fsum0np6d

February 14, 2025

[487]:

```python
import pandas as  pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
```

[488]:

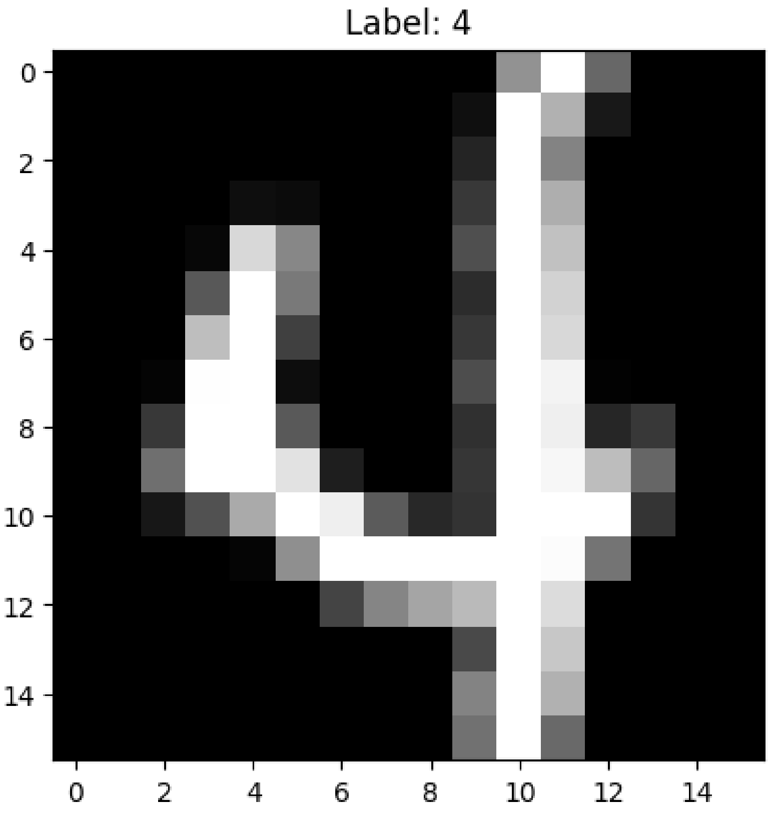
```python
path = "/content/usps.h5"
```

[489]:

```python
import h5py
with h5py.File(path, 'r') as hf:
        train = hf.get('train')
        X_train = train.get('data')[:]
        y_train = train.get('target')[:]
        test = hf.get('test')
        X_test = test.get('data')[:]
        y_test = test.get('target')[:]
```

[490]:

```python
plt.imshow(X_train[2].reshape(16,16), cmap='gray')
plt.title(f'Label: {y_train[2]}')
plt.show()
```



[491]:
```
print("Shape of the Train data features")
print(X_train.shape)
print("Shape of the Train data target")
print(y_train.shape)
print("Shape of the Test data features")
print(X_test.shape)
print("Shape of the Test data target")

print(y_test.shape)
```

```
Shape of the Train data features
(7291, 256)
Shape of the Train data target
(7291,)
Shape of the Test data features
(2007, 256)
Shape of the Test data target
(2007,)
```

[492]:
```
print(np.unique(y_train))
```

```
[0 1 2 3 4 5 6 7 8 9]
```

Using PCA and Standard

[493]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=50)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
```

[494]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

[494]:
```

```

KNN Models

[495]:
```
from sklearn.model_selection import KFold
from sklearn.neighbors import KNeighborsClassifier
```

[496]:
```
from sklearn.model_selection import cross_val_score
clf = KNeighborsClassifier(n_neighbors=3)
scores = cross_val_score(clf, X_train, y_train, cv=10)
```

[497]:
```
print("Train accuracy: ", scores.mean())
```

