Research on NumPy in Python

Q: What is 'Model' in AI/ML?

Ans: A mathematical function that learns patterns from data to make predictions or decisions.

Model = f(data, parameters)

Data -> In ML predictions made on numeric values so data must be numbers

Parameters -> Again those number that model can learn

Output -> Prediction Obviously

Example:

Input: Study hour -> 5

Output: Exam score -> 78

What model do is the rule that map 5->78 and use these leanings for further predictions

Models are not Magic

A model:

- does NOT understand words
- does NOT see images
- does NOT think like humans

A model only understands numbers

Everything:

- text
- images
- audio
- video

must be converted into numbers(arrays) to generate result or predictions

That why NumPy is the backbone of Machine Learning

Q: How Model “See” Data?

To a model it is like:

Real World thing	Model sees
Image	Matrix of Pixel values
Text	Sequence of number can be ascii values
Sound	Number Array of amplitudes
Table	2d numerical array

Core Idea

A ML model never sees:

[percentage, attendance, previous_avg_percentage]

It only sees numbers arranged in space:

[72.5, 80, 68.7]

This is how model see a student.

A single data point is not a data it’s a vector. And converting data into numeric values is called features data as vectors.

For example, data of multiple students:

[

[72.5, 80, 68],

[55.0, 60, 50],

[88.0, 90, 85]

]

Each row contains one student and each column contain one feature like 1st column contain feature of percentage.

This idea is the foundation of ALL ML and DL.

Supporting video link: https://www.youtube.com/watch?v=VGQRQHw_Vgw

Data Representation: The Heart of ML

Hours Studied	Attendance	Score
2	60	45
4	75	65
6	90	85

To humans -> this is a table contain data of students

To model -> we have to create matrix of that

```
X = [
  [2, 60],
  [4, 75],
  [6, 90]
]
```

```
y = [45, 65, 85]
```

This is liner algebra involved here, not just python lists

Features, Labels, and Dimensions

Features: Input to the model

Example: Hours Studied, Attendance

Label (Target): What we want to predict

Example: Score

Dimensions Matter

- **1D array** → vector
- **2D array** → matrix
- **3D+ array** → tensors

Example: X.shape = (3,2)

Means:

- 3 samples of student's data are there
- 2 Features are there against the samples that provide data on which prediction has to be done

So, Models are EXTREMELY sensitive to shape.

Q: How Models Learn?

Models learn by:

- Making a predictions
- Measuring error
- Adjusting Parameters
- Repeating

Example:

$y = w_1x_1 + w_2x_2 + b$

All variables:

- x → data
- w, b → learned numbers

Training = finding the **best numbers**

If you don't understand don't worry, you'll learn all this in future and coming lectures.

Q: Why Python lists are NOT enough?

Python lists:

- slow
- no math awareness
- no broadcasting
- no linear algebra

Example:

Input: [1, 2, 3] * 2

Output: [1, 2, 3, 1, 2, 3]

And this is wrong for ML so for that we need NumPy here.

NumPy was built to:

- store numerical data efficiently
- perform **vectorized math**
- handle large datasets
- support linear algebra

ChatGpt Example:

- text → tokens → numbers → vectors
- everything = math

Q: Why NumPy Exists?

Python lists are for humans

NumPy arrays are for math + speed + models

Python Lists vs NumPy arrays

Python List: A list is a general-purpose container. **Characteristics** • Can store mixed data types • Elements stored as references • No built-in math understanding • Slower for numerical computation	**NumPy Array:** A NumPy array is a numerical data structure. **Characteristics** • Stores **only one data type** • Stored in **contiguous memory** • Optimized for math & linear algebra • Fast and vectorized

Example:

```
lst = [1, 2, 3]

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

print(lst * 2)

print(arr * 2)
```

Output:

```
[1, 2, 3, 1, 2, 3]

[2 4 6]
```

NumPy Arrays:

1D Array (Vector):

```
a = np.array([1, 2, 3, 4])
```

What will be shape: (4,) which means

- 4 elements
- 1 dimension

2D Array (Matrix):

```
b = np.array([
  [1, 2, 3],
  [4, 5, 6]
])
```

What will be shape: (2, 3) which means

- 2 rows
- 3 columns

3D Array (Tensor):

```
c = np.array([
  [[1, 2], [3, 4]],
  [[5, 6], [7, 8]]
])
```

What will be shape: (2, 2, 2) which means

- 2 blocks
- each block has 2 rows
- each row has 2 columns

Used in images, video, deep learning.

Special Array Creator:

```
np.zeros((3, 4))      # all zeros
np.ones((2, 2))       # all ones
np.arange(0, 10, 2)   # range
np.linspace(0, 1, 5)  # evenly spaced
np.eye(3)             # identity matrix
```

What is "Shape" of an Array?

Array	Shape	Meaning
[1,2,3]	(3,)	**1D vector**
[[1,2],[3,4]]	(2,2)	**2x2 matrix**
Image (RGB)	(H, W, 3)	**Height, width, channels**

Models REQUIRE correct shapes.

You can check my practice in my Git hub profile by clicking on the link.

Think like a Model >>

Q: Why a model cannot use variables or strings containing data?

Ans: Model always understand numeric values to give predictions and store data by featuring data as vectors in which columns represent the feature and row represents the single instance or sample of data.

Q: Why scaling features matters? Is [1, 100] same importance as [50, 60]? What happens if one feature dominates others?

Ans: Scaling matters because ML models rely on numerical magnitude and distance. If features are on different scales, larger-valued features dominate the learning process unfairly, leading to biased or unstable models.

If one feature dominates, the model becomes biased toward it, ignores other useful features, and may generalize poorly on unseen data.

Feature Scaling is making features comparable in magnitude. Because model don't know the importance of feature, they only see magnitude.

Question arises: Which number will dominate the multiplication? So, it will be Attendance (100) while study hour (1) so it is not logical in real life and model don't know what we mean, its show the predictions based on just magnitudes.

In [1,100], one feature **completely dominates the space**.

In [50,60], features are: closer in scale, comparable, balanced

That's why we scale: We don't change meaning but we change range.

Examples:

- Marks: 0–100
- Attendance: 0–100
- Hours studied: 0–10

Without scaling -> model is biased.

With scaling -> model is fair

[1, 100] is not same like [50, 60]

If one feature dominates:

- Model focuses mostly on that feature
- Other features become almost invisible
- Learning becomes biased
- Model may perform poorly on new data