# **--> Non-Linear Algorithms**

# **2. K-Nearest Neighbors (KNN) – Detailed Notes**

## **USE → Iris dataset for model predict**

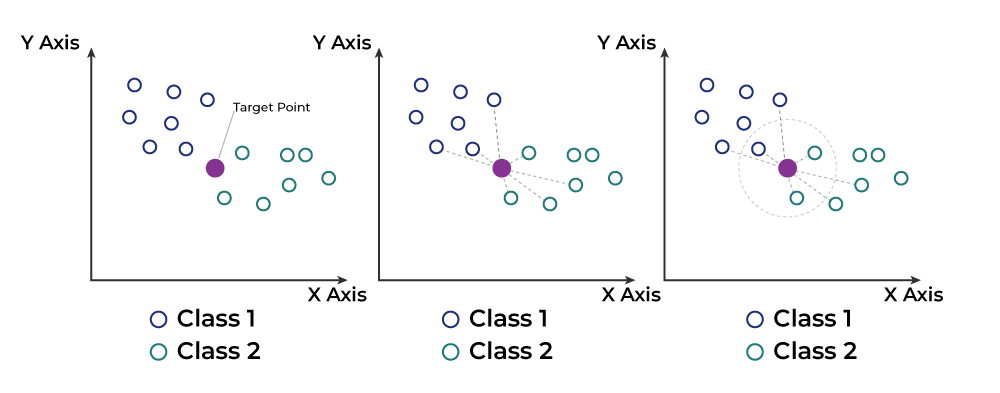
<https://scikit-learn.org/1.5/auto_examples/datasets/plot_iris_dataset.html>

<https://scikit-learn.org/stable/api/sklearn.datasets.html>

## 

## **1. What is K-NN?**

* **Definition:** K-Nearest Neighbors (KNN) is a **supervised learning algorithm** used for both **classification** and **regression** problems.  
   It works by **finding the K nearest data points** to a new data point and making predictions based on their majority vote (classification) or average value (regression).
* **Key Idea:** *"Similar data points are likely to have similar outcomes."*
* **Type:**
  + **Lazy learning**: No model is built during training; all computation happens during prediction.
  + **Instance-based learning**: Stores training data and uses it directly.

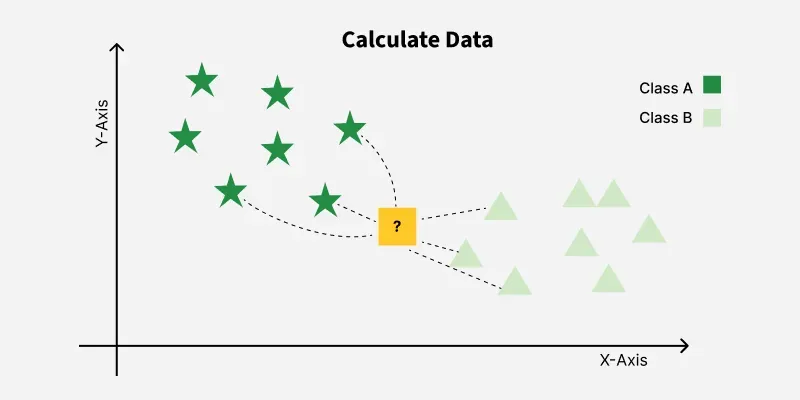


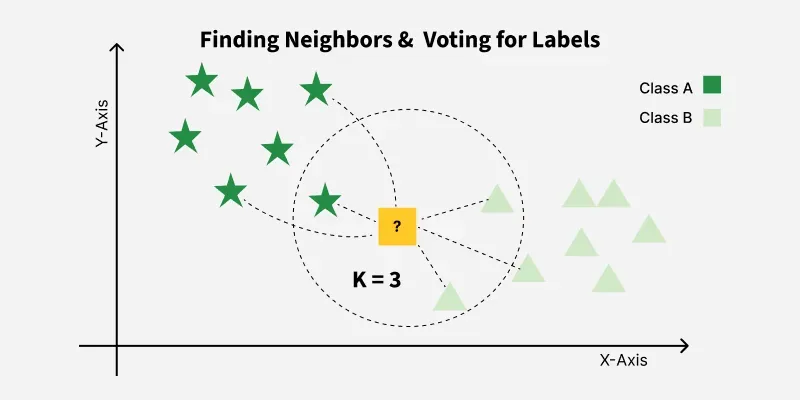
## 

## **2. KNN Algorithm**

**Steps:**

1. **Store** all the training data.
2. **Choose** the number of neighbors K.
3. **Calculate** the distance (e.g., Euclidean) between the new point and all training points.