```
Train accuracy:  0.9729802506717778
```

[498]:
```
clf.fit(X_train, y_train)
```

[498]:
```
KNeighborsClassifier(n_neighbors=3)
```

[499]:
```
# Evaluate on test set
from sklearn.metrics import accuracy_score
y_pred = clf.predict(X_test)
test_accuracy = accuracy_score(y_test, y_pred)
```

[500]:
```
print("Test accuracy :",test_accuracy)
```

```
Test accuracy : 0.9486796213253612
```

[501]:
```
from sklearn.metrics import confusion_matrix, classification_report
```

[502]:
```
print(confusion_matrix(y_true=y_test, y_pred= y_pred))
```

```
[[355   0   3   0   0   0   0   0   0   1]
 [  0 257   0   0   4   0   2   1   0   0]
 [  7   0 185   1   1   0   0   1   3   0]
 [  3   0   1 153   0   6   0   0   1   2]
 [  0   1   2   0 183   1   2   2   1   8]
 [  3   1   2   2   0 148   0   0   1   3]
 [  3   1   1   0   2   0 163   0   0   0]
 [  0   1   1   1   3   0   0 140   0   1]
 [  4   0   2   1   0   2   1   1 153   2]
 [  1   1   1   0   3   0   0   3   1 167]]
```

[503]:

```
from sklearn.metrics import precision_score, recall_score

print("Precison score")
print(precision_score(y_true=y_test, y_pred=y_pred, average="macro"))
print("Recall score")
print(recall_score(y_true=y_test, y_pred=y_pred, average="macro"))
```

```
Precison score
0.9484154196887384
Recall score
0.9434767022174325
```

[504]:

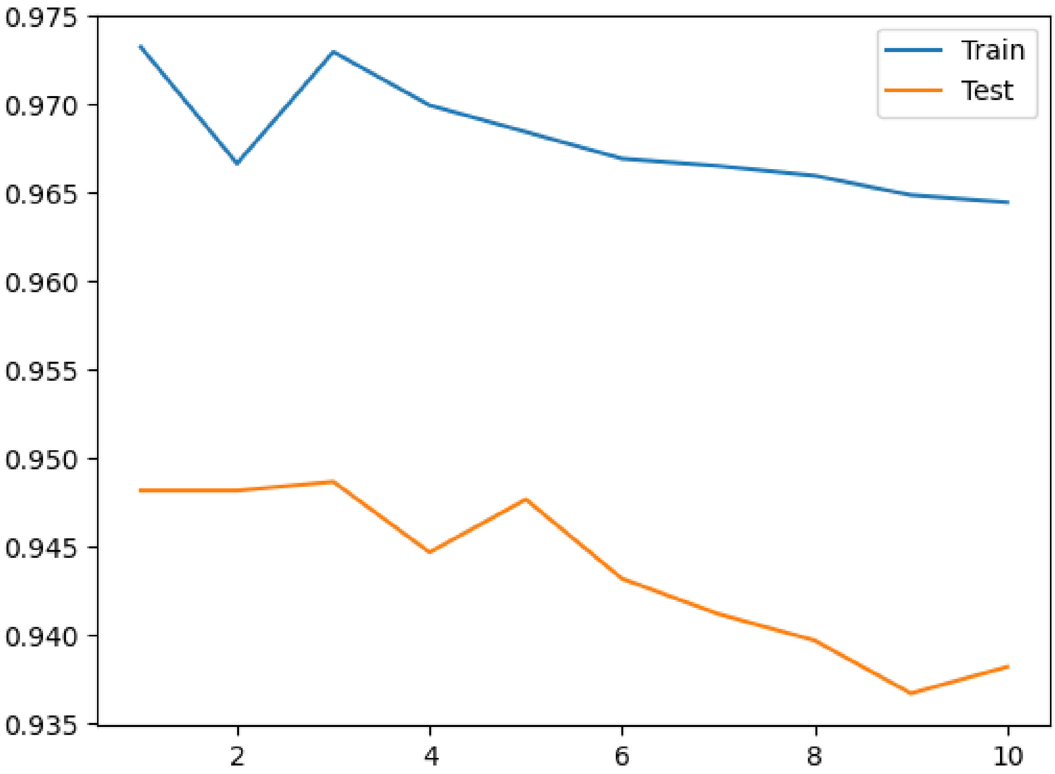
```
train_scores = []
test_scores = []

