchers, and organizations who want to build scalable Machine Learning and Deep Learning applications.

TensorFlow is a popular open-source library released in 2015 by the Google Brain team for building machine learning and deep learning models. It is based on Python programming language and performs numerical computations using data flow graphs to build models.

Features of TensorFlow 2.0

- TensorFlow 2.0 supports easy model building with Keras and eager execution.
- It has robust model deployment in production on any platform.
- You can perform robust experimentation for research.
- TensorFlow 2.0 simplifies the API by cleaning up deprecated APIs and reducing duplication.
- TensorFlow 2.0 works efficiently with multi-dimensional arrays.
- TensorFlow 2.0 supports fast debugging and model building.

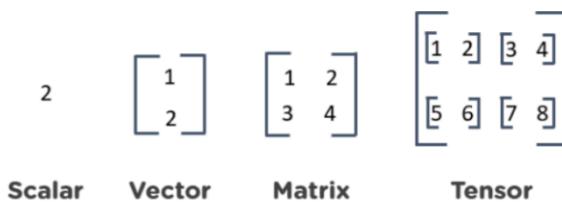
TensorFlow Applications

Below are a few examples where TensorFlow is being widely used:

- Face detection in electronic devices.
- Machine language translation through apps such as Google Translate.
- Fraud detection in the banking and financial sectors.
- Object detections on videos.

Tensor

TensorFlow is derived from its core component known as a tensor. A tensor is actually a vector or a matrix of n-dimensions that represents all types of data.



In TensorFlow, we define tensors by a unit of dimensionality known as a rank.

Example	Entity	Rank
s = 300	Scalar	0
v = [1, 2, 3]	Vector	1
m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]	Matrix	2
t = [[[2], [4], [6]], [[8], [9], [10]], [[11], [12], [13]]]	Tensor	3

TensorFlow performs computations with the help of dataflow graphs. It has nodes that represent the operations in your model.

Let's compute the function depicted below and see how TensorFlow works:

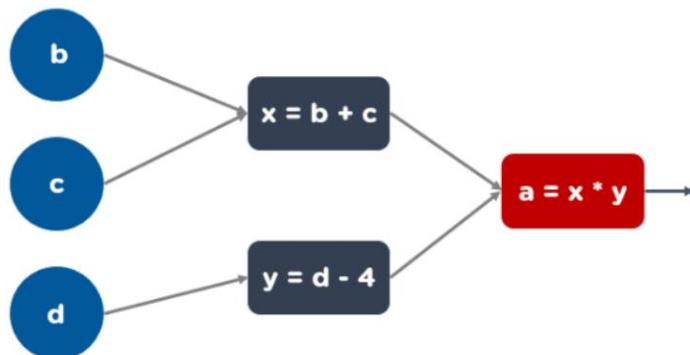
$$a(b, c, d) = (b + c) * (d - 4)$$

$$x = b + c$$

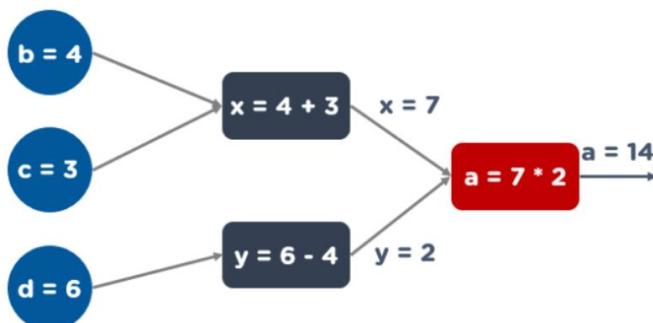
$$y = d - 4$$

$$a = x * y$$

The graph nodes are the inputs and perform mathematical computations, while the connections carry the weights. In this case, it's the result of an expression.



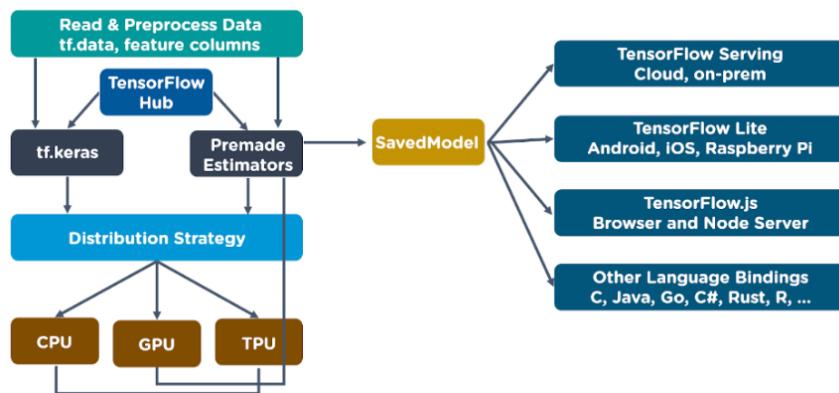
Displayed below is the output:



TensorFlow 2.0 Architecture

Over the last few years, the developer community has added many components to TensorFlow. These components will be packaged together into a comprehensive platform that supports machine learning workflows from training through deployment.

Shown below is the new architecture of TensorFlow 2.0:



Keras API

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research.

KERAS IS Simple -- but not simplistic. Keras reduces developer cognitive load to free you to focus on the parts of the problem that really matter.

Flexible -- Keras adopts the principle of progressive disclosure of complexity: simple workflows should be quick and easy, while arbitrarily advanced workflows should be possible via a clear path that builds upon what you've already learned.

Powerful -- Keras provides industry-strength performance and scalability: it is used by organizations and companies including NASA, YouTube, or Waymo.

Keras & TensorFlow 2

TensorFlow 2 is an end-to-end, open-source machine learning platform. You can think of it as an infrastructure layer for differentiable programming. It combines four key abilities:

- Efficiently executing low-level tensor operations on CPU, GPU, or TPU.
- Computing the gradient of arbitrary differentiable expressions.
- Scaling computation to many devices, such as clusters of hundreds of GPUs.
- Exporting programs ("graphs") to external runtimes such as servers, browsers, mobile and embedded devices.

Keras is the high-level API of TensorFlow 2: an approachable, highly-productive interface for solving machine learning problems, with a focus on modern deep learning. It provides essential abstractions and building blocks for developing and shipping machine learning solutions with high iteration velocity.

Keras empowers engineers and researchers to take full advantage of the scalability and cross-platform capabilities of TensorFlow 2: you can run Keras on TPU or on large clusters of GPUs, and you can export your Keras models to run in the browser or on a mobile device.N5T

Unit 3: Computer Vision

Learning Outcomes:

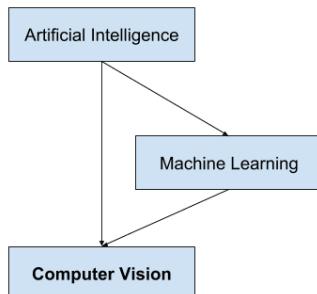
- Understand basic key concepts of computer vision
- Understand concept of different layers in Convolution neural network
- Implement practical of computer vision using openCV

3.1 What is Computer Vision (CV)?

Computer vision (CV) is an artificial intelligence (AI) subcategory that focuses on developing and deploying digital systems that process, analyze, and interpret visual input.

The objective of computer vision is to allow computers to recognize an item or person in a digital image and take appropriate action.

Convolutional neural networks (CNNs) are used in computer vision to analyze visual input at the pixel level.



Difference between Image Processing and Computer Vision:

Image Processing	Computer Vision
Image processing is mainly focused on processing the raw input images to enhance them or preparing them to do other tasks	Computer vision is focused on extracting information from the input images or videos to have a proper understanding of them to predict the visual input like human brain.
Image processing uses methods like Anisotropic diffusion, Hidden Markov models, Independent component analysis, Different Filtering etc.	Image processing is one of the methods that is used for computer vision along with other Machine learning techniques, CNN etc.
Image Processing is a subset of Computer Vision	Computer Vision is a superset of Image Processing.
Examples of some Image Processing applications are- Rescaling image (Digital Zoom), Correcting illumination, Changing tones etc.	Examples of some Computer Vision applications are- Object detection, Face detection, Hand writing recognition etc.

3.2 Image Fundamentals:

What's an image?

An image refers to a 2D light intensity function $f(x,y)$, where (x,y) denote spatial coordinates and the value of f at any point (x,y) is proportional to the brightness or gray levels of the image at that point.

A digital image is an image $f(x,y)$ that has been discretized both in spatial coordinates and brightness.

The elements of such a digital array are called image elements or pixels.

Types of Images



(a)

Color Image



(b)

Grayscale Image



(c)

Binary Image

Pixels

A pixel is the smallest unit of a digital image or graphic that can be displayed and represented on a digital display device. A pixel is the basic logical unit in digital graphics. Pixels are combined to form a complete image, video, text, or any visible thing on a computer display.

Gray-scale images

Grayscale images are monochrome images, Means they have only one color. Grayscale images do not contain any information about color. Each pixel determines available different grey levels.

A normal grayscale image contains 8 bits/pixel data, which has 256 different grey levels. In medical images and astronomy, 12 or 16 bits/pixel images are used.

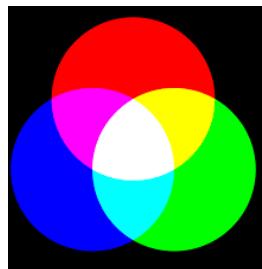


<https://www.javatpoint.com/dip-types-of-images>

Colour images

Colour images are three band monochrome images in which, each band contains a different color and the actual information is stored in the digital image. The color images contain gray level information in each spectral band.

The images are represented as red, green and blue (RGB images). And each color image has 24 bits/pixel means 8 bits for each of the three color band(RGB).



<https://www.javatpoint.com/dip-types-of-images>

3.3 Computer Vision Using OpenCV

OpenCV is a very famous library for computer vision and image processing tasks. It one of the most used pythons open-source library for computer vision and image data. It is used in various tasks such as image denoising, image thresholding, edge detection, corner detection, contours, image pyramids, image segmentation, face detection and many more.

*Let's understand computer vision with Practical
IMPORT LIBRARIES*

```
import os
import numpy as np
import cv2
import matplotlib.pyplot as plt
%matplotlib inline
```

RGB IMAGE AND RESIZING

An RGB image where RGB indicates Red, Green, and Blue respectively can be considered as three images stacked on top of each other. It also has a nickname called ‘True Color Image’ as it represents a real-life image as close as possible and is based on human perception of colors.

```
height = 224
width = 224
```

```
font_size = 20
plt.figure(figsize=(15, 8))
for i, path in enumerate(paths):
    name = os.path.split(path)[-1]
    img = cv2.imread(path, cv2.IMREAD_COLOR)
    resized_img = cv2.resize(img, (height, width))
    plt.subplot(1, 2, i+1).set_title(name[ : -4], fontsize = font_size);
    plt.axis('off')
    plt.imshow(cv2.cvtColor(resized_img, cv2.COLOR_BGR2RGB))
plt.show()
```

GRAYSCALE IMAGE

Grayscale images are images that are shades of grey. It represents the degree of luminosity and carries the intensity information of pixels in the image. Black is the weakest intensity and white is the strongest intensity.

```
plt.figure(figsize=(15, 8))
for i, path in enumerate(paths):
    name = os.path.split(path)[-1]
    img = cv2.imread(path, 0)
    resized_img = cv2.resize(img, (height, width))
    plt.subplot(1, 2, i + 1).set_title(f'Grayscale {name[ : -4]} Image',
                                    fontsize = font_size); plt.axis('off')
    plt.imshow(resized_img, cmap='gray')
plt.show()
```

IMAGE DENOISING

Image denoising removes noise from the image. It is also known as ‘Image Smoothing’. The image is convolved with a low pass filter kernel which gets rid of high-frequency content like edges of an image

```
for i, path in enumerate(paths):
    name = os.path.split(path)[-1]
    img = cv2.imread(path, cv2.IMREAD_COLOR)
    resized_img = cv2.resize(img, (height, width))
    denoised_img = cv2.medianBlur(resized_img, 5)
    plt.figure(figsize=(15, 8))
```

```
plt.subplot(1, 2, 1).set_title(f'Original {name[ : -4]} Image',  
    fontsize = font_size); plt.axis('off')  
  
plt.imshow(cv2.cvtColor(resized_img, cv2.COLOR_BGR2RGB))  
  
plt.subplot(1, 2, 2).set_title(f'After Median Filtering of {name[ :  
-4]} Image', fontsize = font_size); plt.axis('off')  
  
plt.imshow(cv2.cvtColor(denoised_img, cv2.COLOR_BGR2RGB))  
  
plt.show()
```

IMAGE THRESHOLDING

Image Thresholding is self-explanatory. If the pixel value in an image is above a certain threshold, a particular value is assigned and if it is below the threshold, another particular value is assigned.

```
for i, path in enumerate(paths):  
  
    name = os.path.split(path)[-1]  
  
    img = cv2.imread(path, 0)  
  
    resized_img = cv2.resize(img, (height, width))  
  
    denoised_img = cv2.medianBlur(resized_img, 5)  
  
    th = cv2.adaptiveThreshold(denoised_img, maxValue = 255,  
        adaptiveMethod = cv2.ADAPTIVE_THRESH_GAUSSIAN_C, thresholdType =  
        cv2.THRESH_BINARY, blockSize = 11, C = 2)  
  
    plt.figure(figsize=(15, 8))  
  
    plt.subplot(1, 2, 1).set_title(f'Grayscale {name[ : -4]} Image',  
        fontsize = font_size); plt.axis('off')  
  
    plt.imshow(resized_img, cmap = 'gray')  
  
    plt.subplot(1, 2, 2).set_title(f'After Adapative Thresholding of  
    {name[ : -4]} Image', fontsize = font_size); plt.axis('off')  
  
    plt.imshow(cv2.cvtColor(th, cv2.COLOR_BGR2RGB))  
  
    plt.show()
```

IMAGE GRADIENTS

Gradients are the slope of the tangent of the graph of the function. Image gradients find the edges of a grayscale image in the x and y-direction. This can be done by calculating derivates in both directions using Sobel x and Sobel y operations.

```
for i, path in enumerate(paths):
    name = os.path.split(path)[-1]
    img = cv2.imread(path, 0)
    resized_img = cv2.resize(img, (height, width))
    laplacian = cv2.Laplacian(resized_img, cv2.CV_64F)
    plt.figure(figsize=(15, 8))
    plt.subplot(1, 2, 1).set_title(f'Grayscale {name[ : -4]} Image',
        fontsize = font_size); plt.axis('off')
    plt.imshow(resized_img, cmap = 'gray')
    plt.subplot(1, 2, 2).set_title(f'After finding Laplacian Derivatives
        of {name[ : -4]} Image', fontsize = font_size); plt.axis('off')
    plt.imshow(cv2.cvtColor(laplacian.astype('float32'),
        cv2.COLOR_BGR2RGB))
    plt.show()
```

EDGE DETECTION

Edge Detection is performed using Canny Edge Detection which is a multi-stage algorithm. The stages to achieve edge detection are as follows. Noise Reduction – Smoothen image using Gaussian filter

Find Intensity Gradient – Using the Sobel kernel, find the first derivative in the horizontal (Gx) and vertical (Gy) directions.

```
for i, path in enumerate(paths):
    name = os.path.split(path)[-1]
    img = cv2.imread(path, 0)
    resized_img = cv2.resize(img, (height, width))
    edges = cv2.Canny(resized_img, threshold1 = 100, threshold2 = 200)
    plt.figure(figsize=(15, 8))
    plt.subplot(1, 2, 1).set_title(f'Grayscale {name[ : -4]} Image',
        fontsize = font_size); plt.axis('off')
    plt.imshow(resized_img, cmap = 'gray')
```

```
plt.subplot(1, 2, 2).set_title(f'After Canny Edge Detection of {name[ : -4]} Image', fontsize = font_size); plt.axis('off')
plt.imshow(cv2.cvtColor(edges, cv2.COLOR_BGR2RGB))
```

FOURIER TRANSFORM ON IMAGE

Fourier Transform analyzes the frequency characteristics of an image. Discrete Fourier Transform is used to find the frequency domain.

```
for i, path in enumerate(paths):
    name = os.path.split(path)[-1]
    img = cv2.imread(path, 0)
    resized_img = cv2.resize(img, (height, width))
    freq = np.fft.fft2(resized_img)
    freq_shift = np.fft.fftshift(freq)
    magnitude_spectrum = 20 * np.log(np.abs(freq_shift))
    plt.figure(figsize=(15, 8))
    plt.subplot(1, 2, 1).set_title(f'Grayscale {name[ : -4]} Image', fontsize = font_size); plt.axis('off')
    plt.imshow(cv2.cvtColor(resized_img, cv2.COLOR_BGR2RGB))
    plt.subplot(1, 2, 2).set_title(f'Magnitude Spectrum of {name[ : -4]} Image', fontsize = font_size); plt.axis('off')
    plt.imshow(magnitude_spectrum, cmap = 'gray')
    plt.show()
```

MORPHOLOGICAL TRANSFORMATION OF IMAGE

Morphological Transformation is usually applied on binary images where it takes an image and a kernel which is a structuring element as inputs. Binary images may contain imperfections like texture and noise.

