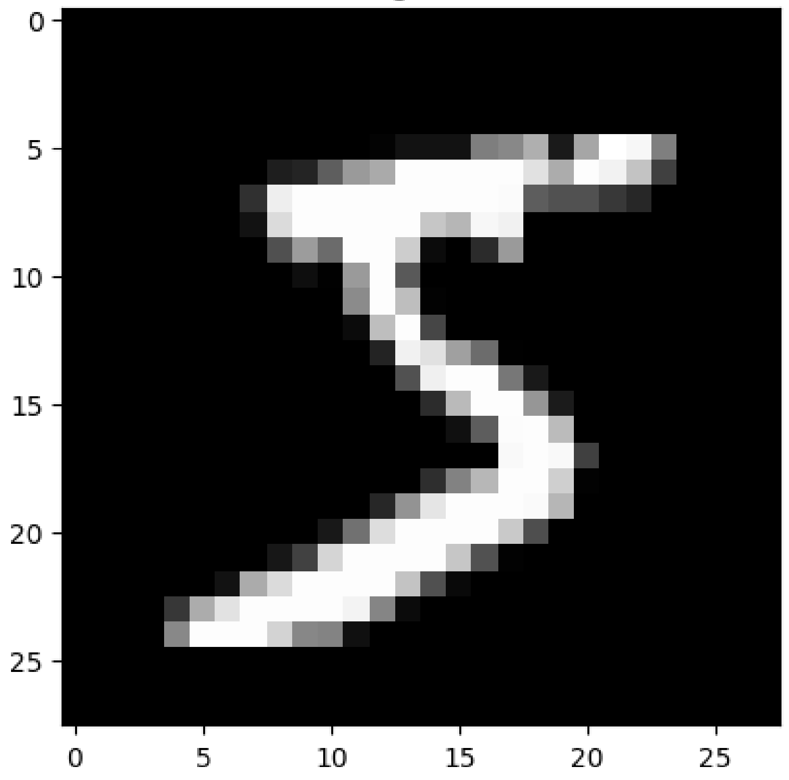
```python
# ===============================================
# MNIST Image Processing & Analysis Assignment
# Framework: Google Colab
# Dataset: MNIST Handwritten Digits
# Algorithms: KMeans Clustering (unsupervised), Logistic Regression
(supervised)
# ===============================================

# ----------------------------
# 1 Install / Import Libraries
# ----------------------------
# Google Colab usually has these pre-installed
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.preprocessing import StandardScaler
import seaborn as sns
import tensorflow as tf

# ----------------------------
# 2 Load MNIST Dataset
# ----------------------------
# MNIST comes preloaded with TensorFlow
(x_train, y_train), (x_test, y_test) =
tf.keras.datasets.mnist.load_data()

print("Training set shape:", x_train.shape)
print("Test set shape:", x_test.shape)
```

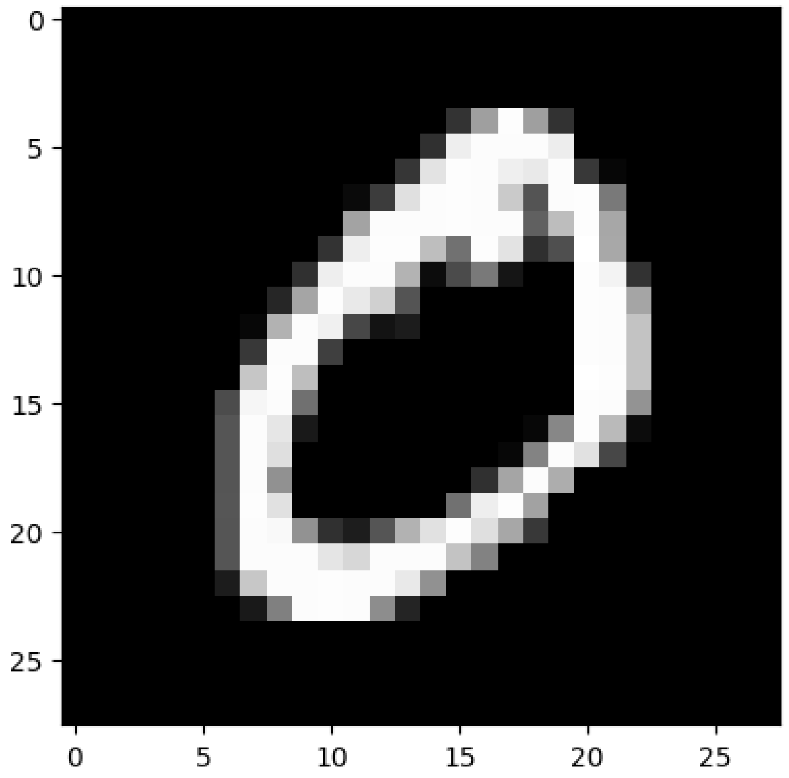
```
Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Training set shape: (60000, 28, 28)
Test set shape: (10000, 28, 28)
```

