DO'S - ML SYSTEM DESIGN

i) Only Write Down the
- Structure
- Keywords
- Imp. pts

Explain options, reason for choosing one option over other/trade-off and only write down the chosen option

ii) Time-box every section and don't look back
Ask the interviewer if there is anything else they want to touch upon in a section before moving to next section

iii) Be conversational with your interviewer throughout
+ Keep clarifying
+ Or make assumptions, let the interviewer know and move forward

iv) You can't cover both modeling and deployment in detail.
Ask the interviewer where he/she wants to focus

v) It is possible that interviewer wants to dive deep into few sections instead of few lines about each section, so
i) Write the full outline with imp. pts first
ii) focus on answering the question interviewer asks instead of looking at time-remaining.

→ Technical Depth (e.g. focal loss, deep hash embedding, GraphSage etc.)

→ Potential Issues & their solutions (training-serving skew, Deequ)

→ Trade-offs / rationale for decisions.

→ Concise: one statement abt each bullet pt.

THINGS TO STAND OUT IN ML SYSTEM DESIGN

→ 2 Tower:
- Sampling
- ANN (Quantization & Indexing)

→ Multi-task learning:
- Single Task vs Multi-task (both regression & classification target together)
- Shared bottom vs MMoE

→ Single Task: Wide & Deep Model and Factorization Machine (advantages) / DeepFM

→ Ways to encode sequential features (e.g. watch history etc.)

→ Features: Context features: sine & cosine fn to encode 23 is closer to 1

Sparse features: Embedding Table, Deep hash Embedding

→ Loss fn: Focal Loss

→ Trade-offs:
- Pointwise, Pairwise, Listwise
- Xgboost vs NN
- Factorization Machine Adv.

→ RecSys special issues:
1) Cold start: user & item
2) Bias
   - Popularity
   - Position
   - Serving

→ <u>Relevant</u> feature list for the problem + their encoding + intuition

CLARIFICATIONS

→ Explanation of terms in question: Search → product / web / Social media

Harmful → Nudity / Violence / Hate speech

→ Think of all user actions for the problem
: will inform abt i) +ve } labels — regular / counter → online metrics
ii) -ve }

→ Explore features of system e.g.
- modality of data supported
- kinds of reaction (likes, hide, report) : how users interact with the system
- interaction with other actors (subscribe / follow etc)

→ High-level business objective :
- inc. engagement
- inc. revenue
- reduce cost
- reduce harmful content
- improve platform safety

→ Scale of system: How many users? How many items?
Daily Active Users?
Provide ballpark estimate

→ Performance of system:
- Batch Vs. Real-time
- QPS + Latency
- No. of times update is needed (e.g. # of timeline updates)

Provide ballpark estimate

→ Any constraints: Legal / Compliance, Interpretability, Online training?
re-train frequency

→ Objective Fn:
(In terms of things ML can optimize not necessarily business obj)

Maximize watch Time / Reactions / Click
or a combination of these
num of connections established (sent + accepted)

→ Type of Problem: RecSys (Retrieval + Ranking) / Classification / only Ranking

→ Modeling Architecture
— Multi-stage or One-stage
— Multi-task or single task

PAUSE:
TOP THINGS TO HIGHLIGHT
→ What are the top things in this problem that I want to highlight to gain brownie pts? (since time-limit)

→ KEY CHALLENGES:
RecSys: Cold Start, Bias
Fraud & Abuse: Keeping up, friction with user for false +ve, Label scarcity & delay
Human-in-the-loop

→ KEY FEATURES:
Only describe features that seem most predictive for the problem
: intuition + encoding (e.g. strength of conn. → people you may know)

ONLINE METRICS : Think of metrics to measure when the system is live based on user-item interactions

→ Primary metrics
→ Secondary metrics
→ Counter Metrics

→ Justify online metrics to high-level business objective

User-item interaction level (can avg. over entire user base): impression, click (CTR), dwell time (7 sec), reaction (→ like, comment, share, subscribe, follow), further action e.g. purchase, conversion, watch time, event registration, video completion rate, conn. req sent + accepted

Session Level:
- → Time spent on recommended content vs search in a session
- → Time to success (in search)
  No. of queries
- → $\frac{\text{sessions with clicks (or intended action e.g. purchase)}}{\text{Total \# of sessions}}$

→ Counter Metrics: hide, report, search time → in RecSys search time high ⇒ recommendations not good

Business level (Avg. over entire user base):
- → Stickiness : Users who re-visited in the next X days (retention)
- Conversion Rate
- Revenue lift
- Bookmark Rate
- Satisfaction Survey

