In [1]:

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
df=pd.read_csv("train.csv")
df.head()
```

Out[2]:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pi
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	

5 rows × 785 columns

In [3]:

```
x=df.iloc[:,1:]
x
```

Out[3]:

	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel774	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel78
0	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	
2	0	0	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	0	0	
4	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	
41996	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
41997	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
41998	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
41999	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	

42000 rows × 784 columns

In [4]:

```
y=df.iloc[:,0]
y
```

Out[4]:

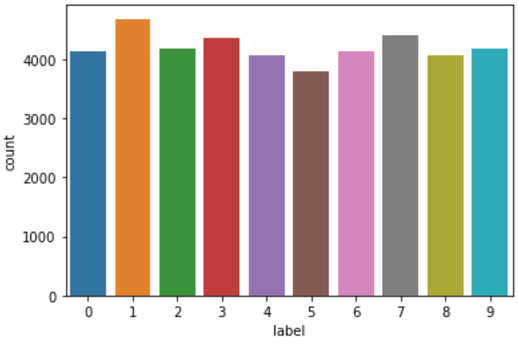
```
0        1
1        0
2        1
3        4
4        0
        ..
41995    0
41996    1
41997    7
41998    6
41999    9
Name: label, Length: 42000, dtype: int64
```

In [5]:

```
import seaborn as sns
```

```
In [6]: sns.countplot(data=df,x=y)
```

```
Out[6]: <AxesSubplot:xlabel='label', ylabel='count'>
```



```
In [7]: df.sample(1)
```

Out[7]:

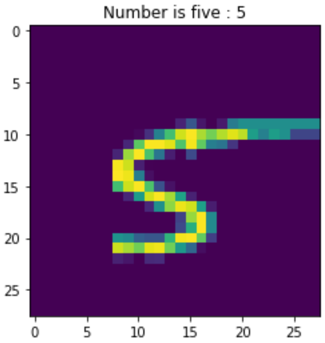
	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781
5434	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0

1 rows × 785 columns

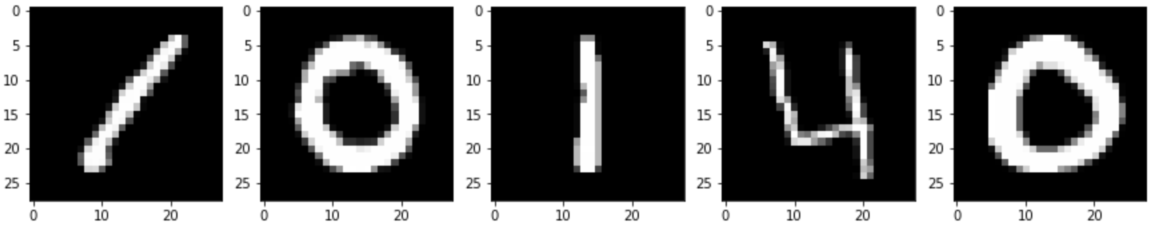
```
In [8]: import matplotlib.pyplot as plt
```

```
In [9]: plt.imshow(x.iloc[244,:].values.reshape(28,28))
        plt.title(f"Number is five : {y[244]}")
```

```
Out[9]: Text(0.5, 1.0, 'Number is five : 5')
```



```
In [10]: 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 [11]: 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 [12]: x_train.shape
```

```
Out[12]: (29400, 784)
```

```
In [13]: x_test.shape
```

```
Out[13]: (12600, 784)
```

```
In [14]: from sklearn.neighbors import KNeighborsClassifier
```

```
In [15]: knn=KNeighborsClassifier()
```

```
In [16]: knn.fit(x_train,y_train)
```

```
Out[16]: KNeighborsClassifier()
```

```
In [17]: y_pred=knn.predict(x_test)
```

```
In [18]: from sklearn.metrics import accuracy_score
         accuracy_score(y_test,y_pred)
```

```
Out[18]: 0.9657142857142857
```

```
In [19]: from sklearn.preprocessing import StandardScaler
```

```
In [20]: std=StandardScaler()
```

```
In [21]: x_train_trf=std.fit_transform(x_train)
         x_test_trf=std.fit_transform(x_test)
```

```
In [22]: x_train_trf
```

```
Out[22]: array([[0., 0., 0., ..., 0., 0., 0.],
                [0., 0., 0., ..., 0., 0., 0.],
                [0., 0., 0., ..., 0., 0., 0.],
                ...,
                [0., 0., 0., ..., 0., 0., 0.],
                [0., 0., 0., ..., 0., 0., 0.],
                [0., 0., 0., ..., 0., 0., 0.]])
```

```
In [23]: from sklearn.decomposition import PCA
```

```
In [24]: pca= PCA(n_components=100)
```

```
In [25]: x_train_pca=pca.fit_transform(x_train_trf)
         x_test_pca=pca.transform(x_test)
```

```
In [26]: x_train_pca.shape
```

```
Out[26]: (29400, 100)
```

```
In [27]: knn_pca=KNeighborsClassifier()
         knn_pca.fit(x_train_pca,y_train)
```

```
Out[27]: KNeighborsClassifier()
```

```
In [28]: y_pred_pca= knn_pca.predict(x_test_pca)
```

```
In [29]: accuracy_score(y_test,y_pred_pca)
```

```
Out[29]: 0.7686507936507937
```

In [30]:
```
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)}")
```

```
    608
    609 Callback to :func:`sklearn.metrics.pairwise.pairwise_distances_chunked`
   (...)
    631     The neighbors indices.
    632 """
    633 sample_range = np.arange(dist.shape[0])[:, None]
--> 634 neigh_ind = np.argpartition(dist, n_neighbors - 1, axis=1)
    635 neigh_ind = neigh_ind[:, :n_neighbors]
    636 # argpartition doesn't guarantee sorted order, so we sort again

File <__array_function__ internals>:5, in argpartition(*args, **kwargs)

File ~\anaconda3\lib\site-packages\numpy\core\fromnumeric.py:839, in argpartition(a, kth, axis, kind, order)
    763 @array_function_dispatch(_argpartition_dispatcher)
    764 def argpartition(a, kth, axis=-1, kind='introselect', order=None):
    765     """
    766     Perform an indirect partition along the given axis using the
    767     algorithm specified by the `kind` keyword. It returns an array of
   (...)
    837
```

In [37]:
```
pca_dim=PCA(n_components=3)
x_train_pca= pca_dim.fit_transform(x_train_trf)
x_test_pca=pca_dim.transform(x_test)
x_train_pca.shape
```

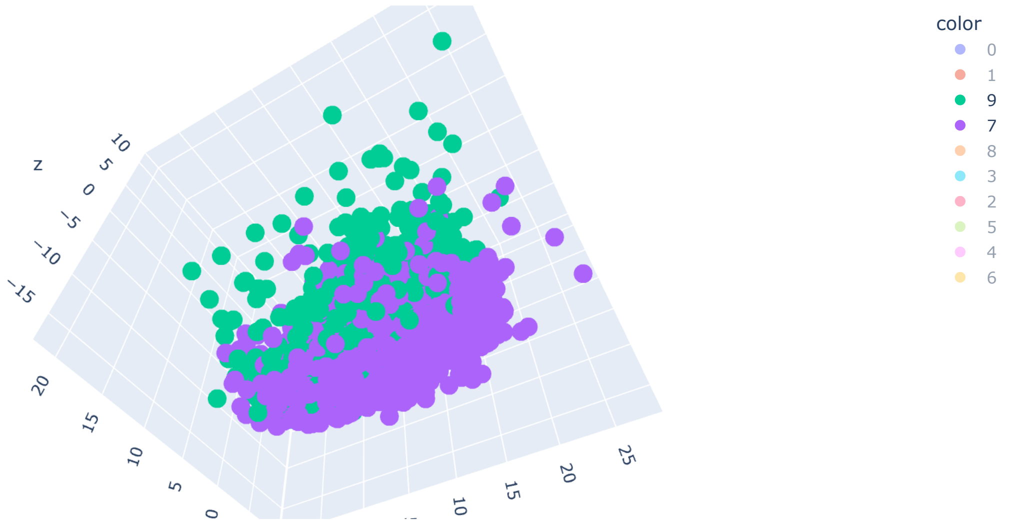
Out[37]:
```
(29400, 3)
```

In [38]:
```
!pip install plotly
```

```
Requirement already satisfied: plotly in c:\users\user20\anaconda3\lib\site-packages (5.6.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\user20\anaconda3\lib\site-packages (from plotly) (8.0.1)
Requirement already satisfied: six in c:\users\user20\anaconda3\lib\site-packages (from plotly) (1.16.0)
```

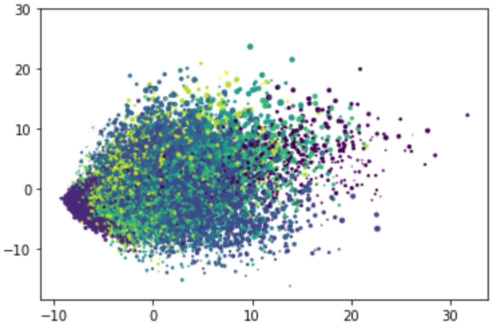
In [42]:
```
d=y_train.astype(str)
```

In [44]:
```
import plotly.express as px
px.scatter_3d(x=x_train_pca[:,0],y=x_train_pca[:,1],z=x_train_pca[:,2],color=d)
```



```
In [45]: plt.scatter(x_train_pca[:,0],x_train_pca[:,1],x_train_pca[:,2],c=y_train)

Out[45]: <matplotlib.collections.PathCollection at 0x25d408b3f70>
```



```
In [46]: pca.explained_variance_

Out[46]: array([40.59588171, 29.31939629, 26.705399  , 20.79061607, 18.0643079 ,
                15.75866135, 13.9306296 , 12.52037795, 11.12731117, 10.08218143,
                 9.71768038,  8.7115118 ,  8.02335861,  7.89624998,  7.48525489,
                 7.1691588 ,  6.73477221,  6.64457689,  6.48382311,  6.33577897,
                 5.956529  ,  5.78179099,  5.58505005,  5.34889243,  5.1959452 ,
                 4.95422385,  4.91988014,  4.76273887,  4.5199611 ,  4.42355281,
                 4.36165595,  4.32013579,  4.16712305,  4.07220011,  4.02333094,
                 3.93391034,  3.81037088,  3.7162528 ,  3.63304523,  3.53297947,
                 3.46010138,  3.41856809,  3.31344851,  3.22330275,  3.1967853 ,
                 3.16744218,  3.13107582,  3.05360221,  3.01379846,  2.93730978,
                 2.8794287 ])
```

```
In [48]: pca.explained_variance_ratio_*100

Out[48]: array([5.84935171, 4.22455317, 3.84790932, 2.99566411, 2.60283768,
                2.27062325, 2.00722706, 1.80402768, 1.60330442, 1.45271448,
                1.4001945 , 1.25521837, 1.15606422, 1.13774948, 1.0785303 ,
                1.03298486, 0.97039526, 0.95739926, 0.93423668, 0.9129054 ,
                0.85826029, 0.83308276, 0.80473488, 0.77070756, 0.74866981,
                0.71384083, 0.70889233, 0.68625027, 0.65126907, 0.63737786,
                0.62845931, 0.62247678, 0.60042958, 0.58675239, 0.57971096,
                0.56682659, 0.54902612, 0.5354649 , 0.52347574, 0.50905754,
                0.49855673, 0.49257231, 0.47742591, 0.46443705, 0.46061622,
                0.45638825, 0.45114832, 0.43998535, 0.43425014, 0.42322909,
                0.41488916])
```

```
In [ ]: 
```