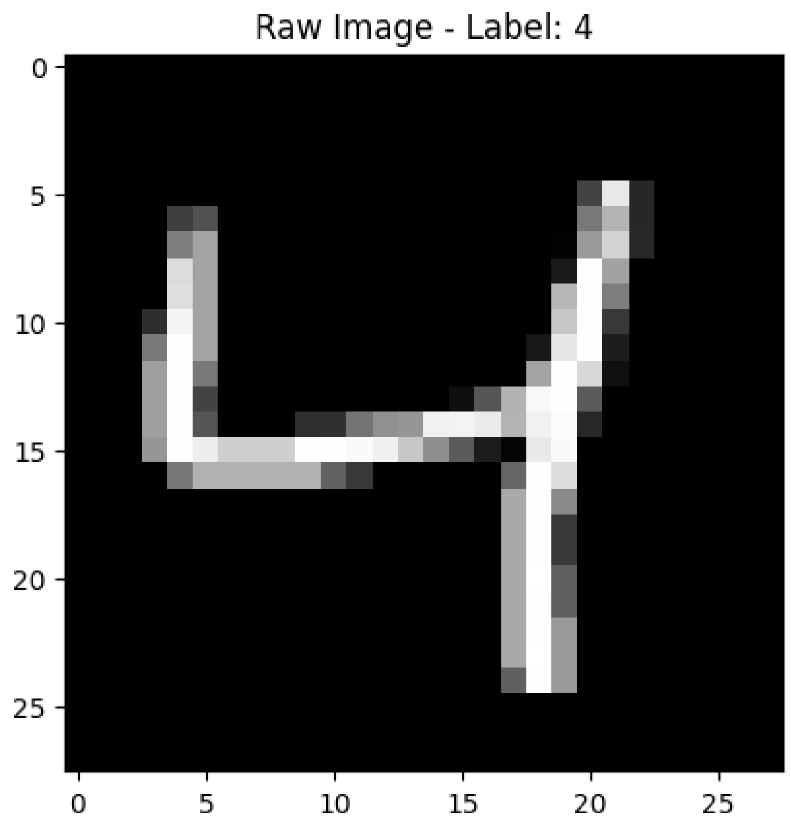
```python
# ----------------------------
# 3 Show Example of Raw Data
# ----------------------------
# Display first 3 raw images
for i in range(3):
    plt.imshow(x_train[i], cmap='gray')
    plt.title(f"Raw Image - Label: {y_train[i]}")
    plt.show()
```

Raw Image - Label: 5



Raw Image - Label: 0





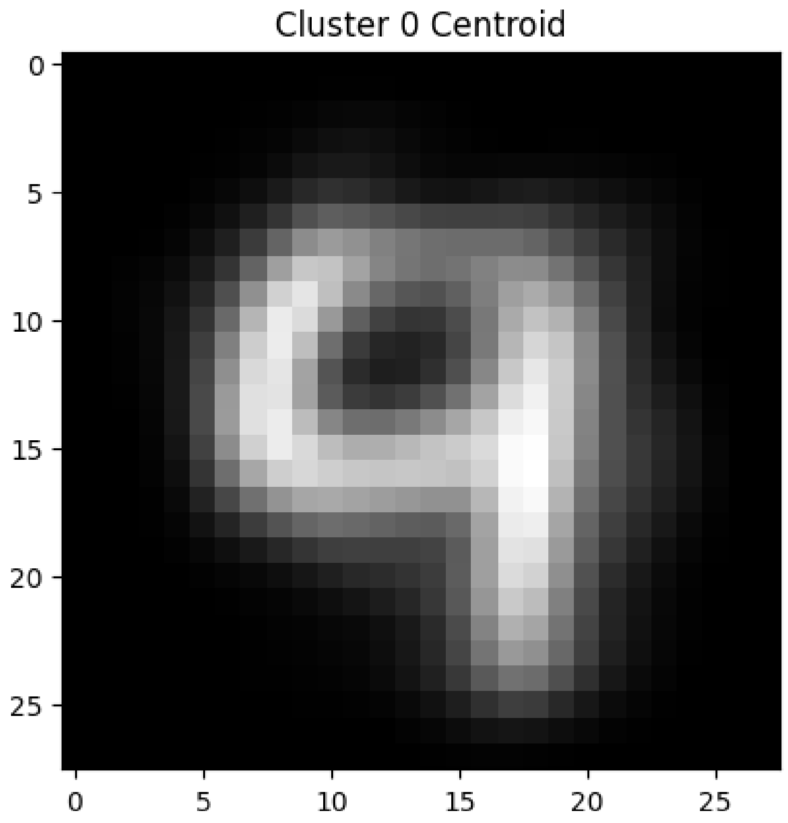
```
# ------------------------------
# 4 Preprocessing (Handling Unstructured Data)
# ------------------------------
# Normalize pixel values to 0-1
x_train_norm = x_train / 255.0
x_test_norm = x_test / 255.0

# Flatten images for ML algorithms (28x28 -> 784 features)
x_train_flat = x_train_norm.reshape(len(x_train_norm), -1)
x_test_flat = x_test_norm.reshape(len(x_test_norm), -1)

print("Example of processed vector (first 10 features):")
print(x_train_flat[0][:10])
```

```
Example of processed vector (first 10 features):
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

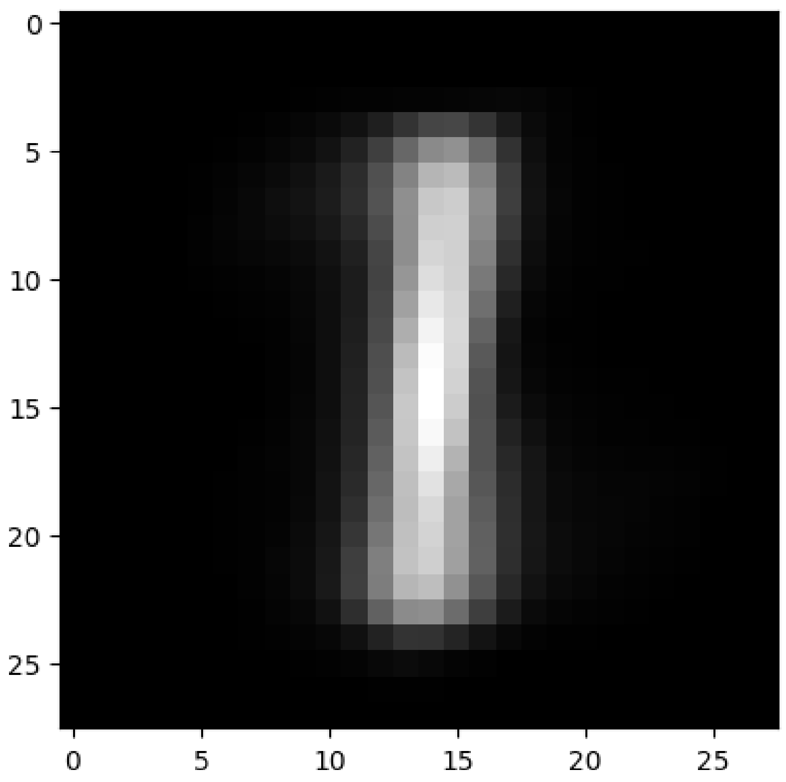
```
# ------------------------------
# 5 Unsupervised Learning: KMeans Clustering
# ------------------------------
# Cluster MNIST images into 10 clusters (digits 0-9)
kmeans = KMeans(n_clusters=10, random_state=42)
kmeans.fit(x_train_flat)
```

