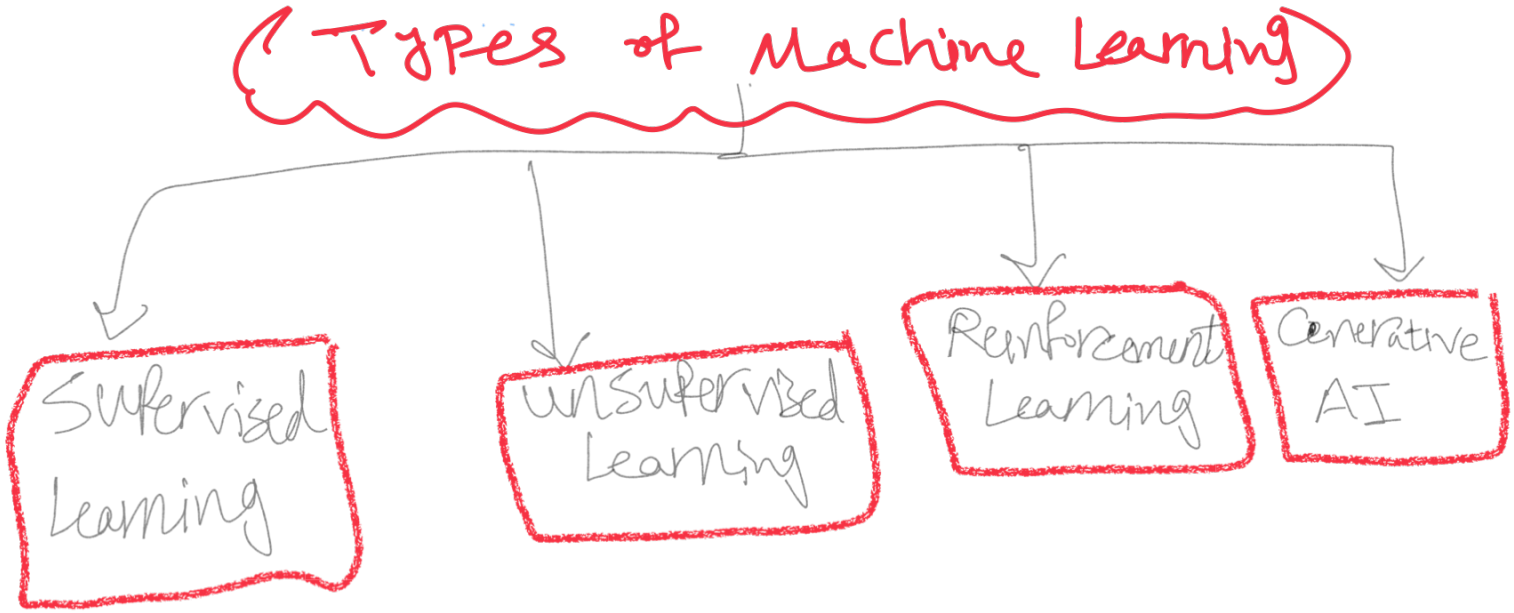
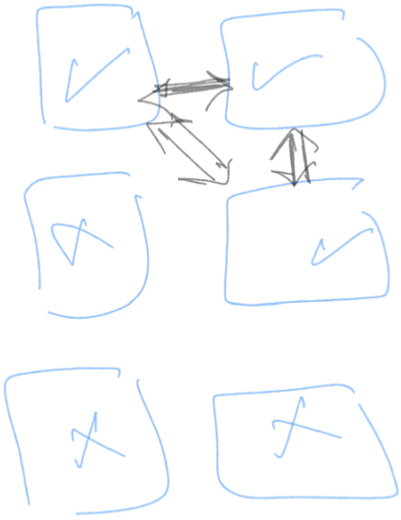
Machine Learning

Course Link: https://developers.google.com/machine-learning/crash-course/

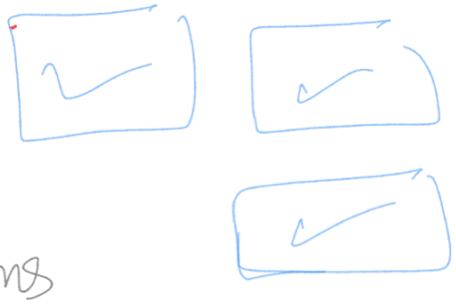


Supervised Learning:



connecting

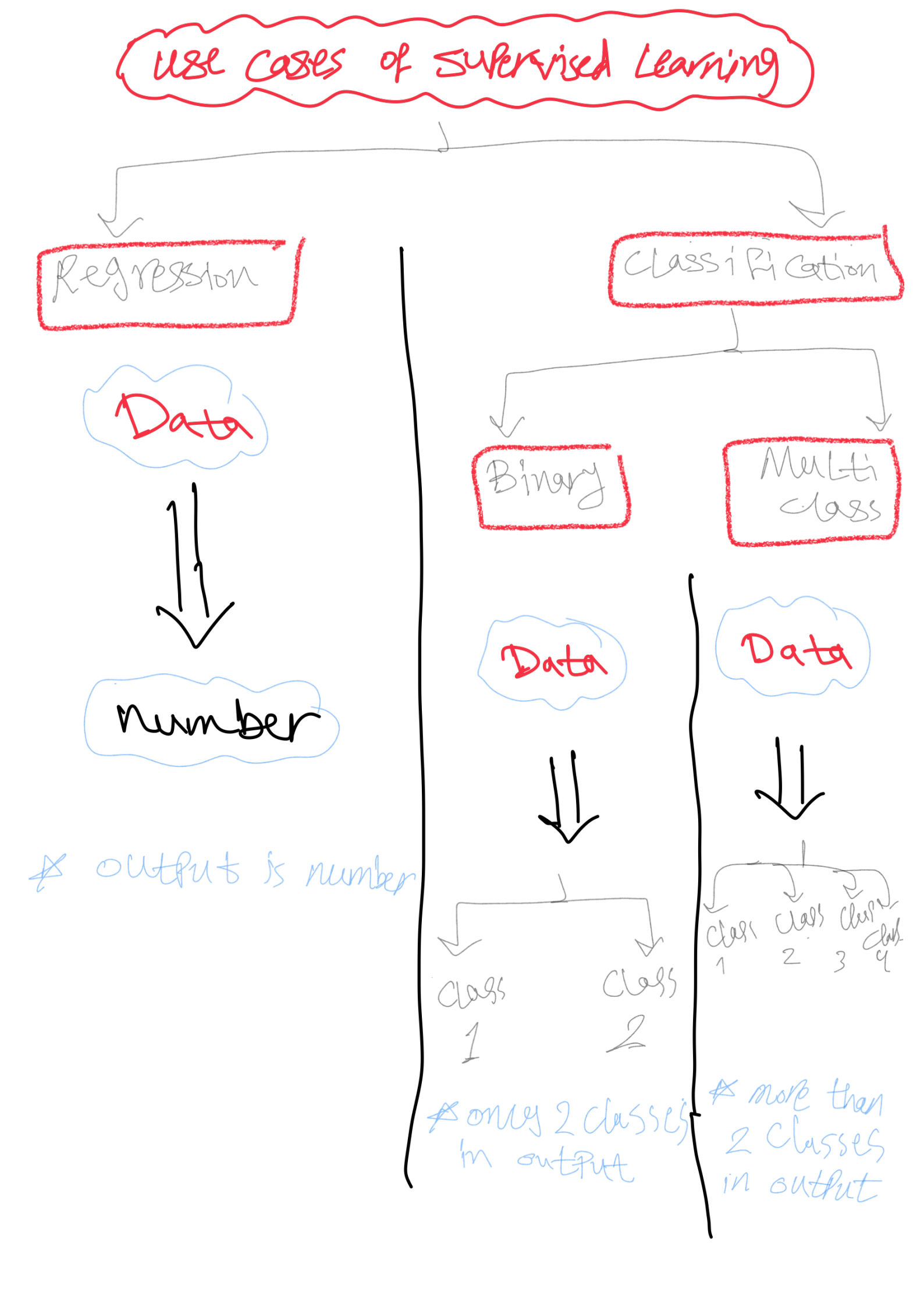
"Finding connections in the data that produce correct answer"



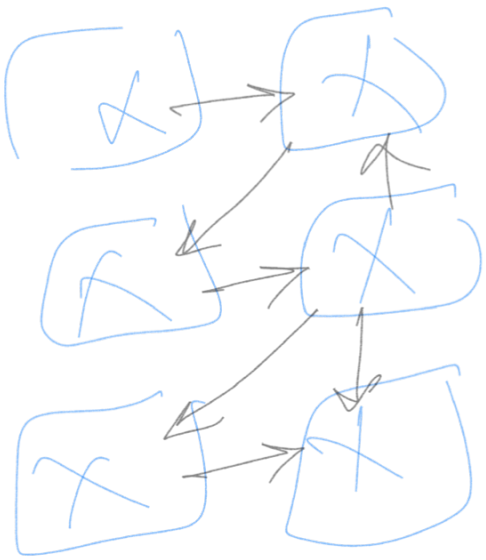
Predictions

Data

"has correct Answers"


Use cases of supervised Learning
Regression
Classification
Data
number
* output is number
Binary
Multi class
Data
Class 1
Class 2
* only 2 classes in output
Data
Class 1
Class 2
Class 3
Class 4
* more than 2 classes in output

Unsupervised Learning:

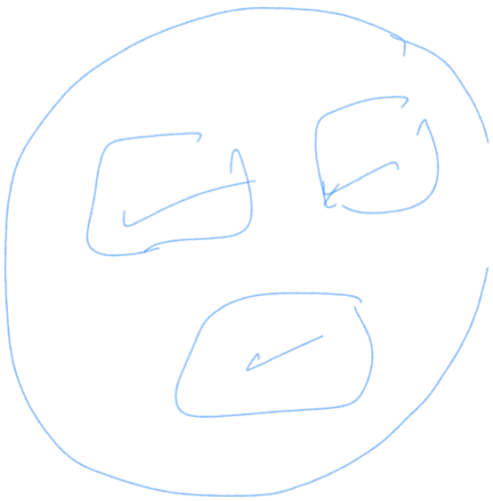


Random Data

" has no correct Answers "

clustering

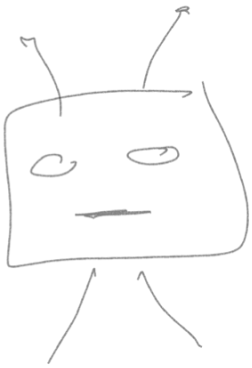
" Finding groupings in the data "



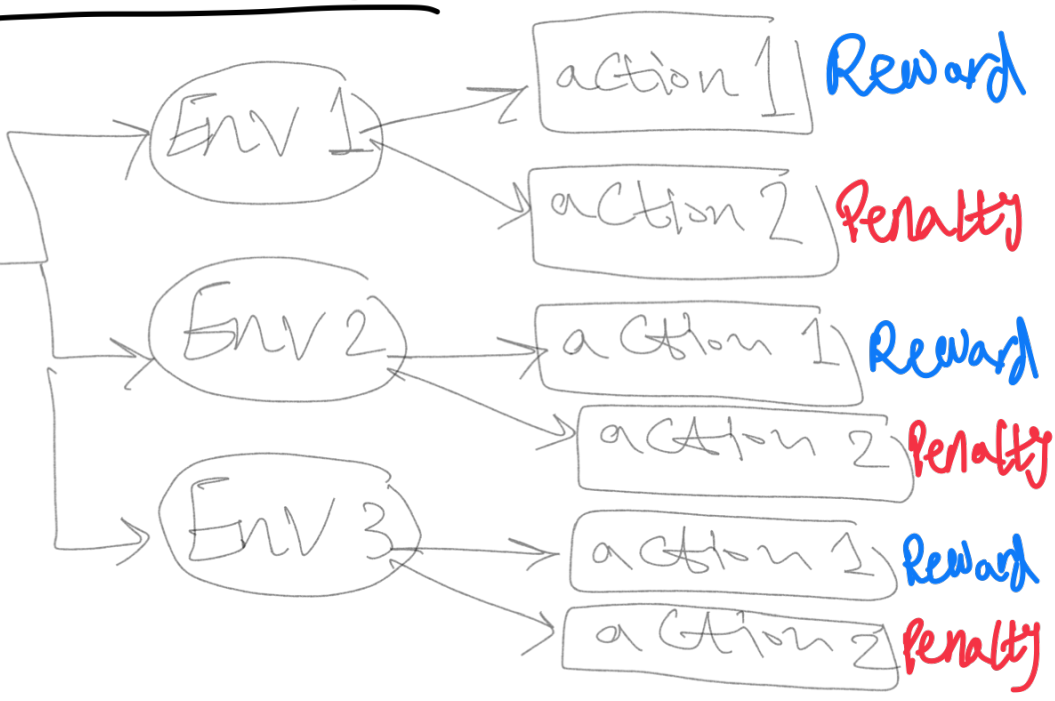
Predictions

" Defined by ML model "

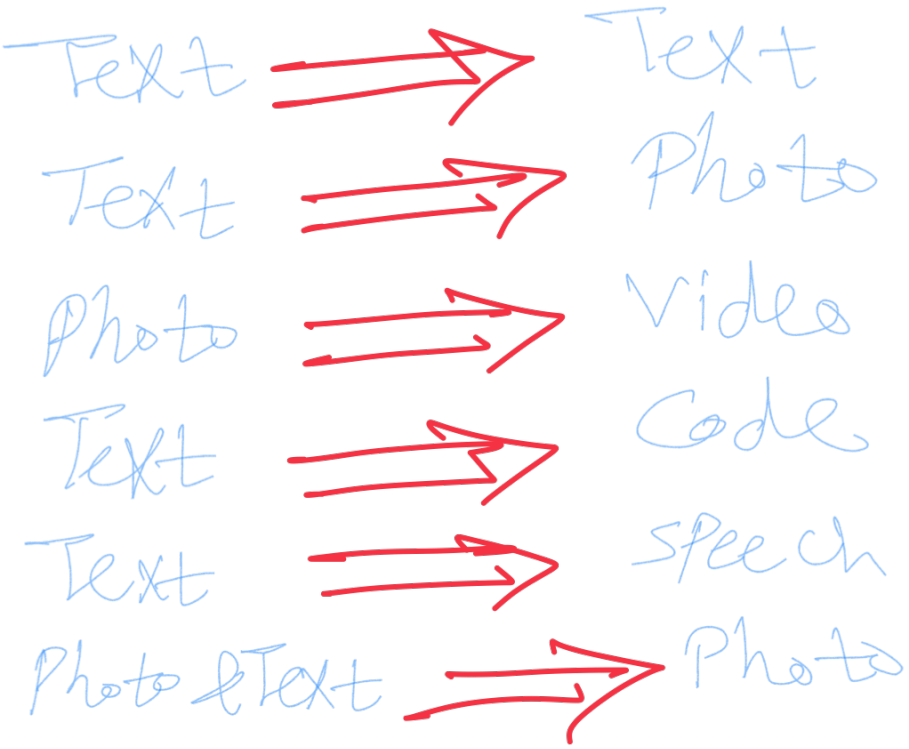
Reinforcement Learning:



Model



Generative AI :



