dtype = np.float64 // দশমিক সংখ্যা 64 bit ধারণ করে।

Code → 41

```
import numpy as np

x = np.array ( [[1,2] ,[3,4] ], dtype = np.float64)
y = np.array ( [[5,6], [7,8] ], dtype = np.float64)

Print (x+y)
Print (np.add(x,y))
```

Output :

এখানে দুটো Matrix (ম্যাট্রিক্স) যোগ করেছি, Element wise যোগ করেছি।

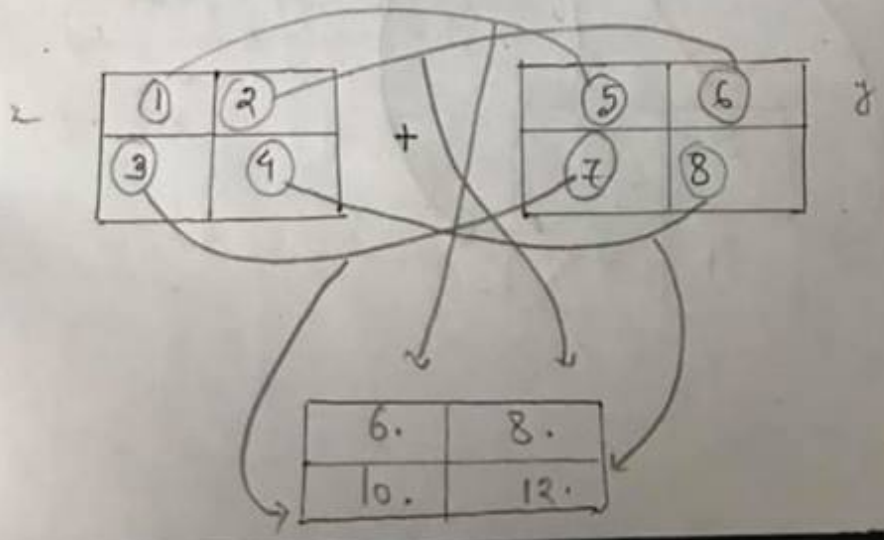
```
[ [ 6.   8.]
  [ 10.  12. ] ]

[ [6.   8.]
  [10.  12.]]
```

x:

1	2
3	4

y:

5	6
7	8

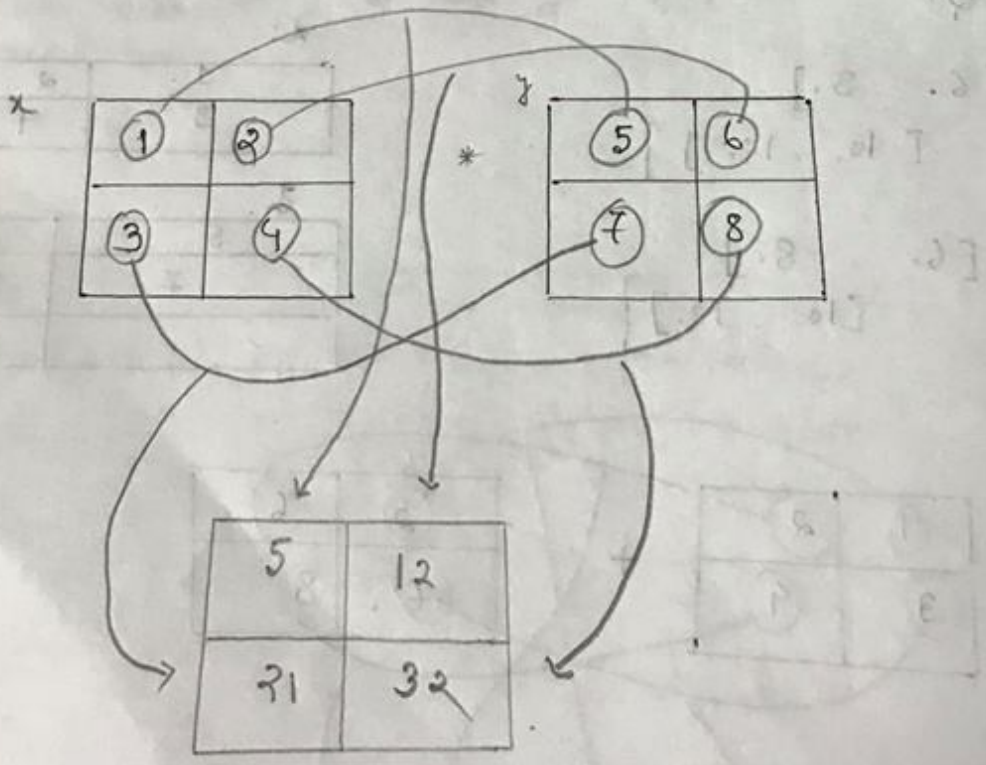


Code → 42

```
import numpy as np
x = np.array([[1,2],[3,4]], dtype = np.float64)
y = np.array([[5,6],[7,8]], dtype = np.float64)

z = x*y
Print (z)
```

Output: এটা এর Element wise হবে।

```
[ [5.  12.]
  [21. 32.]]
```



Code→43 — Matrix এর প্রতিটা element এর বর্গমূল বের করা যায়।

```
import numpy as np
x = np.array([[1,2],[3,4]], dtype = np.float64)
Print (np.sqrt(x))
```

Output:

```
[ [1.        1.41421]
  [1.73205   2.     ]]
```

Code→44 — Matrix গুণের জন্য dot function ব্যবহার করা হয়।

```
import numpy as np
x = np.array([[1,2],[3,4]])
y = np.array([[5,6],[7,8]])

v = np.array([9,10])
w = np.array([11,12])
print(v.dot(w))
print(np.dot(v,w))
```

Output:

219
219

[9, 10] . dot [11, 12]
=> [9, 10] . [11, 12]
=> 9×11 + 10×12
=> 99 + 120
=> 219

np.dot(v, w)
=> np. [9, 10] . [11, 12]
=> np. 9×11 + 10×12 => np. 99+120 => np.

Code->45

```
import numpy as np
x = np.array([[1,2],[3,4]])
y = np.array([[5,6],[7,8]])
v = np.array([9,10])
w = np.array([11,12])

Print(x.dot(v))
Print(np.dot(x,v))
```

Output:

```
[29  67]
[29  67]
```

x.dot(v)

1	2
3	4

.dot([9 10])

=> | 1 | 2 | . [9 10]
| 3 | 4 |

=> [9×1+2×10 3×9+4×10]

=> [9+20 27+40]

=> [29 67]

Code → 46

```
import numpy as np
x = np.array ( [ [1,2], [3,4] ])
y = np.array ( [ [5,6], [7,8] ])
Print (x.dot (y))
Print ( np.dot (x,y))
```

Output:

```
[ [19  22]
  [43  50] ]

[ [19  22]
  [43  50] ]
```

x

1	2
3	4

y

5	6
7	8

$$\begin{bmatrix} 5\times1+2\times7 & 6\times1+2\times8 \\ 3\times5+4\times7 & 3\times6+4\times8 \end{bmatrix}$$

$$= \begin{bmatrix} 5+14 & 6+16 \\ 15+28 & 18+32 \end{bmatrix}$$

$$= \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

=> list এ যোগ করলে, list এর element গুলো concate হয়ে যায়।

=> Matrix এর Element wise multiplication এর জন্য dimension একই হওয়া দরকার।

=> Matrix Multiplication এর জন্য

$[m \times n] * [n \times z] = m \times z$

$n = n$

[row x column] [row x column]

Code -> 47

Column হওয়া, Column গুলো row হওয়া।

```
import numpy as np
x = np.array([[1,2],[3,4]])
Print(x)
Print(x.T)  // .T দিলে Transpose হয়,
```

Output:

Transpose করার আগে =>

1	2
3	4

Transpose করার পর =>

1	3
2	4

Code → 48

```
import numpy as np

x = np.array([[1,2],[3,4]])
Print(np.sum(x))
Print(np.sum(x, axis=0))
Print(np.sum(x, axis=1))
```

Output:

→ 10 // 1+2+3+4

→ [4 6] // axis=0 হলে Column গুলো sum করে return করবে

1	2
3	4

→ [3 7] // axis=1 হলে row গুলো sum করে return করবে।

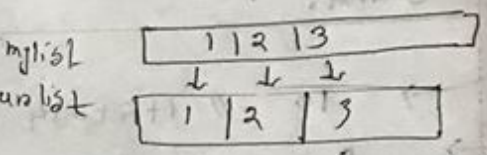
দ্বিতীয় 2d array // return করবে 1d array.

Value চুজের যেভাবে যায়,

i. By Reference
ii. By Value.

Code → 49.

```
import numpy as np

mylist = [1, 2, 3]
unlist = mylist // Reference assign
mylist.append(4)
Print(mylist)
Print(unlist)

unlist.remove(1)
Print(mylist)
Print(unlist)
```



↓ এখানে reference Pointing

Output:

```
[1, 2, 3, 4]
[1, 2, 3, 4]

[2, 3, 4]
[2, 3, 4]
```

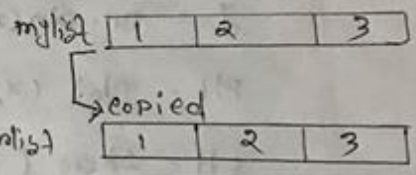
এটা বলতেই হবে যে জায়গায় থেকে remove হচ্ছে আরো আগের জায়গায় a value ও রয়েছে।

এখন আমরা বলছি, unlist = mylist.

So আমরা by reference বলতেছি।

Code → 50

```
import numpy as np
mylist = [1,2,3]
unlist = mylist.copy() # mylist এর element গুলোর copy করে unlist এ রাখো
mylist.append(4)
Print(mylist)
Print(unlist)
unlist.remove(1)
Print(mylist)
Print(unlist)
```



