Axana Chhetri
HW#1
Report 1

Problem1: Compute an NxN matrix, whose entries are pairwise Euclidean distance between any data a) using loop and b) without any loop but with the matrix operations by numpy. The Euclidean Distance tool is used frequently as a stand-alone tool for applications, such as finding the nearest hospital for an emergency helicopter flight. Alternatively, this tool can be used when creating a suitability map, when data representing the distance from a certain object is needed.

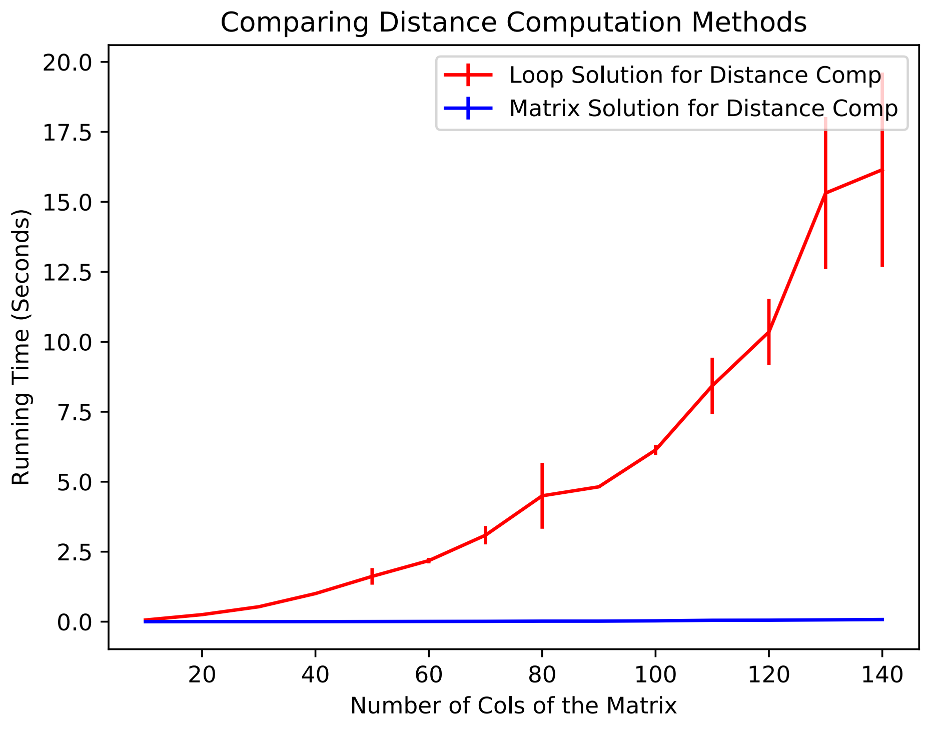
Solutions: I used the following source code with loops

```
def compute_distance_naive(X):
   N = X.shape[0]      # num of rows
   D = X[0].shape[0]   # num of cols

   M = np.zeros([N,N])
   for i in range(N):
      for j in range(N):
         xi = X[i,:]
         xj = X[j,:]
         M[i,j] = np.sqrt(np.dot((xi-xj).T,(xi-xj)))

   return M
```

and the following code without loop

```
def compute_distance_smart(X):
   N = X.shape[0]  # num of rows
   D = X[0].shape[0]  # num of cols

   # use X to create M
   XX = (X * X).sum(axis=1)[:, np.newaxis]
   YY = XX.T
   Z = -2*np.dot(X, X.T) + XX + YY
   np.maximum(Z, 0, out=Z)
   Z.flat[::Z.shape[0] + 1] = 0.0
  Z = np.sqrt(Z)
   return Z
```

Experiment:
I got this as an output. The grapah clearly shows that the loop function takes a very long time compared to numpy function.



Problem 2: Compute the correlation matrix (D x D) a) using the nested loops b) without any loop but use matrix operations by numpy. Correlation is a statistical measure that describes the association between random variables.In almost any business, it is useful to express one quantity in terms of its relationship with others.

Solution:

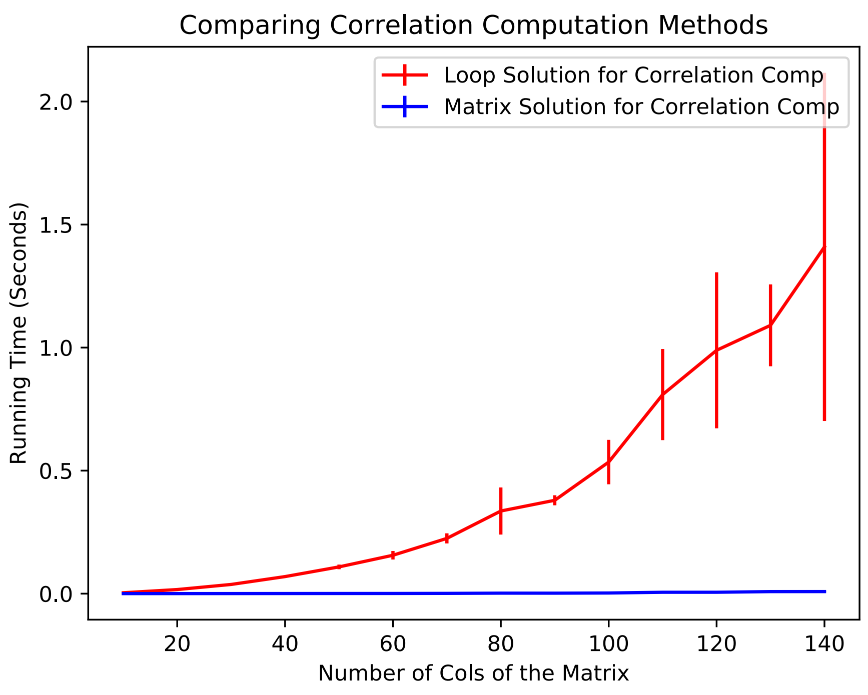
For correlation matrix using loops I used the following source code using nested loops

```
def compute_correlation_naive(X):
    N = X.shape[0]  # num of rows
    D = X[0].shape[0]  # num of cols

    M = np.zeros([D, D])
    for i in range(D):
        for j in range(D):
            xi = X[:, i]
            xj = X[:, j]
            mui = np.sum(xi).astype(float)/ N
            muj = np.sum(xj).astype(float)/N
            xni= xi- mui
            xnj = xj- muj
            zij = (np.dot(xni, xnj)).astype(float)/ (N-1)
            sigi = np.sqrt(np.dot(xni, xni).astype(float)/(N-1))
            sigj = np.sqrt(np.dot(xnj, xnj).astype(float)/(N-1))
            sig = sigi * sigj
            Zmatrix = zij.astype(float)/sig
            M[i,j] = Zmatrix


    return M
```

