p = int(input("Enter a string: "))
print(p)

Day: 1.10 Date: / /

0 - 9

① format sa values fix kr skty hai

print("value = {}, value1 = {}".format(3.1, 2))
// value = 3.1, value1 = 2

② ⇒ List ek data type beh.

```
listst = [2,3,7,9]
print(listst[1:4])   // print(379)
```

⇒ nested list.

```
l = [2,3,[1,2,4],5,6]
```

⇒ print(list(range(1,10))) // 1-9.

③ Sorting

We cannot sort string directly. First, we have to convert them into list than we can sort them.

```
s = 'Tushi is good'
s2 = list(s)
s2.sort()
print(s2)
```

Ais ma char ma bara bara.

```
s2.sort()
s3 = ''.join(s2)
print(s3)
```

Ais ma wapis join kr dia string ma aaiega karay ga.

④ l.append("A") // It will add to the last of the string/list.

l.insert(1, 'B')

Ja hamesha parameter ma 1 hae value la ga.

l.remove("A") ← You want to like 'A'

↳ Ais kay andr bs wohi aye ga jo remove wala scene hua hai. koi index nai hai.

length calculate krnay ka liy

Day: print (len(name jis ki calculate krni)) Date: / /

Agr index ka pta. hy tou remove.

def l[2]

→ tou second index removed.

⑤ Tuples:-

p = 1,2 — or p=(1,2)

x,y = p

print (y) print (p[1:])

→ Tuple can not be changed.

⑥ Dictionaries:-

key values.

dict = { 122 : 'Zaibi',
127 : 'Hashi',
2041 : 'Shabu'
}

print (dict) // Full dictionary print. (dict), ma

print (dict[127]) // 'Hashi' ⇒ Agr mai agr 127 nhi hy tou error

print (dict.get(2041)) ⇒ Agr ma none aya ga. agr 2042 dictionary ma nhi hy

print (dict.keys())

print (dict.values())

print (dict.items())

Day: Date: / /

Question:-
① colon list ka last ma atta hy. ya nai.
② def fun(l1):
return
print [x**2 for x in l1]

⇒ // if else

⇒ loop:-

```
for i in [1,2,3,4,5]:
    print (i)
```

⇒ Range:-

```
for con in range(5)    # it starts from 0
    print (i)
```

⇒ for dictionaries:

```
for key, value in Params.items()
    print (Key + "=" + str(value))
```

⇒ Example:-

```
for( int i=0; i<= num; i++ )
cout << i << " " << a[i]
```

C++

↓ Same in python

print number and their names.

```
for idx, x in enumerate (lists_my)
    print (idx, x)
```

⇒ l1 = [x**2 for x in range (0,5)]

⇒ while i<10
print(i)

Date: / /

⇒ A Keyword for function called def,

⇒ int Square (int n) // C++

⇒ # python

```
def fun(n):
    return n**2, n**3, n**4

v = square(2)
print(square(n))
```

⇒

```
def function():
    """ Hey """
help(function) # it tells what is happening in the function
```

⇒

```
def Square(n, y, debug)
```

Varietical

```
square(debug=True, y=1, n=2)
```

if you write name then

Day: Date: / /

~~Square / (int x)~~ // can only do small things like addition, multiplication?

=> f1 = lambda x : x**2

syntax => lambda argument : expression

we use this, we don't use list

def Square (x) :
    return x**2 ;

list

=> print (list(map (lambda x: x**2, range(4))))

" map (function, iterable)

=> Range ka square lae aur list bana dae

=> Classes :- def __init__(self

```
class Point :

    def __init__(self, x, y) :      # this
        self.x = x
        self.y = y
```

} member functions are always starts with

def __str__ (self) : //By default self

print

ostream wala kaam / print krnay ka liye operator overloading

};
cout << p
point p;

Point . P1
print P1

↓

```
def __str__(self)
    return f"{self.x}, {self.y}"
```

⇒ Modules :-

import types
dir.

import math

OR

→ from math import *
→ from math import pi;

eg.

→ math.pi
math.log.

⇒ File :-

```
files = open("1.one.txt", 'r')
lines = file1.read().splitlines()
print(lines)
lst1 = [
```

⇒ Exceptions :-.

```
if:
else:
    raise Exception(...)
```

```
try:
    #normal code.
except:
or
except Exception as e:
    print("Exception" + str(e))
```

→ Ags ma compiler khud bata dega error bata dega

y: Date: / /

⇒ Numpy:- → numpy array → famous data science library.

import time import numpy as np
import timeit ⇒

%timeit [i+1 for i in range(10000)]

%timeit np.arange(10000) + 1

↓

jaisy python ma range hy numpy ma arrange.

