MNIST DATA SET

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

data_test = pd.read_csv('/content/sample_data/MNIST_test.csv')
data_train = pd.read_csv('/content/sample_data/MNIST_train.csv')

Total_train = data_train.to_numpy()
Total_test = data_test.to_numpy()

y_train = Total_train[:,2]
y_test = Total_test[:,2]

X1_train = Total_train[:,3:]
X1_test = Total_test[:,3:]
```

Min Max scaling

```
X_train = X1_train/255
X_test = X1_test/255

X_train.shape
```

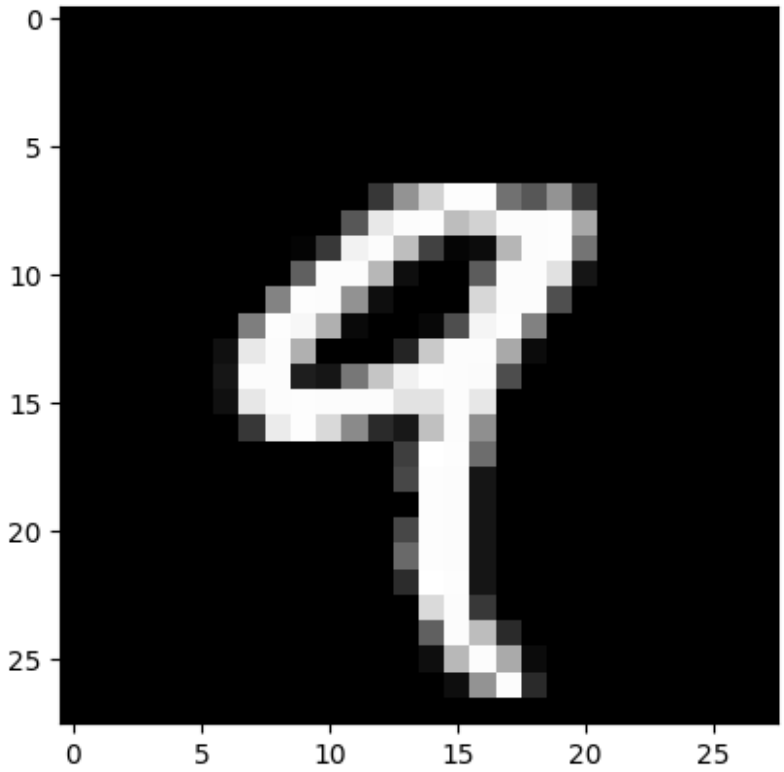
```
(60000, 784)
```

```
X_test.shape
```

```
(10000, 784)
```

Sample Grayscale image

```
plt.imshow(X_train[4].reshape(28,28),cmap='gray')
