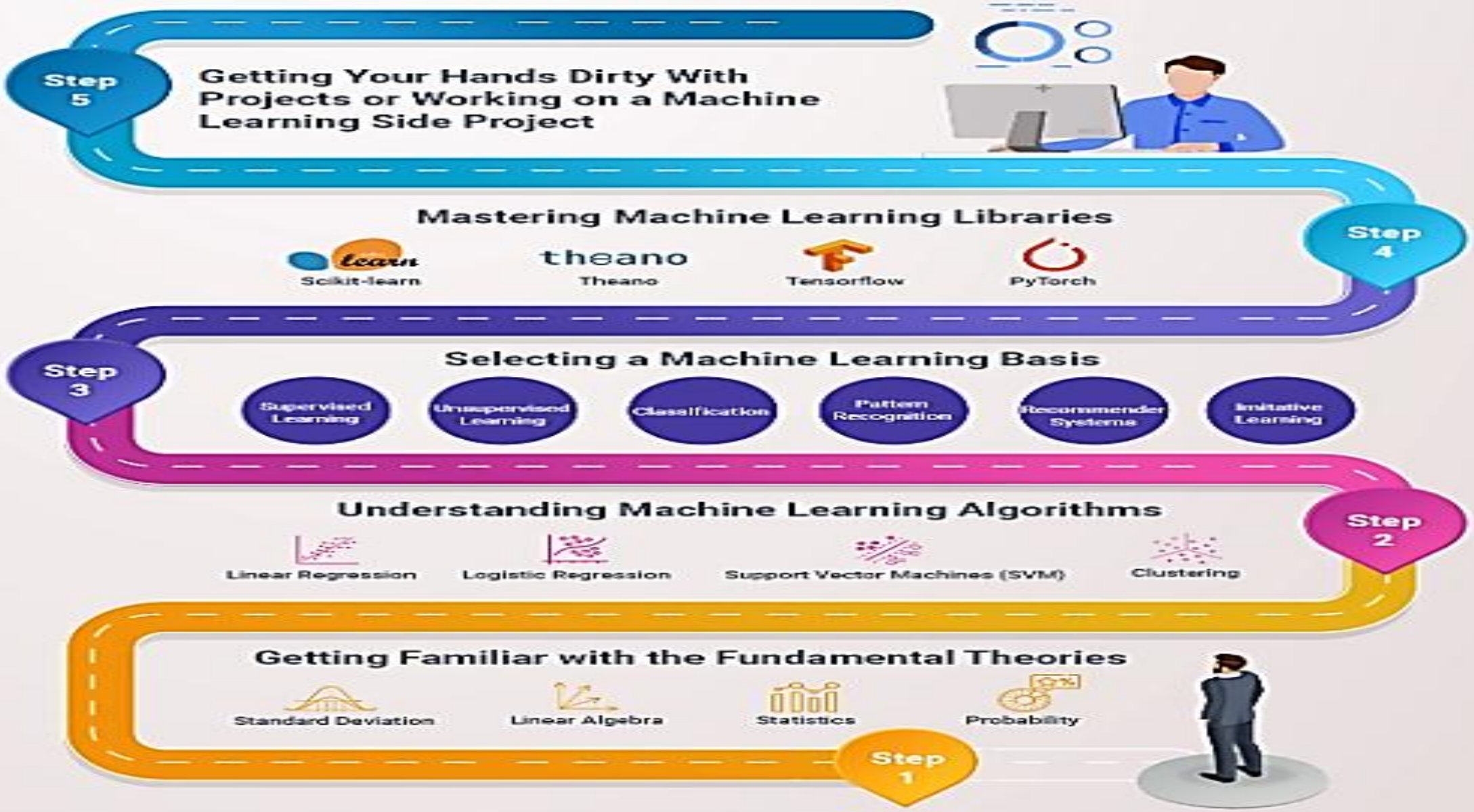
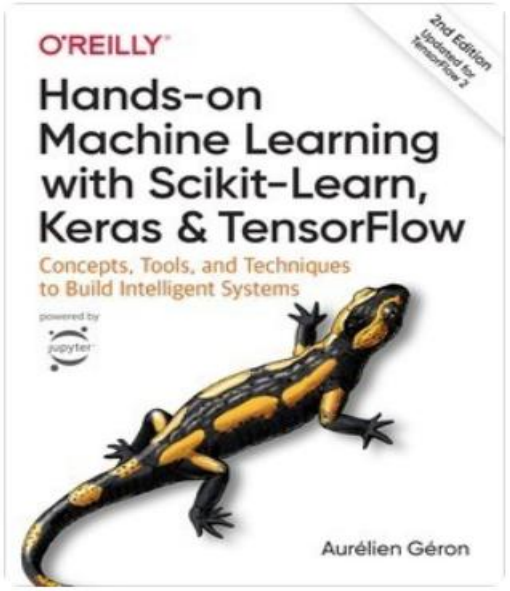
ACC-HPC June 2025