FRAUD / ABUSE Use Cases:

TP, FP, FN, TN:
- Impressions of harmful content
- $\frac{\text{\# of harmful content detected}}{\text{\# of harmful content detected + reported by users}}$
- % of content detected harmful but appealed & reversed

LABEL CRITERIA AND TRAINING SET

→ Annotated/Human labeled Vs Existing onboarding/engagement flow data

Pros/Cons:

- Not scalable
- High quality

- Scalable
- Noisy

→ +ve

- Click
- Click + dwell time > t sec
- Reactions } may be weighted by diff. reactions
- purchase
- registered for event
- conn. sent + accepted
- Session-based: items in context window

-ve

- hide
- report
- impression less than t sec

→ Graded Relevance | for pairwise or listwise LTR

- only impression, no click = 0
- Click = 1
- Click + dwell time > t sec = 2
- Click + dwell time > t sec + further engagement = 3
- Click + dwell time > t sec

→ Delay or Unavailability of labels:

- Proxy
- Re-formulate the problem e.g. whether user will return after x days
- Human labellers
- Unsupervised
- One-class classification

→ Popularity bias: Restrict pop. items by a certain % of training set (limit # of labels per user or per item)

→ Privacy ? need anonymization in data

→ Fairness / Bias for gender, age, race, underrepresented grp : make sure they are appropriately represented in data (e.g. % by vol.)

→ Sampling -ve set:

Ranking / Classification:

- Cluster -ve items & pick certain %
- Pop. based sampling
- Hard -ve sampling (select -ve items that are similar to +ve items)
- undersample -ve set
- oversample +ve set e.g. SMOTE
- Class wts (e.g. scale_pos_wt in xgboost)

Instead of balancing use i) balanced cross-entropy or better ii) FOCAL Loss

Cand. Generation in 2 tower:

i) in-batch random -ve
ii) in-batch -ve
iii) Hard -ve

} combination

→ CANDIDATE GENERATION

- Labels & Sampling
- Multiple ways of generating candidates
- How to limit candidates (based on some heuristic or imp. feature)
- Why multiple cand. generators? Diversity
- Metrics: (Recall focussed)

→ TRAIN-TEST SPLIT:

- Usually by time frame/days, same user not in train and test, no leakage
- only downsample/take care during training, val & test set keep the same distribution of +ve & -ve labels

→ SOURCING TRAINING DATA: Upstream pipeline monitoring which populates the base tables e.g. using "Deequ"

FEATURE-ENGINEERING

→ Identify Actors and their historical interaction with "all" of other actors

ITEM: – main "properties" of item e.g. price, location, duration, cuisine etc
- modality of data e.g. text, video, image → embeddings
- engagement + reputation : likes, shares, etc. pagerank, subscribed } rate + counter features
- freshness/age
  Location
  Position

USER: – Demographics: age, gender, lang., country, city, account age
- User – item (all) historical interaction
  - Identify some imp. properties of item and user's interaction history with these properties of "all" item
    ↓
    counter and rate features
  - seq. features
    - watch history of items
    - Like history of items
    - Search history of items
- User – "all" other actor (author, advertizer) historical interaction

SPARSE USER & ITEM FEATURES: – User Id
(can avoid them since they come at huge computational/storage cost → embedding table size but Deep hash embedding can solve for it)
- Item Id
  - Id of imp. properties of item e.g. director id, actor id etc.

NOTE: ONLY the above features go into 2 tower so that isolation is maintained for creating User and item embeddings. User Tower: User (dense + sparse)
Item Tower: Item (dense + sparse)

CONTEXT:
- specific to user-item pair : time of day, day of week, is holiday, device
- aggregate (ratio & counter feature) at user level : → can be ingested in user tower of 2 tower architecture
  : %. of media consumed on mobile

CROSS-FEATURES: "specific" User-item-other Actor interaction
- "specific" User-item interaction
  e.g. how many times user has watched the genre of movie that is to be ranked
- "specific" User-other actor interaction
  e.g. posts liked by user created by auth

NOTE: The above features go into Ranking / Classification where specific user-item pairs are ranked / classified (along with user & item features : dense + sparse)

NOTE: In case of Ranking / Classification after 2 tower N/W, the features used are :
1). Candidate generation score (dot product b/w user & item embeddings)
2) Avg. embedding of user-item historical interaction e.g. watch history, like history (seq. of items)
3) Cross-features.

