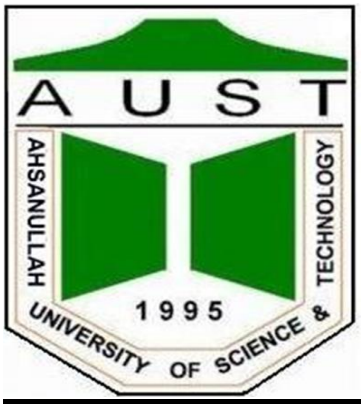
AHSANULLAH UNIVERSITY OF SCIENCE & TECHNOLOGY



Course No: CSE 4238

Course Name: Soft Computing Lab

Section: C Lab Group: C1

Semester: Fall 2020

Submitted to:

Mr. Nibir Chandra Mandal

Lecturer,

Department of CSE, AUST

Submitted By:

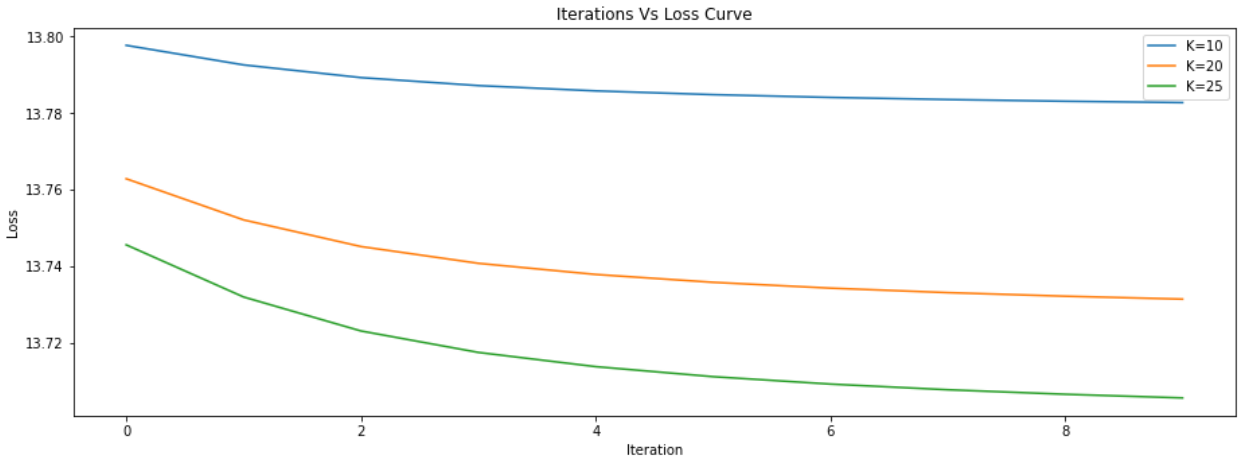
Name: Md. Siam Islam

ID:170104124

Email: 170104124@aust.edu

Generate Loss curves for the hyperparameters K

Loss curves for each K's of each iterations has been plotted in a graph which has been shown below.




```
[11]  ==============For 10 Latent Factors======================
      Iteration-0: Loss=13.79773072900103
      Iteration-1: Loss=13.792611290717785
      Iteration-2: Loss=13.789291985166923
      Iteration-3: Loss=13.78720001986388
      Iteration-4: Loss=13.785811060882878
      Iteration-5: Loss=13.784832297385373
      Iteration-6: Loss=13.784108122445721
      Iteration-7: Loss=13.783551575753663
      Iteration-8: Loss=13.783110870296552
      Iteration-9: Loss=13.782753391597328

      ==============For 20 Latent Factors======================
      Iteration-0: Loss=13.762837739519858
      Iteration-1: Loss=13.752075866702647
      Iteration-2: Loss=13.745104686844995
      Iteration-3: Loss=13.740717806658505
      Iteration-4: Loss=13.73780544187576
      Iteration-5: Loss=13.735750054534975
      Iteration-6: Loss=13.734225028440479
      Iteration-7: Loss=13.733048963573042
      Iteration-8: Loss=13.732114473997163
      Iteration-9: Loss=13.731354193872436

      ==============For 25 Latent Factors======================
      Iteration-0: Loss=13.745550937643095
      Iteration-1: Loss=13.731912382892281
      Iteration-2: Loss=13.723022882168904
      Iteration-3: Loss=13.71741330394252
      Iteration-4: Loss=13.713691272731168
      Iteration-5: Loss=13.711071255916677
      Iteration-6: Loss=13.709134199305897
      Iteration-7: Loss=13.707645979773165
      Iteration-8: Loss=13.706467525737452
      Iteration-9: Loss=13.70551153760774
```

Optimal Hyperparameters

K and iterations are used as hyperparameters.For each K's of each iterations, we update U and V matrix for convergence.As our machine's aren't suitable for larger K's and iterations.We compare the consecutive loss of each iterations.If the consecutive loss for last 10 iterations is less than 0.1,then we stop convergence for this certain K.Code of implementation has been given below.

```
def matrix_factorization(K_list,itr):
  for K in K_list:
    U=np.random.rand(N,K)
    U_norm = Normalizer(norm='l2').fit(U)
    U = U_norm.transform(U)
    lam_u=(0.00015 + 0.0001 * (index_id % 8))
    lam_v=(0.00025 - 0.0001 * (index_id % 7))
    Loss[K]=[]
    error=0
    error_diff=0
    print(f'\n==============For {K} Latent Factors====================')
    for i in range(itr):
      V=np.dot(np.linalg.inv(np.dot(U.T,U)+lam_v*np.identity(K)),np.dot(X.fillna(0).T,U).T)
      U=np.dot(np.linalg.inv(np.dot(V,V.T)+lam_u*np.identity(K)),np.dot(X.fillna(0),V.T).T).T
      L=np.nansum(np.square(X-np.dot(U,V)))/np.nansum(X.count())
      Loss[K].append(L)
      print(f'Iteration-{i}: Loss={L}')
      if(math.floor(L)==0 or (len(Loss[K])%10==0 and abs(Loss[K][-1]-Loss[K][-10])<0.1)):
        UxV[K]=np.dot(U,V)
        break
```

In the plotted graph,it is shown that optimal value for K is 25 for 10 iterations.