Project Deliverable 3: Final Deliverable

Group 4

Members - Dhruv Jani, Dhruvil Patel, Karan Issrani, Mohit Gupta, Yaw Frempong

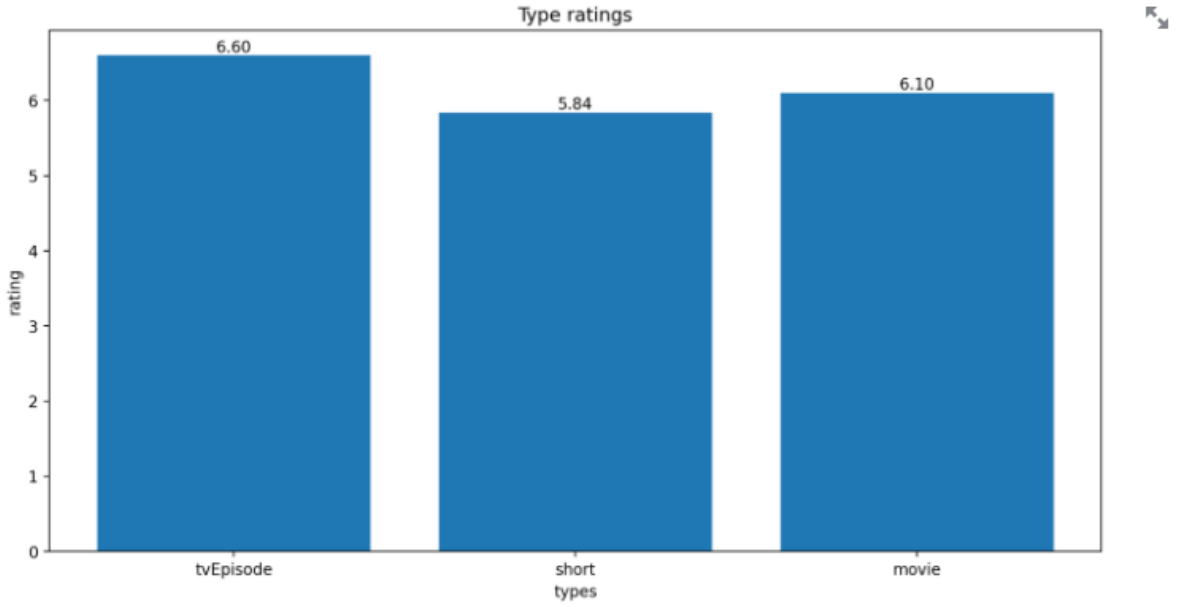
Analytics, Machine Learning

Here, We selected the KMeans Clustering algorithm. k-means clustering is a vector quantization method that seeks to partition n observations into k clusters, with each observation belonging to the cluster with the nearest mean (cluster centers or cluster centroid), which serves as the cluster's prototype. As a result, the data space is partitioned into Voronoi cells. Within-cluster variances (squared Euclidean distances) are minimized by k-means clustering, but regular Euclidean distances are not, which is the more difficult Weber problem: the mean optimizes squared errors, while only the geometric median reduces Euclidean distances. Using k-medians and k-medoids, for example, better Euclidean solutions can be obtained.

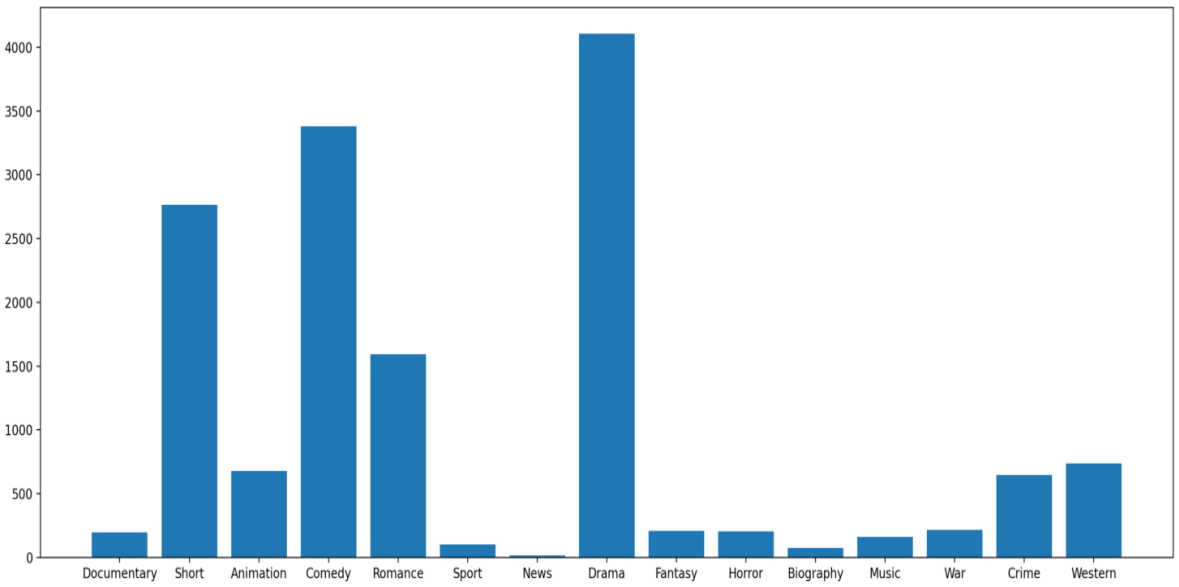
We made use of the K-means clustering algorithm from sklearn python library.

Analytics:

We also gave our code a front end to make the model interactive. We made use of Streamlit for this UI. Streamlit is a python library making it easy to make our Machine Learning models interactive. Here, we plotted a few analytical graphs and also took the users input to make real-time recommendations.



Average Ratings vs types of formats



Number of Movies in the dataset with respect to genres

Evaluation and Optimization

Evaluation:-

1) We broke the code into three portions to make recommendations. 1) weighted scores 2) clustering 3) the last function To begin, we'll utilize a weighted rating system to assign a rating to each piece of data based on the average rating and number of votes. The following is the formula for calculating weighted rating:

We use the Weighted rating formula for our analysis

$$W=(v/(v+m) * M) + (m/(m+v) * C)$$

Where:
M = Mean for the video format
v = number of votes for the video format
m = minimum votes required to be counted
C = the mean vote across the whole dataset

Optimization: -

We need a boolean column for all the Genres for that we'll make a new column for each genre and assign a value of 1 or 0 to it based on the value. There will be a total of 38 columns created. We'll use a scaler to standardize the data in the all genres column. It converts the data so that the mean is 0 and the standard deviation is 1. In a nutshell, it normalizes the data. For data with negative values, standardization is beneficial. It uses a typical normal distribution to organize the data.

```
def weighted_ratings(model_data):
    v = model_data['numVotes']
    R = model_data['averageRating']
    return (v/(v+m) * R) + (m/(m+v) * C)

model_data['wr'] = model_data.apply(weighted_ratings, axis=1)
model_data = model_data.sort_values(by = ['wr'], ascending = False)

word_cloud_list = list(set(word_cloud_list))
for i in word_cloud_list:
    model_data[i] = 0

for i in model_data.index:
    x = model_data['genres'][i].split(',')
    for k in x:
        model_data[k][i] = 1

cluster_data = model_data[['primaryTitle'] + word_cloud_list]

scaler = StandardScaler()
data_scaled = scaler.fit_transform(cluster_data.iloc[:,1:])
```

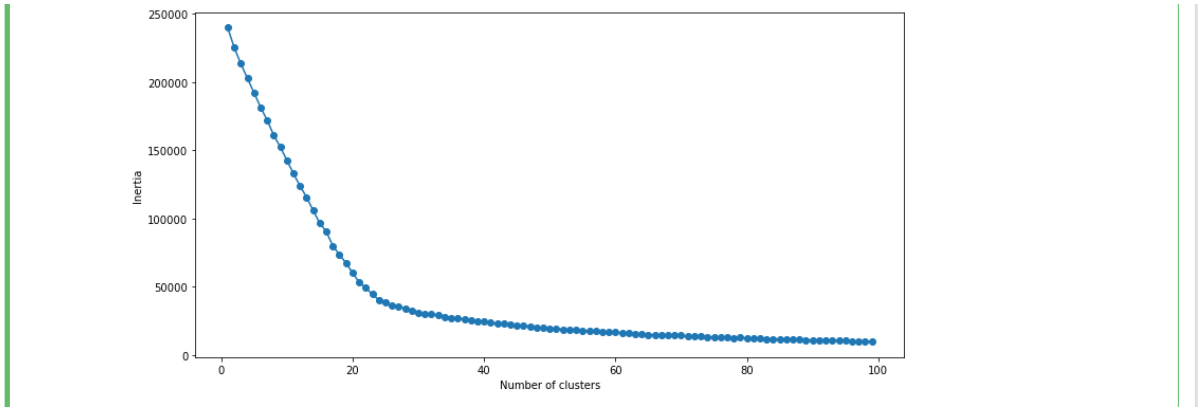
Results

1) We make the famous elbow graph for our data and try to see which value of k is feasible for us.

2) When we see the graph we see that we that after k=23 the values go down linearly. For further process, We will make use of K=23. Now we'll utilize the k-means cluster to forecast the clustering result. This will assist us in making genre-based recommendations.

