⇒ How often we see 2 relevant fields.

- Better Output ✓
  - Kaylee / EMA :- To review the output ✓

⇒ Categories Can't change

{Refine the model} → provide → Stratified data.

a) incident description. (Initial Breach assessment)

+

b) ? (Bring more data) {We have to look at that data.}

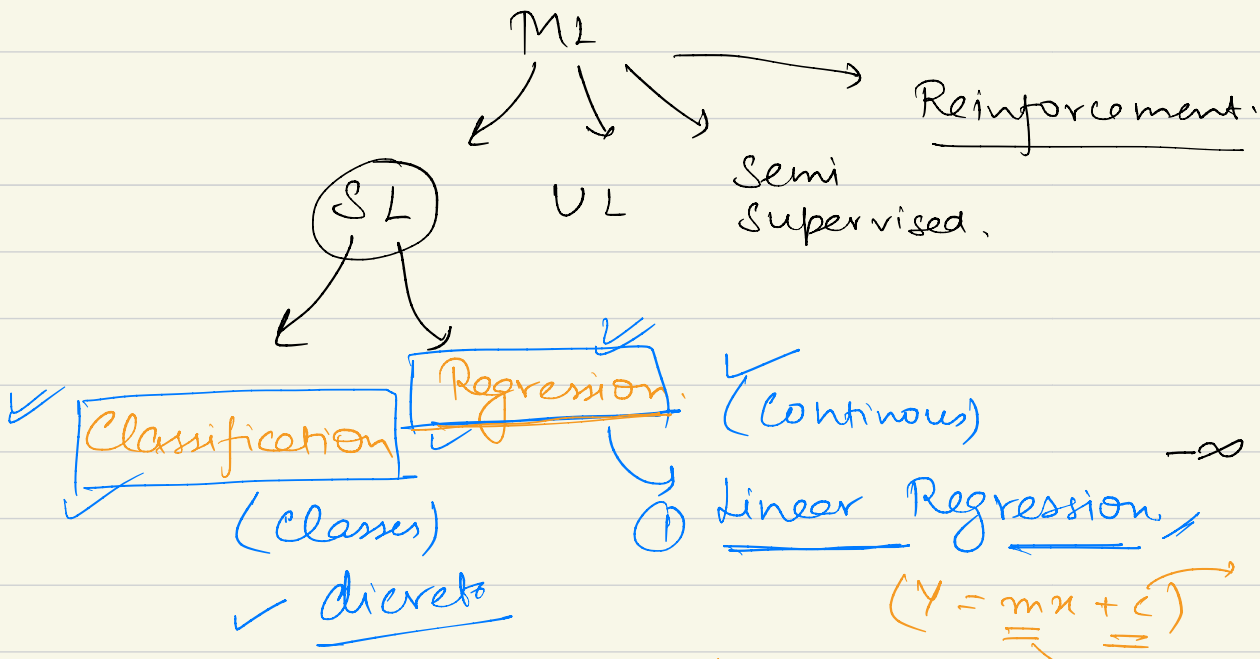
(Commentary)

