Package

In [14]:

```
import numpy as np
from sklearn.metrics import pairwise_distances
from scipy.spatial.distance import cosine
```

In [15]:

```
import pandas as pd
import numpy as np
#from sklearn.neighbors import NearestNeighbors
from numpy import array
from numpy.linalg import norm

from operator import itemgetter
import timeit
```

In [16]:

```
from sklearn.linear_model import LinearRegression
```

Functions

In [17]:

```
def Average(lst):
    return sum(lst) / len(lst)
```

In [18]:

```
def rmse(predictions, targets):
    return np.sqrt(((predictions - targets) ** 2).mean())
```

In [19]:

```
def pq_check_rate(test_movie_id,df,userid):
    """
    Input
    ----
        df: df_pq"""

    user_index = b_user_dic.get(str(userid))

    movie_index =b_item_dic.get(str(test_movie_id))

    rating_predict = df[user_index][movie_index]
    return rating_predict
```

In [20]:

```
def item_check_value(test_movie_id,mat):
    """
    Input
    ----
        mat: b_item,matrix
    Return
    -----
        a float"""

    #user_index = b_user_dic.get(str(userid))

    movie_index =b_item_dic.get(str(test_movie_id))

    value_predict = mat[0][movie_index]
    return value_predict
```

In [21]:

```
def user_check_value(userid,mat):
    """
    Input
    ----
        mat: b_user,matrix"""

    user_index = b_user_dic.get(str(userid))

    #movie_index =b_item_dic.get(str(test_movie_id))

    value_predict = mat[0][user_index]
    return value_predict
```

In [22]:

```
def time_check_value(test_movie_id,df,time_bin):
    """
    Input
    ----
        mat: b_time"""
    #print(test_movie_id,time_bin)
    time_index = b_time_dic.get(str(time_bin))
    #print(time_bin,time_index)

    movie_index =b_item_dic.get(str(test_movie_id))


    rating_predict = df[time_index][movie_index]
    return rating_predict
```

In [23]:

```
def pariwise_cosin(q):
    """
    Input
    -----
        q: q.csv
    Return
    -----
        """

#     q_matrix = q.copy()
#     q_matrix = q_matrix.values

#     q_knn_matrix = q_matrix.T

    return 1-pairwise_distances(q.values.T, metric="cosine")
```

In [24]:

```
def pair_cos_id (test_movie_id):
    """
    Input
    -----
        test_movie_id:int
        pair_cos_q: nparray, pair_cosin_q (global var)
    Return
    -----
        list with index of nearst movie,id of nearest movie"""
    test_movie_index = b_item_dic.get(str(test_movie_id))
    df_cos = pd.DataFrame(pair_cos_q)


    high_2_info = df_cos[test_movie_index].nlargest(2)
    knn_near_index = np.where(df_cos[test_movie_index] == high_2_info.values[1
])[0][0]

    #knn_movie_id = b_item_dic.get(str(test_movie_id))
    knn_movie_id  = list(b_item_dic.keys())[list(b_item_dic.values()).index(kn
n_near_index)]
    #[knn_near_index,int(knn_movie_id)]
    return int(knn_movie_id)
```

In [25]:

```
def avg_rating_train_look_up(movie_id_int):
    """
    Global variable
    ------
        df_data_train:after groupby mean and reset index

    Return
    ------
        rating"""
    #print(df_data_train[df_data_train['movieId'] == movie_id_int])

    return rating[rating['movieId'] == movie_id_int]['rating'].values[0]
```

In [26]:

```
def main_get_rating_knn(test_movie_id):
    """
    Global var
    ------
        pair_cos_q
        df_data_train

    Return
    -----
        Get KNN rating """
    nearest_movie_id = pair_cos_id(test_movie_id)
    #print(nearest_movie_id)
    avg_rating_nearest_movie_id = avg_rating_train_look_up(nearest_movie_id)
    #print(avg_rating_nearest_movie_id)


    return avg_rating_nearest_movie_id
```

In [27]:

```
def b_time_user(user_id):
    rating_user = rating[rating['userId'] == user_id]
    rating_group = rating_user[['userId','movieId','bin_num']].values

    user_time_movie_list = []
    for j, pair in enumerate(rating_group):
        user_time_movie_list.append((time_check_value(pair[1],b_time,pair[2]))
)
    return user_time_movie_list
    #time_check_value(test_movie_id,df,time_bin)
    #b_item_col_qiqi.append(b_item_col_qiqi)
```

