## CS5691 - PRML
## Music Recommender System Course Project

# Code Explanation

**Shreyas Kulkarni, Kshitij Shah**
**ME19B193, CS19B027**

# Part I: Setting up the code

Line 1-13:
Importing all the required libraries

Line 14-15:
Setting the sample seed at 42 so that we can receive the same results when the code is run again.

Line 16-18:
Remove any warnings we receive from the code so as to specifically focus on the output.

## Part 2: Data Cleaning and Feature Engineering

Line 19:
Importing train.csv into a pandas data frame variable named train.

Line 20:
We were trying to figure out whether there is any pattern associated with the way the Customer ID was allotted to each song, so we created a histogram to look for patterns with the first character of each Customer ID. However, it looked like the Customer ID was very much uniformly distributed and that feature was not very useful.

Line 21-22:
Imported songs.csv into a data frame variable named songs

Line 23:
Similar to Line 20, but here we have experimented on Platform ID

Line 24:
Merged the song's metadata with the train so that in one table we can have all the data we need.

Line 25-26:
Some songs in the train.csv were not there in the metadata so the number_of_comments was replaced with 0 when it was null in the table to avoid errors later.

Line 27-54:
The purpose of these manipulations was to generate the average of all ratings, number of comments, and any other continuous variables with respect to the customer and the song separately as columns in our train dataframe - to later mean adjust them.

Line 55-80:
Similar manipulations have been performed for the test data frame but by using averages from the train dataset instead of the test dataset to avoid errors.

Line 81-83:
Saving the variables as they would undergo many changes in the upcoming steps.

We worked on various methods and our plan was that we would create an ensemble of different techniques and not rely on one methodology.
After our first week, we concluded on using an ensemble of Collaborative Filtering (methods that rely on using ratings-based data only) and Content-Based Filtering (Methods that use customer and song-based characteristics). This intuition mainly came from Netflix Challenge

# Part 3: Content-Based Model

We created multiple features for songs and users by utilizing all the metadata. These features were created by using pivot tables and then using PCA over them so as to extract denser features for our Machine Learning Model (XGBoost in our case). We came up with this idea on our own and thus do not believe this has been used by anyone else/plagiarised.

Line 84-89:
Importing data frames.

Line 90-91:
Train val split. The major reason for doing this here is that we would need the val set to tune the parameters for the final Ensemble model.

Line 92-93:
Pivot to create features for every song from the song labels csv

Line 94-96:
Pivot to create features for every customer and every song using the save_for_later.csv
Here, the pivot shall have values in 0 or 1 - 1 if the song has been saved for later by the customer else 0, similarly for songs.

Line 97-100:
Pivot to create features for every customer and song using train.csv ratings

Line 101-127:
As these features were very large in size and sparse, we used KernelPCA for every such feature generated and tuned the parameters.
The number of components was decided by using the average number of non-zero entries in the features.

Line 128-165:
Motivation for the following idea: Smart Music Rating Predictor
We have created features such as customer rating count, variance, standard deviated etc. These features could give us insights into other patterns followed by the ratings. Similarly for Songs.

Line 166-203:
Compilation of all the features that we have created in Line 101-127 and Line 128-165.

Line 204-217:
XGBRegressor Model creation, training on the train part of the train dataset and prediction on the val part of the train dataset.

# Part 4: Matrix Factorization Method

This next idea has been motivated by this paper: Repeated Matrix Reconstruction and the code snippet has been inspired by - Netflix Data Challenge.
This might be the place where plagiarism has been detected.

Intuition: In matrix factorization, it is difficult to differentiate between rated and non-rated entries when creating low-rank approximations. Repeated Matrix Reconstruction repeatedly uses Matrix Factorization by assuming that the unrated entries are equal to the average of the ratings given to the song before using SVD. As these approximations are not guaranteed to preserve the rated entries, all the known entries are reset to their original values. A new low-rank matrix is made from this matrix and the process continues till convergence.

Line 218-224:
The song_matrix is created which would contain all the predicted ratings of the user-song pair. Initially, it just contains the matrix from the train dataset.

Line 225-229:
To remove the song bias, we subtract by the song average.

Line 230-232:
A mask is created to identify where we have the ground truth of ratings.

Line 233-239:
SVD transformed the matrix into smaller matrices. Later we inverse transform to get a reconstructed matrix

Line 240-241:
The loss is just on the train set

Line 242-248:
Unless it's the last iteration, after every iteration, the train part of the matrix is again added back to the reconstructing matrix. This is the main trick behind the repeated matrix factorization which helps in preserving already rated entries from the train set.

Line 249-252:
Again mean shifting back the predictions

Line 253-261:
Tuned the parameters so that we don't depend on single construction and then later took the average.

Line 262-270:
Predicting to add it back to the Ensemble matrix

## Part 5: Ensemble Training

Here, we have rating predictions on the validation set, we use these predictions from the Content-based model and Matrix Factorization Method. Now we use them to create a Linear Regression model to predict the final ratings.

Line 271-276:
Training and Prediction using Linear Regression

## Part 6: Training on the entire Train Set

Line 277-423:
We had done the above training and predictions by splitting the train set. This was done so that we can train the Linear Regression model. Now, we train the Matrix Factorization and Content-Based Model on the entire train dataset and then finally use the linear regression model for final predictions on the test dataset.

Other references:


1. Stanford Paper on Repeated Matrix Reconstruction
   https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjV04qH_OnxAhXljuYKHeapCA4QFjABegQIBhAD&url=http%3A%2F%2Fcs229.stanford.edu%2Fproj2006%2FKleemanDenuitHenderson-MatrixFactorizationForCollaborativePrediction.pdf&usg=AOvVaw1tZ2VbdZeIa9kQA2Hmrjsy
2. Wikipedia - Collaborative Filtering
   https://en.wikipedia.org/wiki/Collaborative_filtering
3. Paper on Smart Music Rating Predictor
   ChaoYangSong-SmartMusicRatingPredictor.pdf
4. Movie Rating Predictor
   AutomaticMovieRatingsPrediction_MIPRO.pdf
5. How to improve recommendations for highly sparse datasets using Hybrid Recommender Systems?
6. AI for Socializing: Build Recommendation Engines Using PredictionIO
7. Introductory Blogs
   In-Depth Guide: How Recommender Systems Work | Built In
   An Easy Introduction to Machine Learning Recommender Systems - KDnuggets