

Machine Learning With Python

- Section 1 : Introduction to Machine Learning
- Section 2 : Implementing ML Algorithms in Python
- Section 3 : Simple Linear Regression
- Section 4 : Multiple Linear Regression
- Section 5 : Classification Algorithms: K-Nearest Neighbors
- Section 6 : Classification Algorithms: Decision Tree
- Section 7 : Classification Algorithms: Logistic regression
- Section 8 : Clustering
- Section 9 : Recommender System
- Section 10: Conclusion

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

What is Machine Learning?

01

Application of machine learning

02

Machine learning Methods

03

What is Supervised learning?

04

What is Unsupervised learning?

05

Supervised learning vs Unsupervised learning

06



Introduction to Machine Learning

What is Machine Learning?

- Machine learning is a form of artificial intelligence, where the machine are taught to do tasks without being explicitly programmed.

Characteristics of Machine Learning:

1. Learns relationship between data.
2. It can predict events for other data.
3. Improve performance with respect to experience.



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

What is Machine Learning?

01

Application of machine learning

02

Machine learning Methods

03

What is Supervised learning?

04

What is Unsupervised learning?

05

Supervised learning vs Unsupervised learning

06



Introduction to Machine Learning

Applications

★ Medical Diagnosis

★ Self-Driving Cars

★ Spam email Filtering

★ Stock market Trading

★ Automatic Language Translation

★ Product Recommendations

★ Traffic Prediction

★ Image Recognition

★ Speech Recognition

And many more



Introduction to Machine Learning

Real World Applications

- ★ Detect cancer and brain tumours
 - ★ Tesla self-driving car
 - ★ Gmail spam emails filtering
 - ★ Google translator
 - ★ Netflix, YouTube, Amazon, and other such companies
- Google maps, face detection to unlock a device, Siri, Cortana, Alexa all these technologies use machine learning in the background

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

What is Machine Learning?

01

Application of machine learning

02

Machine learning Methods

03

What is Supervised learning?

04

05

Supervised learning vs Unsupervised learning

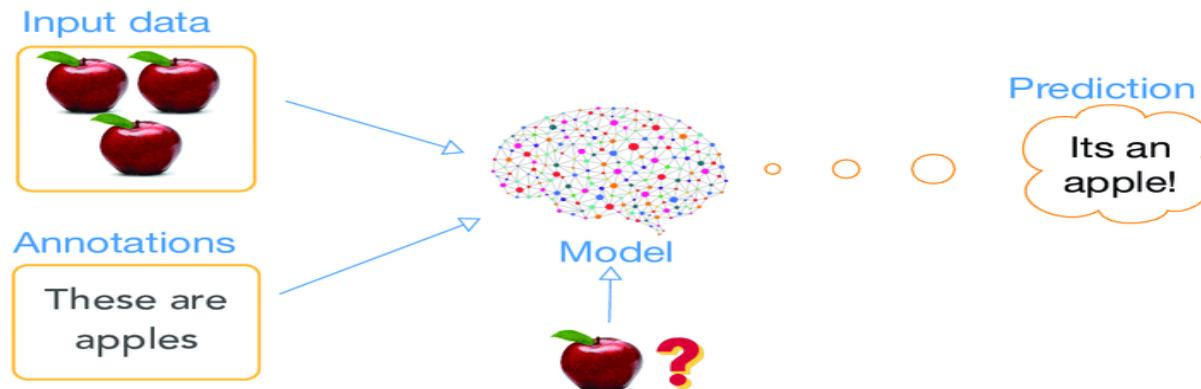
06



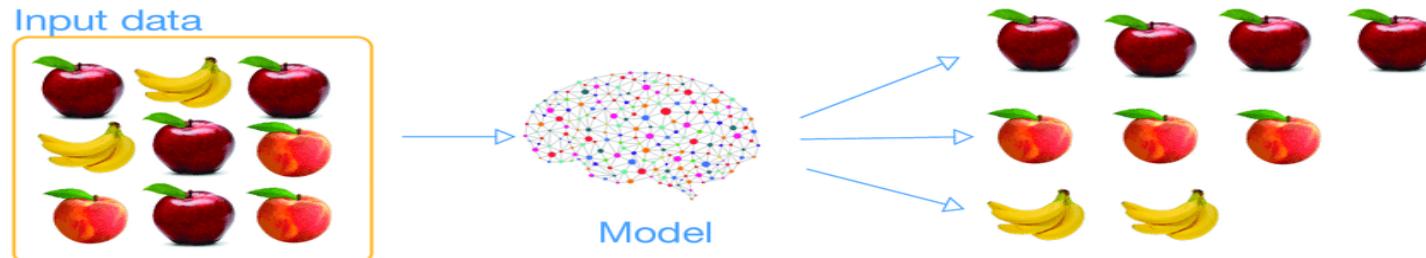
Machine Learning

Machine learning Methods

supervised learning



unsupervised learning



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

What is Machine Learning?

01

Application of machine learning

02

Machine learning Methods

03

What is Supervised learning?

04

What is Unsupervised learning?

05

Supervised learning vs Unsupervised learning

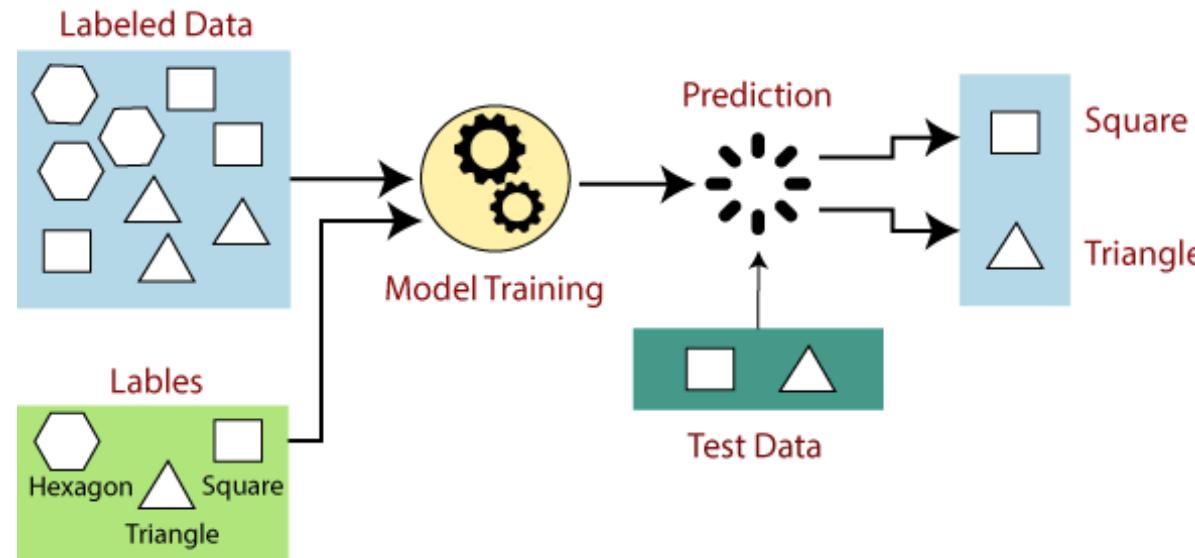
06



Machine Learning

What is Supervised learning?

The term "supervise" refers to the act of watching and directing the completion of a work, project, or operation. It uses labelled training data and a collection of training sample to estimate a function.



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

What is Machine Learning?

01

Application of machine learning

02

Machine learning Methods

03

What is Supervised learning?

04

What is Unsupervised learning?

05

Supervised learning vs Unsupervised learning

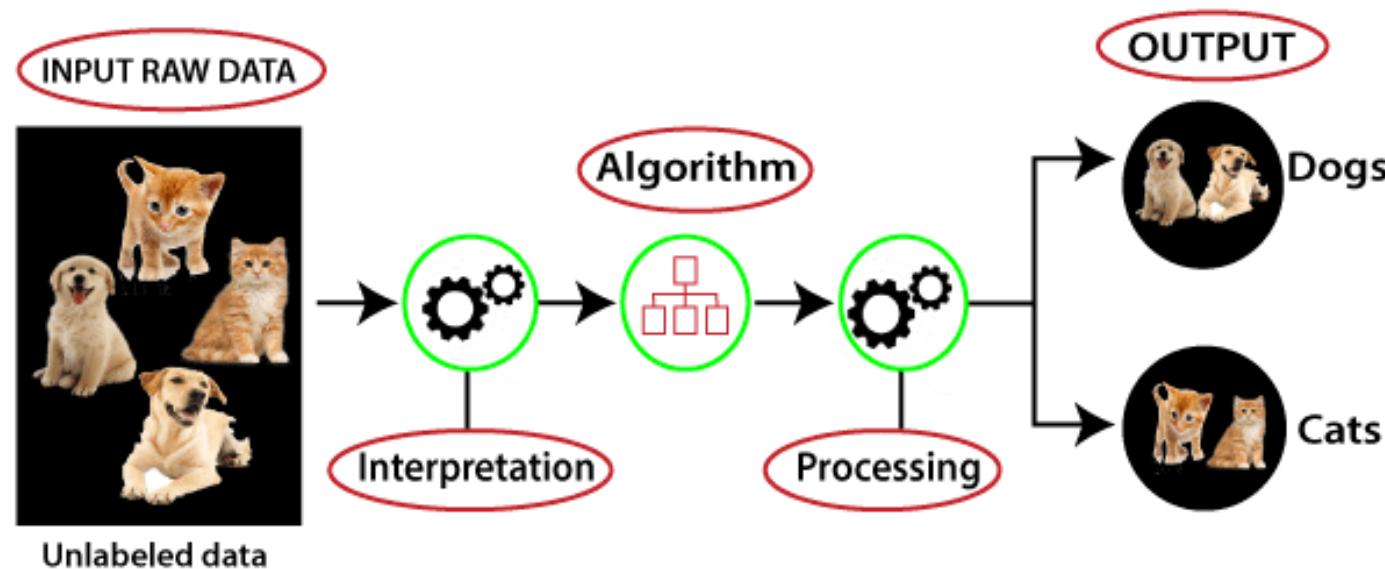
06



Machine Learning

What is Unsupervised learning?

Unsupervised learning is a kind of machine learning in which the training data is supplied to the system without any pre-assigned labels or values.



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

What is Machine Learning?

01

Application of machine learning

02

Machine learning Methods

03

What is Supervised learning?

04

What is Unsupervised learning?

05

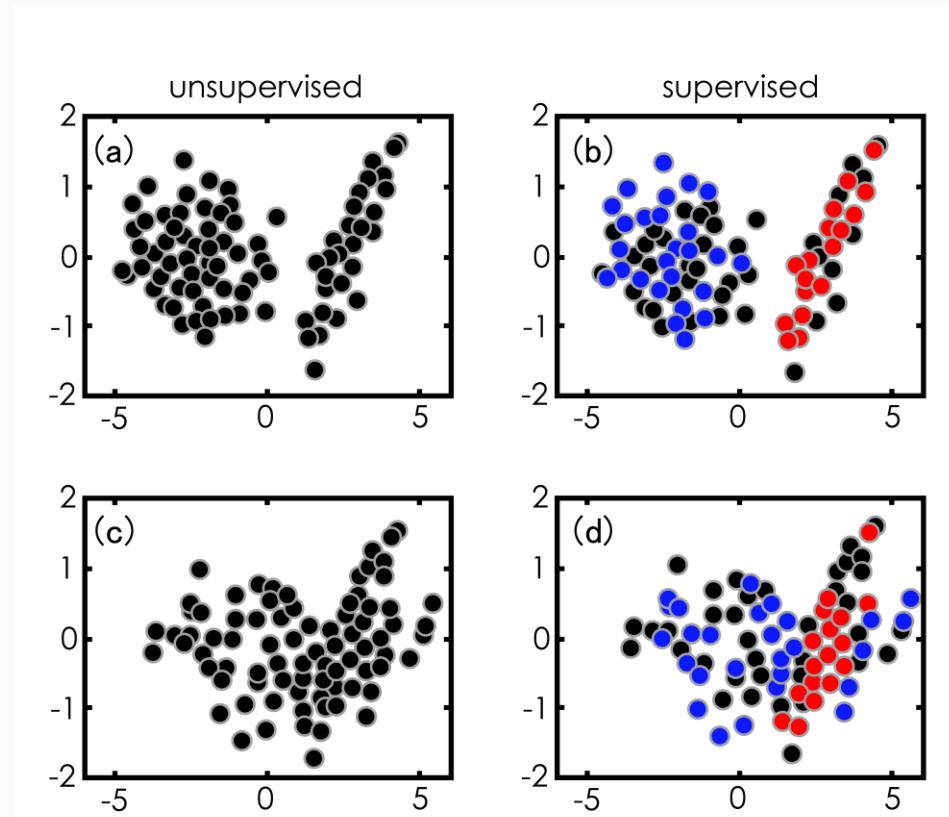
Supervised learning vs Unsupervised learning

06



Machine Learning

Supervised learning vs Unsupervised learning





Quiz Time

Which of the following is *TRUE* about unsupervised machine learning?

1. *unsupervised learning comprises algorithms with no pre-existing outcomes.*
2. *learning algorithms with no control over quality of their predictions.*
3. *a semi-autonomous machine learning where researchers control some parts of the modelling process.*
4. *a fully autonomous machine learning with no human interference*



Quiz Time

Which of the following is *TRUE* about unsupervised machine learning?

1. *unsupervised learning comprises algorithms with no pre-existing outcomes.*
2. *learning algorithms with no control over quality of their predictions.*
3. *a semi-autonomous machine learning where researchers control some parts of the modelling process.*
4. *a fully autonomous machine learning with no human interference*

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction

01

Python libraries for Machine Learning

02

Setting up Python

03

What is Jupyter?

04

Anaconda Installation for Windows OS

05

Anaconda Installation for Mac OS

06

Anaconda Installation for Ubuntu OS

07

Implementing Python in Jupyter

08

Managing Directories in Jupyter Notebook

09



Python Language for Machine Learning

Python

Python is a widely used and efficient programming language that has lately become the language of choice for data scientists.



Libraries of Python

Following are the few libraries of Python.



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction

01

Python libraries for Machine Learning

02

Setting up Python

03

What is Jupyter?

04

Anaconda Installation for Windows OS

05

Anaconda Installation for Mac OS

06

Anaconda Installation for Ubuntu OS

07

Implementing Python in Jupyter

08

Managing Directories in Jupyter Notebook

09



Python Language for Machine Learning

Libraries of Python



NumPy



SciPy



Numpy helps to deal with large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Scipy used for scientific computing and technical computing. SciPy contains modules for optimization, linear algebra, integration, special functions, signal and image processing, and other tasks common in science and engineering.

Matplotlib is a plotting library, and it is easy to use with Numpy because it's numeric is integrated with Numpy.



Python Language for Machine Learning

Libraries of Python



Pandas is designed for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

Scikit-learn (also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms. And it is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction

01

Python libraries for Machine Learning

02

Setting up Python

03

What is Jupyter?

04

Anaconda Installation for Windows OS

05

Anaconda Installation for Mac OS

06

Anaconda Installation for Ubuntu OS

07

Implementing Python in Jupyter

08

Managing Directories in Jupyter Notebook

09



Introduction to Python

What is Python?

- Python is one of the most popular programming languages, created by **Guido van Rossum** and released in 1991.
- Python is an **interpreted** programming language as opposed to a compiled programming language.



Why python?

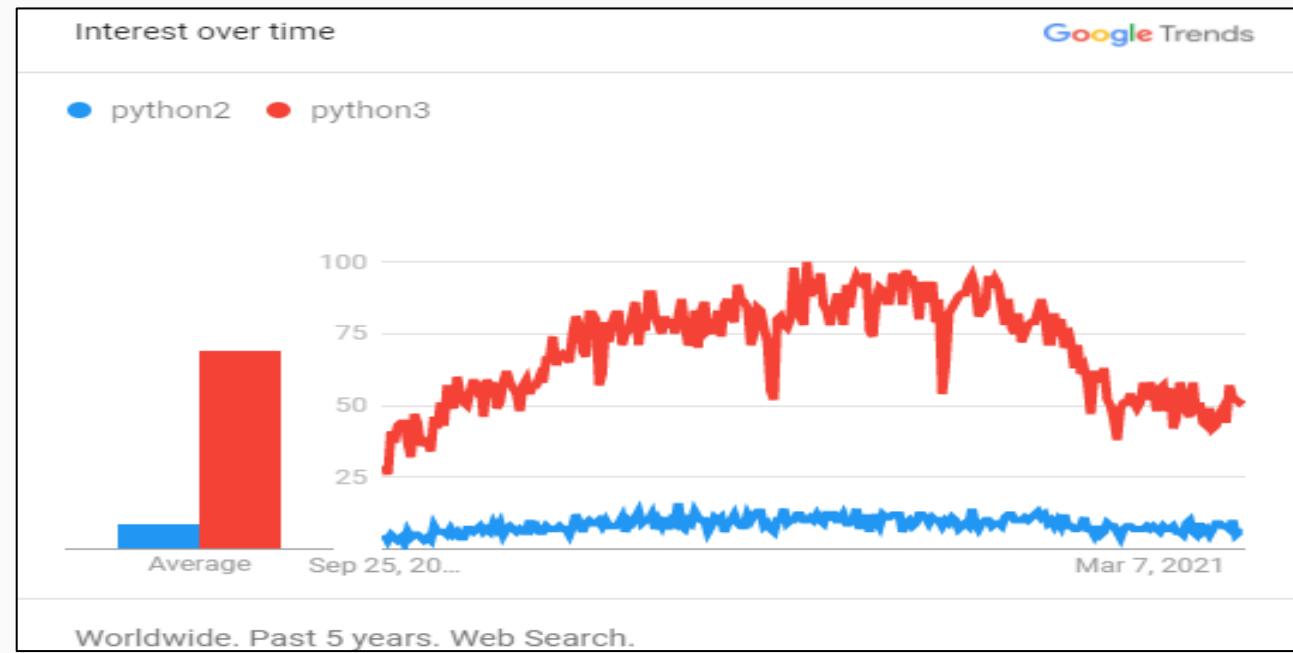
- Python is a beginner-friendly language due to its simplified syntax which is close to natural language.
- It has been designed to write clear, logical code for small and large-scale projects.
- It is used extensively in the industry for software development, data science, artificial intelligence, and machine learning.



Setting Up Python

Python3 vs Python2

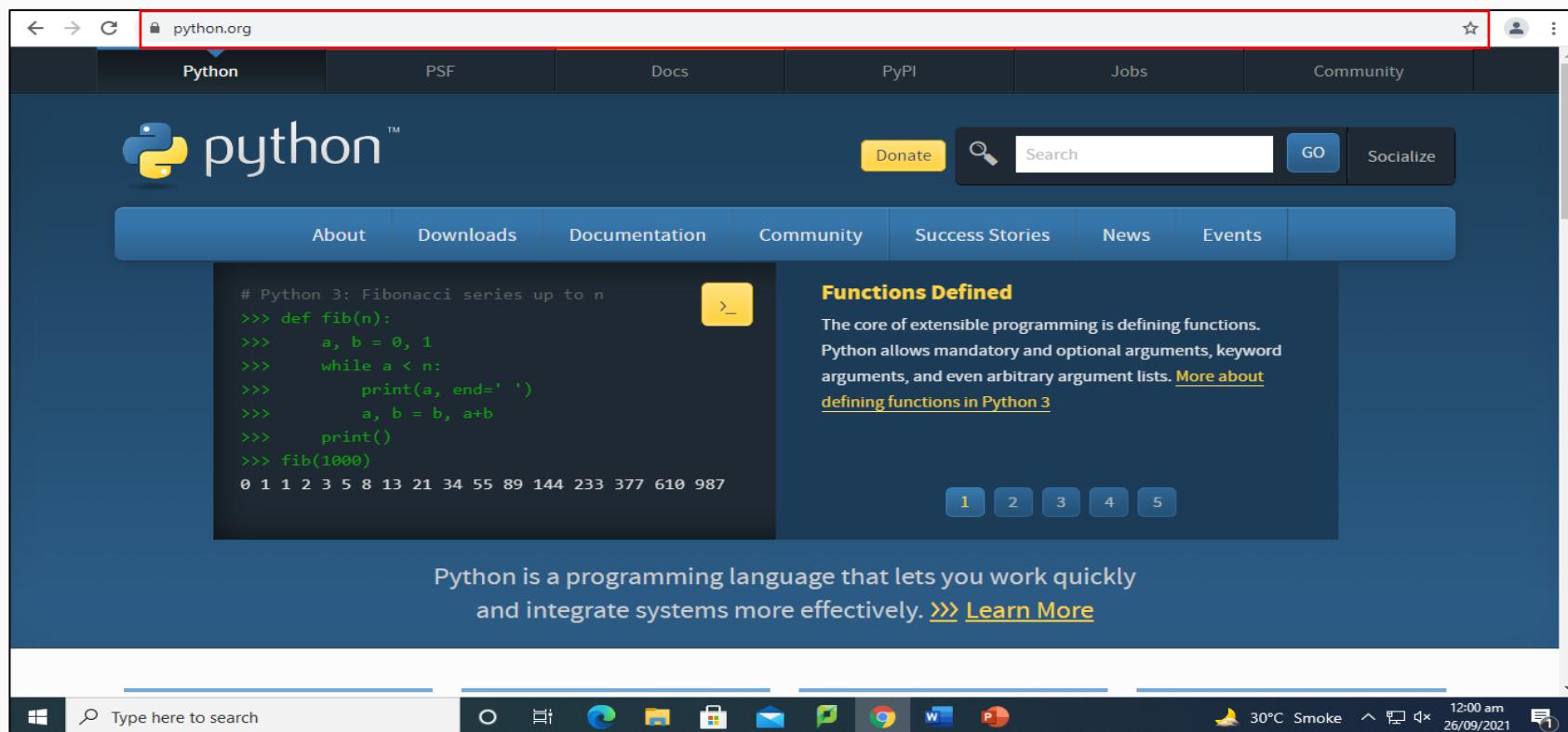
- Python2 is an outdated version of python. Python3 is the current version and is in-demand.
- In this course, we will be learning Python3 .





Setting Up Python – Windows (1/8)

- Visit: <https://www.python.org/>





Setting Up Python – Windows (2/8)

- Click on Downloads.

The screenshot shows the Python.org homepage. The navigation bar includes links for Python, PSF, Docs, PyPI, Jobs, and Community. Below the navigation bar is the Python logo and a search bar with a 'GO' button. The main content area has tabs for About, Downloads, Documentation, Community, Success Stories, News, and Events. The 'Downloads' tab is highlighted with a red box. On the left, there's a code snippet demonstrating Python's syntax for calculating Fibonacci numbers. To the right of the code, a sidebar titled 'Download for Windows' offers Python 3.9.7, along with links for All releases, Source code, Windows, macOS, Other Platforms, License, and Alternative Implementations. A note states that Python 3.9+ cannot be used on Windows 7 or earlier. The footer contains a brief description of Python and a link to learn more, followed by a Windows taskbar at the bottom.

python.org

Python PSF Docs PyPI Jobs Community

Donate Search GO Socialize

python™

About Downloads Documentation Community Success Stories News Events

```
# Python 3: Fib
>>> def fib(n):
>>>     a, b =
>>>     while a
>>>         pri
>>>         a,
>>>         print()
>>>     fib(1000)
0 1 1 2 3 5 8 1
```

All releases
Source code
Windows
macOS
Other Platforms
License
Alternative Implementations

Download for Windows

Python 3.9.7

Note that Python 3.9+ cannot be used on Windows 7 or earlier.

Not the OS you are looking for? Python can be used on many operating systems and environments.

[View the full list of downloads.](#)

Python is a programming language that lets you work quickly and integrate systems more effectively. [» Learn More](#)

https://www.python.org/downloads/

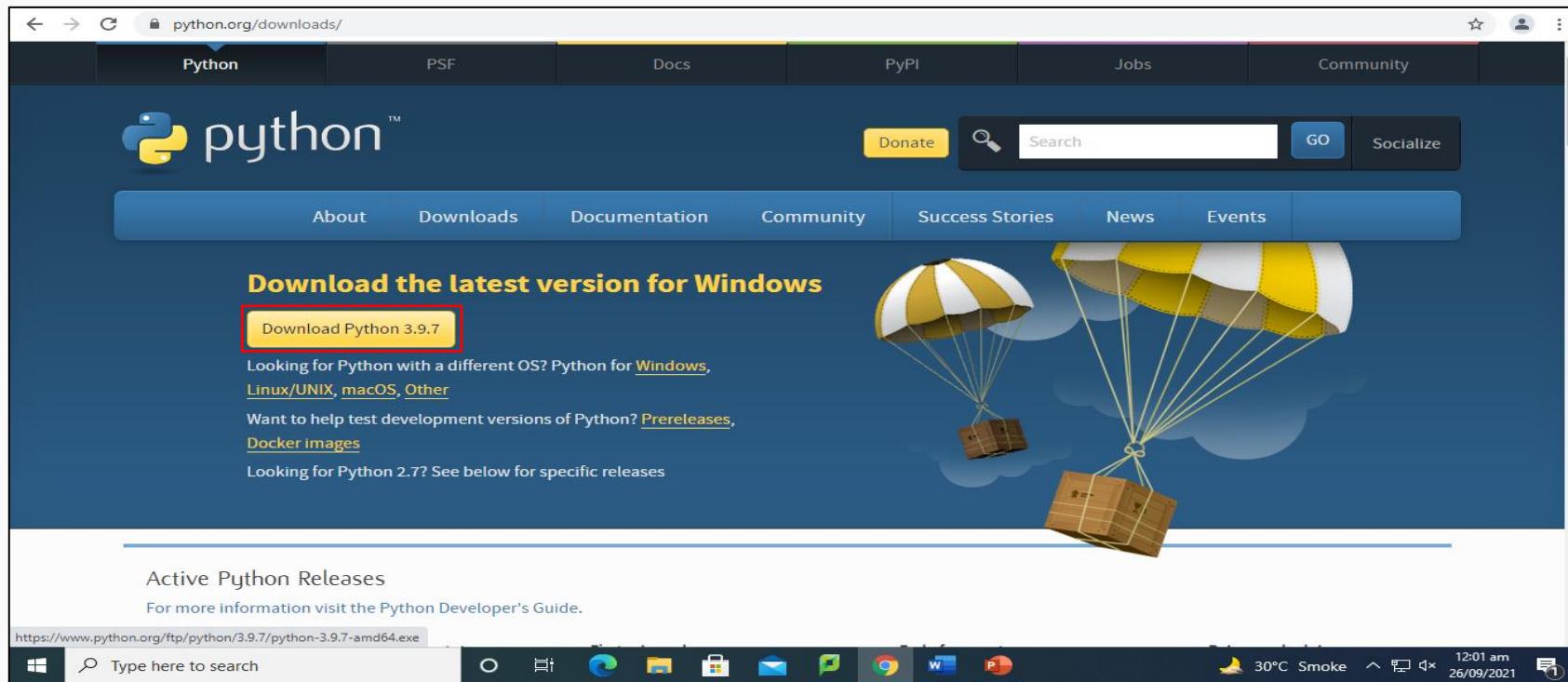
Type here to search

12:01 am 26/09/2021



Setting Up Python – Windows (3/8)

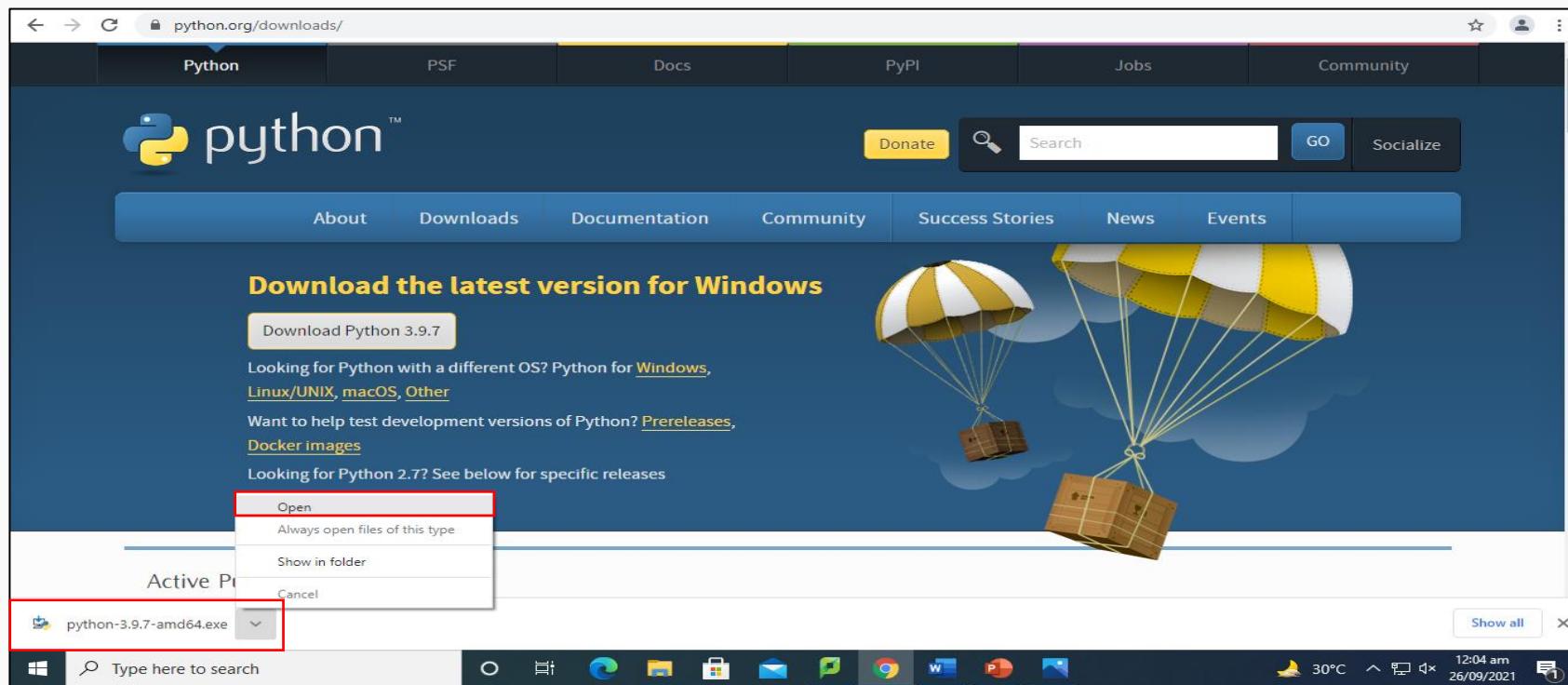
- This will show you the latest version of Python3 for your operating system. Download it.





Setting Up Python – Windows (4/8)

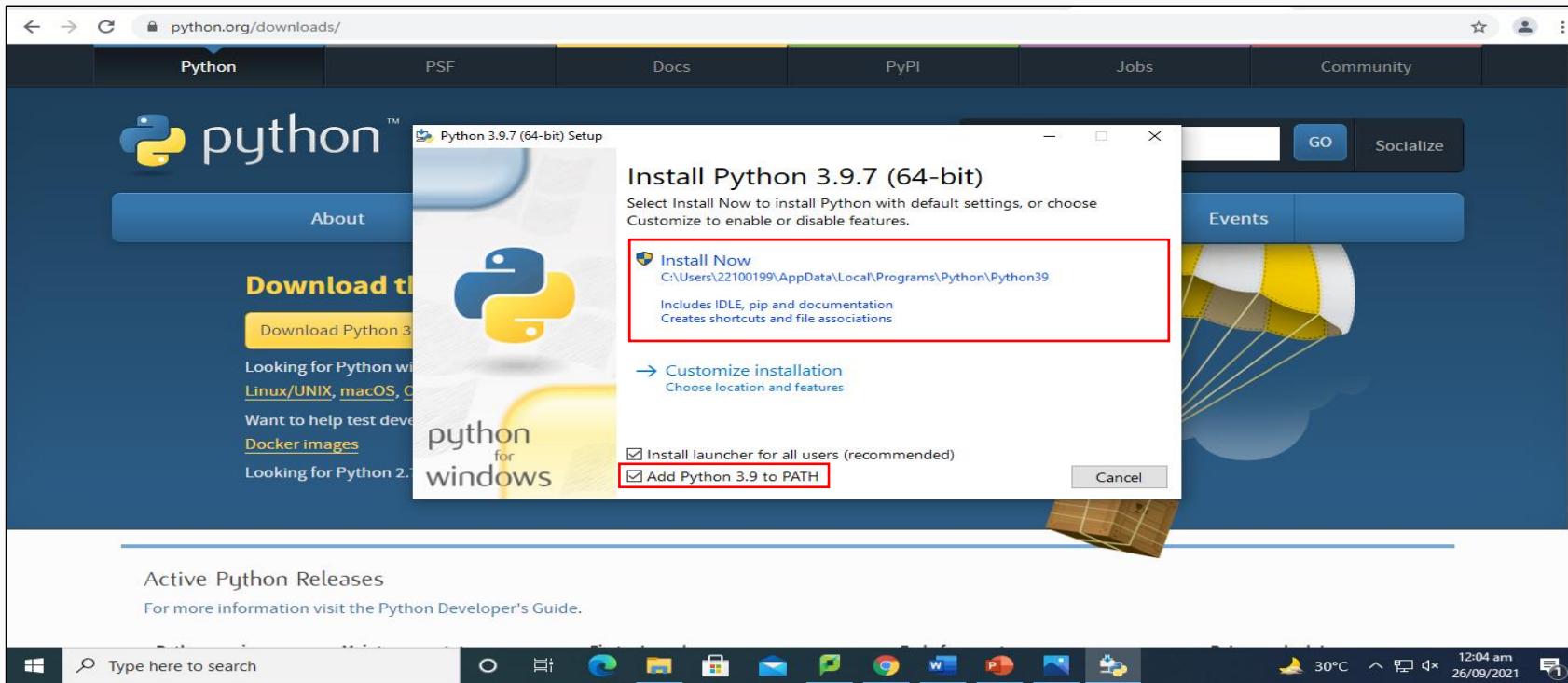
- Once downloaded, click Open.





Setting Up Python – Windows (5/8)

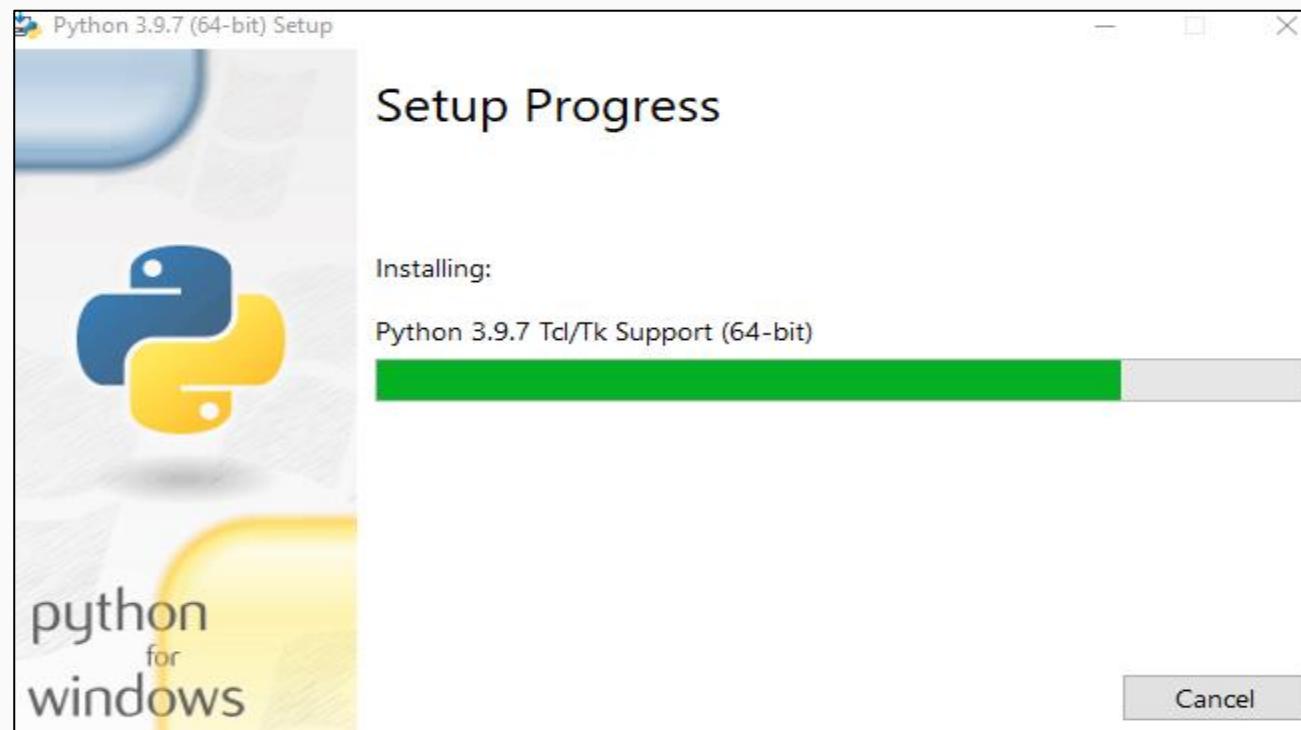
- Installation wizard will open up. Make sure that ‘Add Python 3.9 to Path’ is checked. Click on Install Now.





Setting Up Python – Windows (6/8)

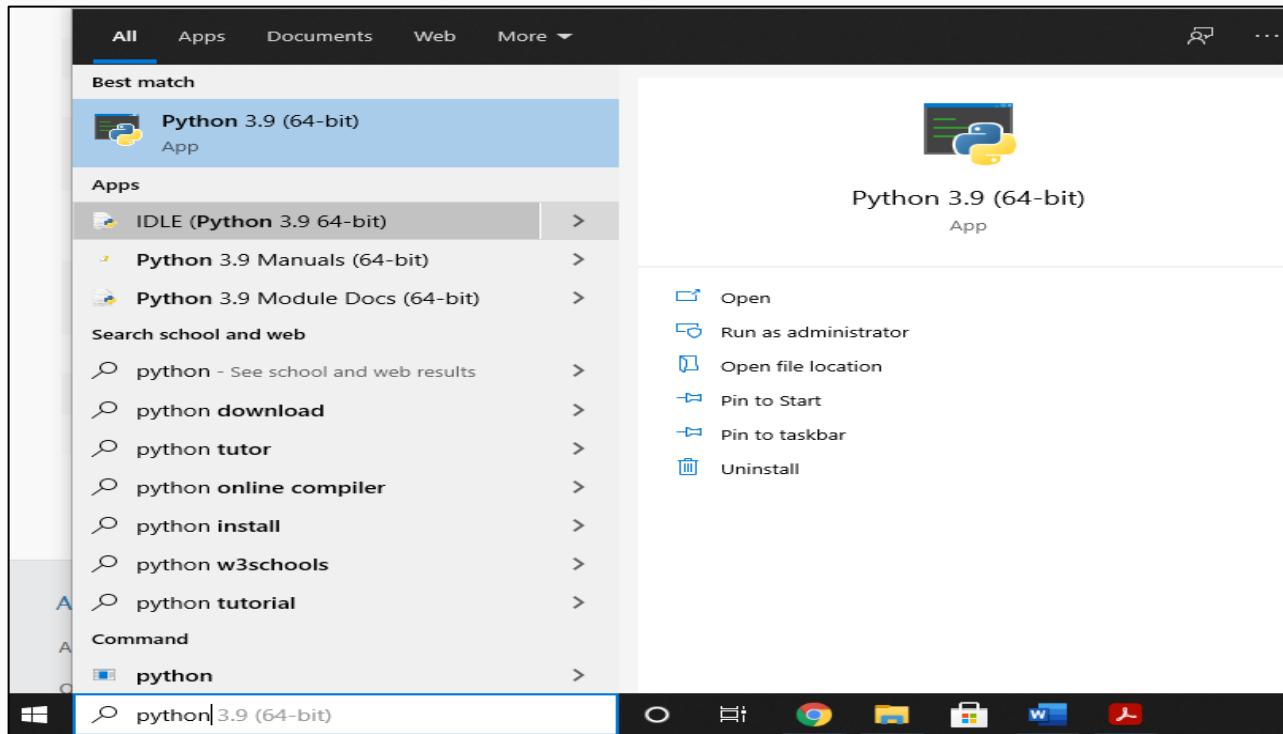
- It will take sometime to install.





Setting Up Python – Windows (7/8)

- Once installed, type python in your search bar and click on ‘IDLE (Python 3.9 64-bit)’.





Setting Up Python – Windows (8/8)

- This will open up Python IDLE. Type print("Hello") and press Enter to execute.

The screenshot shows the Python IDLE Shell 3.9.7 window. The title bar reads "IDLE Shell 3.9.7". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The status bar at the bottom indicates "Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32". The main window displays the following text:
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello")
Hello
>>> |



Setting Up Python – Mac (1/13)

- Visit: <https://www.python.org/>

The screenshot shows the Python.org homepage. A red box highlights the browser's address bar, which displays the URL <https://www.python.org/>. The page features a dark blue header with the Python logo and navigation links for Python, PSF, Docs, PyPI, Jobs, and Community. Below the header is a search bar with a 'Search' button and a 'Socialize' button. A navigation menu at the top includes About, Downloads, Documentation, Community, Success Stories, News, and Events. The main content area contains a code snippet demonstrating a Fibonacci series and a section titled 'Functions Defined' explaining Python's function definition capabilities. At the bottom, a quote emphasizes Python's efficiency and integration capabilities, followed by a 'Learn More' link.

