CS 155 Group Project 2 Report

Joon Young Park, Derek Ing, Daniel Li, Hernan Caceres

February 2022

1 Introduction

1. **Group members**:

   Hernan Caceres, Daniel Li, Derek Ing, Joon Young Park

2. **Colab Link**:

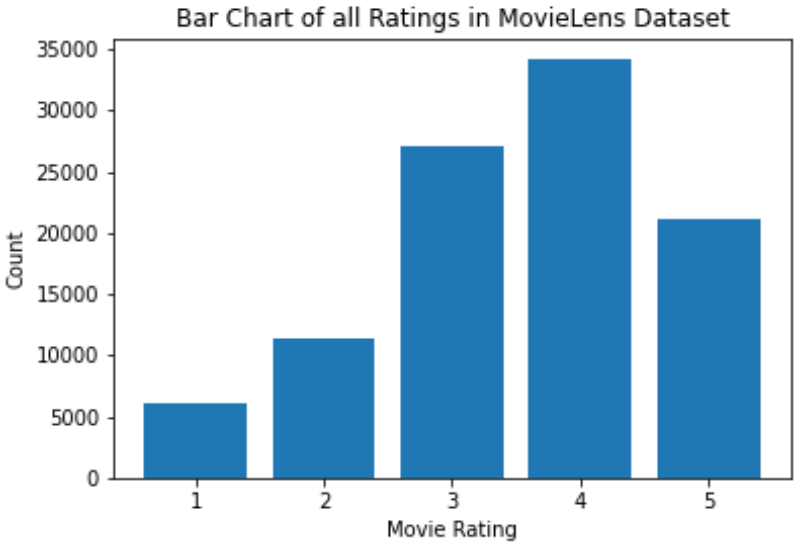
   Click here.

3. **Division of Labor**:

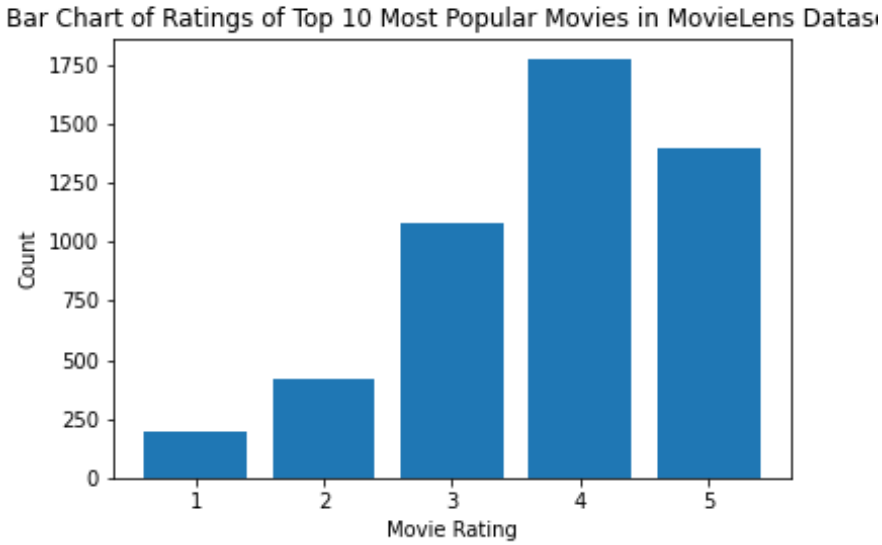
   (a) Derek Ing: Matrix Factorization 2
   (b) Daniel Li: Matrix Factorization 1
   (c) Joon Young Park: Basic Visualizations and GridSearch for parameters
   (d) Hernan Caceres: Matrix Factorization 3

