How Classification Works :-

Amazon Dataset

$r_i \longrightarrow$ Text $\longrightarrow$ Vectors

$\hookrightarrow$ +ve/-ve

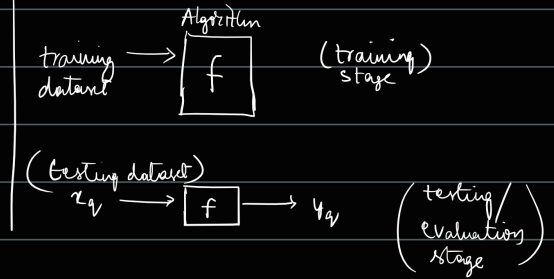
$\rightarrow r_i \longrightarrow (v_i, +ve/-ve)$

{Given a review Text, determine/predict if review is +ve/-ve

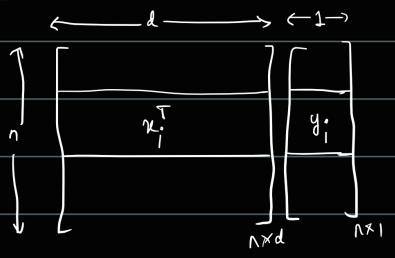
$\rightarrow y = f(x)$

↳ In most ML problems, we are trying to find this f() function.

Amazon Dataset :-



Data Matrix Notation :-



$r_i \longrightarrow$ Text $\rightarrow$ vectors

↳ class (+ve/-ve)

$y = f(x)$

For Amazon Dataset, $D = \{(x_i, y_i)_{i=1}^{n} \mid x_i \in \mathbb{R}^d, y_i \in \{0,1\}\}$

$r_i \longrightarrow$ vector x_i

$y = f(x)$

→ If target classes = 2, (like Amazon Dataset), classifications is called Binary classifications

= n, " " Multi Class Classifications. (IRIS dataset, MNIST dataset)

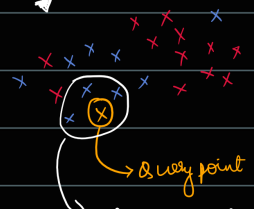
If y_i is not finite, it becomes regression. $y_i \in \mathbb{R}$ not a set of finite values.

→ classification → Discrete values, Regression → continuous variable

→ Algorithms like Decision trees, random forests can perform classifications as well as regression

K-Nearest Neighbours :-

Assume $D = \{(x_i, y_i) \mid x_i \in \mathbb{R}^2, y_i \in \{0,1\}\}$



X → positive (1)

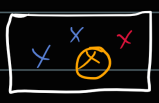
X → Negative (0)

↳ Neighbourhood of x_q. Geometrically all points close to x_q are positive.

We can conclude x_q is positive

Steps in KNN :-

① Find 'K' nearest points. Let K = 3 in this example. = $\{x_1, x_2, x_3\}$



$\{y_1, y_2, y_3\}$ → In this case, majority = positive

② Majority vote b/w $\{y_1, y_2, y_3 \ldots, y_K\}$

(Avoid taking even 'K', as there could be a chance of tie)

→ In scikit learn, there are 3 methods to calculate nearest neighbours

(i) Brutoforce :- Effective in small datasets.

(ii) KD tree :-

> distance information for the sample. The basic idea is that if point A is very distant from point B, and point B is very close to point C, then we know that points A and C are very distant, *without having to explicitly calculate their distance*. In this way, the

(iii) Ball tree :-

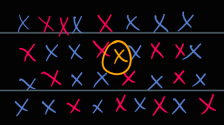
> A ball tree recursively divides the data into nodes defined by a centroid C and radius r, such that each point in the node lies within the hyper-sphere defined by r and C. The number of candidate points for a neighbor search is reduced through use of the *triangle inequality*:
>
> $$|x+y| \leq |x| + |y|$$

→ KNN is not used in realtime applications as it is a lazy algorithm.

Failure cases of KNN :-

→ when point is very far away from rest of the points. Taking the nearest neighbours doesn't make much sense.

→ Very jumbled randomly spread data.



No useful information in this type of spread.

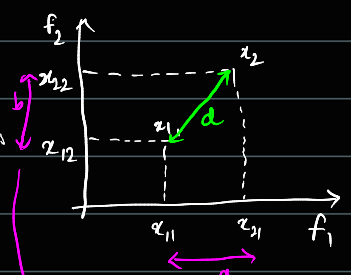
Most ML algos fail when data is like this.

→ Determining threshold.

