In [ ]:
```
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPool2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
```

In [ ]:
```
(X_train,y_train) , (X_test,y_test)=mnist.load_data()
```

Reshaping data

In [ ]:
```
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], X_train.shape
[2], 1))
X_test = X_test.reshape((X_test.shape[0],X_test.shape[1],X_test.shape[2],1))
```

Checking the shape after reshaping

In [ ]:
```
print(X_train.shape)
print(X_test.shape)
```

Normalizing the pixel values

In [ ]:
```
X_train=X_train/255
X_test=X_test/255
```

Model building

```
In [ ]: #defining model
        model=Sequential()
        #adding convolution layer
        model.add(Conv2D(32,(3,3),activation='relu',input_shape=(28,28,1)))
        #adding pooling layer
        model.add(MaxPool2D(2,2))
        #adding fully connected layer
        model.add(Flatten())
        model.add(Dense(100,activation='relu'))
        #adding output layer
        model.add(Dense(10,activation='softmax'))
        #compiling the model
        model.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=
        ['accuracy'])
        #fitting the model
```

Fitting the model

```
In [ ]: model.fit(X_train,y_train,epochs=10)
```

Model evaluation

```
In [ ]: model.evaluate(X_test,y_test)
```

Predictions

```
In [ ]: y_pred=model.predict_classes(X_test)
```

```
In [ ]: y_pred
```

```
In [ ]: y_test
```

```
In [ ]: 
```