CSL7670 : Fundamentals of Machine Learning

Lab Report

॥ त्वं ज्ञानमयो विज्ञानमयोऽसि ॥

Name:	**Ganesh Kumar Nagal**
Roll Number:	**M23MEA004**
Program:	**M.Tech Adavance Manufacturing And Design**

Chapter 1

Lab-2

1.1 Objective

The objective of this lab assignment is to get familiarity with the K-Nearest Neighbors (K-NN) algorithm. By completing the problems and exercises in this lab.

1.2 Problem-1

1. (Apple vs Orange) You are given a K-NN code for the Apple vs Orange problem. Please read and understand the code. Now perform the following tasks:

1. Synthetically increase the dataset size to 50 samples.
2. Edit the code so that random 80%, 10%, and 10% samples are used for training, testing, and validation respectively.
3. Change the value of K to 3, 5, and 7 and compare the validation set and test set results.
4. Write a code that draws confusion matrices for different K. Use the following link to understand about Confusion Matrix.

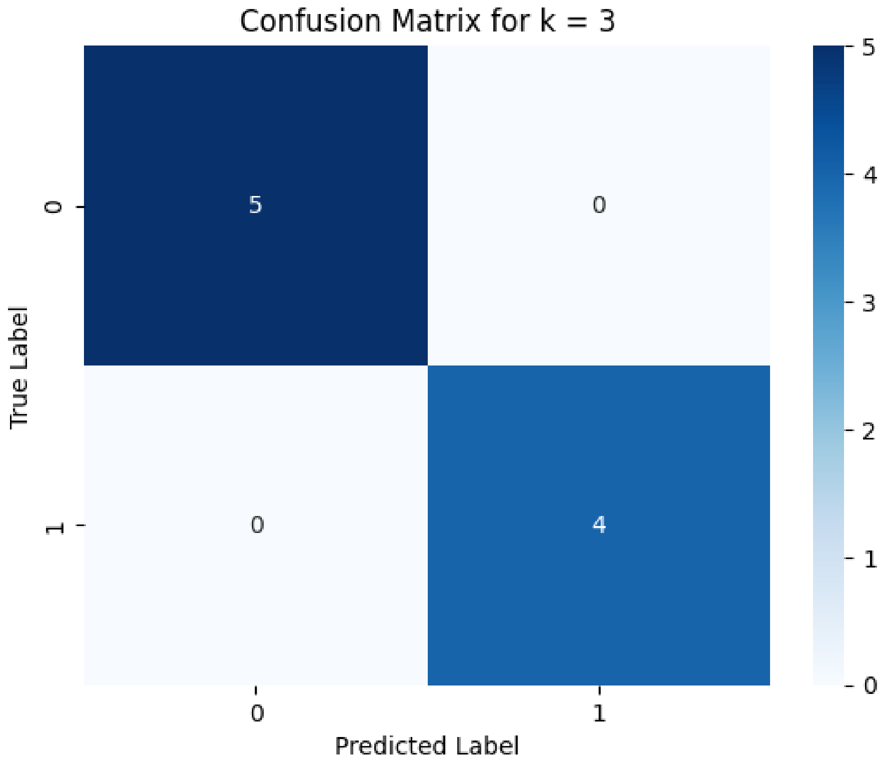
Solution 1:

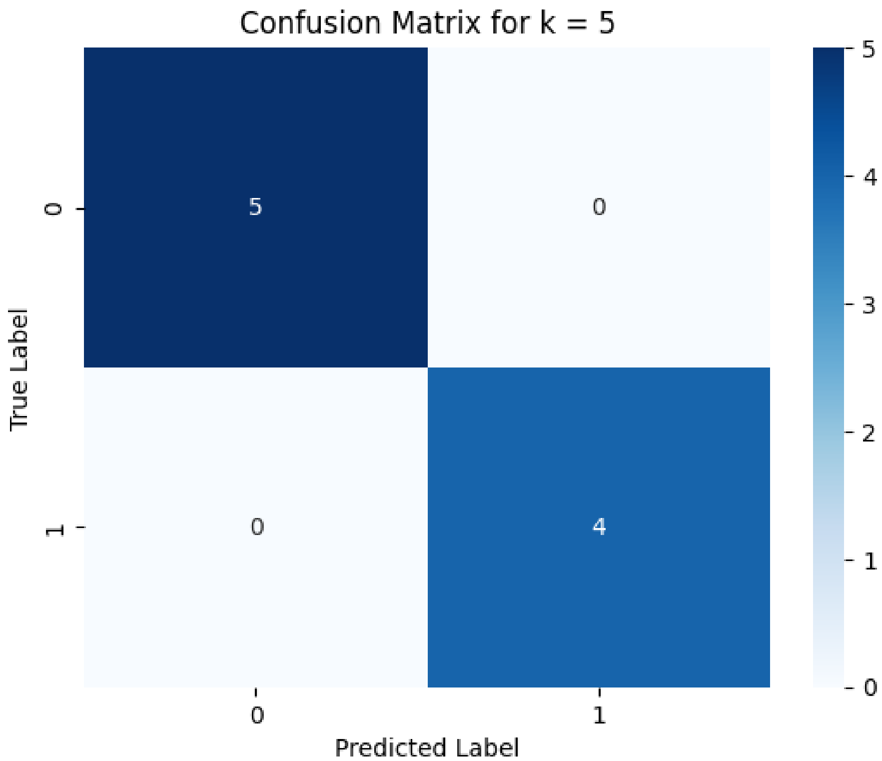
```
# -*- coding: utf-8 -*-
"""prob-1.py

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1Sp9P_8FqwjvNQWMe5biq5010OP1oDWXl
"""

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

# Sample synthetic dataset, 1=Apple, -1=Orange
data_apples = np.array([
```