1. **Sort** the distances and select the K nearest neighbors.
2. **Vote / Average**:  
   * **Classification:** Pick the class with the highest frequency among neighbors.
   * **Regression:** Take the average of neighbors' target values.
3. **Return** the predicted class/value.  
   

## **3. Working of KNN – Example**

**Dataset Example (Fruit classification):**

|  |  |  |
| --- | --- | --- |
| **Color** | **Size** | **Fruit Type** |
| Red | 5 | Apple |
| Green | 6 | Pear |
| Red | 6 | Apple |
| Yellow | 4 | Banana |

**If we have a new fruit:**

* Color = Red, Size = 5.5
* KNN will:  
  1. Calculate distances to all fruits in dataset.
  2. Pick nearest K neighbors.
  3. See which fruit type is most common among neighbors.
  4. Predict that type.

## **4. How to Choose the Value of K?**

* **Small K (e.g., K=1, K=3)**:  
  + Very sensitive to noise.
  + May overfit.
* **Large K (e.g., K=20)**:  
  + More stable predictions.
  + May underfit.
* **Best practice:**
  + Use **Cross-Validation** to choose K.
  + Often odd K is chosen for classification to avoid ties.
* **Rule of Thumb:** K≈Number of SamplesK \approx \sqrt{\text{Number of Samples}}

## **5. Advantages**

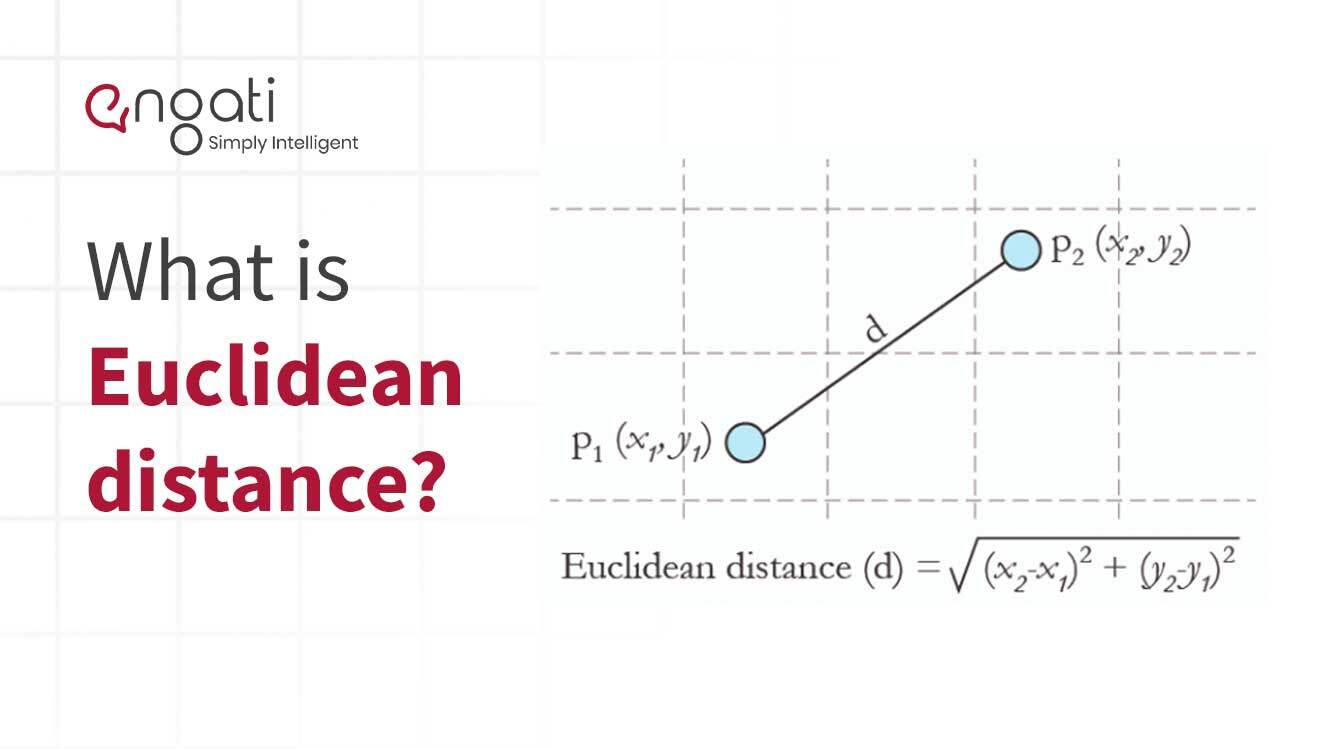
✅ Simple to implement and understand  
 ✅ Works for classification and regression  
 ✅ No training phase (fast training)