%timeit ⇒ for one line
%%timeit ⇒ for program

→ Creating numpy arrays:-

a = np.array([1,2,3,4,5,6])
print(np.array(a))

1D → (6,) ∴ 1D array hai jis mn 6 elements hai.
2D → (3,3) ∴ 2D array hai jis ma 3 rows aur 3 columns

→ h = np.arange

k = np.linspace(1,10,9) → means 10 ko 9 parts ma kro. 10/9 = 1.1

1. 1.1 2.1
2.1 1.1 3.2

→ import matplotlib.pyplot as plt.
import numpy as np.
y = np.ones(9) → rows.
y = np.zeros(9) // Created an array with 9 zeros
y = np.zeros((9,1)) → 2D double so, write in 2 brackets.
plt.plot(k,y,'o')
plt.ylim([0.5,1])

Day: Date: / /

→ y = np.loadtxt("file.txt", delimiter=",")
print(y)

↓
ignores , in the text.

for random // → np.empty(3,3) // We just have to make an array compiler khud jae initialize kr dega.

→ np.identity(4) // identity matrix, diagnols all 1s.

⇒ indexes:-

a = np.array([1,2,3,4,5])

a[0:5] # a[:5]

a[0,5] #

a = 1, 2, 3, 4, 5
False true false true false

b = a%2 == 0

⇒ Function:-

dtype= those are the bits.

Eg: a = np.array([1,2,3], dtype=np.int8)
print(a)
a.nbytes
a.size (3,3) // tells size (a)
a.type
a.shape (3,)
a.ndim (1) // pta karay ga kitni array hy, 1D ya 2D
a.dtype.

Date: / /

a.argmax() // returns index where the max value is.

a.argsort() // sort the indexes ~~by value~~ indexes ko

z = np.zeros(10, np.int32)

np.ones(10)

k = p.reshape(2,3)

k.ravel() → multidimensional ko 1D mai krdoo. (flatten krdoon)

np.sum()

np.

~~z = // Arithmetic~~

numpy

a.shape → (rows, columns)

Day: b= [1,2,3] → for array. Date: / /

⇒ a= np.array([1,2,3,4,5], np.int8), b= np.array([[1,2,3],[4,5,6]])

print (a+2) # 3,4,5

print (a-2) # -1, 0, 1, 2, 3

We want to compare.

Print (a==b) # True, True

Print (a>=b) #

⇒ Transpose.

b.T

⇒ Mathematical:-

b.max()

b.argmax()

np.mean(a)

np.std(a) ⇒ standard deviation.

Pandas:- numpy had limit. So, we use Pandas.

Data Series. (1D array)

Data frames (2D array)

⇒ import Pandas as pd.

df= pd.read_csv("data.csv")

↑ obj

Day: Date: / /

df = pd.Series(a)

df = pd.Dataframe(b)

↓

2D Array.

(1) Modules (no errors, bhr sa utha lo)

Built-In (pip sa install)

External (people wrote)

(2) Files:

r=read w=writing a=appending

'r' mode is default

f = open("name.txt", 'r') → Modes

l = f.read()

print(l)

f.close()

OR

with open("name.txt", 'r') as f:

f.read(),

// Now, we don't use close.

* With → say khud hee close.

// In case of append.

f = open("name.txt", 'a')

l = l.write("a")

OR l = f.write("Text")

Modes

(1) read (r)

(2) write (w)

(3) append (a)

(4) create (x)

if file phly ki tou error.

(5) text (t)

(6) binary (b)

f = open("name.txt", 'rt')

f = open("name.txt", "rb")

Exclusive Falcon

Day: error handling. Date: / /

(3) Exceptions:-

(1)
```
a= input ("Enter a number:")

try:
    for i in range (1,11):
        print (f"{a} x {i} = {int(a)*i}")
except Exception as e:
    print ("Invalid")

print("Haiba")
```

(2) try:
q#1

(4) Numpy:- efficient space gives.
khud memory handle karlenge.

=> Numpy Array vs lists.
(more faster)
we can work on only one data types (for big programs).

=> To check dimensions of arrays

```
a= np.array ([[1,2,3,4], [1,12,13,14]])
```
4 4
they should be same

Day: Date: / /

// While printing array, they print in lists

print(a.ndim) // prints the dimension

print(np.array([1,2,3,4], ndim = 10))

=> Creating dataframe from dictionary

✓ import panda as pd.

✓ ① data = { name : ["Ali", "Haleema", "Tashi"], ages : [1, 2, 3] }

② a) df = pd.DataFrame(data) or pd.series( )

b) df = pd.DataFrame(data, [a, b, c]) we can not put data b/c series is 1D and it is column representation

=> // extracting data from index

df.iloc[2]

=> // saving to csv format.

df.to_csv(".csv") (csv file created)

// reading the csv file.

df1 = pd.read_csv(".csv", index_col=1) (parsing first col to index)

Ais Sa na, phly 1st col first pay aye jae gy.