ENCODING FEATURES

SPARSE FEATURES: EMBEDDING LAYER

Issue → Using embedding layer for sparse features with high cardinality results in embedding lookup table getting huge and slows the process of training/inference significantly

e.g. user id (5000 users) mapped to embedding vector (128 dim.)

results in embedding lookup table of size (5000, 128)

Sol^n:

1) Hashing to reduce size of lookup table
2) Using rate & counter features
   - rate → summary statistics count by time: count_by_time, max_in_3_months, min_in_2_weeks
   - counter → summary statistics count: count, max, min
3) Deep Hash Embedding

NUMERICAL FEATURES:

Normalization or Standardization or Binning

CATEGORICAL FEATURES:

- one-hot encoding: if cardinality is low
- Feature-hashing: when lot of values/high cardinality
  Adv: can handle unseen categories
- Embedding: usually jointly learnt with DNN model (extra layer to convert sparse one-hot encoded vector to dense embeddings)

SEQUENTIAL FEATURES:

- If 2 tower used in cand. generation phase then fetching embeddings for items and avg. them (e.g. watch history)
- Attention layer and then concatenating the o/p
- Embedding layer and then aggregating the embedding of each item

TIME BASED FEATURES: sine & cosine fns so that 23 is closer to 1
(e.g. CONTEXT FEATURES) i.e. cyclic nature is captured

LOCATION BASED (Lat, Long): Haversine Distance

IMAGE, VIDEO, AUDIO: Pre-processing & using pre-trained models to extract features

TEXT-BASED: word-level: Word2Vec (not context-aware)
clause or sentence level: pre-trained models like BERT embeddings

MODELING :

→ Baseline Model
→ Multi-stage or One-stage < binary / multi-task (Classification)
(retrieval, filtering, ranking, re-ranking) (usual 2 main actors: user-item pair for every user-item) e.g. classify new post as harmful

1st Stage < 2-tower / filtering based } Diff. ways of generating candidates + limiting their number
(first degree connection posts (unseen), pages/people followed, groups joined
events in the 5 mile radius,
ads for specific country, age, gender)

→ Single Task Vs Multi-task in Ranking/Classification
(MMOE) → does not req. user-item cross-features

→ Inference is usually Real-Time
→ Search System: 2-tower: 1) User-Query Tower
2) Item Tower (Get the score of 2 tower model as feature in ranking phase)

→ For Ranking/Classification Model: Start with baseline
a) Baseline model / Rule-based system : Pros & Cons (more accurate but more difficult to implement & train)
→ i) Pointwise, Pairwise, Listwise
training data: graded relevance

→ 2) After choosing pointwise, primarily the decision is b/w boosting Vs DNN

Choosing NN: ✓ data is unstructured
✓ lot of sparse features & embedding is needed
✓ continual learning
Training & inference (slower than xgboost)

→ For Search/RecSys : 1) Google Wide & Deep Model
(since these models select either model 2) DCN
generalize better) 3) Factorization Machine or DFM

→ Loss Fn: Focal loss ✓(imbalanced + multiple modalities in data)
or
Balanced Cross-Entropy (✓ imbalanced data)

→ Training pipeline, Managing Model Version, Distributed Training

Adv
1. Development velocity
2. Less # of labels for fine-tuning

→ For NLP or Image based models: 1. fine-tuning on task specific labelled dataset
2. Use embeddings from pre-trained e.g. Transformer (BERT, GPT) or ResNet, VGG, Inception)
(if classification or 2 tower arch.)

- Other ways of generating cand: popular items, trending items

- Cold start in 2 tower architecture: (no interaction data)
  or in general for cand. generation
  - new users: Only use user profile info to get embedding & find top K nearest neighbor items
    (2 tower architecture accepts new users)
  - new items: if item profile available, use 2 tower to get item embedding and store in embedding table and use it while finding top K nearest neighbor items
    (2 tower can work with new items if features available

- In case of Recsys, give pros & cons of Matrix Factorization V 2 Tower

Matrix Factorization	2 Tower
- Training efficient	Costly to train
- Inference faster (as embeddings are static & can be pre-computed)	User features need to be transformed into embeddings at query time
→ Cold Start (only relies on iteration data)	Cold start is substancially reduced
- Quality of Recsys: Not Ideal (does not use item or user expansion features)	Better since it relies on user & item features

- For faster ANN: Vector encoding / e.g. product quantization
  2) ~~Inverted~~ Index → inverted index, HNSW (hierarchical Navigable Small world), LSH (Locality sensitive hashing)

