AML – 64016 – Assignment 1

apeter@kent.edu

GitHub link - https://github.com/Aloysius95/apeter_64061/tree/65d87b8b96d0fc29d268b80e1b72cec8ee1c7e0f/Assignment1

1. Read, run, and understand the Python code in the template project using iris data

Code given to analyze the KNN and its working

```
[24] # import modules for this project
     from sklearn import datasets
     from sklearn.metrics import accuracy_score
     from sklearn.model_selection import train_test_split

     # load iris dataset
     iris = datasets.load_iris()
     data, labels = iris.data, iris.target

     # training testing split
     res = train_test_split(data, labels,
                            train_size=0.8,
                            test_size=0.2,
                            random_state=12)
     train_data, test_data, train_labels, test_labels = res

     # Create and fit a nearest-neighbor classifier
     from sklearn.neighbors import KNeighborsClassifier
     # classifier "out of the box", no parameters
     knn = KNeighborsClassifier()
     knn.fit(train_data, train_labels)

     # print some interested metrics
     print("Predictions from the classifier:")
     learn_data_predicted = knn.predict(train_data)
     print(learn_data_predicted)
     print("Target values:")
     print(train_labels)
     print(accuracy_score(learn_data_predicted, train_labels))
```

```
    # re-do KNN using some specific parameters.
    knn2 = KNeighborsClassifier(algorithm='auto',
                                leaf_size=30,
                                metric='minkowski',
                                p=2,          # p=2 is equivalent to euclidian distance
                                metric_params=None,
                                n_jobs=1,
                                n_neighbors=5,
                                weights='uniform')

    knn.fit(train_data, train_labels)
    test_data_predicted = knn.predict(test_data)
    accuracy_score(test_data_predicted, test_labels)
```

```
Predictions from the classifier:
[0 1 2 0 2 0 1 1 0 1 1 0 0 0 0 0 0 0 2 0 2 1 1 1 0 2 1 1 2 0 2 0 2 1 2 2 1
 1 1 2 2 0 2 2 0 1 0 2 2 0 1 1 0 0 1 1 1 1 2 1 2 0 0 1 1 2 0 2 1 0 2 2 1 2
 2 0 0 2 1 1 2 0 1 1 0 1 1 2 2 1 0 2 0 2 0 0 1 2 2 1 2 2 0 1 1 0 2 2 2 1 2
 2 2 0 0 1 0 2 2 1]
Target values:
[0 1 2 0 2 0 1 1 0 1 1 0 0 0 0 0 0 0 2 0 2 1 1 1 0 2 1 1 2 0 2 0 2 2 2 2 1
 1 1 1 2 0 2 2 0 1 0 2 2 0 1 1 0 0 1 1 1 1 2 1 2 0 0 1 1 1 0 2 1 0 2 2 1 2
 2 0 0 2 1 1 2 0 1 1 0 1 1 2 2 1 0 2 0 2 0 0 1 2 2 1 2 2 0 1 1 0 2 2 2 1 2
 2 2 0 0 1 0 2 2 1]
0.975
0.9666666666666667
```

The professor provided the code to understand the workings of the code.

- Dataset Iris was uploaded to Python and the dataset was divided into training and testing datasets of 80 and 20 percent respectively.
- Initialize KNN with default parameters and train using the training set.
- Redo KNN on testing tests with different parameters.
- Print the accuracy of the models.

2. Replicate the study using a new simulated dataset

- Step 1

```
# Simulated data set provided in the assignment question/reference.
from sklearn.datasets import make_blobs
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import numpy as np

# Define the centers for the blobs
centers = [[2, 4], [6, 6], [1, 9]]
n_classes = len(centers)

# Generate the synthetic dataset
data, labels = make_blobs(n_samples=150,
                          centers=np.array(centers),
                          random_state=1)
```

We created a simulated dataset as per instructions.

- Step 2

```
# Split the data into training and testing sets
res = train_test_split(data, labels,
                       train_size=0.8,
                       test_size=0.2,
                       random_state=12)
train_data, test_data, train_labels, test_labels = res

# Create and fit a nearest-neighbor classifier
knn = KNeighborsClassifier()
knn.fit(train_data, train_labels)

# Predict on the training data
learn_data_predicted = knn.predict(train_data)

# Print metrics
print("Predictions from the classifier:")
print(learn_data_predicted)
print("Target values:")
print(train_labels)
print("Accuracy on training data:")
print(accuracy_score(train_labels, learn_data_predicted))
```

```
Predictions from the classifier:
[0 2 1 0 0 1 1 2 2 0 2 2 2 1 1 0 0 2 1 1 0 0 0 1 1 2 0 0 1 0 1 1 1 0 1 2 0
 1 0 1 2 2 2 0 2 0 2 2 0 0 0 1 2 2 2 2 1 1 0 1 2 1 2 2 2 0 0 0 0 0 0 0 1 1
 2 1 2 1 2 2 1 1 1 0 2 1 2 1 0 1 2 1 0 2 0 1 2 2 0 2 1 0 0 2 1 1 2 2 0 1 1
 1 2 2 2 1 1 2 1 2]
Target values:
[0 2 1 0 0 1 1 2 2 0 2 2 2 1 1 0 0 2 1 1 0 0 0 1 1 2 0 0 1 0 1 1 1 0 1 2 0
 1 0 1 2 2 2 0 2 0 2 2 0 0 0 1 2 2 2 2 1 1 0 1 2 1 2 2 2 0 0 0 0 0 0 0 1 1
 2 1 2 1 2 2 1 1 1 0 2 1 2 1 0 1 2 1 0 2 0 1 2 2 0 2 1 0 0 2 1 1 2 2 0 1 1
 1 2 2 2 1 1 2 1 2]
Accuracy on training data:
1.0
```

Split the dataset into 80/20 training and testing data sets, used similar KNN code to run the training test, and printed the results. The accuracy achieved was 100% for the training set.

- Step 3

```
[15] # re-do KNN using some specific parameters.
     knn2 = KNeighborsClassifier(algorithm='auto',
                                 leaf_size=30,
                                 metric='minkowski',
                                 p=2,          # p=2 is equivalent to Euclidean distance
                                 metric_params=None,
                                 n_jobs=1,
                                 n_neighbors=5,
                                 weights='uniform')

     knn2.fit(train_data, train_labels)
     test_data_predicted = knn2.predict(test_data)
     accuracy_score(test_data_predicted, test_labels)

     1.0
```

Running the KNN code with adjusted parameters, with 2 as the Euclidean distance, and printing the accuracy score for the testing set. The accuracy achieved was 100% for the testing set.
(The simulated dataset is well-distributed, leading to both the testing and training accuracy reaching 100%.)

- Step 4

```
import matplotlib.pyplot as plt # import the matplotlib library

train_acc_knn = accuracy_score(learn_data_predicted, train_labels)
test_acc_knn2 = accuracy_score(test_data_predicted, test_labels)

# Plotting the results to compare the training and testing accuracy
labels = ['Train Accuracy', 'Test Accuracy']
knn_acc = [train_acc_knn, test_acc_knn2]


x = range(len(labels))

plt.figure(figsize=(8, 5))
plt.bar(x, knn_acc, width=0.4, label='KNN', align='center')


plt.xticks(x, labels)
plt.ylabel('Accuracy')
```

Plotting the training and testing accuracy to have a visual graph comparing the results.

Below is the graph output comparing the results of the training and testing set.