KNN (K-Nearest Neighbours)

What is KNN (K-Nearest Neighbour) Algorithm?

The K-Nearest Neighbor (KNN) algorithm is a popular machine learning technique used for classification and regression tasks. It relies on the idea that similar data points tend to have similar labels or values.

During the training phase, the KNN algorithm stores the entire training dataset as a reference. When making predictions, it calculates the distance between the input data point and all training examples, using a chosen distance metric such as Euclidean distance.

Next, the algorithm identifies the k nearest neighbors to the input data point based on their distances. In the case of classification, the algorithm assign the most common class label among the k neighbors as the predicted label ~~for~~ for the input data point. For regression, it calculates the average or weighted average of the target value of the k neighbors to predict the value for the input datapoint.

The KNN algorithm is straightforward and easy to understand, making it a popular choice in various domains. However, its performance can be affected by the choice of k and the distance metric, so careful parameter tuning is necessary for optimal results.

When Do We Use the KNN Algorithm?

KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problem in the industry. To evaluate any technique, we generally look at 3 important aspects:

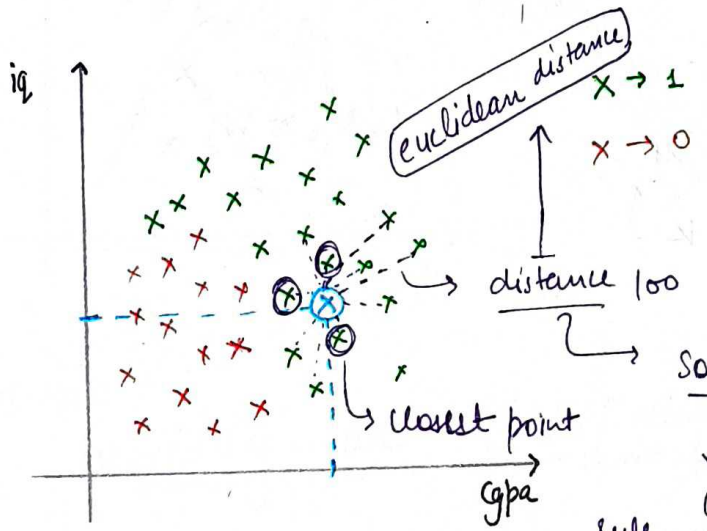
1. Ease of interpreting output
2. calculation time
3. Predictive Power

KNN classifier fair across all parameter of consideration, It is commonly used for its ease of interpretation and low calculation time.

KNN Intution

100 student

cgpa	iq	placement
8	80	1
7	70	0
⋮	⋮	⋮



K = 3

↑

3 nearest neighbour

Sort ascending

↓ Find

Closest

rule → majority count

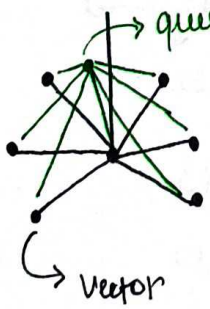
1 1 0 → (1)

closest point

majority class

* Work on N-Dimensional

$X_1 \mid X_2 \mid X_3 \cdots X_n$



3 vectors → majority count → class label of query point

How to Select K?

data

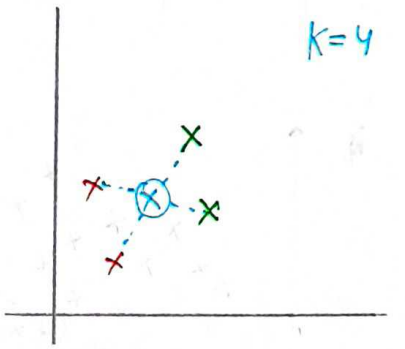
- heuristic (jugad)
- experimental → cross-validation

$\sqrt{n}$ n → observation

let 400 → $\sqrt{400} = 20$ → 19, 21

↑ data

* avoid even value for k



2(1) 2(0)

conflict

Best → experiment

↓

cross-validation

1000 = n

- 800 (train)
- 200 (test)