for i in range(1,11):
  print("For k value ", i, " :")
  clf = KNeighborsClassifier(n_neighbors=i)
  scores = cross_val_score(clf, X_train, y_train, cv=10)
  score = scores.mean()
  test_score = clf.fit(X_train, y_train).score(X_test, y_test)
  train_scores.append(score)
  test_scores.append(test_score)
  print(f"Neighbors: {i}, Train accuracy: {score:.4f}, Test accuracy:␣
 ↪{test_score:.4f}")
  print("Confusion matrix :")
  print(confusion_matrix(y_true=y_test, y_pred= clf.predict(X_test)))
  print("Precision score : ")
  print(precision_score(y_true=y_test, y_pred=clf.predict(X_test),␣
 ↪average="macro"))
  print("Recall score : ")
  print(recall_score(y_true=y_test, y_pred=clf.predict(X_test),␣
 ↪average="macro"))
  print("----------------------------")
```

```
For k value  1  :
Neighbors: 1, Train accuracy: 0.9733, Test accuracy: 0.9482
Confusion matrix :
[[355   0   2   0   0   0   0   1   0   1]
 [  0 255   0   0   6   0   2   1   0   0]
 [  4   0 186   2   1   0   0   1   4   0]
 [  2   0   1 153   0   9   0   0   0   1]
 [  0   2   0   0 184   2   2   2   1   7]
 [  2   1   2   2   0 149   0   0   3   1]
 [  0   0   1   0   2   4 163   0   0   0]
 [  0   1   1   1   4   0   0 136   2   2]
 [  2   0   2   5   0   2   0   0 151   4]
 [  0   0   1   0   1   0   0   3   1 171]]
Precision score :
0.9438701530347882
Recall score :
0.9426831562580743
----------------------------
For k value  2  :
Neighbors: 2, Train accuracy: 0.9667, Test accuracy: 0.9482
Confusion matrix :
[[355   0   3   0   0   0   0   0   0   1]
 [  0 260   0   0   3   0   1   0   0   0]
 [  9   0 185   1   1   0   0   1   1   0]
 [  3   0   1 155   0   6   0   0   0   1]
 [  1   2   2   0 187   1   0   2   0   5]
 [  3   1   2   8   0 146   0   0   0   0]
 [  3   0   1   0   2   3 161   0   0   0]
 [  0   2   1   1   4   0   0 139   0   0]
 [  4   0   4   3   0   2   0   1 150   2]
 [  1   1   1   0   4   1   0   4   0 165]]
Precision score :
0.9491384108099208
Recall score :
0.9417739700153381
----------------------------
For k value  3  :
Neighbors: 3, Train accuracy: 0.9730, Test accuracy: 0.9487
Confusion matrix :
[[355   0   3   0   0   0   0   0   0   1]
 [  0 257   0   0   4   0   2   1   0   0]
 [  7   0 185   1   1   0   0   1   3   0]
 [  3   0   1 153   0   6   0   0   1   2]
 [  0   1   2   0 183   1   2   2   1   8]
 [  3   1   2   2   0 148   0   0   1   3]
 [  3   1   1   0   2   0 163   0   0   0]
 [  0   1   1   1   3   0   0 140   0   1]
 [  4   0   2   1   0   2   1   1 153   2]
 [  1   1   1   0   3   0   0   3   1 167]]
Precision score :
0.9484154196887384
Recall score :
0.9434767022174325
```

