A Project Report

On

Movie Recommendation System

……

Submitted

By

Sikandar Burnwal

Roll No.: 21MA40026

2 Year M.Sc. Mathematics

Under the Supervision of

Prof. Sourav Mukhopadhyay

Department of Mathematics

Indian Institute of Technology, Kharagpur- 721302

28 November, 2023

…… ----------

(Signature of Student) (Signature of Supervisor)

ACKNOWLEDGEMENT

I am heartily thankful to my supervisor, **Prof. Sourav Mukhopadhyay**, whose encouragement, guidance and support from the initial to final level enabled me to develop an understanding of the subject. Finally, I am thankful to Department of Mathematics, IIT KHARAGPUR, which provide all the facilities to continue my project work in this semester smoothly.

Sikandar Burnwal

Abstract

In recent years, movie industry is getting more and more profitable and prosperous. Thousands of movies are released every year across the World. Now days, it became more popular as a research topic. In this project, we propose a Movie Recommendation System by Combining the Naïve Bayes Algorithm with K- means culturing. Recommendation systems are ubiquitous in versatile online platforms these days. Our aim is to build a movie recommendation system based on dataset. We wish to Segment our Movies Data in clusters which is relatable to Specific Customers data.

Keywords: Dataset, One- Hot encoding, EDA (Outlier detection), data cleaning (Outlier removal)

Problem definition

In this report, we want to implement Movies recommendation System using Naïve Bayes Classifier and K- means culturing. Here we have dataset of the people from different cultures having different interest in movies web series. We are predicting people lying in the same group according to movies in particular language.

Contents

Introduction

A recommender system is a simple algorithm whose aim is to provide the most relevant information to a user by discovering patterns in a dataset. The algorithm rates the items and shows the user the items that they would rate highly. An example of recommendation in action is when you visit Amazon and you notice that some items are being recommended to you or when Netflix recommends certain movies to you. They are also used by Music streaming applications such as Spotify and Deezer to recommend music that you might like.

The most common types of recommendation systems are content-based and collaborative filtering recommender systems. The recommendation is based on the preference of other users. A simple example would be recommending a movie to a user based on the fact that their friend liked the movie.

Collaborative Filtering: In collaborative filtering, the behaviour of a group of users is used to make recommendations to other users. They are divided into two parts:

- User-based collaborative filtering: In this model, products are recommended to a user based on the fact that the products have been liked by users similar to the user. For example, if Derrick and Dennis like the same movies and a new movie come out that Derick like, then we can recommend that movie to Dennis because Derrick and Dennis seem to like the same movies.
- Item-based collaborative filtering: These systems identify similar items based on users' previous ratings. For example, if users A, B, and C gave a 5-star rating to books X and Y then when a user D buys book Y they also get a recommendation to purchase book X because the system identifies book X and Y as similar based on the ratings of users A, B, and C.

Content-based system: It uses metadata such as genre, producer, actor, musician to recommend items say movies or music. Such a recommendation would be for instance recommending Infinity War that featured Vin Diesel because someone watched and liked The Fate of the Furious. Similarly, you can get music recommendations from certain artists because you liked their music. Content-based systems are based on the idea that if you liked a certain item, you are most likely to like something that is similar to it.

1. Dataset

Classifying objects in the field of computer science, in general, is a difficult task. However, if large data sets consisting of similar objects to classify are available, a predictive model can be obtained using various techniques. As the problem of object classification became increasingly relevant in computer applications, developments in the field of Machine Learning (ML), a sub-field of Artificial Intelligence (AI), allowed for the construction of Machine Learning Algorithms that efficiently predict, categorize, and classify distinct objects based on different features and their association to classification labels. Recommendation System is generally a subclass of information filtering system that helps to predict the preference that users would give to an item. We use "Naïve Bayes Algorithm" in this project to getting our results. Object classification problems occur frequently and are complex problems that often involve many features. Machine learning algorithms are used to effectively handle classification problems and are highly effective.

Related Work: With new applications of computer algorithms in fields that often require the solution to object classification problems such as medicine, defence contracting, transportation, and retail, machine learning techniques drastically increase safety, consumer products, and health due to the use of machine learning to solve critical problems. Furthermore, as the field of machine learning expands, new learning techniques are implemented to further increase model accuracy and reliability.

The Naïve Bayes (NB) classifier method is one simple but effective example of how supervised machine learning algorithms proficiently solve complex problems.

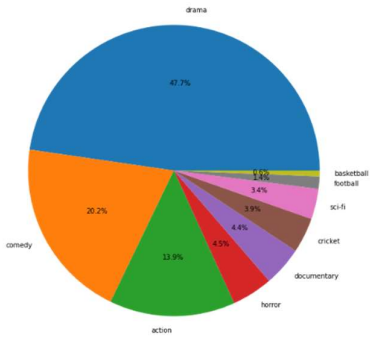
Often data that are analysed and learned using ML techniques are sensitive. In other words, the data set, such as medical records, security footage, and financial information, that predictive models are trained on cannot be accessible to the end user. This problem concerns supervised ML models such as the NB classifier.

- **About Data**: The dataset we used for this project is the "content.csv". The dataset has 48,646 rows and 9 columns. Some features of this dataset include content_id, content_type, language, genre, rating etc. The size of the "content.csv" is 2.84 MB.
- **Data Pre-processing**: It is the technique which used to covert raw data into useful and efficient format so that we can perform other processes. This involves many steps:
  1. Data cleaning like handle missing data, Noisy data that cannot be interpreted by our machine, handling data entry errors etc. Here we detect the Outlier by Clustering the data and we will manipulate this.
  2. Data transformation to transform data in appropriate form. Here we construct new attributes which helps in further process.
  3. Data Reduction used when we have huge amount of data. Since we know handling data is big task and if that data is very large with very huge volume. Then analysing data became the harder part of Data pre-processing. It aims to decrease data storage and cost of data analysis and moreover to increase the storage efficiency. In this method we use "One Hot encoding for removing some columns".

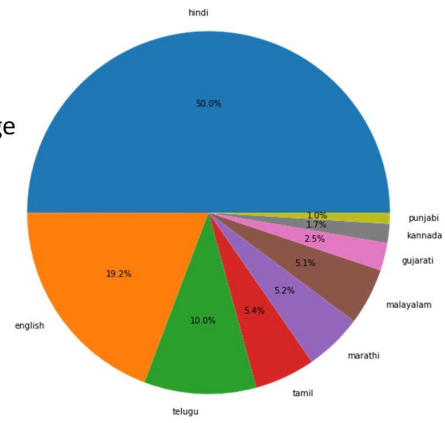
  We split the dataset into test and train based on content_id, such that 80% of the content by every single user is used for training the model and the remaining 20% of the content to validate the correctness of our model which are known as training set and test set respectively. We achieve this by using the train_test_split method in scikit-learn library.

- **Data Analysis**:

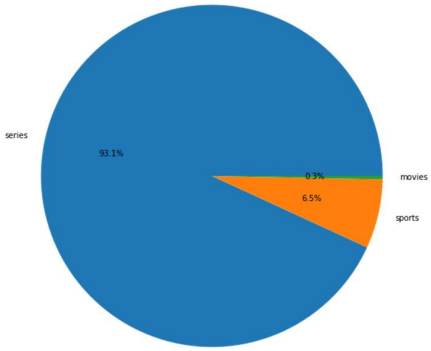
  1. Genre Distribution of data:
     Most of the contents are drama of distribution 47.7% followed by comedy 20.2%, action 13.9%, horror 4.5% and so on.

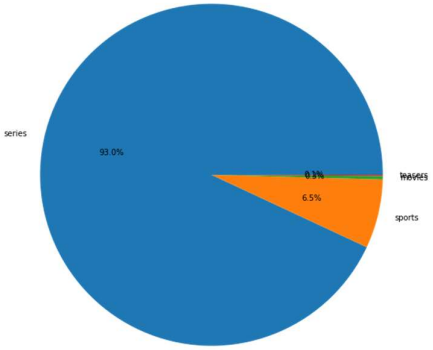


  2. Language Distribution of the data:
     Most of the contents are in hindi language of distribution 50% followed by english 19.2%, telugu 10%, tamil 5.4%, marathi 5.2% and so on.



3. Content type distribution of data:





(Top 3 content type in our data) (Distribution of all content type in our data)

We can see, 93% of the contents is series only followed by sports 6.5%, movies 0.3% and remaining are have very small contribution.

2. Prerequisites

Probability

Experiment: An experiment or trial is any procedure that can be infinitely repeated and has a well-ordered set of possible outcomes. The result of an activity or experiment is called an outcome.

Ex: - "Throw a die" is an experiment and {1,2,3,4,5,6} are the possible outcomes of the given experiment.

Sample Space: The set of all possible outcomes of an experiment is called Sample Space and is denoted by S.

Event: It is generally a subset of sample space and denoted by A, B, E, etc.

For any experiment, if |S|=n then number of events = 2^n

Independent events: Let A and B be two events that are said to be independent if

$$P(A/B) = P(A)$$

i.e., $P(A \cap B) = P(A)\,P(B)$

Partitions of Sample Space: Let S be any non-empty sample space for an experiment. The collection $\{A_i\}_{i=1}^{n}$ of non-empty subsets of S such that

(i) $\bigcup_{i=1}^{n} A_i = S$ (ii) $A_i \cap A_i = \phi$

Conditional probability: Let A and B be random variables then

$P(A/B) = P(A \cap B) / P(A)$

Mutually Exclusive events: Let A and B be two events that are said to be Mutually exclusive if

$P(A/B) = P(B/A) = 0$ i.e., $P(A \cap B) = 0$

Bayes theorem: Let $\{A_i\}_{i=1}^{n}$ be non-empty events, constitute a partition on a sample space S or are mutually exclusive and exhaustive events. Let E be the event occurs with any A_k then

$$P(A_k/E) = \frac{P(A_k)\ P(E/A_k)}{\sum_{i=1}^{n} P(A_i)\ P(E/A_i)}$$

- The formula for Bayes' theorem is given as:

$$P(A/B) = \frac{P(A)\ P(B/A)}{P(B)}$$

Where, P(A/B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B/A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

Bayes Theorem used in Machine learning to predict the classes accurately. It describes the probability of occurrence of an event related to any given condition i.e., for the case of conditional probability.

Normal Distribution: Suppose Z ~ B (n, p) then

$$P(Z = z) = \binom{n}{z} p^z q^{n-z}, \quad p + q = 1$$

If n is large and neither p nor q is very small (i.e., they are close to ½) then distribution of B (n, p) approaches to Normal Distribution.

We know for Binomial Distribution,

Mean(μ) = np, Variance (σ^2) = npq and Standard deviation(σ)= $\sqrt{npq}$

Then consider $X = \frac{Z-\mu}{\sigma} = \frac{Z-np}{\sqrt{npq}}$

When $Z = 0 : X = -\sqrt{\frac{np}{q}} \to -\infty$ and $Z = n : X = \sqrt{\frac{nq}{p}} \to \infty$ as n → ∞

So, X is Continuous random Variable, known as Normal Distribution denoted by $N(\mu, \sigma^2)$.

Suppose $X \sim N(\mu, \sigma^2)$ then Probability distribution function (p.d.f) is given by

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}}\ e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

The expression $f_X(x)$ is symmetric about $x = \mu$ and $f_X(x) > 0$ for every finite value of x.

