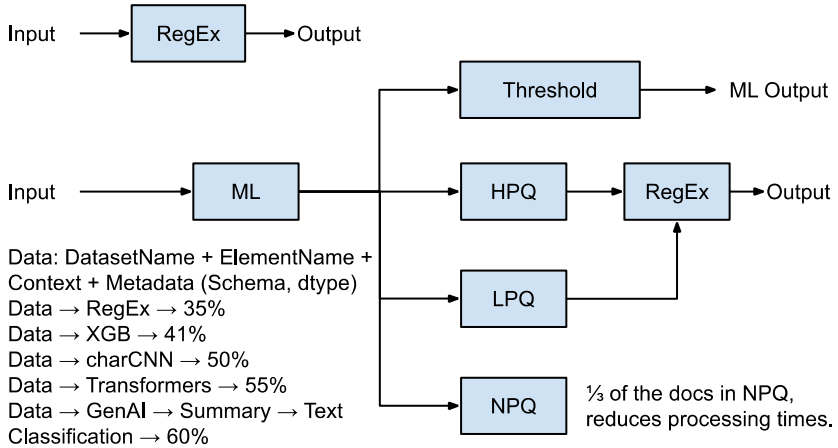
Information Retrieval and Detection Systems



Data: DatasetName + ElementName + Context + Metadata (Schema, dtype)
Data → RegEx → 35%
Data → XGB → 41%
Data → charCNN → 50%
Data → Transformers → 55%
Data → GenAI → Summary → Text Classification → 60%

With LLaMA-2:
- Lexical-based metrics:
  - Exact match: 8%
  - Rouge-1: 0.49, Rouge-2: 0.41
  - Meteor: 0.47
- Semantic-based metrics:
  - BERTScore (F1): 0.69
- Human review metrics: Accurate, complete, consistent, coherent. Avoidance of undesired outputs.

Open Source Models:
1. **Llama 2** (Meta; 7B, 13B, and 70B)
2. **MPT** (MosaicML; 7B and 30B)
3. **Falcon** (TII; 7B and 40B)
4. **Dolly 2.0** (Databricks; 12B)
5. **FLAN-T5** (Google; 80M to 11B)
6. **Mistral** (Mistral AI; 7B)
7. **BERT** (Google; 110M, 340M)
8. **allMiniML-L6** (HF; 11M)

Example Prompt: Given the input comprising DatasetName, ElementName, Context, and Metadata (including Schema and data type), summarize the potential presence of confidential user information. Consider the implications of dataset labels, the nature of data elements, the context in which data is used, and the structural metadata that defines data organization. Highlight any indicators or patterns that might suggest sensitive information such as social security numbers, bank account details, and other personal identifiers. Provide an overview of the data's characteristics, focusing on aspects that are particularly relevant for privacy and data protection.

Document Recommendation

- 3 steps: Graph, TF-IDF, and Transformer
- Transformer BERT approach details:
  - **Data preparation**
    - A-A*: Similar documents. A* is prepared by replacing words in A by their synonyms.
    - A-B: Dissimilar documents. Use usage graph to find dissimilar docs with distance >= 3.
  - **Data table:**

Pair	Label (y)
A-A	0
A-B	1

  - **Contrastive loss**
    - $L = (1-y) * \|x_i - x_j\|_2^2 + y * \max\left(0, m - \|x_i - x_j\|_2^2\right)$
    - $y = 0 \Rightarrow L = \|x_i - x_j\|_2^2 \Rightarrow$ for similar docs, larger euclidean distance is penalized.
    - $y = 1 \Rightarrow L = \max\left(0, m - \|x_i - x_j\|_2^2\right) \Rightarrow$ for dissimilar docs, euclidean distance $< m$ is penalized.
    - m too large => dissimilar docs fall farther apart. Distance < m will be penalized, so too large value of m might be good for two docs that are fully different, but models won't learn to distinguish docs with similarity distance less than m.
  - **Training: With LoRA (Low Rank Adoption)**
    - $W = W + W_r = W + AB,\ W^*, W, W_r \in \mathbb{R}^{m \times n}; A \in \mathbb{R}^{m \times r}, B \in \mathbb{R}^{r \times n}; r \ll m, n$
  - **Evaluation**
    - Precision: Proportion of likes improves from 60% to 80%, i.e., 8 out of 10 recommended documents were liked by the users, while recall remains the same at 90% for TF-IDF vs. BERT models.
  - **Candidate selection model**
    - Select candidate documents from usage graph with distance <= 2. If the number of candidates > 500, apply TF-IDF cosine similarity to filter out the candidates.

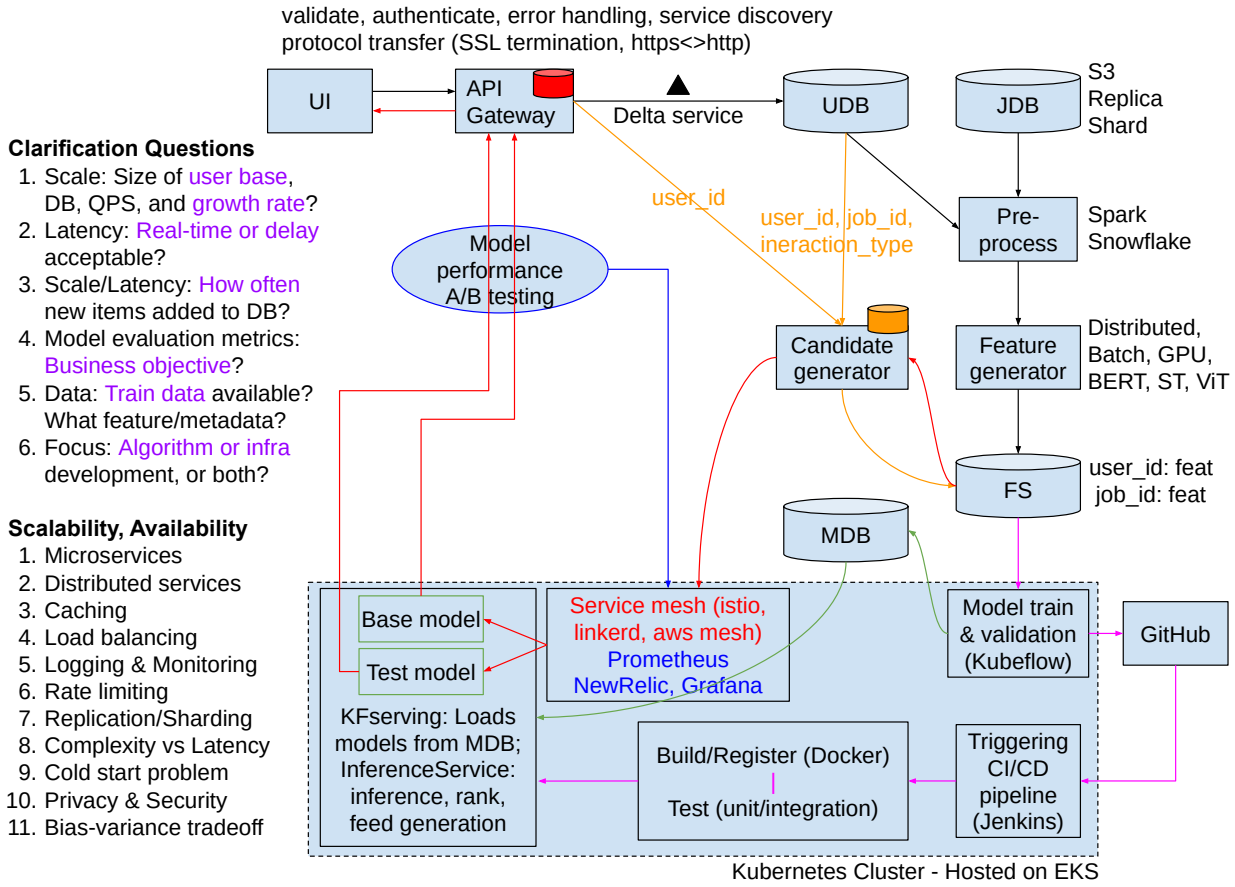
ML System Design: E2E ML Workflow

3-3-RULE

- Data Engineering
  - Collection (Run experiments)
  - Preprocessing (Cleaning, Labeling)
  - Feature Engineering (Which features, and how can they be used or useful?)
- Model Development
  - Selection (Identify the appropriate models depending on the problem)
  - Training/Fine Tuning (transfer learning, LoRA, QLoRA, RAG, Feedback Loop)
  - Evaluation (Offline metrics: MSE, Precision, Recall, F1-Score, Accuracy)
- Model Deployment
  - Setup infra (ECS cluster, Docker container, networking-scaling (ALB, ASG, SNS, SQS), and storage)
  - Serve the model (deploy model services using Kubernetes and implement A/B testing)
  - CI/CD (Setup GitHub and Jenkins for automatic building from changes)

