Name: Harsh Patil
Class: D15C/37

Experiment No. 4

1. Dataset Source

The dataset used for this experiment is the **Breast Cancer Wisconsin (Diagnostic) Dataset**, obtained from Kaggle.

- **Dataset Link:** https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data

2. Dataset Description

The Breast Cancer Wisconsin dataset is a real-world medical dataset used to predict whether a breast tumor is **malignant** or **benign**.

- **Total instances:** 569
- **Number of features:** 30 numeric features
- **Target variable:** `diagnosis`
  - `M` → Malignant (encoded as 1)
  - `B` → Benign (encoded as 0)

Feature Description

The features are computed from digitized images of fine needle aspirate (FNA) of breast masses and include measurements such as:

- Radius
- Texture
- Perimeter
- Area
- Smoothness
- Compactness
- Concavity
- Symmetry

The dataset contains no missing values and is well-suited for classification tasks.

3. Mathematical Formulation of KNN Algorithm

K-Nearest Neighbors (KNN) is a **non-parametric, instance-based learning algorithm**.

Distance Calculation

The most commonly used distance metric is **Euclidean distance**:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Classification Rule

Given a test instance:

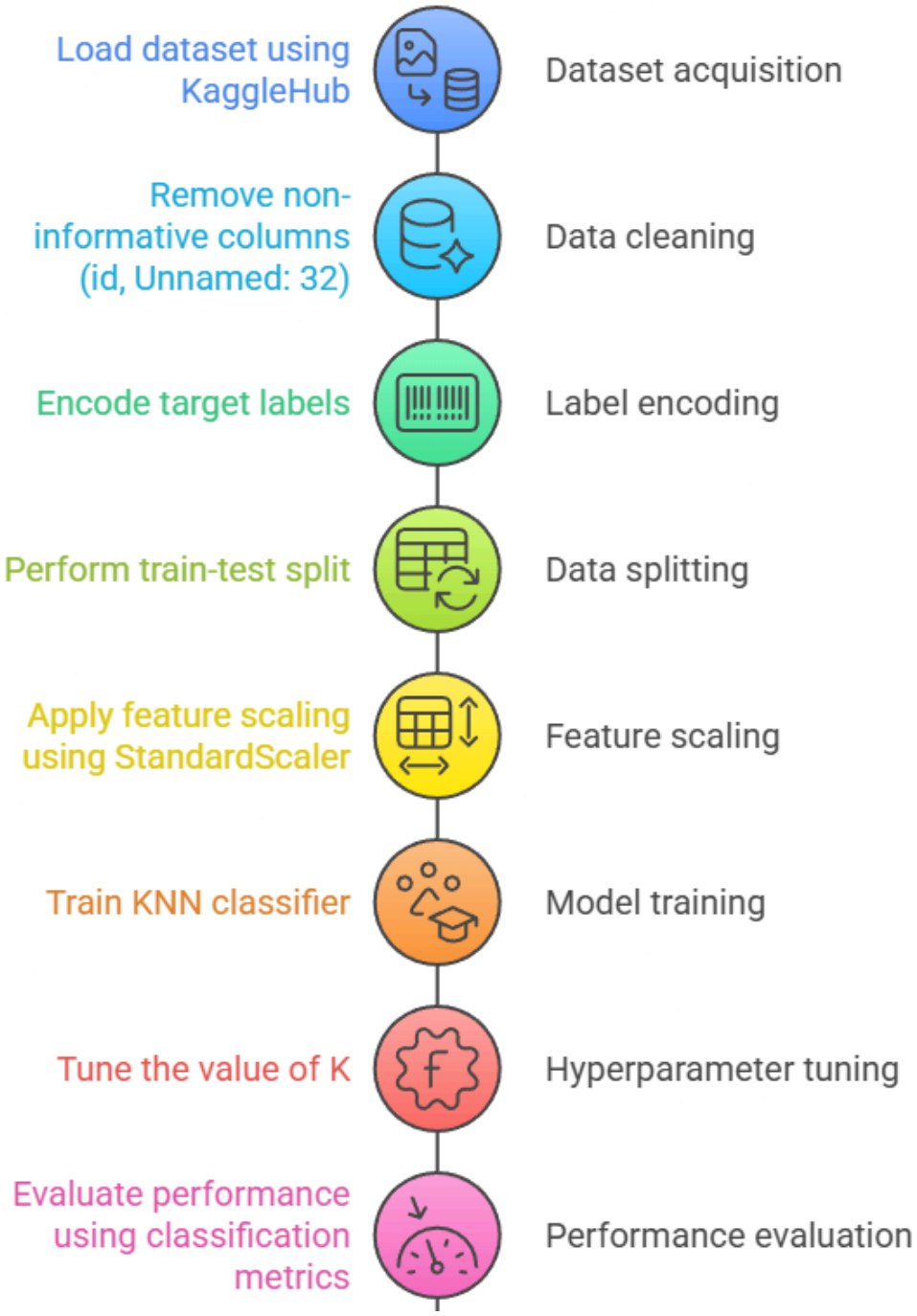
1. Compute the distance between the test instance and all training instances.
2. Select the **K nearest neighbors**.
3. Assign the class based on **majority voting** among the K neighbors.

