K-Nearest Neighbour (KNN) Algorithm

Outline

1. Introduction
2. The KNN Algorithm
3. Measuring Model Accuracy
4. Implementation
5. Pros & Cons
6. When To Use KNN

Introduction

- In supervised learning, predictions are either
  - **Classification:** Predicting the output to be one of a predefined class/category
    - *Binary Classification*: For example, Email is a spam or not
    - *Multiclass Classification*: For example, Email belongs to Primary, Work, Social, Promotion, etc
  - **Regression:** Predicting the output to be a continuous/numerical variable
    - For example, predict price of a house given features such as property size, number of bedrooms etc

- KNN algorithm is used to predict both Classification & Regression

Introduction

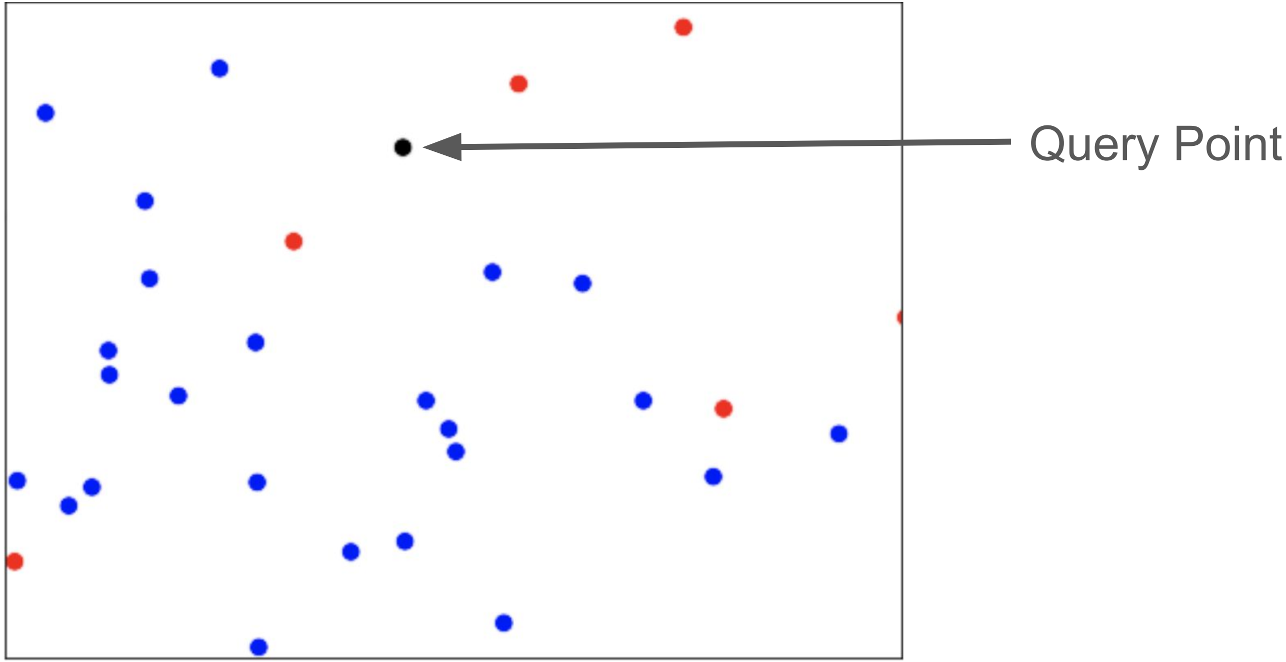
- An algorithm that predicts the label of a query point based on the majority observation of its **K neighbours (Similarity Measure)**

- It tries to answer:
    - **Classification**: What **class** does a query point belong to based on the **majority vote of its K nearest neighbors?**
    - **Regression**: What **value** should I assign to this query point, based on **the average of its K nearest neighbors' values?**

