In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [6]:
```
df = pd.read_csv("train.csv")
df.shape
```

Out[6]: (42000, 785)

In [7]:
```
df
```

Out[7]:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781
0	1	0	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	0	0	0
2	1	0	0	0	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	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
41995	0	0	0	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	0	0	0
41997	7	0	0	0	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	0	0	0
41999	9	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0

42000 rows × 785 columns

In [8]:
```
x = df.iloc[:,1:]
y = df.iloc[:,0]
```

In [10]:
```
y
```

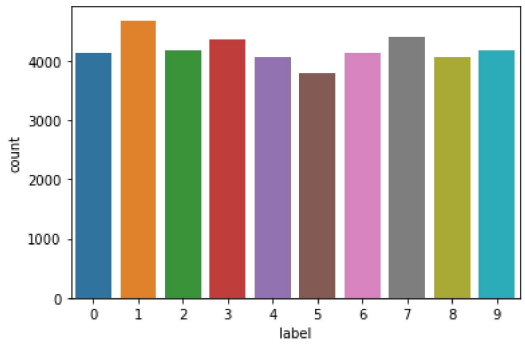
Out[10]:
```
0        1
1        0
2        1
3        4
4        0
        ..
41995    0
41996    1
41997    7
41998    6
41999    9
Name: label, Length: 42000, dtype: int64
```

In [11]:
```
y
```

Out[11]:
```
0        1
1        0
2        1
3        4
4        0
        ..
41995    0
41996    1
41997    7
41998    6
41999    9
Name: label, Length: 42000, dtype: int64
```

In [13]:
```
sns.countplot(data= df,x=y)
```

Out[13]: <AxesSubplot:xlabel='label', ylabel='count'>



In [16]:
```
x.sample(1)
```

Out[16]:

	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel774	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781
244	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0

1 rows × 784 columns

In [24]:
```
plt.imshow(x.iloc[244:255,:].values.reshape(28,28))
plt.title(f"Number is {y[244]}")
```

...

In [22]:
```
x.sample()
```

Out[22]:

	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel774	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781
5483	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0

1 rows × 784 columns

In [33]:
```
plt.figure(figsize=(15,10))
for i in range(5):
    plt.subplot(2,5,i+1)
    plt.imshow(x.iloc[i,:].values.reshape(28,28),cmap='gray')
plt.show()
```

...

In [34]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3, random_state=0)
```

In [35]:
```
x_train.shape
```

Out[35]: (29400, 784)

In [36]:
```
x_test.shape
```

Out[36]: (12600, 784)

In [37]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [38]:
```
knn = KNeighborsClassifier()
```

In [40]:
```
knn.fit(x_train,y_train)
```

Out[40]: KNeighborsClassifier()

In [43]:
```
y_pred = knn.predict(x_test)
```

In [44]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[44]:
```
0.9657142857142857
```

In [45]:
```
from sklearn.preprocessing import StandardScaler
```

In [48]:
```
std = StandardScaler()
x_train_trf = std.fit_transform(x_train)
x_test_trf = std.transform(x_test)
```

In [50]:
```
from sklearn.decomposition import PCA
```

In [58]:
```
pca = PCA(n_components=100)
```

In [59]:
```
x_train_pca  = pca.fit_transform(x_train_trf)
x_test_pca = pca.transform(x_test)
```

In [60]:
```
x_train_pca.shape
```

Out[60]:
```
(29400, 100)
```

In [65]:
```
x_test_pca.shape
```

Out[65]:
```
(12600, 100)
```

In [69]:
```
knn_pca = KNeighborsClassifier()
knn_pca.fit(x_train_pca,y_train)
```

Out[69]:
```
KNeighborsClassifier()
```

In [72]:
```
y_pred_pca = knn.predict(x_test_pca)
```

Out[72]:
```
array([0, 6, 9, ..., 1, 6, 0], dtype=int64)
```

In [73]:
```
accuracy_score(y_test,y_pred_pca)
```

Out[73]:
```
0.7673809523809524
```

In [77]:
```
for i in range(1,785):
    pca = PCA(n_components=i)
    x_train_pca  = pca.fit_transform(x_train_trf)
    x_test_pca = pca.transform(x_test)
    knn.fit(x_train_pca,y_train)
    y_pred_pca = knn.predict(x_test_pca)
    print(f"Iteration : {i} {accuracy_score(y_test,y_pred_pca)}")
```

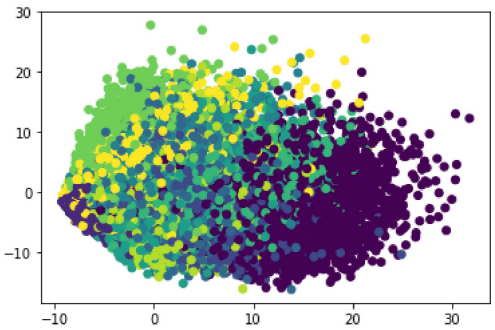
```
Iteration : 1 0.18611111111111112
Iteration : 2 0.1353968253968254
Iteration : 3 0.23103174603174603
Iteration : 4 0.2594444444444444
Iteration : 5 0.363015873015873
Iteration : 6 0.4015873015873016
Iteration : 7 0.43047619047619046
Iteration : 8 0.47095238095238096
Iteration : 9 0.4723809523809524
Iteration : 10 0.4635714285714286
Iteration : 11 0.468015873015873
Iteration : 12 0.48404761904761906
Iteration : 13 0.5506349206349206
Iteration : 14 0.5697619047619048
Iteration : 15 0.5819841269841269
Iteration : 16 0.5842063492063492
Iteration : 17 0.5814285714285714
Iteration : 18 0.5795238095238096
Iteration : 19 0.5876190476190476
```

In [79]:
```
pca_dim = PCA(n_components=2)
x_train_pca  = pca_dim.fit_transform(x_train_trf)
x_test_pca = pca_dim.transform(x_test)
x_train_pca.shape
```

Out[79]:
```
(29400, 2)
```

In [87]:
```
#import plotly.express as px
plt.scatter(x_train_pca[:,0],x_train_pca[:,1],c=y_train)
```

Out[87]: <matplotlib.collections.PathCollection at 0x1948f6be250>




```
Collecting plotly
  Downloading plotly-5.13.0-py2.py3-none-any.whl (15.2 MB)
Collecting tenacity>=6.2.0
  Downloading tenacity-8.1.0-py3-none-any.whl (23 kB)
Installing collected packages: tenacity, plotly
Successfully installed plotly-5.13.0 tenacity-8.1.0
Collecting plotly
  Downloading plotly-5.13.0-py2.py3-none-any.whl (15.2 MB)
Collecting tenacity>=6.2.0
  Using cached tenacity-8.1.0-py3-none-any.whl (23 kB)
Installing collected packages: tenacity, plotly
Successfully installed plotly-5.13.0 tenacity-8.1.0
```

In [83]:
```
!pip install plotly
```

```
^C
```

In [ ]:
```
```