$$\hat{y} = \text{mode}(y_1, y_2, ..., y_K)$$

4. Algorithm Limitations

- Computationally expensive for large datasets
- Sensitive to feature scaling
- Choice of K significantly impacts performance
- Performs poorly with high-dimensional data (curse of dimensionality)
- Requires storing entire training dataset

5. Methodology / Workflow



Workflow Diagram:

Dataset → Preprocessing → Scaling → KNN Training → Prediction → Evaluation

6. Performance Analysis

The KNN model performance was evaluated using:

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

The model achieved high accuracy due to:

- Well-separated classes
- Proper feature scaling
- Optimal choice of K

Comparison with previous models shows that KNN performs competitively but is more computationally expensive during prediction.

7. Hyperparameter Tuning

The key hyperparameter in KNN is **K (number of neighbors)**.

Tuning Strategy

- Tested multiple K values (e.g., K = 3, 5, 7, 9, 11)
- Selected K with highest validation accuracy

Impact

- Small K → Overfitting
- Large K → Underfitting
- Optimal K provides best bias-variance tradeoff

After tuning, the selected K resulted in improved generalization and reduced misclassification.

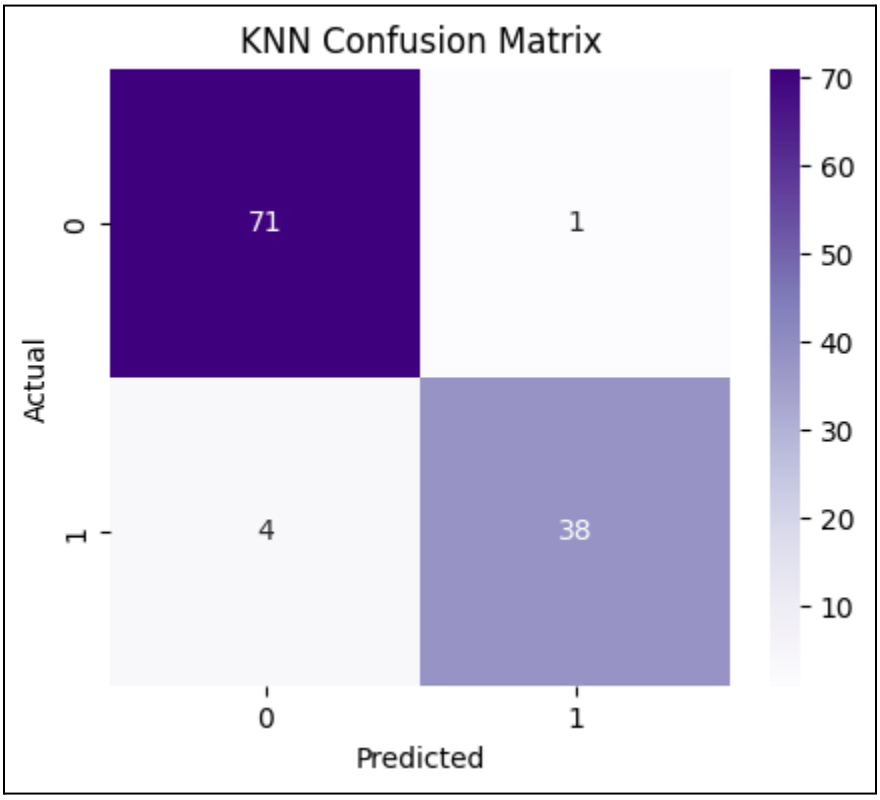
OUTPUT:

