**Solution of Assignment-6**

#1

def find\_second\_largest(nums):

    nums = list(set(nums))

    nums.sort()

    return nums[-2] if len(nums) > 1 else None

nums = [10, 20, 4, 45, 99, 99]

print("Second Largest:", find\_second\_largest(nums))

Second Largest: 45

#2

def swap\_elements(string\_list, i, j):

    if i < len(string\_list) and j < len(string\_list):

        string\_list[i], string\_list[j] = string\_list[j], string\_list[i]

    return string\_list

string\_list = ["apple", "banana", "cherry"]

print("Swapped List:", swap\_elements(string\_list, 0, 2))

Swapped List: ['cherry', 'banana', 'apple']

#3

def remove\_empty\_lists(lst):

    return [i for i in lst if i]

# Example usage

lst = [1, 2, [], 4, [], 5]

print("List after removing empty lists:", remove\_empty\_lists(lst))

List after removing empty lists: [1, 2, 4, 5]

#4

def reverse\_row\_sort(matrix):

    return [sorted(row, reverse=True) for row in matrix]

# Example usage

matrix = [[1, 3, 2], [4, 6, 5], [9, 7, 8]]

print("Matrix after reverse row sort:", reverse\_row\_sort(matrix))

Matrix after reverse row sort: [[3, 2, 1], [6, 5, 4], [9, 8, 7]]

#5

def pair\_with\_rear(matrix):

    return [[(elem, row[-1]) for elem in row] for row in matrix]

# Example usage

matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

print("Matrix paired with rear element:", pair\_with\_rear(matrix))

Matrix paired with rear element: [[(1, 3), (2, 3), (3, 3)], [(4, 6), (5, 6), (6, 6)], [(7, 9), (8, 9), (9, 9)]]

#6

def sort\_dict\_by\_key(d):

    return dict(sorted(d.items()))

# Example usage

d = {'banana': 3, 'apple': 4, 'cherry': 2}

print("Sorted Dictionary:", sort\_dict\_by\_key(d))

Sorted Dictionary: {'apple': 4, 'banana': 3, 'cherry': 2}

#7

def merge\_dictionaries(d1, d2):

    return {\*\*d1, \*\*d2}

# Example usage

d1 = {'a': 1, 'b': 2}

d2 = {'c': 3, 'd': 4}

print("Merged Dictionary:", merge\_dictionaries(d1, d2))

Merged Dictionary: {'a': 1, 'b': 2, 'c': 3, 'd': 4}

#8

def replace\_words(text, word\_dict):

    words = text.split()

    return ' '.join([word\_dict.get(word, word) for word in words])

# Example usage

text = "hello world"

word\_dict = {'hello': 'hi', 'world': 'earth'}

print("Replaced Text:", replace\_words(text, word\_dict))

Replaced Text: hi earth

#9

from sklearn import datasets

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

from sklearn.neighbors import KNeighborsClassifier

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

# Load Iris dataset

iris = datasets.load\_iris()

X = iris.data

y = iris.target

# 80-20 train-test split

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

# Apply KNN classifier

knn = KNeighborsClassifier(n\_neighbors=3)

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

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

# Display accuracy

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

print("Accuracy:", accuracy)

# Generate confusion matrix

print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))

# Display precision, recall, F1-score, sensitivity, specificity, kappa

print("Classification Report:\n", classification\_report(y\_test, y\_pred))

Accuracy: 1.0

Confusion Matrix:

[[10 0 0]

[ 0 9 0]

[ 0 0 11]]

Classification Report:

precision recall f1-score support

0 1.00 1.00 1.00 10

1 1.00 1.00 1.00 9

2 1.00 1.00 1.00 11

accuracy 1.00 30

macro avg 1.00 1.00 1.00 30

weighted avg 1.00 1.00 1.00 30

#10

import matplotlib.pyplot as plt

# Function to train KNN and return accuracy

def knn\_accuracy(X, y, test\_size):

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

    knn = KNeighborsClassifier(n\_neighbors=3)

    knn.fit(X\_train, y\_train)

    y\_pred = knn.predict(X\_test)

    return accuracy\_score(y\_test, y\_pred)

# 70-30 and 65-35 splits

accuracy\_80\_20 = accuracy

accuracy\_70\_30 = knn\_accuracy(X, y, test\_size=0.3)

accuracy\_65\_35 = knn\_accuracy(X, y, test\_size=0.35)

# Display the accuracies

print("80-20 Accuracy:", accuracy\_80\_20)

print("70-30 Accuracy:", accuracy\_70\_30)

print("65-35 Accuracy:", accuracy\_65\_35)

# Bar graph for the accuracies

splits = ['80-20', '70-30', '65-35']

accuracies = [accuracy\_80\_20, accuracy\_70\_30, accuracy\_65\_35]

plt.bar(splits, accuracies, color=['blue', 'orange', 'green'])

plt.xlabel('Train-Test Split')

plt.ylabel('Accuracy')

plt.title('KNN Accuracy for Different Splits')

plt.show()

80-20 Accuracy: 1.0

70-30 Accuracy: 1.0

65-35 Accuracy: 0.9811320754716981

