**Topic : NETFLIX MOVIES AND TV SHOWS CLUSTERING**

**For capstone project using unsupervised machine learning**

**Kundan Lal, Abhijeet Kulkarni, Pankaj Ganjare,**

**Mohd Sharik**

**Data Science Trainees**

**Alma Better**

**Abstract:**

1. Our study encompasses the findings done Predicting whether Understanding what type content is available in different countries
2. Is Netflix has increasingly focusing on TV rather than movies in recent years.
3. Clustering similar content by matching text-based features

Netflix is a company that manages a large collection of TV shows and movies, streaming it anytime via online. This business is profitable because users make a monthly payment to access the platform. However, customers can cancel their subscriptions at any time. Therefore, the company must keep the users hooked on the platform and not lose their interest. This is where recommendation systems start to play an important role, providing valuable suggestions to users is essential.

**Problem Statement**

This dataset consists of tv shows and movies available on Netflix as of 2019. The dataset is collected from Flixable which is a third-party Netflix search engine.

In 2018, they released an interesting report which shows that the number of TV shows on Netflix has nearly tripled since 2010. The streaming service’s number of movies has decreased by more than 2,000 titles since 2010, while its number of TV shows has nearly tripled.

**In this project, you are required to do**

1. Exploratory Data Analysis
2. Understanding what type content is available in different countries
3. Is Netflix increasingly focused on TV rather than movies in recent years?
4. Clustering similar content by matching text-based features.

# Data Description

**Attribute Information**

1. show\_id : Unique ID for every Movie / Tv Show
2. type : Identifier - A Movie or TV Show
3. title : Title of the Movie / Tv Show
4. director : Director of the Movie
5. cast : Actors involved in the movie / show
6. country : Country where the movie / show was produced
7. date\_added : Date it was added on Netflix
8. release\_year : Actual Release year of the movie / show
9. rating : TV Rating of the movie / show
10. duration : Total Duration - in minutes or number of seasons
11. listed\_in : Genere
12. description: The Summary description

## **Approach taken:**

This project is divided into multiple section where each section will have it's own importance towards our problem statements. the approach that we will be following in this project is given as-

Section 1: In this section we will simply loading our dataset into google colab and will explore the basic information about data.

Section 2: In section 2 we will be dealing with missing values of this dataset and will impute the values into missing places.

Section 3: In this section We fill focus on Exploratory data analysis of the dataset using various methods and visualization plot and will be extracting the information from this dataset as much as we can.

Section 4: In this section we will be dealing with outliers in out dataset and will see how to define our outlier criteria and deal with outliers.

Section 5: In this Section we will be applying NLP in order to get most similar contents like most similar movies.

Section 6: In this section we will be perform various clustering methods to find out best no. of clusters and for validation we will be using Silhouette score and elbow curve where ever it is applicable.

Section 7: In this section we will be giving a quick summary of entire notebook.

# importing important libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')

from kmodes.kmodes import KModes

from sklearn.cluster import KMeans

from sklearn.feature\_extraction.text import TfidfVectorizer

from yellowbrick.cluster import SilhouetteVisualizer

from sklearn.preprocessing import StandardScaler,normalize

import umap.umap\_ as umap

# mounting drive

from google.colab import drive

drive.mount('/content/drive')

#loading the dataset

initial\_df =  pd.read\_csv('/content/drive/MyDrive/Almabetter/Capstone Project Unsupervised ML/NETFLIX MOVIES AND TV SHOWS CLUSTERING (1).csv')

# basic info about data

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 7787 entries, 0 to 7786

Data columns (total 12 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 show\_id 7787 non-null object

1 type 7787 non-null object

2 title 7787 non-null object

3 director 5398 non-null object

4 cast 7069 non-null object

5 country 7280 non-null object

6 date\_added 7777 non-null object

7 release\_year 7787 non-null int64

8 rating 7780 non-null object

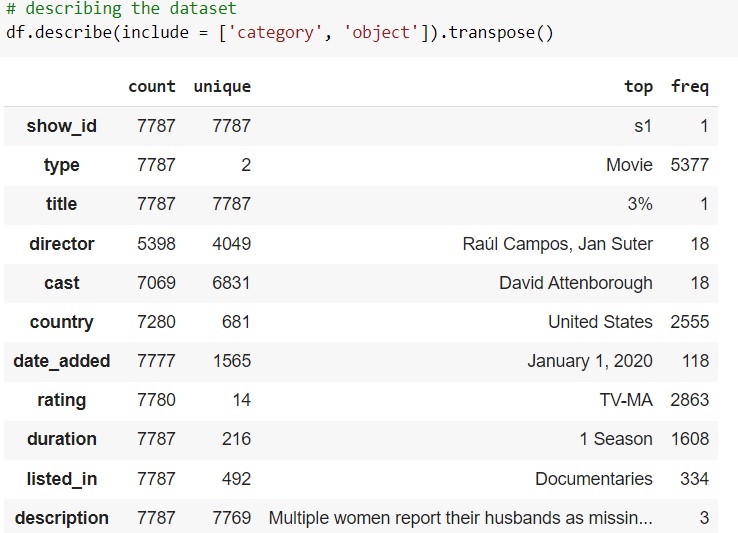
9 duration 7787 non-null object

10 listed\_in 7787 non-null object

11 description 7787 non-null object

dtypes: int64(1), object(11)

memory usage: 730.2+ KB



From above info and table we knows the basic information about data set

* We have seen that we are given 7787 rows and 12 columns and most columns are categorical.
* We are given two different content type as movies and tv shows hence we will be dealing with these columns separately in other sections.
* We also have observed that some columns are nested columns where values are given comma( , ) separable. Since these columns are cast and listed\_in(genres) and these two features play very important part in movies and tv shows hence we will be treating them carefully.
* There is no duplicated datapoint/row in our dataset.
* Since we have mostly categorical features hence we do not get too much descriptive statistical values apart from mode of the features.
* We also have checked unique values of all the features.
* We also have dropped show\_id columns as all the values of this columns are unique and will not be relevant for our investigation and objective
* # checking missing values
* df.isnull().sum()
* type 0
* title 0
* director 2389
* cast 718
* country 507
* date\_added 10
* release\_year 0
* rating 7
* duration 0
* listed\_in 0
* description 0
* dtype: int64

# filling the missing values in director, cast and country column

df['director'] =df.director.fillna('None')

df['cast'] =  df['cast'].fillna('not available')

df['country'] = df['country'].fillna('missing')

# dropping date\_added and rating rows where values are missing

df = df.dropna(axis =0, subset = ['date\_added', 'rating'] )

df['date\_added'].isnull().sum()