Python 3: Fibonacci series up to n

```
>>> def fib(n):
>>>     a, b = 0, 1
>>>     while a < n:
>>>         print(a, end=' ')
>>>         a, b = b, a+b
>>>     print()
>>> fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610
987
```

Functions Defined

The core of extensible programming is defining functions. Python allows mandatory and optional arguments, keyword arguments, and even arbitrary argument lists. [More about defining functions in Python 3](#)

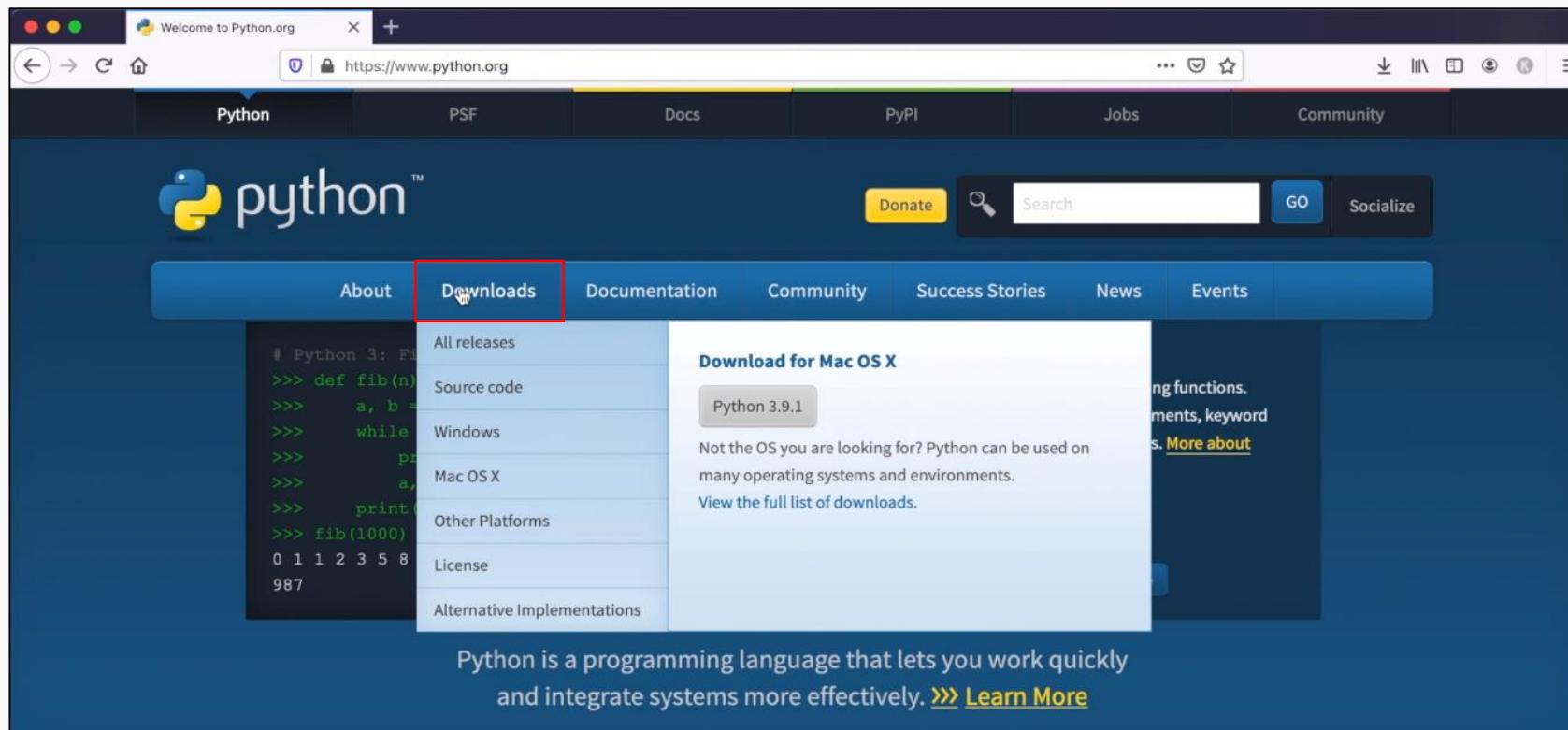
1 2 3 4 5

Python is a programming language that lets you work quickly and integrate systems more effectively. [» Learn More](#)



Setting Up Python – Mac (2/13)

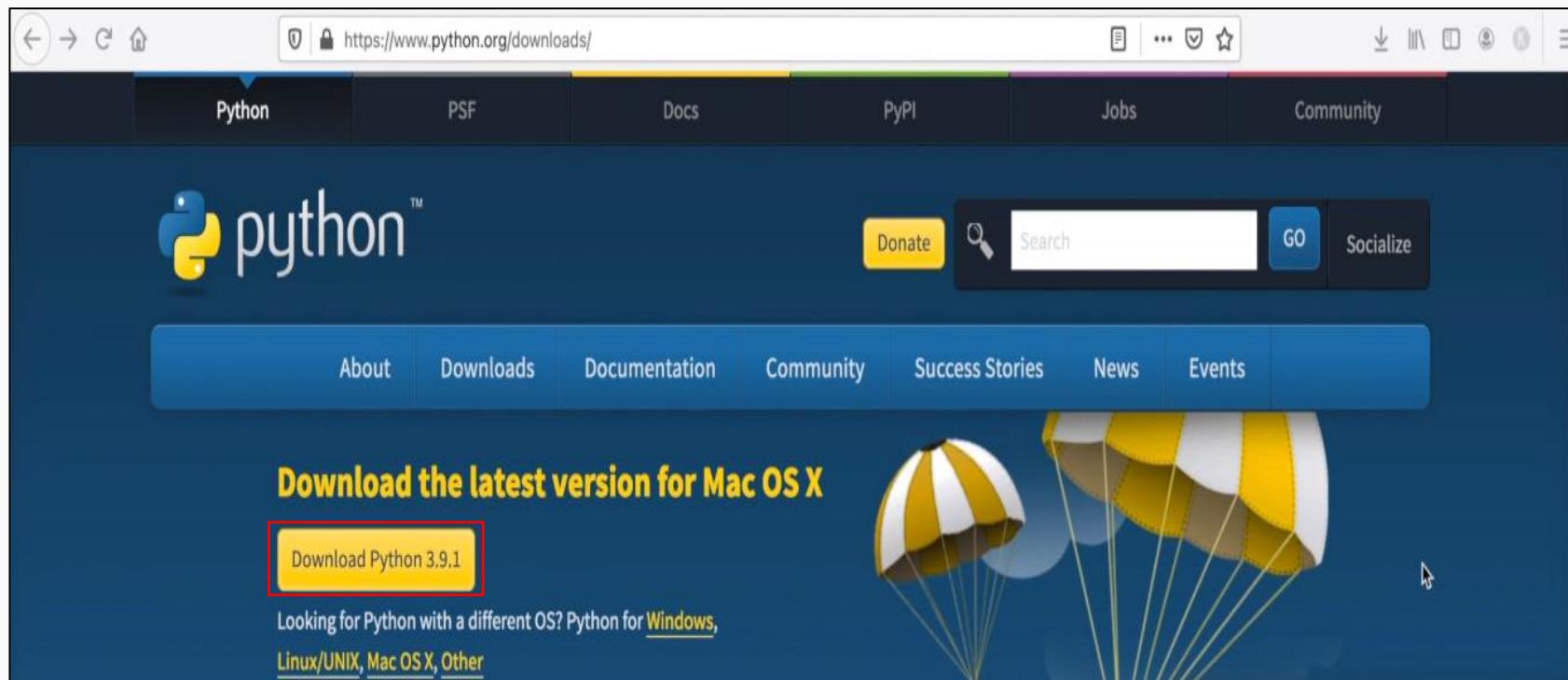
- Click on Downloads.





Setting Up Python – Mac (3/13)

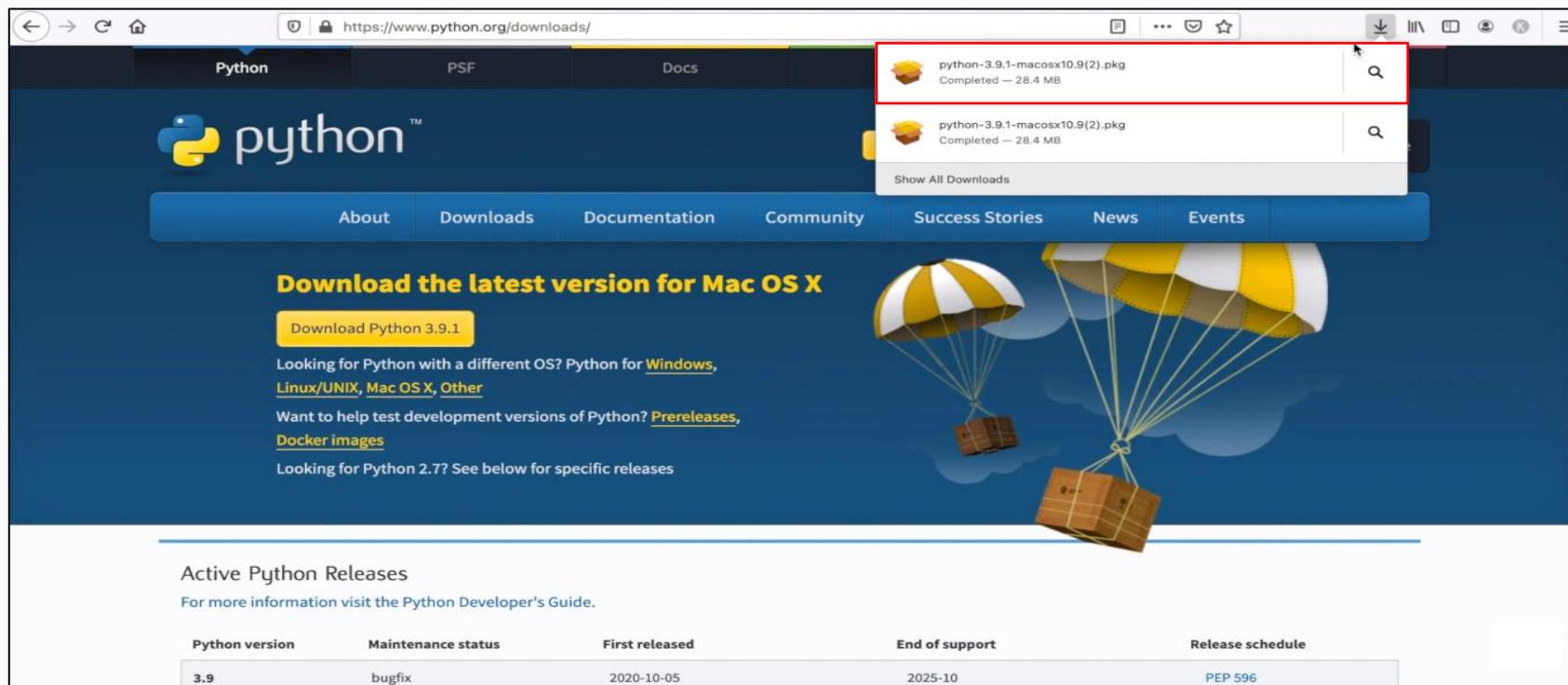
- This will show you the latest version of Python3 for your operating system. Download it.





Setting Up Python – Mac (4/13)

- Once downloaded, click on the file.





Setting Up Python – Mac (5/13)

- Click continue.





Setting Up Python – Mac (6/13)

- Click continue.





Setting Up Python – Mac (7/13)

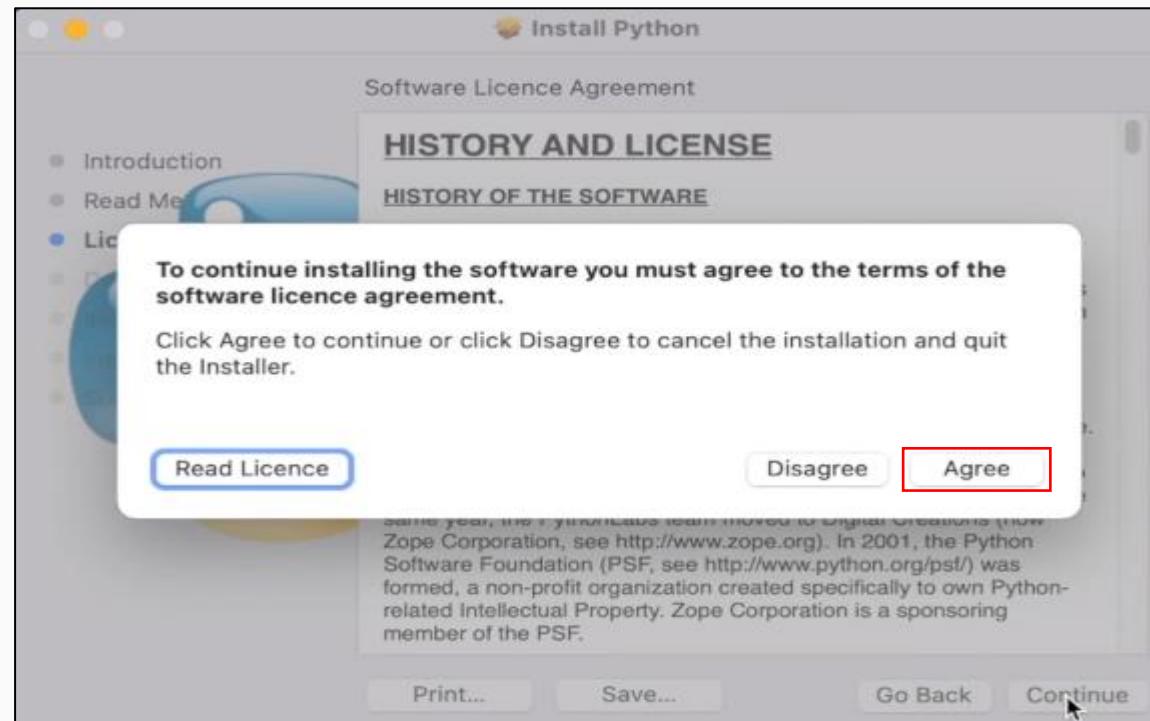
- Click continue.





Setting Up Python – Mac (8/13)

- Read and accept the license agreement.





Setting Up Python – Mac (9/13)

- Click continue.





Setting Up Python – Mac (10/13)

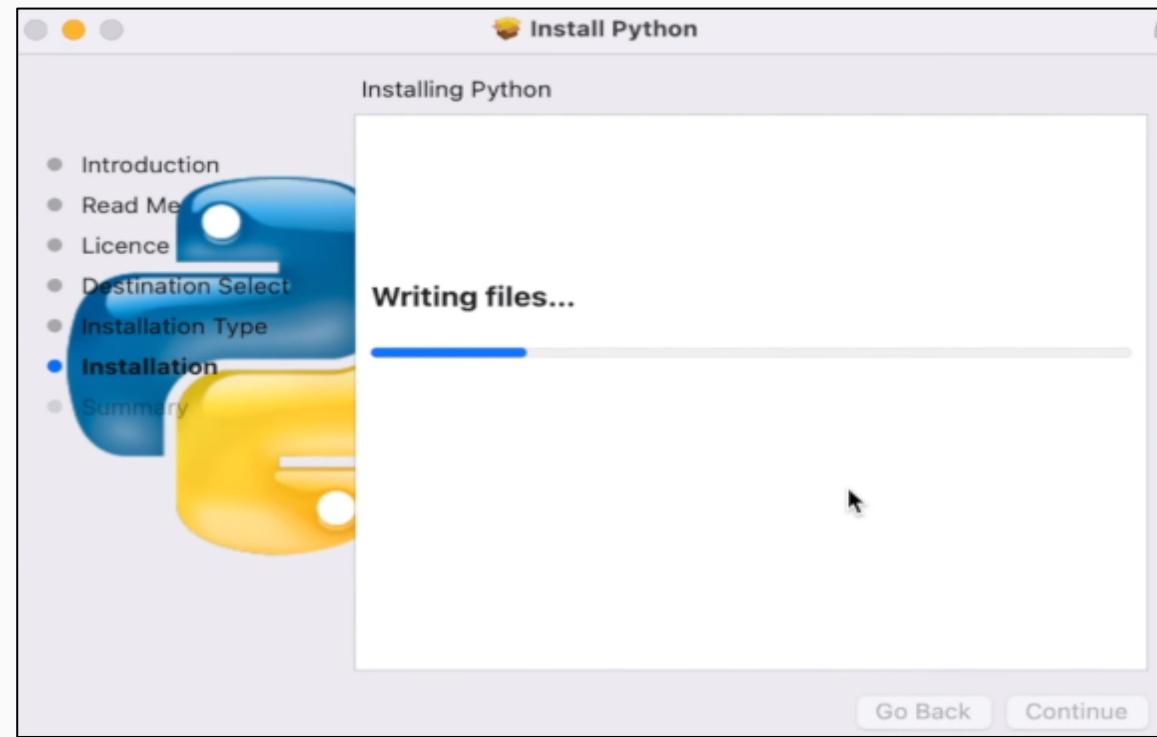
- Click install. If prompted, enter your password.





Setting Up Python – Mac (11/13)

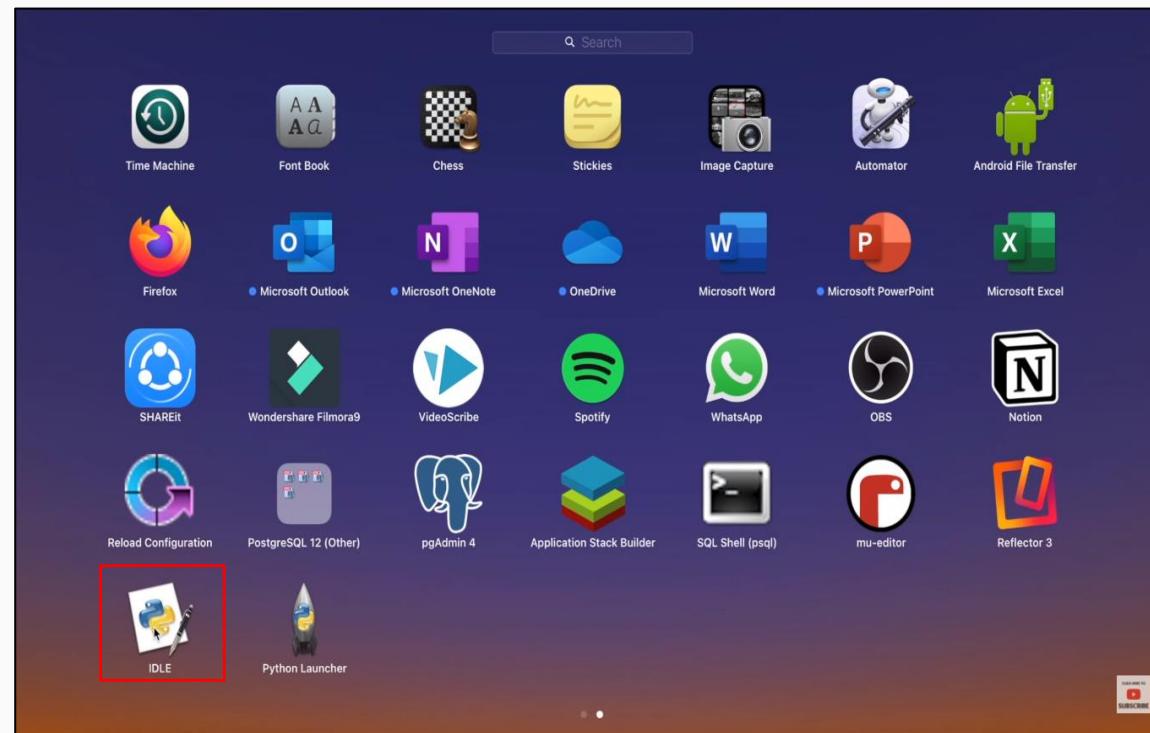
- Wait for the installation to finish.





Setting Up Python – Mac (12/13)

- Once installed, search for IDLE in your spotlight and click on it.





Setting Up Python – Mac (13/13)

- This will open up Python IDLE. Type print("Hello World") and press Enter to execute.

The screenshot shows the Python IDLE Shell 3.9.1 window. The title bar reads "IDLE Shell 3.9.1". The main area displays the following text:

```
Python 3.9.1 (v3.9.1:1e5d33e9b9, Dec  7 2020, 12:10:52)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
WARNING: The system preference "Prefer tabs when opening documents" is set to
"Always". This will cause various problems with IDLE. For the best experience,
change this setting when running IDLE (via System Preferences -> Dock).
>>> print('Hello World')
Hello World
>>>
>>>
>>> |
```

The bottom status bar indicates "Ln: 11 Col: 4". A cursor arrow is visible in the center of the window.

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction

01

Python libraries for Machine Learning

02

Setting up Python

03

What is Jupyter?

04

Anaconda Installation for Windows OS

05

Anaconda Installation for Mac OS

06

Anaconda Installation for Ubuntu OS

07

Implementing Python in Jupyter

08

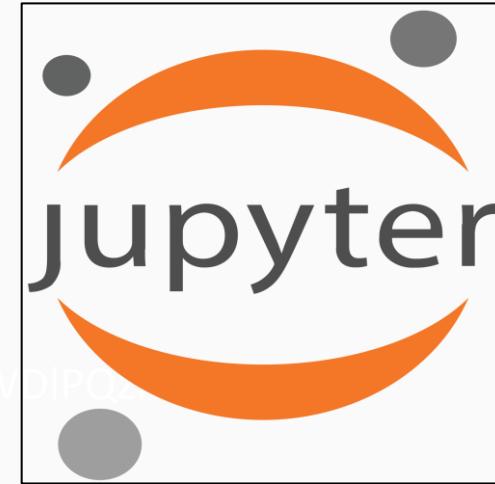
Managing Directories in Jupyter Notebook

09



Jupyter Notebook

- Jupyter Notebook is a web application that allows you to create documents that may contain code, visualizations, and narrative text.
- It is very popular and is used extensively in data science, artificial intelligence, and machine learning.
<https://uniho.zoom.us/j/83422537997?pwd=Z3VENmI0QTV2Q2xIWDP0ZkFzZGZzZz09>
- A notebook has the .ipynb extension



Alternatives to Jupyter Notebook

- Google Colaboratory is an alternative to Jupyter Notebook and does not require downloading anything.

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction

01

Python libraries for Machine Learning

02

Setting up Python

03

What is Jupyter?

04

Anaconda Installation for Windows OS

05

Anaconda Installation for Mac OS

06

Anaconda Installation for Ubuntu OS

07

Implementing Python in Jupyter

08

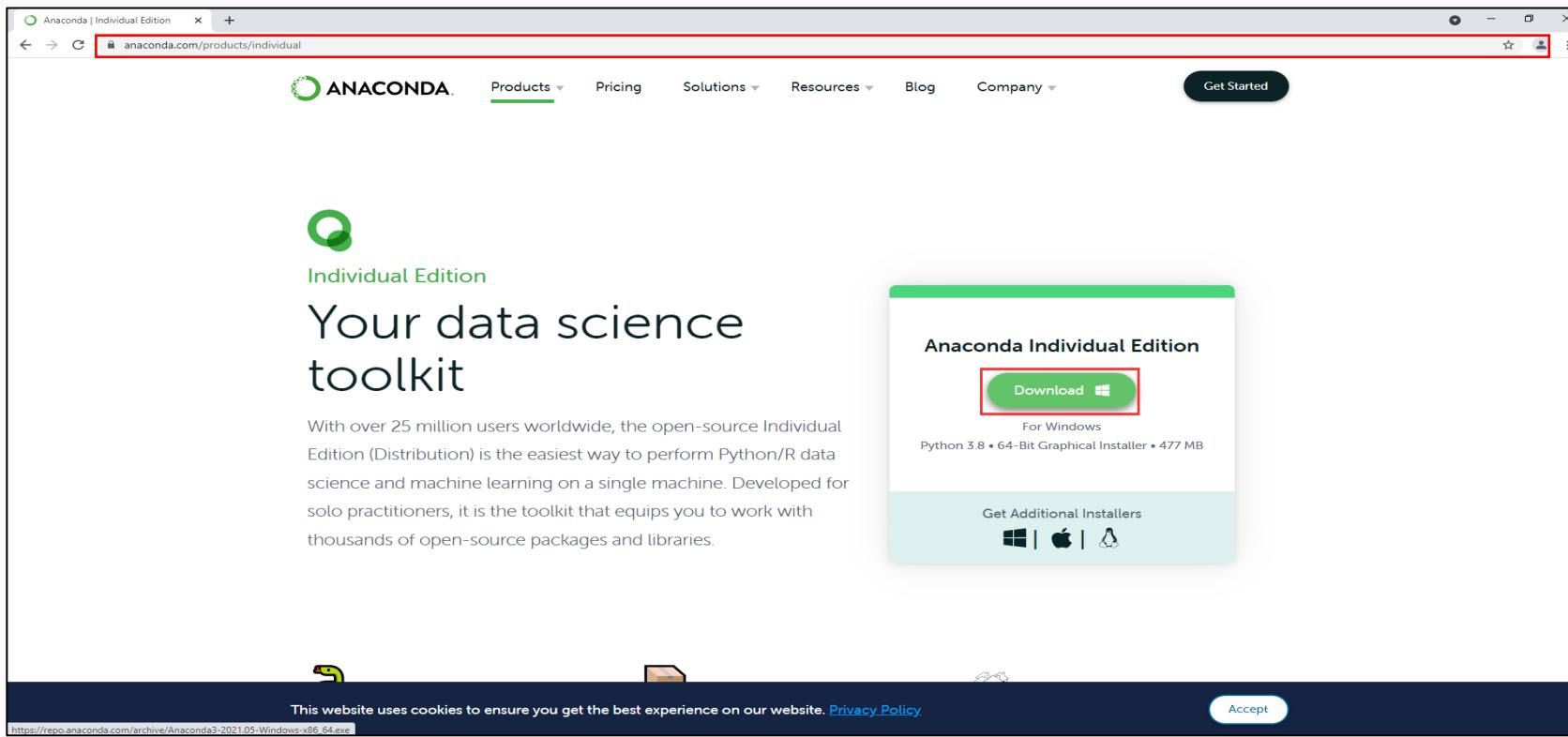
Managing Directories in Jupyter Notebook

09



Anaconda for Windows (1/13)

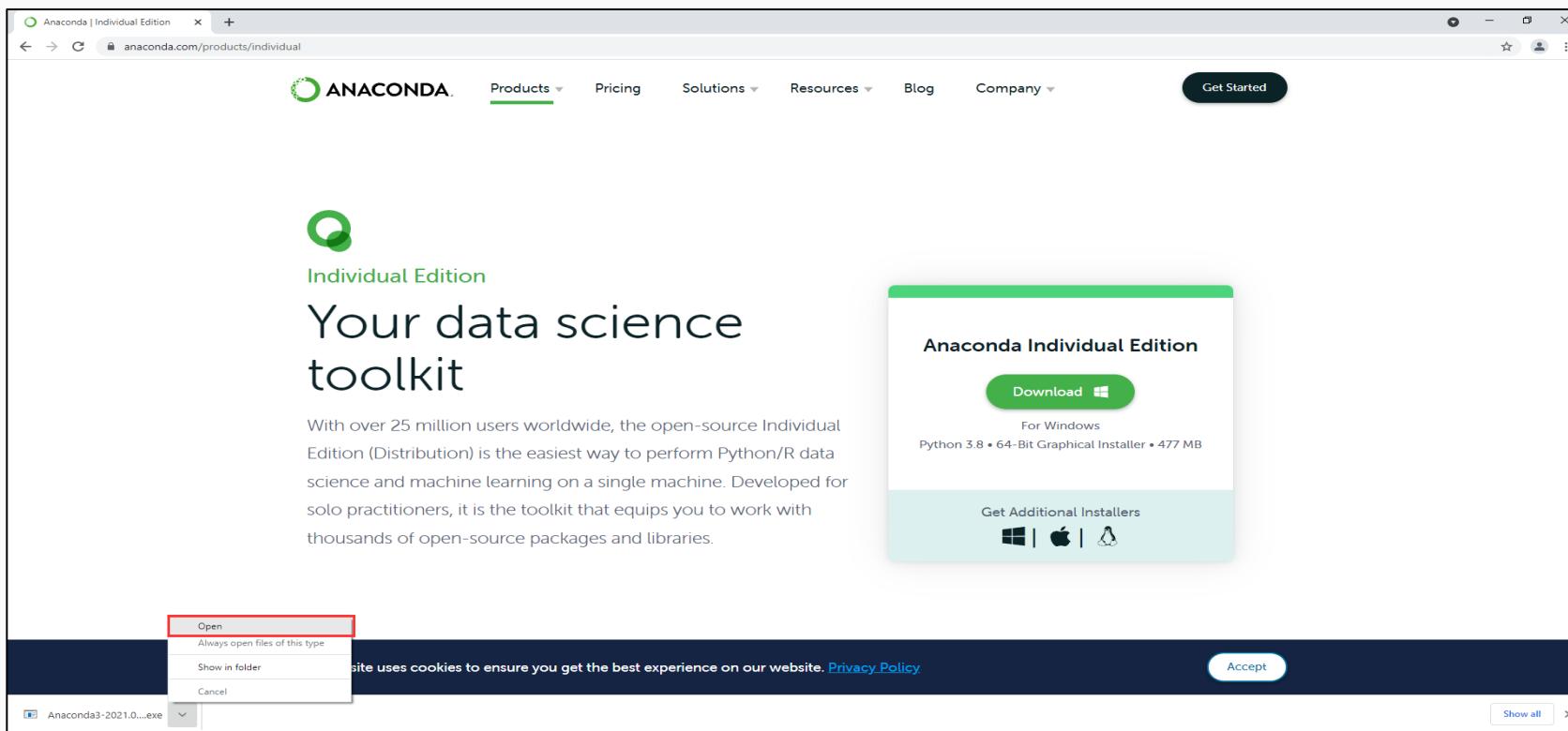
- Visit: <https://www.anaconda.com/products/individual>
- Click on the Download button.





Anaconda for Windows (2/13)

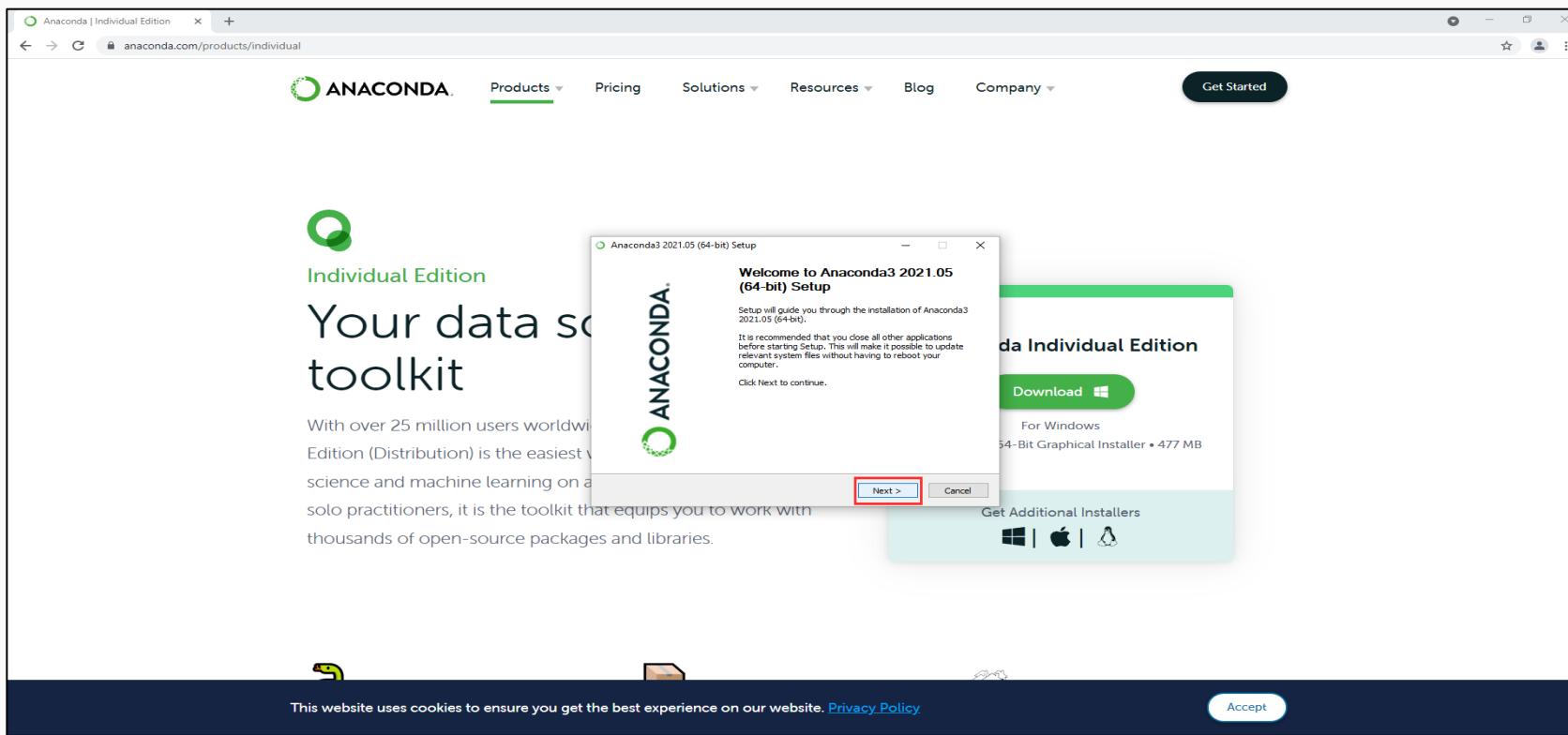
- Once Downloaded, click on Open.





Anaconda for Windows (3/13)

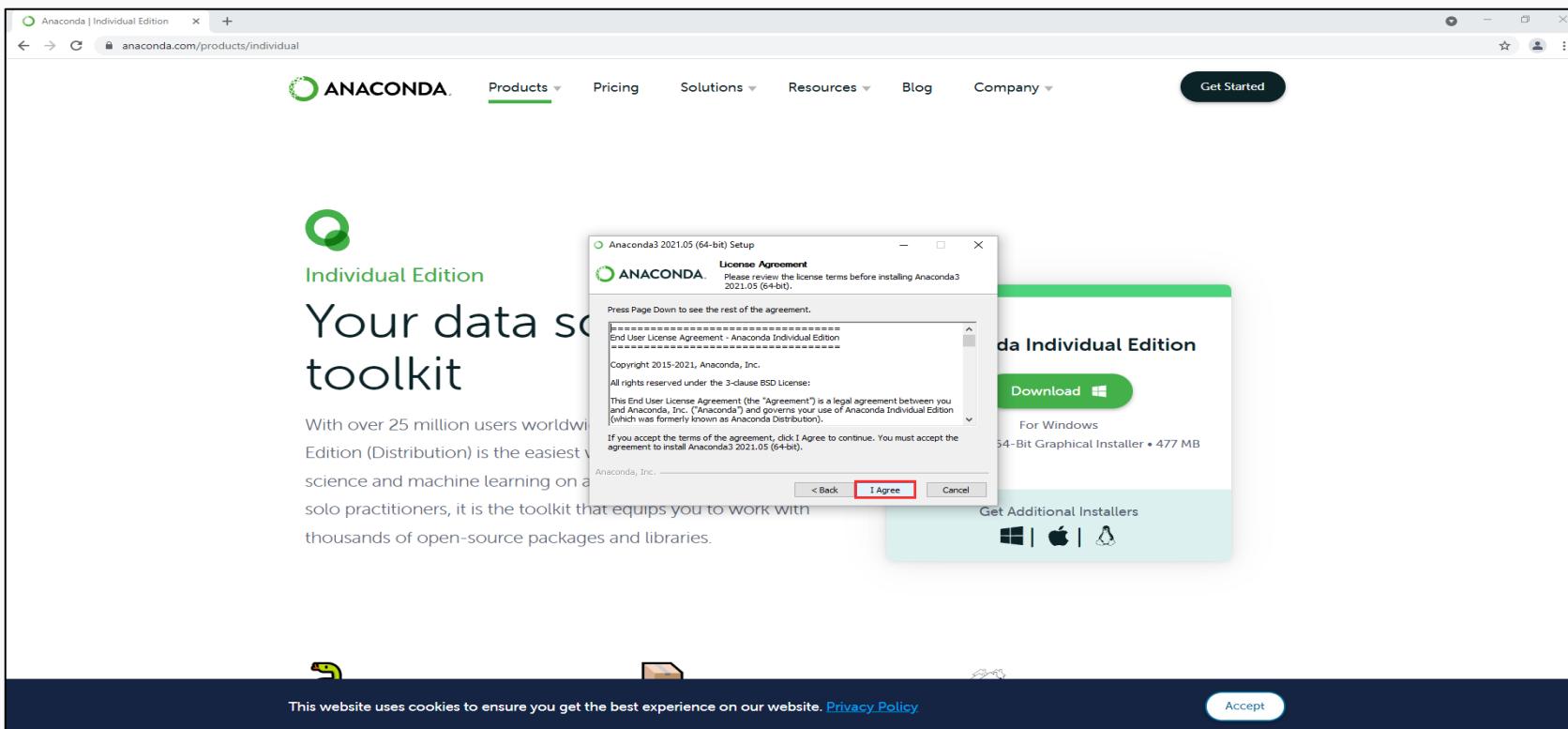
- An installation Wizard will open up.
- Click on Next.





Anaconda for Windows (4/13)

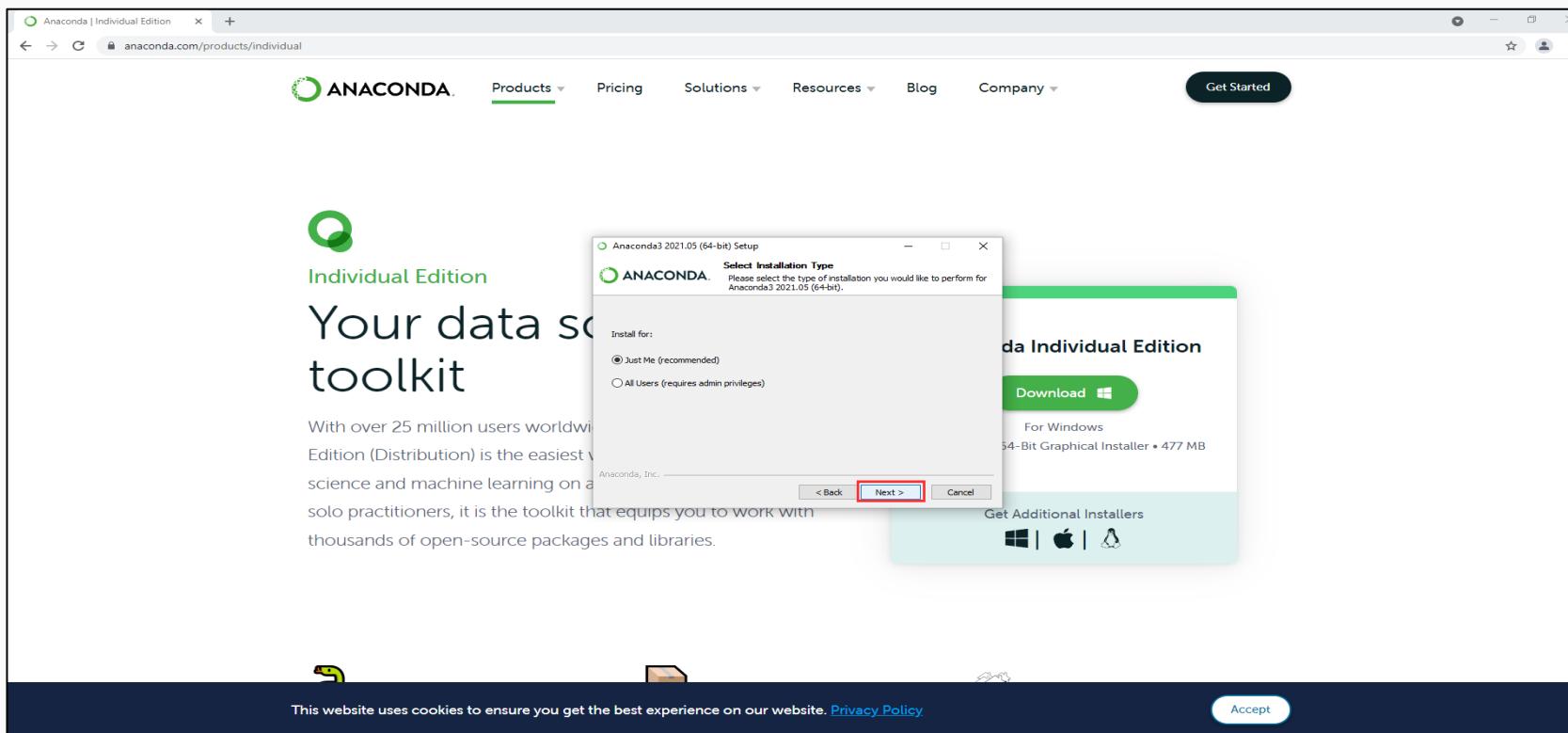
- Read the License Agreement and click I Agree.





Anaconda for Windows (5/13)

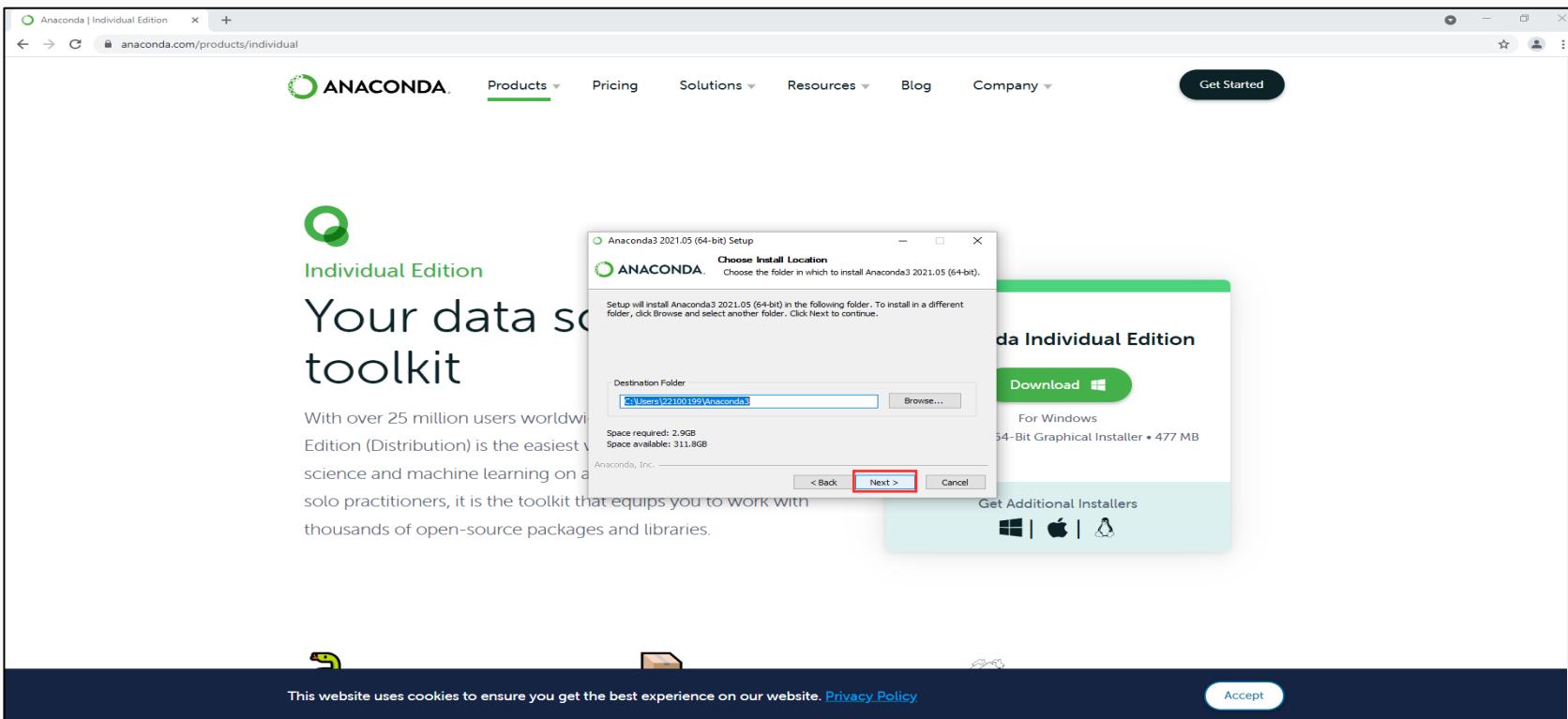
- Click Next.