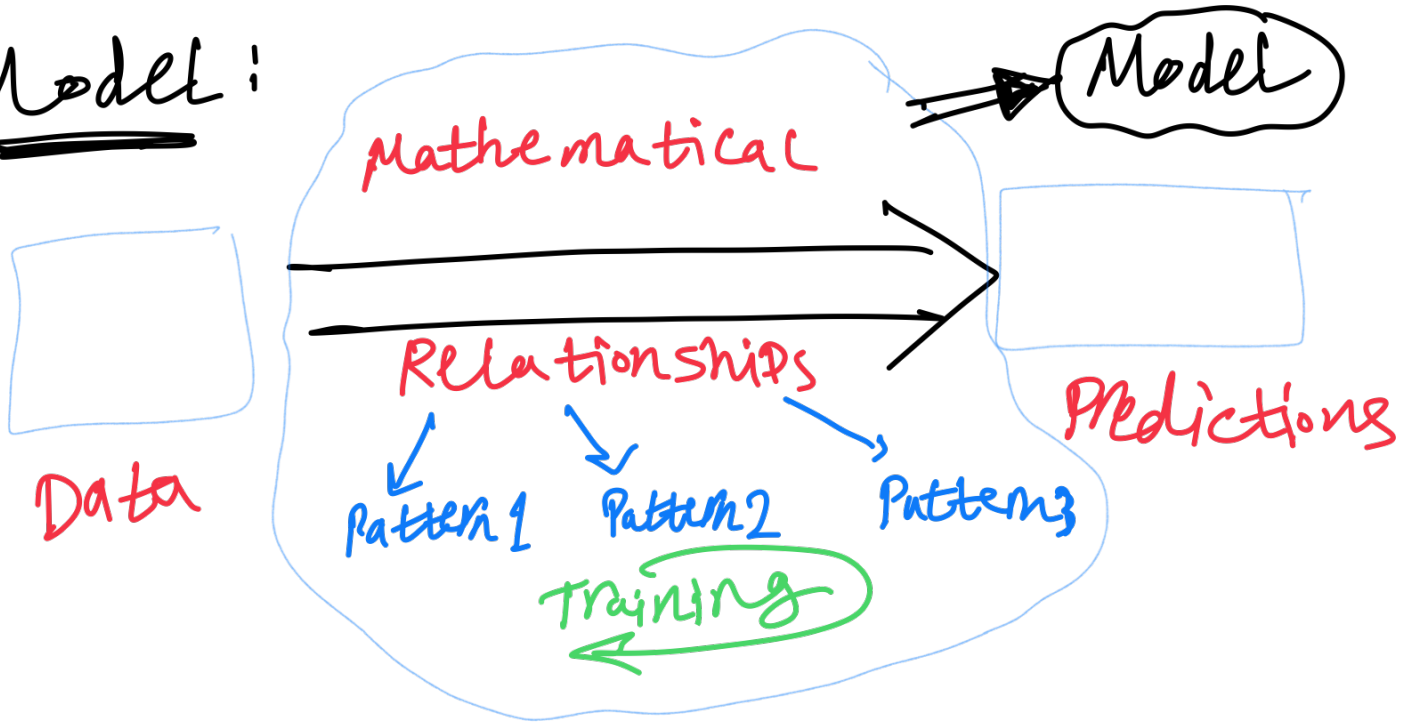
Supervised learning Concepts

Data :

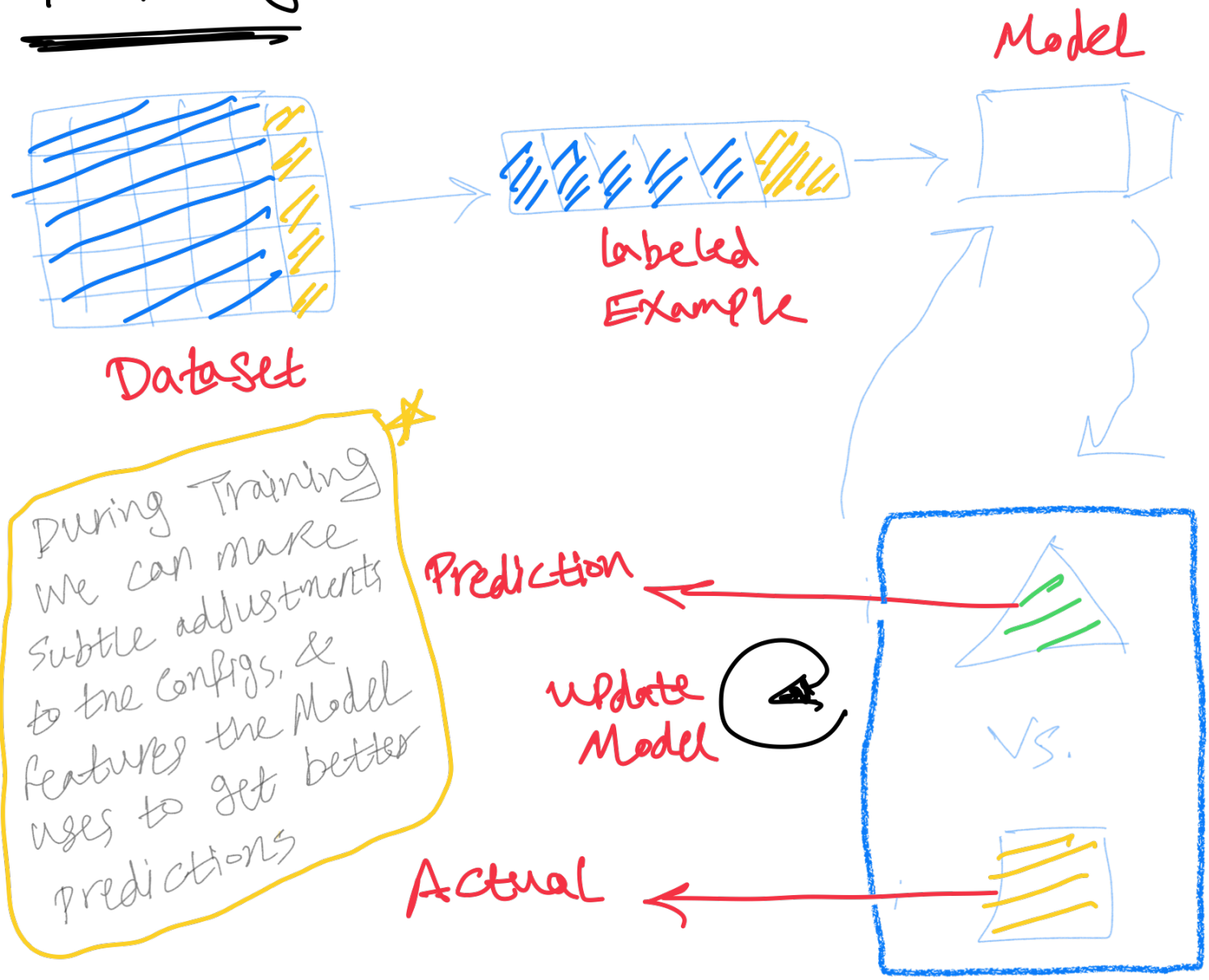
Large size Data & High Diversity

Precision Predictions

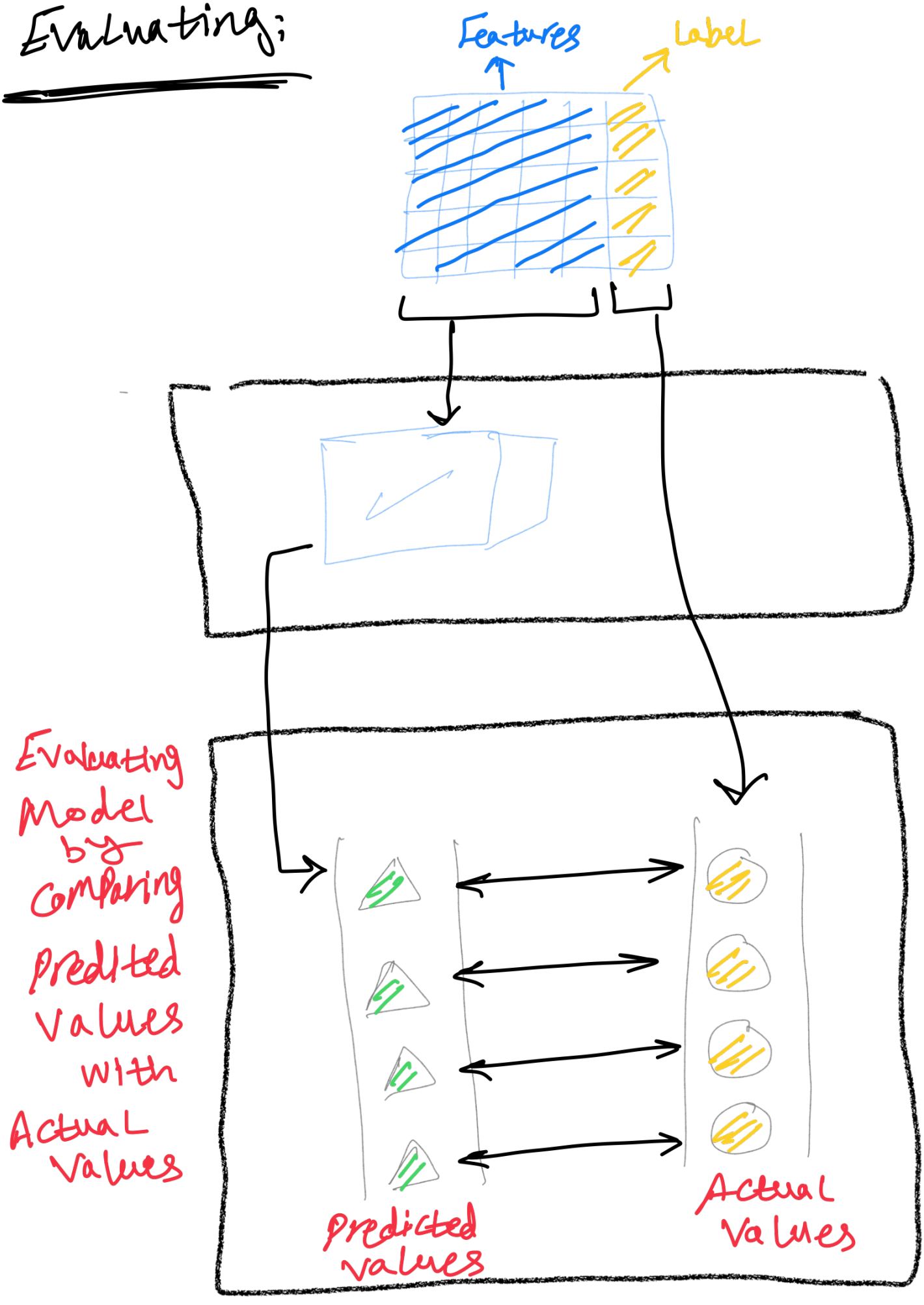
Model:



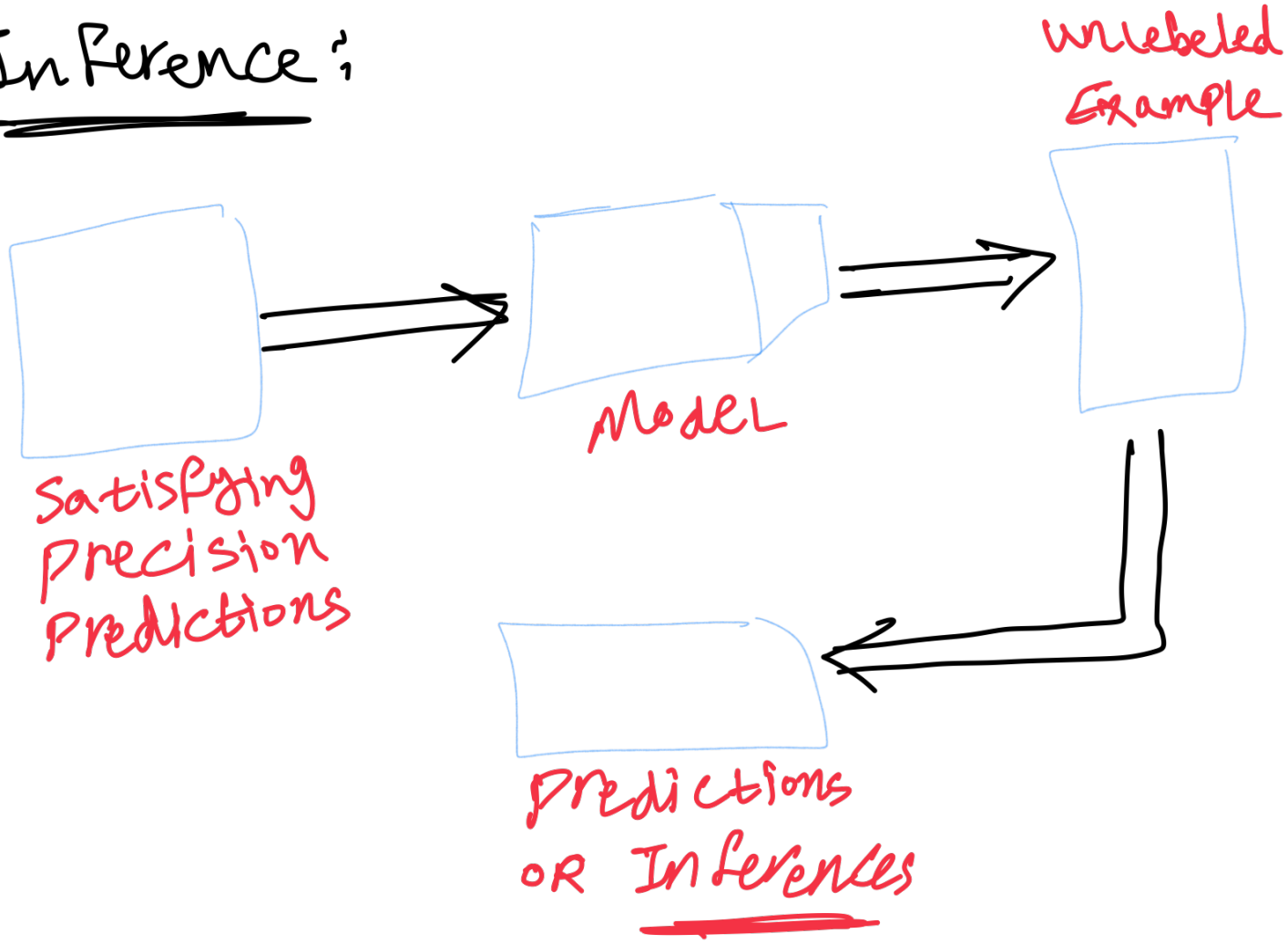
Training:

