1. (1%)請比較有無 normalize(rating)的差別。並說明如何 normalize.
   normalized: rmse= 0.96900 normalize 方法: (train['Rating'] - train['Rating'].min())/ 5
   Unnormalized: rmse=0.85458, model as below:

Layer (type)	Output Shape	Param #	Connected to
UserID (InputLayer)	(None, 1)	0	
MovieID (InputLayer)	(None, 1)	0	
embedding_1 (Embedding)	(None, 1, 120)	724920	UserID[0][0]
embedding_2 (Embedding)	(None, 1, 120)	474360	MovieID[0][0]
flatten_1 (Flatten)	(None, 120)	0	embedding_1[0][0]
flatten_2 (Flatten)	(None, 120)	0	embedding_2[0][0]
embedding_3 (Embedding)	(None, 1, 1)	6041	UserID[0][0]
embedding_4 (Embedding)	(None, 1, 1)	3953	MovieID[0][0]
dot_1 (Dot)	(None, 1)	0	flatten_1[0][0] flatten_2[0][0]
flatten_3 (Flatten)	(None, 1)	0	embedding_3[0][0]
flatten_4 (Flatten)	(None, 1)	0	embedding_4[0][0]
add_1 (Add)	(None, 1)	0	dot_1[0][0] flatten_3[0][0] flatten_4[0][0]

Total params: 1,209,274
Trainable params: 1,209,274
Non-trainable params: 0

2. (1%)比較不同的 latent dimension 的結果。
   latent dimension=120: rmse=0.85458
   latent dimension=110: rmse=0.85642

3. (1%)比較有無 bias 的結果。
   with bias: rmse=0.85458 ，比較好 without bias: rmse=0.86758

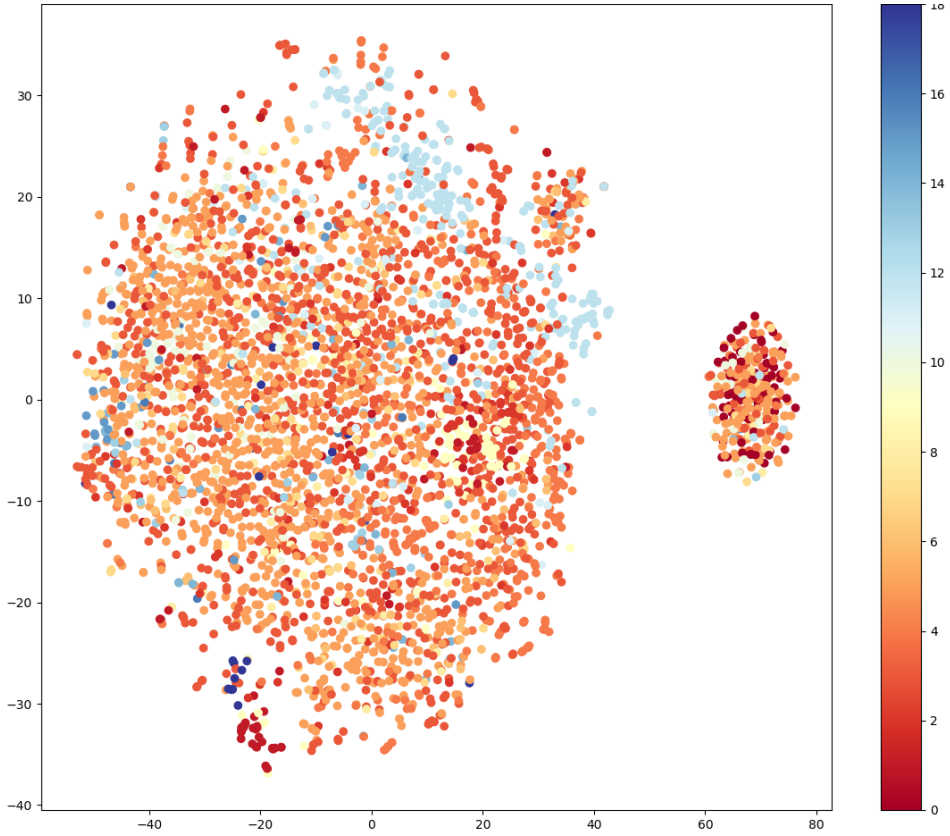
4. (1%)請試著用 DNN 來解決這個問題，並且說明實做的方法(方法不限)。並比較 MF 和 NN 的結果，討論結果的差異。

Layer (type)	Output Shape	Param #
merge_1 (Merge)	(None, 256)	0
dropout_1 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 128)	32896
batch_normalization_1 (Batch	(None, 128)	512
dropout_2 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 32)	4128
batch_normalization_2 (Batch	(None, 32)	128
dropout_3 (Dropout)	(None, 32)	0
dense_3 (Dense)	(None, 8)	264
batch_normalization_3 (Batch	(None, 8)	32
dropout_4 (Dropout)	(None, 8)	0
dense_4 (Dense)	(None, 1)	9

Total params: 1,317,201Trainable params: 1,316,865
Non-trainable params: 336

   rmse=0.86555 稍微差一點，但是 DNN 的 epoch 要多跑很多(100 vs 10)

5. (1%)請試著將 movie 的 embedding 用 t-sne 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。

   直接將 movie 的 Genre 做成 one-hot encoding 進行 DNN。結果並沒有比 MF 好，但比 DNN 好一點 rmse=0.85455