Anaconda for Windows (6/13)

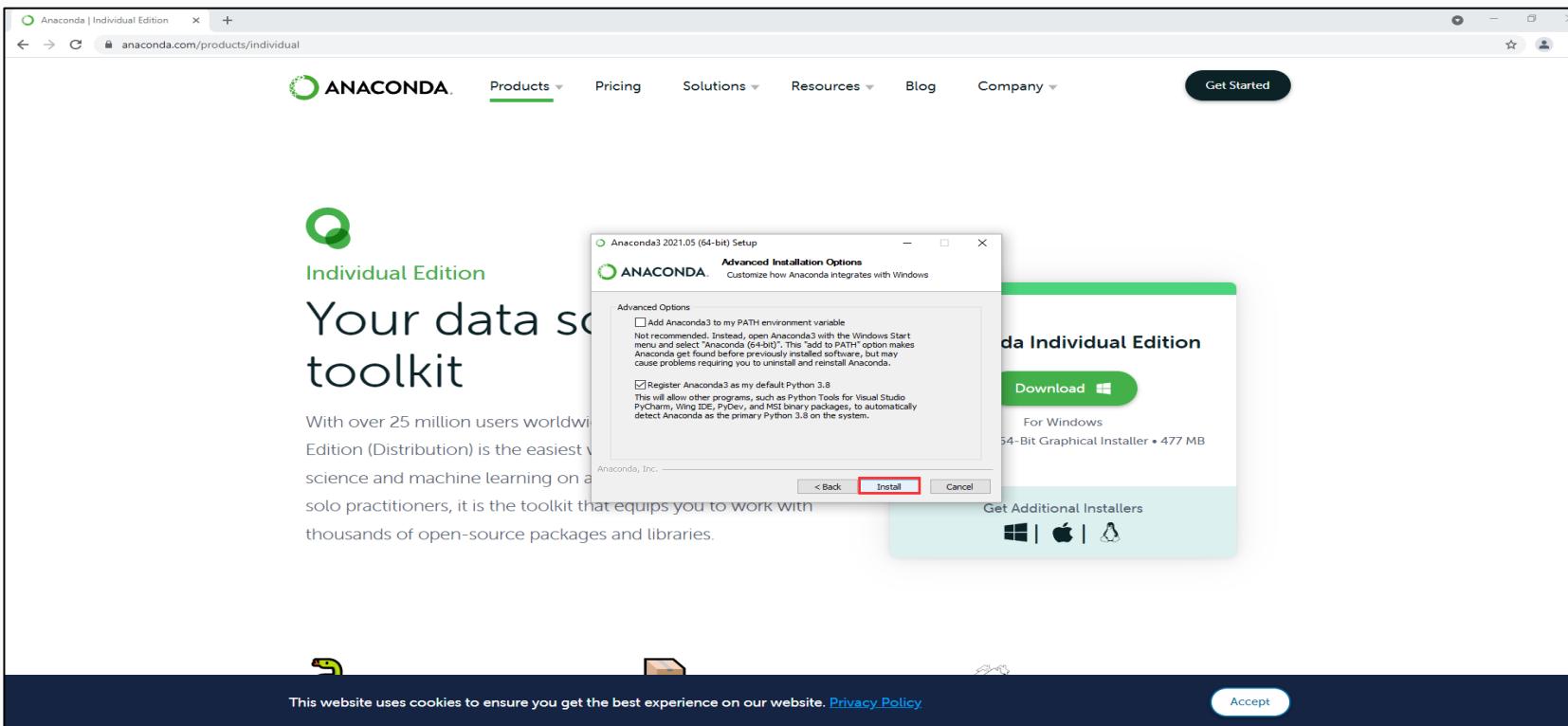
- Choose the destination and click Next.





Anaconda for Windows (7/13)

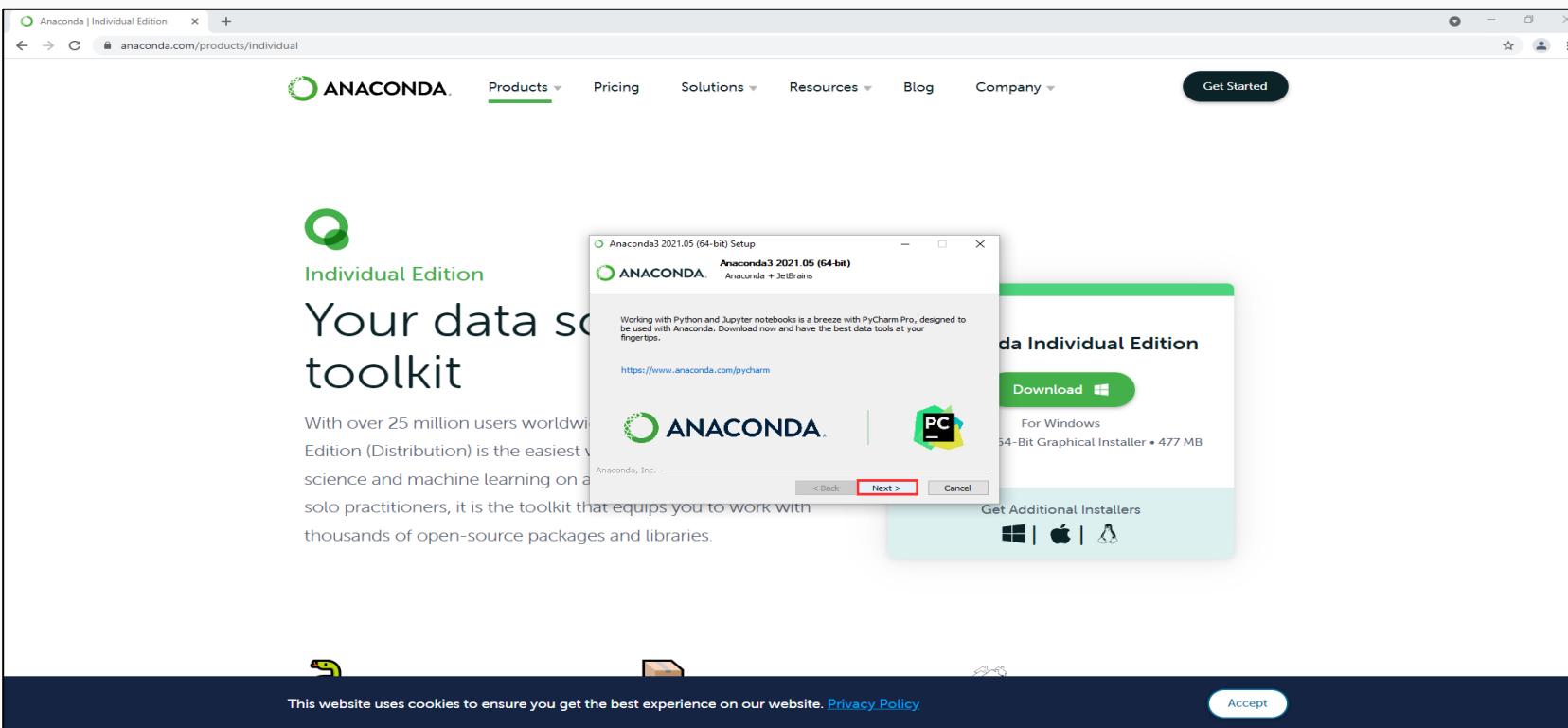
- Click Install.





Anaconda for Windows (8/13)

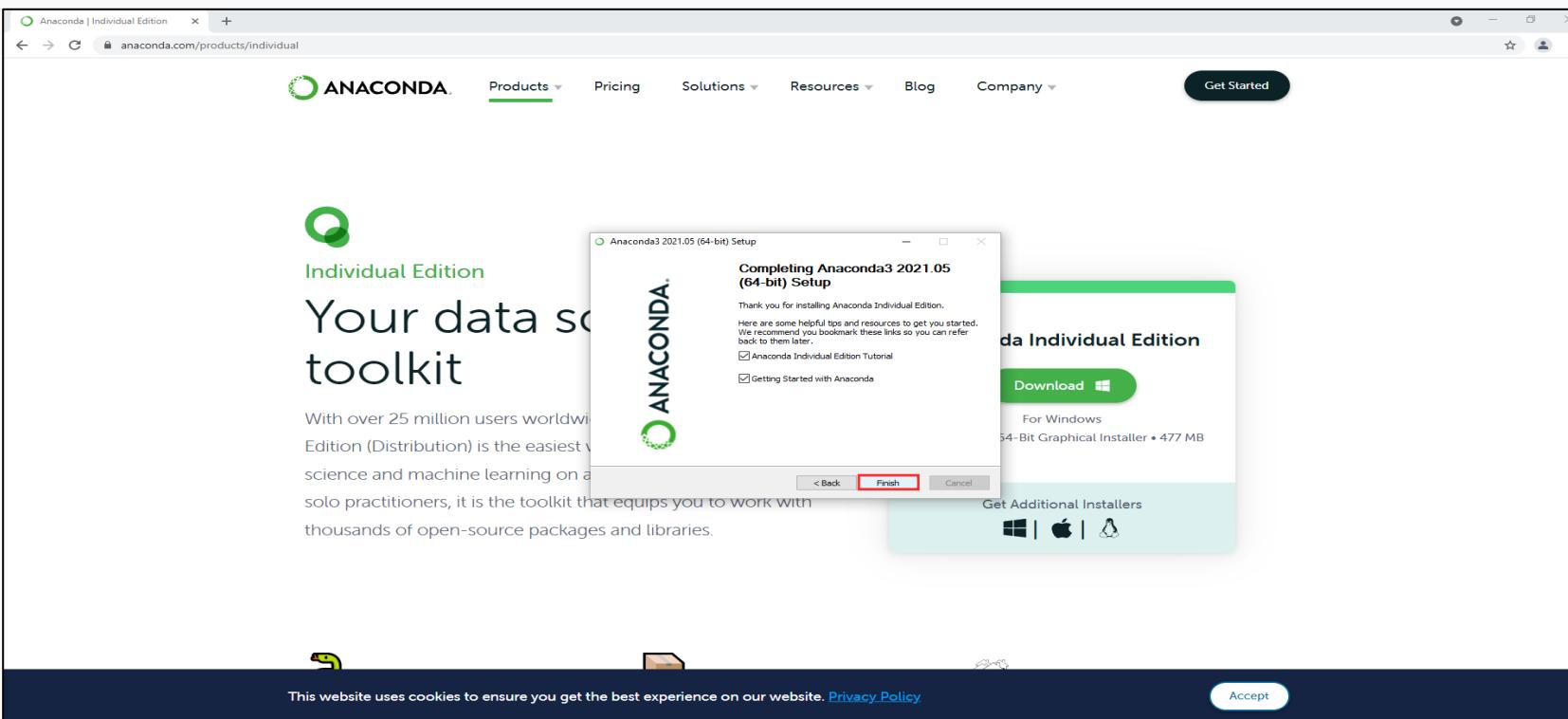
- Wait for the installation to finish and click Next.





Anaconda for Windows (9/13)

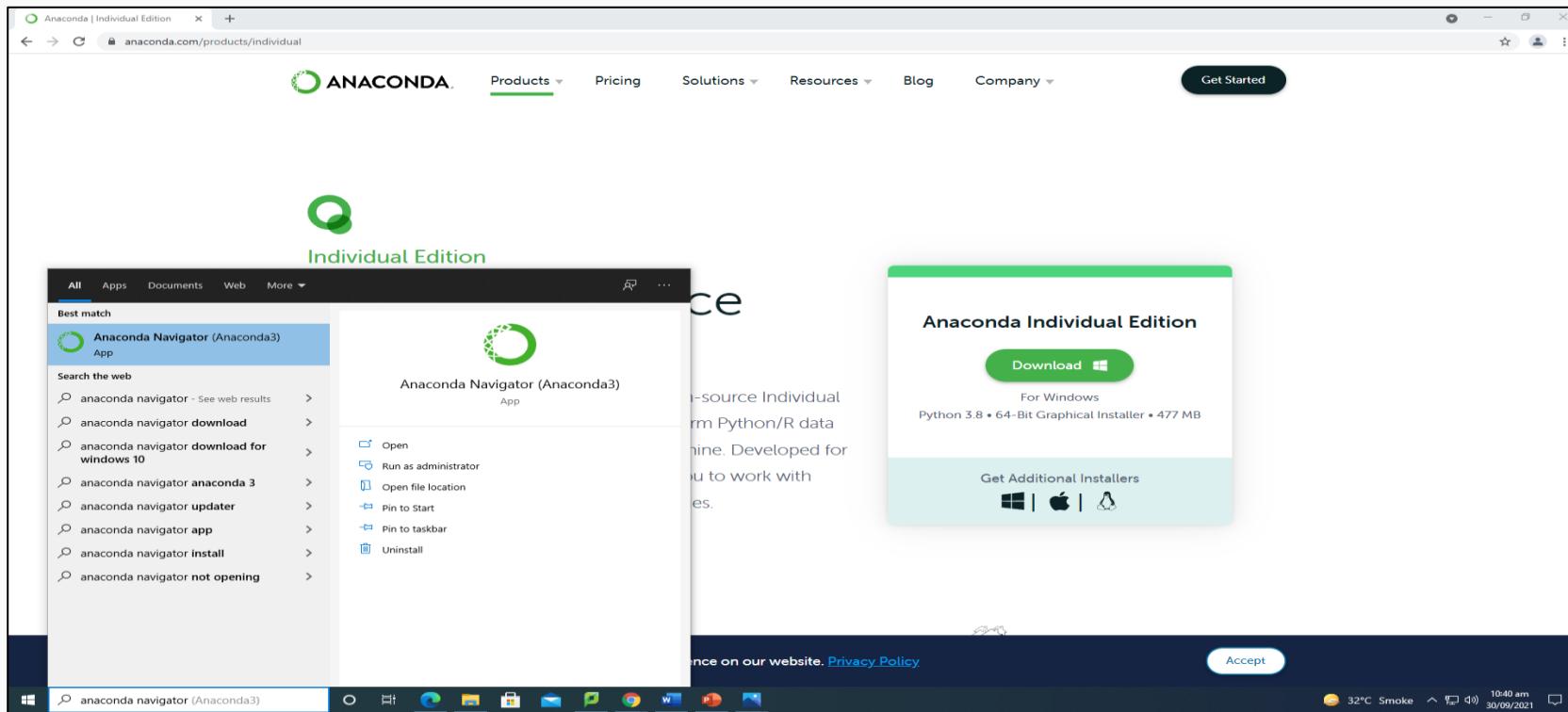
- Click Finish.





Anaconda for Windows (10/13)

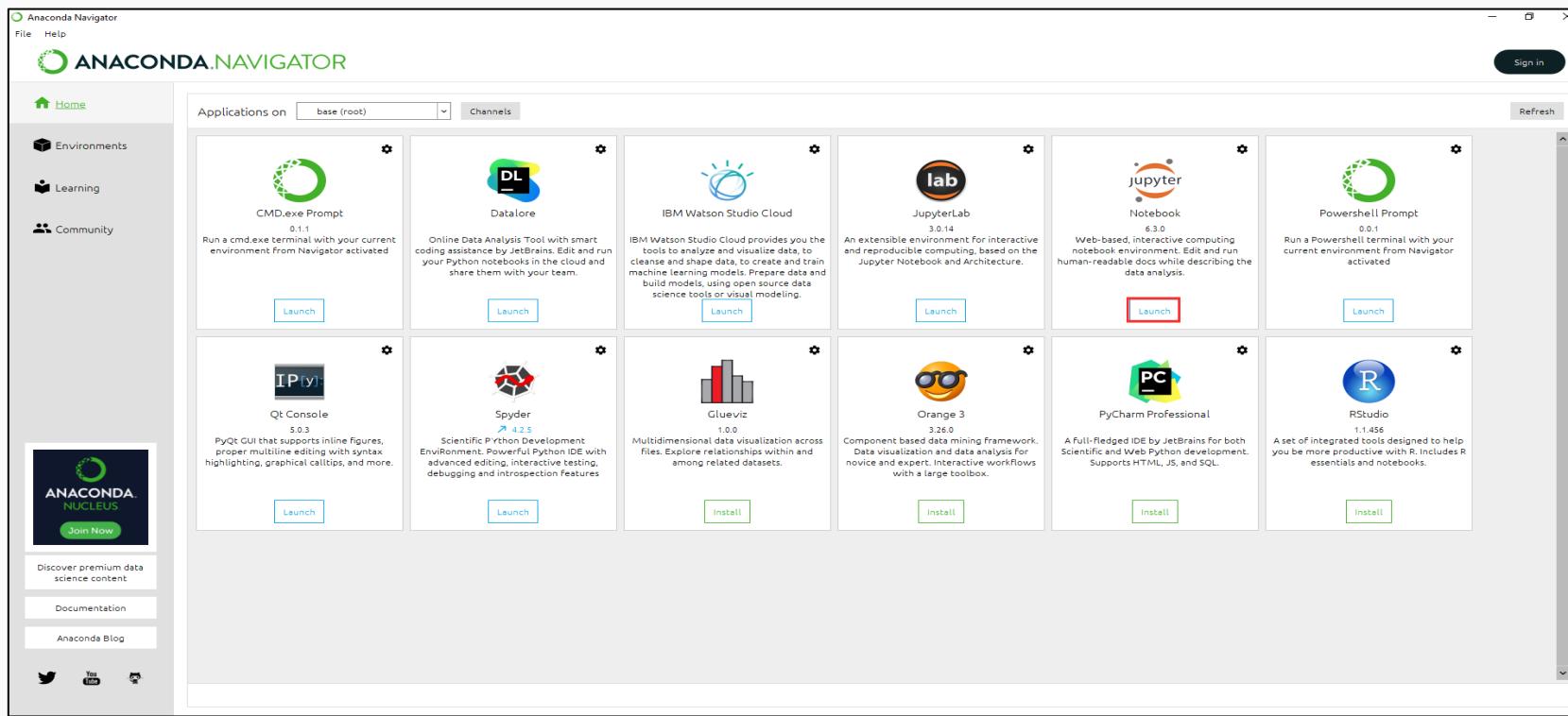
- To launch Jupyter Notebook, search for anaconda navigator and open it.





Anaconda for Windows (11/13)

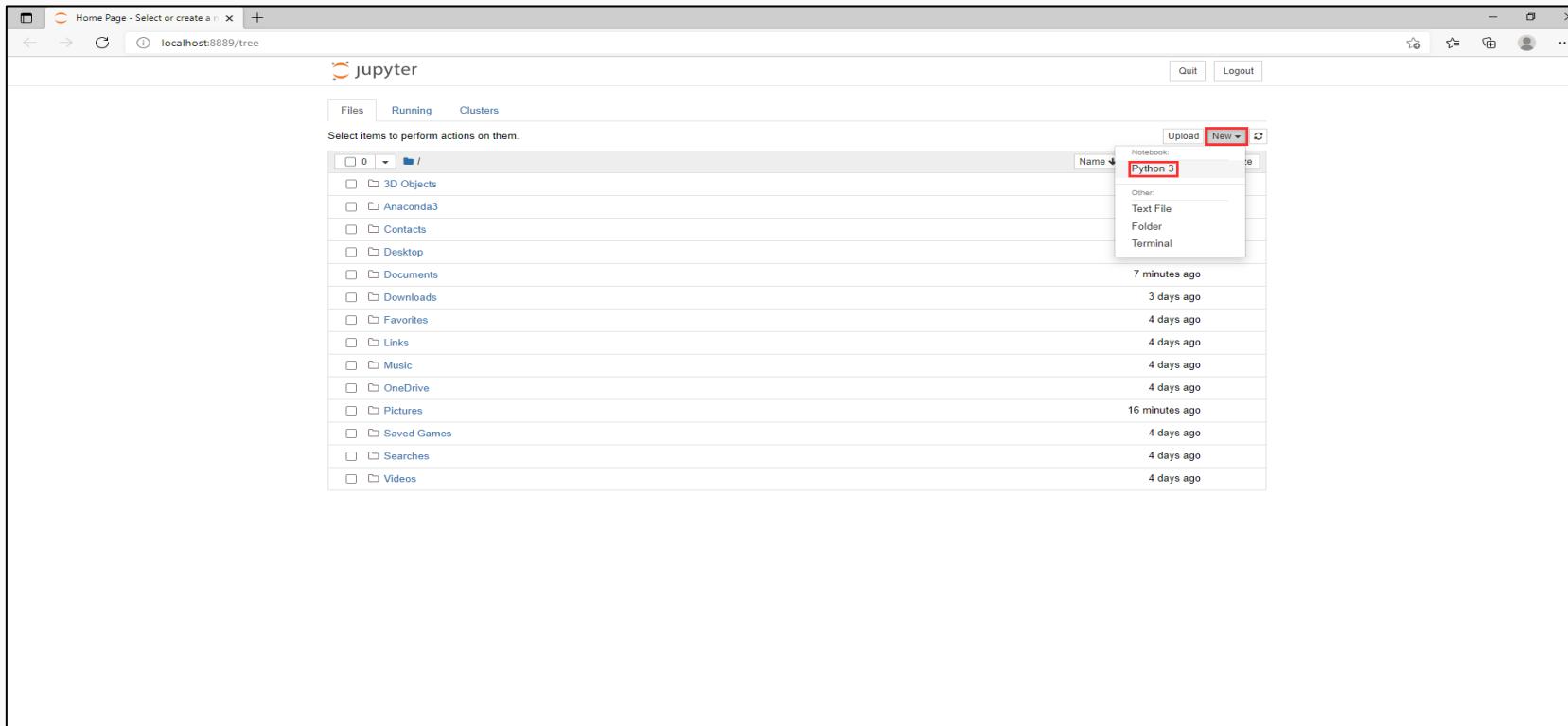
- Following screen will open up.
- Launch Jupyter Notebook from here.





Anaconda for Windows (12/13)

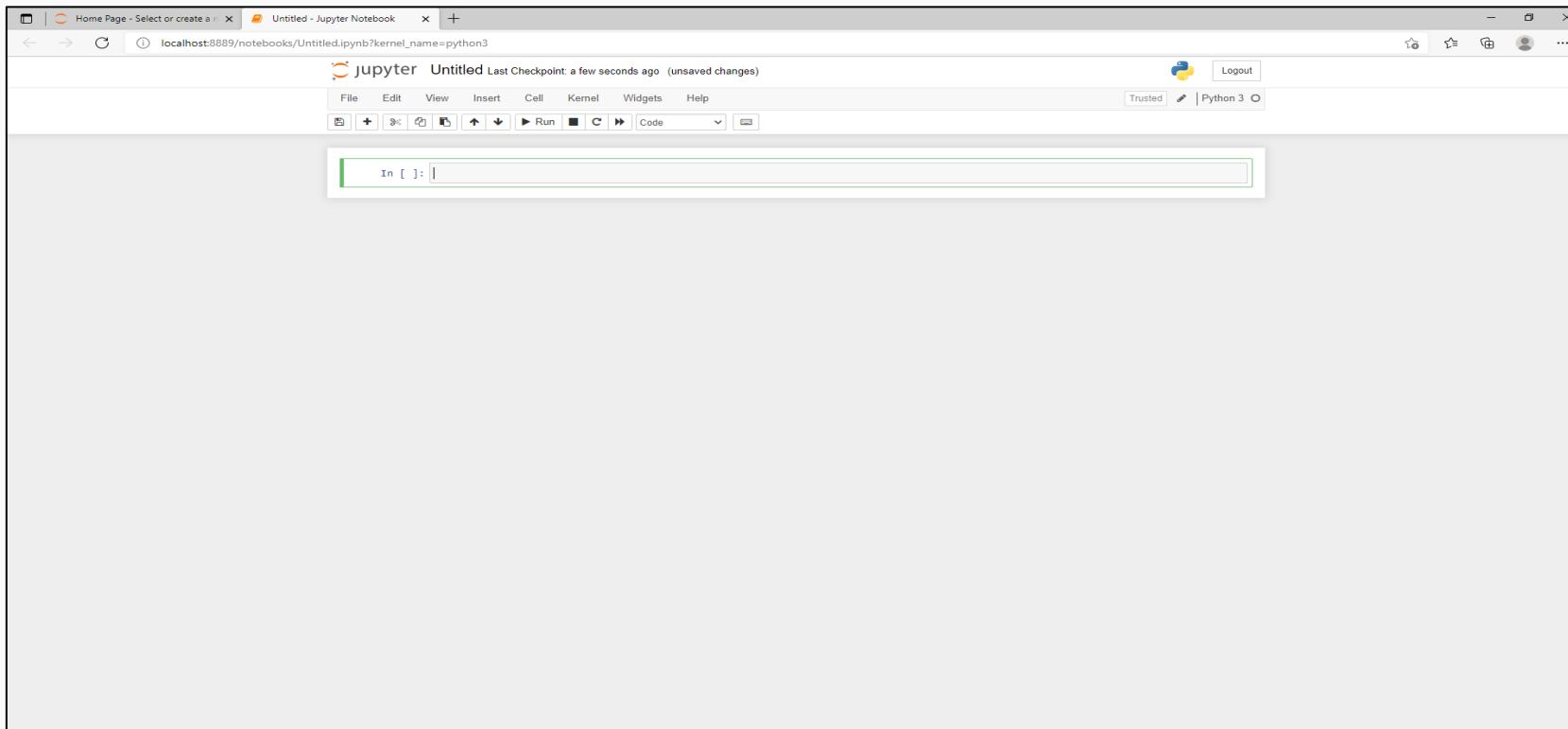
- This will launch Jupyter Notebook in your default browser.
- Click on New -> Python3 to create a new Notebook.





Anaconda for Windows (13/13)

- This is how a Notebook looks like.



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction

01

Python libraries for Machine Learning

02

Setting up Python

03

What is Jupyter?

04

Anaconda Installation for Windows OS

05

Anaconda Installation for Mac OS

06

Anaconda Installation for Ubuntu OS

07

Implementing Python in Jupyter

08

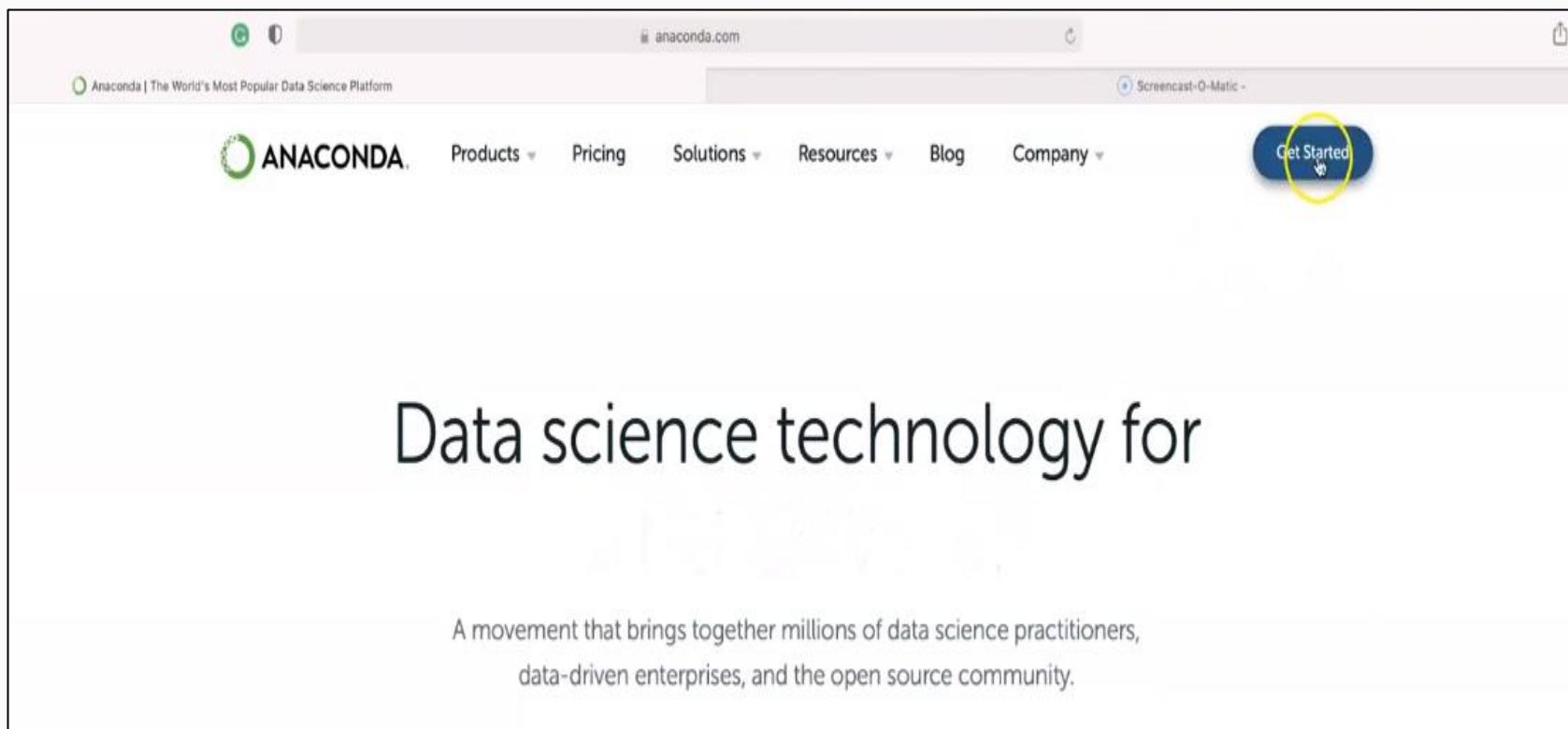
Managing Directories in Jupyter Notebook

09



Anaconda for Mac (1/14)

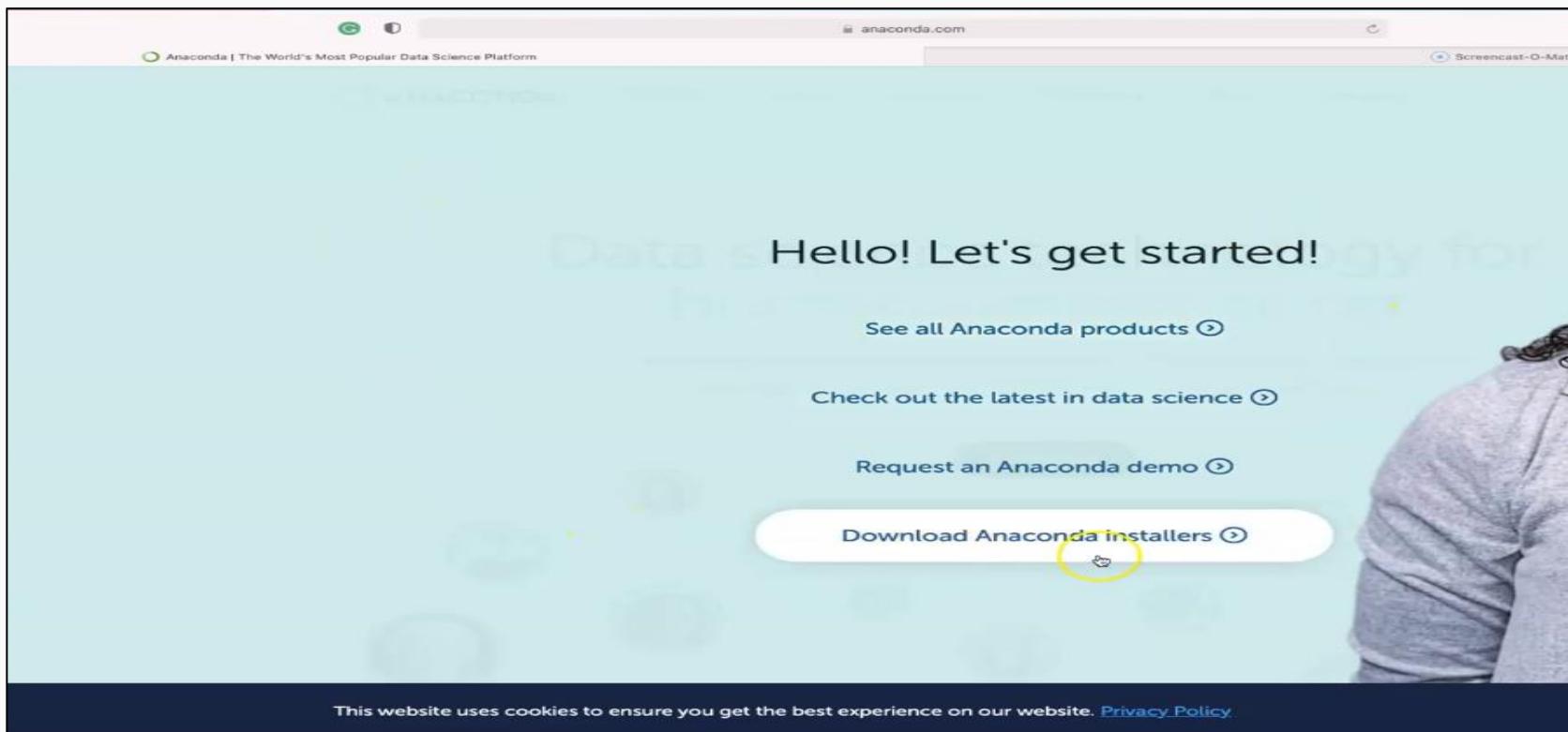
- Visit: <https://www.anaconda.com/>
- Click on Get Started.





Anaconda for Mac (2/14)

- Click Download Anaconda Installers.





Anaconda for Mac (3/14)

- Download the graphical installer for Mac.

The screenshot shows the 'Anaconda Installers' page. It has three main sections: 'Windows' (with icons for 64-bit and 32-bit), 'MacOS' (with icons for Python 3.8, 64-bit graphical, and 64-bit command line installers), and 'Linux' (with icons for Python 3.8, 64-bit x86, 64-bit Power8/Power9, 64-bit AWS Graviton2/ARM64, and 64-bit Linux on IBM Z & LinuxONE). A yellow circle highlights the '64-Bit Graphical Installer' for macOS. Below the sections, there's a 'ADDITIONAL INSTALLERS' box containing text about older versions and a link to the Miniconda installer homepage. At the bottom, a dark bar includes a cookie consent message, an 'Accept' button, and a 'SUBSCRIBE' button with a bell icon.

Windows

MacOS

Linux

ADDITIONAL INSTALLERS

The archive has older versions of Anaconda Individual Edition installers. The [Miniconda installer homepage](#) can be found here.

This website uses cookies to ensure you get the best experience on our website. [Privacy Policy](#)

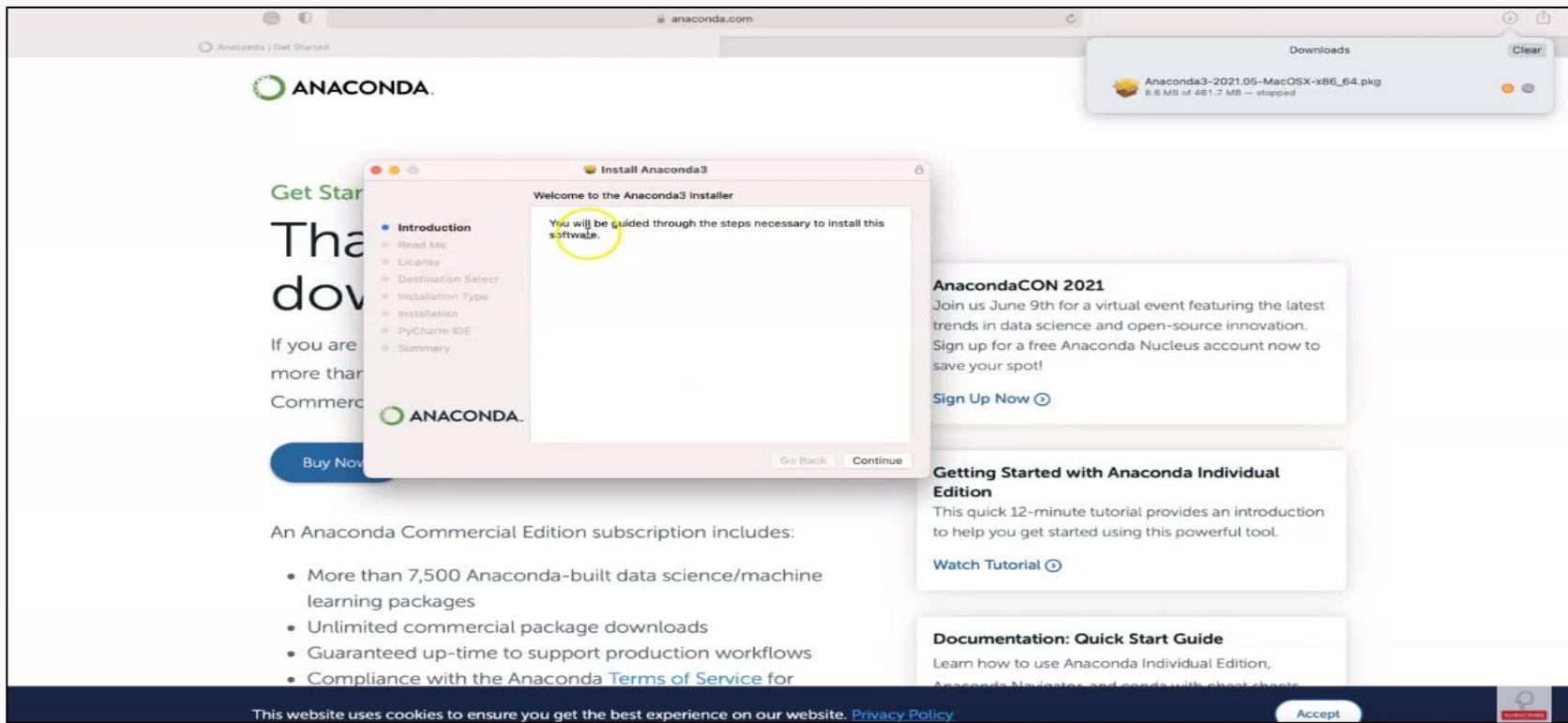
Accept

SUBSCRIBE



Anaconda for Mac (4/14)

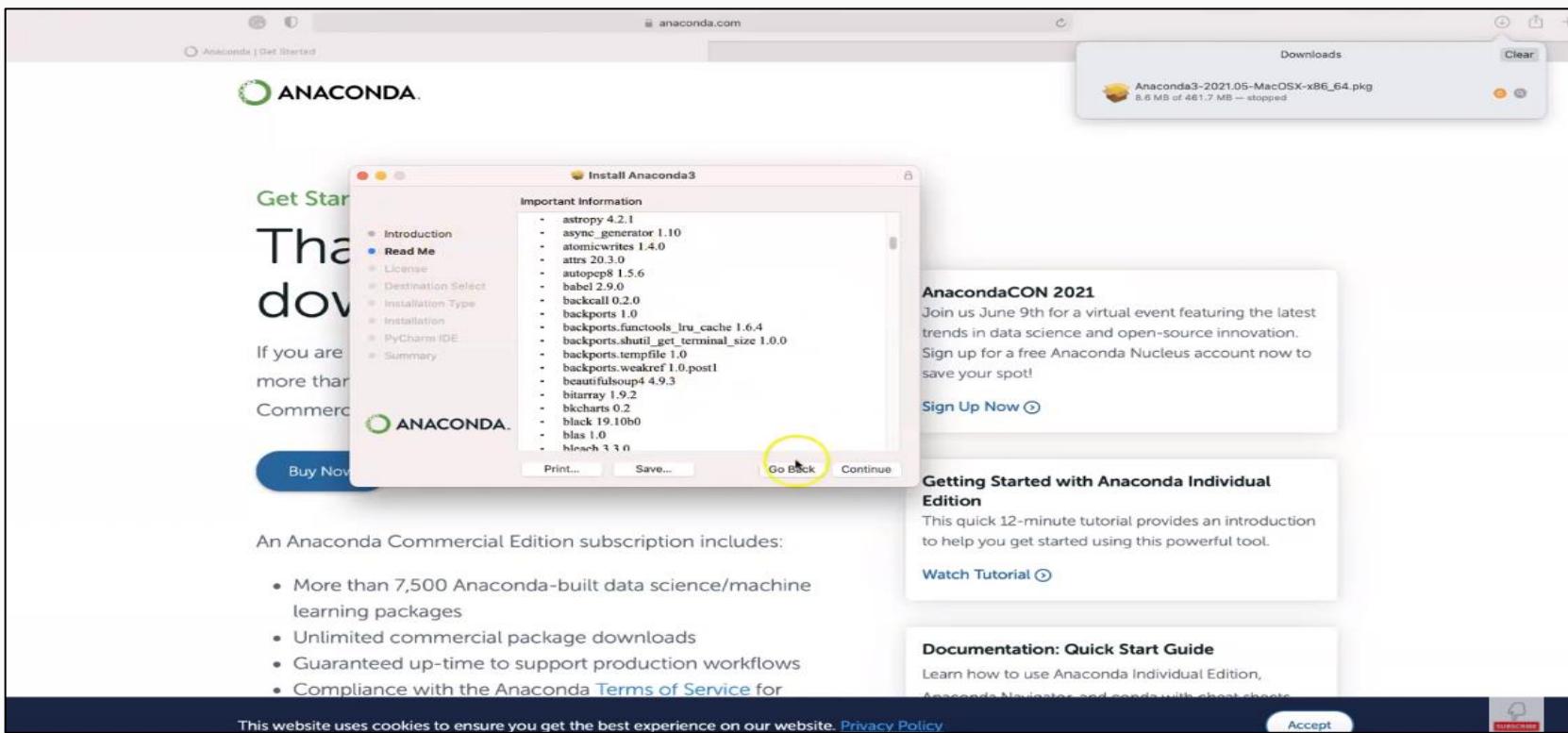
- Once download, launch the installation wizard by double clicking on the file.
- Click Continue.





