學號：R05921087 系級： 電機碩一 姓名：鍾智堅

1. (1%)請比較有無 normalize(rating)的差別。並說明如何 normalize.
   答：在 latent dimension 為 128 下，還未 normalize rating 的 kaggle 分數為 0.86651。而 normalize 后，則分數變差了，kaggle 分數為 0.91986。

```
#normalize output
Y_data = np.array(Y_data, dtype='float')
y_std = np.std(Y_data)
y_mean = np.mean(Y_data)
Y_data = (Y_data - y_mean) / float(y_std)
```

```
#denormalized output
Y_test = Y_test * y_std + y_mean
Y_test = np.around(Y_test)
Y_test = np.array(Y_test, dtype='int')
```

2. (1%)比較不同的 latent dimension 的結果。
   答：原始的 latent dimension 為 128，即 kaggle 分數為 0.86651。當 latent dimension 調整為 256 時，kaggle 分數變差了，即為 0.86945。當把 latent dimension 減少為 64 和 16 時，kaggle 分數分別為 0.86601 和 0.86540，比原本更好。但是當降到為 2 時卻變差了，kaggle 分數為 0.89079。
   因此，可以看得出 latent dimension 為 16 可以得到最好的 kaggle 分數。

3. (1%)比較有無 bias 的結果。
   答：有 bias 的情況下，kaggle 分數為 0.87550，但是取消 bias 后，發現效果變得好很多，即 0.86651。

4. (1%)請試著用 DNN 來解決這個問題，並且說明實做的方法(方法不限)。並比較 MF 和 NN 的結果，討論結果的差異。
   答：以下是我的 DNN 架構，batch size 為 256，validation split 為 0.01。kaggle 分數比 MF 更好，即 DNN 的是 0.85669，MF 的是 0.86651。

```
#MODEL
user_input = Input(shape=[1])
user_embedding = Embedding(n_users+1, 256, input_length=1)(user_input)
user_vec = Flatten()(user_embedding)
user_vec = Dropout(0.2)(user_vec)

movie_input = Input(shape=[1])
movie_embedding = Embedding(n_movies+1, 256, input_length=1)(movie_input)
movie_vec = Flatten()(movie_embedding)
movie_vec = Dropout(0.2)(movie_vec)

vec_inputs = Concatenate()([user_vec, movie_vec])
model = Dense(1024, activation='relu')(vec_inputs)
model = Dropout(0.4)(model)
model = BatchNormalization()(model)
model = Dense(256, activation='relu')(model)
model = Dropout(0.4)(model)
model = BatchNormalization()(model)
model = Dense(64, activation='relu')(model)
model = Dropout(0.3)(model)
model = BatchNormalization()(model)
model = Dense(32, activation='relu')(model)
model = Dropout(0.4)(model)
model = BatchNormalization()(model)
model_out = Dense(1, activation='linear')(model)

model = Model([user_input, movie_input], model_out)
```

5. (1%)請試著將 movie 的 embedding 用 tsne 降維後，將 movie category 當作 label 來作圖。

6. (BONUS)(1%)試著使用除了 rating 以外的 feature, 並說明你的作法和結果，結果好壞不會影響評分。

   答：首先，把 user.csv 里的資料轉換成數字，特別是性別部分，男生為 0 女生為 1。然後對 train 和 test 檔案的 user 做搜尋和對 user.csv 資料做配對，然後再用 DNN 方法去測試，發現 kaggle 分數反而變好很多，即 0.85221，主要是因為有關 user 的資訊變多了。

   以下是 preprocessing data。

```
numUserGender = {"M":0, "F":1}
users_data.Gender = users_data.Gender.replace(numUserGender)
users_data.Gender = users_data.Gender.astype('int')

n_movies = ratings_data['MovieID'].drop_duplicates().max()
n_users = ratings_data['UserID'].drop_duplicates().max()
movieID = ratings_data.MovieID.values
userID = ratings_data.UserID.values

print("Processing train %d User's Data..." %(len(userID)))
from progress.bar import Bar
bar = Bar('Processing', max=len(userID))
userDetailsInput = []
for i in range(len(userID)):
    userDetailsInput.append(users_data[users_data.UserID == userID[i]].values[0][0:4])
    bar.next()
bar.finish()
userDetailsInput = np.asarray(userDetailsInput, dtype='int')
print('User Input data: ', userDetailsInput.shape)

test_users = test_data.UserID.values
print("Processing test %d User's Data..." %(len(test_users)))
bar = Bar('Processing', max=len(test_users))
testUserDetailsInput = []
for i in range(len(test_users)):
    testUserDetailsInput.append(users_data[users_data.UserID == test_users[i]].values[0][0:4])
    bar.next()
bar.finish()
testUserDetailsInput = np.asarray(testUserDetailsInput, dtype='int')
print('Test User Input data: ', testUserDetailsInput.shape)
```

以下是 model 的程式

```
#MODEL
user_input = Input(shape=[4])
user_embedding = Embedding(n_users+1, 256)(user_input)
user_vec = Flatten()(user_embedding)
user_vec = Dropout(0.2)(user_vec)

movie_input = Input(shape=[1])
movie_embedding = Embedding(n_movies+1, 256)(movie_input)
movie_vec = Flatten()(movie_embedding)
movie_vec = Dropout(0.2)(movie_vec)

vec_inputs = Concatenate()([user_vec, movie_vec])
model = Dense(1024, activation='relu')(vec_inputs)
model = Dropout(0.4)(model)
model = BatchNormalization()(model)
model = Dense(256, activation='relu')(model)
model = Dropout(0.4)(model)
model = BatchNormalization()(model)
model = Dense(64, activation='relu')(model)
model = Dropout(0.3)(model)
model = BatchNormalization()(model)
model = Dense(32, activation='relu')(model)
model = Dropout(0.4)(model)
model = BatchNormalization()(model)
model_out = Dense(1, activation='linear')(model)


model = Model([user_input, movie_input], model_out)
```