These transformations help in correcting these imperfections by accounting for the form of the image

```
kernel = np.ones((5,5), np.uint8)
plt.figure(figsize=(15, 8))
img = cv2.imread('../input/cv-images/morph-min.jpg',
cv2.IMREAD_COLOR)
resized_img = cv2.resize(img, (height, width))
morph_open = cv2.morphologyEx(resized_img, cv2.MORPH_OPEN, kernel)
```

```
morph_close = cv2.morphologyEx(morph_open, cv2.MORPH_CLOSE, kernel)
plt.subplot(1,2,1).set_title('Original Digit - 7 Image', fontsize = font_size); plt.axis('off')
plt.imshow(cv2.cvtColor(resized_img, cv2.COLOR_BGR2RGB))
plt.subplot(1,2,2).set_title('After Morphological Opening and Closing of Digit - 7 Image', fontsize = font_size); plt.axis('off')
plt.imshow(cv2.cvtColor(morph_close, cv2.COLOR_BGR2RGB))
plt.show()
```

GEOMETRIC TRANSFORMATION OF IMAGE

Geometric Transformation of images is achieved by two transformation functions namely cv2.warpAffine and cv2.warpPerspective that receive a 2×3 and 3×3 transformation matrix respectively.

```
pts1 = np.float32([[1550, 1170],[2850, 1370],[50, 2600],[1850, 3450]])
pts2 = np.float32([[0,0],[4160,0],[0,3120],[4160,3120]])
img = cv2.imread('../input/cv-images/book-min.jpg',
cv2.IMREAD_COLOR)
transformation_matrix = cv2.getPerspectiveTransform(pts1, pts2)
final_img = cv2.warpPerspective(img, M = transformation_matrix,
dsize = (4160, 3120))

img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img = cv2.resize(img, (256, 256))
final_img = cv2.cvtColor(final_img, cv2.COLOR_BGR2RGB)
final_img = cv2.resize(final_img, (256, 256))
plt.figure(figsize=(15, 8))

plt.subplot(1,2,1).set_title('Original Book Image', fontsize = font_size); plt.axis('off')
plt.imshow(img)

plt.subplot(1,2,2).set_title('After Perspective Transformation of Book Image', fontsize = font_size); plt.axis('off')
plt.imshow(final_img)
plt.show()
```

3.4 PRACTICAL: Canny Edge Detection

Canny edge detection is a image processing method used to detect edges in an image while suppressing noise. The main steps are as follows:

ALGORITHM STEPS

1. Grayscale conversion
2. Gaussian blur
3. Determine the intensity gradients
4. Non maximum suppression
5. Double thresholding
6. Edge tracking by hysteresis
7. Cleaning up

CODE

```
import org.opencv.core.Core;
import org.opencv.core.Mat;

import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;

public class CannyEdgeDetection {
    public static void main(String args[]) throws Exception {
        // Loading the OpenCV core library
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);

        // Reading the Image from the file and storing it in to a Matrix object
        String file = "E:/OpenCV/chap17/canny_input.jpg";

        // Reading the image
        Mat src = Imgcodecs.imread(file);

        // Creating an empty matrix to store the result
        Mat gray = new Mat();

        // Converting the image from color to Gray
        Imgproc.cvtColor(src, gray, Imgproc.COLOR_BGR2GRAY);
        Mat edges = new Mat();

        // Detecting the edges
        Imgproc.Canny(gray, edges, 60, 60*3);
```

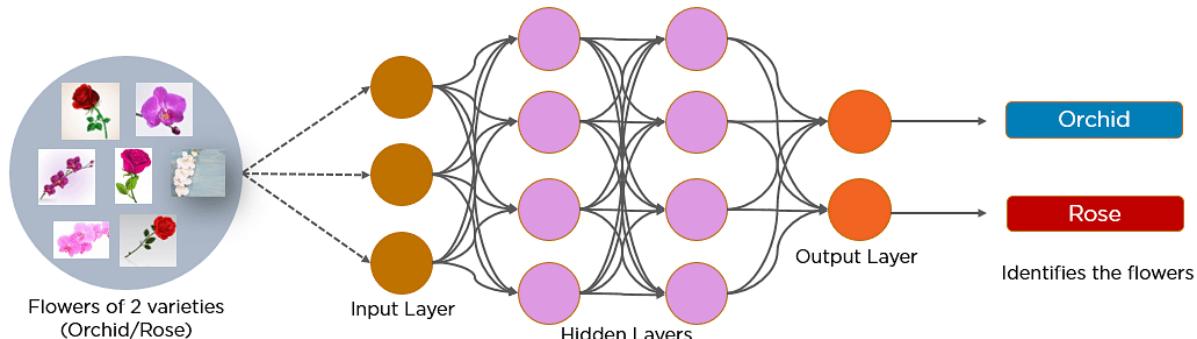
```

// Writing the image
Imgcodecs.imwrite("E:/OpenCV/chap17/canny_output.jpg", edges);
System.out.println("Image Loaded");
}
}

```

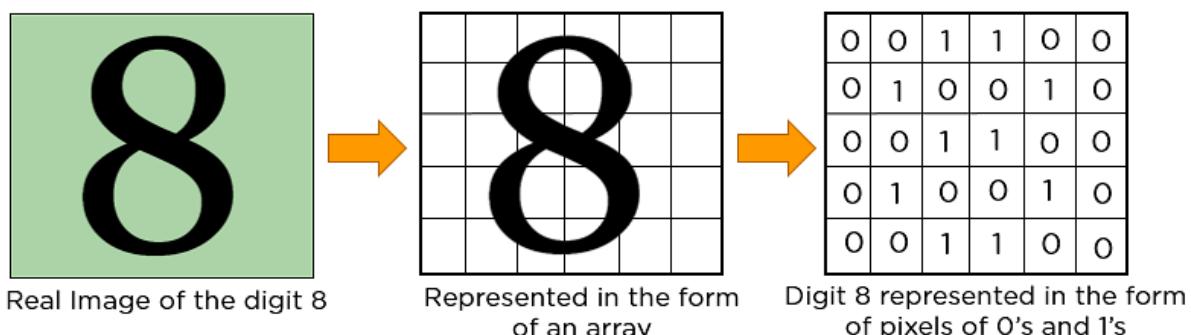
3.5 Convolutional Neural Network

A convolutional neural network is a feed-forward neural network that is generally used to analyze visual images by processing data with grid-like topology. It's also known as a ConvNet. A convolutional neural network is used to detect and classify objects in an image.



<https://www.simplilearn.com/tutorials/deep-learning-tutorial/convolutional-neural-network>

In CNN, every image is represented in the form of an array of pixel values



Layers in a Convolutional Neural Network

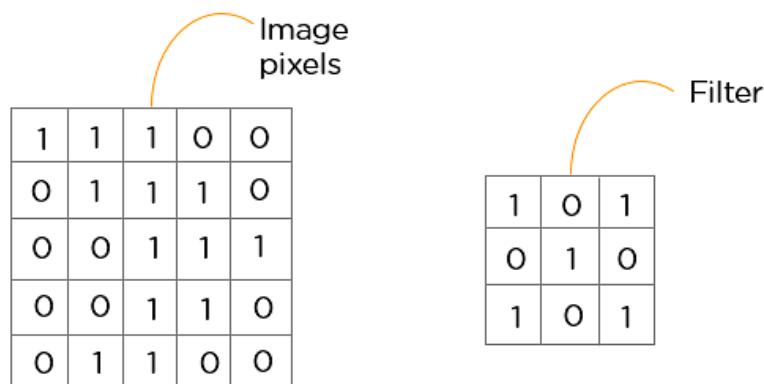
A convolution neural network has multiple hidden layers that help in extracting information from an image. The four important layers in CNN are:

- Convolution layer
- ReLU layer
- Pooling layer
- Fully connected layer

Convolution Layer

This is the first step in the process of extracting valuable features from an image. A convolution layer has several filters that perform the convolution operation. Every image is considered as a matrix of pixel values.

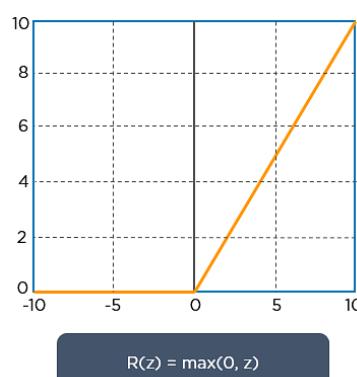
Consider the following 5x5 image whose pixel values are either 0 or 1. There's also a filter matrix with a dimension of 3x3. Slide the filter matrix over the image and compute the dot product to get the convolved feature matrix



ReLU layer

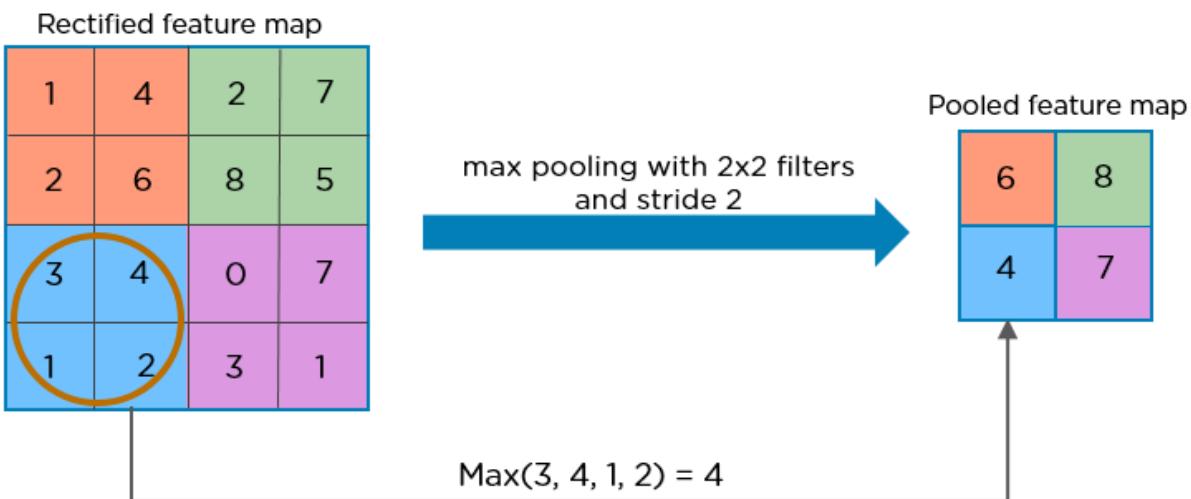
ReLU stands for the rectified linear unit. Once the feature maps are extracted, the next step is to move them to a ReLU layer.

ReLU performs an element-wise operation and sets all the negative pixels to 0. It introduces non-linearity to the network, and the generated output is a rectified feature map. Below is the graph of a ReLU function:



Pooling Layer

Pooling is a down-sampling operation that reduces the dimensionality of the feature map. The rectified feature map now goes through a pooling layer to generate a pooled feature map.



code

```
#importing the required libraries
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPool2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Dense
#loading data
(X_train,y_train) , (X_test,y_test)=mnist.load_data()
#reshaping data
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], X_train.shape[2], 1))
X_test = X_test.reshape((X_test.shape[0],X_test.shape[1],X_test.shape[2],1))
#checking the shape after reshaping
print(X_train.shape)
print(X_test.shape)
#normalizing the pixel values
X_train=X_train/255
X_test=X_test/255
#defineing model
model=Sequential()
#adding convolution layer
model.add(Conv2D(32,(3,3),activation='relu',input_shape=(28,28,1)))
#adding pooling layer
model.add(MaxPool2D(2,2))
#adding fully connected layer
model.add(Flatten())
model.add(Dense(100,activation='relu'))
#adding output layer
```

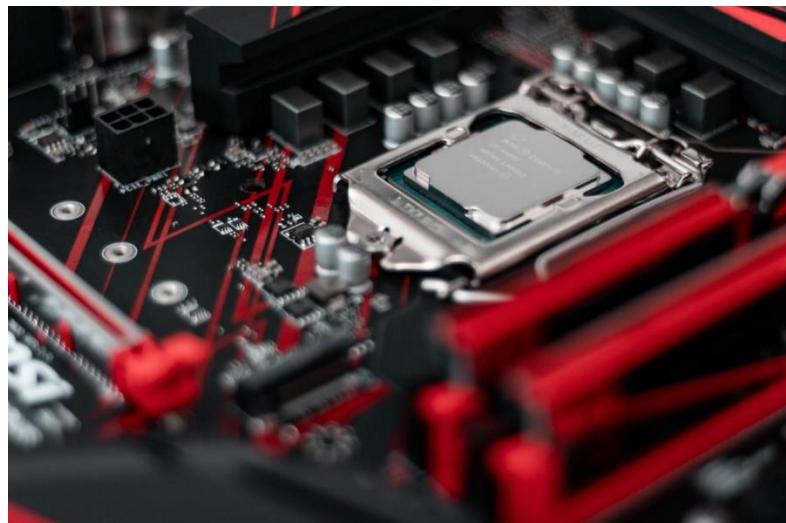
```
model.add(Dense(10,activation='softmax'))  
#compiling the model  
model.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['a  
ccuracy'])  
#fitting the model  
model.fit(X_train,y_train,epochs=10)
```

Unit 4: Computer Vision with OpenVINO?

Learning Outcomes:

- Understand the concept of OpenVINO Toolkit
- Understand the Workflow of OpenVINO Toolkit
- ModelZoo & Model Optimizers use for OpenVINO-IR
- Able to create application on Edge Computing Using OpenVINO & Raspberry-PI

4.1 Introduction to OpenVINO



Reference : [gray and green computer processor and black motherboard photo – Free Image on Unsplash](#)

OpenVINO stands for Open Visual Inference and Neural Network Optimization. It is a toolkit provided by Intel to facilitate faster inference of deep learning models. It helps developers to create cost-effective and robust computer vision applications. It enables deep learning inference at the edge and supports heterogeneous execution across computer vision accelerators — CPU, GPU, Intel® Movidius™ Neural Compute Stick, and FPGA. It supports a large number of deep learning models out of the box. You can check out this [link](#) to know more about the model zoo.