```
----------------------------
For k value  4  :
Neighbors: 4, Train accuracy: 0.9700, Test accuracy: 0.9447
Confusion matrix :
[[355   0   3   0   0   0   0   0   0   1]
 [  0 258   0   0   4   0   2   0   0   0]
 [  6   0 183   1   1   0   2   2   3   0]
 [  2   0   2 155   0   4   0   1   0   2]
 [  0   3   3   0 185   1   2   2   0   4]
 [  5   0   2   4   0 144   0   0   1   4]
 [  3   1   2   0   2   0 162   0   0   0]
 [  0   1   1   1   4   1   0 138   0   1]
 [  5   0   1   3   0   3   2   1 149   2]
 [  1   1   1   0   2   1   0   4   0 167]]
Precision score :
0.9439571786232257
Recall score :
0.9381917902972061
----------------------------
For k value  5  :
Neighbors: 5, Train accuracy: 0.9685, Test accuracy: 0.9477
Confusion matrix :
[[355   0   2   0   1   0   0   0   0   1]
 [  0 258   0   0   4   0   2   0   0   0]
 [  6   0 183   1   1   0   1   2   4   0]
 [  2   0   2 154   0   5   0   1   0   2]
 [  0   3   2   0 184   1   2   2   0   6]
 [  4   0   2   2   0 147   0   0   1   4]
 [  3   0   2   0   2   0 163   0   0   0]
 [  0   1   1   1   4   1   0 137   0   2]
 [  2   0   1   4   0   3   1   1 152   2]
 [  1   0   1   0   1   1   0   4   0 169]]
Precision score :
0.9458380951546841
Recall score :
0.9418095162624136
----------------------------
For k value  6  :
Neighbors: 6, Train accuracy: 0.9669, Test accuracy: 0.9432
Confusion matrix :
[[355   0   2   0   1   0   0   0   0   1]
 [  0 258   0   0   4   0   2   0   0   0]
 [  6   0 183   1   1   0   1   2   4   0]
 [  2   0   2 154   0   5   0   1   0   2]
 [  0   3   4   0 184   0   2   2   0   5]
 [  4   0   2   4   0 144   0   0   1   5]
 [  3   0   3   0   2   2 160   0   0   0]
 [  0   1   1   1   4   1   0 139   0   0]
 [  4   2   1   2   0   4   2   2 148   1]
 [  1   0   1   0   2   1   0   4   0 168]]
Precision score :
0.9415792732878817
Recall score :
0.9365557442921183
----------------------------
For k value  7  :
Neighbors: 7, Train accuracy: 0.9665, Test accuracy: 0.9412
Confusion matrix :
[[355   0   2   0   1   0   0   0   0   1]
 [  0 258   0   0   4   0   2   0   0   0]
 [  7   0 182   1   1   0   1   2   4   0]
 [  3   0   1 154   0   5   0   1   0   2]
 [  0   3   4   0 182   0   2   2   0   7]
 [  6   0   2   2   0 144   0   0   1   5]
 [  3   1   2   0   3   2 159   0   0   0]
 [  0   1   1   1   4   1   0 138   0   1]
 [  4   1   1   3   0   5   0   0 149   3]
 [  1   0   0   0   2   1   0   4   1 168]]
Precision score :
0.9407027284393328
Recall score :
0.9343845960226608
----------------------------
For k value  8  :
Neighbors: 8, Train accuracy: 0.9660, Test accuracy: 0.9397
Confusion matrix :
[[355   0   2   0   1   0   0   0   0   1]
 [  0 258   0   0   4   0   2   0   0   0]
 [  7   0 182   1   1   0   2   2   3   0]
 [  2   0   2 154   0   5   0   1   0   2]
 [  0   3   4   0 184   0   2   2   0   5]
 [  6   0   3   4   0 141   0   0   1   5]
 [  3   0   3   0   2   2 159   0   1   0]
 [  0   1   1   1   4   1   0 137   1   1]
 [  4   2   0   3   0   5   1   0 148   3]
 [  1   0   0   0   2   1   0   4   1 168]]
Precision score :
0.9383173966638152
Recall score :
0.9322269142752629
----------------------------
For k value  9  :
Neighbors: 9, Train accuracy: 0.9649, Test accuracy: 0.9367
Confusion matrix :
[[354   0   2   0   1   0   1   0   0   1]
 [  0 257   0   0   4   0   3   0   0   0]
 [  7   0 181   1   1   0   2   2   4   0]
 [  3   0   1 154   0   6   0   1   0   1]
 [  0   3   4   0 181   0   2   2   0   8]
 [  5   0   3   4   0 143   0   0   1   4]
 [  3   0   3   0   2   2 159   0   1   0]
 [  0   1   1   1   4   1   0 136   1   2]
 [  5   2   0   5   0   3   1   0 147   3]
 [  1   0   0   0   2   1   0   4   1 168]]
Precision score :
0.9351246929315924
Recall score :
0.9295318426119934
----------------------------
For k value  10  :
Neighbors: 10, Train accuracy: 0.9645, Test accuracy: 0.9382
Confusion matrix :
[[354   0   2   0   1   0   1   0   0   1]
 [  0 257   0   0   4   0   3   0   0   0]
 [  7   0 182   1   1   0   2   2   3   0]
 [  3   0   2 154   0   5   0   1   0   1]
 [  0   3   5   0 181   0   2   2   0   7]
 [  4   0   2   5   0 144   0   0   1   4]
 [  3   0   3   0   2   2 159   0   1   0]
 [  0   2   1   1   3   1   0 137   1   1]
 [  4   2   0   4   0   5   1   2 147   1]
 [  1   0   0   0   1   1   0   5   1 168]]
Precision score :
0.935985155090839
Recall score :
0.9313421652258874
----------------------------
```

[505]:

```
plt.plot(range(1,11), train_scores, label="Train")
plt.plot(range(1,11), test_scores, label="Test")
plt.legend()
plt.show()
```



[505]:

```

```

Naive Bayes Model

[506]:

```python
from scipy.ndimage.interpolation import shift
from sklearn.metrics import accuracy_score
```

```
<ipython-input-506-7b5c807e8aaf>:1: DeprecationWarning: Please import `shift`
from the `scipy.ndimage` namespace; the `scipy.ndimage.interpolation` namespace
is deprecated and will be removed in SciPy 2.0.0.
  from scipy.ndimage.interpolation import shift
```

[506]:

```

```

[507]:

```python
from sklearn.naive_bayes import GaussianNB

nb = GaussianNB()

scores  = cross_val_score(nb, X_train, y_train, cv=10)
print("Train accuracy: ", scores.mean())
nb.fit(X_train, y_train)
```

```python
# Predict on test set
y_pred_nb = nb.predict(X_test)

# Compute metrics
accuracy_nb = accuracy_score(y_test, y_pred_nb)
precision_nb = precision_score(y_test, y_pred_nb, average='macro')
recall_nb = recall_score(y_test, y_pred_nb, average='macro')

# Print results
print(f"Naïve Bayes Test Accuracy: {accuracy_nb:.4f}")
print(f"Naïve Bayes Test Precision: {precision_nb:.4f}")
print(f"Naïve Bayes Test Recall: {recall_nb:.4f}")

print("Confusion matrix naive bayes")
print(confusion_matrix(y_true=y_test, y_pred= y_pred_nb))
```

```
Train accuracy:  0.9009720578010786
Naïve Bayes Test Accuracy: 0.8670
Naïve Bayes Test Precision: 0.8603
Naïve Bayes Test Recall: 0.8568
Confusion matrix naive bayes
[[338   0   6   1   2   6   6   0   0   0]
 [  1 233   0   2   5   3   9   1   3   7]
 [  1   0 171   5   8   1   2   2   8   0]
 [  1   0   5 138   1  16   1   1   2   1]
 [  1   1   9   0 177   1   1   1   0   9]
 [  3   0   2  10   4 136   0   0   2   3]
 [  4   0   8   0   5  13 139   0   1   0]
 [  0   0   2   0   8   3   0 124   2   8]
 [  2   0   6  10   5  12   1   0 128   2]
 [  2   0   2   1   9   2   0   2   3 156]]
```

[507]:

[507]:

[507]: