K-Nearest Neighbors (KNN) Algorithm

K-Nearest Neighbors (KNN) is a **supervised machine learning algorithm** used for **classification** and **regression** tasks. It is a **non-parametric** and **instance-based learning** algorithm, meaning it makes predictions without explicitly learning a model; instead, it uses the stored training data to predict outputs.

When to Use KNN?

1. **Classification Tasks:**
   a. When the decision boundaries are not linear.
   b. When the dataset is not large (as KNN can be computationally expensive for large datasets).
2. **Regression Tasks:**
   a. When we need to predict continuous values based on similarity (e.g., salary prediction, house price prediction).
3. **Real-Time Applications:**
   a. **Recommendation Systems**: To find similar items or users.
   b. **Fraud Detection**: To classify transactions as fraudulent or genuine.
   c. **Image Recognition**: For recognizing objects based on pixel similarity.

Real-Time Example: Salary Estimation

Dataset Description:

We will use a hypothetical dataset containing the following features:

- `YearsExperience` (Numeric): The number of years a person has worked.
- `Age` (Numeric): The age of the person.
- `EducationLevel` (Categorical): The highest degree attained (e.g., Bachelor's, Master's).
- `Salary` (Numeric - Target): The salary of the individual.

How KNN Works

1. **Initialization:**
    a. Choose the number of neighbors (k).
2. **Compute Distance:**
    a. For a given data point, calculate the distance (e.g., Euclidean, Manhattan) to all other data points in the dataset.
3. **Sort Neighbors:**
    a. Sort the data points by their distance to the given point.
4. **Determine the Output:**
    a. For **classification**: Assign the label most common among the k nearest neighbors.
    b. For **regression**: Compute the average (or weighted average) of the k nearest neighbors' values.
5. **Decision Making:**
    a. Return the result for the given data point.

Real-Time Example: Salary Estimation

Let's consider an example dataset of employees' experience and salaries:

Dataset

Experience (Years)	Age	Education Level (1: High School, 2: Bachelor, 3: Master)	Salary ($)
1	22	2	35,000
2	25	2	42,000
3	28	2	50,000
4	30	3	65,000
5	35	3	75,000
6	40	3	90,000

Experience (Years)	Age	Education Level (1: High School, 2: Bachelor, 3: Master)	Salary Class
1	22	2	≤50K
2	25	2	≤50K
3	28	2	≤50K
4	30	3	>50K
5	35	3	>50K
6	40	3	>50K

Goal: Predict Salary

We want to predict the salary of a new employee with the following characteristics:

- **Experience:** 3.5 years
- **Age:** 29
- **Education Level:** 3 (Master's)

Steps to Apply KNN

1. **Choose k:**
    a. Suppose k = 3.
2. **Calculate Distance (Euclidean Distance):**

$$d = \sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2 + (Z_1 - Z_2)^2}$$

For each data point, compute the distance to the new employee (3.5 years, 29, level 3).

3. **Find Nearest Neighbors:** Sort the distances and find the k nearest neighbors.
4. **Predict Salary:**
    a. For regression: Take the average salary of the k nearest neighbors.
    b. For classification (if applicable): Use the majority class.

Problem (KNN classification):

For a **classification problem** predicting whether a person's salary is **≤50K** or **>50K**, we adapt the K-Nearest Neighbors (KNN) algorithm to perform **binary classification** based on a given dataset.

Dataset

Here is an example dataset with features like **Experience (Years)**, **Age**, and **Education Level**, and the target variable **Salary Class** (≤50K or >50K):

Experience (Years)	Age	Education Level (1: High School, 2: Bachelor, 3: Master)	Salary Class
1	22	2	≤50K
2	25	2	≤50K
3	28	2	≤50K
4	30	3	>50K
5	35	3	>50K
6	40	3	>50K

Goal

Predict whether the salary of a **new employee** with the following characteristics falls into **≤50K** or **>50K**:

- Experience: **3.5 years**
- Age: **29**
- Education Level: **3 (Master's)**

Steps to Solve Using KNN

Step 1: Choose k

We choose $k=3$, which means we will look at the **3 nearest neighbors**.

Step 2: Calculate Euclidean Distance

Use the **Euclidean distance** formula to calculate the distance between the new employee and each data point in the dataset.

$$d = \sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2 + (Z_1 - Z_2)^2}$$

Distances Calculation Table:

