

HomeWork04

Problem 1: Use sklearn's implementation of k-Nearest Neighbors for regression purposes, which is found in `sklearn.neighbors.KNeighborsRegressor`. You will find the best value of `k` using 10-fold Cross- Validation (CV), which is found in `sklearn.model_selection.KFold`.

(a) You will modify the python code below to generate 1000 data points, or alternatively you could use part of your semester project dataset if it is related to regression.

Solution:

To generate a set of 1000 data points, `genDataSet(1000)` is to be used instead of `genDataSet(100)`

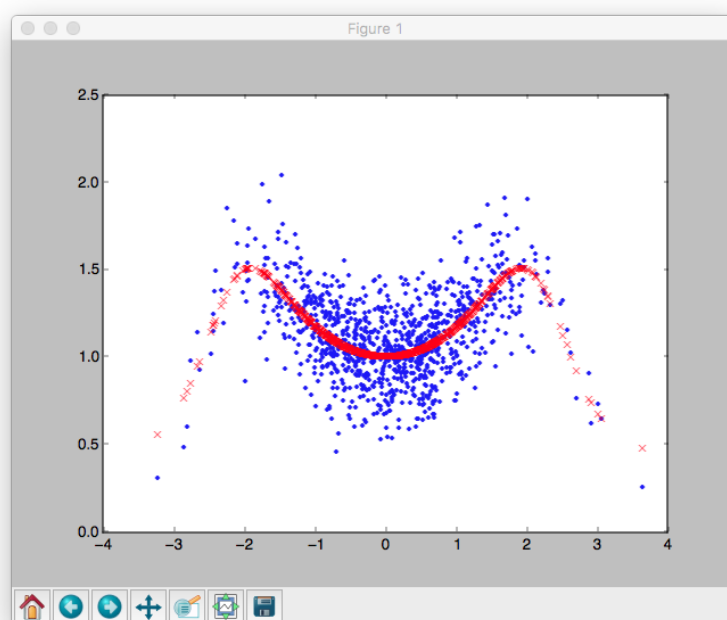
Hence, the program used is

```
import numpy as np
from matplotlib import pyplot as plt

def genDataSet (N) :
    x = np.random.normal(0, 1, N)
    ytrue = (np.cos(x) + 2) / (np.cos(x * 1.4) + 2)
    noise = np.random.normal(0, 0.2, N)
    y = ytrue + noise
    return x, y, ytrue

x ,y, ytrue = genDataSet(1000)
plt.plot(x,y, '.')
plt.plot(x , ytrue , 'rx' )
plt.show()
```

The output of the 1000 data set points program is



(b) Using 10-fold CV, you will report the three best values of k-neighbors that yield the best CV E_{out} . You will vary the values of k in the following range: $k=1,3,5,\dots,2\lceil\frac{N+1}{2}\rceil-1$.

(c) You will report the best CV E_{out} .

Solution:

The program used for these problems is

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn import neighbors
#from numpy import genDataSet
from sklearn.model_selection import KFold
import time

# read digits data & split it into X (training input) and y (target output)
def genDataSet (N) :
    X = np.random.normal(0, 1, N)
    ytrue = (np.cos(X) + 2) / (np.cos(X * 1.4) + 2)
    noise = np.random.normal(0, 0.2, N)
    y = ytrue + noise
    return X, y, ytrue
X ,y, ytrue = genDataSet(1000)
#plt.plot(X,y, '.')
#plt.plot(X , ytrue , 'rx' )
#plt.show()

X = X.reshape(len(X), 1) #reshaping Training dataset
bestk = []
kc = 0

for n_neighbors in range(1,900,2): #90% Training Data
    kf = KFold(n_splits=10)
    #n_neighbors = 85
    kscore=[]
    k=0

    for train, test in kf.split(X):
        #print("%s %s" % (train, test))
        X_train, X_test, y_train, y_test = X[train], X[test], y[train], y[test]

        #time.sleep(100)