OpenVINO is a cross-platform deep learning toolkit developed by Intel. The name stands for “Open Visual Inference and Neural Network Optimization.” OpenVINO

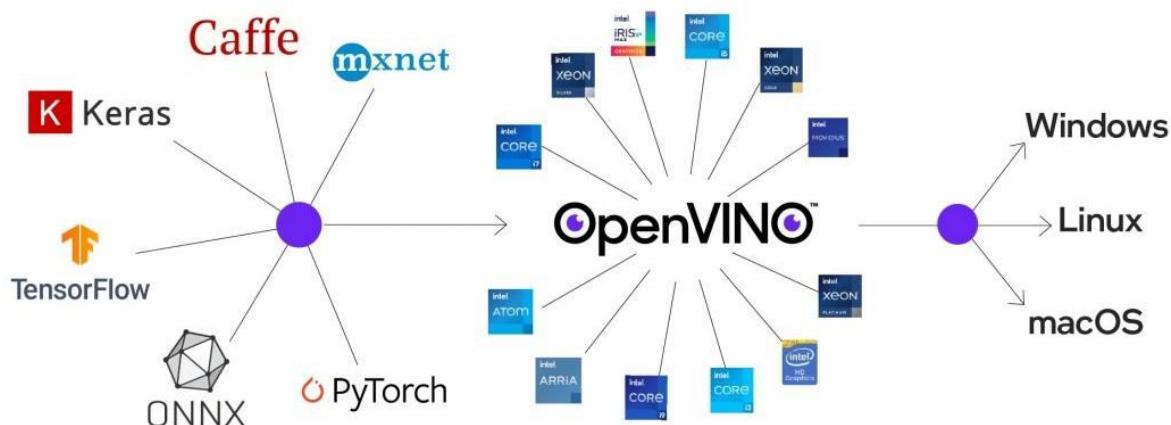
focuses on optimizing neural network inference with a write-once, deploy-anywhere approach for Intel hardware platforms.

The toolkit is free for use under Apache License version 2.0 and has two versions:

- [OpenVINO toolkit](#), which is supported by the open-source community and the
- [Intel Distribution of OpenVINO toolkit](#), which is supported by Intel.

Using the OpenVINO toolkit, software developers can select models and deploy pre-trained deep learning models ([YOLO v3](#), [ResNet 50](#), etc.) through a high-level C++ Inference Engine API integrated with application logic.

Hence, OpenVINO offers integrated functionalities for expediting the development of applications and solutions that solve several tasks using [computer vision](#), automatic speech recognition, natural language processing, recommendation systems, machine learning, and more.

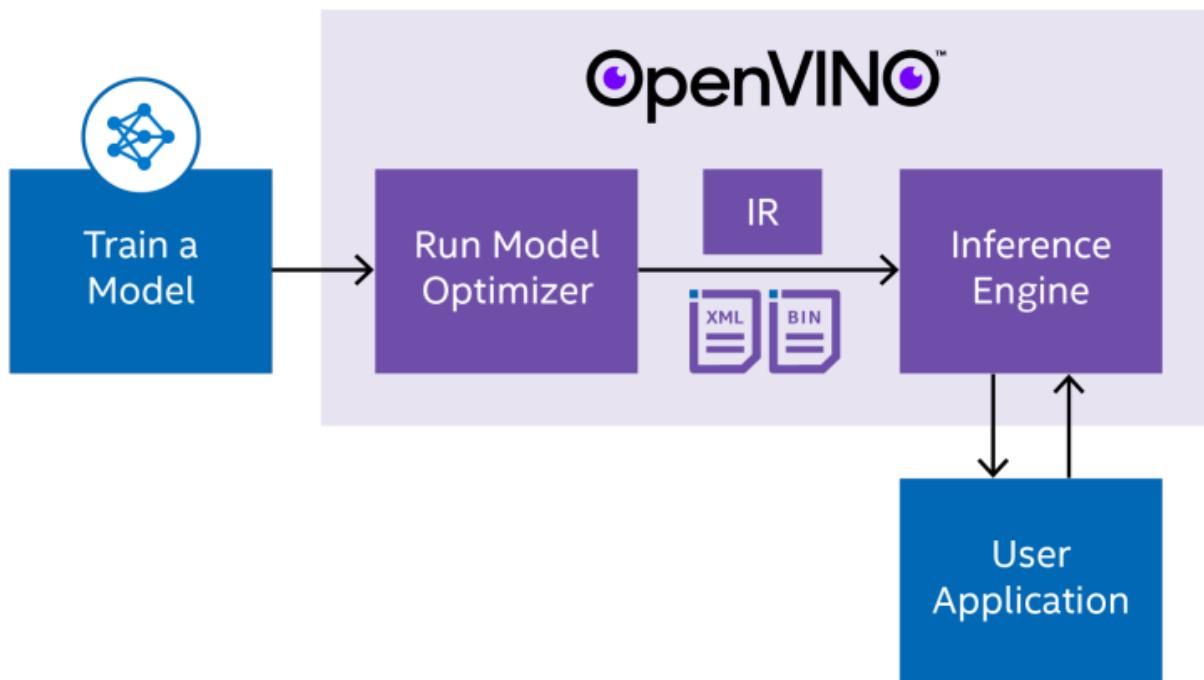


Reference: [What is OpenVINO? - The Ultimate Overview \(Updated\) - viso.ai](#)

Deep Neural Networks (DNNs) have made considerable advances in many industrial domains in the past few years, bringing the accuracy of computer vision algorithms to a new level. However, deploying and producing such accurate and useful models requires adaptations for the hardware and computational methods.

4.2 OpenVINO Toolkit Components

The two main components of the OpenVINO toolkit are Model Optimizer and Inference Engine. So, we will dive into their details, to better understand their role and internal working.



Reference: [What is OpenVINO? - The Ultimate Overview \(Updated\) - viso.ai](https://viso.ai/what-is-openvino-the-ultimate-overview-updated/)

Model Optimizer

Model optimizer is a cross-platform command-line tool that facilitates the transition between the training and deployment environment. It adjusts the deep learning models for optimal execution on end-point target devices.

Working

Model Optimizer loads a model into memory, reads it, builds the internal representation of the model, optimizes it, and produces the **Intermediate Representation**. Intermediate Representation is the only format that the Inference Engine accepts and understands. The Model Optimizer does not infer models. It is an offline tool that runs before the inference takes place. Model Optimizer has two main purposes:

1. **Produce a valid Intermediate Representation.** The primary responsibility of the Model Optimizer is to produce two files (.xml and .bin) that form the Intermediate Representation.
2. **Produce an optimized Intermediate Representation.** Pretrained models contain layers that are important for training, such as the Dropout layer. These layers are useless during inference and might increase the inference time. In many cases, these layers can be automatically removed from the resulting Intermediate Representation. However, if a group of layers can be represented as one mathematical operation, and thus as a single layer, the Model Optimizer recognizes such patterns and replaces these layers with only one. The result is an Intermediate Representation that has fewer layers than the original model. This decreases the inference time.

Operations of Model Optimizer

Reshaping —The Model Optimizer allows us to reshape our input images. Suppose you have trained your model with an image size of 256 * 256 and you want to convert the image size to 100 * 100, then you can simply pass on the new image size as a command-line argument and the Model Optimizer will handle the rest for you.

Batching — We can change the batch size of our model at inference time. We can just pass the value of batch size as a command-line argument. We can also pass our image size like this [4,3,100,100]. Here we are specifying that we need 4 images with dimensions 100*100*3 i.e RGB images having 3 channels and having width and height as 100. Important thing to note here is that now the inference will be slower as we are using a batch of 4 images for inference rather than using just a single image.

Modifying the Network Structure — We can modify the structure of our network, i.e we can remove layers from the top or from the bottom. We can specify a particular layer from where we want the execution to begin or where we want the execution to end.

Standardizing and Scaling — We can perform operations like normalization (mean subtraction) and standardization on our input data.

Quantization — It is an important step in the optimization process. Most deep learning models generally use the FP32 format for their input data. The FP32 format consumes a lot of memory and hence increases the inference time. So, intuitively we may think, that we can reduce our inference time by changing the format of our input data. There are various other formats like FP16 and INT8 which we can use, but we need to be careful while performing quantization as it can also result in loss of accuracy. Using the INT8 format can help us in reducing our inference time significantly, but currently, only certain layers are compatible with the INT8 format: Convolution, ReLU, Pooling, Eltwise and Concat. So, we essentially perform hybrid execution where some layers use FP32 format whereas some layers use INT8 format. There is a separate layer that handles these conversions. i.e we don't have to explicitly specify the type conversion from one layer to another.

Calibrate layer— handles all these intricate type conversions. The way it works is as follows:

- Initially, we need to define a threshold value. It determines the drop in accuracy are we willing to accept.
- The Calibrate layer then takes a subset of data and tries to convert the data format of layers from FP32 to INT8 or FP16.
- It then checks the accuracy drop and if it less than the specified threshold value, then the conversion takes place.

Inference Engine

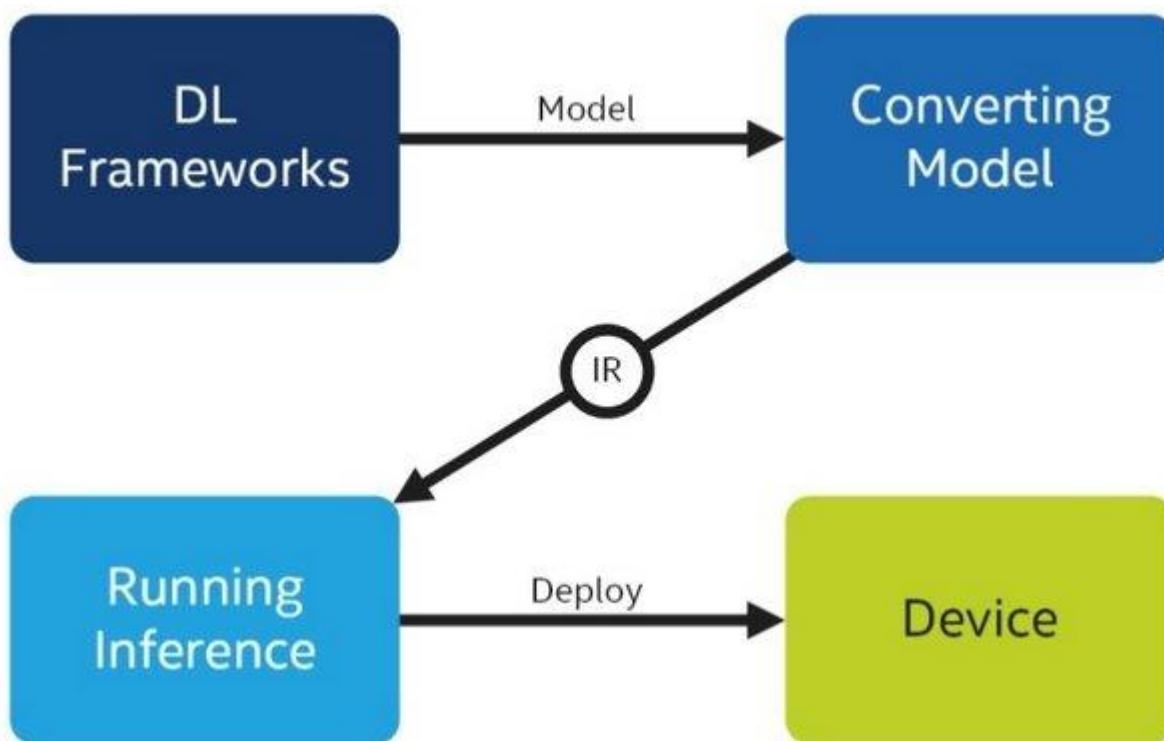
After using the Model Optimizer to create an intermediate representation (IR), we use the Inference Engine to infer input data. The Inference Engine is a C++ library with a set of C++ classes to infer input data (images) and get a result. The C++ library provides an API to read the Intermediate Representation, set the input and output formats, and execute the model on devices. The heterogeneous execution of the model is possible because of the Inference Engine. It uses different plug-ins for different devices.

Heterogeneous Plug-in — We can execute the same program on multiple devices. We just need to pass in the target device as a command-line argument and the Inference Engine will take care of the rest i.e. we can run the same piece of code on a CPU, GPU, VPU or any other device compatible with the OpenVINO toolkit.

- We can also execute parts of our program on different devices i.e. some part of our program might run on CPU and other parts might be running on a FPGA or a GPU. If we specify HETERO: FPGA, CPU then the code will run primarily on an FPGA, but if suppose it encounters a particular operation which is not compatible with FPGA then it will switch to CPU.
- We can also execute certain layers on a specific device. Suppose you want to run the Convolution layer only on your GPU then you can explicitly specify it.
- The important thing to note here is that we need to be careful about the data format while specifying different hardware. Not all devices work with all the data types. Example — The Neural Compute Stick NCS2, which comes with a Movidius chip, doesn't support the INT8 format. You can check out [this](#) link to get complete information about the supported devices and their respective formats.

OpenVINO allows the optimization of DNN models for inference to be a streamlined, efficient process through the integration of various tools. The OpenVINO toolkit is based on the latest generations of [Artificial Neural Networks \(ANN\)](#), such as Convolutional Neural Networks (CNN) as well as recurrent and attention-based networks. For more information on what Artificial Neural Networks (ANN) are all about and how they are incorporated in computer vision, we suggest you read [ANN and CNN: Analyzing Differences and Similarities](#).

The OpenVINO toolkit covers both computer vision and non-computer vision workloads across Intel hardware. It maximizes performance and accelerates application development. OpenVINO aims to accelerate AI workloads and speed up time to market using a library of predetermined functions as well as pre-optimized kernels. In addition, other [computer vision tools](#) such as [OpenCV](#), OpenCL kernels, and more are included in the OpenVINO toolkit.



Reference: [OpenVINO™ Documentation](#) — [OpenVINO™ documentation](#) — Version(latest)

Benefits of OpenVINO

1. **Accelerate Performance:** Expedite computer vision workloads by enabling simple execution methods across different Intel processors and accelerators such as CPU, GPU/Intel Processor Graphics, VPU ([Intel AI Stick NCS2](#) with Myriad X), and FPGA.
2. **Streamline Deep Learning Deployment:** Utilize Convolutional Neural Network (CNN)-based deep learning functions using one common API in addition to more than 30 pre-trained models and documented code samples. With more than 100 public and custom models, the OpenVINO toolkit streamlines deep learning innovation by providing one centralized method for implementing dozens of deep learning models.
3. **Extend and Customize:** [OpenCL](#) (Open Computing Language) Kernels and other tools offer an open, royalty-free standard way to add custom code pieces straight into the workload pipeline, customize deep learning model layers without the burden of framework overheads, and implement parallel programming of various accelerators.
4. **Innovate Artificial Intelligence:** The complete [Deep Learning Deployment Toolkit](#) within OpenVINO allows users to extend artificial intelligence within private applications and optimize artificial intelligence “[all the way to the cloud](#)” with processes such as the Model Optimizer, Intermediate Representation, nGraph Integration, and more.
5. **Full Viso Suite Integration (End-To-End):** OpenVINO is fully integrated with the enterprise no-code computer vision platform [Viso Suite](#). Viso Suite provides

pre-built modules to fetch the video feed of any digital camera (IP cameras, webcams, etc.) and multi-camera support. Visual programming with logic workflows allows fast building and updating of complete computer vision applications that can be deployed to edge devices – all [within one platform](#).

4.3 PRACTICAL: Installing Intel OpenVINO Toolkit

In this section, we will cover the installation steps for the **Ubuntu 20.04** OS, 2021.3 version (the latest version available currently) of the Intel OpenVINO Toolkit. The same steps also apply, if you are on Ubuntu 18.04. We've listed each step to install Intel OpenVINO Toolkit in detail here. Let's go over it together now.

Step 1: Download the Correct Version of OpenVINO Toolkit

Head over to the [official download page](#) to get the correct version of OpenVINO, after choosing all the prerequisites according to your needs and OS.