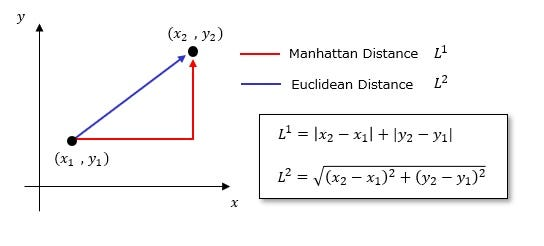
## **6. Disadvantages**

❌ Slow prediction for large datasets (must calculate distance for all points)  
 ❌ Sensitive to irrelevant features and different scales (needs feature scaling)  
 ❌ Curse of dimensionality – performance drops with too many features

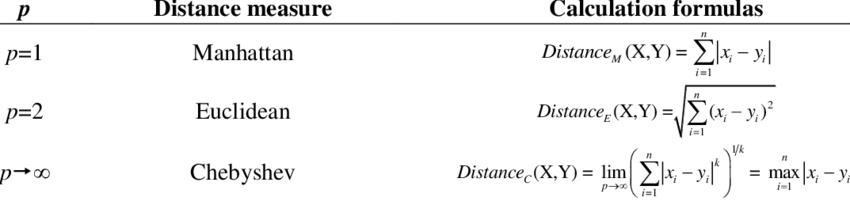
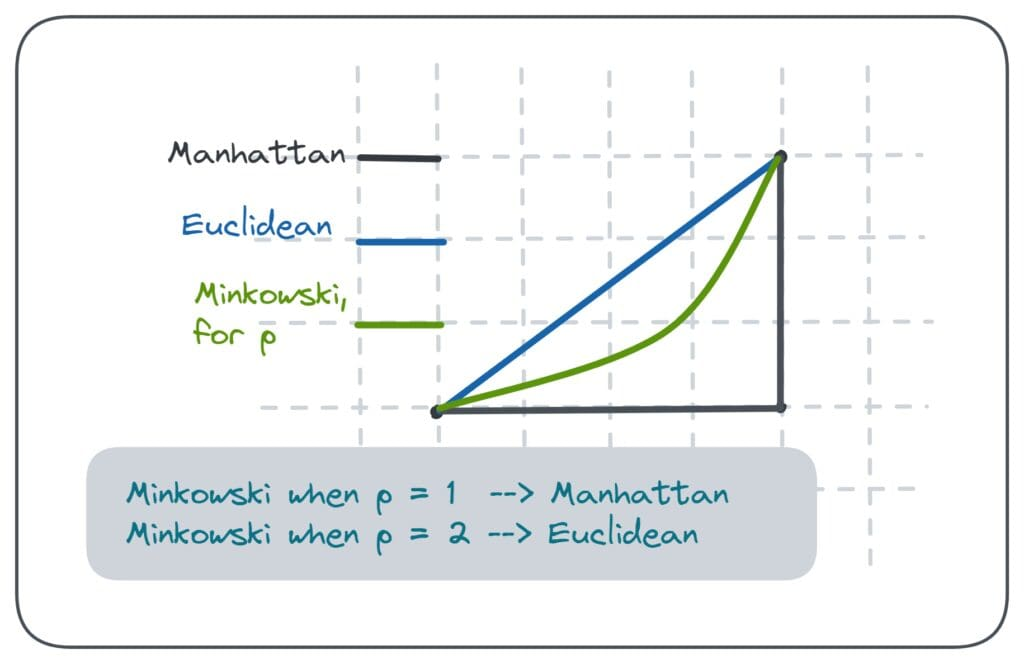
## **7. Distance Metrics Used in KNN**