Problem Name	**Best ML Models**	**Cost, Optimization, Evaluation**
Recommendation (LinkedIn job, Netflix movie, YouTube video, E-commerce product, Events, News feed) (A/B tests: t-test with equal var; Welch's t-test if group sizes vary largely)	TF-IDF, Collaborative Filtering, Matrix Factorization, BERT, ST Pairwise Loss is for Ranking Cross-Entropy is for Classification	• Personalized? User-X interaction data? • Pairwise Loss=max(0, 1 – (sim^+ – sim^-)) Data Table: (uid, +/-pair, +sim, -sim) • Local tests: Precision@k, NDCG@k
Classification (Ads click, Article tagging, Harmful content detection, Transaction fraud, Server logs anomaly)	Logistic Regression, Gradient Boosting, Random Forests, Isolation Forests (unsupervised), LSTM, CNN, BERT	• Number of tags? harmful/anomaly/fraud? • Cross-Entropy • Accuracy • Precision, Recall, F1-score, AUC-ROC
Object Detection/Segmentation (Face, Pedestrians, Facemask)	U-Net, YOLO, ViT (classification), Swin Transformer (detection)	• Partial visibility or occlusion? • Cross-Entropy, Dice Coeff, IOU • Precision, Recall, F1-score, mAP
Regression Prediction (Traffic pattern for GPS app, Stock price)	ARIMA, LSTM, Prophet	• Time horizon? Factors (weather, event)? • MSE, MAE, RMSE

ML System Design: Deployment Infrastructure and Operational Pipelines



Clarification Questions

1. Scale: Size of user base, DB, QPS, and growth rate?
2. Latency: Real-time or delay acceptable?
3. Scale/Latency: How often new items added to DB?
4. Model evaluation metrics: Business objective?
5. Data: Train data available? What feature/metadata?
6. Focus: Algorithm or infra development, or both?

Scalability, Availability

1. Microservices
2. Distributed services
3. Caching
4. Load balancing
5. Logging & Monitoring
6. Rate limiting
7. Replication/Sharding
8. Complexity vs Latency
9. Cold start problem
10. Privacy & Security
11. Bias-variance tradeoff

- Data
  - Exploration and handling missing data and outliers
    - Missing data: Imputation, Regression, SVD
    - Outliers: data point >= 2 * z-score/standard deviation, or use anomaly detection algorithms
  - Feature engineering/generation (embeddings, transformation, normalization, scaling)
  - Train/test split: 80/20, Cross-validation for limited amount of data
- Model Deployment
  - Train and validate model (using in-house tools and workflows, or within kubeflow) --> save the model in .onnx format and upload to artifactory --> push changes to github with yaml configuration file --> jenkins runs builds, tests, and deploy ServiceInference (kubectl apply -f file.yaml) --> KFServing is now operational, handles incoming inference requests, and scales based on the traffic.
  - Now, ServiceInference is accessible via a specific URL provided by Kubernetes cluster's ingress or load balancer. To send inference request, create a script that runs in a regular basis. The script contain the URL and link to dataset that is to make inference for. YAML file contains storageUri (model path), predictor (torch), compute resources, scaling parameters (pods), deployment command.
  - CI/CD pipeline
    - CI (change, push, build, test); CD (canary deployment or whole system deployment).
    - **GitHub Actions** (hosted by GitHub, suitable for simple workflows); **Jenkins** (self-hosted, suitable for more complex workflow since it is more configurable).
  - **Kubeflow** (open-source, customizable, uses Kubernetes infra); **Sagemaker** (AWS hosted); **MLflow** (open-source, less developed); **Airflow** (used for all workflows, not specialized for ML).

General DL Concepts

- Concepts in neural network (input, initialization, layers, activations, output, cost function, backpropagation, optimization, weights, and biases, hyperparameters, overfitting, regularization, batch, and layer normalization, shifting and scaling, input data normalization and scaling)
- Initialization
    - Zero vs. Random
        - Zero -> cannot learn well -> outputs of all neurons are same -> weight updates be same -> can represent one/limited number of features -> required randomness
        - Random -> vanishing/exploring gradient if random values are very low/high.
    - He vs. Xavier/Glorot
        - He -> mu=0, sigma=2/√n_input -> better for ReLU -> Higher sigma than Xavier -> probability of generating larger positive values is higher -> negative values doesn't count since they become zero in ReLU.
        - Xavier -> mu=0, sigma=2/√(n_in + n_out)
- Activation function
    - Required to introduce non-linearity and therefore, to capture complex patterns.
    - Sigmoid (for binary class probabilities), Tanh (when data needs to be centered around zero), Softmax (for multiclass probabilities), ReLU, Leaky ReLU, GeLU
    - For multi-label multi-class problems use Sigmoid separately for each output neuron.
- Cost function
    - Properties of good cost functions: Convex, continuous, differentiable. If not differentiable at any x, use sub-gradient, e.g., gradient of MAE or ReLU at x = 0 is 0.
    - Cross entropy -> Difference between two probability distributions, e.g., prediction vs. ground truth.
    - Cross-Entropy loss is widely used:
        - CE measures the difference between the probability distributions of the classes that we want.
        - CE is convex, continuous, differentiable for classification.
        - CE measures the negative log-likelihood of binary class labels under binomial model, which aligns with the assumption of (logistic regression) classification—outputs follow binomial distribution.
- Backpropagation
    - An algorithm or process of updating weights based on the derivatives of cost function.
    - Compute the gradient of the loss function w.r.t. the params using chain rule and update the params.
- Optimizer
    - Gradient descent, SGD: Gradient is rate of change -> we want to go in the direction with maximum change, i.e., to the steepest direction -> take the directional derivative to find that direction -> $\nabla_{\hat{u}} f = \nabla f(x).\hat{u} = |\nabla f(x)||\hat{u}| \cos\theta = |\nabla f(x)| \cos\theta$ -> which is maximum when $\theta = 0$, and the max rate of change is $|\nabla f(x)|$ -> which implies the max is in the direction of derivative -> In gradient descent, we need to go downhill, so use negative sign in the weight update formula.
    - Adam (Adam uses both SGD with momentum (first moment = mean = EMA of derivative in each weight's direction -> oscillation cancelled by positive and negative -> faster in other directions) and RMSProp (second moment = variance = EMA of squared derivative in each weight's direction -> learning rate is divided by second moment))

- Hyperparameters
  - Network architecture
    - Number of layers, hidden size, activation function
  - Regularization
    - Concepts: Overfitting, underfitting, and bias-variance tradeoff
      - Overfitting: Models learn training data too well, including its noises and outliers, but fail on new data → model is too complex → introduce error due to variance.
      - Underfitting: Models fail to learn enough from training data; poor performance on both training and new data → model is too simple → introduce error due to bias.
      - Bias-variance tradeoff: Find a good balance between bias and variance, minimizing total error. Total error = Bias^2 + Variance (+ irreducible errors, e.g., measurement errors)
      - Use regularization (e.g., L1, L2, dropout), ensembles, normalization, data augmentation to overcome overfitting.

    - L1 vs. L2 regularization
      - $L = (y - \hat{y})^2 + \lambda \|w\|_k$ where $k = 1$ for L1 or $k = 2$ for L2 regularization.
      - L2-norm is linked to Gaussian, while L1-norm is linked to Laplacian distributions, i.e., negative log-likelihood of Gaussian is proportional to L2-norm and negative log-likelihood of Laplacian in proportional to L1-norm.

$$G = c\exp(-\gamma(x-\mu)^2) \rightarrow \log G = \prod_{i=1}^{n} c\exp(-\gamma(x-\mu)^2) \rightarrow -\log G = -n\log c + \gamma\sum_{i=1}^{n}(x-\mu)^2$$

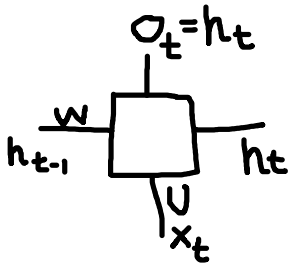
$$L = c\exp(-\gamma|x-\mu|) \rightarrow \log L = \prod_{i=1}^{n} c\exp(-\gamma|x-\mu|) \rightarrow -\log L = -n\log c + \gamma\sum_{i=1}^{n}|x-\mu|$$

    - Dropout: Exclude some neurons/parameters during training
      - Training: scale the output of the neurons by $1/(1-p_{dropout})$
      - Testing: Use the full network with no dropping.
    - Batch normalization vs. layer normalization (scaling and shifting after normalization)
      - Batch: Normalizes the inputs to a layer for each mini batch → depends on batch size.
      - Layer: Normalizes the outputs of each neuron using the outputs of all neurons in that layer → does not depend on batch size.
      - Scaling and shifting: Normalization makes the input distribution more consistent. However, if normalization always results in mean=0 and variance=1, the network might be unable to represent data that naturally have a different distribution. Scaling and shifting after normalization helps overcoming this issue: $x_i = \gamma.\frac{x-\mu}{\sigma} + \beta$
      - Normalization during testing: Use the same normalization parameters used in training, except for layer normalization, in which normalize the output regardless of inputs.
    - Data scaling vs. data normalization
      - Example: Scaling (-1, 1), normalization (mu=0, sigma=1) → scaling is good for distance-based models, e.g., SVM and KNN. In SVM, if one feature has a larger scale than the others, it will dominate in the margin calculations.