2 Basic Visualizations

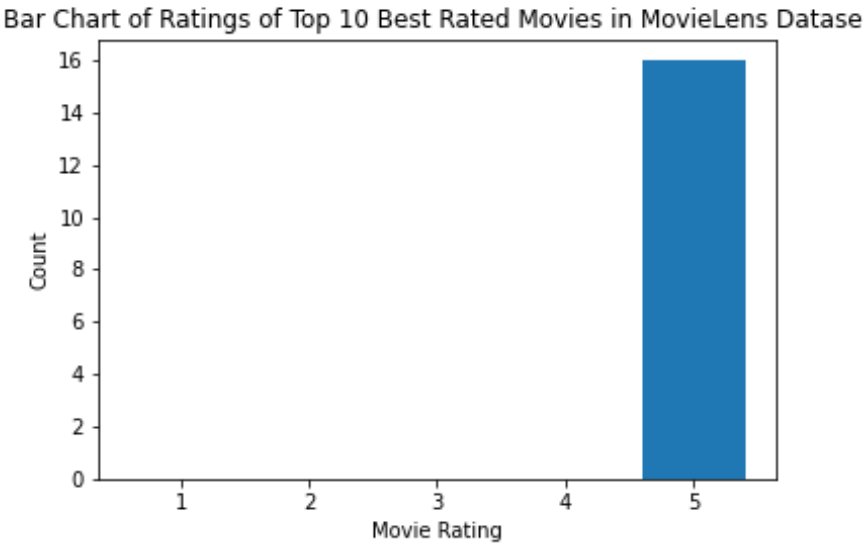
All ratings in the MovieLens Dataset



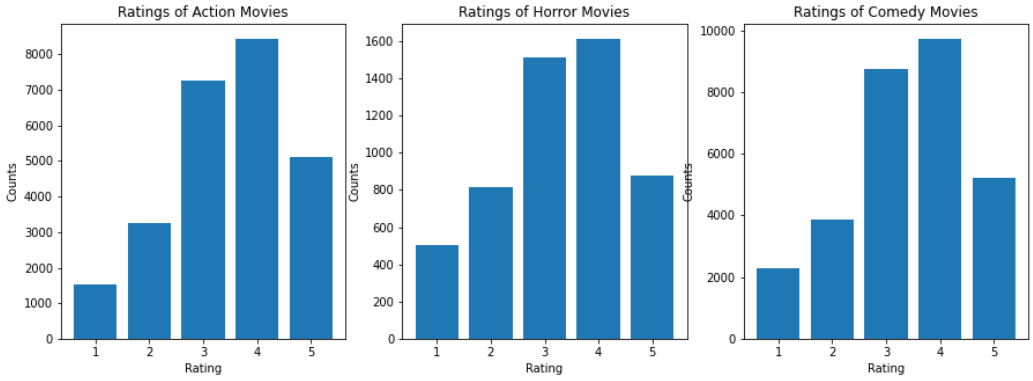
All ratings of the ten most popular movies



All ratings of the ten best movies



All ratings of movies from three genres of your choice



3 Matrix Factorization Methods

For method 1, we use collaborative filtering to construct two matrices, $U \in \mathbb{M}_{kxm}$ and $V \in \mathbb{M}_{kxn}$, to represent a matrix $Y \in \mathbb{M}_{mxn} \cong U^T V$. The matrix Y represents each user's ratings for each movie in the dataset, where $m = 943$ is the number of users, $n = 1682$ is the number of movies, and $k = 20$ is the number of latent factors. The element Y_{ij} represents user i's rating for movie j. In the given data set, users do not rate all the movies, thus we have unknown rating values. Thus, we use collaborative filtering to a learn latent representation of movies U and users V to explain the observed ratings, and thus predict all users ratings of all the movies. We learn a latent representation such that

$$\arg\min_{U,V} \frac{\lambda}{2}\left(\|U\|_F^2 + \|V\|_F^2\right) + \frac{1}{2}\sum_{i,j}\left(y_{ij} - u_i^T v_j\right)^2$$

where u^i and v^j are the ith and jth columns of U and V, respectively. We first initialize U and V matrices by randomly drawing samples from a uniform distribution between $(-0.5, 0.5)$. Then, for each epoch, randomly iterate through each data point, (i, j, y_{ij}), in the training set and update u_i and v_j by stochastic gradient descent. We calculate gradients

$$\delta u_i = \lambda u_i - v_j(y_{ij} - u_i^T v_j)$$

$$\delta v_j = \lambda v_j - u_i(y_{ij} - u_i^T v_j)$$

and update $u_i = u_i - \eta \delta u_i$ and $v_j = v_j - \eta \delta v_j$ accordingly, where we chose our learning rate, η. We implement an early stopping condition and stop at epoch t if $\Delta_{t-1,t}/\Delta_{0,1} \leq \epsilon = 0.0001$, where $\Delta_{t-1,t}$ is the loss reduction between epoch t and $t-1$ and $\Delta_{0,1}$ is the loss reduction between before training and after the first epoch.