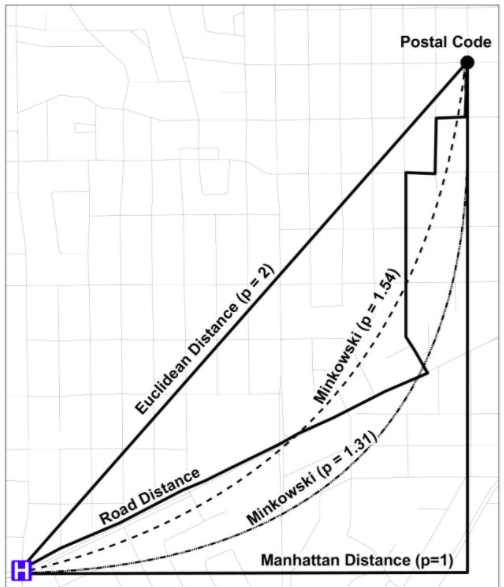
* **Euclidean Distance:**

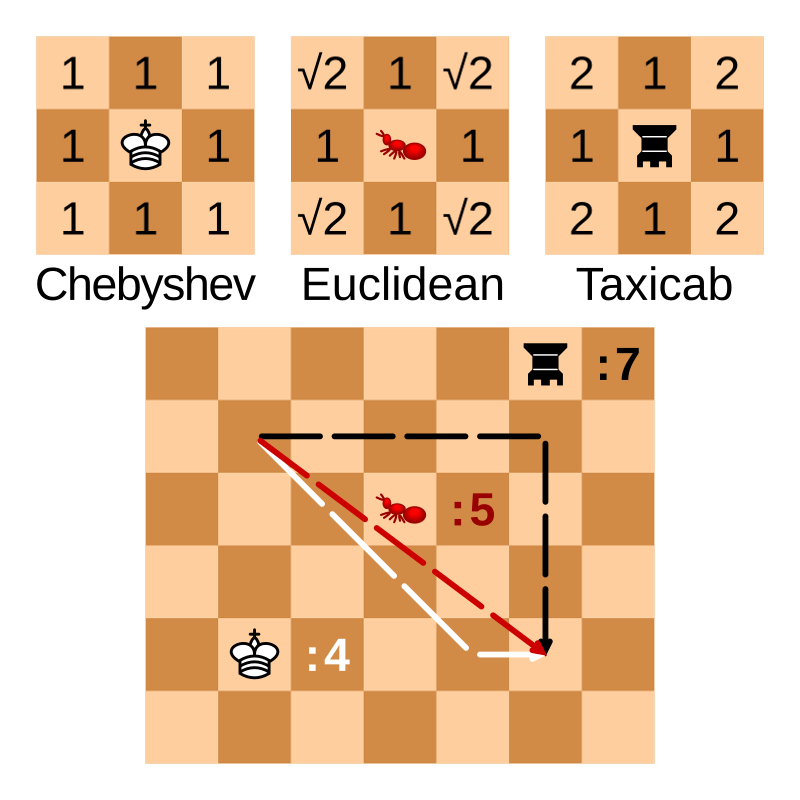


* **Manhattan Distance:**
* **Minkowski Distance** (generalized form)

Minkowski distance is a generalized distance metric used in multi-dimensional space, named after Hermann Minkowski, that encompasses other common distance measures like [Euclidean distance](https://www.google.com/search?sa=X&sca_esv=a46a1b8edd07c513&biw=1366&bih=607&sxsrf=AE3TifOWkrYRC-8srbOKZpUMZchM15QmtA%3A1756964498193&q=Euclidean+distance&ved=2ahUKEwjN3rDmsr6PAxVJz6ACHezkAEYQxccNegQIJhAC&mstk=AUtExfCJY46ypk-SOcHlkVIxp9cAtdijd_rgiVpW_si46SncG1KeTjFJqnyeh4l6h9Sc7t30PTDMwMGPqOV6G3IYrp3aBLrk87Yt9YGGpL5RTksVu1uXOiISamkKC5yoLB6ry_F_jw79i8nSAsfG9e7-Tx4j_7bwidyKQm5rGqih3u7NiXK-gQisME8VBLq9hEQgPtNA5EeJFazVE88znyPF_uY36sSqiLLfNsBV45lns_USOjkZTmBa25MtcwVSVbJNFIavdlOndOPKMqujcPWLQ21abEsd0XiBrvqtm6wr0pvGHQ&csui=3) and [Manhattan distance](https://www.google.com/search?sa=X&sca_esv=a46a1b8edd07c513&biw=1366&bih=607&sxsrf=AE3TifOWkrYRC-8srbOKZpUMZchM15QmtA%3A1756964498193&q=Manhattan+distance&ved=2ahUKEwjN3rDmsr6PAxVJz6ACHezkAEYQxccNegQIJhAD&mstk=AUtExfCJY46ypk-SOcHlkVIxp9cAtdijd_rgiVpW_si46SncG1KeTjFJqnyeh4l6h9Sc7t30PTDMwMGPqOV6G3IYrp3aBLrk87Yt9YGGpL5RTksVu1uXOiISamkKC5yoLB6ry_F_jw79i8nSAsfG9e7-Tx4j_7bwidyKQm5rGqih3u7NiXK-gQisME8VBLq9hEQgPtNA5EeJFazVE88znyPF_uY36sSqiLLfNsBV45lns_USOjkZTmBa25MtcwVSVbJNFIavdlOndOPKMqujcPWLQ21abEsd0XiBrvqtm6wr0pvGHQ&csui=3) through a parameter 'p'.







