1st Assignment - Omar Rashad Salem - prof.Heba Gamal

Question 1

ans: higher `k` value help eleminating outliers in dataset resulting in more accurate prediction, but increases prediction time complexity *i.e.(*`O(n*k)` *instead of* `O(n)`*)*

Question 2

ans: Advantages: 1.constant training time `O(1)` *fast* 2.Easy to implement
Disadvantages: 1. bad with large datasets due to high cost of calc. distances 2.sensitive to noise and missing data

Question 3

b) ans:

- K = 1
  - L1 = 0.219 L2 = 0.219
- K = 3
  - L1 = 0.222 L2 = 0.222
- K = 5
  - L1 = 0.227 L2 = 0.227

c) **ans** test accuracy = 0.2314

Code:

```
!pip install tensorflow
!pip install numpy
!pip install sklearn


# implementation of KNN with L1 , L2 distances

import numpy as np

class KNN:
    def __init__(self, k= 3, l : int = 1):
        self.k = k
        self.l = l

    def train(self, x, y):
        self.xtr = x
```

```
           self.ytr = y

     def predict(self, x):
           num_test = x.shape[0]
           ypred: np.ndarray = np.zeros(num_test , dtype=
self.ytr.dtype)

           if self.l == 1 : # using L1 distance
                for i in range(num_test):
                      dist = np.sum(np.absolute(self.xtr - x[i,:]),
axis= 1)

                      dist_tuples = zip( range(len(dist)), dist )
                      dist_tuples = sorted(dist_tuples , key= lambda a
: a[1] )

                      freq_arr = np.zeros(len(dist))
                      for j in range(self.k): freq_arr[dist_tuples[j]
[0]] += 1 #first element is the org image index

                      kmin_index = np.argmax(freq_arr)
                      ypred[i] = self.ytr[kmin_index]
           else : # using L2 distance
                for i in range(num_test):
                      L2dist = np.sum(np.sqrt( np.power(self.xtr -
x[i,:] , 2) ), axis= 1)

                      L2dist_tuples = zip( range(len(L2dist)),
L2dist )
                      L2dist_tuples = sorted(L2dist_tuples , key=
lambda a : a[1] )

                      freq_arr = np.zeros(len(L2dist))
                      for j in range(self.k):
freq_arr[L2dist_tuples[j][0]] += 1 #first element is the org image
index

                      kmin_index = np.argmax(freq_arr)
                      ypred[i] = self.ytr[kmin_index]



           return ypred

#using KNN model: Define all KNNs
from tensorflow.keras.datasets import cifar10  # ~160mb
from sklearn.model_selection import train_test_split

(xtr, ytr), (xtst, ytst) = cifar10.load_data()
xtr, xval, ytr, yval = train_test_split(xtr, ytr, test_size=0.1,
random_state=42)
```

```
xtr = xtr.reshape(xtr.shape[0],-1) / 255.0
xtr = xtr[:1000]
ytr = ytr[:1000]
xtst = xtst.reshape(xtst.shape[0],-1) / 255.0
xval = xval.reshape(xval.shape[0],-1) / 255.0

ytst = ytst.squeeze()
yval = yval.squeeze()
ytr = ytr.squeeze()

xval = xval[:1000]
yval = yval[:1000]


k1l1nn_obj = KNN(k= 1, l= 1)
k1l2nn_obj = KNN(k= 1, l= 2)

k3l1nn_obj = KNN(k= 3, l= 1)
k3l2nn_obj = KNN(k= 3, l= 2)

k5l1nn_obj = KNN(k= 5, l= 1)
k5l2nn_obj = KNN(k= 5, l= 2)


#QUESTION 3a
#k=1
k1l1nn_obj.train(xtr , ytr)
k1l2nn_obj.train(xtr , ytr)

y11pred = k1l1nn_obj.predict(xval)
y12pred = k1l2nn_obj.predict(xval)

print (f"k = 1 L1 ans = {np.sum(y11pred == yval[:])/len(y11pred)}")
print (f"k = 1 L2 ans = {np.sum(y12pred == yval[:])/len(y12pred)}")
```

```
k = 1 L1 ans = 0.219
k = 1 L2 ans = 0.219
```

```
#k=3
k3l1nn_obj.train(xtr , ytr)
k3l2nn_obj.train(xtr , ytr)

y31pred = k3l1nn_obj.predict(xval)
y32pred = k3l2nn_obj.predict(xval)

print (f"k = 3 L1 ans = {np.sum(y31pred == yval[:])/len(y31pred)}")
print (f"k = 3 L2 ans = {np.sum(y32pred == yval[:])/len(y32pred)}")
```

```
k = 3 L1 ans = 0.222
k = 3 L2 ans = 0.222
```

```
#k=5
k5l1nn_obj.train(xtr , ytr)
k5l2nn_obj.train(xtr , ytr)

y51pred = k5l1nn_obj.predict(xval)
y52pred = k5l2nn_obj.predict(xval)

print (f"k = 5 L1 ans = {np.sum(y51pred == yval[:])/len(y51pred)}")

print (f"k = 5 L2 ans = {np.sum(y52pred == yval[:])/len(y52pred)}")
```

```
k = 5 L1 ans = 0.227
k = 5 L2 ans = 0.227
```

```
#QUESTION 3C

#tst with best model at k=5 L1
y51pred_tst = k5l1nn_obj.predict(xtst)

print (f"Best mode is k=5 L1 ... accuracy score = {np.sum(y51pred_tst
== ytst[:])/len(y51pred_tst)}")
```

```
Best mode is k=5 L1 ... accuracy score = 0.2314
```