plt.show()
```



Naive Bayes Classifier

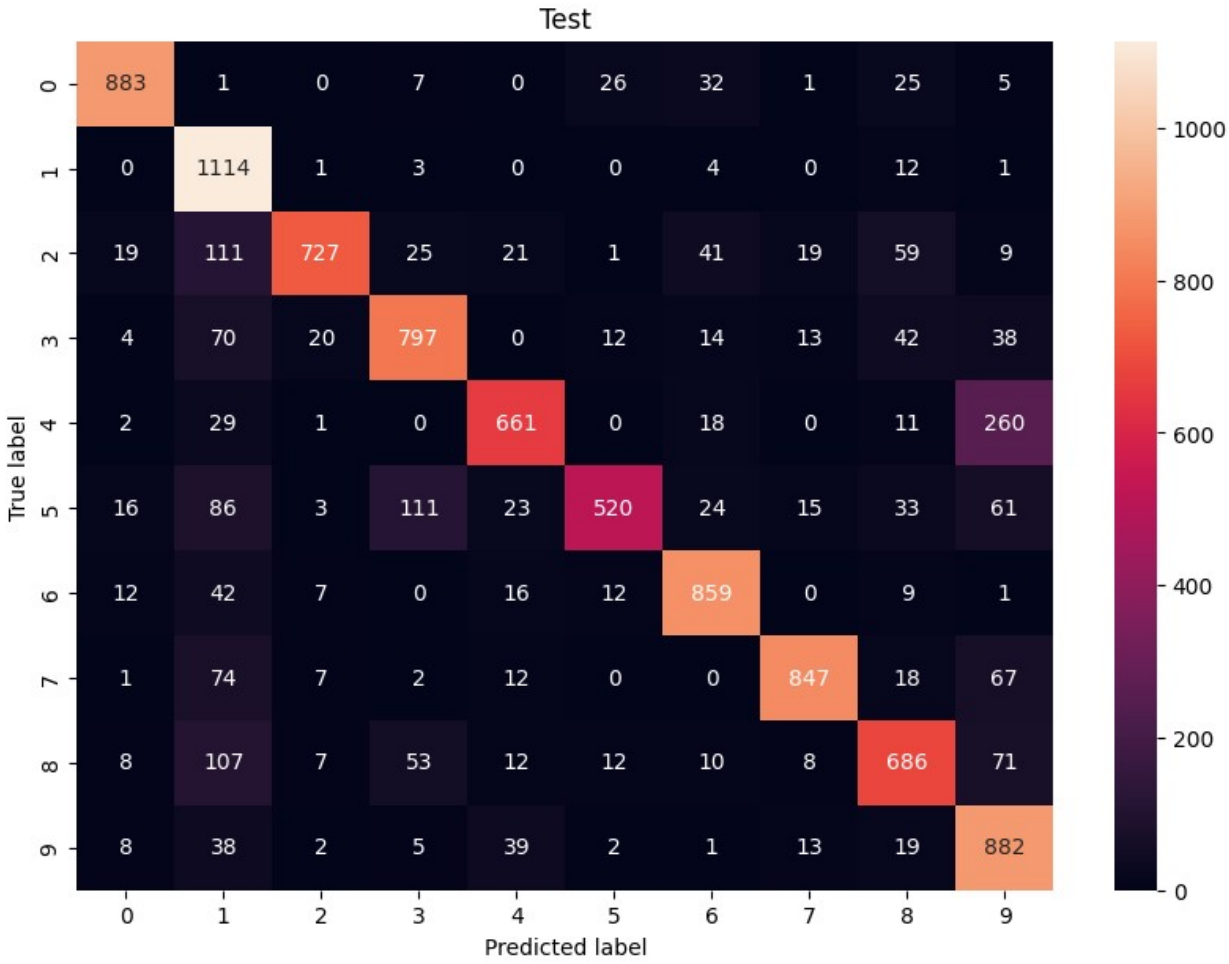
```
from scipy.stats import multivariate_normal as mvn

class GausNB():

  def fit(self, X,y,epsilon= 0.06):

    self.likelihoods = dict()
    self.priors = dict()
    self.K = set(y.astype(int))
    for k in self.K:
      X_k = X[y==k,:]
      self.likelihoods[k] =
{"mean":X_k.mean(axis=0),"cov":X_k.var(axis=0)+epsilon}
      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)
```

Accuracy function Definition

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

gnb = GausNB()

gnb.fit(X_train,y_train)

y_hat = gnb.predict(X_test)
```