<https://tuhinmukherjee74.medium.com/different-types-of-distances-used-in-machine-learning-explained-550e2979752c>

## **8. Assignment**

**Dataset:** Use the Iris dataset from sklearn.  
 **Task:**

1. Implement KNN for classification to predict flower species.
2. Experiment with different K values (1, 3, 5, 7, 11).
3. Compare accuracy for each K.
4. Plot accuracy vs. K graph.

## 

## 

## **9. Python Code Example**

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

import matplotlib.pyplot as plt

# Load dataset

iris = load\_iris()

X, y = iris.data, iris.target

# Train-Test Split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Try different values of K

k\_values = [1, 3, 5, 7, 9, 11]

accuracies = []

for k in k\_values:

model = KNeighborsClassifier(n\_neighbors=k)

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

acc = accuracy\_score(y\_test, y\_pred)

accuracies.append(acc)

print(f"K={k}, Accuracy={acc:.4f}")

# Plot accuracy vs K

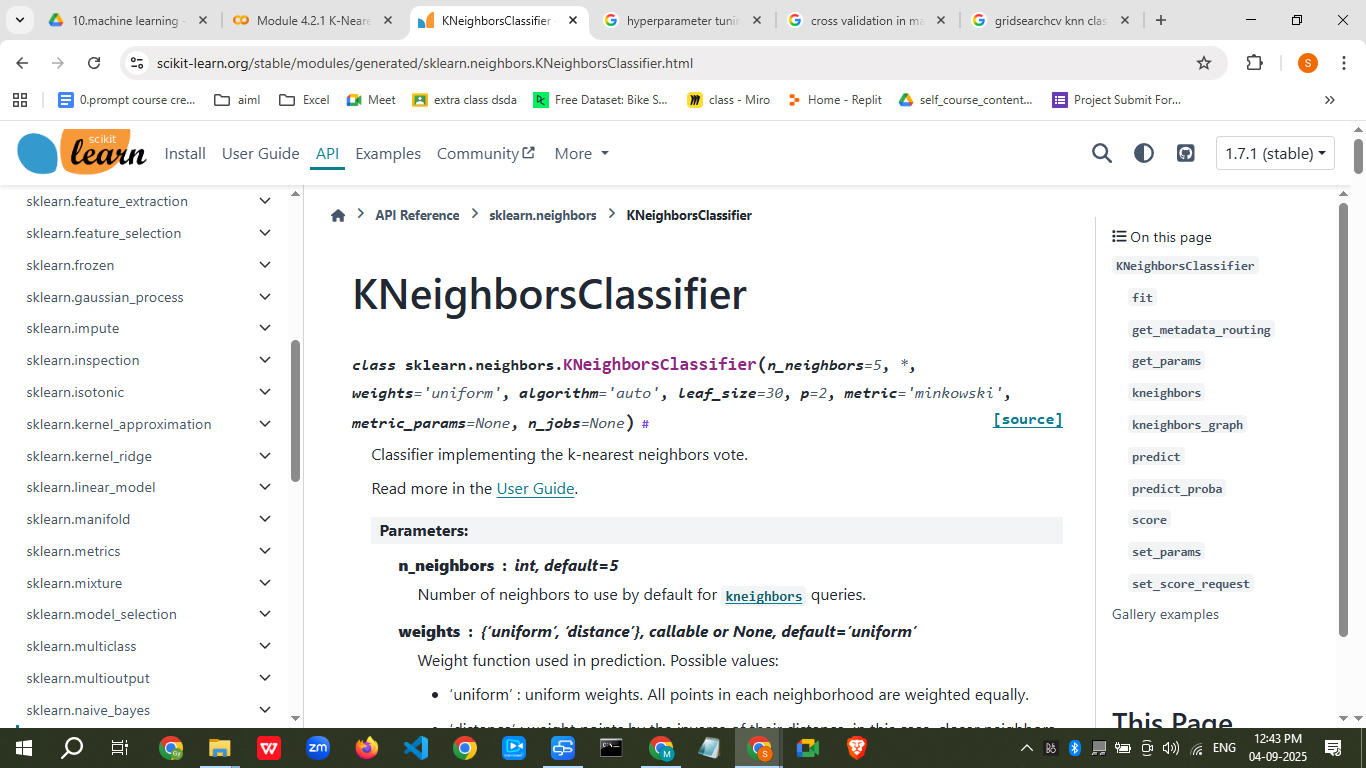
plt.plot(k\_values, accuracies, marker='o')