Data

In [28]:

```
# ################################## #sgd data
# p = pd.read_csv('../output/p_5.csv',index_col='Unnamed: 0')
# #rmse = pd.read_csv('../data/rmse.csv')
# q = pd.read_csv('../output/q_5.csv',index_col='Unnamed: 0') #movie latent va
lues
# #rating = pd.read_csv('../data/ml-latest-small/ratings.csv')
# rating = pd.read_csv('../output/data_bin.csv',index_col = 'Unnamed: 0')
# data_train = pd.read_csv('../output/data_train_add.csv',index_col='Unnamed:
0')
# data_test = pd.read_csv('../output/data_test.csv',index_col='Unnamed: 0')

# b_item = pd.read_csv('../output/b_item.csv',index_col='Unnamed: 0')
# b_time = pd.read_csv('../output/b_time.csv',index_col='Unnamed: 0')
# b_user = pd.read_csv('../output/b_user.csv',index_col='Unnamed: 0')
```

In [29]:

```
############################## #als data
p = pd.read_csv('../output/P_ALS.csv',index_col='Unnamed: 0')
#rmse = pd.read_csv('../data/rmse.csv')
q = pd.read_csv('../output/Q_ALS.csv',index_col='Unnamed: 0') #movie latent va
lues
#rating = pd.read_csv('../data/ml-latest-small/ratings.csv')
rating = pd.read_csv('../output/data_bin.csv',index_col = 'Unnamed: 0')
data_train = pd.read_csv('../output/data_train_add.csv',index_col='Unnamed: 0'
)
data_test = pd.read_csv('../output/data_test.csv',index_col='Unnamed: 0')

b_item = pd.read_csv('../output/b_item_ALS.csv',index_col='Unnamed: 0')
b_time = pd.read_csv('../output/b_time_ALS.csv',index_col='Unnamed: 0')
b_user = pd.read_csv('../output/b_user_ALS.csv',index_col='Unnamed: 0')
```

In [30]:

```
b_time.head(2)#movie id,bin
```

Out[30]:

	1	2	3	4	5	6	7	8
0	-3.642852	-3.458309	-3.258477	-2.836368	-2.943978	-3.791145	-3.206189	-3.113072
1	0.077626	-0.203374	-0.298689	0.073975	-0.073602	0.049277	-0.163726	-0.000800

2 rows × 9724 columns

Processing Data

In [31]:

```
##calculate predict rating matrix
b_time_dic = dict(zip(list(b_time.columns),range(len(b_time.columns)))) #id, i
ndex
b_item_dic = dict(zip(list(b_item.columns),range(len(b_item.columns)))) #id, i
ndex
b_user_dic = dict(zip(list(b_user.columns),range(len(b_user.columns)))) #id, i
ndex

p_matrix = p.values
q_matrix = q.values
b_item = b_item.values
b_time = b_time.values
b_user = b_user.values
```

In [32]:

```
mat_pq = q_matrix.T.dot(p_matrix)
df_pq = pd.DataFrame(mat_pq)
```

In [33]:

```
pair_cos_q = pariwise_cosin(q)
```

In [34]:

```
data_test_sort = data_test.copy()
data_test_sort = data_test_sort.sort_values(by=['userId'])

userId = list(set(rating['userId'])) #a list of user id, int

movie_id = list(q.columns[:]) #list of all movie id,str
```

In [35]:

```
###for loop for the KNN part, only movie id considered

tic=timeit.default_timer()

knn_result = []
for i, item in enumerate(movie_id): #1_train_movie
    #print(i,item)
    #print(i,item,type(i),type(item)) ##0,1 int, int, index, id
    #print(main_get_rating_knn(item),item,i)
    knn_result.append(main_get_rating_knn(item))


toc=timeit.default_timer()
toc - tic #elapsed time in seconds
```

