While we wait for the tutorial to begin

GET SET UP TO FOLLOW THE TUTORIAL

1. Navigate to **https://colab.research.google.com/**
2. Click on the '**GitHub**' tab.
3. Enter '**UCLAIS/ml-tutorials-season-3**'.
4. Click '**week-2/1 - Introduction to Python.ipynb**'

UCL ARTIFICIAL
INTELLIGENCE SOCIETY

TUTORIAL #2

Session 2

An introduction to Python & scientific computing libraries

Introduction

WHO WE ARE

Jeremy Lo Ying Ping

Head of Development

Aliff Noorisman

Machine Learning Officer

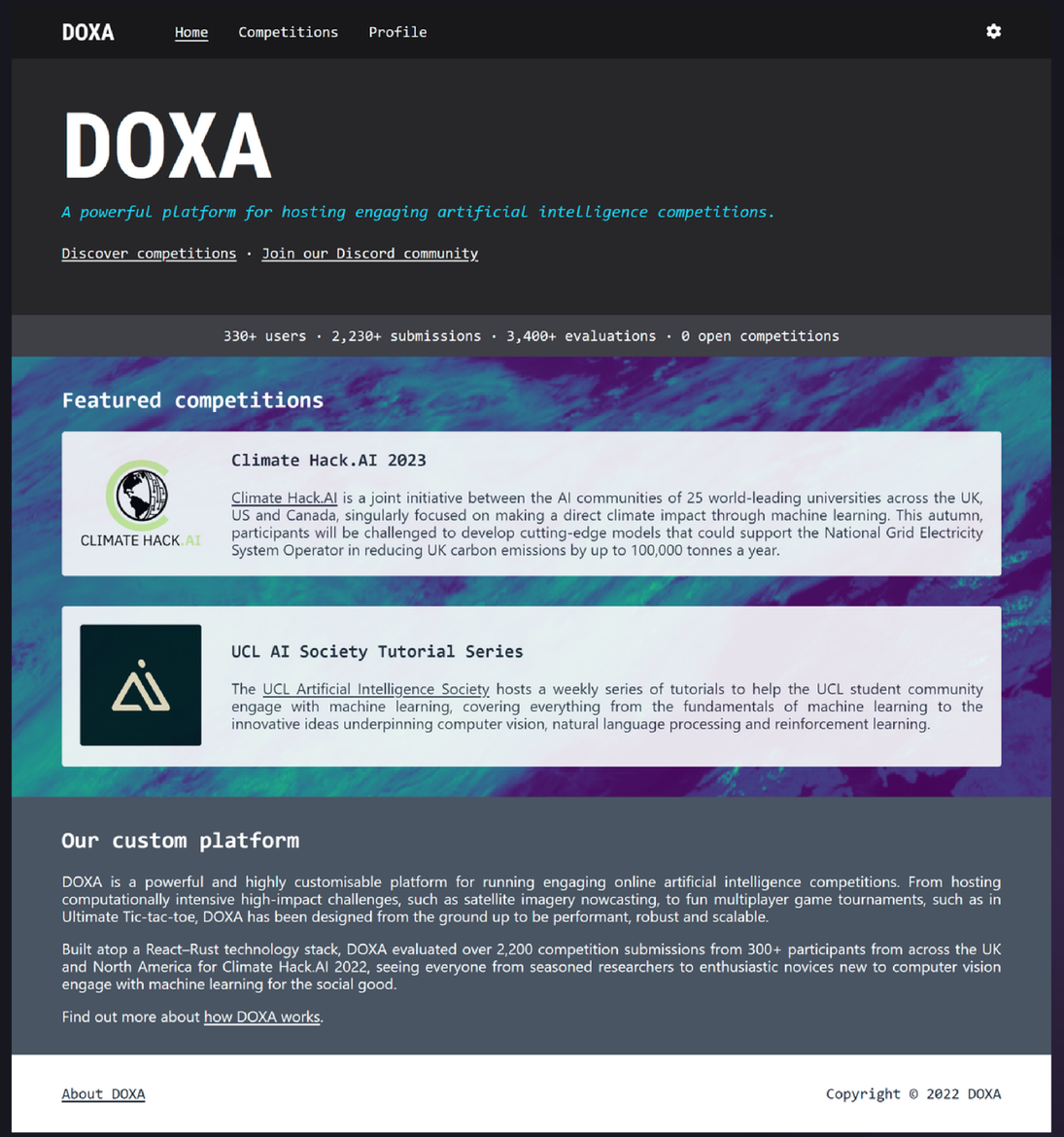
Introduction

DOXA

What does the development team work on?

We develop DOXA: a customisable platform for hosting engaging artificial intelligence competitions, such as the upcoming UCLAIS tutorial series challenges. Look out for those!

For the most part, DOXA is built using Rust, Python & React (JavaScript).



UCL ARTIFICIAL
INTELLIGENCE SOCIETY

LECTURE OVERVIEW

01 | **Using Colab**
How to follow along with the tutorial

02 | **A tour of Python**
Getting familiar with Python's syntax

03 | **Useful Libraries**
Common libraries in the scientific computing world

04 | **Questions**
We would be more than happy to take any questions!

Introduction

HOW TO FOLLOW ALONG WITH THIS TUTORIAL

You don't need to install anything!
You only need a web browser.

For the first part of the tutorial, we'll be doing a fast-paced whistle-stop tour of Python to show you examples of what Python code looks like and some of the things it can do.

Don't worry if you feel like this is going quickly! There will be plenty of time afterwards for you to go through the code examples and ask us questions, either after this tutorial or on the society discord!

Introduction

HOW TO FOLLOW ALONG WITH THIS TUTORIAL

Resources for this tutorial are available on GitHub: https://github.com/UCLAIS/ml-tutorials-season-3

It can be accessed via https://linktr.ee/uclaisociety

You will find the first Jupyter notebook at "**week-2/Introduction to Python.ipynb**".

https://github.com/UCLAIS/ml-tutorials-season-3/blob/main/week-2/Introduction%20to%20Python.ipynb

HOW TO FOLLOW ALONG WITH THIS TUTORIAL

1. Navigate to **https://colab.research.google.com/**
2. Click on the '**GitHub**' tab.
3. Enter '**UCLAIS/ml-tutorials-season-3**' (or paste in *https://github.com/UCLAIS/ml-tutorials-season-3/blob/main/week-2/1%20-%20Introduction%20to%20Python.ipynb*).
4. Click '**week-2/1 - Introduction to Python.ipynb**'

Introduction

DEVELOPING WITH PYTHON LOCALLY

You will not need to install Python locally for this tutorial, but you may wish to do so afterwards if you have not yet done so!

There are several different ways to install Python locally:

- Directly from the Python website (https://www.python.org/downloads/)
- Using miniconda3 (https://docs.conda.io/en/latest/miniconda.html)
- Through a package manager (primarily for Linux users, although you can on other systems, e.g. Windows using chocolatey)

For the time being, you may wish to stick to Python 3.9 for better machine learning library compatibility, even if Python 3.11 is about to be released.

Introduction

CODE EDITORS

https://code.visualstudio.com/

While you could write Python in *notepad.exe* if you really wanted to, using a proper code editor or IDE brings several advantages, such as syntax highlighting and checking, code formatting, autocompletion, linting and so on.

Visual Studio Code with the Python extension pack (and GitLens!) is a great option if you would like to follow along with the tutorial series on your local machine.

Introduction

WHAT IS PYTHON?

Introduction

THE ZEN OF PYTHON, BY TIM PETERS

>>> import this

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!