Choose a Preferred Package
You can customize the selections to fit your needs.

Environment

Dev Tools
Best option to develop and optimize deep-learning models

Runtime
You already have a model and want to run inference on it

Operating System

Windows macOS Linux

OpenVINO™ Version

2022.1 (recommended)
Latest standard release

2021.4.2 LTS
Latest LTS release

2020.3 LTS
Previous LTS release

Language

Python
Included by default, and cannot be unselected

C++

Distribution

Offline Installer
Recommended option

Online Installer

PIP

APT
(Requires Ubuntu Linux)

YUM
(Requires Red Hat Linux)

Gitee
Source

Docker

[Learn more about distribution options](#)

[Try OpenVINO on Intel® DevCloud](#)

Download Intel® Distribution of OpenVINO™ Toolkit

[Install instructions](#)

[Get started guide](#)

[OpenVINO Notebooks](#)

Download

Ref: [Download_page](#)

For example, the following image shows us selecting to download the **local installer** of **OpenVINO 2021.3** version for the **Linux operating system**. Intel OpenVINO Toolkit official download page. Choosing the correct version of OpenVINO Toolkit, as per the operating system.

Which version should you download? Any of the older, available versions will work too. But better to go with the latest version for it provides the most updated functionalities of the toolkit. Choose any version, starting from OpenVINO 2020. **Just keep in mind that all these versions are fully supported only on Ubuntu 18.04 LTS and 20.04 LTS.**

Click the download button. It will download a file named `I_openvino_toolkit_p_<version>.tgz`, where `<version>` is the version number you have chosen to download.

Step 2. Unpack the Installer File

Open your terminal and cd into the directory where you kept the OpenVINO installer file. Then unpack the file, using the following command:

```
1 | tar -xvzf I_openvino_toolkit_p_<version>.tgz
```

After unpacking, you will see a `I_openvino_toolkit_p_<version>` folder in your current working directory. This contains the OpenVINO installer and other required files.

Step 3. Install OpenVINO Toolkit

Now, cd into the `I_openvino_toolkit_p_<version>`. You get three options to install OpenVINO on your system.

- The graphical user interface (GUI) installation wizard
- The command line installer
- The command line installer, with silent instructions.

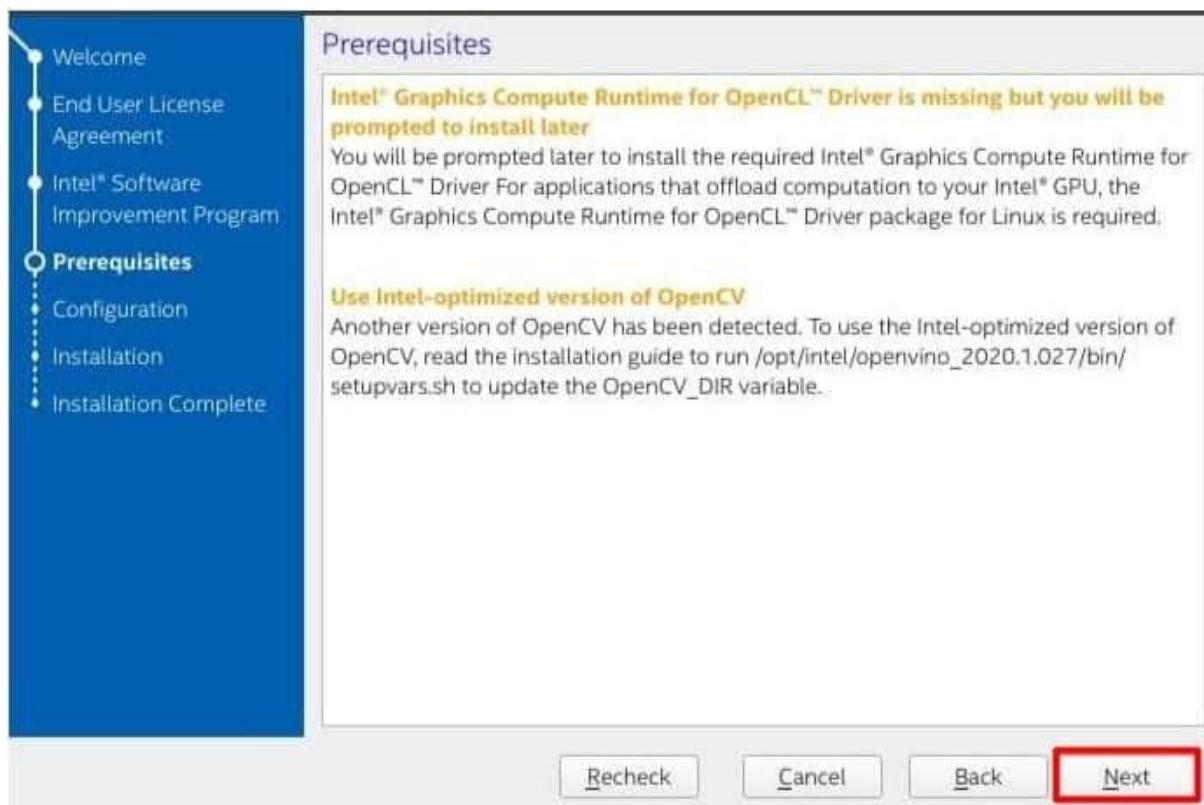
We will use the GUI installation wizard, that is the `install_GUI.sh` file, as it is the easiest and most intuitive to follow. To start the installation, type the following command in your terminal.

```
1 | sudo ./install_GUI.sh
```

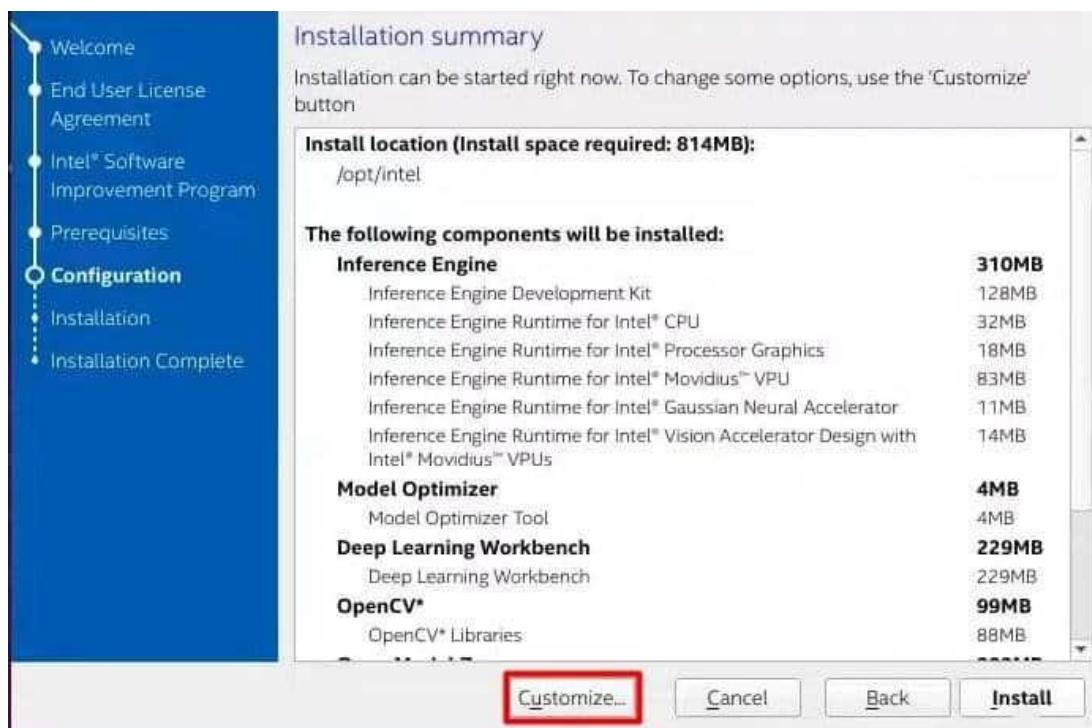
Note: You can also continue with the installation processes, without the **sudo** access.

Step 4: Following the Installation Instruction on Screen

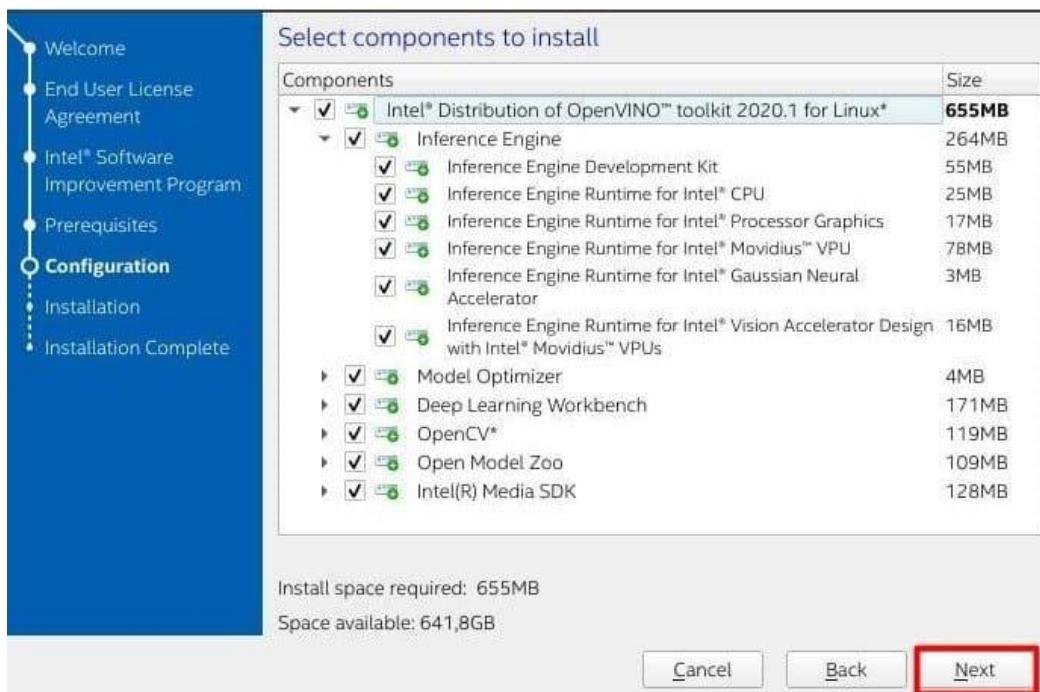
First, comes the welcome screen, and then the license agreement. Next, you will see the prerequisites screen, similar to the image below.



Intel OpenVINO prerequisites installation page shown above. Even if you get warnings for not meeting the prerequisites, you can still proceed with the installation. These can be installed later, when you are installing the dependencies. After this comes the installation-configuration summary screen.



Configuring the Intel OpenVINO installation. If you wish, you can also customize the configurations, and choose what you want to install.



Selecting the desired configuration, while installing OpenVINO Toolkit. First, choose the components, then move forward with the installation. It takes some time to complete the installation, and you will see a screen, as in the image below:



Finish the Intel OpenVINO Installation. If you installed OpenVINO Toolkit with administrator privileges, in line with the steps listed above, then it will be installed in the /opt/intel/openvino_<version>/ directory.

The next few steps ensure you face no issues while using this toolkit. These steps are necessary irrespective of you wanting to use any advanced functionality of OpenVINO or not. They are also needed to properly run the models available in the Intel Model Zoo.

Step 5: Installing Software Dependencies

These include software dependencies for:

- Intel-optimized build of OpenCV library
- Deep Learning Inference Engine
- Deep Learning Model-Optimizer tools

Change your current working directory to the install_dependencies folder.

```
1 | cd /opt/intel/openvino_2021/install_dependencies
```

Run the following script to install the dependencies.

```
1 | sudo -E ./install_openvino_dependencies.sh
```

Step 6: Set Up the Environment Variables

To set the environment variables, open a new terminal and type:

```
1 | vi ~/.bashrc
```

Go to the end of the file, and add the following line:

```
1 | source /opt/intel/openvino_2021/bin/setupvars.sh
```

Note: If you have installed the OpenVINO Toolkit in any other directory of your choice, then please provide that path. In such cases, it should be <path_to_your_directory>/openvino_2021/bin/setupvars.sh.

Now, save and close the file. Shut down your current terminal and open a new one so that system-wide changes can take place. We're only one step away from completing installation now. Just configure the Model Optimizer and you're done.

Step 7: Installing Model Optimizer Prerequisites

The Model Optimizer is a command-line tool in the OpenVINO Toolkit. Use this Model Optimizer to convert models, trained with different frameworks, into a format accepted by the OpenVINO Toolkit for inference. For your information, the OpenVINO Toolkit does not support inference directly. None of the models trained with Deep-Learning frameworks like TensorFlow, Caffe, MXNet, ONNX, or even Kaldi can help you infer. First, you'll need to convert these trained models into an Intermediate Representation (IR), which consists of:

- A .xml file, which describes the network architecture
- A .bin file that holds the weights and biases of the trained model

To convert, run the models through the Model Optimizer, after installing the necessary prerequisites. To configure the Model Optimizer, open your terminal and go to the Model Optimizer prerequisites directory.

```
1 | cd opt/intel/openvino_2021/deployment_tools/model_optimizer/install_prerequisites
```

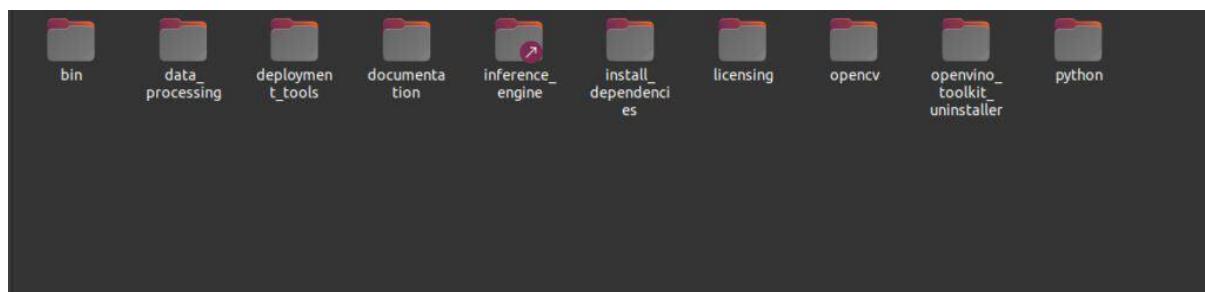
Execute the following command to run the script that will configure the Model Optimizer for Caffe, TensorFlow, MXNet, ONNX and even Kaldi, at one go.

```
1 | sudo ./install_prerequisites.sh
```

This completes the installation process for the OpenVINO Toolkit. Now, you are all set to use any model from the Intel Model Zoo, or to convert the models for inference.

4.4 Exploring OpenVINO Toolkit Directories

Navigating through the freshly installed OpenVINO Toolkit can be a bit overwhelming for newcomers. It contains a lot of directories, with each directory having numerous sub-directories.

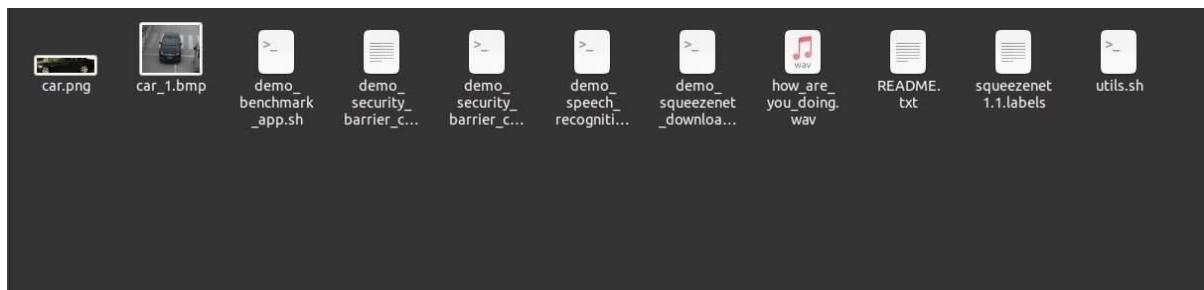


Look at the image above. You will see a similar structure inside your installed directory. We are going to discuss the directories and contents present in OpenVINO installed super directory. As you can refer to installed folder, in total the OpenVINO directory

contains approx 1266 sub directories and 6032 files. Let's go through the important ones now. While using the OpenVINO Toolkit, for most of our work, we'll be referring to the deployment_tools directory. So, let's explore the sub directories in this directory first.

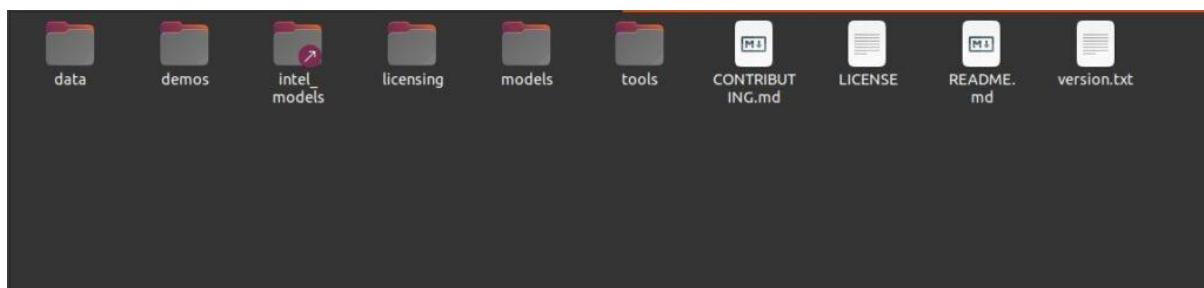
The demo Directory

The demo directory contains some demos that you can run directly, using off-the-shelf optimized models. These models will download automatically when you execute their respective scripts. Your directory structure should look, as shown below:



The open_model_zoo Directory

This is one of the most important directories, so you better get comfortable with it. You'll find here a list of all the official and public models available for use with the OpenVINO Toolkit. Along with that, there are utility scripts to download these models. The following image shows the sub directories and files inside the open_model_zoo directory.



Going over some of its subdirectories:

- The demos directory contains a lot of demo codes that we can execute using the OpenVINO Toolkit, after downloading the models from the Model Zoo. These range from classification to action recognition to object detection, and many more. This is one directory you must definitely explore.
- Coming to the models directory, it contains two subdirectories:

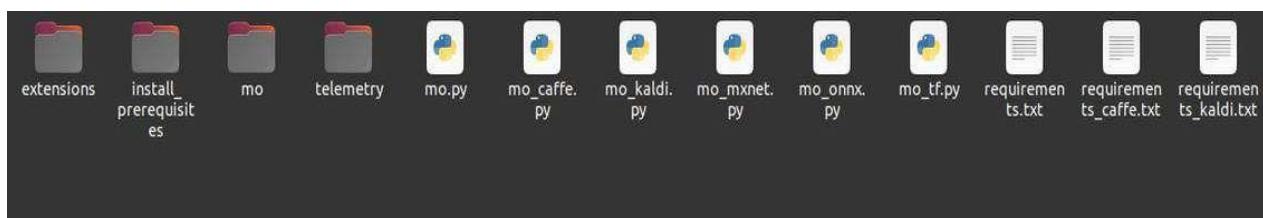
- intel – The intel folder contains a list of all the official pre-models provided by Intel. These models get downloaded directly in the FP32, FP16, and INT8 precision format, and need no further conversion.
- public – The models from this folder however are downloaded in the format of the Deep Learning framework they were trained on. For example, TensorFlow models will have .pb format, Caffe models will have .caffemodel, and so on.

Another thing to note here is that these model folders do not contain the actual model weight files. They only have the model documentation and some .yml configuration files. The documentation is a Markdown file containing the model benchmark results, the original framework, parameters, along with information on the input and output format. To download one of the intel or public models, you need the downloader.py script, which is present in the **tools/downloader** subdirectory. While executing the script, you will have to provide the name of the model you want to download. Either give the exact folder name from the models directory, or get the model name from the Markdown documentation file. More on this, when we execute one of the demos.

Another important subdirectory is the accuracy_checker inside tools. For now, just know that we can check the accuracy of different models, using the scripts in this directory.

The model_optimizer Directory

The model_optimizer also happens to be a very important directory, so understand it well. It contains the scripts that convert the trained neural network models to the Intermediate Representation (.xml and .bin files) that OpenVINO accepts. This is mainly for the public models which are not downloaded in the IR format by default.



There are separate Python scripts to convert models from different frameworks like TensorFlow, Caffe, ONNX and MXNet to the OpenVINO IR format. These scripts are named as follows:

1. mo_tf.py for TensorFlow models
2. mo_caffepy for Caffe models
3. mo_onnx.py for ONNX models
4. mo_mxnet.py for MXNet models

Along with these, there is also one **mo.py** which acts like a universal model converter script. You can just use this to convert any trained model into the OpenVINO IR format. In next coming section, we will also show you how to use the Model Optimizer, and the way models are converted from Caffe and TensorFlow frameworks to the OpenVINO IR format.

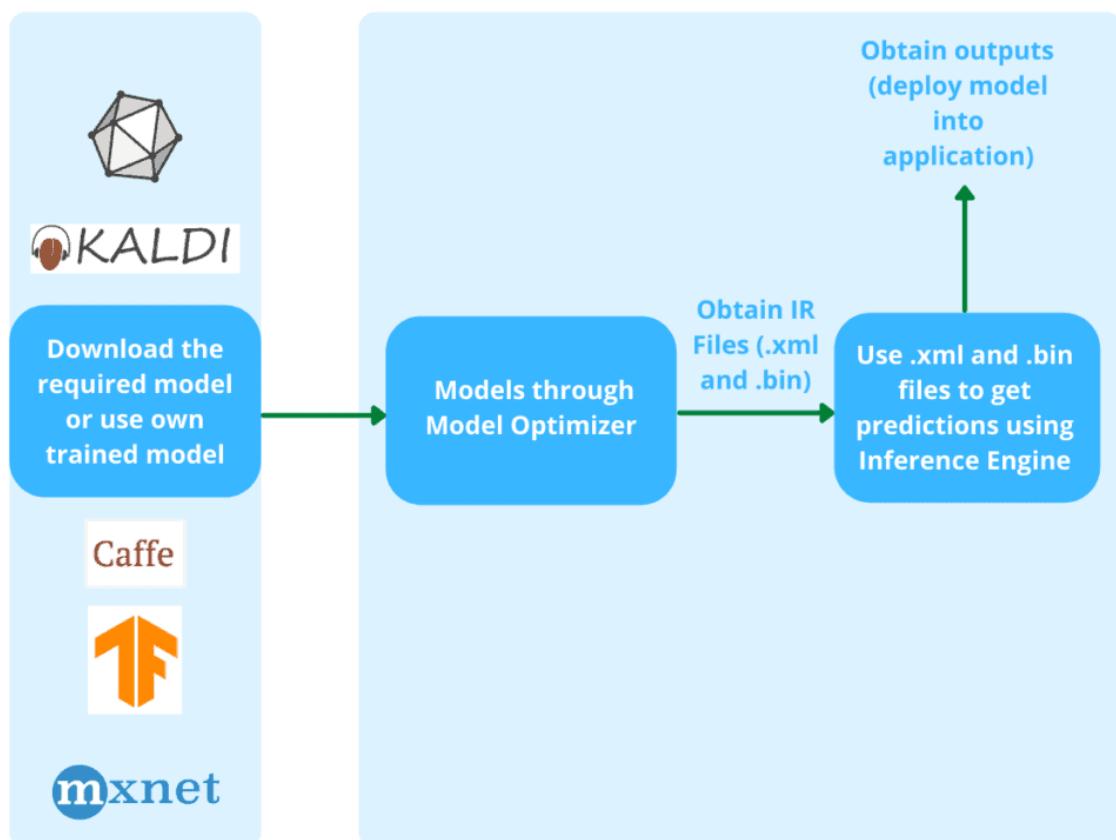
4.5 Working with Model Optimizer

Let's now focus on the model-conversion process. Here, you will basically learn to convert pre-trained Deep Learning models from supporting frameworks to the OpenVINO IR format. Well, these are the frameworks that OpenVINO toolkit supports: Caffe, TensorFlow, MXNet, Kaldi, PyTorch and ONNX.

Specifically, this section will cover the conversion of models from:

- Caffe to OpenVINO IR format

The basic conversion workflow of a pre-trained model from a specific framework to the OpenVINO Toolkit format looks like this:



Reference: [Model Optimizer](#)

The basic conversion workflow of a pre-trained model into the OpenVINO format. Obviously, there is more to the conversion process. We will cover those details in coming sections, where we will be talking about IR files generation from deep learning model

Converting a SqueezeNet Caffe Model to OpenVINO-IR Format

Let's start with an image classification model, that is the **SqueezeNet Caffe model**. It is one of the publicly available models from Model Zoo.

Start the process by following these simple steps:

1. Download the SqueezeNet Caffe model from the public Model Zoo.
2. Run the model optimizer to convert the Caffe model to IR format.

Begin by downloading the SqueezeNet-Caffe model. Our model is named **squeezezenet1.1** (a pretrained deep learning mode) in the **public** subdirectory, inside the **open_model_zoo** directory. Just provide the model name, while executing the downloader.py script to ensure the correct model is downloaded. Now head over to the tools/downloader directory, inside the deployment_tools. Go there directly by using this specific command:

```
1 | cd /opt/intel/openvino_2021/deployment_tools/open_model_zoo/tools/downloader
```

Here, we execute the downloader.py script, with the following command:

```
1 | python3 downloader.py --name squeezezenet1.1
```

Note: The --name flag accepts the exact name of the model we want to download. Giving a model name that is neither in intel nor in the public model directory will surely result in an error.

Post execution, you will see a public folder in the current working directory, containing the squeezezenet1.1 sub directory. It will consist of three files:

1. squeezezenet1.1.caffemodel
2. squeezezenet1.1.prototxt
3. squeezezenet1.1.prototxt.orig

Out of these, the ones that interest us the most are the squeezezenet1.1.caffemodel and squeezezenet1.1.prototxt files. We need these to run the Model Optimizer and obtain the .xml and .bin files. Details of individual file is listed below:

- The .caffemodel file contains the model weights.
- And the .prototxt file contains the model architecture required by the Model Optimizer.

Next, we run the model optimizer and convert the Caffe model into IR format. So, head over to the model_optimizer directory.

```
1 | cd /opt/intel/openvino_2021_3_latest/openvino_2021/deployment_tools/model_optimizer
```

Next, execute the following command:

```
1 | python mo.py --input_model squeezenet1.1.caffemodel --batch 1 --output_dir squeezenet_ir
```

Let us go over the flags we have used:

- **--input_model**: This is the path to the Caffe model that we want to convert into IR format. In the above example, we assume the Caffe model is present in the same directory as the mo.py script. **Please note that even the squeezenet1.1.prototxt file should be present in the same directory, so that the script can infer the path to the file on its own.** Else you will have to provide the path to the .prototxt file, using the **--input_proto** flag.
- **--batch**: This flag specifies the batch size for building the OpenVINO models. It comes into play during inference. By default, the batch size is 1. It is the batch size that determines the number of images/frames the model will infer on, when executing the inference scripts.
- **--output_dir**: This is the output directory in which the resulting .xml and .bin files will be stored. In the above example, we have provided it as squeezenet_ir. If absent, the directory will be automatically created.

If everything runs successfully, you should see an output similar to the one below:

Note that `install_prerequisites` scripts may install additional components.

```
[SUCCESS] Generated IR version 10 model.  
[SUCCESS] XML file: /home/diwakar/my_data/Data_Science/Projects/openvino_experiments/squeezenet1.1_caffemodel/squeezenet_ir/squeezenet1.1.xml  
[SUCCESS] BIN file: /home/diwakar/my_data/Data_Science/Projects/openvino_experiments/squeezenet1.1_caffemodel/squeezenet_ir/squeezenet1.1.bin  
[SUCCESS] Total execution time: 6.39 seconds.  
[SUCCESS] Memory consumed: 344 MB.
```

Inside the squeezenet_ir directory, you will find the two files we need:

1. squeezenet1.1.bin: This contains the model weights.

2. squeezeNet1.1.xml: It has the model topology/architecture.

We have successfully converted our first image classification Caffe model to the appropriate IR format, which we can now leverage to run inference on Intel hardware.

4.6 OpenVINO Deep Learning Workbench

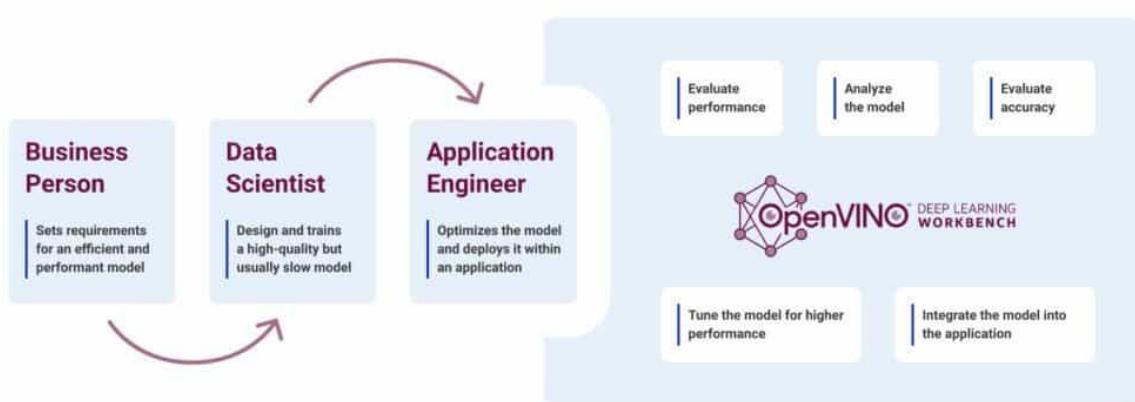
The Intel-OpenVINO Toolkit provides many great functionalities for Deep-Learning model optimization, inference and deployment. Perhaps the most interesting and practical tool among them is the Deep-Learning (DL) workbench. Not only does model optimization, calibration, and quantization get easier, but the **OpenVINO-Deep Learning Workbench** also makes the final model deployment-ready in a matter of minutes.

Let's understand what exactly is the DL workbench and why it's important. It is a web-based application provided by the Intel-OpenVINO toolkit that essentially runs in the browser. And its goal is to minimize the inference-to-deployment workflow timing for Deep-Learning models.

The DL workbench strongly integrates many of the optimization, quantization and deployment processes that OpenVINO supports, but are done manually. Its easy-to-use Graphical-User Interface lets you access almost everything, no need to bother what's going on below the hood. You can not only import models and datasets, but also visualize and optimize these models. Even compare accuracies across various runs and parameters. Also, you can export the final model, which will be deployment-ready.

Functionalities and Components of the DL Workbench

Let's study the functionalities and components that make the DL workbench so special.



OpenVINO Deep Learning Workbench

The general workflow of the DL workbench, showing the basic functionalities is shown

in above figure. The general workflow of the DL workbench and know the basic functionalities that come integrated with the following components:

1. Model Evaluator
2. Model Optimizer
3. Model Quantization Tool
4. Accuracy Checker
5. Deployment Package Manager

Now, let's go over them in a bit more detail.

1. Evaluating Model Performance

The DL workbench allows you to import any model from its list of supported frameworks, which includes TensorFlow, ONNX, Caffe, MXNet and Kaldi. But did you know you can evaluate their performance too? Also, you can simply import a dataset of your choice, like the MS COCO or the PASCAL VOC dataset, and run an evaluation on them. Even if you do not have a standardized dataset at hand, you can always generate random data in the DL workbench itself. Not only that, along with models from various frameworks, you can also import and evaluate models that are already in the OpenVINO-IR format.

2. Analyzing the Model

Furthermore, you get a great environment and visualization tools to analyze the architecture of your imported models.

Layer Name	Execution Time, ms	Layer Information
Inputs	Not Executed	Layer Type: Input Runtime Precision: U8 Execution Order: 3
inputs_U8_FP32_detector/yolo-v4-tiny/Conv/BatchNorm/FusedBatchNorm...	0.149	Layer Type: Reorder Runtime Precision: U8 Execution Order: 4
detector/yolo-v4-tiny/Conv/BatchNorm/FusedBatchNorm...	0.407	Layer Type: Convolution Runtime Precision: FP32 Execution Order: 5
detector/yolo-v4-tiny/Conv_1/BatchNorm/FusedBatchNorm...	1.24	Layer Type: Convolution Runtime Precision: FP32 Execution Order: 6
detector/yolo-v4-tiny/Conv_2/BatchNorm/FusedBatchNorm...	2.293	Layer Type: Convolution Runtime Precision: FP32 Execution Order: 7
detector/yolo-v4-tiny/Crop...	0.118	Layer Type: Crop

Layers Download Report Download the inference report in the .csv format.

Select Column
Select Filter

+ Add new filter
Apply Filter
Clear Filter

Expand Graph
Runtime Graph
Visualize Original IR
Search Layer

Coloring

By Layer Type

```

graph TD
    Inputs[Inputs] --> Reorder[Reorder]
    Reorder -- "1x3x416x416" --> Conv1[Convolution]
    Conv1 -- "1x32x208x208" --> Conv2[Convolution]
    Conv2 -- "1x64x104x104" --> Conv3[Convolution]
    Conv3 -- "1x64x104x104" --> Crop[Crop]
    
```

[Model Architecture](#)

You can check each layer, each operation in the layers, and even how much time each layer took for every single operation. Then use these insights to further optimize your model and make it all the better.

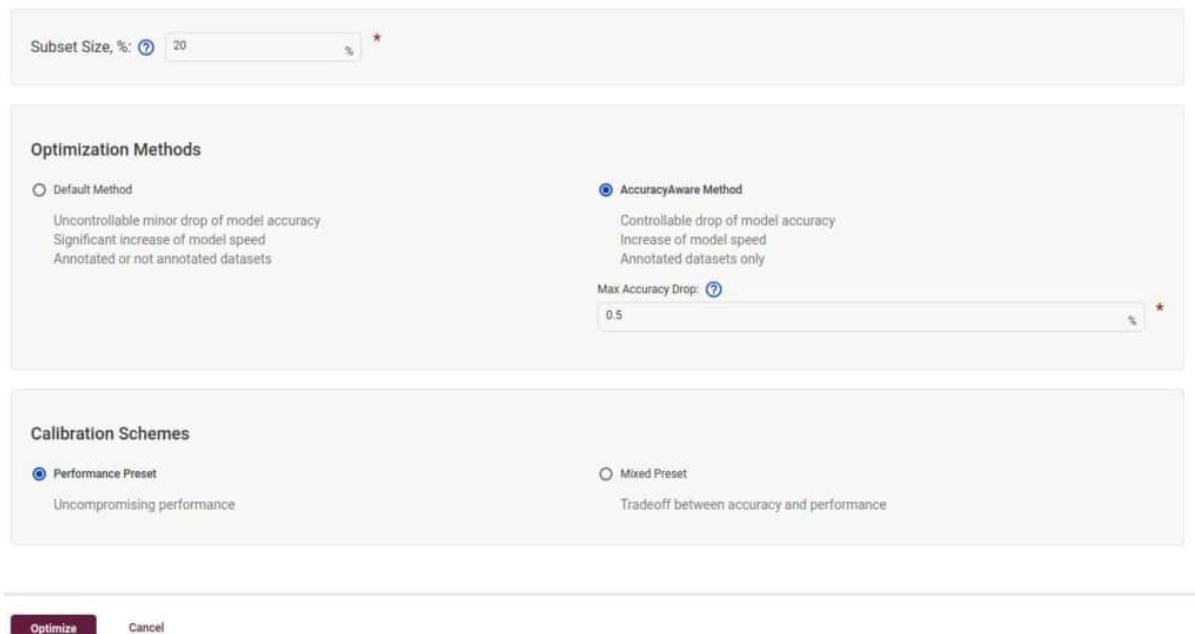
3. Accuracy Evaluation

You can evaluate the accuracy of your imported models on various datasets, and this perhaps is the most important functionality available in the DL workbench. Initial accuracy evaluation done by the Deep Learning Workbench.

Simply choose the model, the target environment and the dataset on which to run the evaluation. And the workbench handles the rest. It will give you not only the accuracy, but also the FPS of the model for your chosen target environment. From there on, you decide whether to optimize it further or straightaway deploy.

4. Model Tuning and Quantization

Be it Default-Quantization or Accuracy-Aware-Quantization, the process of model quantization becomes easier and hassle-free with the DL workbench.



The screenshot shows the DL Workbench interface for Model Tuning and Quantization. It includes:

- Optimization Methods:** A section with two radio button options:
 - Default Method: Uncontrollable minor drop of model accuracy, Significant increase of model speed, Annotated or not annotated datasets.
 - AccuracyAware Method: Controllable drop of model accuracy, Increase of model speed, Annotated datasets only. A "Max Accuracy Drop:" input field is set to 0.5%.
- Calibration Schemes:** A section with two radio button options:
 - Performance Preset: Uncompromising performance.
 - Mixed Preset: Tradeoff between accuracy and performance.
- Bottom Toolbar:** Buttons for "Optimize" (highlighted in dark blue) and "Cancel".

It need to be followed a series of manual steps to convert an FP32-Tiny YOLOv4 model into an INT8-precision model. Besides having to take care of each step and configuration file, we also had to ensure that each path and command was correct. With the DL workbench, you need not worry about any of these. Just provide the FP32 model, the dataset, and the desired accuracy. Within minutes, you will have an optimizer and a quantized model ready for use.

Quantization methods

5. Integration and Deployment

Lastly, the DL workbench also provides a final-deployable model which you can just click and export. The exported, deployable package will contain:

1. the optimized model
2. all the configuration files
3. the results of the various runs and experiments that you carry out

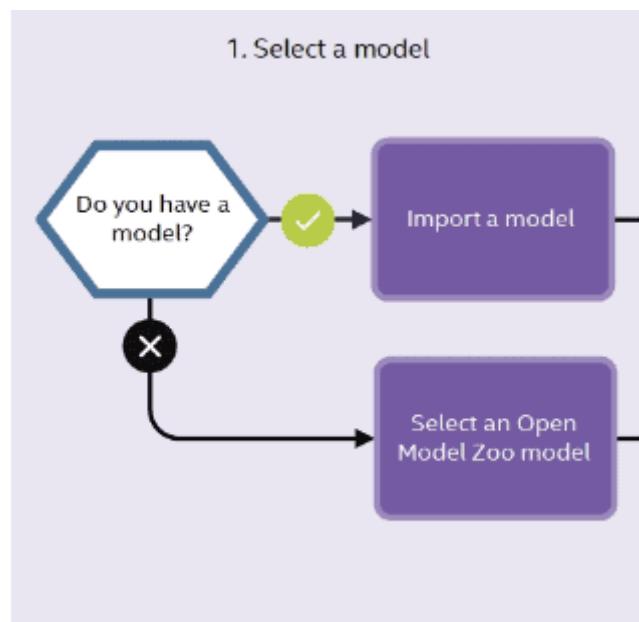
With all this information, you can easily zero in on the best possible way to deploy your model, and know exactly how it will perform. You can integrate it within an application, deploy it on the edge, or simply decide to run inference using the model.

Workflow of the Deep-Learning Workbench

The following stages presents the workflow of the Deep-Learning workbench, illustrating all the steps, starting from model selection right up to model deployment. The general workflow consists of 7 steps. Let's break these down into different components for greater clarity.

1. Model selection

You always start by selecting the model you want to optimize.

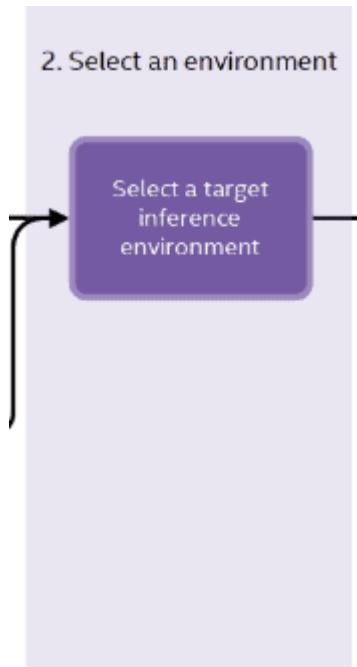


[Model Selection](#)

You are not constrained to choose a model from a specific framework, just ensure it comes from any of the OpenVINO-supported frameworks. You are even free to choose an OpenVINO-IR format model, provided you have already converted it using the Model Optimizer from any of the above frameworks.

2. Selecting the Target Environment

Next, select the target environment. This is important because the DL workbench will then optimize the model to run best on this particular hardware environment.



[Target Selection](#)

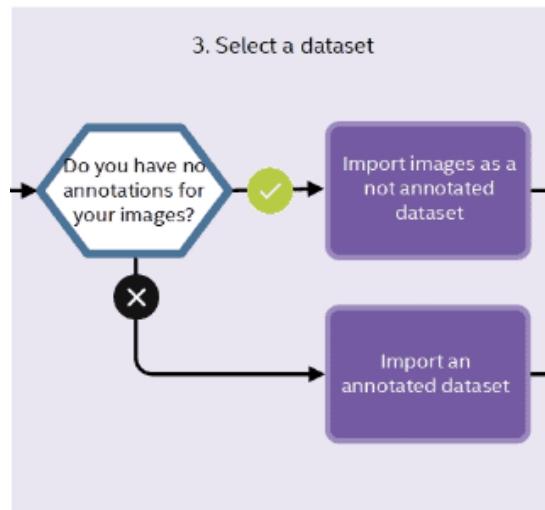
The target environment can be:

- a CPU
- the Intel Integrated GPU
- even VPUs like the Myriad X

In fact, you can even target a remote environment, which is not local to the system in which you are running the DL workbench.

3. Selecting the Dataset

Now that you've chosen the environment, select the dataset on which you want to run the evaluation and optimize the model.



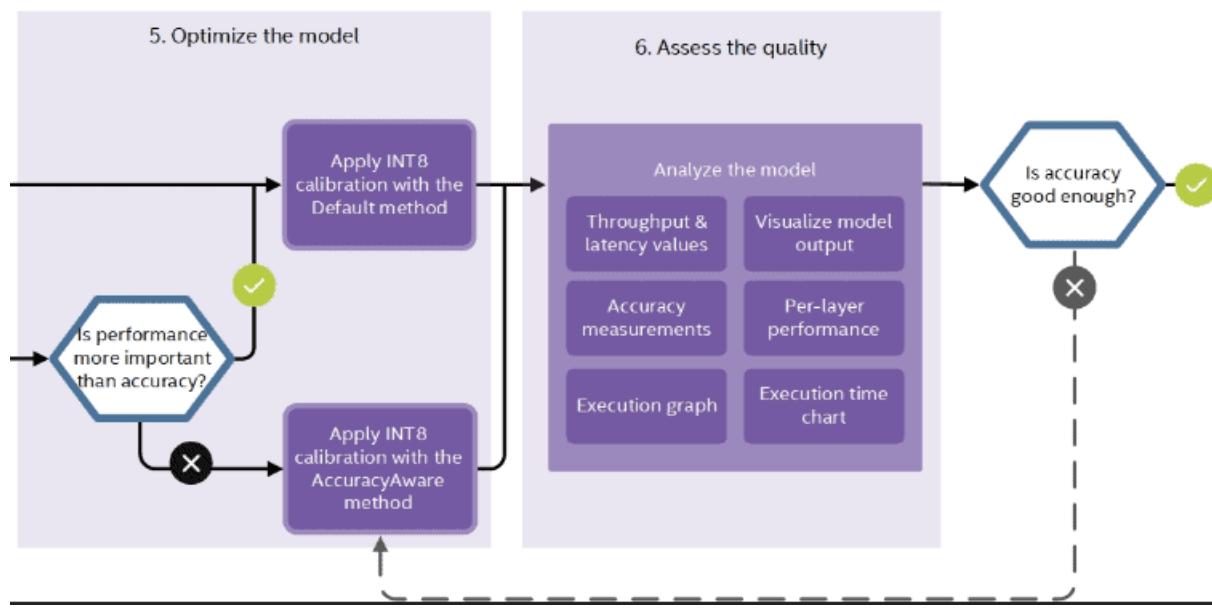
Dataset Selection

The DL workbench supports a number of datasets, including:

- Object-detection datasets like MS COCO, Pascal VOC
- Image-classification datasets like the ImageNet dataset
- Common Semantic Segmentation (CSS) dataset for semantic segmentation
- Common Super Resolution (CSR) dataset for super resolution, image inpainting and style transfer

4. Model Optimization

After model and dataset selection, optimize your model.



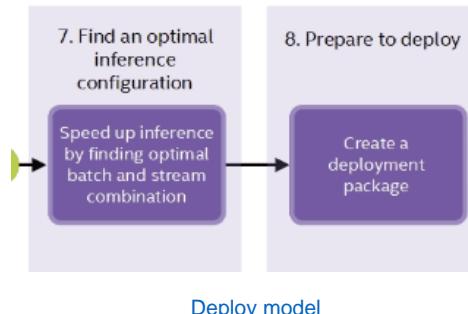
Model Optimization

Apply quantization to convert the FP32 models into INT8-precision models. Also,

assess the precision of your model, so that you know for sure it will perform well in the real-world.

5. Configure and Deploy

The final few steps will configure the model according to your requirements and deploy it.



Experiment with different batch sizes and compare the throughput across different runs. When you get the desired tradeoff between speed and throughput:

- create the deployment package
- download it
- deploy it on an edge device

4.7 Utilizing OpenVINO for inference at the edge

Installing Deep-Learning Workbench Through Docker

Only after you install the Deep-Learning workbench in your system can you use it for optimization. The easiest way to install it is through the Docker Hub. First, install the Docker Engine, that being a necessary prerequisite. **Please follow the instructions given here to install the Docker Engine on Ubuntu.** Now, follow these steps to install the DL workbench through Docker Hub on Linux (**Ubuntu 20.04**):

1. Go inside the workbench Folder in the OpenVINO Installation Directory

```
1 | cd /opt/intel/openvino_2021.3.394/deployment_tools/tools/workbench
```

2. Download the Workbench Starting Script

To start the DL workbench, you need to download the script. Give the following command to download the starting script for DL workbench, inside the current working directory.

```
1 | wget https://raw.githubusercontent.com/openvinotoolkit/workbench_aux/master/start_workbench.sh
```

3. Ensure the File is Executable

In many cases, execution permissions are disabled by default for security reasons. Execute the following command to ensure the script is executable.

```
1 | chmod +x start_workbench.sh
```

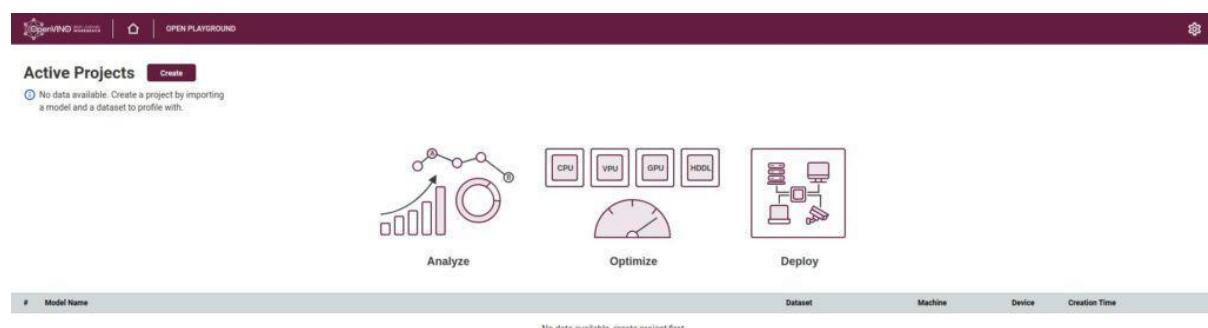
4. Run the DL Workbench

Finally, start the DL workbench through the below command execution terminal. `./start_workbench.sh`

You may need to wait for some time till the DL workbench is ready. After the script runs all the commands successfully, a prompt on the terminal informs you that the DL workbench is available on the local host at port 5665.