Confusion Matrix

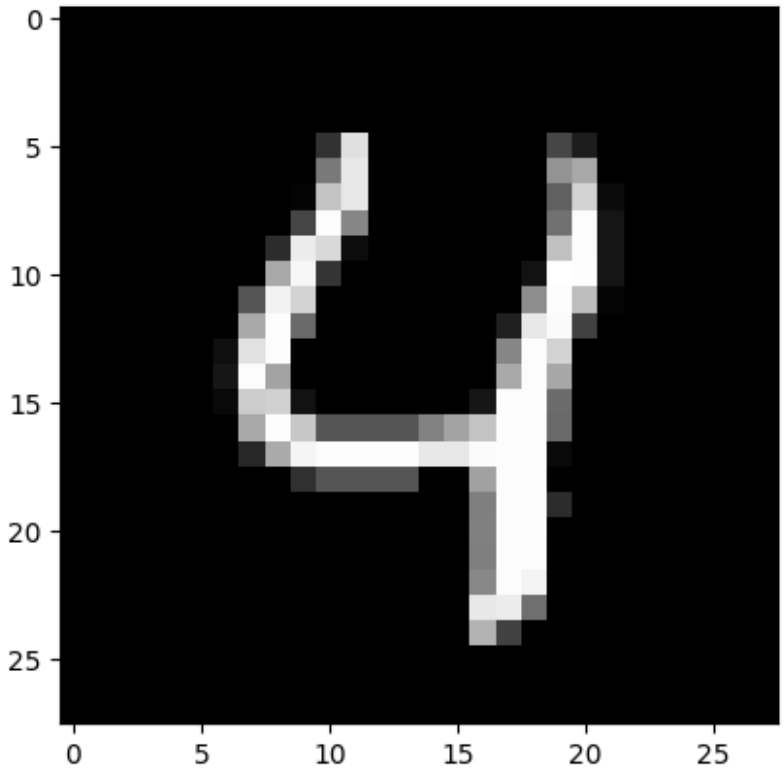
```
y_actu = pd.Series(y_test, name='Actual').to_numpy()
y_pred = pd.Series(y_hat, name='Predicted').to_numpy()
cm = pd.crosstab(y_actu, y_pred)

plt.figure(figsize=(10,7))
ax = sns.heatmap(cm, annot=True, fmt="d")
plt.title('Test')

plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()


print(f'                                        Train accuracy
{accuracy(y_actu,y_pred)*100}')
```




```
                                    Train accuracy 79.75999999999999
```

Grayscale image after applying Scaling & Naive Model

```
plt.imshow(X_test[4].reshape(28,28),cmap='gray')
plt.show()
```



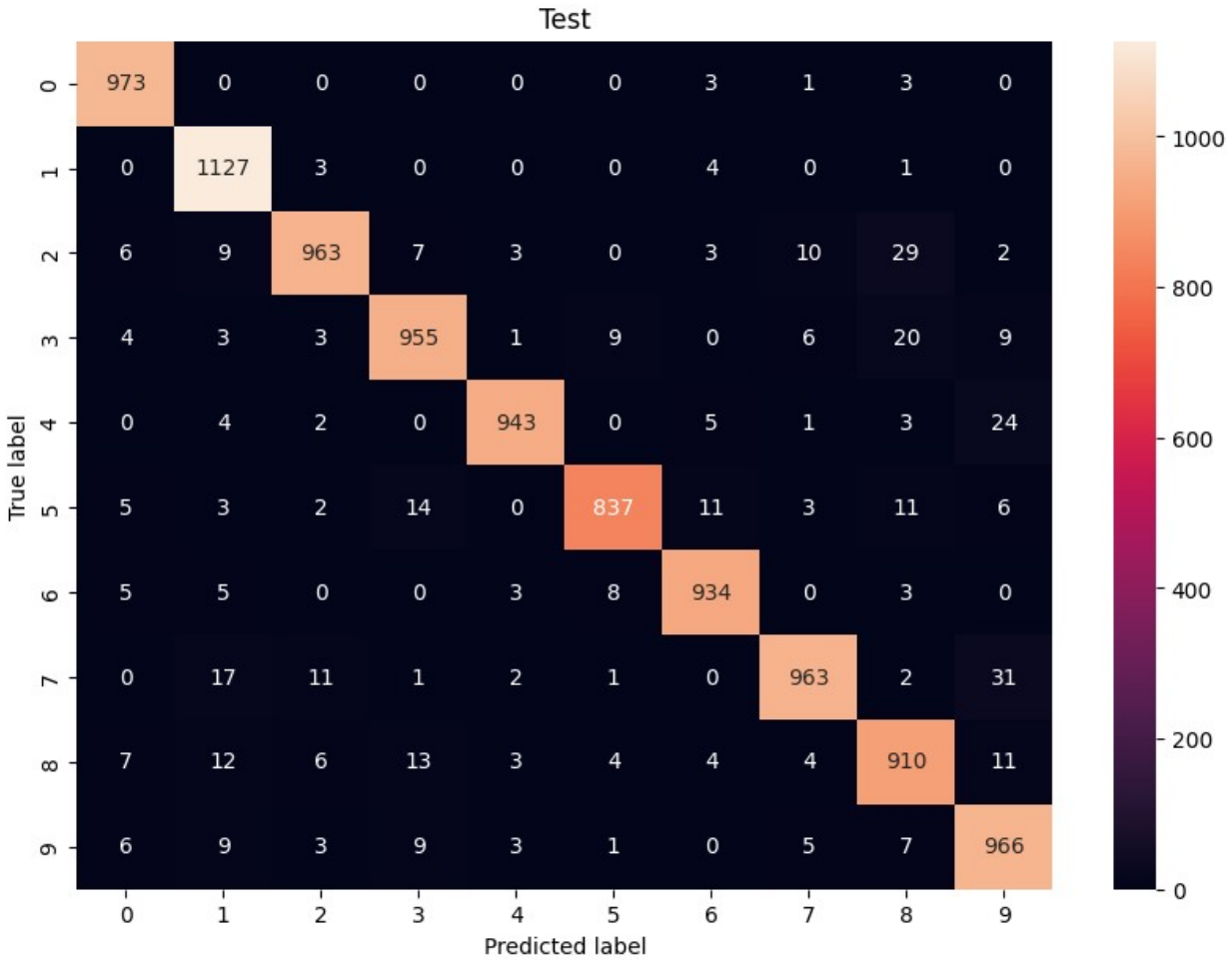

```
accuracy(y_test,y_hat)
```

```
np.float64(0.7976)
```

Non Naive Bayes Classifier

```
class GaussBayes():

  def fit(self, X,y,epsilon= 0.06):

    self.likelihoods = dict()
    self.priors = dict()
    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":X_k.mean(axis=0),
                             "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):
```

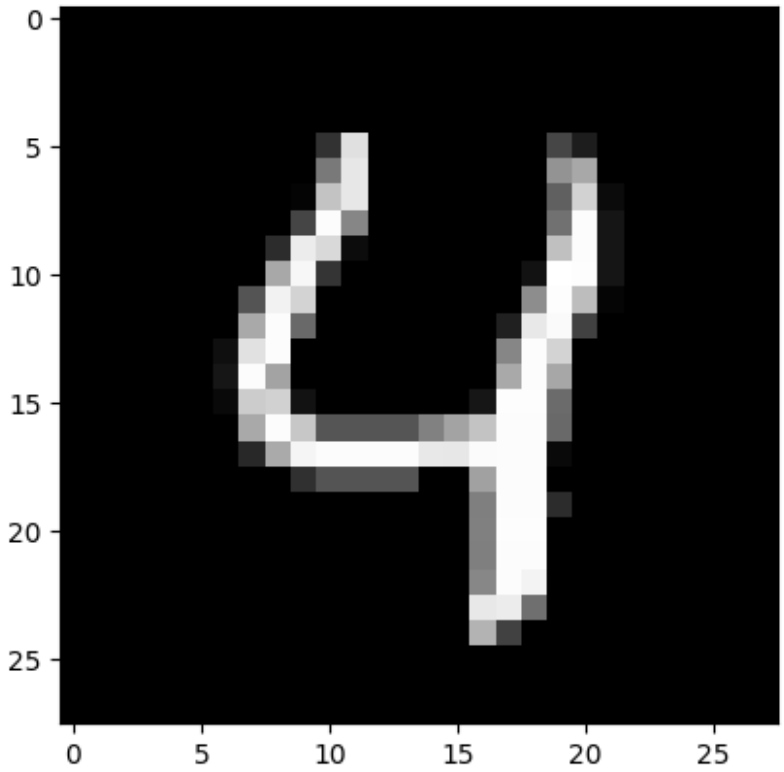
```
    N, D = X.shape
    P_hat = np.zeros((N,len(self.K)))

    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)

gbays_non_naive = GaussBayes()

gbays_non_naive.fit(X_train,y_train)

y_hat_gbayes = gbays_non_naive.predict(X_test)

Confusion Matrix after applying Scaling & Non-Naive Model

y_actu = pd.Series(y_test, name='Actual').to_numpy()
y_pred = pd.Series(y_hat_gbayes, name='Predicted').to_numpy()
cm = pd.crosstab(y_actu, y_pred)

plt.figure(figsize=(10,7))
ax = sns.heatmap(cm, annot=True, fmt="d")
plt.title('Test')

plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()

print(f'                                Train accuracy
{accuracy(y_actu,y_pred)*100}')
```




```
Train accuracy 95.71
```

Grayscale image after applying Scaling & Non-Naive Model

```
plt.imshow(X_test[4].reshape(28,28),cmap='gray')
plt.show()
```




```
accuracy(y_test,y_hat_gbayes)
```

```
np.float64(0.9571)
```

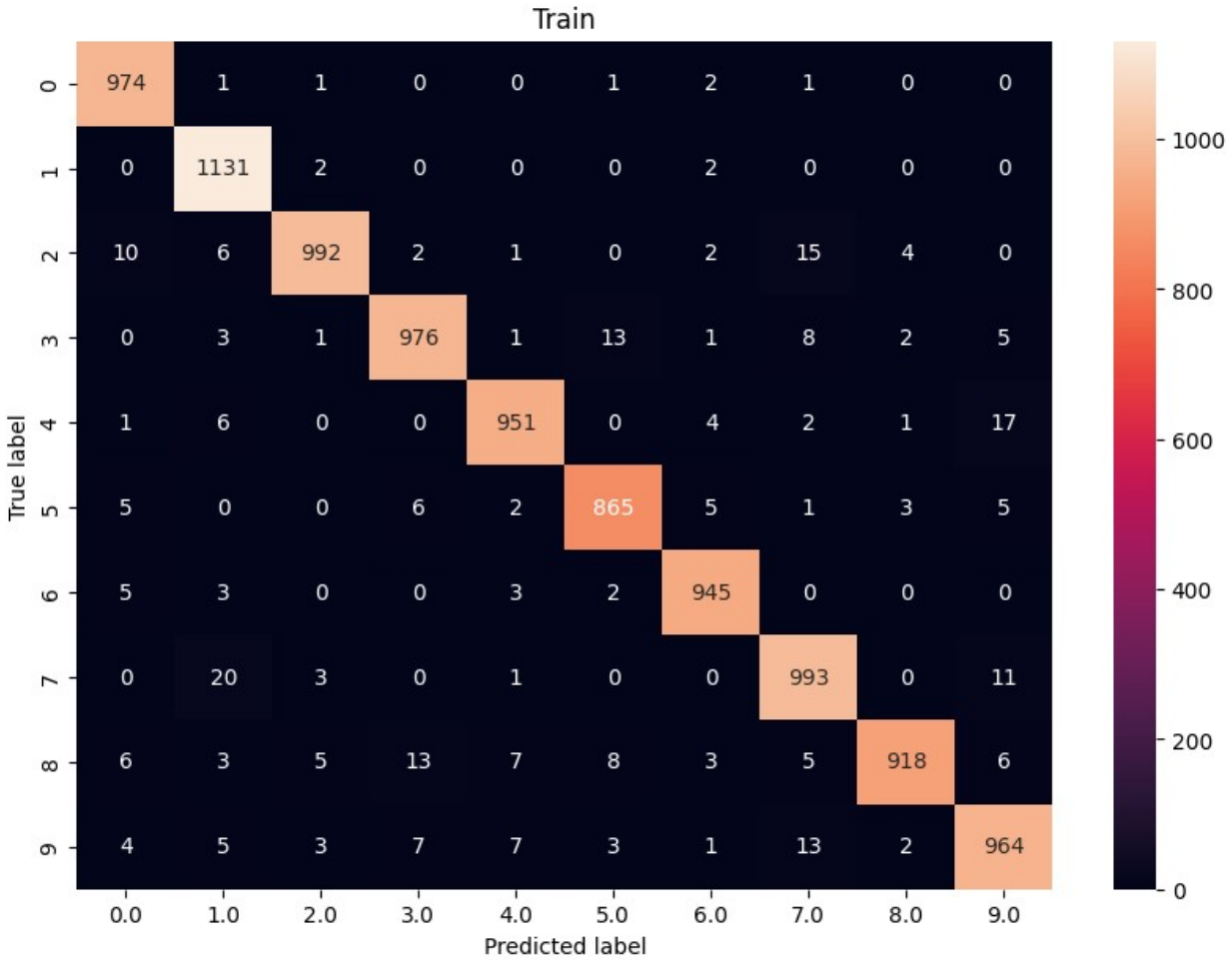
KNN

```
from ast import Return
class KNNClassifier():

  def fit(self,X,y):
    self.X =X
    self.y =y

  def predict(self,X,K,epsilon= 0.06):

    N = len(X)
    y_hat = np.zeros(N)

    for i in range(N):
      dist2 = np.sum((self.X-X[i])**2,axis=1)
      idxt = np.argsort(dist2)[:K]
      gamma_k = 1/(np.sqrt(dist2[idxt]+epsilon))

      y_hat[i] = np.bincount(self.y[idxt],weights=gamma_k).argmax()

    return y_hat
```

```
Knn_instance = KNNClassifier()

Knn_instance.fit(X_train,y_train)

y_hat_Knn = Knn_instance.predict(X_test,K=6)
```

