學號：r06946004 系級： 資科學程碩一 姓名：蔡尚錡

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

說明：

對training data的Rating值做normalize，首先用np.mean()計算所有Rating的平均數，接著用np.std()計算所有Rating的標準差，把原來的所有Rating減去平均數再除以標準差，得到的結果作為training data的新target，testing時再把prediction的結果乘上先前training時算出的標準差，接著加上training時算出的平均數，得到最後的output。

下方的比較數據是使用簡單的 MF model，latent dims = 200，optimizer = adamax，batch_size = 128 ， epochs = 10，兩者只有是否加上normalize的差別。

無normalize的MF model -> Public RMSE : **0.86217**, Private RMSE : **0.86563**

有normalize的MF model -> Public RMSE : **0.87340**, Private RMSE : **0.87388**

從數據上看起來，加上normalize之後會有少許的退步跡象，有可能是因為 normalize 之後會讓 data 彼此之間的差距變小了，導致 model 無法有效地將他們分開，才導致退步跡象。

2.比較不同的latent dimension的結果。

以下的表格比較的是不同的 latent dimension對MF model 在RMSE上面的影響，我使用的是簡單的MF model，有bias但沒有加上normalize，optimizer = adamax，batch_size = 128，epochs數目則會隨著latent dimension 的不同而收斂在不同地方，latent dimension 則從 16 開始以翻雙倍的方式一路增加到 1024 看看不同維度的影響。

latent dimension	Public RMSE	Private RMSE
16	0.86280	0.86635
32	0.86194	0.86549
64	0.86228	0.86485
128	0.86273	0.86435
256	0.86468	0.86715
512	0.86646	0.87130
1024	0.87795	0.88053

從數據結果可以看到，latent dimension 從16 到 128 的過程，private RMSE會一路下降，public RMSE 則會先下降再上升，然而其實在這個範圍內都會保持著還不錯的準確率，變動幅度不大，但是在 128 維度之後，維度繼續上升，會導致不管是 public RMSE 還是 private RMSE 都有不小的上升幅度，甚至到 1024 維度之後整個準確率就不是很好了，所以也沒必要往後比較。

3.比較有無bias的結果。

下方的比較數據是使用 MF model，沒有加上 rating 的 normalize，參數設定 latent dims = 128，optimizer = adamax，batch_size = 128 ， epochs = 12，兩者只有是否加上 bias 的差別。

有 bias 的MF model -> Public RMSE : **0.86140** , Private RMSE : **0.86225**

無 bias 的MF model -> Public RMSE : **0.86519** , Private RMSE : **0.86825**

由數據可以看到，加上 bias 之後，準確率不管在 private 或是 public 都有顯著的提升，原因是不同的 user 可能會有自己給電影評分的的傾向，例如有些人就會傾向都給予高分的評分，或是不論什麼

類型都只給偏低的評分；同樣的，不同的電影也會也這樣的趨勢，可能某些類型的電影題材或類型通常不被普羅大眾所接受，可能就會獲得偏低的評分，因此加上 bias 是有助於我們確切預估評分。

4.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。

說明：

我所實作的 NN model 是把 user ID 的 embedding 跟 movie ID 的 embedding 藉由 keras.layers.Concatenate() 合併起來之後丟進一個 DNN 作為 input，output 則是把它當作一個 regression 的問題預測是 1 到 5 哪個選項，而 DNN 的架構我選用三層 hidden layer，第一層大小是 512，第二層大小是 256，第三層大小是 128，加入Dropout 預防 overfitting。

下方的比較數據 MF model，是沒有加上 rating 的 normalize，參數設定 latent dims = 128，optimizer = adamax，batch_size = 128 ， epochs = 12，並加上 bias ; 而 NN model 則是依照上述的實作方法的架構，但是 latent dims = 256，optimizer = adam，batch_size = 128 ， epochs = 8

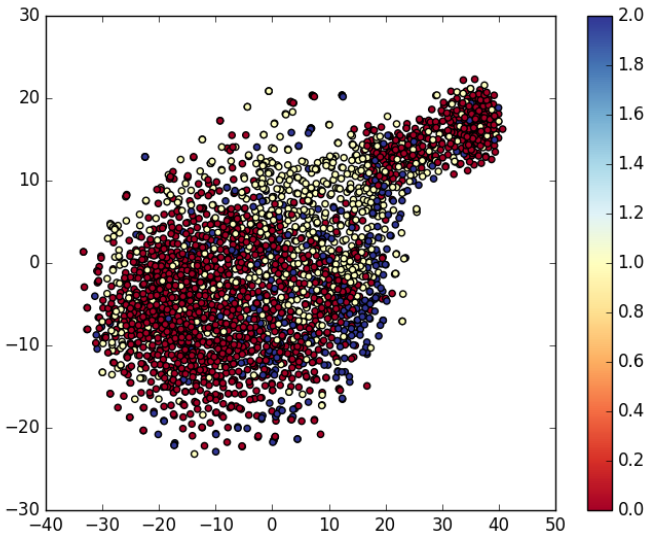
使用 MF model -> Public RMSE : **0.86248**, Private RMSE : **0.86380**

使用 NN model -> Public RMSE : **0.86581**, Private RMSE : **0.86625**

從結果看來似乎還是用MF model的準確率會比較高一些，但使用NN model 也可以分別超過 public and private strong baseline，照理來說，在 NN model 的參數通常會比 MF model 參數量來的比較多的情況下，應該可以學到比較細微的 feature ，讓準確率來得比較好，不過由於我還沒有辦法找出一組可以超越 MF model 的 NN model 的參數和架構，才會讓結果顯得 NN model 表現不如預期。

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

這張圖我是把Drama、Musical 作為一類，把 Thriller、Horror、Crime 作為一類，另外則是 Adventure、Animation、Children's 做為一類所畫的圖。由此可見Drama、Musical 相似的音樂戲劇類型分散在左下角紅色，而犯罪懸疑驚悚類則分布在右上米白色，最後是小孩子類型的冒險動畫片則是在右下方的藍色，三種類別分別由於彼此之間電影的類型相同，會讓 embedding 之後的分佈落在差不多範圍的區域，當然由於這個 dataset 電影的種類分很細，所以似乎會有一些分不是很清楚的情況，但大致上的趨勢分佈還算清楚。



(BONUS)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。

我把 user data 裡面的 gender 、age、occupation 分別取出都當做是一種 id ，然後來做 embedding，再跟原來的 UserID embedding 合併起來，作為新的input feature，通過一個 DNN，而另一個movieID embedding 也通過一個另一個 DNN 再把兩個DNN分別的 output vector 做內積得到我想要的rating值。用這個方法來做training，最後得到的結果 private RMSE = 0.86872，感覺沒有特別的進步，可能還是取到了比較不重要的 feature，或是參數無法取到最佳。