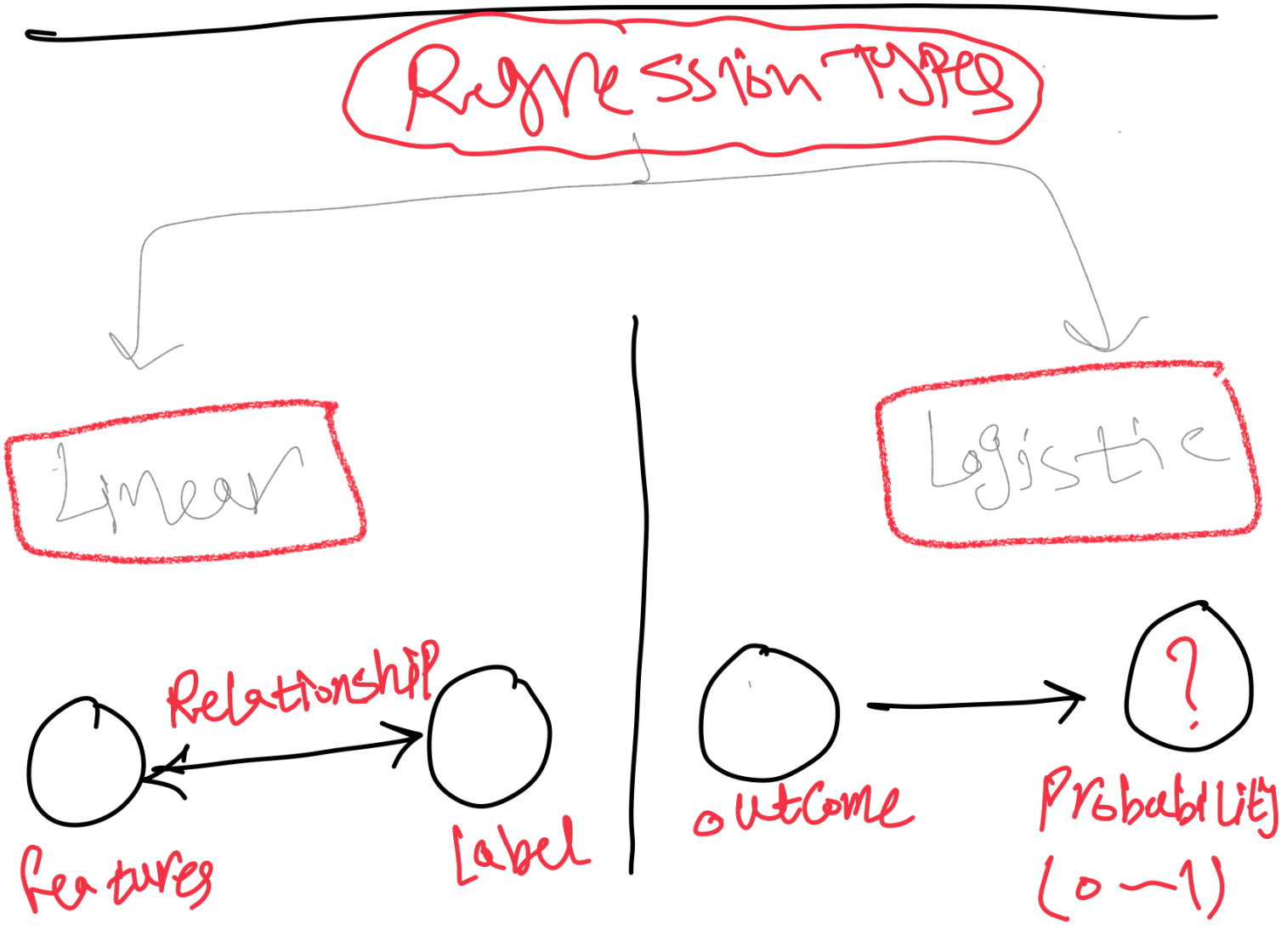


Evaluating:


Features
Label
Evaluating Model by Comparing Predicted Values with Actual Values
Predicted values
Actual Values

Inference:


Unlabeled Example
Model
Satisfying Precision Predictions
Predictions OR Inferences


Regression Types
Linear
Logistic
Relationship
Features
Label
Outcome
?
Probability (0-1)

Linear Regression:

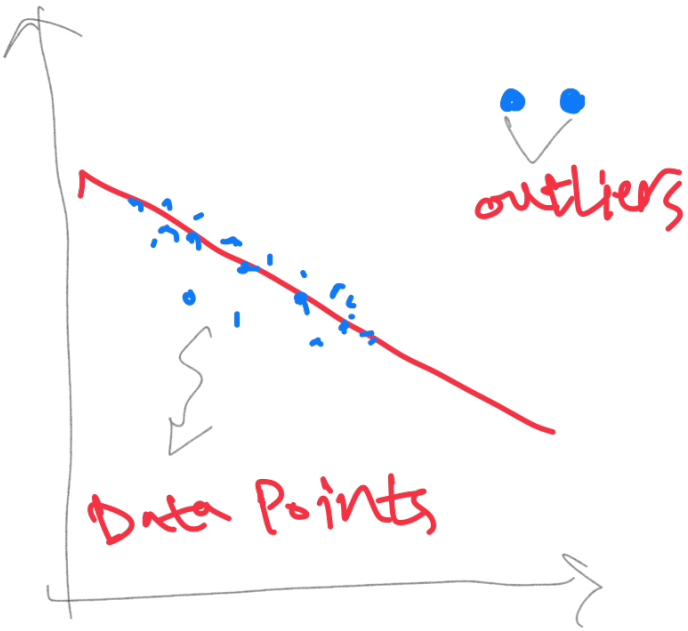
Loss: Numeric metric that describes how wrong a model's predictions are.

Types of Loss

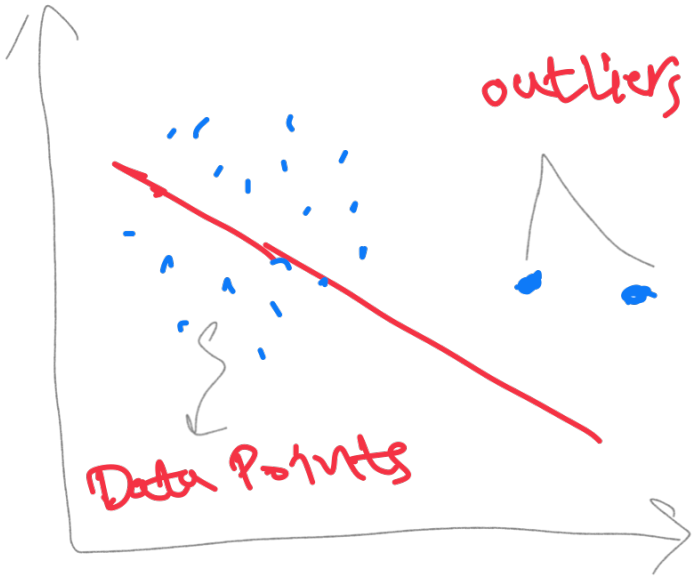
- Mean Absolute Error **MAE**
- Mean Squared Error **MSE**

MAE: The Model is further away from the outliers but closer to most of other data Points.

MSE: The Model is closer to the outliers but further away from most of other data Points.



Model trained using **MAE**



Model trained using **MSE**

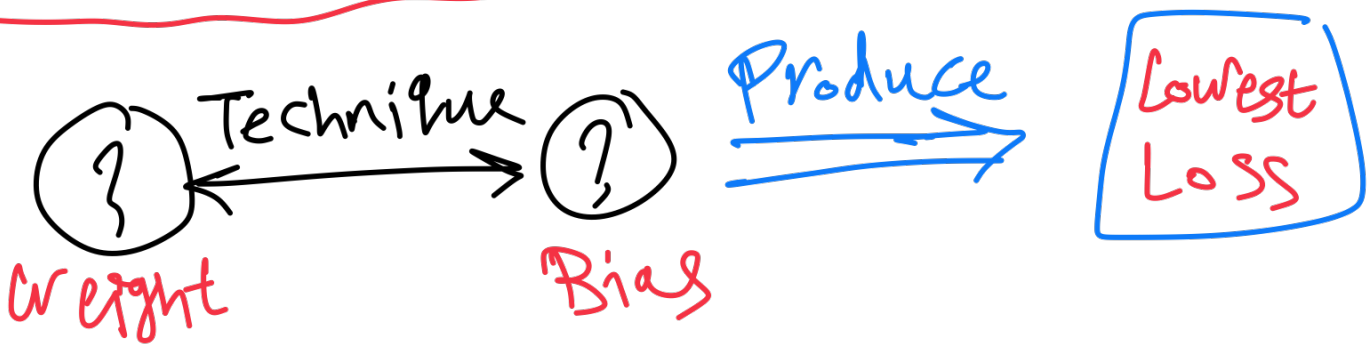
Equation:

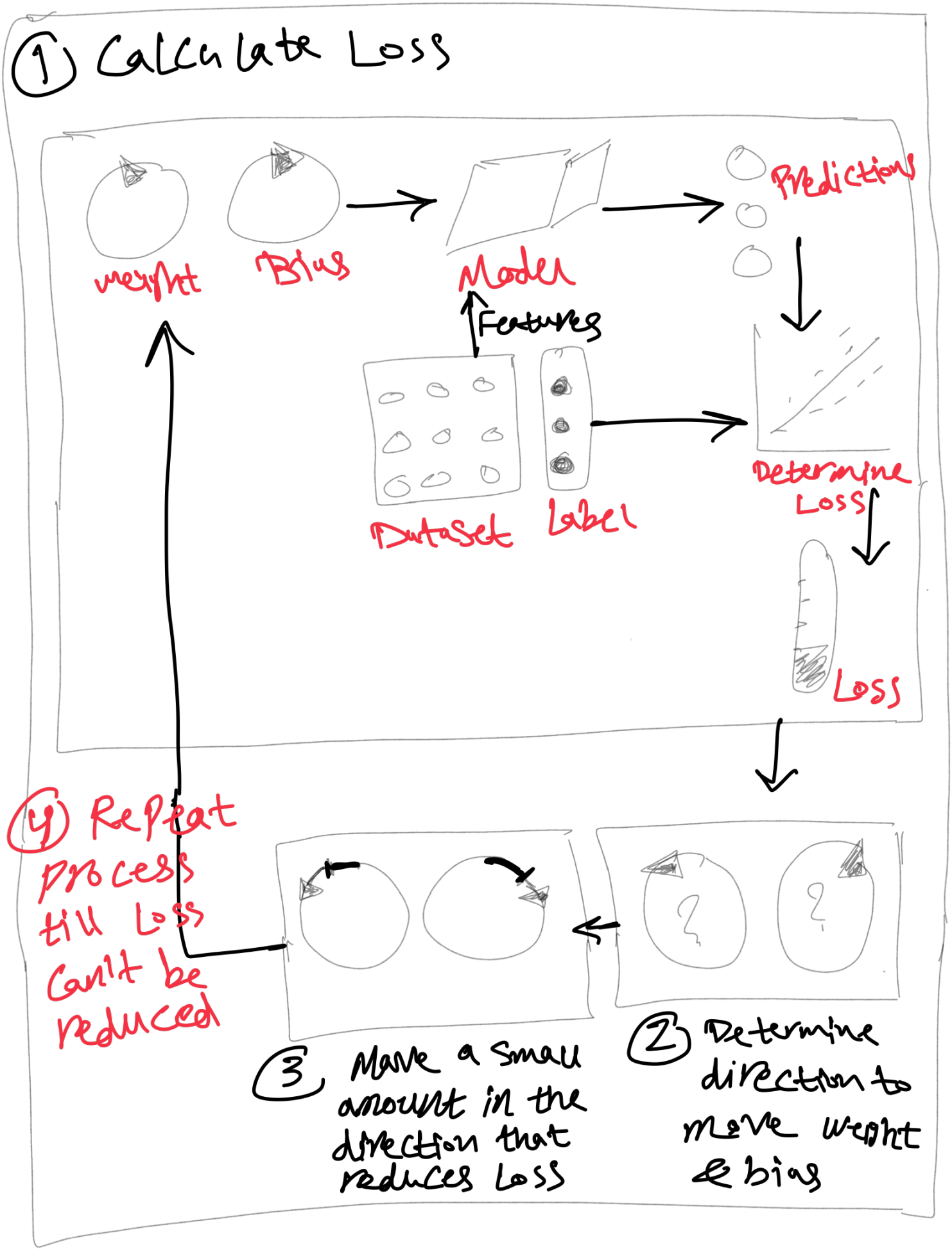
$$y' = b + w_1 x_1$$

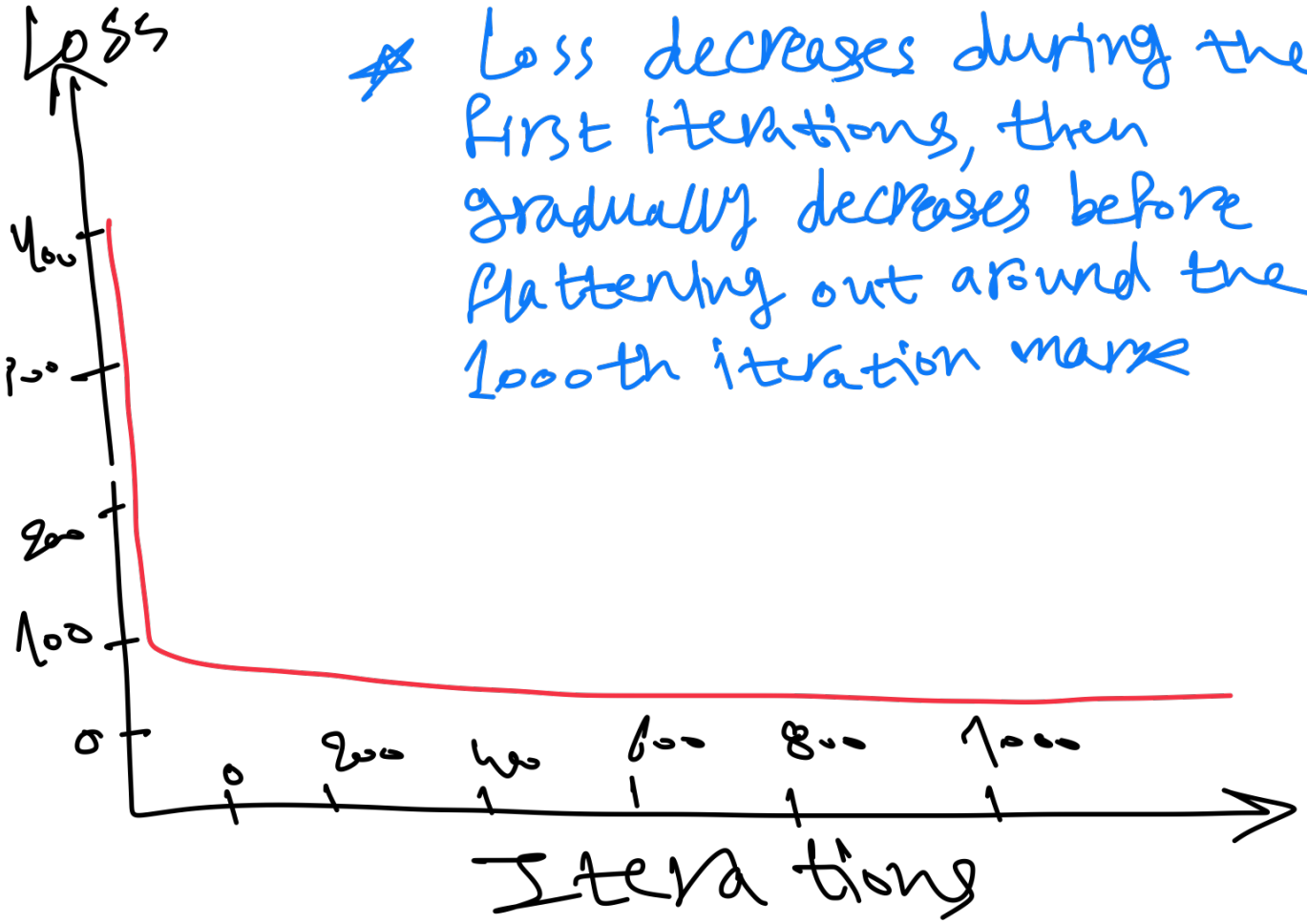
- y' — Prediction
- b — Bias
- w_1 — Weight
- x_1 — Feature value