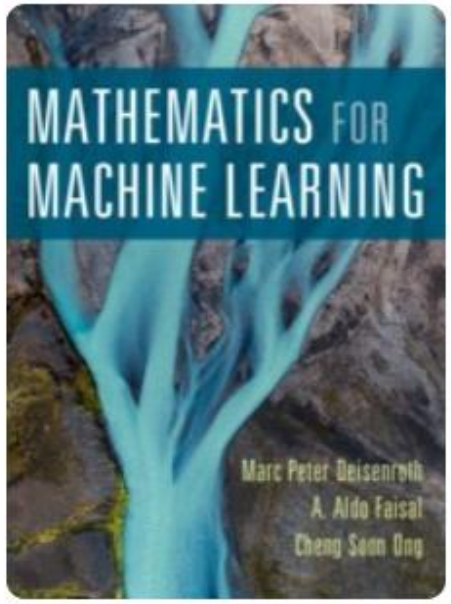
INTRODUCTION TO MACHINE LEARNING

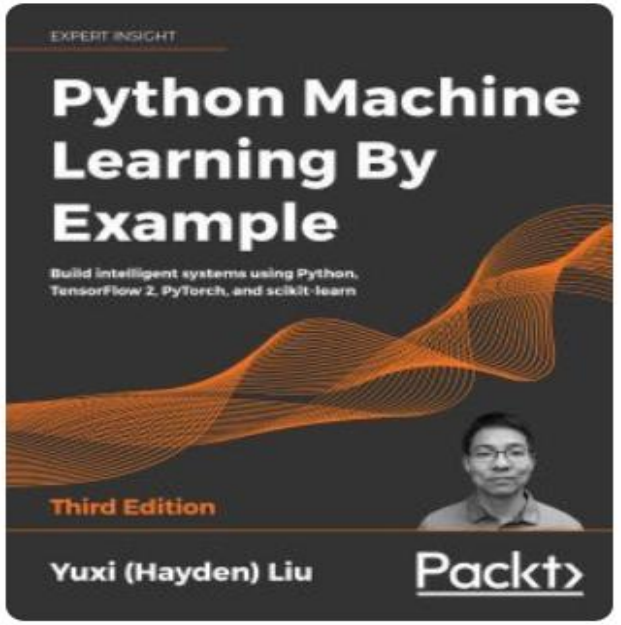
Dr Kiran Waghmare
CDAC Mumbai

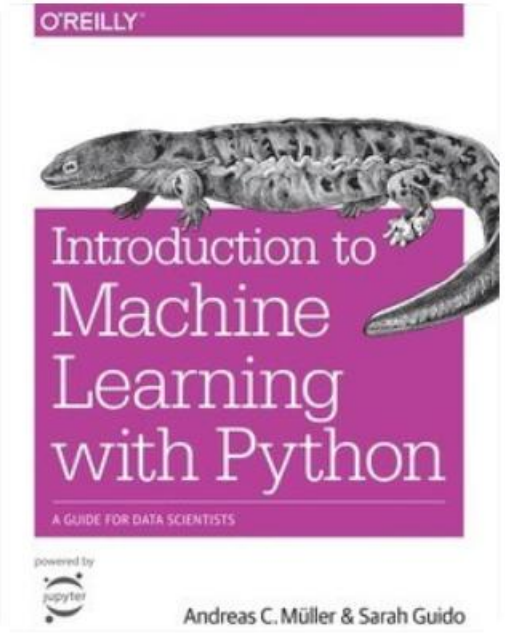
Machine Learning Roadmap

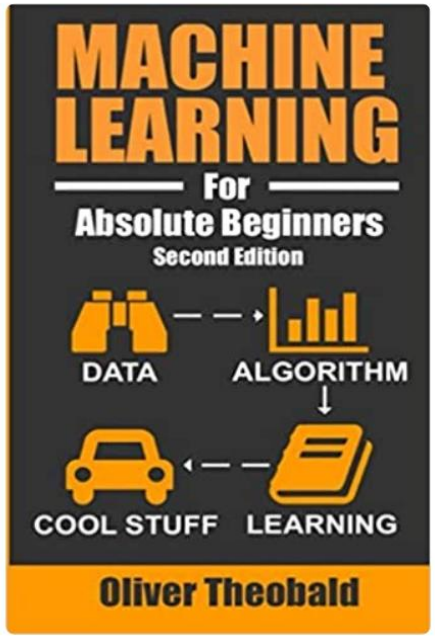


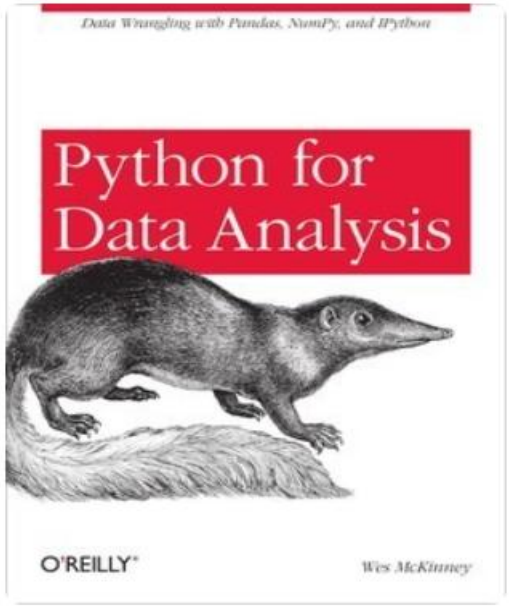

O'REILLY
2nd Edition
Updated for TensorFlow 2
Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow
Concepts, Tools, and Techniques to Build Intelligent Systems
powered by jupyter
Aurélien Géron

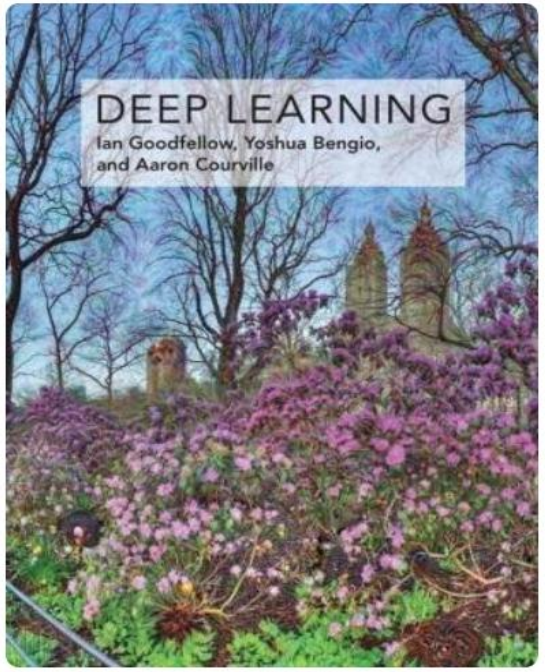

MATHEMATICS FOR MACHINE LEARNING
Marc Peter Deisenroth
A. Aldo Faisal
Cheng Soon Ong

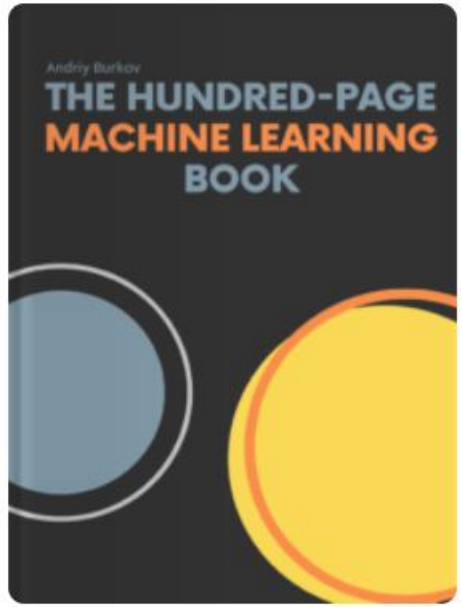

EXPERT INSIGHT
Python Machine Learning By Example
Build intelligent systems using Python, TensorFlow 2, PyTorch, and scikit-learn
Third Edition
Yuxi (Hayden) Liu
Packt


O'REILLY
Introduction to Machine Learning with Python
A GUIDE FOR DATA SCIENTISTS
powered by jupyter
Andreas C. Müller & Sarah Guido


MACHINE LEARNING
For
Absolute Beginners
Second Edition
DATA
ALGORITHM
COOL STUFF
LEARNING
Oliver Theobald


Data Wrangling with Pandas, NumPy, and IPython
Python for Data Analysis
O'REILLY
Wes McKinney

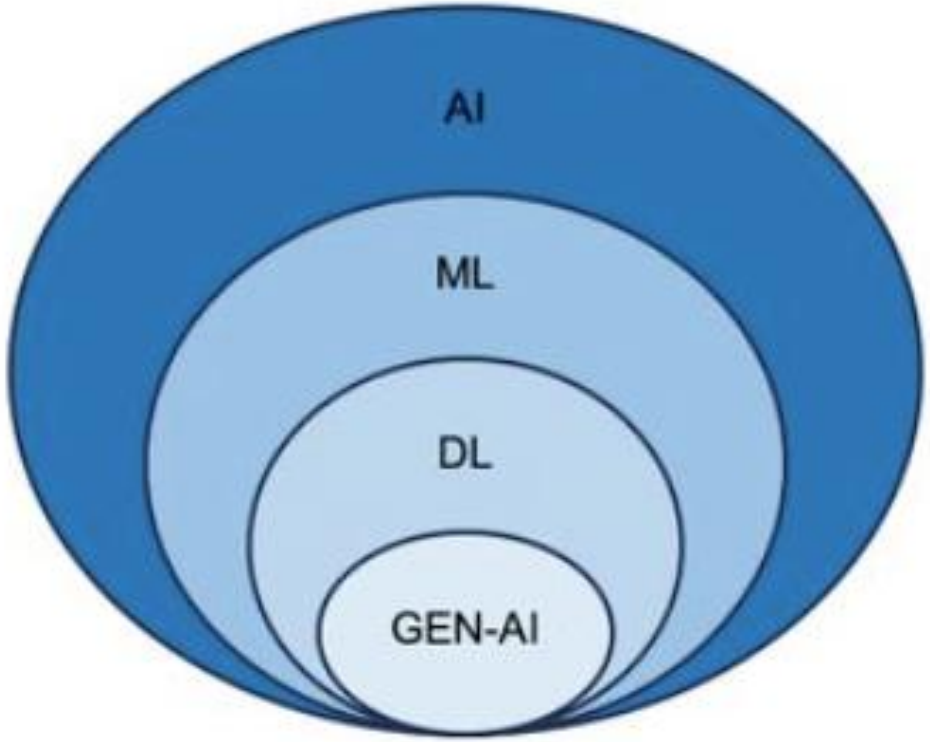

DEEP LEARNING
Ian Goodfellow, Yoshua Bengio, and Aaron Courville


Andriy Burkov
THE HUNDRED-PAGE MACHINE LEARNING BOOK

Agenda

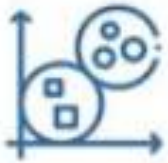
- What is Machine Learning?
- Applications
- Types of ML
- ML Workflow
- Data Pre-processing
- Feature Engineering
- Evaluating ML Models

Overview on AI, ML, DL & GEN-AI



- AI
  - A broad field
  - Creating machines to mimic human intelligence

- Machine learning (ML)
  - Subset of AI
  - Learn patterns from data
  - Make decisions/predictions

