LAB TEST-2(EVEN)

1.A.

from sklearn.datasets import load\_breast\_cancer

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion\_matrix, classification\_report

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

data = load\_breast\_cancer()

df = pd.DataFrame(data.data, columns=data.feature\_names)

df['target'] = data.target

print("First 10 rows of the dataset:\n", df.head(10))

B.

print("Basic statistics for the first 10 rows:")

print(df.head(10).describe())

C.

print("\nMissing values check:\n", df.isnull().sum())

print("Handling missing values is important because they can distort the statistical analysis, \

and most ML models cannot handle missing data without preprocessing.")

D.

X = df.drop('target', axis=1)

y = df['target']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42, stratify=y)

E.

scaler = StandardScaler()

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

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

F.

corr\_matrix = pd.DataFrame(X\_train\_scaled, columns=data.feature\_names).corr()

plt.figure(figsize=(12, 10))

sns.heatmap(corr\_matrix, annot=False, cmap='coolwarm', cbar=True)

plt.title("Correlation Matrix")

plt.show()

G.

k\_range = range(1, 26)

accuracy\_scores = []

for k in k\_range:

knn = KNeighborsClassifier(n\_neighbors=k)

scores = cross\_val\_score(knn, X\_train\_scaled, y\_train, cv=5, scoring='accuracy')

accuracy\_scores.append(scores.mean())

H.

plt.plot(k\_range, accuracy\_scores, marker='o')

plt.title('Accuracy for Different k Values')

plt.xlabel('Number of Neighbors (k)')

plt.ylabel('Cross-validated Accuracy')

plt.show()

best\_k = accuracy\_scores.index(max(accuracy\_scores)) + 1

print(f"\nOptimal k value: {best\_k}")

knn = KNeighborsClassifier(n\_neighbors=best\_k)

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

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

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

class\_report = classification\_report(y\_test, y\_pred)

print("\nConfusion Matrix:\n", conf\_matrix)

print("\nClassification Report:\n", class\_report)

2.

import numpy as np

def reverse\_and\_filter(strings):

reversed\_strings = [s[::-1] for s in strings if len(s[::-1]) >= 5]

return reversed\_strings

inputs = ["hello", "world", "abc", "python", "mod"]

result = reverse\_and\_filter(inputs)

print("Result:", result)