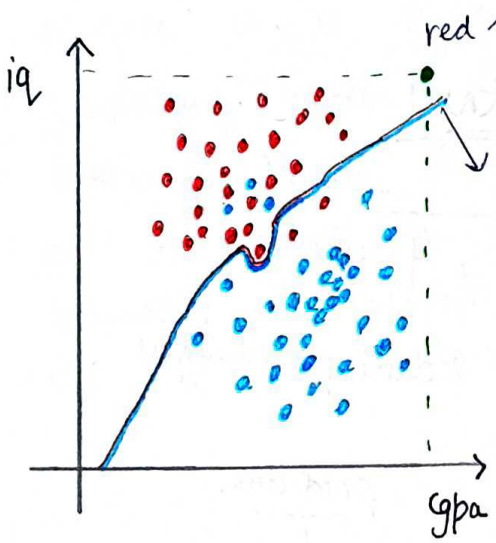
accuracy

diff KNN →

- KNN = 2 → 81
- KNN = 1 → 83
- KNN = 3 → 87
- KNN = 25 → 91 → best accuracy

Decision Surface

↳ tool → Classification → KNN
→ SVM
→ LR
→ DT
→ NN

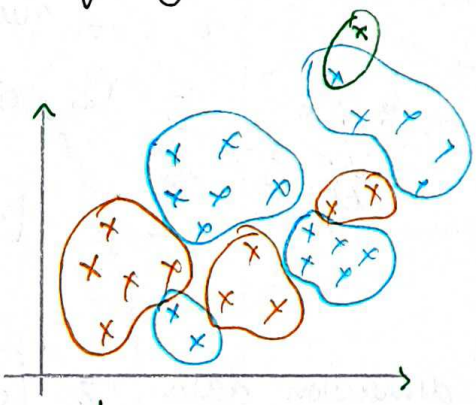


cgpa | iq | binary

Overfitting and Underfitting in KNN

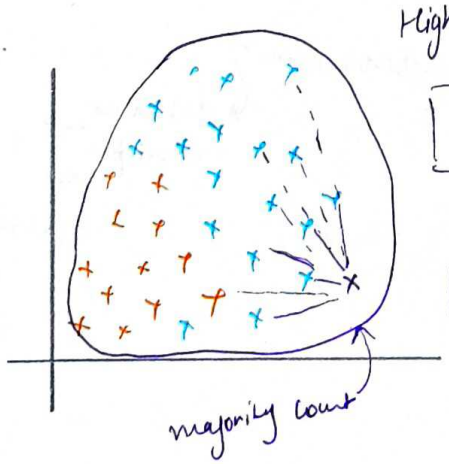
cgpa | iq | placed

200 student



K=1

↳ minor changes → overfitting ↓ High variance



Highest value ↓

K=?

n=200

K=200

blue > orange

simplistic ↓ underfitting

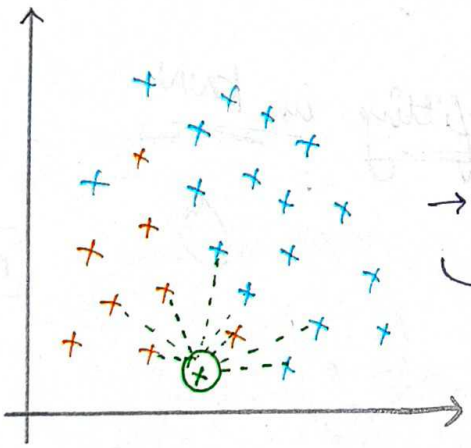
K = low → Overfitting

K = high → Underfitting

Limitation of KNN

1) large datasets ⇒ $n = 5L$, $f = 100$ → $(500000, 100)$

↳ KNN is a lazy learning technique



prediction → slow → dataset

→ query point → prediction

↳ training → nothing

5L distance → sort → majority

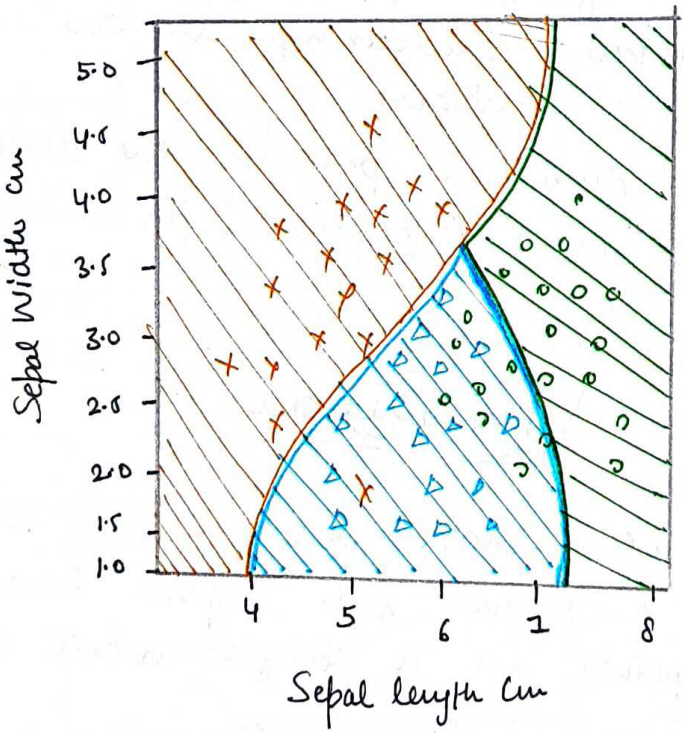
2. High dimension data → $f = 500$

↓

curse of dimension → distance concept → reliable

What is Decision Boundary?

