In [1]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```python
train_data = pd.read_csv('train.csv')
test_data = pd.read_csv('test.csv')
```

In [3]:

```python
train_data.head()
```

Out[3]:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel775	pixel776	pixel777	pixel778	pixe
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
3	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	

5 rows × 785 columns

In [4]:

```python
train_data.tail()
```

Out[4]:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel775	pixel776	pixel777	pixel778
41995	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
41996	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
41997	7	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
41998	6	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
41999	9	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0

5 rows × 785 columns

In [5]:

```python
test_data.tail()
```

Out[5]:

	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel774	pixel775	pixel776	pixel777	pixel778
27995	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
27996	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
27997	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
27998	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
27999	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0

5 rows × 784 columns

In [6]:

```python
test_data.head()
```

Out[6]:

	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel774	pixel775	pixel776	pixel777	pixel778	pix
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	

5 rows × 784 columns

In [7]:

```python
train_data.shape
```

Out[7]:

```
(42000, 785)
```

In [8]:

```python
test_data.shape
```

Out[8]:

```
(28000, 784)
```

In [9]:

```python
train_data.columns
```

Out[9]:

```
Index(['label', 'pixel0', 'pixel1', 'pixel2', 'pixel3', 'pixel4', 'pixel5',
       'pixel6', 'pixel7', 'pixel8',
       ...
       'pixel774', 'pixel775', 'pixel776', 'pixel777', 'pixel778', 'pixel779',
       'pixel780', 'pixel781', 'pixel782', 'pixel783'],
      dtype='object', length=785)
```

In [10]:

```python
test_data.columns
```

Out[10]:

```
Index(['pixel0', 'pixel1', 'pixel2', 'pixel3', 'pixel4', 'pixel5', 'pixel6',
       'pixel7', 'pixel8', 'pixel9',
       ...
       'pixel774', 'pixel775', 'pixel776', 'pixel777', 'pixel778', 'pixel779',
       'pixel780', 'pixel781', 'pixel782', 'pixel783'],
      dtype='object', length=784)
```

In [11]:

```python
train_data.duplicated().sum()
```

Out[11]:

```
0
```

In [12]:

```python
test_data.duplicated().sum()
```

Out[12]:

```
0
```

In [13]:

```python
train_data.isnull().sum()
```

Out[13]:

```
label       0
pixel0      0
pixel1      0
pixel2      0
pixel3      0
           ..
pixel779    0
pixel780    0
pixel781    0
pixel782    0
pixel783    0
Length: 785, dtype: int64
```

In [14]:

```python
test_data.isnull().sum()
```

Out[14]:

```
pixel0      0
pixel1      0
pixel2      0
pixel3      0
pixel4      0
           ..
pixel779    0
pixel780    0
pixel781    0
pixel782    0
pixel783    0
Length: 784, dtype: int64
```

In [15]:

```python
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42000 entries, 0 to 41999
Columns: 785 entries, label to pixel783
dtypes: int64(785)
memory usage: 251.5 MB
```

In [16]:

```python
test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28000 entries, 0 to 27999
Columns: 784 entries, pixel0 to pixel783
dtypes: int64(784)
memory usage: 167.5 MB
```

In [17]:

```python
train_data.describe()
```

Out[17]:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixe
count	42000.000000	42000.0	42000.0	42000.0	42000.0	42000.0	42000.0	42000.0	42000.0	42000.0	...	42000.000000	42000.000
mean	4.456643	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.219286	0.117
std	2.887730	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	6.312890	4.633
min	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000
25%	2.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000
50%	4.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000
75%	7.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000
max	9.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	254.000000	254.000

8 rows × 785 columns

In [18]:

```python
test_data.describe()
```

Out[18]:

	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel774	pixel775
count	28000.0	28000.0	28000.0	28000.0	28000.0	28000.0	28000.0	28000.0	28000.0	28000.0	...	28000.000000	28000.000000
mean	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.164607	0.073214
std	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	5.473293	3.616811
min	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000
25%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000
50%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000
75%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000
max	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	253.000000	254.000000

8 rows × 784 columns

In [19]:

```python
train_data.nunique()
```

Out[19]:

```
label       10
pixel0       1
pixel1       1
pixel2       1
pixel3       1
            ..
pixel779     3
pixel780     1
pixel781     1
pixel782     1
pixel783     1
Length: 785, dtype: int64
```

In [20]:

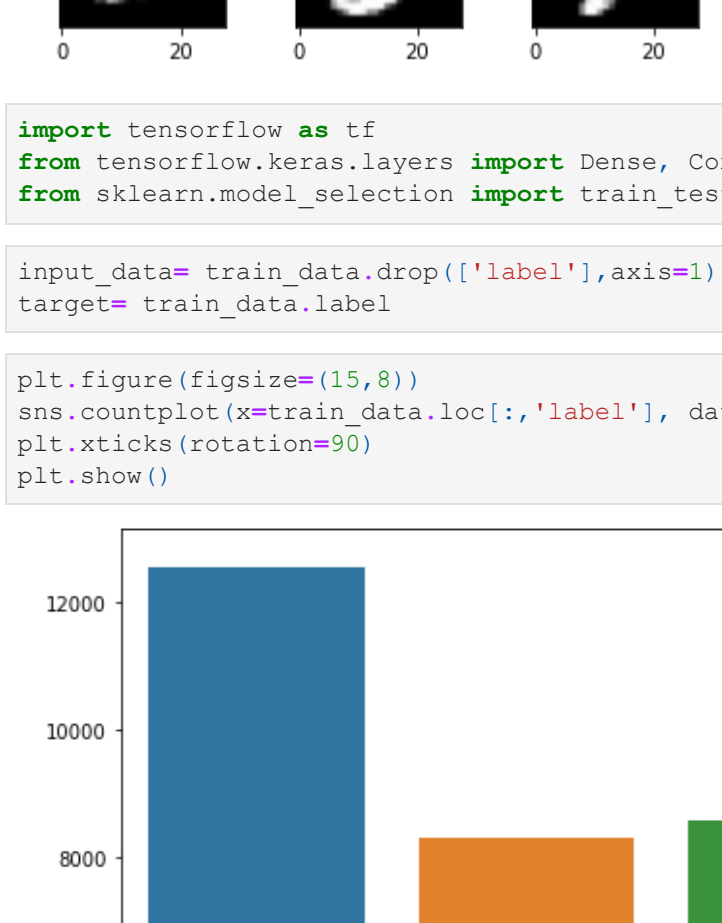
```python
test_data.nunique()
```

Out[20]:

```
pixel0      1
pixel1      1
pixel2      1
pixel3      1
pixel4      1
           ..
pixel779    1
pixel780    1
pixel781    1
pixel782    1
pixel783    1
Length: 784, dtype: int64
```

In [22]:

```python
fig,ax = plt.subplots(4,3,figsize=(6,7))
arr = [i for i in range(len(train_data.loc[:,'label'].unique()))]
ax = ax.flatten()
for i in range(12):
    ax[i].imshow(train_data.iloc[i,1:].to_numpy().reshape(28,28),cmap='gray')
    ax[i].set_title(f'Digit : {train_data.iloc[i,0]}')
```



In [23]:

```python
import tensorflow as tf
from tensorflow.keras.layers import Dense, Conv2D, MaxPooling2D, Dropout, Flatten
from sklearn.model_selection import train_test_split
```

In [24]:

```python
input_data= train_data.drop(['label'],axis=1)
target= train_data.label
```

In [25]:

```python
plt.figure(figsize=(15,8))
sns.countplot(x=train_data.loc[:,'label'], data=train_data.loc[:,'label'].dropna())
plt.xticks(rotation=90)
plt.show()
```

In [26]:

```python
x_train,x_test,y_train,y_test = train_test_split(input_data,target,test_size=0.3)
```

In [27]:

```python
y_train_enc = tf.keras.utils.to_categorical(y_train)
y_test_enc = tf.keras.utils.to_categorical(y_test)
```

In [28]:

```python
y_train_enc[0]
```

Out[28]:

```
array([0., 0., 0., 0., 0., 0., 0., 1., 0., 0.], dtype=float32)
```

In [29]:

```python
x_train = x_train.values.reshape(-1,28,28,1)
x_test = x_test.values.reshape(-1,28,28,1)
```

In [31]:

```python
def model():
    nn = tf.keras.models.Sequential()
    nn.add(Conv2D(64,kernel_size=(3,3),input_shape=(28,28,1),activation='relu'))
    nn.add(Conv2D(64,kernel_size=(3,3),activation='relu'))
    nn.add(MaxPooling2D(pool_size=(2,2)))
    nn.add(Dropout(0.2))
    nn.add(Conv2D(128,kernel_size=(3,3),activation='relu'))
    nn.add(Conv2D(128,kernel_size=(3,3),activation='relu'))
    nn.add(MaxPooling2D(pool_size=(2,2)))
    nn.add(Dropout(0.2))
    nn.add(Conv2D(256,kernel_size=(3,3),activation='relu'))
    nn.add(Flatten())
    nn.add(Dense(10,activation='softmax'))
    nn.compile(optimizer='Adam',loss='categorical_crossentropy',metrics=['accuracy'])
    return nn
```

In [32]:

```python
model = model()
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 64)        640

 conv2d_1 (Conv2D)           (None, 24, 24, 64)        36928

 max_pooling2d (MaxPooling2D  (None, 12, 12, 64)       0
 )

 dropout (Dropout)           (None, 12, 12, 64)        0

 conv2d_2 (Conv2D)           (None, 10, 10, 128)       73856

 conv2d_3 (Conv2D)           (None, 8, 8, 128)         147584

 max_pooling2d_1 (MaxPooling  (None, 4, 4, 128)        0
 2D)

 dropout_1 (Dropout)         (None, 4, 4, 128)         0

 conv2d_4 (Conv2D)           (None, 2, 2, 256)         295168

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 10)                10250

=================================================================
Total params: 564,426
Trainable params: 564,426
Non-trainable params: 0
_________________________________________________________________
```

