學號：b04501127 系級： 土木三 姓名：凌于凱

1. (1 %)請比較有無 normalize 的差別。並說明如何 normalize. (ref: 去年手把手)

   MF model(no normalize): public: 0.88594 private: 0.88139

   MF model(normalize): public: 0.87487 private: 0.86755

Layer (type)	Output Shape	Param #	Connected to
input_5 (InputLayer)	(None, 1)	0	
input_6 (InputLayer)	(None, 1)	0	
embedding_9 (Embedding)	(None, 1, 100)	604100	input_5[0][0]
embedding_10 (Embedding)	(None, 1, 100)	395300	input_6[0][0]
flatten_8 (Flatten)	(None, 100)	0	embedding_9[0][0]
flatten_9 (Flatten)	(None, 100)	0	embedding_10[0][0]
embedding_11 (Embedding)	(None, 1, 1)	6041	input_5[0][0]
embedding_12 (Embedding)	(None, 1, 1)	3953	input_6[0][0]
dot_3 (Dot)	(None, 1)	0	flatten_8[0][0] flatten_9[0][0]
flatten_10 (Flatten)	(None, 1)	0	embedding_11[0][0]
flatten_11 (Flatten)	(None, 1)	0	embedding_12[0][0]
add_3 (Add)	(None, 1)	0	dot_3[0][0] flatten_10[0][0] flatten_11[0][0]

Total params: 1,009,394
Trainable params: 1,009,394
Non-trainable params: 0

   方法：rating = (rating – mean(rating)) / std(rating)

2. (1 %)比較不同的 embedding dimension 的結果。(ref: 去年手把手)

   NN model(dim = 500): public: 0.87449 private: 0.86563

Layer (type)	Output Shape	Param #	Connected to
input_7 (InputLayer)	(None, 1)	0	
input_8 (InputLayer)	(None, 1)	0	
embedding_11 (Embedding)	(None, 1, 500)	3020500	input_7[0][0]
embedding_12 (Embedding)	(None, 1, 500)	1976500	input_8[0][0]
flatten_9 (Flatten)	(None, 500)	0	embedding_11[0][0]
flatten_10 (Flatten)	(None, 500)	0	embedding_12[0][0]
concatenate_2 (Concatenate)	(None, 1000)	0	flatten_9[0][0] flatten_10[0][0]
dense_4 (Dense)	(None, 150)	150150	concatenate_2[0][0]
batch_normalization_1 (BatchNorm	(None, 150)	600	dense_4[0][0]
leaky_re_lu_1 (LeakyReLU)	(None, 150)	0	batch_normalization_1[0][0]
dropout_3 (Dropout)	(None, 150)	0	leaky_re_lu_1[0][0]
dense_5 (Dense)	(None, 150)	22650	dropout_3[0][0]
batch_normalization_2 (BatchNorm	(None, 150)	600	dense_5[0][0]
leaky_re_lu_2 (LeakyReLU)	(None, 150)	0	batch_normalization_2[0][0]
dropout_4 (Dropout)	(None, 150)	0	leaky_re_lu_2[0][0]
dense_6 (Dense)	(None, 50)	7550	dropout_4[0][0]
batch_normalization_3 (BatchNorm	(None, 50)	200	dense_6[0][0]
leaky_re_lu_3 (LeakyReLU)	(None, 50)	0	batch_normalization_3[0][0]
dropout_5 (Dropout)	(None, 50)	0	leaky_re_lu_3[0][0]
dense_7 (Dense)	(None, 1)	51	dropout_5[0][0]

Total params: 5,178,801
Trainable params: 5,178,101
Non-trainable params: 700

   NN model(dim = 100): public: 0.86752 private: 0.85880

Layer (type)	Output Shape	Param #	Connected to
input_5 (InputLayer)	(None, 1)	0	
input_6 (InputLayer)	(None, 1)	0	
embedding_5 (Embedding)	(None, 1, 100)	604100	input_5[0][0]
embedding_6 (Embedding)	(None, 1, 100)	395300	input_6[0][0]
flatten_5 (Flatten)	(None, 100)	0	embedding_5[0][0]
flatten_6 (Flatten)	(None, 100)	0	embedding_6[0][0]
concatenate_3 (Concatenate)	(None, 200)	0	flatten_5[0][0] flatten_6[0][0]
dropout_9 (Dropout)	(None, 200)	0	concatenate_3[0][0]
dense_9 (Dense)	(None, 150)	30150	dropout_9[0][0]
batch_normalization_7 (BatchNorm	(None, 150)	600	dense_9[0][0]
leaky_re_lu_7 (LeakyReLU)	(None, 150)	0	batch_normalization_7[0][0]
dropout_10 (Dropout)	(None, 150)	0	leaky_re_lu_7[0][0]
dense_10 (Dense)	(None, 150)	22650	dropout_10[0][0]
batch_normalization_8 (BatchNorm	(None, 150)	600	dense_10[0][0]
leaky_re_lu_8 (LeakyReLU)	(None, 150)	0	batch_normalization_8[0][0]
dropout_11 (Dropout)	(None, 150)	0	leaky_re_lu_8[0][0]
dense_11 (Dense)	(None, 50)	7550	dropout_11[0][0]
batch_normalization_9 (BatchNorm	(None, 50)	200	dense_11[0][0]
leaky_re_lu_9 (LeakyReLU)	(None, 50)	0	batch_normalization_9[0][0]
dropout_12 (Dropout)	(None, 50)	0	leaky_re_lu_9[0][0]
dense_12 (Dense)	(None, 1)	51	dropout_12[0][0]

Total params: 1,061,201
Trainable params: 1,060,501
Non-trainable params: 700

Dim = 500 的情況下，參數量有點過多，導致 overfitting。

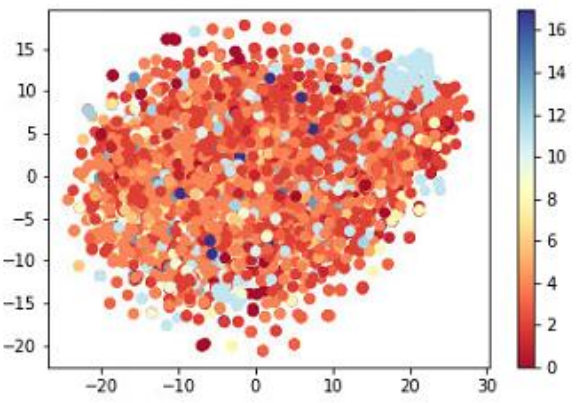
3. (1 %)比較有無 bias 的結果。(ref: 去年手把手)

   MF model(有 bias): public: 0.87487 private: 0.86755

   MF model(無 bias): public: 0.87545 private: 0.86861

   可以發現有 bias 雖然成績會比較好，但沒有影響那麼多。

4. (1 %)請試著將 movie 的 embedding 用 tsne 降維後，將 movie category 當作 label 來作圖。(ref: 去年手把手)

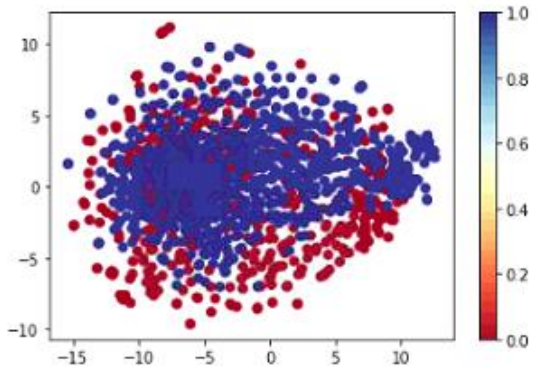



```
{'Action': 3,
 'Adventure': 1,
 'Animation': 0,
 "Children's": 8,
 'Comedy': 2,
 'Crime': 6,
 'Documentary': 9,
 'Drama': 4,
 'Fantasy': 16,
 'Film-Noir': 14,
 'Horror': 11,
 'Musical': 17,
 'Mystery': 13,
 'Romance': 7,
 'Sci-Fi': 10,
 'Thriller': 5,
 'War': 15,
 'Western': 12}
```

PS: 當有超過一個 category 時，取第一個 category。(MF model with bias)

效果並沒有很好，沒有把不同的 category 分堆，可能是因為只取第一個 category 的關係。

如果分成[Adventure, Animation, Children's]，[Horror, Crime, Thriller]兩類，其他不加進去。




```
type0 = ['Adventure', 'Animation', 'Children\'s']
type1 = ['Horror', 'Crime', 'Thriller']
```

效果還是不好，可能還是因為取第一個 label 的關係。

5. (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。

   方法：將 movie.csv 裡面的 Genres 拿來用，因為每一個 movie 可能會有很多種種類，所以將此做成一個 shape=(18,1)的 one hot encoding 的 np.array 加入 nn model 的 input layer 去 train。

Layer (type)	Output Shape	Param #	Connected to
input_4 (InputLayer)	(None, 1)	0	
input_5 (InputLayer)	(None, 1)	0	
embedding_3 (Embedding)	(None, 1, 100)	604100	input_4[0][0]
embedding_4 (Embedding)	(None, 1, 100)	395300	input_5[0][0]
flatten_3 (Flatten)	(None, 100)	0	embedding_3[0][0]
flatten_4 (Flatten)	(None, 100)	0	embedding_4[0][0]
input_6 (InputLayer)	(None, 18)	0	
concatenate_2 (Concatenate)	(None, 218)	0	flatten_3[0][0] flatten_4[0][0] input_6[0][0]
dropout_5 (Dropout)	(None, 218)	0	concatenate_2[0][0]
dense_5 (Dense)	(None, 150)	32850	dropout_5[0][0]
batch_normalization_4 (BatchNorm	(None, 150)	600	dense_5[0][0]
leaky_re_lu_4 (LeakyReLU)	(None, 150)	0	batch_normalization_4[0][0]
dropout_6 (Dropout)	(None, 150)	0	leaky_re_lu_4[0][0]
dense_6 (Dense)	(None, 150)	22650	dropout_6[0][0]
batch_normalization_5 (BatchNorm	(None, 150)	600	dense_6[0][0]
leaky_re_lu_5 (LeakyReLU)	(None, 150)	0	batch_normalization_5[0][0]
dropout_7 (Dropout)	(None, 150)	0	leaky_re_lu_5[0][0]
dense_7 (Dense)	(None, 50)	7550	dropout_7[0][0]
batch_normalization_6 (BatchNorm	(None, 50)	200	dense_7[0][0]
leaky_re_lu_6 (LeakyReLU)	(None, 50)	0	batch_normalization_6[0][0]
dropout_8 (Dropout)	(None, 50)	0	leaky_re_lu_6[0][0]
dense_8 (Dense)	(None, 1)	51	dropout_8[0][0]

```
Total params: 1,063,901
Trainable params: 1,063,201
Non-trainable params: 700
```

結果：

public score: 0.86752 → 0.86103 private: 0.85880 → 0.84892，有明顯進步。