Bias, Weight → Calculated from Training

Gradient Descent:



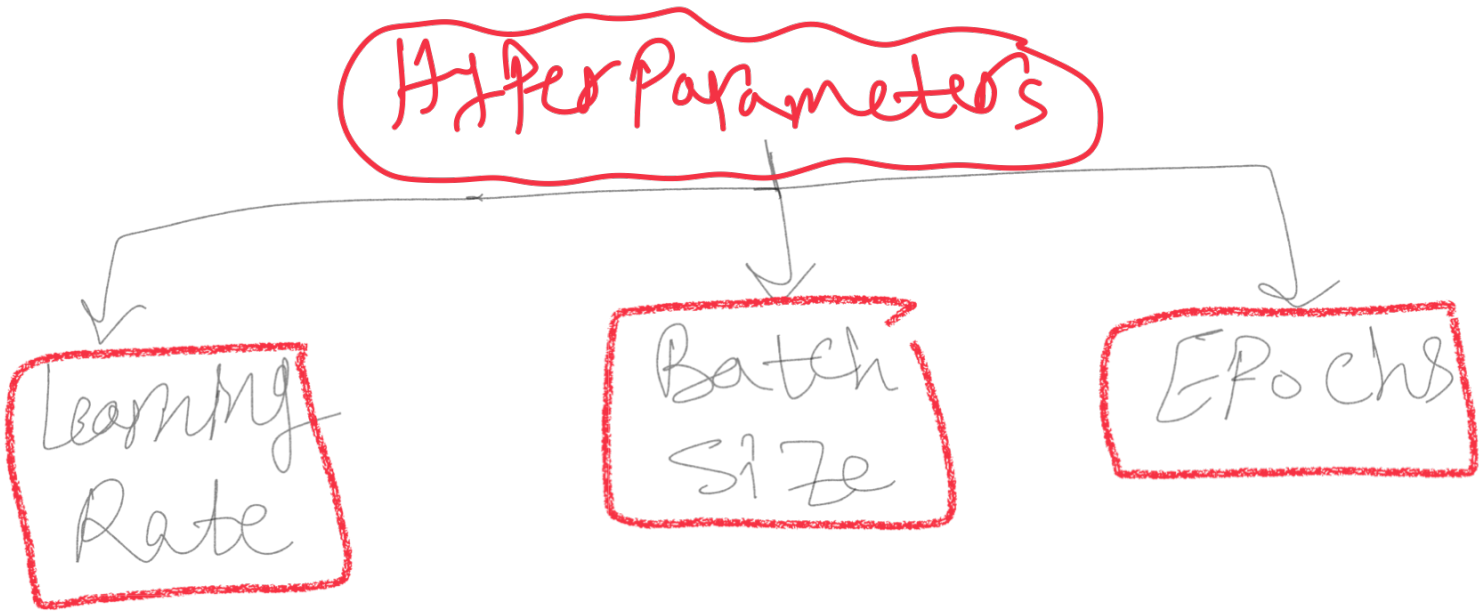

1 Calculate Loss
weight
Bias
Model
Predictions
Features
Dataset
Label
Determine Loss
Loss
4 Repeat process till Loss can't be reduced
3 Move a small amount in the direction that reduces Loss
2 Determine direction to move weight & bias



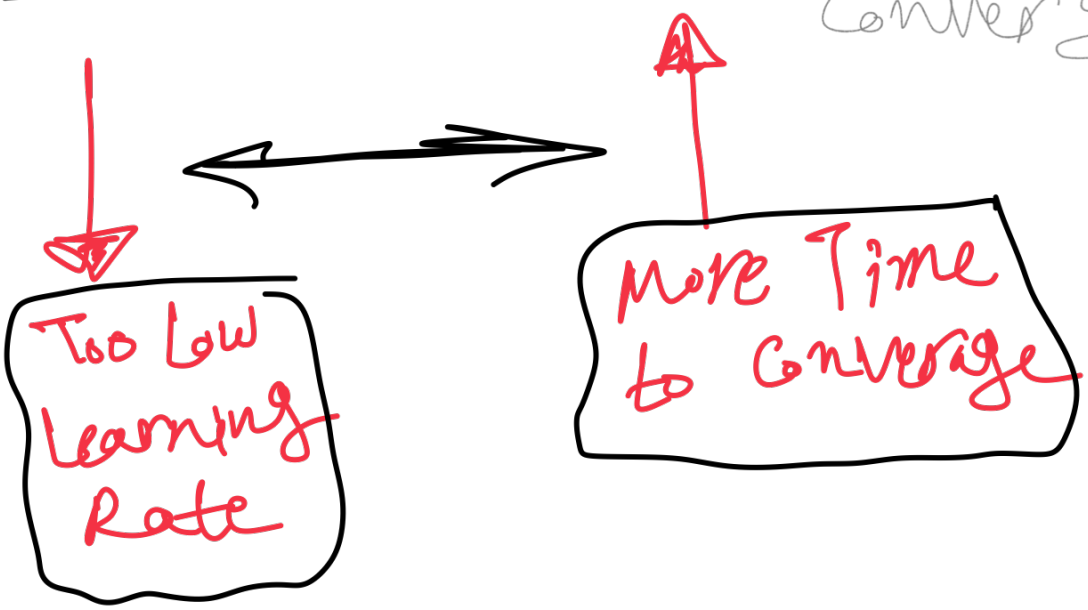
* Loss decreases during the first iterations, then gradually decreases before flattening out around the 1000th iteration mark

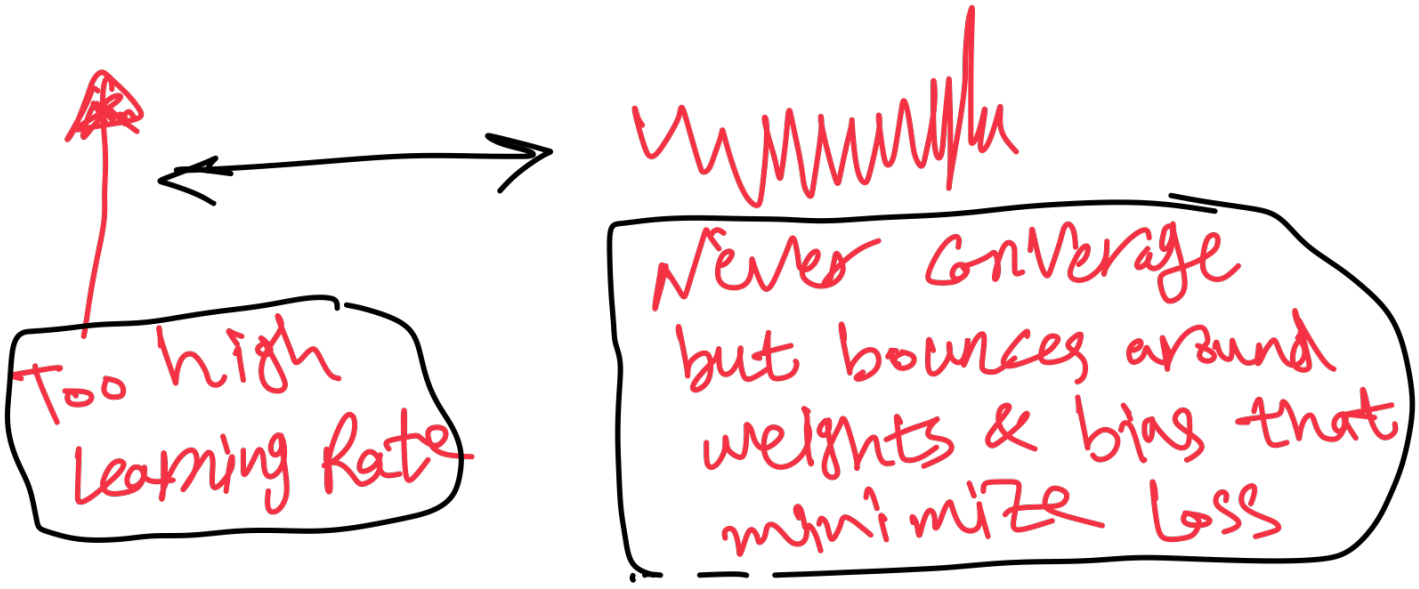
* After 1000th iterations, we can be mostly certain that the model has converged.

HyperParameters: are variables that control different aspects of training.



Learning Rate: number refers to how quickly the model converges.





Not too High
Not too low

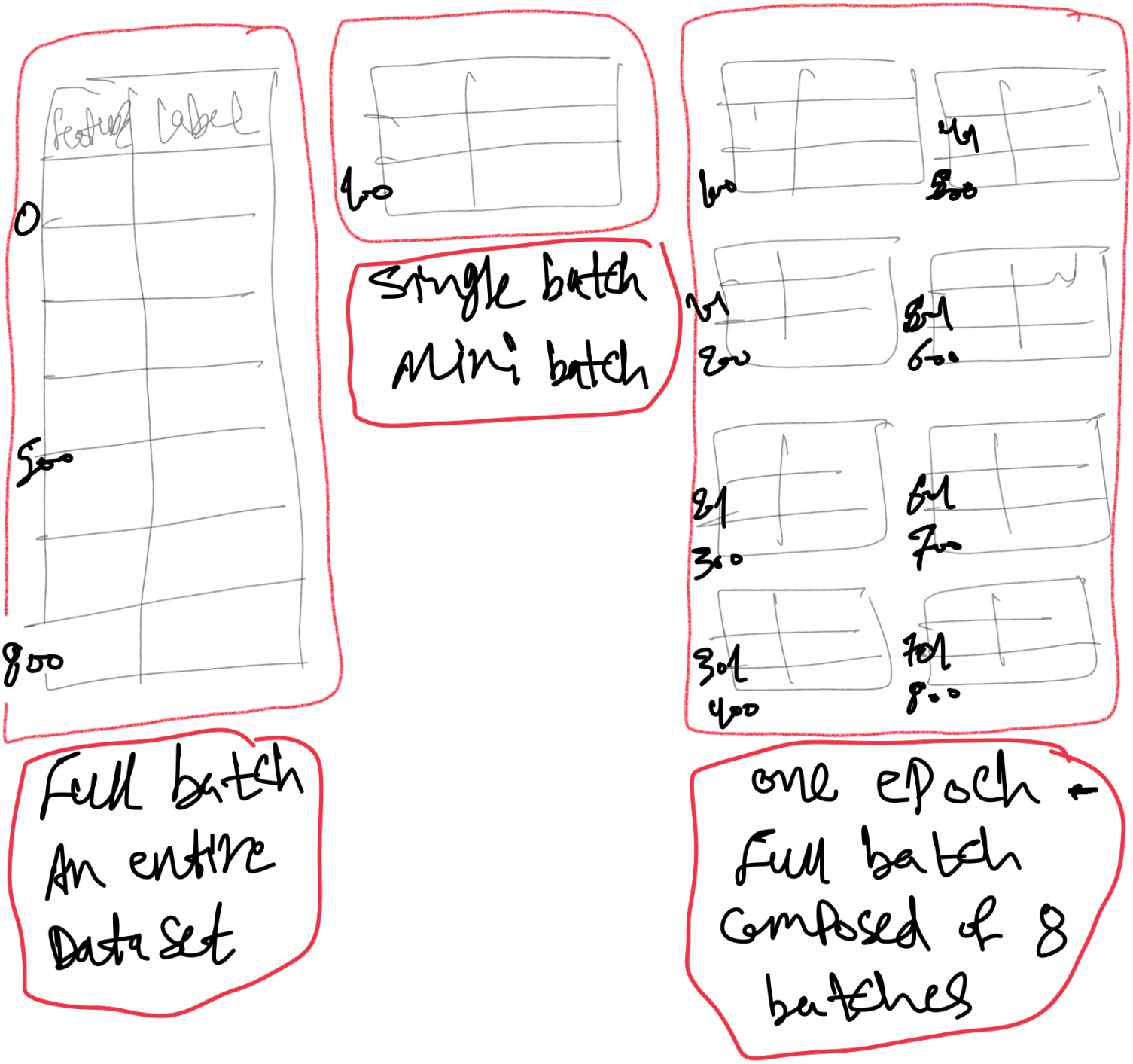
Converge quickly

Batch Size: Number of examples the model processes before updating weights & bias.

Common Techniques

- Stochastic gradient descent (SGD)
  - only one example per iteration
  - very Noisy
- Mini-batch stochastich gradient descent
  - update weights & bias once per iteration

Epochs : The model processed every example in the training set once.



Full batch
An entire Dataset

one epoch = Full batch composed of 8 batches

Batch Type	when weight & bias updates occure
Full Batch	After the model looks at all the examples in dataset
SGD	After the model looks at single example from dataset
Mini-SGD	After the model looks at the examples in each batch

Classification

Confusion Matrix: There are four possible outcomes for each output from a **binary classifier**

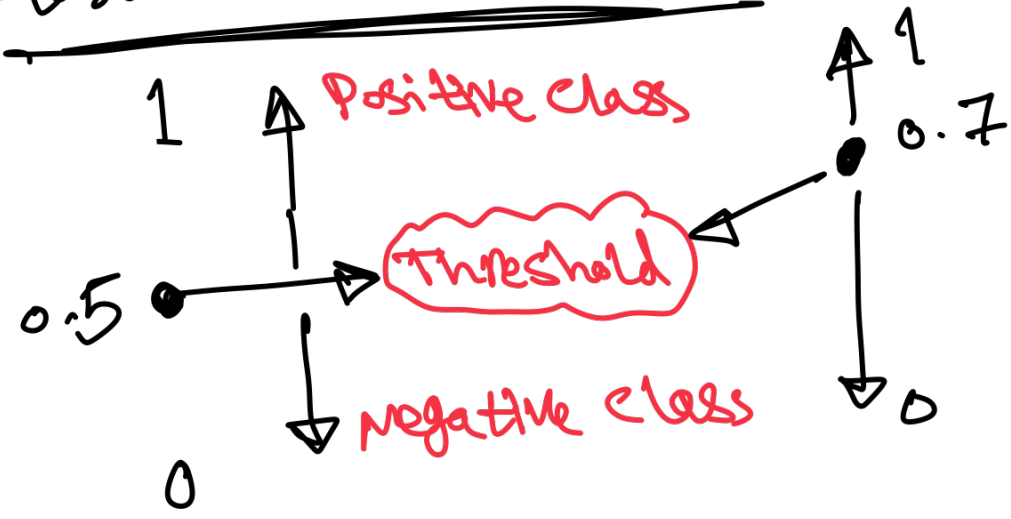
- class 1 (+)
- class 2 (-)

* In the Spam Email Spam/NotSpam
  - Spam Email (Positive class)
  - NotSpamEmail (Negative class)

	Actual Positive	Actual Negative
Predicted Positive	True Positive (TP) Spam email correctly classified as spam	False Positive (FP) A not spam Email misclassified as Spam
Predicted Negative	False Negative (FN) Spam Email misclassified as not spam	True Negative (TN) A not spam Email correctly classified as not spam

- TP + FP ⇒ All Predicted Positives
- FN + TN ⇒ All Predicted Negatives
- TP + FN ⇒ All Real Positives
- FP + TN ⇒ All Real Negatives

Classification Threshold:



Accuracy: The proportion of all classifications that were correct, whether positive or negative.