For methods 2 and 3, we used the Surprise package's SVD algorithm. In method 3, we set the bias parameter to False so that there are no bias terms, and in method 2 we include bias terms. The Surprise SVD algorithm learns $U \in \mathbb{M}_{mxk}$ and $V \in \mathbb{M}_{nxk}$, to represent a matrix $Y \in \mathbb{M}_{mxn} \cong UV^T$.

For method 2, all biases are initialized to 0 and U and V are randomly initialized by drawing from a normal distribution with $\mu = 0$ and $\sigma = 1$. The predictions are set to $\hat{r} = \mu + b_i + b_j + u_i^T v_j$.
The algorithm also minimizes

$$\sum_{r_{ij} \in R_{train}} (r_{ij} - \hat{r}_{ij})^2 + \lambda(b_i^2 + b_j^2 + ||u_i||^2 + ||v_j||^2)$$

Surprise's implementation also uses stochastic gradient descent to update u_i, v_j, and the bias terms, b_i and b_j. The updates are performed by

$$b_i = b_i + \gamma(e_{ij} - \lambda b_i)$$

$$b_j = b_j + \gamma(e_{ij} - \lambda b_j)$$

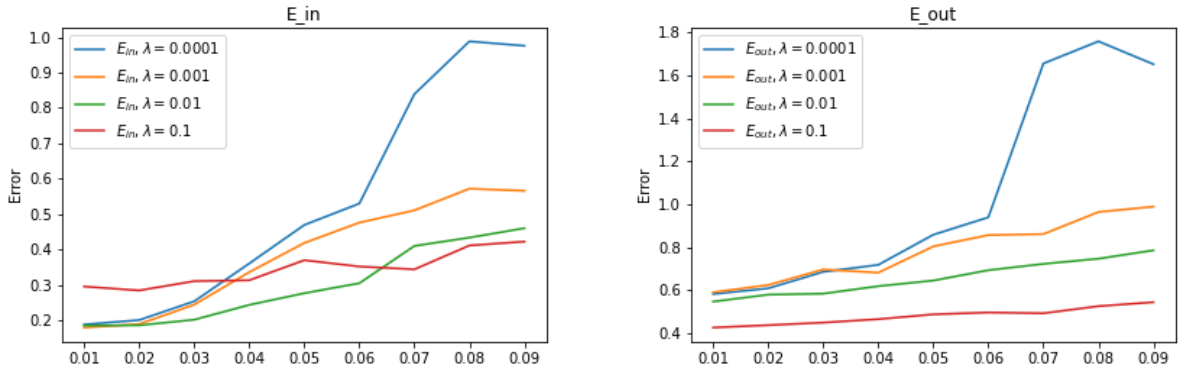
$$u_i = u_i + \gamma(e_{ij} v_j - \lambda u_i)$$

$$v_j = v_j + \gamma(e_{ij} u_i - \lambda v_j)$$

where $e_{ij} = r_{ij} - \hat{r_{ij}}$ and γ is the learning rate. The updates are performed for n epochs and there is no early stopping condition. This is the same for method 3 except biases are not included in the learning objective and are not updated.

For method 1, we trained for different values of λ and η to find the parameters that minimize unregularized mean squared error on the test set. For methods 2 and 3, we performed a grid search for different values of λ and γ to minimize root mean squared error.

Loss curves for method 1 matrix factorization, the x axis represents values of η:

