Name : Bhoomi Mangesh Naik
Class : D15C
Roll No. : 34
Practical No. : 4

Aim : Implement K-Nearest Neighbors (KNN) and evaluate model performance.

Theory :

1. Dataset Source

The dataset used for this experiment is obtained from Kaggle:

Breast Cancer Wisconsin Dataset
https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data

2. Dataset Description

This dataset contains diagnostic measurements used to predict whether a breast tumor is malignant or benign.

Dataset Characteristics

- **Total Records:** 569
- **Type:** Structured numerical dataset
- **Target Variable:** diagnosis

Class Labels

- M → Malignant (Cancer present)
- B → Benign (Cancer absent)

Features

The dataset includes 30 numerical features such as:

- Mean radius
- Mean texture
- Mean perimeter
- Mean area
- Mean smoothness

3. Mathematical Formulation of KNN

K-Nearest Neighbors is a **non-parametric, distance-based** algorithm.

Distance Metric (Euclidean Distance):

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

Prediction Rule:

The class of a test sample is determined by **majority voting** among the *k*k nearest neighbors.

4. Algorithm Limitations

- Computationally expensive for large datasets
- Sensitive to noisy data
- Requires proper feature scaling
- Performance depends heavily on the choice of *k*k

5. Methodology / Workflow

1. Dataset collection from Kaggle
2. Data upload in Google Colab
3. Data cleaning and duplicate removal
4. Feature-target separation
5. Feature scaling using StandardScaler
6. Train-test split
7. KNN model training
8. Model evaluation

Workflow Diagram:

Dataset → Cleaning → Scaling → Train-Test Split → KNN Training → Evaluation

6. Performance Analysis

Evaluation Metrics Used

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

Interpretation:
The confusion matrix shows correct and incorrect classifications. High accuracy and balanced precision-recall values indicate effective classification by the KNN model.

7. Hyperparameter Tuning

Hyperparameter Tuned

- n_neighbors (k)

Different values of *k*k were tested (e.g., 3, 5, 7, 9).

Impact of Tuning:

- Small *k*k → Overfitting
- Large *k*k → Underfitting
- Optimal *k*k improves accuracy and generalization

Code and Output :

```
from google.colab import files
import pandas as pd
# Upload dataset
uploaded = files.upload()
# Load dataset
df = pd.read_csv("data.csv")
print("Initial Shape:", df.shape)
# Drop unnecessary columns
df = df.drop(columns=['id', 'Unnamed: 32'], errors='ignore')
# Remove duplicate rows
df = df.drop_duplicates()
# Check missing values
print("\nMissing Values:\n", df.isnull().sum())
print("\nShape after cleaning:", df.shape)
df.head()
```

```
Choose Files No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving data.csv to data (1).csv
Initial Shape: (569, 33)

Missing Values:
 diagnosis                 0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
dtype: int64

Shape after cleaning: (569, 31)
```

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	...	radius_worst	texture_worst	perimeter_worst	area_worst	smoothness_w
0	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	...	25.38	17.33	184.60	2019.0	0.
1	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	...	24.99	23.41	158.80	1956.0	0.
2	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	...	23.57	25.53	152.50	1709.0	0.
3	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	...	14.91	26.50	98.87	567.7	0.
4	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	...	22.54	16.67	152.20	1575.0	0.

5 rows × 31 columns

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
# Features and target
X = df.drop("diagnosis", axis=1)
y = df["diagnosis"].map({'M': 1, 'B': 0})
print("X shape:", X.shape)
print("y shape:", y.shape)
# Feature scaling
```