Exclusive Falcon

pd.DataFrame(np.random.randint(0,5, size= 10))

Day: Date: / /

// df.iloc [0] ['ages'].

// df= pd.DataFrame(np.random.randint(0,100 size=(100,3)))

movies = pd.read_csv("....csv"),

// movies.head(0) → displaying fixed number of rows.

if movies.head(1) it will display 1st row

// movies.iloc [0]

movies.

// movies.loc ['split']

movies.

// df.shape — tells about row, column

// movies.column — // how much columns.

// movies.rename(columns = {"Votes": "Vote"} , inplace = True)

Votes se name vote ho jae ga.

new dataset nai bante

Agr false ho jae too ho ga yani, pr peechay ek new file jo purani se votes wale wapis le gye

movies.rename(columns = {"Names":

it og used to change the columns names.

Day: df.info()
df.describe() // mean, median, mode, std, Date: / /

// for i in movies:
print(i.lower()) → Sb ko small kr dae ga.

// movies.columns = [i.lower() for i in movies]

// movies.isnull()

// movies.isnull().sum()

// movies.dropna() → Null values wali row hae nikal jae.

// movies.dropna(axis=1) → A'say column nikal jae ab jin ma null values

// Extracting data from columns.

2. re = movies['votes']

// m = re.mean()

// re.fillna(m, inplace=True)
↓
ab agr ali ki salary nhi hy
tu remove krne nhi hy.
Ari lia mean value nikal
wo aur aus ma values
daal dae ga.

Day: Date: / /

```
import matplotlib as mpl
import matplotlib.pyplot as plt
x = np.linspace(1,100)
x = np.linspace(1,100,100)
fig = plt.figure()
plt.plot(x, np.sin(x), '-')
                       'o'
fig.savefig("-png")
```

It is the range and it prints a graph with base of x-axis as 5-100

fig stores more.

'o' → we use coloring in scatter plot

// display a picture.

```
from IPython.display import image
image("-png")
```

I.P

capital

Simple line plots.

```
// %matplotlib inline
plt.style.use("")
// fig = plt.figure()
   ax = plt.axes()
// plt.plot(x, x+1, linestyle='solid')
                    or
                    :='-'
plt.plot(x, x+1, :-g')
```

solid

solid

Day:

Date: / /

plt.plot(x, np.sin(x))

// plt.xlim(-1, 11)
plt.ylim(-2, 11)

// plt.axis('tight')
('equal')

x, np.sin(x), label = "sin(x)")

=> // Labelling plots.

plt.plot(x, np.sin(x))
plt.title(" ")
plt.xlabel(" ")
plt.ylabel(" ")

} write these 3 things must

plt.legend( ) → top right corner by

=> " Scatter plot

2 ways

① • plt.plot(x, y, 'o', color = "black")

• rng = np.random

// plt.legend(numpoints=1, fontsize=13) → this say legend key symbol ka

// -p => for polygon.

Day: Co-variance => C.O.V => Do variables ko nikalnay kay lie   Date: / /

(2) plt.scatter(x, y, ...)

// Heat maps.

// plt.colorbar()    => side pay bar ban jae gi colors ki

// from sklearn.datasets import load_iris

=> Customizing - legends

Binning => It's on numerical data
↓
to catogarized numerical data.

Ali	17	(17-20)
Shan	23	(23-25)
Shah	21	(21-22)

(EDA) ~~Datasets~~ from sklearn.datasets import fetch_openml
boston = fetch_openml(data_id=42165)

Its not running version 1.2 sa pehly {
=> from sklearn import datasets
import warnings // is ko import krtay hain warnings atti, aun to ignore krnay kay liye
warnings.filterwarnings("ignore")
// n = datasets.load_boston()
}

n=boston

n. DESCR // tells details
type(n)

n.values(); n.keys() // last column is target
~~shape()~~ boston.target.shape. // only rows
boston.data.shape // only rows, columns

n.feature_names // target variable means result nai aye ga. (All columns names except last one).

boston_pd = pd.DataFrame(n.data)
type(boston_pd)
boston_pd.head() // By default 5 print rows.
boston_pd.tail() // Last se.
n.data.shape

// Sklearn ma target nai atta phr alag include kry latay hain

boston_pd['MEDV'] = n.target.
n = dataset.load_iris()

iris_pd = pd.DataFrame() Same as above one.

Day: IDS (continued) Date: / /

⇒ Exploratory Data Analysis (EDA) (mostly graphical)

Def:- ⇒ Data exploration technique to understand data.
⇒ Approach of data analysis that employs a variety of techniques.

⇒ Why Data Exploration?

get data ⇒ Exploratory data analysis ⇒ Preprocessing ⇒ Data mining

motivations:-
⇒ Select right tool for preprocessing, data analysis and data mining.