<http://127.0.0.1:5665/>

Open the link and you will see the following browser window:



The initial window of the DL Workbench.

With this, you have completed the installation of DL workbench.

Applying AccuracyAwareQuantization to Tiny YOLOv4, Using DL Workbench

Just follow these steps:

1. Start the DL Workbench

First, open the DL workbench, by following all the steps discussed in the installation section.



Active Projects

[Create](#)

 No data available. Create a project by importing a model and a dataset to profile with.

#	Model Name
---	------------

You will see the **Create** button at the top of the initial DL workbench window. Click on it.

2. Create Project

That takes you to the **Create Project** page, which should look similar to this:

[Start Page](#) / Create Project

Create Project

- ⓘ Select a model, dataset, and environment.
Then click Create to perform an inference.

Project Details

- ✗ Model: Selection Required
- ✗ Target: Selection Required
- ✗ Device: Selection Required
- ✗ Dataset: Selection Required

Model ^ Import

Model Name

- ⓘ To continue working, import a model.

Environment ^ Add Remote Target

ⓘ Target Platform

Target Name	Available Devices
Local Workstation	CPU

You need four things to successfully optimize a model, using the DL workbench:

1. The Deep-Learning model
2. The target environment
3. The target device or hardware
4. An evaluation dataset

Initially, all these options will be marked with red crosses (as in the above image), indicating that none of the requirements are met.

3. Import the Deep-Learning Model

Next, import the Deep-Learning model. For this example, we will be importing the Tiny-YOLOv4 FP32 model, which is already in the IR format. So no need to run it through the model optimizer. Simply, click on the **Import** button, which should take you to the following screen:

Import Model

1. Import

Open Model Zoo Original Model

Framework: OpenVINO IR

IR XML file: [Select](#) frozen_darknet_yolov4_model.xml

IR BIN file: [Select](#) frozen_darknet_yolov4_model.bin

Model name: [?](#) frozen_darknet_yolov4_mode

Import Model Cancel

You will need both:

- The .xml file containing the network topology
- The .bin file containing the model weights

Finally, click on the **Import Model** button. The model might take some time to upload to the DL-workbench environment.

4. Select the Target Environment and Hardware

Then you need to select the target environment and hardware.

Environment [Add Remote Target](#)

Target Platform

Base Platform: Intel Core

Processor Family	Processor Numbers	Available Devices	Platform Tag	Configuration Details
8th Generation Intel(R) Core(TM) i7 Processor	i7-8750H	CPU	Local Workstation	6 cores – 0.8-4.1 GHz

Device: Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz

Selecting the target environment and hardware.

For this example, we are using the **Local Environment** and the local CPU as our targets. When running on your local system, you should see your own CPU model and can choose accordingly.

5. Import the Evaluation Dataset

The final step before carrying out evaluation and optimization is to choose the evaluation dataset. If you just needed to carry out the evaluation, you wouldn't have to import any official dataset. Just creating a random dataset in the DL workbench would do. But here we need to quantize the model as well, so we will require a validation subset on which the quantized model can be evaluated.

[Start Page](#) / [Create Project](#) / Import Dataset

Import Validation Dataset

- ⓘ Import an dataset in one of the supported formats: [ImageNet](#), [Pascal VOC](#), [COCO](#), [Common Semantic Segmentation](#), [Common Super-resolution](#), [LFW](#), [VGGFaces2](#), or not annotated.

Select File: coco.zip

Imported Dataset Name: [?](#)

Here, we are importing the MS COCO 2017 validation dataset for evaluation purposes. This zip file contains 5000 validation images in total, along with their corresponding images. To download the dataset from the official website, click [here](#). After this, you should see a *green tick mark* across all the requirements.

Create Project

- ⓘ Select a model, dataset, and environment.
Then click Create to perform an inference.

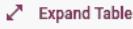
Project Details

- ✓ Model: frozen_darknet_yolov4_model
- ✓ Target: Local Workstation
- ✓ Device: Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz
- ✓ Dataset: coco

These are all the things you need to start the model evaluation/optimization. Next, click on the **Create** button on the browser window to start the process.

Initial Run Results

After this, run one initial evaluation on the entire validation dataset, using the full-precision model we selected above. Check out the results in the following image:

 Expand Table						
#	Stream	Batch	Throughput, FPS	Latency, ms	Last Status	Display Experiment
A	1	1	27.6	35.17	✓	<input checked="" type="checkbox"/>

The above results are from the i7 CPU machine, as mentioned in the previous section. Your results may vary depending on the hardware. For the FP32 model, the initial is giving 27.6 FPS on average and latency of 35.17 milliseconds. It will be interesting to compare these results with the quantized model, after applying AccuracyAwareQuantization.

6. Optimize Performance

To start the AccuracyAwareQuantization:

- Click on the **Perform** tab on the current screen,
- Then click the **Optimize Performance** button. This should let you select the INT8-optimization method.

Selecting the optimization method for quantization. Next, click the **Optimize** button.

7. Configure the Accuracy Settings and Select Dataset-Subset Size

Now you need to configure the accuracy settings. It is safe to leave the settings to default value initially. You should see a screen similar to this:

Configuring the accuracy settings for AccuracyAwareQuantization. For the optimized configuration settings:

- The metric is mAP (mean Average Precision)
- The dataset is COCO with 80 classes
- Both IOU (Intersection Over Union) and NMS (Non-Max Suppression) have 0.5 threshold
- The usage type is object detection, as we are using the Tiny-YOLOv4 model
- The *Model Type* is set to Tiny YOLOv2

As you must have noticed by now, instead of v3 or v4, the *Model Type* is Tiny YOLOv2. Because the recent Tiny YOLO versions are not-fully supported by OpenVINO, we are bound to choose the Tiny YOLOv2 as the Model Type. Though this will not cause issues while quantizing the model, it will fail to give us any accuracy results. So, we will be checking the accuracy manually, by using the COCO evaluator to run evaluation manually on the entire COCO-validation set. After completing the above settings, choose AccuracyAwareQuantization as the quantization method.

The screenshot shows the calibration tool's configuration interface. In the 'Optimization Methods' section, 'AccuracyAware Method' is selected over 'Default Method'. It includes a note about uncontrollable minor drops in accuracy and controllable drops with increased speed. A 'Max Accuracy Drop' input field is set to 0.5. In the 'Calibration Schemes' section, 'Performance Preset' is selected over 'Mixed Preset', described as providing uncompromising performance or a tradeoff between accuracy and performance. At the bottom, there are 'Optimize' and 'Cancel' buttons.

For this example, we have set the *Max Accuracy Drop* value to 0.5. This means, while doing INT8 calibration, if the accuracy drops below this specified threshold for any particular layer, then that layer will revert back to the original precision.

Also, take note of the dataset Subset Size. We are using just 20% of the 5000 images. So the calibration and evaluation will be done only on 1000 images. Selecting the whole dataset would have eaten up too much time, sometimes it takes hours to complete. Finally, you can click on the **Optimize** button.

8. Check the Results

Let the calibration tool run the AccuracyAwareQuantization. It could take some time, depending on your hardware (CPU).

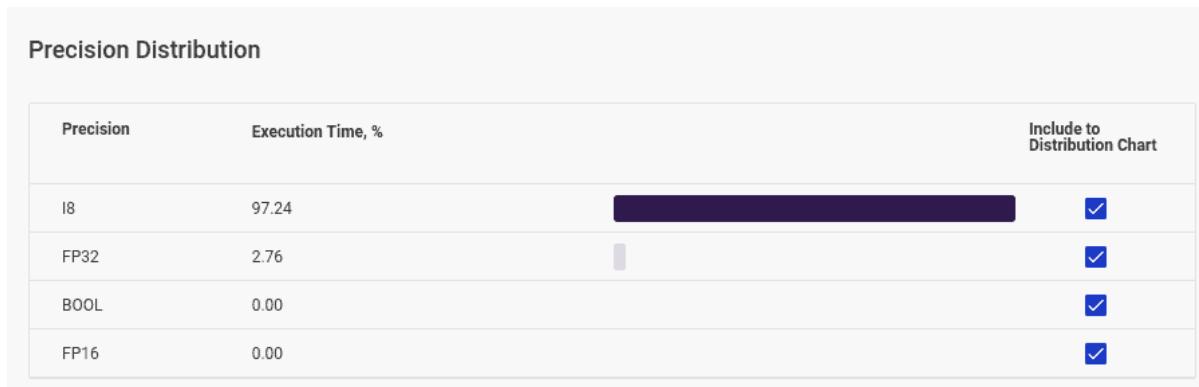
Experiment Summary							Expand Table
#	Stream	Batch	Throughput, FPS	Latency, ms	Last Status	Display Experiment	
A	1	1	60.48	17.66	✓	<input checked="" type="checkbox"/>	

We got:

- 60 FPS, in terms of throughput
- The latency dropped to 17.66 milliseconds, after the INT8 calibration

This is a huge improvement compared to the 27 FPS and 35 milliseconds latency seen in the case of the full precision model. We can also check the **Precision**

Distribution in our calibrated model to ensure the model was actually converted to the INT8 format.



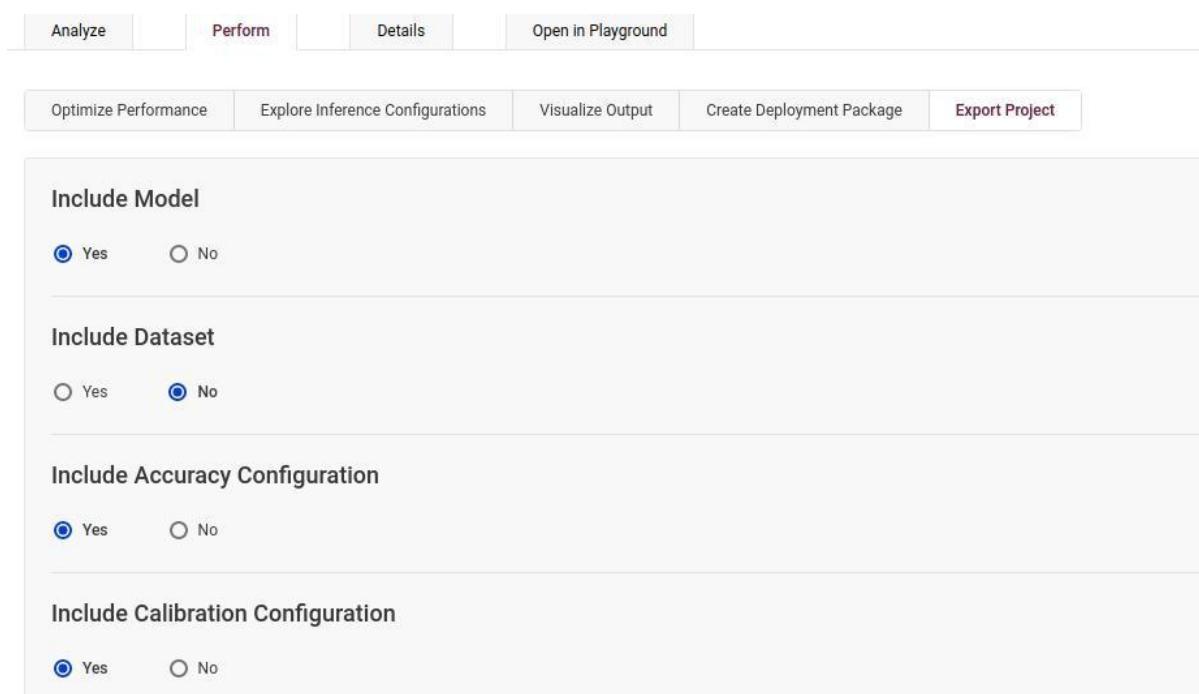
You can see that:

- More than 97% of the execution time was spent in the INT8 layers
- Around 2.7% of the time was spent in the FP32 layers

Okay, so this means most of the layers have been successfully converted to INT8 precision. Only a few layers reverted back to their original FP32-precision format, probably because they exceeded the accuracy-drop threshold of 0.5.

9. Export the Calibrated Model

The final step is to export your INT8-calibrated model so that you can run an inference with it.



Analyze Perform Details Open in Playground

Optimize Performance Explore Inference Configurations Visualize Output Create Deployment Package Export Project

Include Model
 Yes No

Include Dataset
 Yes No

Include Accuracy Configuration
 Yes No

Include Calibration Configuration
 Yes No

Go to the **Export Project** sub-tab on the Perform tab, choose the calibrated model, and click on the Export button. The downloaded file will contain the .xml as well as the .bin file.

10. Run Inference Using INT8-Calibrated Tiny-YOLOv4 Model

Now that we have obtained the INT8 models from the above steps, let's try running inference on the following video. To run inference, we use almost the same commands as in the previous posts in this series. The only difference being the path to the INT8-calibrated model. As the input video also remains the same, we can compare performance with the FP32 model.

```
1   python object_detection_demo.py --model frozen_darknet_yolov4_model.xml  
-at yolo -i video_1.mp4 -t 0.5 -o acc_aw_int8_default.mp4
```

For the above run:

- the average FPS is 30.3
- latency is 27.3 milliseconds

Now, check out the following video output:



The above results are really interesting. Not only are we getting throughput very similar to the Default Quantized INT8 model, but also the detections look exactly the same as the FP32 model. This means **we are getting good speed and accuracy at the same time.**

4.8 Edge Computing Using OpenVINO and Raspberry-Pi

Install OpenVINO on R-Pi

1. Go to Downloads folder
2. cd Downloads

3. Install the OpenVINO (2020.4) toolkit using wget command

```
wget --no-check-certificate
https://storage.openvinotoolkit.org/repositories/openvino/packages/2020.4/I_openvino_toolkit_runtime_raspbian_p_2020.4.287.tgz
```

By default, the package file is saved as
I_openvino_toolkit_runtime_raspbian_p_<version>.tgz.

Note: If the file isn't run by wget, use go to this link and download from GUI
Link: <https://storage.openvinotoolkit.org/repositories/openvino/packages/2020.4/>

4. Create an installation folder.

```
sudo mkdir -p /opt/intel/openvino_2021
```

5. Unpack the archive:

```
sudo tar -xf I_openvino_toolkit_runtime_raspbian_p_2020.4.287.tgz --strip 1 -C /opt/intel/openvino_2021
```

6. Set the Environment Variables

You must update several environment variables before you can compile and run OpenVINO toolkit applications. Run the following script to temporarily set the environment variables:

```
source /opt/intel/openvino_2021/bin/setupvars.sh
```

(Optional) The OpenVINO environment variables are removed when you close the shell. As an option, you can permanently set the environment variables as follows:

```
echo "source /opt/intel/openvino_2021/bin/setupvars.sh" >> ~/.bashrc
```

7. Add USB Rules for an Intel® Neural Compute Stick 2 device

This task applies only if you have an Intel® Neural Compute Stick 2 device.