Anaconda for Mac (5/14)

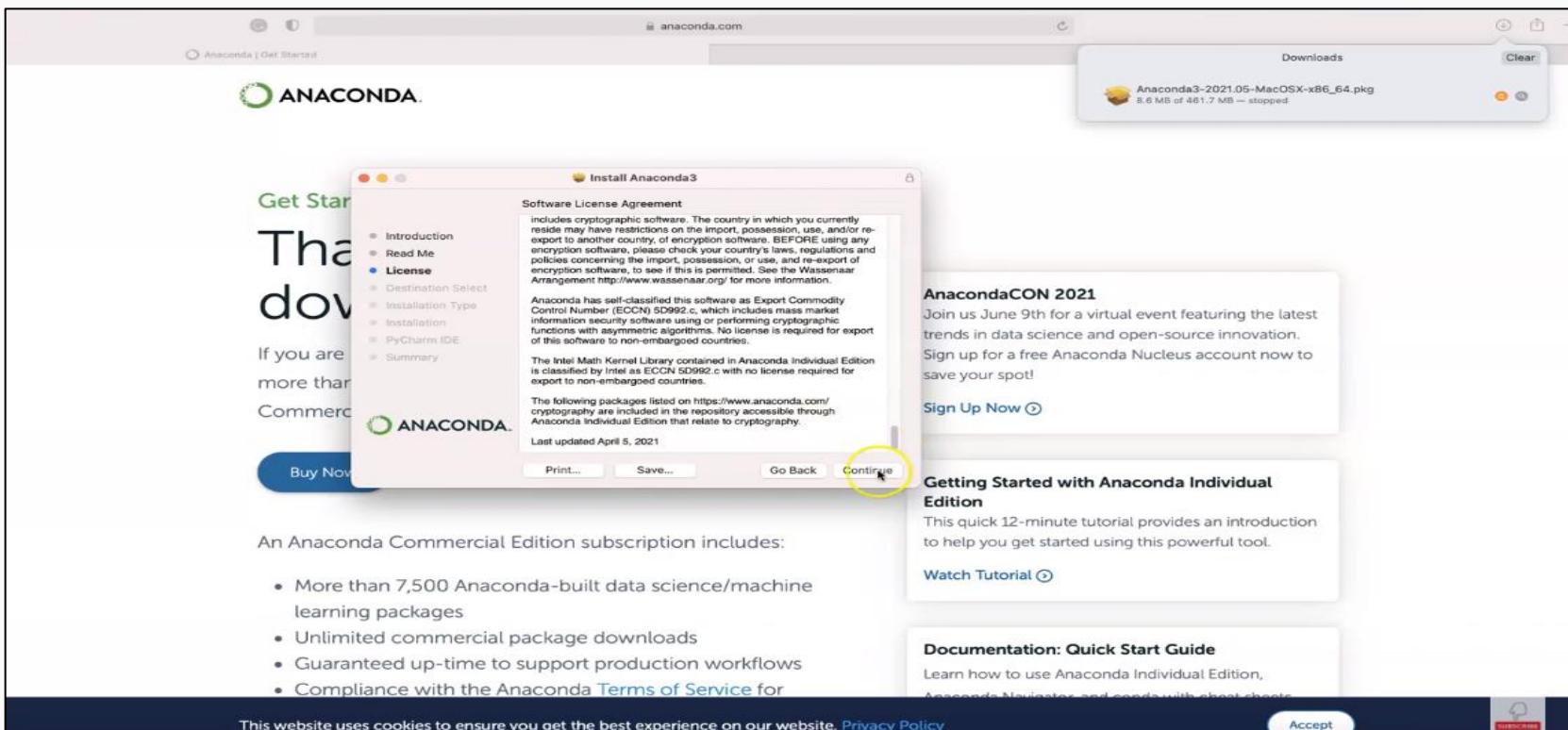
- Read the Read Me file and click continue.





Anaconda for Mac (6/14)

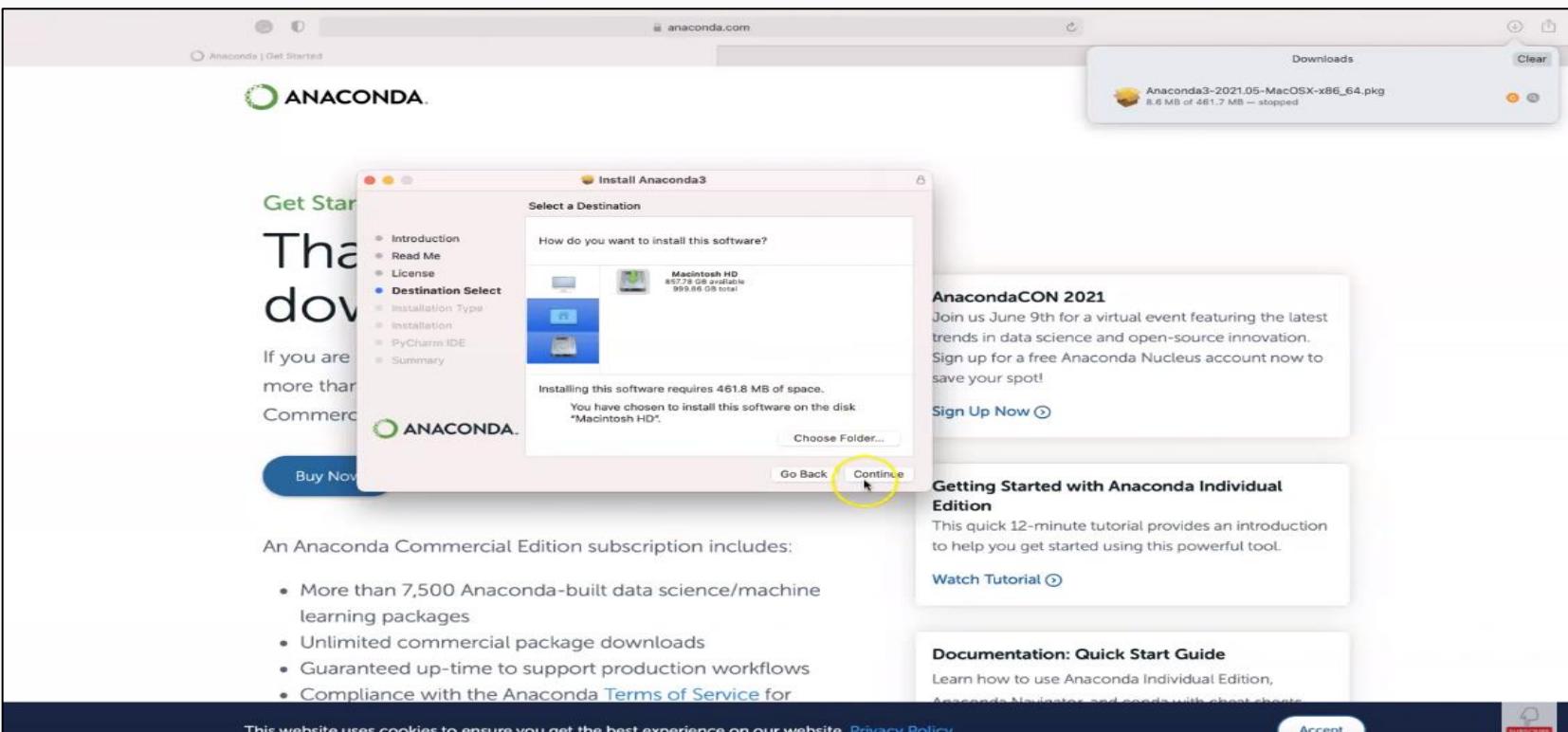
- Read the License Agreement and click continue.





Anaconda for Mac (7/14)

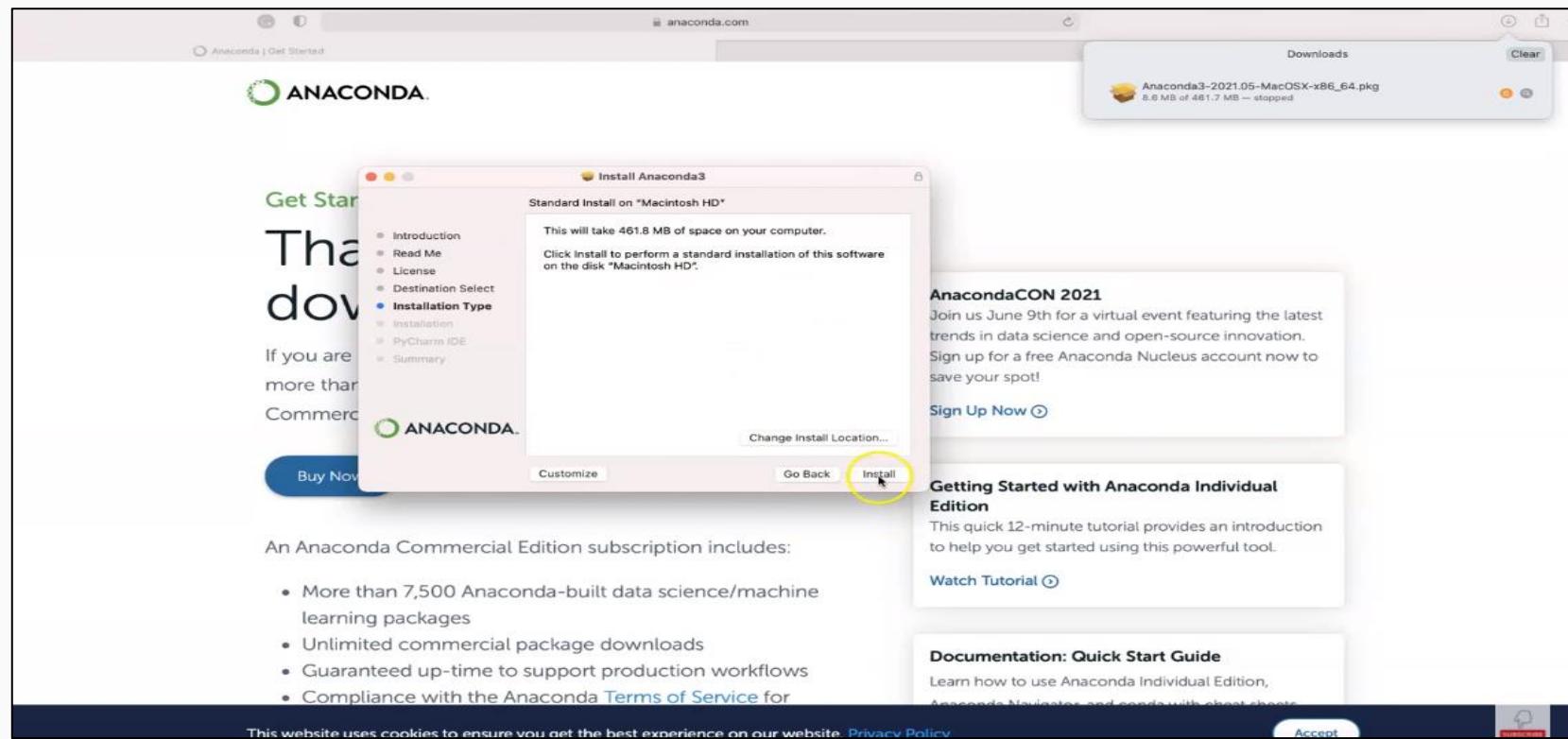
- Select the destination and click Continue.





Anaconda for Mac (8/14)

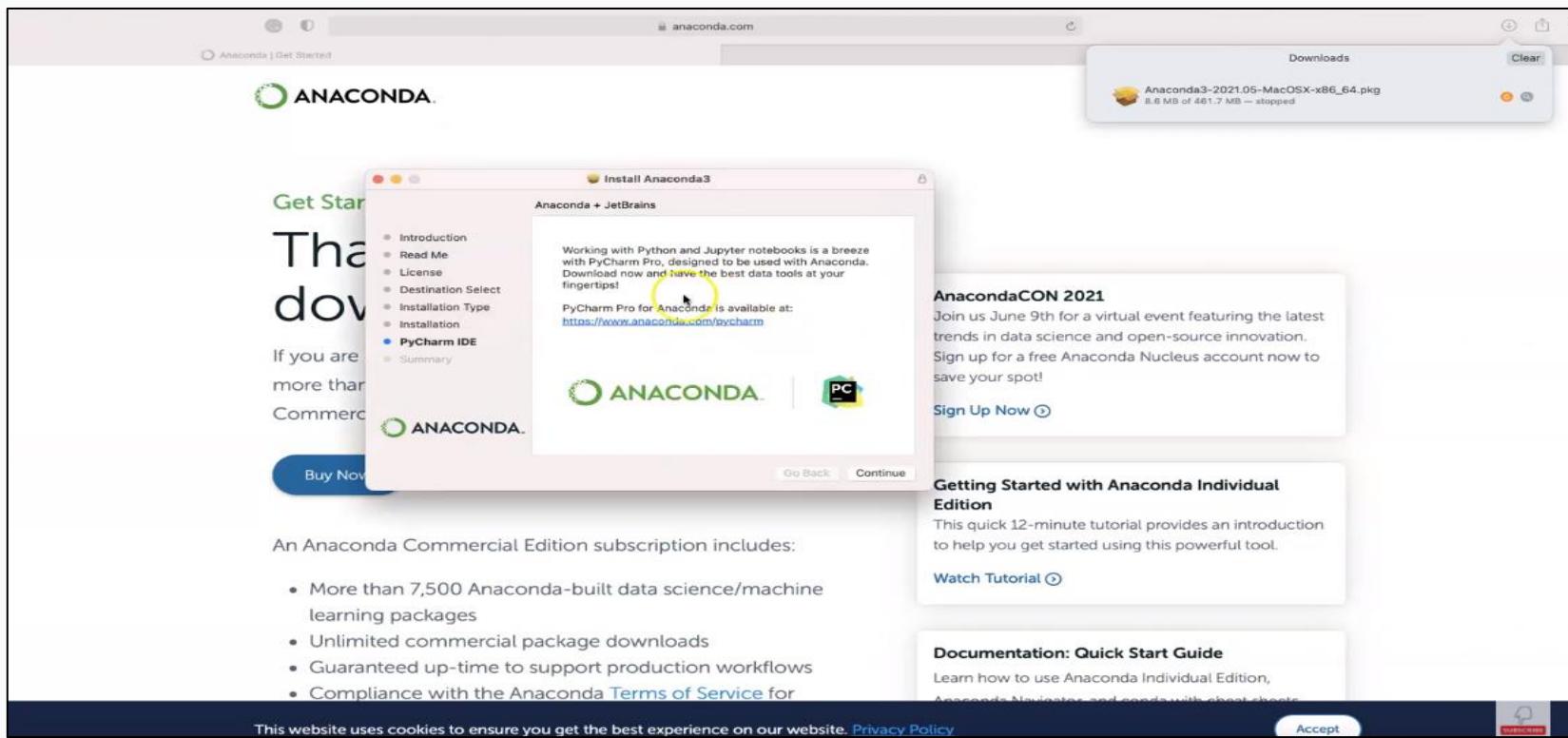
- Click Install.





Anaconda for Mac (9/14)

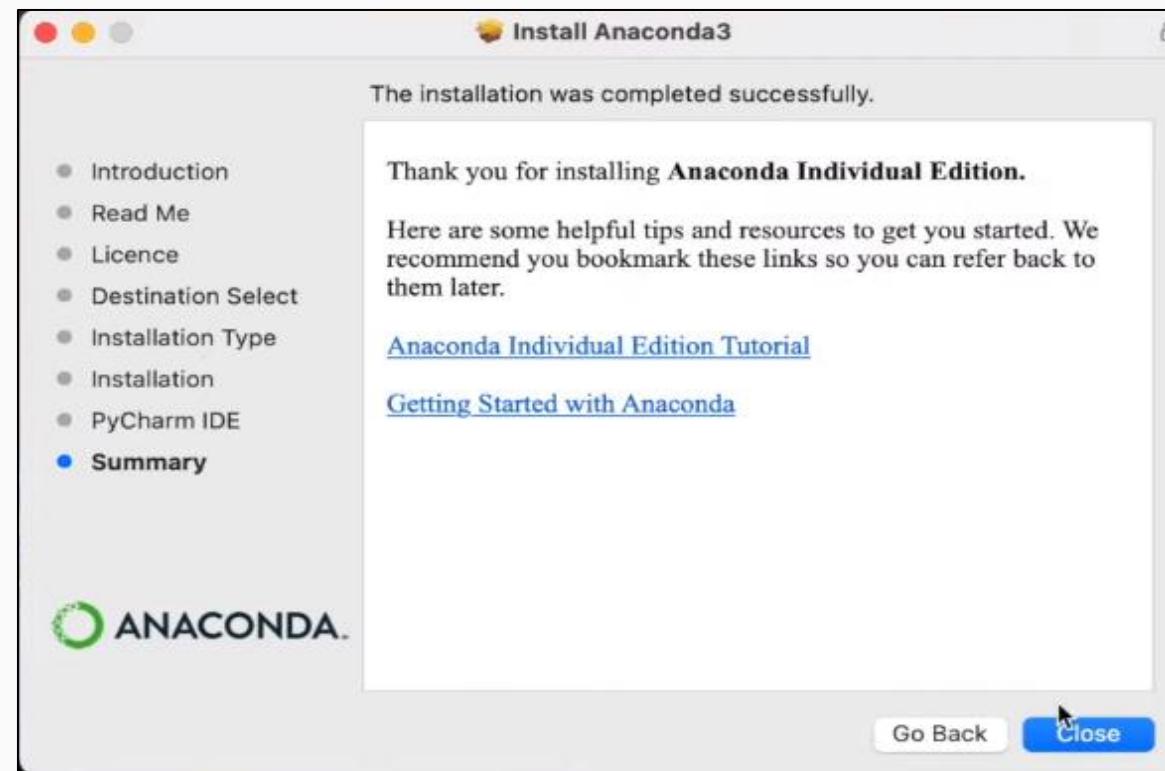
- Click Continue.





Anaconda for Mac (10/14)

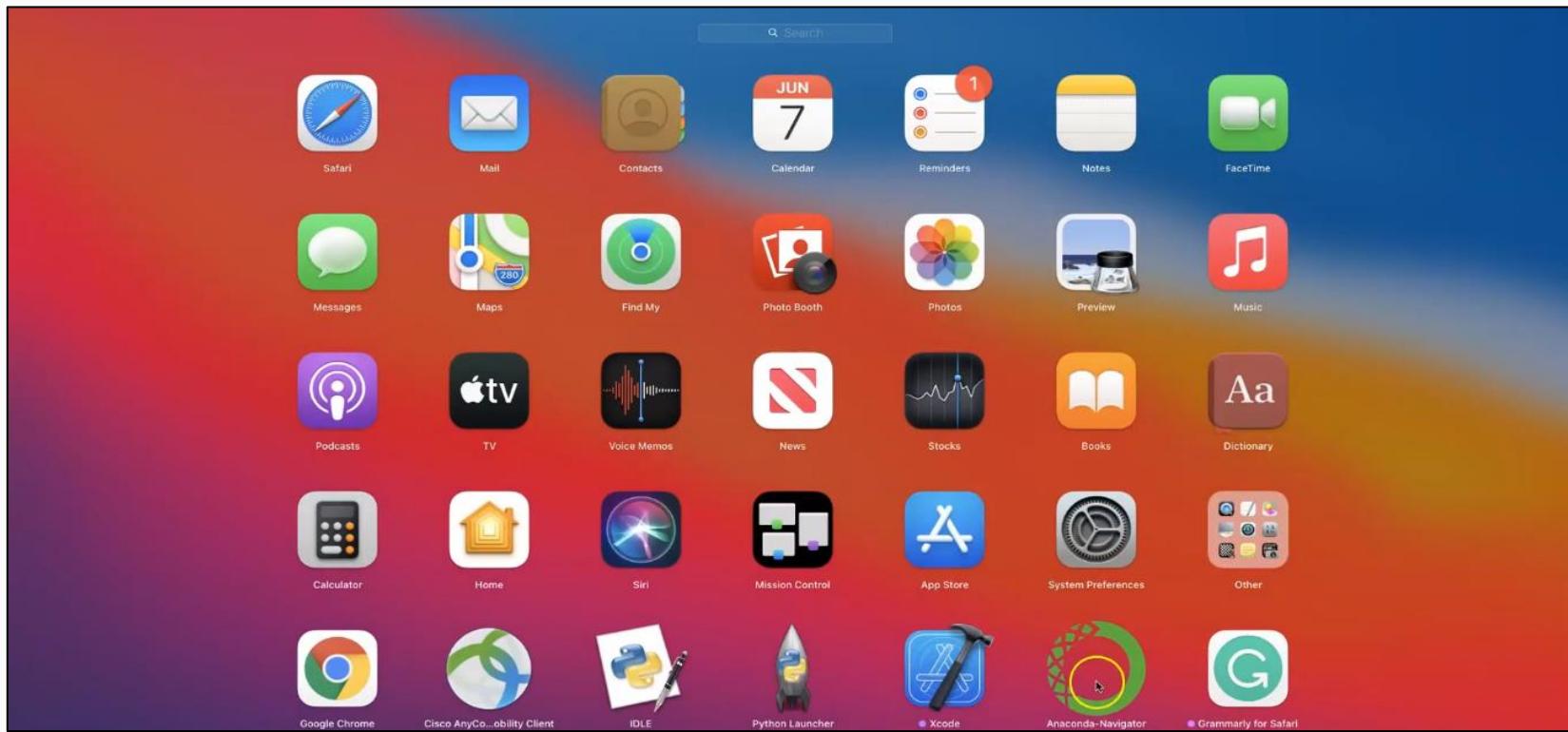
- Once the installation finishes, click Close.





Anaconda for Mac (11/14)

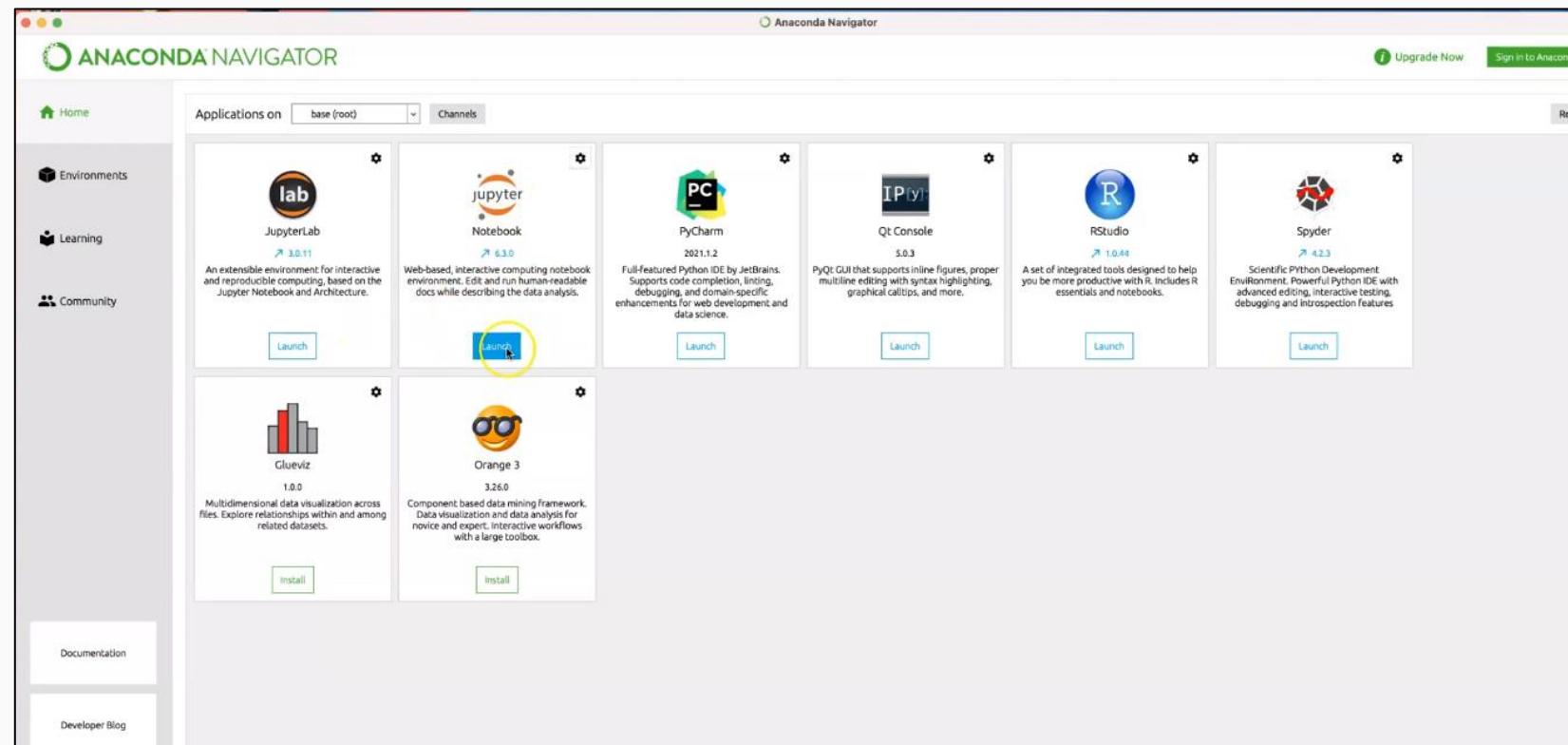
- Open your spotlight and click on Anaconda Navigator to launch it.





Anaconda for Mac (12/14)

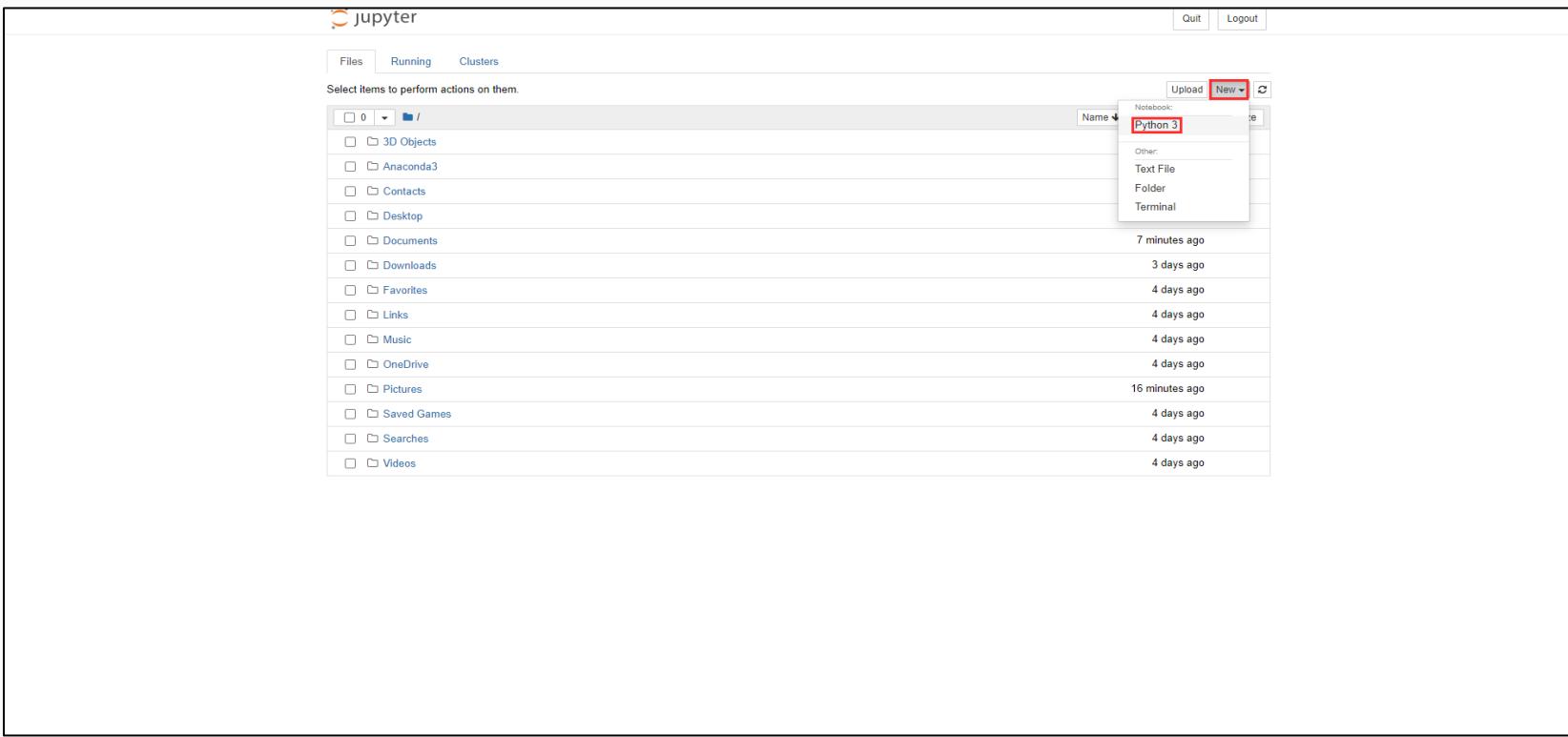
- Launch Jupyter Notebook. It will take sometime to launch.





Anaconda for Mac (13/14)

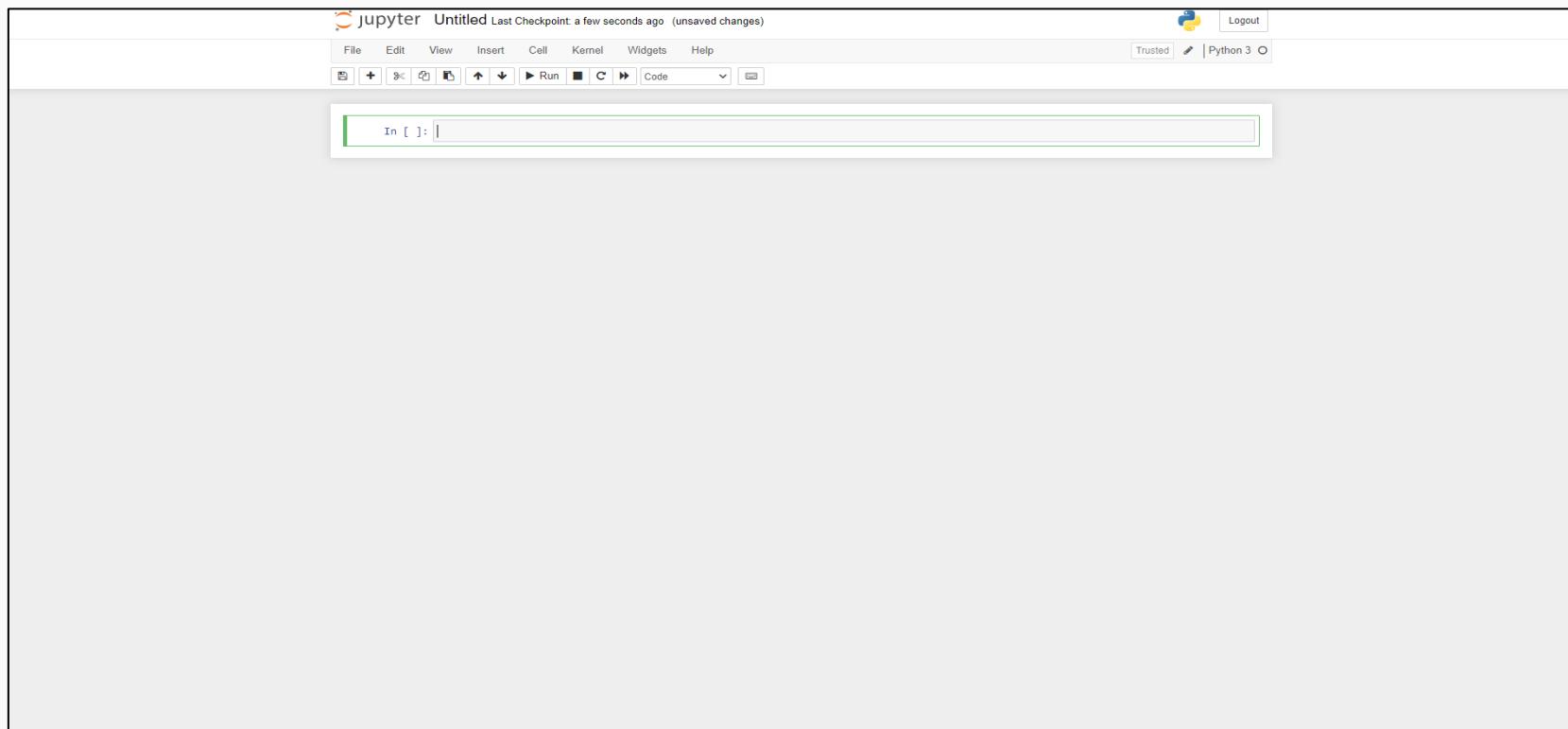
- This will launch Jupyter Notebook in your default browser.
- Click on New -> Python3 to create a new Notebook.





Anaconda for Mac (14/14)

- This is how a Notebook looks like.



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction

01

Python libraries for Machine Learning

02

Setting up Python

03

What is Jupyter?

04

Anaconda Installation for Windows OS

05

Anaconda Installation for Mac OS

06

Anaconda Installation for Ubuntu OS

07

Implementing Python in Jupyter

08

Managing Directories in Jupyter Notebook

09



Anaconda for Ubuntu (1/11)

- Open your terminal, paste the following command, and hit enter
sudo apt install libgl1-mesa-glx libegl1-mesa libxrandr2 libxrandr2 libxss1 libxcursor1 libcomposite1 libasound2 libxi6 libxtst6
- Type your password and hit Enter.

A screenshot of a terminal window titled "waqar@waqar-N750BU: ~". The window contains the following text:
waqar@waqar-N750BU:~\$ sudo apt install libgl1-mesa-glx libegl1-mesa libxrandr2 libxrandr2 libxss1 libxcursor1 libcomposite1 libasound2 libxi6 libxtst6
[sudo] password for waqar:

The terminal has a dark background with light-colored text. The cursor is located at the end of the password prompt line.



Anaconda for Ubuntu (2/11)

- Next give the following command and hit enter

```
wget https://repo.anaconda.com/archive/Anaconda3-2020.11-Linux-x86_64.sh -O  
~/Downloads/Anaconda3-2020.11-Linux-x86_64.sh
```

The screenshot shows a terminal window titled "waqar@waqar-N750BU: ~". The window displays the output of a command that first installs several Debian packages (libasound2, libasound2-data, libegl1-mesa, libegl1-mesa-glx) and then downloads the Anaconda installer script. The terminal shows the progress of the wget command, indicating a download speed of 7.34MB/s and an estimated time of 61 seconds remaining.

```
Preparing to unpack .../libasound2_1.2.2-2.1ubuntu2.4_amd64.deb ...
Unpacking libasound2:amd64 (1.2.2-2.1ubuntu2.4) over (1.2.2-2.1ubuntu2.3) ...
Preparing to unpack .../libasound2-data_1.2.2-2.1ubuntu2.4_all.deb ...
Unpacking libasound2-data (1.2.2-2.1ubuntu2.4) over (1.2.2-2.1ubuntu2.3) ...
Preparing to unpack .../libegl1-mesa_21.0.3-0ubuntu0.3~20.04.2_amd64.deb ...
Unpacking libegl1-mesa:amd64 (21.0.3-0ubuntu0.3~20.04.2) over (20.2.6-0ubuntu0.20.0
4.1) ...
Preparing to unpack .../libegl1-mesa-glx_21.0.3-0ubuntu0.3~20.04.2_amd64.deb ...
Unpacking libegl1-mesa-glx:amd64 (21.0.3-0ubuntu0.3~20.04.2) over (20.2.6-0ubuntu0.2
0.04.1) ...
Setting up libegl1-mesa:amd64 (21.0.3-0ubuntu0.3~20.04.2) ...
Setting up libasound2-data (1.2.2-2.1ubuntu2.4) ...
Setting up libegl1-mesa-glx:amd64 (21.0.3-0ubuntu0.3~20.04.2) ...
Setting up libasound2:amd64 (1.2.2-2.1ubuntu2.4) ...
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
waqar@waqar-N750BU:~$ wget https://repo.anaconda.com/archive/Anaconda3-2020.11-Linu
x-x86_64.sh -O ~/Downloads/Anaconda3-2020.11-Linux-x86_64.sh
--2021-10-03 12:51:56-- https://repo.anaconda.com/archive/Anaconda3-2020.11-Linux-
x86_64.sh
Resolving repo.anaconda.com (repo.anaconda.com)... 104.16.130.3, 104.16.131.3, 2606
:4700::6810:8303, ...
Connecting to repo.anaconda.com (repo.anaconda.com)|104.16.130.3|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 554535580 (529M) [application/x-sh]
Saving to: '/home/waqar/Downloads/Anaconda3-2020.11-Linux-x86_64.sh'

x86_64.sh          16%[==>]   85.64M  7.34MB/s    eta 61s
```



Anaconda for Ubuntu (3/11)

- Next give the following command and hit enter
cd ~/Downloads

The screenshot shows a terminal window with the following output:

```
waqar@waqar-N750BU: ~/Downloads
4.1) ...
Preparing to unpack .../libgl1-mesa-glx_21.0.3-0ubuntu0.3~20.04.2_amd64.deb ...
Unpacking libgl1-mesa-glx:amd64 (21.0.3-0ubuntu0.3~20.04.2) over (20.2.6-0ubuntu0.2
0.04.1) ...
Setting up libegl1-mesa:amd64 (21.0.3-0ubuntu0.3~20.04.2) ...
Setting up libasound2-data (1.2.2-2.1ubuntu2.4) ...
Setting up libgl1-mesa-glx:amd64 (21.0.3-0ubuntu0.3~20.04.2) ...
Setting up libasound2:amd64 (1.2.2-2.1ubuntu2.4) ...
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
waqar@waqar-N750BU:~$ wget https://repo.anaconda.com/archive/Anaconda3-2020.11-Linu
x-x86_64.sh -O ~/Downloads/Anaconda3-2020.11-Linux-x86_64.sh
--2021-10-03 12:51:56-- https://repo.anaconda.com/archive/Anaconda3-2020.11-Linux-
x86_64.sh
Resolving repo.anaconda.com (repo.anaconda.com)... 104.16.130.3, 104.16.131.3, 2606
:4700::6810:8303, ...
Connecting to repo.anaconda.com (repo.anaconda.com)|104.16.130.3|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 554535580 (529M) [application/x-sh]
Saving to: '/home/waqar/Downloads/Anaconda3-2020.11-Linux-x86_64.sh'

/home/waqar/Downloads/Anaconda3-2020.11-Linux-x86_64.sh
/home/waqar/Download 100%[=====] 528.85M 7.25MB/s in 75s
2021-10-03 12:53:12 (7.08 MB/s) - '/home/waqar/Downloads/Anaconda3-2020.11-Linux-x8
6_64.sh' saved [554535580/554535580]

waqar@waqar-N750BU:~$ cd ~/Downloads
waqar@waqar-N750BU:~/Downloads$
```



Anaconda for Ubuntu (4/11)

- Next give the following command and hit enter

```
sudo chmod +x Anaconda3-2020.11-Linux-x86_64.sh
```

The screenshot shows a terminal window with the following output:

```
Preparing to unpack .../libgl1-mesa-glx_21.0.3-0ubuntu0.3~20.04.2_amd64.deb ...
Unpacking libgl1-mesa-glx:amd64 (21.0.3-0ubuntu0.3~20.04.2) over (20.2.6-0ubuntu0.2
0.04.1) ...
Setting up libegl1-mesa:amd64 (21.0.3-0ubuntu0.3~20.04.2) ...
Setting up libasound2-data (1.2.2-2.1ubuntu2.4) ...
Setting up libgl1-mesa-glx:amd64 (21.0.3-0ubuntu0.3~20.04.2) ...
Setting up libasound2:amd64 (1.2.2-2.1ubuntu2.4) ...
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
waqar@waqar-N750BU:~$ wget https://repo.anaconda.com/archive/Anaconda3-2020.11-Linu
x-x86_64.sh -O ~/Downloads/Anaconda3-2020.11-Linux-x86_64.sh
--2021-10-03 12:51:56-- https://repo.anaconda.com/archive/Anaconda3-2020.11-Linux-
x86_64.sh
Resolving repo.anaconda.com (repo.anaconda.com)... 104.16.130.3, 104.16.131.3, 2606
:4700::6810:8303, ...
Connecting to repo.anaconda.com (repo.anaconda.com)|104.16.130.3|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 554535580 (529M) [application/x-sh]
Saving to: '/home/waqar/Downloads/Anaconda3-2020.11-Linux-x86_64.sh'

/home/waqar/Download 100%[=====] 528.85M 7.25MB/s   in 75s

2021-10-03 12:53:12 (7.08 MB/s) - '/home/waqar/Downloads/Anaconda3-2020.11-Linux-x8
6_64.sh' saved [554535580/554535580]

waqar@waqar-N750BU:~$ cd ~/Downloads
waqar@waqar-N750BU:~/Downloads$ sudo chmod +x Anaconda3-2020.11-Linux-x86_64.sh
waqar@waqar-N750BU:~/Downloads$
```



Anaconda for Ubuntu (5/11)

- Next give the following command and hit enter
./Anaconda3-2020.11-Linux-x86_64.sh
- When prompted, press enter to continue.