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

Recall, or True Positive Rate: the proportion of all actual positives that were classified correctly as positives.

$$\text{Recall (TPR)} = \frac{TP}{TP + FN}$$

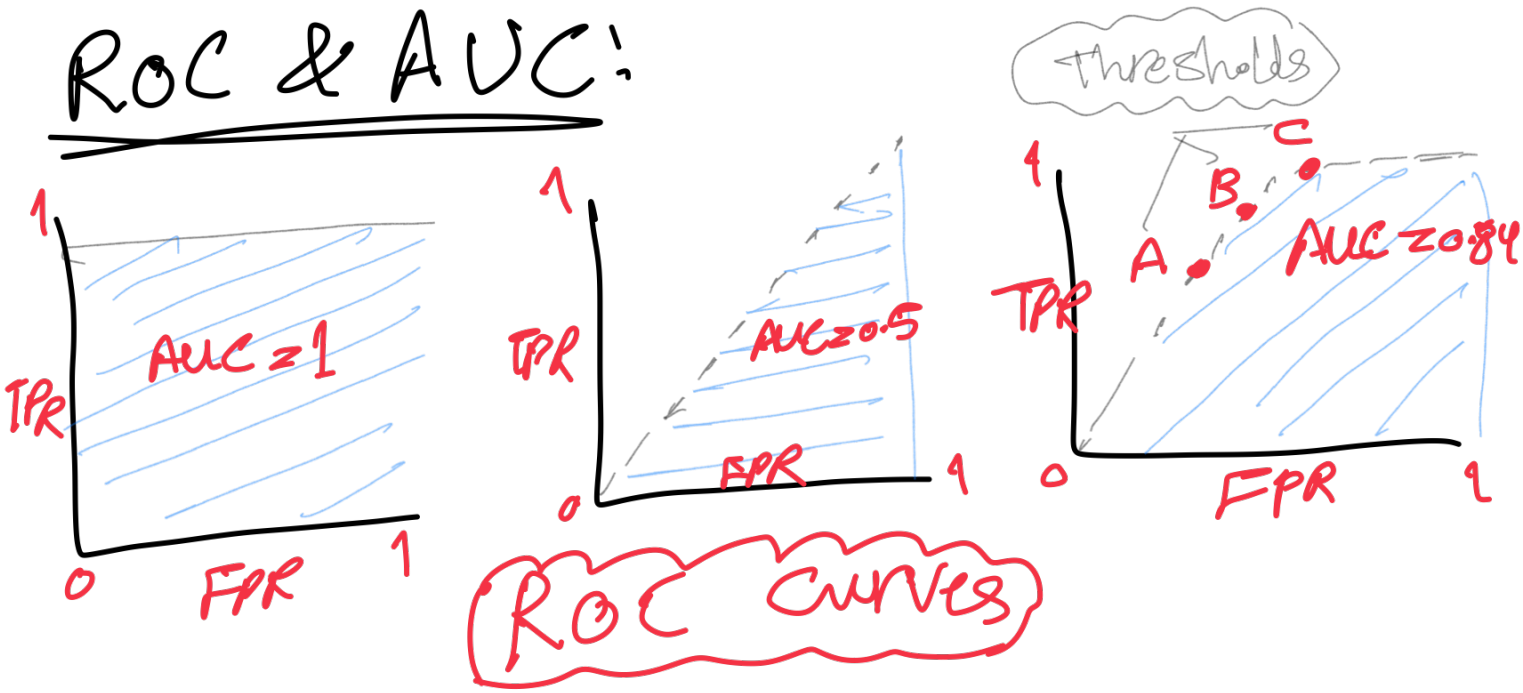
False Positive Rate: The proportion of all actual negatives that were classified incorrectly as positives.

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

Precision: The proportion of all positive classifications that are actually positive.

$$\text{Precision} = \frac{TP}{TP + FP}$$

ROC & AUC:



ROC curves

@ (A) threshold: ↓FP & ↓TP

@ (C) threshold: ↑FP & ↑TP

Prediction Bias: The difference between the mean of model's predictions and the mean of ground-truth labels in the data.

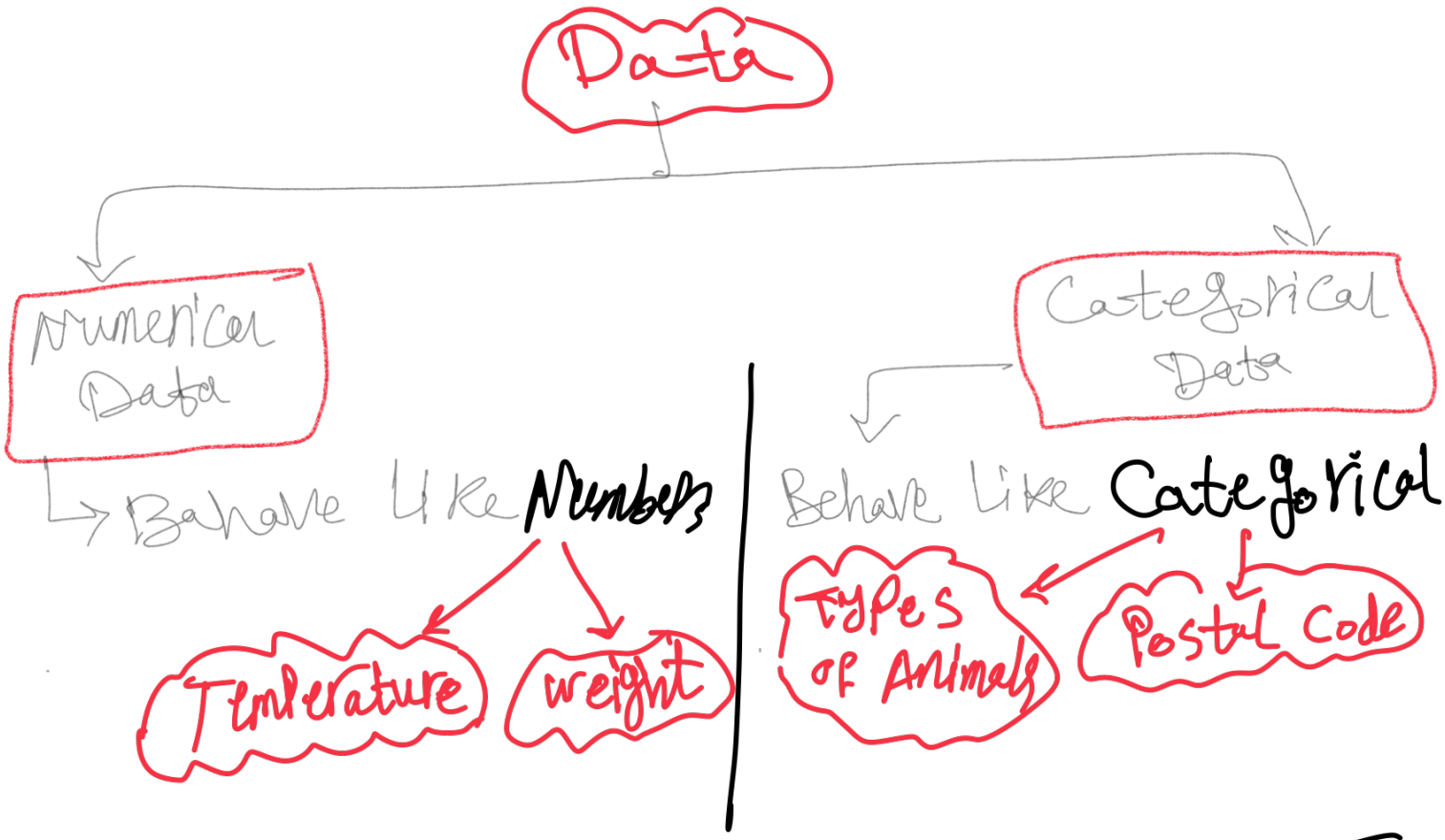
* Caused by::

→ Biases or noise in the data.

→ Too strong regularization.

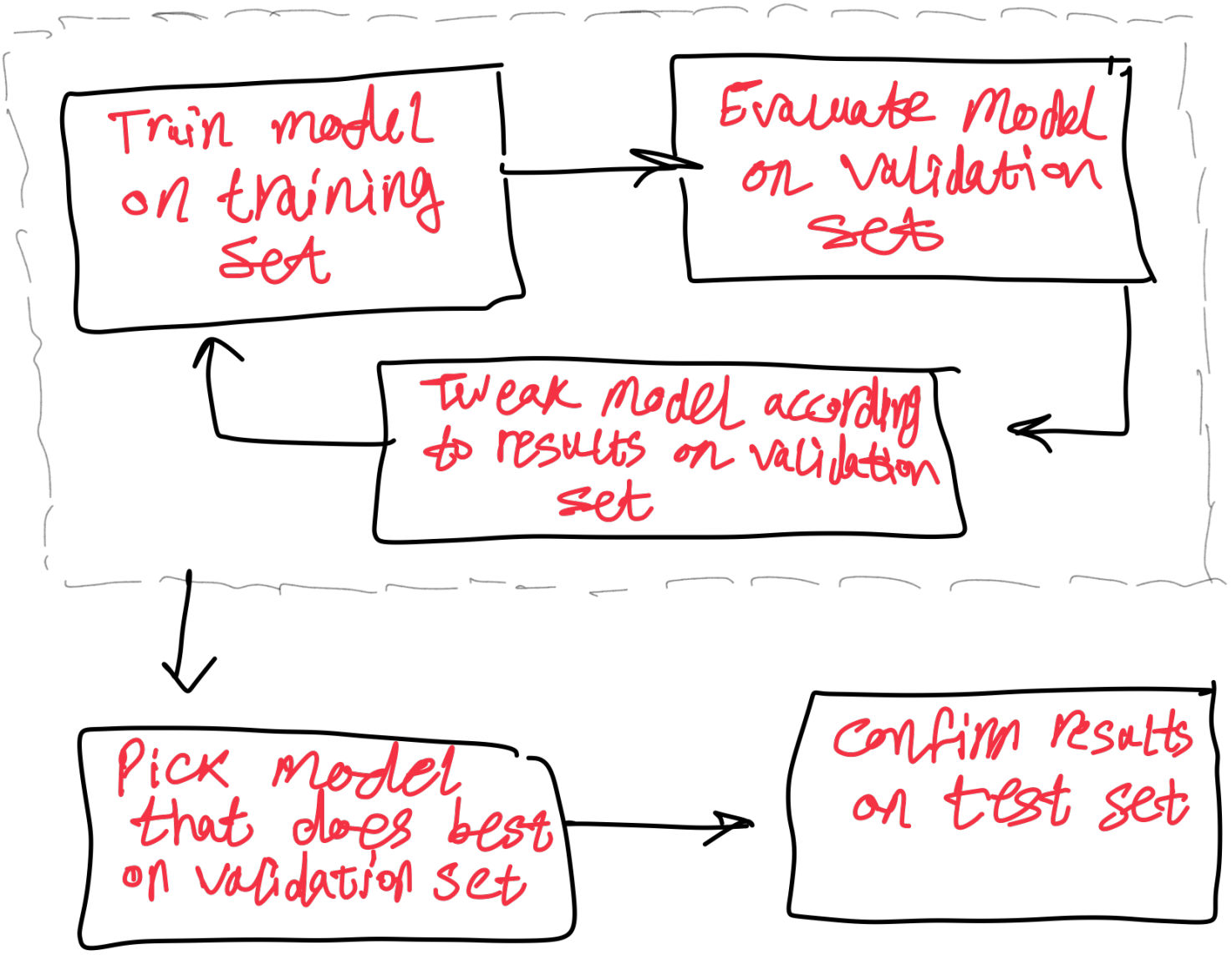
→ Bugs in the model training pipeline.

→ The features provided to the model being insufficient for the task.

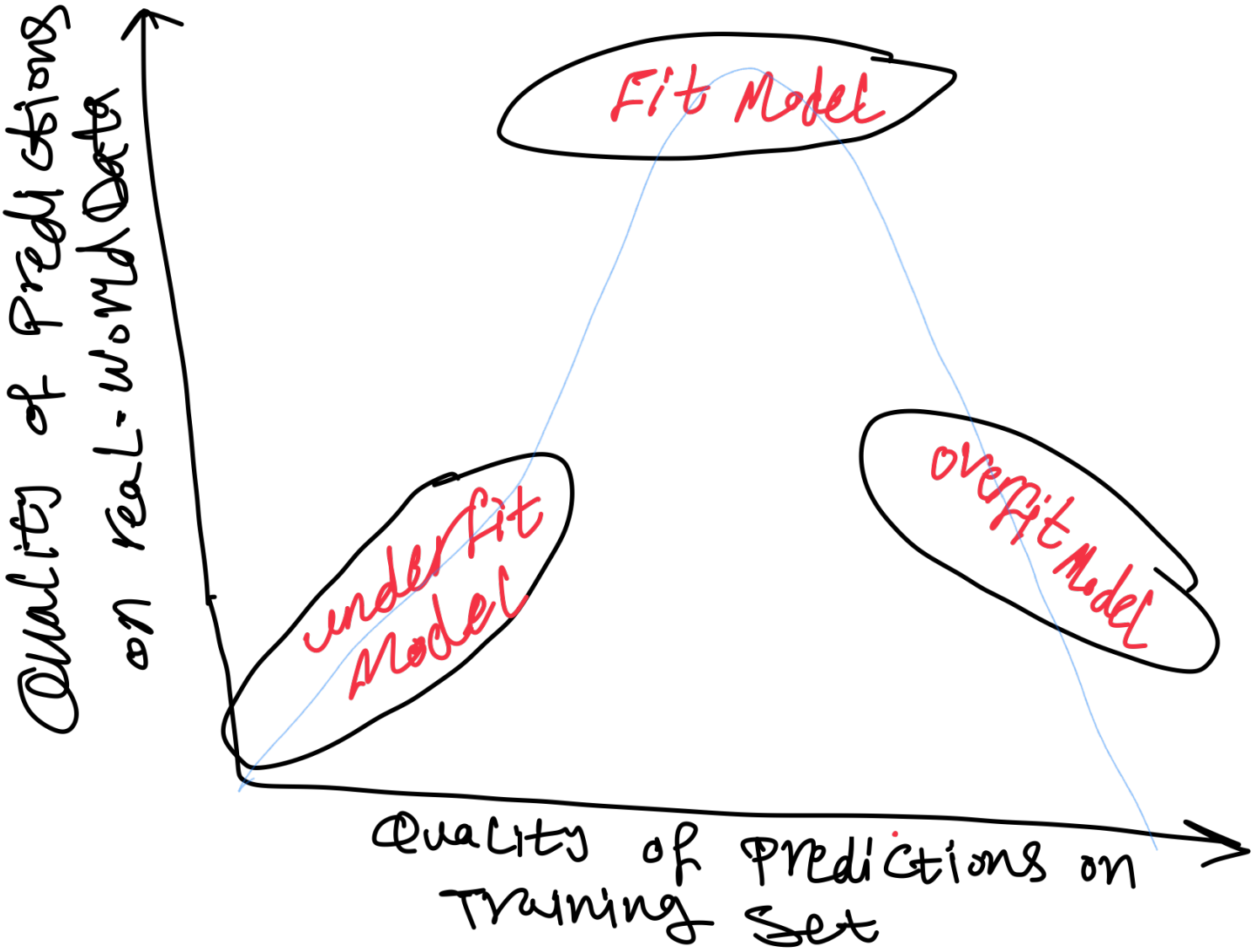


Dividing Dataset:

Workflow of Development & testing:



overfitting: means creating model that memorizes the training set so closely but the model fails to make correct predictions on new data.



