# CS506 Midterm Report - Lingyan Jiang

## Kaggle Name - Lingyan Jiang

The midterm of class CS506 is to predict the star rating associated with user reviews from Amazon Movie Reviews using the available features such as 'Text', 'ProductId', 'UserId', 'HelpfulnessNumerator', 'HelpfulnessDenominator' and 'Summary'. During this exam, I used machine learning algorithms such as linear regression, naive bayes, decision tree, random forest, and SVM to try to get better predictions with higher accuracy.

- **Data Preprocessing**

For this train data, there are a few important steps of data preprocessing that need to be mentioned.

1. **Missing data filling** - In the train data, there are a few missing values in both summary and text features. Instead of using the dropna() function to drop the rows of all the missing values, I used summary value to fill the missing text value and vice versa. Thus, all the data can be kept and used in the training process.
2. **NLP clean processing** - For this training process, I used the text feature instead of summary because I tried both features in the training process and the summary will yield a lower accuracy. For the text feature, it is raw data that can not directly be put into the training process. So I cleaned the text data by removing numbers, splitting and lowercasing, deleting the stop words from the 'English' library, stemming words, and removing punctuation.
3. **Data Transformation** - Although the text data has been cleaned up, the data are still not prepared for the training process. So I transformed the data as follows. For the Product Id and User Id data, I chose to project one same value to one number instead of using the one-hot-encoder, due to its sparsity. For the text data, I used TfidfVectorizer to vectorize the text data and use transform language to store the text data as csr_matrix to save the memory.

- **Training Model Process**

For the training process, I chose the 'Text', 'ProductId', 'UserId', 'HelpfulnessNumerator', and 'HelpfulnessDenominator' features. I dropped the 'time' feature because its interaction with scores is not so strong and I tried to add the 'Time' feature into the training process, but the accuracy became lower. As to the algorithms, I tried linear regression, naive bayes, decision tree, random forest, and SVM. I will explain more details as follows.

1. **Train test split and K fold** - I used the train_test _split() function to split the train dataset and set test_size as 0.2. In addition, I used KFold language to split the data

into 5 folds and train each fold into the algorithms to prove the accuracy. Due to the slow iteration time, I only tried KFold in the SVM and linear regression algorithms. With the help of KFold, the accuracy did improve but not as significantly as I thought.

2. **Modeling** - It's my first time training the data with the sparse matrix, so I chose different algorithms to see its accuracy. I tried linear regression, naive bayes, decision tree, random forest, and SVM. But their accuracies were very different from each other. Random forest and naive bayes did not perform as well as I thought and their accuracies were lower than others. Linear regression, SVM, and decision tree performed similarly in terms of accuracy. I chose the linear regression with the highest accuracy which reacted at 0.71.
3. **Accuracy** - Because this exam will grade the result with mean_squared error, I imported mean_squared_error to calculate the accuracy.

- **Prediction and Submission**

1. **Missing value filling** - There are still some missing values needed to fill in the prediction process. As I mentioned above, I used summary value to fill the missing text value and vice versa. But during the cleaning data process, the text data may still become NaN, because the filling value is summary and the summary is way too short compared with the text. Instead of using a KNN imputer, I used the same algorithms to train the summary data and filled the missing score value with the value obtained from summary data training.
2. **Prediction score transformation** - After using linear regression as the best algorithm, its prediction is decimal with different values instead of integer five-star scores. So I wrote a for loop to change the prediction to an integer in an interval between [0,5]. The reason why I didn't use the round() function is that even if I use the round() function, the integer score may still be over 5.
3. **Submit and Result** - I submitted several times to prove which algorithm would be suitable for this exam. And the best score I got was 0.87852.

- **Pitfall and Challenge**

1. **Pitfall** - I solved a pitfall when dealing with the TfidfVectorizer to vectorize the text data. At first, I used the to_array() function to change the text feature to the array to train in the model. But during the training process, the server always interrupted the activity and stopped working. In order to solve this problem, I found the transform() function to change the text data into crs_matrix so that it saves lots of memory and iteration time.
2. **Challenge** - Although I tried several algorithms, the accuracy didn't reach 0.8. After this exam, I plan to use CNN to see if the accuracy will be a little better and I hope Professor can still open the kaggle so that we can test whether our result will be better or not.