        # we create an instance of Neighbors Classifier and fit the data.
        clf = neighbors.KNeighborsRegressor(n_neighbors, weights='distance')
        clf.fit(X_train, y_train)

        kscore.append(clf.score(X_test,y_test))
        #print kscore[k]
```

```

    k=k+1

    #print (n_neighbors)
    bestk.append(sum(kscore)/len(kscore))
    #print bestk[kc]
    kc+=1

# to do here: given this array of E_outs in CV, find the max, its
# corresponding index, and its corresponding value of n_neighbors
print "Eout="
print clf.score(X,y)
#print "Eout True = "
#print clf.score(X,ytrue)

N_Bestk = sorted(bestk, reverse=True)
#print N_Bestk
#print N_Bestk[0], N_Bestk[1], N_Bestk[2]

index = sorted(range(len(bestk)), key=bestk.__getitem__)
print (index[-1]*2)+1 #odd numbers
print (index[-2]*2)+1 #odd numbers
print (index[-3]*2)+1 #odd numbers

```

For the (b), the data set has to be taken from the **def** genDataSet(N).
Next, the training data set is to be reshaped using

```

X = X.reshape(len(X), 1) #reshaping Training dataset

```

After that, the training data (90% of the complete data) have to be taken and a total of n = 10 neighbours have to be added using

```

for n_neighbors in range(1,900,2): #90% Training Data
    kf = KFold(n_splits=10)
    #n_neighbors = 85
    kscore=[]
    k=0

    for train, test in kf.split(X):
        #print("%s %s" % (train, test))
        X_train, X_test, y_train, y_test = X[train], X[test], y[train], y[test]

        #time.sleep(100)

        # we create an instance of Neighbors Classifier and fit the data.
        clf = neighbors.KNeighborsRegressor(n_neighbors, weights='distance')
        clf.fit(X_train, y_train)

```

```

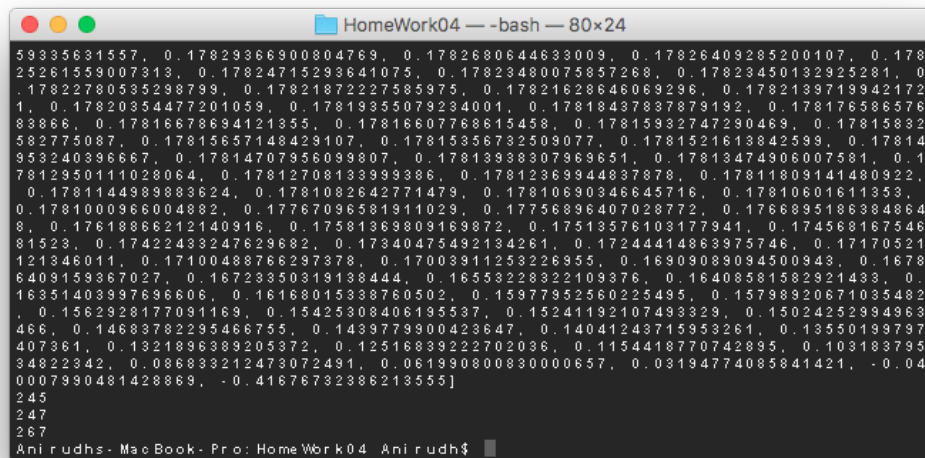
kscore.append(clf.score(X_test,y_test))
#print kscore[k]
k=k+1