* overfit Model: Best Prediction in training set, but bad Predictions in new data

* underfit Model: Bad Predictions in both training set & new data

Generalization Curve that implies overfitting:

* After a number of iterations, loss declines or holds steady for training set, but increases for the validation set, this suggests **overfitting**

overfitting Reasons:

1. training set doesn't represent real life data.
2. The model is too complex.

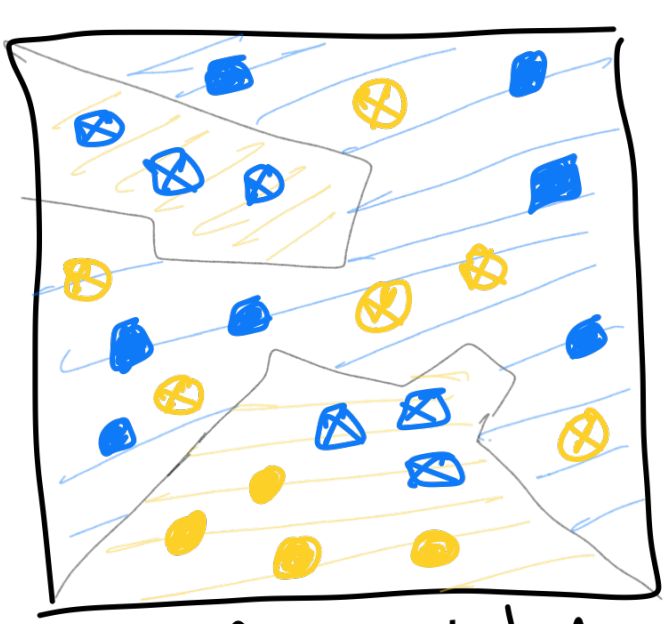
Generalization Conditions:

Model trains on training set, but the real test of a model's worth is how well it makes predictions on new examples on real-world data. Training a model that generalize well implies the following dataset conditions:

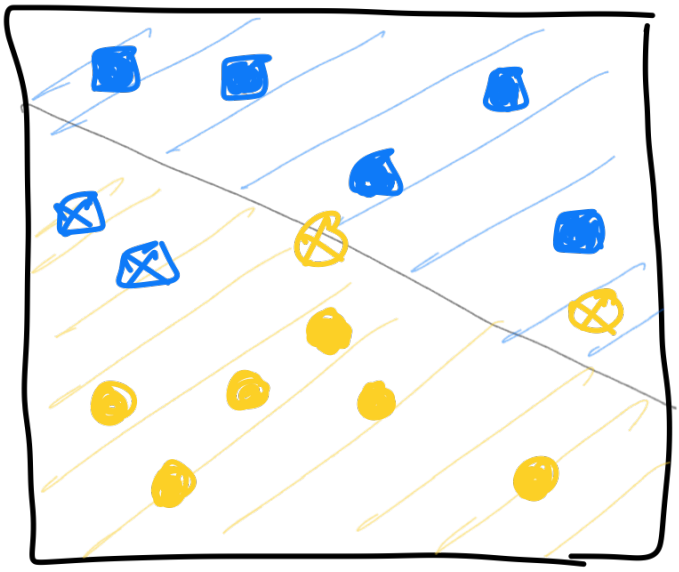
- Examples must be independently and identically distributed
- Dataset is stationary, meaning that dataset doesn't change significantly over time.
- Dataset Partitions have the same distribution, [Examples in training set are similar to examples in validation set, test set and real-world set]

* To have a similar distribution to the examples in all dataset types, you should shuffle the examples in the dataset before partitioning them.

Model Complexity:



Complex Model



Simple Model

Model predicts healthy trees in this region

Model predicts sick trees in this region

- Model correctly predicted healthy tree.
- Healthy tree, but mistakenly model predicted it as sick tree.

- Model correctly predicted sick tree.
- Sick tree, but mistakenly model predicted it as healthy tree.

* Simple model generalizes better than complex model on new data, simple model made better predictions on the test set than complex model.

Regularization: Machine learning models must simultaneously meet two conflicting goals:

- Fit data well.
- Fit data as simply as possible.

* One approach to keep model simple is to penalize complex models, forcing & penalizing complex models during training is one form of Regularization to become simpler.

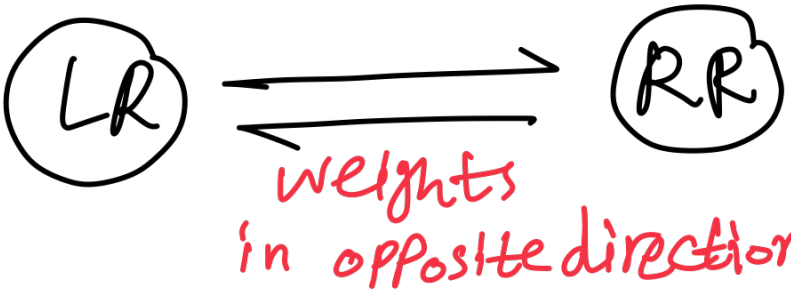
Loss & Complexity: Loss & Complexity are inversely related, as complexity increases, loss decreases, as complexity decreases, loss increases, you should find a reasonable middle ground where model makes good predictions on both training data and real-world data.

Regularization Rate:

$$\text{minimize}\ (\text{loss} + \lambda\ \text{complexity})$$

λ → Regularization Rate

Equilibrium between Learning Rate & Regularization Rate:



* High LR → Pulls weights away from Zero.

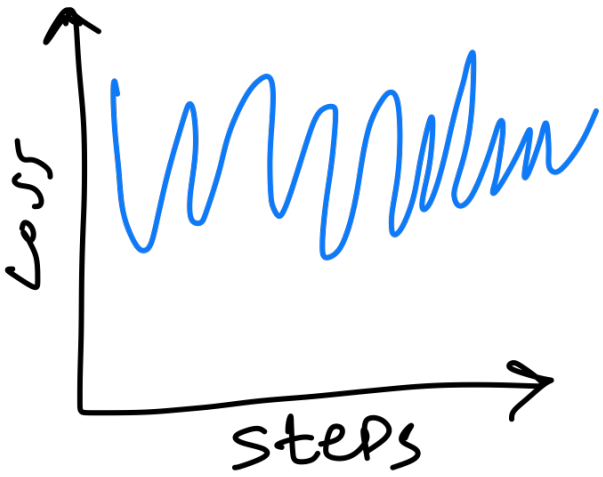
* High RR → Pulls weights towards Zero.

↑ RR ↓ LR → Weak weights tend to produce a model that makes poor predictions.

↓ RR ↑ LR → strong weights tend to produce an overfit model

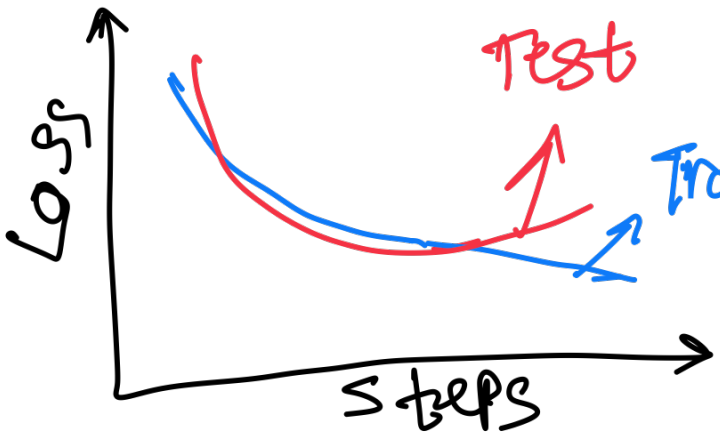
* optimal case is to find balance between (RR) & (LR)

Interpreting Loss curves:

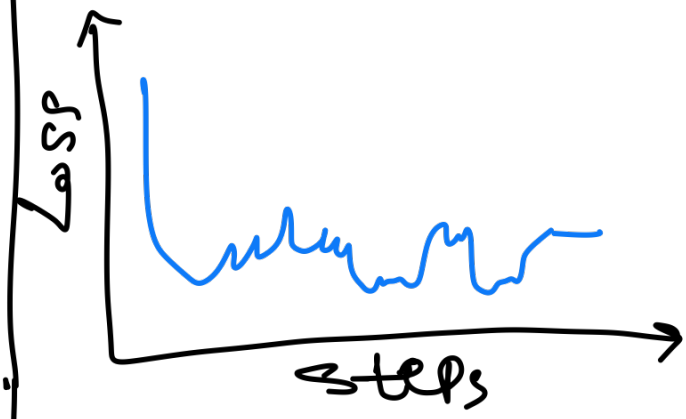


* Training set has large number of untrusted examples.
* High LR
* There is bad examples in the data

* Input data contains a burst of outliers
* There is one or more NaNs in the input data (a value caused by a division by zero)



* The model is overfitting the training set.



* The training set contains repetitive sequences of examples

Neural Networks

Neural Networks are a family of model architectures designed to find nonlinear patterns in data.

Large Language Model

Language Model estimates the probability of a token or sequence of tokens occurring within a longer sequence of tokens. A token could be a word, subword or even a single character.

when I hear rain on my roof, I _____ in my kitchen.

Probability	Token
9.4%	cook soup
5.2%	warm up kettle
3.6%	cower
2.5%	nap

* A language model determines probabilities of different tokens or sequence of tokens to complete that blank in the previous sentence.

* In some situations, the sequence of tokens could be an entire sentence, paragraph or even an entire essay.

* An application can use the probability table to make predictions. Prediction might be the highest probability, for example (cook soup) or random selection from tokens having probability greater than a certain threshold.

* Estimating probability of what fills in the blank in a text sequence can be extended to more complex tasks, including:
  - Generating text from one language to another
  - Translating text from one language to another
  - Summarizing documents.

N-gram Language Model:

N-gram are ordered sequences of words used to build LM, where N is the number of words in the sequence.

you are very nice	you are very nice
2-grams based Model → you are → are very → very nice	3-grams based Model → you are very → are very nice

* Given two words as input, LM based on 3-grams can predict the likelihood of the third word.

orange is

→ LM examines all the different 3-grams derived from its training corpus that start with orang is to determine the most likely third word.

→ orange is ripe ⇝ is about orange fruit.

→ orange is cheerful ⇝ is about color orange.

Context:

Context is helpful information before or after the target token. it can help LM determines whether "orange" refers to fruit or color.

* Context can help LM make better predictions, the only context the 3-gram provides is the first two words. For example the two words orange is doesn't provide enough context for the LM to predict the third word. Due to lack of context, LM based on 3-grams make a lot of mistakes.

* Longer N-grams would certainly provide more context than shorter N-grams.

Recurrent Neural Networks!

Recurrent Neural Networks provid more context than N-grams. It is a type of Neural Network that trains on a sequence of tokens, for example, it can gradually learn selected context from each word in a sentence, kind of like you would

Listening to someone speak. Recurrent neural Network can gain context from a Passage of several sentences.

* Although recurrent neural networks learn more context than N-grams, the amount of useful context that can intuit is still relatively limited.

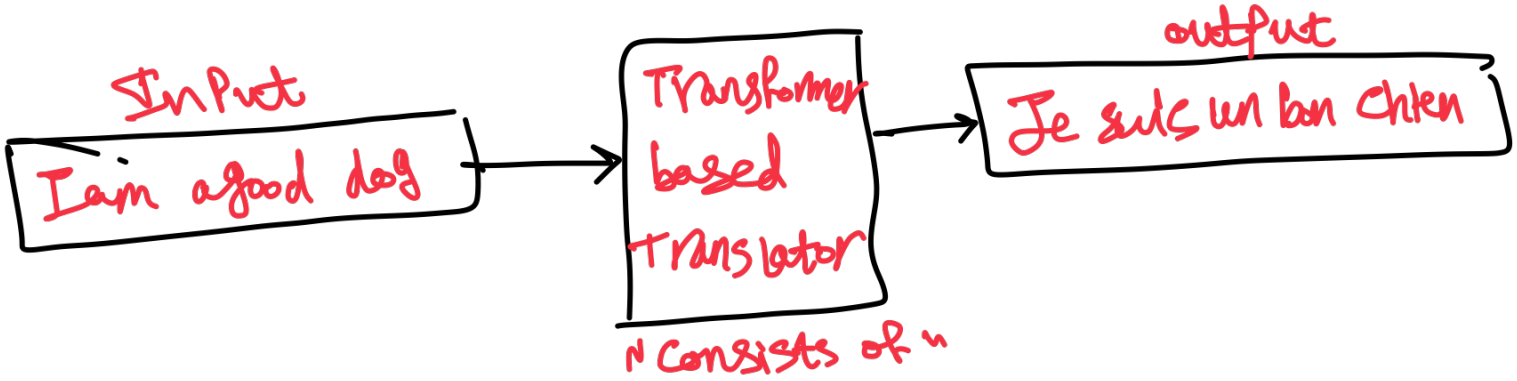
* Recurrent Neural Networks evaluate information "token by token". In contrast the LM can evaluate "The whole context at once".

Large Language Model: LLMs predict token or sequence of tokens, sometimes many paragraphs worth of predicted tokens. Token can be a word, subword or even a single character.

* LLMs make much better predictions than N-gram language models or recurrent networks.

* LLMs contain more parameters than recurrent models.
* LLMs gather far more context.