⇒

Next steps :- {Breach Assessment}^{R} BRA → description

1. Detailed Breach Commentary → Commentary.
2. Running that on Model
3. Showcase.

In IRL : Which field

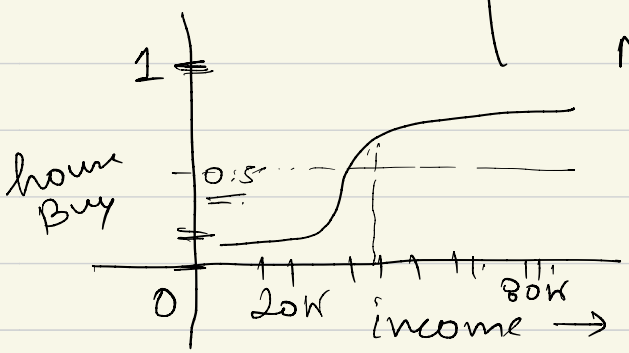


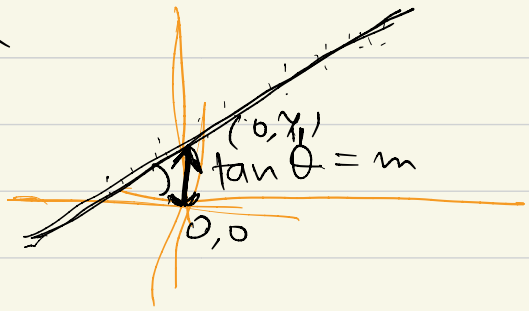
① Linear Regression — $-\infty$ to $+\infty$

$(Y = mx + c)$ — m: slope, c: y-intercept

② Logistic Regression :- (Classification)

Income ↑ ⇒ { Buy a house / or / Not } ↑





Mathematics :-

probability :- $\frac{3}{10} = 0.3$ (3 → rain, 10 → moving out)

② log odds :- $\log\left(\frac{\text{prob(event)}}{1 - \text{prob(event)}}\right)$ — Chances.

$= \log\left(\frac{0.3}{1-0.3}\right)$ (0.3 → raining, 1−0.3 → not raining)

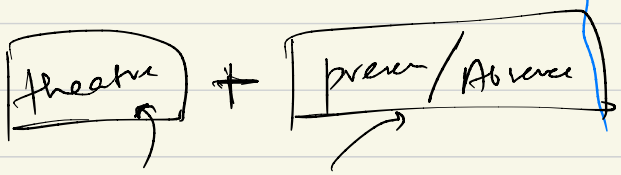
$= \log(3/7)$.

(3) Odds Ratio :-

ratio of Event happening in the presence of some variable vs Event happening in the absence of that variable.

10 → Raining.
↓ theatre.
3 times.

10 → Non-Raining.
↓ theatre
7 times.



$(p/1-p)$.

Odds ratio (Non-Raining) $= \frac{7/10-7}{3/10-3}$

$= \frac{7/3}{3/7} = \frac{7}{3} \times \frac{7}{3}$

$= 49/9 = 5.4$

* { prob. of ∧ Non-raining. is 5.4 times more that when it is raining. going out of theatre when }

$9/49 = 0.18$

" Equation of Logistic Regression "

$\tilde{y} = mx + c$ ... (1) Linear Regression

step 1

$$\log\left(\frac{p}{1-p}\right) = mx + c$$

$$p = \frac{1}{1 + e^{-(mx+c)}}$$

$-\infty$ to $+\infty$

linear regression

final step

Logistic regression

0 − 1

$$p = \frac{1}{1 + e^{-f(x)}} = \frac{1}{1 + e^{-(m_1x_1 + m_2x_2 \ldots + c)}}$$

$$p = \frac{e^{mx+c}}{1 + e^{mx+c}}$$

$$= \frac{1}{(1 + e^{mx+c})/e^{mx+c}} = \frac{1}{1/e^{mx+c} + 1}$$

$$= \frac{1}{e^{-(mx+c)} + 1}$$

Evaluation of Logistic Regression

Doctor

① Ground truth (Labels) Actual. [0,1]

↕ Comparision..

② Model Output: [0,1]

4 combination

given :- Binary Classification

↙ event (1) ↘ event (0)

Confusion Matrix :-

pred \ Actual	1	0
1	TP	FP
0	FN	TN

{ 1 → Diseased
0 → Healthy }

Type I Error

When Null Hypothesis (H_0) is mistakenly rejected.. (By Model)

Type II Error.

When [Null Hypothesis that is actually false] → diseased.

is Accepted (By Model).

Class Imbalance

X	y	pred$_y$ (0, 1)
10	0	0 ✓
20	1	0 ✓
30	0	0 ✓
40	0	0 ✓
15	0	0 ✓
18	0	0 ✓

→ rule.

$$\text{Accuracy} = \frac{TP + TN}{All} = \frac{0 + 5}{6} = \frac{5}{6} \Rightarrow 80\%$$

$$\text{Recall} = \frac{0}{0 + 1} = 0$$

$$\text{Precision} = \frac{0}{0 + 0} = \text{undefined}$$

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

$$\text{Recall} = \frac{TP}{TP + FN} \qquad \text{Precision} = \frac{TP}{TP + FP}$$

{ Out of all the actual +ve values, How many our model gave correct. +ve }

{ Out of all the +ve values which our model gave, How many were actually +ve by Expert }

$$\Rightarrow \left\{ F_1 \text{ score} = 2 \cdot \frac{\text{precision} \cdot \text{Recall}}{\text{precision} + \text{Recall}} \right\}$$

$$2\left(\frac{1}{P} + \frac{1}{R}\right)$$

$\Rightarrow$ (stable enough)

98 people Healthy
2 people not healthy =

⇒ (Rajat Sinha)

User specific chatbot

Stakeholder :-

① Regional HR lead :-

All the yr end performance Bonus.

Ensure :

⇒ Chatbot

⇒ Managers :-

- Ratings (1-5) ≤
- top-p
- Problems

⇒ NLP :-

" Subheading "

I.S

Hypothesis Testing

→ Assumption / Intermediate

Steps :-

1. Frame the Hypothesis.
2. Collect the data.
3. Choose specific margin of Error.
4. Conduct the Experiment
5. Analyse the Data using Statistical testing.
6. Communicate the Results

H_0 : Null Hypothesis (Statement of no change)
H_a : Alternate Hypothesis (Statement of change)

→ Day to day activity → New change.
(trying to prove)

[Water / Milk]

eg. H_a : DA for Mechanical | Innocent
H_0 : DA jobs for CS. | Guilty.

{ 100 : Water
100 : Milk } 3 months.
Skin getting better...

In order to introduce a drug or Cosmetic ...

= [In lines of probability]

Margin of Error (α) : 0.05

C.I (Confidence Interval) = 0.95

CLT → Avg. (Mean) sample = Avg (Mean) population

100 → 95% +ve result towards drinking milk.
Hence; We go towards (Alternate Hypothesis)

p-value (probal) < 0.05 ; We accept H_a
> 0.05 ; We accept H_0

⇒ If sample size < 30 ; We use the t-test ;

$$t = \frac{\bar{x} - \mu}{s/\sqrt{n}}$$

$\bar{x}$ → mean (sample)
μ → overall mean
n → samples

t > C.V
H_0 rejected

⇒ For 2-tailed Hypothesis testing :-

$$Z = \frac{\bar{x} - \mu}{\sigma/\sqrt{n}}$$

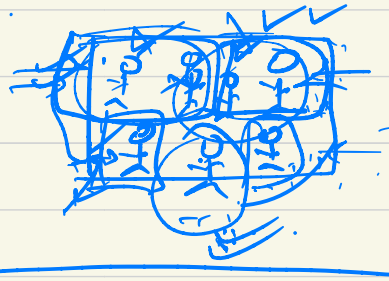
Z > C.V
H_0 rejected.

- Last Visualisation.. :-
- *(Hypothesis + Assignment)

Table

Y	X	Test.
Continous.	Discrete >2 categories.	Annova. (single factor)
Continous.	Discrete in 2 Categories	t-test.
Continous.	Continous.	Regression
Discrete.	Discrete. in 2 categories.	2-proportion Test
Discrete.	Discrete > 2Categories.	chi-square test

large no.
↓
from 1 platform
↳ sample



- Random Sampling
- Sequential

- Stratified sampling
  ↓
  subgroups :- strata.

- Cluster sampling.
- Convenience sampling.