plt.xlabel('K Value')

plt.ylabel('Accuracy')

plt.title('KNN Accuracy vs K')

plt.show()



—-------------------------

—

### **6. Use Cases**

* Recommender systems (finding similar items/users).
* Handwriting recognition (digit classification).
* Medical diagnosis (classifying patient conditions).
* Fraud detection.

### **7. Assignment**

**Dataset:** Iris.csv or any classification dataset.  
 **Tasks:**

1. Implement KNN from scratch (without scikit-learn).
2. Compare with KNeighborsClassifier from scikit-learn.
3. Find best K using cross-validation.
4. Plot accuracy vs K.

## **💻 Streamlit App: KNN Interactive Tool**

import streamlit as st

import pandas as pd

import numpy as np

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, classification\_report

import matplotlib.pyplot as plt

import seaborn as sns

st.title("🔍 K-Nearest Neighbors (KNN) Classifier App")

# File upload

uploaded\_file = st.file\_uploader("Upload CSV file", type=["csv"])

if uploaded\_file is not None:

df = pd.read\_csv(uploaded\_file)

st.subheader("📊 Dataset Preview")

st.write(df.head())

# Feature & target selection

all\_columns = df.columns.tolist()

target\_col = st.selectbox("Select Target Column", all\_columns)

feature\_cols = st.multiselect("Select Feature Columns", [col for col in all\_columns if col != target\_col])

if feature\_cols and target\_col:

X = df[feature\_cols]

y = df[target\_col]

# Train-test split

test\_size = st.slider("Test Size (fraction)", 0.1, 0.5, 0.2)

random\_state = st.number\_input("Random State", value=42, step=1)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=test\_size, random\_state=random\_state)

# Scaling

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# K selection

k\_value = st.slider("Select K", 1, 20, 5)

# Model training

knn = KNeighborsClassifier(n\_neighbors=k\_value)

knn.fit(X\_train\_scaled, y\_train)

y\_pred = knn.predict(X\_test\_scaled)

# Metrics

acc = accuracy\_score(y\_test, y\_pred)

st.subheader(f"✅ Accuracy: {acc:.2f}")

st.text("Classification Report")

st.text(classification\_report(y\_test, y\_pred))

# Confusion Matrix

cm = confusion\_matrix(y\_test, y\_pred)

fig, ax = plt.subplots()

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', ax=ax)

ax.set\_xlabel("Predicted")

ax.set\_ylabel("Actual")

st.pyplot(fig)

✅ **How this helps you in class:**

* **Theory + Hands-on** in one session.
* Students can **upload any dataset** and experiment.
* They will understand **how K affects predictions** in real-time.
* Encourages **interactive learning**.

If you want, I can **extend this app** to also include:

* **Decision boundary plots** (for 2D datasets)
* **Regression version of KNN**
* **Auto K tuning** using cross-validation.

Do you want me to add these **advanced features** too? That would make it a complete KNN learning tool.

[K-Nearest Neighbor(KNN) Algorithm - GeeksforGeeks](https://www.geeksforgeeks.org/machine-learning/k-nearest-neighbours/)

[KD-TREE AND BALL TREE IN KNN ALGORITHM | by Narasimharaodevisetti | Medium](https://medium.com/@narasimharaodevisetti14/kd-tree-and-ball-tree-in-knn-algorithm-09a86d1bc6e6)

