tarcotba-64061

February 4, 2025

Advanced Machine Learning Assignment_1

tarcotba@kent.edu

1. Code given in the assignment for understanding.

[1]:
```
# 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
```

[1]:
```
0.9666666666666667
```

Observations:

From the code it is evident that we are performing classification using KNN method for the mentioned dataset IRIS.

Firstly the data set was uploaded and is split into two i.e., Training and Testing, 80 and 20 respectively.

Training the train dataset with KNN after initializing it with the default parameters.

The same is repeated on the test dataset and the respected results are displayed.

Interpretation of results:

Predictions from the classifier: These are the training data's predicted class labels. Every sample in the training data is assigned to one of the three classes by the classifier (0, 1, or 2).

Target Values: These represent the training data's true class labels, or the real target values. They show which group each sample actually belongs into.

The accuracy value **97.5%** shows that the classifier correctly predicted the class for 97.5% of the samples in the training set.

About **96.67%** of the test set's samples had the class properly predicted by the model.

2. Replicating the study using new simulated data set.

[14]:
```
from sklearn.datasets import make_blobs
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
import numpy as np

centers = [[2, 4], [6, 6], [1, 9]]
n_classes = len(centers)
data, labels = make_blobs(n_samples=150,
                          centers=np.array(centers),
                          random_state=1)

# Splitting the data in train and test 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

# Creating nearest neighbour classifier
knn = KNeighborsClassifier()
knn.fit(train_data, train_labels)

# Train data predictions
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_score(learn_data_predicted, train_labels))
```

```
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]
1.0
```

Observations:

The same process is repeated on the new simulated dataset. i.e., it is split into 80-20 training and

testing data set. Using the same KNN code, full accuracy (**1.0**) is acheived.

[15]:
```
# 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)
```

[15]:
```
1.0
```

The accuracy score for the testing set is printed after the KNN code is run with the parameters changed and the Euclidean distance set to 2. For the testing set, **100%** accuracy was attained. This means that the simulated dataset is **well-distributed.**

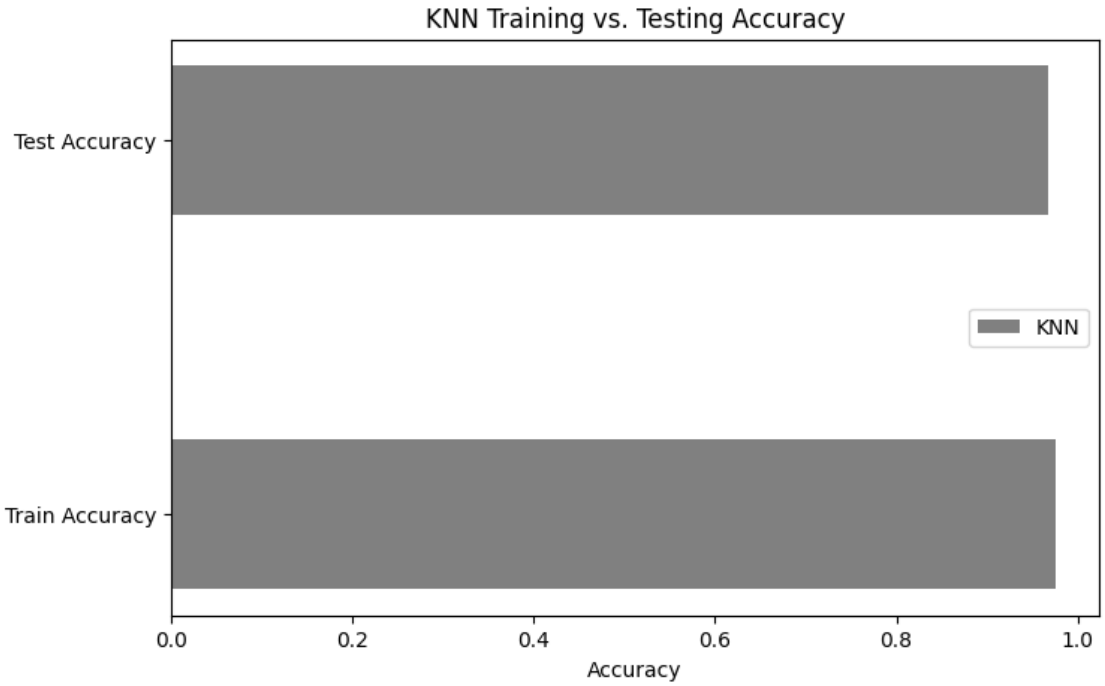
3. The below code shows the visual representation of the train-test accuracy.

[12]:
```
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score

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

labels = ['Train Accuracy', 'Test Accuracy']
knn_acc = [train_acc_knn, test_acc_knn2]

plt.figure(figsize=(8, 5))
plt.barh(labels, knn_acc, height=0.4, label='KNN', align='center',␣
 ↪color="grey")  # barh for horizontal
plt.xlabel('Accuracy')
plt.title('KNN Training vs. Testing Accuracy')
plt.legend()
plt.show()
```


KNN Training vs. Testing Accuracy
Test Accuracy
Train Accuracy
KNN
0.0
0.2
0.4
0.6
0.8
1.0
Accuracy