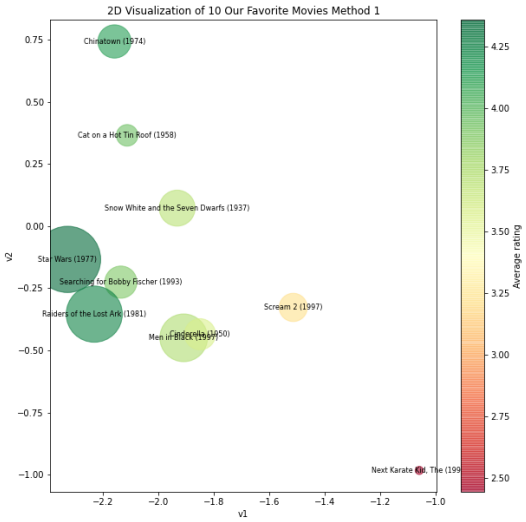


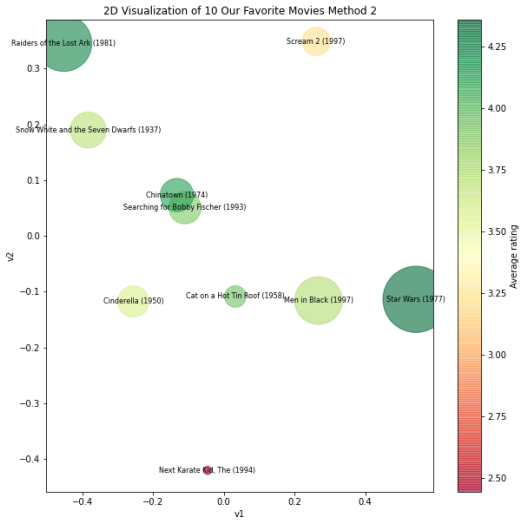
After training, our errors on the test set were: method 1 = 0.429, method 2 = 1.329, and method 3 = 1.430. Method 2 is different from methods 1 and 3 because it includes bias terms. Methods 2 and 3 also do not use an early stopping condition while method 1 does. Method 1 is different from methods 2 and 3 because we initialize U and V according to a uniform distribution as opposed to initializing according to a normal distribution.
The early stopping condition could attribute to method 1 performing better than method 2. With early stopping, we prevent the model from overfitting to the training set. Without early stopping, the model can overfit to the training set, and thus cannot generalize and perform well on the test set.

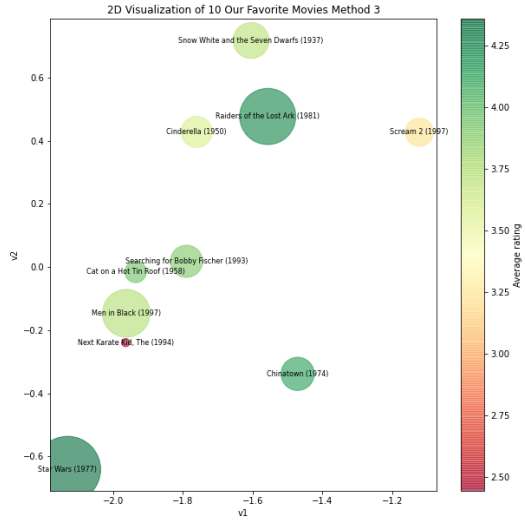
Between methods 2 and 3, the model with the bias term performs slightly better than the model without the bias term. Some movies are rated higher/lower on average compared to other movies. On the other hand, some users may give on average higher/lower ratings to movies compared to other users. Thus, to compare a user's ratings to other users, we do not want to take into account the bias of a user. The bias term can help us remove this user bias. In addition, when we make predictions for a user's rating of a certain movie, we can also take into account a certain user's rating tendencies. As a result, the bias terms can help the model better capture the interactions between the users and movies, and thus the model can perform better than without bias terms.