Output:

```
[1, 2, 3, 4]
[1, 2, 3]
[1, 2, 3, 4]
[2, 3]
```

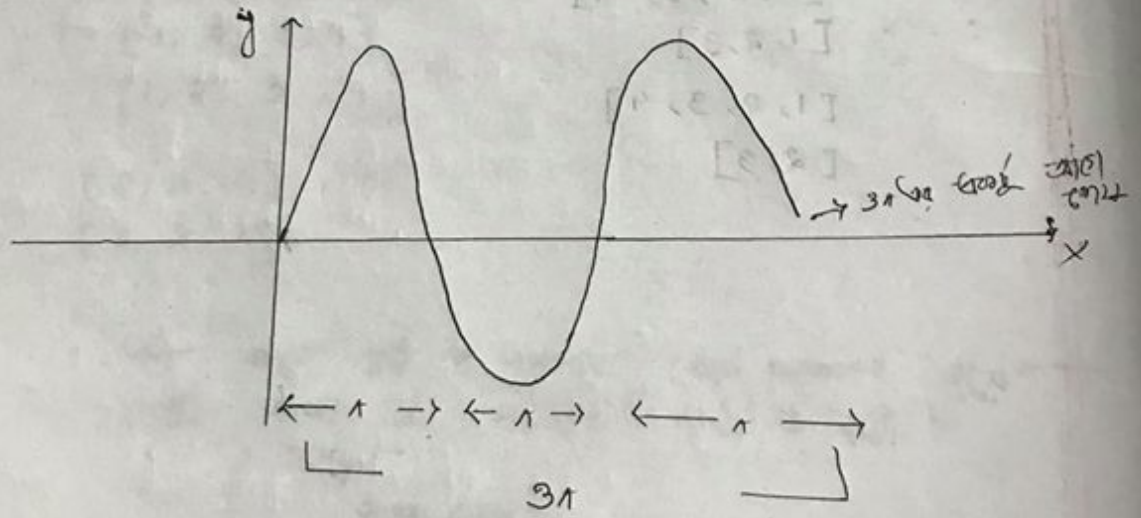
Python এ graph আঁকার জন্য

Code-১

matplotlib.pyplot ব্যবহার করা হয়,

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

x = np.arange(0, 3*np.pi, 0.1)
y = np.sin(x)
plt.plot(x,y)
plt.show()
```

Output:



"arange() => Returns evenly spaced values within a given interval.

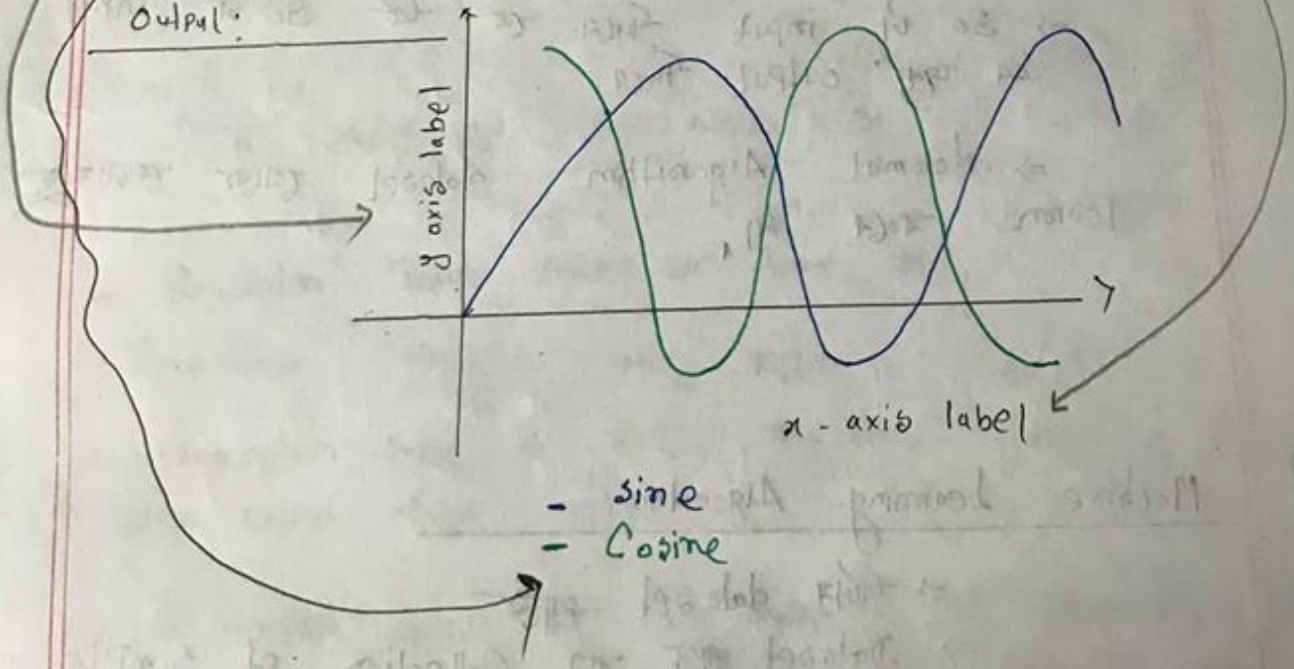
for (0, 3*np.pi, 0.1)

start, end, stepsize

Code→50