Now, $f_X'(x) = -\frac{1}{\sigma\sqrt{2\pi}}\ e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}\ \frac{x-\mu}{\sigma}$, Where μ = mean, σ^2 = Variance

Also, for $x < \mu : f_X'(x) > 0$ i.e., $f_X(x)$ is increasing function.

for $x > \mu : f_X'(x) < 0$ i.e., $f_X(x)$ is decreasing function.

So, curve of p.d.f of X is bell-shaped with x-axis as asymptote.

Note that, $P(\mu - \sigma \leq X \leq \mu + \sigma) = 0.68$

$P(\mu - 2\sigma \leq X \leq \mu + 2\sigma) = 0.95$

$P(\mu - 3\sigma \leq X \leq \mu + 3\sigma) = 0.99$

Standard Normal Distribution: for μ=0 and σ=1, Normal distribution is known as Standard Normal Distribution. i.e., $Z \sim N(0, 1)$

For $X \sim N(\mu, \sigma^2)$, the Standard Normal Distribution is given by

$$Z = \frac{X - \mu}{\sigma}$$

Z- score: Let x be the test value then z-score is calculated as

$$z = \frac{x - \mu}{\sigma}$$

Where μ is the mean and σ is the standard value.

If the number of elements in the set is large, about 68% of the elements have a z-score between -1 and 1; about 95% have a z-score between -2 and 2 and about 99% have a z-score between -3 and 3.

Machine Learning

Feature transformation: It is a technique used to boost our model performance. In this method, we apply a mathematical expression or formula to a particular feature and transform the values of that feature or column's value to useful value for further analysis. There are of various types of transformations: Log transformation, Reciprocal transformation (not defined for Zero), Square transformation, square root transformation etc.

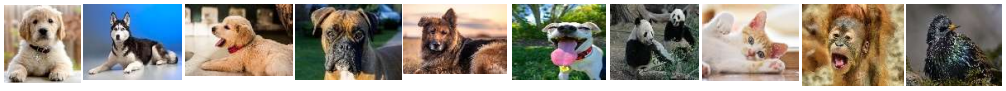
In this Project, we used Log transformation because after this transformation, our data is almost Normally distributed.

Outlier detection: An Outlier is a data that generally far from mainstream of data or piece of data that deviates drastically from average dataset. There are different Outlier Detection techniques such DBSCAN (density-based clustering), using Z-score etc.

In this Project, we use Z-score outlier detection technique. There is an assumption for using this technique that the data should be Normal distributed or almost Normally Distributed.

Cross validation: It is a method to make model better before the final implementation of the model. The data we use in Cross validation is known as Cross Validation data. We could not fit the model on training data and cannot say that the model will work fine and accurately for the real data in Machine Learning. So, we must assure that our model got the correct pattern from the data, we use Cross validation Technique.

True Positive, True Negative, False Positive and False Negative:

Images: 

Predict(Dog):	X	√	X	√	√	√	√	√	X	√
Actual:	√	√	√	√	√	√	X	X	X	X

True Positive = 4, False Positive = 3, True Negative = 1, False Negative = 2

So, Accuracy = number of accurate predictions / total number of predictions

= 5/10 = 0.5

Precision: Precision is the ratio between the True Positives and all the Positives.

$$Precision = \frac{True\ Positive}{True\ Positive + \ False\ Positive}$$

In above case: Precision = 0.57

Recall: The recall is the measure of our model correctly identifying True Positives. Recall also gives a measure of how accurately our model can identify the relevant data. It is also known as Sensitivity.

$$Recall = \frac{True\ Positive}{True\ Positive + \ False\ Negative}$$

In above case: Recall = 0.67

3. Naïve Bayes Algorithm

Classifier: A classifier is a machine learning model used to differentiate different objects on certain properties.

Naïve Bayes Classifier Algorithm: It is technique used for text classification where you have a large data. This algorithm is based on Bayes' theorem with naïve assumption that features are independence among themselves. Let $X = (x_1, x_2, x_3, \dots, x_n)$

$$P(C_k/x_1, x_2, x_3, \dots, x_n) = P(C_k/X) = \frac{1}{z}\ P(C_k) \prod_{i=1}^{n} P(x_i/C_k)$$

Where $z = P(x_1)P(x_2) \dots . P(x_n)$

Among $P(C_1/X)$, $P(C_2/X)$,...., $P(C_k/X)$, the greatest one will tag as Class Label.

Suppose we have a dataset of **weather conditions** and corresponding target variable "**Play**". So, using this dataset we need to decide that whether we should play or not on a particular day according to the weather conditions. So, to solve this problem, we need to follow the below steps:

1. Convert the given dataset into frequency tables
2. Generate Likelihood table by finding the probabilities of given features
3. Now, use Bayes theorem to calculate the posterior probability

Problem: If the weather is sunny, then the Player should play or not?

Solution: To solve this question, consider the following dataset:

	Outlook	Play
0	Rainy	Yes
1	Sunny	Yes
2	Overcast	Yes
3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes
6	Sunny	Yes
7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No
12	Overcast	Yes
13	Overcast	Yes

Frequency table for the Weather Conditions:

Weather	Yes	No	Probability
Overcast	5	0	5/14 = 0.35
Rainy	2	2	4/14 = 0.29
Sunny	3	2	5/14 = 0.35
Total	10	5	
Likelihood Probability	10/14	4/14	

Applying Bayes' theorem:

Since, P(Sunny)= 0.35, P(Yes)=0.71, P(No)= 0.29

P (Sunny/ Yes) = 3/10= 0.3, P (Sunny/ No) = 2/4=0.5

So, P (Yes/ Sunny) = P (Sunny/ Yes) *P (Yes)/ P (Sunny)= 0.3*0.71/0.35= 0.60

P (No/ Sunny) = P (Sunny/ No) *P (No)/P(Sunny) = 0.5*0.29/0.35 = 0.41

So, as we can see from the above calculation that P (Yes/ Sunny)>P (No/ Sunny)

Hence on a Sunny day, Player can play the game.

Another Example:

Suppose a table given below:

Outlook	*Yes*	*No*	*P(Yes)*	*P(No)*
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0
Rainy	3	2	3/9	2/5
Total	9	5		

Temp	*Yes*	*No*	*P(Yes)*	*P(No)*
Hot	2	2	2/9	2/5
Mild	4	2	4/9	2/5
Cool	3	1	3/9	1/5
Total	9	5		

Humidity	Yes	No	P(Yes)	P(No)
High	3	4	3/9	4/5
Normal	6	1	6/9	1/5
Total	9	5		

Wind	Yes	No	P(Yes)	P(No)
False	6	2	6/9	2/5
True	3	3	3/9	3/5
Total	9	5		

Play		Probability
Yes	9	9/14
No	5	5/14
Total	14	

Probability of playing given that the temperature is cool is

P (Temp = cool / play = Yes) = 3/9

Let us consider the set of new features as

today = (Sunny, Hot, Normal, False)

Then, Probability of Playing and Not Playing are respectively given by

P(Yes/ today) = P(Sunny/Yes) P(Hot/Yes) P(Normal/Yes) P(False/yes) P(Yes) / P(today)

P(No/ today) = P(Sunny/ No) P(Hot/ No) P(Normal/ No) P(False/ No) P(No) / P(today)

Since P(today) is common in both probabilities, so we can ignore it then

P (Yes/ today) α (2/9) * (2/9) * (6/9) * (6/9) * (9/14) = 0.0141

P (No/ today) α (3/5) * (2/5) * (1/5) * (2/5) * (5/14) = 0.0068

And we know, P (Yes/ today) + P (No/ today) =1

Then, P (Yes/ today) = (0.0141) / (0.0141 + 0.0068) = 0.67

P (No/ today) = (0.0068) / (0.0141 + 0.0068) = 0.33

Since P (Yes/ today) > P (No/ today) So, Prediction that they can play is “Yes”.

Advantages:

1. Does not require large training data
2. Easy to implement and fast to predict the class of test data
3. High scalability and unsensitive to irrelevant data

Disadvantage: It assumes that all features are independent, so it cannot learn the relationship between features.

Application:

1. It is used in Text classification such as Spam filtering
2. It can be used in real time predictions

4. K-means Clustering

Clustering: Culturing is unsupervised machine learning technique and used to find meaningful structure for further analysis. It is a task in which we divide data points into several groups such that points in the same group are more like other data points in same group and dissimilar to the data points in other groups i.e., a collection of objects based on similarity and dissimilarity between them.

Culturing is very important to determine the Outlier and for analysis of data points. There are no criteria for good culturing but it depends on the user that what is the criteria they may use to satisfy their need.

There are many Clustering algorithms such as Partitional Clustering eg, K-means (we used this in this project); Hierarchical clustering etc. A distance or similarity function need to define such as Minkowski metric, Euclidean metric, Cosine distance etc. for measuring the distance between two data points that will help to check in which cluster the data point needs to be there. The quality of Clustering result depends on the algorithm, distance function and the application for which we are using it.

K -means Clustering: It is Partitioning Clustering method in which we have K and we need to produce K clusters. Also, objects withing the cluster are similar to each other and objects belonging to different clusters are different to each other according to features. In this project we use Sum of Square for measuring distance.

The algorithm works as follows:

a. We initialize K points, called cluster centroids randomly
b. Categorize each item to its closest centroid and update centroid location which is averages of items categorized in that cluster
c. Repeat the process for given number of times to get clusters

Elbow Method: Choosing the value of K is literally a big task. There are many ways to calculate this and Elbow Method is one of the most popular methods used to find optimal number of clusters. This method used the concept of WCSS (Within Cluster Sum of Squares) value. The formula for calculating WCSS value (for 3 clusters i.e., K=3) is given by:

$$WCSS= \sum_{Pi \text{ in Cluster1}} dist(P_i\ C_1)^2 + \sum_{Pi \text{ in Cluster2}} dist(P_i\ C_2)^2 + \sum_{Pi \text{ in Cluster3}} dist(P_i\ C_3)^2$$

Where, $\sum_{Pi \text{ in Cluster1}} dist(P_i\ C_1)^2$: sum of the square of the distances between each data point and its centroid within a cluster1

To find optimal value of clusters, this method uses steps as follows:

a. Execute K-means clustering on given dataset for different k values (ranges from 1- 10)
b. For each value of k, calculate WCSS value
c. Plot a curve between WCSS value and k
d. the sharp point of bend is considered as the best value for K

5. Procedure

Aim: To segment our Movies data in clusters which is relatable to specific Customers data

Algorithm:

Naive Bayes is a popular algorithm for classifying text. It is Supervised learning algorithm. Although it is simple, it often performs as well as much more complicated solutions. A Naive Bayes' classifier works by figuring out the probability of different attributes of the data being associated with a certain class. This is based on Bayes' theorem.

The Theorem is P(A|B) =P(B|A), P(A)P(B)P(A|B) =P(B|A), P(A)P(B).

This states "the probability of A given that B is true equals the probability of B given that A is true times the probability of A being true, divided by the probability of B being true."

It is called Naïve because it assumes occurrence of certain feature is independent of that of other features. For example, if the fruit is identified on bases of shape, colour and taste then red, spherical, and sweet fruit recognized as an apple. Also, it called Bayes because it is based on Bayes' theorem.