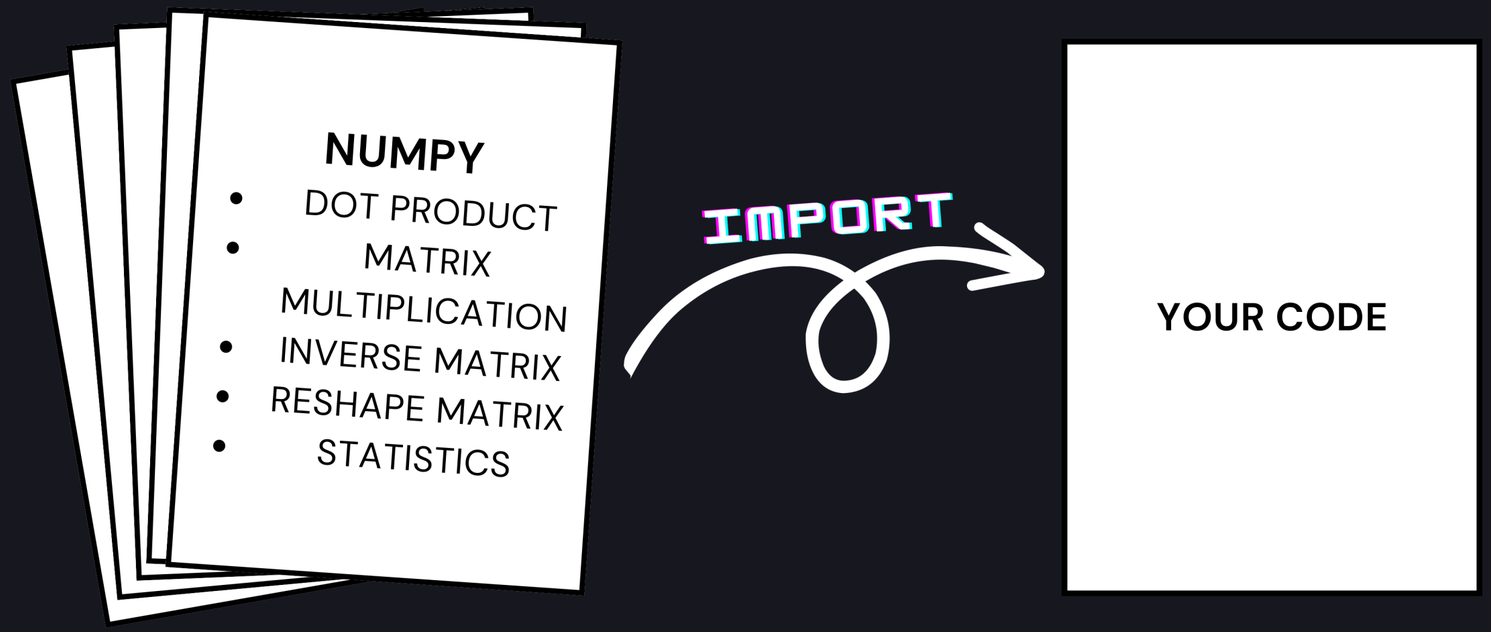
LET'S SWITCH OVER TO THE JUPYTER NOTEBOOK

IT'S TIME FOR US TO GET INTO LIBRARIES

Libraries for scientific computing

WHAT IS A LIBRARY?

A collection of codes that we can use in a program for specific operations



Motivation

WHY DO WE NEED LIBRARIES?

- **Layer of abstraction**

  Allow you to deal with the thing that only matters to you
- **Significant reduction in lines of code**

  Code has already been implemented by other 10x developers 😎.
  You just need to import them
- **Fast to implement**

  Increase in productivity, does not need to wait long for operation to be completed as the function implemented has been mostly optimised
- **Maintability**

  You don't need to worry about maintaining the functionality that we use

Scientific Computing Library

DOMAIN

1. Data Analysis

- NumPy
- Pandas

2. Visualization

- Matplotlib
- Seaborn

3. Machine Learning

- Scikit-learn
- TensorFlow
- PyTorch

Scientific Computing Library Domain

1. DATA ANALYSIS

When it comes to doing a Machine Learning project, whether we like it or not we will always be dealing with **data**, and it is integral to have a **high-quality data** feeding into our Machine Learning model. On top of that, **fully understanding your data** can provide you with some intuition on what type of ML model to use.

High-Quality Data?

- Data completeness
  (NaN / null value)
- Data consistency
  (Consistent data type for each feature,
  range of data that is not wildly different)

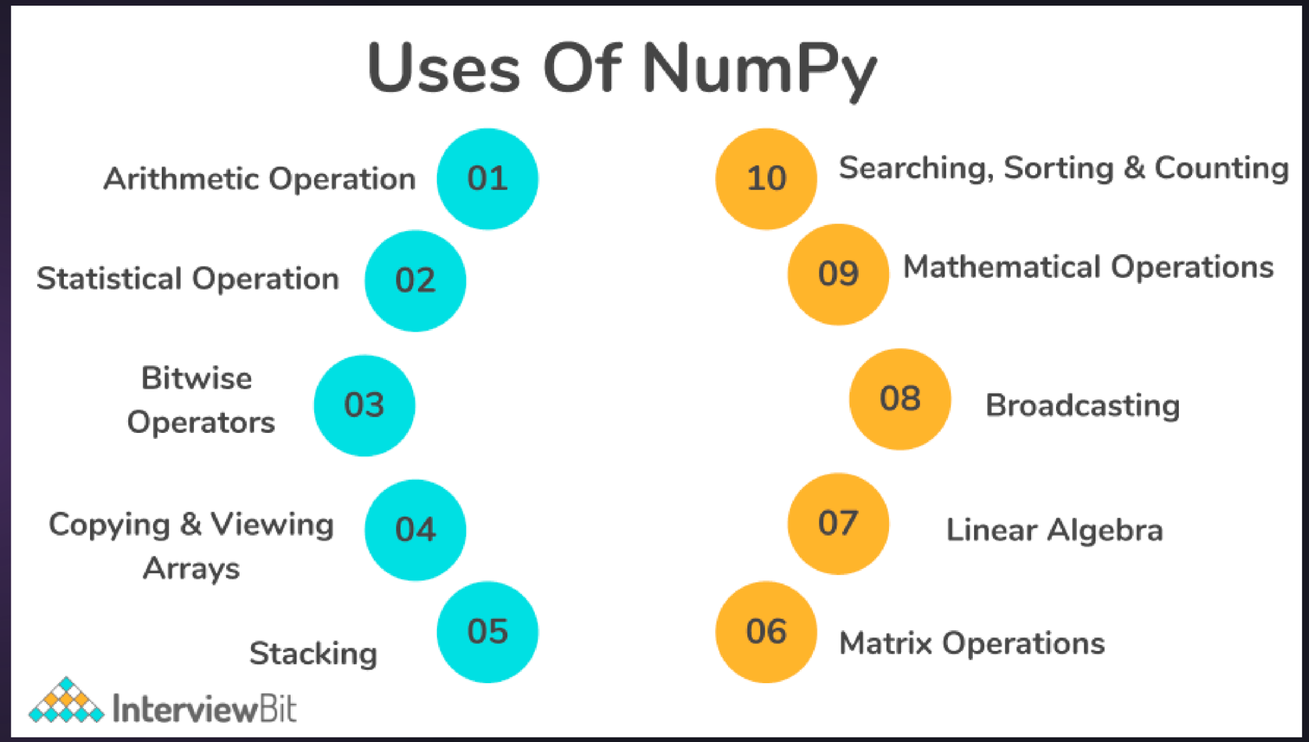
Data Understanding?

- Statistical Analysis
  (is there any underlying distribution,
  mode, median, mean)

Library

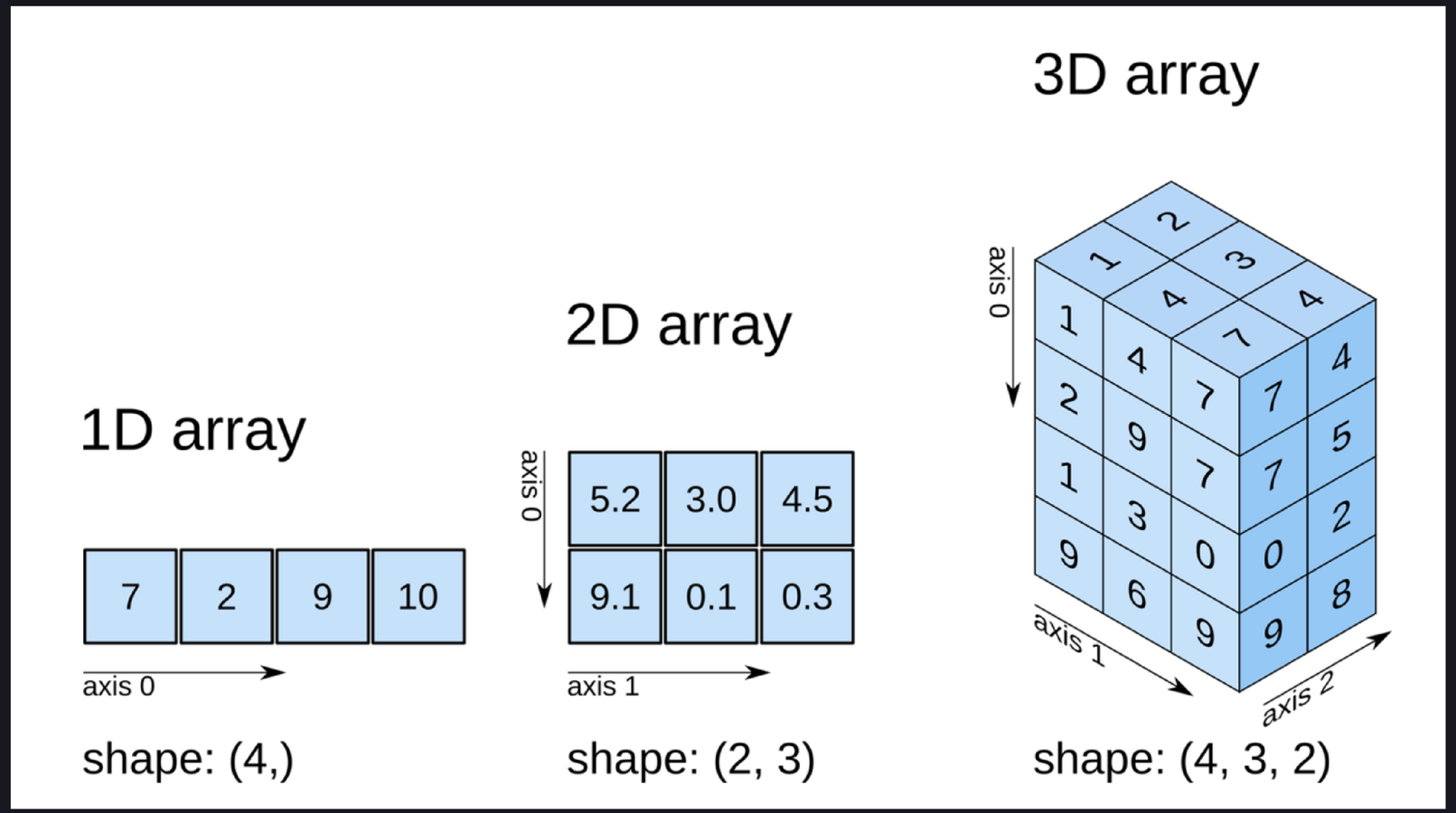
NUMPY

NumPy (Numerical Python) is the core library for scientific computing in Python. It deals with mathematical computation and enables users to compute on multi-dimensional data structures more efficiently and easily.



At its core, Numpy data structure is consist of **NUMPY ARRAY**, more precisely **ndarray** (n-dimensional array)

ndarray: It is basically an N-dimensional array which supports a wide variety of calculations



Why NumPy array is so fast?

CONTIGUOUS ARRAY

arr = np.arange(12).reshape(3,4)

0	1	2	3
4	5	6	7
8	9	10	11

In computer's memory, the values of arr are stored like this:

ROWS ARE STORED AS CONTIGUOUS (CONTINUOUS) BLOCKS OF MEMORY

Performance-wise, accessing memory addresses which are next to each other is very often faster than accessing addresses which are more "spread out"

Why NumPy array is so fast?

For standard Python list, they are heterogeneous,
meaning that they can accept any dataype.

ARR2 = ['ITEM1', 2, 3.5, TRUE]

Each of the element in arr2 has a different datatype, meaning that it requires different size / number of bits to store a particular element. And because of this, it will take a lot more time and not straightforward to access each particular element in the list

From this, you can see the tradeoff that exists between Python list, and NumPy array

INITIALIZING NUMPY ARRAY

np.array([[1,2,3],[4,5,6]]) : Create a matrix by specifying entry for each element. For this case, the matrix created will be of shape 2x3

[[1, 2, 3]
[4, 5, 6]]

np.ones((2,2)) : Create a 2x2 matrix full of ones

np.zeros((2,2)) : Create a 2x2 matrix full of zeros

np.eye(3) : Create a 3x3 identity matrix

np.random.random((2,2)) : Create a 2x2 matrix filled with random numbers between 0 and 1

np.arange(6) : Create an array of 6 element in which its elements are arranged in sorted order

np.empty((2,2)) : Create a 2x2 matrix placeholder

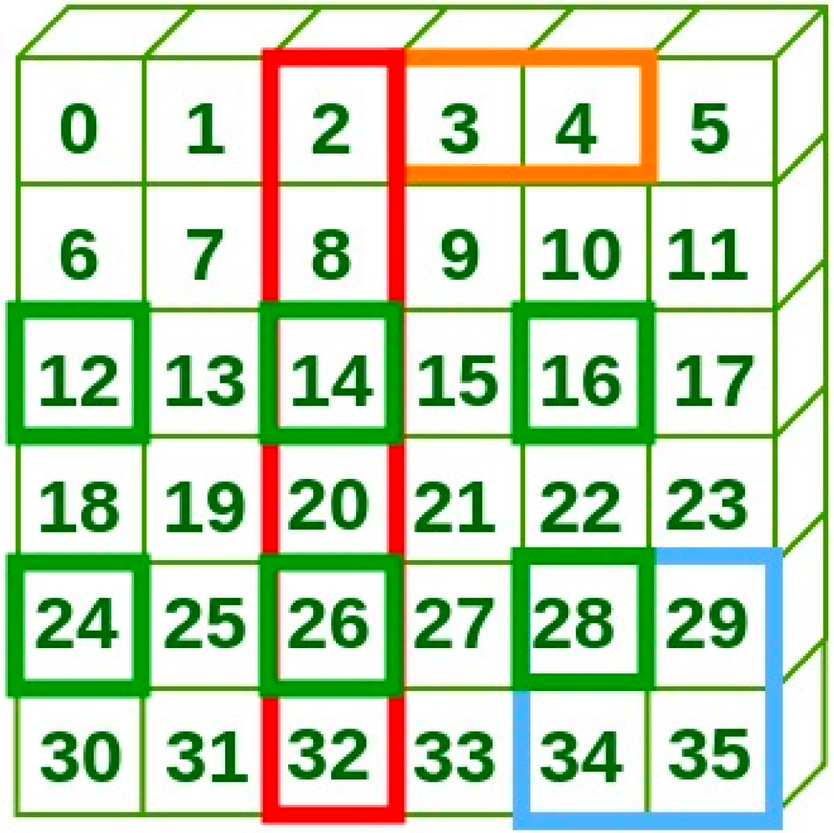
SLICING NUMPY ARRAY

```
>>> a[0,3:5]
array( [3,4] )

>>> a[4:, 4:]
array( [ 28, 29],
       [ 34, 35] ] )

>>> a[ :, 2]
array( [2, 8, 14, 20, 26, 32] )

>>> a[2 : : 2, : : 2]
array( [ 12, 14, 16],
       [ 24, 26, 28] ] )
```



OPERATION ON NUMPY ARRAY

np.transpose(a) : Find the transpose of array/matrix a

np.dot(a, b) : Calculate the Dot product of matrix a and b

np.linalg.inv(a) : Calculate the inverse matrix of matrix a
(only valid for square matrices, whose dimension is n * n)

np.diagonal(a) : Extract diagonal components of array a

np.reshape(a, (x,y)) : Reshape array a into the given dimension which is (x,y)

STATISTICS ON NUMPY ARRAY

np.sum(b) : Sum of all elements in an array/matrix b (if a is b matrix, you need to specify which axes it is acting upon, whether you want to add it column-wise or row-wise)

np.max(b) : Find the maximum element in array/matrix b

np.min(b) : Find the minimum element in array/matrix b

np.mean(b) : Mean of elements in an array/matrix b

np.median(b) : Median value among elements in array/matrix b

np.var(b) : Variance of the elements in the array/matrix b

np.std(b) : Standard deviation of the elements in the array/matrix b

Library

PANDAS

Pandas is another library that is commonly used. It is built on top of another package named Numpy. It is mostly known for its data wrangling ability and can be easily integrated with other data science modules in Python.

