✅ K-Nearest Neighbors (KNN) Classification — Detailed Explanation

KNN is a **supervised machine learning algorithm** used for **classification**.
It classifies a new data point by looking at the **K nearest neighbors** (closest data points).

🧠 How KNN Works (Simple Explanation)

1. Choose **K** (example: K = 3).
2. Calculate **distance** between the new sample and all training samples (Euclidean distance).
3. Pick the **K closest points**.
4. The **majority class** among those K neighbors becomes the **prediction**.

📌 Steps You Should Follow (Based on Your Image)

1. Choose dataset → Example: Iris Dataset
2. Normalize features

3 Use KNeighborsClassifier
4 Try different K values
5 Evaluate accuracy + confusion matrix
6 Visualize decision boundaries

✅ Complete Working Code (With Output Explanation)

📌 Python Code

```
import pandas as pd
from sklearn.datasets import load_iris

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

import matplotlib.pyplot as plt
import seaborn as sns

# 1. Load Dataset
iris = load_iris()
X = iris.data
y = iris.target

# 2. Normalize Features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 3. Train-Test Split
```

```
X_train, X_test, y_train,
y_test =
train_test_split(X_scaled, y,
test_size=0.2, random_state=42)


# 4. KNN Model

knn =
KNeighborsClassifier(n_neighbor
s=3)

knn.fit(X_train, y_train)


# 5. Predictions

y_pred = knn.predict(X_test)


# 6. Accuracy


accuracy =
accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy)


# 7. Confusion Matrix


cm = confusion_matrix(y_test,
y_pred)

print("Confusion Matrix:\n",
cm)


# 8. Plot Confusion Matrix

sns.heatmap(cm, annot=True,
cmap="Blues", fmt="d",



xticklabels=iris.target_names,



yticklabels=iris.target_names)

plt.title("KNN Confusion
Matrix")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()
```

🎯 OUTPUT DETAILS

1 Accuracy Output

Example output will look like:

```
Accuracy: 1.0
```

```
Meaning → Model correctly
predicted all test samples
(Iris dataset is easy).
```

2 Confusion Matrix Output

```
Confusion Matrix:
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

Interpretation

Row	Meaning
Row 1 → Iris-setosa	All 10 predicted correctly
Row 2 → Iris-versicolor	All 9 correct
Row 3 → Iris-virginica	All 11 correct

No misclassification → Perfect model

3 Confusion Matrix Graph Output

You will see a heatmap with:

3x3 grid

Dark blue squares on diagonal

Values like (10, 9, 11)

🎨 Decision Boundary (Only for 2 Features)

If you want, I can also give you **decision boundary code and graph**, but it needs selecting only 2 columns.

Tell me: