PyMC3 :-

1. Probability → ~~Area~~ integration of the curve

Likelihood → Product of values $\left(e^{\frac{x-\mu}{2\sigma^2}}\right)$ across entire distribution for a set of parameters.

→ Then clicks = $(x < 0.5) \times 1$.

⇒ x = np.random.rand(1000) → will provide us set of ~~prob~~ values.

↳ We can postulate a distribution for this data.

↳ We can also enumerate our confidence

⇒ We can repeat this experiment → Find μ, σ for mean values.

↳ CLT

μ of mean ≈ μ of the distribution

↳ statistic

⇒ Click = $(x < 0.7) \times 1$

↳ This is a biased experiment as Area of x is bounded between 0, 1 and uniformly distributed. But we set the partition of two experiments at 0.7

0 0.7 1

BOOTSTRAP → proxy of probability

x_b = np.random.choice(df[col], n_samples, replace = True).

$\Rightarrow \dfrac{(x\_b > 0.7) \times 1}{n\_samples}$

⇒ Probability distribution for a coin, toss

↳ Has two outcomes ∴ can be stated in terms of binomial distribution.

If fair (Null hypothesis) → $P(H) = \frac{1}{2}$, $P(T) = \frac{1}{2}$

If experimented n times ∴ probability distribution of H →

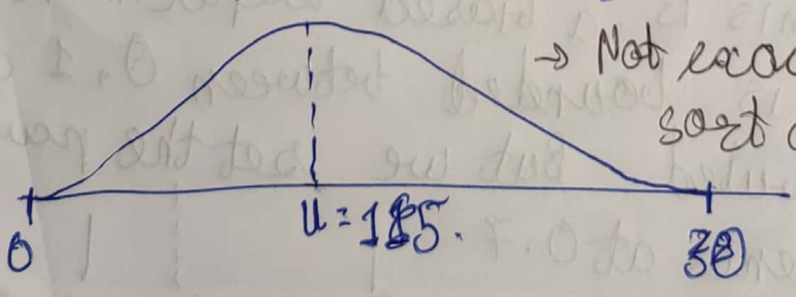
$^{n}C_{r} \, (H)^{r} \, (T)^{1-r}$ r → number of times we obtain heads.

⇒ Expected mean of binomial dist = $n \times p$

std dev = $p(1-p)$

If $n = 30$; $\hat{u} = 0.5 \times 30$ (# of heads)

→ Not exactly normal but sort of.



⇒ For $\overset{30}{\cancel{22}}$ tosses if we get 22 heads, there is a probability that this coin is fair

↳ Meaning $\mu = 0.5 \times 30 = 15$

↳ But it may come from some unfair coin / distribution too.

$P(x = 22 | 30)$ for a fair dist → $P(x \geq 22 | 30)$ → Extreme + give case

↳ $0.8\% = 0.008 \Leftarrow {}^{30}C_{22} \left(\frac{1}{2}\right)^{22} \left(\frac{1}{2}\right)^{8}$

↳ less than assumed 0.05 α

∴ Unfair coin chances ↑

⇒ Bootstrap helps avoid this calculation

```
m = 0
for _ in range(10000):
    trials = randint(2, size=30)
        ↳ Uniform dist for 0,1.
    m += (trials.sum() ≥ 22) × 1

p = m/100000
```

npr.binomial (n, p) → returns number of success ~~rand~~ in n trials

$^nC_x \, p^x (1-p)^{n-x}$ ↳ This is from a distribution

[npr.binomial(n, p) for _ in range(100)]

= npr.binomial(n, p, 100)

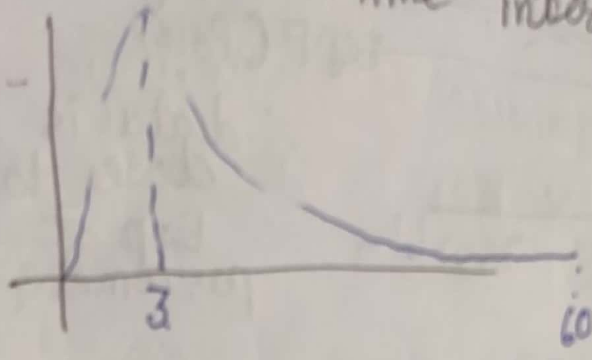
npr.binomial (n, p) → gives a statistic from the distribution

∴ npr.binomial (n, p, 1000)
↳ is normally distributed about $\hat{u} = np$ of the parent dist

ECDF :- Cummulative distribution of x
↳ x → sort x
Then y → np.linspace (0 + (1/n), 1, n)
↳ Each element thus gives cdf till that point

Poisson distribution → Count of events in a unit time.