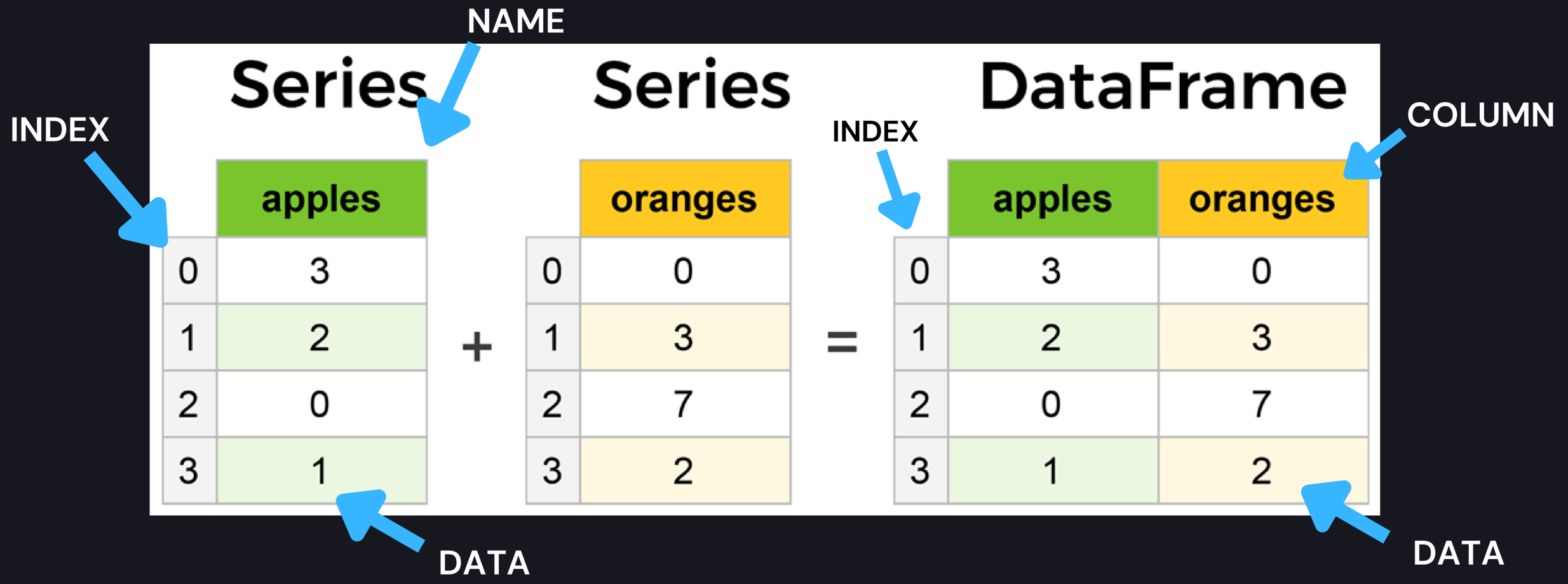
Pandas makes it simple to do many of the time consuming, repetitive tasks associated with working with data, including:

- **Data cleansing**
- **Data fill**
- **Data normalization**
- **Merges and joins**
- **Statistical analysis**
- **Data inspection**
- **Loading and saving data**
- **And much more...**

At its core, Pandas data structures are consist of **SERIES** and **DATAFRAME**

SERIES: A pandas Series is a one-dimensional array. It holds any data type supported in Python and uses labels to locate each data value for retrieval. It is very similar to Python list

DATAFRAME: DataFrame is a 2-dimensional labeled data structure with columns of potentially different types.



INITIALISING SERIES

pd.Series() : Create a Pandas Series by passing the data as its argument

INITIALISING DATAFRAME

pd.DataFrame() : Create a Pandas DataFrame by passing data as its argument

IMPORTING DATAFRAME

pd.read_csv() : Import a file that is in .csv format into DataFrame
pd.read_hdf() : Import a file that is in .hdf format into DataFrame
pd.read_excel() : Import a file that is in .xls format into DataFrame

IMPORTING DATAFRAME

pd.DataFrame.to_csv() : Export DataFrame into a .csv file
pd.DataFrame.to_hdf() : Export DataFrame into a .hdf file
pd.DataFrame.to_excel() : Export DataFrame into a .xls file

DATA MANIPULATION

df.sort_index() : Sort the index of a DataFrame

df.sort_values() : Sort the DataFrame by the value in particular column

df.drop(columns=['A', 'B']) : Drop columns from DataFrame

df.head(n) : Select first n rows; Default: n = 5

df.tail(n) : Select last n rows; Default: n = 5

Scientific Computing Library Domain

2. VISUALISATION

Another important aspect in Machine Learning is visualisation, which can be categorised into two functionalities;

Data

This can be plotting any sorts of data distribution, or if we are doing Computer Vision problem, data visualisation also involves plotting the data itself

Performance

Another area where visualisation comes in handy is for plotting the performance of our model, which includes:

- Comparing how training loss progresses
- Plotting a confusion matrix to understand on which feature our model is performing well and vice-versa

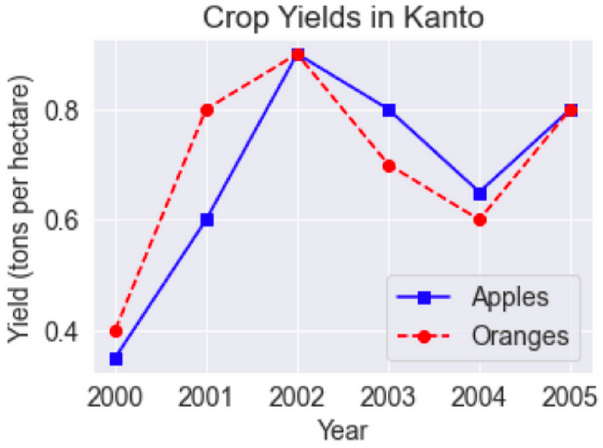
Library

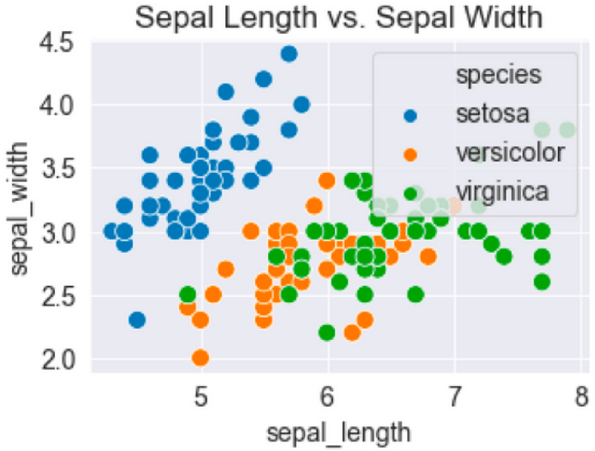
MATPLOTLIB

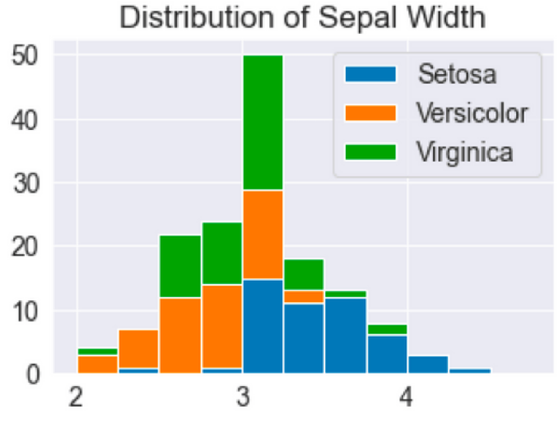
Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It provides both a very quick way to visualise data from Python and publication-quality figures in many formats. Among other features of matplotlib are:

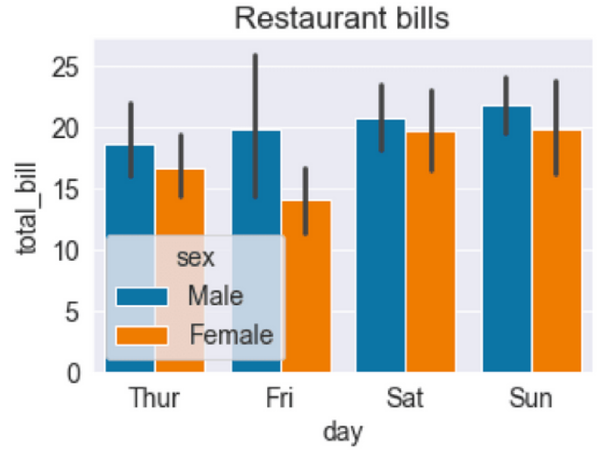
- Create publication quality plots.
- Make interactive figures that can zoom, pan, update.
- Customise visual style and layout.
- Export to many file formats.
- Embed in JupyterLab and Graphical User Interfaces.
- Use a rich array of third-party packages built on Matplotlib.

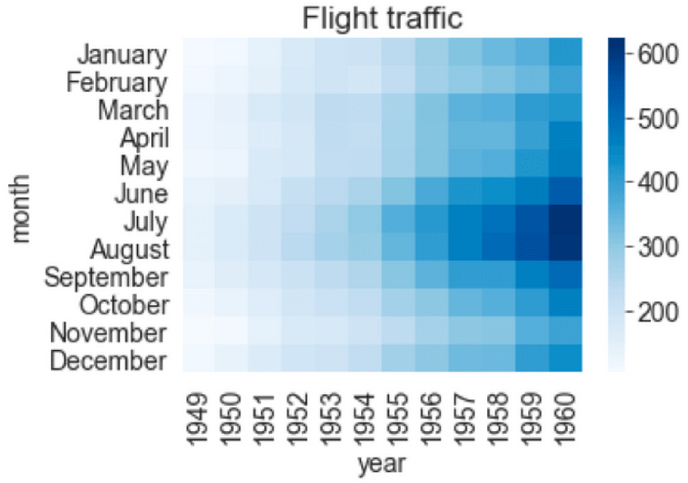
SOME TYPE OF GRAPHS YOU CAN DRAW WITH MATPLOTLIB

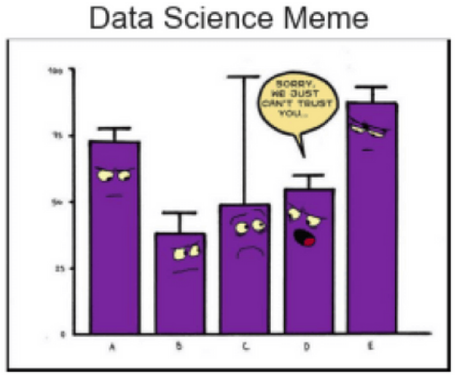












Library

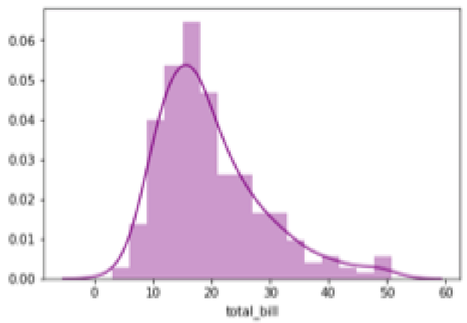
SEABORN

Seaborn is an amazing data visualisation library for statistical graphics plotting in Python. It is built on the top of the matplotlib library and also closely integrated to the data structures from panda.

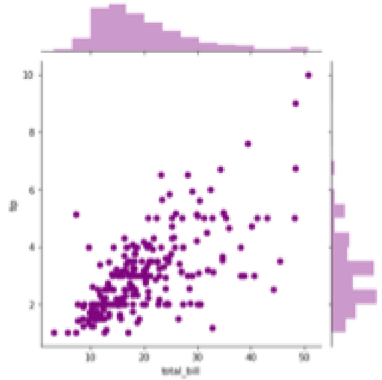
On Seaborn’s official website, they state:

"If matplotlib 'tries to make easy things easy and hard things possible', seaborn tries to make a well-defined set of hard things easy too."

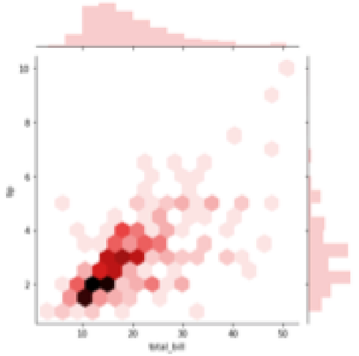
SOME TYPE OF GRAPHS YOU CAN DRAW WITH SEABORN



