KNN on MNIST dataset

```
import tensorflow as tf
from tensorflow.keras.datasets import mnist
import numpy as np
import matplotlib.pyplot as plt
```

```
(X_train, y_train), (X_test, y_test)= mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
```

```
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(60000, 28, 28)
(60000,)
(10000, 28, 28)
(10000,)
```

```
plt.imshow(X_test[0])
```

```
<matplotlib.image.AxesImage at 0x7f00f27a0a20>
```


```
X_train= X_train.reshape(-1, 28*28)
X_test= X_test.reshape(-1, 28*28)
```

```
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(60000, 784)
(60000,)
(10000, 784)
(10000,)
```

```
train_digits= tf.Variable(X_train, dtype="float", shape=[None, 784])
test_digits= tf.Variable(X_test[0], dtype="float", shape=[784]) # only the first test digit in X_test
```

```
print(train_digits.shape)
print(test_digits.shape)
```

```
(None, 784)
(784,)
```

```
def kNearestNeighbour(test_digits):
  l1_distance = tf.abs(tf.add(train_digits,tf.negative(test_digits)))
  dist= tf.reduce_sum(l1_distance,axis=1)
  return np.array(tf.argsort(dist))
```

```
classes=[0,1,2,3,4,5,6,7,8,9]
```

```
test_digits.assign(X_test[0])
sort_indices= kNearestNeighbour(test_digits)

print(sort_indices)
print(y_test[0])
print(len(sort_indices))
```

```
[53843 27059 38620 ... 25321 59439 41358]
7
60000
```

```
k=int(input("Enter the value of k:"))
correct=0
for i in range(100): #for 1st 100 test digits
  ind=[0,0,0,0,0,0,0,0,0,0]
  test_digits.assign(X_test[i,:])
  indices = kNearestNeighbour(test_digits)
  labels=[]
  for j in range(k):
    labels.append(y_train[indices[j]])
  for l in labels:
    if l==0:
      ind[0]=ind[0]+1
    elif l==1:
      ind[1]=ind[1]+1
    elif l==2:
      ind[2]=ind[2]+1
    elif l==3:
      ind[3]=ind[3]+1
    elif l==4:
      ind[4]=ind[4]+1
    elif l==5:
      ind[5]=ind[5]+1
    elif l==6:
      ind[6]=ind[6]+1
    elif l==7:
      ind[7]=ind[7]+1
    elif l==8:
      ind[8]=ind[8]+1
```

```
    ind[8]=ind[8]+1
  else:
    ind[9]=ind[9]+1
print("Epoch: ",(i+1)," Predicted: ", classes[np.argmax(ind)], " actual: ",y_test[i])
if classes[np.argmax(ind)]==y_test[i]:
  correct+= 1
print("correctly predicted: ",correct)
```

```
correctly predicted:  69
Epoch:  72  Predicted:  0  actual:  0
correctly predicted:  70
Epoch:  73  Predicted:  2  actual:  2
correctly predicted:  71
Epoch:  74  Predicted:  9  actual:  9
correctly predicted:  72
Epoch:  75  Predicted:  1  actual:  1
correctly predicted:  73
Epoch:  76  Predicted:  7  actual:  7
correctly predicted:  74
Epoch:  77  Predicted:  3  actual:  3
correctly predicted:  75
Epoch:  78  Predicted:  2  actual:  2
correctly predicted:  76
Epoch:  79  Predicted:  9  actual:  9
correctly predicted:  77
Epoch:  80  Predicted:  7  actual:  7
correctly predicted:  78
Epoch:  81  Predicted:  7  actual:  7
correctly predicted:  79
Epoch:  82  Predicted:  6  actual:  6
correctly predicted:  80
Epoch:  83  Predicted:  2  actual:  2
correctly predicted:  81
Epoch:  84  Predicted:  7  actual:  7
correctly predicted:  82
Epoch:  85  Predicted:  8  actual:  8
correctly predicted:  83
Epoch:  86  Predicted:  4  actual:  4
correctly predicted:  84
Epoch:  87  Predicted:  7  actual:  7
correctly predicted:  85
Epoch:  88  Predicted:  3  actual:  3
correctly predicted:  86
```

```
Epoch:  89  Predicted:  6  actual:  6
correctly predicted:  87
Epoch:  90  Predicted:  1  actual:  1
correctly predicted:  88
Epoch:  91  Predicted:  3  actual:  3
correctly predicted:  89
Epoch:  92  Predicted:  6  actual:  6
correctly predicted:  90
Epoch:  93  Predicted:  9  actual:  9
correctly predicted:  91
Epoch:  94  Predicted:  3  actual:  3
correctly predicted:  92
Epoch:  95  Predicted:  1  actual:  1
correctly predicted:  93
Epoch:  96  Predicted:  4  actual:  4
correctly predicted:  94
Epoch:  97  Predicted:  1  actual:  1
correctly predicted:  95
Epoch:  98  Predicted:  7  actual:  7
correctly predicted:  96
Epoch:  99  Predicted:  6  actual:  6
correctly predicted:  97
Epoch:  100  Predicted:  9  actual:  9
correctly predicted:  98
```