:- Central limit theorem.
↓
Multiple platforms
↳ Σ (mean)
⇓
Overall pop (mean)

Coding - Python

Adv :-

i) Automation.
ii) Libraries ↑ Huge community ↑
iii) Simple.
iv) ~~large dataset.~~ → slow (wrt. C, C++, Java) inclined towards HLL.
v) Dynamic
vi) Debug Errors
vii) Data Analytics task.
viii) OOPS.
ix) Full stack. → complete language.
x) Interpreted language.

Pallindrome :-

NAYAN ; MALAYALAM.

Logic :-

string1 = "NAYAN"

Logic ?

```
if —
    print(" P... ")
else :
    print (" -- -- ")
```

```
str1 = "MALAYALAM"
str2 = str1[::-1]
if str1 == str2:
    print ("yes")
```

iii) [8, 4, 9, 14, 21, 6, 5]

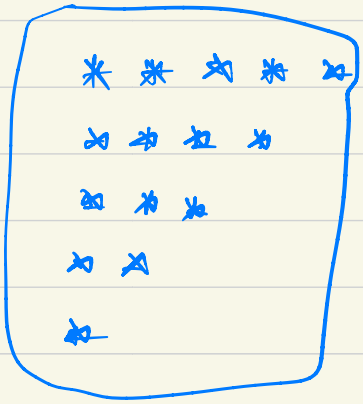
↳ input a no :-

And get all the nos. that are divisible by that no from the list.

i) pattern matching :-

```
5
5 4
5 4 3
5 4 3 2
5 4 3 2 1
```

5 min



5 min

ii) Creating a game of 2 players.

(A) 5, 1, 2, 3, 4, 5, 6, 5, 6, 2,

Chance of playing a dice having nos. b/w 1 – 9

(B) 9, 5, 8, 7, 6, 5, 4, 5, 4, 9

B lose

Till the time the sum total exceeds 100, the last player who played that chance looses.

iv) Magic no. ?

v) Armstrong no. ?

} Google

vi)

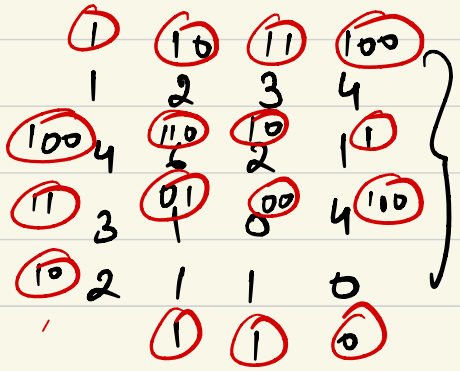
vii) $y = x^4 + bx^2 + cx^{1/2} + d$

1.4 1.1

$x \rightarrow$ input

$y \rightarrow$ output.

viii)



multiply this with Binary representation

$[4 \times 4]$

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

Output → {1:2, 2:1, 3:2, 4:1, 5:1, 6:2},

x) [4, 3, 2, 1, 5, 9, 13, 18, 6]

9 third largest no..

0 : A
25 : Z

str → E

AGILOFT

Contract Management System

- Contracts
- individual data

→ .... ?

"30 min".

- product
- metadata structured ✓

- Upload Contracts. :- 200-300
  - ↳ label

find
- NDA clause.
-

- redlining → language translation problem.
- summarising a Contract
  - risk

Python - Introduction.

History :-

→ Guido Van Rossum.

{ 1970's ⇒ BBC comedy series.
- Monty Python flying circus. }

Advantages :-

- Simple & short.
- Vast Application. → Complete language → front End / DA/DS / Back End.
- Lots of libraries | Community.
- Open-source.

- Dynamic Typing. → As we type, it interprets.
- Automatic Memory Management;
- Cross-platform support;;

- High level Data Types.
  - i) Lists.
  - ii) Tuples.
  - iii) Dictionaries.
  - iv) Sets.

For Swap :-

a = 2 , b = 6 ✓

output ⇒ a = 6 , b = 2.

other languages

int

```
a = 2
b = 6
temp = b
b = a
a = temp.
```

Multiple assignments :-

```
a, b = 2, 6
b, a = a, b
```

} integer types. (Automatically)

Palindrome :-

eg.; MALAYALAM.