```
import numpy as np
x = np.arange(0, 3*np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)

plt.plot(x, y_sin)
plt.plot(x, y_cos)
plt.xlabel('x axis label')
plt.ylabel('y axis label')
plt.title('Sine & Cosine')
plt.legend(['Sine', 'Cosine'])
plt.show()
```

Output:



legend() => labeling existing plot elements.

Machine Learning Concept

Normal Algorithm Vs Machine Learning Algorithm

Normal Algorithm:

=> Normal Program এটা নির্দিষ্ট input এর জন্য নির্দিষ্ট output দিবে,

=> 30 টা input দিলে যে output দিবে, ঠিক 30 টা input এর জন্য

=> Normal Algorithm dataset থেকে কোনো কিছু learn করে না,

Machine Learning Algorithm:

=> এটার dataset লাগে

=> Dataset মানে হল Collection of samples.

=> Sample মানে, কিছু input দিলে feature এর উপর base করে output দিবে,

[মানে Sample => input feature output]

=> Machine Learning অনেকটা একটা ছোট বাচ্চা এভাবে দেখে দেখে শিখে যেভাবে কাজ করে।

=> Sample একটা হতে পারে, অনেক অধিক হতে পারে।

=> Dataset এ sample যত বেশি হবে, তত বেশি Accurate result পাওয়া সম্ভব হবে।

=> আমার কাছে একটা dataset আছে, যার মধ্যে 50 টা bus, 50 টা motor-cycle, 50 টা car এর ছবি আছে। image size 1024 x 1024 করে, অর্থাৎ 1024 টি Pixel। এই Pixel কে আমরা বলবো input feature.

=> image size হবে (1024 x 1024 x 3)
↳ sample.

=> Parameter : Input - Output কে tune করে।

=> Parameter : Matrix আকারে থাকে।

=> Parameter tune না করলে Machine Learning Algorithm সঠিক output দিতে পারবে না,

=> যদি আমার dataset এ 500 বাসের আর 500 মোটর-বাইকের ছবি আছে। আমি যদি প্লেনের ছবি দেই, তাহলে Algorithm সঠিক output দিবে না। কারণ dataset কে আমি প্লেনের ছবি দেখাই নাই আগে। মানে tune করি নাই।

=> Machine Learning Algorithm এর ২টা type হয়।

i. Classification
ii. Regression

→ Machine Learning এ Value ২ ধরনের হতে পারে

i. Discrete Value: Value এর সংখ্যা limited
যেমন: আমরা বলতাম ০ থেকে ৭ পর্যন্ত সংখ্যা, যেখানে দশমিক সংখ্যা not allowed. তাহলে 10 টা সংখ্যা।

ii. Continuous Value: Value এর সংখ্যা limited নয়। যেমন: ০ থেকে ৭ পর্যন্ত সংখ্যা নিলাম আর বলতাম দশমিক সংখ্যাও allowed.

Classification: যে সব Machine Learning model discrete value return করে।

Regression: যে সব Machine Learning model continuous value return করে।

→ Google এ কোন কিছু search করার সময় word suggestion আসে। যে word এর Probability বেশি, সেটার index return করে দেয়, এটা হল sequence to sequence model. এটা হয় sequence wise Probability wise

→ Word by word suggestion দুই ধরনের হয়।

i. global
ii. Personal

→ Estimation এর ক্ষেত্রে Regression Model use হয়।

Univariate: Input feature যদি ১টা

Multivariate: ১ এর বেশি input feature হলে multivariate.

→ Dataset যে format এ থাকে

i. .csv
ii. .data
iii. .txt
iv. .xlsx [Excel file]

=> Missing Value: কোন কোন feature এর জন্য input value missing.

=> input value missing থাকলে, সবগুলো value -এর average নিয়ে বসিয়ে দিতে হয়।

=> Balanced Data: এ প্রতিটা class এ সমান সংখ্যক data থাকবে। যেমন:

class 1 : 50 data
class 2 : 50 data
class 3 : 50 data

