Guided Projects Artificial Intelligence & Machine Learning

Guided Projects: Supervised Learning

Handwritten Digit Recognition

Name	Lakshmi Thirunavukkarasu
Course	AI and ML (Batch 5)
Problem Statement	Compare the performance of the SVM and MLP classifier for MNIST dataset.

Software requirements prerequisites

Anaconda

Python 3.8

Python Packages

NumPy

Pandas

Scikit

Matplotlib

Steps

1. Load the Dataset

Download the dataset

```
In [2]:
digits = datasets.load_digits()
dir(digits)
```

```
Out[2]: ['DESCR', 'data', 'feature_names', 'frame', 'images', 'target', 'target_names']
```

```
In [3]:
print(digits.images.shape)
print(digits.target.shape)
print(digits.target)
```

```
(1797, 8, 8)
(1797,)
[0 1 2 ... 8 9 8]
```

```
In [4]:
plt.imshow(digits.images[0])
plt.show()
digits.target_names[0]
```

2. Preprocessing

Squeeze the dataset into a vector and apply Normalization technique.

squeezing the data into vector

In [5]:
```
image_reshaped = digits.images.reshape((len(digits.images), -1))
image_reshaped.shape
```

Out[5]: (1797, 64)

In [6]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_scaled = scaler.fit_transform(image_reshaped)
```

In [7]:
```
y = digits.target
y.shape
```

Out[7]: (1797,)

3. Validate whether the data is balanced or not

In [8]:
```
unique, counts = np.unique(digits.target,return_counts = True)
result = np.column_stack((unique, counts))
print(result)
```

```
[[  0 178]
 [  1 182]
 [  2 177]
 [  3 183]
 [  4 181]
 [  5 182]
 [  6 181]
 [  7 179]
 [  8 174]
 [  9 180]]
```

4. Build MLP Classifier

MLP Classifier

In [11]:
```
kf = KFold(n_splits = 5, shuffle = True, random_state = 2)
mlp = MLPClassifier(hidden_layer_sizes=(300,300), activation='relu', solver='adam',
                    random_state=1, max_iter=2000,shuffle=True,momentum=0.9,alpha=0.001, learning_rate = 'adaptive')
acc_score_mlp = []
for k, (train_index, test_index) in enumerate(kf.split(x_scaled)):

    #print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = x_scaled[train_index], x_scaled[test_index]
    y_train, y_test = digits.target[train_index], digits.target[test_index]

    #print(x_train)
    #mlp = MLPClassifier(hidden_layer_sizes=(15,), activation='logistic', alpha=1e-4, solver='sgd', tol=1e-4, random_state=
    mlp.fit(X_train,y_train)
    predictions = mlp.predict(X_test)
    score = accuracy_score(y_test, predictions)
    acc_score_mlp.append(score)
    print(accuracy_score(y_test, predictions))
    print(classification_report(y_test,predictions))
    print(confusion_matrix(y_test,predictions))
```

5. Build SVM (Linear Classifier)

SVC LInear Model

In [12]:
```
kf = KFold(n_splits = 5, shuffle = True, random_state = 2)
svc_linear = SVC(kernel='linear')
acc_score = []

acc_score_svc_linear = []

for k, (train_index, test_index) in enumerate(kf.split(x_scaled)):

    #print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = x_scaled[train_index], x_scaled[test_index]
    y_train, y_test = digits.target[train_index], digits.target[test_index]

    #print(x_train)
    #mlp = MLPClassifier(hidden_layer_sizes=(15,), activation='logistic', alpha=1e-4, solver='sgd', tol


    svc_linear.fit(X_train,y_train)
    predictions = svc_linear.predict(X_test)
    score = accuracy_score(y_test, predictions)
    acc_score_svc_linear.append(score)
    print(score)
    print(classification_report(y_test,predictions))
```

6. Build SVM (RBF Kernel)

SVC - RBF Kernel

In [13]:
```
kf = KFold(n_splits = 5, shuffle = True, random_state = 2)
svc_rbf = SVC(kernel='rbf')
acc_score_svc_rbf = []

for k, (train_index, test_index) in enumerate(kf.split(x_scaled)):

    #print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = x_scaled[train_index], x_scaled[test_index]
    y_train, y_test = digits.target[train_index], digits.target[test_index]

    #print(x_train)
    #mlp = MLPClassifier(hidden_layer_sizes=(15,), activation='logistic', alpha=1e-4, sol
    svc_rbf.fit(X_train,y_train)
    predictions = svc_rbf.predict(X_test)
    score = accuracy_score(y_test, predictions)
    acc_score_svc_rbf.append(score)
    print(score)
```

7. Mean Performance Report for MLP and SVM classifiers

In [14]:
```
print("Total SVC(RBF Kernel) model accuracy: " , np.mean(acc_score_svc_rbf))
print("Total SVC(Linear) model accuracy: " , np.mean(acc_score_svc_linear))
print("Total MLP model accuracy: " , np.mean(acc_score_mlp))
```

```
Total SVC(RBF Kernel) model accuracy:  0.981641906530486
Total SVC(Linear) model accuracy:  0.9799736923553081
Total MLP model accuracy:  0.9816434540389972
```