3)We consider 5 parameters for our recommendation. At least one argument must be presented. A title is a movie/show name, a type is a video format, a rating is the minimum rating required, a year is a specific year search, and a genre is a genre name, and it will recommend up to 10 titles based on the arguments provided

.



In [6]:
```
def get_recommendations(title=None,types = None,ratings=None,year=None,genres=None):
    x = model_data
    if title != None:
        cluster_no = list(x[x['primaryTitle'].str.lower() == title.lower()]['cluster'])
        x = x[x['cluster'] == random.choice(cluster_no)]
    if types != None:
        x = x[x['titleType'].str.lower() == types.lower()]
    if year != None:
        x = x[x['startYear'] == year]
    if genres != None:
        list_of_genres = genres.split(',')
        x = x[x[genres.capitalize()] == 1]
    if ratings != None:
        x = x[x['wr'] >= ratings]

    return x[:15]
```

In [7]:
```
get_recommendations(types='short', genres='comedy', year=1918)
```

```
2022-05-05 08:16:25.485 INFO    numexpr.utils: NumExpr defaulting to 8 threads.
```

Out[7]:

	tconst	averageRating	numVotes	titleType	primaryTitle	originalTitle	isAdult	startYear	endYear	runtimeMinutes	...	Film-Noir	Mystery	Musical	Sci-Fi	S
2430	tt0009018	7.7	8496	short	A Dog's Life	A Dog's Life	0	1918	\N	33	...	0	0	0	0	
2565	tt0009611	7.3	6739	short	Shoulder Arms	Shoulder Arms	0	1918	\N	36	...	0	0	0	0	
2391	tt0008874	6.7	1251	short	The Bell Boy	The Bell Boy	0	1918	\N	33	...	0	0	0	0	
2420	tt0008975	6.6	1594	short	The Cook	The Cook	0	1918	\N	22	...	0	0	0	0	
2525	tt0009466	6.4	1045	short	Out West	Out West	0	1918	\N	25	...	0	0	0	0	
2580	tt0009678	6.5	518	short	Take a Chance	Take a Chance	0	1918	\N	10	...	0	0	0	0	
2055	tt0007151	7.8	57	short	Out of the Inkwell	Out of the Inkwell	0	1918	\N	2	...	0	0	0	0	
2497	tt0009316	6.5	187	short	Look Pleasant, Please	Look Pleasant, Please	0	1918	\N	10	...	0	0	0	0	
2447	tt0009099	6.4	156	short	A Gasoline Wedding	A Gasoline Wedding	0	1918	\N	9	...	0	0	0	0	

Future Work, Comments

The data we selected was not from Kaggle, but it was the official data from IMDb. The dataset was huge. The free instance of Sagemaker won't process so much data. So we reduced the size of our data. The datasets were tab spaced versions and had a const id column, making it easy to merge 2 datasets and make a dataset we require.

The dataset we merged had a single column of genres. As we wanted separate values. We added more columns to the dataset and each column represented a genre. The input in these columns was boolean to make it easier for us to process the data. We also removed some null values present in the dataset.

As the dataset was otherwise clean, We didn't have to deal with outliers of any kind.

The imbd has some movies whose genre is not known. We cant categorize this data and we need further information on these movies to know the genre. We leave them as N for now.

Some important points.

1. Did you create any new additional features/variables?

Yes, We created a few features for our genres.

2. What was the process you used for evaluation? What was the best result?

We used the K-means clustering algorithm till convergence and then we used the clusters for further recommendation analysis.

3. What were the problems you faced? How did you solve them?

If this is a funded project then you can freely use the AWS resources. In college-level instances, you can't have a large dataset and you can’t build a pipeline. The data cant is kept in s3 because for every access you will call the S3 storage. This may incur charges on your account. The dataset we choose must have appropriate data. If not, the accuracy of the model declines. To develop a project like this you need to have knowledge of Machine Learning algorithms and AWS. We used a private account for our process and made which caused fewer problems. We selected the datasets which had appropriate data and were from a legitimate source.

4. What future work would you like to do?

In the future, We can do regression analysis on the data. The dataset had a lot of movies without a genre. We will try to label the data there. We can also shift to graphical databases for processing unstructured data. Further enhancements are possible to give it a professional-looking UI.

5. Instructions for individuals that may want to use your work
    1. Make sure you install all the necessary python libraries before working on the code.
    2. Find a good dataset with enough rows and relevant columns.
    3. While creating an S3 Bucket and a new notebook instance make sure to use a new IAM role and store data in a random s3 bucket.
    4. Use a good processing instance for higher processing power.
    5. There are 2 Jupyter notebooks. You can use our code on your local machine as well as a notebook for AWS instance.
    6. Read about the boto library for using AWS.
    7. We created a separate python file for Streamlit.