In [2]:

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

In [3]:

```
train_data = pd.read_csv('digit_train.csv')
test_data = pd.read_csv('digit_test.csv')
```

In [4]:

```
train_data.head()
```

Out[4]:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel
0	1	0	0	0	0	0	0	0	0	0	...	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	
2	1	0	0	0	0	0	0	0	0	0	...	0	
3	4	0	0	0	0	0	0	0	0	0	...	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	

5 rows × 785 columns

In [5]:

```
train_data.tail()
```

Out[5]:

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

5 rows × 785 columns

In [6]:

```
test_data.head()
```

Out[6]:

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

5 rows × 784 columns

In [7]:

```
test_data.tail()
```

Out[7]:

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

5 rows × 784 columns

In [8]:

```
train_data.shape
```

Out[8]:

```
(42000, 785)
```

In [9]:

```
test_data.shape
```

Out[9]:

```
(28000, 784)
```

In [10]:

```
train_data.columns
```

Out[10]:

```
Index(['label', 'pixel0', 'pixel1', 'pixel2', 'pixel3', 'pixel4', 'pixel5',
       'pixel6', 'pixel7', 'pixel8',
       ...
       'pixel774', 'pixel775', 'pixel776', 'pixel777', 'pixel778', 'pixel77
9',
       'pixel780', 'pixel781', 'pixel782', 'pixel783'],
      dtype='object', length=785)
```

In [11]:

```
test_data.columns
```

Out[11]:

```
Index(['pixel0', 'pixel1', 'pixel2', 'pixel3', 'pixel4', 'pixel5', 'pixel6',
       'pixel7', 'pixel8', 'pixel9',
       ...
       'pixel774', 'pixel775', 'pixel776', 'pixel777', 'pixel778', 'pixel77
9',
       'pixel780', 'pixel781', 'pixel782', 'pixel783'],
      dtype='object', length=784)
```

In [12]:

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

Out[12]:

```
0
```

In [13]:

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

Out[13]:

```
0
```

In [14]:

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

Out[14]:

```
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 [15]:

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

Out[15]:

```
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 [16]:

```
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 [17]:

```
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 [18]:

```
train_data.describe()
```

Out[18]:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	4
count	42000.000000	42000.0	42000.0	42000.0	42000.0	42000.0	42000.0	42000.0	42000.0	
mean	4.456643	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
std	2.887730	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
min	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
25%	2.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
50%	4.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
75%	7.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
max	9.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

8 rows × 785 columns

In [19]:

```
test_data.describe()
```

Out[19]:

	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9
count	28000.0	28000.0	28000.0	28000.0	28000.0	28000.0	28000.0	28000.0	28000.0	28000.0
mean	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
std	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
min	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
25%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
50%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
75%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
max	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

8 rows × 784 columns

In [20]:

```
train_data.nunique()
```

Out[20]:

```
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 [21]:

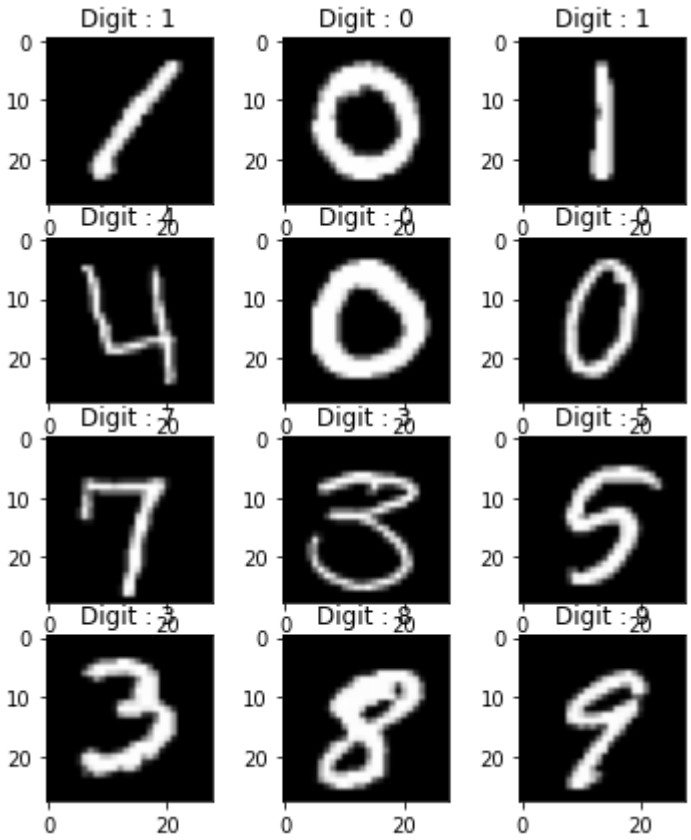
```
test_data.nunique()
```

Out[21]:

```
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 [23]:

```
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 [24]:

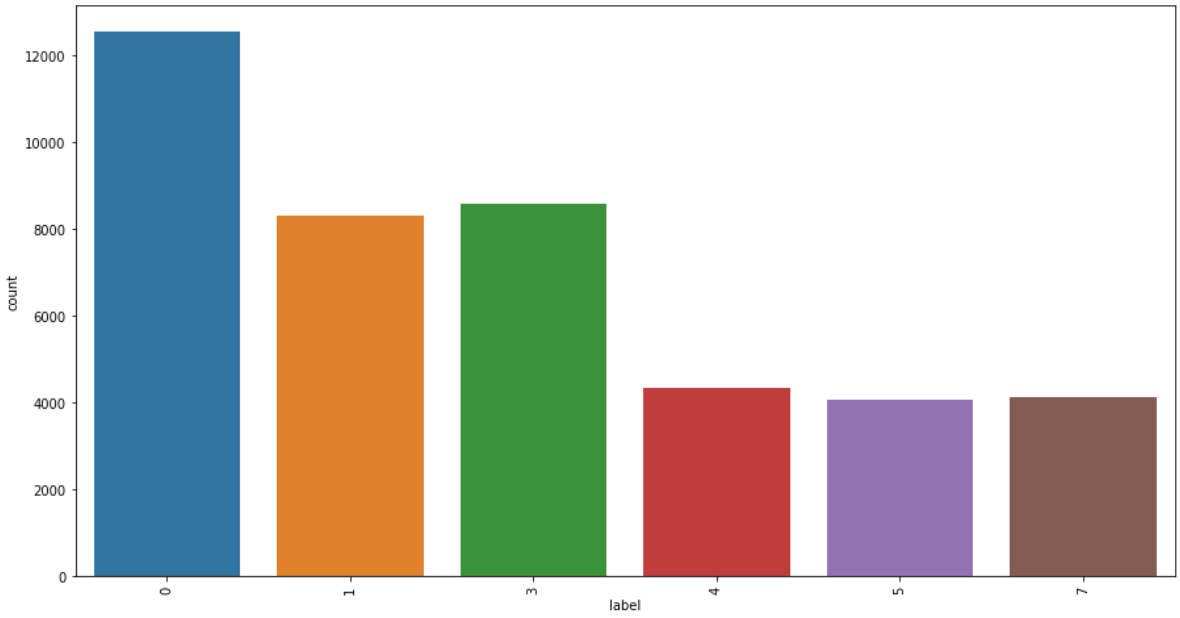
```
import tensorflow as tf
from tensorflow.keras.layers import Dense, Conv2D, MaxPooling2D, Dropout, Flatten
from sklearn.model_selection import train_test_split
```

In [25]:

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

In [26]:

```
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 [27]:

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

In [28]:

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

In [29]:

```
y_train_enc[0]
```

Out[29]:

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

In [30]:

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

In [31]:

```
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]:

```
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
_________________________________________________________________
```

In [33]:

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

```
Epoch 1/20
919/919 [==============================] - 66s 69ms/step - loss: 0.2962 - ac
curacy: 0.9270
Epoch 2/20
919/919 [==============================] - 68s 74ms/step - loss: 0.0784 - ac
curacy: 0.9755
Epoch 3/20
919/919 [==============================] - 66s 71ms/step - loss: 0.0570 - ac
curacy: 0.9823
Epoch 4/20
919/919 [==============================] - 82s 90ms/step - loss: 0.0523 - ac
curacy: 0.9835
Epoch 5/20
919/919 [==============================] - 89s 97ms/step - loss: 0.0484 - ac
curacy: 0.9863
Epoch 6/20
919/919 [==============================] - 92s 100ms/step - loss: 0.0446 - a
ccuracy: 0.9863
Epoch 7/20
919/919 [==============================] - 191s 208ms/step - loss: 0.0400 -
accuracy: 0.9876
Epoch 8/20
919/919 [==============================] - 210s 228ms/step - loss: 0.0365 -
accuracy: 0.9885
Epoch 9/20
919/919 [==============================] - 209s 228ms/step - loss: 0.0333 -
accuracy: 0.9899
Epoch 10/20
919/919 [==============================] - 210s 229ms/step - loss: 0.0309 -
accuracy: 0.9907
Epoch 11/20
919/919 [==============================] - 219s 239ms/step - loss: 0.0343 -
accuracy: 0.9901
Epoch 12/20
919/919 [==============================] - 207s 226ms/step - loss: 0.0282 -
accuracy: 0.9913
Epoch 13/20
919/919 [==============================] - 213s 231ms/step - loss: 0.0295 -
accuracy: 0.9914
Epoch 14/20
919/919 [==============================] - 2731s 3s/step - loss: 0.0201 - ac
curacy: 0.9943
Epoch 15/20
919/919 [==============================] - 410s 446ms/step - loss: 0.0335 -
accuracy: 0.9909
Epoch 16/20
919/919 [==============================] - 211s 230ms/step - loss: 0.0222 -
accuracy: 0.9933
Epoch 17/20
919/919 [==============================] - 210s 229ms/step - loss: 0.0288 -
accuracy: 0.9920
Epoch 18/20
919/919 [==============================] - 607s 662ms/step - loss: 0.0217 -
accuracy: 0.9944
Epoch 19/20
919/919 [==============================] - 209s 227ms/step - loss: 0.0283 -
accuracy: 0.9923
```

```
Epoch 20/20
919/919 [==============================] - 208s 226ms/step - loss: 0.0233 -
accuracy: 0.9938
```

Out[33]:

```
<keras.callbacks.History at 0x23b703321f0>
```

In [34]:

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

```
394/394 [==============================] - 27s 63ms/step - loss: 0.0626 - ac
curacy: 0.9890
```

In [35]:

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

```
Loss =  0.06256796419620514
Accuracy =  98.9047646522522
```

In [36]:

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

```
875/875 [==============================] - 45s 51ms/step
```

Out[36]:

```
(28000, 10)
```

In [37]:

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

In [38]:

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

In [39]:

```
sub
```

Out[39]:

	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 [40]:

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