```
KMeans(n_clusters=10, random_state=42)
```

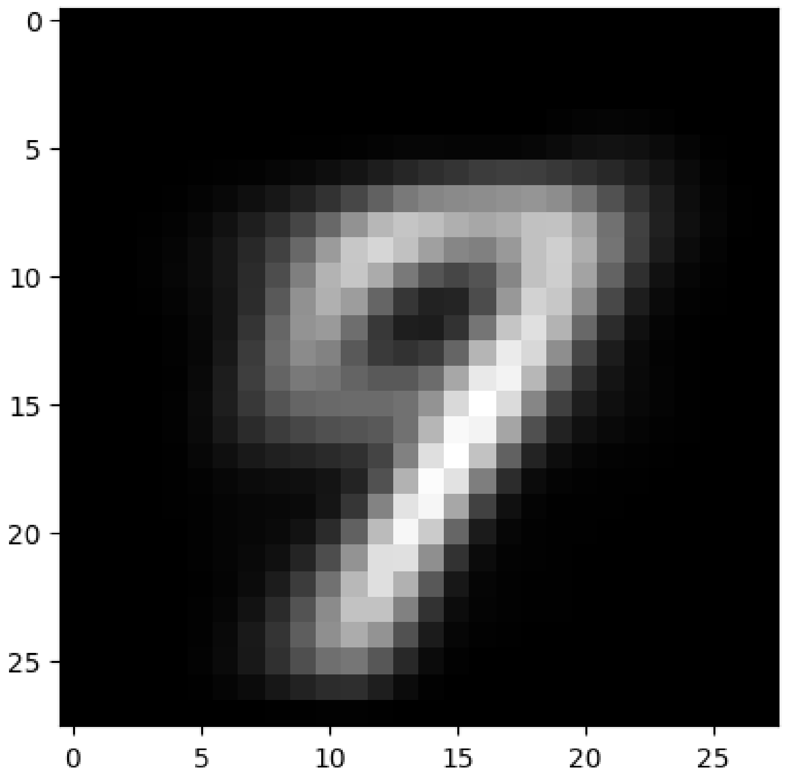
```
# Cluster assignments
clusters = kmeans.labels_

# ----------------------------
# 5a Show Cluster Centroids as Images
# ----------------------------
for i in range(10):
    centroid = kmeans.cluster_centers_[i].reshape(28,28)
    plt.imshow(centroid, cmap='gray')
    plt.title(f"Cluster {i} Centroid")
    plt.show()
```



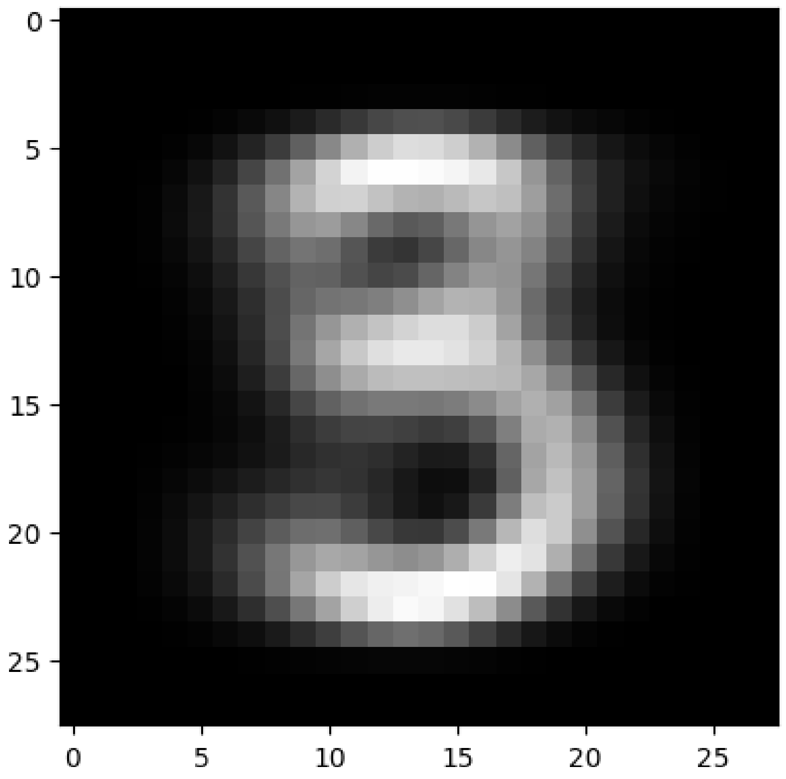
Cluster 1 Centroid



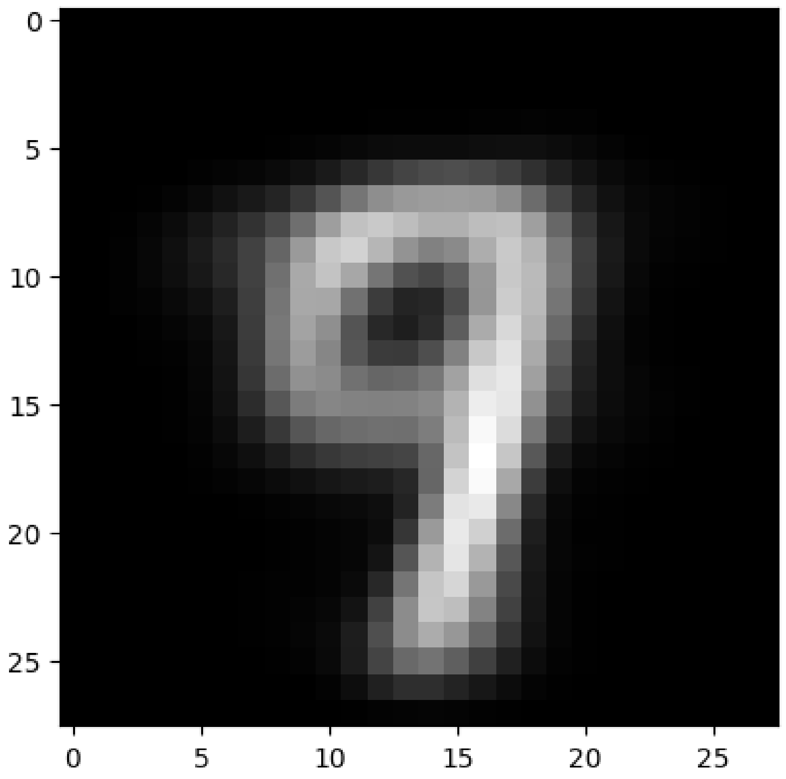
Cluster 2 Centroid



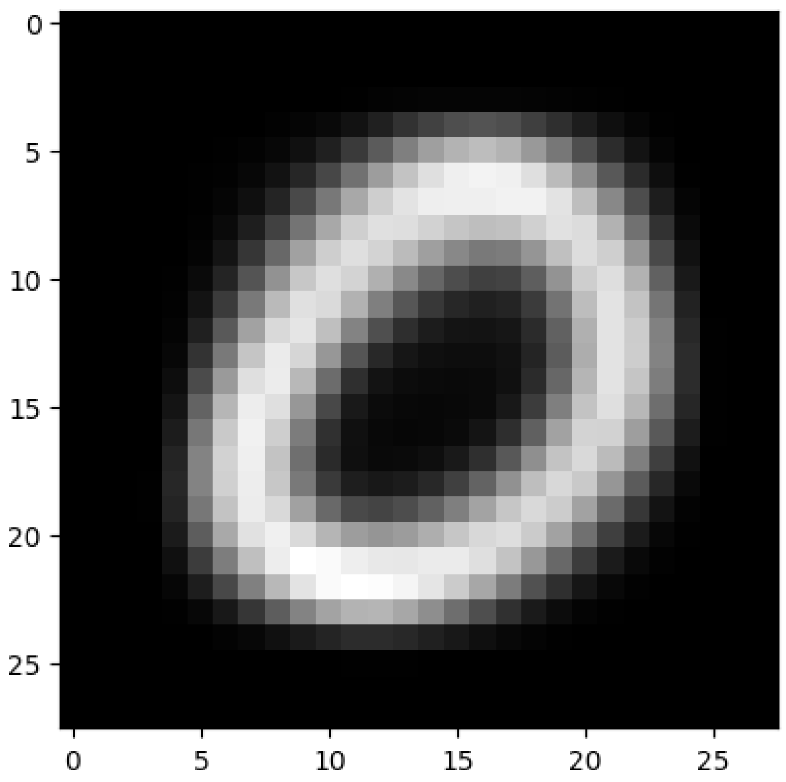
Cluster 3 Centroid



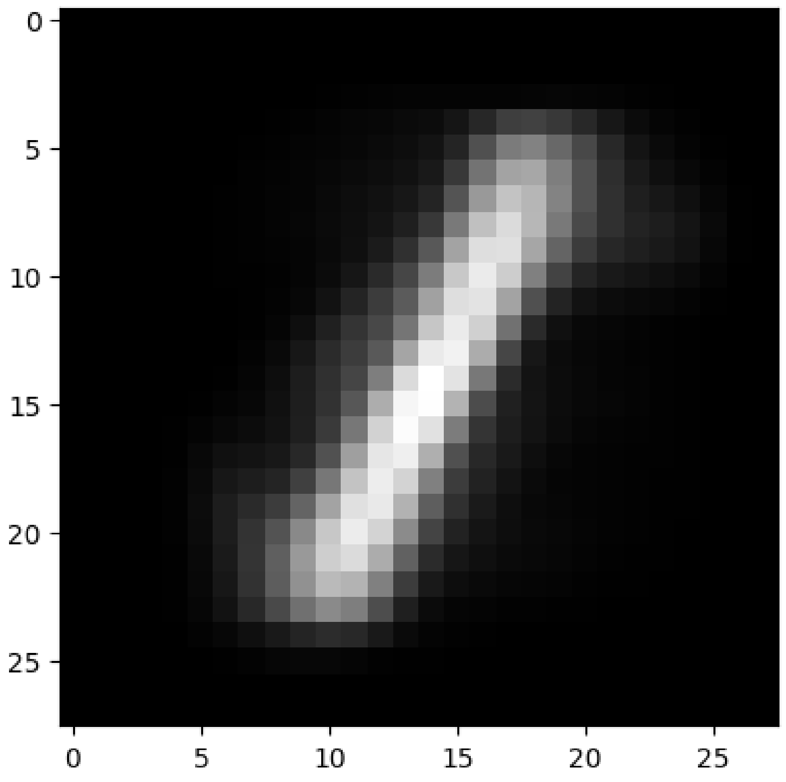
Cluster 4 Centroid



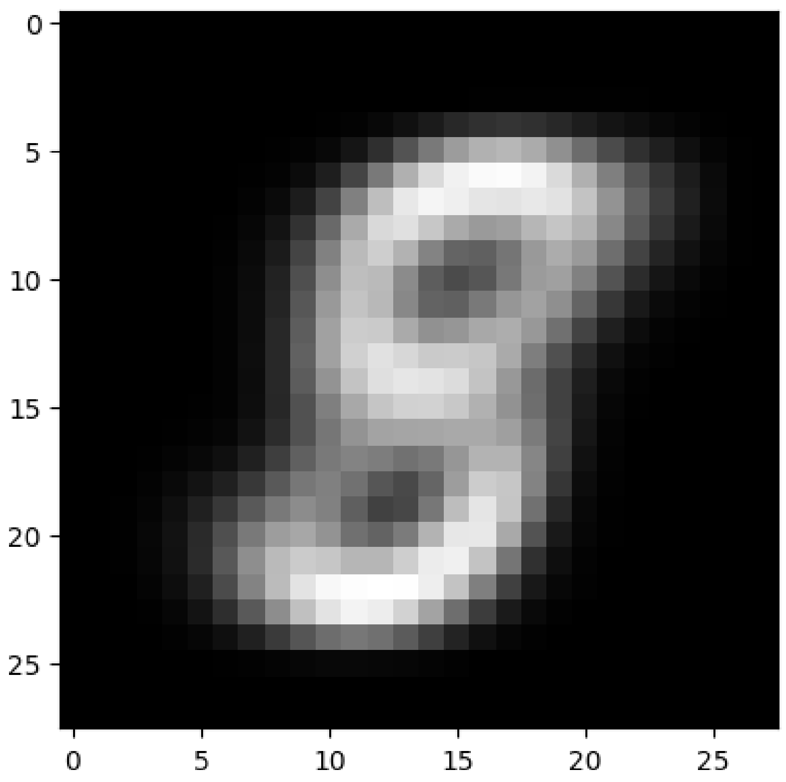
Cluster 5 Centroid



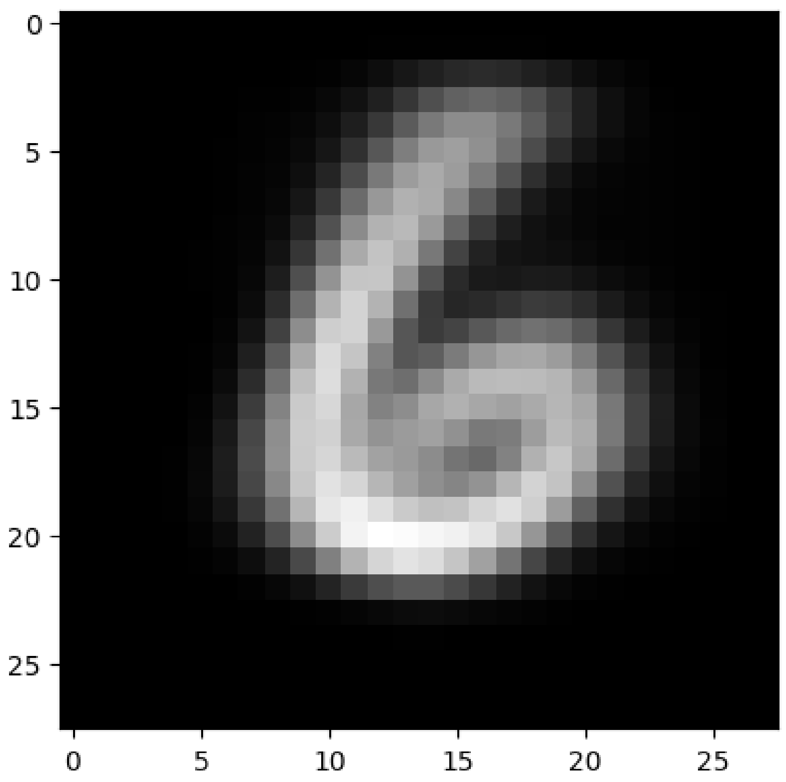
Cluster 6 Centroid

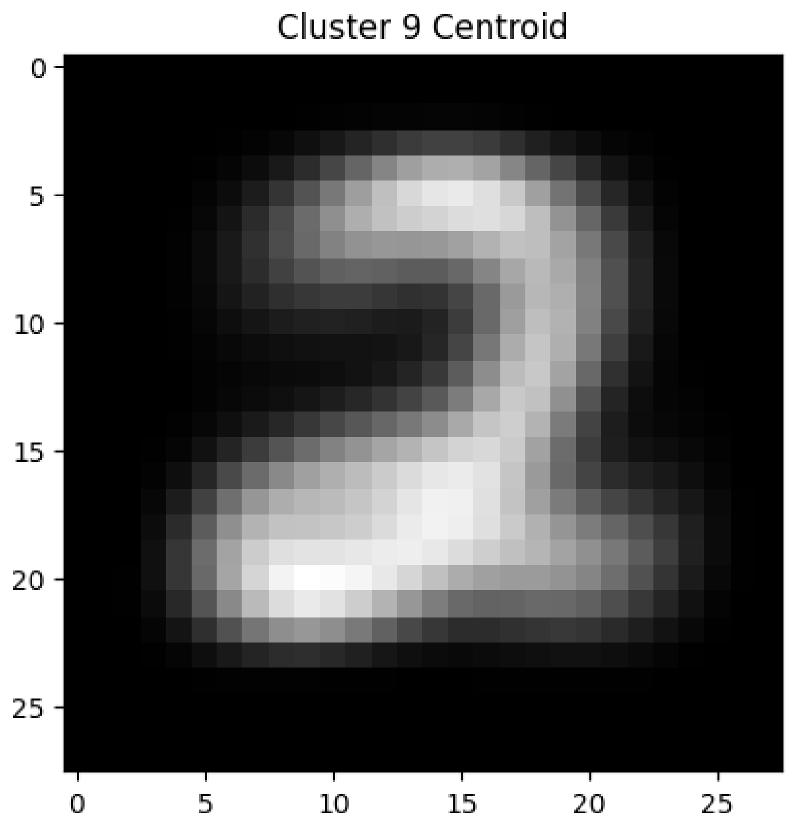


Cluster 7 Centroid



Cluster 8 Centroid





```
# ------------------------------
# 5b Map Clusters to Actual Digits (Insight)
# ------------------------------
# Count which label appears most in each cluster
cluster_label_mapping = {}
for i in range(10):
    indices = np.where(clusters == i)[0]
    most_common_label = np.bincount(y_train[indices]).argmax()
    cluster_label_mapping[i] = most_common_label
    print(f"Cluster {i} mostly contains digit: {most_common_label}")
```

```
Cluster 0 mostly contains digit: 4
Cluster 1 mostly contains digit: 1
Cluster 2 mostly contains digit: 7
Cluster 3 mostly contains digit: 3
Cluster 4 mostly contains digit: 9
Cluster 5 mostly contains digit: 0
Cluster 6 mostly contains digit: 1
Cluster 7 mostly contains digit: 8
Cluster 8 mostly contains digit: 6
Cluster 9 mostly contains digit: 2
```

```
# ------------------------------
# 6 Supervised Learning: Logistic Regression
# ------------------------------
# Logistic Regression classifier on flattened features
lr = LogisticRegression(max_iter=1000, solver='lbfgs',
multi_class='auto')
lr.fit(x_train_flat, y_train)
```

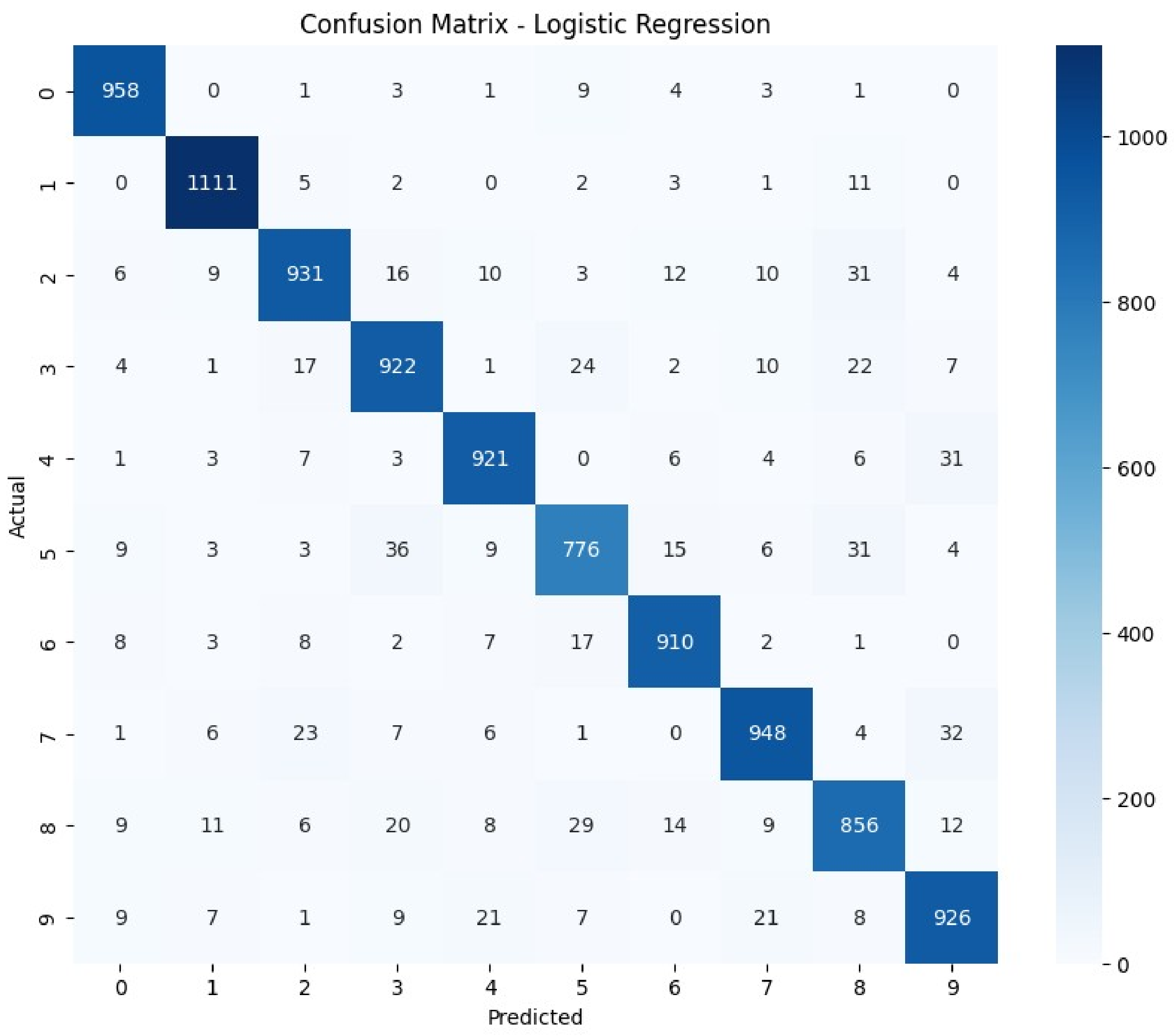
```
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/
_logistic.py:1247: FutureWarning: 'multi_class' was deprecated in
version 1.5 and will be removed in 1.7. From then on, it will always
use 'multinomial'. Leave it to its default value to avoid this
warning.
  warnings.warn(
```

```
LogisticRegression(max_iter=1000, multi_class='auto')
```

```
# Predict on test set
y_pred = lr.predict(x_test_flat)

# ------------------------------
# 6a Evaluate Model
# ------------------------------
accuracy = accuracy_score(y_test, y_pred)
print(f"Logistic Regression Accuracy: {accuracy*100:.2f}%")
```

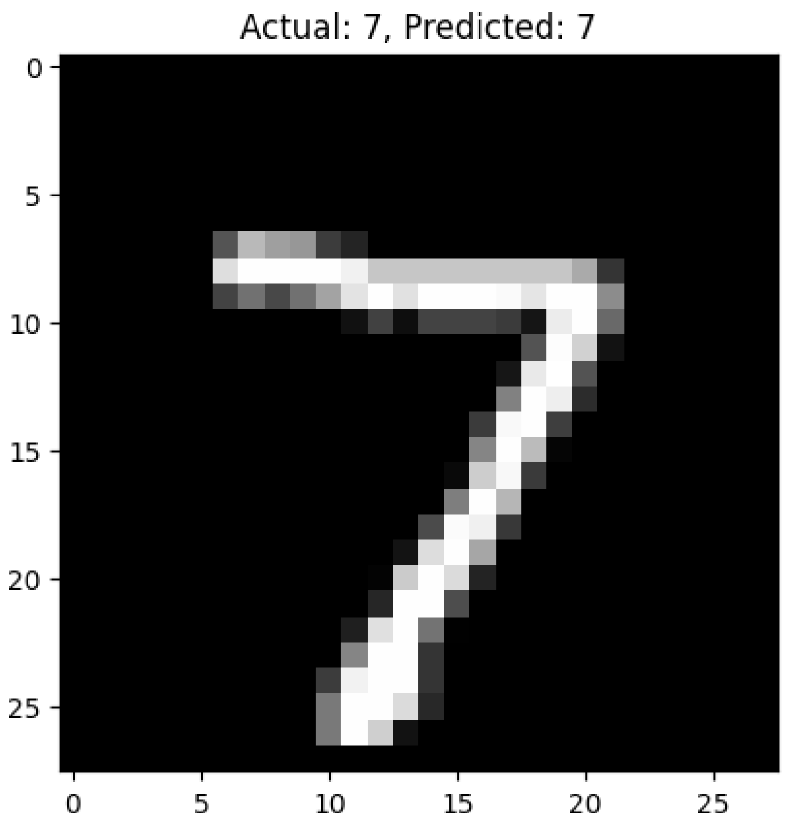
```
Logistic Regression Accuracy: 92.59%
```

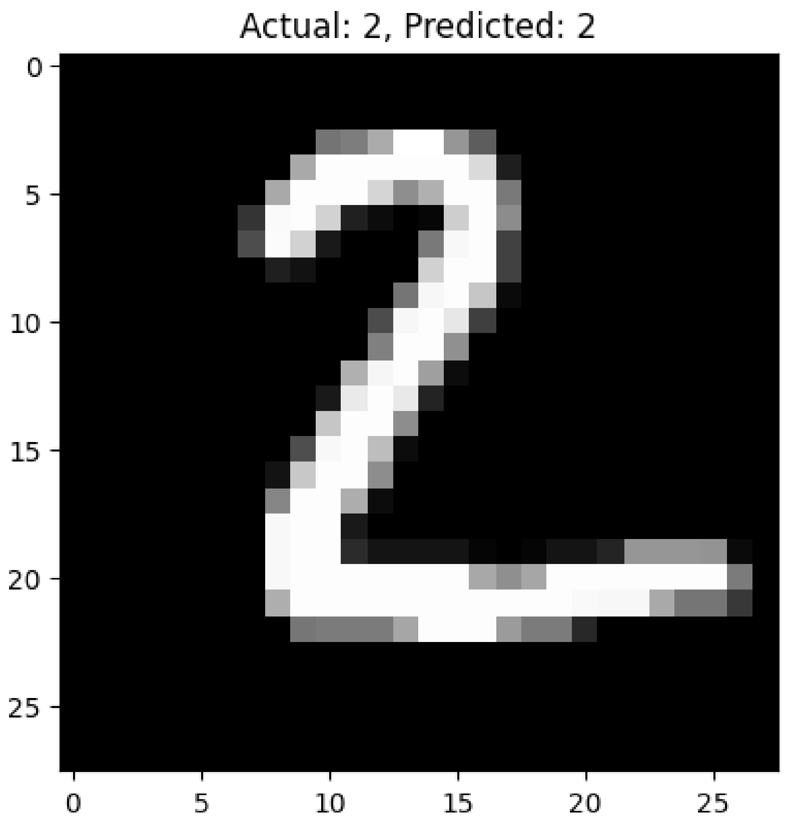
```
# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(10,8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix - Logistic Regression")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



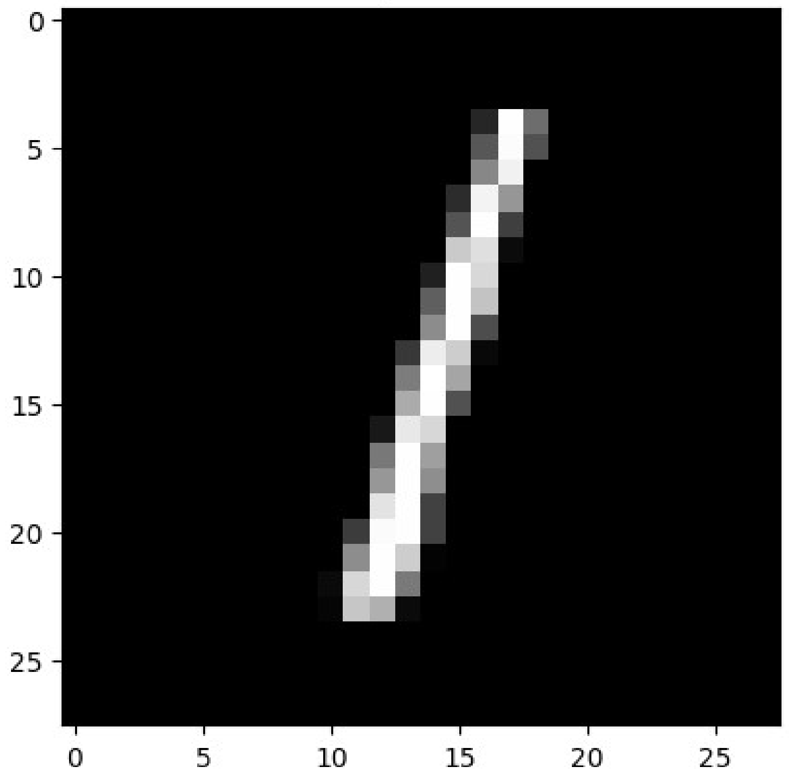
Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	958	0	1	3	1	9	4	3	1	0
1	0	1111	5	2	0	2	3	1	11	0
2	6	9	931	16	10	3	12	10	31	4
3	4	1	17	922	1	24	2	10	22	7
4	1	3	7	3	921	0	6	4	6	31
5	9	3	3	36	9	776	15	6	31	4
6	8	3	8	2	7	17	910	2	1	0
7	1	6	23	7	6	1	0	948	4	32
8	9	11	6	20	8	29	14	9	856	12
9	9	7	1	9	21	7	0	21	8	926

```
# ------------------------------
# 7 Some Predictions vs Actual
# ------------------------------
for i in range(5):
    plt.imshow(x_test[i], cmap='gray')
    plt.title(f"Actual: {y_test[i]}, Predicted: {y_pred[i]}")
    plt.show()
```

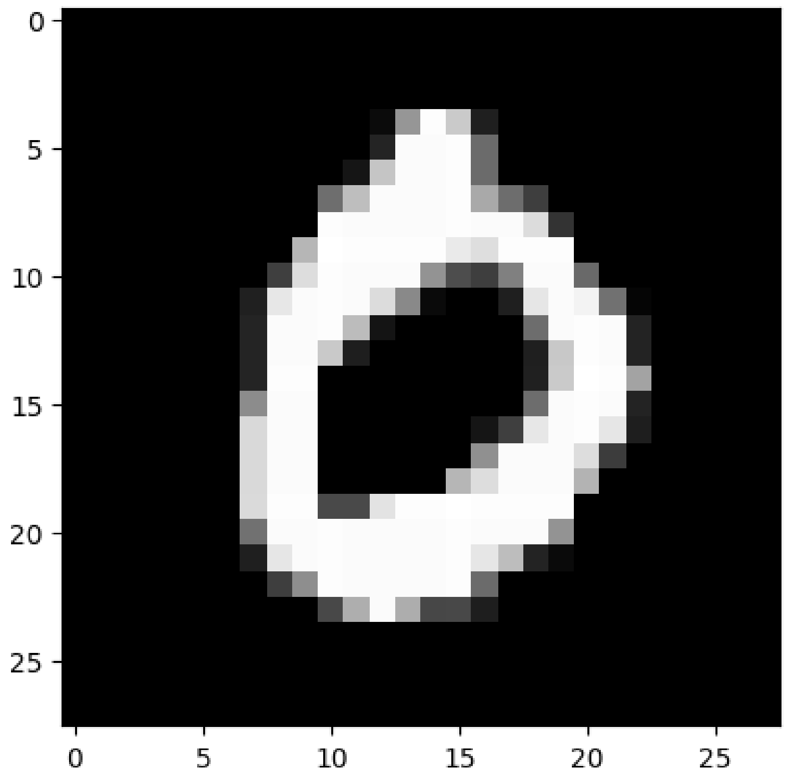

Actual: 7, Predicted: 7
0
5
10
15
20
25
0
5
10
15
20
25

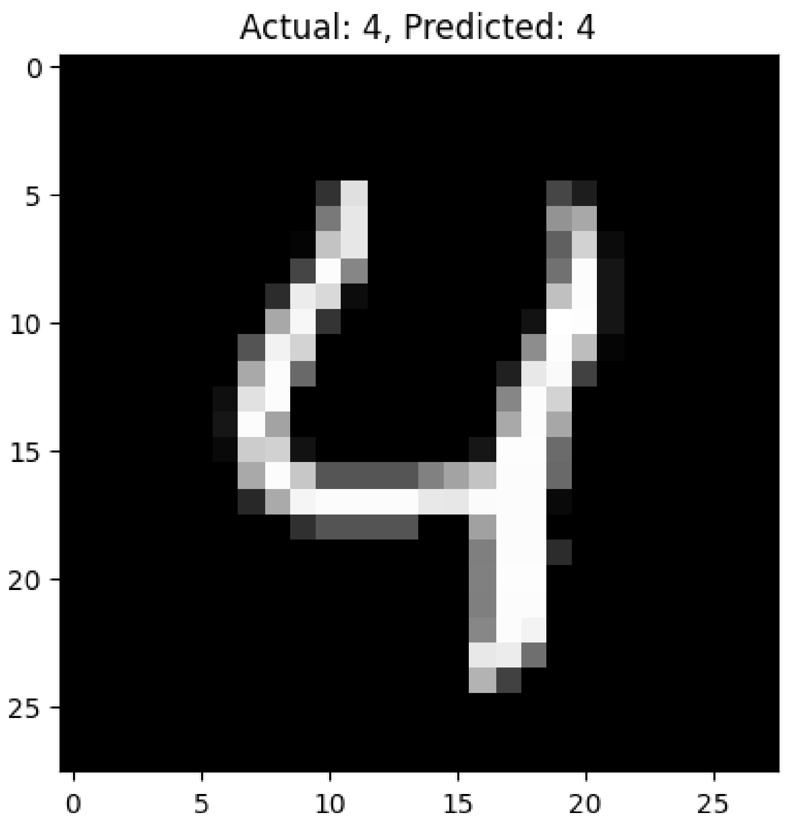

Actual: 2, Predicted: 2
0
5
10
15
20
25
0
5
10
15
20
25

Actual: 1, Predicted: 1



Actual: 0, Predicted: 0




Actual: 4, Predicted: 4
0
5
10
15
20
25
0
5
10
15
20
25