

How Classification Works:-

Amazon Dataset

Amazon Dataset :-

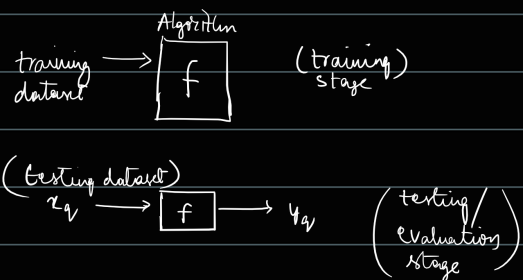
$r_i \rightarrow \text{Text} \rightarrow \text{Vectors}$
 $\rightarrow +ve/-ve$

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

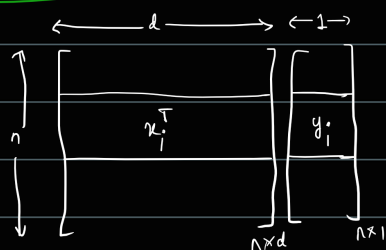
{ Given a review text, determine/predict if review is +ve/-ve

$\rightarrow y = f(x)$

In most ML problems, we are trying to find this $f()$ function -



Data Matrix Notation:-



$r_i \rightarrow \text{Text} \rightarrow \text{vectors}$

\hookrightarrow class (+ve/-ve)

$y = f(x)$

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

$x_i \rightarrow \text{vector } x_i$

$y = f(x)$

\rightarrow If target classes = 2, (like Amazon Dataset), classification is called Binary Classification

= n, " " " Multi-Class Classification (IRIS dataset, MNIST dataset)

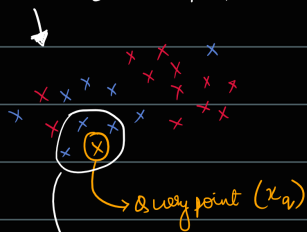
If y_i is not finite, it becomes regression. $y_i \in \mathbb{R}$ not a set of finite values.

\rightarrow Classification \rightarrow Discrete values, Regression \rightarrow continuous variables

\rightarrow Algorithms like Decision trees, random forests can perform classification as well as regression.

K-Nearest Neighbours:-

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



$x \rightarrow \text{positive}(1)$

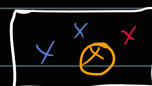
$x \rightarrow \text{Negative}(0)$

Neighborhood 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. $= \{x_1, x_2, x_3\}$



$\{y_1, y_2, y_3\}$

\rightarrow In this case, majority = positive

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

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

→ In Scikit Learn, there are 3 methods to calculate nearest neighbours

(i) Bruteforce :- 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 real-time applications as it is a very slow 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.

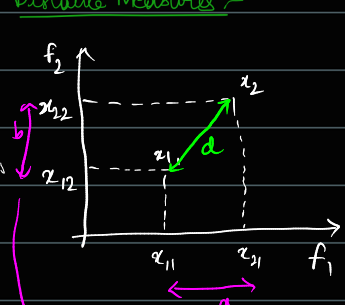


→ Determining threshold.

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 distances
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 :-



$$x_1 = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix} = (x_{11}, x_{12})$$

$$x_2 = (x_{21}, x_{22})$$

d = len of shortest line from x_1 to x_2

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

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

L2 Norm of a vector

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

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

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

Lp 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 = Euclidean 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:- Euclidean distance $(x_1, x_2) = \text{L2 Norm of } (x_1 - x_2) = \|x_1 - x_2\|_2$

Hamming Distance :-

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

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

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

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 differ
- Can also be used for Gene Sequencing. They are basically sequence of letters
- sklearn KNN uses Minkowski by default.