```
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, random_state=42
)
# KNN model
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
# Predictions
y_pred = knn.predict(X_test)
print("KNN Accuracy:", accuracy_score(y_test, y_pred))
```

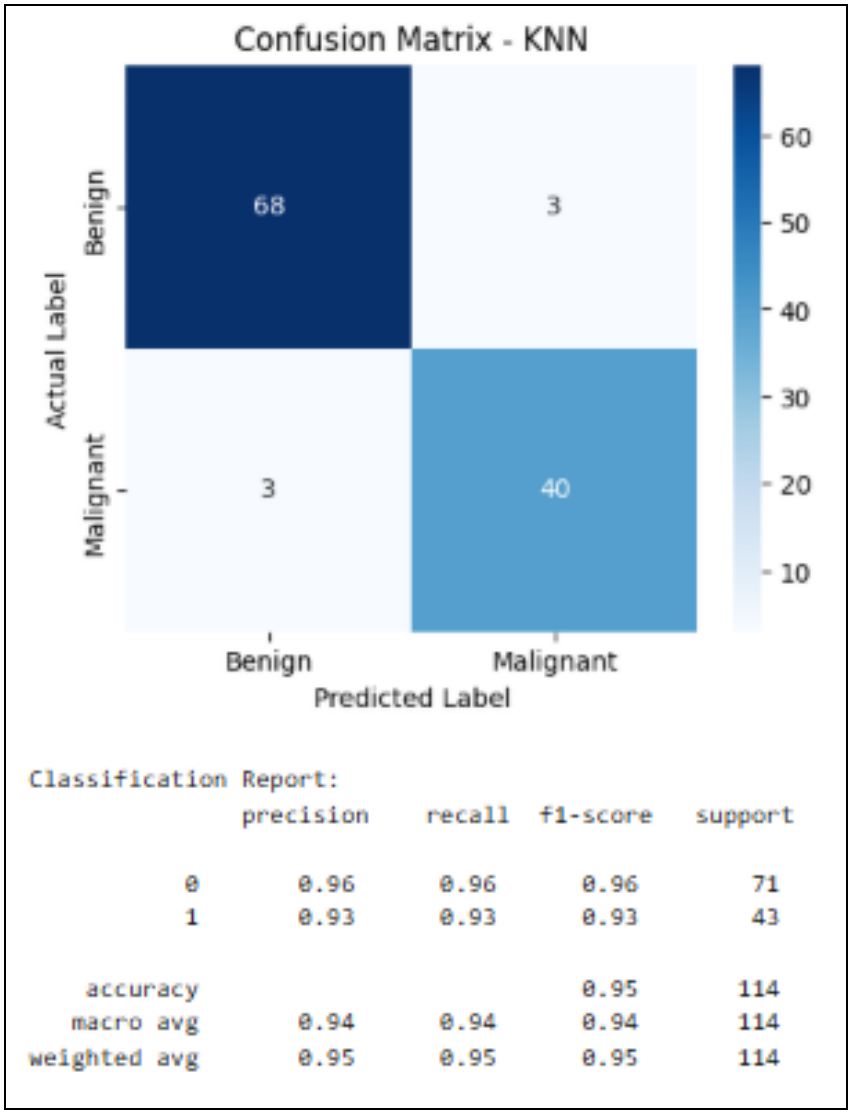
```
X shape: (569, 30)
y shape: (569,)
KNN Accuracy: 0.9473684210526315
```

```
from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt
import seaborn as sns

cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(5,4))
sns.heatmap(
    cm, annot=True, fmt="d", cmap="Blues",
    xticklabels=["Benign", "Malignant"],
    yticklabels=["Benign", "Malignant"]
)
plt.xlabel("Predicted Label")
plt.ylabel("Actual Label")
plt.title("Confusion Matrix - KNN")
plt.show()

print("\nClassification Report:\n", classification_report(y_test, y_pred))
```



Classification Report:

	precision	recall	f1-score	support
0	0.96	0.96	0.96	71
1	0.93	0.93	0.93	43
accuracy			0.95	114
macro avg	0.94	0.94	0.94	114
weighted avg	0.95	0.95	0.95	114

Google Colab Link for Code and Output : [Link for Code and Output]

Conclusion :

In this experiment, the K-Nearest Neighbors algorithm was successfully implemented on a real-world breast cancer dataset. Proper data cleaning and feature scaling significantly improved performance. The model achieved high accuracy and demonstrated the effectiveness of KNN for medical classification tasks.