ANN not needed if # of items is small, NN can work

- Filtering:
  - Age restriction
  - Geography restriction
  - De-dupe
  - Harmful content

- Re-ranking service:
  → Diversity (avg. pairwise similarity b/w ranked list of items)
  → Freshness
  ~~+ De-dup~~
  ~~+ Harmful content~~
  - If the item has been recommended recently / product purchased (then discount the score by certain amt ⇒ don't show it for a certain duration ~~make it~~ available
  or reported / hide → remove them

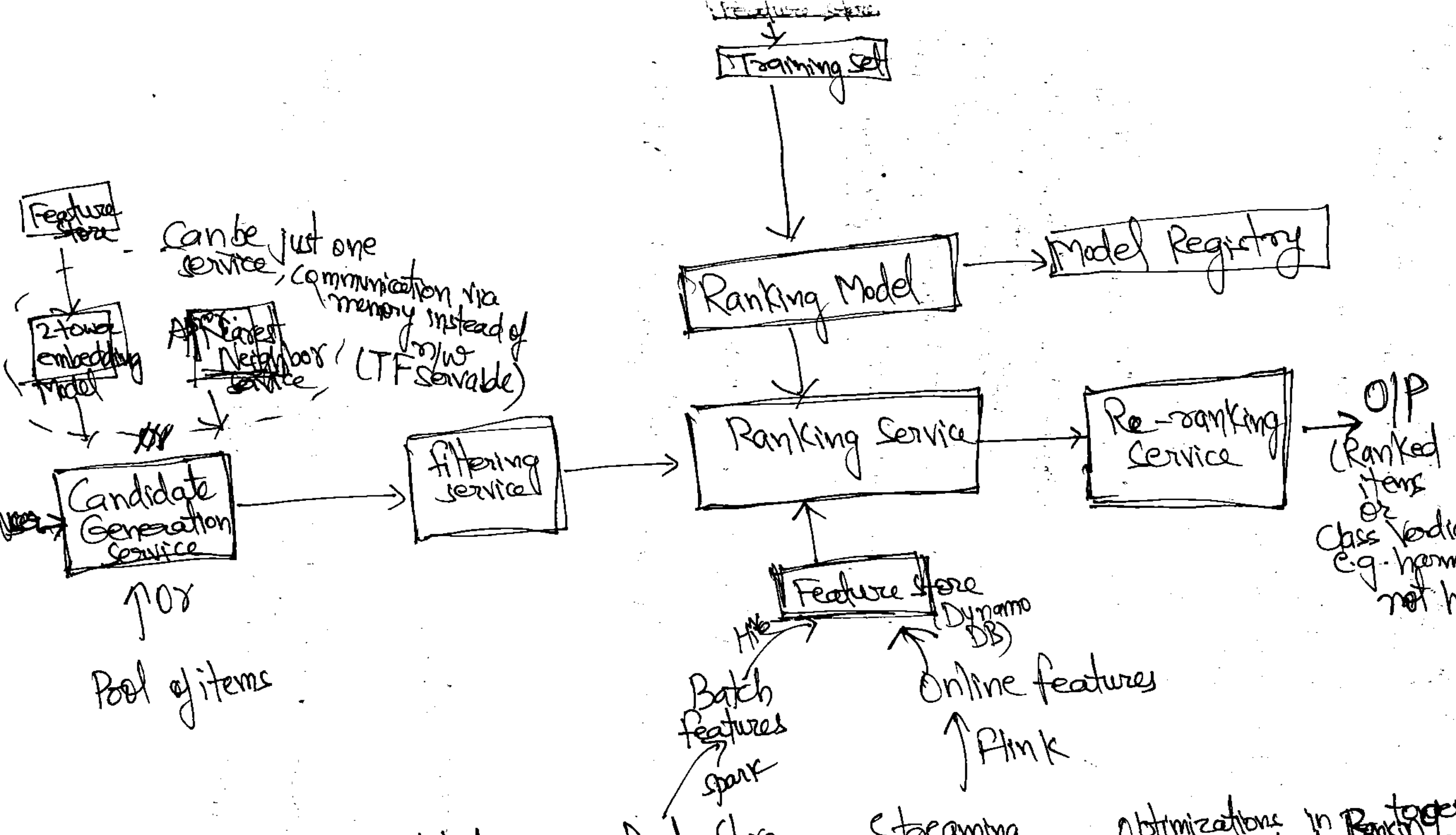
OFFLINE METRICS

→ Multi-stage: Retrieval/Cand. Generation phase : Recall specific metrics e.g. Recall @K