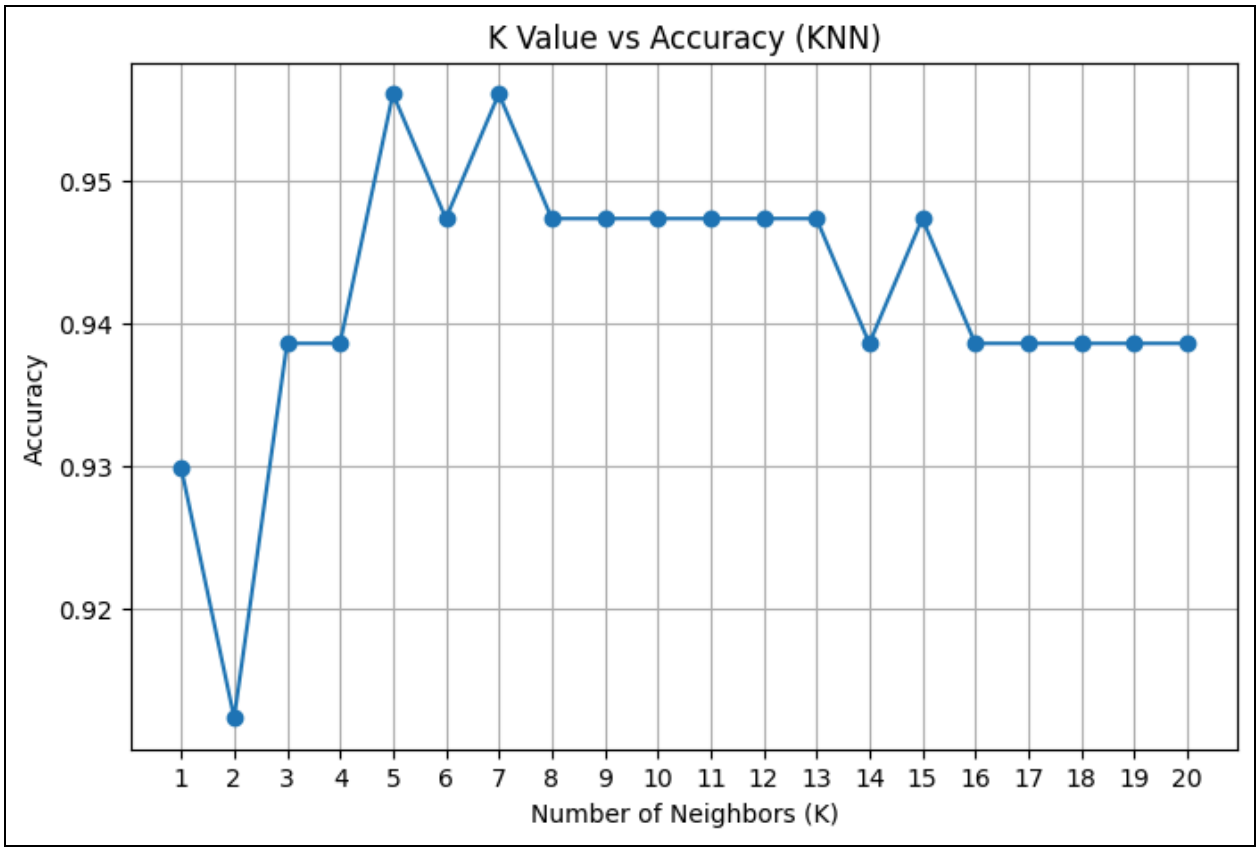
Code:

```
!pip install kagglehub -q

import kagglehub
from kagglehub import KaggleDatasetAdapter

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

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

df = kagglehub.load_dataset(
    KaggleDatasetAdapter.PANDAS,
    "uciml/breast-cancer-wisconsin-data",
    "data.csv"
)

print("Dataset Shape:", df.shape)
df.head()

df.drop(columns=["id", "Unnamed: 32"], inplace=True)
df["diagnosis"] = df["diagnosis"].map({"M": 1, "B": 0})

X = df.drop("diagnosis", axis=1)
y = df["diagnosis"]

X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42,
    stratify=y
)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train_scaled, y_train)

y_pred = knn.predict(X_test_scaled)

print("\n=== KNN Performance ===")
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))

plt.figure(figsize=(5, 4))
sns.heatmap(confusion_matrix(y_test, y_pred),
            annot=True, fmt="d", cmap="Purples")
plt.title("KNN Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

k_values = range(1, 21)
accuracies = []

for k in k_values:
    model = KNeighborsClassifier(n_neighbors=k)
    model.fit(X_train_scaled, y_train)
    preds = model.predict(X_test_scaled)
    accuracies.append(accuracy_score(y_test, preds))

plt.figure(figsize=(8, 5))
plt.plot(k_values, accuracies, marker='o')
plt.xlabel("Number of Neighbors (K)")
plt.ylabel("Accuracy")
plt.title("K Value vs Accuracy (KNN)")
plt.xticks(k_values)
plt.grid(True)
plt.show()
```

```
=== KNN Performance ===
Accuracy: 0.956140350877193
              precision    recall  f1-score   support

           0       0.95      0.99      0.97        72
           1       0.97      0.90      0.94        42

    accuracy                           0.96       114
   macro avg       0.96      0.95      0.95       114
weighted avg       0.96      0.96      0.96       114
```





8. Conclusion

In this experiment, the K-Nearest Neighbors algorithm was successfully implemented on a real-world medical dataset. The model demonstrated strong classification performance when proper scaling and hyperparameter tuning were applied. Although KNN is simple and effective, it is less scalable compared to tree-based ensemble methods such as Random Forest.

This experiment highlights the importance of distance-based learning and parameter tuning in classification tasks.