⇒ Aim of EDA?
① Extract important variables
② Develop valid model.

hypothesis ⇓ Proposed explanation on basis of limited evidence

⇒ EDA	CDA
• No hypothesis at first.	Start with hypothesis
• Generate hypothesis	Test null hypothesis
• Uses graphical methods	Uses statistical models

⇒ Steps of EDA
i) Generate good research questions
ii) Data restructuring - instead of using 2 variables, make %ages of them.
creating ⇒ dummy variables for categorical variables.
iii) Based on research questions, obtain descriptive statistics.
iv) Handle missing observations
v) Decide on need of transformation
vi) Decide on hypothesis based on researched

Exclusive Falcon

Day: NLP ki library. Date: / /

⇒ After EDA:-

(i) Confirmatory Data Analysis
(ii) Get conclusions and present results nicely.

⇒ Classification of EDA:-

Graphical / non-graphical

Summarize data in diagram

Calculation of summary statistics

univarient / multivarient

(not bivarient)

one data column (How it varies?)

more variables (before performing multivariant only univarient not all too)

Data Profiling = Data calculation to Examine.

Data.

Categorical (Categories) (Qualitative)

Nominal (order not important) / Unordered Data

ordinal (order matters) (Ascending or descending) level of education Ordered Rating.

Numerical

Discrete (counted data)

Continuous measured data

Day: For text ↓ Date: / /

Introduction to Natural Language Processing.

-- Human-to-computer language

```
import nltk.
s = nltk.sent_tokenize(text)
for i in s:
    print(s)
for i in s:
    w = nltk.word_tokenize(i)
    print(w)
```

To write big para. Use """ 3 Comma at start. and " one at the end.

Text stemming and lemitization (Its more accurate)

Stemming: Change, Changing, Changes → Chang

Lemitization: Change, Changing, Changes → change.

Stop words :- → English me jaisay (.)

in nlp it is used as dataset. (Text dataset ⇒ Corpus)

```
from nltk.corpus import stopwords
s = stopwords.words("english")
```

Variables

Catagorical (Bar chart)

Continuous (Histogram)

Univariate

```
import seaborn as sns.

=> iris_pd.[illegible].mean
iris_pd.median
pd.DataFrame([mean, mean], index=["Mean","median"])

i = iris_pd.mode(axis=1)

// if I want to change the type
```

it runs on a full dataframe.

```
i.astype("category") or ("int8")
i.describe(Number summary)
i.info
a = i.cov()
i.corr()
sns.heatmap(i.corr(), annot=True, cmap="Set3")

plt.figure(figsize=(5,5))
sns.boxplot(=).
```

Values comes inside box

```
sns.boxplot(,
            orient="h")
// i.cov          // ↓
// i.corr("Pearson"   // += Size
// i.corr("spearman")  // color
                       (palette="Set1",
//
```

Day:

Date: / /

- boston_pd.skew()

n = np.log(boston_pd.skew()) (log transformation)

sns.distplot(n)

n = np.sqrt(boston_pd.skew()) (square root transformation)

sns.distplot(n)

import statistics

n = statistics.variance(iris_pd)

iris_pd.var() # for every row variance

Intro to Data Science (IDS)

// import nltk
nltk.download()
// from nltk.tokenize import sent_tokenize, word_tokenize

aText = " . . . "
print (sent_tokenize(Text)) // word tokens ⇓ list of words.

// from nltk.text import Text
Text (aText) // Ab tokens ko text mai convert.
// t.count (" ; ; ") → If you want to count the words.

// t.vocab () // text kay sath ye batae ga kay kitni baar repeat.
↓ text kay according dictionary bana dae ga.

// from nltk.corpus import stopwords // dataset in nltk is corpus.
set (stopwords.words ('English'))
↓ ye unique words ko le kr aye ga.

// filtered_words = []
for i in word_tokens:
if i not in stop_words:
filtered_sentence.append(i)

// stop.words.fileids () // languages in stop words.

// stopwords.raw ('arabic'). replace ('\n', ' ')
↓ it is print karna ← arabic language kay stop words.

(1) // tokenize
(2) // lamitization
(3) // stopwords.

// from nltk.stem import PorterStemmer
// tokenize word:

// Ais say orignal word aye gy. ing, ed sb Khatam.

ps = PorterStemmer()
e = ['python', 'pythoned']
for w in e:
    print(ps.stem(w))

// from nltk, WordNetLemmatizer

// ///
for .....
    print(ps.WordNetLemmatizer())

// Stemmer 3 types.
from nltk.stem.snowball import SnowBall — same with
// .lancaster // lancaster Stemmer
// .porter // PorterStemmer
used porter.

Porter improvement → Snowball

// import string

punctuation string ma hai aya jatay hain ai e.g nltk vle ma kro.

p = string.punctuation
p.

// First Limitization than stopwords
// yield => It just save the value jo agy ho gae
used in for loops.

// from textblob import TextBlob //spellings to theek krnay kay liye.

```
t = "...."
tr = TextBlob (t)
tc = tr.correct()
```

// Ngrams.

```
text = "...."
import nltk
nltk.download('punkt')
```

// ya khud sentence ko correct. puntuation lga dee ge.

```
m = nltk.FreqDist(text)
```

(uni ma tokens hae ban rahe, faida hae nhi ais ka) Freq Dist ⇓ frequency distribution

// unigrams / bigrams / trigram
// predictions llay lie. Two ka group bana dee ge.

```
unigrams = nltk.ngrams(tokens,1)
uf = nltk.FreqDist(unigrams)
for token in list(uf.items()):
```

// Time Series (jis date mention hti hy, string ma)

//

```
df_hbl['Date'] = pd.to_datetime(df_hbl['Date'])
df_hbl.index
```

// ya start, stop aur step ka batee gai

//Ais ya jo date excel ma hy wo string ma, hum ne int me time lena.

Bgr → RGB → ya hota hai

Red green blue. ↖ RGB ⇒ 3 images hoti ais ma. (3 dimensions)
(Ais ma sa hum 1 image nikal letay hain. Phr ais ma kaam krna easy)

⇒ OpenCV

```
img = cv2.imread(path) // read the image.

img = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)

fig, ax = plt.subplots(1, 2, figsize=(10,10))
```

(1 row, 2 images / 2 colums)

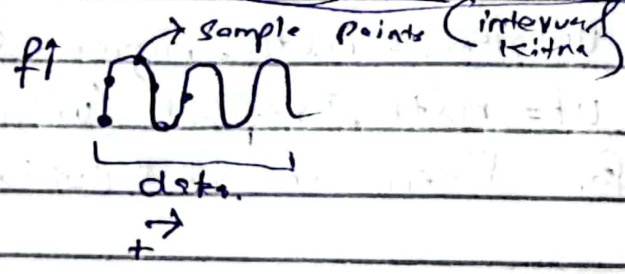
```
ax[0].imshow(img) // orignal image wo phly show karayga.

ax[1].imshow(gray_img) // uper gray image ko 2nd number pr dikhae ga.
```

// RGB ma split krnay ka lia. 3 variables needed.

```
a, b, c = cv2.split(img)
```

---

import librosa //



⇒ Spectrogram → visual representation of audio.

⇒ time = data / sample rate.

⇒ fill_between #

[a, b, c] = plt. magnitude

↓ we had used this b/c

•- Edge detection → Search pixcels where there is sharp change.
→ Sobel and Prewitt filters.
• → Perwitt

Image Processing :-

• - Sobel

```
q = perwitt_h(img)
    "   _v   "
b = filters.sobel(img)
```

• - Canny → highest accuracy

```
feature.canny(img)
```

```
import cv2
img = cv2.imread("profile.jpg")
grey_img = cv2.cvtColor(img, COLOR_BGR2GRAY)
fig, ax = plt.subplots(1,2, figsize=(10,4))
ax[0].imshow(img)
ax[1].imshow(gray_img)
// Split image in colors
b,g,r = cv2.split(img)
fig, ax = plt.subplots(1,3, figsize=(10,5))

ax[0].imshow(b, cmap="Blues")
ax[1].imshow(g, cmap="Greens")
ax[2].imshow(r, cmap="Reds")
ax[0].axis("off")
// Blur images
g_blur = cv2.GaussianBlur(img, (19,19), 15)
```

(19,19): kernal size (larger size, more blurry)

15: std

blur in width.

```
m_blur = cv2.medianBlur(img, 9)
```

9: blur value

```
fig, ax = plt.subplots(1,2, figsize=(10,4))
ax[0].imshow(g_blur)
ax[1].imshow(m_blur)
```

Image Rotation

```
angle = 180
scale = 1          // scale means zoom
h, w = img.shape[:2]
centre = (w//2, h//2)   // center of rotation (It is important)

rot_matrix = cv2.getRotationMatrix2D(centre, angle, scale)
rot_img = cv2.warpAffine(img, rot_matrix, (w,h))
```

Image Resizing

```
newwidth = 500
newheight = 500
h, w = img.shape[:2]
resized_img = cv2.resize(img, (newwidth, newheight))
```

