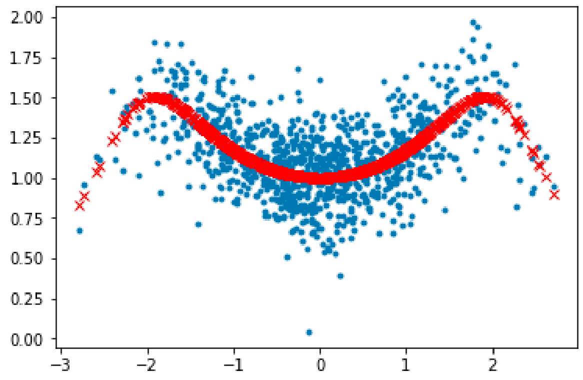
HW03

Cameron Brenner

May 1 2019

1 Problem 1

In problem 1, we are asked to perform 10-fold k-validation on the given data set and give the three best k-values for the set $k = 1, 3, 5, ..., 2[\frac{N+1}{2}] - 1$ which give the best cross validation values and report the best E_{out}. The given data set is:

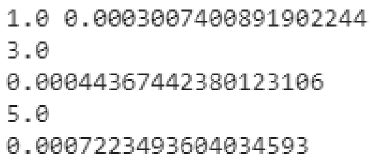


Here is the code:

```
# -*- coding: utf-8 -*-
"""hw3prob1.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1NDYNwnP1qZDthWPKB9sLFBscsqhwNOMN
"""

import numpy as np
from matplotlib import pyplot as plt
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error

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()


kf = KFold(n_splits=10)
x, y, ytrue = genDataSet(1000)
X = np.array([x,y]).T
kf.get_n_splits(X)

msek = {}
#plt.figure(figsize=(10,10))
bestk = 0
bestmse = 100000
for k in np.arange(1,2*np.floor(((len(ytrue)*0.9)+1)/2),2):
  print(k)
  mse = []
  for train_index, test_index in kf.split(X):
    X_train, X_test =      X[train_index],      X[test_index]
    y_train, y_test = ytrue[train_index], ytrue[test_index]


    neigh = KNeighborsRegressor(n_neighbors=int(k))
    neigh.fit(X_train, y_train)
    predictions = neigh.predict(X_test)

    mse.append(mean_squared_error(y_test, predictions))

  print(np.mean(mse))
  msek[k] = np.mean(mse)
  if bestmse > msek[k]:
    bestmse = msek[k]
    bestk = k
    print(bestk,bestmse)

bestkall.append(bestk)
#plt.plot(k,msek[k],'r.')

#plt.show()
print(bestkall)
```

Here are the resulting three best CV E_{out}.



```
1.0 0.0003007400891902244
3.0
0.00044367442380123106
5.0
0.0007223493604034593
```

This makes 1.0 = 0.0003007400891902244 the k-value that returns the best CV E_{out}.

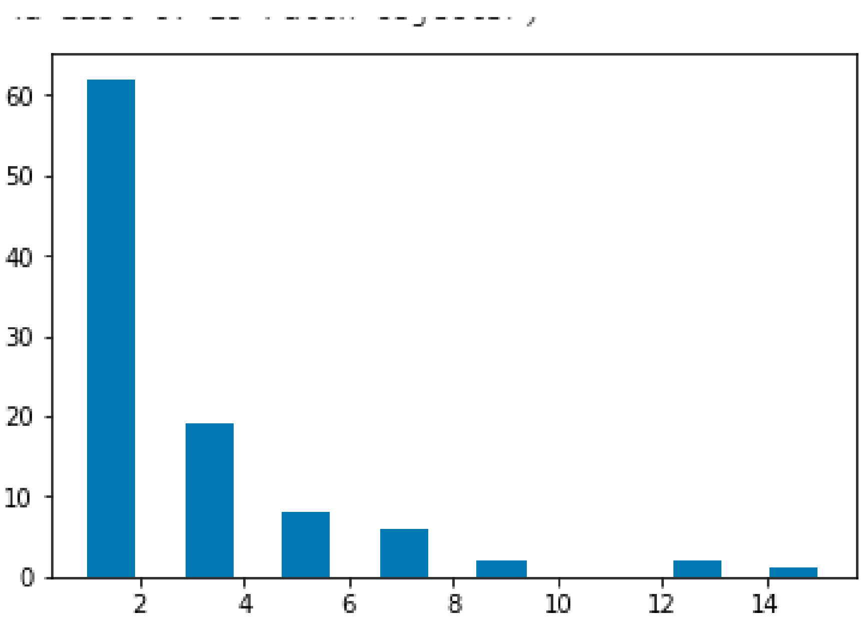
2 Problem 2

In problem 2, we are asked to use the dataset in the above problem, and repeat the experiment 100 time, storing the best k-values, then make a histogram of the k-values and their results.