Out[35]:

```
50.07721570000001
```

In [36]:

```
dict_knn_id_rating = dict(zip(movie_id,knn_result))
```

In [37]:

```
#produce knn col
tic=timeit.default_timer()

rating_knn_col =[]
for i, item in enumerate(rating['movieId']):
    #print(i,item,type(i),type(item)) #0,1,int,int, index,id
    #based id find rating
    rating_knn_col.append(dict_knn_id_rating.get(str(item)))
toc=timeit.default_timer()
toc - tic #elapsed time in seconds
```

Out[37]:

```
0.14450890000000527
```

In [38]:

```
rating_sub = rating[['userId','movieId','bin_num']].values
```

In [39]:

```
#produce pq col

tic=timeit.default_timer()

search_col_result = []
for i, group in enumerate(rating_sub):
    search_col_result.append(pq_check_rate(group[1],df_pq,group[0]))

toc=timeit.default_timer()
toc - tic #elapsed time in seconds
```

Out[39]:

```
1.8990631000000349
```

In [40]:

```
#produce b_item col

tic=timeit.default_timer()

b_item_col = []
for i, group in enumerate(rating_sub): #userid, movieid
    b_item_col.append(item_check_value(group[0],b_item))

toc=timeit.default_timer()
toc - tic #elapsed time in seconds
```

Out[40]:

```
0.259394699999916 16
```

In [41]:

```
#produce b_user col
tic=timeit.default_timer()

b_user_col = []
for i, group in enumerate(rating_sub): #userid, movieid
    b_user_col.append(user_check_value(group[0],b_user))

toc=timeit.default_timer()
toc - tic #elapsed time in seconds
```

Out[41]:

```
0.251794300000028 67
```

In [42]:

```
#produce b_time col
tic=timeit.default_timer()

b_time_col = []
for i, group in enumerate(rating_sub): #userid, movieid,time
    #print(i,group,'group')
    b_time_col.append(time_check_value(group[1],b_time,group[2]))

toc=timeit.default_timer()
toc - tic #elapsed time in seconds
```

Out[42]:

```
0.406207699999981 74
```

In [43]:

```
rating_merge = rating.copy()
rating_merge['knn'] = rating_knn_col
rating_merge['pq'] = search_col_result
rating_merge['b_item'] = b_item_col
rating_merge['b_user'] = b_user_col
rating_merge['b_time'] = b_time_col
```

In [44]:

```
rating_merge_train = rating_merge[:80670]
rating_merge_train.to_csv('rating_merge_train_ALS.csv')
```

In [45]:

```
rating_merge_test = rating_merge[80670:]
rating_merge_test.to_csv('rating_merge_test_ALS.csv')
```

In [46]:

```
len(rating) == len(rating_merge_test)+len(rating_merge_train)
```

Out[46]:

```
True
```

check

In [47]:

```
rating_merge_train.head()
```

Out[47]:

	userId	movieId	rating	timestamp	bin_num	knn	pq	b_item	b_user	b_tin
1	1	1	4.0	964982703	5	3.5	-0.261606	1.03597	0.088535	0.0751
2	1	3	4.0	964981247	5	2.5	-0.454195	1.03597	0.088535	0.0902
3	1	6	4.0	964982224	5	4.0	-0.644210	1.03597	0.088535	0.0025
4	1	47	5.0	964983815	5	3.0	-0.254080	1.03597	0.088535	0.0995
5	1	50	5.0	964982931	5	4.0	-0.244089	1.03597	0.088535	0.0898

In [48]:

```
rating_merge_train.shape
```

Out[48]:

```
(80670, 10)
```

In [49]:

```
rating_merge_test.head()
```

Out[49]:

	userId	movieId	rating	timestamp	bin_num	knn	pq	b_item	b_user
80671	509	53322	3.0	1435994136	20	1.0	0.231352	0.359473	-0.791639
80672	509	53956	3.0	1436101754	20	4.5	0.220205	0.359473	-0.791639
80673	509	53993	2.5	1435995535	20	4.0	-0.000888	0.359473	-0.791639
80674	509	54259	3.5	1436393817	20	5.0	-0.418584	0.359473	-0.791639
80675	509	54780	3.0	1435998294	20	2.0	0.014711	0.359473	-0.791639

In [50]:

```
x_label = list(rating_merge_test.columns[5:])
x_label
```

Out[50]:

```
['knn', 'pq', 'b_item', 'b_user', 'b_time']
```

In [ ]:

```

```