Ranking (precision based metrics): NDCG@5, MAP, Precision @K
Freshness, Diversity

→ Classification: Precision ~~go~~ Recall, PRAUC, F1, AUCROC
(sometimes have to decide b/w precision & recall)

INFERENCE&SERVING DESIGN



Optimizations in 2 tower Architecture:

1. Each tower could be less Complicated NN eg. FC layers only
2. Distributed Training
3. Embedding Model +ANN service could comm. via memory instead of network (Tensorflow Servable framework)
4. Vector encoding (e.g. product quantization) + Inverted Index

Optimizations in Ranking Model

1. Distributed training
2. Scaling using Load balancer or Auto scaling (Docker + Kubernetes)

- How to mitigate serving bias / feedback loop issue / Explore-Exploit
  - Multiple cand. generator
  - ε-Greedy approach / heuristics (sometimes use random items not from O/P of RecSys).
- How to reduce latency (in case real-time)
  - Pre-computing feature / embeddings
    Caching
  - Model Compression
  - Use simpler feature-engg.
  - Use more hardware / infra

- Infrastructure Design:

  Kafka → Flink → Dynamo DB (online features)

  Data store → Hive (offline features)
  not time-sensitive

  Model Registry consists of
  - Object storage: to store model binary files etc
  - Relational DB: to store model metadata
  - APIs: to retrieve model artifacts and metadata

  ↓

  E.g. Sagemaker Model Registry

- COST: Spot vs dedicated instances (for training)
  GPUs
  Cost Monitoring via dashboard (weekly, monthly, projections etc.)

DEPLOYMENT

Model Registry
↓
- Docker + Kubernetes

A/B Testing
- Shadow Mode
- Canary Deployment → gradually rolling out an update to a small % of users before making it available to the entire user base
- Refreshing model

MONITORING & LOGGING:

At the very least: talk about 1) Model Metrics
(if label is available
if label is not available)
↓flag rate etc.

2) Operation metrics

3) Training-serving skew

Log Model Request + Response → sanity check & future debugging

A/B Test ~~or Multi Armed Bandit~~

1. Ranking - Interleaving
2. Classification - Usual A/B test setup with MDE, sample size and basic steps

→ Novelty Effect

SPECIAL ISSUES IF NOT MENTIONED

1. Serving bias
2. Popularity bias
3. Position bias
4. Cold start problem

→ Calibration

Problem exploration
Modeling approaches
Infrastructure design
Training/testing
Deployment

Strengths
Gave an overview of the template used
Asked about the framework
Mentioned what type of actions
Asked about where this is surfaced eg homepage
Candidate mentioned maximizing both the send and the acceptance
Identified the top objectives as multiobjective on click + acceptance
Mentioned users see new results on each refresh
Mentioned a bunch of online metrics to measure the system
Mentioned user engagement after x days as additional metrics
Talked about the main actors in the system
Mentioned about interaction features
Mentioned network features eg number of common friends
Also mentioned second hop friends as candidates
Proposed a heuristic way to threshold the multihop candidates
On mentioning cold start came up with an alternative recall source to mitigate this
Mentioned categorical feature encodings
Mentioned using sparse id and deep hash embeddings -> technical depth
Mentioned deep and wide models and factorization machines as they are good with feature interaction
Mentioned using graph sage - graph NN for ranking
Mentioned using focal loss for ranking
Mentioned using recall specific metrics
Mentioned using NDCG ranking specific metrics
Mentioned training data and serving have no distributions

Areas to improve
Mentioned 10 times refreshed the system without justifying
Better answer is ask about latency of search
What is the goal? User experience. How do users interact with the system? How much personalization?
Scale, users, DAU, items, QPS, explainability, retrain frequency, latency
How much data do we have? Whether data is available or existing onboarding flow
Need to talk about primary metrics, secondary metrics and guardrail metrics
Justify business metrics to target, connect business needs to ML decisions
Different characteristics in the industry + risks of a complex design
Ask about the end to end product flow of the system
How fast should the model be? Is precision or recall more important
Can include more graph based features eg page rank
Can mention collaborative filtering on candidates
Need to mention cold start eg using friends of friends
Can mention filtering friends in candidate selection based on user affinity
When mentioning embeddings need to mention what we embed and how
Can use graded labels instead of 1 or 0
No need to explain focal loss in detail - need better explanations eg importance weighting
Its very important to talk about training data -> how to source it/pitfalls etc
Need to talk about ab testing/canary deployment
Important to touch upon biases -> eg position bias, popularity bias etc