the Process of training a Piece of software, called a model to make useful Predictions (or) generate content from data.

Weather Prediction :

* Traditional approach : Physics based approach of earth's atmosphere and surface, computing massive amounts of fluid dynamics equations (Hard method)

* ML approach : we give a ML model an enormous amount of data until in learns mathematical relationship between Patterns that Produce diffent amounts of rain.

Types of ML Systems :

→ supervised learning
→ unsupervised learning
→ Reinforcement learning
→ Generative AI learning.

* supervised learning : "the supervised learning models make prediction based on seeing large amounts of data and discover connection between the elements in the data that Produce the correct answers". these ML systems are "supervised" in the sense that a human gives the ML system data with correct results.

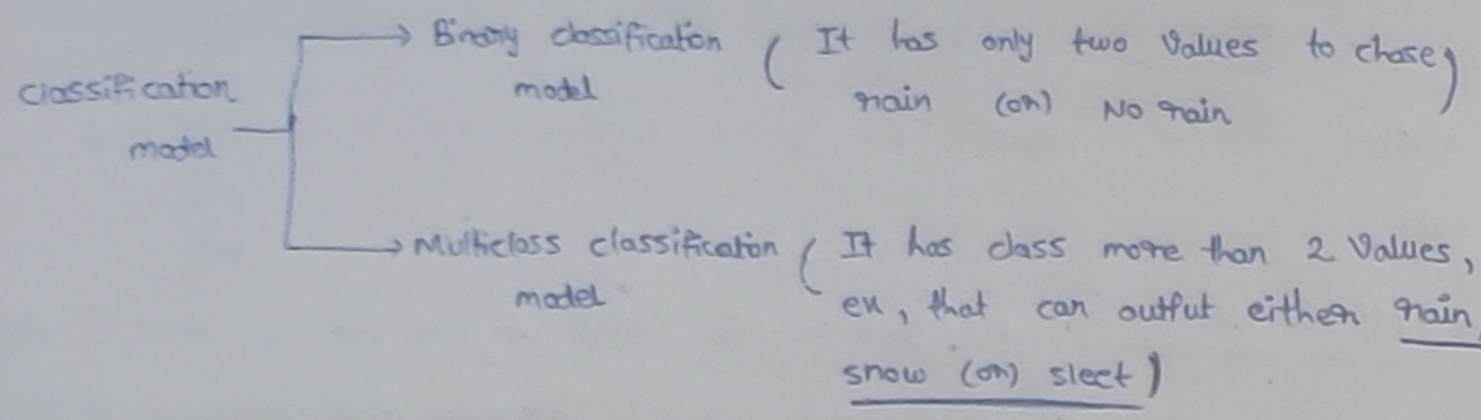
"Most common cases for supervised learning are regression and classification".

⇒ Regression : It Predicts a numeric value ex: amount of rain in mm / inches, is a regression model.

ex: Future House Price

Future Ride time

* Classification model : unlike regression classification model doesn't give a numeric value as output they only state that the value belongs to particular category (or) not ex : Rain (or) No Rain , email is spam (or) not etc



* Unsupervised Learning : It can make Predictions by being given data that doesn't contain any correct answers . It's main goal is to identify meaningful Patterns among data . It has no hints / no category of data , but instead infer to it's own rules.

A commonly used unsupervised learning model employs a technique called clustering . the model finds data that demarcate natural groupings.

clustering differs from classification as categories aren't defined by us.

* Reinforcement learning : These models make Predictions by getting rewards (or) Penalities based on actions Performed within an environment It generates a Policy that defines best strategy for getting most rewards.

ex : Used to train robots to Perform tasks , like walking around room etc AlphaGo (A model which defeated the master of the game GO)

* Generative AI : It is a class of model that creates content for user input . for ex , it can create unique images , music compositions , and jokes it can summarize articles explain how to Perform a task , or edit a Photo.

↑ip ↑op
→ Text - to - Text
→ Text - to - image
→ Text - to - Video
→ Text - to - code
→ Text - to - speech
→ image and text - to - image

* Foundational supervised learning Concepts :

→ Data
→ Model
→ Training
→ Evaluating
→ inference

* Data : Data is the driving force of ML . It comes in words, numbers or values of pixel . we store it in datasets .

ex : Image of cats weather information
Housing Prices

They are makeup of individual examples that contain features and a label

features : components that ML model depends on to Predict

label : It is the answer / the value we want ML to Predict

If has Examples that contain both features and label are called labelled examples.

date	lat	Lon	temp	Humidity	Cloud_coverage	rainfall ← label
2021-09-09	49.71	82.168	24	20	3	.01
2021-09-09	39.71	87.168	82	42	6	.23

↳ this row is example

↑ features

Unlabelled examples contain features, but no label. After the model is created the model Predicts label from features.

* characteristics : It is characterized by its size and diversity

Good datasets are large and highly diverse. A dataset can be characterised by the number of features. Data set with more features has more chance to find the new Patterns and make better Predictions.

