```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# from sklearn.datasets import fetch_openml
# from sklearn.model_selection import train_test_split
# from sklearn.preprocessing import StandardScaler
# from sklearn.neighbors import KNeighborsClassifier
# from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

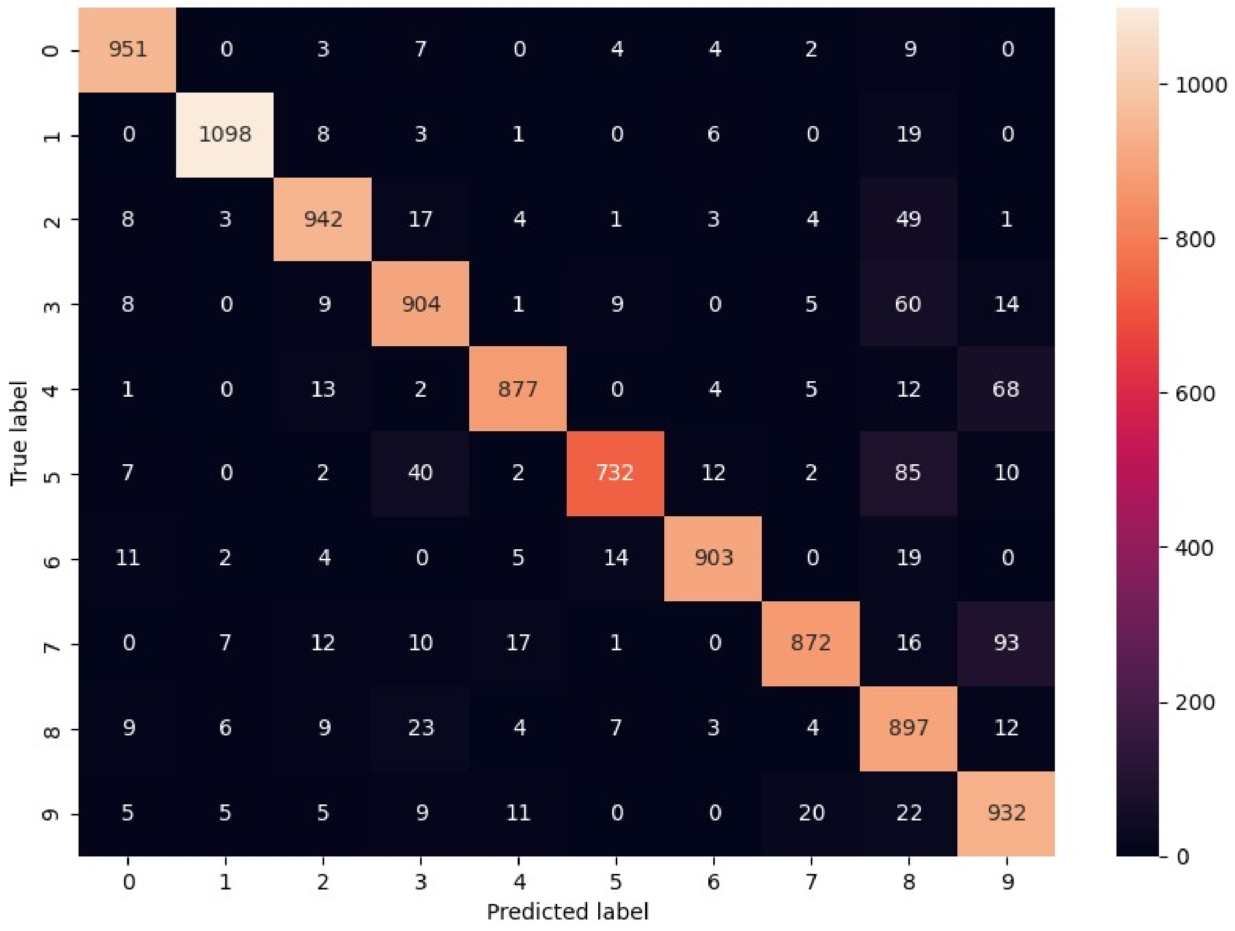
```
from scipy.stats import multivariate_normal as mvn

class GaussBayes():
  def fit(self, X, y, epsilon=1e-3):
    self.likelihoods = dict()
    self.priors = dict()
    #only ints in classes and getting structured set
    self.K = set(y.astype(int))

    for k in self.K:
      X_k=X[y==k]
      N_k, D = X_k.shape
      mu_k = X_k.mean(axis=0)
      self.likelihoods[k] = {"mean": mu_k, "cov": (1/(N_k-1)) *
np.matmul((X_k-mu_k).T, X_k-mu_k)+epsilon*np.identity(D)}
      self.priors[k] = len(X_k)/len(X)

  def predict(self, X):
      N, D = X.shape
      P_hat = np.zeros((N, len(self.K)))

      for k , l in self.likelihoods.items():
        P_hat[:,k] = mvn.logpdf(X, l["mean"],l["cov"]) +
np.log(self.priors[k])

      return P_hat.argmax(axis = 1)

def ConfusionMatrix(y_actual,y_predicted):
  plt.figure(figsize=(10,7))
  y_actu = pd.Series(y_actual, name='Actual')
  y_pred = pd.Series(y_predicted, name='Predicted')
  cm = pd.crosstab(y_actu, y_pred)
  ax = sns.heatmap(cm, annot=True, fmt="d")
  plt.ylabel('True label')
```

```python
  plt.xlabel('Predicted label')
  return cm

def accuracy(y,y_hat):
  return np.mean(y==y_hat)

gbays = GaussBayes()

# Load CSV data
train_df = pd.read_csv("/content/drive/MyDrive/Colab
Notebooks/MNIST_train.csv")
test_df = pd.read_csv("/content/drive/MyDrive/Colab
Notebooks/MNIST_test.csv - MNIST_test.csv.csv")

# Features start at column 3
X_train = train_df.iloc[:, 3:].values.astype(float)
X_test  = test_df.iloc[:, 3:].values.astype(float)

# □ FIX: labels must be 1-D int array
y_train = train_df.iloc[:, 2].values.astype(int)
y_test  = test_df.iloc[:, 2].values.astype(int)

# Normalize
X_train /= 255.0
X_test  /= 255.0

print("X_train:", X_train.shape)
print("y_train:", y_train.shape)
print("X_test:", X_test.shape)
print("y_test:", y_test.shape)

gbays.fit(X_train, y_train)
```

```
X_train: (60000, 784)
y_train: (60000,)
X_test: (10000, 784)
y_test: (10000,)
```

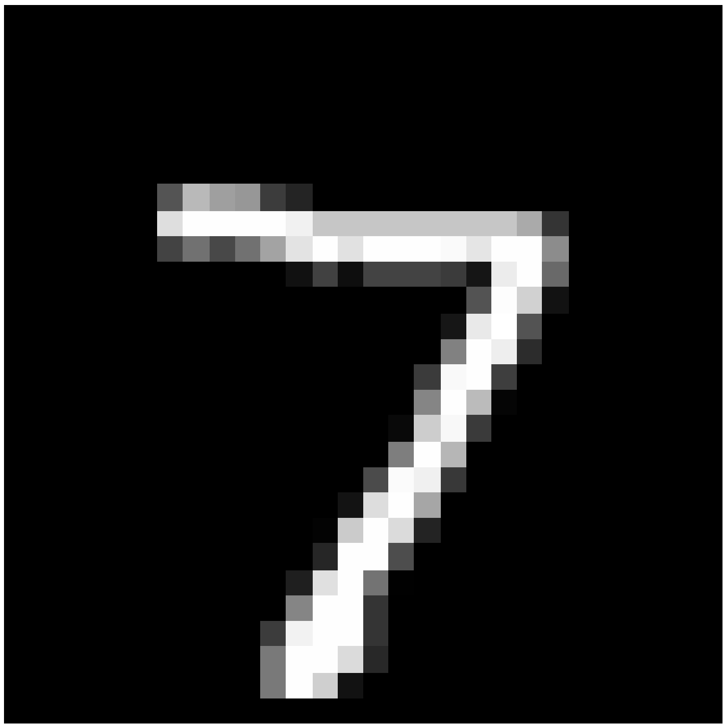
```python
# Predict
# y_hat_bayes=gbays.predict(X_train)

y_pred = gbays.predict(X_test)

