

Assignment 1 -
ADVANCED MACHINE LEARNING (BA-64061-001)
vvedula@kent.edu

GitHub: [GitHub_Assignment 1](#)

Part A: Read, run, and understand the Python code in the template project using iris data

```
hw1.ipynb  Cannot save changes
File Edit View Insert Runtime Tools Help
+ Code + Text Copy to Drive
A case study using iris dataset for KNN algorithm

# 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))
```

```
hw1.ipynb  Cannot save changes
File Edit View Insert Runtime Tools Help
+ Code + Text Copy to Drive

# 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')

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

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

Use this command to help with choice of parameters in the KNeighborsClassifier function.

[ ] help(KNeighborsClassifier)
```

Key Takeaways from the code:

- The Iris dataset was loaded into Python and split into two parts: 80% for training and 20% for testing.
- A K-Nearest Neighbors (KNN) model was first initialized with default settings and trained on the training data.
- The model was then fine-tuned by testing different parameter settings on the test data.
- Finally, the accuracy of each model was calculated and displayed to evaluate performance.(0.975)

Part B: Replicate the study using a new simulated dataset.

```
Use the following code to generate an artificial dataset which contain three classes. Conduct a similar KNN analysis to the dataset and report your accuracy.

#Generated dataset which is 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 matplotlib.pyplot as plt
import numpy as np

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

# Synthetic Dataset:
data, labels = make_blobs(n_samples=150,
                          centers=np.array(centers),
                          random_state=1)

# Question: do a 80-20 split of the data
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

# Question: perform a KNN analysis of the simulated data
knn = KNeighborsClassifier()
knn.fit(train_data, train_labels)

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:")
```

```
hw1.ipynb  Cannot save changes
File Edit View Insert Runtime Tools Help

+ Code + Text Copy to Drive
print("Accuracy on training data:")
print(accuracy_score(train_labels, learn_data_predicted))

# re-do KNN using specific parameter:
knn2 = KNeighborsClassifier(algorithm='auto',
                           leaf_size=30,
                           metric='minkowski',
                           p=2,
                           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)

# Question: plot your different results
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))

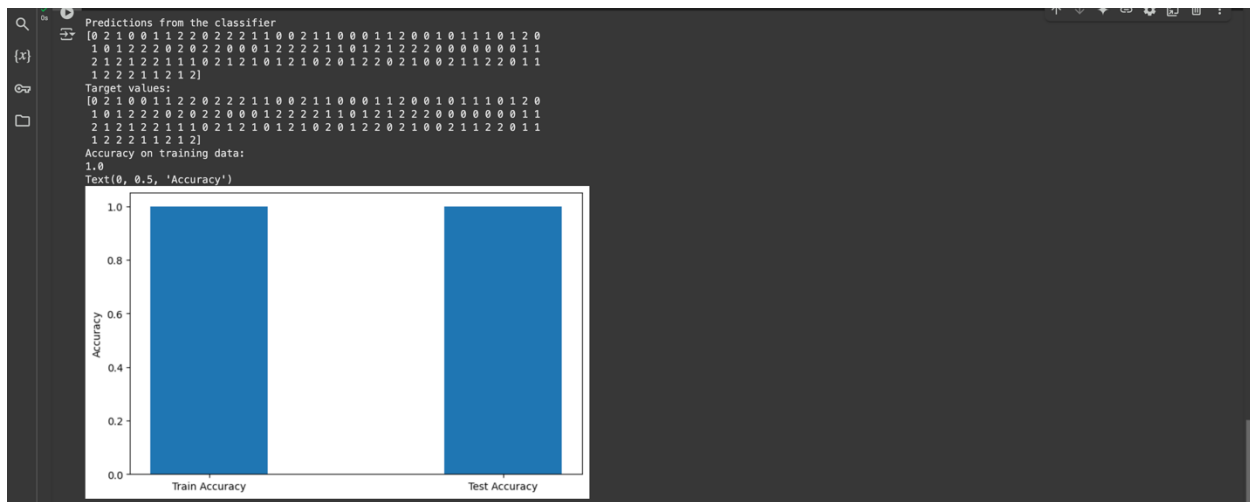
plt.figure(figsize=(8, 5))
plt.bar(x, knn_acc, width=0.4, label='KNN', align='center')

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

Notes:

- We generated a simulated dataset following the given instructions and split it into 80% training and 20% testing data.
- Using a similar K-Nearest Neighbors (KNN) approach, we trained the model on the training set and achieved 100% accuracy.
- The model was then tested with adjusted parameters, setting the Euclidean distance to 2, and the accuracy score for the test set was also 100%.
- The exceptionally high accuracy is attributed to the well-distributed nature of the simulated dataset.

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



(Graph – Output comparing the results of the training and testing set.)