Description

In this assignment, we are going to implement the k-means algorithm over MongoDB using IMDB data. We will use the Euclidean distance in which points will be formed by the start year of the movies and their average rating. You should only consider movies (documents whose type='movie') with a number of votes greater than 10,000 for all questions in this assignment and where both startYear and avgRating exist.

⇒ CREATED VIEW 'ASSIGNMENT-8-FILTERS' BASED ON THE FOLLOWIN PIPELINE:

```
{
  type: "movie",
  numvotes: { $gt: 10000 },
  startyear: { $exists: true },
  avgrating: { $exists: true}
}
```

Your tasks

1. For each document, create a field called 'kmeansNorm'. This should contain an array in which the first position is the normalized start year and the second position is the normalized average rating. Let v be the value of a field and m and M the minimum and maximum values among all movies. The normalized value of v is computed as (v – m) / (M – m). (Normalization helps adjust for the different scales used in average rating and start year.) **(10 points)**

- Min, max values identified post-filtration for cleaner clustering.

- Documents from aforementioned View moved to a new collection "KMeans-Scores" with the value calculated in part 1.

2. Write a program which selects *k* random documents from genre *g* (*k* and *g* are inputs to your program). The two fields from the previous question should be inserted into new documents in a collection named centroids. Assign the centroid documents IDs from 1 to *k*. You will need to erase any previous documents in this collection if it already exists.

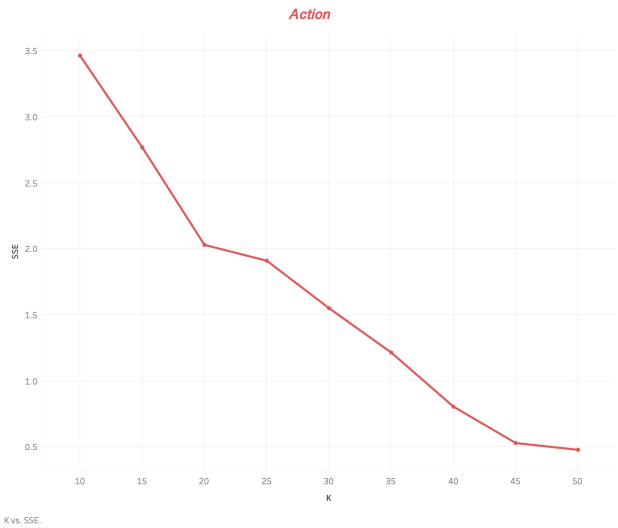
3. Implement one step of the *k*-means algorithm by assigning a new field 'cluster' in each document with the genre *g* (a parameter) _id of the closest centroid. Then update the documents in the centroids collection with the new centroids. **(20 points)**

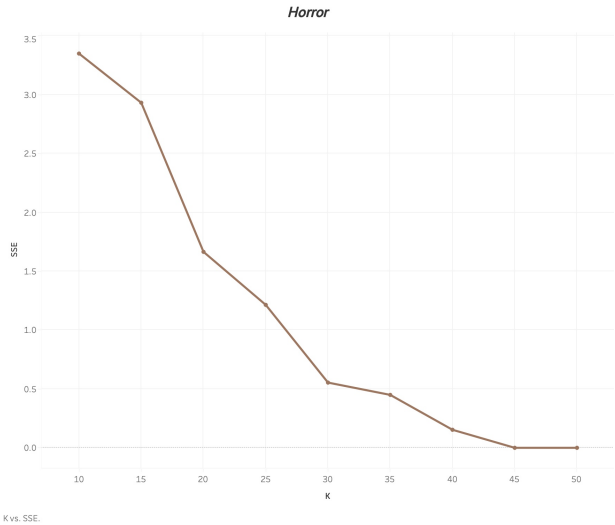
- I calculated a 2D array like:

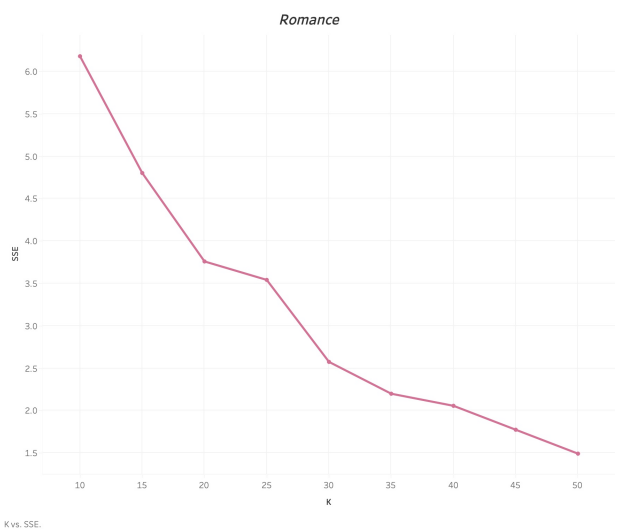
id	1	2	3	4 ...	n
1					
2					
3					
4					
n					

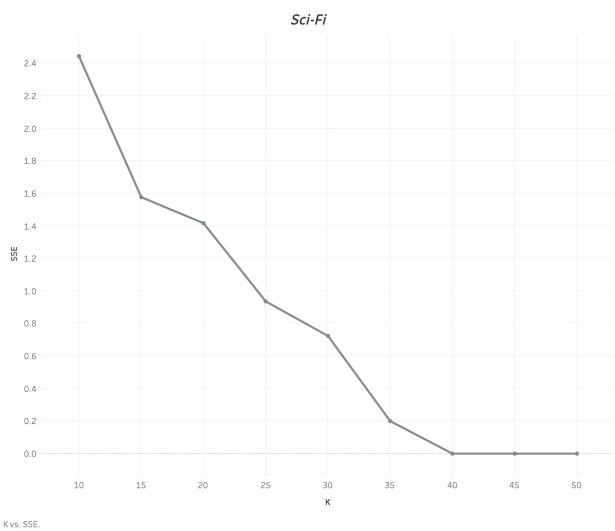
and populated the cells with the Euclidean distance between the two documents.