Image flip

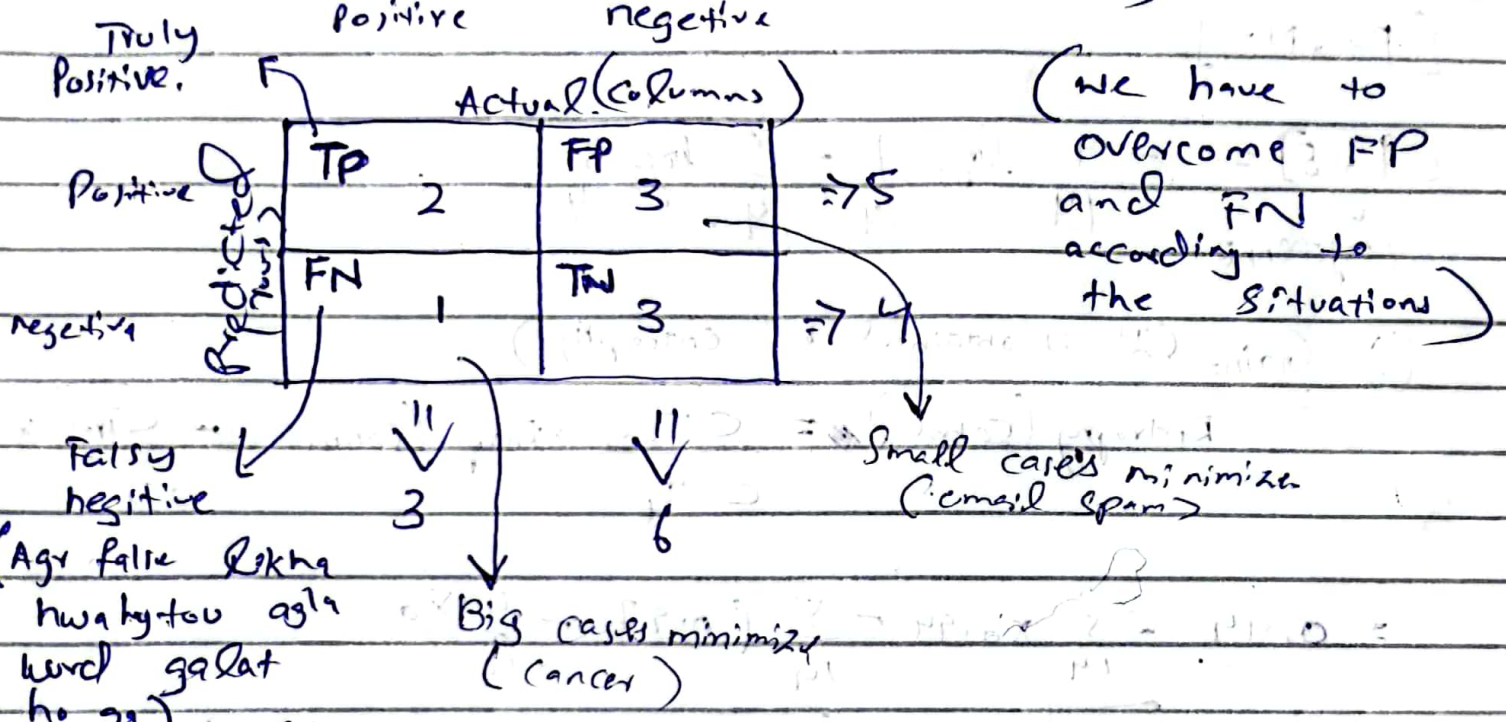
```
flip_img = cv2.flip(img, 1)
```

Data imbalancing ⇒ Red Block] Jo mam ne samjhaya tha

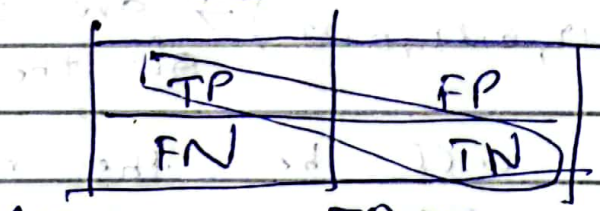
Performance metrices

(1) Data Preparation → feature engineering (features nikaltay hain) → data modeling → Performance measure.

(2) Confusion matrix (used in classification) (like ek banday ko cancer machine yes. diya ya cancer mi, No, lekin agr app usko ho jae. yes ko No aur No ko yes. Ab aj table ko confusion matrix)



Accuracy



(Accuracy achi nahi hy, data imbalancing)

Accuracy = $\frac{TP+TN}{TP+FP+FN+TN}$.

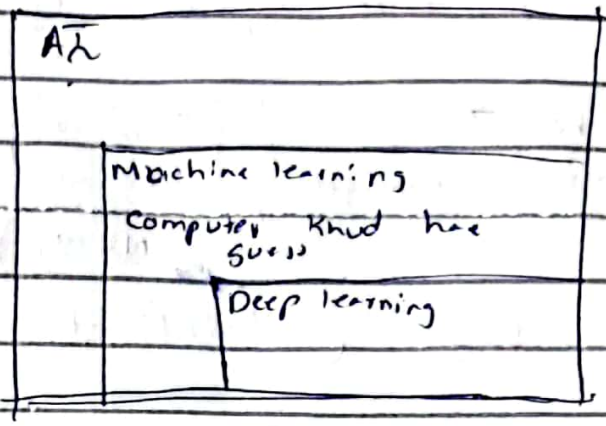
Precision:

$\frac{TP}{TP+FP}$

Session 1-2.

