## Choosing Attributes and Processing Original CSV file

In this project, I decided to use 5 attributes: popularity, vote average, genre, profit and spoken languages. Normalization is applied to them to ensure they don't overwhelm other attributes. Categorical data, such as genre, is converted to multiple dimensions via 1-hot encoding. Profit is not an original attribute. It is derived from the difference between budget and revenue; assuming budget is equal to cost. Textual data was omitted because the frequency of word is not important, but rather, the content of it is. However, text processing technique, such as sentiment analysis, is beyond the scope of this project. Other attributes were omitted either because of their relevance in the project's goal of movie classification or because the 'meaning' of their data overlapped with that of others attributes.

In this project, I randomly selected 2000 rows of datapoint from the original movies.csv to use. The reason for this is because of the computation power limit on my computer. I am confident it will work with the original csv input file if provided more time and computation power. The code for randomly selecting datapoint is in generate_shorter_movie_list.py (more instructions in README.md)

## K-means & K-means++ Convergence Test

I iterated different values of K in both K-means and K-means++, and calculated their distance (from the closest centroid). Each iteration of K for both tests are repeated 5 times to average out the randomness from K-means' centroid initialization step.

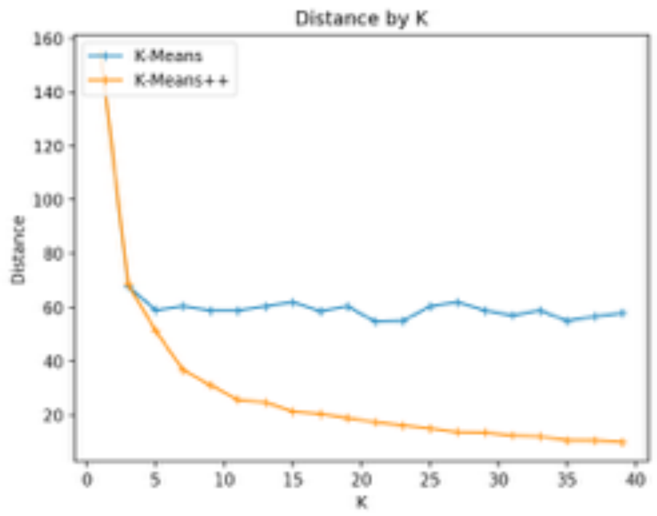

Figure 1. Distance of K-means and K-means++ by K (each iteration repeated 5 times)

As shown in the graph above, it is obvious k-means++ performs better than k-means as K increases. K-means++ is better because the initial centroids are distributed over the datapoint as fairly as possible using probability. K-means does this randomly. Based on the graph above, I chose the **optimal K for k-means to be 3** because the reduction in distance levels off right before K=3. I chose the **optimal K for k-means++ to be 12** because the decrease in distance is not as significant after K=12. In both cases, I didn't choose the K value that brings the lowest distance because doing so will result in overfitting (and we know distance = 0 when K= # of datapoint, and that is not what we want).

## Principal Component Analysis (PCA)

PCA is very useful in this project because there are too many attributes for us to take account of. It would simply take too long to run clustering algorithms with all dimensions. The advantage of PCA is it reduces the dimensions of our dataset, and this in turn reduces computation time.

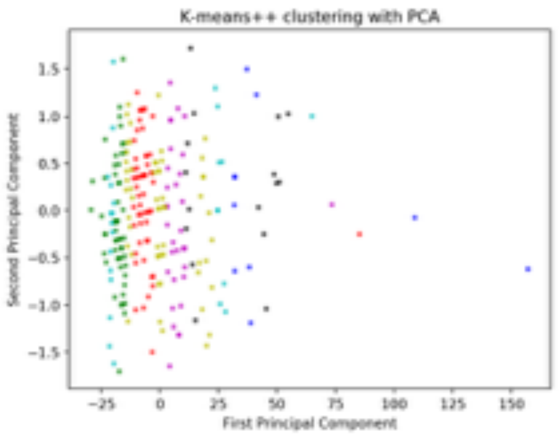

Figure 2. Plot of datapoint after applying PCA. Clustering is based on K-means++ ran before applying PCA

As shown from the graphs above, all datapoints that belong in the same cluster (before we performed PCA) are in some sort of grouping after we perform PCA. This is a good indication that PCA worked because we have not broken up the clusters.

## 1 Dimensional K-means clustering

Implementation of 1-dimensional k-means is based on the paper, “Optimal k-means clustering in One Dimension by Dynamic Programming”. In this algorithm, we start by assuming we have 1 clustering, and find the possible distances from first to last datapoint using sliding window technique. Then, we gradually increment the number of clustering to a pre-set upper limit (where the max is the number of datapoint). We can implement this via dynamic programming because the sliding window step is repeated multiple times. This algorithm works because, for a given K (other than K=1), we can calculate the best place to put the next optimal centroid by utilizing the calculated distances from the previous K.

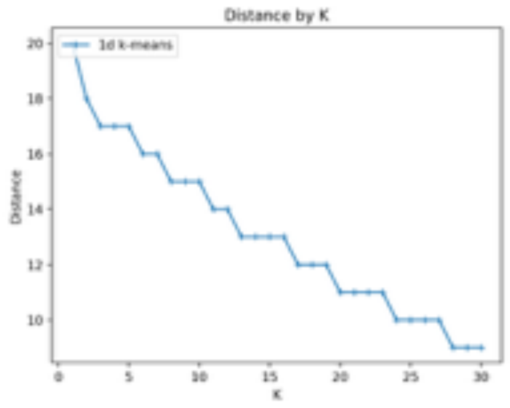

Figure 3. Distance of 1 dimensional K-means by K.

As shown in the graph above, the reduction in distance slows down as K increases. I chose the **optimal K to be 12 for 1-dimensional k-means** because the reduction after K=12 seems to level off gradually. If we choose more clustering, we might overfit.

## Disagreement Distance

*Computed Disagreement Distance = 59. datapoints=500, K_in_kmpp=12. K_in_1d=12.*

Figure 4. Result of Disagreement Distance between K-means++ and 1d K-means

## Instructions to execute the Project

Please refer to README.md