The screenshot shows a terminal window titled "waqar@waqar-N750BU: ~/Downloads". The terminal output is as follows:

```
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
waqar@waqar-N750BU:~$ wget https://repo.anaconda.com/archive/Anaconda3-2020.11-Linu
x-x86_64.sh -O ~/Downloads/Anaconda3-2020.11-Linux-x86_64.sh
--2021-10-03 12:51:56-- https://repo.anaconda.com/archive/Anaconda3-2020.11-Linux-
x86_64.sh
Resolving repo.anaconda.com (repo.anaconda.com)... 104.16.130.3, 104.16.131.3, 2606
:4700::6810:8303, ...
Connecting to repo.anaconda.com (repo.anaconda.com)|104.16.130.3|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 554535580 (529M) [application/x-sh]
Saving to: '/home/waqar/Downloads/Anaconda3-2020.11-Linux-x86_64.sh'

/home/waqar/Download 100%[=====] 528.85M 7.25MB/s   in 75s

2021-10-03 12:53:12 (7.08 MB/s) - '/home/waqar/Downloads/Anaconda3-2020.11-Linux-x8
6_64.sh' saved [554535580/554535580]

waqar@waqar-N750BU:~$ cd ~/Downloads
waqar@waqar-N750BU:~/Downloads$ sudo chmod +x Anaconda3-2020.11-Linux-x86_64.sh
waqar@waqar-N750BU:~/Downloads$ ./Anaconda3-2020.11-Linux-x86_64.sh

Welcome to Anaconda3 2020.11

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



Anaconda for Ubuntu (6/11)

- Press the Enter key to scroll through the License Agreement.
- After reading the Agreement, type yes in the prompt and hit enter.

A network authentication protocol designed to provide strong authentication for client/server applications by using secret-key cryptography.

cryptography
A Python library which exposes cryptographic recipes and primitives.

pycryptodome
A fork of PyCrypto. It is a self-contained Python package of low-level cryptographic primitives.

pycryptodomex
A stand-alone version of pycryptodome.

libsodium
A software library for encryption, decryption, signatures, password hashing and more.

pynacl
A Python binding to the Networking and Cryptography library, a crypto library with the stated goal of improving usability, security and speed.

Last updated September 28, 2020

Do you accept the license terms? [yes|no]
[no] >>> yes



Anaconda for Ubuntu (7/11)

- Anaconda installer will ask you where do you want to install Anaconda. We suggest pressing Enter key to install it in the home directory.
- After sometime, Anaconda will be installed on your machine.

The screenshot shows a terminal window titled "waqar@waqar-N750BU: ~/Downloads". The window contains the following text:

```
A stand-alone version of pycryptodome.  
libsodium  
A software library for encryption, decryption, signatures, password hashing and more.  
pynacl  
A Python binding to the Networking and Cryptography library, a crypto library with the stated goal of improving usability, security and speed.  
  
Last updated September 28, 2020  
  
Do you accept the license terms? [yes|no]  
[no] >>> yes  
  
Anaconda3 will now be installed into this location:  
/home/waqar/anaconda3  
  
- Press ENTER to confirm the location  
- Press CTRL-C to abort the installation  
- Or specify a different location below  
  
[/home/waqar/anaconda3] >>>  
PREFIX=/home/waqar/anaconda3
```



Anaconda for Ubuntu (8/11)

- Once installation is finished, you will be asked “Do you wish the installer to initialize Anaconda3 by running conda init?” type “yes” and hit Enter. Setup will finish the installation process.

The screenshot shows a terminal window titled "waqar@waqar-N750BU: ~/Downloads". The window displays a list of packages installed, followed by transaction preparation and execution messages, and finally a question about initializing Anaconda3.

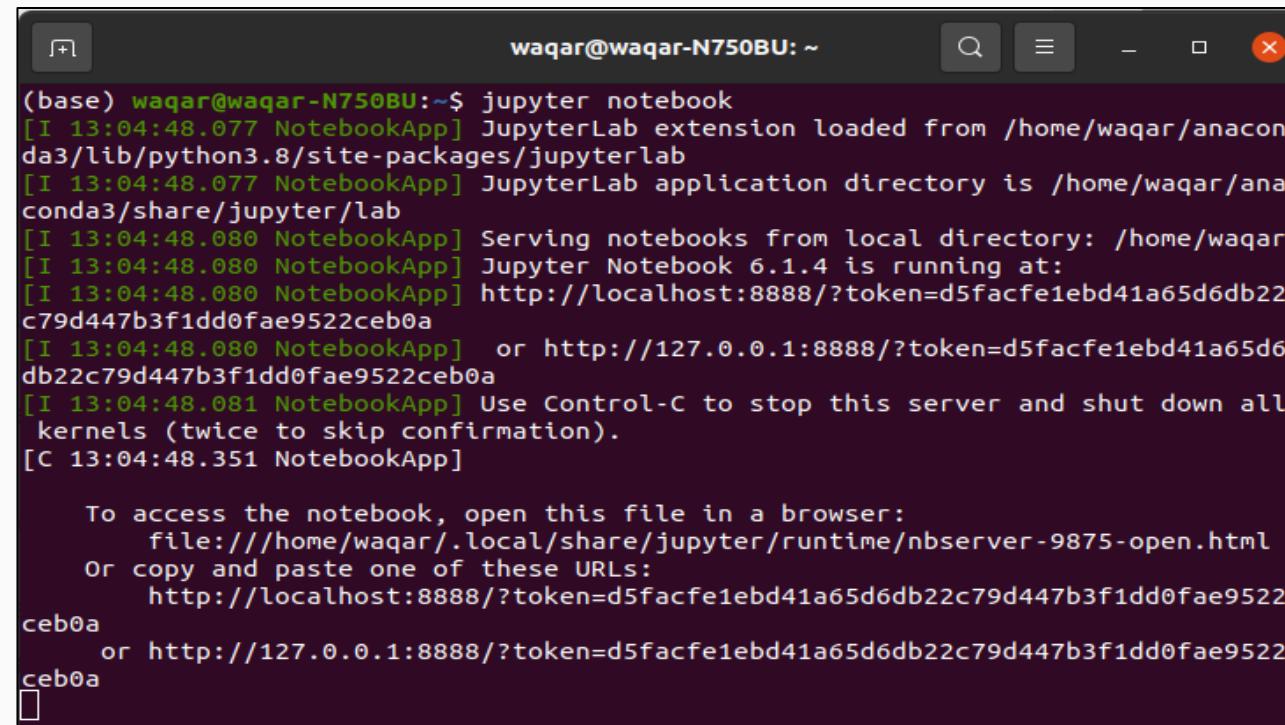
```
waqar@waqar-N750BU: ~/Downloads
wheel                  pkgs/main/noarch::wheel-0.35.1-py_0
widgetsnbextension    pkgs/main/linux-64::widgetsnbextension-3.5.1-py38_0
wrapt                  pkgs/main/linux-64::wrapt-1.11.2-py38h7b6447c_0
wurlitzer              pkgs/main/linux-64::wurlitzer-2.0.1-py38_0
xlrd                   pkgs/main/noarch::xlrd-1.2.0-py_0
xlsxwriter             pkgs/main/noarch::xlsxwriter-1.3.7-py_0
xlwt                   pkgs/main/linux-64::xlwt-1.3.0-py38_0
xmltodict              pkgs/main/noarch::xmltodict-0.12.0-py_0
xz                     pkgs/main/linux-64::xz-5.2.5-h7b6447c_0
yaml                   pkgs/main/linux-64::yaml-0.2.5-h7b6447c_0
yapf                   pkgs/main/noarch::yapf-0.30.0-py_0
zeromq                 pkgs/main/linux-64::zeromq-4.3.3-he6710b0_3
zict                   pkgs/main/noarch::zict-2.0.0-py_0
zipp                   pkgs/main/noarch::zipp-3.4.0-pyhd3eb1b0_0
zlib                   pkgs/main/linux-64::zlib-1.2.11-h7b6447c_3
zope                   pkgs/main/linux-64::zope-1.0-py38_1
zope.event              pkgs/main/linux-64::zope.event-4.5.0-py38_0
zope.interface          pkgs/main/linux-64::zope.interface-5.1.2-py38h7b6447c_0
zstd                   pkgs/main/linux-64::zstd-1.4.5-h9ceee32_0

Preparing transaction: done
Executing transaction: done
installation finished.
Do you wish the installer to initialize Anaconda3
by running conda init? [yes|no]
[no] >>> yes
```



Anaconda for Ubuntu (9/11)

- To launch Jupyter Notebook, restart your terminal.
- You will see (base) written before your home directory name.
- Type “jupyter notebook” without the quotation marks and hit enter.
- After a few minutes, Jupyter Notebook will run on your default browser.



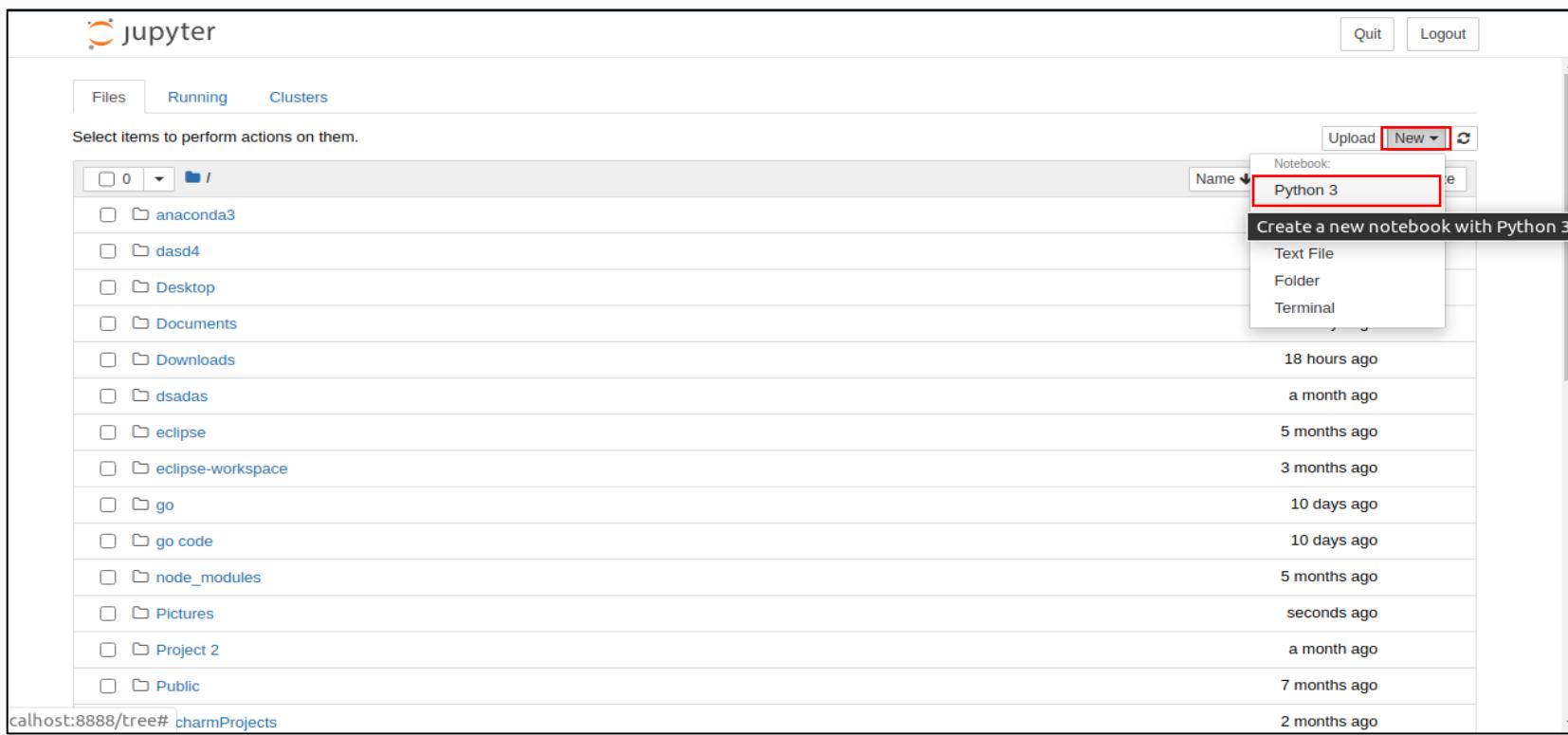
```
waqar@waqar-N750BU: ~
(base) waqar@waqar-N750BU:~$ jupyter notebook
[I 13:04:48.077 NotebookApp] JupyterLab extension loaded from /home/waqar/anaconda3/lib/python3.8/site-packages/jupyterlab
[I 13:04:48.077 NotebookApp] JupyterLab application directory is /home/waqar/anaconda3/share/jupyter/lab
[I 13:04:48.080 NotebookApp] Serving notebooks from local directory: /home/waqar
[I 13:04:48.080 NotebookApp] Jupyter Notebook 6.1.4 is running at:
[I 13:04:48.080 NotebookApp] http://localhost:8888/?token=d5facfe1ebd41a65d6db22c79d447b3f1dd0fae9522ceb0a
[I 13:04:48.080 NotebookApp] or http://127.0.0.1:8888/?token=d5facfe1ebd41a65d6db22c79d447b3f1dd0fae9522ceb0a
[I 13:04:48.081 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 13:04:48.351 NotebookApp]

To access the notebook, open this file in a browser:
    file:///home/waqar/.local/share/jupyter/runtime/nbserver-9875-open.html
Or copy and paste one of these URLs:
    http://localhost:8888/?token=d5facfe1ebd41a65d6db22c79d447b3f1dd0fae9522ceb0a
    or http://127.0.0.1:8888/?token=d5facfe1ebd41a65d6db22c79d447b3f1dd0fae9522ceb0a
```



Anaconda for Ubuntu (10/11)

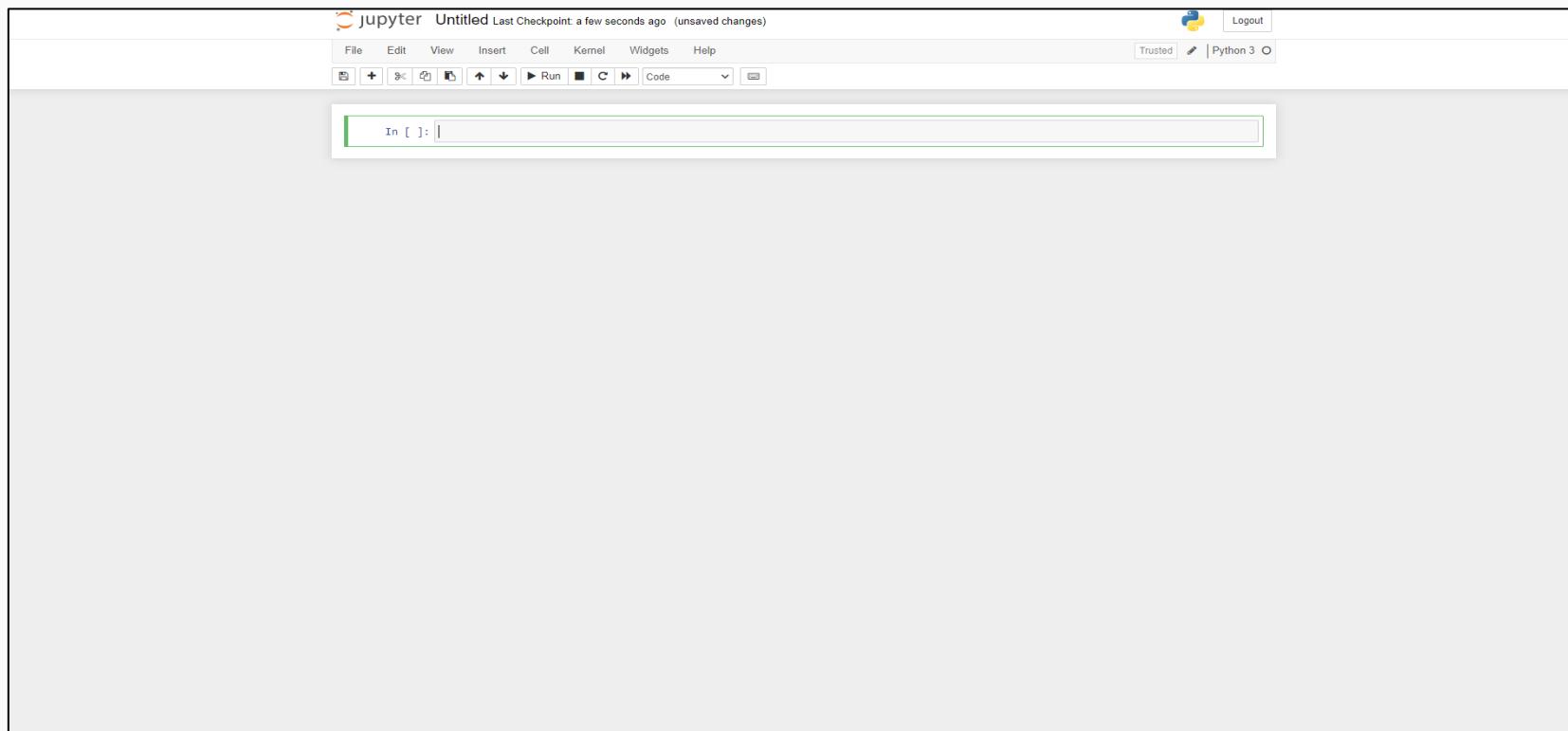
- Click on New -> Python3 to create a new Notebook.





Anaconda for Ubuntu (11/11)

- This is how a Notebook looks like.



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction

01

Python libraries for Machine Learning

02

Setting up Python

03

What is Jupyter?

04

Anaconda Installation for Windows OS

05

Anaconda Installation for Mac OS

06

Anaconda Installation for Ubuntu OS

07

Implementing Python in Jupyter

08

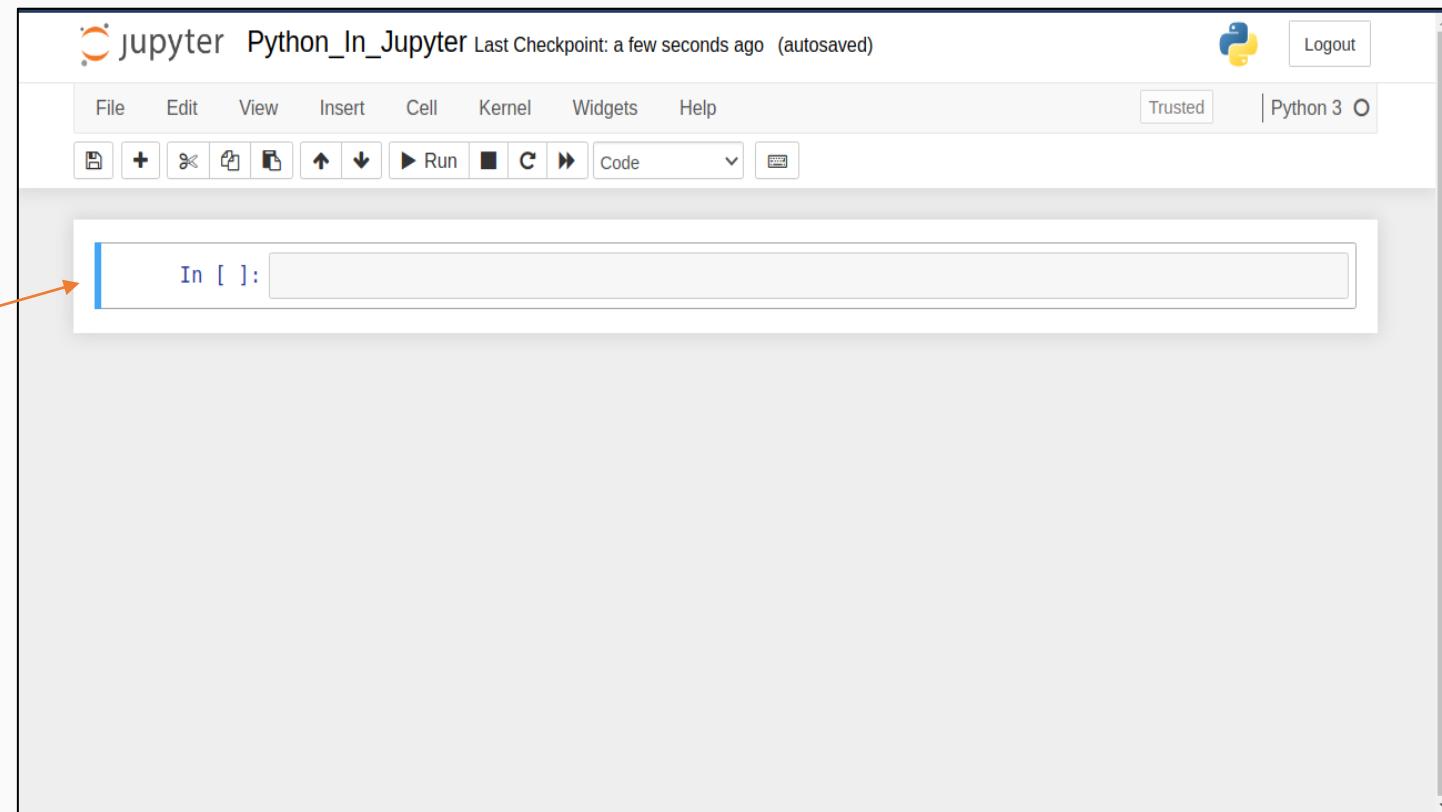
Managing Directories in Jupyter Notebook

09



Implementing Python in Jupyter Notebook (1/2)

Upon launching Jupyter Notebook, this is what you will see!



This is where we write our code, called a cell. You can have as many of them as you want.



Implementing Python in Jupyter Notebook (2/2)

- Write your Python code inside a cell.
- Select the cell by clicking on it and then click on the Run button to execute the code.
- Output of a cell is produced right below it.

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** jupyter Python_In_Jupyter Last Checkpoint: 2 minutes ago (unsaved changes) Logout
- Toolbar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help. The "Cell" button is highlighted.
- Status Bar:** Trusted | Python 3
- Cell Content:** In [1]: `print("Hello World")`
- Output:** Hello World
- Annotations:** A red arrow labeled "output" points to the "Hello World" text in the output cell. A mouse cursor is hovering over the "Run" button in the toolbar.

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction

01

Python libraries for Machine Learning

02

Setting up Python

03

What is Jupyter?

04

Anaconda Installation for Windows OS

05

Anaconda Installation for Mac OS

06

Anaconda Installation for Ubuntu OS

07

Implementing Python in Jupyter

08

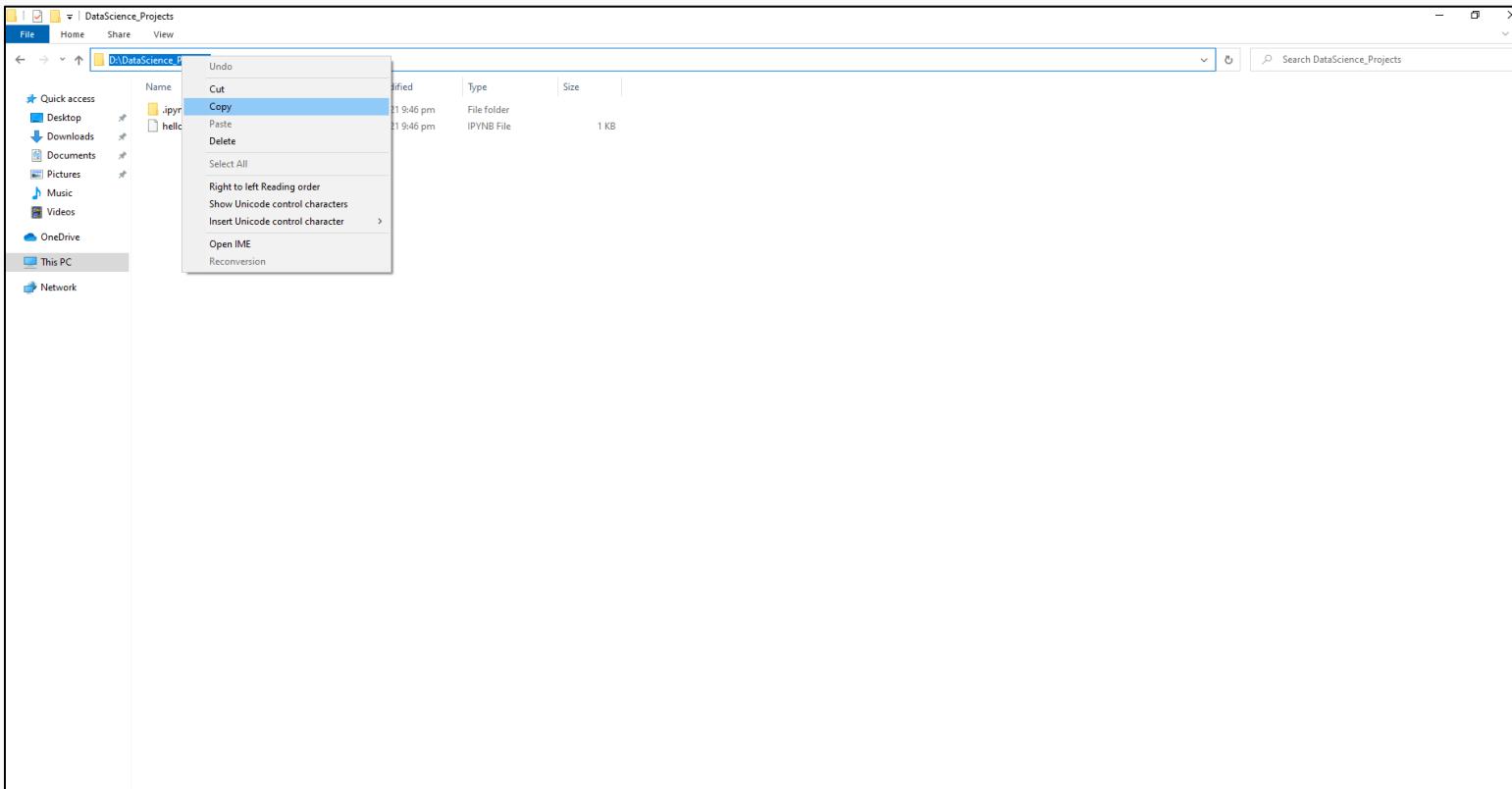
Managing Directories in Jupyter Notebook

09



Managing Directories in Jupyter Notebook – Windows (1/7)

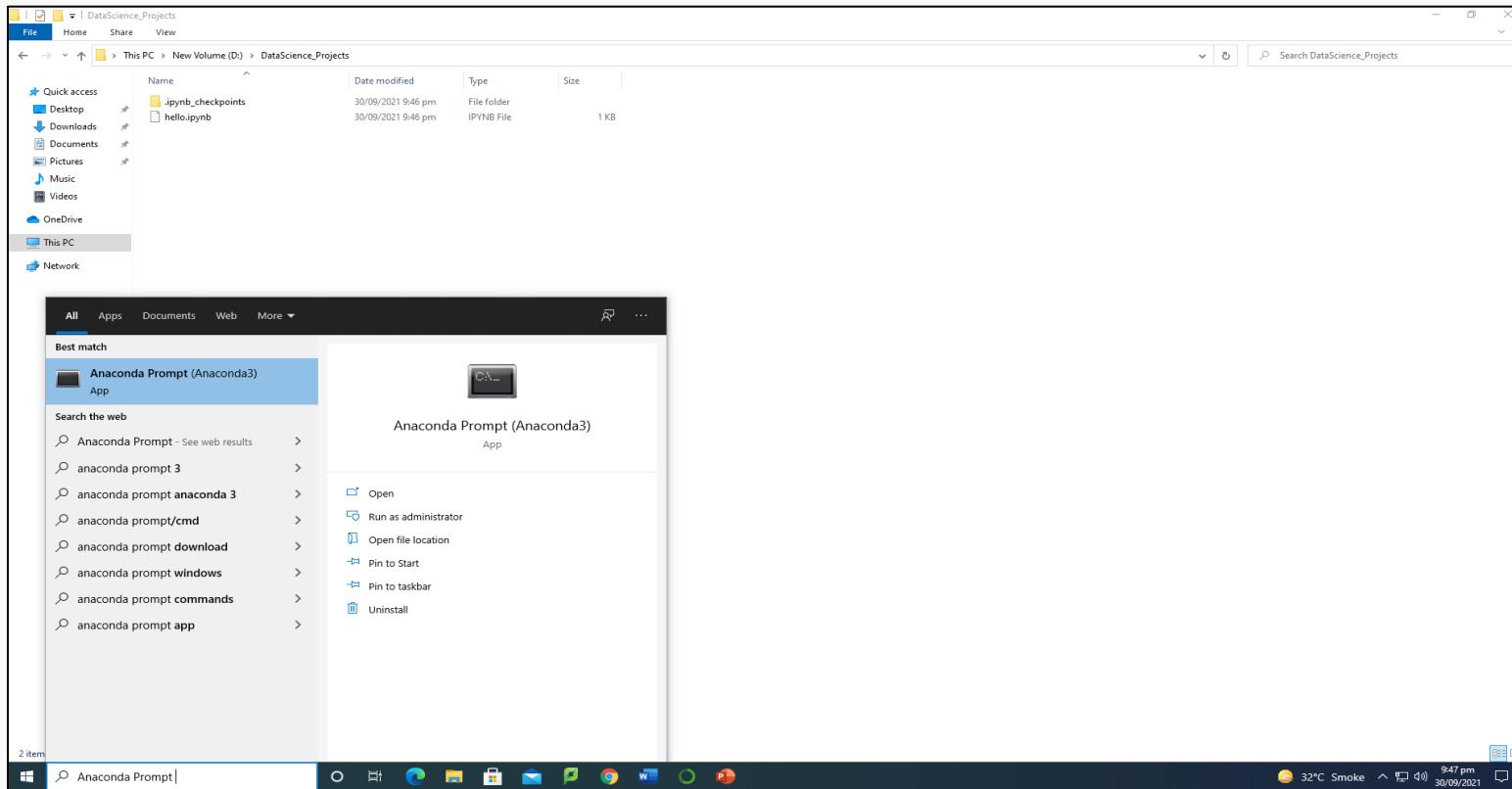
- To open Jupyter Notebook in a particular directory/folder in windows, copy the path of the directory.





Managing Directories in Jupyter Notebook – Windows (2/7)

- In your search bar type Anaconda Prompt and launch it
 - Anaconda Prompt is the CLI of Anaconda





Managing Directories in Jupyter Notebook – Windows (3/7)

- Anaconda Prompt opens up in your C drive. Since we want to move to the D drive, we type d: and hit enter.

The screenshot shows a Windows-style terminal window titled "Anaconda Prompt (Anaconda3)". The command prompt "(base)" is followed by the current directory path "C:\Users\22100199>". A user has typed "d:" and pressed enter, resulting in the new directory path "D:\>".



Managing Directories in Jupyter Notebook – Windows (4/7)

- Next, to switch to the desired directory we type **cd** and then **paste the path of the directory**.

A screenshot of the Anaconda Prompt window titled "Anaconda Prompt (Anaconda3)". The window has a black background and white text. It shows the following command sequence:

```
(base) C:\Users\22100199>d:  
(base) D:\>cd D:\DataScience_Projects  
(base) D:\DataScience_Projects>
```



Managing Directories in Jupyter Notebook – Windows (5/7)

- Finally, we launch Jupyter Notebook by typing jupyter notebook.
- Jupyter Notebook takes a while to open.

The screenshot shows a terminal window titled "Anaconda Prompt (Anaconda3) - jupyter notebook". The command entered was "jupyter notebook". The output indicates that the port 8888 is already in use, so it tries another port. It also shows that the JupyterLab extension is loaded and provides the URL for the Jupyter Notebook application, which is running at port 8890. The command "Use Control-C to stop this server and shut down all kernels (twice to skip confirmation)." is displayed at the end.

```
Anaconda Prompt (Anaconda3) - jupyter notebook

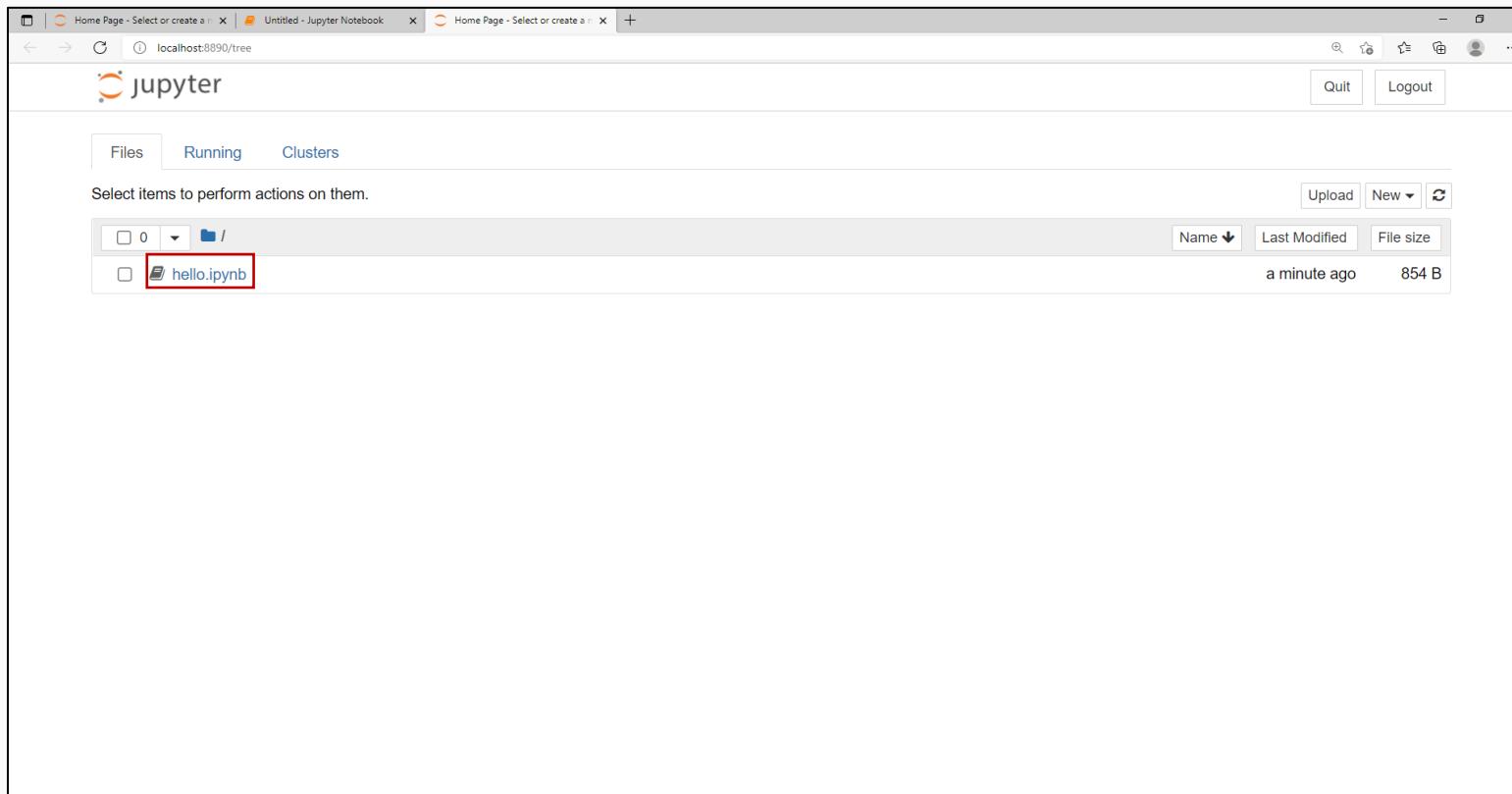
(base) C:\Users\22100199>d:
(base) D:\>cd D:\DataScience_Projects

(base) D:\DataScience_Projects>jupyter notebook
[I 21:47:44.481 NotebookApp] The port 8888 is already in use, trying another port.
[I 21:47:44.481 NotebookApp] The port 8889 is already in use, trying another port.
[I 2021-09-30 21:47:45.068 LabApp] JupyterLab extension loaded from C:\Users\22100199\Anaconda3\lib\site-packages\jupyterlab
[I 2021-09-30 21:47:45.068 LabApp] JupyterLab application directory is C:\Users\22100199\Anaconda3\share\jupyter\lab
[I 21:47:45.068 NotebookApp] Serving notebooks from local directory: D:\DataScience_Projects
[I 21:47:45.068 NotebookApp] Jupyter Notebook 6.3.0 is running at:
[I 21:47:45.068 NotebookApp] http://localhost:8890/?token=14d64985db6992c640f1d073c32482fc04aa213dc8262852
[I 21:47:45.068 NotebookApp] or http://127.0.0.1:8890/?token=14d64985db6992c640f1d073c32482fc04aa213dc8262852
[I 21:47:45.068 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
```



Managing Directories in Jupyter Notebook – Windows (6/7)

- Once it opens, we can either open an existing notebook or create a new one.
- In this case we open the existing notebook named hello.ipynb.





Managing Directories in Jupyter Notebook – Windows (7/7)

- This is how hello.ipynb looks like.

The screenshot shows a Jupyter Notebook interface running in a web browser. The title bar indicates the notebook is titled "hello". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The toolbar below the menu bar contains icons for file operations, cell creation, and execution. A status bar at the bottom shows "Not Trusted" and "Python 3". The main workspace displays a single code cell:

```
In [1]: print("Hello World")
Hello World
```

The output "Hello World" is displayed in the cell's output area.



Managing Directories in Jupyter Notebook – Mac (1/3)

- To open Jupyter Notebook in a particular directory/folder in Mac, open that directory in terminal.
- Type ‘jupyter notebook’ without the quotation marks and hit Enter.

A screenshot of a Mac OS X terminal window. The window has red, yellow, and green close buttons at the top left. The title bar is dark grey. The main area of the terminal shows the following text:

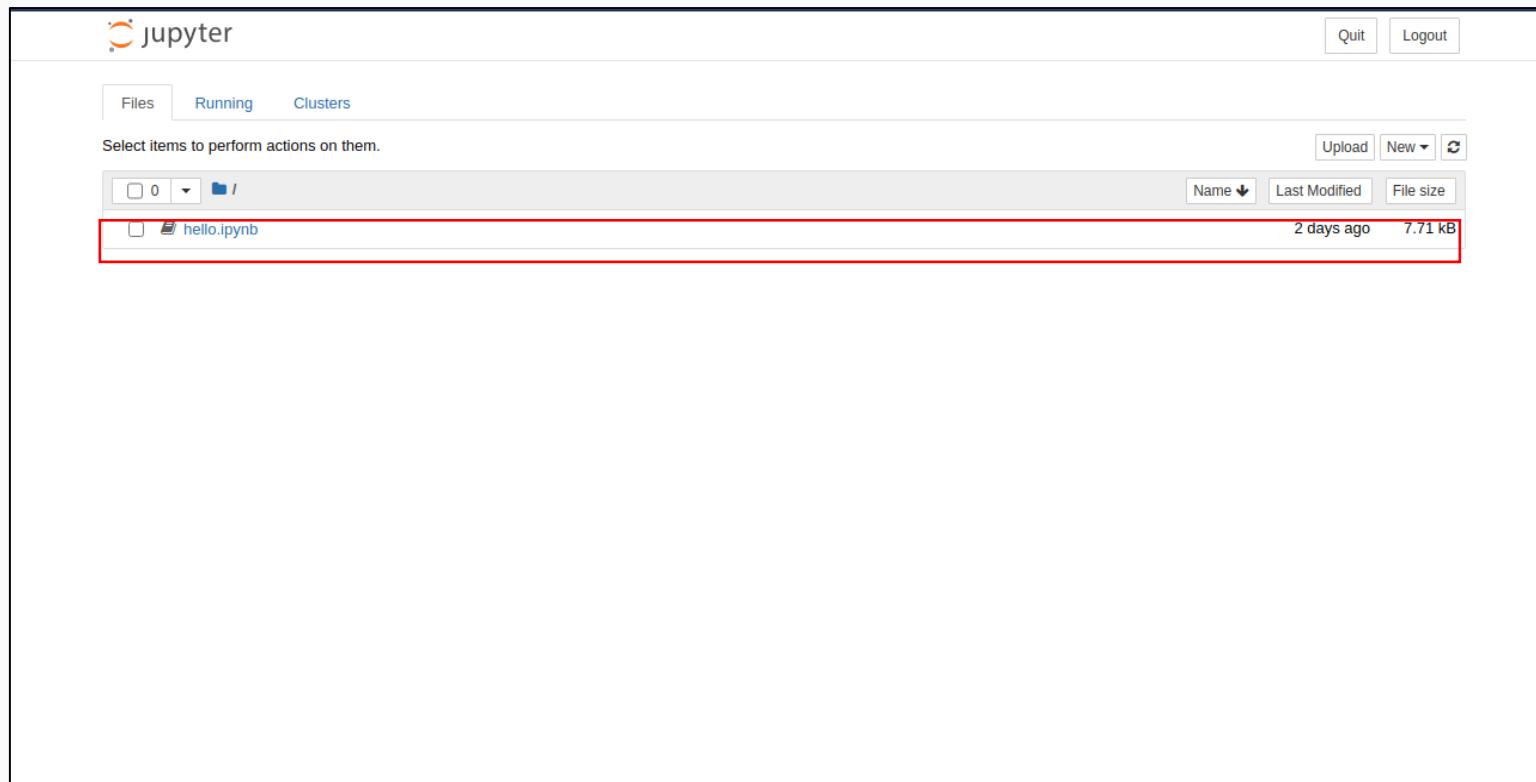
```
Last login: Fri Mar 27 23:32:59 on ttys000
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base)      $ jupyter notebook
```

The cursor is visible at the end of the command line, indicating it is ready for input.



Managing Directories in Jupyter Notebook – Mac (2/3)

- Jupyter Notebook will launch in your browser.
- You can create a new Notebook or open an existing one.
- Let's open this 'hello.ipynb' file.





Managing Directories in Jupyter Notebook – Mac (3/3)

- ‘hello.ipynb’ contains a simple print statement.

The screenshot shows a Jupyter Notebook interface titled "jupyter hello". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The toolbar below the menu has icons for file operations like Open, Save, and New, along with Run, Cell, and Kernel selection buttons. A status bar at the bottom indicates "Trusted" and "Python 3".

The notebook content area displays a single code cell:

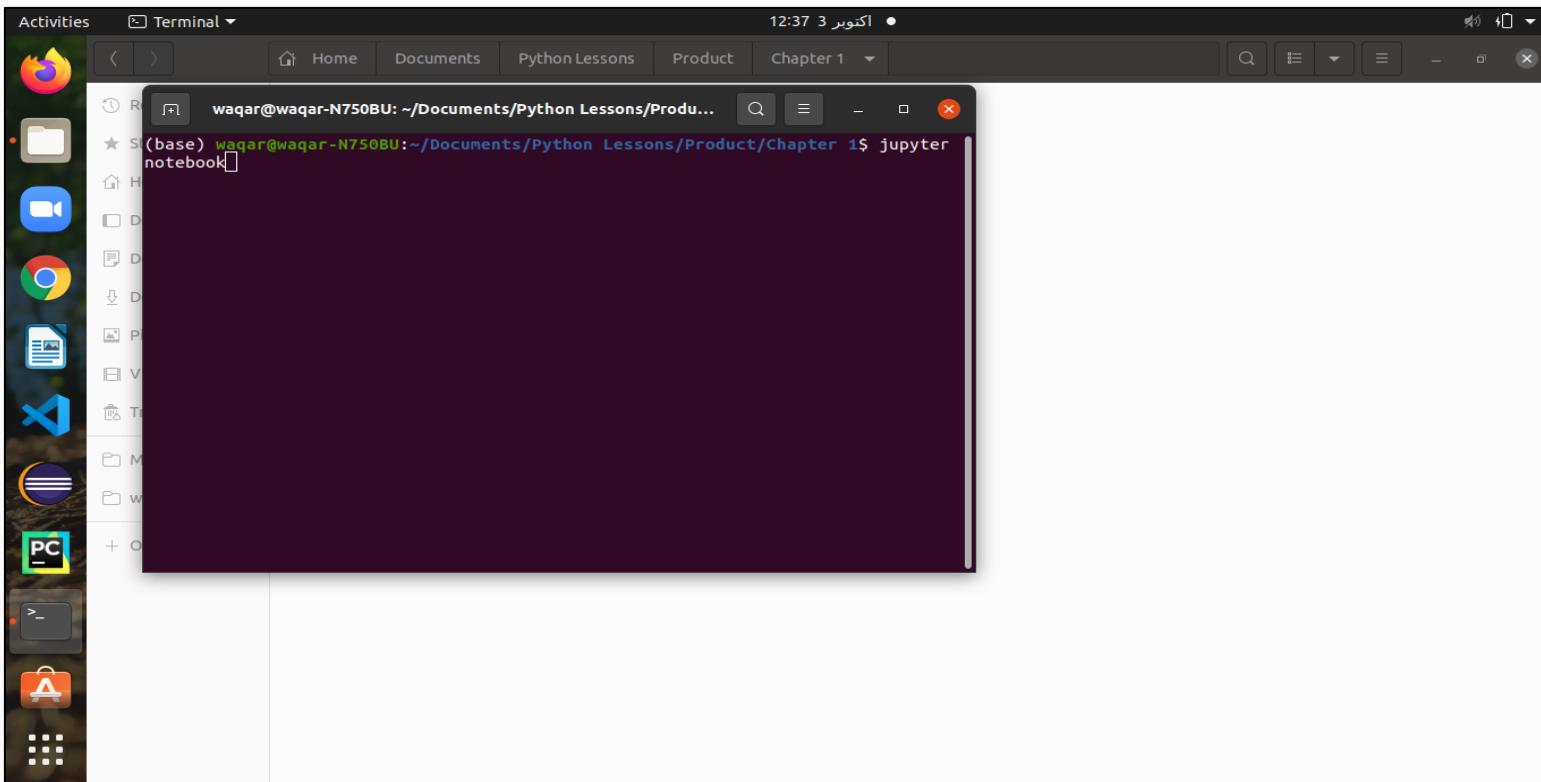
```
In [1]: print("Hello World")
Hello World
```

The output "Hello World" is displayed directly below the code. A new code cell is currently being edited, indicated by the placeholder "In []:" with a cursor.



Managing Directories in Jupyter Notebook – Ubuntu (1/4)

- To open Jupyter Notebook in a particular directory/folder in Ubuntu, go to that directory, right click and select ‘open in terminal’.
- Type ‘jupyter notebook’ without quotation marks and hit Enter.





Managing Directories in Jupyter Notebook – Ubuntu (2/4)

- Wait for sometime.