Experience (Years)	Age	Education Level	Distance to (3.5, 29, 3)
1	22	2	$\sqrt{(3.5-1)^2 + (29-22)^2 + (3-2)^2} = \sqrt{2.5^2 + 7^2 + 1^2} = \sqrt{6.25 + 49 + 1} = \sqrt{56.25} = 7.5$
2	25	2	$\sqrt{(3.5-2)^2 + (29-25)^2 + (3-2)^2} = \sqrt{1.5^2 + 4^2 + 1^2} = \sqrt{2.25 + 16 + 1} = \sqrt{19.25} \approx 4.39$
3	28	2	$\sqrt{(3.5-3)^2 + (29-28)^2 + (3-2)^2} = \sqrt{0.5^2 + 1^2 + 1^2} = \sqrt{0.25 + 1 + 1} = \sqrt{2.25} = 1.5$
4	30	3	$\sqrt{(3.5-4)^2 + (29-30)^2 + (3-3)^2} = \sqrt{(-0.5)^2 + (-1)^2 + 0^2} = \sqrt{0.25 + 1} = \sqrt{1.25} \approx 1.12$
5	35	3	$\sqrt{(3.5-5)^2 + (29-35)^2 + (3-3)^2} = \sqrt{(-1.5)^2 + (-6)^2 + 0^2} = \sqrt{2.25 + 36} = \sqrt{38.25} \approx 6.18$
6	40	3	$\sqrt{(3.5-6)^2 + (29-40)^2 + (3-3)^2} = \sqrt{(-2.\text{[illegible]} + (-11)^2 + 0^2} = \sqrt{6.25 + 121} = \sqrt{127.25} \approx 11.29$

Distances Calculation Table:

Experience (Years)	Age	Education Level	Distance to (3.5, 29, 3)	Salary Class
1	22	2	7.5	≤50K
2	25	2	4.39	≤50K
3	28	2	1.5	≤50K
4	30	3	1.12	>50K
5	35	3	6.18	>50K
6	40	3	11.29	>50K

Step 3: Find Nearest Neighbors

Sort the distances and select the **k = 3 nearest neighbors**:

Experience (Years)	Age	Education Level	Distance	Salary Class
4	30	3	1.12	>50K
3	28	2	1.5	≤50K
2	25	2	4.39	≤50K

Step 4: Predict Salary Class

The **majority class** among the 3 nearest neighbors determines the prediction:

- Neighbor 1: >50K
- Neighbor 2: ≤50K
- Neighbor 3: ≤50K

Majority Class: ≤50K

Final Prediction

The new employee's salary is predicted to be in the class **≤50K**.

Problem – KNN Regressor

Dataset:

Experience (Years)	Age	Education Level (1: High School, 2: Bachelor, 3: Master)	Salary ($)
1	22	2	35,000
2	25	2	42,000
3	28	2	50,000
4	30	3	65,000
5	35	3	75,000
6	40	3	90,000

New Employee Characteristics:

- Experience: **3.5 years**
- Age: **29**
- Education Level: **3 (Master's)**

Steps to Apply KNN for Salary Prediction:

Step 1: Choose k

We set $k=3$, meaning the algorithm will consider the **3 nearest neighbors** to predict the salary.

Step 2: Calculate Distances

We use the **Euclidean distance formula** to calculate the distance between the new employee and each point in the dataset:

$$d = \sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2 + (Z_1 - Z_2)^2}$$

Where:

- X: Experience
- Y: Age
- Z: Education Level

Distances Calculation Table:

Experience (Years)	Age	Education Level	Distance to (3.5, 29, 3)
1	22	2	$\sqrt{(3.5-1)^2+(29-22)^2+(3-2)^2} = \sqrt{2.5^2+7^2+1^2} = \sqrt{6.25+49+1} = \sqrt{56.25} = 7.5$
2	25	2	$\sqrt{(3.5-2)^2+(29-25)^2+(3-2)^2} = \sqrt{1.5^2+4^2+1^2} = \sqrt{2.25+16+1} = \sqrt{19.25} \approx 4.39$
3	28	2	$\sqrt{(3.5-3)^2+(29-28)^2+(3-2)^2} = \sqrt{0.5^2+1^2+1^2} = \sqrt{0.25+1+1} = \sqrt{2.25} = 1.5$
4	30	3	$\sqrt{(3.5-4)^2+(29-30)^2+(3-3)^2} = \sqrt{(-0.5)^2+(-1)^2+0^2} = \sqrt{0.25+1} = \sqrt{1.25} \approx 1.12$
5	35	3	$\sqrt{(3.5-5)^2+(29-35)^2+(3-3)^2} = \sqrt{(-1.5)^2+(-6)^2+0^2} = \sqrt{2.25+36} = \sqrt{38.25} \approx 6.18$
6	40	3	$\sqrt{(3.5-6)^2+(29-40)^2+(3-3)^2} = \sqrt{(-2.5)^2+(-11)^2+0^2} = \sqrt{6.25+121} = \sqrt{127.25} \approx 11.29$

Step 3: Find Nearest Neighbors

Sort the distances to identify the **3 nearest neighbors**:

Experience (Years)	Age	Education Level	Distance	Salary ($)
4	30	3	1.12	65,000
3	28	2	1.5	50,000
2	25	2	4.39	42,000

Step 4: Predict Salary

For **regression**, we calculate the **average salary** of the 3 nearest neighbors:

$$\text{Predicted Salary} = \frac{65,000+50,000+42,000}{3} = \frac{157,000}{3} = 52,333.33$$

Final Prediction: The salary for the new employee is approximately $52,333.