- Optimization/training related hyper-parameters
  - Learning rate (high (faster convergence, low stability) vs. low (slow convergence, high stability))
  - Epoch vs. batch vs. iterations (in each iteration, weights are updated for one batch)
  - Low Rank Adaptation (hyper-parameter: rank)

$$W = W + W_r = W + AB, \qquad W^*, W, W_r \in \mathbb{R}^{m\times n}; A \in \mathbb{R}^{m\times r}, B \in \mathbb{R}^{r\times n}; r \ll m, n$$

- CNN
  - Layers: input, convolution, activation, pooling, fully connected.
  - Convolution operation: A kernel/filter sliding over the input and producing a dot product.
    - Filters work as feature detectors, e.g., edges, corners, blobs, etc.
    - Dot product computes the similarity, i.e., part of the image similar to the filter.
  - Pooling: Max and average pooling → used for dimensionality reduction → may help to reduce noise.
  - Concept of padding and stride? What is padding used for?
    - Padding: Ensures no border information is lost during convolution.
    - Stride: Step size of convolution filer's move → affects the spatial dimension of the output.
    - H x W x D → H' x W' x N, where H' = (H – F + 2*P) / S) + 1
    - H=height, W=width, D=depth, P=padding, S=stride, F=filter size, N=filter number.
- RNN
  - Sequential processing
  - Hidden state: Captures info processed so far → works as network's memory.
  - Challenge: vanishing gradient
    - Hidden state outputs at time point 2 and 1 are:
    - $h_2 = \sigma(Wh_1 + Ux_2) = \sigma(z_2),\ h_1 = \sigma(Wh_0 + Ux_1) = \sigma(z_1)$
    - $\frac{\partial h_2}{\partial W} = \frac{\partial h_2}{\partial h_1}.\frac{\partial h_1}{\partial W} = \frac{\partial h_2}{\partial z_2}.\frac{\partial z_2}{\partial h_1}.\frac{\partial h_1}{\partial z_1}.\frac{\partial z_1}{\partial W} = \sigma'(z_2)W\sigma'(z_1)W = W^2\sigma'(z_2)\sigma'(z_1)$



  - LSTM: Overcomes vanishing gradient problem using gating mechanisms, making it easier to remember the long-term dependencies. **Cell state** stores the information; **forget gate** determines what info can be discarded from the cell state; **input gate** decides what new information to add in the cell state; **output gate** decides which information to take from cell state to generate output.
- NLP
  - Text processing: Tokenization, stemming & lemmatization, punctuations and stop words removal.
  - Token embeddings:
    - **One-hot-encoding**: Create a list of words present in all documents. For each sentence, put 1 for a word if that word in present in that sentence (regardless of number of frequency of the word).
    - **Label encoding**: Assign a random number for each category (Red=0, Green=1, Blue=2)
    - **Target encoding**: Take target into account. Red = target_mean_Red -> Data leakage → variants: weighted target encoding, K-fold target encoding
    - TF-IDF: Like one-hot-encoding but it considers the number of frequency of the word in the sentence, which is term frequency. Then it computes IDF and finally TF * IDF = TF * log(N/n) → N=total documents, n = document where the word is present)
    - Example: 2 sentences – I like Apple. Apple is good. → Processing → like apple. apple good.

Word level encoding

Tokens	Words		
	like	apple	good
like	1	0	0
apple	0	1	0
good	0	0	1

Sentence level encoding

Tokens	Words		
	like	apple	good
i like apple	1	1	0
apple is good	0	1	1

Sentence level TF-IDF

Tokens	Words		
	like	apple	good
i like apple	1	0	0
apple is good	0	0	1

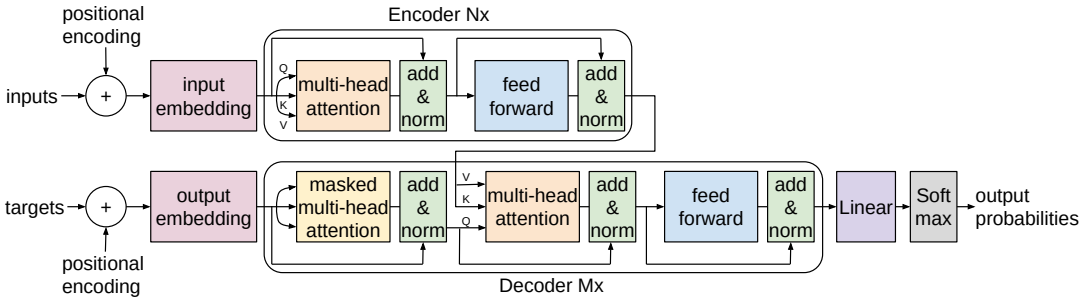
        - Word2Vec
            - What is Word2Vec? How does it work and what are its limitations?
                - NN-based model -> learn to generate embeddings of the words in a sentence -> considers only the local context (e.g., “I like eating apple” and “apple released iPhone” have different contexts for apple, so the embeddings will be different) -> cannot capture the context when words have same local context but different meaning (e.g., “I like apple fruit” and “I like apple products” -> show affection for apple, i.e., same context -> generate same embedding for apple although their meanings are different) -> Also it does not consider the order of the words in the sentence.
            - Two architectures of word2vec
                - CBOW: Continuous bag of words -> predict the target words given context words.
                - Skip-gram -> predict the context words given the target word.

CBOW	Skip-gram
(In → out): like, apple → I	(In → out): I → like, apple
(In → out): I, apple → like	(In → out): like → I, apple
(In → out): I, like → apple	(In → out): apple → I like
The NN model learn the vector reprecipitations of the input and outputs words, so that the probability of observing the output words is maximized given the input words.	

        - Transformer-based encoding
            - Context-aware embeddings that considers the position/order of the words in the sentence.
        - How to evaluate the quality of the embeddings?
            - Performance of the downstream tasks
    - Common NLP tasks: Sentiment analysis; Named entity recognition; Parts of speech tagging; Machine translation; Speech recognition.
- LLMs
    - Foundation model: the pre-trained base model.
    - Pretraining: The foundation model is trained with a large amount and varieties of text data.
    - Optimize LLM for specific applications:
        - Fine-tuning: Further tune for specific task with limited and domain specific dataset.
        - RAG: Models first retrieves from a dataset and then use that context to generate responses.
    - Neural scaling law: Performance tends to increase with model size (parameters) and training dataset size.
    - Evaluation: Exact matching, BLEU, ROUGE, METEOR, BERTScore, Human review (accurate, complete, coherent, consistent, desired, undesired)
    - RLHF: is a training approach where the models learn the desired behavior based on feedback from human interactions, rather than solely relying on pre-collected datasets.
    - Bias

	Data	**Model**
Sources of biases	Underrepresented train data	Complex/overfitting
How to measure biases?	Statistics: Proportion & Distribution Metrics (e.g., accuracy) per category	Stability analysis: Introduce noise in inputs → Small input changes results in larger output shifts
How to remediate biases?	Resampling Data augmentation	Regularization Multi-objective optimization

- **Transformer**
  - Fundamentals: Input embedding, positional encoding, self-attention mechanism, multi-head attention, masked multi-head attention, layer normalization, residual connection, output probabilities, sampling, sources of variability in generation.

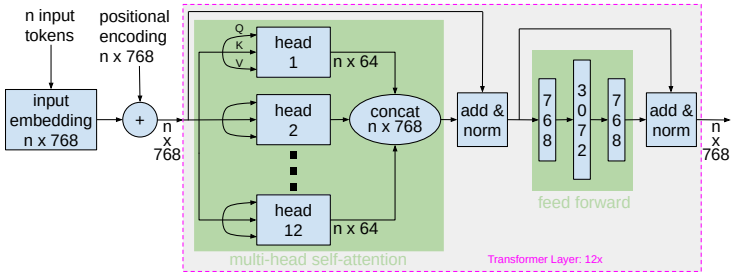


$$Attention(Q, K, V) = Softmax\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) V$$

  - Q: how important a word is in the sentence? K: how each word is related to the Q word? V: represents the context and significance of a word; it helps the model to understand and interpret the sentence. Analogy: **Q=Glasses, K=Named badge, V=Summary note**.
  - Limitations (remedy): memory-intensive (weight quantization), resource-intensive (knowledge distillation), scalability of self-attention (use sparse attention patterns)
  - Handling multimodal inputs: Joint embedding and then general self-attention; dedicated encoding layers for each modality and combine their representations; cross-modal attention mechanism.

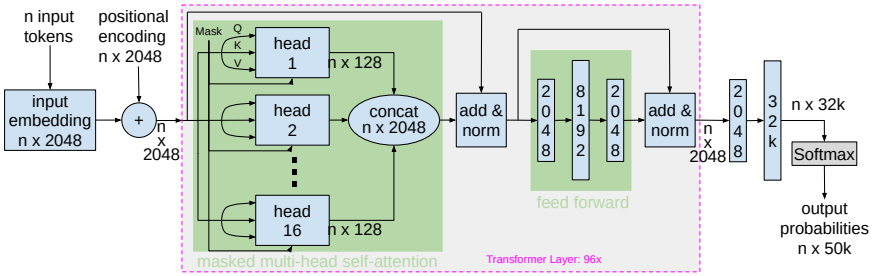
$$\text{CrossModalAttention}\left(\text{Q}_{\text{text}}, \text{K}_{\text{image}}, \text{V}_{\text{image}}\right) = \text{Softmax}\left(\frac{\text{Q}_{\text{text}} \cdot \text{K}_{\text{image}}^{\text{T}}}{\sqrt{\text{d}_{\text{k,image}}}}\right)\text{V}_{\text{image}}$$

  - Cross-modal attention mechanism, dedicated encoding layers for each modality, joint embedding space.
  - Responsible AI: Bias, misinformation, privacy; enhance interpretability by attention weights visualization.
- **BERT**



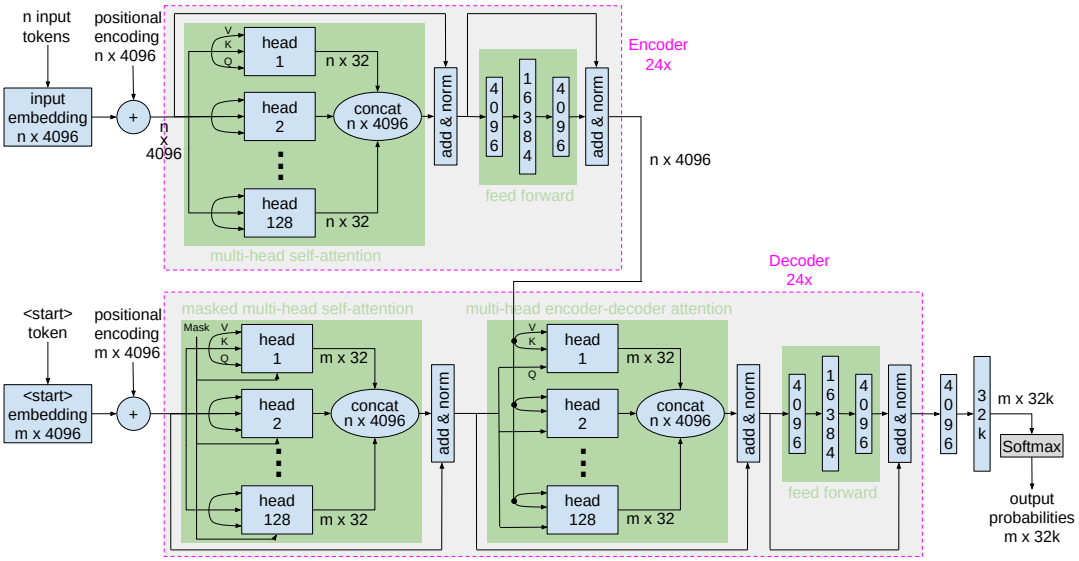
  - Uses only the encoding mechanism of transformer and is suitable for tasks related to understand context.
  - Inputs (max 512 embeddings for base and 1024 embeddings for large model)
    - [CLS] + (n-1) tokens → input embeddings + position embeddings
  - Training
    - Multi-task learning -> Task 1: Masked Language Model (MLM, 15% of words are masked), and Task 2: Next Sentence Prediction (NSP, 50% of the second sentences are actual second sentences and the remaining 50% of the second sentences are randomly chosen from corpus).
    - Two pre-trained versions (L = #attention layer, A = #heads, H=hidden layer size)
      - BASE (L=12, A=12, H=768; Total Parameters=110M)
      - LARGE (L=24, A=16, H=1024, Total Parameters=340M)

- **GPT-2**



  - Uses only the decoding mechanism with only masked multi-head attention and is suitable for text generation tasks.
  - Input tokens -> embeddings -> decoder block (multi-head self-attention, position-wise FFNN with layer norm and residual connections) -> Linear -> Softmax -> Probabilities of all possible words in dictionary usually 30k-50k words and pick the one with the highest probability or randomly pick one from a set of words with probabilities > threshold.
  - After generating the first output tokens, it is used with the original input tokens and generates the second output token, and so on.

- **T5/BART**



- **Vision Transformers**
  - **ViT vs. Swin**
    - ViT -> image patches -> transformer model -> image classification (max 197 inputs).
    - Swin Transformer -> image patches of different scales -> transformer model -> object detection / semantic segmentation (max 224 input embeddings).
  - **Oscar vs. ViLBERT**
    - Oscar -> Object-Semantics Aligned Pre-training -> image embeddings and text embeddings -> same transformer model -> cross-modal attention-> decoder -> generate text caption of the image (max 74 input embeddings)
    - ViLBERT -> Visual Linguistic BERT -> image embeddings and text embeddings -> two-stream transformer models -> cross-modal attention -> decoder -> generate text caption of the image (max 128 input embeddings)

General ML Concepts

- MLE vs. MAP: MLE seeks for params that maximize the likelihood $\mathbf{L}(\boldsymbol{\theta}|\mathbf{x})$ (or equivalently the probability $\mathbf{P}(\mathbf{x}|\boldsymbol{\theta})$), while MAP seek for params that maximize posterior probability: $\mathbf{P}(\boldsymbol{\theta}|\mathbf{x}) \propto \mathbf{L}(\boldsymbol{\theta}|\mathbf{x}) * \mathbf{P}(\boldsymbol{\theta})$.

$$\widehat{\boldsymbol{\theta}}_{\mathbf{MLE}} = \underset{\boldsymbol{\theta}}{\mathbf{argmax}}\, \mathbf{L}(\boldsymbol{\theta}|\mathbf{x})$$

$$\widehat{\boldsymbol{\theta}}_{\mathbf{MAP}} = \underset{\boldsymbol{\theta}}{\mathbf{argmax}}\, \mathbf{L}(\boldsymbol{\theta}|\mathbf{x}) * \mathbf{P}(\boldsymbol{\theta})$$

  - Both MLE and MAP are used for parameters estimation in probabilistic models. The key difference lies in how they treat prior knowledge. MLE does not consider any prior distribution over the parameters (purely data-driven), while MAP incorporates a prior, aligning with Bayesian principles.
- MSE vs. Cross-Entropy Loss
  - MSE loss is for linear regression and cross-entropy loss is for logistic regression.
    - Linear regression assumes a normal distribution of error, and MSE aligns with this assumption, while logistic regression assumes a binomial distribution of the output, and cross-entropy aligns with this assumption.
    - Both MSE and CE are convex functions. Second derivative of convex functions is positive.
      - $\mathrm{MSE} \approx (\mathrm{y} - \mathrm{XW})^2 \rightarrow \nabla^2 = \mathrm{X}^2$
      - $\mathrm{CE} = -\mathrm{y}\log\mathrm{p} - (1-\mathrm{y})\log(1-\mathrm{p}) \rightarrow \mathrm{z}(1-\mathrm{y}) + \log(1+\mathrm{e}^{-\mathrm{z}}) \rightarrow \nabla^2 = \mathrm{e}^{-\mathrm{z}}/(1+\mathrm{e}^{-\mathrm{z}})^2$
    - MSE cannot be used for logistic regression because of two reasons:
      - The assumptions do not align.
      - MSE for logistic regression becomes non-convex in nature since it involves squaring of a non-linear logistic function. $\mathrm{MSE}_{\mathrm{logistic}} \approx (\mathrm{y} - \mathrm{p})^2 \rightarrow \nabla^2 = ?$
    - MSE is equivalent to MLE for linear regression, because of normality assumption of the residuals.

$$\epsilon \sim \mathrm{N}(0, \sigma^2) \approx \mathrm{Y} \sim \mathrm{N}\left(\mathrm{X}\hat{\beta}, \sigma^2\right) = \mathrm{N}\left(\widehat{\mathrm{Y}}, \sigma^2\right)$$

$$\mathrm{lik}\,\mathrm{Y} = \prod_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{N}\left(\widehat{\mathrm{Y}}_{\mathrm{i}}, \sigma^2\right) = \prod_{\mathrm{i}=1}^{\mathrm{n}} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(\mathrm{Y}_{\mathrm{i}} - \widehat{\mathrm{Y}}_{\mathrm{i}}\right)^2}{2\sigma^2}\right)$$

$$\log\mathrm{lik}\,\mathrm{Y} = \mathrm{n} * \log\frac{1}{\sqrt{2\pi\sigma^2}} - \sum_{\mathrm{i}=1}^{\mathrm{n}} \frac{\left(\mathrm{Y}_{\mathrm{i}} - \widehat{\mathrm{Y}}_{\mathrm{i}}\right)^2}{2\sigma^2} = \mathrm{n} * \log\frac{1}{\sqrt{2\pi\sigma^2}} - \frac{\mathrm{n}}{2\sigma^2} - \sum_{\mathrm{i}=1}^{\mathrm{n}} \left(\mathrm{Y}_{\mathrm{i}} - \widehat{\mathrm{Y}}_{\mathrm{i}}\right)^2$$

$$\max\log\mathrm{lik}\,\mathrm{Y} \approx \min \sum_{\mathrm{i}=1}^{\mathrm{n}} \left(\mathrm{Y}_{\mathrm{i}} - \widehat{\mathrm{Y}}_{\mathrm{i}}\right)^2$$

- LR vs. LR (Linear vs. Logistic)
  - Linear -> Regression -> continuous output -> OLS -> residuals are normally distributed
  - Logistic -> Classification -> binomial output -> MLE -> residuals may not normally distribute
- Logistic Regression vs. SVM
  - SVM tries to maximize the separation between the margins, while LR tries to maximize the log-likelihood.
  - Outliers can shift the decision boundary of LR, while only the support vectors determine the decision boundary of SVM; other data points do not matter in finding the decision boundary.
- NB vs. LDA: NB assumes identity covariance matrices for each class, i.e., features are independent, which is Naïve assumption, LD assumes same covariance matrices for each class.
- PCA vs. MDS
  - PCA preserve covariance of the data matrix, while MDS preserve pair-wise distance between the data.
- Feature selection process: Correlation analysis; Stepwise methods (forward or backward) using AIC/BIC; Lasso regression; Dimensionality reduction like PCA and SVD

REGRESSION

Linear regression:

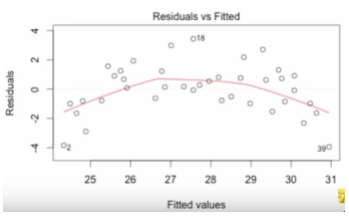
Definition: $Y = X\beta + \epsilon$, OLS solution by setting derivative of RSS to zero: $\hat{\beta} = (X^T X)^{-1} X^T y$

$$RSS = (y - X\beta)^T (y - X\beta) = y^T y - y^T X\beta - \beta^T X^T y + \beta^T X^T X\beta = y^T y - 2\beta^T X^T y + \beta^T X^T X\beta \rightarrow \nabla = -2X^T y + 2\beta X^T X$$

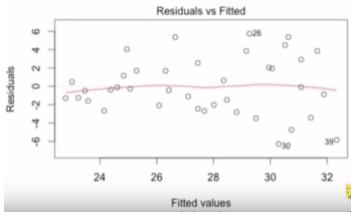
- Assumptions for linear regression:
  - Response variable is linearly related to the predictor variables. Plot normalized residuals vs. predictors. If this assumption is violated, then try to transform the data (either Y or X or both) and use them in the model. Common transformation may include log (for positive data!), X^2 etc.
  - Samples are independent.
  - Predictors are independent (no multi-co-linearity).
    - Rule of thumb: VIF > 10 reflects high co-linearity, cutoff VIF = 5.

$$VIF_i = \frac{1}{1 - R_i^2}, \quad \text{where,} \quad R_1^2 \text{ is for } X_1 = \beta_2 X_2 + \cdots + \beta_n X_n + \epsilon$$

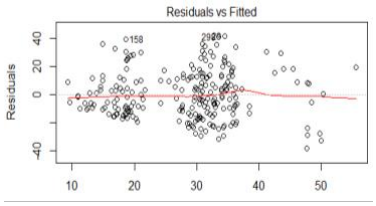
    - We can remove collinear predictors or use Ridge penalty, reduce dimensions.
  - Variances of errors are constant for all X's. Plot residuals vs. fitted values. If this assumption is not violated, then the residuals should approximately equal for all fitted values.
  - Residuals are normally distributed. Observe QQ plot or histogram of residuals to test the normality. Also k-s test or shapro-wilk test can be used.
  - For a fixed value of x, e.g., x=0, the outputs are normally distributed.
- All the above assumptions (except the linearity one) are also the assumptions for ANOVA.



Linearity assumption is violated.



Equal variance assumption is violated.

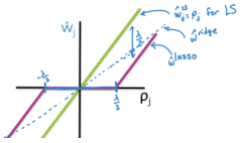


Clusters indicated multi-collinearity!

Types of linear regression:

- Simple: $Y = X\beta + \epsilon$ with p = 1 and multiple: $Y = X\beta + \epsilon$ with p > 1, polynomial: $Y = X\beta + \epsilon$ with x's powers.
- Stepwise: Forward stepwise and backward stepwise (feature selection, $O(D^2)$ << $O(2^D)$ for all subset method)
- Ridge: Cost with L2 penalty: $C = (y - \hat{y})^2 + \lambda |w|^2$ -- Good for prediction
  - Optimization using gradient descent: $w_j = w_j - \alpha \frac{dC}{dw_j}$, α is the learning rate.
- Lasso: Cost with L1 penalty: $C = (y - \hat{y})^2 + \lambda |w|$ -- Good for feature selection
  - If ρ_j be the derivative of RSS without j^{th} predictor (or coordinate), Optimization using coordinate descent:

$$w_j = \begin{cases} \rho_j + \frac{\lambda}{2}, & \rho_j < -\frac{\lambda}{2} \\ 0, & \text{otherwise} \\ \rho_j - \frac{\lambda}{2}, & \rho_j > \frac{\lambda}{2} \end{cases}$$



K-Nearest Neighbor (kNN)

- Given x_{test}, find x_{train} that are closer to x_{test} (based on computed distance like eucledian, cosine, etc.). Then $y_{test} = \frac{1}{k} * (y_{train,1} + y_{train,2} + \ldots + y_{train,k})$

Weighted kNN: $y_{test} = \frac{1}{w_1 + \cdots + w_k} (w_1 * y_{train,1} + \ldots + w_k * y_{train,k})$, where $w_i = \frac{1}{distance(x_{test}, x_i)}$, or other function, which is called the "kernel".

CLASSIFICATION

- Precision: Proportion of relevant information retrieved; measures how much junk information is given to the users.
- Recall: Fraction of all relevant information found; measures how much good information we miss.
- ROC-AUC: TPR vs. FPR and PR-AUC: Precision vs. Recall.