↳ T → Time interval.



Approximated as binomial dist with $n \to \infty$ and $p \to 0$

↳ Occurance of rare events.

↳ $p = 0.05$

Exp at every min for an hour → $n = 60$.

$np = 60 \times 0.05 = 3$ → success rate

↳ mean.

13)

⇒ scipy.stas.wassertein_distance

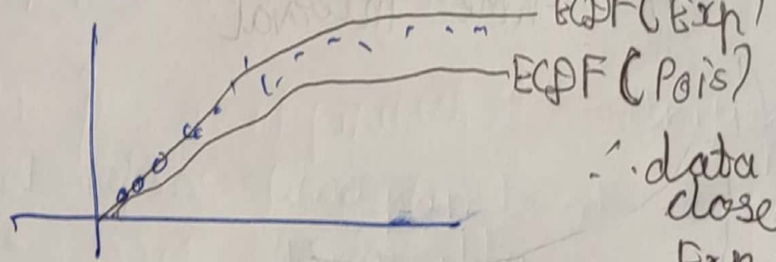
↳ To compute distance between distributions

↳ Compute difference between likelihood of two models for same data with different parameters

⇒ Number of buses per hour - Poisson dist

Interval between buses (time) → Exp dist.

→ We can compare theoretical ecdf of distributions with the given data distribution to generalise dist



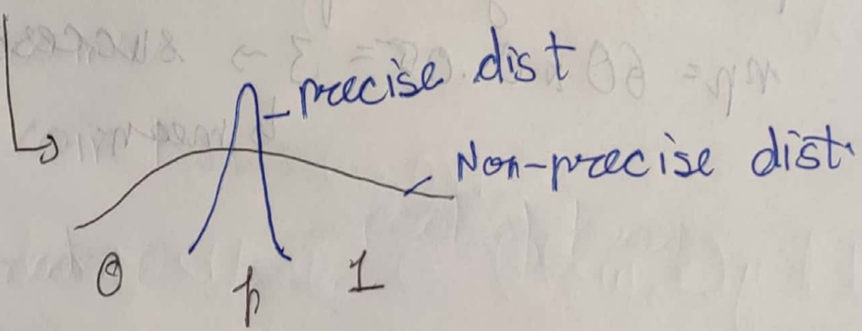
∴ data is closer to Exp at parameter.

→ Distribution + Parameter.

↳ gives configuration.

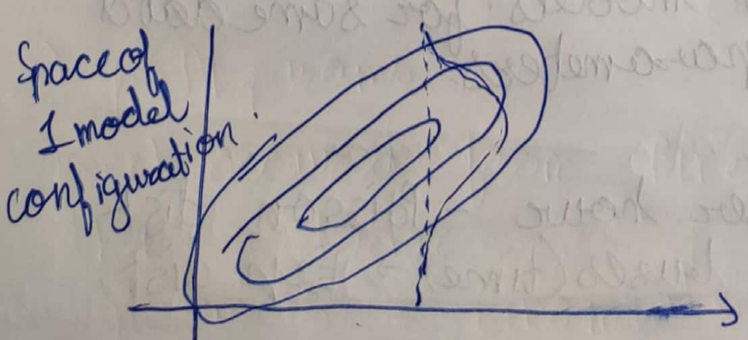
Let model be Binomial$(1, p)$ → # of possibilities = Infinite.] Hard problem

Data = H, T, T, T, H, H ... → can be arranged in infinite ways.



↳ Bayes theorem tries to link these two possibilities.

↳ Space of model + Data configs.