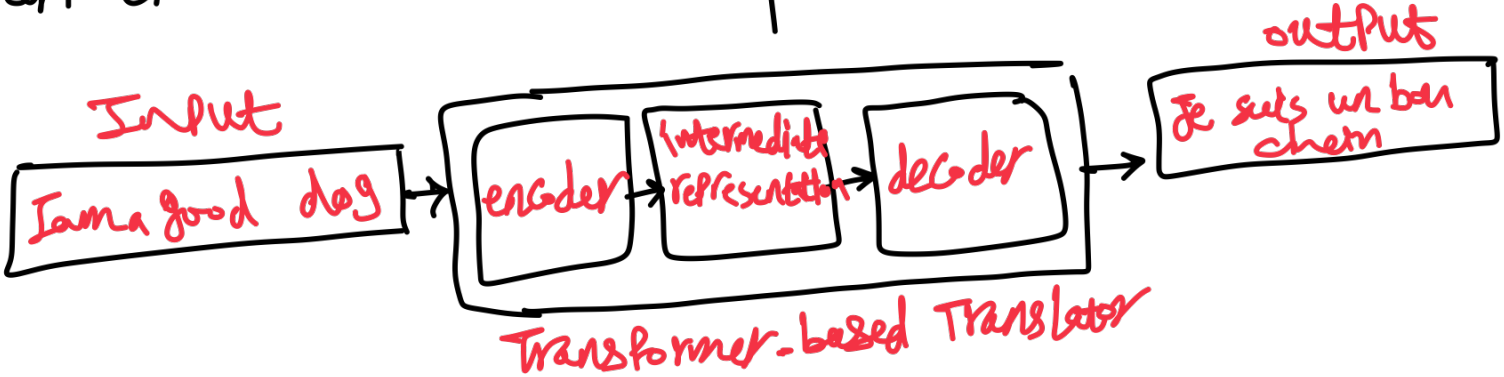
Transformer! is the ~~state-of-the-art~~ architecture for a wide variety of LM applications.



N consists of "

* Encoder converts input text into intermediate representation. Encoder is an enormous neural net.

* Decoder converts that intermediate representation into useful text. Decoder is an enormous neural net.



Self Attention: Transformers rely heavily on concept called self-attention.

* Effectively on behalf of each token of input, Self Attention asks this question:-

How much does each other token of input affect the interpretation of this token.

The Animal didn't cross the street because it was too tired

→ Self Attention determines the relevance of each nearby word to the pronoun (it).

[Animal is more important than street to the pronoun (it)]

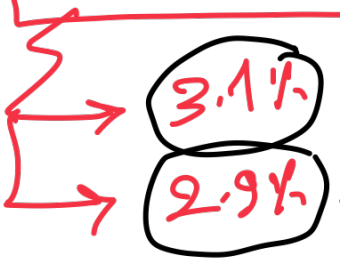
The Animal didn't cross the street because it was too wide

→ [street is more relevant than animal to the pronoun (it)]

How does LLM generate text:

LLMs are auto complete mechanisms that can automatically predict thousands of tokens.

My dog, Max, knows how to perform many traditional dog tricks.



⇒ he can sit, stay and Roll over.

⇒ he knows how to sit, stay and Roll over.

Problems with LLMs:

1- Gathering an enormous training set.

2- Consuming multiple months and enormous computational resources and electricity.

3- Solving Parallelism challenges.

Prompt Engineering: Prompt Engineering enables

LLM's end users to customize model's output. That is, end users clarify how LLM should respond to their prompt.

* LLMs learn well from examples, showing one example to LLM is called **one-shot prompting**.

* Suppose you want model to use the following format to output a fruit's family

Peach: drupe
apple: ----

apple: Pome

one-shot prompt shows LLM single example of a format and then asks LLM to complete a query based on that example

* In other situations, single example is insufficients, the user must show LLM multiple examples.

Plum: drupe
Pear: Pome
Lemon: ----

Providing multi examples is called **few-shot prompting**, you can think of the first two lines of the preceding prompt as training examples.

* Can LLM provide useful predictions with no examples **zero-shot prompting**? Sometimes, but LLMs like context, without context, the zero shot prompt might return information about the technology company rather than fruit.

apple: ----- → Fruit ??
→ Company ??

Production ML systems

Static Training (offline Training):

Train model only once. Then serve that same trained model for awhile.

Dynamic Training (online Training):

Train model frequently. Then serve recently trained model.

	Static Training	Dynamic Training
Advs	Simpler: You only need to develop and test model once.	More Adaptable: Your model keeps up with any changes to the relationship between features and labels.
Di's - Advs	Sometimes Stale: If relationship between features and labels changes over time, your model's predictions will degrade.	More Work: You must build, test and release a new product all the time.

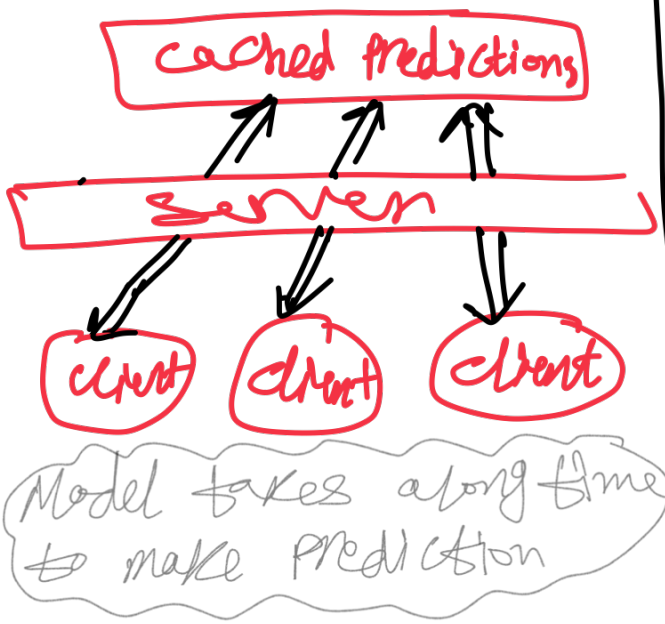
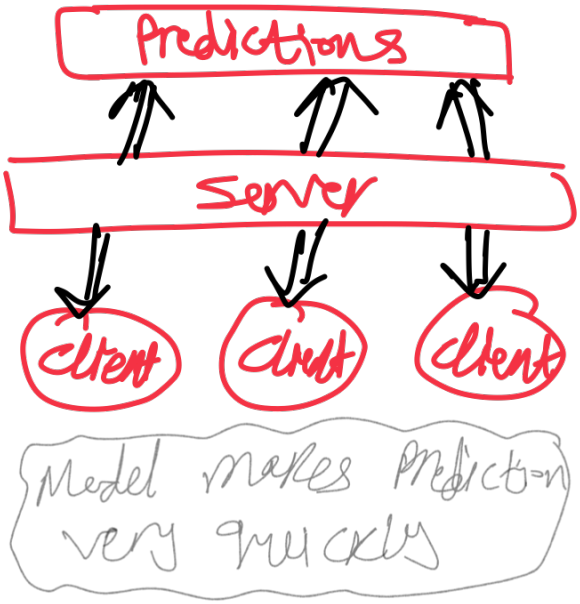
Interface: Interface is the process of making predictions by applying a trained model to unlabeled examples.

Static Inference (offline Inference):

Model makes predictions on a bunch of common unlabeled examples and then caches those predictions somewhere.

Dynamic Inference (online Inference):

Model makes predictions on demand, for example when a client requests a prediction.

	Static Inference	online Inference
		
Advs	- Don't need to worry about cost. - Can do Post-verification of Predictions before Pushing.	- Can infer prediction on any new item as it comes in, which is great for long tail predictions.

	Static Inference	Dynamic Inference
Dis - Advs	- Can only serve cached predictions, so the system might not be able to serve predictions for uncommon input examples. - Update latency is likely measured in hours or days.	- Compute intensive and latency sensitive. - Monitoring needs are more intensive.

(Fairness)

Types of Bias:

ML models are not inherently objective. ML practitioners train models by feeding them a dataset of examples, and human involvement in the provision and curation of this data can make model's predictions susceptible to bias.

* When building models, it is important to be aware of common human biases that can manifest in your data, so you can take steps to mitigate their effects.

Reporting bias

DEFINITION

Reporting bias occurs when the frequency of events, properties, and/or outcomes captured in a dataset does not accurately reflect their real-world frequency. This bias can arise because people tend to focus on documenting circumstances that are unusual or especially memorable, assuming that the ordinary does not need to be recorded.

Click › for an example.

Historical bias

DEFINITION

Historical bias occurs when historical data reflects inequities that existed in the world at that time.

Click › for an example

Automation bias

DEFINITION

Automation bias is a tendency to favor results generated by automated systems over those generated by non-automated systems, irrespective of the error rates of each.

Click › for an example

Selection bias

Selection bias occurs if a dataset's examples are chosen in a way that is not reflective of their real-world distribution. Selection bias can take many different forms, including coverage bias, non-response bias, and sampling bias.

Coverage bias

DEFINITION

Coverage bias occurs if data is not selected in a representative fashion.

Click › for an example

Non-Response bias

DEFINITION

Non-response bias (also knows as **participation bias**) occurs if data ends up being unrepresentative due to participation gaps in the data-collection process.

Click › for an example

Sampling bias

DEFINITION

Sampling bias occurs if proper randomization is not used during data collection.

Click › for an example

Group attribution bias

Group attribution bias is a tendency to generalize what is true of individuals to the entire group to which they belong. Group attribution bias often manifests in the two following forms.

In-group bias

DEFINITION

In-group bias is a preference for members of your own group *you also belong*, or for characteristics that you also share.

Click › for an example

Out-group homogeneity bias

DEFINITION

Out-group homogeneity bias is a tendency to stereotype individual members of a group to which *you do not belong*, or to see their characteristics as more uniform.

Click › for an example

Implicit Bias

DEFINITION

Implicit bias occurs when assumptions are made based on one's own model of thinking and personal experiences that don't necessarily apply more generally.

Click › for an example

Confirmation bias

DEFINITION

Confirmation bias occurs when model builders unconsciously process data in ways that affirm pre-existing beliefs and hypotheses.

Click › for an example

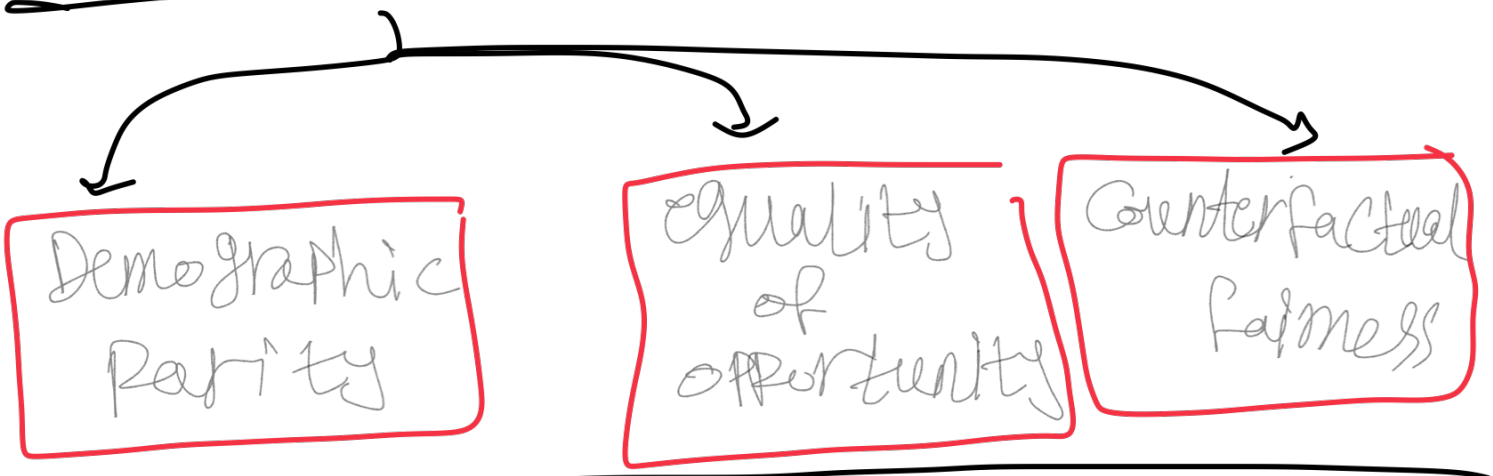
Experimenter's bias

DEFINITION

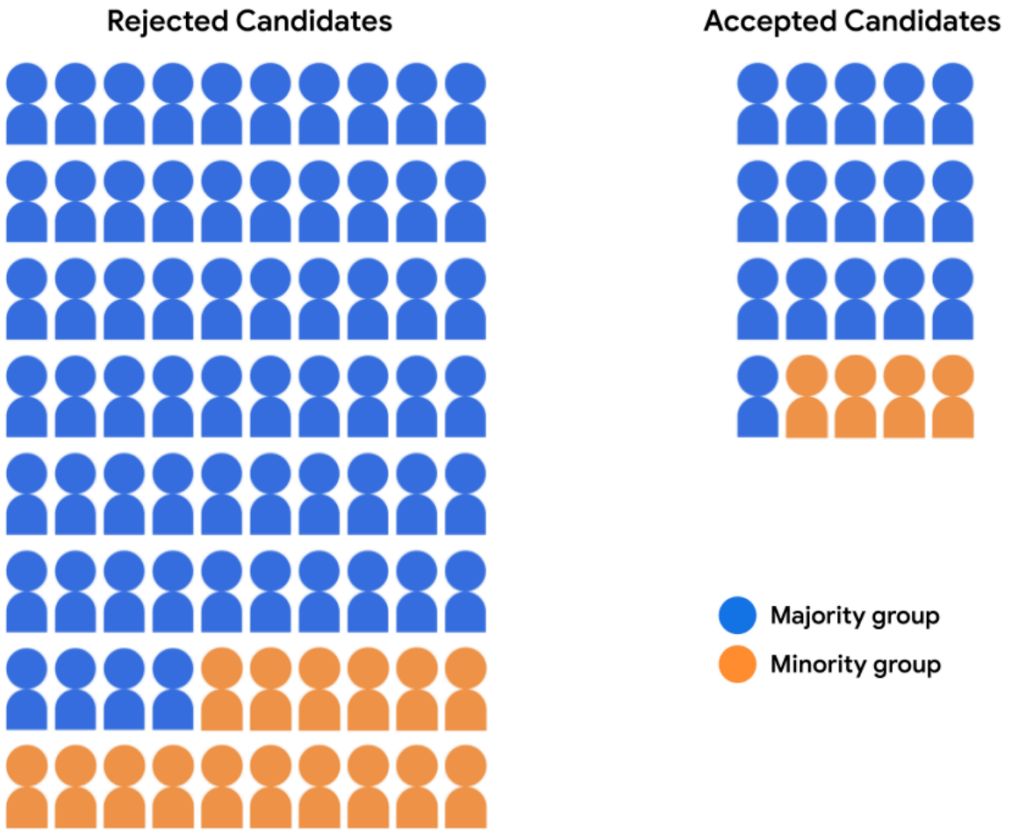
Experimenter's bias occurs when a model builder keeps training a model until it produces a result that aligns with their original hypothesis.

Click › for an example

Fairness Metrics!



Demographic Parity: one method we can use to evaluate model's predictions for fairness is to compare admissions rate for the majority group and the minority group. If the 2 admissions rates are equal, then the model's predictions exhibit demographic parity.