#print (n_neighbors)
bestk.append(sum(kscore)/len(kscore))
#print bestk[kc]
kc+=1

```

Now, the 'Best k' is to be obtained and then sorted so that it yields 3 best values of k-neighbours

The output showing the 3 - best values of k is

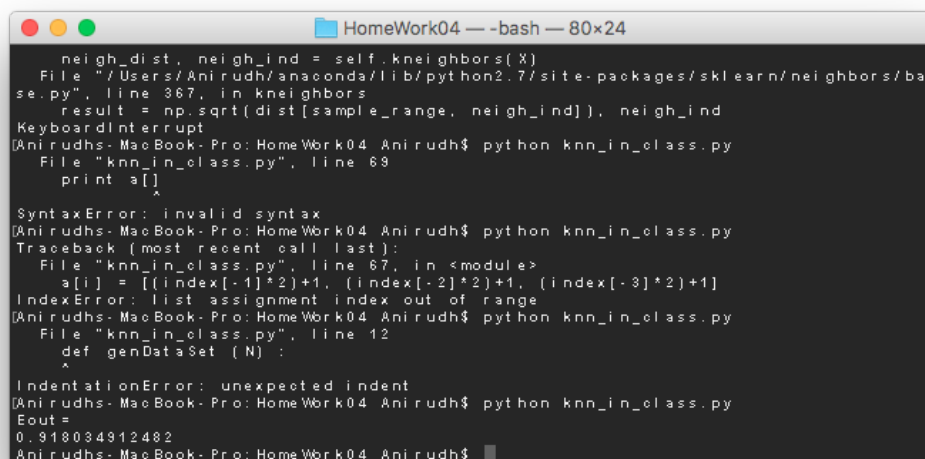


```

59335631557, 0.17829366900804769, 0.1782680644633009, 0.17826409285200107, 0.178
25261559007313, 0.17824715293641075, 0.17823480075857268, 0.17823450132925281, 0
.17822780535298799, 0.17821872227585975, 0.17821628646069296, 0.1782139719942172
1, 0.17820354477201059, 0.17819355079234001, 0.17818437837879192, 0.178176586576
83866, 0.17816678694121355, 0.17816607768615458, 0.17815932747290469, 0.17815832
582775087, 0.17815657148429107, 0.17815356732509077, 0.1781521613842599, 0.17814
953240396667, 0.17814707956099807, 0.17813938307969651, 0.17813474906007581, 0.1
7812950111028064, 0.17812708133999386, 0.17812369944837878, 0.17811809141480922,
0.1781144989883624, 0.1781082642771479, 0.17810690346645716, 0.17810601611353,
0.1781000966004882, 0.17767096581911029, 0.17756896407028772, 0.1766895186384864
8, 0.17618868212140916, 0.17581369809169872, 0.17513576103177941, 0.174568167546
81523, 0.17422433247629682, 0.17340475492134261, 0.17244414863975746, 0.17170521
121346011, 0.17100488766297378, 0.17003911253226955, 0.16909089094500943, 0.1678
6409159367027, 0.16723350319138444, 0.16353228322109376, 0.16408581582921433, 0.
16351403997696606, 0.16168015338760502, 0.15977352560225495, 0.15798920671035482
, 0.1562328177091169, 0.15425308406195537, 0.15241192107493329, 0.15024252994963
466, 0.14683782295466755, 0.1439779900423647, 0.14041243715953261, 0.13550199797
407361, 0.1321896389205372, 0.12516839222702036, 0.1154418770742895, 0.103183795
34822342, 0.086833212473072491, 0.061990800830000657, 0.03194774085841421, -0.04
0007990481428869, -0.41676732386213555]
245
247
267
Anirudhs-MacBook-Pro: HomeWork04 Anirudh$

```

The output showing the best E_{out} value is



```

neigh_dist, neigh_ind = self.kneighbors(X)
File ~/Users/Anirudh/anaconda/lib/python2.7/site-packages/sklearn/neighbors/ba
se.py", line 367, in kneighbors
    result = np.sqrt(dist[sample_range, neigh_ind]), neigh_ind
KeyboardInterrupt
[Anirudhs-MacBook-Pro: HomeWork04 Anirudh$ python knn_in_class.py
File "knn_in_class.py", line 69
    print a[]
          ^
SyntaxError: invalid syntax
[Anirudhs-MacBook-Pro: HomeWork04 Anirudh$ python knn_in_class.py
Traceback (most recent call last):
  File "knn_in_class.py", line 67, in <module>
    a[i] = [(index[-1]^2)+1, (index[-2]^2)+1, (index[-3]^2)+1]
IndexError: list assignment index out of range
[Anirudhs-MacBook-Pro: HomeWork04 Anirudh$ python knn_in_class.py
File "knn_in_class.py", line 12
    def genDataSet (N) :
    ^
IndentationError: unexpected indent
[Anirudhs-MacBook-Pro: HomeWork04 Anirudh$ python knn_in_class.py
Eout=
0.918034912482
Anirudhs-MacBook-Pro: HomeWork04 Anirudh$

```

Problem 2: Using the same dataset you just tried in the previous problem, repeat the experiment 100 times storing the best three k number of neighbors in every single trial, and at the end of all trials plot a histogram of all the values of k that you saved.

Solution:

Repeating the experiment for 100 time can be done easily using a for loop.
The syntax for the for loop is

