KNN

K-Nearest Neighbor is simplest algorithm

Classification algorithm for numerical data

Supervised learning algorithm

It assumes the similarity between the new case / data and available cases & puts the new cases into the catagory that is most similar to the available catagories

Classifies a data point based on its similarity in the form of a distance heuristic

Non Parametric algorithm ie does not make any assumptions on the data
(eg in linear regression we made assumptions on values of W & each time we refine our algorithm)

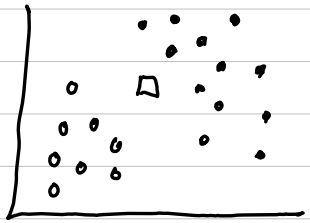
No training is done
Only evaluation is done
} Lazy learning algorithm

Takes a lot of time for its prediction if the dataset is very very large

KNN

K - Nearest Neighbor

If there is a data spread of huge data



where will sample □ be placed?

eg

Percentile of marks	student get placed
6 - 7	Yes
5 - 6	no
7 - 8	yes

Now a new sample comes → % 6.8
will the student get placed?

It will be decided on class 6 - 7

We can find the distance of the value 6.8 from the classes

Average Marks	k - avg marks $\lvert x - x_i \rvert$
6.5	0.3
5.5	1.3
7.5	0.7

The distance $|x - x_i|$ is called as manhatten distance

Minimum distance is in class 5-6 so it will decide accordingly

So every data point is treated as an coordinate

Distance of every point from the data point is measured

Then based on the value of k, the nearest neighbors are found

Classification is done on basis of nearest neighbor

Here k=1 nearest neighbor is class 5-6

Types of distance

Manhattan $|x - x_1| + |y - y_1|$

or $\sum |x_1 - x_2|$

Euclidian distance $\sqrt{(x-x_1)^2 + (y-y_1)^2}$

or $\sqrt{\sum (x - x_i)^2}$

This data was a simple 1D data
In higher dimentions the data points translate into coordinates in an f dimentional space where f is the number of features

We can classify a new data based on the distance between the coordinates of sample points

Consider sample table

x_1 = acid durability
x_2 = strength (kg/m^2)
y = classification B → Bad, G → good

	x_1	x_2	y
S_1	7	7	B
S_2	7	4	B
S_3	3	4	G
S_4	1	4	G

Classify $x_1 = 3$ & $x_2 = 7$

→ Calculate Nearest neighbor $k = 3$

Here value of k is number of nearest neighbours

Calculate distance between (3,7) with all the input samples

lets consider the euclidian distance

We will find euclidian distance of the point from other points

$$S_1 = \sqrt{(7-3)^2 + (7-7)^2} = 4$$

$$S_2 = \sqrt{(7-3)^2 + (4-7)^2} = 5$$

$$S_3 = 9 \quad , \quad S_4 \cong 3.7$$

	X_1	X_2	d^2	d	Rank	Included in Neighborhood
S_1	7	7	16	4	3	✓
S_2	7	4	25	5	4	✗ $4 > k$
S_3	3	4	9	3	1	✓
S_4	3	4	13	3.7	2	✓

k is the number of samples to consider

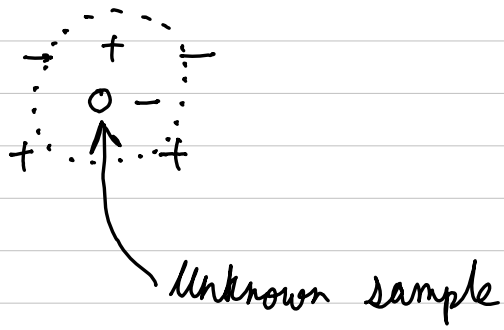
Here $k = 3$, so we consider first 3 neighbors ie. S_3, S_4, S_1 (Good, Good, Bad)

Majority of the label is Good

Consider the label with majority

k-NN is based on the principle

"Man is known by the company he keeps"



(1)

	Brightness	Saturation	Class
1.	40	20	Red
2.	50	50	Blue
3.	60	90	Blue
4.	10	25	Red
5.	70	70	Blue
6.	60	10	Red
7.	25	80	Blue

Classify (20, 35) for $k = 5$

→

	d^2	d	Rank	In neighborhood	Class
1.	625	25	2	✓	R
2.	1125	33.54	3	✓	B
3.	4625	⋮	7		B
4.	200	⋮	1	✓	R
5.	3725	⋮	6		B
6.	2225	⋮	5	✓	R
7.	2050	⋮	4	✓	B

R, B, R, R, B → Majority → R

Hence point will be classified as Red

Instance Based Learning

We need to calculate the distance from the points every time for every instance at time of prediction

No prior rules (like decision tree) or parameters (like ANN, linear regression) are learnt

No Training on train data is done, no model weights

All work is at the time of inference

Hence the algorithm is called as lazy learning

Every time, the distances have to be calculated, making the time complexity of inference

$O(N)$ $N \rightarrow$ size of dataset

Inference

Majority voting can be applied to multiclass as well

$$y = \max_c \sum_{i=1}^{k} \begin{cases} 1 \text{ if } y_i = c \\ 0 \text{ if } y_i \neq c \end{cases}$$

Prediction can be improved if the influence of each neighbor is weighted

$$y = \frac{\sum w_i y_i}{\sum w_i}$$

Weights depend on distance eg $w_i = \frac{1}{d(x_i, x)^2}$

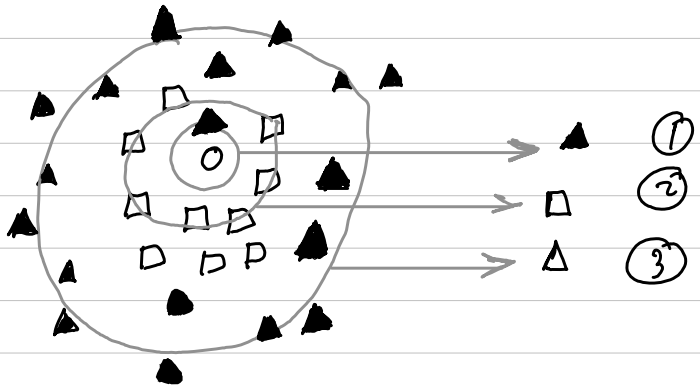
Further points should have lesser influence

The idea of classification can be extended to regression by taking average value of classes from k neighbours

Choosing the value of k

If k is too small, then model will be sensative to noise

If k is too large then model will suffer



(1) → too small circle (k) hence thinks that it is ▲

(2) → Perfect classifier

(3) → too large k hence doesnt classify properly

Correct value of k must be found by crossvalidation

KNN Imputer

Used to impute missing values & fill incomplete data

eg.

Age	Income	Buy Computer?
Young	??	No
Old	Low	Yes
Middle	High	Yes

What will be the missing value of income?

Model (eg decision tree) needs us to get rid of all missing values from dataset before it can train on the data and predict if person May buy computer or not

Hence we can guess value of Income or remove row. Removing row will reduce the data

K-NN is used to fill the missing values by checking what its neighbours do.

KNN needs numeric data. Hence the catagorical data must be numeric encoded.

KNN requires us to normalize the data, else biased replacements may be found.

KNN imputer replaces the NaN values with the nearest neighbour estimated values

Since KNN doesnt make any assumptions about data distribution, it can be applied to a wide range of data values.

Advantages → No training time
Simple model
Explainable
No assumptions on data made
(eg gaussian distribution like GMM)

Disadvantages → Inference time is high for large datasets O(N)

Sensative to noise & missing data

Doesnt work well with high dimentions

Applications → Recommendation systems

Data preprocessing (kNN imputer)

Credit rating