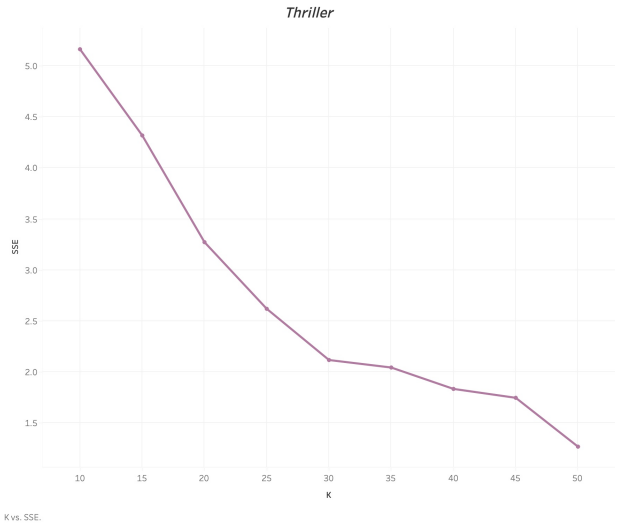
4. For each of the following genres: Action, Horror, Romance, and Sci-Fi, and Thriller, initialize the cluster centers and run k-means for up to 100 iterations (or until the clusters converge) starting with k=10 up to k=50 with a step of 5 using only movies of that genre. Plot the sum of squared errors vs the value of *k* for each of the genres (i.e. one plot per genre). Provide your code and the plots. **(30 points)**











5. For each of the previous genres, decide which one of the computed number of clusters is the best in each case. For each genre, present an example cluster of movies grouped together such that you can explain it.
   (25 points)

- Action
  - # Clusters = 20
  - Example:

_id	avgrating	cluster	genres	kMeansNorm	movie	numvotes	runtime	startyear
71402	**6.9**	**13**	["Action","Drama", "Crime"]	[0.6582278481012658, 0.42500000000000016]	71402	40335	93	**1974**
75223	**6.9**	**13**	["Crime","Comedy" ,"Action"]	[0.6835443037974683, 0.42500000000000016]	75223	20703	114	**1976**

The datapoints in this cluster are very close. If k is increased to 25, SSE will fall even further but the difference would not really be meaningful.

- Horror
  - #Clusters = 20
  - Example:

_id	avgrating	cluster	genres	kMeansNorm	movie	numvotes	runtime	startyear
36027	**7**	**11**	["Drama","Fantasy ","Horror"]	[0.26582278481012656 ,0.45000000000000007]	36027	12801	69	**1943**
37988	**7.5**	**11**	["Drama","Fantasy ","Horror"]	[0.2911392405063291, 0.5750000000000001]	37988	13602	110	**1945**

The total datapoints in the genre are 43 which is why the SEE reaches "0" eventually. A high K would not produce good clustering because at that point every point is its own cluster. k=20 is reasonable as it lets similar points be collected. A second elbow is at k=30 but that value is too

close to the total number of datapoints.

- Romance
  - #Clusters = 20
  - Example:

_id	avgrating	cluster	genres	kMeansNorm	movie	numvotes	runtime	startyear
61385	7	11	["Romance","Comedy"]	[0.569620253164557,0.45000000000000007]	61385	19348	106	1967
65134	7	11	["Adventure","Romance","War"]	[0.6075949367088608,0.45000000000000007]	65134	28646	116	1970
66011	6.9	11	["Drama","Romance"]	367088608,0.42500000(	66011	35166	100	1970
70849	6.9	11	["Romance","Drama"]	[0.6329113924050633,0.42500000000000016]	70849	55007	129	1972

The datapoints in this cluster are very close. After k=20, the curve flattens a little which indicates decreasing gain in accuracy for increase in k.

- Sci-Fi
  - #Clusters = 25
  - Example:

_id	avgrating	cluster	genres	kMeansNorm	movie	numvotes	runtime	startyear
67756	6.6	5	["Drama","Sci-Fi"]	[0.6329113924050633,0.3499999999999999]	67756	29928	89	1972
68713	6.5	5	["Sci-Fi","Adventure","Horror"]	[0.6329113924050633,0.325]	68713	11505	91	1972

The total datapoints in the genre are 39 which is why the SEE reaches "0" eventually. A high K would not produce good clustering because at that point every point is its own cluster. k=25 is reasonable as it lets similar points be collected. A second elbow is at k=35 but that value is too close to the total number of datapoints.

- Thriller
  - #Clusters = 30
  - Example:

_id	avgrating	cluster	genres	kMeansNorm	movie	numvotes	runtime	startyear
65143	7.1	5	["Horror","Thriller","Mystery"]	[0.6075949367088608,0.475]	65143	22109	96	1970
67927	7.2	5	["Action","Thriller","Crime"]	[0.6202531645569620,0.5000000000000001]	67927	29153	99	1971
70379	7.2	5	["Thriller","Crime","Drama"]	6202531646,0.50000000	70379	111478	112	1973

The datapoints in this cluster are very close. After k=30, the curve flattens a little which indicates decreasing gain in accuracy for increase in k.