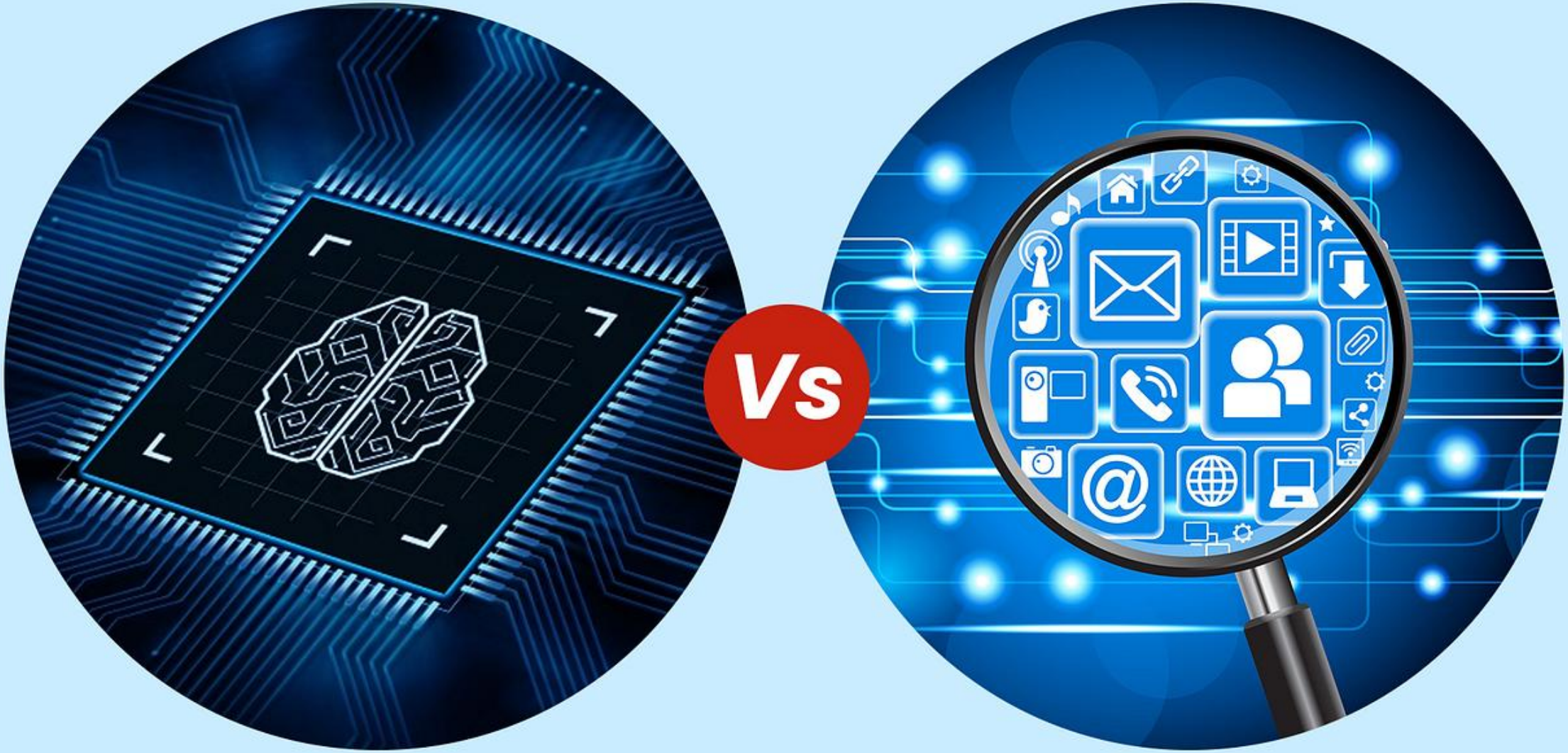


- Deep learning (DL)
  - Subset of ML
  - Using neural networks
  - Learn complex patterns

- Generative AI (GEN-AI)
  - Subset of DL
  - Generating new content

Comparison

- **Traditional Programming**



Data Information

Machine Learning

- **Herbert Alexander Simon:**
  "Learning is any process by which a system improves performance from experience."
- "Machine Learning is concerned with computer programs that automatically improve their performance through experience. "

Herbert Simon
Turing Award 1975
Nobel Prize in Economics 1978

What is Machine Learning?

- [Arthur Samuel, 1959]
  - Field of study that gives computers
  - the ability to learn without being explicitly programmed

- [Kevin Murphy] algorithms that
  - automatically detect patterns in data
  - use the uncovered patterns to predict future data or other outcomes of interest

- [Tom Mitchell] algorithms that
  - improve their performance (P)
  - at some task (T)
  - with experience (E)

The concept of learning in a ML system

- Learning = Improving with experience at some task
  - Improve over task *T*,
  - With respect to performance measure, *P*
  - Based on experience, *E*.

Definition

A computer program is said to
learn from **experience E** with respect to
some class of **tasks T** and
performance **measure P**,
if its performance at tasks T, as measured by P, **improves** with experience E.

Examples

i) Handwriting recognition learning problem

- Task T: Recognising and classifying handwritten words within images
- Performance P: Percent of words correctly classified
- Training experience E: A dataset of handwritten words with given classifications

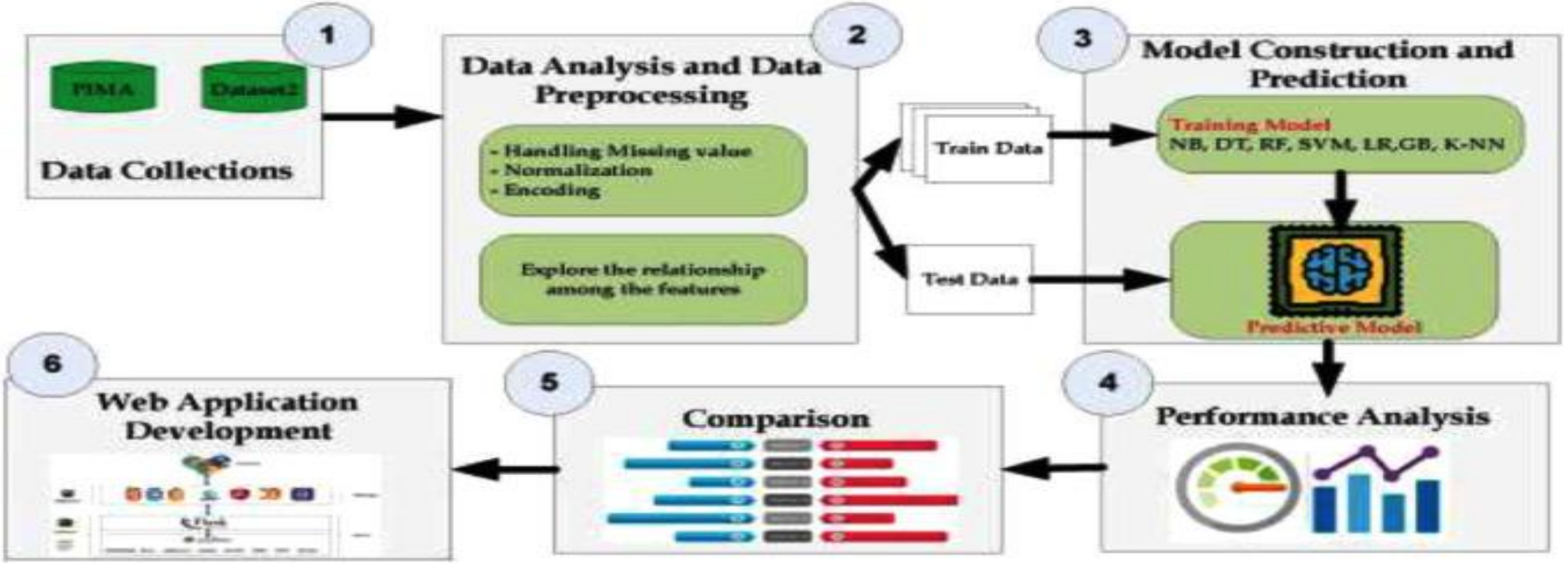
ii) A robot driving learning problem

- Task T: Driving on highways using vision sensors
- Performance measure P: Average distance traveled before an error
- training experience: A sequence of images and steering commands recorded while observing a human driver

iii) A chess learning problem

- Task T: Playing chess
- Performance measure P: Percent of games won against opponents
- Training experience E: Playing practice games against itself

Industry Case Study : 1



Definition

A computer program which learns from experience is called a *machine learning program* or simply a *learning program*. Such a program is sometimes also referred to as a *learner*.

What is Machine Learning?