* Model : A complex collection of numbers that define the mathematical relationship from specific input feature Pattern to specific output Values.

* Training : Before making Predictions a model should be trained. To train a model we give the model a dataset with labeled examples. The model training is done by giving a dataset and Predict an output and compare to the actual output and the difference between them is called Loss. the model gradually updates it's soln in other words, it learns the mathematical relationship b/w the features and the Label so that it can make best Predictions on unseen data.

Linear Regression :

Take an example of weight of a car and how many miles it can cover you are given a data in pounds and Miles per gallon

Pounds in 1000's (Feature)	Miles Per gallon (label)
3.5	18
3.69	15
3.44	18
3.43	16
4.34	15
4.42	14
2.37	24

Visual representation :

```
import numpy as np
import matplotlib.pyplot as plt

X = np.array([Pounds in 1000's values])

Y = np.array([MilesPer gallon values])

plt.scatter(X,Y)

x0, y0 = 0, C   (X intercept)

x1, y1 = -C/m, 0   (Y intercept)

plt.plot([x0,y0],[x1,y1], colour='blue')

plt.ylim(0,35)

plt.xlim(0,50)
        ⟶ plt.grid()
plt.show()
```

$y = mx + C$ ⟶ algebra

$y' = b + w_1x_1$ ⟶ feature
↑ bias
↓ weight

Prediction

if a model has multiple features

$$y' = b + \sum_{i=1}^{n} w_i x_i$$

for ex :

$$y' = b + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5$$

x_1 → Pounds displacement
x_2 → acceleration
x_3 → no of cylinders
x_4 → Horsepower
x_5 → no of people travelling

***Loss :** It is a numerical metric that describes how wrong a model's Prediction are. the goal of the model is to minimize the loss reducing to the lowest possible.

***Distance of Loss :** loss measures the diff b/w the values, not the direction for ex : if Predicted value is 2 but the actual value is 5 we get loss as $-3(2-5=-3)$ but we do not care about the sign we only care about the value no the sign.

Types of Losses :

L1 Loss	$\sum \lvert \text{actual value} - \text{Predicted value} \rvert$
Mean absolute Error	$\frac{1}{N} \sum \lvert \text{actual value} - \text{Predicted value} \rvert$
L2 Loss	$\sum [\text{actual value} - \text{Predicted value}]^2$
Mean square error	$\frac{1}{N} \sum (\text{actual value} - \text{Predicted value})^2$

that Produce the model

The model begins training with randomized weights and biases near zero,

1.) calculate the Loss with current weight and bias
2.) Determine the direction to move to reduce Loss
3.) Move the W and B a small amount in direction that ↓ loss.
4.) Reduce and Repeat until the model can't reduce Loss.

By taking Previous data set Values

step-1 : consider

weight = 0

bias = 0

$y = Wi$

(MSE) Loss = $\frac{(18-0)^2 + (15-0)^2 + (13-0)^2 + (16-0)^2 + (15-0)^2 + (14-0)^2 + (24-0)^2}{7}$

step-2 : Loss = 303.71

step-3 : calculate the slope of tangent to each weight and bias

$f_{(w,b)}(x) = w*x + b$

$L = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2 \quad N=7$

$\frac{dL}{dw} = \frac{-2}{N}\sum_{i=1}^{N} x_i(y_i - \hat{y_i})$ $\quad$ $\frac{dL}{db} = \frac{-2}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})$

$L = \frac{1}{N}\sum_{i=1}^{N} -x_i(y_i - \hat{y_i})^2$

$\frac{\partial L}{\partial w} = \frac{-2}{N}\sum_{i=1}^{N} x_i(y_i - \hat{y_i})$

here $w=0$ so $\hat{y_i} = 0$

$\frac{\partial L}{\partial w} = \frac{-2}{N}\sum_{i=1}^{N} x_i y_i$

$\sum x_i y_i = 419.01$

$\frac{-2}{7}(419.01) = \frac{\partial L}{\partial w}$

$\frac{\partial L}{\partial w} = -119.7171$

$L = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2 - 2$

$L = \frac{-2}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})$

here $b=0$ so $\hat{y_i} = 0$

$\frac{\partial L}{\partial b} = \frac{-2}{N}\sum_{i=1}^{N} y_i$

$\sum y_i = 120$

$\frac{\partial L}{\partial b} = \frac{-2}{7} \times 120$

$\frac{\partial L}{\partial b} = -34.287$

step-4 : Move small amount in the direction of ~~Ave~~ slope to get next weight and bias

New weight = old weight - (small change * weight slope)

for ex 0.01

New weight = 0 - (0.01 * -119.7)

New bias = 0 - (0.01 * -34.3)

New weight = 1.2

New bias = 0.34

* Gradient descent :

GD is a mathematical technique that iteratively finds the weights and bias that Produce the model with the Lowest Loss.

the model begins training with randomized weights and biases near zero,

1.) calculate the Loss with current weight and bias
2.) Determine the direction to move to reduce Loss
3.) Move the W and B a small amount in direction that ↓ loss.
4.) Reduce and Repeat until the model can't reduce Loss.

By taking Previous data set Values

step-1 : consider

weight = 0

bias = 0

$y = wx$

(MSE) Loss = $\frac{(18-0)^2 + (15-0)^2 + (18-0)^2 + (16-0)^2 + (15-0)^2 + (14-0)^2 + (24-0)^2}{7}$

step-2 : Loss = 303.71

step-3 : calculate the slope of tangent to each weight and bias

$f_{(w,b)}(x) = w * x + b$

$L = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \quad N=7$

$\frac{dL}{dw} = \frac{-2}{N} \sum_{i=1}^{N} x_i (y_i - \hat{y}_i) \qquad \frac{dL}{db} = \frac{-2}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)$

here $y_i = w_i x_i + b$

for b = 0

$\hat{y}_i = w_i x_i$

$L = 1/N \sum_{i=1}^{N} (y_i - w_i x_i)^2$

$L = 2/N \sum_{i=1}^{N} -x_i (y_i - \hat{y}_i)^2$

$\frac{\partial L}{\partial w} = \frac{-2}{N} \sum_{i=1}^{N} x_i (y_i - \hat{y}_i)$

here w = 0 so $\hat{y}_i = 0$

$\frac{\partial L}{\partial w} = \frac{-2}{N} \sum_{i=1}^{N} x_i y_i$

$\sum x_i y_i = 419.01$

$\frac{-2}{7} (419.01) = \frac{\partial L}{\partial w}$

$\frac{\partial L}{\partial w} = -119.7171$

for w = 0

$\hat{y}_i = b_i$

$L = 1/N \sum_{i=1}^{N} (y_i - b_i)^2$

$L = 1/N \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 - 2$

$L = -\frac{2}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)$

here b = 0 so $\hat{y}_i = 0$

$\frac{\partial L}{\partial b} = \frac{-2}{N} \sum_{i=1}^{N} y_i$

$\sum y_i = 120$

$\frac{\partial L}{\partial b} = \frac{-2}{7} \times 120$

$\frac{\partial L}{\partial b} = -34.287$

step-4 : Move small amount in the direction of the slope to get new weight and bias

New weight = old weight - (small change * weight slope)

for ex 0.01

New weight = 0 - (0.01 * -119.7)

New bias = 0 - (0.01 * -34.3)

New weight = 1.2

New bias = 0.34

Import numpy as np

X = np.array

Y = np.array

N = len(X)

W, b = 0.0, 0.0

alpha = 0.01

for i in range (6):

Y_pred = W * X + b

dW = -2/N * np.sum (x * (Y - Y_pred))

db = -2/N * np.sum (Y - Y_pred)

W = W - alpha * dW

b = b - alpha * bW

mse = np.mean ((Y - Y_pred) ** 2)

Print ("iter", i+1, ": W =", round (W, 4), ": b =", round (b, 4),

"MSE =", round (mse, 4))

* Convergence and cover functions :

The Loss function for linear models always Produce a convex surface. As a result when model converges we know that model has found the weig, bias for Lowest Loss.

If we graph the Loss surface for a model with one feature, we can see it's convex shape.

Additional iterations only cause GD to move the weights and bias in very small amounts around them min.

* Hyper Parameters :

they are Variables that control different aspects of training

→ Learning rate
→ Batch size
→ Epochs

"

hyperparameters are Values that we control, Parameters are Values that the model calculates during training".

* Learning Rate :

It is a floating Point number we set that how quickly the model converges.

If the Learning rate is too Low the model takes longtime to converge. However, If it is too high the model never converges but bounce around the weights and bias te minimize the Loss.

It determines the mag of the changes to make to the weights and bias during each step of GD Process. the model multiply the LR with gradient to determine model parameters for next iteration. in third iteration step the small value to move in-ve direction of slope refers to Learning rate.

Slope of loss $\propto$ new model parameters
Parameters

if slope is large the model takes Large step, if the slope is small it takes small steps.

* **Batch size :**

Batch size is a Hyperparameter that refers to the number of examples that model Processes before updating it's weight and bias. When a dataset contains hundreds of thousands or even millions of examples, using full batch isn't Practical.

Two common techniques to get the right gradient on average without needing to look at every example in dataset are

→ Stochastic gradient descent

→ mini-batch stochastic gradient descent

* **SGD** : It uses only a single example (a batch size of one) per iteration. Given enough iterations SGD works but it's very noisy. "Noise" refers to variations during training the ↑ the Loss rather than ↓ for iteration, "stochastic" tells that one ex compromising each batch chosen at random.

* **mini-batch SGD** : Mini-batch SGD is a compromise b/w full batch and SGD, for N no of data Points, the batch size can be any number greater than 1 Less than N. The model chooses the examples included in each batch at random, average their gradients, and then update weights and bias once per iteration.

* **Epochs** : An epoch means that the model has Processed every example in the training set once. for ex, given a training set of 1000 examples each of batch size of 100 examples it will take model 10 iterations to complete 1 epoch.

The system needs to process every example in the training set multiple times.