```
For i in range (0, 100):
```

Now all the obtained 3 best values of k during every iteration has to be stored in an array (abc). This was done using

```
abc = np.append(abc, [[[index[-1]*2)+1, (index[-2]*2)+1, (index[-3]*2)+1]])
```

To create histograms, the commands used are

```
n, bins, patches = plt.hist(abc, facecolor='red', alpha=0.5)
plt.title('Histogram')
plt.xlabel('Iterations')
plt.ylabel('E')
plt.axis([0, 900, 0, 150])
plt.show()
```

/**Complete program attached separately**/

The final outputs obtained are

300 values of E

```
HomeWork04 — python knn_in_class.py — 80x31
Traceback (most recent call last):
  File "knn_in_class.py", line 100, in <module>
    plt.xlabel('Iterations')
AttributeError: 'module' object has no attribute 'xlabel'
Anirudhs-MacBook-Pro:HomeWork04 Anirudh$ python knn_in_class.py
[ 377. 379. 381. 237. 223. 245. 297. 299. 311. 295. 293. 291.
 259. 235. 233. 363. 365. 371. 337. 333. 341. 323. 325. 333.
 279. 271. 291. 319. 311. 309. 315. 317. 313. 267. 265. 257.
 285. 287. 295. 295. 289. 291. 293. 275. 277. 269. 267. 273.
 361. 363. 367. 351. 387. 385. 377. 347. 379. 397. 405. 395.
 295. 293. 281. 307. 355. 309. 343. 345. 329. 281. 279. 275.
 353. 351. 355. 263. 259. 269. 345. 341. 347. 301. 297. 303.
 315. 317. 327. 205. 211. 207. 279. 297. 277. 305. 307. 309.
 423. 405. 407. 337. 323. 335. 373. 367. 371. 245. 267. 251.
 315. 309. 311. 281. 277. 279. 255. 253. 249. 265. 263. 251.
 247. 233. 245. 277. 285. 271. 273. 275. 271. 341. 339. 329.
 259. 271. 257. 303. 299. 301. 389. 391. 387. 345. 349. 367.
 391. 389. 363. 235. 237. 233. 283. 281. 279. 277. 269. 279.
 277. 279. 297. 367. 365. 363. 343. 339. 325. 301. 299. 293.
 269. 285. 271. 399. 391. 397. 307. 305. 281. 233. 237. 239.
 253. 255. 259. 353. 343. 345. 289. 269. 271. 267. 263. 269.
 337. 347. 349. 215. 217. 219. 281. 283. 285. 287. 299. 291.
 327. 329. 331. 291. 293. 289. 277. 279. 269. 279. 281. 299.
 265. 267. 271. 235. 225. 233. 263. 289. 291. 325. 337. 333.
 319. 315. 317. 313. 305. 311. 333. 339. 341. 317. 315. 311.
 301. 299. 277. 289. 291. 287. 371. 369. 373. 295. 281. 293.
 319. 323. 325. 255. 233. 245. 309. 311. 307. 309. 313. 311.
 289. 287. 291. 301. 291. 293. 285. 283. 287. 299. 275. 273.
 285. 295. 303. 275. 245. 249. 281. 279. 289. 341. 335. 349.
 313. 315. 311. 255. 257. 269. 277. 279. 281. 285. 279. 283.]
```

Histogram