In a classification problem with two or more classes, a decision boundary or decision surface is a hypersurface that partitions the underlying vector space into two or more sets, one for each class. The classifier will classify all the points on one side of the decision boundary as belonging to one class and all those on the other side as belonging to the other class.



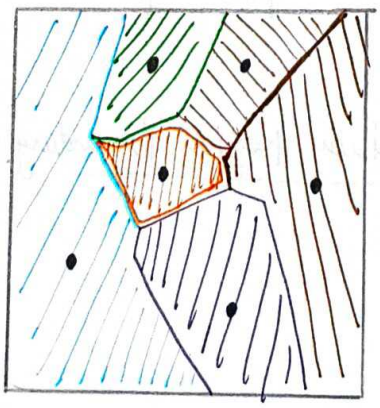
Decision Region Boundary: Sepal length and Sepal width

Important Points

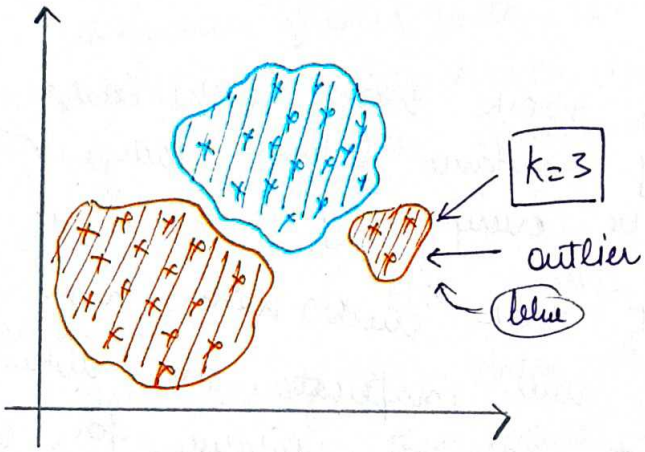
1. We can draw decision boundary for all the classification algorithms including Neural Networks
2. Decision boundary can be both linear (as in the case of SVM) or non-linear (as in the case of Decision tree classifier or knn)
3. Decision boundaries are not always clear cut. That is, the transition from one class in the feature space to another is not discontinuous, but gradual. This effect is common in fuzzy logic based classification algorithms, where membership in one class or another is ambiguous.
4. For higher dimension problems the decision boundry acts as a hyperplane (for linear ones)

Voronoi Diagram

In mathematics, a Voronoi diagram is a partitioning of a plane into regions based on distance to points in a specific subset of the plane.



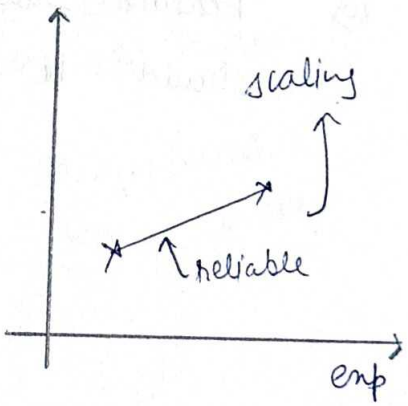
3) Outliers



4) Non- homogeneous scales

experience	salary	fine
0-25	20k-1Cr	

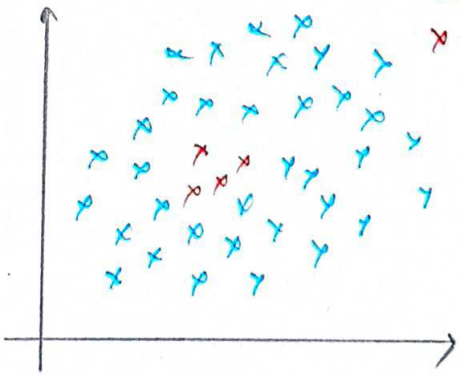
dominant because of scale



5) Imbalance dataset

- Yes → 98%
- No → 2%

→ biased



6) Inference (fail KNN or not good KNN for Inference)

↳ black box model

7) Does not work well with large dataset as calculating distance between each data instance would be every costly.

8) Does not work well with high dimensionality as this will complicate the distance calculating process to calculate distance for each dimension.

9) Sensitive to noisy and missing data

10) Feature Scaling - Data in all the dimension should be scaled properly.

3. Steps to plot Decision Boundary for KNN

(Assume 2 input cols)

1. Train the classifier on the training set.
2. Create a uniform grid (with the help of Numpy Meshgrid) of points that densely cover the region of input space containing the training set.
3. Classify each point on the grid. Store the result in an array A, where Aij contains the predicted class for the point at row i, column j on the grid.
4. Plot the array of an image, where each pixel corresponds to a grid point and its color represents the predicted class, The decision boundary can be seen as contours where the image changes color.
5. Finally print out the training data with their respective color on the same contours.