4 Matrix Factorization Visualizations

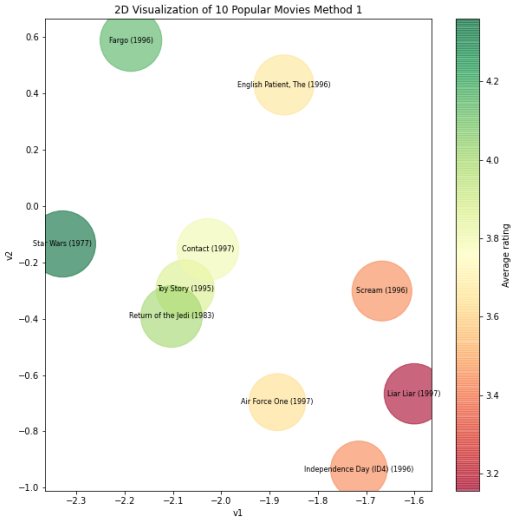
10 Movies of Our Choice

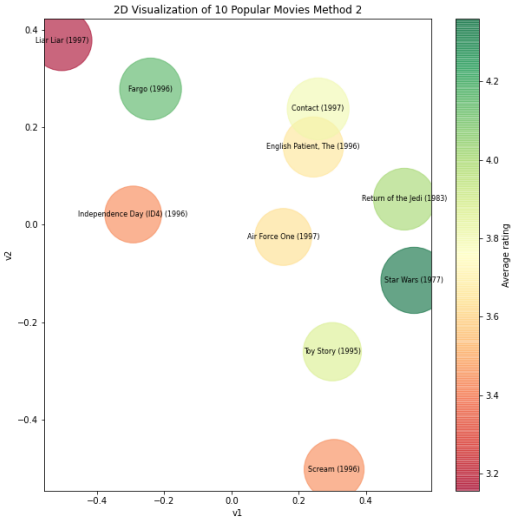


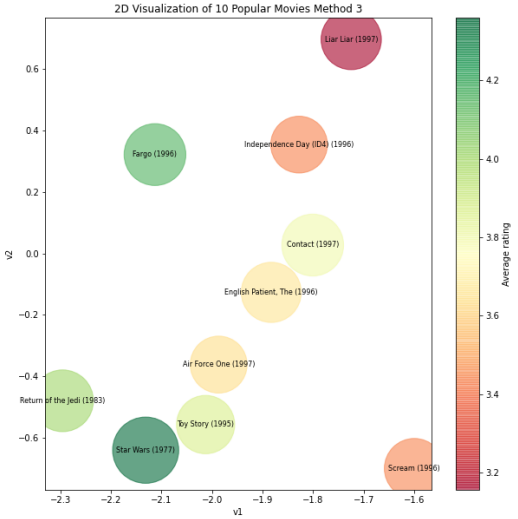




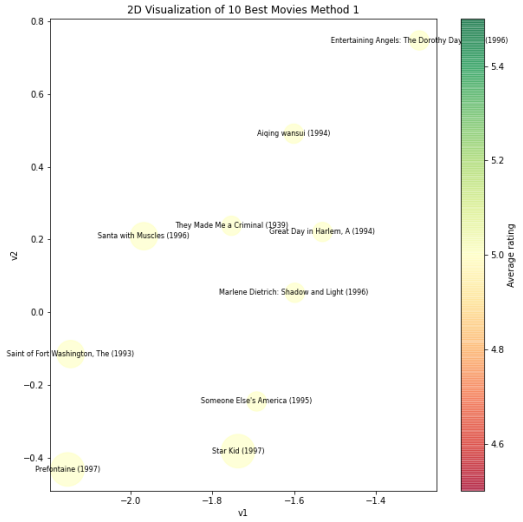
10 Most Popular Movies

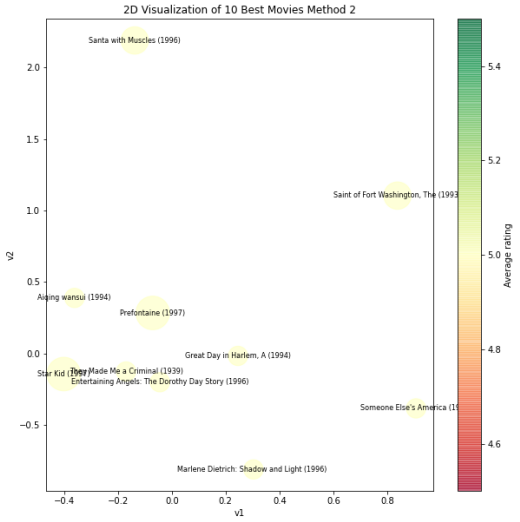


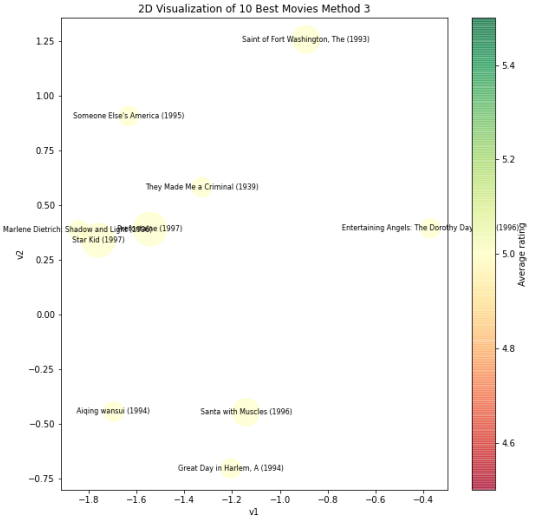




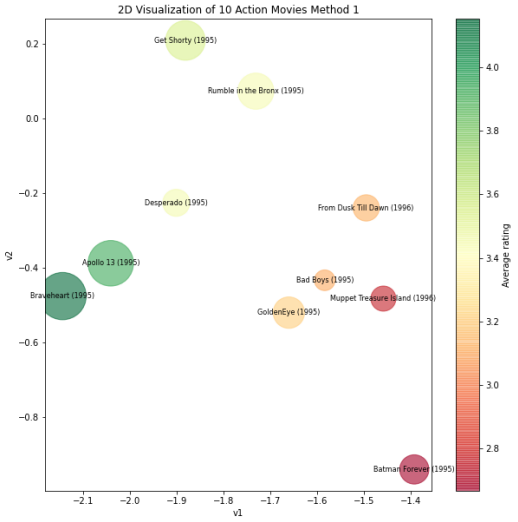
10 Best Movies

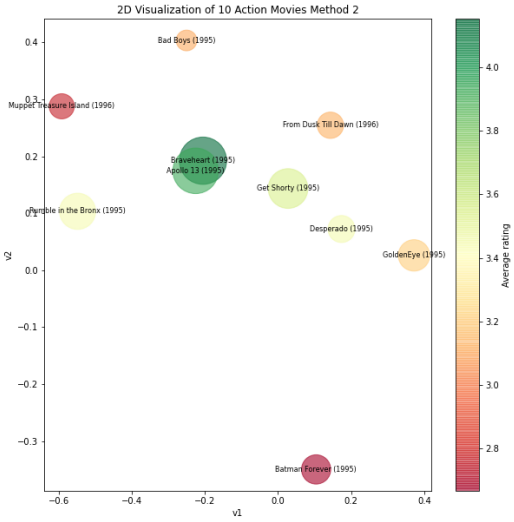


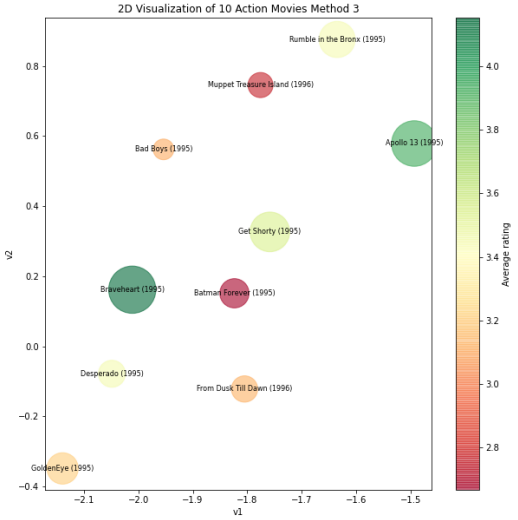




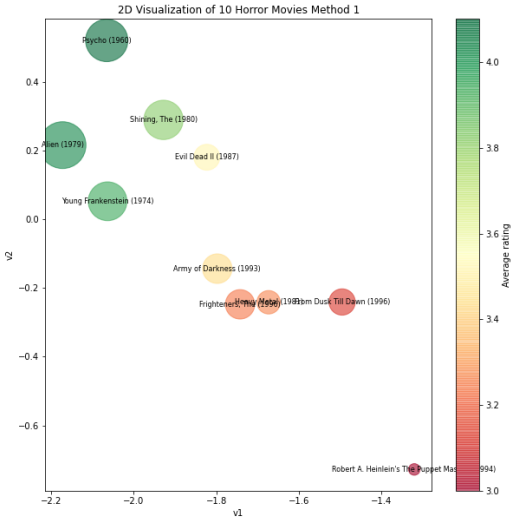
10 Movies from Three Genres Selected

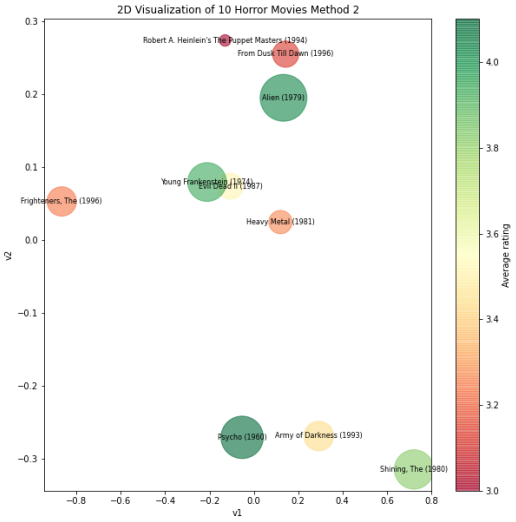


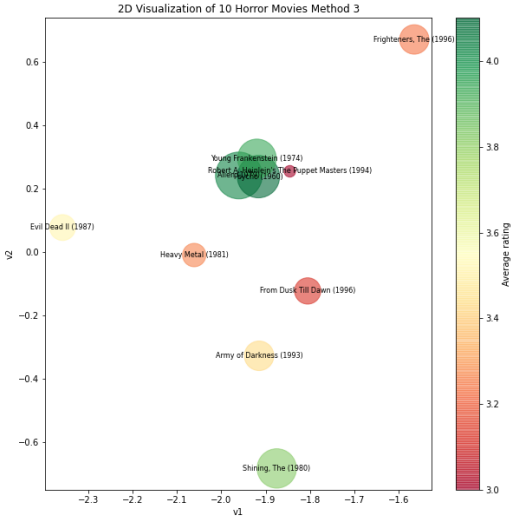




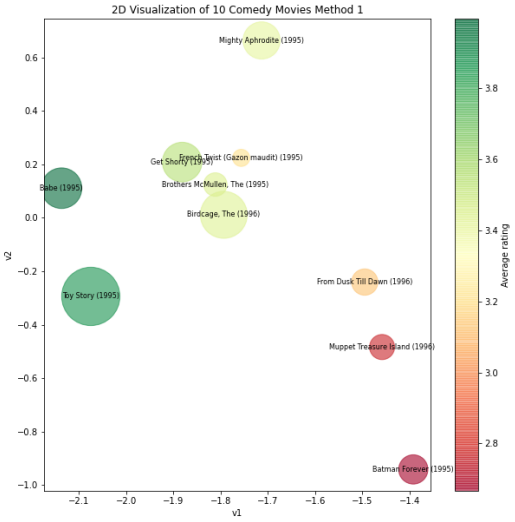
Action Above

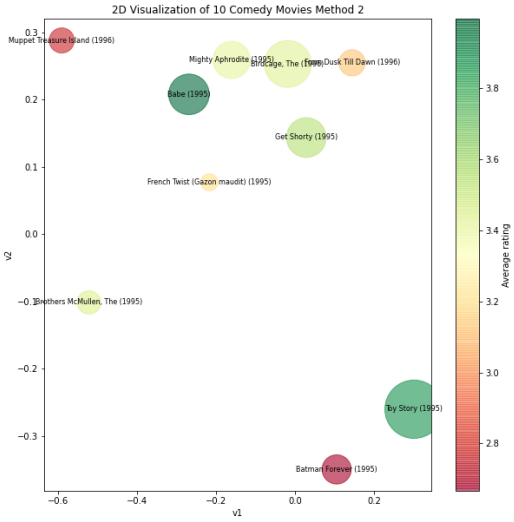


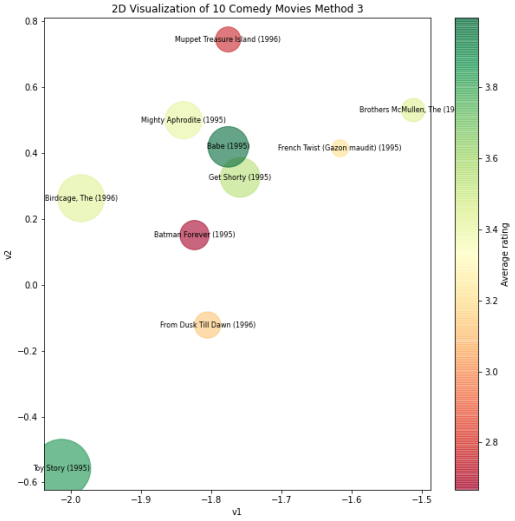




Horror Above







Comedy Above

It can be observed that movies that that are similar in nature are classified in similar regions in the visualization. For example, in our action movie classification, the movies Apollo 13 and Braveheart are visualized to be very similar, which makes sense since they are similar movies. This matches what we expect to see, since these movies are similar and are closely place in the visualization. There is not any clear pattern between the visualization of the best movies and the visualization of the popular movies. However, for the best movies, there is a smaller pool of ratings, while for the most popular movies there is a variety of ratings. Amongst the genres chosen, generally the movies with the highest average rating are grouped together. This leads us to believe that in any specific category, there are usually only a few types or general patterns of movies that lead to higher ratings. This is especially shown in the horror category: