**K-Nearest Neighbor (KNN)**

# Assumptions:

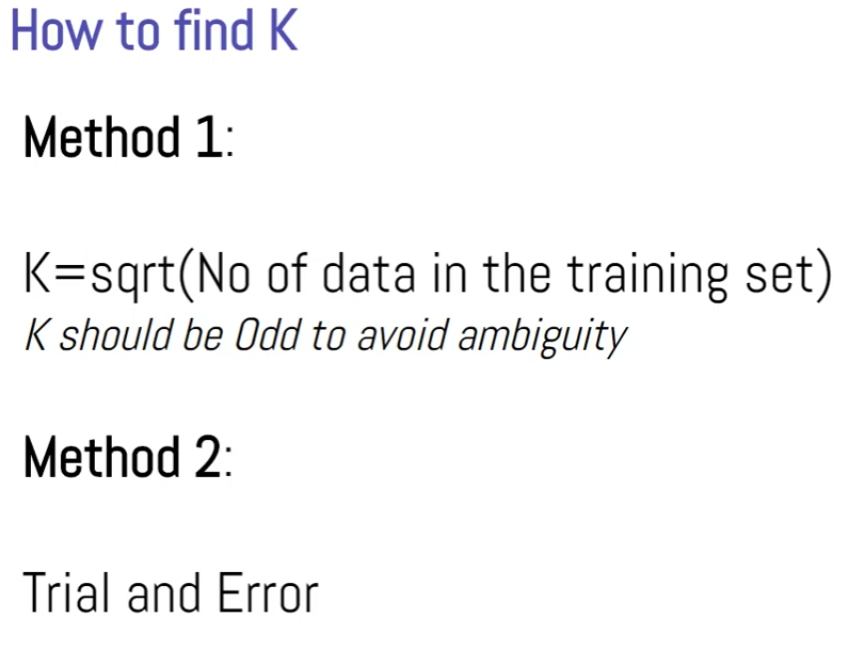
1. KNN assumes data is in metric space and there is a notion of distance
2. Each training data consists of labels associated with it wither +ve or -ve. It also supports multi-Class classification.
3. K decides how many neighbors influence the classification. This is usually Odd number.

# How KNN Works

1. **Classification**: To classify a new data point, KNN finds the k closest points (neighbours) in the training dataset and assigns the class that is most common among these neighbours.
2. **Regression**: For regression, KNN takes the average (or weighted average) of the values of the k nearest neighbours to predict the value of the new data point.

# Choosing the Value of K

1. The value of k is crucial in KNN and can greatly affect the model's performance.
2. **Small k**: Can lead to high variance and overfitting, as the model becomes sensitive to noise in the data.
3. **Large k**: Can lead to high bias and underfitting, as the model becomes too smooth and may miss complex patterns.
4. Common practice: Start with an odd number to avoid ties in classification and use cross-validation to determine the optimal k.



# Distance Metrics

1. KNN relies on a distance metric to measure the similarity between data points.
2. **Euclidean Distance**: Most common for continuous variables; calculated as the straight-line distance between points.
3. **Manhattan Distance**: Used for grid-like data; calculated as the sum of the absolute differences between coordinates.
4. **Minkowski Distance**: Generalization of Euclidean and Manhattan distances; includes a parameter p where p=2 is Euclidean and p=1 is Manhattan.
5. **Hamming Distance**: Used for categorical variables; counts the number of positions at which the corresponding elements are different.

# Weaknesses of KNN

1. **Computationally Intensive**: KNN can be slow for large datasets because it requires calculating the distance to all training points for each prediction.
2. **Memory Intensive**: Requires storing the entire training dataset, leading to high memory usage.
3. **Sensitive to Noise**: KNN can be sensitive to noisy data, outliers, and irrelevant features, especially with low k values.
4. **Curse of Dimensionality**: In high-dimensional spaces, the distance between points becomes less meaningful, and KNN performance can degrade.

# Improving KNN

1. **Feature Scaling**: Normalize or standardize features to ensure that all features contribute equally to the distance calculation.
2. **Dimensionality Reduction**: Techniques like PCA (Principal Component Analysis) can reduce the number of features, helping to mitigate the curse of dimensionality.
3. **Weighted KNN**: Assign weights to the neighbors, so closer neighbors have a greater influence on the prediction.
4. **KD-Tree or Ball Tree**: Data structures that can be used to optimize the search for nearest neighbors, making KNN faster for large datasets.

# Applications of KNN

1. **Classification**: Commonly used in image recognition, recommendation systems, and medical diagnosis.
2. **Regression**: Applied in predicting continuous outcomes, such as prices, temperatures, and scores.
3. **Imputation**: Used to fill in missing values by averaging the values of the nearest neighbours.