K-Nearest Neighbors (KNN)

1. What is KNN?

K-Nearest Neighbors (KNN) is a **non-parametric**, **lazy learning** algorithm used for:

- Classification
- Regression

It predicts based on the ***k* closest data points** in the training set.

2. Core Idea

1. Store all training data.
2. For a new point:
   (a) Compute distance to all training points.
   (b) Pick the k nearest ones.
   (c) **Classification:** majority vote of the neighbors.
   (d) **Regression:** average (or weighted average) of neighbor values.

13. Math Example (Marriage Dataset)

Problem: Predict whether a new person `P` has a wife based on their age and salary using k-Nearest Neighbors ($k = 3$).

Training Data (Features: Age, Salary, Has_Wife)

ID	**Age**	**Salary (K)**	**Has_Wife (Label)**
A	25	50	0
B	30	60	1
C	28	58	1
D	22	45	0
E	35	65	1

New Person (Test Point)

`P`: Age = 27, Salary = 55K

Step 1: Convert Salary to Numerical Scale (in thousands)

- All salaries: divide by 1000 $\rightarrow$ use 50 instead of 50K, etc.

Step 2: Compute Euclidean Distances

$$d(P,A) = \sqrt{(27-25)^2 + (55-50)^2} = \sqrt{4+25} = \sqrt{29} \approx 5.39$$
$$d(P,B) = \sqrt{(27-30)^2 + (55-60)^2} = \sqrt{9+25} = \sqrt{34} \approx 5.83$$
$$d(P,C) = \sqrt{(27-28)^2 + (55-58)^2} = \sqrt{1+9} = \sqrt{10} \approx 3.16$$
$$d(P,D) = \sqrt{(27-22)^2 + (55-45)^2} = \sqrt{25+100} = \sqrt{125} \approx 11.18$$
$$d(P,E) = \sqrt{(27-35)^2 + (55-65)^2} = \sqrt{64+100} = \sqrt{164} \approx 12.81$$

Step 3: Pick $k = 3$ Nearest Neighbors

Top 3 nearest:

- C (3.16) – Yes
- A (5.39) – No
- B (5.83) – Yes

Step 4: Majority Vote

Labels: Yes, No, Yes $\Rightarrow$ Majority = **Yes**

Final Prediction:

$$\boxed{\text{Person P is predicted to have a wife (Yes)}}$$

Note: If you used $k = 5$, you would include all points, and result might change.

3. Key Parameters

- **k** – Number of neighbors to consider.
- **Distance metric** – Common: Euclidean, Manhattan, Minkowski.
- **Weights** – All neighbors equal or closer ones weighted more.
- **Algorithm** – `'auto'`, `'kd_tree'`, `'ball_tree'`, `'brute'`.

4. Distance Metrics

- **Euclidean Distance**:

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

- **Manhattan Distance**:

$$d(p,q) = \sum_{i=1}^{n} |p_i - q_i|$$

- **Minkowski Distance**:

$$d(p,q) = \left(\sum_{i=1}^{n} |p_i - q_i|^p \right)^{1/p}$$

5. Pros and Cons

Advantages

- Simple and intuitive.
- No explicit training phase.
- Works well for multi-class problems.
- Good with well-separated data.

Disadvantages

- Prediction can be slow for large datasets.
- Sensitive to irrelevant features and unscaled data.
- Poor performance in high dimensions (curse of dimensionality).
- Choice of k is crucial.

6. Choosing k

- **Small** k: Sensitive to noise (high variance).
- **Large** k: Smoother, more stable (but higher bias).
- **Best practice**: Use **cross-validation** to choose k.

7. Preprocessing Tips

- Normalize or standardize data.
- Handle missing values.
- Consider dimensionality reduction (e.g., PCA).

8. Python Implementation (Scikit-learn)

Classification Example:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

# Preprocessing
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2)

# Model
model = KNeighborsClassifier(n_neighbors=5)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

Regression Example:

```
from sklearn.neighbors import KNeighborsRegressor
model = KNeighborsRegressor(n_neighbors=3)
```

9. Evaluation Metrics

For Classification:

- Accuracy
- Precision, Recall, F1-score
- Confusion Matrix

For Regression:

- Mean Squared Error (MSE)
- Mean Absolute Error (MAE)
- R^2 Score

10. Variants and Improvements

- **Weighted KNN** – Weights inversely proportional to distance.
- **K-d Trees / Ball Trees** – Fast lookup for neighbors.
- **Approximate Nearest Neighbors** – e.g., FAISS, Annoy.
- **Outlier-resistant methods**.

11. Applications of KNN

- Recommender systems
- Image classification
- Text categorization
- Anomaly detection
- Imputation of missing data

12. Summary Table

Term	Explanation
Lazy Learner	No explicit training, just memory-based
Instance-based	Uses all data to make prediction
Non-parametric	No assumption on data distribution
Curse of Dimensionality	Distance loses meaning in high dimensions