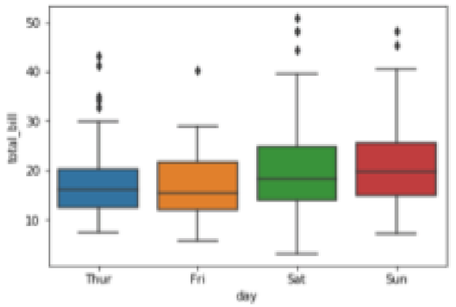
distplot



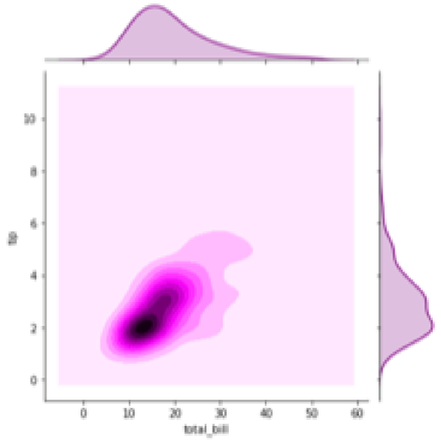
Jointplot



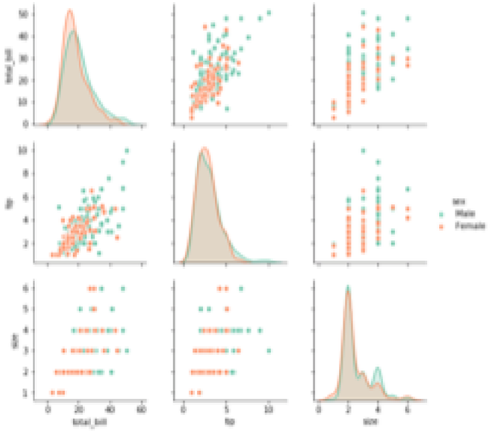
Hexplots



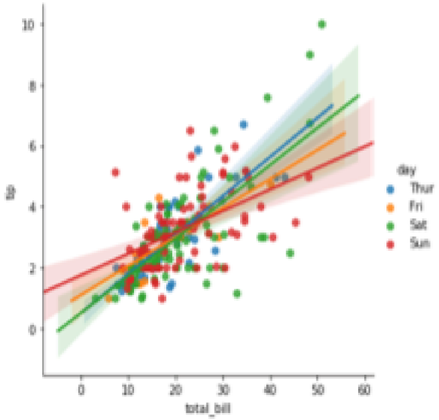
Boxplots



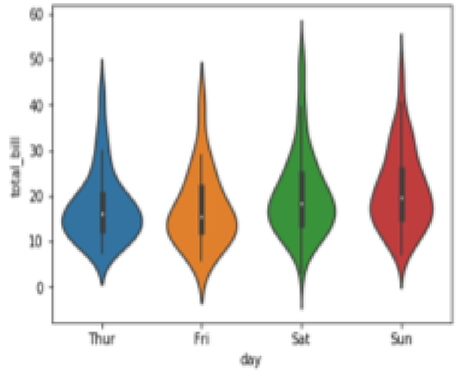
KDE Plot



Pair Plots



LM Plots



Violin Plots

Library

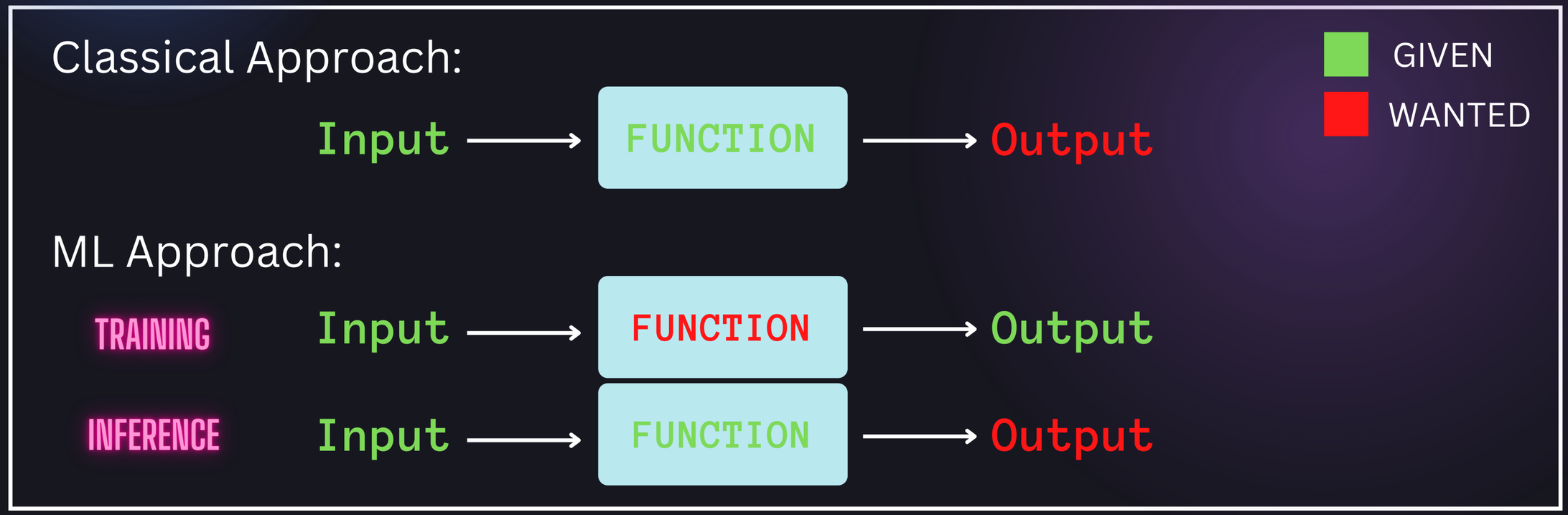
MATPLOTLIB VS. SEABORN

Characteristics	Matplotlib	Seaborn
Use Cases	Matplotlib plots various graphs using Pandas and Numpy	Seaborn is the extended version of Matplotlib which uses Matplotlib along with Numpy and Pandas for plotting graphs
Complexity of Syntax	It uses comparatively complex and lengthy syntax.	It uses comparatively simple syntax which is easier to learn and understand.
Multiple figures	Matplotlib has multiple figures can be opened	Seaborn automates the creation of multiple figures which sometimes leads to out of memory issues
Flexibility	Matplotlib is highly customizable and powerful.	Seaborn avoids a ton of boilerplate by providing default themes which are commonly used.

Scientific Computing Library Domain

3. MACHINE LEARNING

"Field of study that gives computers the ability to learn without being explicitly programmed" – Arthur Samuel



Machine Learning - Introduction

7 STEPS IN MACHINE LEARNING

ML Library

ML Library covers a very wide range of processes. Some of the library in fact does provide some functionality for Step 2 (Data Preparation)

Example

ML PROJECT

Penguin Classification

We will be using 'Palmer Dataset'

STEP 1 - DATA COLLECTION

Data were collected and made available by Dr. Kristen Gorman and the Palmer Station, Antarctica LTER, a member of the Long Term Ecological Research Network.

STEP 2 - DATA UNDERSTANDING + PREPARATION

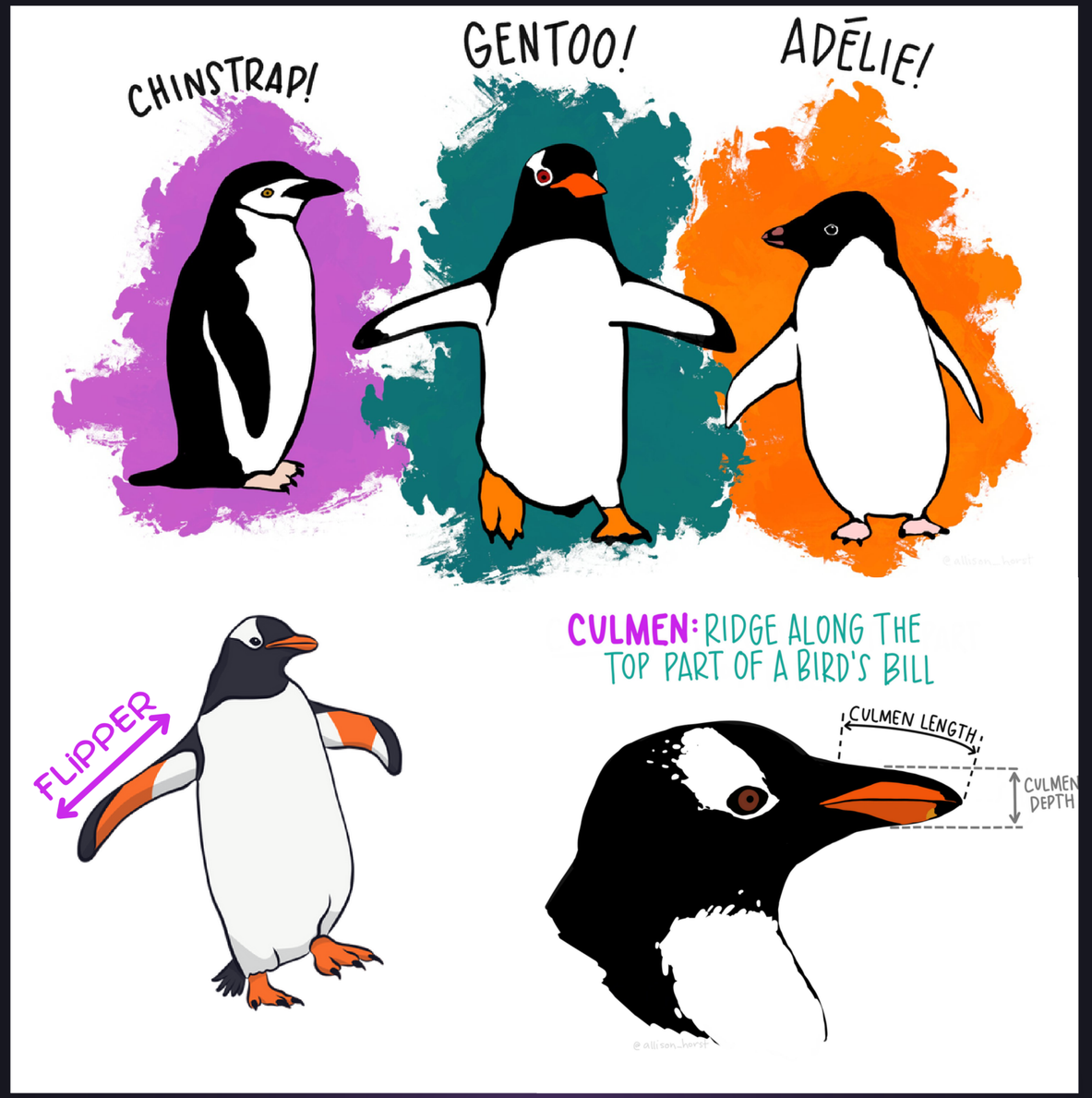
GOAL: Predict what species of penguin it is given the features of the penguin

Species: Adelie, Gentoo, Chinstrap

FEATURES:

culmen_length_mm: culmen length (mm)
culmen_depth_mm: culmen depth (mm)
flipper_length_mm: flipper length (mm)
body_mass_g: body mass (g)
island: island name (Dream, Torgersen, or Biscoe) in the Palmer Archipelago
sex: penguin sex

REFER AND ATTEMPT THE PROVIDED NOTEBOOK FOR FURTHER DATA UNDERSTANDING AND DATA PREPARATION TECHNIQUE. AND TEST YOURSELF WITH THE TOOLS YOU'VE LEARNT FROM THE PREVIOUS SLIDES



AS YOU GO THROUGH THE NOTEBOOK, BE MINDFUL OF THE 7 STEPS IN MACHINE LEARNING

Library

SCIKIT-LEARN

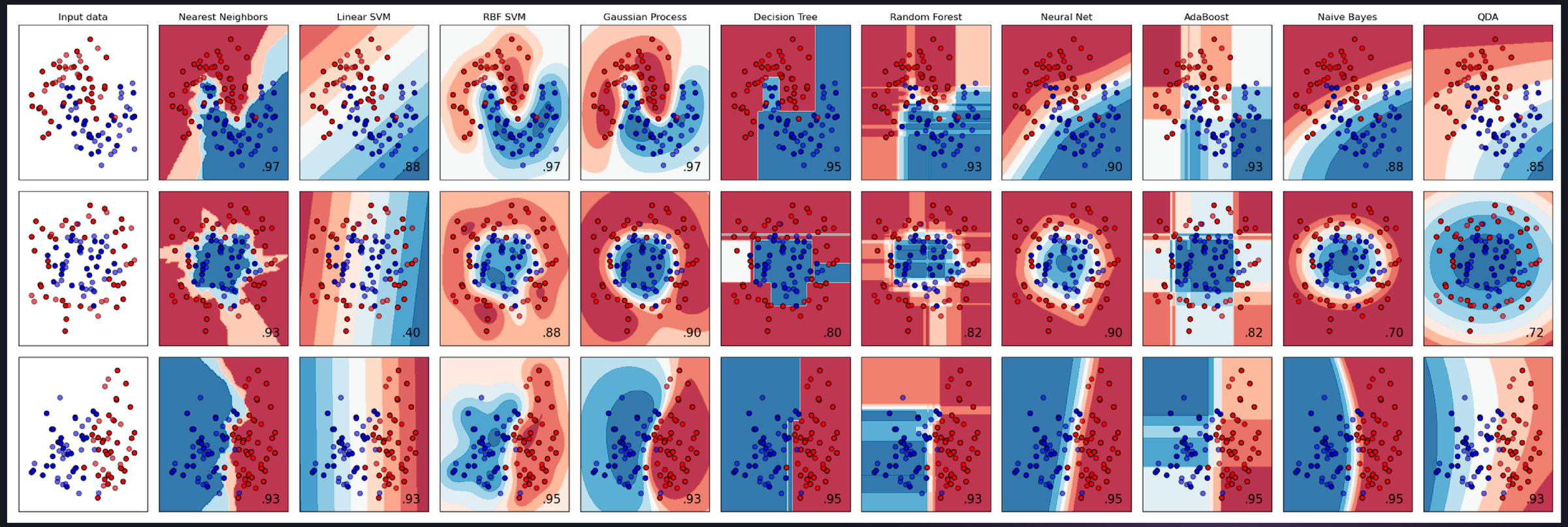
Scikit-learn (SciPy Toolkit) is a library in Python that provides many unsupervised and supervised learning algorithms. It’s built upon some of the technology you might already be familiar with, like NumPy, pandas, and Matplotlib!

- There is a vast collection of ML algorithms that are all included (It’s like a repo, but a repo of ML algorithm)
- An amazing way to get a handle of what different types of models do, as well as giving some intuition about some algorithm perform
- It also contains a lot of useful function that we can utiliza during data preprocessing step

MACHINE LEARNING MODEL

A lot of classical ML model :

- Classification (Nearest Neighbors, SVM, Decision Tree, Random Forest, ...)
- Regression (Linear Regression, Logistic Regression, SVM, ...)
- Clustering (k-Means, Spectral Clustering, Mean Shift, ...)



FUNCTION THAT YOU WILL SEE IN NOTEBOOK

StandardScaler.fit_transform(): Standardize features by removing the mean and scaling to unit variance.

model_selection.train_test_split(X): Split an array/matrix 'X' into random train and test subsets

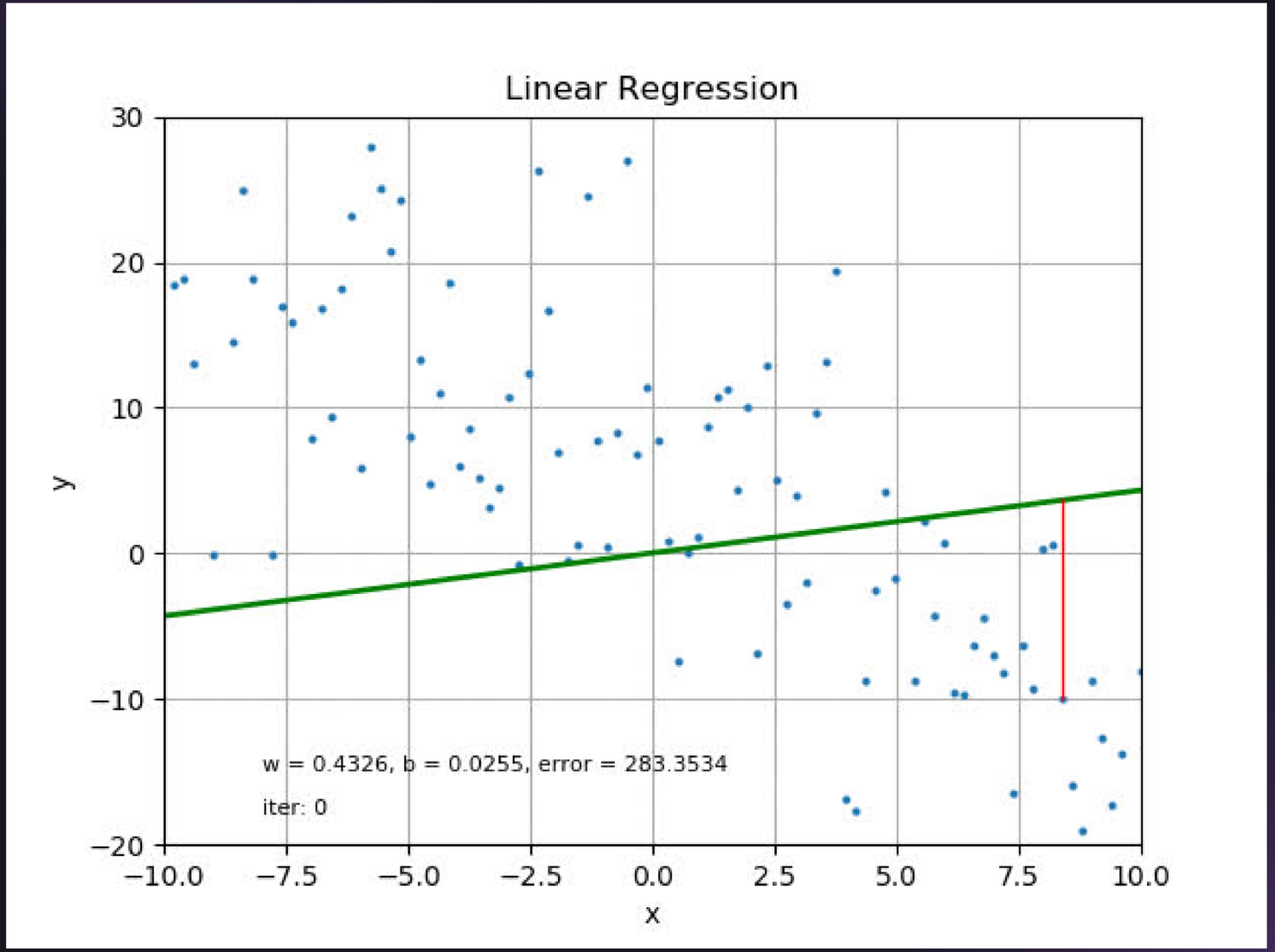
model.fit(X, y): Fit the any chosen ML model according to the given training data.
In our notebook, model comes from svm.SVC class

model.predict(X): Perform classification on samples in 'X'.

metrics.accuracy_score(y_predicted, y): Compute a simple score between 'y_predicted' and 'y'

metrics.confusion_matrix(): Compute confusion matrix to evaluate the accuracy of a classification.

WHAT HAPPEN DURING TRAINING?



Animation from: https://medium.com/swlh/from-animation-to-intuition-linear-regression-and-logistic-regression-f641a31e1caf

STANDARDISATION

The operation that we do when stardizing our feature is removing the mean and scaling it to unit variance.

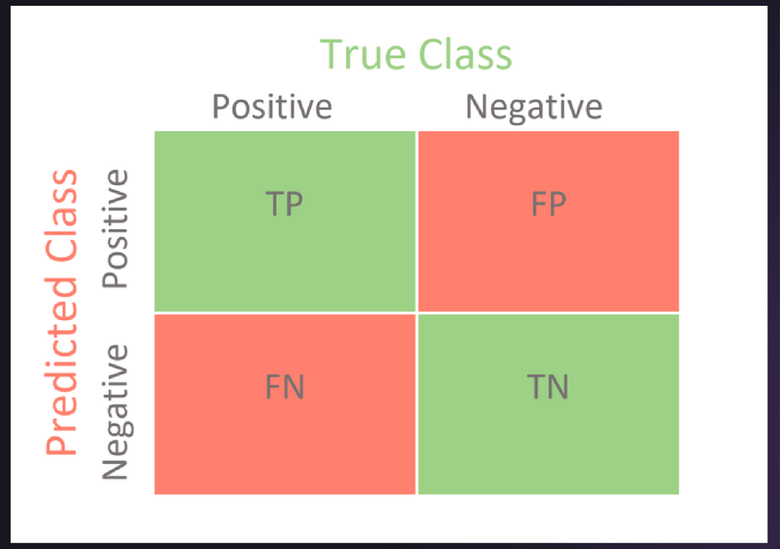
$$Z = \frac{x - \mu}{\sigma}$$

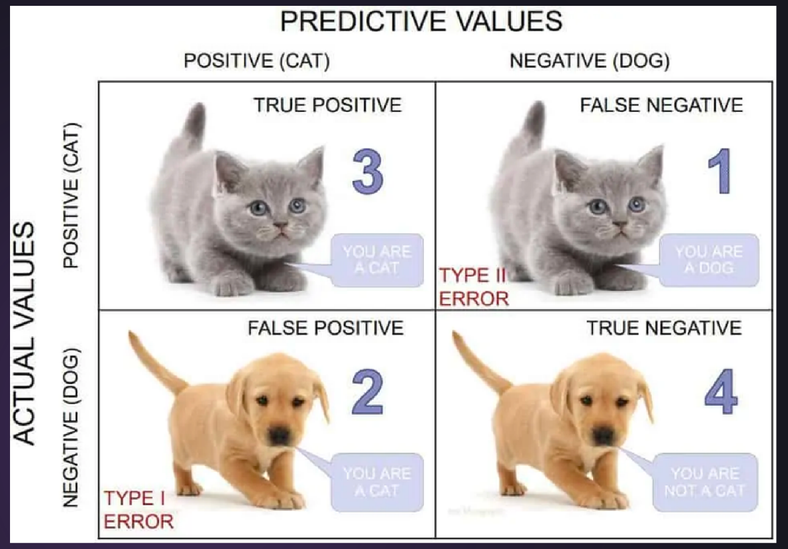
WHY?: Standardisation makes all variables contribute equally. A variable that ranges between 0 and 1000 will outweigh a variable that ranges between 0 and 1. Using these variables without standardization will give the variable with the larger range weight of 1000 in the analysis.

