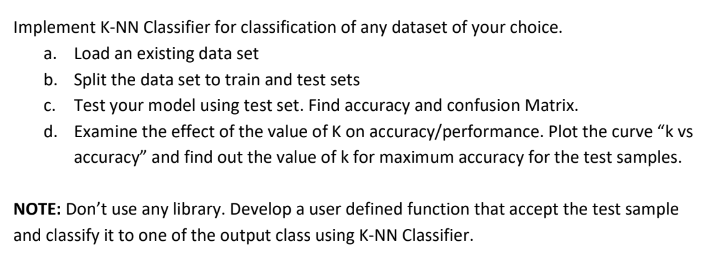
**Machine Learning Lab CSE 336L**

**Week – 5 K-NN Classifier**



**Code:**

import csv

import random

import math

import matplotlib.pyplot as plt

def load\_dataset(filename, split\_ratio):

training\_data = []

test\_data = []

with open(filename, 'r') as file:

lines = csv.reader(file)

dataset = list(lines)

for i in range(len(dataset) - 1):

for j in range(4):

dataset[i][j] = float(dataset[i][j])

if random.random() < split\_ratio:

training\_data.append(dataset[i])

else:

test\_data.append(dataset[i])

return training\_data, test\_data

def euclidean\_distance(instance1, instance2, length):

distance = 0

for i in range(length):

distance += (instance1[i] - instance2[i]) \*\* 2

return math.sqrt(distance)

def get\_neighbors(training\_set, test\_instance, k):

distances = []

length = len(test\_instance) - 1

for i in range(len(training\_set)):

dist = euclidean\_distance(test\_instance, training\_set[i], length)

distances.append((training\_set[i], dist))

distances.sort(key=lambda x: x[1])

neighbors = []

for i in range(k):

neighbors.append(distances[i][0])

return neighbors

def predict\_classification(neighbors):

class\_votes = {}

for i in range(len(neighbors)):

response = neighbors[i][-1]

if response in class\_votes:

class\_votes[response] += 1

else:

class\_votes[response] = 1

sorted\_votes = sorted(class\_votes.items(), key=lambda x: x[1], reverse=True)

return sorted\_votes[0][0]

def evaluate\_model(test\_set, predictions):

correct = 0

confusion\_matrix = {}

for i in range(len(test\_set)):

actual\_class = test\_set[i][-1]

predicted\_class = predictions[i]

if actual\_class == predicted\_class:

correct += 1

if actual\_class not in confusion\_matrix:

confusion\_matrix[actual\_class] = {}

if predicted\_class not in confusion\_matrix[actual\_class]:

confusion\_matrix[actual\_class][predicted\_class] = 0

confusion\_matrix[actual\_class][predicted\_class] += 1

accuracy = (correct / float(len(test\_set))) \* 100.0

return accuracy, confusion\_matrix

def k\_nearest\_neighbors(training\_set, test\_set, k):

predictions = []

for test\_instance in test\_set:

neighbors = get\_neighbors(training\_set, test\_instance, k)

result = predict\_classification(neighbors)

predictions.append(result)

return predictions

filename = "data/iris.csv"

split\_ratio = 0.7

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

training\_set, test\_set = load\_dataset(filename, split\_ratio)

accuracies = []

for k in k\_values:

predictions = k\_nearest\_neighbors(training\_set, test\_set, k)

accuracy, confusion\_matrix = evaluate\_model(test\_set, predictions)

accuracies.append(accuracy)

print(f'Accuracy for k={k}: {accuracy:.2f}%')

print('Confusion Matrix:')

for actual\_class, pred\_classes in confusion\_matrix.items():

print(f'Actual: {actual\_class}')

for pred\_class, count in pred\_classes.items():

print(f' Predicted: {pred\_class}, Count: {count}')

print('-----------------------------------')

plt.plot(k\_values, accuracies)

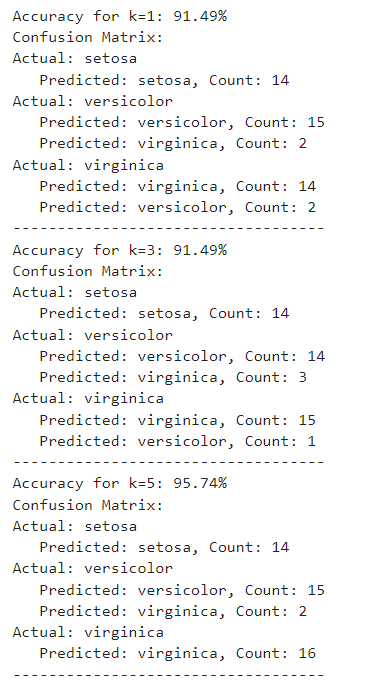
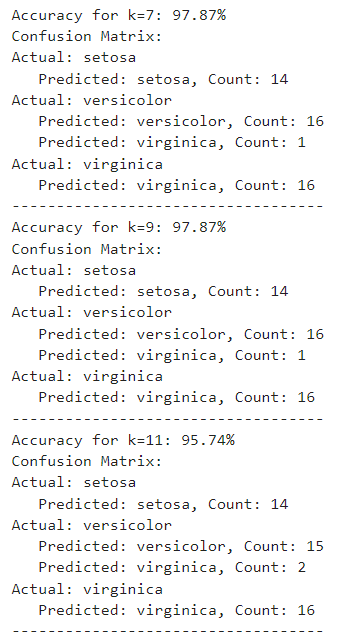
plt.xlabel('k values')

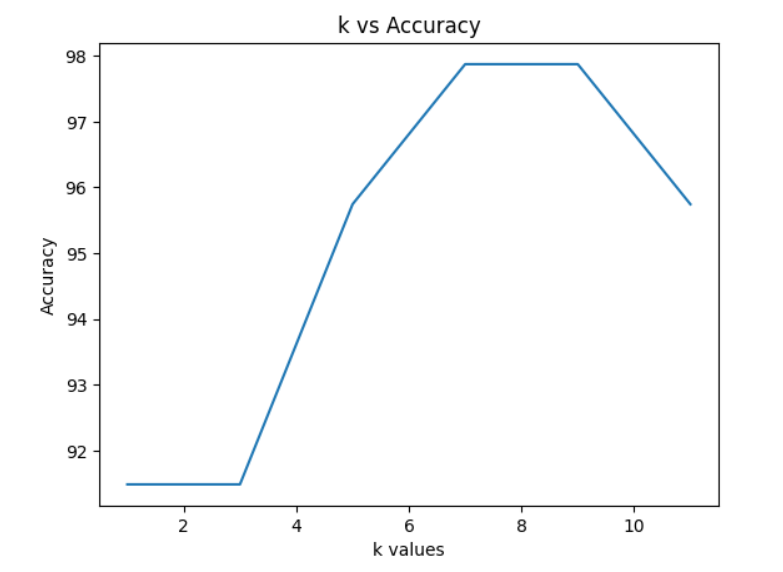
plt.ylabel('Accuracy')

plt.title('k vs Accuracy')

plt.show()

**Output:**

** **

****

**Submitted By:**

AP21110011080

Kuncham Pavan Vitesh

CSE-Q

Lab Date: 20-Mar -2024