# checking the null values after removal

df.isnull().sum()

type 0

title 0

director 0

cast 0

country 0

date\_added 0

release\_year 0

rating 0

duration 0

listed\_in 0

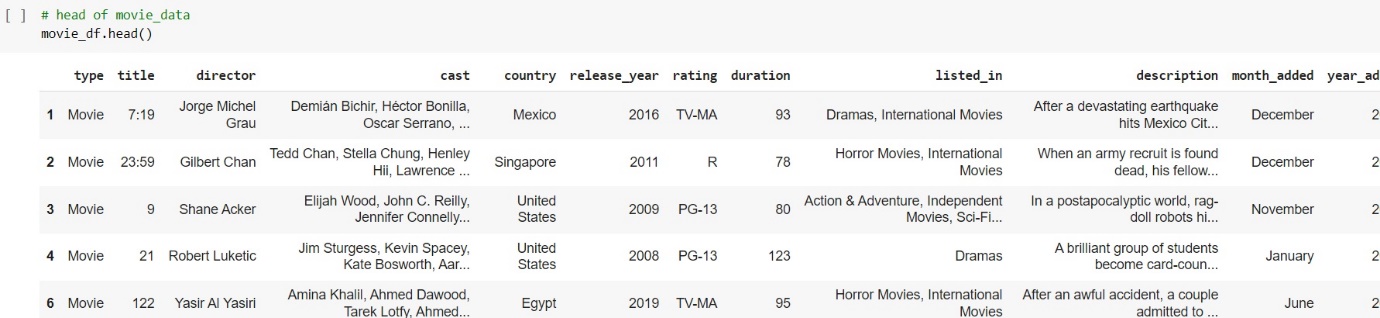
description 0

dtype: int6

* First we have summarized the missing values in entire dataset and found that the most missing values are available in director feature followed by cast and country. and the features which has minimum values are date\_added and rating.
* Then we have visualized the missing values across the data by plotting heatmap and seen that values are missing completely at random.
* since the columns which were containing missing values were quite important hence we didn't drop those missing values as it could have lead to information shortage and finally we have decided to fill these missing values by some dummy value except date\_added feature and rating as these columns are having less missing values 10, 7 respectively and hence we have dropped the rows containing these null values.

**Exploratory Data Analysis**:

Exploratory Data Analysis (EDA) as the name suggests, is used to analyze and investigate datasets and summarize their main characteristics, often employing data visualization methods. It helps determine how best to manipulate data sources to get the answers you need, making it easier for data scientists to discover patterns, spot anomalies, test a hypothesis, or check assumptions. It also helps to understand the relationship between the variables (if any) and it will be useful for feature engineering. It helps to understand data well before making any assumptions, to identify obvious errors, as well as better understand patterns within data, detect outliers, anomalous events, find interesting relations among the variables. After mounting our drive and fetching and reading the dataset given, we performed the Exploratory Data Analysis for it.



* As we all know that EDA is one of the most important part of any Data Science project and if not done properly it may lead to incorrect insight of historical data. Keeping this in mind we have performed a very detailed data analysis for this first we have divided our data set into two different datasets named movie\_df which contains all the movies given in dataset and it's associated information. The other dataset that we have created contains the information about TV shows named shows\_df. basis these two datasets we have performed EDA and found which type has which insights in terms of given features.
* In this path first we have changed nature of movie duration as it was given to in string format

The above table shows the dataset in the form of Pandas Data Frame

**Finding the % distribution of type content plotting the type distribution**

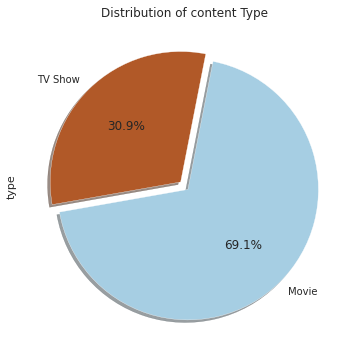
sns.set\_style('darkgrid')

plt.figure(figsize = (8,6))

df['type'].value\_counts().plot.pie(autopct="%1.1f%%", cmap = 'Paired', shadow=True, startangle=190,explode=(0.04,0.04));

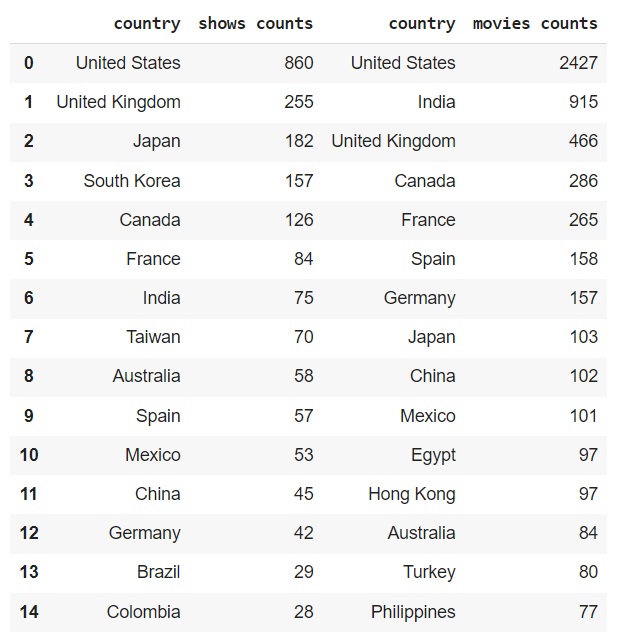
plt.title('Distribution of content Type')

plt.show()



we have seen what type of content is available in what amount and ratio and found that 69% of dataset is occupied by movies and 31% is occupied by TV shows.

**Finding top 15 countries based on content type**



**Finding seasons distribution for TV shows**

plt.figure(figsize = (10,7))

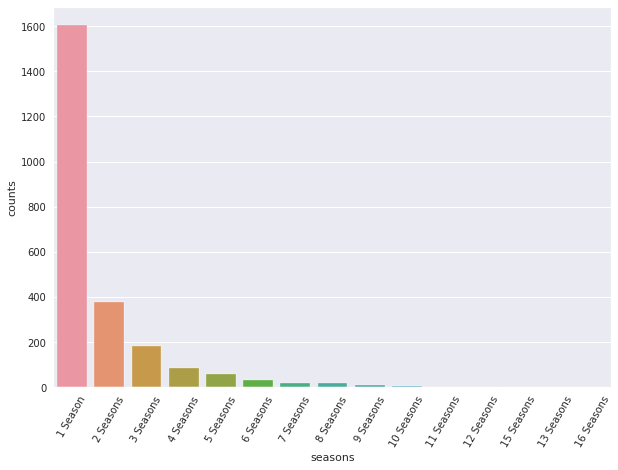
seasons\_df = pd.DataFrame(shows\_df['duration'].value\_counts()).reset\_index()

seasons\_df.rename(columns = {'index': 'seasons', 'duration': 'counts'}, inplace = True)

sns.barplot(data =seasons\_df, x= 'seasons', y= 'counts')

plt.xticks(rotation = 60)

plt.show()



 we have seen the seasons distriution of TV shows and found that there are only few TV shows which crossed 8 season most of seasons have released in one 1 season.Also when plotted movies duration distribution then we have found that this distribution is positively skewed and there are very less moves which crossed 175 mins. Thus we information that we need to remove certain outliers from out dataset

**Findings the distribution of various ratings**

order =  ['G', 'TV-Y', 'TV-G', 'PG', 'TV-Y7', 'TV-Y7-FV', 'TV-PG', 'PG-13', 'TV-14', 'R', 'NC-17', 'TV-MA']

plt.figure(figsize=(15,7))

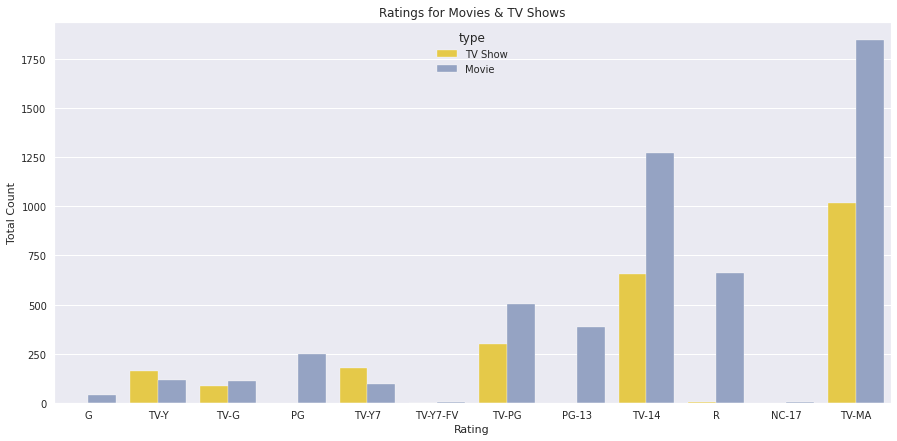
g = sns.countplot(df.rating, hue=df.type, order=order, palette='Set2\_r');

plt.title("Ratings for Movies & TV Shows")

plt.xlabel("Rating")

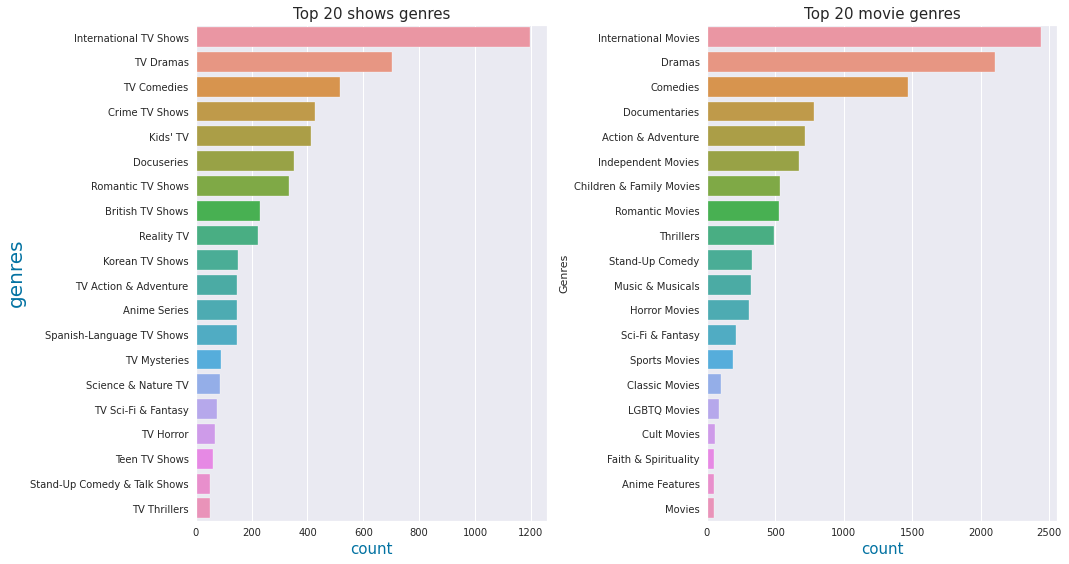
plt.ylabel("Total Count")

plt.show()



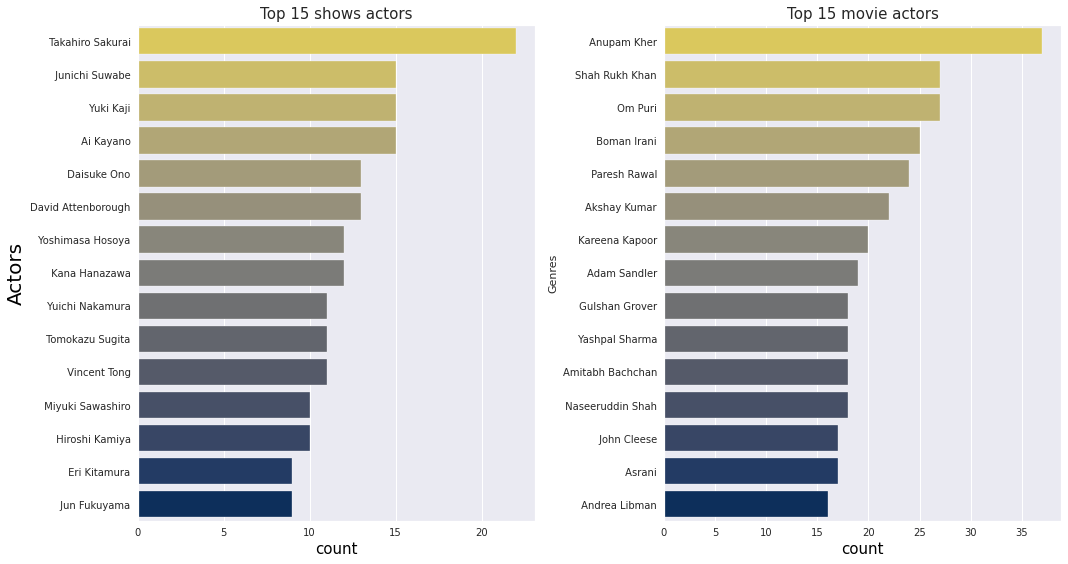
 ratings distribution and found that most contents are adult rated on Netflix and there are very less G-rated contents.

**Finding top 20 movie genres in movies and TV shows**



 Analyzed genres for movies and TV shows and figured out that in both type internation genres is on top in the list followed by drama and comedies respectively.

**Finding the top actors who worked in most content in movies and TV shows**

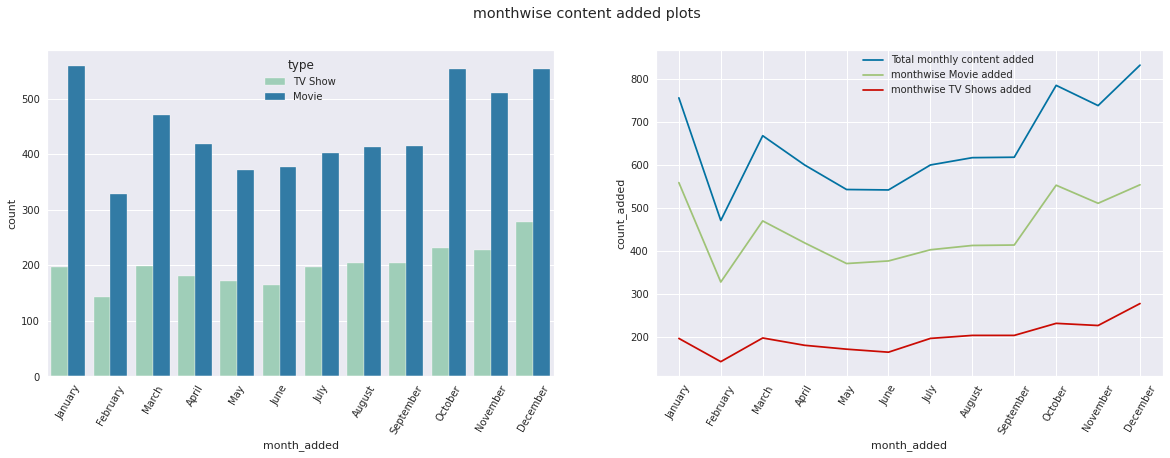
****

Actors who were part of most shows and movies which gave us that Takahiro sukurai has worked in most shows and Anupam kher has worked in most movies.

**Finding month wise content distribution**

**Month wise content distribution in movies dataset**

**Month wise content distribution in TV shows dataset**

****

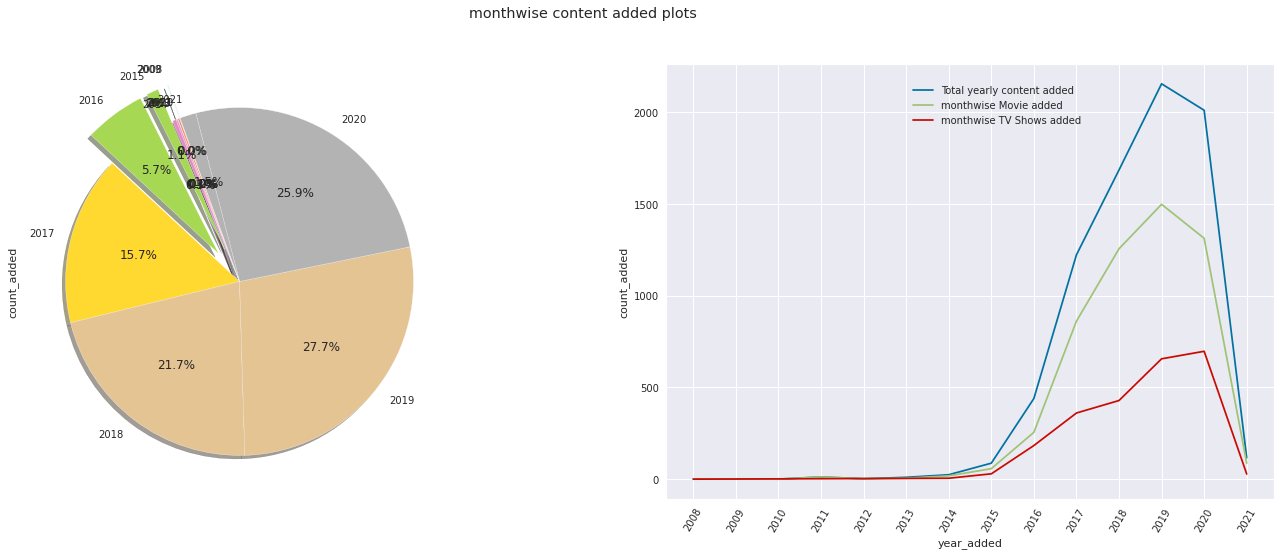
we have visualized that how the content is added on the platform in different month. With the help of month wise analysis we have found that most content is added on the platform in the month of Nov, Dec, and Jan.

**Findings**

**year wise content distribution**

**year wise content distribution in movies dataset**

**year wise content distribution in TV shows dataset**

****

* we have seen what type of content and how much content is added on the platform on the basis of year. with this distribution we have found that in recent year content is added on the platform exponentially. also we have found that between 2019 to 2020 Netfilx has added more shows in comparison of movies. And Since the dataset is not given for complete 2021 hence we are not telling about it.

**Outlier detection**

depending on the nature of columns and values of these columns most of the columns content a lot unique values hence we cannot detect outliers however we check for the outliers in country and duration column if exist.

### Outliers in duration

since we have two content types (TV shows and movies ) and the values are different in both these types hence we will perform these tasks separately in two individual dataframes. i.e shows\_df & movie\_df

# detecting outlier detection in shows\_df as in this dataframe duration columns contains categorical values i.e seasons

# cheking for the percent distribution in duraion column of shows\_df

shows\_df['duration'].value\_counts(normalize =True)\*100

shows\_duration = pd.DataFrame(shows\_df['duration'].value\_counts()).reset\_index().rename(columns = {'index' : 'season', 'duration': 'counts'})

shows\_duration['percent\_occupied'] = shows\_duration['counts']/len(shows\_df)\*100

shows\_duration

# based on this result we can assume that there are less than 1 percent shows which are 7 seasons or long hence we can consider these seasons as outliers

# however we will be removing content which has more than 10 seasons

* In this way first we have decided the columns where we can perform the outlier detection as most of the features are categorical hence our base of removing outliers will value counts for different categories. Since most of the columns contains large no. of less counted values hence we will have to decide which can be gone through the process of outlier detection. In this scenario we have decided to perform this on country column and duration column of movies and TV shows. but here we found many points which contains a chain of countries hence the value count is one. by removing these columns we might loose important information and finally we have dropped the idea of removing outlier from this column.

