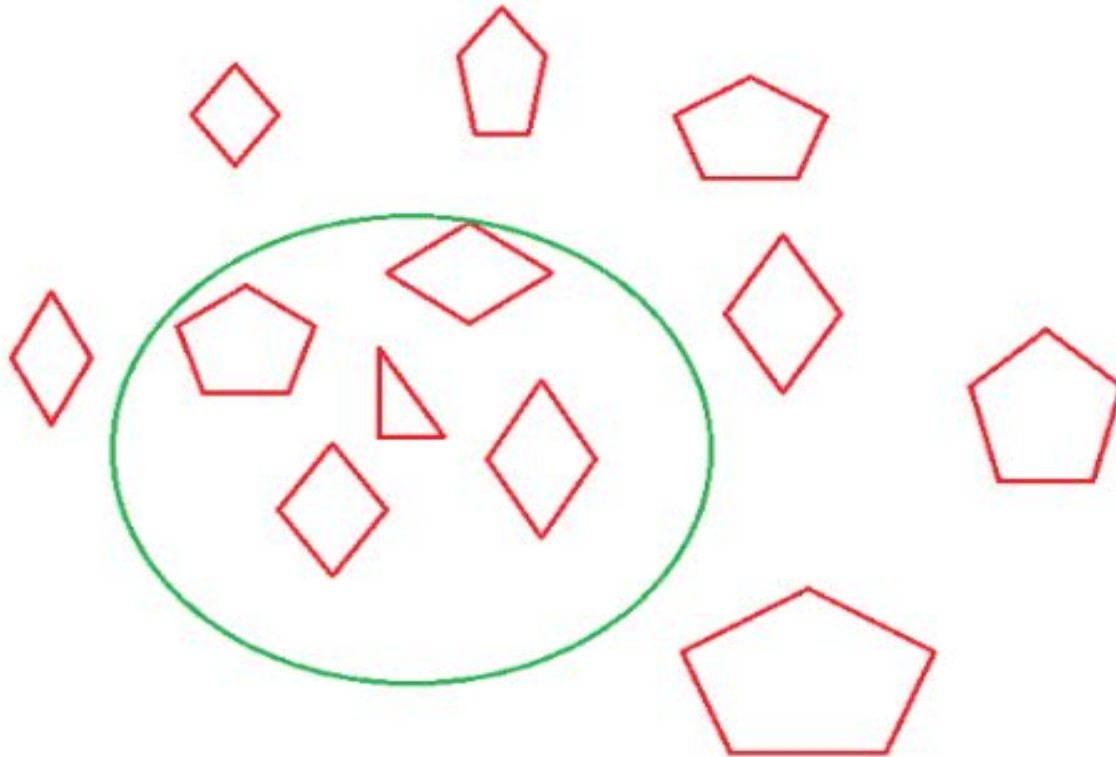


KNN

- One of the simplest ML algorithms based on Supervised Learning technique.
- At the training phase, K-NN algorithm stores all the available data (dataset)
- When a new case/data point appears, it classifies the new data point based on the similarity
 - Assigns the category from the available categories that is most similar among from the nearest neighbors.
- **a lazy learner algorithm**

Contd..

- K in KNN represents the number of the nearest neighbors used to classify new data points.
- Different k values can and will produce different classifications
- K=4



KNN algorithm

- The K-NN working can be explained on the basis of the below algorithm:
 - **Step-1:** Determine parameter k = number of nearest neighbors.
 - **Step-2:** Calculate the distance between the query instance (new data point) and all the training samples.
 - **Step-3:** Take the K nearest neighbors as per the calculated distance. (Sort the distance and determine the nearest neighbors)
 - **Step-4:** Gather the category of the nearest neighbors
 - **Step-5:** Use simple majority of the category of nearest neighbors as the prediction value of the query instance.

1. Selection of K value

- Choosing the optimal value for K is best done by first inspecting the data.
- Choosing the right value of K is called **parameter tuning** and it's necessary for better results
- In general, a large K value is more precise as it reduces the overall noise but there is no guarantee.
 - Historically, the optimal K for most datasets has been between 3-10. That produces much better results than 1NN.
 - The most preferred value for K is 5.

KNN – Number of Neighbors

- If $K=1$, select the nearest neighbour
- If $K>1$,
 - For classification select the most frequent neighbour.
 - For **regression** calculate the average of K neighbours.
- **Rule of thumb is $k < \sqrt{n}$** , n is number of examples
 - Odd value of K is always selected to avoid confusion between 2 classes.

2. Distance Calculation

- There are various methods for calculating the distance between the new point and each training point
- The most commonly known methods are:
 - Euclidian (for continuous)
 - Manhattan (for continuous)
 - Hamming distance (for categorical)

Distance functions

Euclidean

$$\sqrt{\sum_{i=1}^k (x_i - y_i)^2}$$

Manhattan

$$\sum_{i=1}^k |x_i - y_i|$$

Minkowski

$$\left(\sum_{i=1}^k (|x_i - y_i|)^q \right)^{1/q}$$

Hamming Distance

- It is used for categorical variables
- If the value (x) and the value (y) are the same, the distance D will be equal to 0 . Otherwise D=1.