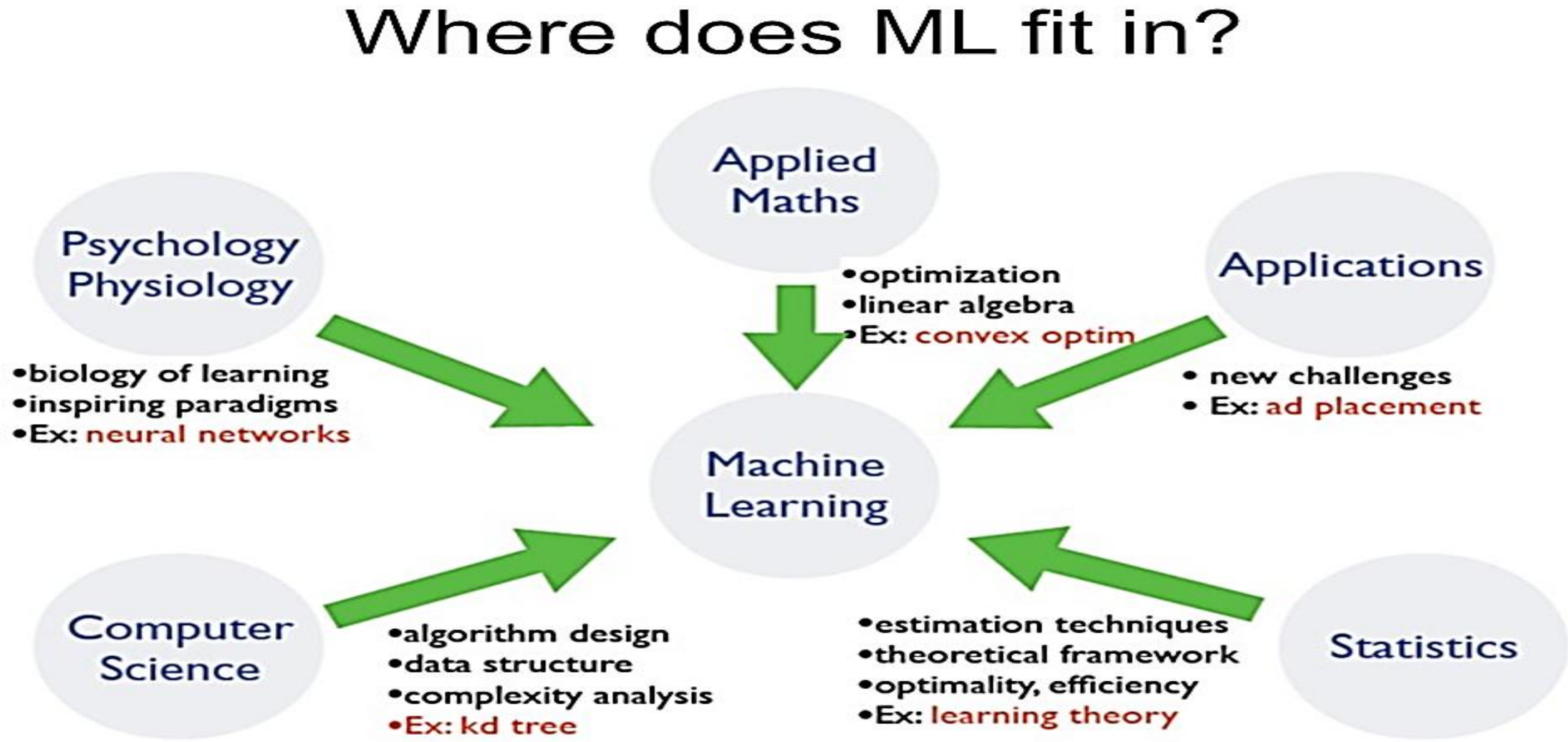
- If you are a Scientist

Why Study Machine Learning?

The Time is Ripe

- Algorithms
  - Many basic effective and efficient algorithms available.
- Data
  - Large amounts of on-line data available.
- Computing
  - Large amounts of computational resources available.

Domains include in ML



(C) Dhruv Batra Slide Credit: Fei Sha 19


Google
Google Search
AUSTRALIA NEWS
australia news - Google Search
australia news today
australia news now
australia news latest
australia news covid
australia news code


NETFLIX
Category Codes 2022
NETFLIX
CROWN
how i met your mother
Breaking Bad
SPECTRAL

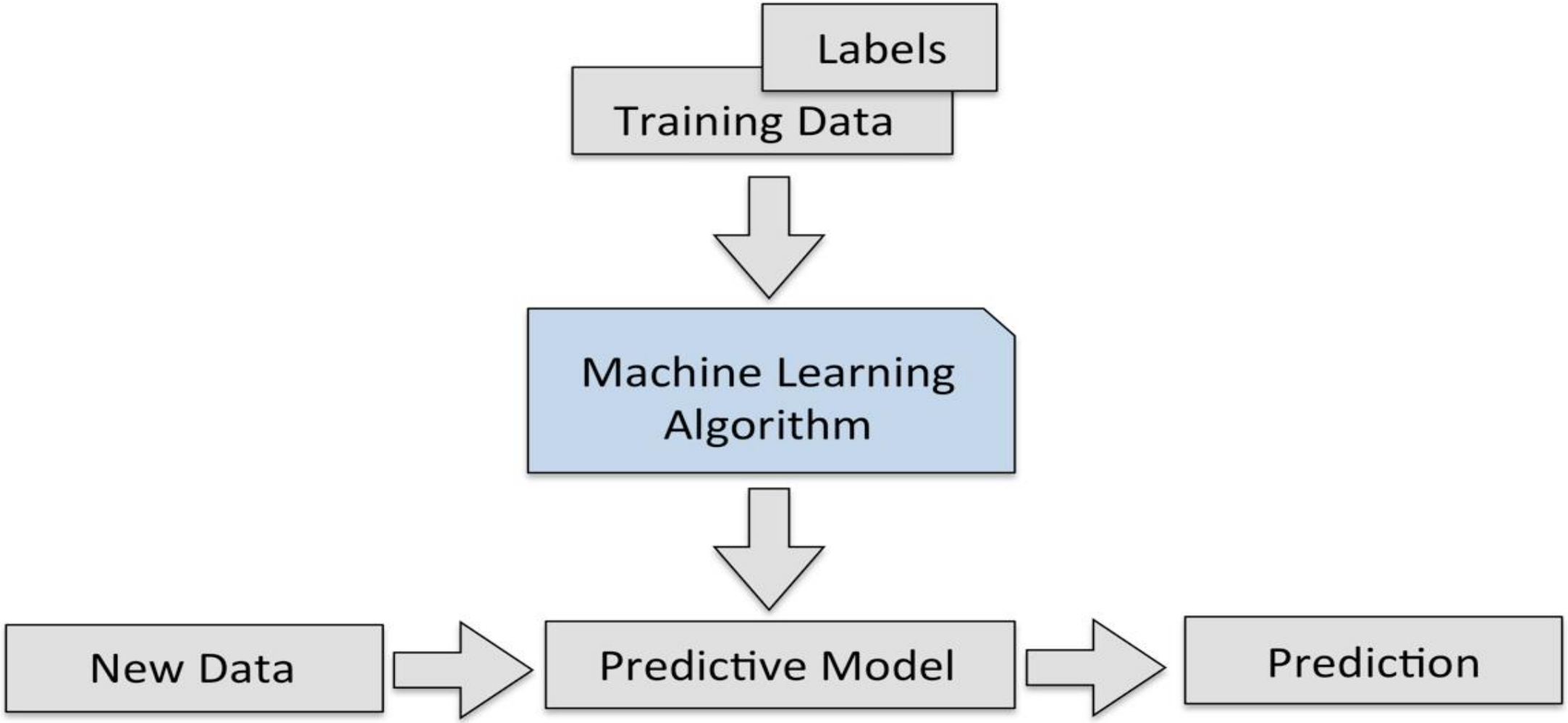
How does Machine Learning Work?

Input Data → Analyze Data → Find Patterns → Prediction → Stores the Feedback





Machine Learning Model



Features of Machine Learning:

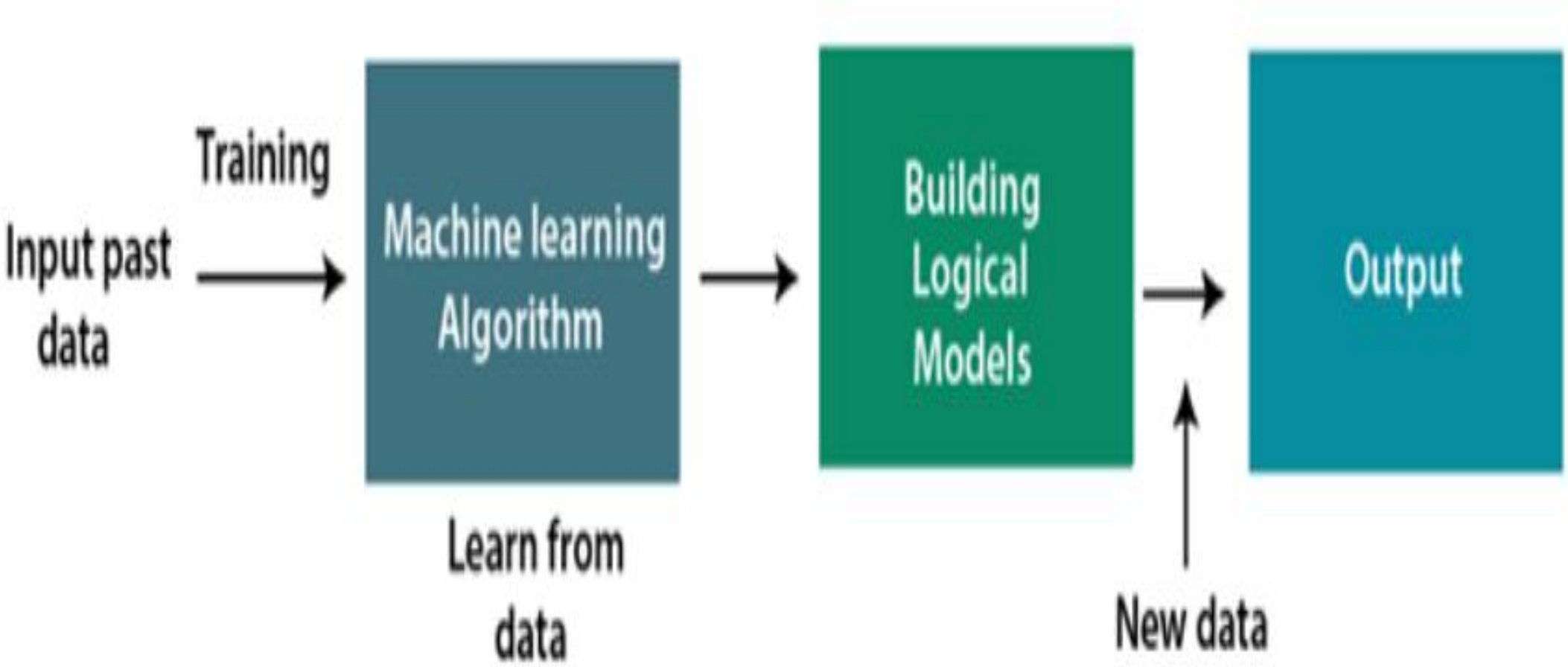
- Machine learning **uses data to detect various patterns** in a given dataset.
- It can **learn from past data** and improve automatically.
- It is a **data-driven technology**.
- Machine learning is much **similar to data mining** as it also deals with a huge amount of data.

- **Following are some key points that show the importance of Machine Learning:**
  - **Rapid increment** in the production of data
  - **Solving complex problems**, which are difficult for a human
  - **Decision-making in various sectors** including finance
  - Finding **hidden patterns and extracting useful information** from data.

What is Machine Learning Model?

- **Definition:**
  - Machine Learning is a concept which allows the machine
  - **to learn from examples and experience**,
  - and that **too without being explicitly programmed**.

- **Machine Learning algorithms are an evolution of normal algorithms.**
- **They make your programs "smarter", by allowing them to automatically learn from the data you provide.**
- **The algorithm is mainly divided into:**
  - **Training Phase**
  - **Testing phase**


Training
Input past data
Machine learning Algorithm
Learn from data
Building Logical Models
Output
New data

Training Phase : Mango example

- **You take a randomly selected specimen of mangoes from the market (training data),**
- **make a table of all the physical characteristics of each mango,**
  - like **color, size, shape, grown** in which part of the country,
  - **sold by which vendor**, etc (**features**),
  - along with the **sweetness, juiciness, ripeness** of that mango (**output variables/classifier/predictor**).
- **You feed this data to the machine learning algorithm (classification/regression), and it learns a model of the correlation between an average mango's physical characteristics, and its quality.**

Machine Learning End Product

Ordinary System

With AI

Machine Learning

Learns

Predicts

Testing Phase

- **Next time when you go shopping, you will measure the characteristics of the mangoes which you are purchasing(test data)and feed it to the Machine Learning algorithm.**
- **It will use the model which was computed earlier to predict**
  - if the mangoes are sweet, ripe and/or juicy.
- **The algorithm may internally use the rules, similar to the one you manually wrote earlier (for eg, a decision tree).**
- **Finally, you can now shop for mangoes with great confidence, without worrying about the details of how to choose the best mangoes.**

Conclusion as an Algorithm

- You know what! you can make your algorithm improve over time (reinforcement learning) so that it will improve its accuracy as it gets trained on more and more training dataset.
- In case it makes a wrong prediction it will update its rule by itself.
- The best part of this is, you can use the same algorithm to train different models.
- You can create one each for predicting the quality of apples, grapes, bananas, or whatever you want.

Machine Learning Libraries – Comparison Chart

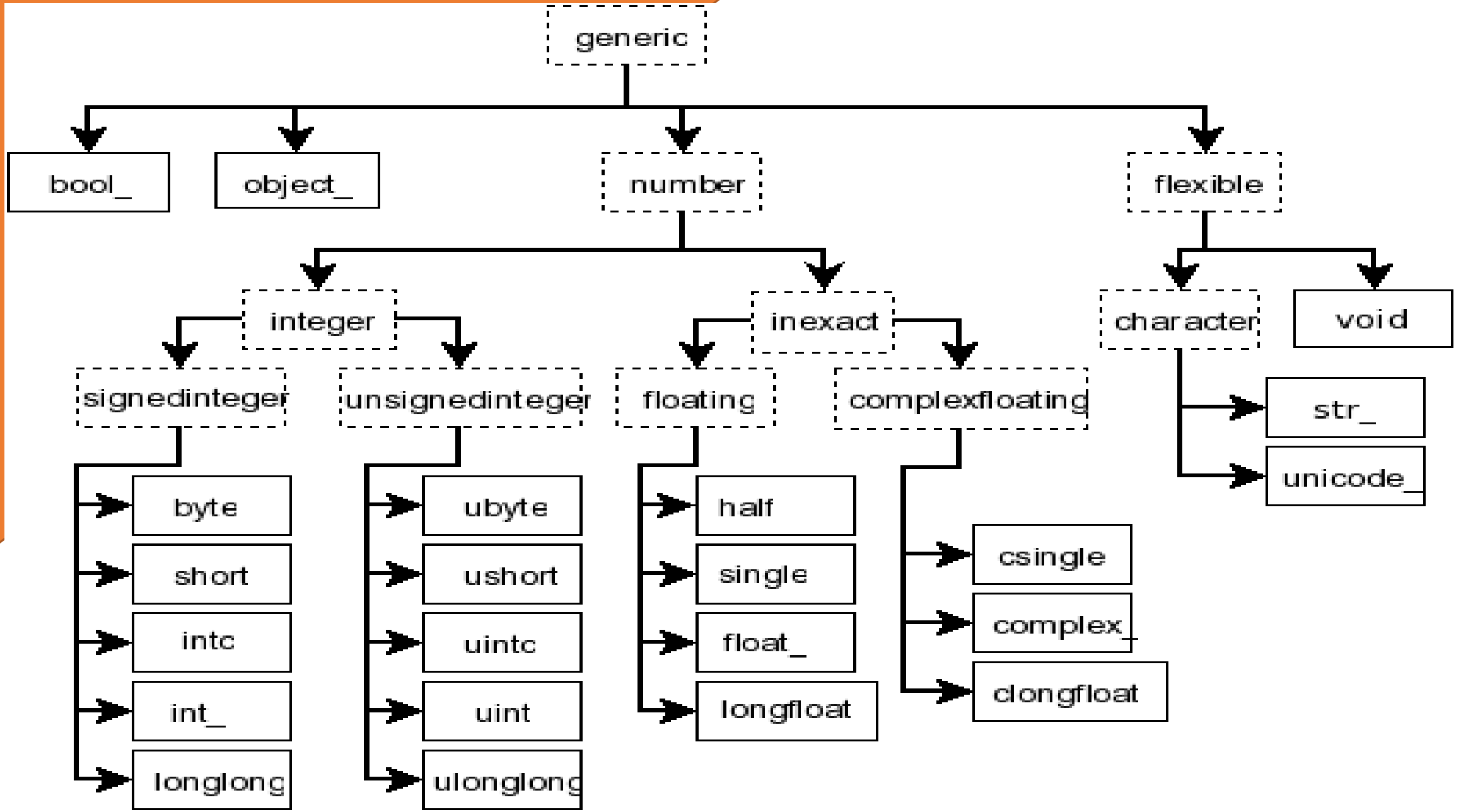
Category	Library	Primary Purpose	Key Features
Numerical Computing	NumPy	Fast numerical operations	Arrays, linear algebra, mathematical functions
Data Manipulation	Pandas	Data cleaning & analysis	DataFrames, merging, filtering, grouping
Visualization	Matplotlib	Basic plotting	Line, bar, scatter, histograms
	Seaborn	Statistical visualization	Heatmaps, boxplots, pairplots
Machine Learning	Scikit-Learn	Classical ML algorithms	Preprocessing, ML models, pipelines, metrics

What is Scikit-Learn?

- **Main concept**
  - A powerful open-source machine learning library for Python.
  - Built on NumPy, SciPy, and Matplotlib.
  - Provides simple, consistent APIs for building ML models.
  - Ideal for beginners and production-grade prototypes.
- **Key Features**
  - Easy-to-use APIs for ML operations.
  - Wide range of ML algorithms: regression, classification, clustering, etc.
  - Preprocessing tools: scaling, encoding, feature extraction.
  - Model evaluation tools: metrics, cross-validation.
  - Pipelines: automate workflow from pre-processing → modeling.
  - Built-in toy datasets: Iris, Digits, Wine, Breast Cancer.

