**CPE695 Final Project Report  
Group Member: Mu,Yu; Pang,Liangfan**

**Abstract:** ​ Currently, MovieLens allow users to associate each movie with a set of arbitrary words, colloquially called tags. Tags often describe the movie’s genres, but can also be any other attribute the user desires. One common application with genre tags is to find other movies with similar tags to generate a list of user-specific recommendations. However, we noticed a potential shortcoming of the current tag system: since all tags are user-defined, they are prone to user error and subjectivity. This raises the question, is there a better way to generate tags or give predictive ratings? This project explores ways to predict ratings and recommend movies better: given the features of the movie itself, we try to utilize a movie’s title and genome-scores to predict its rating.

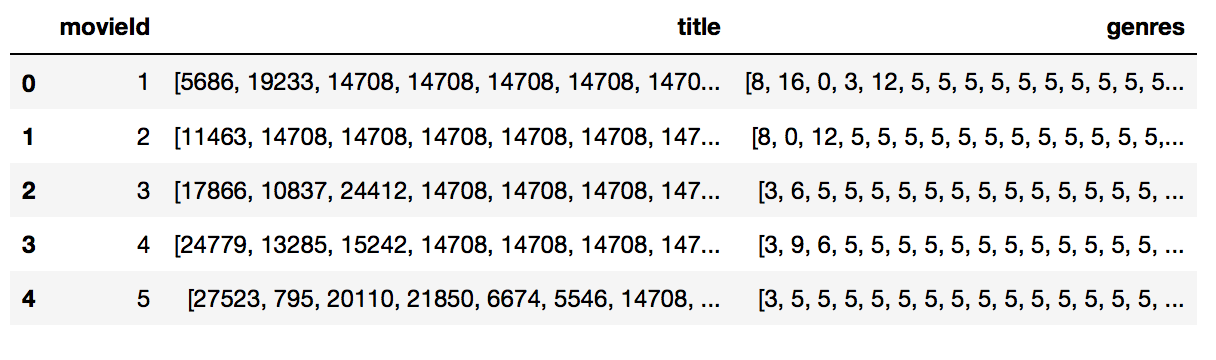
**Introduction:** For this project, We first encoded text datasets using regular expression and calculated digital datasets using functions in python. Then, for each encoded dataset, we built a function to merge the code into a float number. Then we use three machine learning algorithms(SVM, KNN and DecisionTree) to train this model. For each movie, we built a classifier by retrieving training examples, each composed of a movie’s average rating and its associated feature(title, tags, movieID and genome-relevance). To test the classifier, we reserved another set of movies, and used our classifier to predict their ratings. Finally, we determined accuracies by comparing the predictions to their known ratings in the database.

**Solutions:**

1. **Datasets and data processing:**

For datasets, we used the publicly available (and legally produced) MovieLens Dataset[ml-20m]. This dataset describes 5-star rating and free-text tagging activity from [MovieLens](http://movielens.org), a movie recommendation service. It contains 20000263 ratings and 465564 tag applications across 27278 movies. These data were created by 138493 users between January 09, 1995 and March 31, 2015. Users were selected at random for inclusion. All selected users had rated at least 20 movies. No demographic information is included. Each user is represented by an id, and no other information is provided. The data are contained in six files, `genome-scores.csv`, `genome-tags.csv`, `links.csv`, `movies.csv`, `ratings.csv` and `tags.csv`.

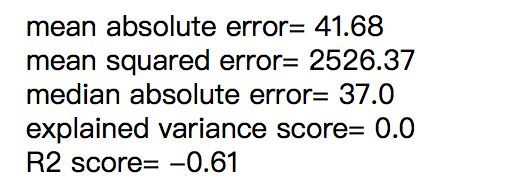
We split the datasets to two parts—80% of them as training samples and 20% of them as test samples. In our data cleansing, firstly, we imported six csv datasets reading as string for the accuracy of data observations. Then we adjusted some data types from string to float and int such as relevance and ratings. and checked for logic error such as whether there is negative ratings and relevances. Then we Uniform the format of text data—movie title. We removed redundant spaces and symbols. Then we remove the missing data in tags, movies, and ratings. In the sixth step, we encoded two text datasets using regular expression. For the digital datasets, we calculated the average ratings for every movie. And then the digital features were multiplied by 100 or 1000 to become integer to train model use function in sklearn. And at last, datasets were joined to one dataframe.



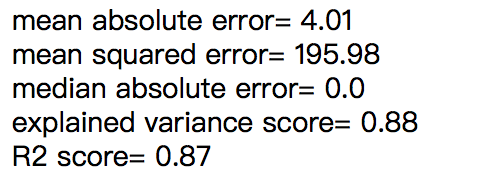
1. **Algorithms and processing:**

In order to predict ratings, we trained three algorithms that would calculate ratings, given a list of movie-ratings and movies’ associated features. We used SVM algorithms, KNN algorithms and DecisionTree to construct the prediction. In all cases, one training examples was the features for one movie, so the set of training examples would be a large set of features for many movies. The output was the rating of this movie.

SVM: We first elected to use SVM algorithm. Compared to other methods, support vector machines make few assumptions about our data and provide theoretical performance guarantees. As a tradeoff, the computation speed is highly degraded for large feature sets. Thus, we chose to only use 50000 of the movies as our training samples. Three classes of kernels (linear, quadratic and RBF) were tested using the SMO algorithm and all provided similar results. Below is some results of accuracies for this model.



KNN: Then, we elected to use KNN algorithm. Compared to other methods, K-nearest neighbors algorithm can model complex target functions by a collection of less complex local approximations and the fact that information present in the training examples is never lost. Thus, we chose to use all of the training samples as our training samples. Below is some results of accuracies for this model.



DecisionTree: At last, we elected to use DecisionTree to train this model. We chose to only use 50000 of the movies as our training samples. As the result, this model is overfitting.

**Comparison of results**

According to the results, as a instance-based learning, we found that KNN algorithm has the best training effect and the shortest time. And the fitting effect of decision tree is not stable enough.

**Future research directions**

For the future research, we think we can make more efforts in three directions. One is using more detailed movie information to train this model, the other one is process the encoded text datasets using Convolutional Neural Network. The last one is writing our own algorithm code to use float data type of samples.

**Reference**

**Predicting Music Tags from Lyrics** Elmer Le/Quentin Moy/Jerry Zhou