**Natural Language Processing (NLP) Model:**

For the NLP portion of this project, I will first convert all plot descriptions to word vectors so they can be processed by the NLP model. Then, the similarity between all word vectors will be calculated using cosine similarity (measures the angle between two vectors, resulting in a score

between -1 and 1, corresponding to complete opposites or perfectly similar vectors). Finally, I will extract the 5 movies or TV shows with the most similar plot description to a given movie or TV show.

**Tf-idf vectorization:**

TF-IDF is an abbreviation for Term Frequency Inverse Document Frequency. This is a very common algorithm to transform text into a meaningful representation of numbers which is used to fit a machine learning algorithm for prediction.

We have also utilized the PCA because it can help us improve performance at a very low cost of model accuracy. Other benefits of PCA include reduction of noise in the data, feature selection (to a certain extent), and the ability to produce independent, uncorrelated features of the data.

So, it's essential to transform our text into tf-idf vectorizer, then convert it into an array so that we can fit into our model.

* **Finding number of clusters**

The goal is to separate groups with similar characteristics and assign them to clusters.

We used the Elbow method and the Silhouette score to do so, and we have determined that 28 clusters should be an optimal number of clusters.

**K-modes Clustering Algorithm:**

K- modes algorithm is an algorithm that works similar to the K-means. however, there is a major difference between both of that K-means clustering algorithm is limited to numerical data and it do not work on categorical as we cannot measure distance between two categorical values.

It uses the dissimilarities (total mismatches) between the data points. The lesser the dissimilarities the more similar our data points are. It uses Modes instead of means. Steps that are followed in K-modes are as follows.

1. Select random observations and treat them as leader/cluster centers.
2. Calculate the dissimilarities and assign each observation to closest cluster.
3. Define new modes for each cluster.
4. Repeat step no. 2-3 until there is no re-assignment required.

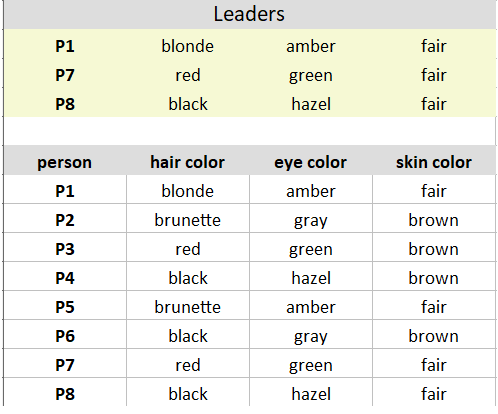
Let’s understand with an example:

Imagine we are given a dataset as follows-



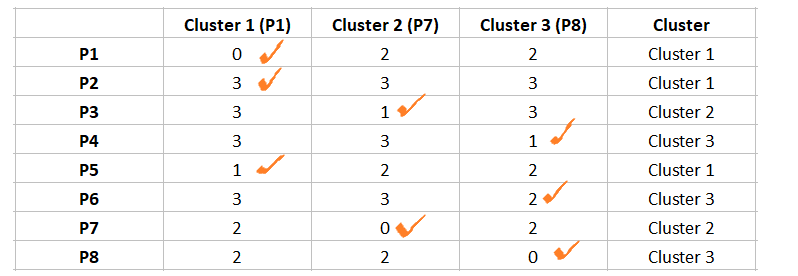
1. Select random observations and treat them as leader/cluster centers.

I am choosing P1, P7, P8 as leaders/clusters



1. Calculate the dissimilarities and assign each observation to closest cluster.

Iteratively compare the cluster data points to each of the observations. Similar data points are given 0, dissimilar data points are given 1. Thus we create a matrix as below-

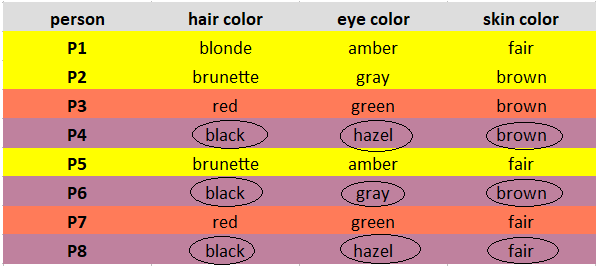


After step 2, the observations P1, P2, P5 are assigned to cluster 1; P3, P7 are assigned to Cluster 2; and P4, P6, P8 are assigned to cluster 3.

1. Define new modes for each cluster.

Mode is simply the **most observed value**.

Mark the observations according to the cluster they belong to. Observations of Cluster 1 are marked in Yellow, Cluster 2 are marked in Brick red, and Cluster 3 are marked in Purple.

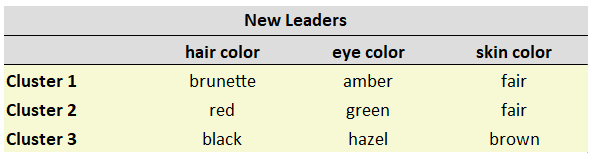


Considering one cluster at a time, for each feature, look for the Mode and update the new leaders.

**Explanation:**Cluster 1 observations (P1, P2, P5) has brunette as the most observed hair colour, amber as the most observed eye colour, and fair as the most observed skin colour.

Note: If you observe the same occurrence of values, take the mode randomly. In our case, the observations of Cluster 3(P3, P7) have one occurrence of brown, fair skin colour. I randomly chose brown as the mode.

Below are our new leaders after the update.

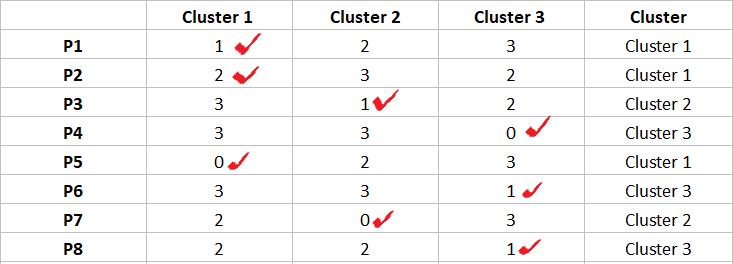


**Repeat steps 2–4**

After obtaining the new leaders, again calculate the dissimilarities between the observations and the newly obtained leaders.

Comparing Cluster 1 to the observation P2 gives 2 dissimilarities.

Likewise, calculate all the dissimilarities and put them in a matrix. Assign each observation to its closest cluster.



The observations P1, P2, P5 are assigned to Cluster 1; P3, P7 are assigned to Cluster 2; and P4, P6, P8 are assigned to Cluster 3.

We stop here as we see there is no change in the assignment of observations.

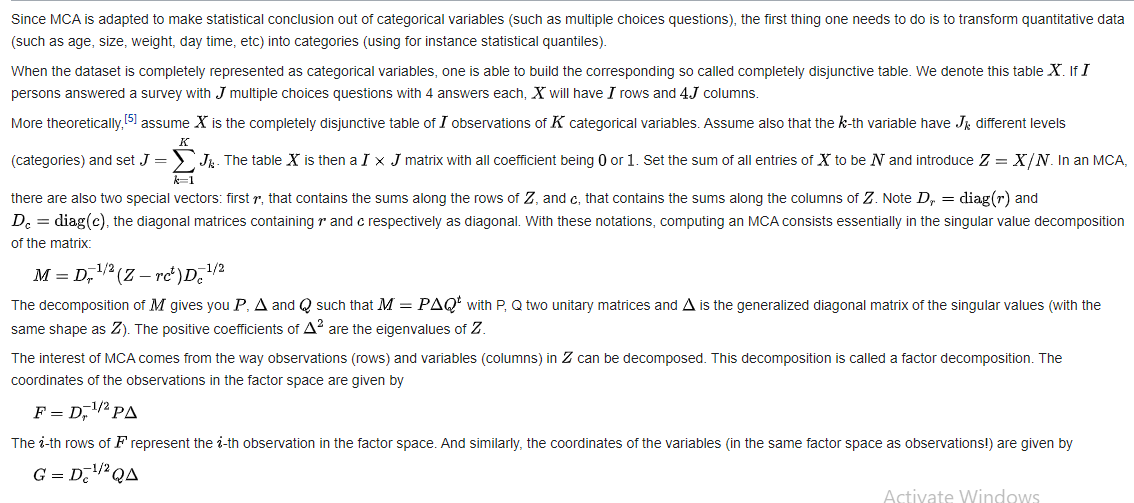
**Converting categorical algorithm.**

Unfortunately, the majority of clustering algorithms can only work with data that exclusively contains either numeric or categorical features. This is a huge problem. as most real world datasets will contain multiple types of features. Two of the methods that we have used in the project are given below.

1. **MCA ( Multiple correspondence analysis )**

multiple correspondence analysis (MCA) is a data analysis technique for nominal categorical data, used to detect and represent underlying structures in a data set. It does this by representing data as points in a low-dimensional Euclidean space. The procedure thus appears to be the counterpart of principal component analysis for categorical data MCA can be viewed as an extension of simple correspondence analysis (CA) in that it is applicable to a large set of categorical variables.

Since MCA is adapted to make statistical conclusion out of categorical variables (such as multiple choices questions), the first thing one needs to do is to transform quantitative data (such as age, size, weight, day time, etc) into categories (using for instance statistical quantiles).

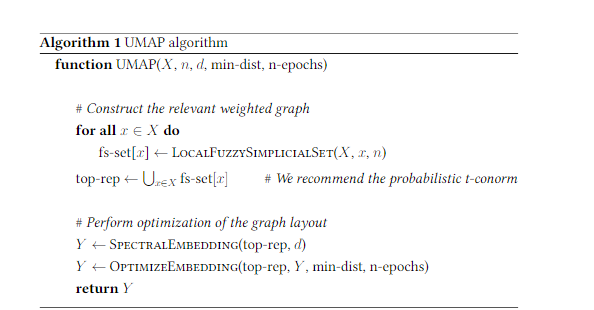


**UMAP** (**Uniform Manifold Approximation and Projection for Dimension Reduction)**

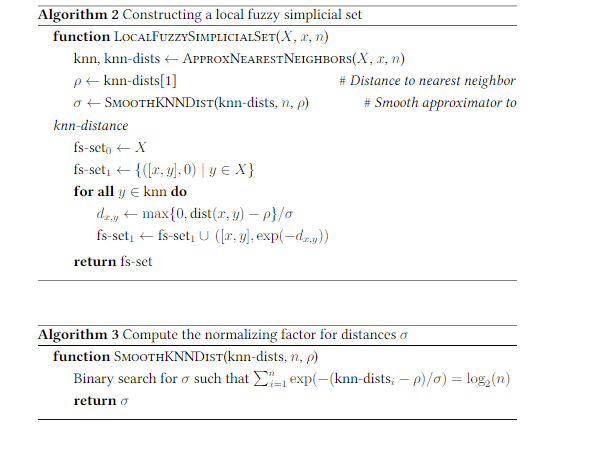
UMAP is a dimensionality reduction technique predicated upon manifold learning & ideas from topological data analysis. It provides a general framework for applying dimensionality reduction and is an incredibly powerful tool in any data scientist’s arsenal. In order to apply UMAP to mixed data we must create two separate manifolds one for each data type, as UMAP still needs to compute distances between points. These two manifolds are then combined either by computing the Union which preserves the categorical embedding more, or by calculating the intersection which will cause the embedding to more closely resemble the numerical manifold.

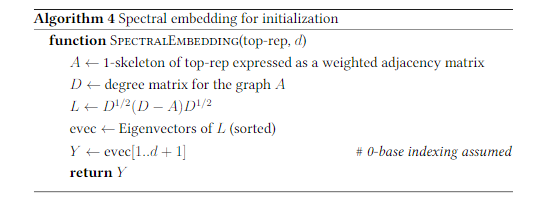
**Algorithm description**

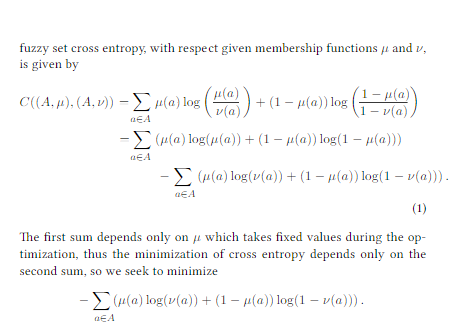
In overview the UMAP algorithm is relatively straightforward (see Algo-rithm 1). When performing a fuzzy union over local fuzzy simplicial sets we have found it most effective to work with the probabilistic t-conorm (as one would expect if treating membership strengths as a probability that the simplex exists). e individual functions for constructing the local fuzzy simplicial sets, determining the spectral embedding, and optimizing the embedding with regard to fuzzy set cross entropy, are described in more detail below

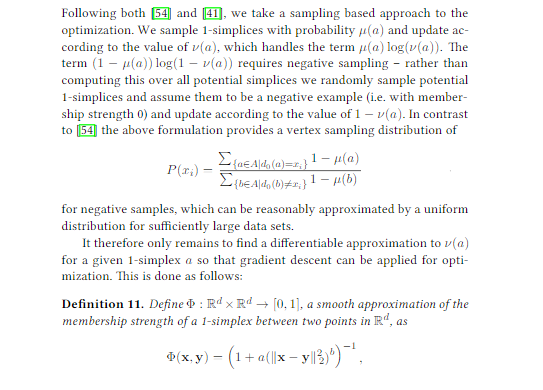


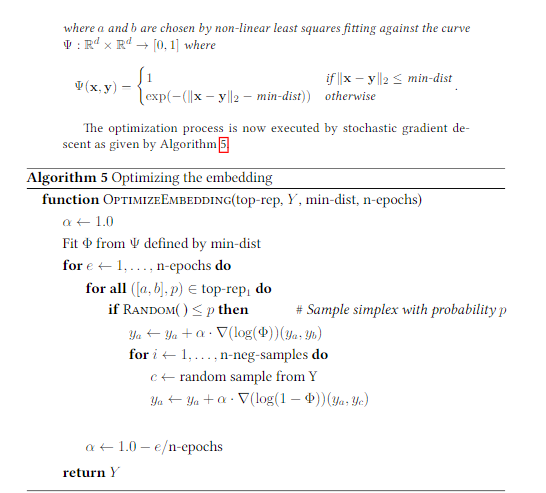
Algorithm 2 describes the construction of local fuzzy simplicial sets. To represent fuzzy simplicial sets we work with the fuzzy set images of [0]and [1] (i.e. the 1-skeleton), which we denote as fs-set0and fs-set1. One can work with higher order simplices as well, but the current implementation does not. We can construct the fuzzy simplicial set local to a given point x by finding the n-nearest neighbours, generating the appropriate normalised distance on the manifold, and then converting the finite metric space to asimplicial set via the functor FinSing, which translates into exponential of the negative distance in this case.Rather than directly using the distance to the nth nearest neighbor as the normalization, we use a smoothed version of knn-distance that fixes the cardinality of the fuzzy set of 1-simplices to a fixed value. We selectedlog2(n)for this purpose based on empirical experiments. fiis is described briefly in Algorithm 3. Spectral embedding is performed by considering the 1-skeleton of the global fuzzy topological representation as a weighted graph and using standard spectral methods on the symmetric normalized Laplacian. is pro-cess is described in Algorithm 4. final major component of UMAP is the optimization of the embedding through minimization of the fuzzy set cross entropy.











**Spectral Clustering:**

Spectral clustering is an algorithm in unsupervised machine learning that helps to cluster the data by projecting the points to higher dimensions. A simple working of spectral clustering can be stated as-

Spectral clustering works by first transforming the data from Cartesian space into similarity space and then clustering in similarity space. The original data is projected into the new coordinate space which encodes information about how nearby data points are. The similarity transformation reduces the dimensionality of space and, loosely speaking, pre-clusters the data into orthogonal dimensions. This pre-clustering is non-linear and allows for arbitrarily connected *non-convex* geometries which is the main advantage of spectral clustering.

The mapping from Cartesian space to similarity space is facilitated by the creation and diagonalization of the similarity matrix. In the case where you have k spatially separated, well defined clusters, regardless of the geometrical shape of the cluster, the resulting similarity matrix is block diagonal. Each block will correspond to a different cluster.

When you stack the lowest k-eigenvectors of this matrix as columns in a new matrix and normalize them, the rows of the matrix are the new coordinates for each data point in the new space. Ignoring degeneracy, if you inspect these new coordinates you will see that the data lies along each of the axes of your new space. These are the coordinates which are used to do the clustering and assign the original data cluster labels.

The K-means part is run because the eigenvectors can be degenerate, and the clusters don't have to be so cleanly separated. The eigenvectors span the linear space defined by the clusters. But, the clusters could sit at any coordinates in the space as long as they are rotated 90 degrees from each other relative to the origin.

On a side note: Up to the normalization steps, the similarity transformation procedure is the same one would do if applying the kernel PCA [2]. If we don't use a kernel, we get linearly separated data. Non-linear clusters can't be separated this way. Using a kernel lets us separate clustered non-convex groups using the kernel to define what "close" means. To understand the similarity transformation better, you can read a little kernel PCA to get insight to the procedure. Wikipedia has a good example using a Gaussian kernel.

* **Fitting into model**

In this task, we have implemented a K means clustering algorithm. K-means is a technique for data clustering that may be used for unsupervised machine learning. It is capable of classifying unlabeled data into a predetermined number of clusters based on similarities (k).

**Implementing DB SCAN**

# implimenting DBSCAN

from sklearn.cluster import DBSCAN

from sklearn import metrics

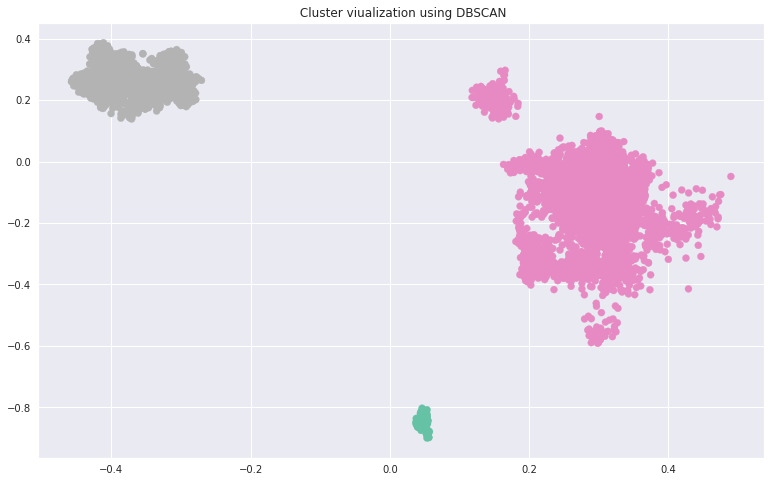
y\_pred = DBSCAN(eps=0.5, min\_samples=5, algorithm ='auto', leaf\_size = 25).fit\_predict(X)

plt.figure(figsize=(13,8))

plt.scatter(X[:,0], X[:,1], c=y\_pred, cmap = 'Set2\_r')

plt.title(' Cluster viualization using DBSCAN')

plt.show()



**Our Findings**

When it comes to nested data which is not linearly separable and is dense in nature then DB Scan is one of the best algorithm to be used , also it works best in 3 dimensional data , so we have implemented this clustering algorithm also like other algorithms .