		Predicted		
		+	-	
True	+	TP	FN (Type 2)	TPR = TP/(TP+FN) = Recall = Sensitivity
	-	FP (Type 1)	TN	FPR = FP / (FP + TN) TNR = Specificity = TN/(TN+FP)
		PPV = Precision = TP/(TP+FP)	NPV = TN/(TN+FN)	

(1) Logistic regression

- Link function for logistic regression: sigmoid, i.e.

$$y = \text{sigmoid}\left(-(w^T X + b)\right) = \frac{1}{1+\exp(-(w^T X + b))} \rightarrow w^T X + b = \log\frac{p}{1-p} = \text{logit}(p) = \text{logodds}$$

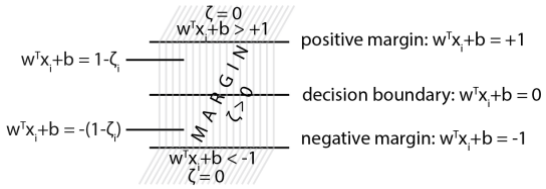
- Log-odds are linear combination of the predictors and have a nice symmetric property.
  - For example, if log-odds of occurring an even = 0.2, then the log-odds of not occurring the event is -0.2.
  - Log-odds of occurring the event = logit(p)=0.2 → p = 0.55; log-odds of not occurring the event = logit(1-p)=-0.2 → 1-p=0.45, which confirms the symmetric property of the log-odds.
- Estimate parameters (w) using MLE
- Use log-loss function: $J = -y \log A - (1-y)\log(1-p)$
- Assumptions: (1) Independent observations (2) Response variables are binomial (2) No multi-co-linearity
- Algorithm: X (mxn), Y (mx1) -> Z (mx1), A (mx1), C (mx1) -> dW (nx1), db (1x1)
  - Initialize w (nx1) and b(1x1)
  - While not converged:
    - Forward propagation: (i) Compute Z = w^T X + b (ii) Compute A = sigmoid(Z)
    - Cost, C = – Y log A – (1 – Y) log (1 – A)
    - Backward propagation:
      - $dW = \frac{\partial C}{\partial W} = \frac{\partial C}{\partial A}\frac{\partial A}{\partial Z}\frac{\partial Z}{\partial W} = \frac{A-Y}{A(1-A)} * A(1-A) * X = (A-Y)X = \text{np.dot}(X.T, A-Y)$
      - $db = \frac{\partial C}{\partial b} = \frac{\partial C}{\partial A}\frac{\partial A}{\partial Z}\frac{\partial Z}{\partial b} = A - Y = \text{np.sum}(A-Y)$
    - Update parameter: (i) w = w – learning_rate * dW (ii) b = b – learning_rate * dB

(2) Bayes classifier: Uses Bayes' Theorem to classify data points. First, compute prior and likelihood for the classes, and then assign a data point to the class whose posterior probability is the highest.

- $p(y=k|X) = \frac{\pi_k f_k(X)}{\sum_{j=1}^{K}\pi_j f_j(X)}$, where likelihood $f_k(X) = p(X|y=k)$; $f_k(X)$ is computed differently for different classifiers.
- Naïve Bayes: $f_k(X) = N(\mu_k, \sigma^2 I)$
  - Assumption: Predictors are independent from each other, i.e., identity covariance matrix for all classes
  - Linear classifier. Observations are independent.
- LDA: $f_k(X) = N(\mu_k, \Sigma_k)$: Same covariance matrix for all classes, i.e., $\Sigma_k = \Sigma, \forall k$.
- QDA: $f_k(X) = N(\mu_k, \Sigma_k)$, covariance matrix is different for different classes. $\Sigma_k = \left(x^{(k)} - \mu_k\right)^T \left(x^{(k)} - \mu_k\right)$

(3) Maximum margin classifier: SVM

- Class labels: y= $\{+1, -1\}$
- Margins: $w^T X + b \geq +1,$ and, $w^T X + b \leq -1$. Multiply each side by y, we get a single equation to represent the margin: $y(w^T X + b) \geq 1$
- Distance between two margins: $d = \frac{2}{\|w\|}$. So, for maximum margin classification: $\max d \approx \min w \approx \min \frac{1}{2}\|w\|^2$



- Hard margin: Minimize $L = \frac{1}{2}\|w\|^2$ subject to $y_i(w^T x_i + b) \geq 1$ → primal form
- Lagrange found that minimum of f(x) is found under constraint g(x)=0 when their gradients point in the same direction, i.e., $\nabla f(x) = \alpha \nabla g(x)$ or $\nabla f(x) - \alpha \nabla g(x) = 0$. Converting loss primal into dual form using Lagrange multiplier:

$$L = \frac{1}{2}\|w\|^2 - \sum \alpha_i \{y_i(w^T x_i + b) - 1\}, \qquad \frac{\partial L}{\partial w} = w - \sum \alpha_i y_i x_i = 0 \rightarrow w = \sum \alpha_i y_i x_i, \qquad \text{and } \frac{\partial L}{\partial b} = -\sum \alpha_i y_i = 0$$

$$\text{So, } L = \frac{1}{2} w^T w - \sum \alpha_i \{y_i(w^T x_i + b) - 1\} = \frac{1}{2} w^T \sum \alpha_i y_i x_i - w^T \sum \alpha_i y_i x_i - \sum \alpha_i y_i b + \sum \alpha_i = \sum \alpha_i - \frac{1}{2} w^T \sum \alpha_i y_i x_i$$

$$= \sum \alpha_i - \frac{1}{2} \sum \alpha_i y_i x_i^T \sum \alpha_j y_j x_j = \sum \alpha_i - \frac{1}{2} \sum\sum \alpha_i \alpha_j y_i y_j (x_i^T x_j)$$

  - $w = \sum \alpha_i x_i y_i$, i.e., weight is a linear combination of some inputs
  - $L = \sum \alpha_i - \frac{1}{2} \sum\sum \alpha_i \alpha_j y_i y_j (x_i^T x_j)$, i.e., optimization depends on the dot product of the inputs—kernel trick!
  - Kernels: Linear: $k(x, x') = (x^T x')^d$; Polynomial: $k(x, x') = (1 + x^T x')^d$; RBF: $k(x, x') = \exp(-\gamma \|x - x'\|^2)$
- Soft margin: Minimize $\frac{1}{2}\|w\|^2 + C\sum \zeta_i$ subject to $y_i(w^T x_i + b) \geq 1 - \zeta_i$
  - $\zeta_i > 0$ is slack variables, represent distance from margins; allow misclassification with $\zeta_i > 1$ when positive margin $y_i(w^T x_i + b) = 1 - \zeta_i$ falls below the decision boundary, vice versa for negatives.
  - $\zeta_i = \begin{cases} 1 - y_i(w^T x_i + b) & \text{if } y_i \text{ is inside MARGIN, } y_i(w^T x_i + b) < 1 \\ 0 & \text{if } y_i \text{ is outside MARGIN, } y_i(w^T x_i + b) \geq 1 \end{cases} = \underbrace{\max\left(0, 1 - y_i(w^T x_i + b)\right)}_{\text{hinge loss}}$
  - Optimization: $\frac{1}{2}\|w\|^2 + C\sum \zeta_i = \underbrace{\frac{1}{2}\|w\|^2}_{\text{L2 regularizer}} + C \sum \underbrace{\max\left(0, 1 - y_i(w^T x_i + b)\right)}_{\text{hinge loss}}$
  - Same math for soft margin for w and L, but $0 < \alpha_i < C$.

(4) Trees: decision trees

- Which feature to split? Split based on the feature that gives minimum Gini impurity and maximum information gain.

$$GI = 1 - \sum_i p_i^2, \qquad IG = -\sum_i p_i \log p_i$$

  - Example: Play (yes/no) golf if the wind is high/low?
    - High (yes: 2; no: 3) → $GI = 1 - P(\text{yes})^2 - P(\text{no})^2 = 1 - \left(\frac{2}{5}\right)^2 - \left(\frac{3}{5}\right)^2 = 1 - \frac{4}{25} - \frac{9}{25} = \frac{12}{25}$
    - Low (yes: 6, no: 3) → $GI = 1 - P(\text{yes})^2 - P(\text{no})^2 = 1 - \left(\frac{6}{9}\right)^2 - \left(\frac{3}{9}\right)^2 = 1 - \frac{36}{81} - \frac{9}{81} = \frac{36}{81}$
    - GI for wind: 12/25 * (5/(5+9)) + 36/81 * (9/(5+9))
- Early stopping criteria (hyper-parameter optimization):
  - Max depth, max leaf nodes, minimum #sample to split a node, max #features to split, GI/IG tolerance etc.
  - Pruning (cost = misclassification error + lambda * #leaf nodes)