Note:-Determining Threshold is Data-Specific

1. Take Train Data
2. Take each point in data-set and compute its distance from its first neighbor and store all these distance
3. Sort them, some are small and some are large
4. Take 99.99% percentile as your threshold, let's name it D-99
5. Now for our Query point, if its distance from its first neighbor is greater than D-99 then take it as an outlier and don't classify it

Distance Measures :-



$$x_1 = (x_{11}, x_{12}) \quad x_2 = (x_{21}, x_{22})$$

(f_1, f_2)

d = len of shortest line from x_1 to x_2

$$d = \sqrt{(x_{21}-x_{11})^2 + (x_{22}-x_{12})^2} = \|x_1 - x_2\| \longrightarrow \text{Euclidian distance.}$$

if $x_1 \in \mathbb{R}^d$ & $x_2 \in \mathbb{R}^d$, $\|x_1 - x_2\|_2 = \left(\sum_{i=1}^{d} (x_{1i} - x_{2i})^2\right)^{1/2}$

L2 Norm of a vector

→ if only one vector ie; $\|x_1\|_2$ → distance of x_1 from origin $= \left(\sum_{i=1}^{d} x_{1i}^2\right)^{1/2}$

(a+b)

Manhattan Distance :- $\sum_{i=1}^{d} |x_{1i} - x_{2i}|$.

Also written as $\|x_1 - x_2\|_1$. L1 Norm. $\|x_1\|_1 = \sum_{i=1}^{d} |x_i|$

L_p Norms :- corresponding value is called Minkowski Distance

$$\|x_1 - x_2\|_p = \left(\sum_{i=1}^{d} |x_{1i} - x_{2i}|^p\right)^{1/p}$$

If p=2, Minkowski distance = Euclidian Distance

If p=1, Minkowski distance = Manhattan Distance.

$$\|x_1\|_p = \left(\sum_{i=1}^{d} |x_{1i}|^p\right)^{1/p}$$

→ Distances are b/w two points. Norms are b/w two vectors.

ex:- Euclidian distance (x_1, x_2) = L2 Norm of $(x_1 - x_2) = \|x_1 - x_2\|_2$

Hamming Distance :-

Used for Boolean Values (ex:- Binary Bag of Words)

$x_1 = [0,1,1,0,1,0,1,1,1]$

$x_2 = [1,0,1,1,0,1,0,0,1]$

Hamming Dist (x_1, x_2) = # of locations where Binary Vectors differ.

For above example :- HD = 7

→ Can also be used for strings - Number of locations where strings differs

→ can also be used for Gene Sequencing. They are basically sequence of letters

→ Sklearn kNN uses Minkowski by default.

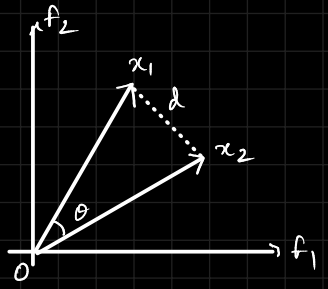
Cosine Similarity and Cosine Distance :-

→ As similarity increases, distance decreases. Similarity $\propto \frac{1}{\text{distance}}$

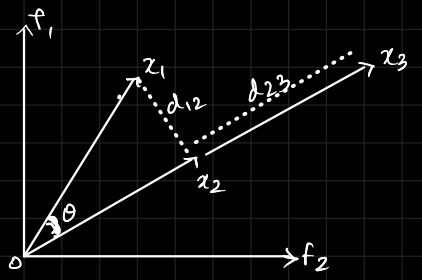
→ 1 − cos-similarity (x_1, x_2) = cos-dist (x_1, x_2)

this lies b/w $[-1, 1]$. If they are very similar, cos-sim = +1
if they are very dissimilar, cos-sim = −1

→



d = Euclidian distance
cos-similarity = cos(θ)



cos-sim (x_1, x_2) = cos(θ)

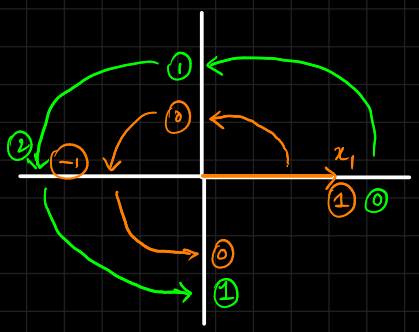
cos-sim (x_2, x_3) = 1 $(\cos(0) = 1)$

cos-dist (x_2, x_3) = 1 − cos-sim (x_2, x_3)

= 1 − 0 = 1

But $d_{12} < d_{23}$ (Euclidian)

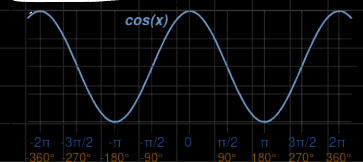
→



green = cos-dist
orange = cos-sim

angle b/w (x_1, x_2)	cos-dist
0° − 90°	0 → 1
90° − 180°	1 → 2
180° − 270°	2 → 1
270° − 360°	1 → 0

cosine - graph :-



→ We know that $\cos(\theta) = \frac{x_1 \cdot x_2}{\|x_1\|_2 \, \|x_2\|_2}$

↳ if no L_p is mentioned, it is 2 by default.

if x_1, x_2 are unit vectors, $\cos(\theta) = x_1 \cdot x_2$

→ Relationship b/w euc-dist & cos-sim :-

if x_1 & x_2 are unit vectors, $[\text{Euc-dist}(x_1, x_2)]^2 = 2(1 - \cos(\theta)) = 2(\text{cos-dist})$

→ Whenever we find cosine distance, we need to perform row normalization → converting them to unit vectors basically.