- **K** refers to how many neighbours to observe

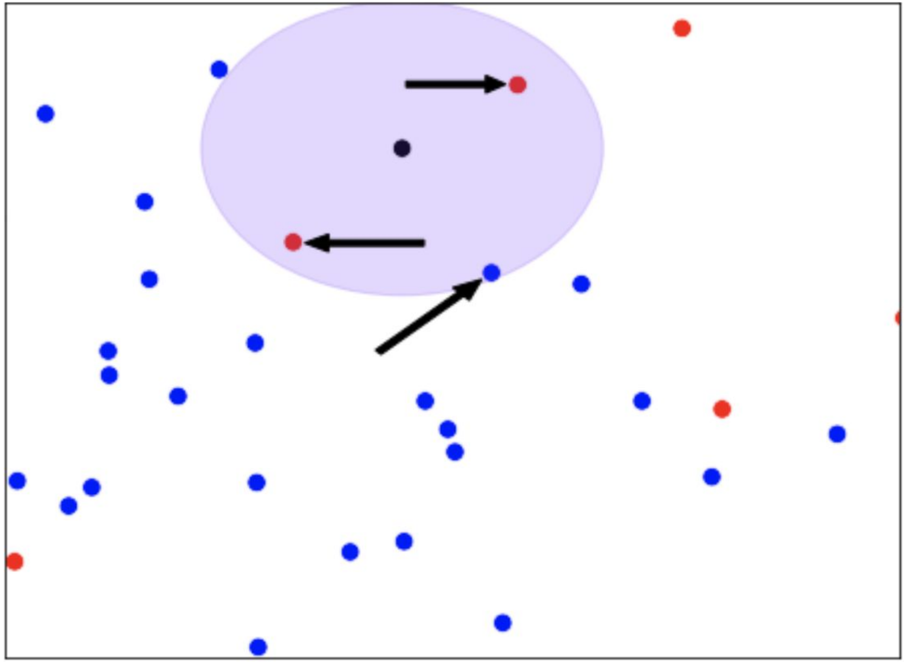
The K-Nearest Neighbour Algorithm

- From the scatterplot, classify the black observation to be red or black in color



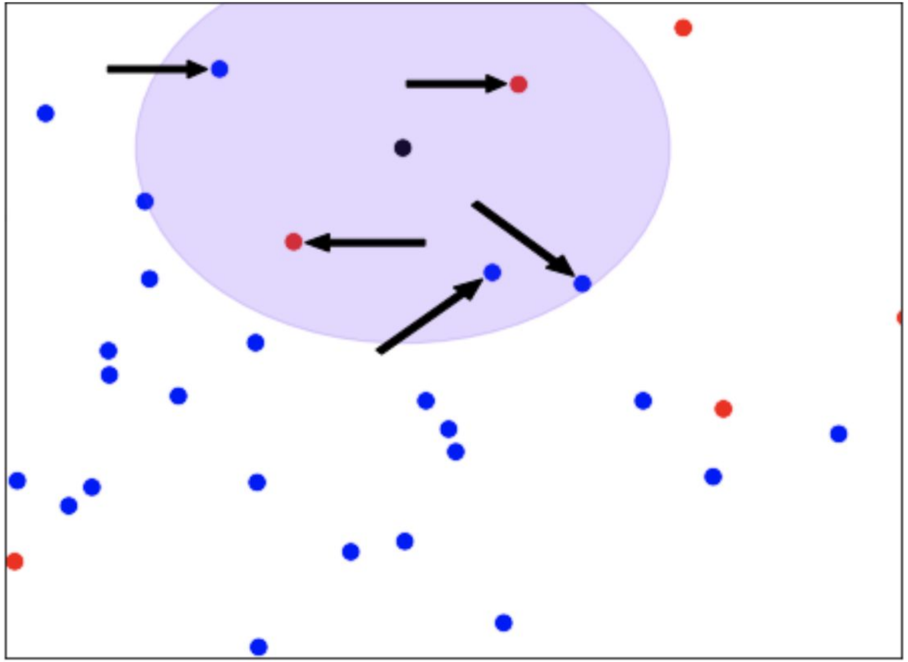
The K-Nearest Neighbour Algorithm

- **K = 3,** 2 of the 3 observations are red so the color is **classified as red**