0 → n-1 ←

prev.

```
str1 = _____
  ↳ length.
for: 0 - (len-1)
    if str1[0] == str1[len-1]
        ↓
      count
      ↳
    count = count + 1
    len = len - 1
```

```
str1[::1] == str1[::-1]
```

Disadvantage :-

'Python' is a higher level language. w.r.t others.

Code :-

High level language

↕ Interpreter.

Machine Code (Low-level language)

Java :-

System.out.Println( ) → Developer

↕ Interpreter → 01010101 (Machine Code / Binary Code)

MCode → Machine

Python :-

Print ( ) → Developer

↓

01010101 → Machine.

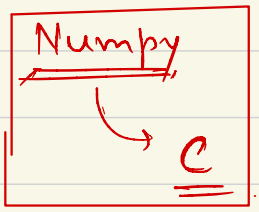
import Print type.

Devopers Ease ↑ Machines complexity ↑

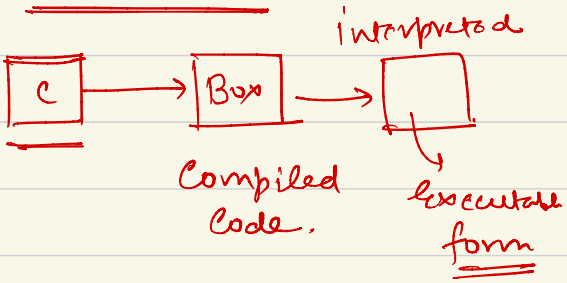
- Slow language.

Flavors of Python :-

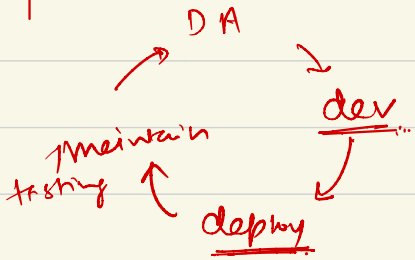
i) C Extension : Cython
ii) Java Extension : Jython.
iii) .Net Extension : IronPython.



Abstraction :-

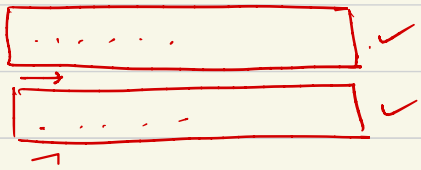


- Anaconda
- Spyder.



Notebooks

- They have cells.



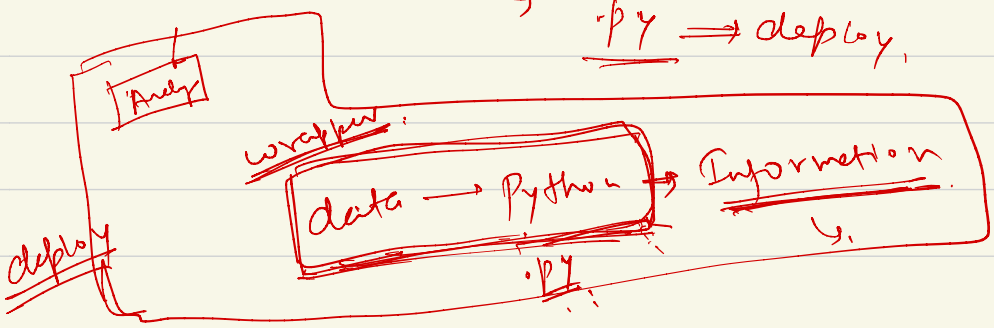
- Its handy for 'Dev' work.
- Extension : .ipynb

IDE

- One frame code Architecture

- output.
- Its handy for 'Deployment' work.
- Extension : .py

.ipynb → .py → deploy.



- Leetcode
- Hackerrank