In [321…

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

In [322…

```
train_data= pd.read_csv('/content/MNIST_train.csv')
test_data=pd.read_csv('/content/MNIST_test.csv')
train_data.shape
```

Out[322…

```
(60000, 787)
```

In [323…

```
X=train_data.to_numpy()
X_test=test_data.to_numpy()
```

In [324…

```
y=X[:10000,2]
y_test=X_test[:10000,2]
```

In [325…

```
y
y_test
```

Out[325…

```
array([7, 2, 1, ..., 4, 5, 6])
```

In [326…

```
set(y)
set(y_test)
```

Out[326…

```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
```

In [327…

```
X=X[:10000,3:]
X_test=X_test[:10000,3:]
```

In [328…

```
X.shape
X_test.shape
```

Out[328…

```
(10000, 784)
```

In [329…

```
X = X / 255
X_test = X_test/ 255
```

In [330…

```
from scipy.stats import multivariate_normal as mvn
```

In [331…

```
class KNNClassifier():
  def fit(self,X,y):
    self.X=X
    self.y=y

  def predict(self, X, K, epsilon=1e-1):
    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].astype(int), weights=gamma_k).argmax()
    return y_hat
```

In [332…

```
knn=KNNClassifier()
knn.fit(X,y)
```

In [333…

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

In [313…

```
y_hat=knn.predict(X,3)
y_hat_test=knn.predict(X_test,3)
```

In [314…

```
accuracy(y,y_hat)
```

Out[314…

```
1.0
```

In [315…

```
acc=accuracy(y_test,y_hat_test)
acc
```

Out[315…

```
0.9494
```

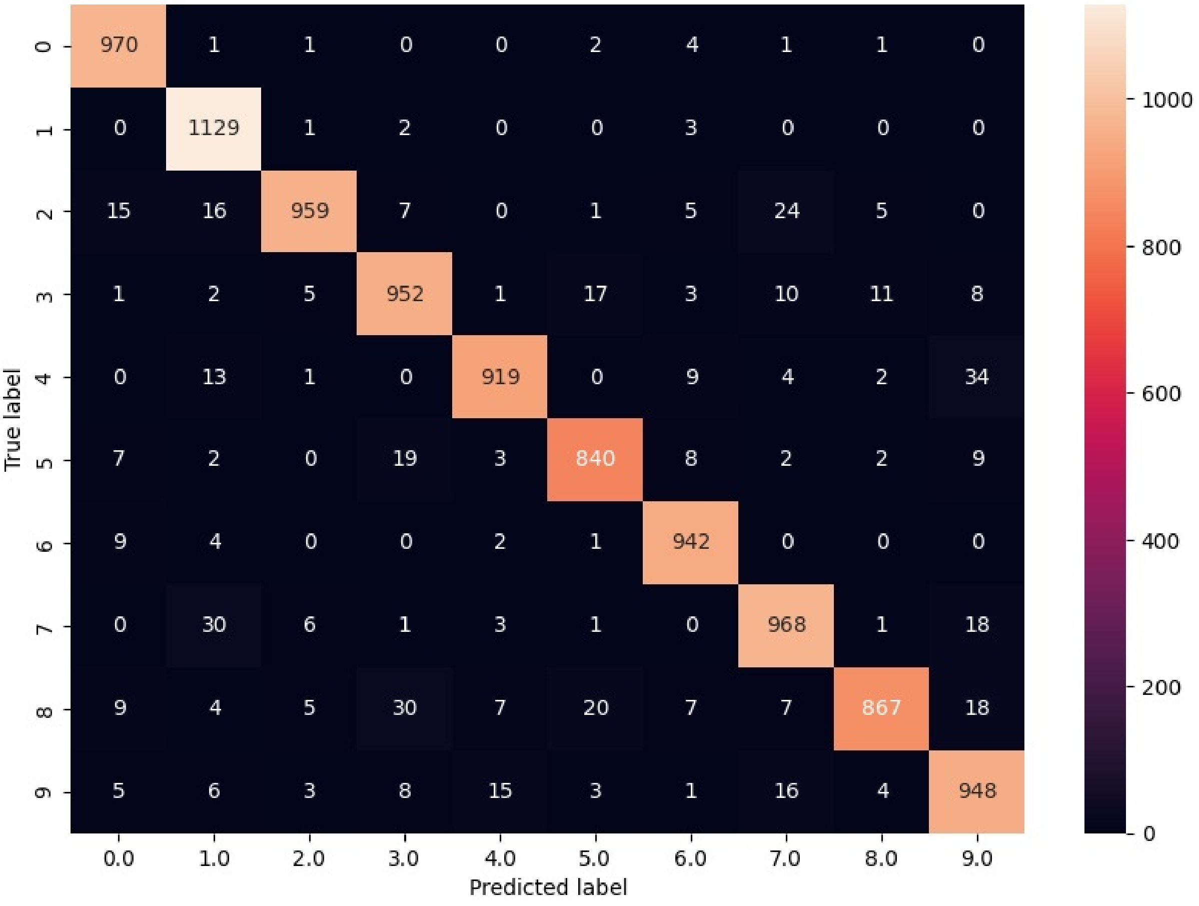
In [316…

```
import seaborn as sns
plt.figure(figsize=(10,7))
y_actu = pd.Series(y_test, name='Actual')
```

```
y_pred = pd.Series(y_hat_test, name='Predicted')
cm = pd.crosstab(y_actu, y_pred)
ax = sns.heatmap(cm, annot=True, fmt="d")
plt.ylabel('True label')
plt.xlabel('Predicted label')
```

Out[316…

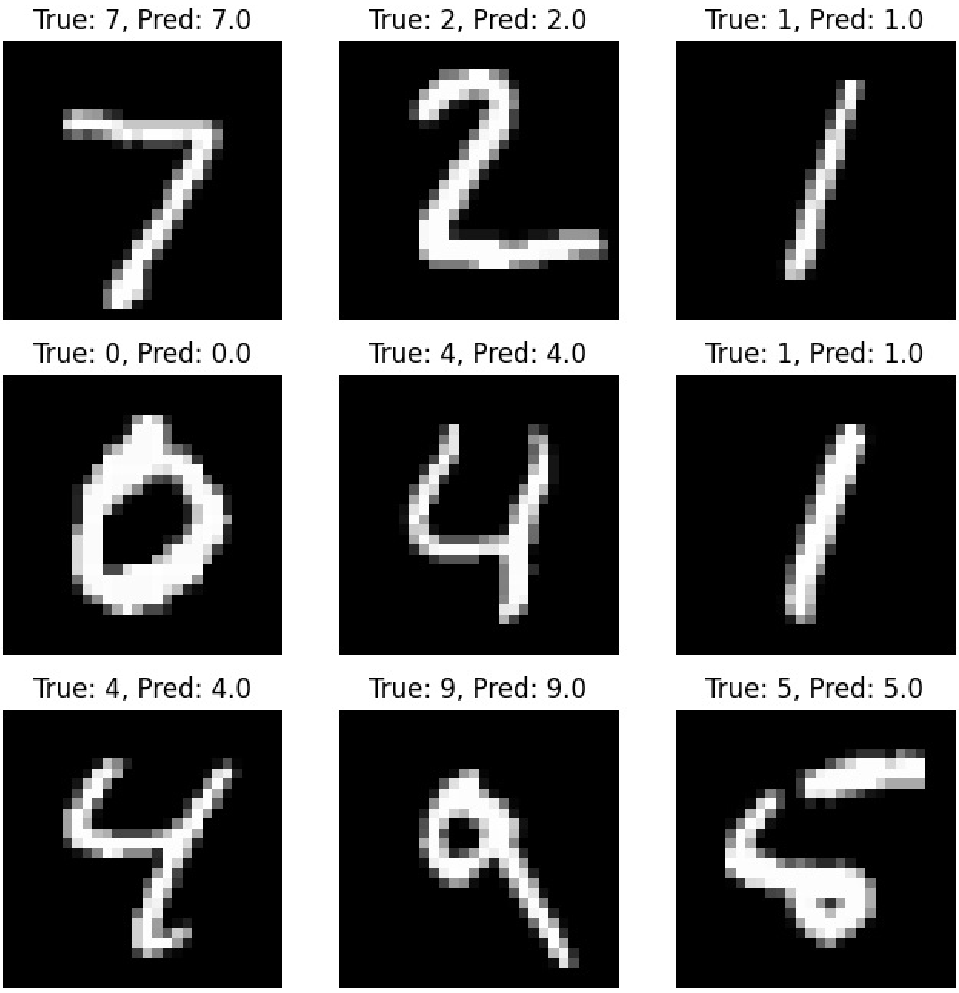
```
Text(0.5, 47.7222222222222, 'Predicted label')
```