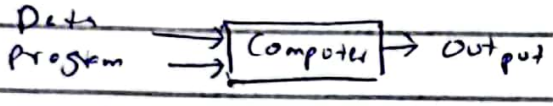
1 Machine learning

"Field of study that gives computer the ability to learn without being explicitly programmed"

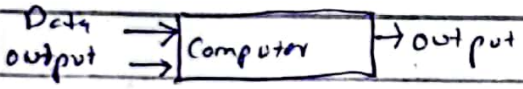


a) Start with data → E (experience)
b) perform some sort of task → T (task) (Task is called supervised learning, classification or inductive learning)
c) Validation test → Performance measure P

Traditional



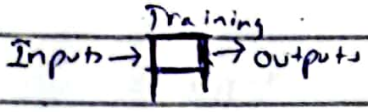
ML



Test complexity than US b/c output is known

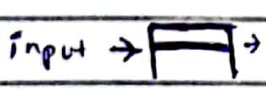
ML

Supervised (• Classification • Regression → numerical)
- learn explicitly
- Output clear
- Predits future



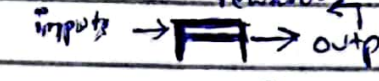
- Medical Diagnosis
- labeled data
- 2 steps - learn (training)
  - Testing
- Decision trees
- KNN

Unsupervised (• detection of anomalies • Association • clustering • Dimensionality Reduction)
- machine learn
- do not predict



- Fast Students new
- un labled data.
- K-means clustering

Reinforcement
- Approach to AI
- learning from +ve or -ve reinforcement
- Maximize rewards



- Reward based learning
- netflix, youtube recommendations.

Machine learning Process

gather data → clean data for homogeneity → Model building (select right ML algo) → Gaining insights → Data visualization

Accuracy:- (Classification problem)

$$\text{Accuracy} = \frac{\text{No. of correct classification}}{\text{Total no of test cases}}$$

Classification:- (2 things)

- We predict y labels (classes) for input n.
- Yes or No (fraud or not fraud)
- $y = f(n)$ → input
  - output
  - Classification function

Regression:- Numerical data (1 thing)

Supervised

- feedback
- Used for prediction
- Known number of classes
- less complex than →

Examples

- Text recognition (training no right)
- Spam detection
- Face detection

Unsupervised

no feedback
Used for analysis
Unknown number of classes

Examples

Noise removal from datasets!

Decision trees (for classification & regression)

Classification:-

2 steps
- learning step
- prediction step.

Decision tree algorithm:

- Topmost node → root
- Internal node → feature
- Branch → decision rule.
- Outcome → leaf
- It learns to partition on attribute values
- It partitions the tree in recursive manner called recursive partitioning

Entropy
- measures impurity of input set

$$-\frac{c(p)}{c(T)} \times \log_2 \frac{c(p)}{c(T)} - \frac{c(N)}{c(T)} \times \log_2 \frac{c(N)}{c(T)}$$

$$-\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \times \log_2 \frac{5}{14}$$

Gain:-

$$\text{Entropy (label)} = \frac{CT_{sunny}}{CT_{total}} \times \text{Entropy}_{sunny}$$

DTA

- Attribute selection Measures (ASM).
  - also known as
  - Splitting rules.

- Different Matric :-
  - Confusion Matric
  - Accuracy
  - Precision
  - Auc (Area under the curve)
  - MAPE (Mean absute % error)
  - MAE (Mean absute error)
  - MSE (Mean squared error)

	Actual	
Predicted	TP	FP
	FN	TN

- Accuracy :- $\frac{TP + TN}{Total}$ (TP, TN)
- Precision :- $\frac{TP}{TP + FP}$ (TP, FP) (How many did we match)
- Recall or Sensitivity :- $\frac{TP}{TP + FN}$ (TP, FN) (How many we miss)
  - If every person has cancer, recall is 100%.
  - If focus on minimize FN, Recall → 100% without precision too bad
  - If on " FP, Precision → 100%.

- **Specificity :-** (Opposite to recall)

$$\frac{TN}{TN+FP}$$

(FP / TN)

- **F1 Score :-**

Single score that represent Precision and recall.

F1 Score = Harmonic Mean (Precision, recall)

$$\text{F1 Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **AUC :-** (Scale - invariant) (0 - 1 range) (If 100% correct AUC 1 hoti hai)

a) True - Positive rate (TPR)

$$TPR = \frac{TP}{TP+FN}$$

b) False - Positive rate (FPR)

$$FPR = \frac{FP}{FP+TN}$$

- **MAE :-** (Avg diff b/w orignal and predicted values)

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

- **MSE :-**

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

- **MAPE :-**

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

(X) → Independent / predictor / explanatory

Y = Dependent / response variable.