Numpy Library

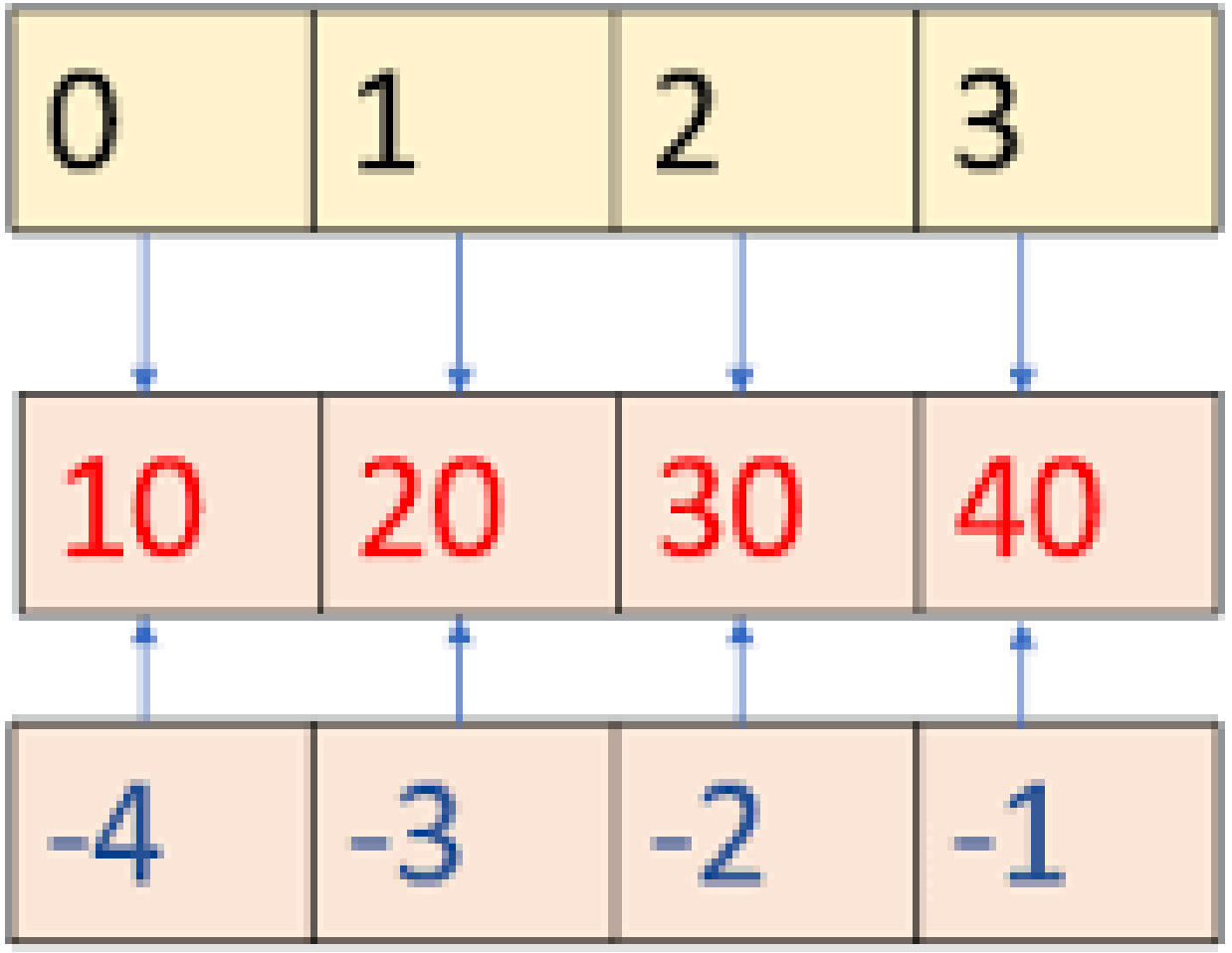
- Provides the foundation for **scientific computing in Python.**
- Supports **multi-dimensional arrays (ndarrays)** that are faster and more efficient than Python lists.
- Enables vectorized operations, reducing the need for loops and making code faster.
- Offers **powerful tools** for linear algebra, statistics, and mathematical operations.
- Forms the core dependency for many **ML libraries** like Pandas, Scikit-Learn, TensorFlow, PyTorch.
- Provides **broadcasting,** allowing operations on arrays of different shapes.
- Handles **large datasets** efficiently, crucial for machine learning workflows.
- Highly optimized using C/C++ backend, giving significant performance boosts.


generic
bool_
object_
number
flexible
integer
inexact
character
void
signedinteger
unsignedinteger
floating
complexfloating
str_
unicode_
byte
short
intc
int_
longlong
ubyte
ushort
uintc
uint
ulonglong
half
single
float_
longfloat
csingle
complex_
clongfloat

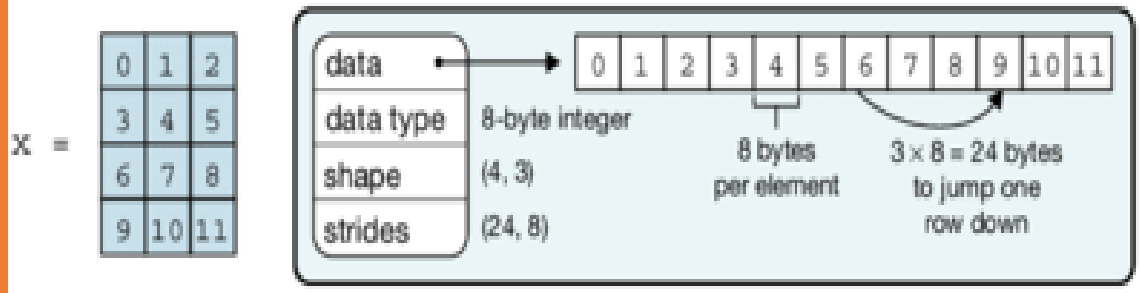
Positive Indexing

One_d

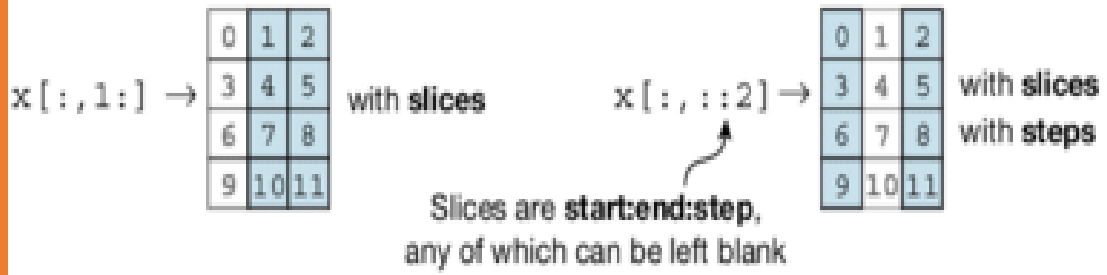
Negative Indexing



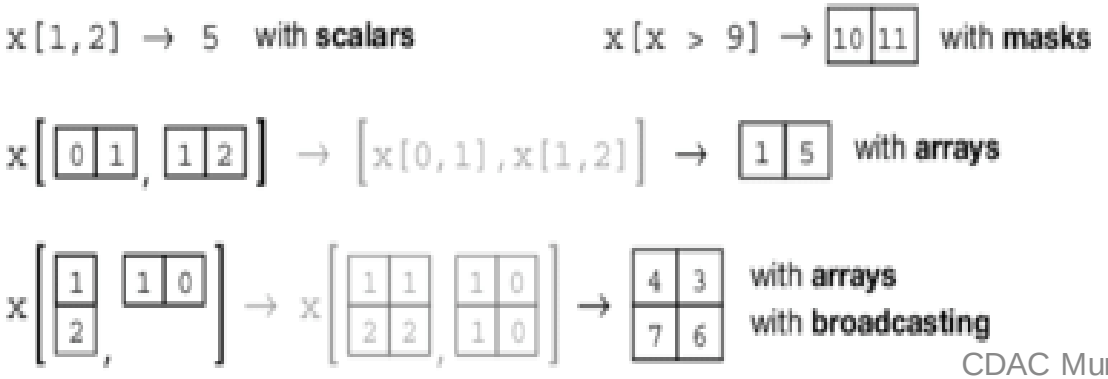
a Data structure



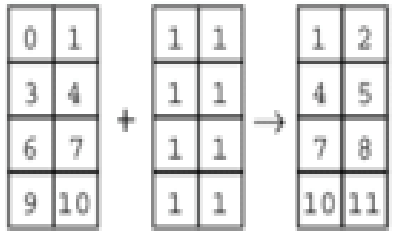
b Indexing (view)