	Majority group	Minority group
Accepted	16	4
Rejected	64	16
Acceptance Rate	20%	20%

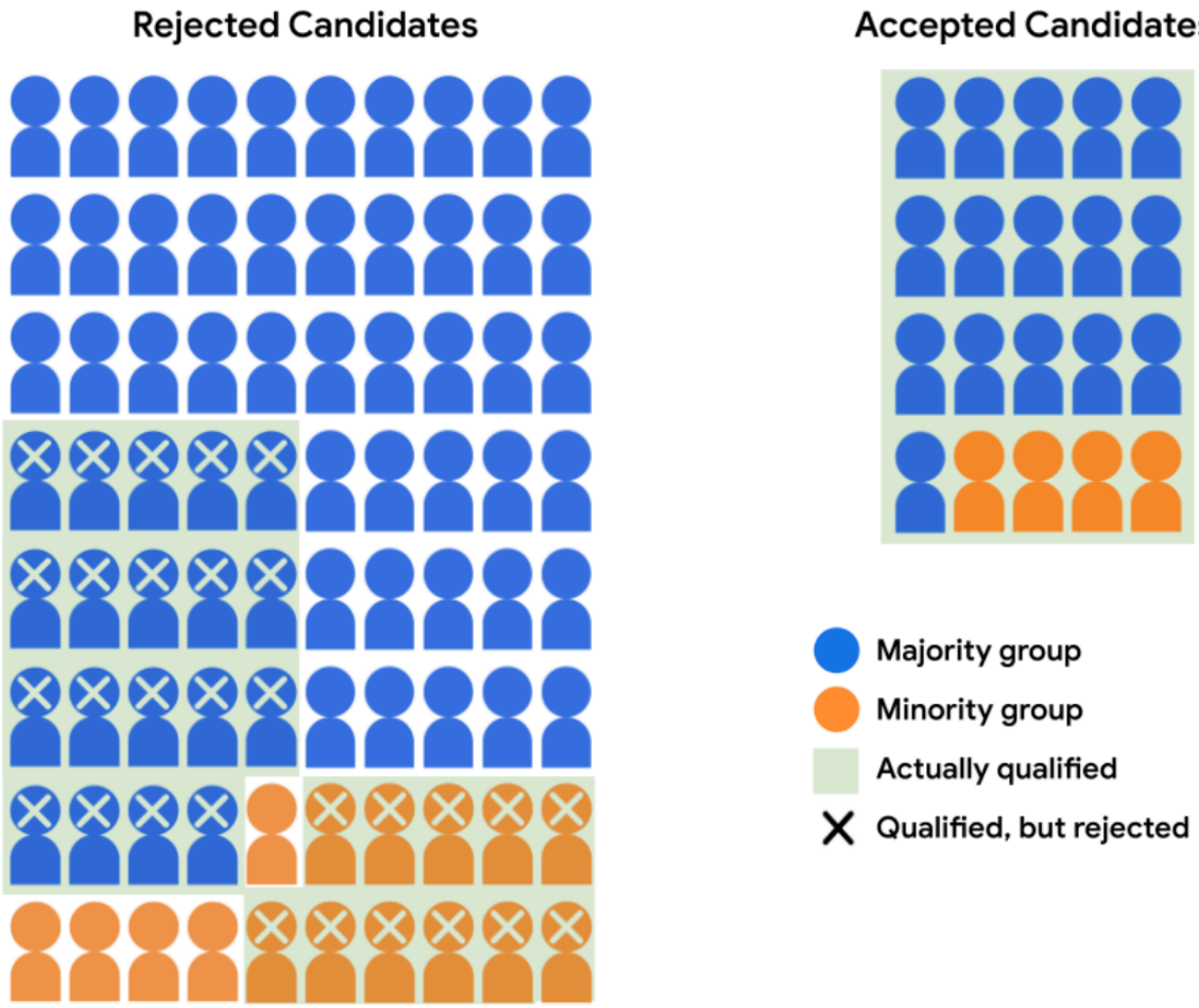
Benefits & Drawbacks :

Benefits

- Ensures that both majority & Minority are represented in the admitted class of students in the same proportions as they are in the candidate pool.

Drawbacks

- It doesn't take the distribution of predictions of each demographic group (qualified Vs unqualified) into account when evaluate how to admission slots should be allocated.



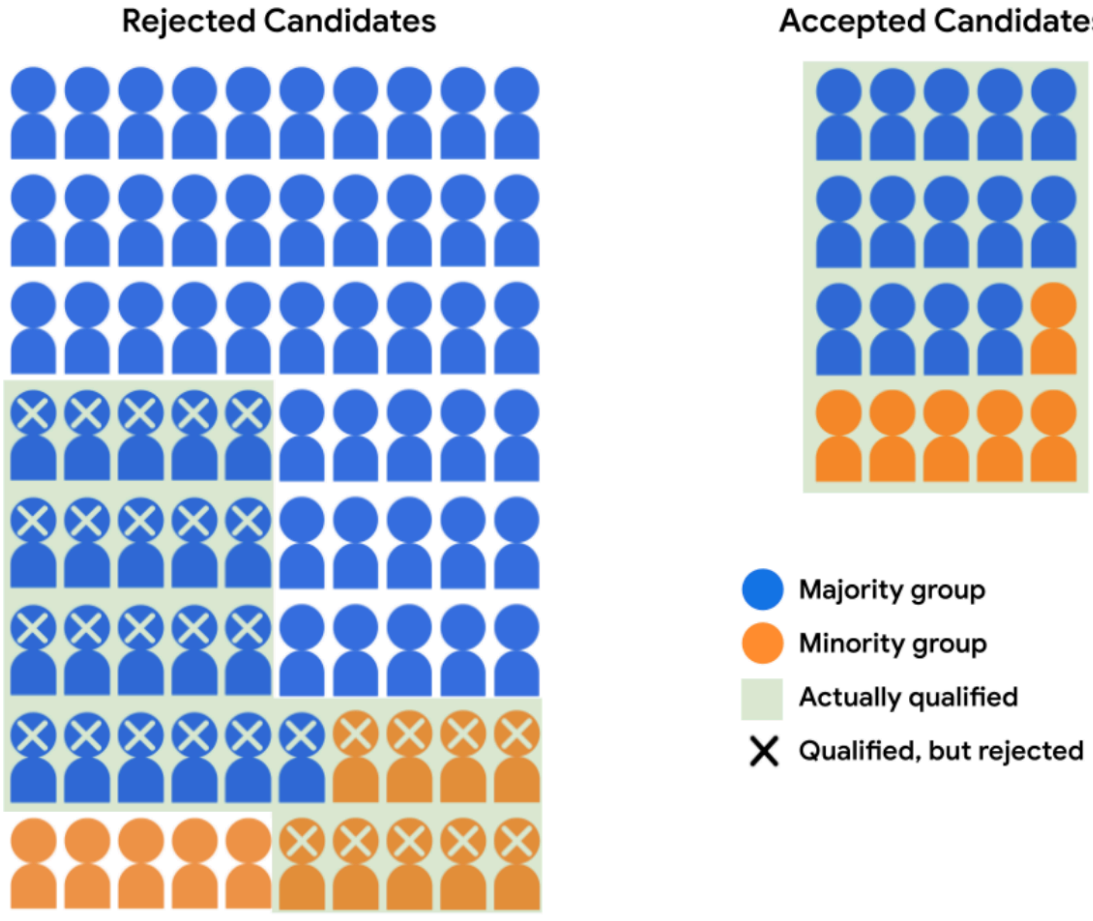
	Majority group		Minority group	
	Accepted	Rejected	Accepted	Rejected
Qualified	16	19	4	11
Unqualified	0	45	0	5

Majority acceptance Rate = $\frac{16}{35}$ = 46%

Minority acceptance Rate = $\frac{4}{15}$ = 27%

Equality of opportunity: We could compare the acceptance rates for just the qualified candidates in the majority and the minority group. If the acceptance rates for qualified students in both groups are equal, the model exhibits **equality of opportunity**

	Majority group	Minority group
Qualified	35	15
Unqualified	45	5



	Majority group		Minority group	
	Accepted	Rejected	Accepted	Rejected
Qualified	14	21	6	9
Unqualified	0	45	0	5

Benefits & Drawbacks:

Benefits

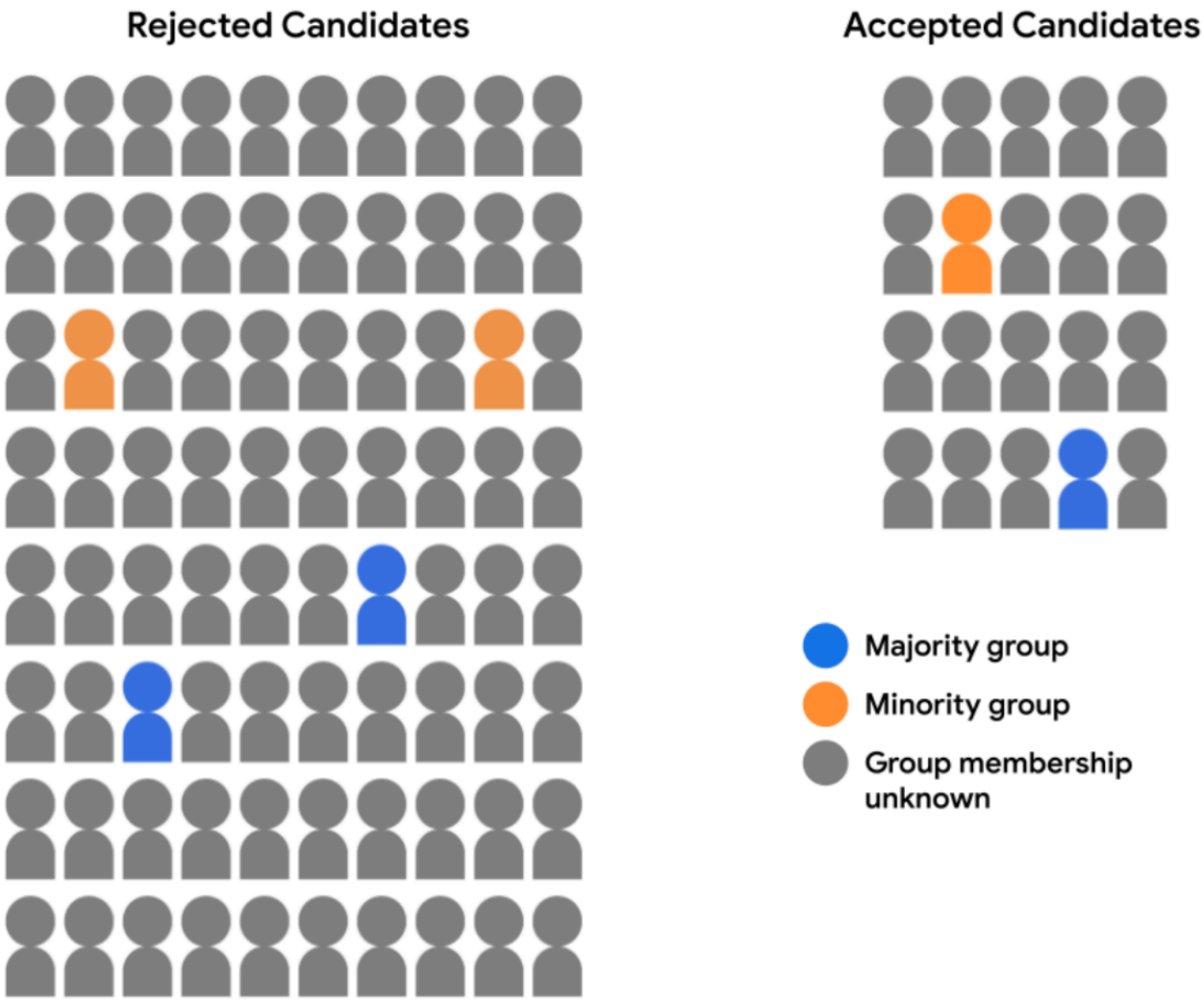
- Allows model's Ratio of positive to negative predictions to vary across demographic groups, provided that model is equally successful at predicting the preferred label for both groups.

Drawbacks

- Designed for use cases where there is clear-cut preferred label.
- It assesses fairness by comparing error rates in aggregate for demographic groups, which may not always be feasible.

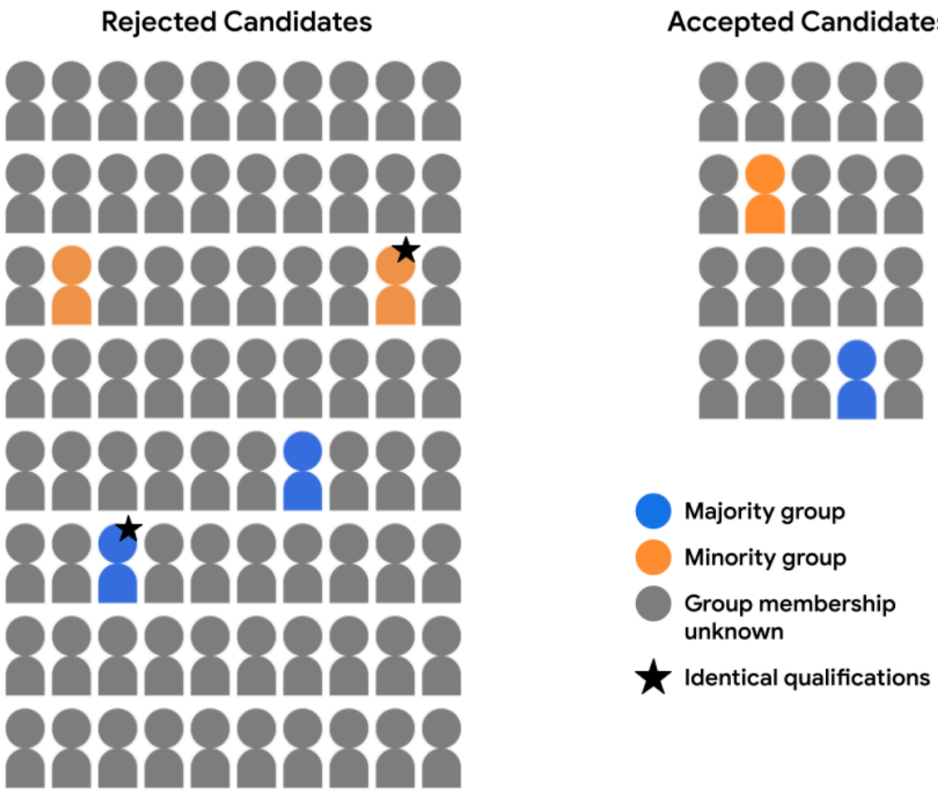
* $\frac{14}{80} = 17.5\%$ "being Accepted in Majority"
* $\frac{6}{20} = 30\%$ "being Accepted in Minority"
* 40% "being Accepted in qualified in both groups"

Counterfactual Fairness? Suppose that our admissions dataset doesn't contain complete demographic data, instead, demographic group membership is recorded for just a small percentage of examples, such as students who opted to self-identify which group they belonged to.



* If we have thoroughly reviewed the feature data available for 2 candidates (annotated with a star in the image below), and have determined that they are identically qualified for admission in all respects. If the model makes same prediction for both of these candidates, (either rejects both or accepts both), it is said to satisfy **Counterfactual fairness**, for these examples.

* Counterfactual fairness stipulates that 2 examples that are identical in all respects, except a sensitive attribute (Demographic group membership), should result in the same model prediction.



Benefits & Drawbacks:

Benefits

- Can be used to evaluate predictions for fairness in many cases where using other metrics wouldn't be feasible.

Drawbacks

- It doesn't provide as holistic a view of bias in model predictions.