Date / /

Linear Regression (For prediction or forecasting)

- Dependence of one variable on one or more variables (called independent variables).

⇒ Primary used to input to a system

- Dependent & independent variables kay relationship ka estimate.
- Effect of each explanatory variables on dependent variables.
- Predict value of dependent variable for given ind variable.

⇒
- Dependent are those which are changed due to indep.
- least square linear regression ⇒ method for determining Y on basis of X.

⇒ **Assumptions of Linear Regression Model.**

- Linear Functional form
- Fixed independent Variables
- Equality of varience of the errors.
- No multicollinearity
- No outlier distortion
- No autocorrelation of the errors.

⇒ **Linear Regression Model.**

First order linear model

$$Y = b_0 + b_1 X + \varepsilon$$

b_0 = Y-intercept

b_1 = Slope of the line

ε = error Variable.

slope. ↑ Y-intercept.

$y = mx + b$

predict

- Independent change, 1 dependent change
- 2 Independent change, 1 dependent change.

Date / /

- If there is only one driver variable, X, it is simple linear regression.
- Model involves multiple driver variables called multiple linear regression.

- If Relationship b/w X and Y is curvilinear, the regression line will be be a curved line

- Greater strength of relationship b/w X and Y better is prediction.

⇒ **Least Squares Estimation of b_0, b_1**

Least Square estimates of slope co-efficient b_1 of true regression line

① $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$ → β_0 = Mean response when x=0 (is)

② $SSE = \sum_{i=1}^{n} \left(y_i - \left(\hat{\beta}_0 + \hat{\beta}_1 x_i\right)\right)^2$

③ $\beta_1 = \dfrac{\sum \left(x_i - x'\right)\left(y_i - y'\right)}{\sum \left(x_i - x'\right)^2}$ β_1 = change in mean response when x increases by 1 unit.

$\beta_0 = y' - \beta_1 x'$

- β_0, β_1 are unknown parameters.

⇒
- Regression generates least squares regression line.
- Squaring difference and adding up squared differences across all predictions is called residual / error sums / squares / SSerror.
- Regression generates formula such that SSerror is as small as it can possibly be.
- Minimizing this number minimizes average error.

Date / /

Regression in Python

1. from sklearn.linear_model import LinearRegression
2. Do transformation. Use reshape(), set x, y
3. • model = LinearRegression().fit(x,y)

• fit_intercept ⇒ Boolean, if true decides to calculate intercept bo.
(By default true)
• if false consider it equal to zero.

• normalize ⇒ Boolean, if true decides to normalize the input variables
(By default false)
↓
It doesn't normalize input variables.

model.intercept_ (bo)
model.coef_ (b1)

4. Check results of model fitting
• Obtain co-efficient of determination, R^2 with .score().
r1 = model.score(x,y)
pre = model.predict(x)

Applications

Economic growth
Product sale
Housing Sales
Score prediction.

If $K = n$ (no. of obs), then distance = 0 (case of overfitting)

K-mean Clustering

Date / /

	x	y	
1	1.0	1.0	→ Mean 1
2	1.5	2.0	
3	3.0	4.0	
4	5.0	7.0	→ Mean 2
5	3.5	5.0	
6	4.5	5.0	
7	3.5	4.5	

Advantages:-
- Easy to represent
- Can work in multiple dimension

Diradvantages:-
- Time consuming to find optimal number of clusters.
- Time consuming feature engineering

Iteration 1:-

⇒ For point 1:- $x_1 = 1.0$, $y_1 = 1.0$

$$D_1 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$
$$= 0$$
$$D_2 = \sqrt{(5-1)^2 + (7-1)^2}$$
$$= \sqrt{(4)^2 + (6)^2}$$
$$= \sqrt{16+36}$$
$$= \sqrt{52} = 6.10$$

⇒ For Point 2:- $x_1 = 1.5$, $y_1 = 2.0$

$$D_1 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$
$$D_1 = \sqrt{(1-1.5)^2 + (-2)^2}$$
$$= \sqrt{(-0.5)^2 + (1)^2}$$
$$= 1.12$$
$$D_2 = \sqrt{(5-1.5)^2 + (7-2)^2}$$
$$= 6.10.$$

Point	D_1	D_2	which cluster to choose.
1 (1.0, 1.0)	0	6.10	1
2 (1.5, 2.0)	1.12	6.10	1

Date / /

Iteration 2:

For Mean 1:- $\frac{1.0+1.5+3}{3}, \frac{1.0+2.5+4.0}{3}$ (Points in cluster 2)

$(1.8, 2.33)$

$m_2 \quad y_2$

For Mean 2:- $(4.12, 5.38)$

$m_2 \quad y_2$