Add the current Linux user to the users group:

```
sudo usermod -a -G users "$(whoami)"
```

To perform inference on the Intel® Neural Compute Stick 2, install the USB rules running the install_NCS_udev_rules.sh script:

```
sh /opt/intel/openvino_2021/install_dependencies/install_NCS_udev_rules.sh
```

8. Log out and log in for it to take effect. (Reboot)

```
sudo reboot
```

To test your change, open a new terminal. You will see the following:

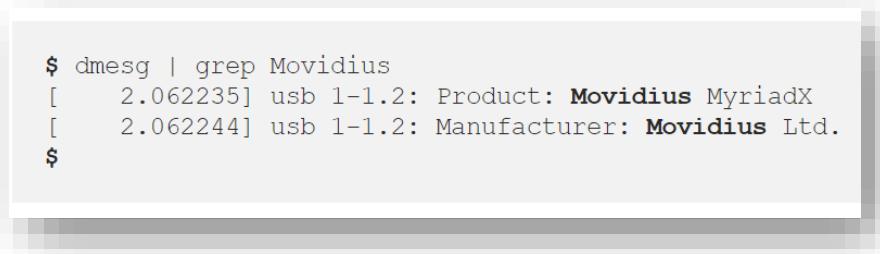
```
[setupvars.sh] OpenVINO environment initialized
```

9. Plug in your Intel® Neural Compute Stick 2 in USB 2.0

10. Checking for Movidius Device (VPU)

```
dmesg | grep Movidius
```

You will get the following output indicating that we have Movidius MyriadX connected to Raspberry Pi



```
$ dmesg | grep Movidius
[    2.062235] usb 1-1.2: Product: Movidius MyriadX
[    2.062244] usb 1-1.2: Manufacturer: Movidius Ltd.
$
```

Face Detection using OpenVINO on R-Pi

Project 01: Face Detection

1. Clone the github repository into local system

```
git clone https://github.com/ameer-aiml/face-detect-ov-rpi
```

This repository contains both xml and bin file of Face Detection model.

2. Go to Face Detection folder

```
cd face-detect-ov-rpi /Face Detection
```

Note: Add a single before and after Face Detection wording if not recognized

3. Run the python file (it should be python3)

```
python3 face_detection.py
```

Method 2: If git clone isn't working

1. Create a folder named face
`mkdir face && cd face`
2. **Download xml and bin file using wget for Face Detection Retail from web browser and move it to face folder**

Link : https://download.01.org/opencv/2019/open_model_zoo/R1/models_bin/face-detection-retail-0004/FP16/

NOTE: Take only FP16 because VPU works only on FP16 .

3. Copy and paste the face_detection.py file in face folder of Raspberry Pi
4. Run python file using python3 command
`python3 face_detection.py`

Errors which might be faced:

1. cv2.error: OpenCV(4.4.0-opencvino)
..../opencv/modules/dnn/src/ie_ngraph.cpp:638: error: (-2:Unspecified error)
Failed to initialize Inference Engine backend (device = MYRIAD): Can not init Myriad device: NC_ERROR in function 'initPlugin'

Either VPU isn't working or you are using VPU in USB 3.0 (Connect it to USB 2.0)

Project 02: Face, Age & Gender Detection using OpenVINO & R-PI

1. Clone the github repository into local system

```
git clone https://github.com/ameer-aiml/age-gender-openvino-rpi
```

2. Go to Age Gender Detection folder

```
cd age-gender-openvino-rpi/AgeGender
```

This repository contains 3 different models.

One for Face Detection, another for Age and last one for Gender Detection.

Both the files of Face Detection is available. The .caffemodel files of Age and Gender are missing.

3. Download the Age and Gender Caffe model from this site using the command
`wget`
https://www.dropbox.com/s/iyy483wz7ztr9gh/gender_net.caffemodel

```
wget https://www.dropbox.com/s/xfb20y596869vbb/age_net.caffemodel
```

4. Run the python file (it should be python3)

```
python3 face_detection.py
```

Errors which might be faced:

5. **cv2.error:** OpenCV(4.4.0-opencvino)
..../opencv/modules/dnn/src/ie_ngraph.cpp:638: error: (-2:Unspecified error) Failed to initialize Inference Engine backend (device = MYRIAD): Can not init Myriad device: NC_ERROR in function 'initPlugin'

Either VPU isn't working or you are using VPU in USB 3.0 (Connect it to USB 2.0)

Challenge to try out:

Convert all the above mentioned 6 files into xml and bin file respectively using model optimizer for unified way of model building.

If you want to use your model for inference, the model must be converted to the .bin and .xml Intermediate Representation (IR) files that are used as input by Inference Engine. OpenVINO™ toolkit support on Raspberry Pi only includes the Inference Engine module of the Intel® Distribution of OpenVINO™ toolkit. The Model Optimizer is not supported on this platform.

Reference

1. <https://www.edureka.co/blog/variables-and-data-types-in-python/>
2. https://www.tutorialspoint.com/python/python_variable_types.htm
3. <https://data-flair.training/blogs/python-variables-and-data-types/>
4. <https://www.programiz.com/python-programming/variables-datatypes>
5. <https://www.tutorialspoint.com/numpy>
6. <https://numpy.org/>
7. https://www.tutorialspoint.com/python_data_science
8. <https://www.geeksforgeeks.org/generating-random-number-list-in-python/>
9. [https://www.w3schools.com/python\(numpy_random.asp](https://www.w3schools.com/python(numpy_random.asp)
10. <https://www.geeksforgeeks.org/numpy-asscalar-in-python/>
11. [https://data-flair.training/blogs\(numpy-statistical-functions/](https://data-flair.training/blogs(numpy-statistical-functions/)
12. <https://www.w3schools.in/python-tutorial/decision-making/>
13. <https://www.geeksforgeeks.org/python-broadcasting-with-numpy-arrays/>
14. <https://realpython.com/>
15. https://www.tutorialspoint.com/python/python_variable_types.htm
16. https://www.tutorialspoint.com/python/python_variable_types.htm
17. <https://www.edureka.co/blog/variables-and-data-types-in-python/>
18. <https://data-flair.training/blogs/python-variables-and-data-types/>
19. <https://www.programiz.com/python-programming/variables-datatypes>
20. [Jupyter Notebook: An Introduction – Real Python](#)
21. <https://jakevdp.github.io/PythonDataScienceHandbook/04.12-three-dimensional-plotting.html>
22. [https://www.tutorialspoint.com\(numpy/numpy_matplotlib.htm](https://www.tutorialspoint.com(numpy/numpy_matplotlib.htm)
23. <https://www.edureka.co/blog/python-numpy-tutorial/>
24. <https://pypi.org/>
25. <https://datatofish.com/install-package-python-using-pip/>
26. <https://packaging.python.org/>
27. https://www.tutorialspoint.com/operating_system/os_linux.html
28. <https://buildmedia.readthedocs.org/media/pdf/lym/latest/lym.pdf>
29. <https://phoenixnap.com/kb/linux-commands-cheat-sheet>
30. <https://www.mongodb.com/docs/manual/core/views/>
31. <https://www.geeksforgeeks.org/mongodb-an-introduction/>
32. <https://www.mongodb.com/developer/how-to/SQL-to-Aggregation-Pipeline/>
33. <https://www.mongodb.com/docs/compass/current/import-export/#export-data-from-a-collection>
34. <https://www.broadbandsearch.net/blog/internet-statistics>
35. <https://www.cloudcredential.org/blog/knowledge-byte-building-blocks-of-iot-architecture/>
36. <https://www.engineersgarage.com/iot-building-blocks-and-architecture-iot-part-2/>
37. <https://www.c-sharpcorner.com/UploadFile/f88748/internet-of-things-part-2/>
38. <https://www.oracle.com/in/internet-of-things/what-is-iot/>
39. <https://www.simplilearn.com/iot-applications-article>

40. <https://electronicscoach.com/electronic-components.html>
- 41.
42. https://en.wikipedia.org/wiki/Electronic_component#:~:text=An%20electronic%20component%20is%20any,electrons%20or%20their%20associated%20fields.\
43. <https://tesckt.com/transistor-application-circuits-and-it-application-in-daily-life/>
44. <https://www.elprocus.com/buzzer-working-applications/>
45. <https://www.vedantu.com/iit-jee/basic-logic-gates>
46. <https://www.monolithicpower.com/en/analog-vs-digital-signal>
47. <https://byjus.com/physics/pulse-width-modulation/>
48. <https://dengarden.com/home-improvement/Using-a-Multimeter>
49. <https://circuitdigest.com/tutorial/different-types-of-sensors-and-their-working>
50. <https://robu.in/wp-content/uploads/2019/09/Grove-Rotary-Angle-Sensor-User-Manual.pdf>
51. <https://www.elprocus.com/robot-sensor/>
52. <https://robu.in/ultrasonic-sensor-working-principle/>
53. <https://www.watelectronics.com/lcd-16x2/>
54. <https://www.automate.org/blogs/what-kinds-of-applications-are-best-for-stepper-motors>
55. [https://circuitdigest.com/article/servo-motor-working-and-basics#:~:text=Servo%20motor%20works%20on%20PWM,\(potentiometer\)%20and%20some%20gears.](https://circuitdigest.com/article/servo-motor-working-and-basics#:~:text=Servo%20motor%20works%20on%20PWM,(potentiometer)%20and%20some%20gears.)
56. <https://www.raspberrypi.com/documentation/computers/getting-started.html>
57. [GrovePi+Getting-started-Guide.pdf \(storage.googleapis.com\)](https://GrovePi+Getting-started-Guide.pdf (storage.googleapis.com))
58. [Seeed Studio GrovePi Plus - Seeed Wiki \(seeedstudio.com\)](https://Seeed Studio GrovePi Plus - Seeed Wiki (seeedstudio.com))
59. [Setting Up The Software for GrovePi \(seeedstudio.com\)](https://Seeed Studio Setting Up The Software for GrovePi (seeedstudio.com))
60. [Grove - Ultrasonic Ranger - Seeed Wiki \(seeedstudio.com\)](https://Seeed Studio Grove - Ultrasonic Ranger - Seeed Wiki (seeedstudio.com))
61. [Data Communications and Networking](#)
62. <https://www.quora.com/How-does-the-Internet-of-Things-work-in-a-LAN-network.>
63. <https://www.f5.com/services/resources/glossary/multi-homing>
64. <https://behrtech.com/blog/wireless-iot-protocols-breaking-down-the-network-stack/>
65. <https://www.iotcommunications.com/blog/types-of-iot-networks/>
66. <https://www.lairdconnect.com/resources/blog/all-about-iot>
67. <https://www.guru99.com/difference-ipv4-vs-ipv6.html>
68. <https://www.geeksforgeeks.org/python-gui-tkinter/#:~:text=Python%20offers%20multiple%20options%20for,to%20create%20the%20GUI%20applications.>
69. <https://medium.com/teamresellerclub/the-6-best-python-gui-frameworks-for-developers-7a3f1a41ac73>
70. <https://www.computerhope.com/jargon/g/gui.htm>
71. <https://www.cs.mcgill.ca/~hv/classes/MS/TkinterPres/#Overview>
72. <https://www.geeksforgeeks.org/introduction-to-tkinter/>
73. <https://www.studytonight.com/tkinter/python-tkinter-geometry-manager>

74. <https://www.javatpoint.com/logistic-regression-in-machine-learning>
75. <https://www.datasciencecentral.com/profiles/blogs/understanding-the-applications-of-probability-in-machine-learning>
76. <https://www.allerin.com/blog/how-to-fine-tune-your-artificial-intelligence-algorithms>
77. <https://www.mygreatlearning.com/blog/gridsearchcv/>
78. <https://www.kdnuggets.com/2020/05/hyperparameter-optimization-machine-learning-models.html>
79. <https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm>
80. <https://towardsdatascience.com/support-vector-machines-svm-c9ef22815589>
81. <https://medium.com/analytics-vidhya/role-of-distance-metrics-in-machine-learning-e43391a6bf2e>
82. <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>
83. <https://towardsdatascience.com/basic-probability-theory-and-statistics-3105ab637213>
84. <https://www.javatpoint.com/machine-learning-naive-bayes-classifier>
85. Andreas C. Müller and Sarah Guido , Introduction to Machine learning with Python , O'reilly , October 2016.
86. <https://www.guru99.com/unsupervised-machine-learning.html#:~:text=Unsupervised%20Learning%20is%20a%20machine,deals%20with%20the%20unlabelled%20data.>
87. <https://towardsdatascience.com/unsupervised-learning-and-data-clustering-eeecb78b422a>
88. <https://www.geeksforgeeks.org/clustering-in-machine-learning/#:~:text=Clustering%20is%20the%20task%20of,data%20points%20in%20other%20groups.>
89. <https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1>
90. Chire, CC BY-SA 4.0 <<https://creativecommons.org/licenses/by-sa/4.0>>, via Wikimedia Commons\
91. <https://www.javatpoint.com/deep-learning-algorithms>
92. <https://www.guru99.com/deep-learning-tutorial.html>
93. https://www.tutorialspoint.com/python_deep_learning/index.htm
94. <https://www.datacamp.com/tutorial/tutorial-deep-learning-tutorial>
95. <https://www.javatpoint.com/gradient-descent-in-machine-learning>
96. <https://neptune.ai/blog/performance-metrics-in-machine-learning-complete-guide>
97. <https://www.section.io/engineering-education/understanding-loss-functions-in-machine-learning/>
98. <https://analyticsindiamag.com/a-beginners-guide-to-cross-entropy-in-machine-learning/#:~:text=The%20average%20number%20of%20bits,of%20actual%20and%20expected%20results.>
99. <https://www.v7labs.com/blog/overfitting#:~:text=It%20is%20a%20common%20pitfall,the%20noise%20and%20random%20fluctuations>
100. <https://www.javatpoint.com/regularization-in-machine-learning>

101. <https://www.simplilearn.com/tutorials/deep-learning-tutorial/tensorflow-2>
102. <https://www.cs.ryerson.ca/~aharley/vis/>
103. <https://setosa.io/ev/image-kernels/>
104. <https://towardsdatascience.com/understand-transposed-convolutions-and-build-your-own-transposed-convolution-layer-from-scratch-4f5d97b2967>
105. <https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorithm>
106. <https://medium.com/voice-tech-podcast/text-classification-using-cnn-9ade8155dfb9>
107. ["Release Notes for Intel Distribution of OpenVINO toolkit 2022"](#). March 2022.
108. ["OpenVINO Toolkit: Welcome to OpenVINO"](#).
109. ["Introduction to Intel Deep Learning Deployment Toolkit – OpenVINO Toolkit"](#).
110. Wilbur, Marcia. ["Use the Model Downloader and Model Optimizer for the Intel® Distribution of OpenVINO™ Toolkit on Raspberry Pi"](#).
111. Agrawal, Vasu (2019). [Ground Up Design of a Multi-modal Object Detection System](#) (PDF) (MSc). Carnegie Mellon University Pittsburgh, PA. [Archived](#) (PDF) from the original on 26 January 2020.
112. Driaba, Alexander; Gordeev, Aleksei; Klyachin, Vladimir (2019). ["Recognition of Various Objects from a Certain Categorical Set in Real Time Using Deep Convolutional Neural Networks"](#) (PDF). Institute of Mathematics and Informational Technologies Volgograd State University. [Archived](#) (PDF) from the original on 26 January 2020. Retrieved 26 January 2020.
113. Nanjappa, Ashwin (31 May 2019). Caffe2 Quick Start Guide: Modular and scalable deep learning made easy. Packt. pp. 91–98. [ISBN 978-1789137750](#).