Confusion Matrix after applying Scaling & KNN Model

```
y_actu = pd.Series(y_test, name='Actual').to_numpy()
y_pred = pd.Series(y_hat_Knn, name='Predicted').to_numpy()
cm = pd.crosstab(y_actu, y_pred)

plt.figure(figsize=(10,7))
ax = sns.heatmap(cm, annot=True, fmt="d")
plt.title('Test')

plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()

print(f'                                    Train accuracy
{accuracy(y_actu,y_pred)*100}')
```



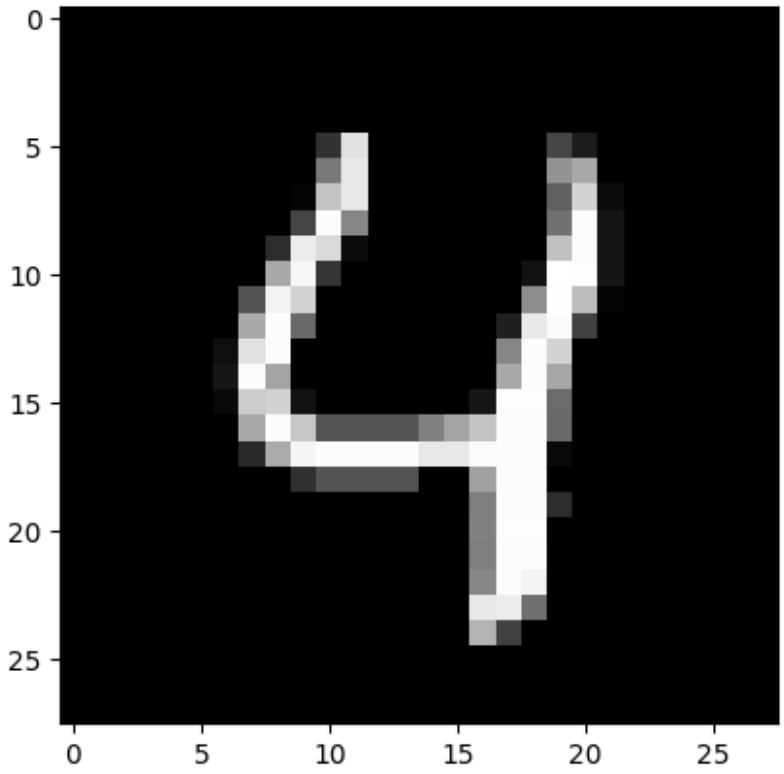

```
                                        Train accuracy 97.09
```

```
accuracy(y_test,y_hat)
```

```
np.float64(0.7746)
```

Grayscale image after applying Scaling & KNN Model

```
plt.imshow(X_test[4].reshape(28,28),cmap='gray')
plt.show()
```


0
5
10
15
20
25
0
5
10
15
20
25