CONFUSION MATRIX

A table with a combination of predicted class and true class.

- Useful for identifying Type I Error and Type II Error
- Also useful for measuring Precision, Accuracy, Recall, and Specifity

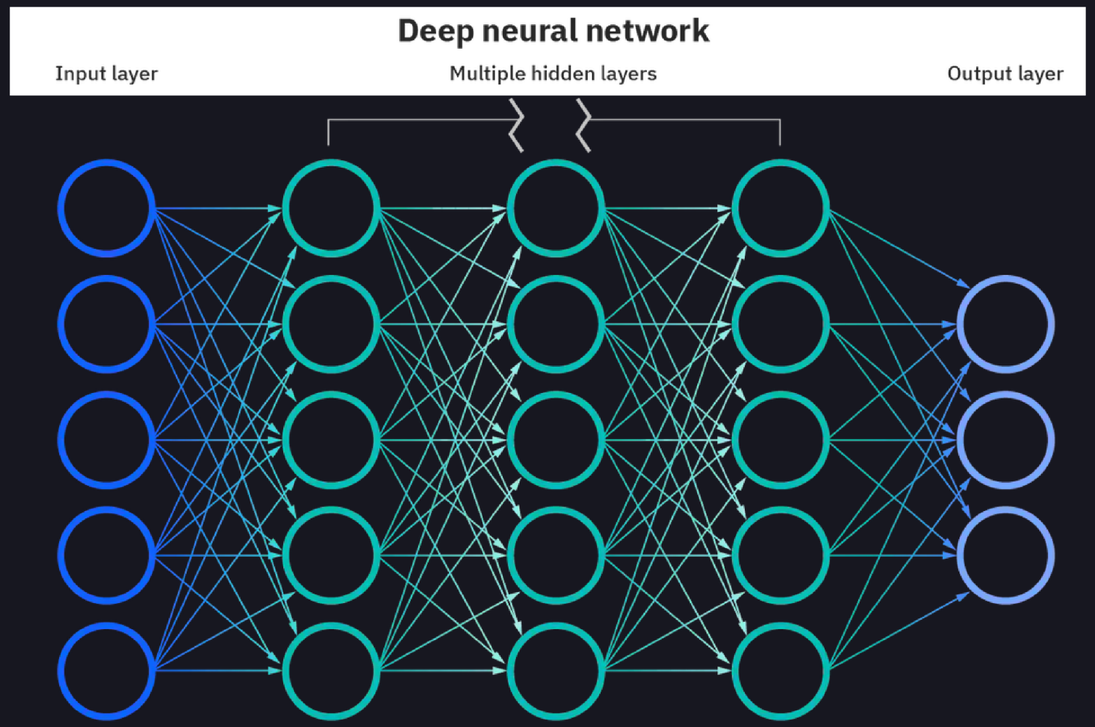




Library

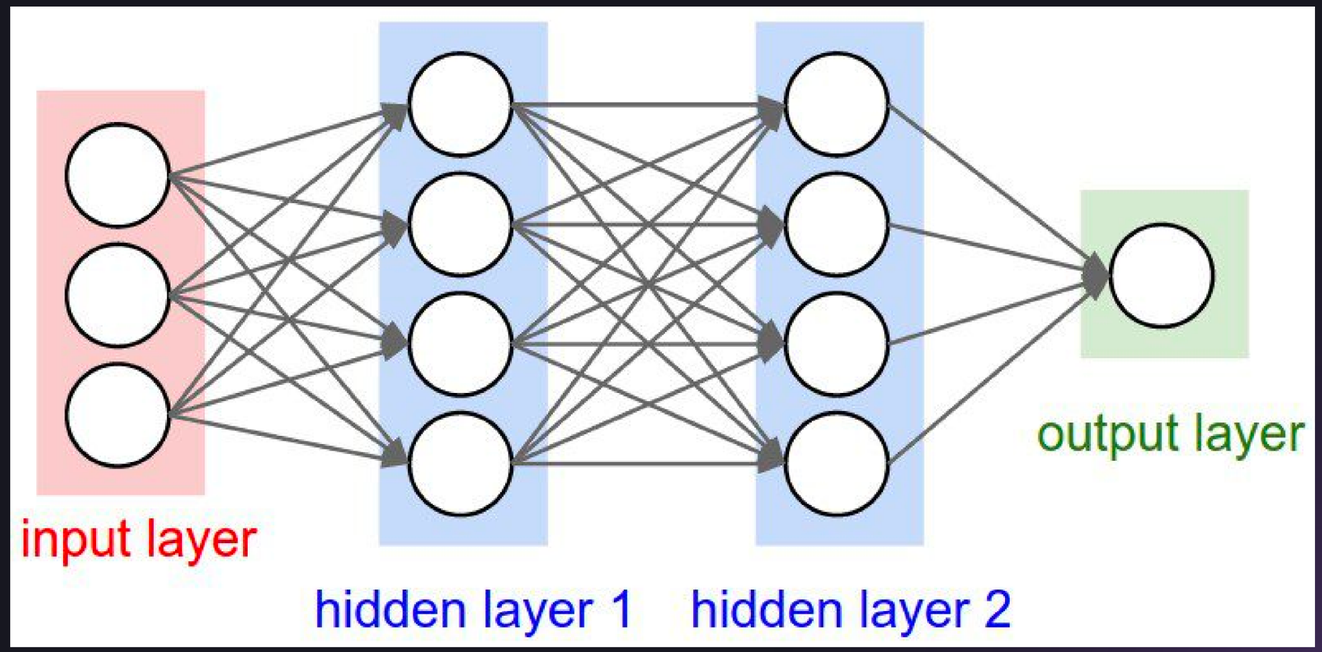
TENSORFLOW

TensorFlow is a library that makes machine learning and particularly developing neural networks faster and easier. It is mainly used for developing deep learning model. It makes building models easier, faster and more reproducible



HOW TO CREATE NN USING TENSORFLOW

```
model = Sequential()          # create an object called model of class Sequential
model.add(Dense(units=4, activation='relu', input_dim=3)))
model.add(Dense(units=4, activation='relu'))
model.add(Dense(units=1, activation='relu'))
```



IF YOU'RE INTERESTED IN UNDERSTANDING WHAT IS INSIDE NEURAL NETWORK, AND WHAT HAPPENED DURING TRAINING, GO TO YOUTUBE AND SEARCH **3BLUE1BROWN NEURAL NETOWRK**

FUNCTION THAT YOU WILL SEE IN NOTEBOOK

model.compile(): Configures the model for training (metrics, loss, accuracy)

model.fit(X, y, epochs): Trains the model on feature 'X' and target 'y' for a fixed number of epochs (iterations on a dataset)

model.predict(X): Perform classification on samples in 'X'.

Library

PYTORCH

- An open source machine learning framework that accelerates the path from research prototyping to production deployment. **– PyTorch website**

- PyTorch is an optimized Deep Learning tensor library based on Python and Torch. PyTorch is favored over other Deep Learning frameworks like TensorFlow and Keras since it uses dynamic computation graphs and is completely Pythonic. **– Simplilearn**

- PyTorch is a machine learning framework based on the Torch library, used for applications such as computer vision and natural language processing, originally developed by Meta AI and now part of the Linux Foundation umbrella. **– Wikipedia**

QUESTIONS?

CONGRATULATIONS