and the second one without any loop

```
def compute_correlation_smart(X):
    N = X.shape[0]  # num of rows
    D = X[0].shape[0]  # num of cols
    M = np.zeros([D,D])
    vectormu = (np.sum(X, axis = 0)).astype(float)/N
    matrixmu = vectormu * np.ones([N,1])
    x = X- matrixmu
    xT = np.transpose(x)
    cov= (np.dot(xT,x)).astype(float) / (N-1)
    x2 = np.multiply(x,x)
    varience = (np.sum(x2, axis =0)).astype(float) / (N-1)
    sig = np.sqrt(varience)
    Zmatrix = np.outer(sig, sig)
    M = np.multiply(cov, np.power(Zmatrix, -1))
    return M
```

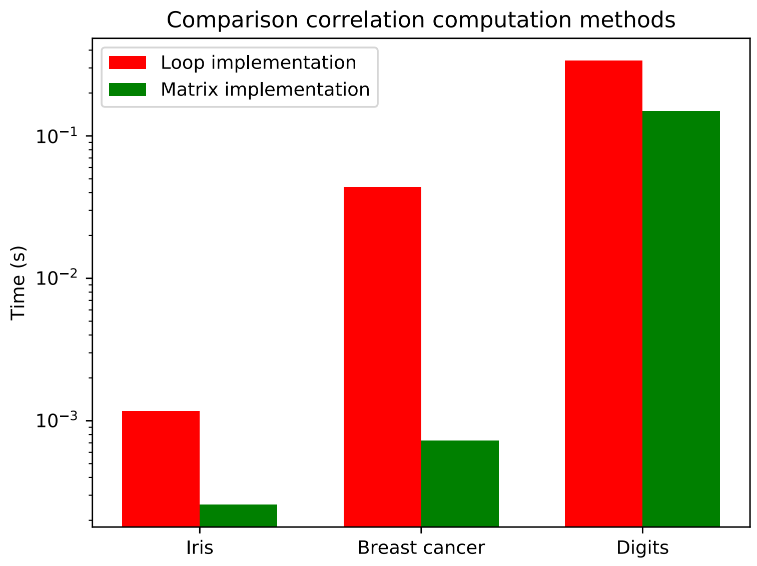
<u>Experiment:</u>
I got this as an output. The grapah clearly shows that the loop function takes a very long time compared to numpy function.

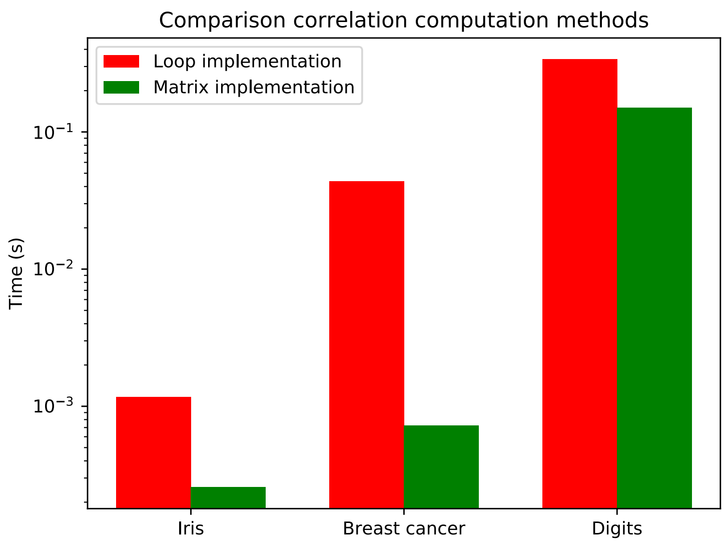


<u>Problem 3</u>: Compute the pairwise distance matrix and correlation matrix for the three datasets from sklearn:

Iris (N = 150 , D = 4);
Breast cancer (N = 569 , D = 30);
Digits (N = 1797 , D = 64).

When using the three datasets from the sklearn, i go the following graph which also shows that the loop function takes a very long time compared to without loops function


Comparison correlation computation methods
Loop implementation
Matrix implementation
10^{-1}
10^{-2}
10^{-3}
Time (s)
Iris
Breast cancer
Digits


Comparison correlation computation methods
Loop implementation
Matrix implementation
10^{-1}
10^{-2}
10^{-3}
Time (s)
Iris
Breast cancer
Digits