	0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0
0	970	1	1	0	0	2	4	1	1	0
1	0	1129	1	2	0	0	3	0	0	0
2	15	16	959	7	0	1	5	24	5	0
3	1	2	5	952	1	17	3	10	11	8
4	0	13	1	0	919	0	9	4	2	34
5	7	2	0	19	3	840	8	2	2	9
6	9	4	0	0	2	1	942	0	0	0
7	0	30	6	1	3	1	0	968	1	18
8	9	4	5	30	7	20	7	7	867	18
9	5	6	3	8	15	3	1	16	4	948

In [317…

```
correct_indices = np.where(y_pred == y_test[:10000])[0]
fig, axes = plt.subplots(3, 3, figsize=(8, 8))
for i, ax in enumerate(axes.flat):
    if i < len(correct_indices):
        index = correct_indices[i]
        image = X_test[index].reshape(28, 28)
        ax.imshow(image, cmap='gray')
        ax.set_title(f"True: {y_test[index]}, Pred: {y_pred[index]}")
        ax.axis('off')
    else:
        ax.axis('off')
plt.show()
```



True: 7, Pred: 7.0

True: 2, Pred: 2.0

True: 1, Pred: 1.0

True: 0, Pred: 0.0

True: 4, Pred: 4.0

True: 1, Pred: 1.0

True: 4, Pred: 4.0

True: 9, Pred: 9.0

True: 5, Pred: 5.0

In [334…

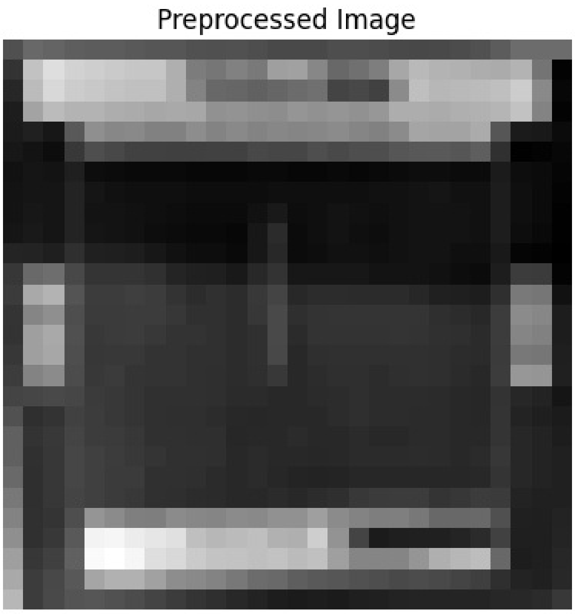
```
!pip install pillow
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt

def load_and_preprocess_image(filepath):
    img = Image.open(filepath).convert('L')
    img = img.resize((28, 28))
    img = np.array(img)
    img = 255 - img
    img = img.flatten()
    img = img / 255.0
    return img

image_path = '/content/Skype_Picture_2024_09_02T20_11_17_444Z.jpeg'
test_image = load_and_preprocess_image(image_path)

plt.imshow(test_image.reshape(28, 28), cmap='gray')
plt.title('Preprocessed Image')
plt.axis('off')
plt.show()
```

```
Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages (9.4.0)
```



Example Image prediction

In [335…

```
K = 3
predicted_label = knn.predict(test_image, K)
print(f'Predicted Label: {predicted_label[0]}')
```

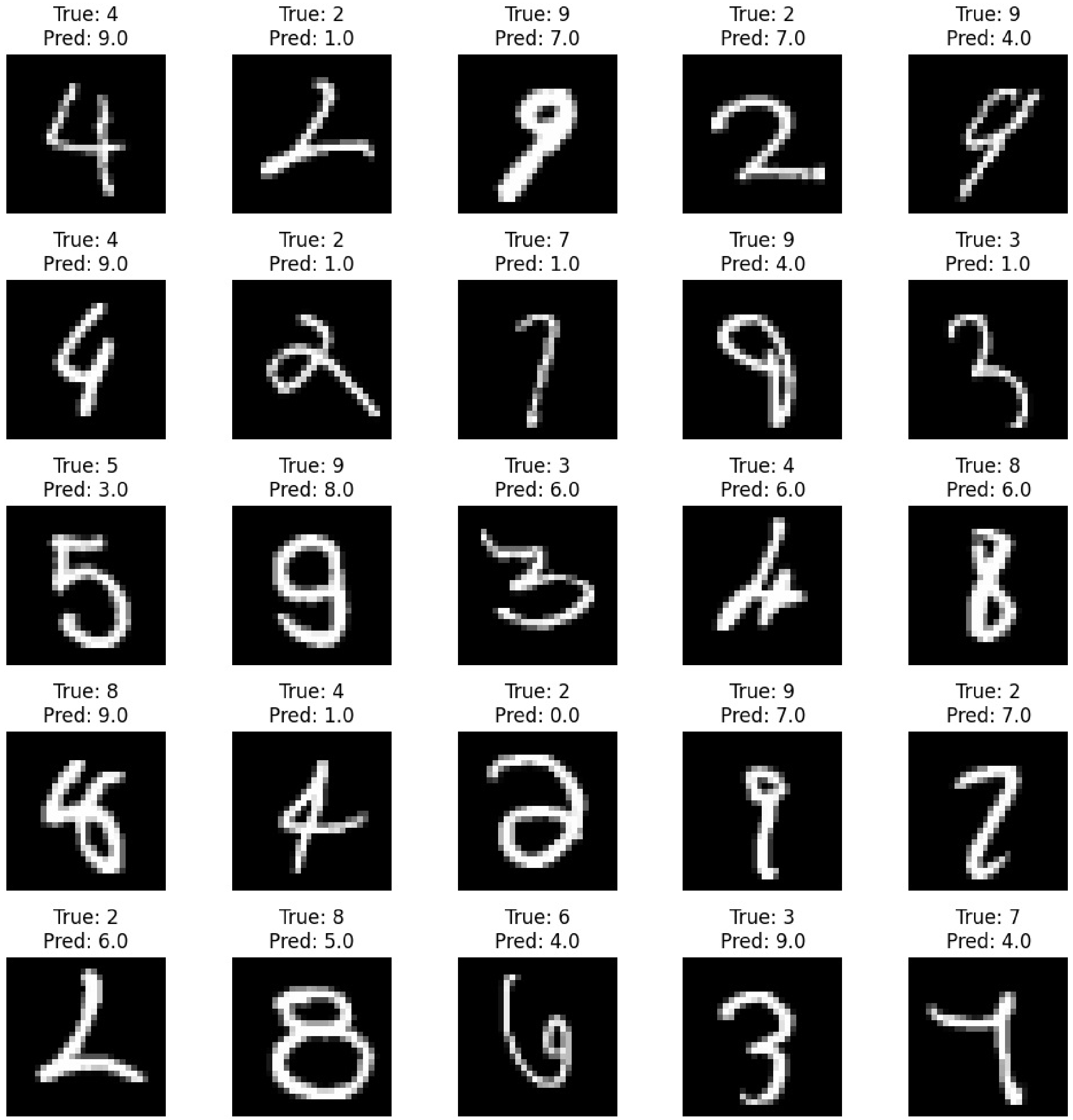
```
Predicted Label: 1.0
```

In [337…

```
def plot_errors(X, y_true, y_pred, num_errors=25):
    incorrect_indices = np.where(y_true != y_pred)[0]

    plt.figure(figsize=(10, 10))
    for i, idx in enumerate(incorrect_indices[:num_errors]):
        plt.subplot(5, 5, i + 1)
        plt.imshow(X[idx].reshape(28, 28), cmap='gray') # Reshape the image data to 28x28
        plt.title(f'True: {y_true[idx]}\nPred: {y_pred[idx]}')
        plt.axis('off')
    plt.tight_layout()
    plt.show()

# Call the function with test images
plot_errors(X_test, y_test, y_hat_test)
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js