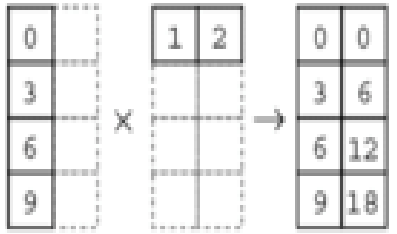
c Indexing (copy)



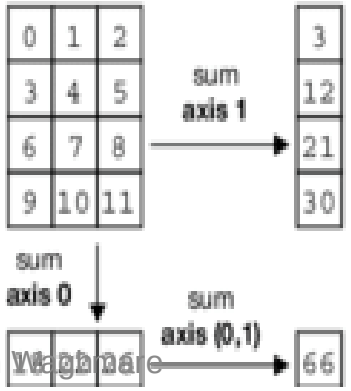
d Vectorization



e Broadcasting

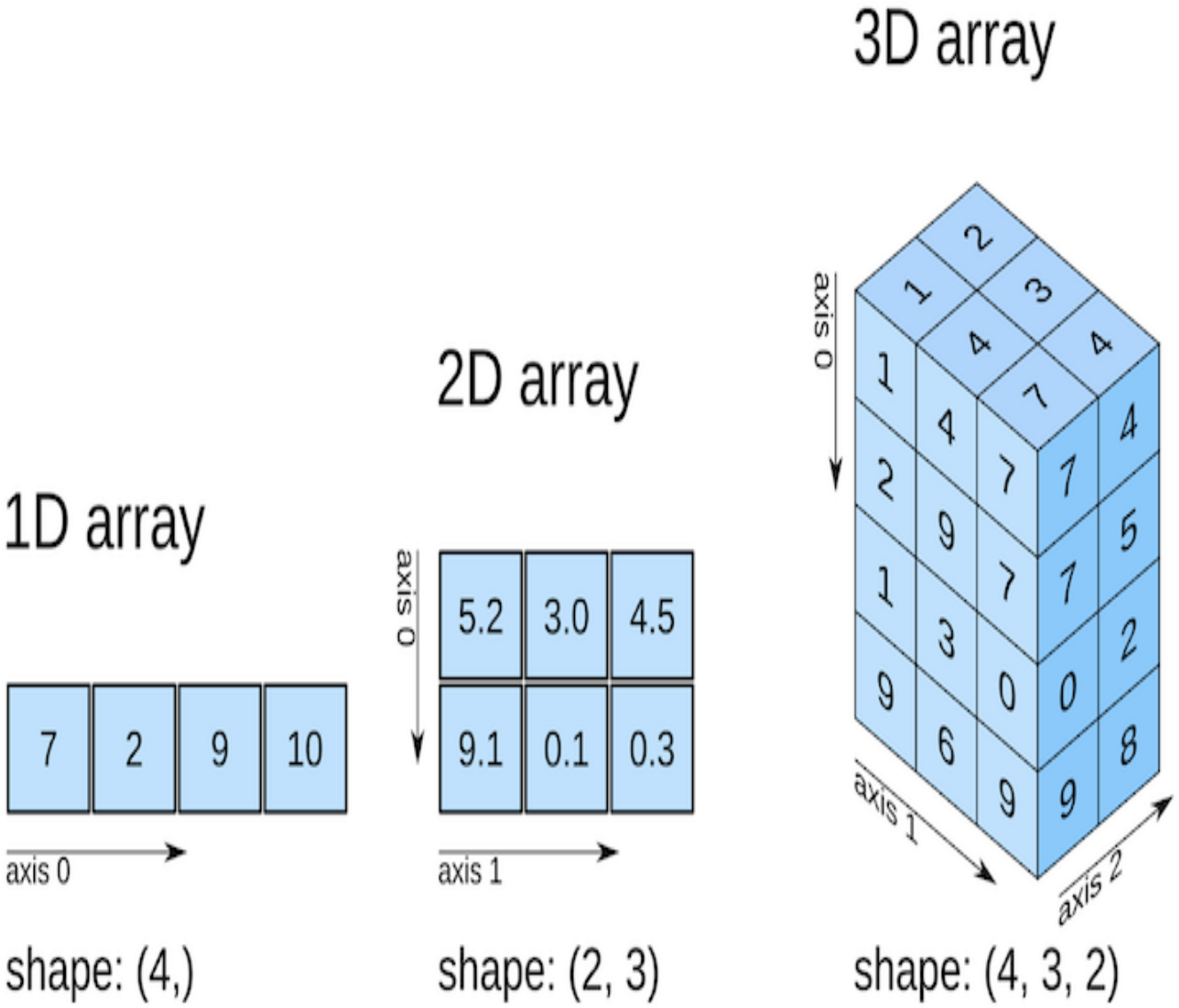


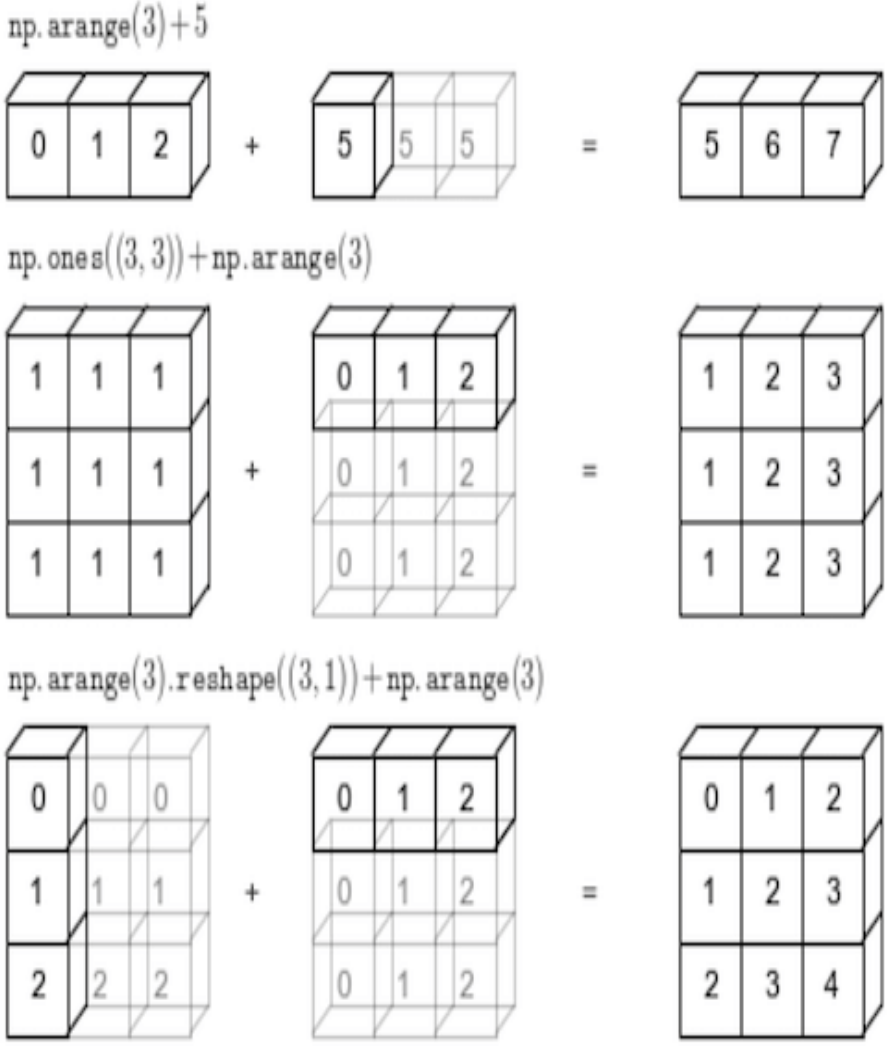
f Reduction



g Example

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

In [2]: x = np.arange(12)

In [3]: x = x.reshape(4, 3)

In [4]: x
Out[4]:
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])

In [5]: np.mean(x, axis=0)
Out[5]: array([4.5, 5.5, 6.5])

In [6]: x = x - np.mean(x, axis=0)

