Classification Algorithm in various situations

1. What is Imbalanced and balanced dataset?

Algorithm may get biased if we have alot of +ve or -ve point in our data set.

ex: lets say we have alot of positive point, then algorithm will always predict +ve for future unseen data

D_n — +ve (9900)
D_n — -ve (100)

1. Balanced data set (2 class classification)

$D_n = n_1 + n_2$ (↓ +ve, ↓ -ve)

if $n_1 \approx n_2$ { its okay if $n_1 \neq n_2$ but it roughly similar

2. Imbalanced dataset:-

if $n_1 << n_2$ or $n_2 << n_1$ { there is very much difference in # of points

ex: K-NN with (imbalanced data)

$n_1 = 50$ (+ve) $n_2 = 950$ (-ve)

* model will be biased towards -ve because of majority vote
* Dominant class.

Problem: You can get high accuracy with imbalanced data with a dumb model

How to work around imbalanced dataset?

1. undersampling:-

D_n — n_1 (100) → minority
D_n — n_2 (900) → majority

Randomly Sample 100 pts from n_2 →

n_1	n_2	
100	900	D_n

↓ undersampling

n_1	n_2	
100	100	D'_n

Problem:- * we are throwing away 800 pts which should be available
* There are very few points in D'_n so model may not even work

Note: Throwing away data should be avoided.

2. Oversampling

D_n — 900 (n_1) +ve
D_n — 100 (n_2) -ve

{ 100 -ve pts (n_2) ↓ repeat 9 times ↓ 900 -ve pts

D'n : 900 +ve pts
900 -ve pts

* placing more points from minority class in the dataset

1. replacing points from minority class
2. Extrapolation (creating artificial or synthetic points)

3. Weightage (not available for KNN)

Dn — 900 (-ve pts) — $w_{+ve} = 9$ → more weight to minority class
100 (+ve pts) — $w_{-ve} = 1$ → less weight to majority class

It is equivalent to repeatation

2. Define Multi-classclassification?

1. Binary classification : $y_i \in \{0, 1\}$
2. Multiclass classification : $y_i \in \{0, 1, 2, 3 \ldots k\}$

Some algorithms can only do binary classification (ex: logistic regression)
→ we can do One Vs rest

 # KNN Given a distance measure.

Sometimes we don't get x_i as a numerical vector, but we get it as Similarity matrix

As K-NN cares mostly about distance or similarity
→ Given a sim-matrix or distance matrix explicitly → K-NN can work

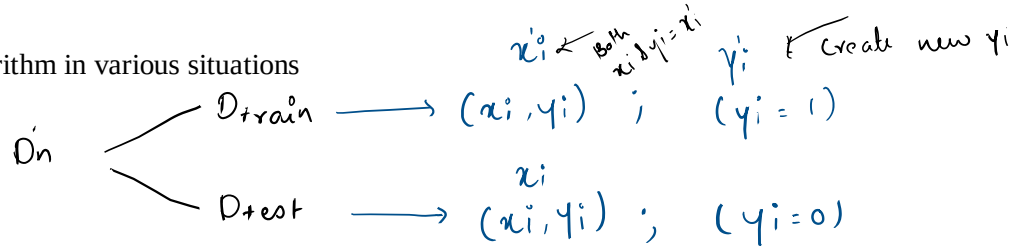
$$\text{distance} = \frac{1}{\text{similarity}}$$

3. How to determine if data is changing over time? Or Distribution (Dtrain) != Distribution (Dtest)?

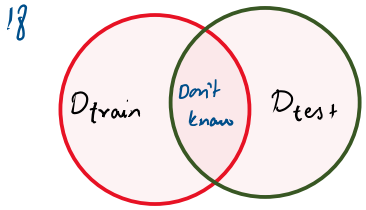
Dn — Dtrain (x_i, y_i)
Dtest (x_i, y_i)

1) Create a new data set using Dn to check the distribution

Classification Algorithm in various situations



② Create a binary classifier on D'_n



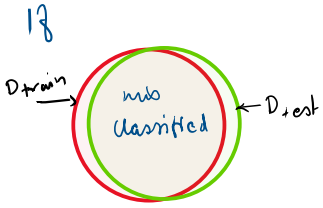
$f_x \longrightarrow 1$ or 0

High error means similar distribution

③ if binary classification has accuracy of 70% ie 30% misclassified

This mean train and test data cannot be seperated completely

∴ distribution is not very similar



→ if is almost overlapping

→ Binary classification accuracy low

This means D_{train} & D_{test} are coming from same distribution

Higher the misclassification mean similar distribution

4. Explain Impact of Outliers?

Outliers can change the model drastically as it can easily affect decision surface

for k-NN →
* if k is small : outlier can easily impact model
* if k is large : it will be less prone to outliers.

5. What is LocalOutlier Factor?

ⓐ k-distance:

k-distance (x_i) = distance to the k^{th} nearest neighbour of x_i from x_i

$N(A)$ = neighbourhood of A - - - set of all point that belong to k-NN of x_i

(b) Reachability distance :-

$$\text{Reachability distance}(x_i, x_j) = \max(k\text{-distance}(x_j), \text{distance}(x_i, x_j))$$

If x_i is in neighbourhood of x_j ($N(x_j)$) then reachability distance is k^{th} distance of x_j (i.e farthest distance of nearest neighbour)

else, reachability distance is equal to distance between (x_i, x_j)

(c) Local reachability density → lrd (A)

$$lrd(x_i) = \frac{1}{\sum_{x_j \in N(x_i)} \left\{ \frac{\text{reachability distance}(x_i, x_j)}{|N(x_i)|} \right\}}$$

size of set or elements of set

↳ Generally k but not necessarily because there can be multiple point at same distance.

Denominator: Avg reachability distance of x_i from its neighbours.

lrd(x_i): inverse of average reachability distance of x_i from its neighbours

$$lrd(x_i) = \frac{N(x_i)}{\sum_{x_j \in N(x_i)} (\text{reachability distance}(x_i, x_j))}$$

$N(x_i)$ ← # of points

(density)

local

reachability distance

∵ density = $\frac{\text{\# of point}}{\text{unit area}}$

(d) Local outlier factor (LOF)

$$LOF = \frac{\sum_{x_j \in N(x_i)} lrd(x_j)}{|N(x_i)|} \propto \frac{1}{lrd(x_i)}$$

Avg lrd of point in neighbourhood of x_i

lrd of x_i

If LOF(x_i) → large → outlier
↳ small → inlier

Classification Algorithm in various situations

1. for each point (xi) compute LOF (xi)
2. pick points with highest LOF - → OUTLIER

Disadvantages:-

* LOF is very hard to interpretate
* use Local outlier probability (modified LOF)