Technical stack:

We use Python in this project. The Python libraries used here are:

1. NumPy: It is one of the most useful libraries used to handle Numerical values. NumPy is short for "Numerical Python".
2. Pandas: Pandas is python library used for working with data sets. It has functions for analysing, cleaning, and manipulating data.
3. Scikit- Learn: It is one of the most powerful and useful libraries used for Machine Learning in Python. It provides lots of efficient tools for modelling the data.
4. Matplotlib: It is one of the most popular Python Package used for Data visualization.
5. Pylab: It is a module in Matplotlib and installed alongside Matplotlib.
6. Scipy: It provides algorithms for Optimization, interpolation, differential equations and many more classes of problems. The data structure provides by Scipy is very helpful for solving such problems.
7. Seaborn: This library used for Statistical plotting in Python and provides beautiful default styles.

Actual Procedure:

1. Importing Raw data from source
2. EDA (Exploratory Data Analysis): Exploratory Data Analysis (EDA) is an approach to analyse the data using visual techniques. It is used to discover trends, patterns, or to check assumptions with the help of statistical summary and graphical representations.
   We use Pie chart, Histogram and Scatter plots to analyse our data.
3. Feature transformation: Data pre-processing is one of the many crucial steps of any Machine Learning project. As we know, our real-life data is often very unorganized and messy and without data pre-processing, there is no meaning in making a machine learning model. We must first pre-process our data and then feed that processed data to our machine learning models for good performance. One part of pre-processing is Feature Transformation.
4. Data cleaning: Our first step to remove Outliers for Predicting our data with better Accuracy

5. Making pipeline for unsupervised algorithm to make Clusters (K- means clustering)
6. Predict each data point using Supervised Learning (Categorical Naïve Bayes) whether the output is belonged to same clusters or not
7. We Transform our data into Z- score and remove the Data Correspondences which having Z -score>3 and Z- score<-3

6. GitHub Link for Python code

https://github.com/sikandarburnwal1408/Movie_Recommendation_system

Python Implementation of Movie Recommendation System

November 26, 2022

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import scipy
import scipy.stats as stats
import pylab
```

[2]:
```
df = pd.read_csv(r"content.csv")
```

[3]:
```
df
```

[3]:

	content_id	content_type	language	genre	duration	release_date	rating	episode_count	season_count
0	cont_475_19_32	series	english	drama	4980000	2018-07-01	10	32	19
1	cont_2185_15_21	series	english	drama	3000000	2016-03-29	4	21	15
2	cont_4857_13_28	series	tamil	comedy	3120000	2006-03-06	8	28	13
3	cont_3340_1_5	sports	hindi	cricket	9900000	2009-01-10	0	5	1
4	cont_1664_10_29	series	hindi	action	3660000	2020-05-25	2	29	10
...	...	...	...	...	...	...	...	...	...
48640	cont_4218_6_15	series	hindi	drama	3360000	2015-02-04	6	15	6
48641	cont_2533_1_14	series	marathi	sci-fi	3120000	2002-01-15	4	14	1
48642	cont_4606_33_5	series	hindi	drama	3180000	2006-02-18	6	5	33
48643	cont_3708_9_1	series	english	drama	4020000	2010-04-12	5	1	9
48644	cont_3470_2_4	series	english	horror	2760000	1997-03-26	8	4	2

```
[48645 rows x 9 columns]
```

[4]:
```
df.dtypes
```

[4]:
```
content_id       object
content_type     object
language         object
genre            object
duration          int64
release_date     object
rating            int64
episode_count     int64
season_count      int64
dtype: object
```

[5]:
```
df.describe
```

[5]:
```
<bound method NDFrame.describe of             content_id content_type language
genre  duration release_date  \
0       cont_475_19_32       series  english    drama   4980000   2018-07-01
1      cont_2185_15_21       series  english    drama   3000000   2016-03-29
2      cont_4857_13_28       series    tamil   comedy   3120000   2006-03-06
3        cont_3340_1_5       sports    hindi  cricket   9900000   2009-01-10
4      cont_1664_10_29       series    hindi   action   3660000   2020-05-25
...                ...          ...      ...      ...      ...          ...
48640   cont_4218_6_15       series    hindi    drama   3360000   2015-02-04
48641   cont_2533_1_14       series  marathi   sci-fi   3120000   2002-01-15
48642   cont_4606_33_5       series    hindi    drama   3180000   2006-02-18
48643    cont_3708_9_1       series  english    drama   4020000   2010-04-12
48644    cont_3470_2_4       series  english   horror   2760000   1997-03-26

       rating  episode_count  season_count
0          10             32            19
1           4             21            15
2           8             28            13
3           0              5             1
4           2             29            10
...       ...            ...           ...
48640       6             15             6
48641       4             14             1
48642       6              5            33
48643       5              1             9
48644       8              4             2

[48645 rows x 9 columns]>
```

[6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48645 entries, 0 to 48644
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   content_id     48645 non-null  object
 1   content_type   48645 non-null  object
 2   language       48645 non-null  object
 3   genre          48645 non-null  object
 4   duration       48645 non-null  int64 
 5   release_date   48645 non-null  object
 6   rating         48645 non-null  int64 
 7   episode_count  48645 non-null  int64 
 8   season_count   48645 non-null  int64 
dtypes: int64(4), object(5)
memory usage: 3.3+ MB
```

[7]:
```
df.drop(["content_id", "release_date"], axis=1, inplace=True)
```

[8]:
```
df
```

[8]:

	content_type	language	genre	duration	rating	episode_count	season_count
0	series	english	drama	4980000	10	32	19
1	series	english	drama	3000000	4	21	15
2	series	tamil	comedy	3120000	8	28	13
3	sports	hindi	cricket	9900000	0	5	1
4	series	hindi	action	3660000	2	29	10
...	...	...	...	...	...	...	...
48640	series	hindi	drama	3360000	6	15	6
48641	series	marathi	sci-fi	3120000	4	14	1
48642	series	hindi	drama	3180000	6	5	33
48643	series	english	drama	4020000	5	1	9
48644	series	english	horror	2760000	8	4	2

```
[48645 rows x 7 columns]
```

[9]:
```
df1 = pd.get_dummies(df.content_type)
```

[10]:
```
df1
```

[10]:
```
       movies  series  sports  teasers
0           0       1       0        0
1           0       1       0        0
2           0       1       0        0
3           0       0       1        0
4           0       1       0        0
...       ...     ...     ...      ...
48640       0       1       0        0
48641       0       1       0        0
48642       0       1       0        0
48643       0       1       0        0
48644       0       1       0        0

[48645 rows x 4 columns]
```

[11]:
```
df1.drop(["teasers"], axis=1, inplace=True)
```

[12]:
```
df1.tail()
```

[12]:
```
       movies  series  sports
48640       0       1       0
48641       0       1       0
48642       0       1       0
48643       0       1       0
48644       0       1       0
```

[13]:
```
df2 = pd.get_dummies(df.genre)
df3 = pd.get_dummies(df.language)
```

[14]:
```
df2.drop(["thriller"], axis=1, inplace=True)
```

[15]:
```
df3.drop(["telugu"], axis=1, inplace=True)
```

[16]:
```
df11 = pd.concat([df, df1], axis=1)
```

[17]:
```
df12 = pd.concat([df11, df2], axis=1)
```

[18]:
```
df13 = pd.concat([df12, df3], axis=1)
```

[19]:
```
df13
```

[19]:

	content_type	language	genre	duration	rating	episode_count	\
0	series	english	drama	4980000	10	32	
1	series	english	drama	3000000	4	21	
2	series	tamil	comedy	3120000	8	28	
3	sports	hindi	cricket	9900000	0	5	
4	series	hindi	action	3660000	2	29	
...	...	...	...	...	...	...	
48640	series	hindi	drama	3360000	6	15	
48641	series	marathi	sci-fi	3120000	4	14	
48642	series	hindi	drama	3180000	6	5	
48643	series	english	drama	4020000	5	1	
48644	series	english	horror	2760000	8	4	

	season_count	movies	series	sports	...	bengali	english	gujarati	\
0	19	0	1	0	...	0	1	0	
1	15	0	1	0	...	0	1	0	
2	13	0	1	0	...	0	0	0	
3	1	0	0	1	...	0	0	0	
4	10	0	1	0	...	0	0	0	
...	...	...	...	...	...	...	...	...	
48640	6	0	1	0	...	0	0	0	
48641	1	0	1	0	...	0	0	0	
48642	33	0	1	0	...	0	0	0	
48643	9	0	1	0	...	0	1	0	
48644	2	0	1	0	...	0	1	0	

	hindi	kannada	malayalam	marathi	oriya	punjabi	tamil
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	1
3	1	0	0	0	0	0	0
4	1	0	0	0	0	0	0
...	...	...	...	...	...	...	...
48640	1	0	0	0	0	0	0
48641	0	0	0	1	0	0	0
48642	1	0	0	0	0	0	0
48643	0	0	0	0	0	0	0
48644	0	0	0	0	0	0	0

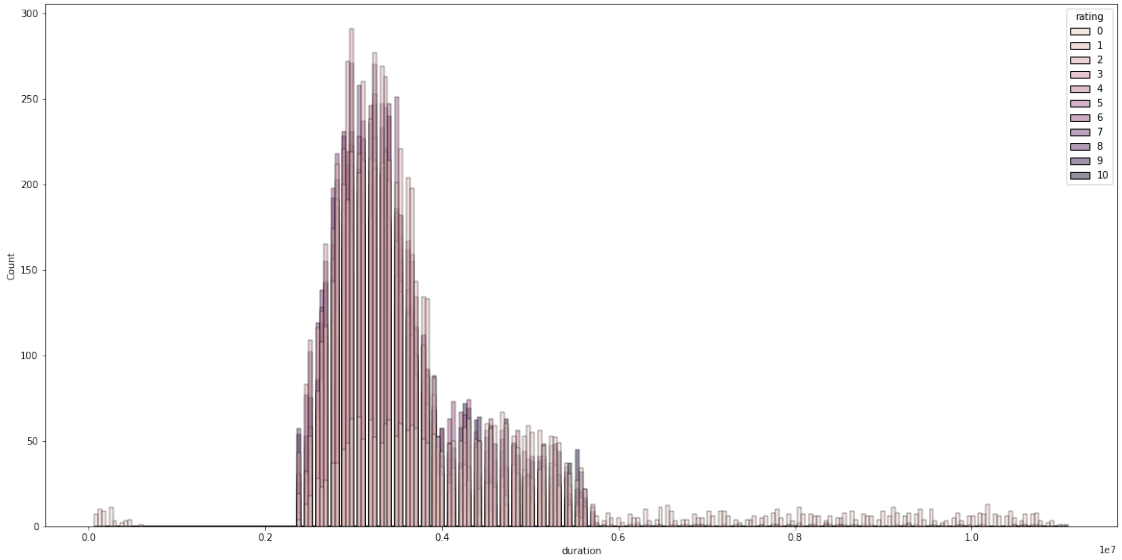
[48645 rows x 41 columns]

1 EDA

2 Histogram

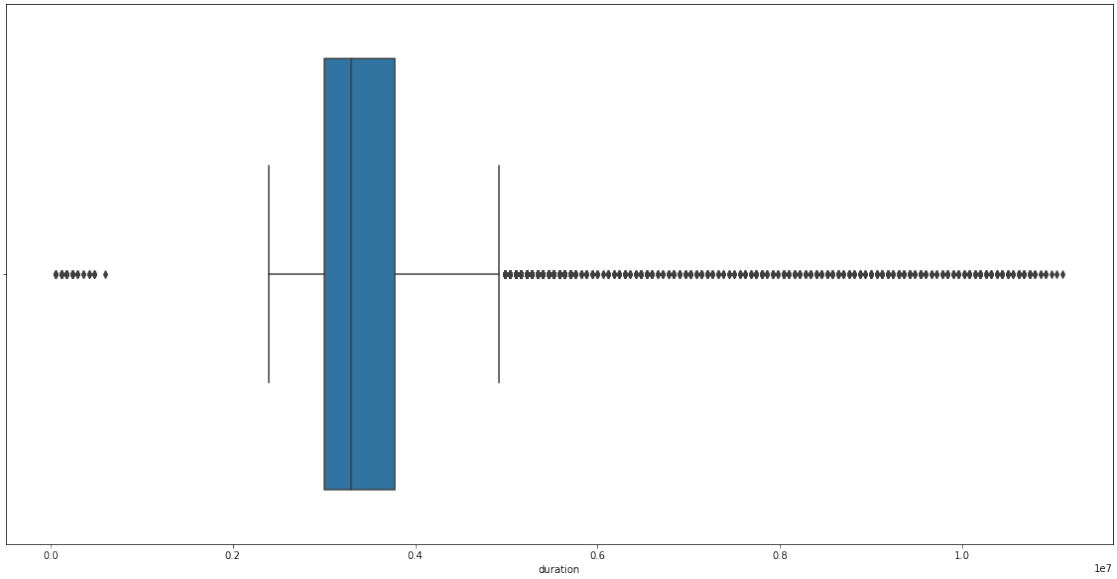
[20]:
```
plt.figure(figsize=(20, 10))
sns.histplot(x="duration", data=df13, hue="rating")
```

[20]: <AxesSubplot:xlabel='duration', ylabel='Count'>



[21]:
```
plt.figure(figsize=(15, 10))
sns.boxplot(x="duration", data=df13)
```

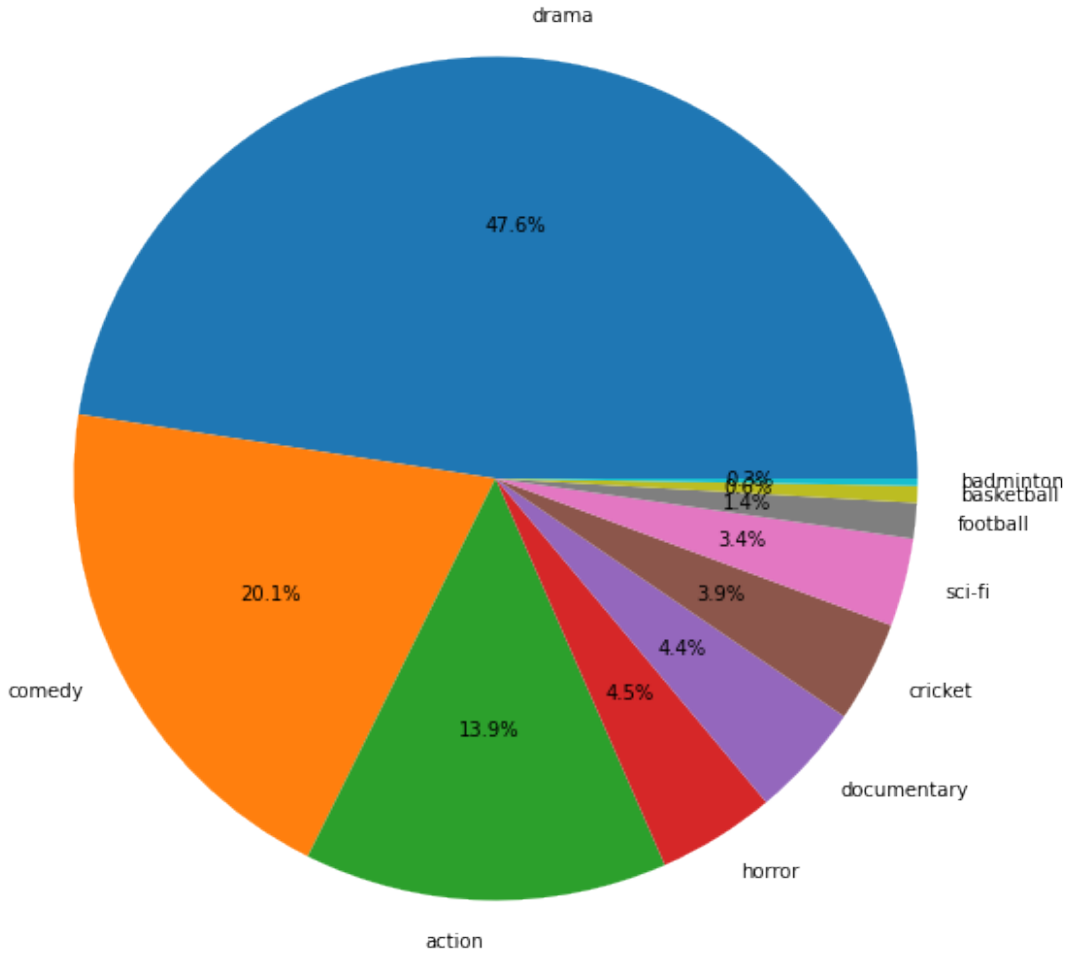
[21]: <AxesSubplot:xlabel='duration'>



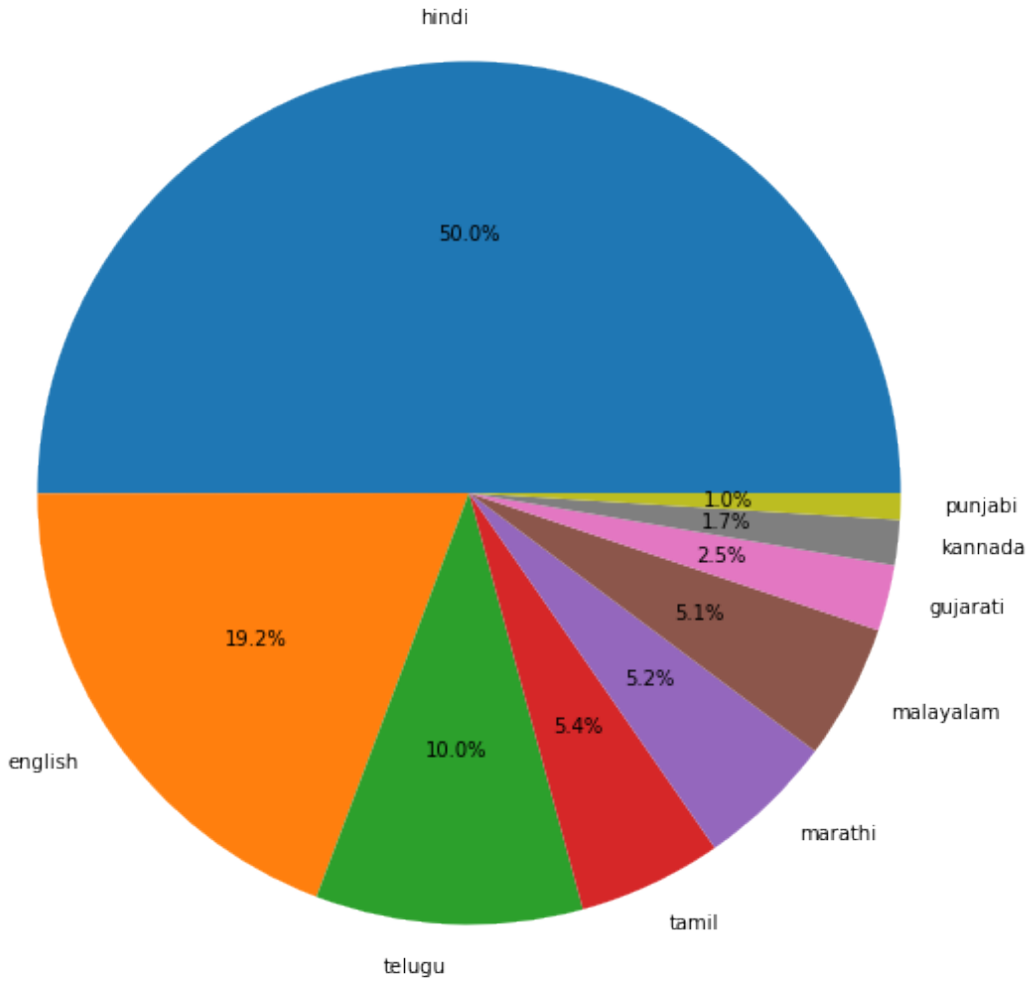
[22]:
```
df13.language.value_counts()
```

[22]:
```
hindi        23912
english       9194
telugu        4781
tamil         2577
marathi       2465
malayalam     2415
gujarati      1179
kannada        810
punjabi        474
bengali        454
oriya          384
Name: language, dtype: int64
```

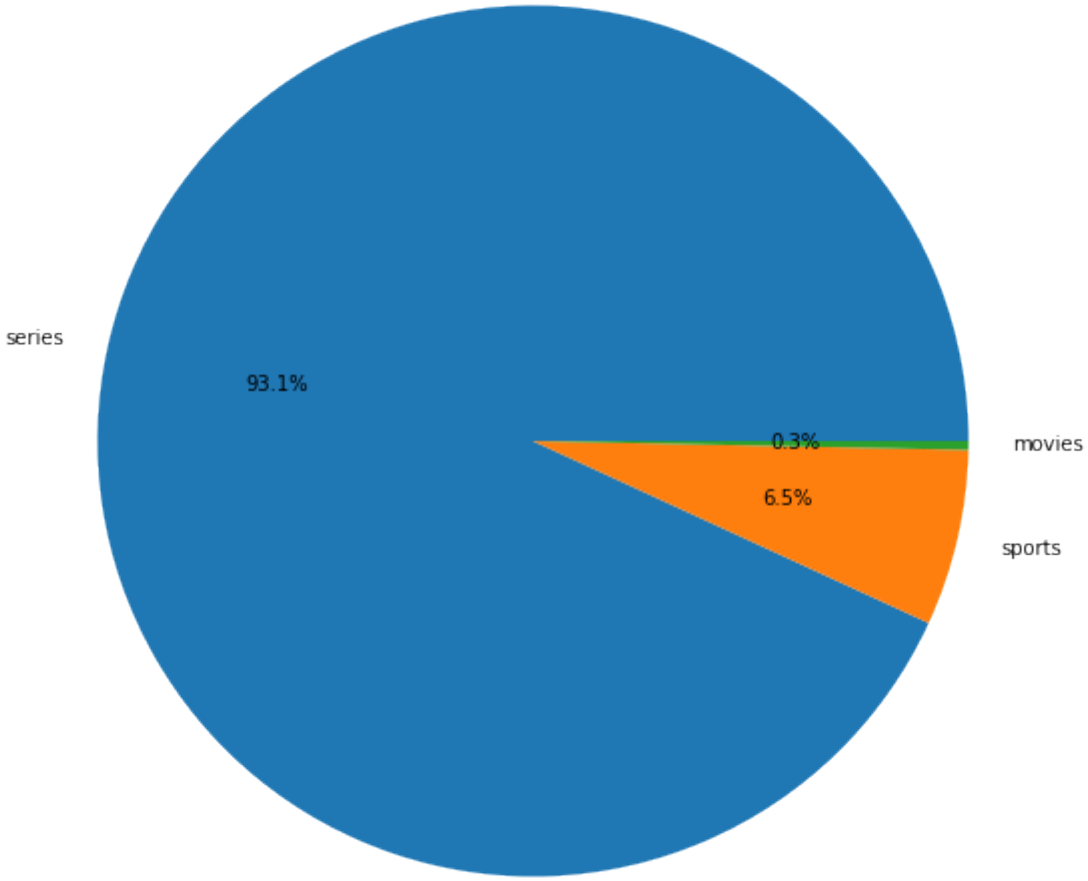
[23]:
```
plt.figure(figsize=(20, 10))
plt.pie(df13.genre.value_counts()[:10], labels=df13.genre.value_counts()[:10].
 ↪keys(), autopct="%.1f%%")
plt.show()
```



