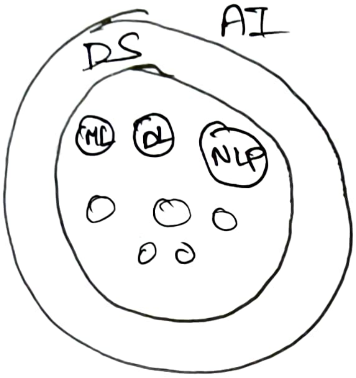
# Machine learning




## Types of ML Systems

- Supervised learning
- Unsupervised learning
- Reinforcement learning
- Generative AI.

## Supervised learning

↳ models can make predictions after seeing lots of data with the correct answers and then discovering the connections b/w the elements in the data that produce the correct answers.

(like a student reffering PYQs)

### Regression

↳ A regression model predicts a numerical value. For eg., a weather model that predicts the amount of rain, in inches or millimeters, is a regression.

### Classification

↳ models predicts likelihood that something belongs to a category.

Binary classification

0 (or) 1 -

T F

Multiclass classification

category

Unsupervised learning

↳ models make predictions by given data that does not contain any correct answers. It's goal is to ~~classify~~ identify meaningful patterns among the data. In other words, the model has no hints on how to categorize each piece of data, but instead it must infer its own rules.

Reinforcement learning

↳ models make predictions by getting rewards or penalties based on actions performed within an environment. A reinforcement learning systems generate a policy that defines the best strategy for getting the most rewards.

# Supervised learning

Supervised machine learning is based

- Data
- Model
- Training
- Evaluating
- Inference

date lat long temp humidity clouds wind pressure

↓

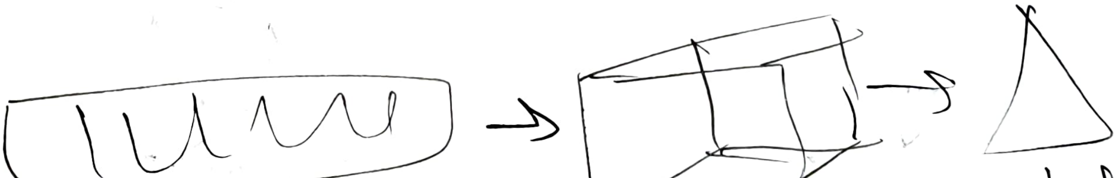
features

↓

rainfall

↳ Label → The column which needs to be predicted.

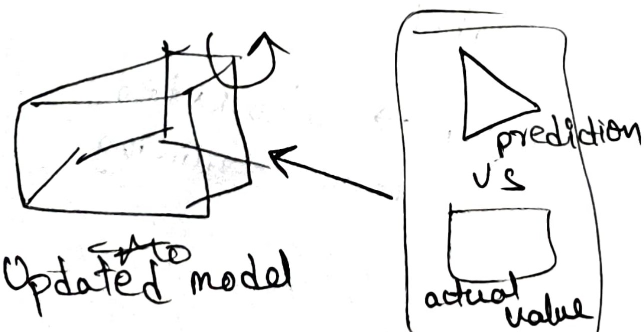
## Training



Labelled Example

Model

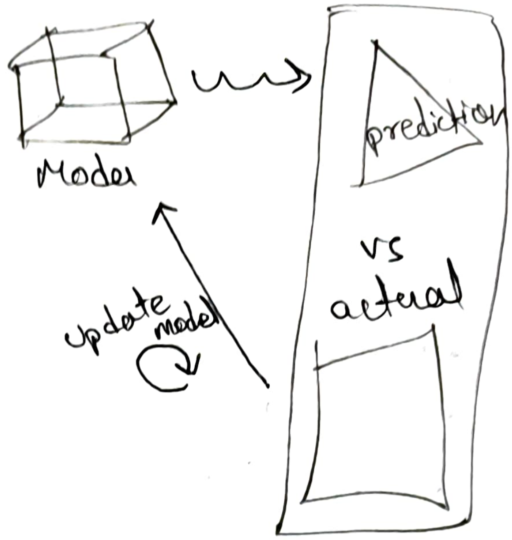
prediction




Updated model

if prediction is wrong it calculates loss and update itself.