In [7]: x
Out[7]:
array([[-4.5, -4.5, -4.5],
       [-1.5, -1.5, -1.5],
       [ 1.5,  1.5,  1.5],
       [ 4.5,  4.5,  4.5]])
```


1D array
7 2 9 10
axis 0
shape: (4,)
2D array
axis 0
5.2 3.0 4.5
9.1 0.1 0.3
axis 1
shape: (2, 3)
3D array
axis 0
axis 1
axis 2
shape: (4, 3, 2)


np.arange(3)+5
0 1 2 + 5 5 5 = 5 6 7
np.ones((3,3))+np.arange(3)
1 1 1 / 1 1 1 / 1 1 1 + 0 1 2 / 0 1 2 / 0 1 2 = 1 2 3 / 1 2 3 / 1 2 3
np.arange(3).reshape((3,1))+np.arange(3)
0 / 1 / 2 + 0 1 2 / 0 1 2 / 0 1 2 = 0 1 2 / 1 2 3 / 2 3 4

NumPy Basics

NumPy Basics

Operator	Description
`np.array([1,2,3])`	1d array
`np.array([(1,2,3),(4,5,6)])`	2d array
`np.arange(start,stop,step)`	range array

Placeholders

Operator	Description
`np.linspace(0,2,9)`	Add evenly spaced values btw interval to array of length
`np.zeros((1,2))`	Create and array filled with zeros
`np.ones((1,2))`	Creates an array filled with ones
`np.random.random((5,5))`	Creates random array
`np.empty((2,2))`	Creates an empty array

Array

Operator	Description
`array.shape`	Dimensions (Rows,Columns)
`len(array)`	Length of Array
`array.ndim`	Number of Array Dimensions
`array.dtype`	Data Type
`array.astype(type)`	Converts to Data Type
`type(array)`	Type of Array

Copying/Sorting

Operator	Description
`np.copy(array)`	Creates copy of array
`other = array.copy()`	Creates deep copy of array
`array.sort()`	Sorts an array
`array.sort(axis=0)`	Sorts axis of array

Array Manipulation

Adding or Removing Elements

Operator	Description
`np.append(a,b)`	Append items to array
`np.insert(array, 1, 2, axis)`	Insert items into array at axis 0 or 1
`np.resize((2,4))`	Resize array to shape(2,4)
`np.delete(array,1,axis)`	Deletes items from array

Combining Arrays

Operator	Description
`np.concatenate((a,b),axis=0)`	Split an array into multiple sub-arrays.
`np.vstack((a,b))`	Split an array in sub-arrays of (nearly) identical size
`np.hstack((a,b))`	Split the array horizontally at 3rd index

More

Operator	Description
`other = ndarray.flatten()`	Flattens a 2d array to 1d
`array = np.transpose(other)`	Transpose array
`array.T`	Transpose array
`inverse = np.linalg.inv(matrix)`	Inverse of a given matrix

Slicing and Subsetting

Operator	Description
`array[i]`	1d array at index i
`array[i,j]`	2d array at index[i][j]
`array[i<4]`	Boolean Indexing, see Tricks
`array[0:3]`	Select items of index 0, 1 and 2
`array[0:2,1]`	Select items of rows 0 and 1 at column 1
`array[:1]`	Select items of row 0 (equals array[0:1, :])
`array[1:2, :]`	Select items of row 1
`[comment]: <> (`	array[1,...]
`array[ : :-1]`	Reverses array

Mathematics

Operations

Operator	Description
`np.add(x,y)`	Addition
`np.substract(x,y)`	Subtraction
`np.divide(x,y)`	Division
`np.multiply(x,y)`	Multiplication
`np.sqrt(x)`	Square Root
`np.sin(x)`	Element-wise sine
`np.cos(x)`	Element-wise cosine
`np.log(x)`	Element-wise natural log
`np.dot(x,y)`	Dot product
`np.roots([1,0,-4])`	Roots of a given polynomial coefficients

Comparison

Operator	Description
`==`	Equal
`!=`	Not equal
`<`	Smaller than
`>`	Greater than
`<=`	Smaller than or equal
`>=`	Greater than or equal
`np.array_equal(x,y)`	Array-wise comparison

Basic Statistics

Operator	Description
`np.mean(array)`	Mean
`np.median(array)`	Median
`array.corrcoef()`	Correlation Coefficient
`np.std(array)`	Standard Deviation

More

Operator	Description
`array.sum()`	Array-wise sum
`array.min()`	Array-wise minimum value
`array.max(axis=0)`	Maximum value of specified axis
`array.cumsum(axis=0)`	Cumulative sum of specified axis

Pandas

- Provides **DataFrame** and **Series** structures for easy data handling
- Makes **data cleaning, filtering, and transformation** simple and fast
- Offers powerful **data analysis** tools (grouping, aggregation, summarization)
- Supports **reading and writing** data from multiple formats (CSV, Excel, JSON, SQL)
- Enables easy **handling of missing values**
- Integrates seamlessly with **NumPy, Matplotlib, Scikit-Learn**
- Allows **label-based indexing** and **intuitive selection** of rows/columns
- Great for **preprocessing data** before machine learning
- Used extensively for **EDA (Exploratory Data Analysis)**
- Highly optimized for performance and real-world data workloads

Pandas Library

Create Test Objects

Operator	Description
`pd.DataFrame(np.random.rand(20,5))`	5 columns and 20 rows of random floats
`pd.Series(my_list)`	Create a series from an iterable my_list
`df.index = pd.date_range('1900/1/30', periods=df.shape[0])`	Add a date index

Viewing/Inspecting Data

Operator	Description
`df.head(n)`	First n rows of the DataFrame
`df.tail(n)`	Last n rows of the DataFrame
`df.shape`	Number of rows and columns
`df.info()`	Index, Datatype and Memory information
`df.describe()`	Summary statistics for numerical columns
`s.value_counts(dropna=False)`	View unique values and counts
`df.apply(pd.Series.value_counts)`	Unique values and counts for all columns

Selection

Operator	Description
`df[col]`	Returns column with label col as Series
`df[[col1, col2]]`	Returns columns as a new DataFrame
`s.iloc[0]`	Selection by position
`s.loc['index_one']`	Selection by index
`df.iloc[0,:]`	First row
`df.iloc[0,0]`	First element of first column

Data Cleaning

Operator	Description
`df.columns = ['a','b','c']`	Rename columns
`pd.isnull()`	Checks for null Values, Returns Boolean Arrray
`pd.notnull()`	Opposite of pd.isnull()
`df.dropna()`	Drop all rows that contain null values
`df.dropna(axis=1)`	Drop all columns that contain null values
`df.dropna(axis=1,thresh=n)`	Drop all rows have have less than n non null values
`df.fillna(x)`	Replace all null values with x
`s.fillna(s.mean())`	Replace all null values with the mean
`s.astype(float)`	Convert the datatype of the series to float
`s.replace(1,'one')`	Replace all values equal to 1 with 'one'
`s.replace([2,3],['two', 'three'])`	Replace all 2 with 'two' and 3 with 'three'
`df.rename(columns=lambda x: x + 1)`	Mass renaming of columns
`df.rename(columns={'old_name': 'new_ name'})`	Selective renaming
`df.set_index('column_one')`	Change the index
`df.rename(index=lambda x: x + 1)`	Mass renaming of index

Filter, Sort, and Groupby

Operator	Description
`df[df[col] > 0.6]`	Rows where the column col is greater than 0.6
`df[(df[col] > 0.6) & (df[col] < 0.8)]`	Rows where 0.8 > col > 0.6
`df.sort_values(col1)`	Sort values by col1 in ascending order
`df.sort_values(col2,ascending=False)`	Sort values by col2 in descending order.5
`df.sort_values([col1,col2],ascending=[True,False])`	Sort values by col1 in ascending order then col2 in descending order
`df.groupby(col)`	Returns a groupby object for values from one column
`df.groupby([col1,col2])`	Returns groupby object for values from multiple columns
`df.groupby(col1)[col2]`	Returns the mean of the values in col2, grouped by the values in col1
`df.pivot_table(index=col1,values=[col2,col3],aggfunc=mean)`	Create a pivot table that groups by col1 and calculates the mean of col2 and col3
`df.groupby(col1).agg(np.mean)`	Find the average across all columns for every unique col1 group
`df.apply(np.mean)`	Apply the function np.mean() across each column
`nf.apply(np.max,axis=1)`	Apply the function np.max() across each row

Join/Combine

Operator	Description
`df1.append(df2)`	Add the rows in df1 to the end of df2 (columns should be identical)
`pd.concat([df1, df2],axis=1)`	Add the columns in df1 to the end of df2 (rows should be identical)
`df1.join(df2,on=col1, how='inner')`	SQL-style join the columns in df1 with the columns on df2 where the rows for col have identical values. The 'how' can be 'left', 'right', 'outer' or 'inner'

Statistics

Operator	Description
`df.describe()`	Summary statistics for numerical columns
`df.mean()`	Returns the mean of all columns
`df.corr()`	Returns the correlation between columns in a DataFrame
`df.count()`	Returns the number of non-null values in each DataFrame column
`df.max()`	Returns the highest value in each column
`df.min()`	Returns the lowest value in each column
`df.median()`	Returns the median of each column
`df.std()`	Returns the standard deviation of each column