classify-handwritten-digits

June 28, 2024

[ ]:
```
from sklearn.datasets import fetch_openml

# Load the MNIST dataset
mnist = fetch_openml('mnist_784', version=1)
X, y = mnist["data"], mnist["target"]
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/datasets/_openml.py:968:
FutureWarning: The default value of `parser` will change from `'liac-arff'` to
`'auto'` in 1.4. You can set `parser='auto'` to silence this warning. Therefore,
an `ImportError` will be raised from 1.4 if the dataset is dense and pandas is
not installed. Note that the pandas parser may return different data types. See
the Notes Section in fetch_openml's API doc for details.
  warn(
```

[ ]:
```
from sklearn.model_selection import train_test_split

# Normalize pixel values to a range of 0 to 1
X = X / 255.0

# Split the dataset into training (60,000 images) and testing sets (10,000␣
  ↪images)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=10000,␣
  ↪random_state=42, stratify=y)
```

[ ]:
```
from sklearn.neighbors import KNeighborsClassifier

# Choose the value of k
k = 3
knn = KNeighborsClassifier(n_neighbors=k)
```

[ ]:
```
# Train the KNN classifier
knn.fit(X_train, y_train)
```

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

[ ]:
```
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
```

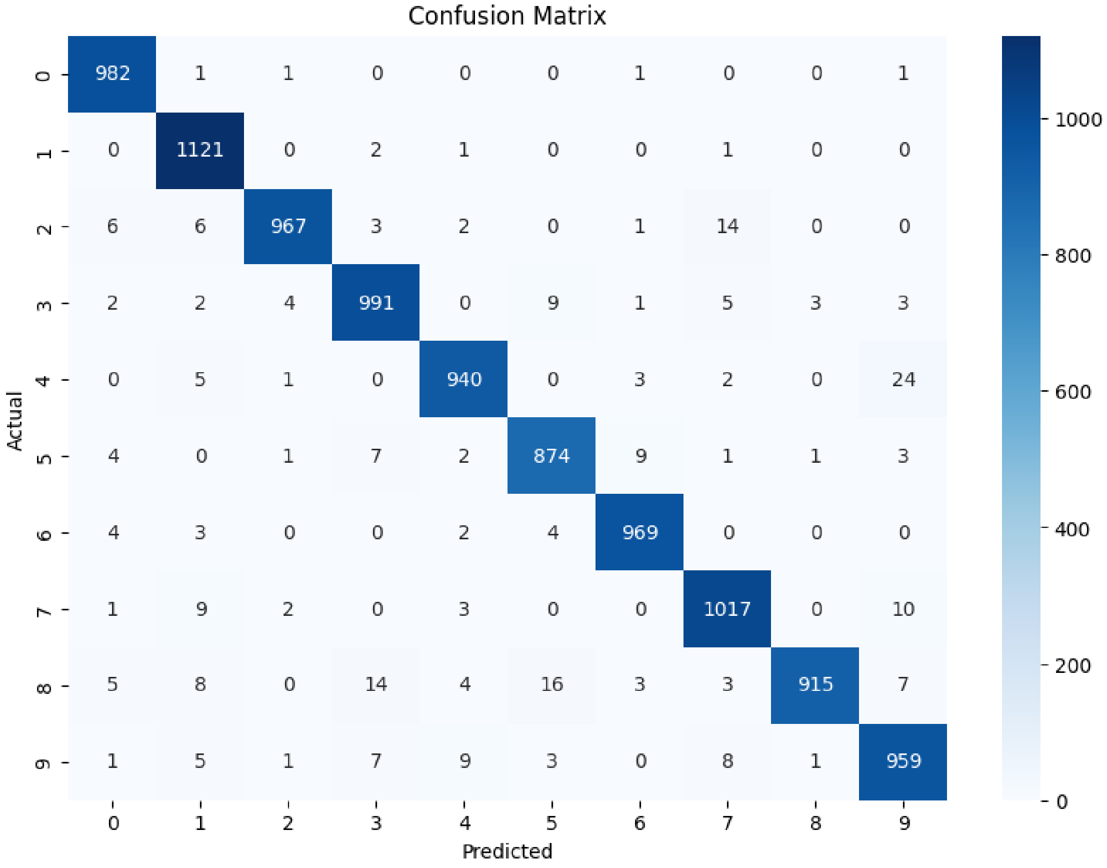
```python
# Predict the labels for the test set
y_pred = knn.predict(X_test)

# Calculate the accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

# Generate the confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Plot the confusion matrix
plt.figure(figsize=(10, 7))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues")
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
Accuracy: 0.97
```



Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	982	1	1	0	0	0	1	0	0	1
1	0	1121	0	2	1	0	0	1	0	0
2	6	6	967	3	2	0	1	14	0	0
3	2	2	4	991	0	9	1	5	3	3
4	0	5	1	0	940	0	3	2	0	24
5	4	0	1	7	2	874	9	1	1	3
6	4	3	0	0	2	4	969	0	0	0
7	1	9	2	0	3	0	0	1017	0	10
8	5	8	0	14	4	16	3	3	915	7
9	1	5	1	7	9	3	0	8	1	959

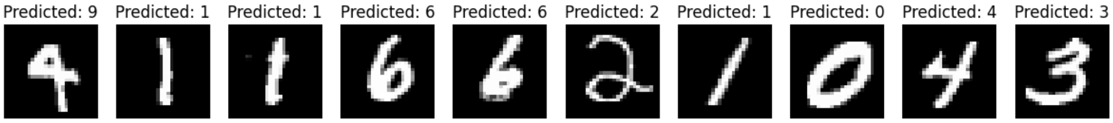
[ ]:

```
import numpy as np

# Select a few test images
num_images = 10
indices = np.random.choice(len(X_test), num_images, replace=False)
selected_images = X_test.iloc[indices].values # Access rows by index and␣
  ↪convert to NumPy array
selected_labels = y_test.iloc[indices].values # Access rows by index and␣
  ↪convert to NumPy array
predicted_labels = knn.predict(selected_images)

# Plot the images with their predicted labels
plt.figure(figsize=(15, 3))
for i in range(num_images):
    plt.subplot(1, num_images, i + 1)
    plt.imshow(selected_images[i].reshape(28, 28), cmap='gray')
    plt.title(f'Predicted: {predicted_labels[i]}')
    plt.axis('off')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but KNeighborsClassifier was fitted with feature
names
  warnings.warn(
```


Predicted: 9
Predicted: 1
Predicted: 1
Predicted: 6
Predicted: 6
Predicted: 2
Predicted: 1
Predicted: 0
Predicted: 4
Predicted: 3