Machine Learning Lab CSE 336L

Week – 5 K-NN Classifier

Implement K-NN Classifier for classification of any dataset of your choice.

a. Load an existing data set
b. Split the data set to train and test sets
c. Test your model using test set. Find accuracy and confusion Matrix.
d. Examine the effect of the value of K on accuracy/performance. Plot the curve "k vs accuracy" and find out the value of k for maximum accuracy for the test samples.

NOTE: Don't use any library. Develop a user defined function that accept the test sample and classify it to one of the output class using 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:
```

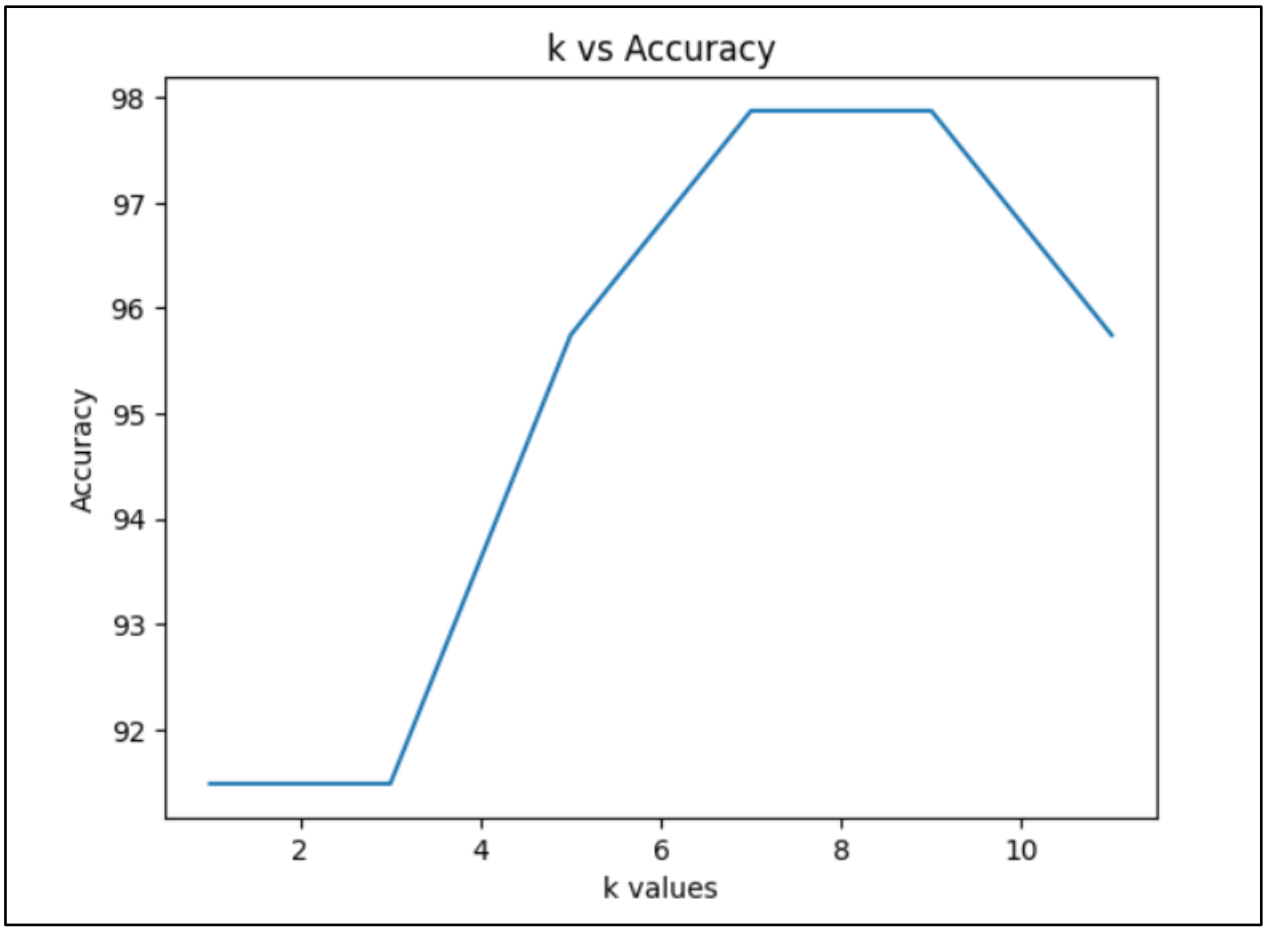
```
    predictions = k_nearest_neighbors(training_set, test_set, k)
    accuracy, confusion_matrix = evaluate_model(test_set, predictions)
    accuracies.append(accuracy)
    print(f'Accuracy for k={k}: {accuracy:.2f}%')
    print('Confusion Matrix:')
    for actual_class, pred_classes in confusion_matrix.items():
        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:

```
Accuracy for k=1: 91.49%
Confusion Matrix:
Actual: setosa
   Predicted: setosa, Count: 14
Actual: versicolor
   Predicted: versicolor, Count: 15
   Predicted: virginica, Count: 2
Actual: virginica
   Predicted: virginica, Count: 14
   Predicted: versicolor, Count: 2
-----------------------------------
Accuracy for k=3: 91.49%
Confusion Matrix:
Actual: setosa
   Predicted: setosa, Count: 14
Actual: versicolor
   Predicted: versicolor, Count: 14
   Predicted: virginica, Count: 3
Actual: virginica
   Predicted: virginica, Count: 15
   Predicted: versicolor, Count: 1
-----------------------------------
Accuracy for k=5: 95.74%
Confusion Matrix:
Actual: setosa
   Predicted: setosa, Count: 14
Actual: versicolor
   Predicted: versicolor, Count: 15
   Predicted: virginica, Count: 2
Actual: virginica
   Predicted: virginica, Count: 16
-----------------------------------
```

```
Accuracy for k=7: 97.87%
Confusion Matrix:
Actual: setosa
   Predicted: setosa, Count: 14
Actual: versicolor
   Predicted: versicolor, Count: 16
   Predicted: virginica, Count: 1
Actual: virginica
   Predicted: virginica, Count: 16
-----------------------------------
Accuracy for k=9: 97.87%
Confusion Matrix:
Actual: setosa
   Predicted: setosa, Count: 14
Actual: versicolor
   Predicted: versicolor, Count: 16
   Predicted: virginica, Count: 1
Actual: virginica
   Predicted: virginica, Count: 16
-----------------------------------
Accuracy for k=11: 95.74%
Confusion Matrix:
Actual: setosa
   Predicted: setosa, Count: 14
Actual: versicolor
   Predicted: versicolor, Count: 15
   Predicted: virginica, Count: 2
Actual: virginica
   Predicted: virginica, Count: 16
-----------------------------------
```



Submitted By:
AP21110011080
Kuncham Pavan Vitesh
CSE-Q
Lab Date: 20-Mar -2024