CSE 158 Assignment 1 Report

Shihao Shen, PID: A15425727, Kaggle Username/Team Name: **KelvinSh**

Task 1: Play Prediction

In this task, given a (user, game) pair, we need to predict whether the user would play the game or not. Accuracy is measured in terms of the categorization accuracy (fraction of correct predictions).

First approach: conducting a massive hyperparameters tuning based on my HW3 solution. In the HW3 solution, we used Jaccard Similarity as well as Popularity combined to make predictions. For each feature, we tried to run it on validation set with different thresholds so that we could find the optimal one. However, we did it separately; that is, we found the threshold for Popularity alone and the threshold for Jaccard Similarity alone, and then we simply combine two features using the independent thresholds. This is not reasonable if we tried to combine two into one single model. Therefore, I conducted a massive hyperparameters tuning on both thresholds to find the optimal pair. I tuned Popularity threshold from 0.4 to 0.85 with a step size of 0.01, and tuned Jaccard Similarity threshold from 0.024 to 0.044 with a step size of 0.002. The lower and upper limits are chosen around the independent optimal threshold found in HW3. I also adopted two ways to combine the model:

1. If Popularity predicts positive and Jaccard predicts negative, trust Popularity
2. If Jaccard predicts positive and Popularity predicts negative, trust Jaccard.

I started by testing on the test set using the two methods with the best thresholds found. The second method gave higher accuracy in general, so I adjusted very small changes to each threshold around 0.84 and 0.03 as given from in *accuracies2*. I finally found 0.83 for Popularity and 0.028 for Jaccard Similarity that would give me the highest accuracy of 70.84%, which ranked me around 200/590.

Second approach: This has shown that a single combination of Jaccard and Popularity would must reach the bottleneck regardless of how much tuning I conducted, so I need to adjust my approach from the bottom up. Changes I have taken can be summarized as follows:

1. Discard the threshold tuning so that now I have validation and training sets together to build my dictionary. This would give a larger training set which might improve the accuracy.
2. Compute more "general" Jaccard Similarity based on all the data. That is to say, given a pair of user and game, $(u,\ g)$, from the test set, for each game g' that the user u has played, find the union of all the users that have played each game g' played by u. And similar to the computation in HW3, find another union of all the users who have played g. Finally compute the Jaccard similarity of these two unions as my new Jaccard Similarity values used for prediction.

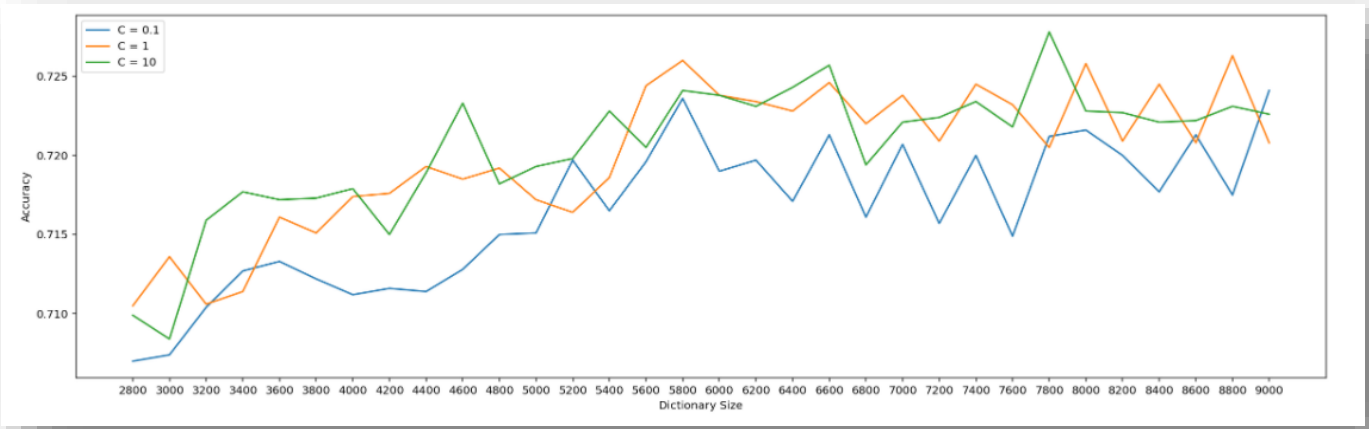
3. Instead of using Popularity or the threshold for similarity, I simply sort the Jaccard similarities for each game played by each user, and check if the game to predict appears in the first half or the other half.

Result: Public Leaderboard 72.53% (69/672); Private Leaderboard 72.81% (49/672). I believe this is attributed to a more general and comprehensive computation of Jaccard similarity and a larger training set.

Task 2: Category Prediction

In this task, we are expected to predict the category of a game given a review for that game. Five categories are used for this task, namely Action, Strategy, RPG, Adventure, and Sport, each encoded from 0 to 5. Performance is measured in terms of the fraction of correct classifications.

Frist approach: In HW3, we built a dictionary of size 1,000 that contains 1,000 most common words across all reviews in the training set. Then we built bag-of-words feature vectors by counting the instances of these 1,000 words in each review. Finally, we trained a Logistic Regression model with a chosen regularization constant C on the feature vectors from each review as well as the game category it corresponds to. Therefore, I conducted a massive tuning of parameters to find the best logistic regression model. I tuned S from 2,800 to 9,000 with a step size of 200 on three possible regularization constants $\{0.1, 1, 10\}$ for the logistic regression model. I plotted my massive tuning for a better interpretation as below. A larger dictionary size is able to increase the performance before it reaches 5,800 because the overall trend in the left half is increasing. After it reaches 5,800, the accuracy mostly maintains around 72.5% for $C = 1$ and $C = 10$. Therefore, we can assume that it's not the case that a larger dictionary size is bound to give better performance. When it reaches a specific size, the bag of words is expected to contain enough keywords that give insight into what a game category might be, so any further increment in the dictionary size might not significantly increase the accuracy any more.



Finally, indeed, the model with dictionary size 7800 and regularization constant 10 gave the best accuracy of 72.58% on test set. However, this seems to be the bottleneck of merely tuning parameters. I decided to change my approach.

Second approach: Using TF-IDF to build my features.

TF-IDF refers to Term Frequency Inverse Document Frequency, which is a measure of originality of a word by comparing the number of times a word appears in a document with the number of documents the word appears in. Here the review text is just our document. Using TF-IDF, we can find the relevance of a word in a review, instead of using a bag of the most frequent words that contain a lot of meaningless words, which is both computational heavy and inefficient. Now frequently occurring words will have little impact on the similarity between a review to predict and all reviews used to train. The similarity is now determined by the words that are most "characteristic" of the review.

I used *TfidVectorizer* from *sklearn* to Convert a collection of reviews to a matrix of TF-IDF features. I applied sublinear tf scaling. I set *min_df* as 5 to ignore words that appear in less than 5 reviews. I extracted unigrams and bigrams by setting *ngram_range* to be (1,2). Due to the memory limit on the datahub, I could only convert the first 140,000 reviews from the training set.

I trained Logistic Regression, Random Forest, and Linear Support Vector Machine on the features extracted. The Linear SVC from *sklearn* appears to be the best one with the highest validation accuracy. Therefore, I chose LinearSVC as the model for prediction.

Result: Public Leaderboard 75.34% (78/401); Private Leaderboard 75.98% (146/401). Note that if I had used the same approach but with all 175,000 reviews for TF-IDF vectorization, I would get 76.5% on hidden tests which could rank me at 96.