**Our Conclusion**

As shown by the color codes of the DB Scan graph, it can be seen that the algorithm is clustering the data into 3 main clusters ..,

**Applying Hierarchical Clustering**

# Using the dendrogram to find the optimal number of clusters

import scipy.cluster.hierarchy as sch

plt.figure(figsize=(13,8))

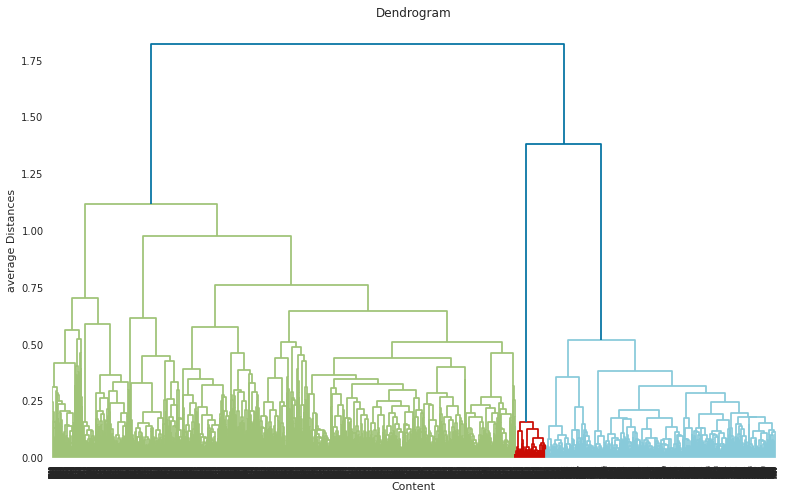
dendrogram = sch.dendrogram(sch.linkage(X, method = 'average'))

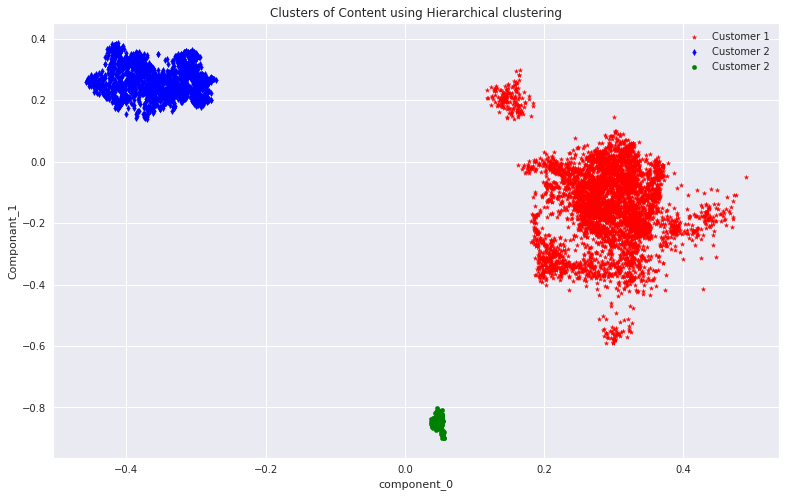
plt.title('Dendogram')

plt.xlabel('Content')

plt.ylabel('average Distances')

plt.show() # find largest vertical distance we can make without crossing any other horizontal line





**Our Findings**

We used Dendrogram here for hierarchical clustering of the same dataset , When we looked at the dendrogram , as per the ideology of reading dendrogram , we will have to horizontally slice it off hypothetically from the longest vertical line and count the no of vertical lines that intersect the horizontal line .

**Our Conclusion**

In our case we got 2 vertical lines, which symbolizes that the given data can be best divided into 2 clusters ..,

**Implementing K –Means Clustering**

#  defining a function for Silhouette scoring and visualization

from sklearn.metrics import silhouette\_score, silhouette\_samples

def silhouette\_score\_analysis(n):

  for n\_clusters in range(2,n):

      km = KMeans(n\_clusters=n\_clusters, random\_state = 42, max\_iter = 100 )

      preds = km.fit\_predict(X)

      centers = km.cluster\_centers\_

      score = silhouette\_score(X, preds, metric='euclidean')

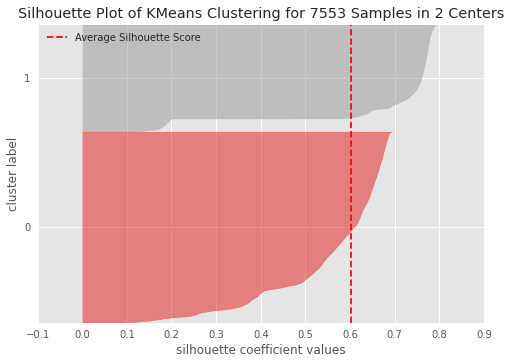
      print ("For n\_clusters = {}, silhouette score is {}".format(n\_clusters, score))

      visualizer = SilhouetteVisualizer(km)

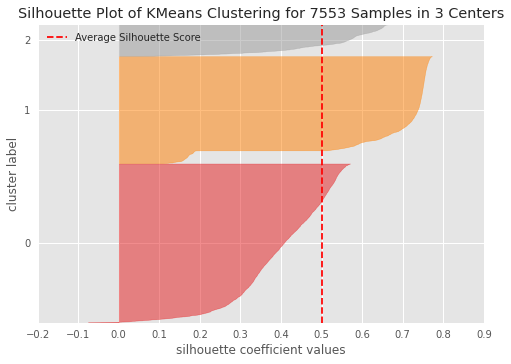
      visualizer.fit(X) # Fit the training data to the visualizer

      visualizer.poof() # Draw/show/poof the data

For n\_clusters = 2, silhouette score is 0.6014919281005859



For n\_clusters = 3, silhouette score is 0.5002426505088806



**Our Findings**

Here we found out that for 2 no. of clusters the silhouette score was 0.6 , which got dropped to 0.5 when the number of clusters became 3.

As a thumb rule the silhouette score will be between -1 to 1 , If its in negative it suggests that the data might be wrong ..,and the closest value towards 1 is considered as the optimal number and its respective no. of clusters will be considered as the optimal no. of clusters for the given data set ,

**Our Conclusion**

Here 2 No. of clusters are optimal number of clusters because its respective silhouette score 0.6 was the highest amongst all ..,

Silhouette score closest to 1 is considered as best and its respective value shown the optimal no. of clusters for the given dataset data..,