=> Dataset যত balanced হবে, classification এর জন্য তত ভাল হবে।

=> Machine learning এ মূলত দুটো কাজ করা হয়,

Machine learning

Algorithm নিয়ে কাজ

Pre Processing dataset.

KNN Algorithm

KNN Classifier / Classification

=> KNN [K Nearest Algorithm]

=> ধরি, green, red, blue এই তিন Color এর বেশ অনেকগুলো Point আছে।

তাহলে এই তিনটা Color হল class

class 1 → green
class 2 → red
class 3 → blue.

=> এখন আমাকে নতুন একটা পয়েন্ট দেওয়া হল x_0, y_0 । এ পয়েন্ট টা কোন class এর হবে সেটা বের করবো,

$\sqrt{(x-x_0)^2 + (y-y_0)^2}$ এর euclidean distance বের করবো, Training set এ 100 টা Point থাকলে, 100 টা euclidean distance বের করবো।

=> K=10 assume করে করি.

=> Euclidean distance গুলো Ascending order এ sort করি,

=> প্রথম 10 টা distance Pick করি,

=> এই ১০ টা পয়েন্ট এর ভিতর যে ক্লাসের সংখ্যা বেশি (x0, y0) পয়েন্টটা সেই ক্লাসের হবে।

KNN Regression:

- Regression এর ক্ষেত্রে

- আমার কাছে ৩ টা class আছে,

k=5

Class 1 → 2
Class 2 → 3.2
Class 3 → 4.5
Class 4 → 6.3
Class 5 → 7.4

Average বের করো = $\frac{2+3.2+4.5+6.3+7.4}{5} = 4.68$

এই average টা হল regression value

=> K এর value কত নিব সেটা experimental

KNN এ আমরা dataset নিয়ে কাজ করি। এই dataset কে তিনভাগে ভাগ করতে পারি,

i. Training set
ii. Validation set
iii. Test set.

আমার কাছে ধরি 100% পয়েন্ট আছে। তাহলে সেটাকে ২ ভাবে ভাগ করতে পারি.

	Training set	Validation set	Test set
1.	80 %	10 %	10 %
2.	70 %	15 %	15 %

⇒ 150 টা পয়েন্ট থেকে 100 টা নিব training set এ। Validation set এর মূল কাজ হল accuracy বাড়ানো

ধরি, $k=5$

আমার 25 টা sample আছে। এই 25 টার class আমি জানি।

এর ভিতর 20টার actual class বিশ্লেষণ করার পর আমলো। 5 টা ভুল আসছে।

তাহলে Accuracy $= \frac{20}{25} \times 100\% = 80\%$

→ k=10 এর জন্য
Accuracy আসলো 90%.
k এর value বাড়ালে, accuracy বাড়বে।

→ k=15 এর জন্য
ধর.
Accuracy আসলো 85%.

তাহলে এমন,
k=10 থেকে 15 এর ভিতর এই range থেকে accuracy টা maximize করার চেষ্টা করব।

K	Accuracy
5	80%
10	90%
15	85%

=> Validation set Accuracy বাড়ায়.
=> Validation Accuracy Maximize করব,
=> Regression এর ক্ষেত্রে আমরা Error minimize করি।

Page-713

Regression

ধরি,
আমার কাছে 25 টা Sample আছে.

	KNN	Actual
Val 1	2.5	2.6
Val 2	2.4	3.5
Val 3	3.5	3.5
⋮		
Val 25	-	

$$Ennon = \sqrt{(2.5-2.6)^2 + (2.4-3.5)^2 + (3.5-3.5)^2 + \dots}$$

$$e = \frac{1}{25} \times Ennon.$$

=> K এর মান কমায় বেশি করে দেখবো, error minimize হচ্ছে কিনা,

=> K-NN algorithm কোন Parameter input নেয় না।

=> K এর মত hyper Parameter ~~tunning~~ tuning

=> যাতে value change করলে ভালো হয় hyper Parameter.

⇒ Test set :

- Test set actual accuracy বলে।
- Test set এ কোন কিছু change করা যাবে না।

⇒ Validation set থেকে sample নিব

⇒ যেই sample ; training set এর data তে —

k NN algorithm apply করবো।

⇒ Apply করলে class বের হবে।

⇒ 25 টা sample থাকলে 25 বার করবো apply

Data Processing

- প্রথম কথা Numpy array তে string support করে না,
- দ্বিতীয় কথা Numpy array তে append() নাই,

⇒ Data কে file থেকে পড়ে নেয়া ও যেভাবে তোয়া করা হয়।