```
waqar@waqar-N750BU: ~/Documents/Python Lessons/Product/Chapter 1$ jupyter notebook
[I 12:37:48.012 NotebookApp] JupyterLab extension loaded from /home/waqar/anaconda3/lib/python3.8/site-packages/jupyterlab
[I 12:37:48.013 NotebookApp] JupyterLab application directory is /home/waqar/anaconda3/share/jupyter/lab
[I 12:37:48.017 NotebookApp] Serving notebooks from local directory: /home/waqar/Documents/Python Lessons/Product/Chapter 1
[I 12:37:48.017 NotebookApp] Jupyter Notebook 6.1.4 is running at:
[I 12:37:48.017 NotebookApp] http://localhost:8888/?token=5ba3e23ed05c480fe4a767b1dbb853ae0c163b31090a34c4
[I 12:37:48.017 NotebookApp] or http://127.0.0.1:8888/?token=5ba3e23ed05c480fe4a767b1dbb853ae0c163b31090a34c4
[I 12:37:48.017 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 12:37:48.180 NotebookApp]

To access the notebook, open this file in a browser:
    file:///home/waqar/.local/share/jupyter/runtime/nbserver-2797-open.html
Or copy and paste one of these URLs:
    http://localhost:8888/?token=5ba3e23ed05c480fe4a767b1dbb853ae0c163b31090a34c4
    or http://127.0.0.1:8888/?token=5ba3e23ed05c480fe4a767b1dbb853ae0c163b31090a34c4
```



Managing Directories in Jupyter Notebook – Ubuntu (3/4)

- Jupyter Notebook will launch in your default browser.
- You can create a new Notebook or open an existing one.
- Let's open this 'hello.ipynb' file.





Managing Directories in Jupyter Notebook – Ubuntu (4/4)

- ‘hello.ipynb’ contains a simple print statement.

The screenshot shows the Jupyter Notebook interface on an Ubuntu system. The title bar reads "jupyter hello (unsaved changes)". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The toolbar below the menu bar includes icons for file operations like Open, Save, and New, along with Run, Cell, and Kernel selection buttons. The main workspace displays a single code cell labeled "In [1]". The cell contains the Python code: `print("Hello World")`. The output of the cell, "Hello World", is displayed directly below the code. A new code cell, "In []:", is currently selected, indicated by a green border around its input field.

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction to regression

- 01 How Does Linear Regression Work?
- 02 Line representation
- 03 Implementation in python: Importing libraries & datasets
- 04 Implementation in python: Distribution of the data
- 05 Implementation in python: Creating a linear regression object



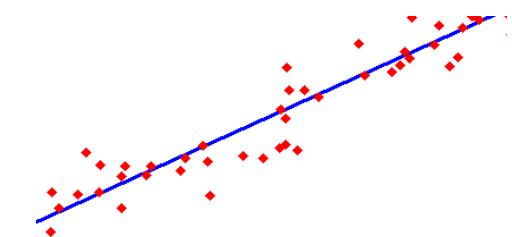
Regression

Regression is a statistical method, which predicts a relationship of a continuous outcome based on the value of one or more input variables.

Types of Regression

- **Linear Regression**

The relationship between input variables and output variable is linear.

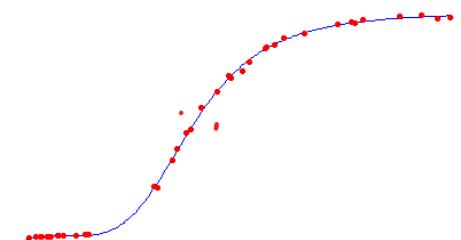


1. **Simple linear Regression:** only single input variable is used to predict an output.

2. **Multiple linear regression:** more than one input variable is involved.

- **Nonlinear Regression**

Output is model by a function which is a nonlinear combination of the inputs.



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction to regression

01

How Does Linear Regression Work?

02

Line representation

03

Implementation in python:
Importing libraries & datasets

04

Implementation in python:
Distribution of the data

05

Implementation in python:
Creating a linear regression object

06

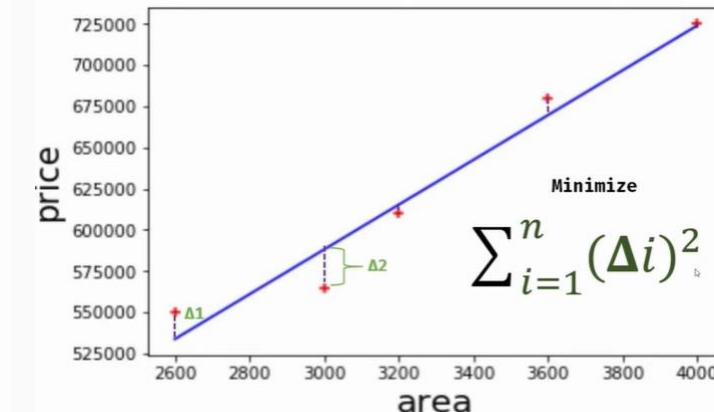
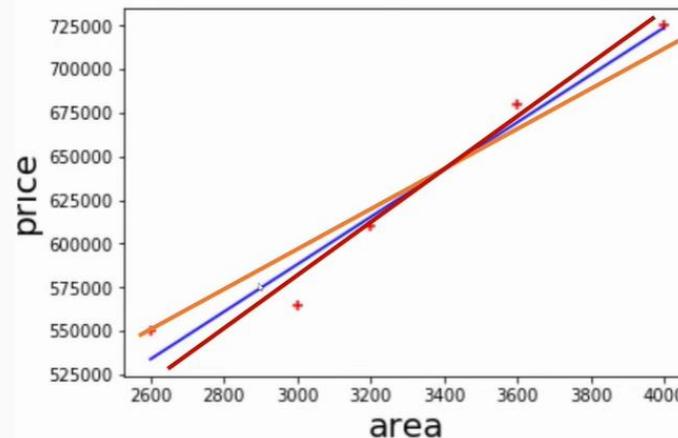


Simple Linear Regression

Working of simple Linear Regression

Let's consider the following example to explain the working of linear regression

area	price
2600	550000
3000	565000
3200	610000
3600	680000
4000	725000



What is the price of a house having an area of 330 square feet?

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction to regression

01

How Does Linear Regression Work?

02

Line representation

03

Implementation in python:
Importing libraries & datasets

04

Implementation in python:
Distribution of the data

05

Implementation in python:
Creating a linear regression object

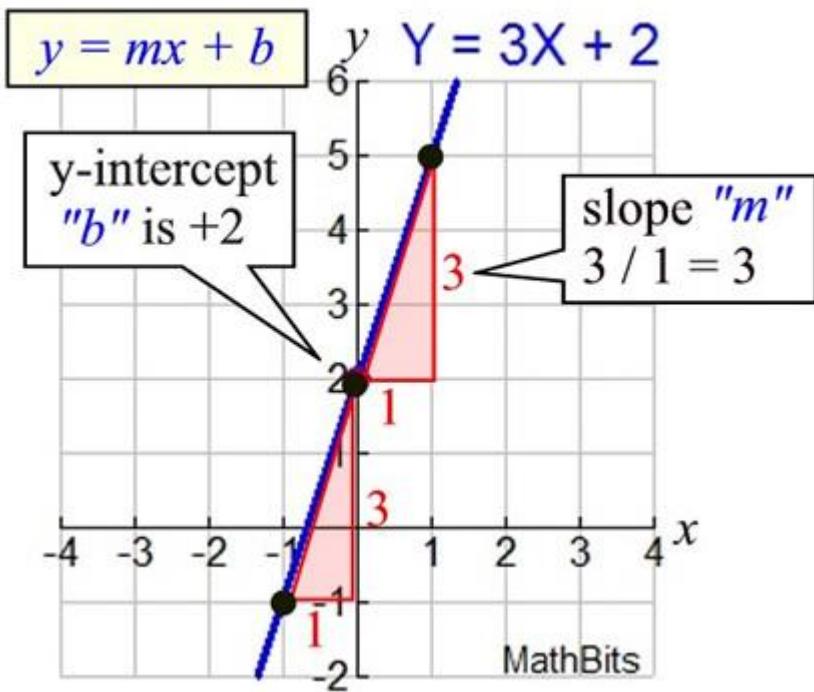
06



Simple Linear Regression

Line representation

$$Y = mx + b$$



Equation of line in terms of area and price

$$\text{price} = m * \text{area} + b$$

Dependent variable

Independent variable

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction to regression

01

How Does Linear Regression Work?

02

Line representation

03

Implementation in python:
Importing libraries & datasets

04

Implementation in python:
Distribution of the data

05

Implementation in python:
Creating a linear regression object

06



Simple Linear Regression

Simple Linear Regression in python

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

df = pd.read_csv("homeprices.csv")
df
```

	Area	Price
0	2600	550000
1	3000	565000
2	3200	610000
3	3600	680000
4	4000	725000

Import the important libraries.

Import linear model from sklearn library.

Import the csv file of the data set.

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction to regression

01

How Does Linear Regression Work?

02

Line representation

03

Implementation in python:
Importing libraries & datasets

04

Implementation in python:
Distribution of the data

05

Implementation in python:
Creating a linear regression object

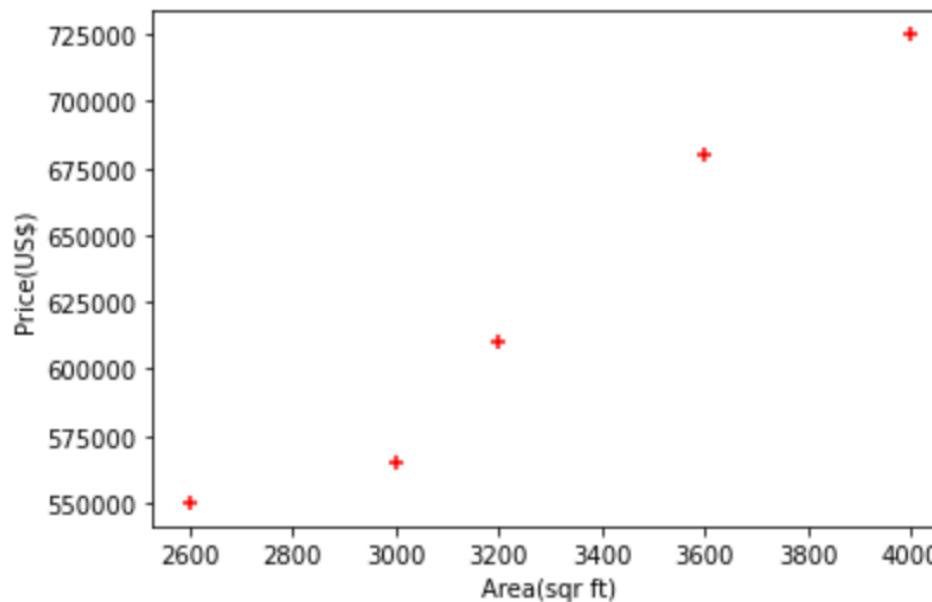
06



Simple Linear Regression

Simple Linear Regression in python

```
plt.xlabel('Area(sqr ft)')  
plt.ylabel('Price(US$)')  
plt.scatter(df.Area, df.Price, color = 'red', marker='+')  
  
<matplotlib.collections.PathCollection at 0x2a2b538cee0>
```



Set x and y labels. x label would be area and y label would be price.

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction to regression

01

How Does Linear Regression Work?

02

Line representation

03

Implementation in python:
Importing libraries & datasets

04

Implementation in python:
Distribution of the data

05

Implementation in python:
Creating a linear regression object

06



Simple Linear Regression

Simple Linear Regression in python

```
In [4]: reg = LinearRegression()
reg.fit(df[['Area']], df.Price)

Out[4]: LinearRegression()

In [5]: reg.predict([[3300]])

Out[5]: array([628715.75342466])

In [6]: reg.intercept_

Out[6]: 180616.43835616432

In [7]: reg.coef_

Out[7]: array([135.78767123])

In [8]: 135.788 * 3300 + 180616.438

Out[8]: 628716.838
```

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

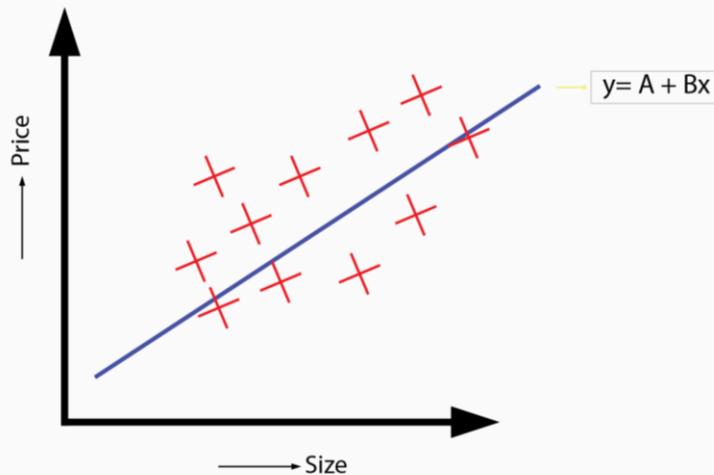
Understanding Multiple linear regression

- 01 Implementation in python: Exploring the dataset
- 02 Implementation in python: Encoding Categorical Data
- 03 Implementation in python: Splitting data into Train and Test Sets
- 04 Implementation in python: Training the model on the Training set
- 05 Evaluating the performance of the regression model
- 06 Implementation in python: Predicting the Test Set results
- 07 Root Mean Squared Error in Python

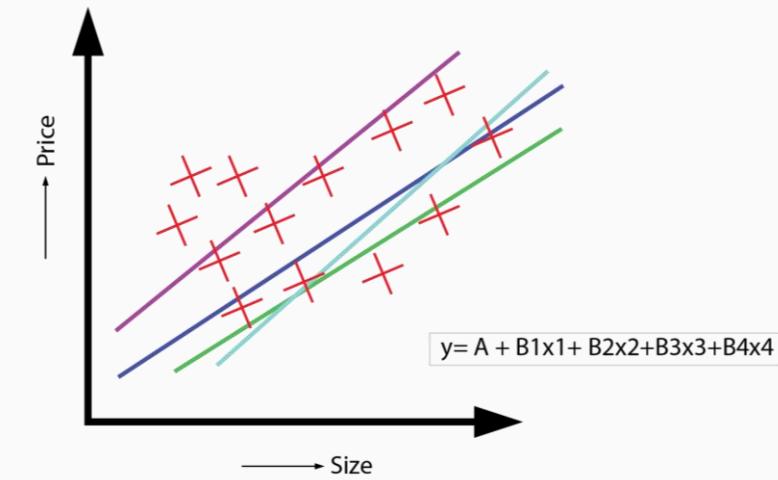


Multiple Linear Regression

Simple Linear Regression



Multiple Linear Regression



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Understanding Multiple linear regression

01

02

03

04

05

06

07

08

**Implementation in python:
Exploring the dataset**

Implementation in python:
Encoding Categorical Data

Implementation in python:
Training the model on the
Training set

Evaluating the performance of
the regression model

Implementation in python:
Splitting data into Train and Test
Sets

Implementation in
python: Predicting the
Test Set results

Root Mean Squared
Error in Python



Multiple Linear Regression

Multiple Linear Regression in python

Dataset

R&D Spend	Administration	Marketing Spend	State	Profit
165349.2	136897.8	471784.1	New York	192261.83
162597.7	151377.59	443898.53	California	191792.06
153441.51	101145.55	407934.54	Florida	191050.39
144372.41	118671.85	383199.62	New York	182901.99
142107.34	91391.77	366168.42	Florida	166187.94
131876.9	99814.71	362861.36	New York	156991.12
124615.46	147100.97	127716.92	California	156122.51



Multiple Linear Regression

Multiple Linear Regression in python

```
#import the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

#reading the dataset
dataset = pd.read_csv('50_Startups.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
dataset.head(5)
```

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Understanding Multiple linear regression

01

02

03

04

05

06

07

08

Implementation in python:
Exploring the dataset

Implementation in python:
Splitting data into Train and Test Sets

Implementation in python:
Predicting the Test Set results

Root Mean Squared Error in Python

**Implementation in python:
Encoding Categorical Data**

Implementation in python:
Training the model on the
Training set

Evaluating the performance of
the regression model



Multiple Linear Regression

Encoding Categorical Data

X:

165349.2	136897.8	471784.1	New York
162597.7	151377.59	443898.53	California
153441.51	101145.55	407934.54	Florida
144372.41	118671.85	383199.62	New York
142107.34	91391.77	366168.42	Florida
131876.9	99814.71	362861.36	New York

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [3])], remainder='passthrough')
X = np.array(ct.fit_transform(X))
```



Multiple Linear Regression

Encoding Categorical Data

After Encoding, X:

0.0	0.0	1.0	165349.2	136897.8	471784.1
1.0	0.0	0.0	162597.7	151377.59	443898.53
0.0	1.0	0.0	153441.51	101145.55	407934.54
0.0	0.0	1.0	144372.41	118671.85	383199.62
0.0	1.0	0.0	142107.34	91391.77	366168.42
0.0	0.0	1.0	131876.9	99814.71	362861.36
1.0	0.0	0.0	134615.46	147198.87	127716.82

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Understanding Multiple linear regression

01

02

03

04

05

06

07

08

Implementation in python:
Exploring the dataset

Implementation in python:
Encoding Categorical Data

**Implementation in python:
Splitting data into Train and Test Sets**

Implementation in python:
Training the model on the
Training set

Implementation in
python: Predicting the
Test Set results

Evaluating the performance of
the regression model

Root Mean Squared
Error in Python



Multiple Linear Regression

Splitting the dataset into the Training set and Test set

```
from sklearn.model_selection import train_test_split  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
```

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Understanding Multiple linear regression

01

02

03

04

05

06

07

08

Implementation in python:
Exploring the dataset

Implementation in python:
Encoding Categorical Data

Implementation in python:
Splitting data into Train and Test Sets

**Implementation in python:
Training the model on the
Training set**

Implementation in
python: Predicting the
Test Set results

Evaluating the performance of
the regression model

Root Mean Squared
Error in Python



Multiple Linear Regression

Training the Multiple Linear Regression model on the Training set

```
from sklearn.linear_model import LinearRegression  
regressor = LinearRegression()  
regressor.fit(X_train, y_train)
```

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Understanding Multiple linear regression

01

02

03

04

05

06

07

08

Implementation in python:
Exploring the dataset

Implementation in python:
Encoding Categorical Data

Implementation in python:
Training the model on the
Training set

Evaluating the performance of
the regression model

Implementation in python:
Splitting data into Train and Test
Sets

**Implementation in
python: Predicting the
Test Set results**

Root Mean Squared
Error in Python



Multiple Linear Regression

Predicting the Test Set results

```
y_pred = regressor.predict(X_test)  
  
df = pd.DataFrame({'Real Values':y_test, 'Predicted Values':y_pred})  
df
```

	Real Values	Predicted Values
0	65200.33	69622.870630
1	14681.40	52846.107901
2	49490.75	62780.406794
3	96712.80	94377.300765
4	97427.84	100871.598266
5	77798.83	77589.217308
6	97483.56	99644.979787
7	132602.65	151447.661176
8	103282.38	106415.733395
9	105008.31	116820.266911

**But, how to measure
how well our model
predicted?**

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Understanding Multiple linear regression

01

02

03

04

05

06

07

08

Implementation in python:
Exploring the dataset

Implementation in python:
Encoding Categorical Data

Implementation in python:
Training the model on the
Training set

Implementation in python:
Splitting data into Train and Test
Sets

Implementation in
python: Predicting the
Test Set results

**Evaluating the performance of
the regression model**

Root Mean Squared
Error in Python



Model Evaluation for Regression

Mean Absolute Error (MAE):

$$\frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

Number of values
Actual value
Predictive value

Mean Squared Error (MSE):

$$\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

Root Mean Squared Error (RMSE):

$$\sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Understanding Multiple linear regression

01

02

03

04

05

06

07

08

Implementation in python:
Exploring the dataset

Implementation in python:
Encoding Categorical Data

Implementation in python:
Training the model on the
Training set

Evaluating the performance of
the regression model

Implementation in python:
Splitting data into Train and Test
Sets

Implementation in
python: Predicting the
Test Set results

**Root Mean Squared
Error in Python**



Model Evaluation for Regression

Python Code for Root Mean Squared Error (RMSE):

```
# RMSE
from sklearn import metrics
print(np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

14430.747692019655



Quiz Time

Which of the following evaluation metrics can be used to evaluate a model while modeling a continuous output variable?

- A) AUC-ROC
- B) Accuracy
- C) Logloss
- D) Mean-Squared-Error



Quiz Time

Which of the following evaluation metrics can be used to evaluate a model while modeling a continuous output variable?

- A) AUC-ROC
- B) Accuracy
- C) Logloss
- D) Mean-Squared-Error

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction to classification

01

K-Nearest Neighbors algorithm

02

03

Example of KNN

04

K-Nearest Neighbours (KNN) using python

05

Implementation in python:
Importing required libraries

06

Implementation in python:
Importing the dataset

07

Implementation in python:
Splitting data into Train and Test Sets

08

Implementation in
python: Feature Scaling

09

Implementation in python:
Importing the KNN classifier

10

Implementation in python:
Results prediction & Confusion matrix



Classification

Classification

Classification is a predictive modelling issue in machine learning where a class label is predicted for a given sample of input data.

Application of Classification

- Filtering spam emails
- Speech and handwriting recognition
- Biometric
- Fingerprint recognition etc.

Algorithms of Classification

- Neural Networks
- Regression
- K-nearest neighbor

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction to classification

01

02

K-Nearest Neighbors algorithm

03

Example of KNN

04

K-Nearest Neighbours (KNN) using python

05

Implementation in python:
Importing required libraries

06

Implementation in python:
Importing the dataset

07

Implementation in python:
Splitting data into Train and Test Sets

08

Implementation in
python: Feature Scaling

09

Implementation in python:
Importing the KNN classifier

10

Implementation in python:
Results prediction & Confusion matrix



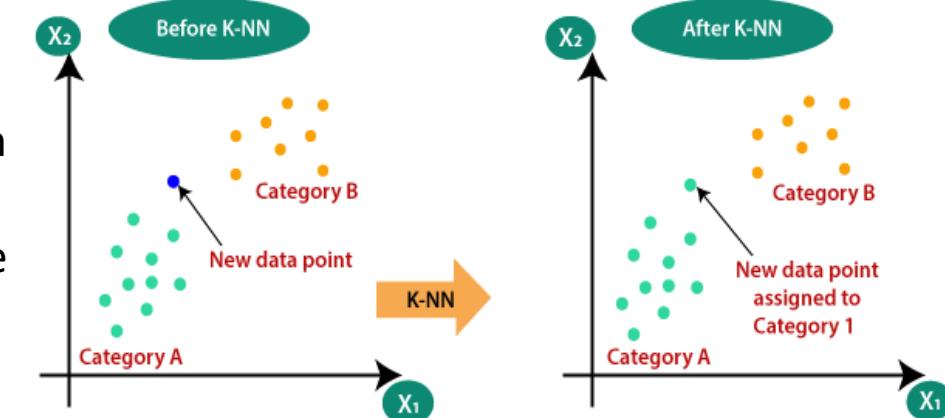
K-Nearest Neighbours (KNN)

K-Nearest Neighbours (KNN)

The K-Nearest Neighbors algorithm is a classification method that utilizes a set of labelled points to learn how to identify new ones

Algorithm of KNN

- Choose a value for K.
- Determine the distance of an unknown point from all other points.
- Now in training data set choose the K which are nearest to the unknown data point.
- Now predict the response of unknown data point.



$$\text{Dis } (x_1, x_2) = \sqrt{\sum_{i=0}^n (x_{1i} - x_{2i})^2}$$

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction to classification

01

K-Nearest Neighbors algorithm

02

03

K-Nearest Neighbours (KNN) using python

04

Implementation in python:
Importing required libraries

05

Implementation in python:
Importing the dataset

06

Implementation in python:
Splitting data into Train and Test Sets

07

Implementation in python:
Feature Scaling

08

Implementation in python:
Importing the KNN classifier

09

Implementation in python:
Results prediction & Confusion matrix

10



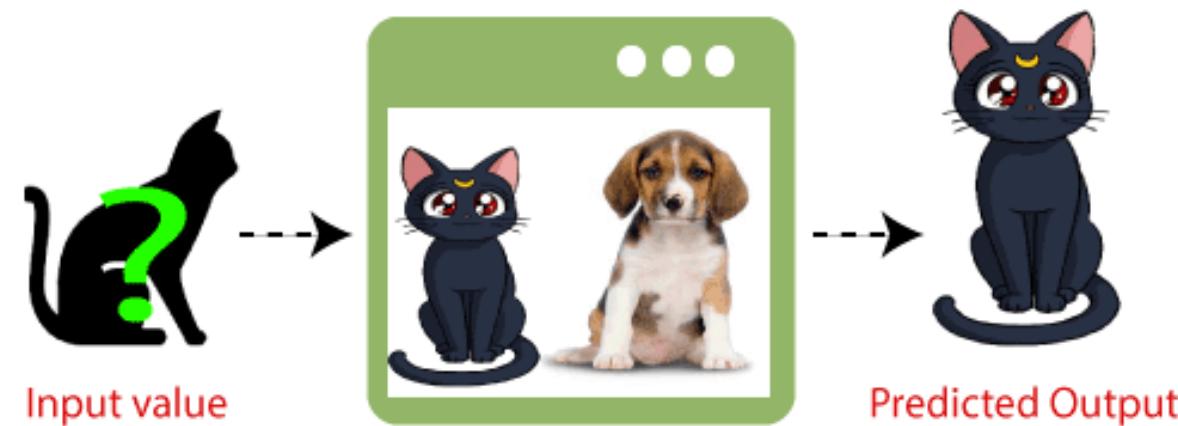
K-Nearest Neighbours (KNN)

K-Nearest Neighbours (KNN)

- **Example**

KNN classifier will be use in this example to classify the input image into cat or dog.

KNN Classifier



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction to classification

01

K-Nearest Neighbors algorithm

02

03

Example of KNN

04

K-Nearest Neighbours (KNN) using python

05

Implementation in python:
Importing required libraries

06

Implementation in python:
Importing the dataset

07

Implementation in python:
Splitting data into Train and Test Sets

08

Implementation in
python: Feature Scaling

09

Implementation in python:
Importing the KNN classifier

10

Implementation in python:
Results prediction & Confusion matrix



KNN Implementation

K-Nearest Neighbours (KNN) in python

Steps to implement KNN

- Data pre-processing like filtering
- Fitting KNN to the training set.
- Predict the test results.
- Test accuracy of the results.

User ID	Gender	Age	EstimatedSalary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	0
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	0
15728773	Male	27	58000	0
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	0
15727311	Female	35	65000	0
15570769	Female	26	80000	0
15606274	Female	26	52000	0
15746139	Male	20	86000	0
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1
15617482	Male	45	26000	1
15704583	Male	46	28000	1
15621083	Female	48	29000	1
15649487	Male	45	22000	1
15736760	Female	47	49000	1

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction to classification

01

K-Nearest Neighbors algorithm

02

03

K-Nearest Neighbours (KNN) using python

04

**Implementation in python:
Importing required libraries**

05

Implementation in python:
Importing the dataset

06

Implementation in python:
Splitting data into Train and Test Sets

07

Implementation in python:
Feature Scaling

08

Implementation in python:
Importing the KNN classifier

09

Implementation in python:
Results prediction & Confusion matrix

10



KNN Implementation

K-Nearest Neighbours (KNN)in python

Implementation steps 1

- First, we will import the required libraries.

The screenshot shows a Jupyter Notebook interface with the title "jupyter KNN_Binary_Classification (autosaved)". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Logout, Not Trusted, and Python 3. Below the toolbar is a toolbar with icons for file operations like Open, Save, New, and Run. A code cell in the notebook contains the following Python code:

```
In [1]: #Importing the Libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics import confusion_matrix
```

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction to classification

01

K-Nearest Neighbors algorithm

02

03

Example of KNN

04

K-Nearest Neighbours (KNN) using python

05

Implementation in python:
Importing required libraries

06

**Implementation in python:
Importing the dataset**

07

Implementation in python:
Splitting data into Train and Test Sets

08

Implementation in
python: Feature Scaling

09

Implementation in python:
Importing the KNN classifier

10

Implementation in python:
Results prediction & Confusion matrix



KNN Implementation

K-Nearest Neighbours (KNN)in python

Implementation steps 2

- Then we will import the given data sets.

```
# Importing the dataset
dataset = pd.read_csv('user data.csv')
X = dataset.iloc[:, 2:4].values
y = dataset.iloc[:, 4].values
```

- Extract independent and dependent variables – X & y

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction to classification

01

K-Nearest Neighbors algorithm

02

03

K-Nearest Neighbours (KNN) using python

04

Implementation in python:
Importing required libraries

05

Implementation in python:
Importing the dataset

06

**Implementation in python:
Splitting data into Train and Test Sets**

07

Implementation in
python: Feature Scaling

08

Implementation in python:
Importing the KNN classifier

09

Implementation in python:
Results prediction & Confusion matrix

10



KNN Implementation

K-Nearest Neighbours (KNN)in python

Implementation steps 3

- Split the data into testing and training data set.

```
In [3]: #Training and Testing Data (divide the data into two part)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =train_test_split(X,y,test_size= 0.25, random_s:
```

- Import sklearn for pre-processing and scale the features.

```
In [4]: from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.fit_transform(X_test)
```

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction to classification

01

K-Nearest Neighbors algorithm

02

03

Example of KNN

04

K-Nearest Neighbours (KNN) using python

05

Implementation in python:
Importing required libraries

06

Implementation in python:
Importing the dataset

07

Implementation in python:
Splitting data into Train and Test Sets

08

**Implementation in
python: Feature Scaling**

09

Implementation in python:
Importing the KNN classifier

10

Implementation in python:
Results prediction & Confusion matrix



KNN Implementation

K-Nearest Neighbours (KNN)in python

Unscaled data

```
print(X_train)
```

```
[[ 44  39000]
 [ 32 120000]
 [ 38  50000]
 [ 32 135000]
 [ 52  21000]
 [ 53 104000]
 [ 39  42000]
 [ 38  61000]
 [ 36  50000]
 [ 36  63000]
 [ 35  25000]
 [ 35  50000]
 [ 42  73000]
 [ 17 100001]]
```

Scaled data

```
print(X_train)
```

```
[[ 0.58164944 -0.88670699]
 [-0.60673761  1.46173768]
 [-0.01254409 -0.5677824 ]
 [-0.60673761  1.89663484]
 [ 1.37390747 -1.40858358]
 [ 1.47293972  0.99784738]
 [ 0.08648817 -0.79972756]
 [-0.01254409 -0.24885782]
 [-0.21060859 -0.5677824 ]
 [-0.21060859 -0.19087153]
 [-0.30964085 -1.29261101]
 [-0.30964085 -0.5677824 ]
 [ 0.38358493  0.09905991]
 [ 0.8787462 -0.59677555]
 [ 2.06713324 -1.17663843]
 [ 1.97681071 -0.13288521]]
```

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction to classification

01

K-Nearest Neighbors algorithm

02

03

Example of KNN

04

K-Nearest Neighbours (KNN) using python

05

Implementation in python:
Importing required libraries

06

Implementation in python:
Importing the dataset

07

Implementation in python:
Splitting data into Train and Test Sets

08

Implementation in
python: Feature Scaling

09

Implementation in python:
Importing the KNN classifier

10

Implementation in python:
Results prediction & Confusion matrix



KNN Implementation

K-Nearest Neighbours (KNN)in python

Implementation steps 4

- Import the KNN classifier from sklearn library

```
In [8]: from sklearn.neighbors import KNeighborsClassifier  
classifier=KNeighborsClassifier(n_neighbors=5, metric="minkowski", p=2)  
classifier.fit(X_train,y_train)  
  
Out[8]: KNeighborsClassifier()
```

Define the different parameters:

- n_neighbors
- metric='minkowski'.
- p=2.

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction to classification

01

K-Nearest Neighbors algorithm

02

03

K-Nearest Neighbours (KNN) using python

04

Implementation in python:
Importing required libraries

05

Implementation in python:
Importing the dataset

06

Implementation in python:
Splitting data into Train and Test Sets

07

Implementation in python:
Feature Scaling

08

Implementation in python:
Importing the KNN classifier

09

Implementation in python:
Results prediction & Confusion matrix

10



KNN Implementation

K-Nearest Neighbours (KNN)in python

Prediction of results and confusion matrix

- Now will predict the results using predict() method
- Get the confusion matrix using y_pred, y_test values

```
y_pred = classifier.predict(X_test)  
  
# Making the Confusion Matrix  
cm = confusion_matrix(y_pred, y_test)  
print(cm)
```

```
[[64  3]  
 [ 4 29]]
```

Correct predictions= $64+29 = 93$
Incorrect predictions= $3+4 = 7$



Quiz Time

Classification is-

- A. Unsupervised learning
- B. Reinforcement learning
- C. Supervised learning
- D. None



Quiz Time

Classification is-

- A. Unsupervised learning
- B. Reinforcement learning
- C. Supervised learning
- D. None

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction to decision trees

- 01 What is Entropy?
- 02 Exploring the dataset
- 03 Decision tree structure
- 04 Implementation in python:
Importing libraries & datasets
- 05 Implementation in python:
Encoding Categorical Data
- 06 Implementation in python:
Splitting data into Train and Test Sets
- 07 Implementation in python:
Results prediction & Accuracy



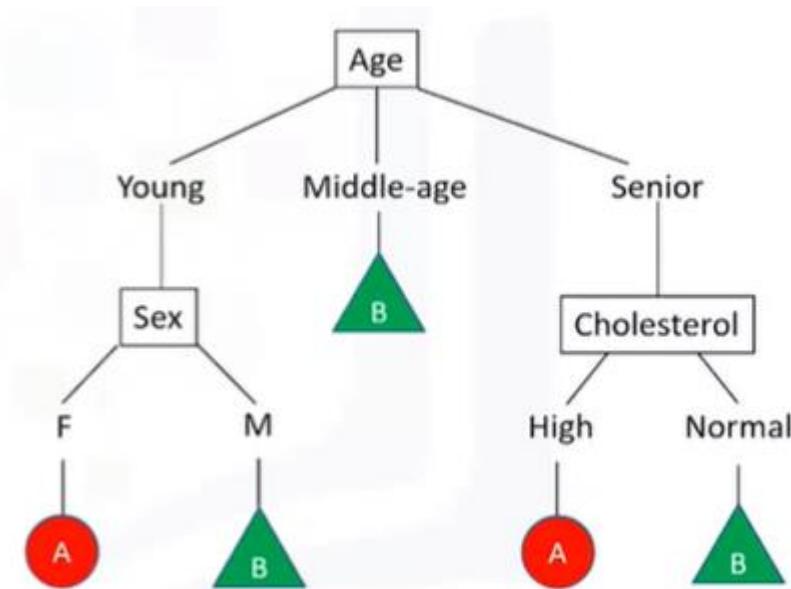
Decision Tree

What is decision tree?

Decision trees are a kind of Supervised Machine Learning in which data is continually segmented based on a parameter.

Learning algorithm of decision tree

- Choose a feature from the given data set.
- Determine the importance of feature in segmenting of data.
- Data segmenting depend upon the best attribute.
- Now repeat the above steps.



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction to decision trees

Exploring the dataset

Implementation in python:
Importing libraries & datasets

Implementation in python:
Splitting data into Train and Test Sets

01

02

03

04

05

06

07

08

What is Entropy?

Decision tree structure

Implementation in python:
Encoding Categorical Data

Implementation in python:
Results prediction & Accuracy



Decision Tree

Entropy

It is the measure of the amount of uncertainty or randomness in data.

$$H(S) = \sum_{x \in X} p(x) \log_2 \frac{1}{p(x)}$$

P(x) is a probability function of the event x to take place

Lowest value: 0 (no randomness)
Highest value: 1 (high randomness)

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction to decision trees

Exploring the dataset

Implementation in python:
Importing libraries & datasets

Implementation in python:
Splitting data into Train and Test Sets

01

02

03

04

05

06

07

08

What is Entropy?

Decision tree structure

Implementation in python:
Encoding Categorical Data

Implementation in python:
Results prediction & Accuracy



Decision Tree Implementation

Building a Decision Tree in python

Data set

In [12]: `data.head(10)`

Out[12]:

	company	job	degree	salary_more_then_100k
0	company	job	degree	salary_more_then_100k
1	google	sale executive	bachelaors	0
2	google	sale executive	masters	0
3	google	business manager	bachelaors	1
4	google	business manager	masters	1
5	google	computer programmer	bachelaors	0
6	google	computer programmer	masters	1
7	abc pharma	sale executive	masters	0
8	abc pharma	computer programmer	bachelaors	0
9	abc pharma	business manager	bachelaors	0

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction to decision trees

Exploring the dataset

Implementation in python:
Importing libraries & datasets

Implementation in python:
Splitting data into Train and Test Sets

01

02

03

04

05

06

07

08

What is Entropy?

Decision tree structure

Implementation in python:
Encoding Categorical Data

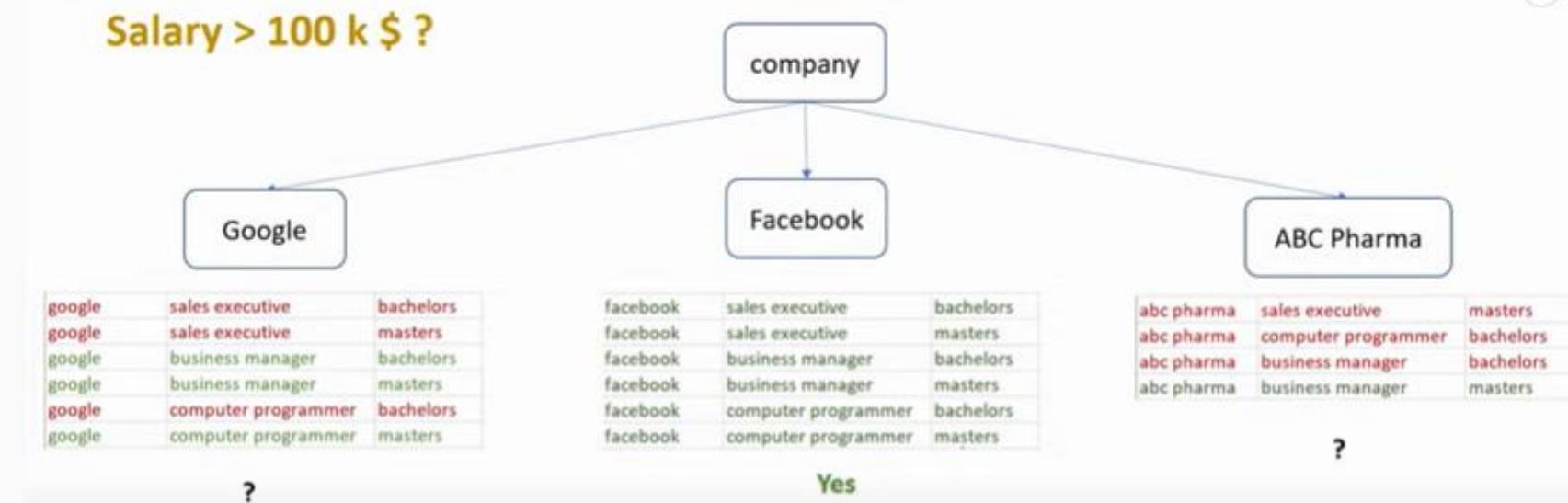
Implementation in python:
Results prediction & Accuracy



Decision Tree Implementation

Building a Decision Tree in python

Decision tree



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction to decision trees

Exploring the dataset

**Implementation in python:
Importing libraries & datasets**

Implementation in python: Splitting data into Train and Test Sets

01

02

03

04

05

06

07

08

What is Entropy?

Decision tree structure

Implementation in python:
Encoding Categorical Data

Implementation in python:
Results prediction & Accuracy



Decision Tree Implementation

Building a Decision Tree in python

Implementation steps 1

- Import the required libraries

The screenshot shows a Jupyter Notebook interface titled "jupyter Decision_tree (unsaved changes)". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, and Python 3. Below the toolbar is a toolbar with icons for file operations like new, open, save, and run. The main area shows a code cell labeled "In [1]:" containing the following Python code:

```
In [1]: # Load libraries
import pandas as pd
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
```



Decision Tree Implementation

Building a Decision Tree in python

Implementation steps 2

- Define the data variables and import the data file

```
In [2]: #Working with the dataset here
```

```
col_names = ['company', 'job', 'degree', 'salary_more_then_100k']# Load dataset
data = pd.read_csv("salaries.csv", header=None, names=col_names)
```



Decision Tree Implementation

Building a Decision Tree in python

Implementation steps 3

- Split the data into features (Inputs - X) and Target variable (output y)

```
In [19]: #Split the dataset in features and target variable  
feature_cols = ['company','job','degree']  
X = data[feature_cols]  
y = data['salary_more_then_100k']  
....
```

- Target variable => Salary_more_then100k
- Defining Features for salary => company, job, degree

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction to decision trees

Exploring the dataset

Implementation in python:
Importing libraries & datasets

Implementation in python:
Splitting data into Train and Test Sets

Implementation in python:
Results prediction & Accuracy

01

02

03

04

05

06

07

08

What is Entropy?

Decision tree structure

**Implementation in python:
Encoding Categorical Data**



Decision Tree Implementation

Building a Decision Tree in python

Implementation steps 4

- Encode the categorical features into numerical values
- Use ordinal encoder

```
In [4]: # Import Label encoder
from sklearn import preprocessing
# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()
# Encode Labels in column
data['company']= label_encoder.fit_transform(data['company'])
data['job']= label_encoder.fit_transform(data['job'])
data['degree']= label_encoder.fit_transform(data['degree'])
print(data.head())
```

	company	job	degree	salary_more_then_100k
0	1	2	1	salary_more_then_100k
1	3	3	0	0
2	3	3	2	0
3	3	0	0	1
4	3	0	2	1



Decision Tree Implementation

Building a Decision Tree in python

```
print(X)
```

	company	job	degree
0	1	2	1
1	3	3	0
2	3	3	2
3	3	0	0
4	3	0	2
5	3	1	0
6	3	1	2
7	0	3	2
8	0	1	0
9	0	0	0
10	0	0	2
11	2	3	0
12	2	3	2
13	2	0	0
14	2	0	2
15	2	1	0
16	2	1	2

```
In [20]: print(y)
```

	salary_more_then_100k
0	0
1	0
2	0
3	1
4	1
5	0
6	1
7	0
8	0
9	0
10	1
11	1
12	1
13	1
14	1
15	1
16	1

Name: salary_more_then_100k, dtype: object

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction to decision trees

Exploring the dataset

Implementation in python:
Importing libraries & datasets

Implementation in python: Splitting data into Train and Test Sets

01

02

03

04

05

06

07

08

What is Entropy?

Decision tree structure

Implementation in python:
Encoding Categorical Data

Implementation in python:
Results prediction & Accuracy



Decision Tree Implementation

Building a Decision Tree in python

Implementation steps 5

- Split the data into training and testing sets

```
In [7]: X_train, X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=100)
```

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction to decision trees

Exploring the dataset

Implementation in python:
Importing libraries & datasets

Implementation in python:
Splitting data into Train and Test Sets

01

02

03

04

05

06

07

08

What is Entropy?

Decision tree structure

Implementation in python:
Encoding Categorical Data

**Implementation in python:
Results prediction & Accuracy**



Decision Tree Implementation

Building a Decision Tree in python

Implementation steps 6

- **Creating Decision Tree**
- **Use entropy to train the model**

```
In [24]: # Create Decision Tree classifier object using entropy
clf_entropy = DecisionTreeClassifier(criterion="entropy", max_depth=3)

# Train Decision Tree Classifier
clf_entropy = clf_entropy.fit(X_train,y_train)
```

- **Predict the results & Find the accuracy of the model**

```
In [8]: # Create Decision Tree classifier object using entropy
clf_entropy = DecisionTreeClassifier(criterion="entropy", max_depth=8)

# Train Decision Tree Classifier
clf_entropy = clf_entropy.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf_entropy.predict(X_test)

print("Accuracy:", metrics.accuracy_score(y_test,y_pred))
```

Accuracy: 0.75



Quiz Time

Decision Trees can be used for Classification Tasks.

- a) True
- b) False



Quiz Time

Decision Trees can be used for Classification Tasks.

- a) True
- b) False

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction

01

02

03

04

05

06

07

08

Implementation steps

Implementation in python:
Importing libraries & datasets

Implementation in python:
Splitting data into Train and Test Sets

Implementation in python:
Training the model

Implementation in python:
Results prediction & Confusion matrix



Logistic Regression

What is logistic Regression?

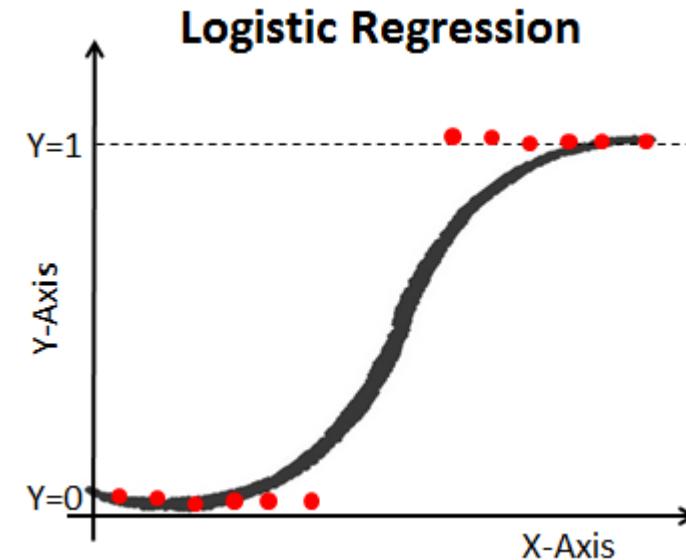
The logistic model is used to predict the likelihood of a specific class or occurrence.

Conditions to use logistic regression

- If the information is binary.
- If you require probabilistic outcomes.
- When a linear decision boundary is required.

Applications

- To forecast survival in wounded patients in the medical profession.
- To estimate the likelihood of a person suffering a heart attack.
- Predicting the likelihood of a procedure or product failing.
- Predicting a homeowner's possibility of defaulting on a lender.



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction

01

02

Implementation steps

03

Implementation in python:
Importing libraries & datasets

04

Implementation in python:
Splitting data into Train and Test Sets

05

Implementation in python: Pre-processing

06

Implementation in python:
Training the model

07

Implementation in python: Results prediction & Confusion matrix

08

Logistic Regression vs Linear Regression



Logistic Regression Implementation

Logistic Regression in python

- **Steps to implement Logistic Regression**
 - Data pre-processing like filtering
 - Fitting KNN to the training set.
 - Predict the test results.
 - Test accuracy of the results.

User ID	Gender	Age	EstimatedSalary	Purchased
15624510	Male	15	19000	0
15810944	Male	35	20000	0
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	0
15728773	Male	27	58000	0
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	0
15727311	Female	35	65000	0
15570769	Female	26	80000	0
15606274	Female	26	52000	0
15746139	Male	20	86000	0
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1
15617482	Male	45	26000	1
15704583	Male	46	28000	1
15621083	Female	48	29000	1
15649487	Male	45	22000	1
15736760	Female	47	49000	1

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

Introduction

01

02

03

04

05

06

07

08

Implementation steps

Implementation in python:
Splitting data into Train and Test Sets

Implementation in python:
Training the model

Logistic Regression vs Linear Regression

Implementation in python:
Importing libraries & datasets

Implementation in python:
Pre-processing

Implementation in python:
Results prediction & Confusion matrix



Logistic Regression Implementation

Logistic Regression in python

Implementation step 1

- First, we will import the important libraries.

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** jupyter logistic_regression_Binary_Classification (unsaved changes)
- User Area:** Python 3
- Toolbar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Not Trusted
- Cell Buttons:** New, Run, Cell, Code, Cell Type
- Code Cell:** In [1]:
```python  
#Importing the Libraries  
import numpy as np  
import matplotlib.pyplot as plt  
import pandas as pd  
from sklearn.metrics import confusion\_matrix



# Logistic Regression Implementation

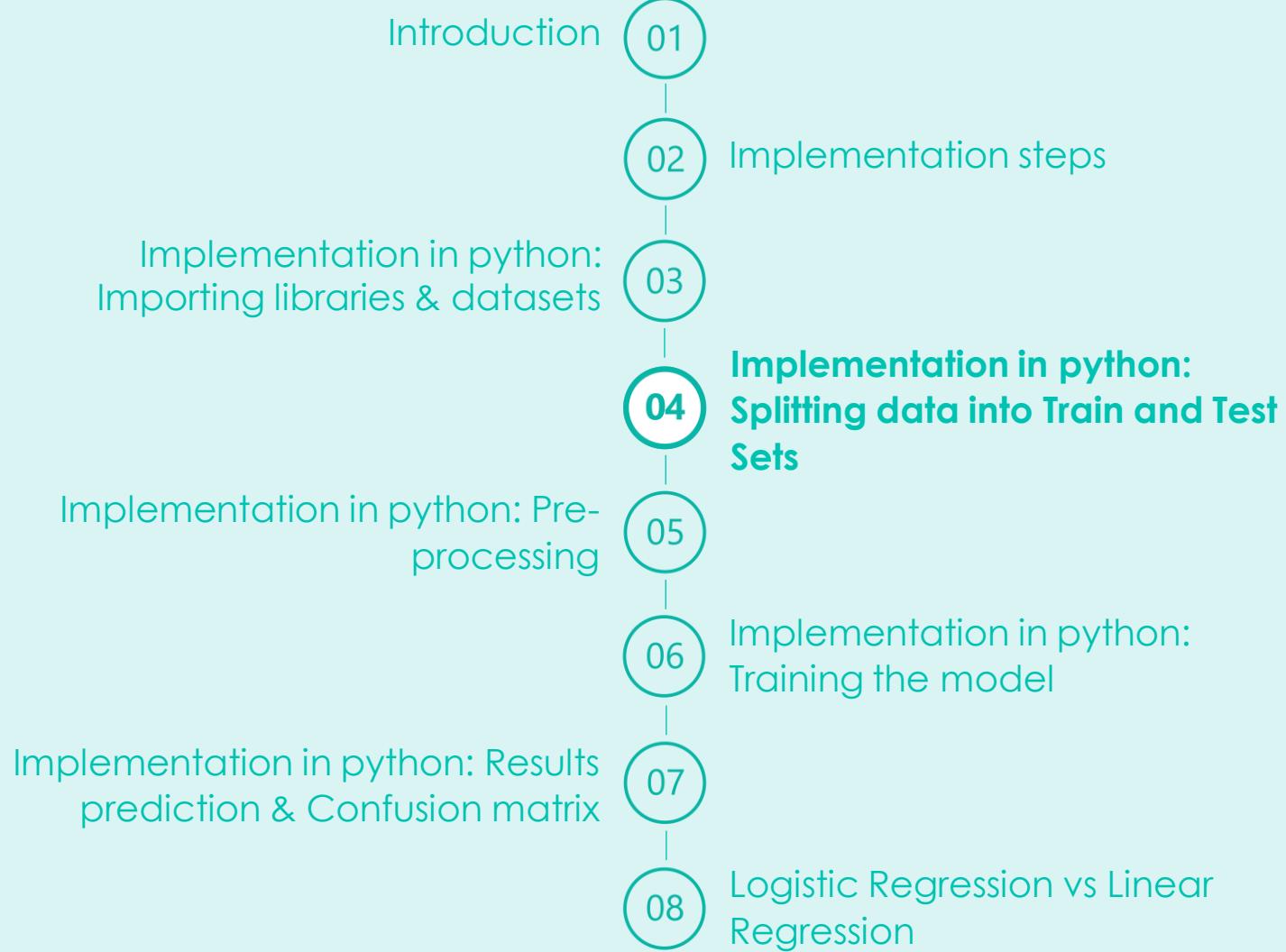
## *Logistic Regression in python*

### **Implementation step 2**

- we will import the given data sets.
- Extract independent and dependent variables.

```
In [3]: # Importing the dataset
dataset = pd.read_csv('user data.csv')
X = dataset.iloc[:, [2,4]].values
y = dataset.iloc[:, 4].values
```

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression**
- Clustering
- Recommender System
- Conclusion





# Logistic Regression Implementation

## *Logistic Regression in python*

### **Implementation step 3**

- Split the data into testing and training data set.
- Import sklearn for pre-processing and scale the features.

```
In [4]: #Training and Testing Data (divide the data into two part)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =train_test_split(X,y,test_size= 0.25, random_state=0)
```

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

**Classification Algorithms: Logistic regression**

Clustering

Recommender System

Conclusion

Introduction

01

02

03

04

05

06

07

08

Implementation steps

Implementation in python:  
Splitting data into Train and Test Sets

Implementation in python:  
Training the model

Logistic Regression vs Linear Regression

Implementation in python:  
Importing libraries & datasets

**Implementation in python: Pre-processing**

Implementation in python: Results prediction & Confusion matrix



# Logistic Regression Implementation

## *Logistic Regression in python*

### **Implementation step 3**

- Import sklearn for pre-processing and scale the features.

```
In [5]: from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.fit_transform(X_test)
```



# Logistic Regression Implementation

*Logistic Regression in python*

**Scaled data**

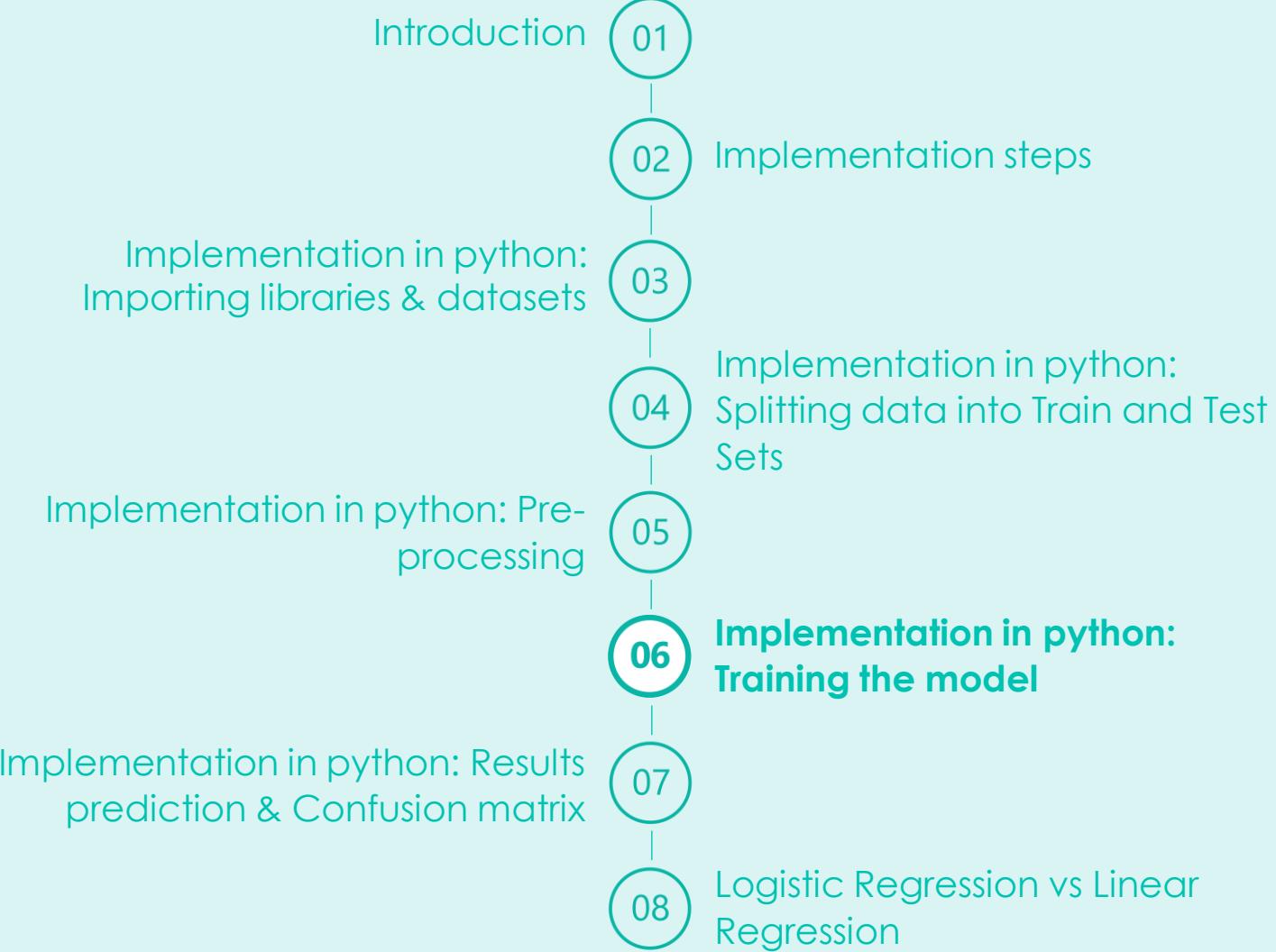
In [11]: `print(X)`

```
[[19 0]
 [35 0]
 [26 0]
 [27 0]
 [19 0]
 [27 0]
 [27 0]
 [32 1]
 [25 0]
 [35 0]
 [26 0]
 [26 0]
 [20 0]
 [32 0]
 [18 0]
 [29 0]
 [47 1]
 [45 1]
 [46 1]
 [48 1]]
```

In [9]: `print(X_train)`

```
[[0.58164944 -0.76635604]
 [-0.60673761 1.30487651]
 [-0.01254409 -0.76635604]
 [-0.60673761 1.30487651]
 [1.37390747 1.30487651]
 [1.47293972 1.30487651]
 [0.08648817 -0.76635604]
 [-0.01254409 -0.76635604]
 [-0.21060859 -0.76635604]
 [-0.21060859 -0.76635604]
 [-0.30964085 -0.76635604]
 [-0.30964085 -0.76635604]
 [0.38358493 1.30487651]
 [0.8787462 1.30487651]
 [2.06713324 1.30487651]
 [1.07681071 -0.76635604]
 [0.68068169 1.30487651]
 [-0.70576986 -0.76635604]
 [0.77971394 -0.76635604]
 [0.3707162 1.30487651]]
```

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression**
- Clustering
- Recommender System
- Conclusion





# Logistic Regression Implementation

## *Logistic Regression in python*

### **Implementation step 4**

- **Fitting Logistic Regression classifier to training data**
- Import sklearn and get our classifier – Logistic Regression

```
In [12]: from sklearn.linear_model import LogisticRegression
classifier=LogisticRegression(random_state=0)
classifier.fit(X_train,y_train)

Out[12]: LogisticRegression(random_state=0)
```

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

**Classification Algorithms: Logistic regression**

Clustering

Recommender System

Conclusion

Introduction

01

02

03

04

05

06

07

08

Implementation steps

Implementation in python:  
Splitting data into Train and Test Sets

Implementation in python:  
Training the model

Logistic Regression vs Linear Regression

Implementation in python:  
Importing libraries & datasets

Implementation in python:  
Pre-processing

**Implementation in python: Results prediction & Confusion matrix**



# Logistic Regression Implementation

## *Logistic Regression in python*

### **Prediction of results and confusion matrix**

Now will predict the results and confusion matrix.

```
y_pred = classifier.predict(X_test)
print(y_pred)
```

```
[0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 1 1 0 0 0
 0 0 1 0 0 0 1 0 0 1 0 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 0 1 0 0 0 1 0 0 1
 0 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 1 0 1 1 1]
```

Correct predictions=  $68+32 = 100$   
Incorrect predictions= 0

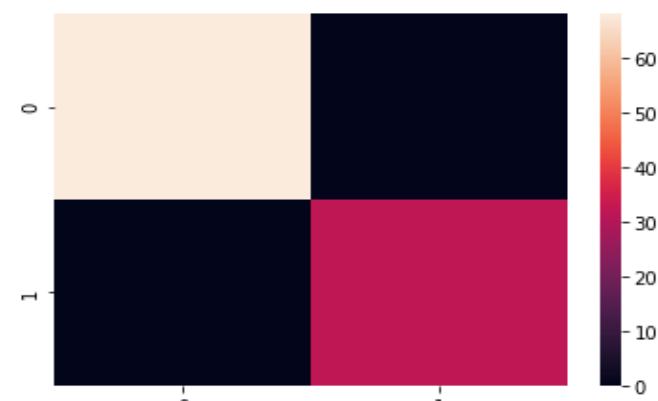
```
In [14]: # Making the Confusion Matrix
import seaborn as sns

cm = confusion_matrix(y_pred, y_test)
print(cm)
sns.heatmap(cm)
```

---

```
[[68 0]
 [0 32]]
```

Out[14]: <AxesSubplot:>



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

**Classification Algorithms: Logistic regression**

Clustering

Recommender System

Conclusion

Introduction

01

02

03

04

05

06

07

08

Implementation steps

Implementation in python:  
Splitting data into Train and Test Sets

Implementation in python:  
Training the model

**Logistic Regression vs Linear Regression**

Implementation in python:  
Importing libraries & datasets

Implementation in python:  
Pre-processing

Implementation in python:  
Results prediction & Confusion matrix



# ***Logistic Regression Vs Linear Regression***

## ***Logistic Regression***

- We estimate the values of categorical variables using logistic regression.
- We discover the S-curve in Logistic Regression and use it to categorize the samples.
- It is not necessary to have a linear connection between the dependent and independent variables in logistic regression.
- There should be no collinearity between the independent variables in logistic regression.

## ***Linear Regression***

- We estimate the outcome of continuous variables using linear regression.
- In linear regression, we look for the best fit line, which allows us to predict the outcome with ease.
- It is necessary for the connection between the dependent and independent variables to be linear.
- There is a possibility of collinearity between the independent factors in linear regression.



## Quiz Time

Is Logistic regression mainly used for Regression?

- A) TRUE
- B) FALSE



## Quiz Time

Is Logistic regression mainly used for Regression?

- A) TRUE
- B) FALSE

Solution: B

Logistic regression is a classification algorithm, don't confuse with the name regression.

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

**Clustering**

Recommender System

Conclusion

## Introduction to clustering

K-Means Clustering Algorithm

Steps of the Elbow method

Hierarchical clustering

Implementation of k-means clustering in python

01

02

03

04

05

06

07

08

09

10

Use cases

Elbow method

Implementation in python

Density-based clustering

Importing the dataset

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

**Clustering**

Recommender System

Conclusion

Introduction to clustering

11

12

13

14

15

Use cases

Elbow method

K-Means Clustering Algorithm

Steps of the Elbow method



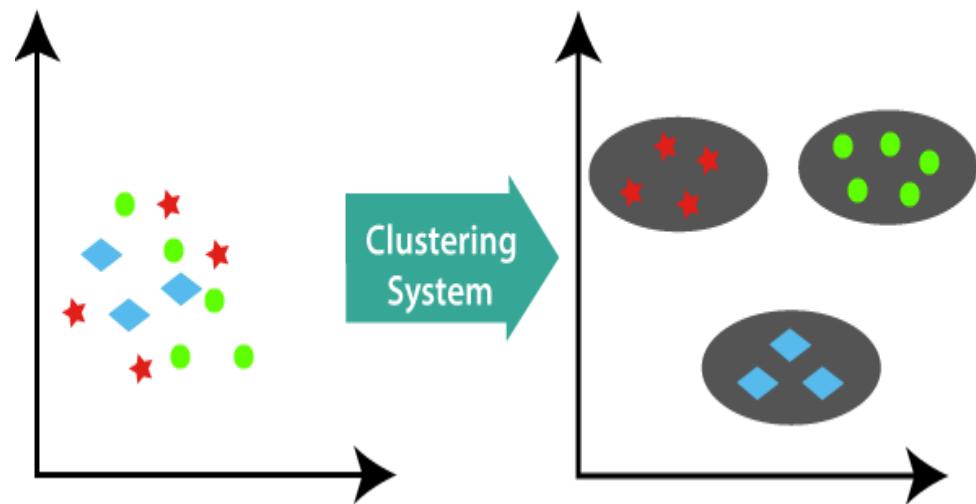
# Clustering

## *What is Clustering?*

Clustering is an unsupervised learning; it is a process of partitioning a set of data into a meaningful sub-classes.

## *Applications of Clustering*

- Pattern recognition
- Spatial data analysis
- Image processing
- Document classification
- Identification of similar entities
- Finding pattern of weather behavior



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

**Clustering**

Recommender System

Conclusion

Introduction to clustering

- 01
- 02 Use cases
- 03
- 04 Elbow method
- 05
- 06 Implementation in python
- 07
- 08 Density-based clustering
- 09
- 10 Importing the dataset

K-Means Clustering Algorithm

Steps of the Elbow method

Hierarchical clustering

Implementation of k-means clustering in python



# Clustering

## *Why do we need Clustering?*

Clustering can be used for the following tasks:

- Analyzing data from the research
- Creating a summary
- Detecting noise
- Duplicate detection
- Procedures for pre-processing

## *Algorithms of Clustering*

- K-Means Clustering
- Hierarchical Clustering
- Density-based Clustering

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

**Clustering**

Recommender System

Conclusion

## Introduction to clustering

### K-Means Clustering Algorithm

Steps of the Elbow method

Hierarchical clustering

Implementation of k-means clustering in python

- 01
- 02
- 03
- 04
- 05
- 06
- 07
- 08
- 09
- 10

Use cases

Elbow method

Implementation in python

Density-based clustering

Importing the dataset



# Clustering

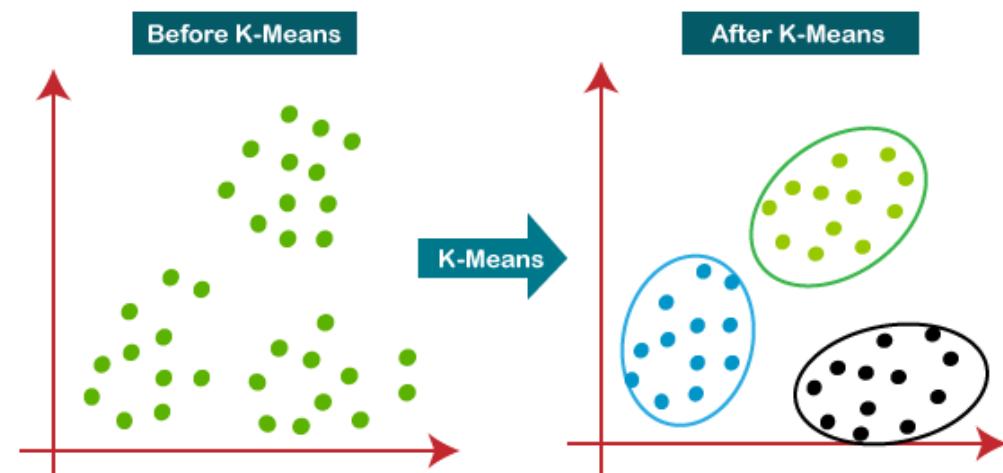
## K-Means Clustering

It is an iterative algorithm that divides the unlabeled dataset into  $k$  different clusters in such a way that each dataset belongs only one group that has similar properties.

## Algorithms of K-Clustering

Following is an algorithm of K-Means Clustering:

- Partition clustering
- Without any cluster-internal structure, K-Means divides data into non-overlapping groups.
- Within a cluster, examples are very similar.
- Various clusters have quite different examples.



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

**Clustering**

Recommender System

Conclusion

## Introduction to clustering

K-Means Clustering Algorithm

Steps of the Elbow method

Hierarchical clustering

Implementation of k-means clustering in python

- 01
- 02
- 03
- 04
- 05
- 06
- 07
- 08
- 09
- 10

Use cases

**Elbow method**

Implementation in python

Density-based clustering

Importing the dataset



# Clustering

## *Determination of K in K-means clustering*

The performance of the K-means clustering algorithm depends upon highly efficient clusters that it forms. But choosing the optimal number of clusters is a big task.

### *Elbow method*

This method uses the concept of WCSS value. WCSS stands for Within Cluster Sum of Squares, which defines the total variations within a cluster. The formula to calculate the value of WCSS.

$$\text{WCSS} = \sum_{P_i \text{ in Cluster}_1} \text{distance}(P_i | C_1)^2 + \sum_{P_i \text{ in Cluster}_2} \text{distance}(P_i | C_2)^2 + \sum_{P_i \text{ in Cluster}_3} \text{distance}(P_i | C_3)^2$$

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

**Clustering**

Recommender System

Conclusion

Introduction to clustering

01

Use cases

02

03

Elbow method

04

**Steps of the Elbow method**

05

Implementation in python

06

Hierarchical clustering

07

Density-based clustering

08

Implementation of k-means clustering in python

09

Importing the dataset

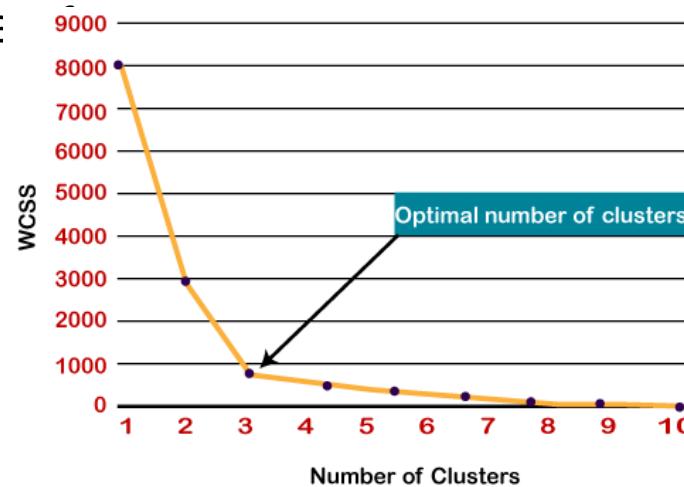
10



# Clustering

## *Steps of Elbow method*

- It executes the K-means clustering on a given dataset for different K values (ranges from 1-10).
- For each value of K, calculates the WCSS value.
- Plots a curve between calculated WCSS values and the number of clusters K.
- The sharp point of bend or a point of the plot looks like an arm, then that point is considered as the best value



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

**Clustering**

Recommender System

Conclusion

Introduction to clustering

- 01
- 02
- 03
- 04
- 05
- 06
- 07
- 08
- 09
- 10

Use cases

K-Means Clustering Algorithm

Steps of the Elbow method

Elbow method

**Implementation in python**

Hierarchical clustering

07

Density-based clustering

Implementation of k-means clustering in python

09

Importing the dataset

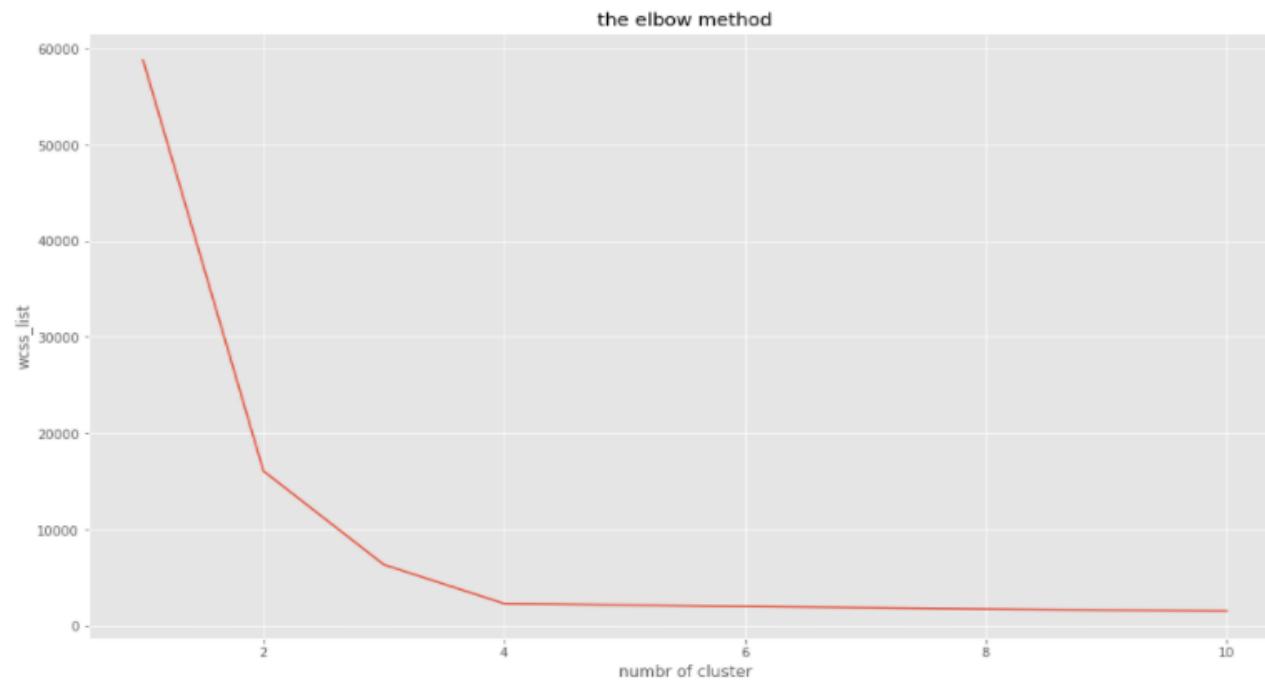


## The Elbow method

```
In [66]: # Finding the optimal number of clusters using elbow method
from sklearn.cluster import KMeans
wcss_list = [] # Intializing the list for the values of WCSS

10 iterations
for i in range(1,11):
 kmeans=KMeans(n_clusters=i, init='k-means++', random_state = 42)
 kmeans.fit(X)
 wcss_list.append(kmeans.inertia_)

plt.plot(range(1,11),wcss_list)
plt.title("the elbow method")
plt.xlabel("numbr of cluster")
plt.ylabel("wcss_list")
plt.show()
```



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

**Clustering**

Recommender System

Conclusion

Introduction to clustering

01

Use cases

02

K-Means Clustering Algorithm

03

Elbow method

Steps of the Elbow method

04

Implementation in python

05

**Hierarchical clustering**

06

Density-based clustering

07

Implementation of k-means clustering in python

08

09

Importing the dataset

10



# Clustering

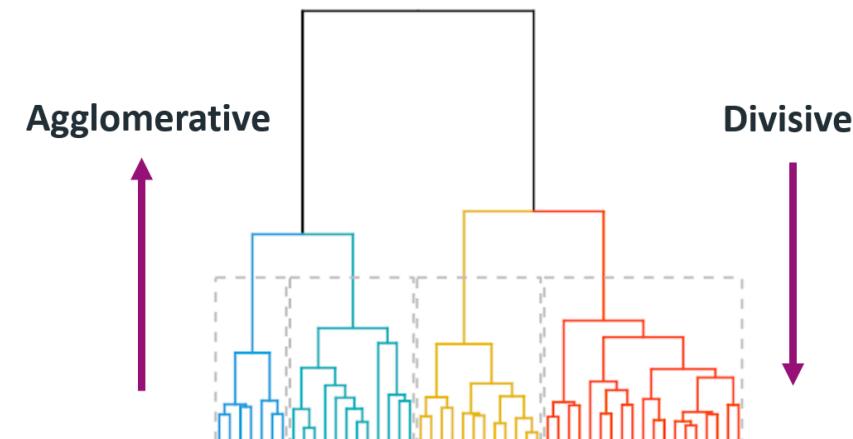
## *What is Hierarchical Clustering?*

It is an algorithm that groups similar objects into groups called clusters. The endpoint is a set of clusters, where each cluster is distinct from each other cluster, and the objects within each cluster are broadly like each other.

## *Types of Hierarchical Clustering*

There are two types of Hierarchical clustering:

- **Divisive clustering** is top to down approach in which observations starts from large cluster and splits into smaller ones.
- **Agglomerative clustering** is a bottom to top approach in which observations start from many clusters and merged as one cluster in the end.



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

**Clustering**

Recommender System

Conclusion

Introduction to clustering

- 01
- 02
- 03
- 04
- 05
- 06
- 07
- 08
- 09
- 10

Use cases

K-Means Clustering Algorithm

Steps of the Elbow method

Hierarchical clustering

Elbow method

Implementation in python

**Density-based clustering**

Implementation of k-means clustering in python

Importing the dataset



# Clustering

## *What is Density-based Clustering?*

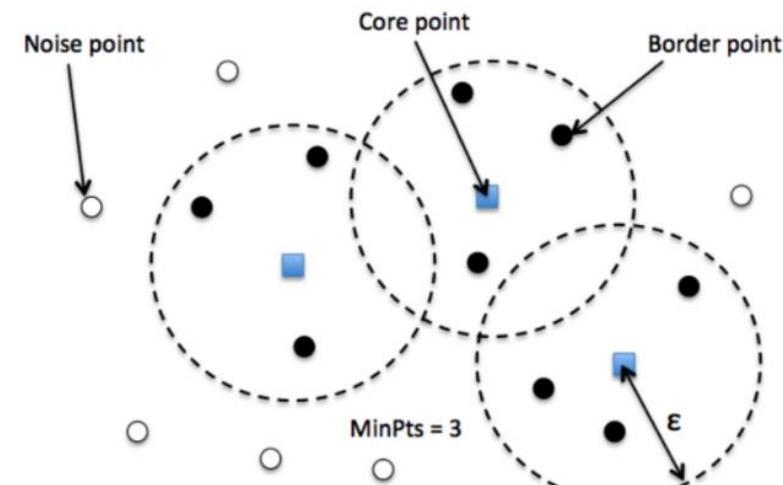
Density-based clustering locates regions of high density and separates outliers. So, the regions of high density get separated from the regions of low density.

## *What is DBSCAN?*

The most powerful attribute of DBSCAN algorithm is that it can find out any arbitrary shape cluster without getting effected by noise.

## *Advantages*

- It can discover arbitrarily shaped cluster.
- Find cluster surrounded by different cluster.
- Robust towards noise detection.



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

**Clustering**

Recommender System

Conclusion

Introduction to clustering

01

Use cases

02

03

Elbow method

04

Steps of the Elbow method

05

Implementation in python

06

Hierarchical clustering

07

Density-based clustering

08

**Implementation of k-means clustering in python**

09

Importing the dataset

10



# Clustering

## K-means clustering in python

### Implementation step 1

- Import the required dataset

```
In [18]: %matplotlib inline
from copy import deepcopy
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
```

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

**Clustering**

Recommender System

Conclusion

Introduction to clustering

- 01
- 02
- 03
- 04
- 05
- 06
- 07
- 08
- 09
- 10

Use cases

K-Means Clustering Algorithm

Steps of the Elbow method

Hierarchical clustering

Implementation of k-means clustering in python

Elbow method

Implementation in python

Density-based clustering

Importing the dataset



# Clustering

## K-means clustering in python

### Implementation step 2

- Import the dataset

```
In [19]: # Importing the dataset
data = pd.read_csv('mallCustomerData.txt', sep=',')
print(data.shape)
data.head(10)|
```

(200, 5)

Out[19]:

|   | CustomerID | Gender | Age | Annual Income (k\$) | Spending Score (1-100) |
|---|------------|--------|-----|---------------------|------------------------|
| 0 | 1          | Male   | 19  | 15                  | 39                     |
| 1 | 2          | Male   | 21  | 15                  | 81                     |
| 2 | 3          | Female | 20  | 16                  | 6                      |
| 3 | 4          | Female | 23  | 16                  | 77                     |
| 4 | 5          | Female | 31  | 17                  | 40                     |
| 5 | 6          | Female | 22  | 17                  | 76                     |
| 6 | 7          | Female | 35  | 18                  | 6                      |
| 7 | 8          | Female | 23  | 18                  | 94                     |
| 8 | 9          | Male   | 64  | 19                  | 3                      |
| 9 | 10         | Female | 30  | 19                  | 72                     |



# Clustering

## K-means clustering in python

### Implementation step 3

- HotEncoding of Gender values

```
In [5]: print(data['Gender'].value_counts())
```

```
Female 112
Male 88
Name: Gender, dtype: int64
```

```
In [6]: data = pd.get_dummies(data,columns=['Gender'])
print(data)
```

```
CustomerID Age Annual Income (k$) Spending Score (1-100) \
0 1 19 15 39
1 2 21 15 81
2 3 20 16 6
3 4 23 16 77
4 5 31 17 40
.. ...
195 196 35 120 79
196 197 45 126 28
197 198 32 126 74
198 199 32 137 18
199 200 30 137 83
```

```
Gender_Female Gender_Male
0 0 1
1 0 1
2 1 0
3 1 0
4 1 0
.. ...
195 1 0
196 1 0
197 0 1
198 0 1
199 0 1
```

```
[200 rows x 6 columns]
```

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

**Clustering**

Recommender System

Conclusion

Visualizing the dataset

3D Visualization of the clusters

Number of predicted clusters

11

12

13

14

15

Defining the classifier

3D Visualization of the predicted values



# Clustering

## K-means clustering in python

### Implementation step 4

- Visualize the dataset

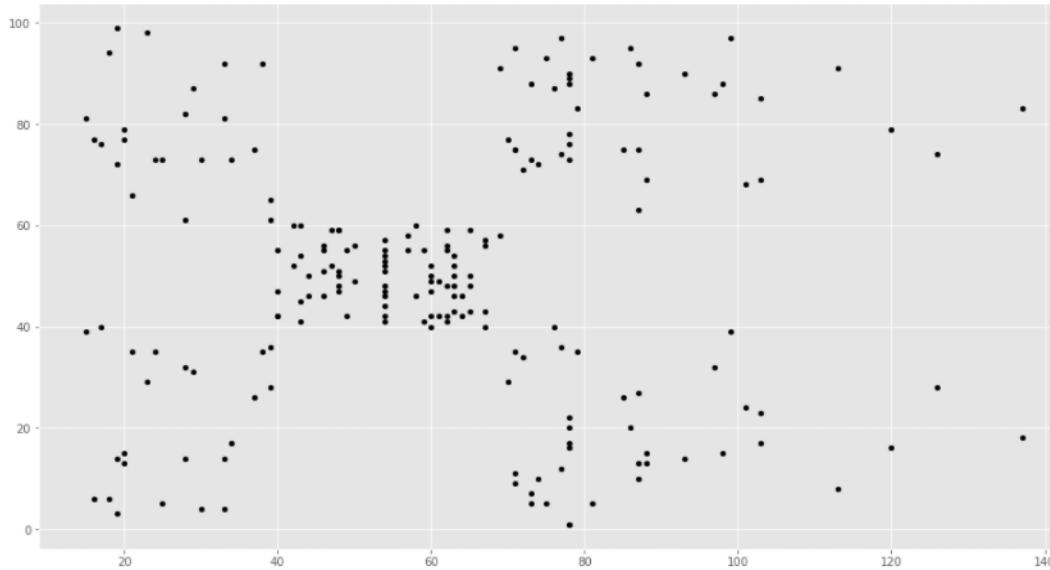
```
In [20]: # Getting the values and plotting it
f1 = data['Annual Income (k$)'].values
f2 = data['Spending Score (1-100)'].values

for key in data.keys():
 print(key)

X = np.array(list(zip(f1, f2)))
#X=[[1, 1], [1, 2], [3, 3],[4,4]]
plt.scatter(f1, f2, c='black', s=20)

CustomerID
Gender
Age
Annual Income (k$)
Spending Score (1-100)

Out[20]: <matplotlib.collections.PathCollection at 0x1460b208be0>
```



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

**Clustering**

Recommender System

Conclusion

Visualizing the dataset

11

12

13

14

15

**Defining the classifier**

3D Visualization of the clusters

3D Visualization of the predicted values

Number of predicted clusters



# Clustering

## K-means clustering in python

### Implementation step 5

- Defining the Classifier
- Fit the kmeans
- Print the values of clusters coordinates

```
In [50]: from sklearn.cluster import KMeans

Number of clusters
kmeans = KMeans(n_clusters=3)
Fitting the input data
kmeans = kmeans.fit(X)
Getting the cluster labels
labels = kmeans.predict(X)
Centroid values
C = kmeans.cluster_centers_
```

```
In [57]: # Values of the clusters
print(C)
```

```
[[-2.59662966 8.17293022 -3.18861476]
 [9.90577958 -1.78901987 4.83348761]
 [-2.67717112 -0.81623897 -7.81162806]
 [-5.19053091 0.683535 -2.16596538]]
```

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

**Clustering**

Recommender System

Conclusion

Visualizing the dataset

**3D Visualization of the clusters**

Number of predicted clusters

11

12

13

14

15

Defining the classifier

3D Visualization of the predicted values



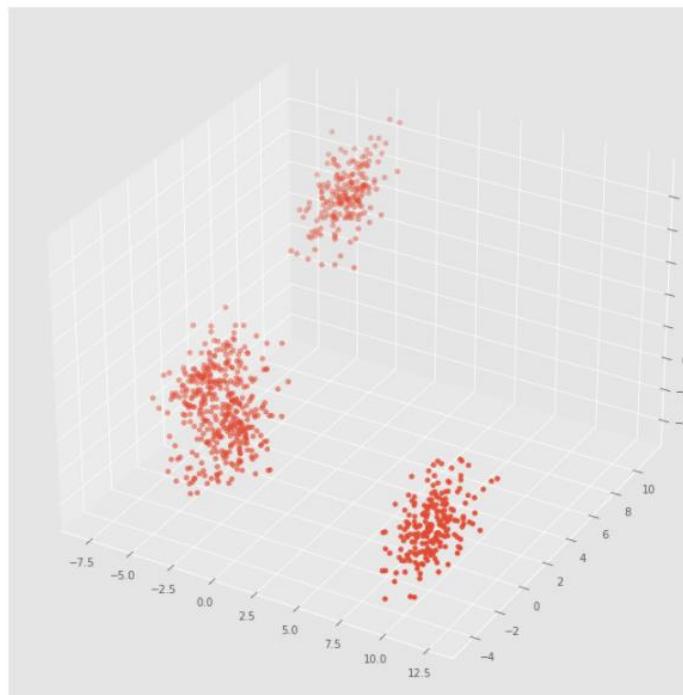
# Clustering

*K-means clustering in python*

**Implementation step 6**  
- A 3D plot of our dataset

```
In [41]: fig = plt.figure()
ax = Axes3D(fig)
ax.scatter(X[:, 0], X[:, 1], X[:, 2])

Out[41]: <mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x1460cc20cd0>
```



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

**Clustering**

Recommender System

Conclusion

Visualizing the dataset

11

12

13

14

15

Defining the classifier

3D Visualization of the clusters

**3D Visualization of the predicted values**

Number of predicted clusters



# Clustering

## K-means clustering in python

### Implementation step 7

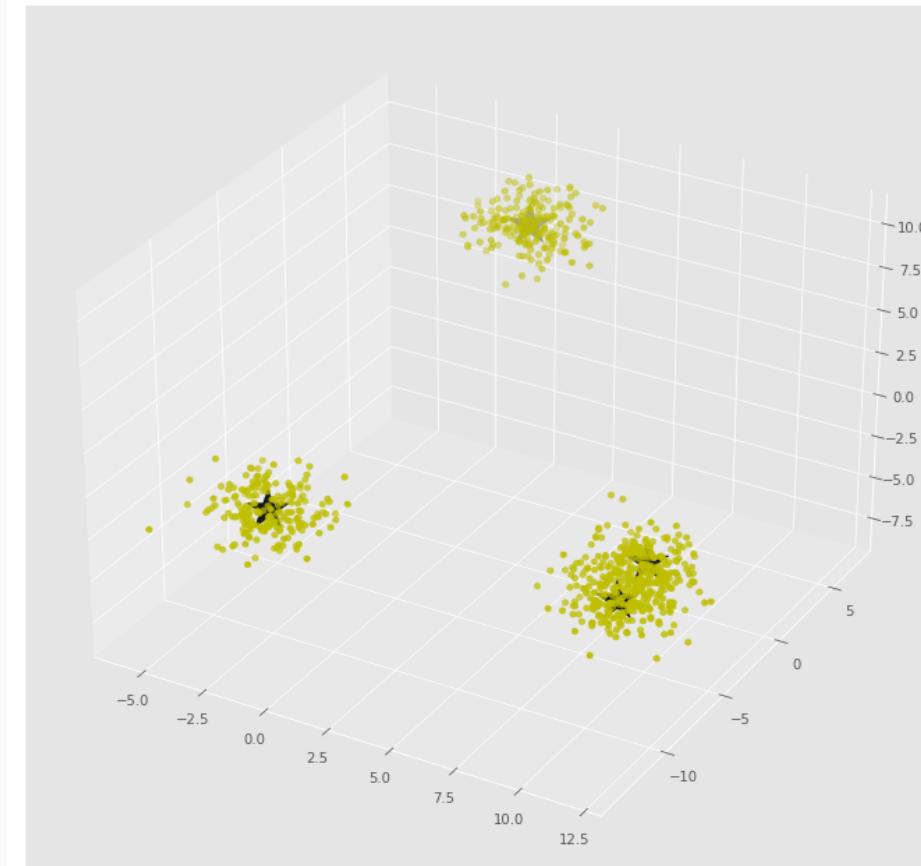
- 3D plot of the prediction

#### Clusters

```
In [57]: # Values of the clusters
print(C)
```

```
[[-2.59662966 8.17293022 -3.18861476]
 [9.90577958 -1.78901987 4.83348761]
 [-2.67717112 -0.81623897 -7.81162806]
 [-5.19053091 0.683535 -2.16596538]]
```

```
fig = plt.figure()
ax = Axes3D(fig)
ax.scatter(X[:, 0], X[:, 1], X[:, 2], c='y')
ax.scatter(C[:, 0], C[:, 1], C[:, 2], marker='*', c='#050505', s=1000)
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x1498114b5b0>
```



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

**Clustering**

Recommender System

Conclusion

Visualizing the dataset

11

12

13

14

15

Defining the classifier

3D Visualization of the clusters

**Number of predicted clusters**

3D Visualization of the predicted values



# Clustering

## K-means clustering in python

### Implementation step 7

#### Checking the predicted Values cluster number

```
In [67]: # Initializing KMeans
kmeans = KMeans(n_clusters=4)
Fitting with inputs
kmeans = kmeans.fit(X)
Predicting the clusters
labels = kmeans.predict(X)
Getting the cluster centers
c = kmeans.cluster_centers_
print(labels)
```

```
print(labels)|
[3 1 3 0 2 1 1 1 2 2 2 1 3 0 2 2 2 1 3 0 3 0 1 3 0 3 0 0 1 0 2 3 3 2 2 1 0
2 2 3 0 1 3 0 3 3 0 1 3 2 0 2 0 0 2 0 2 1 1 2 1 3 0 1 2 1 2 0 0 3 3 2 0 0
2 3 0 2 0 3 0 3 3 0 3 3 0 2 3 1 1 1 3 0 2 2 3 3 0 3 1 1 0 1 3 1 1 0 3 2 1
1 1 0 2 1 0 0 3 2 0 2 1 0 3 2 0 0 3 1 3 3 1 0 2 2 2 0 2 1 1 3 2 3 1 3 0 3
0 2 1 2 0 2 2 1 2 0 1 1 0 2 0 0 3 2 2 2 3 3 2 3 2 1 2 2 2 3 1 3 1 3
3 2 3 1 1 0 3 0 2 1 0 2 0 2 1 3 2 1 2 1 3 3 0 3 1 1 2 3 3 0 0 3 1 2 0 2 2
2 0 3 0 3 2 0 1 0 0 2 0 2 0 3 1 2 2 0 0 3 3 0 1 1 3 0 0 3 3 2 0 0 0 0 0 1
1 2 1 3 3 3 2 2 0 1 0 2 0 0 1 3 3 3 0 2 1 3 3 1 0 2 3 3 1 1 1 0 0 1 0 0
2 2 0 3 3 0 1 3 1 2 0 3 3 3 2 1 3 0 2 3 3 3 1 0 1 2 1 0 2 3 0 0 0 2 3 0 3
0 1 1 2 1 3 1 2 1 2 2 0 1 1 3 3 2 3 2 0 3 3 2 1 1 1 2 0 1 0 1 1 2 1 1 0 2
3 3 3 2 3 0 1 0 1 1 2 2 3 3 2 3 1 0 0 1 0 0 0 3 1 2 1 3 3 2 0 0 1 3 1 1 2
1 1 2 3 1 3 2 0 3 3 0 2 1 3 3 3 1 2 3 1 3 2 2 3 2 0 2 0 1 2 1 3 1 0 1 3 3
2 3 2 0 1 2 2 1 3 0 0 2 2 2 1 0 3 2 1 2 0 1 1 0 3 1 1 3 0 1 2 1 0 2 1 0 1
3 3 2 1 0 0 3 2 3 3 1 2 3 0 0 0 1 2 0 2 3 3 2 3 1 1 0 1 1 0 0 0 1 3 2 0 1
1 2 1 3 2 2 2 2 0 0 2 1 3 0 3 0 1 1 2 2 2 0 0 2 2 2 1 1 3 2 3 1 0 0 3 2 3
1 2 0 1 3 3 2 2 0 2 2 3 2 0 0 0 3 3 1 1 1 2 0 2 1 2 1 3 3 3 0 2 1 2 3 3 3
0 1 0 0 3 3 3 0 1 1 1 3 2 3 2 3 2 0 3 1 2 0 3 2 0 3 0 0 0 1 1 2 2 0 1 2 1
3 1 0 0 0 2 2 0 1 3 2 1 1 1 0 3 3 1 1 1 0 0 0 1 0 2 1 1 2 2 1 2 3 3 1 0 3
0 2 1 1 1 0 1 0 3 2 1 3 2 2 0 0 3 2 3 3 2 2 0 3 1 1 1 0 0 2 3 0 0 3 0 2 1
3 1 2 2 2 1 3 2 2 2 0 1 2 3 1 1 1 2 1 2 0 3 3 3 1 3 1 0 0 3 1 3 0 0 3 3 3
3 2 0 0 2 3 1 2 1 2 1 2 2 1 2 0 0 1 0 2 2 1 3 0 1 0 1 3 2 3 0 1 1 2 3 2 2
3 1 3 1 0 3 3 0 3 3 2 0 0 0 3 0 1 1 0 1 1 2 2]
```



## Quiz Time

Which of the following is required by K-means clustering?

- A) defined distance metric
- B) number of clusters
- C) initial guess as to cluster centroids
- D) all the mentioned



## Quiz Time

Which of the following is required by K-means clustering?

- A) defined distance metric
- B) number of clusters
- C) initial guess as to cluster centroids
- D) all the mentioned

Answer: D

Explanation: because K-means clustering follows partitioning approach.

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

**Recommender System**

Conclusion

## Introduction

01

Collaborative Filtering in Recommender Systems

02

Content-based Recommender System

03

Implementation in python:  
Importing libraries & datasets

04

Merging datasets into one dataframe

05

Sorting by title and rating

06

Histogram showing number of ratings

07

Frequency distribution

08

Jointplot of the ratings and number of ratings

09

Data pre-processing

10

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

**Recommender System**

Conclusion

Sorting the most-rated movies

11

12

13

14

15

16

17

Grab the ratings for two movies

Correlation between the most-rated movies

Filtering out movies

Sorting the data by correlation

Sorting values

Repeating the process for another movie



# Recommender system

## **What is Recommender System?**

Recommender systems are algorithms aimed at suggesting relevant items to users for example items being movies to watch, text to read, products to buy or anything else depending on industries.

## **Applications**

The applications of Recommender systems include movies, music, books, websites, documents, television programs.

## **Benefits**

- Increase customer engagement.
- Increase customer satisfaction by delivering relevant content.
- Reduce the time of content searching.



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

**Recommender System**

Conclusion

Introduction

01

**Collaborative Filtering in Recommender Systems**

02

Content-based Recommender System

03

Implementation in python:  
Importing libraries & datasets

04

Merging datasets into one dataframe

05

Sorting by title and rating

06

Histogram showing number of ratings

07

Frequency distribution

08

Jointplot of the ratings and number of ratings

09

Data pre-processing

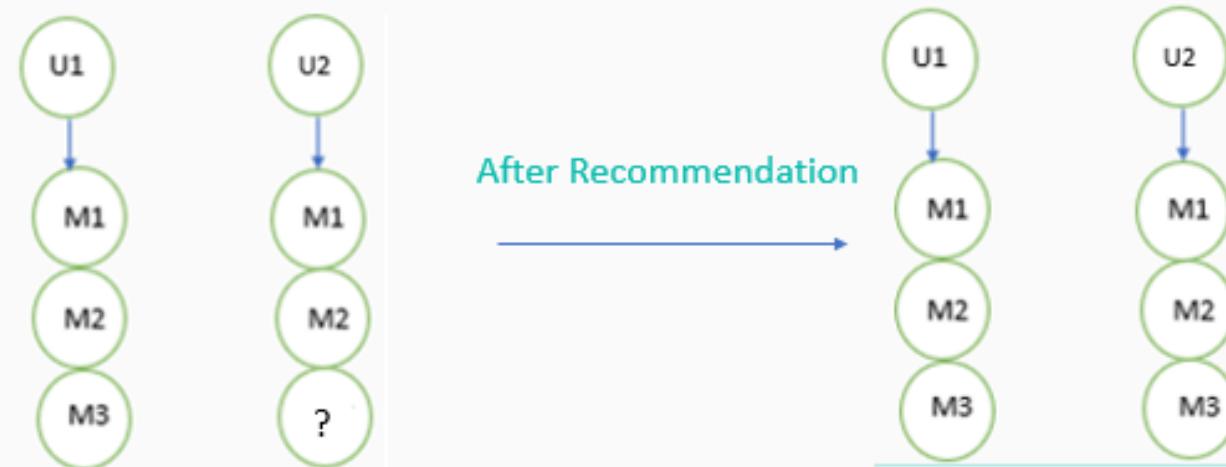
10



# Recommender system

## *Collaborative filtering Recommendation system*

The collaborative filtering method predicts the interests of a user on a product by collecting preferences information from many other users.



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

**Recommender System**

Conclusion

Introduction

01

Collaborative Filtering in Recommender Systems

02

## Content-based Recommender System

03

Implementation in python:  
Importing libraries & datasets

04

Merging datasets into one dataframe

05

Sorting by title and rating

06

Histogram showing  
number of ratings

07

Frequency distribution

08

Jointplot of the ratings  
and number of ratings

09

Data pre-processing

10



# Recommender system

## *Content-based Recommender System*

Content-based filtering method utilizes product features to recommend other products like what the user likes, based on other users' previous actions or explicit feedback such as rating on products.



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

**Recommender System**

Conclusion

Introduction

01

Collaborative Filtering in Recommender Systems

02

Content-based Recommender System

03

**Implementation in python:  
Importing libraries & datasets**

04

Merging datasets into one dataframe

05

Sorting by title and rating

06

Histogram showing number of ratings

07

Frequency distribution

08

Jointplot of the ratings and number of ratings

09

Data pre-processing

10



# Recommender system

## Recommender System in python Implementation step 1

- Import the libraries
- Get the datasets

```
In [4]: movie_titles = pd.read_csv("Movie_Id_Titles")
movie_titles.head()
```

Out[4]:

|   | item_id | title             |
|---|---------|-------------------|
| 0 | 1       | Toy Story (1995)  |
| 1 | 2       | GoldenEye (1995)  |
| 2 | 3       | Four Rooms (1995) |
| 3 | 4       | Get Shorty (1995) |
| 4 | 5       | Copycat (1995)    |

```
In [1]: import numpy as np
import pandas as pd

#for visiulization
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('white')
%matplotlib inline
```

```
In [2]: column_names = ['user_id', 'item_id', 'rating', 'timestamp']
df = pd.read_csv('u.data', sep='\t', names=column_names)
```

```
In [3]: df.head()
```

```
Out[3]:
```

|   | user_id | item_id | rating | timestamp |
|---|---------|---------|--------|-----------|
| 0 | 0       | 50      | 5      | 881250949 |
| 1 | 0       | 172     | 5      | 881250949 |
| 2 | 0       | 133     | 1      | 881250949 |
| 3 | 196     | 242     | 3      | 881250949 |
| 4 | 186     | 302     | 3      | 891717742 |

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

**Recommender System**

Conclusion

Introduction

01

Collaborative Filtering in Recommender Systems

02

Content-based Recommender System

03

Implementation in python:  
Importing libraries & datasets

04

**Merging datasets into one dataframe**

05

Sorting by title and rating

06

Histogram showing  
number of ratings

07

Frequency distribution

08

Jointplot of the ratings  
and number of ratings

09

Data pre-processing

10



# Recommender system

## Recommender System in python Implementation step 2

- Merge the two tables
- Using item\_id

```
In [5]: df = pd.merge(df,movie_titles, on='item_id')
df.head()
```

```
Out[5]:
```

|   | user_id | item_id | rating | timestamp | title            |
|---|---------|---------|--------|-----------|------------------|
| 0 | 0       | 50      | 5      | 881250949 | Star Wars (1977) |
| 1 | 290     | 50      | 5      | 880473582 | Star Wars (1977) |
| 2 | 79      | 50      | 4      | 891271545 | Star Wars (1977) |
| 3 | 2       | 50      | 5      | 888552084 | Star Wars (1977) |
| 4 | 8       | 50      | 5      | 879362124 | Star Wars (1977) |

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

**Recommender System**

Conclusion

Introduction

01

Collaborative Filtering in Recommender Systems

02

Content-based Recommender System

03

Implementation in python:  
Importing libraries & datasets

04

Merging datasets into one dataframe

05

**Sorting by title and rating**

06

Histogram showing  
number of ratings

07

Frequency distribution

08

Jointplot of the ratings  
and number of ratings

09

Data pre-processing

10



# Recommender system

## Recommender System in python

### Implementation step 3

- Sort the rows by group of Title, and rating

**Find :-**

- Count
- Mean

```
In [8]: df.groupby('title')['rating'].count().sort_values(ascending=False).head()
```

```
Out[8]: title
Star Wars (1977) 584
Contact (1997) 509
Fargo (1996) 508
Return of the Jedi (1983) 507
Liar Liar (1997) 485
Name: rating, dtype: int64
```

```
In [7]: df.groupby('title')['rating'].mean().sort_values(ascending=False).head()
```

```
Out[7]: title
They Made Me a Criminal (1939) 5.0
Marlene Dietrich: Shadow and Light (1996) 5.0
Saint of Fort Washington, The (1993) 5.0
Someone Else's America (1995) 5.0
Star Kid (1997) 5.0
Name: rating, dtype: float64
```



# Recommender system

## Recommender System in python

### Implementation step 4

- Sort the rows by group of titles and rating

```
In [9]: ratings = pd.DataFrame(df.groupby('title')['rating'].mean())
ratings.head()
```

Out[9]:

|                           | rating   |
|---------------------------|----------|
| title                     |          |
| 'Til There Was You (1997) | 2.333333 |
| 1-900 (1994)              | 2.600000 |
| 101 Dalmatians (1996)     | 2.908257 |
| 12 Angry Men (1957)       | 4.344000 |
| 187 (1997)                | 3.024390 |

```
In [10]: ratings['num of ratings'] = pd.DataFrame(df.groupby('title')['rating'].count())
ratings.head()
```

Out[10]:

|                           | rating   | num of ratings |
|---------------------------|----------|----------------|
| title                     |          |                |
| 'Til There Was You (1997) | 2.333333 | 9              |
| 1-900 (1994)              | 2.600000 | 5              |
| 101 Dalmatians (1996)     | 2.908257 | 109            |
| 12 Angry Men (1957)       | 4.344000 | 125            |
| 187 (1997)                | 3.024390 | 41             |

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

**Recommender System**

Conclusion

Introduction

01

Collaborative Filtering in Recommender Systems

02

Content-based Recommender System

03

Implementation in python:  
Importing libraries & datasets

04

Merging datasets into one dataframe

05

Sorting by title and rating

06

Histogram showing number of ratings

07

Frequency distribution

08

Jointplot of the ratings and number of ratings

09

Data pre-processing

10



# Recommender system

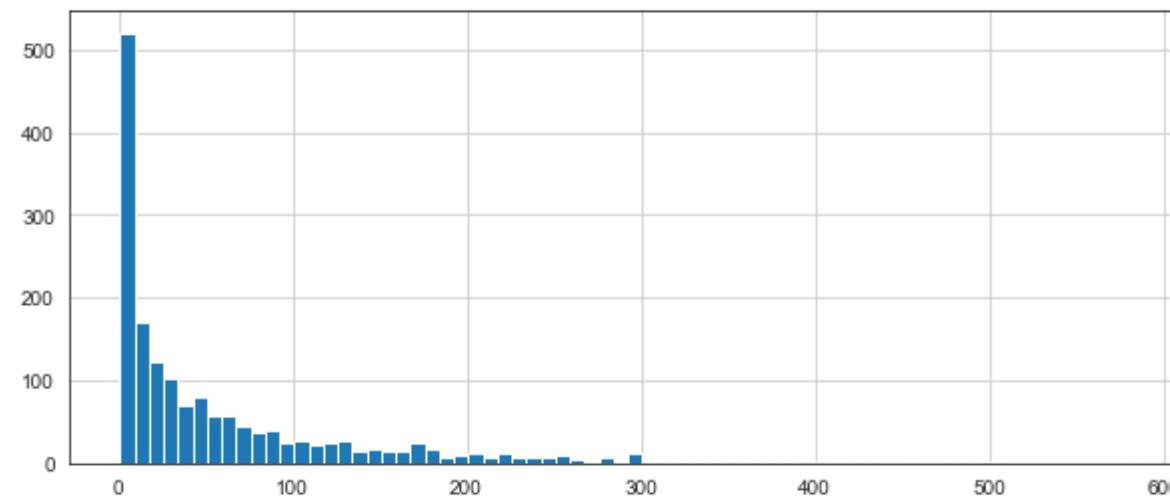
## Recommender System in python

### Implementation step 5

- Plot the Newly created column

```
In [11]: plt.figure(figsize=(10,4))
ratings['num of ratings'].hist(bins=70)
```

```
Out[11]: <AxesSubplot:>
```



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

**Recommender System**

Conclusion

Introduction

01

Collaborative Filtering in Recommender Systems

02

Content-based Recommender System

03

Implementation in python:  
Importing libraries & datasets

04

Merging datasets into one dataframe

05

Sorting by title and rating

06

Histogram showing number of ratings

07

**Frequency distribution**

08

Jointplot of the ratings and number of ratings

09

Data pre-processing

10



# Recommender system

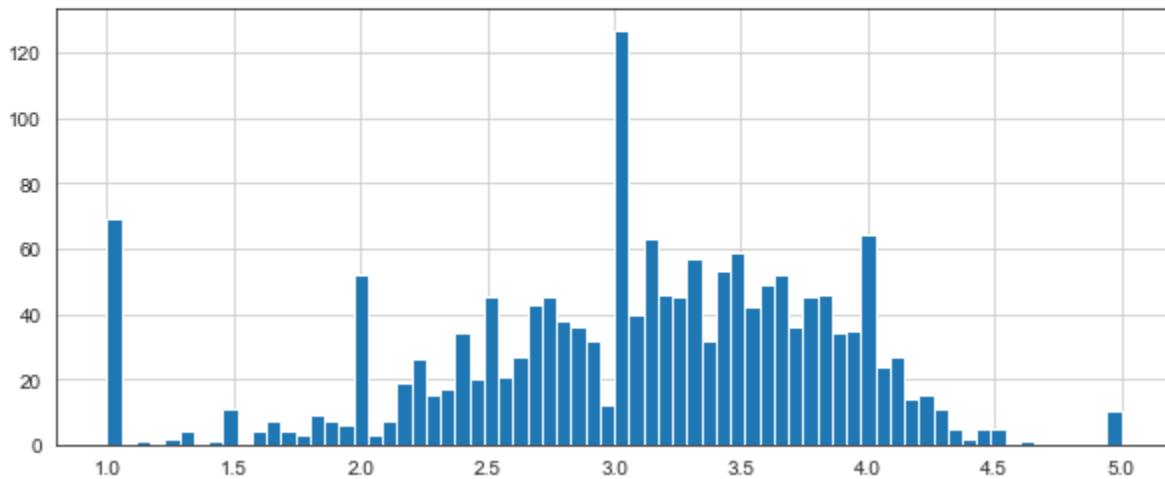
## Recommender System in python

### Implementation step 6

- Histogram of the ratings

```
In [12]: plt.figure(figsize=(10,4))
ratings['rating'].hist(bins=70)
```

```
Out[12]: <AxesSubplot:>
```



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

**Recommender System**

Conclusion

Introduction

01

Collaborative Filtering in Recommender Systems

02

Content-based Recommender System

03

Implementation in python:  
Importing libraries & datasets

04

Merging datasets into one dataframe

05

Sorting by title and rating

06

Histogram showing number of ratings

07

Frequency distribution

08

**Jointplot of the ratings and number of ratings**

09

Data pre-processing

10



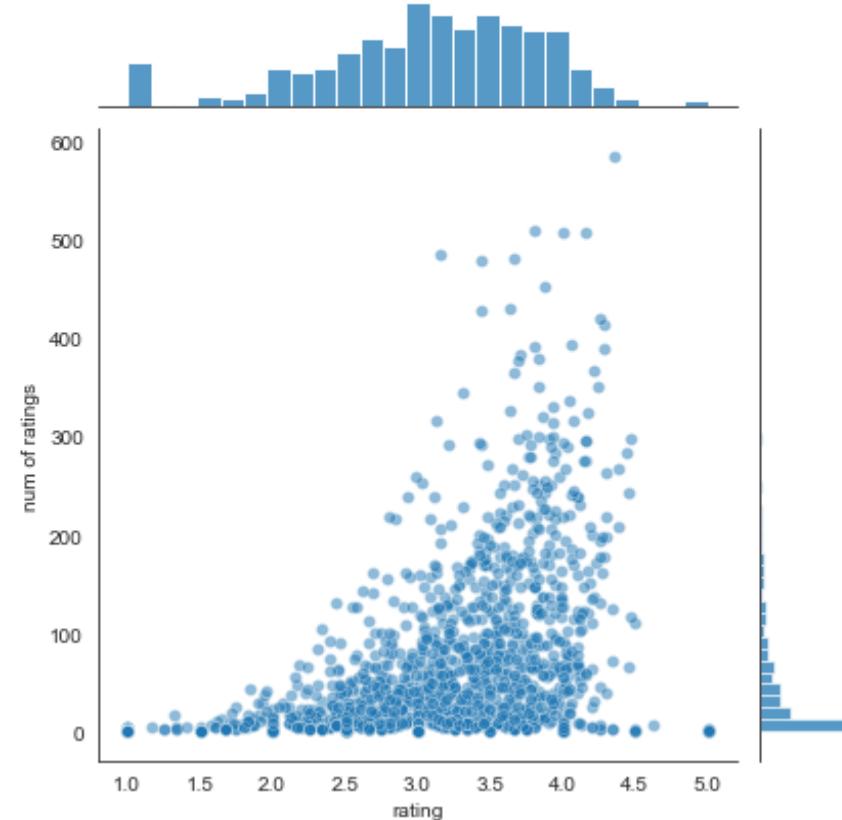
# Recommender system

## Recommender System in python Implementation step 7

- A combined plot of number of Ratings and number of ratings

```
In [13]: sns.jointplot(x='rating',y='num of ratings',data=ratings,alpha=0.5)
```

```
Out[13]: <seaborn.axisgrid.JointGrid at 0x1d21cb69d00>
```



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

**Recommender System**

Conclusion

Introduction

01

Collaborative Filtering in Recommender Systems

02

Content-based Recommender System

03

Implementation in python:  
Importing libraries & datasets

04

Merging datasets into one dataframe

05

Sorting by title and rating

06

Histogram showing  
number of ratings

07

Frequency distribution

08

Jointplot of the ratings  
and number of ratings

09

Data pre-processing

10



# Recommender system

## Recommender System in python

### Implementation step 8

Importing the dataset movies and pre-processing it

```
In [14]: moviemat = df.pivot_table(index='user_id',columns='title',values='rating')
moviemat.head()
```

Out[14]:

| title   | 'Til<br>There<br>Was<br>You<br>(1997) | 1-900<br>(1994) | Dalmatians<br>(1996) | 101<br>Angry<br>Men<br>(1957) | 12<br>(1997) | 187<br>Days<br>in the<br>Valley<br>(1996) | 2<br>Leagues<br>Under<br>the Sea<br>(1954) | 20,000<br>Space<br>Odyssey<br>(1968) | 3 Ninjas:<br>High<br>Noon At<br>Mega<br>Mountain<br>(1998) | 39<br>Steps,<br>The<br>(1935) | ...<br>Yankee<br>Zulu<br>(1994) | Year<br>of the<br>Horse<br>(1997) | You<br>So<br>Crazy<br>(1994) | Frankenstein<br>(1974) | Young<br>Guns<br>(1988) | Young<br>Guns<br>II<br>(1990) | Young<br>Poisc<br>Hand<br>The<br>( |
|---------|---------------------------------------|-----------------|----------------------|-------------------------------|--------------|-------------------------------------------|--------------------------------------------|--------------------------------------|------------------------------------------------------------|-------------------------------|---------------------------------|-----------------------------------|------------------------------|------------------------|-------------------------|-------------------------------|------------------------------------|
| user_id | 0                                     | NaN             | NaN                  | NaN                           | NaN          | NaN                                       | NaN                                        | NaN                                  | NaN                                                        | NaN                           | NaN                             | NaN                               | NaN                          | NaN                    | NaN                     | NaN                           |                                    |
| 1       | NaN                                   | NaN             | 2.0                  | 5.0                           | NaN          | NaN                                       | 3.0                                        | 4.0                                  | NaN                                                        | NaN                           | ...                             | NaN                               | NaN                          | NaN                    | 5.0                     | 3.0                           | NaN                                |
| 2       | NaN                                   | NaN             | NaN                  | NaN                           | NaN          | NaN                                       | NaN                                        | NaN                                  | 1.0                                                        | NaN                           | ...                             | NaN                               | NaN                          | NaN                    | NaN                     | NaN                           | NaN                                |
| 3       | NaN                                   | NaN             | NaN                  | NaN                           | NaN          | 2.0                                       | NaN                                        | NaN                                  | NaN                                                        | NaN                           | ...                             | NaN                               | NaN                          | NaN                    | NaN                     | NaN                           | NaN                                |
| 4       | NaN                                   | NaN             | NaN                  | NaN                           | NaN          | NaN                                       | NaN                                        | NaN                                  | NaN                                                        | NaN                           | ...                             | NaN                               | NaN                          | NaN                    | NaN                     | NaN                           | NaN                                |

5 rows × 1664 columns

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

**Recommender System**

Conclusion

**Sorting the most-rated movies**

11

Grab the ratings for two movies

12

13

Sorting the data by correlation

14

Correlation between the most-rated movies

15

Filtering out movies

16

Sorting values

17

Repeating the process for another movie



# Recommender system

## Recommender System in python Implementation step 9

- Sorting the most rated movies

In [15]: `ratings.sort_values('num of ratings', ascending=False).head(10)`

Out[15]:

|                               | rating   | num of ratings |
|-------------------------------|----------|----------------|
|                               | title    |                |
| Star Wars (1977)              | 4.359589 | 584            |
| Contact (1997)                | 3.803536 | 509            |
| Fargo (1996)                  | 4.155512 | 508            |
| Return of the Jedi (1983)     | 4.007890 | 507            |
| Liar Liar (1997)              | 3.156701 | 485            |
| English Patient, The (1996)   | 3.656965 | 481            |
| Scream (1996)                 | 3.441423 | 478            |
| Toy Story (1995)              | 3.878319 | 452            |
| Air Force One (1997)          | 3.631090 | 431            |
| Independence Day (ID4) (1996) | 3.438228 | 429            |

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

**Recommender System**

Conclusion

Sorting the most-rated movies

11

12

13

14

15

16

17

**Grab the ratings for two movies**

Correlation between the most-rated movies

Filtering out movies

Sorting the data by correlation

Sorting values

Repeating the process for another movie



# Recommender system

## Recommender System in python

### Implementation step 10

- Grab the Users ratings for two Movies
- Star wars (1997)
- Liar Liar (1997)

```
In [17]: starwars_user_ratings = moviemat['Star Wars (1977)']
liarliar_user_ratings = moviemat['Liar Liar (1997)']
starwars_user_ratings.head()

Out[17]: user_id
0 5.0
1 5.0
2 5.0
3 NaN
4 5.0
Name: Star Wars (1977), dtype: float64
```

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

**Recommender System**

Conclusion

Sorting the most-rated movies

11

12

13

14

15

16

17

Grab the ratings for two movies

**Correlation between the most-rated movies**

Filtering out movies

Sorting the data by correlation

Sorting values

Repeating the process for another movie



# Recommender system

## Recommender System in python Implementation step 11

- Use the corrwith() method to get the correlation between our most rated movies.

```
In [18]: similar_to_starwars = moviemat.corrwith(starwars_user_ratings)
similar_to_liarliar = moviemat.corrwith(liarliar_user_ratings)

C:\Users\jawad\anaconda3\lib\site-packages\numpy\lib\function_base.py:2634: RuntimeWarning: Degrees of freedom <= 0 for slice
 c = cov(x, y, rowvar, dtype=dtype)
C:\Users\jawad\anaconda3\lib\site-packages\numpy\lib\function_base.py:2493: RuntimeWarning: divide by zero encountered in true_
divide
 c *= np.true_divide(1, fact)
```

# Error !



# Recommender system

*Recommender System in python  
Implementation step 12*

## Error !

## Lets remove it

```
In [19]: corr_starwars = pd.DataFrame(similar_to_starwars,columns=['Correlation'])
corr_starwars.dropna(inplace=True)
corr_starwars.head(5)
```

Out[19]:

| Correlation               |           |
|---------------------------|-----------|
|                           | title     |
| 'Til There Was You (1997) | 0.872872  |
| 1-900 (1994)              | -0.645497 |
| 101 Dalmatians (1996)     | 0.211132  |
| 12 Angry Men (1957)       | 0.184289  |
| 187 (1997)                | 0.027398  |

Clean all the NaN and Use DataFrame instead of Series

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

**Recommender System**

Conclusion

Sorting the most-rated movies

11

Grab the ratings for two movies

12

Correlation between the most-rated movies

13

**Sorting the data by correlation**

14

Filtering out movies

15

Sorting values

16

Repeating the process for another movie

17



# Recommender system

## Recommender System in python Implementation step 13

Sorting by Correlation

```
In [20]: corr_starwars.sort_values('Correlation', ascending=False).head(10)
```

Out[20]:

|  | Correlation | title                                                                             |
|--|-------------|-----------------------------------------------------------------------------------|
|  |             | Commandments (1997)                                                               |
|  |             | Cosi (1996)                                                                       |
|  |             | No Escape (1994)                                                                  |
|  |             | Stripes (1981)                                                                    |
|  |             | Man of the Year (1995)                                                            |
|  |             | Hollow Reed (1996)                                                                |
|  |             | Beans of Egypt, Maine, The (1994)                                                 |
|  |             | Good Man in Africa, A (1994)                                                      |
|  |             | Old Lady Who Walked in the Sea, The (Vieille qui marchait dans la mer, La) (1991) |
|  |             | Outlaw, The (1943)                                                                |

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

**Recommender System**

Conclusion

Sorting the most-rated movies

11

12

13

14

15

16

17

Grab the ratings for two movies

Correlation between the most-rated movies

**Filtering out movies**

Sorting the data by correlation

Sorting values

Repeating the process for another movie



# Recommender system

## Recommender System in python

### Implementation step 14

Filter out the movies having less than 100 reviews

```
In [21]: corr_starwars = corr_starwars.join(ratings['num of ratings'])
corr_starwars.head()
```

Out[21]:

| title                     | Correlation | num of ratings |
|---------------------------|-------------|----------------|
| 'Til There Was You (1997) | 0.872872    | 9              |
| 1-900 (1994)              | -0.645497   | 5              |
| 101 Dalmatians (1996)     | 0.211132    | 109            |
| 12 Angry Men (1957)       | 0.184289    | 125            |
| 187 (1997)                | 0.027398    | 41             |

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

**Recommender System**

Conclusion

Sorting the most-rated movies

11

12

13

14

15

16

17

Grab the ratings for two movies

Correlation between the most-rated movies

Filtering out movies

Sorting the data by correlation

**Sorting values**

Repeating the process for another movie



# Recommender system

**Recommender System in python**

**Implementation step 15**

Finally Filter out the titles to make sense

```
In [22]: corr_starwars[corr_starwars['num of ratings']>100].sort_values('Correlation', ascending=False).head()
```

```
Out[22]:
```

|  | Correlation                                        | num of ratings |
|--|----------------------------------------------------|----------------|
|  | title                                              |                |
|  | Star Wars (1977)                                   | 1.000000       |
|  | Empire Strikes Back, The (1980)                    | 0.748353       |
|  | Return of the Jedi (1983)                          | 0.672556       |
|  | Raiders of the Lost Ark (1981)                     | 0.536117       |
|  | Austin Powers: International Man of Mystery (1997) | 0.377433       |

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

**Recommender System**

Conclusion

Sorting the most-rated movies

11

Grab the ratings for two movies

12

Correlation between the most-rated movies

13

Sorting the data by correlation

14

Filtering out movies

15

Sorting values

16

Repeating the process for another movie

17



# Recommender system

## Recommender System in python

### Implementation step 16

Doing same for the movie Liar Liar (1997) (Implementation step 11-15)

```
In [23]: corr_liarliar = pd.DataFrame(similar_to_liarliar,columns=['Correlation'])
corr_liarliar.dropna(inplace=True)
corr_liarliar = corr_liarliar.join(ratings['num of ratings'])
corr_liarliar[corr_liarliar['num of ratings']>100].sort_values('Correlation',ascending=False).head()
```

Out[23]:

Correlation num of ratings

| title                 | Correlation | num of ratings |
|-----------------------|-------------|----------------|
| Liar Liar (1997)      | 1.000000    | 485            |
| Batman Forever (1995) | 0.516968    | 114            |
| Mask, The (1994)      | 0.484650    | 129            |
| Down Periscope (1996) | 0.472681    | 101            |
| Con Air (1997)        | 0.469828    | 137            |



## Quiz Time

Why are recommendation engines becoming popular?

- A) Users have less time, more options and face an information overload
- B) It is mandatory to have recommendation engine as per telecom rules
- C) It is better to recommend than ask user to search on cell phones
- D) Users don't know what they want



## Quiz Time

Why are recommendation engines becoming popular?

- A) Users have less time, more options and face an information overload
- B) It is mandatory to have recommendation engine as per telecom rules
- C) It is better to recommend than ask user to search on cell phones
- D) Users don't know what they want



# *Congratulations!*

*Thank You!*