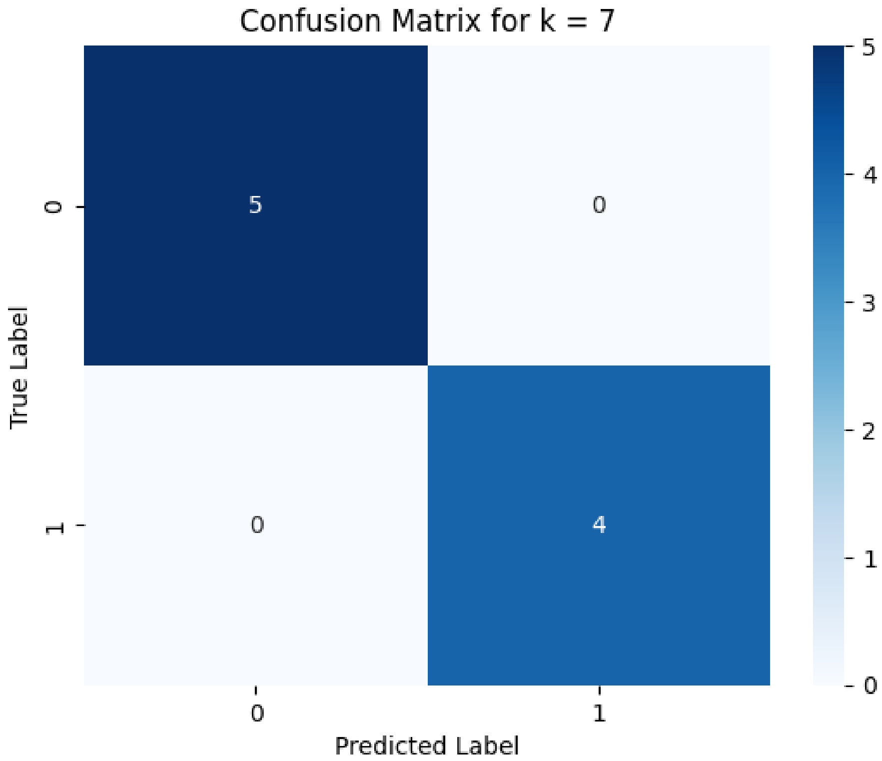
```
    [250, 120, 1],
    [240, 110, 1],
    [245, 130, 1],
    [240, 100, 1],
    [220, 120, 1],
    [230, 125, 1],
    [255, 115, 1],
    [215, 110, 1],
    [210, 130, 1],
    [235, 105, 1],
    [240, 122, 1],
    [245, 125, 1],
    [248, 117, 1],
    [243, 127, 1],
    [225, 112, 1],
    [230, 118, 1],
    [238, 123, 1],
    [222, 115, 1],
    [248, 130, 1],
    [210, 100, 1],
    [255, 121, 1],
    [245, 128, 1],
    [240, 132, 1],
    [235, 120, 1],
    [250, 110, 1],
    [243, 135, 1],
    [227, 125, 1],
    [222, 132, 1],
    [232, 110, 1],
    [240, 105, 1],
    [238, 128, 1],
    [215, 118, 1],
    [255, 112, 1],
    [250, 125, 1],
    [240, 118, 1],
    [223, 130, 1],
    [232, 120, 1],
    [235, 130, 1],
    [245, 112, 1],
    [228, 123, 1],
    [240, 116, 1],
    [220, 128, 1],
    [255, 120, 1],
    [230, 130, 1],
    [225, 122, 1],
    [240, 126, 1],
    [233, 128, 1]
])

data_oranges = np.array([
    [20, 91, -1],
    [30, 95, -1],
    [14, 84, -1],
    [32, 95, -1],
```

```
    [18, 94, -1],
    [23, 90, -1],
    [25, 92, -1],
    [12, 80, -1],
    [29, 87, -1],
    [19, 89, -1],
    [22, 82, -1],
    [30, 88, -1],
    [16, 85, -1],
    [26, 89, -1],
    [28, 86, -1],
    [21, 83, -1],
    [24, 88, -1],
    [15, 81, -1],
    [17, 84, -1],
    [27, 91, -1],
    [23, 84, -1],
    [22, 88, -1],
    [20, 80, -1],
    [25, 87, -1],
    [18, 85, -1],
    [19, 82, -1],
    [28, 83, -1],
    [26, 81, -1],
    [30, 84, -1],
    [29, 82, -1],
    [21, 87, -1],
    [12, 78, -1],
    [27, 84, -1],
    [16, 80, -1],
    [24, 82, -1],
    [14, 88, -1],
    [17, 82, -1],
    [22, 80, -1],
    [23, 86, -1],
    [13, 79, -1],
    [19, 85, -1],
    [21, 89, -1],
    [15, 84, -1],
    [26, 87, -1],
    [25, 85, -1],
    [27, 82, -1],
    [28, 89, -1]
])

# Concatenate apple and orange data
data = np.vstack((data_apples, data_oranges))

# Separate features (redness and weight) and labels (fruit type)
X = data[:, :2]
y = data[:, 2]

# Split dataset into training, testing, and validation sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.2,
```

```
random_state=42)
X_test, X_val, y_test, y_val = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)

# List of k values to try
k_values = [3, 5, 7]

for k in k_values:
    print(f"Testing k = {k}")

    # Create a KNN classifier with the current k value and L1 distance metric
    knn_classifier = KNeighborsClassifier(n_neighbors=k, p=1)

    # Train the classifier on the training data
    knn_classifier.fit(X_train, y_train)

    # Predict the fruit types for the test data
    y_test_pred = knn_classifier.predict(X_test)

    # Create a confusion matrix
    cm = confusion_matrix(y_test, y_test_pred)

    # Plot the confusion matrix
    plt.figure()
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
    plt.title(f"Confusion Matrix for k = {k}")
    plt.xlabel("Predicted Label")
    plt.ylabel("True Label")
    plt.show()

    # Calculate the accuracy of the model on test data
    accuracy_test = accuracy_score(y_test, y_test_pred)
    print("Test Accuracy:", accuracy_test)
    print()
```


Confusion Matrix for k = 3
True Label
Predicted Label
0
1
5
0
0
4
5
4
3
2
1
0


Confusion Matrix for k = 5
True Label
Predicted Label
0
1
5
0
0
4
5
4
3
2
1
0




```
OUTPUT:

Testing k = 3

Test Accuracy: 1.0

Testing k = 5

Test Accuracy: 1.0

Testing k = 7

Test Accuracy: 1.0
```

1.3 Problem-2

2. (Handwritten Digit Classification) Use the provided code for classifying handwritten digits from the MNIST dataset. First, read and understand the code. Then, perform the following tasks:

   1. Modify the code so that it uses L1-distance instead of the default L2-distance (Euclidean).
   2. Determine the value of K that gives better performance.
   3. Report the accuracy achieved with the modified code.
   4. Display the results by showing the image, actual label, and predicted label. Identify some samples where the predicted label is incorrect.

Solution 2:

```python
# -*- coding: utf-8 -*-
"""prob-1b.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1VO0qxSQB624n4QwWHn3-nqJn1KDRPHxX
"""

warnings.simplefilter(action='ignore', category=FutureWarning)
import numpy as np
from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

# Load MNIST dataset
mnist = fetch_openml('mnist_784')
X = mnist.data
y = mnist.target

# Limit to a subset of the data
num_samples = 5000
X_subset = X[:num_samples]
y_subset = y[:num_samples]

# Split dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_subset, y_subset, test_size=0.2, random_state=42)

# Find the optimal value of K using cross-validation
best_accuracy = 0
best_k = 0
k_range = range(1, 6)  # Limiting the search range

for k in k_range:
    knn_classifier = KNeighborsClassifier(n_neighbors=k, p=1)  # p=1 indicates L1-distance
    knn_classifier.fit(X_train, y_train)
    y_pred = knn_classifier.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)

    if accuracy > best_accuracy:
        best_accuracy = accuracy
        best_k = k

print("Best K:", best_k)
print("Best Accuracy:", best_accuracy)

# Train the final classifier with the best K on the entire training data
final_knn_classifier = KNeighborsClassifier(n_neighbors=best_k, p=1)
final_knn_classifier.fit(X_train, y_train)
```

```python

# Predict the labels for the test data
y_pred = final_knn_classifier.predict(X_test)

# Calculate the accuracy of the final model
accuracy = accuracy_score(y_test, y_pred)
print("Final Accuracy:", accuracy)

# Display images, actual labels, and predicted labels for incorrect predictions
incorrect_indices = np.where(y_pred != y_test)[0]

for index in incorrect_indices[:5]:  # Display the first 5 incorrect predictions
    image = X_test[index, :].reshape(28, 28)
    actual_label = y_test[index]
    predicted_label = y_pred[index]

    plt.figure(figsize=(4, 4))
    plt.imshow(image, cmap='gray')
    plt.title(f"Actual: {actual_label}, Predicted: {predicted_label}")
    plt.axis('off')
    plt.show()
```

```
OUTPUT:

Best K: 1
Best Accuracy: 0.939
Final Accuracy: 0.939
```