The K-Nearest Neighbour Algorithm

- **K = 5,** 3 of the 5 observations are blue so the color is **classified as blue**



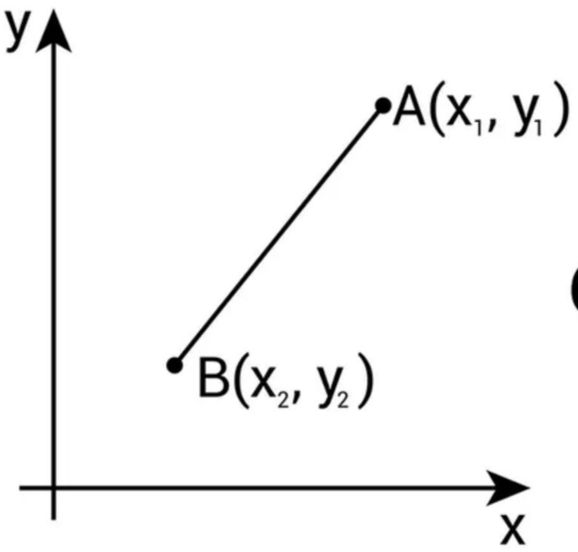
The K-Nearest Neighbour Algorithm

- **Non parametric Algorithm:** KNN does not learn any parameters during training and only has **2 hyperparameters**
    - K
    - Distance Metric (To find nearest points)

- It is referred as a **Lazy Algorithm**
    - It computes all distances and neighbours during prediction time

KNN Distance Metric

- To find the nearest neighbours, we can use **Euclidean distance** between the query point & other data points



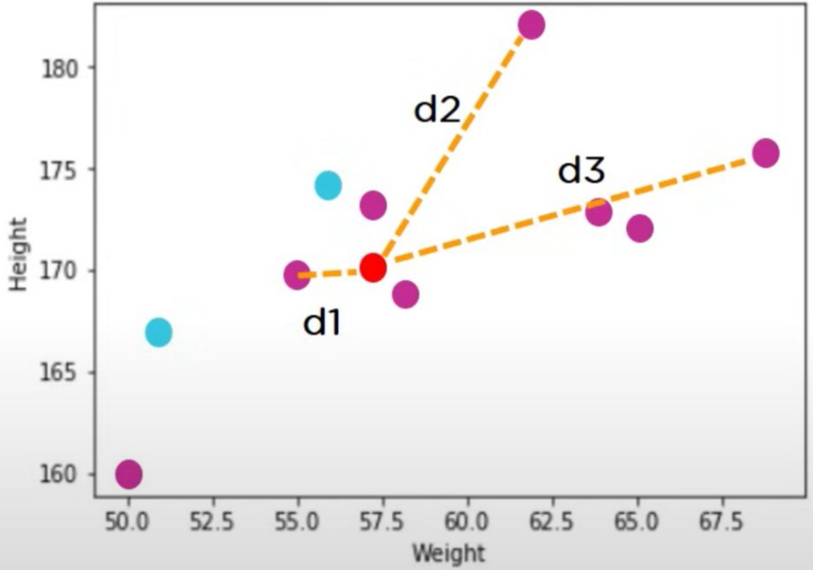
$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

KNN Distance Metric

Weight (KG)	Height (CM)	Class
55	170	Normal
57	173	Normal
58	169	Normal
56	174	Underweight
65	172	Normal
51	167	Underweight

57	170	??

KNN Distance Metric



Unknown data point

dist(**d1**)= $\sqrt{(170-167)^2 + (57-51)^2}$ ~= 6.7

dist(**d2**)= $\sqrt{(170-182)^2 + (57-62)^2}$ ~= 13

dist(**d3**)= $\sqrt{(170-176)^2 + (57-69)^2}$ ~= 13.4

KNN Distance Metric at K=3

Weight (KG)	Height (CM)	Class	Euclidean Distance
58	169	Normal	1.4
55	170	Normal	2
57	173	Normal	3
56	174	Underweight	4.1
51	167	Underweight	6.7
65	172	Normal	8.2

57	170	Normal

Classification: Mode

Regression: Mean

How to pick the right value for K

- Sqrt(n), where n is the total number of data points
    - Total data points, *n**=100***
    - Number of classes is 4
    - K= $\sqrt{100}$ =10
- Odd value of k is recommended to avoid confusion, even value of K can create balance between the features
- Using techniques like **Cross Validation**

Picking the value of K - Overfitting

K is small

- Overly sensitive to noise & variation
- Behaves more like a look up table
- Model performs **well on training data** but **poor on unseen data**

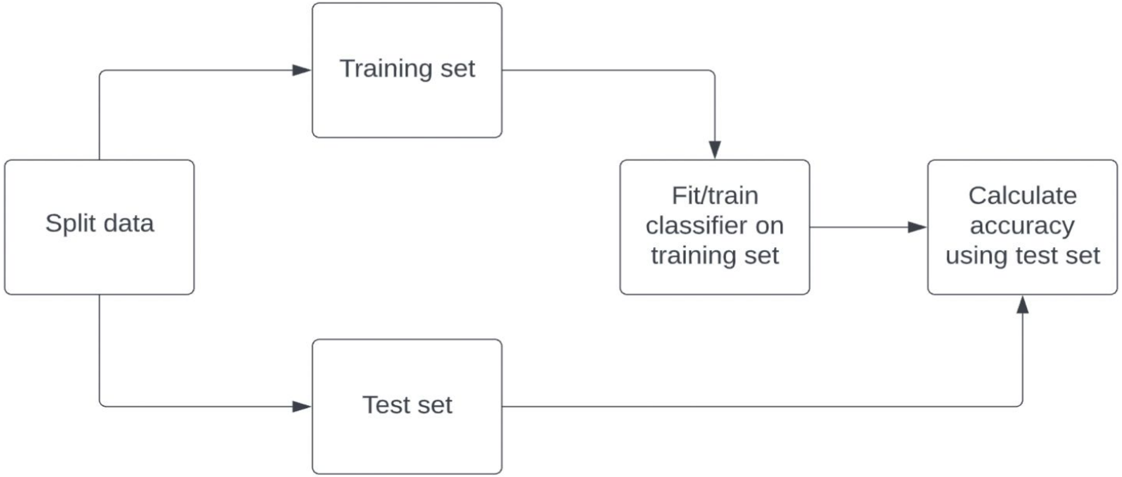
Picking the value of K - Underfitting

K is too large

- It just picks the most occurring label globally
- Model performs **poor on both training & unseen data**

Measuring Model Accuracy

- Accuracy = correct predictions / total observations



Implementation

Cross Validation

Split 1	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Metric 1
Split 2	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Metric 2
Split 3	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Metric 3
Split 4	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Metric 4
Split 5	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Metric 5

Training Data	Test Data

Pros

- Simple to implement
- Few Hyperparameters (only a K value & a distance metric / euclidean dist.)
- Adaptable (Adjusts to new data since all training data is stored in memory)
- No training time needed

Cons

- **Slow at prediction time**: Computes distance at every training point
- **Memory intensive**: Stores entire data set
- Not ideal for large data set as it requires more computational complexity which compromises model performance
- **Curse of dimensionality:** Does not perform well with high dimensional data inputs (causes distances between points to become more similar, making it difficult for KNN to find meaningful neighbours)

When to use KNN

- Data is labeled
- Data set is small to medium
- Data is noise free

Weight	Height	Class
45	100	Underweight
55	200	55
200	133	Overweight
77	150	Normal

Noise

When to use KNN

- Data is low dimensional (1D to 10D)
- The decision boundary is non-linear / irregular
- When nearby points in feature space truly **represent similar labels**

Real World Use Cases

- **Recommendation systems**
  - *Customers who bought this also bought*
- **Credit Scoring or Risk Assessment**
  - *Predict if someone will default on a loan by finding similar past customers*
- **Healthcare**
  - *Finding patients most similar with query point & see what their diagnosis is*
- **Geospatial Applications**
  - Fitted for location data where “closeness” matters
  - Used in Google maps to find nearest points of interest

Q&A

Thank You!