(5) Ensembles

- Bagging: Use bootstrapped observations (i.e., with replacement) to create a tree (split on 'best' features); create n-trees by repeating; train all trees in parallel.
    - Concepts:
        - Parallel training, computationally less expensive, easier to scale; Preferred when the model is complex, and we are concerned about overfitting problem.
        - Uses fully grown trees (individual tree has low bias & high variance) -> tries to reduce variance by adding predictors without affecting the bias (theoretically); for P predictors mu=mu and var=var/P, i.e., the mean of aggregated predictions remains same as the mean of individual models, while the variance of aggregated predictions is reduced by a factor of P, the number of models.
        - Given $x_1, x_2, \dots, x_p$ as predictions from p different models, the aggregated prediction x is the average of these predictions: $x = \frac{x_1+x_2+\cdots+x_p}{p}$
        - $\text{mean}(x) = E[x] = \frac{E[x_1]+E[x_2]\dots+E[x_p]}{P} = P * \frac{\mu}{P} = \mu$
        - $\text{var}(x) = E[(x-\mu)^2] = E\left[\left(\frac{x_1+x_2\dots+x_p}{P} - \mu\right)^2\right] = \frac{1}{P^2} * E\left[\left((x_1-\mu)+(x_2-\mu)\dots+(x_p-\mu)\right)^2\right] \rightarrow$ apply iid assumption $\rightarrow \text{var}(x) = \frac{1}{P^2} * E\left[(x_1-\mu)^2+(x_2-\mu)^2\dots+(x_p-\mu)^2\right] = \frac{1}{P^2} * P * \sigma^2 = \sigma^2/P$
    - Two commonly used variants of bagging methods:
        - Random forest (RF): Use bootstrapped observations and randomly select several features and split on the 'best' feature (from the selected ones); repeat for n-trees.
        - Extremely RF: Bootstrapped observations, select some random features, split a random feature.
- Boosting: Train the first model, increase the weights to the observations that are incorrectly predicted by the first model, and then train the second model, and so on.
    - Concept:
        - Sequential training, computationally expensive, hard to scale; Preferred when the model is simple, and we are concerned about underfitting problem.
        - Uses shallow trees (individual tree has high bias & low variance) -> tries to reduce bias by adding predictors into the ensemble without affecting the variance (theoretically); variance increases after adding certain number of trees/models.
    - Two commonly used boosting methods:
        - Gradient boosting
            - Initialize the prediction with a constant, typically with the mean of the output.
            - For estimator i = 1, 2, …, T
                - If i=1: Train estimator i with the original data
                - Else: Train estimator i with the residuals from estimator [1, i-1] (residuals, so more useful for regression), combine all estimators to reduce error (e.g., MSE).
        - AdaBoost:
            - Start with equal weights (α) to all observations, i.e. $\alpha_i = 1/n$
            - For estimator i = 1, 2, …, T
                - Train estimator i and compute its coefficient (using weighted error)
                - Re-compute the weights of the observations, i.e., increase the weights to misclassified observations, (misclassification, so more useful in classification)
- Stacking: Train k estimators and use the outputs of the k estimator as the input to another estimator.

CLUSTERING

(1) K-means

- Uses a partitioning algorithm; Needs to predefine the number of clusters; Hard assignment.
- Assumption: Clusters are spherical and isotropic (uniform in all directions), and all clusters have same variance.
- Distance metric: Euclidean
- Method: Initialize the center (mean) of the clusters. Compute the distance of each observation from the clusters and assign the observation to its nearest cluster. Update the means of the cluster and repeat until convergence.
- May suffer from local minima problem.
- Determine the number of clusters: Prior knowledge, elbow method.

(2) Gaussian mixture model

- Uses a probabilistic algorithm; Needs to predefine the number of clusters; Soft assignment.
- Assumption: Data can be interpreted as a mixture of Gaussian distributions
- Distance metric: Mahalanobis distance.
- Method: Initialize center (mean) and shape (variance) and update them using EM algorithm
    - Prior: $\pi_c = \frac{n_c}{n}$ = prior probability that a data point falls in cluster c = number of data points in cluster c / total number of datapoints in the dataset.
    - E-step: compute soft probability or responsibility: $r_{ic} = \frac{\pi_c N(x_i;\mu_c,\Sigma_c)}{\sum \pi_k N(x_i;\mu_k,\Sigma_k)}$
    - M-step: Update μ_c and Σ_c
        - $m_c = \sum r_{ic}$ and $\pi_c = \frac{\pi_c}{m_c}$
        - $\mu_c = \frac{1}{m_c}\sum r_{ic}x_i$ and $\Sigma_c = \frac{1}{m_c}\sum r_{ic}(x_i - \mu_c)^T(x_i - \mu_c)$

(3) DBSCAN: Density-Based Spatial Clustering Application with Noise

- Uses a density-based algorithm; no need to predefine the number of clusters; arbitrary shape of clusters.
- Grouping points together in high-density regions and marking points as noise in low-density regions.
- Identifies core points (having a minimum number of other points within a given radius), border points (fewer neighbors but close to a core point), and noise (points not part of any cluster).

(4) Hierarchical clustering

- Uses agglomerative (bottom-up) vs. divisive (top-down) algorithms; No need to predefine the number of clusters.
- Assumption: Data structures maintain a hierarchy.
- Distance/Linkage metrics: Ward's, simple (minimum), complete (maximum), average
- Bottom-up method: Start with each data point as a cluster. Then merge two clusters based on the distances. Repeat the process until all data points form one cluster.

(5) Cluster validation

- Human evaluation > machine evaluation > index-based evaluation
- Machine evaluation: Shows the similarity between true labels and cluster labels (rand index).
- Index: Minimize the variance within a cluster and maximize the separation between the clusters (Dunn index).

Recommender systems

- Recommendation types
  - Content-based (focuses on item attributes)
    - Summary statistics: Recommend popular items, recommend items with average prices, etc.
    - Clustering and kNN and then recommend k products (using distance metrics (Cosine, Euclidean) or document representations (TF-IDF)
  - Collaborative Filtering (relies on item-user similarities)
    - Learn feature vectors for the products and for the users simultaneously, which are used to recommend specific products to specific users.

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	0	3	0	3	4
User 2	4	0	0	2	0
User 3	0	0	3	0	0
User 4	3	0	4	0	3
User 5	4	3	0	4	4

USER-USER SIMILARITY MATRIX

	U_1	U_2		U_j		U_{n-1}	U_n
U_1	1	Sim_{12}	...	Sim_{1j}	...		
U_2		1	...		...		
U_i		Sim_{i2}	...	Sim_{ij}	...		
U_{n-1}			...		...	1	
U_n			...		...		1

ITEM-ITEM SIMILARITY MATRIX

	I_1	I_2		I_j		I_{m-1}	I_m
I_1	1	Sim_{12}	...	Sim_{1j}	...		
I_2		1	...		...		
I_i		Simi2	...	Sim_{ij}	...		
I_{m-1}			...		...	1	
I_m			...		...		1

    - User-user-based collaborative filtering
      - Compute user-user similarity matrix; pick the most similar users (similar to target user); get the items that those users liked; and recommend those to the target user.
    - Item-item-based collaborative filtering
      - Compute item-item similarity matrix; get the most similar items. E.g., user5 already liked items 1, 4, and 5, so find out the items similar to 1, 4, and 5, and recommend the most common ones from the three sets.
    - Matrix Factorization: K = U * V => user-item matrix = user matrix * item matrix
      - K (m x n) = U (m x d) * V (d * n) with d features for users and items
      - Initialize U and V; then update U and V iteratively so that $UV \rightarrow K$.
      - Optimize the number of features (d), learning rate (α), and regularization strength (λ).

$$L = \frac{1}{2}\sum_{i,j}\left(r_{ij} - u_i^T v_j\right)^2 + \frac{\lambda}{2}\left(\sum_i \|u_i\|^2 + \|v_j\|^2\right), \qquad u_i = u_i - \alpha\frac{\partial L}{du_i}$$

  - Hybrid (combination of content-based and collaborative filtering)
- Evaluation: Measuring success of recommender systems
  - Offline: MAE, MSE, Precision, Recall, NDCG, TPR, FPR, ROC-AUC
  - Online: Click Through Rate -> Adoption and Conversion -> Sales and Revenue
- Challenges
  - Cold start problem: no or limited data available for a new user or item -> use content-based recommendation to minimize this problem.
  - Long-tail problem: many items have few or no ratings -> skewed distribution of recommendation towards popular items -> items in the long tail of distribution are overlooked -> solve using matrix factorization.
- Building a recommendation system from scratch
  - Problem/Scope -> data collection -> feature engineering -> algorithm selection -> training/validating -> deploying -> monitoring performance -> CD/CI
  - Scaling: Layered modeling (clustering, matrix factorization, deep learning) and distributed computing.