# Evaluate
print("Model Accuracy: {:.2f}%".format(accuracy(y_test, y_pred) *
100))
# print("\nClassification Report:\n", classification_report(y_test,
y_pred))

# # Confusion Matrix
cm = ConfusionMatrix(y_test, y_pred)
```

```
plt.show()

# Visualize some predictions
#converted xtest to numpy so no longer has .iloc
for i in range(10):
    img = np.array(X_test[i]).reshape(28, 28)
    plt.imshow(img, cmap='gray')
    plt.title(f"True: {y_test[i]} | Predicted: {y_pred[i]}")
    plt.axis('off')
    plt.show()

Model Accuracy: 91.08%
```

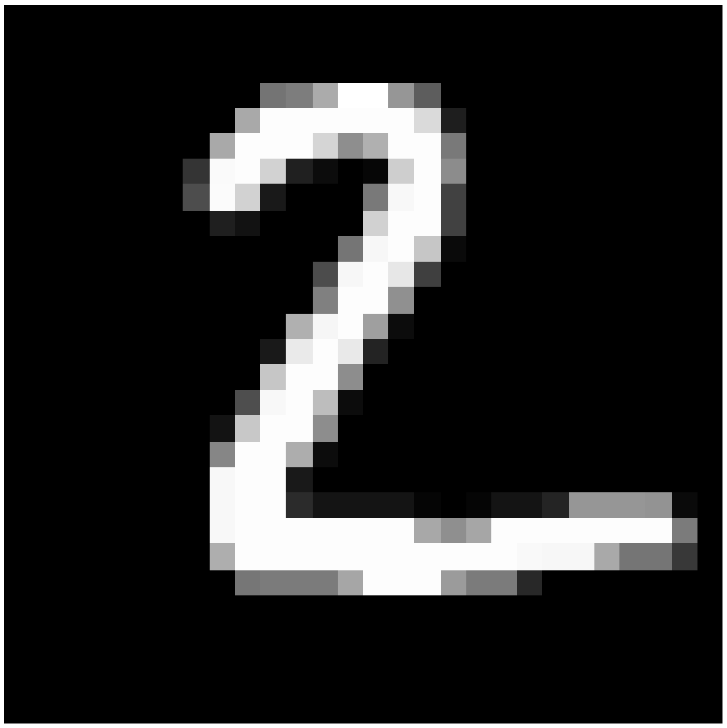


	0	1	2	3	4	5	6	7	8	9
0	951	0	3	7	0	4	4	2	9	0
1	0	1098	8	3	1	0	6	0	19	0
2	8	3	942	17	4	1	3	4	49	1
3	8	0	9	904	1	9	0	5	60	14
4	1	0	13	2	877	0	4	5	12	68
5	7	0	2	40	2	732	12	2	85	10
6	11	2	4	0	5	14	903	0	19	0
7	0	7	12	10	17	1	0	872	16	93
8	9	6	9	23	4	7	3	4	897	12
9	5	5	5	9	11	0	0	20	22	932

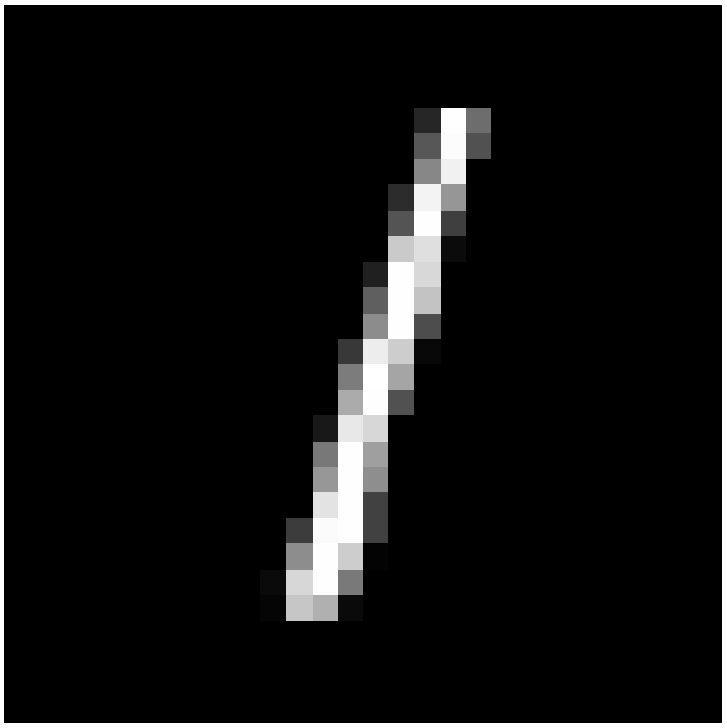
True: 7 | Predicted: 7



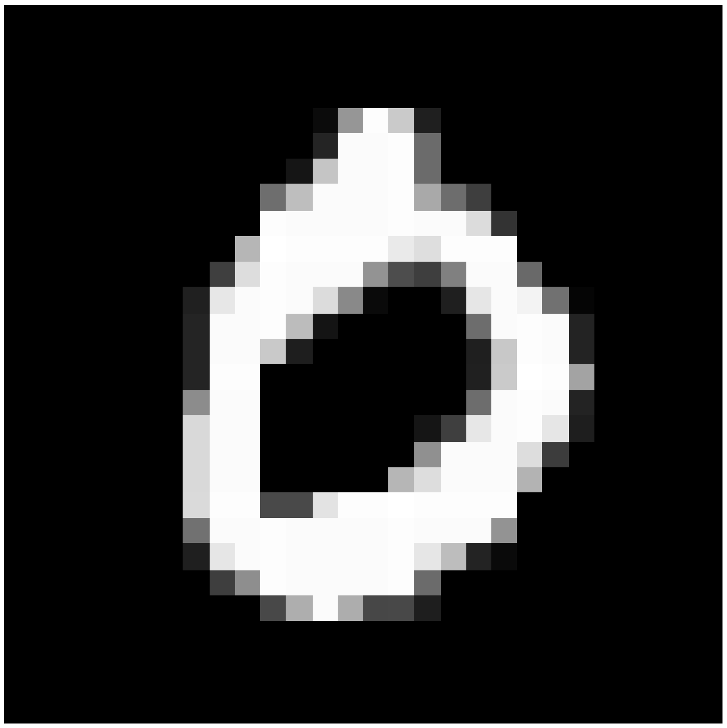
True: 2 | Predicted: 2



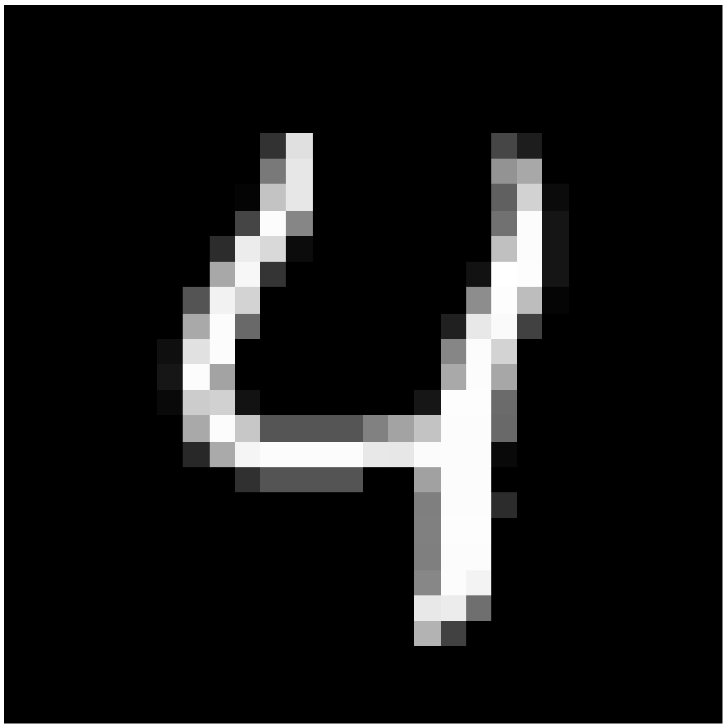
True: 1 | Predicted: 1



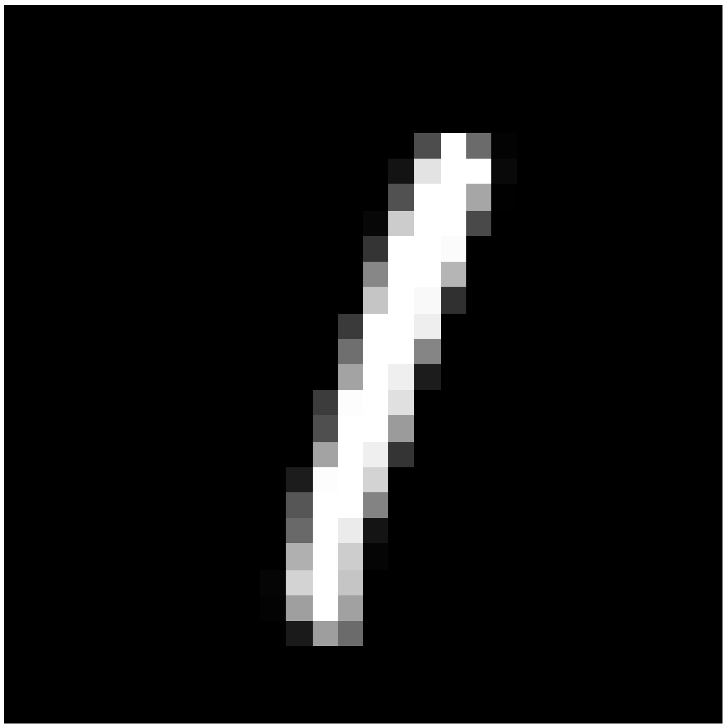
True: 0 | Predicted: 0



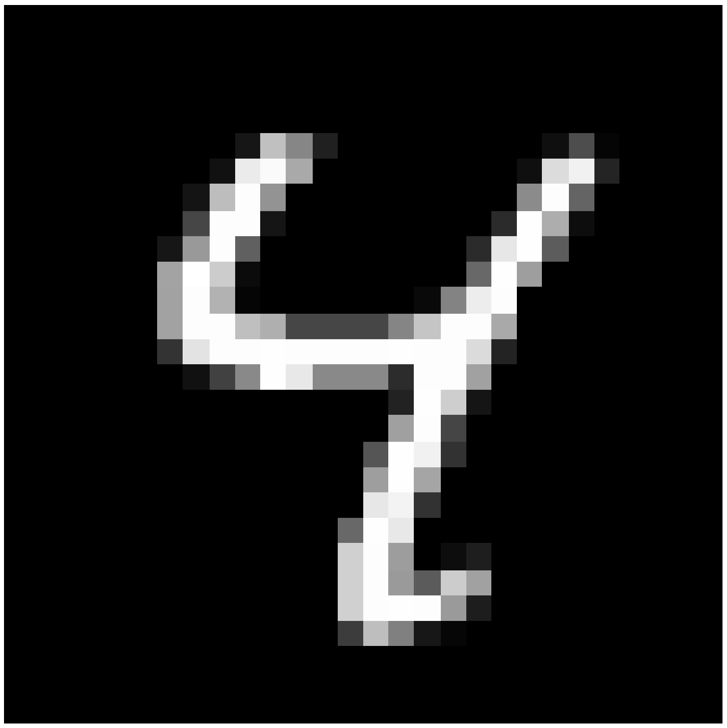
True: 4 | Predicted: 4



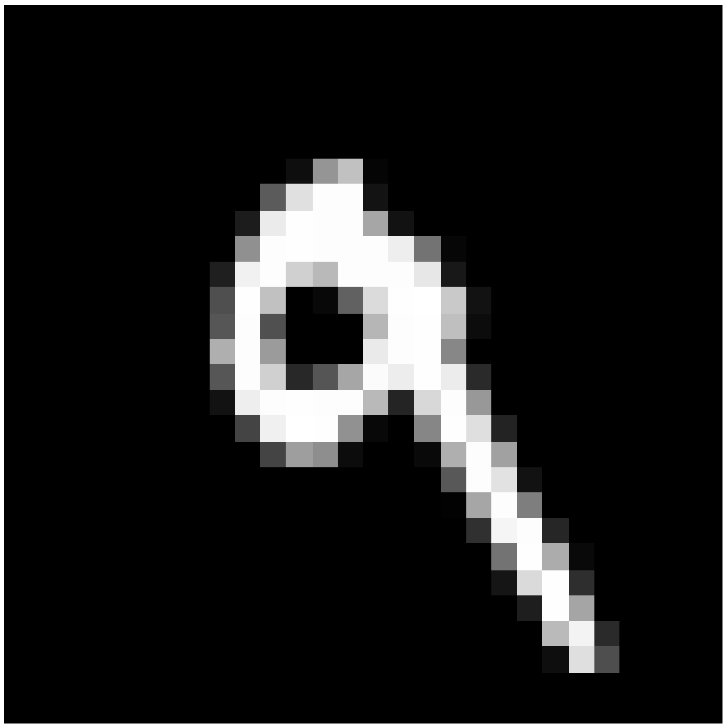
True: 1 | Predicted: 1



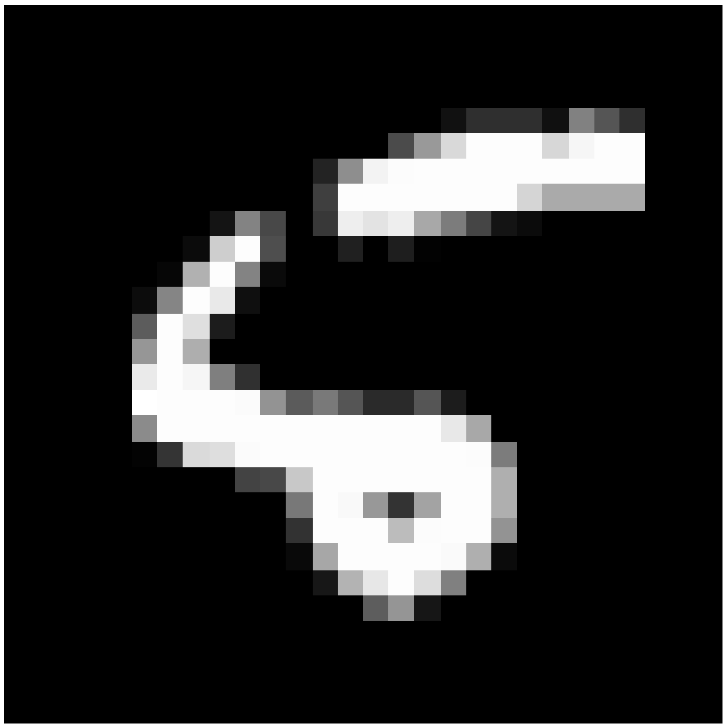
True: 4 | Predicted: 4



True: 9 | Predicted: 9



True: 5 | Predicted: 5



True: 9 | Predicted: 9