Summary

- **New Employee Features**: Experience = 3.5, Age = 29, Education Level = 3.
- **Nearest Neighbors**: Rows 4, 3, and 2 from the dataset.
- **Predicted Salary**: **$52,333**.

```
import numpy as np

import pandas as pd

from sklearn.neighbors import KNeighborsClassifier


# Dataset

data = {

    "Experience": [1, 2, 3, 4, 5, 6],

    "Age": [22, 25, 28, 30, 35, 40],

    "EducationLevel": [2, 2, 2, 3, 3, 3],

    "SalaryClass": ["≤50K", "≤50K", "≤50K", ">50K", ">50K", ">50K"]
```

```
}
df = pd.DataFrame(data)

# Features and target
X = df[["Experience", "Age", "EducationLevel"]]
y = df["SalaryClass"]

# New employee details
new_employee = np.array([[3.5, 29, 3]])

# KNN Classification
knn = KNeighborsClassifier(n_neighbors=3, metric='euclidean')
knn.fit(X, y)
predicted_class = knn.predict(new_employee)

print(f"Predicted Salary Class: {predicted_class[0]}")
```

Working Explanation:

- **Training Phase**: The KNN algorithm stores the training data and precomputes distances between all points in the training set.
- **Prediction Phase**: For a given test point, KNN identifies the K nearest neighbors based on distance and calculates the predicted salary as the average of these neighbors' salaries.

```
import numpy as np
import pandas as pd
```

```python
from sklearn.neighbors import KNeighborsRegressor

# Dataset
data = {
    "Experience": [1, 2, 3, 4, 5, 6],
    "Age": [22, 25, 28, 30, 35, 40],
    "EducationLevel": [2, 2, 2, 3, 3, 3],
    "Salary": [35000, 42000, 50000, 65000, 75000, 90000]
}
df = pd.DataFrame(data)

# Features and target
X = df[["Experience", "Age", "EducationLevel"]]
y = df["Salary"]

# New employee details
new_employee = np.array([[3.5, 29, 3]])

# KNN Regression
knn = KNeighborsRegressor(n_neighbors=3, metric='euclidean')
knn.fit(X, y)
predicted_salary = knn.predict(new_employee)

print(f"Predicted Salary: ${predicted_salary[0]:,.2f}")
```

The **`fit_transform`** method is used on a **training dataset** to calculate the necessary parameters (like mean, standard deviation, min, max, etc.) and apply the transformation. Meanwhile, **`transform`** is used on a **test dataset** (or any other dataset) to apply the same transformation using the parameters learned from the training data. This ensures consistency between how training and testing data are preprocessed.

Why Use `fit_transform` for Training and `transform` for Testing?

1. **Avoid Data Leakage**:
   a. Parameters like mean, standard deviation, or scaling factors should only be computed from the training data. Using the test data during the fitting process can lead to **data leakage**, giving the model access to information about the test set, which it shouldn't have.
2. **Consistency**:
   a. Once the parameters (e.g., scaling factors) are learned from the training data, they are reused to transform both training and testing data, ensuring that the preprocessing remains consistent.
3. **Real-World Simulation**:
   a. In real-world scenarios, you won't have access to future (test) data while training. Using `fit_transform` on test data would simulate an unrealistic scenario.

Example: Using `fit_transform` and `transform`

Let's preprocess a dataset where features need to be standardized (zero mean, unit variance) using **`StandardScaler`** from Scikit-learn.

Dataset

Feature	Target
10	0
20	1
30	0

40	1
50	1

- Training Data (80%): *[10,20,30]*
- Test Data (20%): *[40,50]*

Output

1. **Scaled Training Data:**

$$X_{\text{train\_scaled}} = \begin{bmatrix} -1.2247 \\ 0 \\ 1.2247 \end{bmatrix}$$

2. **Scaled Test Data:**

$$X_{\text{test\_scaled}} = \begin{bmatrix} 2.4495 \\ 3.6742 \end{bmatrix}$$

Explanation

1. **Training** (`fit_transform`):
   - The `StandardScaler` computes the mean and standard deviation from the training data:
     - **Mean**: $\text{mean} = 20$
     - **Std Dev**: $\text{std} = 10$
   - Each value in the training data is scaled as:

$$X_{\text{scaled}} = \frac{X - \text{mean}}{\text{std}}$$

2. **Testing** (`transform`):
   - The same mean (20) and standard deviation (10) are used to scale the test data:

$$X_{\text{test\_scaled}} = \frac{X - \text{mean (training)}}{\text{std (training)}}$$

Key Points

- **`fit_transform`**: Computes the parameters (mean, std, etc.) from training data and applies the transformation.

- **`transform`**: Applies the same transformation (parameters from training) to the test data.
- Using the test data in `fit_transform` leads to **data leakage** and compromises model evaluation.