LINEAR ALGEBRA and DIMENSIONALITY REDUCTION

(1) Matrix (Fundamentals: Transpose, inverse, identity, diagonal, triangular, symmetric, trace)

- Linearly independent matrix: No columns can be written as a linear combination of the others. Columns of a matrix are linearly independent if there exists only a trivial solution to Ax = 0, i.e., x = 0 vector.
    - Rank: Number of independent columns or rows
    - Nullity: Null space is the set of all vectors x for which Ax = 0
    - Rank-Nullity Theorem: Rank(A) + Nullity(A) = Number of columns in A
- Orthogonal/orthonormal matrices: Rows and columns are orthogonal unit vectors, i.e., $AA^T = A^TA = I$.
    - Dot product between any row and any column is zero (orthogonality)
    - Dot product of any row (or column) with itself is one (orthonormality)
    - Gram-Schimdt method: Turns a set of linearly independent vectors into a set of orthogonal vectors.
- Norm: Measures size/weight/intensity of matrix in terms of the magnitudes of elements, e.g., Frobenius (L2) norm.
- Jacobian: First derivative of a multivariate function → used for system analysis, e.g., how small changes in inputs affect output); also used in optimization.

(2) Eigenvectors and eigenvalues (of square matrix)

- $\mathbf{Av = \lambda v}$ → **the vectors multiplied by A results in the same vector with a difference in scaler magnitude are called as the eigenvectors**. Eigenvectors represent the direction of linear transformation of a matrix, and eigenvalues represent the strength of the vectors in their directions.
- To transform the matrix to a certain direction, we need to form a linear combination of the eigenvectors, so the resultant vector directs to that certain direction.
- Eigenvalue decomposition theory states that any real, symmetric matrix ($\Sigma = \Sigma^T$) can be represented as $\Sigma = Q\Lambda Q^T$, where Q is the orthogonal matrix, whose columns are eigenvectors; therefore, **all eigenvectors are orthogonal to each other**. Λ is a diagonal matrix whose diagonal elements are the eigenvalues.
- **Eigenvectors of a matrix (e.g., matrix = covariance matrix of data) give us the directions of maximum variance, which is PCA**. Proof: $\lambda v = \Sigma v \rightarrow v^T\lambda v = v^T\Sigma v \rightarrow \lambda v^T v = v^T\Sigma v = \lambda = v^T\Sigma v$

$$\lambda = v^T \frac{1}{n}\sum_i (x_i - \mu)^T (x_i - \mu)\, v = \frac{1}{n}\sum_i v^T (x_i - \mu)^T (x_i - \mu) v = \frac{1}{n}\sum_i [(x_i - \mu)v]^T [(x_i - \mu)v] = \frac{1}{n}\sum_i p_i^T p_i = \sigma_p^2$$

(3) PCA

- **By projecting data onto these eigenvectors, PCA achieves dimensionality reduction while retaining the most significant variance features of the original data.**
- Assumption for PCA: Data follows a Gaussian distribution
- Eigenvectors are orthogonal, so no collinearity between the dimensions. PCA is often implemented using SVD.

(4) SVD

- $\mathbf{A = U S V^T}$ → **Principal Components (PCs) in PCA correspond to the columns of U, and singular values or diagonal values of S give the amount of variance captured by each PC.**
- Diagonal values of S = Singular values of A = sqrt(eigenvalues of AA^T or A^TA) or $\sigma = \sqrt{\lambda}$
- Columns of U = Eigenvectors of AA^T (covariance matrix of A in its row/observation space) = Left singular vectors. Form an orthonormal basis of column space of A; Rows of A are linear combination o left singular vectors.
- Columns of V = Eigenvectors of A^TA (covariance matrix of A in its col/feature space) = Right singular vectors. Forms an orthonormal basis of row space of A; Columns of A are linear combination of right singular vectors → **Columns of V are used as PCs in PCA**.
- Reduced dimension: $A_{mxd} = U_{mxd} S_{dxd}$ and data approximation: $\hat{A}_{mxn} = U_{mxd} S_{dxd} V^T_{dxn}$

(5) ICA: Data sources are independent and have non-Gaussian distributions; Tries to find and separate different sources.

PROBABILITY and STATISTICS

- Difference between probability and likelihood
  - Probability: Measures chance of event given the known parameters: P(data | parameters)
    - P(weight > 60kg | mu=0, sigma=1) → think of area under the distribution function!
  - Likelihood: Measures the plausibility of parameters given the observed data: L(parameters | data)
    - L(mu=0, sigma=1 | weight > 60kg) → think for fitting the data to a distribution!
- Independent events: Probability of occurring an event does not depend on the other events.
  - P(A or B) = P(A) + P(B) – P(A and B) → for two events
    - P(A and B) = P(A) * P(B) → for independent events
    - P(A and B) = P(A) * P(B|A) = P(B) * P(A|B) → for dependent events
    - P(A and B) = 0 → for mutually exclusive events
- Conditional Probability: Probability of an event occurring given that another event has already been occurred P(A|B)
- Bayes' Theorem: It provides a way to update our knowledge or beliefs (probability) based on prior evidence as: P(A|B) = P(B|A) * P(A) / P(B)
- Random variables: outcome of random phenomena → expressed as binomial, Poisson, and normal distributions.

Distribution	**Definition**	**Mean and Variance**	**Use Cases**
Binomial	Probability of getting exactly k successes in n trials: $P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$	$\mu = np$, $\sigma^2 = npq$	Predicting the number of defective items in batch of products
Poisson	Probability of observing k events in an interval: $P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$	$\mu = \lambda$, $\sigma^2 = \lambda$	Counting something in an interval, e.g. number of error in a page
Normal	Probability density function: $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$	$\mu = \mu$, $\sigma^2 = \sigma^2$	Modeling heights or weights of population

- Sampling methods (Random, Stratified random, cluster random, systematic)
- Statistical measures
  - Central tendency (mean, median, mode) and variability (variance, standard deviation, range, IQR)
  - Asymmetry (skewness) → right/positive skewed → mode < median < mean.
  - Peakness (kurtosis) → k=0 normal; k=3 pointy; k = –0.6 semi-circular; k = –1.2 square.
- Central Limit Theorem: Mean of means follow a normal distribution with μ = population mean, σ = population σ / √n.
- Law of Large Numbers: For large sample size, the sample statistics tend to population/expected statistics.
- Hypothesis testing: Statistical methods of evaluating two propositions (null and alternative hypothesis)
  - A/B testing: Test two versions by measuring a metric from two randomly selected groups.
- P-value and confidence interval
  - P = Probability of getting a statistic as extreme as or more extreme than the observed one given the null hypothesis is true, i.e., the likelihood of getting a data by chance given the null hypothesis is true.
  - 95% CI = If we repeat the same experiment and compute a new CI every time, then the true population statistic is expected to fall the CI for 95% cases.

- Statistical Power: The power of a statistical test is the probability that the test will reject null hypothesis when the alternative hypothesis is true. It's denoted as $1 - \beta$, where β is the probability of making a Type II error (failing to reject null hypothesis when alternative hypothesis is true).
    - Power Analysis: Method of calculating required sample size to achieve a desired significance level ($\alpha =$ the probability of making a Type I error) and effect size (magnitude of difference).
    - There's a tradeoff between α and β → increasing α decreases β, therefore increases statistical power.
    - α (Type I error) is associated with FP that impacts precision, while β (Type II error) is associated with FN that impacts recall).
- Choice of statistical tests

IN\OUT	Categorical	Continuous
Categorical	Chi-Square, ANCOVA	T-tests (smaller n), Z-test (larger n>30), ANOVA
Continuous	Logistic Regression	Regression

    - 1-sample and 2-sample tests (t-test, z-test), n-sample tests (ANOVA, ANCOVA), association tests (correlation between two qualitative (chi-square) or quantitative (regression) variables).
- Parametric and non-parametric methods
    - Parametric: Assume data follow a distribution (used to make inference about population parameters)
    - Non-parametric: No assumption

Parametric Tests	**Equivalent non-parametric tests**
T-tests	Independent: Mann-Whitney-U Dependent: Wilcoxon signed rank
ANOVA	1-way: Kruskal-Wallis 2-way: Friedman
Chi-Square	Fisher's exact test
Correlation	Spearman's rank correlation

Behavioral Questions

- Tell yourself/experience/why be hired? → Passionate, qualified, ability to contribute, change my life.
- Strength → Diversified skills/ability to work across teams/mentoring.
- Weakness → Influence, Difficulty to use top skills in product development.
- Biggest accomplishment/e2e products → Meta recommendation
- Working with cross-functional teams → C1 LLMs/data science team, MLX teams, and privacy compliance teams.
- Failed and learned → Shell oil detection
- Conflict → Replacing scan by ML with VP; Meta recommendation // asset ownership at individual/team level
- Working in a fast-paced environment → C1 sprint-by-sprint, allocate times/resources, use management tools
- Establishing standard code review systems → (works, readable, maintainable) Template, minimum 2 approvals