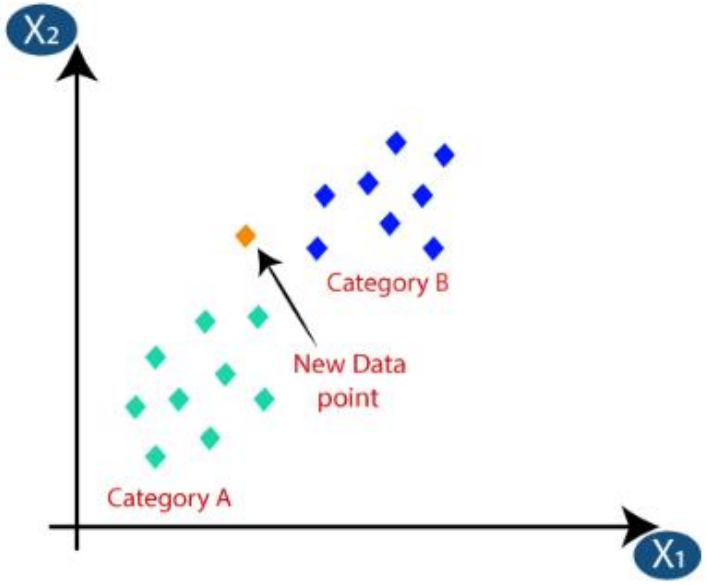
k-Nearest Neighbors

Marcel van Velzen
Junior Marte Garcia

k-Nearest Neighbors



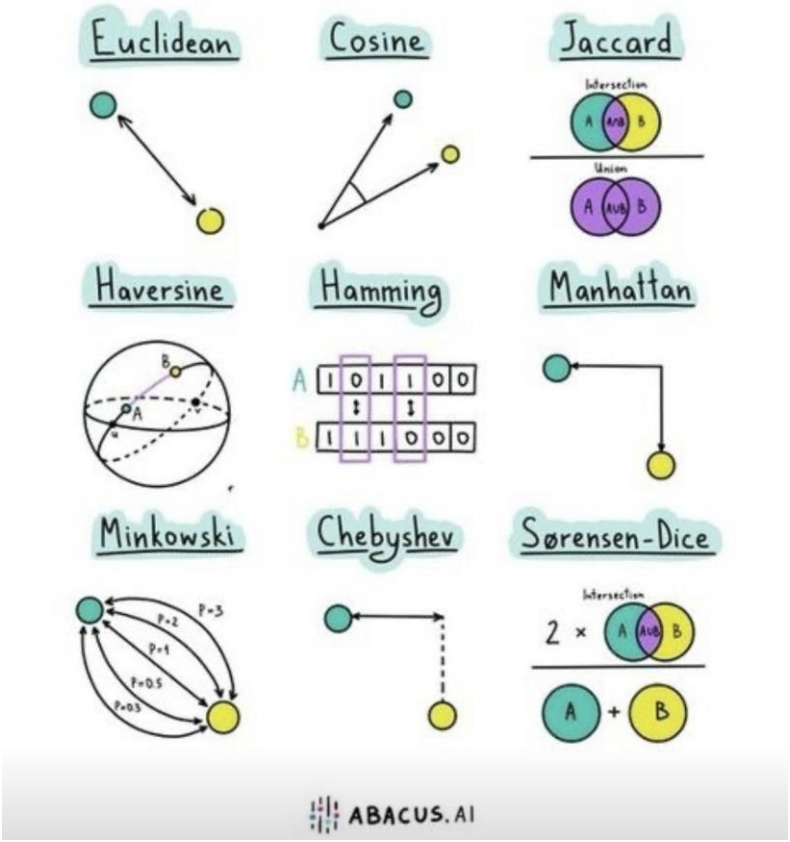
The **k-nearest neighbors** algorithm is based on the principle of similarity, where it classifies new data points by comparing them to the labeled data points in the training set and is used for classification and regression tasks.

For classification tasks, the algorithm assigns the class label of the majority of the k nearest neighbors to the new data point. In regression tasks, it predicts the value of the new data point based on the average or weighted average of the values of its k nearest neighbors.

Benefits	Drawbacks
Simple and effective	Does not produce a model
Makes no assumptions about the underlying data distribution	Slow classification phase
Fast training phase	Requires a large amount of memory
	Nominal features and missing data require additional processing

Calculating distance

In order to determine which data points are closest to a given query point, the distance between the query point and the other data points will need to be calculated.



Choosing an appropriate k

Deciding how many neighbors to use for the kNN determines how well the model generalize future data. The balance between overfitting and underfitting the training data is a problem known as the bias-variance tradeoff.

Smaller values of *k* in kNN have low bias and high variance and larger values of *k* in kNN have high bias and low variance.

A common practise is to start with *k* equal to the square root of the number of training examples.

To determine the optimal value of *k* in k-nearest neighbors (kNN) for both regression and classification problems, you can use the following techniques:

Cross-validation: Evaluate the performance of kNN using techniques like k-fold cross-validation, stratified k-fold cross-validation, or leave-one-out cross-validation, and select the value of k that yields the best performance metric.

Grid search: Define a range of possible values for k, evaluate the performance of kNN for each value using an appropriate performance metric, and select the value of k that gives the highest performance metric.

Domain knowledge and experimentation: Leverage insights or prior knowledge about the data, experiment with different values of k, and observe the model's performance to determine the optimal value of k based on the specific problem and dataset.

Preparing data

Feature scaling: When features have different scales, as it avoids dominance of a single feature in distance calculation, mitigating bias and improving overall performance.

The traditional method of rescaling features for kNN is **min-max normalization**. Another common transformation is called **z-score normalization**.

Feature engineering: Transforming features to reveal patterns, enhance class separability, or strengthen the relationship with the target variable using techniques like feature extraction, dimensionality reduction, or creating new derived features based on domain knowledge.

Handling categorical features: They should be encoded into numerical representations through techniques like one-hot encoding or label encoding before utilizing the algorithm.

Common approaches for categorical variables in KNN include label encoding for ordinal variables, one-hot encoding for non-ordinal variables, and feature hashing for high-cardinality variables.

Example

The table represents our data set and has two columns — **Brightness** and **Saturation**. Each row in the table has a class of either **Red** or **Blue** and the **Euclidean distance** is used as the distance measure. We have a new entry but it doesn't have a class yet. Suppose *k*=3.

BRIGHTNESS	SATURATION	CLASS
40	20	Red
50	50	Blue
60	90	Blue
10	25	Red
70	70	Blue
60	10	Red
25	80	Blue

BRIGHTNESS	SATURATION	CLASS
20	35	?

Calculated distances in ascending order

BRIGHTNESS	SATURATION	CLASS	FORMULA	DISTANCE
10	25	Red	SQRT((20-10)^2 + (35-25)^2)	14,1
40	20	Red	SQRT((20-40)^2 + (35-20)^2)	25,0
50	50	Blue	SQRT((20-50)^2 + (35-50)^2)	33,5
25	80	Blue	SQRT((20-25)^2 + (35-80)^2)	45,3
60	10	Red	SQRT((20-60)^2 + (35-10)^2)	47,2
70	70	Blue	SQRT((20-70)^2 + (35-70)^2)	61,0
60	90	Blue	SQRT((20-60)^2 + (35-90)^2)	68,0

***k*=3**

The majority class within the 3 nearest neighbors to the new entry is **Red**. Therefore, we'll classify the new entry as **Red**.

Business applications

Customer Segmentation: kNN can group customers based on similarities in purchasing behavior or demographics, allowing businesses to tailor marketing strategies and offerings to specific customer segments.

Recommender Systems: kNN can power personalized recommendations by finding similarities between users and suggesting items that similar users have shown interest in, improving customer engagement and sales.

Fraud Detection: kNN can identify suspicious transactions or activities by comparing them to known fraudulent patterns or similar historical instances, helping businesses prevent financial losses due to fraudulent behavior.

Credit Scoring: kNN can assess creditworthiness by comparing the financial profiles of loan applicants to those of existing customers, assisting in the decision-making process for loan approvals and managing credit risk.

Image Recognition: kNN can classify images into categories or identify similar images by comparing their pixel values or extracted features, enabling applications in e-commerce, healthcare, security, and more.

Anomaly Detection: kNN can identify anomalies in data, such as detecting faulty equipment or network intrusions, aiding in proactive maintenance and ensuring system security.

Predictive Maintenance: kNN can predict equipment failures or maintenance needs by analyzing historical patterns and similarities with current operating conditions, reducing downtime and optimizing maintenance schedules.

References

Abba, I. V. (2023). KNN Algorithm – k-Nearest Neighbors Classifiers and Model Example. *freeCodeCamp.org*. https://www.freecodecamp.org/news/k-nearest-neighbors-algorithm-classifiers-and-model-example/

K-Nearest Neighbor(KNN) Algorithm for Machine Learning - Javatpoint. (n.d.-b). www.javatpoint.com. https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning

Lantz, B. (2013). *Machine learning with R: Learn how to use R to apply powerful machine learning methods and gain an insight into real-world applications*. Packt Publishing.