dataset → labelled example →



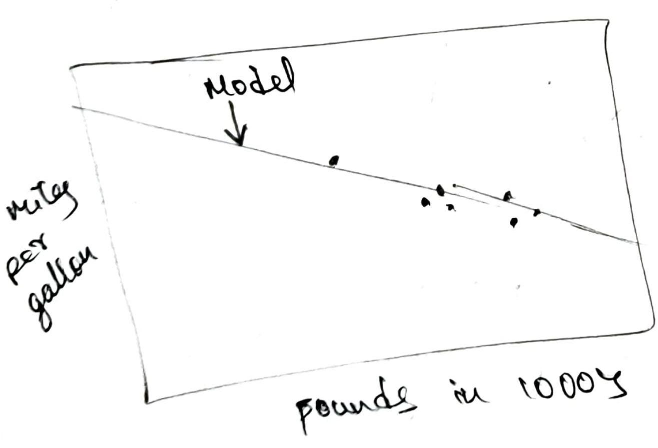

## Linear Regression

Statistical technique used to find the relationship b/w variables.

In ML context,

linear regression finds the relationship b/w features and a label.

linear regression eqn : $y = mx + b$




$y' = b + w_1 x_1$

bias ↑ (b), feature ↑ ($x_1$)

$y'$ ↓ predicted label

$w_1$ ↓ weight of the feature

$b, w_1$ → calculated from training

## Models with multiple features:

$$y' = b + w_1x_1 + w_2x_2 + w_3x_3 + \dots$$

$x_1, x_2, x_3, x_4, \dots$ are features

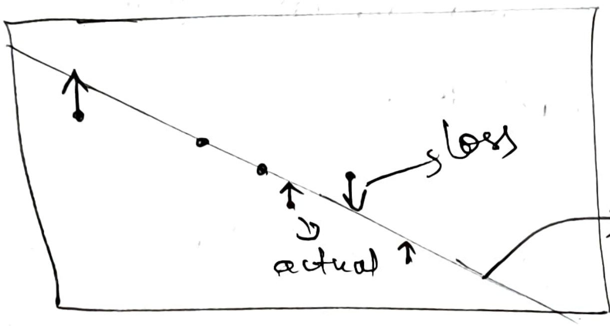
$b \rightarrow$ bias

$w_1, w_2, w_3 \dots$ weights

## Loss

numerical metric that describe how wrong a model's predictions are.

Loss measures the distance b/w the model's predictions and the actual labels.




→ Model ~~pred~~ prediction

→ dosen't care about direction. (use |x|)

loss = abs(loss)

(or)

Square the ~~value~~ diff. b/w actual value and the prediction.

### Types of loss

→ Mean Squared Error.

→ $L_1$ loss

→ Mean absolute Error

→ $L_2$ loss

$L_1$ loss - $\sum |\text{actual value} - \text{predicted value}|$

MAE - $\frac{1}{N} \sum |\text{actual value} - \text{predicted value}|$

$L_2$ loss - $\sum (\text{actual value} - \text{predicted value})^2$

MSE - $\frac{1}{N} \sum (\text{actual value} - \text{predicted value})^2$

Choosing a loss: (refer google dev ML crash course).

choose MSE:

-> If you want to heavily penalize large errors.

-> If you belive the outliers are important and indicative of true data variance that the model should account for.

choose MAE:

-> If your dataset has significant outliers that you don't want to overly influence the model.

MAE is more robust.

-> If you prefer a loss function that is more directly interpretable as the average error magnitude.

## Gradient descent:

↳ mathematical technique that iteratively finds the weights and bias that produce the model with the lowest loss.

The model begins ~~with~~ training with randomized weights and biases near zero, and then repeats following

1. calculate the loss with current weight and bias.
2. Determine the direction to move the weights bias that reduces loss.
3. ~~Mothe~~ Move the weight and bias a small amount in the direction that reduces loss.
4. Return to step one and repeat the process until the model can't reduce the loss any further.

---

Linear regression with MSE is always converge.

**Hyperparameters:** variables that control different aspects of training.

3 common hyperparameters: Learning rate
Batch size
Epochs.

In contrast, parameters are the variables, like weights and bias, that are part of the model itself. In other words, hyperparameters are values that you control;

<u>Learning rate</u>: (LR)

floating point number you set that influence how quickly the model converges.

If LR is too low, the model can take a long time to converge.

if LR is too high, the model never converges, but instead bounces around the weights and bias that minimise the loss.

→ The goal is to pick a LR ~~value~~ that's not ~~too~~ high nor too low so that the model converges quickly.

→ Model multiplies gradeint and LR to determine its parameters (w and b) for next iteration.

→ next step of gradient descent, the "small amount" refers to to move in the direction of -ve ~~direction~~ slope the learning rate.