$$D_H = \sum_{i=1}^k |x_i - y_i|$$

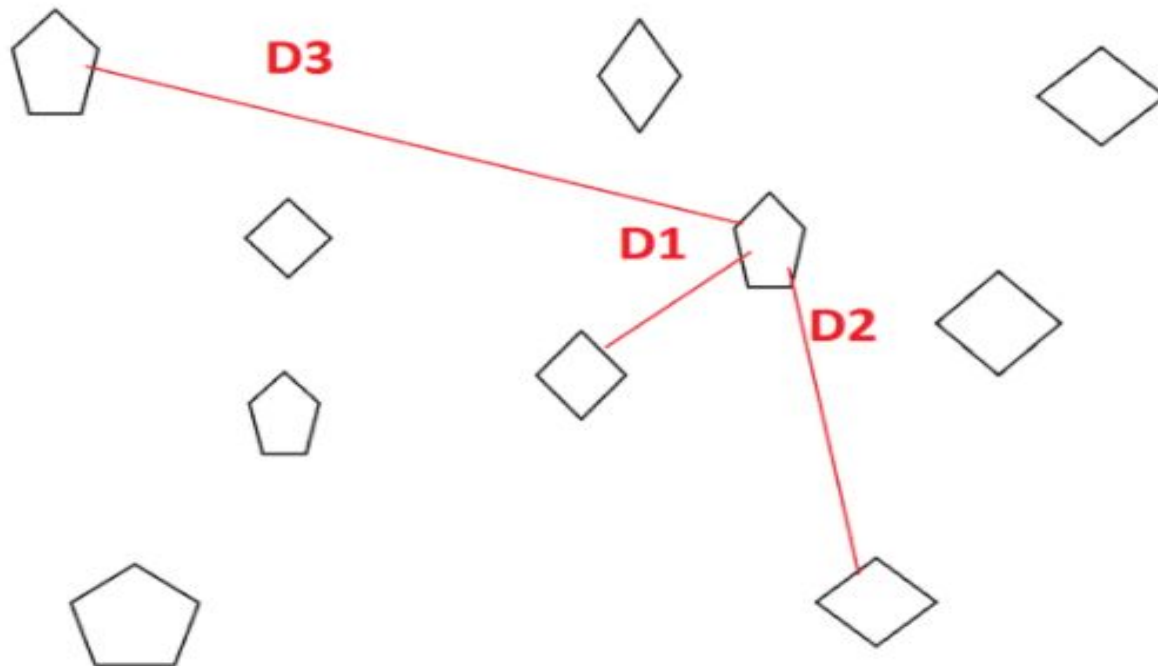
$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

Contd..

- ❑ We usually use **Euclidean distance** to calculate the nearest neighbor.
- ❑ If we have two points (x, y) and (a, b). The formula for Euclidean distance (d) will be


$$d = \sqrt{(x - a)^2 + (y - b)^2}$$



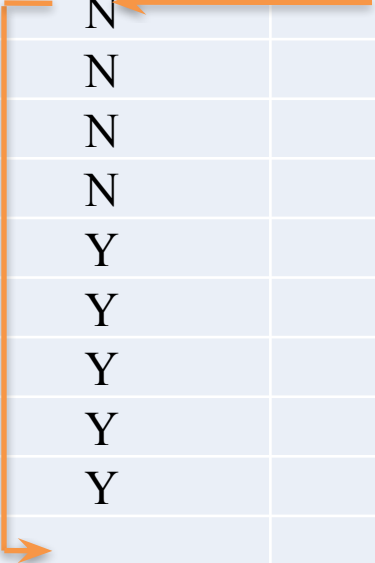
Distance scaling

- One major drawback in calculating distance measures directly from the training set is in the case where variables have different measurement scales or there is a mixture of numerical and categorical variables.
- For example, if one variable is based on annual income in dollars, and the other is based on age in years.
- Then income will have a much higher influence on the distance calculated. One solution is to standardize the training set as shown below

Normalized Distance



Age	Loan	Default	Distance
0.125	0.11	N	0.7652
0.375	0.21	N	0.5200
0.625	0.31	N	0.3160
0	0.01	N	0.9245
0.375	0.50	N	0.3428
0.8	0.00	N	0.6220
0.075	0.38	Y	0.6669
0.5	0.22	Y	0.4437
1	0.41	Y	0.3650
0.7	1.00	Y	0.3861
0.325	0.65	Y	0.3771
0.7	0.61	?	



Normalized Variable

$$X_s = \frac{X - Min}{Max - Min}$$

Nearest Neighbor Complexity

- Expensive for high dimensional data ($d > 20$?)
- $O(Knd)$ complexity for both storage and query time
 - n is the number of training examples,
 - d is the dimension of each sample
 - $O(d)$ to compute distance to one example.
 - $O(nd)$ to compute distance to n numbers of examples
 - $O(knd)$ to find k closest examples
 - Thus total complexity is $O(knd)$.

Advantages/Disadvantages

- **Advantages:**

- Simple to implement
- Training is very fast
- Learns very quickly;
- Provides good generalization accuracy
- Robust to noisy training data;
- Don't lose information

- **Disadvantages:**

- Slow at query
- Always needs to determine the value of K which may be complex some time.
- The computation cost may be high because of calculating the distance between the data points for all the training samples.
 - Expensive, High Storage Requirements

When is KNN?

- Data is **properly labeled**.
 - For example, if we are predicting someone is having diabetes or not the final label can be 1 or 0. It cannot be NaN or -1.
- Data is **noise-free**.
 - For the diabetes data set we cannot have a Glucose level as 0 or 10000. It's practically impossible.
- Comparatively Small dataset.

Some KNN applications

- Classification and Interpretation
 - legal, medical, news, banking, movie prediction for a movie fan
- Problem-solving
 - planning, pronunciation
- Teaching and aiding
 - help desk, user training