1. **(1 %)請比較有無 normalize 的差別。並說明如何 normalize.**

	private	public
Normalize	0.85538	0.86342
Not normalize	0.87019	0.87654

對 Y (rating) normalize，從結果可看出有 normalize 的預測較準確。

2. **(1 %)比較不同的 embedding dimension 的結果。**

	private	public
100	0.85263	0.85888
250	0.85205	0.85979
500	0.85687	0.86345

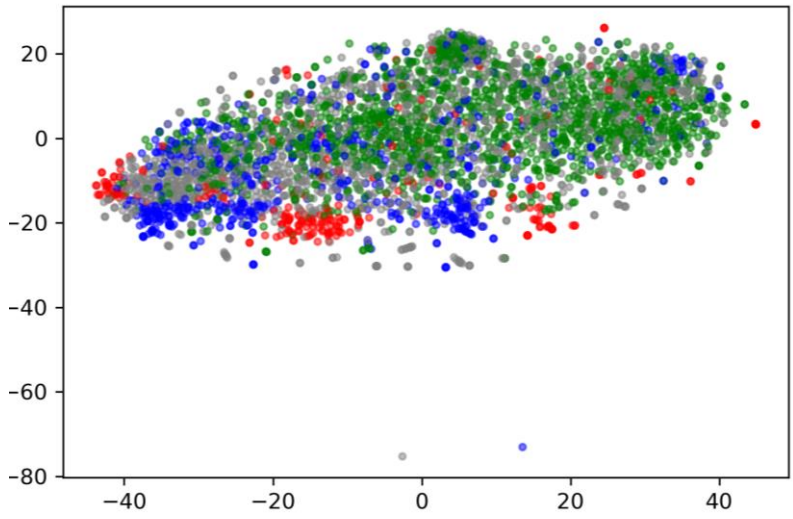
Dimension 為 100 和 250 的結果差不多，250 在 private 的表現上略好；但 dim 到 500 時，RMSE 反而上升。

3. **(1 %)比較有無 bias 的結果。**

	private	public
bias	0.84703	0.85212
Without bias	0.85332	0.85960

加了 bias 的效果較好，可能原因為每個人的評分標準不一，加上 bias 後可針對每個人做調整。

4. **(1 %)請試著將 movie 的 embedding 用 tsne 降維後，將 movie category 當作 label 來作圖。**



Legend
red: ['Animation' "Children's"]
green: ['Drama' 'Musical']
blue: ['Thriller' 'Horror']
grey: [others]

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。**

-added feature: age

-作法：將 age embedding 之後加入最後 merge layer

```
# user & movie
input_user = Input(shape = (1,))
user_emb = Embedding(user_num, emb_dim, input_length = 1, embeddings_regularizer=l2(1e-5))(input_user)
user_vec = Flatten()(user_emb)

input_movie = Input(shape = (1,))
movie_emb = Embedding(movie_num, emb_dim, input_length = 1, embeddings_regularizer=l2(1e-5))(input_movie)
movie_vec = Flatten()(movie_emb)

# feature
input_age = Input(shape = (1,))
age_emb = Embedding(age_num, 1, embeddings_regularizer=l2(1e-5))(input_age)
age = Flatten()(age_emb)

# bias
user_b_emb = Embedding(user_num, 1, embeddings_regularizer=l2(1e-5))(input_user)
user_bias = Flatten()(user_b_emb)

movie_b_emb = Embedding(movie_num, 1, embeddings_regularizer=l2(1e-5))(input_movie)
movie_bias = Flatten()(movie_b_emb)

dot = Dot(axes=1)([user_vec, movie_vec])
out = Add()([user_bias, movie_bias, dot, age])
```

-結果：

	private	public
Without age	0.84703	0.85212
With age	0.84782	0.85395

沒有加入 feature 的結果較好，可能原因為 age embedding 的結果不好，或是 merge 的架構應該用改為 concatenate 後再 add。