Here is the code for problem 2.

```
# -*- coding: utf-8 -*-
"""hw3prob2.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1wccJMgFDGrCSDu7CBJeafDujb5Wr-g7M
"""

import numpy as np
from matplotlib import pyplot as plt
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import KFold

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(100)
plt.plot(x,y,'.')
plt.plot(x,ytrue,'rx')
plt.show()

kf = KFold(n_splits=10)

bestkall = []
for i in range(100):
  x, y, ytrue = genDataSet(100)
  X = np.array([x,y]).T
  kf.get_n_splits(X)

  msek = {}
  #plt.figure(figsize=(10,10))
  bestk = 0
  bestmse = 100000
  for k in np.arange(1,2*np.floor(((len(ytrue)*0.9)+1)/2),2):
    #print(k)
    mse = []
    for train_index, test_index in kf.split(X):
      X_train, X_test =      X[train_index],      X[test_index]
      y_train, y_test = ytrue[train_index], ytrue[test_index]


      neigh = KNeighborsRegressor(n_neighbors=int(k))
      neigh.fit(X_train, y_train)
      predictions = neigh.predict(X_test)

      mse.append(mean_squared_error(y_test, predictions))

    #print(np.mean(mse))
    msek[k] = np.mean(mse)
    if bestmse > msek[k]:
      bestmse = msek[k]
      bestk = k
      print(bestk,bestmse)

  bestkall.append(bestk)
  #plt.plot(k,msek[k],'r.')

#plt.show()
print(bestkall)
plt.hist(bestkall,int(np.max(bestkall)))
```

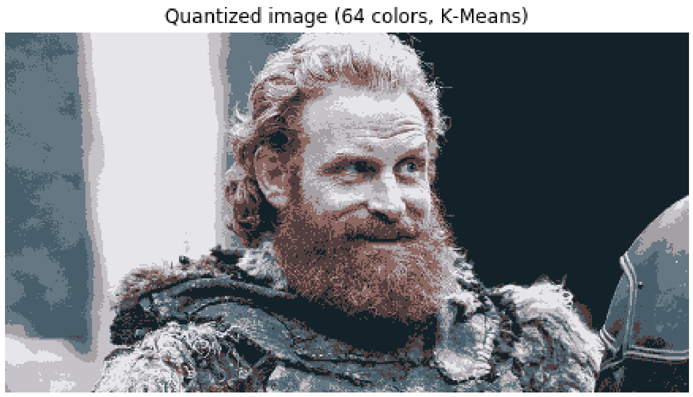
Here is the resulting histogram



3 Problem 3

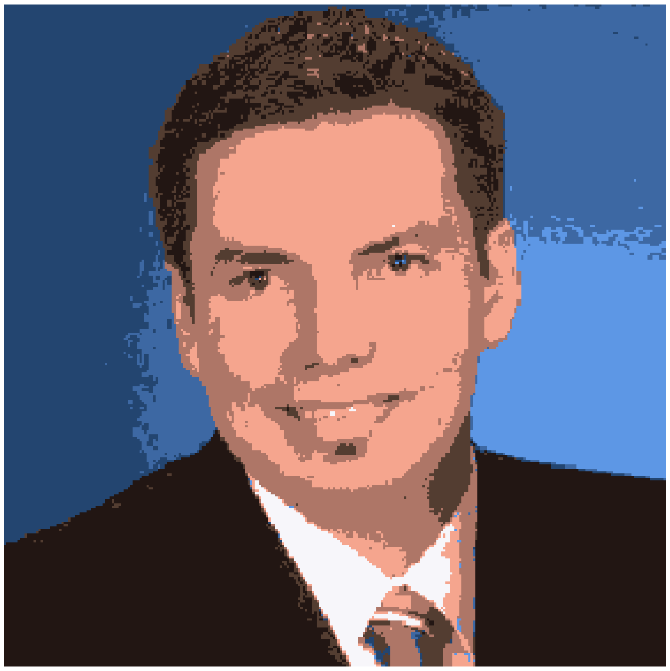
In problem 3, we are asked to use sklearn's implementation of k-means to find the best color clustering of an image. Here is the original image I used.

Here is the result of playing around with the *n_colors* parameter.



I believe that increasing the number of colors increases the spectrum of colors used. The algorithm picks out the number of colors and will replace the similar colors around it with that color. Decreasing the colors decreases the spectrum of colors around a given color. This was represented really well if the image of the professor was used. Both result images are when $n\_colors = 8$.

Quantized image (64 colors, K-Means)



This would be useful in an application where images need to be rendered in lower quality than the default quality of the image, like from high-res DSR cameras to cell-phones. To me, the images looked funny because it is blending in the spectrum of colors usually in a photo, it is erasing shade and depth in the images.

4 Problem 4

In problem 4, we are asked to use an iterative implementation of the Multi-Layered Perceptron to find the best neurons and best eta. My computer received ConvergenceWarnings, so I changed the *max_iter* parameter in the MLPRegressor that was in the loop for a maximum of 500 iterations.

Here are the results of running the code:

```
Neurons 1, eta 0.1. Testing set CV score: -3.167964
Neurons 1, eta 0.2. Testing set CV score: -3.144341
Neurons 18, eta 0.1. Testing set CV score: -2.973462
Neurons 21, eta 0.2. Testing set CV score: -1.605134
Neurons 78, eta 0.1. Testing set CV score: -1.214918
```

From these results, it shows that the best number of neurons that gives the best CV score is 78. I believe that increasing the number of neurons improves the CV testing score, but is costly on the performance. The more neurons, the longer it takes for the code to run. It took me around 20 minutes to run this problem(which was conveniently around the episode length of Curb Your Enthusiasm). The number of neurons can only improve the CV score so much though, as 78 was the best, and not, say, 100. Eventually the number of neurons is costly on the network and the CV score decreases.