Regression Eq $y(x) = E(t|x)$, observed values
↳ true values.

Steps → Generate distribution of $p(t, x)$

↳ Assumption, $\hat{y}$ normally distributed ~~about~~ at each x with error of $\varepsilon \to N(0, \sigma^2)$

MLE for σ each σ, μ.

MLE can be used to find β's.

↳ Find $p(t|X)$

↳ Find $y = \int_0^\infty t \times p(t|x)$ at given x.

↳ $E(t|X)$.

Mean value of x and $y(x)$ follows linear trend.

$y(x, w\text{'s})$ → weights / β

$N(t \mid y(x, w\text{'s}), \varepsilon)$ → Is the model choise.

ε → $N(0, \sigma^2)$

$y(x, w) = w_0 + w_1 x$

or

$w_0 + w_1 x + w_2 x^2 + w_3 x^{10}$.

$w_0 + w_1 \phi_1 x + w_2 \phi_2(x)$

Estimate p given Data → Narrow our choice by selecting distribution

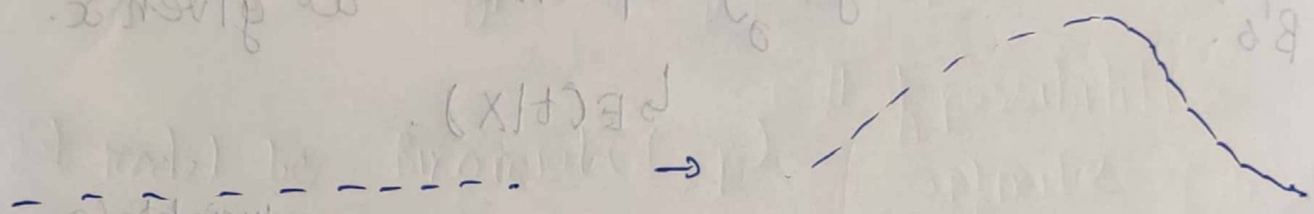
$p(\text{params} \mid \text{Data}) \propto p(\text{Data} \mid \text{params}) \times \underbrace{p(\text{params})}_{\text{prior}}$

1st → Uniform prior.

$x = \text{np.linspace}(0, 1, 100)$ → 100 points at same y possibly.

x $1-x$.

↳ p^k $(1-x)^{n-k}$.



[x ** n_success × (1-x) ** (N-n_success)]

↳ p(D|params)

D → ↳ n coin tosses

N = fixed = 0   p = ? → prob of head

1.) p(params) = prob of head

↳ Uniformly distributed

same → 0 ... 1

2) p(D|params)?

Event occured from Binom(p = 0.6, N = 0)

then ~~posterior = x^{**} n~~

n_success = Binomial(0, p = 0.6)

posterior of params = x ^ (n_success) × (1-x) ^ (n-suc

We wish posterior to accentuate $p = 0.6$ prob.

↳ from where data is achieved.

1:- Informative priors → Here we start with a uniform dist.

2. Avoid outlier data.

3- Here N → controls the amount the data.

PyMC3.

```
with pm.Model() as model:
    prob = pm.uniform('p')
    y = pm.Binomial('y', n=N, p=prob,
                    observed=n_success_a)
```

$\to$ Set context manager do its job.

Define prior

↳ Define likelihood.

We know N

1- Assumptions → Prior is uniform.

$N = 150$.

n_success_a = np.random.binomial(N, 0.3)

And we ~~assum~~ postulate $P(y, p)$ → Binomial. with known N but unknown p.

↳ We finally get posterior.

with Model :
samples = pm.sample(2000)
↳ Runs . 2000 for data.

Example : on click - rates.

with pm.Model() as model :
p = pm.Uniform("p", lower=0, upper=1) — Defining prior.
like = pm.Bernoulli("likelihood", p=p,
observed= control_df["clicks"]
P(Data, params)

↳ Remembers posterior params are finally gonna speak about the data
In this case p (eventually) will tell about the data.

Finally p (→ posterior) tells about ~~how~~ characteri-stic of distribution of Click - rates after observing data
↳ We have assumed a prior and likelihood distribution.