```
2022-08-03 19:24:47.839214: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is opti
mized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-cri
tical operations:  AVX2 AVX512F FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
```

In [33]:

```python
model.fit(x_train,y_train_enc,epochs=20)
```

```
Epoch 1/20
919/919 [==============================] - 66s 71ms/step - loss: 0.3240 - accuracy: 0.9159
Epoch 2/20
919/919 [==============================] - 67s 73ms/step - loss: 0.0792 - accuracy: 0.9756
Epoch 3/20
919/919 [==============================] - 65s 70ms/step - loss: 0.0613 - accuracy: 0.9810
Epoch 4/20
919/919 [==============================] - 67s 73ms/step - loss: 0.0517 - accuracy: 0.9834
Epoch 5/20
919/919 [==============================] - 67s 73ms/step - loss: 0.0422 - accuracy: 0.9875
Epoch 6/20
919/919 [==============================] - 64s 69ms/step - loss: 0.0451 - accuracy: 0.9854
Epoch 7/20
919/919 [==============================] - 66s 72ms/step - loss: 0.0380 - accuracy: 0.9883
Epoch 8/20
919/919 [==============================] - 66s 72ms/step - loss: 0.0325 - accuracy: 0.9902
Epoch 9/20
919/919 [==============================] - 66s 72ms/step - loss: 0.0325 - accuracy: 0.9902
Epoch 10/20
919/919 [==============================] - 71s 77ms/step - loss: 0.0319 - accuracy: 0.9909
Epoch 11/20
919/919 [==============================] - 75s 81ms/step - loss: 0.0303 - accuracy: 0.9907
Epoch 12/20
919/919 [==============================] - 70s 77ms/step - loss: 0.0242 - accuracy: 0.9927
Epoch 13/20
919/919 [==============================] - 70s 77ms/step - loss: 0.0282 - accuracy: 0.9924
Epoch 14/20
919/919 [==============================] - 71s 77ms/step - loss: 0.0295 - accuracy: 0.9915
Epoch 15/20
919/919 [==============================] - 71s 77ms/step - loss: 0.0275 - accuracy: 0.9927
Epoch 16/20
919/919 [==============================] - 70s 76ms/step - loss: 0.0239 - accuracy: 0.9927
Epoch 17/20
919/919 [==============================] - 70s 77ms/step - loss: 0.0264 - accuracy: 0.9932
Epoch 18/20
919/919 [==============================] - 69s 76ms/step - loss: 0.0270 - accuracy: 0.9926
Epoch 19/20
919/919 [==============================] - 70s 77ms/step - loss: 0.0210 - accuracy: 0.9939
Epoch 20/20
919/919 [==============================] - 77s 83ms/step - loss: 0.0260 - accuracy: 0.9930
```

Out[33]:

```
<keras.callbacks.History at 0x7fc523c1c430>
```

In [34]:

```python
loss, acc = model.evaluate(x_test,y_test_enc)
```

```
394/394 [==============================] - 6s 14ms/step - loss: 0.0732 - accuracy: 0.9891
```

In [35]:

```python
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 64)        640

 conv2d_1 (Conv2D)           (None, 24, 24, 64)        36928

 max_pooling2d (MaxPooling2D  (None, 12, 12, 64)       0
 )

 dropout (Dropout)           (None, 12, 12, 64)        0

 conv2d_2 (Conv2D)           (None, 10, 10, 128)       73856

 conv2d_3 (Conv2D)           (None, 8, 8, 128)         147584

 max_pooling2d_1 (MaxPooling  (None, 4, 4, 128)        0
 2D)

 dropout_1 (Dropout)         (None, 4, 4, 128)         0

 conv2d_4 (Conv2D)           (None, 2, 2, 256)         295168

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 10)                10250

=================================================================
Total params: 564,426
Trainable params: 564,426
Non-trainable params: 0
_________________________________________________________________
```

In [36]:

```python
print('Loss = ',loss)
print('Accuracy = ',acc*100)
```

```
Loss =  0.07320599257946014
Accuracy =  98.9126980304718
```

In [37]:

```python
pred = model.predict(test_data.values.reshape(-1,28,28,1))
pred.shape
```

Out[37]:

```
(28000, 10)
```

In [39]:

```python
preds = []
for i in range(28000):
    preds.append(np.argmax(pred[i]))
```

In [40]:

```python
sub = pd.read_csv('sample_submission.csv')
sub['Label'] = preds
```

In [41]:

```python
sub
```

Out[41]:

	ImageId	Label
0	1	2
1	2	0
2	3	9
3	4	0
4	5	3
...	...	...
27995	27996	9
27996	27997	7
27997	27998	3
27998	27999	9
27999	28000	2

28000 rows × 2 columns

In [42]:

```python
sub.to_csv('submission.csv',index=False)
```

In [ ]: