```
# Step 1:---->Importing all the required library and packages


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt


# Step 2:----->Importing the dataset as pandas -dataframe and then preparing datase


df = pd.read_csv("https://raw.githubusercontent.com/coding-blocks-archives/machine-
print(df.shape)
print(df.head())
```

```
(42000, 785)
   label  pixel0  pixel1  pixel2  ...  pixel780  pixel781  pixel782  pixel783
0      1       0       0       0  ...         0         0         0         0
1      0       0       0       0  ...         0         0         0         0
2      1       0       0       0  ...         0         0         0         0
3      4       0       0       0  ...         0         0         0         0
4      0       0       0       0  ...         0         0         0         0

[5 rows x 785 columns]
```

```
data = df.values
print(data.shape)
print(type(data))
```

```
(42000, 785)
<class 'numpy.ndarray'>
```

```
X = data[:,1:]
Y = data[:,0]
print(X.shape,Y.shape)
```

```
(42000, 784) (42000,)
```

```
# Step 3:--->Splitting the dataset into training and testing dataset


split = int(0.80*X.shape[0])
print(split)
```

```
33600
```

```
X_tr,Y_tr = X[:split,:] , Y[:split]
print(X_tr.shape ,Y_tr.shape)
```

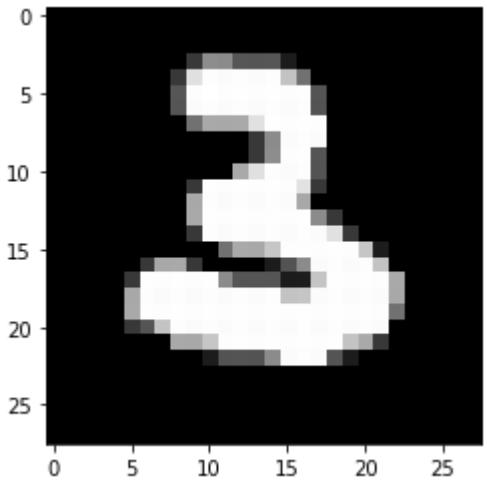
```
(33600, 784) (33600,)
```

```
X_te,Y_te = X[split:,:],Y[split:]
print(X_te.shape,Y_te.shape)
```

```
(8400, 784) (8400,)
```

```
def drawImg(sample):
  img = sample.reshape((28,28))
  plt.imshow(img,cmap='gray')
  plt.show()

# Step 4:------>The KNN Algorithm

def dist(x1,x2):
  return np.sqrt(sum((x1-x2)**2))

def knn(X,Y,q_pt,k=5):
  vals = []
  m = X.shape[0]
  d = 0.0
  for i in range(m):
    #print(X[i])
    d = dist(X[i],q_pt)
    vals.append((d,Y[i]))
  #print(vals)
  vals = sorted(vals)
  vals = vals[:k]
  #print(vals)
  vals = np.array(vals)
  #print(vals)
  new_vals = np.unique(vals[:,1],return_counts=True)
  #print(new_vals)
  #print(new_vals[0])
  #print(new_vals[1])
  index = new_vals[1].argmax()
  return new_vals[0][index]


# Step 5:---->Make prediction

pred = knn(X_tr,Y_tr,X_te[10])
print(pred)
drawImg(X_te[10])
```

```
3.0
```



```
def Acc_calculator(sample,op):
  m = sample.shape[0]
  m = m//100
  count = 0
  for i in range(m):
    pred = knn(X_tr,Y_tr,sample[i])
    if pred == op[i]:
      count+=1
  acc = (count/m)*100
  print("Accuracy of Knn for MNist dataset = %f"%acc)
```

```
Acc_calculator(X_te,Y_te)
```

```
Accuracy of Knn for MNist dataset = 96.428571
```