[24]:
```
plt.figure(figsize=(20, 10))
plt.pie(df13.language.value_counts()[:9], labels=df13.language.value_counts().
↪keys()[:9], autopct="%.1f%%")
plt.show()
```

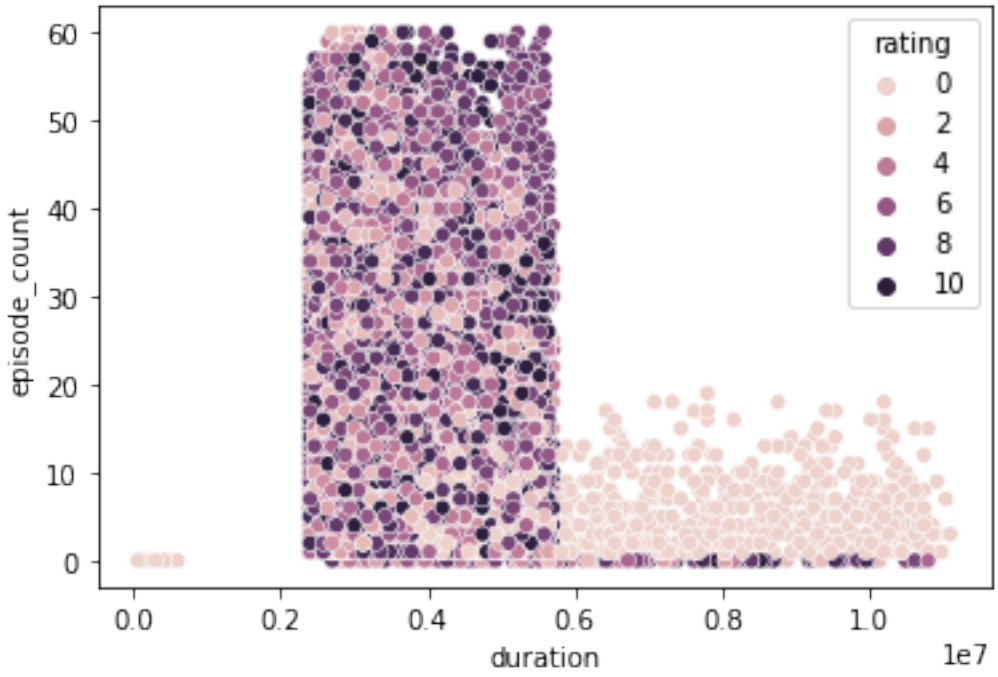


[25]:
```
plt.figure(figsize=(20, 10))
plt.pie(df13.content_type.value_counts()[:3], labels=df13.content_type.
↪value_counts()[:3].keys(), autopct="%.1f%%")
plt.show()
```



[26]:
```
sns.scatterplot(x="duration", y="episode_count", data=df13, hue="rating")
```

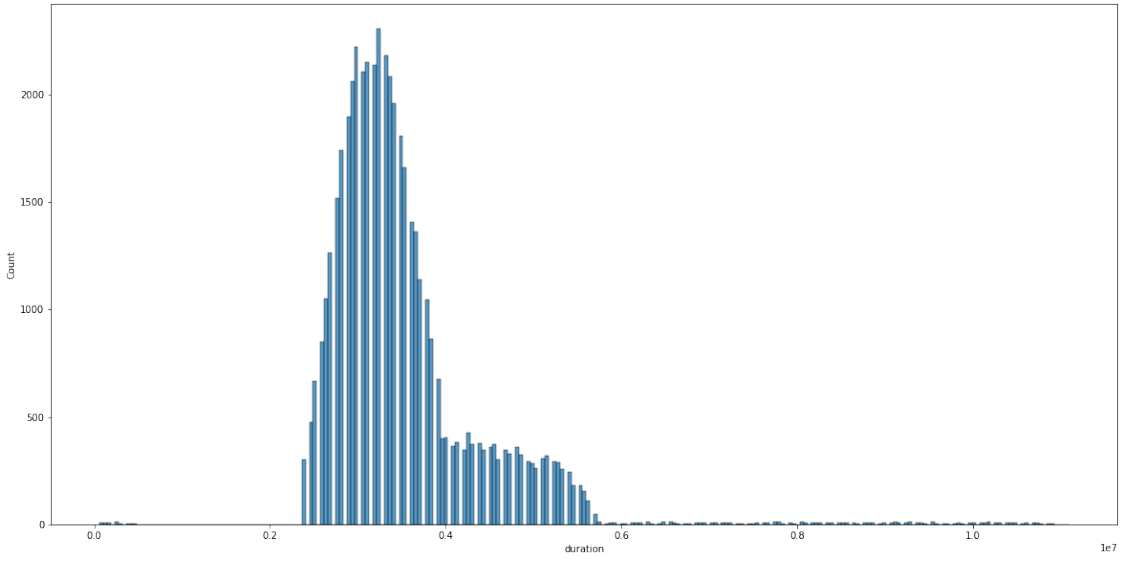
[26]:
```
<AxesSubplot:xlabel='duration', ylabel='episode_count'>
```



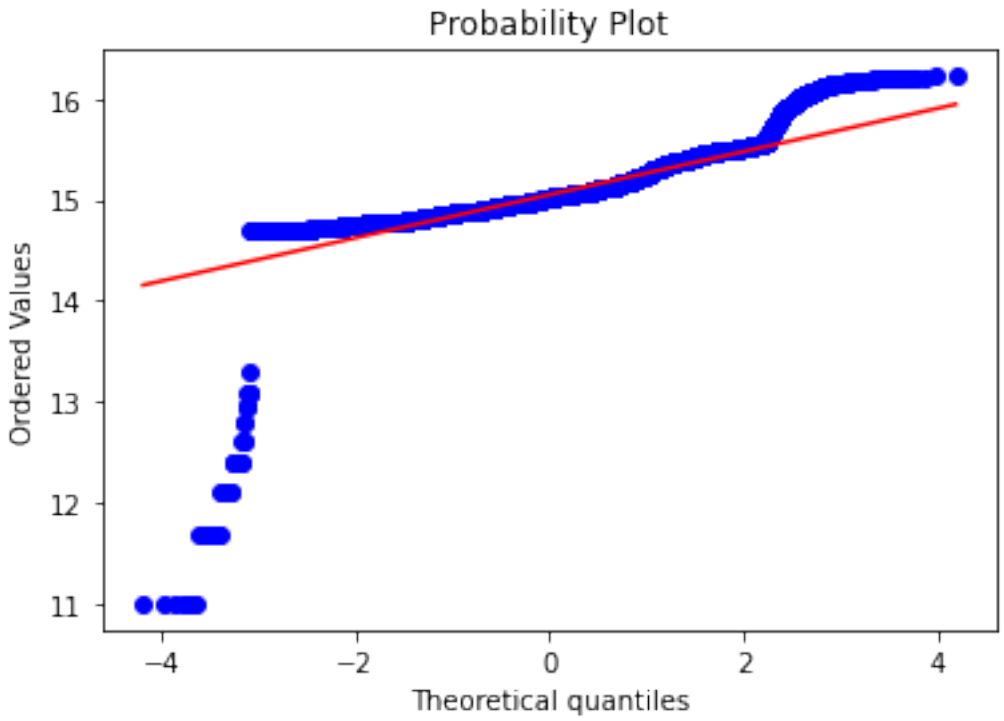
3 feature transformation

[27]:
```
plt.figure(figsize=(20, 10))
sns.histplot(x="duration", data=df13)
```

[27]: <AxesSubplot:xlabel='duration', ylabel='Count'>

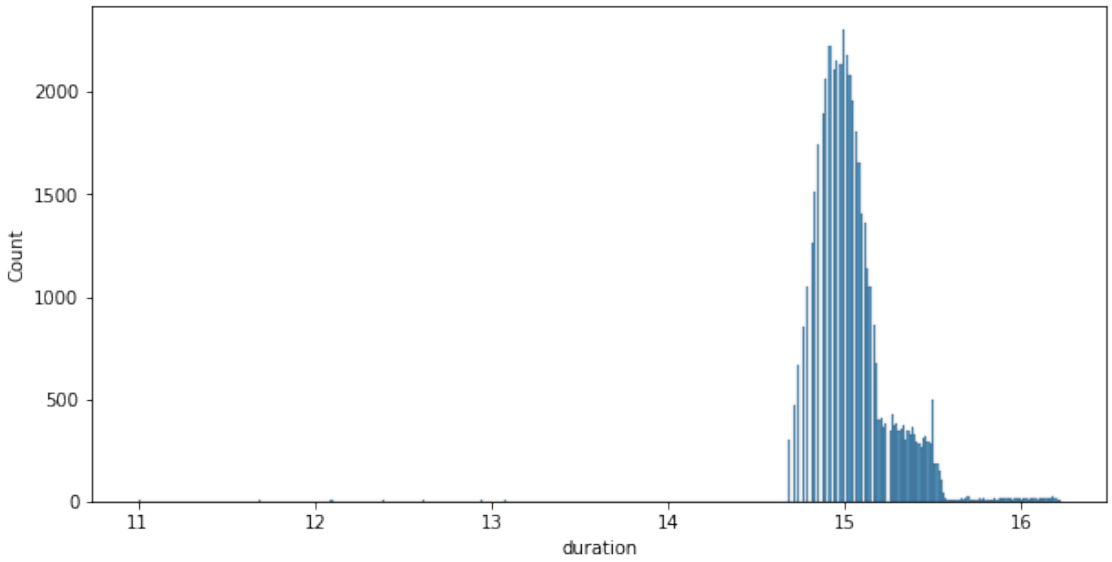


[28]:
```
stats.probplot(np.log(df13.duration), dist="norm", plot=pylab)
pylab.show()
```



[29]:
```
plt.figure(figsize=(10, 5))
sns.histplot(np.log(df13.duration))
```

[29]:
```
<AxesSubplot:xlabel='duration', ylabel='Count'>
```



[30]:
```
df13["normal_duration"] = np.log(df13.duration)
# df13["normal_duration"] = 1/(df13.duration)
```

[31]:
```
df13
```

[31]:

	content_type	language	genre	duration	rating	episode_count	\
0	series	english	drama	4980000	10	32	
1	series	english	drama	3000000	4	21	
2	series	tamil	comedy	3120000	8	28	
3	sports	hindi	cricket	9900000	0	5	
4	series	hindi	action	3660000	2	29	
...	...	...	...	...	...	...	
48640	series	hindi	drama	3360000	6	15	
48641	series	marathi	sci-fi	3120000	4	14	
48642	series	hindi	drama	3180000	6	5	
48643	series	english	drama	4020000	5	1	
48644	series	english	horror	2760000	8	4	

	season_count	movies	series	sports	...	english	gujarati	hindi	\
0	19	0	1	0	...	1	0	0	
1	15	0	1	0	...	1	0	0	
2	13	0	1	0	...	0	0	0	
3	1	0	0	1	...	0	0	1	
4	10	0	1	0	...	0	0	1	
...	...	...	...	...	...	...	...	...	
48640	6	0	1	0	...	0	0	1	
48641	1	0	1	0	...	0	0	0	
48642	33	0	1	0	...	0	0	1	

```
48643            9       0       1       0  ...        1         0      0
48644            2       0       1       0  ...        1         0      0

       kannada  malayalam  marathi  oriya  punjabi  tamil  normal_duration
0            0          0        0      0        0      0        15.420940
1            0          0        0      0        0      0        14.914123
2            0          0        0      0        0      1        14.953344
3            0          0        0      0        0      0        16.108045
4            0          0        0      0        0      0        15.112974
...        ...        ...      ...    ...      ...    ...              ...
48640        0          0        0      0        0      0        15.027452
48641        0          0        1      0        0      0        14.953344
48642        0          0        0      0        0      0        14.972392
48643        0          0        0      0        0      0        15.206792
48644        0          0        0      0        0      0        14.830741

[48645 rows x 42 columns]
```

[32]:
```
droplist = ["content_type", "language", "genre", "duration"]
```

[33]:
```
df13.drop(droplist, axis=1, inplace=True)
```

[34]:
```
df13
```

[34]:
```
       rating  episode_count  season_count  movies  series  sports  action  \
0          10             32            19       0       1       0       0
1           4             21            15       0       1       0       0
2           8             28            13       0       1       0       0
3           0              5             1       0       0       1       0
4           2             29            10       0       1       0       1
...       ...            ...           ...     ...     ...     ...     ...
48640       6             15             6       0       1       0       0
48641       4             14             1       0       1       0       0
48642       6              5            33       0       1       0       0
48643       5              1             9       0       1       0       0
48644       8              4             2       0       1       0       0

       adventure  animation  badminton  ...  english  gujarati  hindi  \
0              0          0          0  ...        1         0      0
1              0          0          0  ...        1         0      0
2              0          0          0  ...        0         0      0
3              0          0          0  ...        0         0      1
4              0          0          0  ...        0         0      1
...          ...        ...        ...  ...      ...       ...    ...
48640          0          0          0  ...        0         0      1
48641          0          0          0  ...        0         0      0
48642          0          0          0  ...        0         0      1
```

```
48643          0          0          0  ...        1         0      0
48644          0          0          0  ...        1         0      0

       kannada  malayalam  marathi  oriya  punjabi  tamil  normal_duration
0            0          0        0      0        0      0        15.420940
1            0          0        0      0        0      0        14.914123
2            0          0        0      0        0      1        14.953344
3            0          0        0      0        0      0        16.108045
4            0          0        0      0        0      0        15.112974
...        ...        ...      ...    ...      ...    ...              ...
48640        0          0        0      0        0      0        15.027452
48641        0          0        1      0        0      0        14.953344
48642        0          0        0      0        0      0        14.972392
48643        0          0        0      0        0      0        15.206792
48644        0          0        0      0        0      0        14.830741

[48645 rows x 38 columns]
```

4 Outlier dect

using z score

[35]:
```
z = np.abs(df13.normal_duration-df13.normal_duration.mean()) / \
    df13.normal_duration.std()
```

[36]:
```
df13["ob"] = z[z > 3]
#ob is set of outlier
```

[37]:
```
df13
```

[37]:
```
       rating  episode_count  season_count  movies  series  sports  action  \
0          10             32            19       0       1       0       0
1           4             21            15       0       1       0       0
2           8             28            13       0       1       0       0
3           0              5             1       0       0       1       0
4           2             29            10       0       1       0       1
...       ...            ...           ...     ...     ...     ...     ...
48640       6             15             6       0       1       0       0
48641       4             14             1       0       1       0       0
48642       6              5            33       0       1       0       0
48643       5              1             9       0       1       0       0
48644       8              4             2       0       1       0       0

       adventure  animation  badminton  ...  gujarati  hindi  kannada  \
0              0          0          0  ...         0      0        0
1              0          0          0  ...         0      0        0
2              0          0          0  ...         0      0        0
```

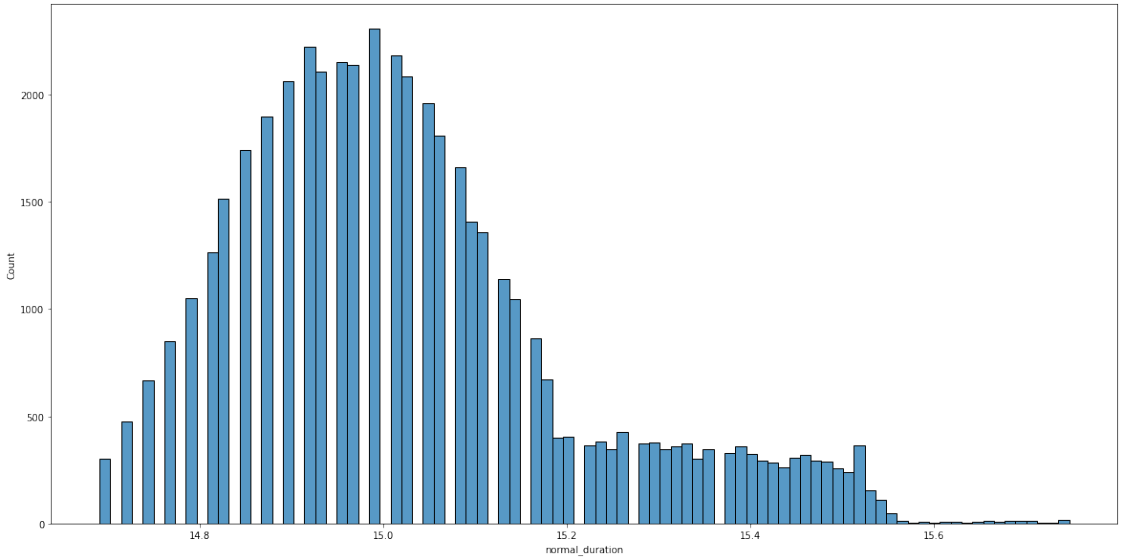
```
3              0          0          0  ...          0      1        0
4              0          0          0  ...          0      1        0
...          ...        ...        ...  ...        ...    ...      ...
48640          0          0          0  ...          0      1        0
48641          0          0          0  ...          0      0        0
48642          0          0          0  ...          0      1        0
48643          0          0          0  ...          0      0        0
48644          0          0          0  ...          0      0        0

       malayalam  marathi  oriya  punjabi  tamil  normal_duration        ob
0              0        0      0        0      0        15.420940       NaN
1              0        0      0        0      0        14.914123       NaN
2              0        0      0        0      1        14.953344       NaN
3              0        0      0        0      0        16.108045  4.526122
4              0        0      0        0      0        15.112974       NaN
...          ...      ...    ...      ...    ...              ...       ...
48640          0        0      0        0      0        15.027452       NaN
48641          0        1      0        0      0        14.953344       NaN
48642          0        0      0        0      0        14.972392       NaN
48643          0        0      0        0      0        15.206792       NaN
48644          0        0      0        0      0        14.830741       NaN

[48645 rows x 39 columns]
```

[38]:
```
a = []
for i in range(0, len(df13["ob"])):
    if df13["ob"][i] > 3:
        a.append(i)
```

[39]:
```
df13.drop(a, inplace=True)
#outlier removed
```

[40]:
```
plt.figure(figsize=(20, 10))
sns.histplot(df13.normal_duration)
```

[40]:
```
<AxesSubplot:xlabel='normal_duration', ylabel='Count'>
```



[41]:
```
df13.drop(["ob"], axis="columns", inplace=True)
```

[42]:
```
df13
```

[42]:

	rating	episode_count	season_count	movies	series	sports	action	\
0	10	32	19	0	1	0	0	
1	4	21	15	0	1	0	0	
2	8	28	13	0	1	0	0	
4	2	29	10	0	1	0	1	
5	10	37	1	0	1	0	0	
...	...	...	...	...	...	...	...	
48640	6	15	6	0	1	0	0	
48641	4	14	1	0	1	0	0	
48642	6	5	33	0	1	0	0	
48643	5	1	9	0	1	0	0	
48644	8	4	2	0	1	0	0	

	adventure	animation	badminton	...	english	gujarati	hindi	\
0	0	0	0	...	1	0	0	
1	0	0	0	...	1	0	0	
2	0	0	0	...	0	0	0	
4	0	0	0	...	0	0	1	
5	0	0	0	...	0	0	1	
...	...	...	...	...	...	...	...	
48640	0	0	0	...	0	0	1	
48641	0	0	0	...	0	0	0	
48642	0	0	0	...	0	0	1	
48643	0	0	0	...	1	0	0	

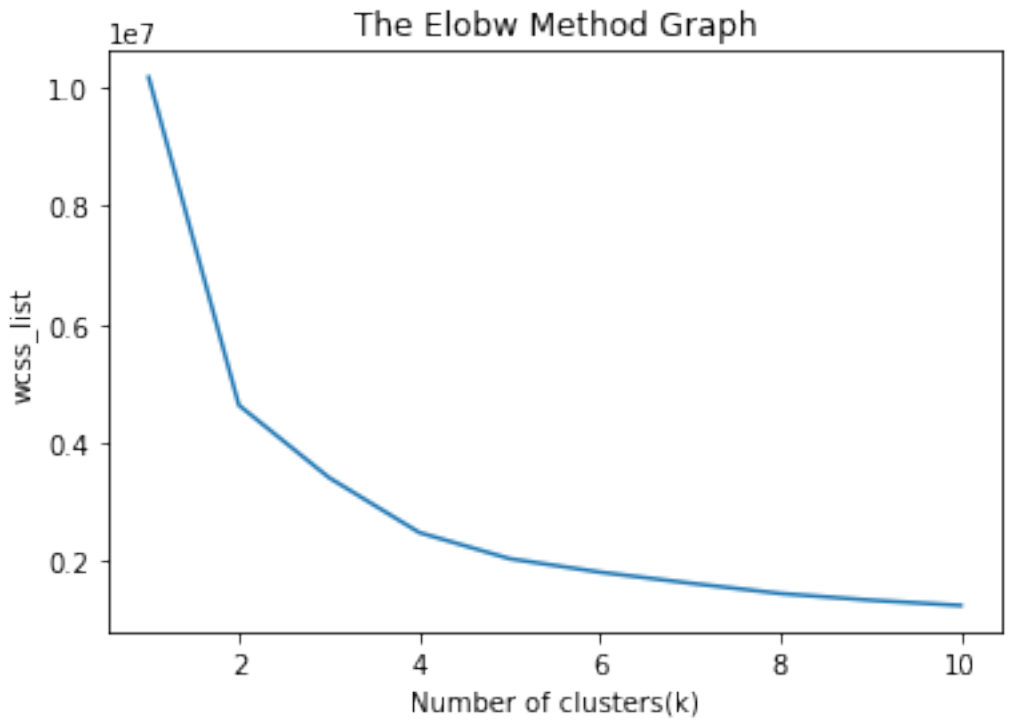
```
48644          0          0          0  ...         1         0      0

       kannada  malayalam  marathi  oriya  punjabi  tamil  normal_duration
0            0          0        0      0        0      0        15.420940
1            0          0        0      0        0      0        14.914123
2            0          0        0      0        0      1        14.953344
4            0          0        0      0        0      0        15.112974
5            0          0        0      0        0      0        14.933925
...        ...        ...      ...    ...      ...    ...              ...
48640        0          0        0      0        0      0        15.027452
48641        0          0        1      0        0      0        14.953344
48642        0          0        0      0        0      0        14.972392
48643        0          0        0      0        0      0        15.206792
48644        0          0        0      0        0      0        14.830741

[48113 rows x 38 columns]
```

5 Kmeans clustering

[43]:
```
from sklearn.cluster import KMeans
```

[44]:
```
x = df13
```

[45]:
```
from sklearn.cluster import KMeans
from sklearn import metrics
from scipy.spatial.distance import cdist
import numpy as np
import matplotlib.pyplot as plt
wcss_list= []  #Initializing the list for the values of WCSS

#Using for loop for iterations from 1 to 10.
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state= 42)
    kmeans.fit(x)
    wcss_list.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss_list)
plt.title('The Elobw Method Graph')
plt.xlabel('Number of clusters(k)')
plt.ylabel('wcss_list')
plt.show()
```



[46]:
```
kmeans = KMeans(n_clusters=4, random_state=0).fit(x)
```

[47]:
```
kmeans.cluster_centers_
```

[47]:
```
array([[ 5.22073237e+00,  2.15353934e+01,  5.79703240e+00,
        -2.68882139e-17,  9.91015519e-01,  8.98448135e-03,
         1.51102641e-01, -2.34458720e-18,  2.81892565e-18,
         3.40321263e-04,  3.40321263e-04, -2.41234983e-18,
         2.09501770e-01,  5.37707596e-03, -4.82469967e-18,
         4.71004628e-02,  5.05513204e-01, -4.68917440e-18,
        -4.82469967e-18,  2.17805608e-03,  2.04192758e-04,
         4.67601416e-02, -1.17229360e-18, -1.17229360e-18,
         3.10372992e-02, -1.17229360e-18,  5.44514021e-04,
         7.75932480e-03,  1.76150286e-01,  2.75660223e-02,
         4.94690988e-01,  1.64034849e-02,  4.87340049e-02,
         4.36291860e-02,  7.14674653e-03,  8.23577457e-03,
         6.03729921e-02,  1.50341519e+01],
       [ 4.70429477e+00,  6.58023938e+00,  4.60023469e+00,
         4.55292185e-03,  8.72330439e-01,  1.23116639e-01,
         1.14527106e-01,  1.87749355e-04,  3.28561371e-04,
         4.55292185e-03,  1.19220840e-02,  1.40812016e-04,
         1.89486036e-01,  7.31283736e-02,  2.81624032e-04,
         4.27129782e-02,  4.48439333e-01,  3.75498709e-04,
```

```
  2.81624032e-04,  2.57685989e-02,  4.03661112e-03,
  4.21497301e-02,  9.38746773e-05,  9.38746773e-05,
  3.73151842e-02,  9.38746773e-05,  3.70804975e-03,
  1.01384651e-02,  2.01361183e-01,  2.70359071e-02,
  4.88336071e-01,  1.59586951e-02,  4.97535790e-02,
  4.57169678e-02,  7.88547289e-03,  9.38746773e-03,
  5.05045764e-02,  1.50508636e+01],
[ 5.50291181e+00,  3.97537438e+01,  5.37118691e+00,
  4.42354486e-17,  1.00000000e+00, -1.06858966e-15,
  1.86633389e-01,  1.49077799e-19, -1.70761842e-18,
  3.81639165e-17,  5.98479599e-17, -1.42301535e-18,
  1.99805879e-01,  2.91433544e-16, -2.84603070e-18,
  4.61730449e-02,  5.00831947e-01,  2.98155597e-19,
 -2.84603070e-18,  1.07552856e-16, -2.16840434e-17,
  4.56184138e-02,  7.45388994e-20,  7.45388994e-20,
  2.09373267e-02,  7.45388994e-20, -1.25767452e-17,
  6.37825846e-03,  1.45729340e-01,  1.15085968e-02,
  5.19412091e-01,  1.78868552e-02,  3.36938436e-02,
  8.20854132e-02,  1.09539656e-02,  1.41430948e-02,
  6.07321131e-02,  1.50345333e+01],
[ 5.76407015e+00,  8.79547308e+00,  2.02601958e+01,
  1.51788304e-17,  1.00000000e+00, -6.93889390e-17,
  1.43352365e-01,  1.76182853e-19, -1.97866896e-18,
  3.12250226e-17, -1.82145965e-17, -5.14996032e-19,
  2.32463295e-01,  2.70616862e-16, -1.02999206e-18,
  3.67047308e-02,  4.87969005e-01,  3.52365706e-19,
 -1.02999206e-18, -6.93889390e-17, -1.56125113e-17,
  5.34257749e-02,  8.80914265e-20,  8.80914265e-20,
  4.60848287e-02,  8.80914265e-20, -1.19262239e-17,
  1.52936378e-02,  2.09624796e-01,  2.26345840e-02,
  4.53915171e-01,  2.01876020e-02,  8.07504078e-02,
  5.15905383e-02,  6.32137031e-03,  1.03996737e-02,
  3.48694943e-02,  1.50386752e+01]])
```

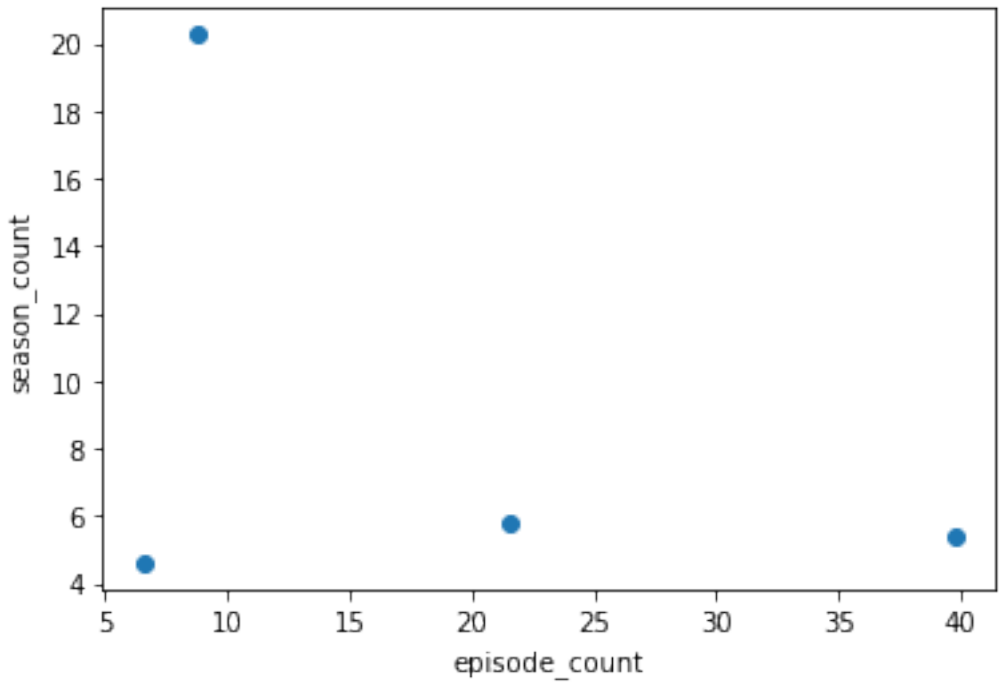
[48]:
```
df13["cluster"] = kmeans.predict(x)
```

[49]:
```
x = kmeans.cluster_centers_[:, 1]
```

[50]:
```
y = kmeans.cluster_centers_[:, 2]
```

[51]:
```
plt.scatter(x, y)
plt.xlabel("episode_count")
plt.ylabel("season_count")
```

[51]:
```
Text(0, 0.5, 'season_count')
```



[52]:
```
df1 = pd.read_csv(r"content.csv")
```

[53]:
```
df1
```

[53]:

	content_id	content_type	language	genre	duration	release_date	rating	episode_count	season_count
0	cont_475_19_32	series	english	drama	4980000	2018-07-01	10	32	19
1	cont_2185_15_21	series	english	drama	3000000	2016-03-29	4	21	15
2	cont_4857_13_28	series	tamil	comedy	3120000	2006-03-06	8	28	13
3	cont_3340_1_5	sports	hindi	cricket	9900000	2009-01-10	0	5	1
4	cont_1664_10_29	series	hindi	action	3660000	2020-05-25	2	29	10
...	...	...	...	...	...	...	...	...	...
48640	cont_4218_6_15	series	hindi	drama	3360000	2015-02-04			
48641	cont_2533_1_14	series	marathi	sci-fi	3120000	2002-01-15			
48642	cont_4606_33_5	series	hindi	drama	3180000	2006-02-18			
48643	cont_3708_9_1	series	english	drama	4020000	2010-04-12			
48644	cont_3470_2_4	series	english	horror	2760000	1997-03-26			

```
48640       6              15              6
48641       4              14              1
48642       6               5             33
48643       5               1              9
48644       8               4              2

[48645 rows x 9 columns]
```

6 Checking Correlation

[54]:
```
df13.corr()
```

[54]:
```
                  rating  episode_count  season_count    movies    series  \
rating          1.000000       0.080658      0.113670  0.005182  0.401427
episode_count   0.080658       1.000000     -0.098487 -0.057595  0.212568
season_count    0.113670      -0.098487      1.000000 -0.048768  0.232964
movies          0.005182      -0.057595     -0.048768  1.000000 -0.179052
series          0.401427       0.212568      0.232964 -0.179052  1.000000
sports         -0.408996      -0.204927     -0.227361 -0.011077 -0.981796
action          0.030891       0.076420      0.024063  0.000634  0.097489
adventure       0.006423      -0.011684     -0.009894  0.202873 -0.036325
animation       0.007513      -0.015458     -0.013088  0.268384 -0.048055
badminton      -0.076492      -0.039331     -0.042522 -0.002072 -0.183619
basketball     -0.122089      -0.064945     -0.067869 -0.003307 -0.293075
biography       0.007493      -0.010119     -0.008568  0.175691 -0.031458
comedy          0.039144       0.011787      0.031856 -0.002943  0.122379
cricket        -0.311455      -0.155980     -0.173138 -0.008435 -0.747648
crime          -0.000327      -0.014311     -0.012117  0.248473 -0.044490
documentary     0.033428       0.011608     -0.005449 -0.000597  0.052109
drama           0.097922       0.043073      0.054142 -0.037423  0.239033
family         -0.000377      -0.016525     -0.013992  0.286917 -0.051373
fantasy         0.002101      -0.014311     -0.012117  0.248473 -0.044490
football       -0.183476      -0.089668     -0.101994 -0.004969 -0.440436
hockey         -0.071442      -0.036659     -0.039714 -0.001935 -0.171496
horror          0.008406      -0.000334      0.020526  0.003597  0.052092
musical         0.000862      -0.008262     -0.006996  0.143450 -0.025685
mystery        -0.002291      -0.008262     -0.006996  0.143450 -0.025685
sci-fi          0.060079      -0.033933      0.014385 -0.003286  0.046003
sport           0.004016      -0.008262     -0.006996  0.143450 -0.025685
tennis         -0.070633      -0.033062     -0.039265 -0.001913 -0.169554
bengali         0.042514      -0.021826      0.015473  0.000436  0.023505
english        -0.080065      -0.060837     -0.011581 -0.001260 -0.187695
gujarati        0.043447      -0.024130     -0.000656  0.007898  0.036865
hindi           0.010255       0.027815     -0.025685  0.001234 -0.010291
kannada        -0.020218       0.006065      0.008483 -0.005878  0.032828
malayalam       0.003703      -0.032441      0.058530 -0.010332  0.057706
```

```
marathi          -0.001591       0.052725     -0.002844 -0.002019  0.056672
oriya             0.036806       0.015305     -0.014117 -0.004026  0.022486
punjabi           0.010736       0.010415      0.003006 -0.004483  0.025039
tamil            -0.007585       0.019454      0.001797  0.001678  0.057307
normal_duration  -0.069210      -0.037832     -0.038589  0.068853 -0.238329
cluster           0.056126       0.032503      0.497782 -0.002352  0.025658

                    sports    action  adventure  animation  badminton  ...  \
rating           -0.408996  0.030891   0.006423   0.007513  -0.076492  ...
episode_count    -0.204927  0.076420  -0.011684  -0.015458  -0.039331  ...
season_count     -0.227361  0.024063  -0.009894  -0.013088  -0.042522  ...
movies           -0.011077  0.000634   0.202873   0.268384  -0.002072  ...
series           -0.981796  0.097489  -0.036325  -0.048055  -0.183619  ...
sports            1.000000 -0.099207  -0.002247  -0.002973   0.187023  ...
action           -0.099207  1.000000  -0.003670  -0.004856  -0.018554  ...
adventure        -0.002247 -0.003670   1.000000  -0.000110  -0.000420  ...
animation        -0.002973 -0.004856  -0.000110   1.000000  -0.000556  ...
badminton         0.187023 -0.018554  -0.000420  -0.000556   1.000000  ...
basketball        0.298509 -0.029614  -0.000671  -0.000887  -0.003391  ...
biography        -0.001946 -0.003179  -0.000072  -0.000095  -0.000364  ...
comedy           -0.123814 -0.202228  -0.004581  -0.006060  -0.023156  ...
cricket           0.761511 -0.075547  -0.001711  -0.002264  -0.008650  ...
crime            -0.002752 -0.004496  -0.000102  -0.000135  -0.000515  ...
documentary      -0.052847 -0.086316  -0.001955  -0.002587  -0.009884  ...
drama            -0.235719 -0.385007  -0.008721  -0.011537  -0.044085  ...
family           -0.003178 -0.005191  -0.000118  -0.000156  -0.000594  ...
fantasy          -0.002752 -0.004496  -0.000102  -0.000135  -0.000515  ...
football          0.448602 -0.044504  -0.001008  -0.001334  -0.005096  ...
hockey            0.174675 -0.017329  -0.000393  -0.000519  -0.001984  ...
horror           -0.053639 -0.087610  -0.001985  -0.002625  -0.010032  ...
musical          -0.001589 -0.002595  -0.000059  -0.000078  -0.000297  ...
mystery          -0.001589 -0.002595  -0.000059  -0.000078  -0.000297  ...
sci-fi           -0.046122 -0.075332  -0.001706  -0.002257  -0.008626  ...
sport            -0.001589 -0.002595  -0.000059  -0.000078  -0.000297  ...
tennis            0.172698 -0.017133  -0.000388  -0.000513  -0.001962  ...
bengali          -0.023974  0.000692  -0.000887  -0.001173  -0.004484  ...
english           0.191009  0.015108   0.001495  -0.001342   0.022071  ...
gujarati         -0.038993 -0.055526  -0.001443  -0.001909  -0.007293  ...
hindi             0.010221  0.007083  -0.004403  -0.001517   0.012541  ...
kannada          -0.032230  0.083152  -0.001192  -0.001578  -0.006028  ...
malayalam        -0.056656 -0.077699  -0.002096  -0.002773  -0.010596  ...
marathi          -0.057209  0.046845  -0.002117  -0.002800  -0.010699  ...
oriya            -0.022077  0.051714  -0.000817  -0.001081  -0.004129  ...
punjabi          -0.024583 -0.016461  -0.000910  -0.001203  -0.004598  ...
tamil            -0.058569  0.049707  -0.002167   0.004794  -0.010954  ...
normal_duration   0.228936 -0.012892   0.018724   0.017119   0.047451  ...
cluster          -0.025624  0.013480  -0.000477  -0.000631  -0.004847  ...
```

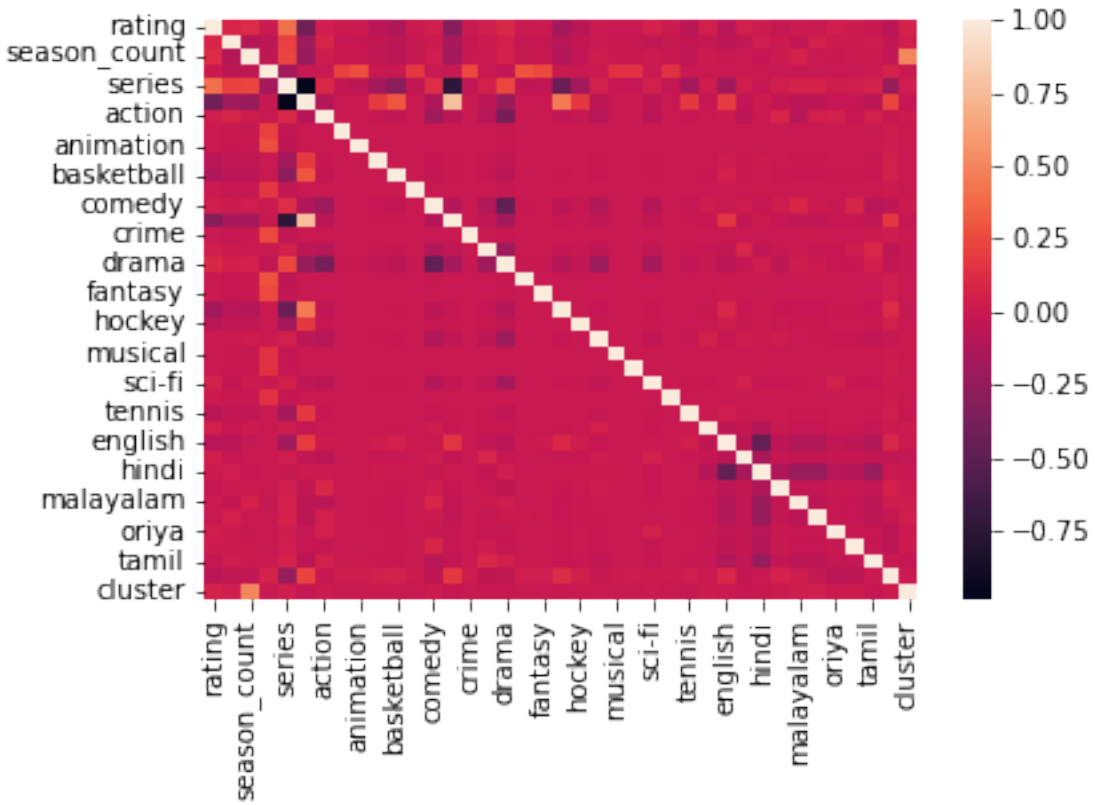
	gujarati	hindi	kannada	malayalam	marathi	oriya	\
rating	0.043447	0.010255	-0.020218	0.003703	-0.001591	0.036806	
episode_count	-0.024130	0.027815	0.006065	-0.032441	0.052725	0.015305	
season_count	-0.000656	-0.025685	0.008483	0.058530	-0.002844	-0.014117	
movies	0.007898	0.001234	-0.005878	-0.010332	-0.002019	-0.004026	
series	0.036865	-0.010291	0.032828	0.057706	0.056672	0.022486	
sports	-0.038993	0.010221	-0.032230	-0.056656	-0.057209	-0.022077	
action	-0.055526	0.007083	0.083152	-0.077699	0.046845	0.051714	
adventure	-0.001443	-0.004403	-0.001192	-0.002096	-0.002117	-0.000817	
animation	-0.001909	-0.001517	-0.001578	-0.002773	-0.002800	-0.001081	
badminton	-0.007293	0.012541	-0.006028	-0.010596	-0.010699	-0.004129	
basketball	-0.011640	0.006091	-0.009621	-0.016912	-0.017077	-0.006590	
biography	-0.001249	0.002768	-0.001033	-0.001815	-0.001833	-0.000707	
comedy	0.019875	-0.020093	0.018523	0.083130	0.001940	-0.011769	
cricket	-0.029694	0.004021	-0.024543	-0.043144	-0.043566	-0.016812	
crime	-0.001767	-0.003532	-0.001460	-0.002567	0.005858	-0.001000	
documentary	0.079049	-0.041649	-0.013060	0.002247	-0.017555	-0.019208	
drama	-0.017349	0.029137	-0.051618	0.017958	0.002972	-0.018719	
family	-0.002040	-0.003003	-0.001686	-0.002965	0.004325	-0.001155	
fantasy	-0.001767	-0.003532	-0.001460	-0.002567	-0.002592	-0.001000	
football	-0.017492	0.002086	-0.014458	-0.025416	-0.025664	-0.009904	
hockey	-0.006811	0.009930	-0.005630	-0.009896	-0.009993	-0.003856	
horror	0.017404	0.011135	0.020546	-0.003767	-0.006475	-0.019496	
musical	-0.001020	0.006559	-0.000843	-0.001482	-0.001497	-0.000578	
mystery	-0.001020	0.006559	-0.000843	-0.001482	-0.001497	-0.000578	
sci-fi	0.050821	-0.028062	-0.024474	-0.009327	-0.001169	0.051794	
sport	-0.001020	0.000111	-0.000843	-0.001482	-0.001497	-0.000578	
tennis	-0.006734	-0.000738	-0.005566	-0.009784	-0.009880	-0.003813	
bengali	-0.015391	-0.095621	-0.012721	-0.022362	-0.022581	-0.008714	
english	-0.075673	-0.470148	-0.062547	-0.109949	-0.111024	-0.042844	
gujarati	1.000000	-0.155528	-0.020691	-0.036372	-0.036727	-0.014173	
hindi	-0.155528	1.000000	-0.128552	-0.225976	-0.228184	-0.088055	
kannada	-0.020691	-0.128552	1.000000	-0.030063	-0.030357	-0.011715	
malayalam	-0.036372	-0.225976	-0.030063	1.000000	-0.053363	-0.020593	
marathi	-0.036727	-0.228184	-0.030357	-0.053363	1.000000	-0.020794	
oriya	-0.014173	-0.088055	-0.011715	-0.020593	-0.020794	1.000000	
punjabi	-0.015782	-0.098053	-0.013045	-0.022931	-0.023155	-0.008935	
tamil	-0.037600	-0.233607	-0.031078	-0.054632	-0.055165	-0.021288	
normal_duration	-0.031217	-0.000920	0.059154	0.023192	-0.015152	-0.071638	
cluster	-0.022794	-0.009523	0.008145	0.020708	0.034446	0.004390	

	punjabi	tamil	normal_duration	cluster
rating	0.010736	-0.007585	-0.069210	0.056126
episode_count	0.010415	0.019454	-0.037832	0.032503
season_count	0.003006	0.001797	-0.038589	0.497782
movies	-0.004483	0.001678	0.068853	-0.002352

```
series           0.025039  0.057307        -0.238329  0.025658
sports          -0.024583 -0.058569         0.228936 -0.025624
action          -0.016461  0.049707        -0.012892  0.013480
adventure       -0.000910 -0.002167         0.018724 -0.000477
animation       -0.001203  0.004794         0.017119 -0.000631
badminton       -0.004598 -0.010954         0.047451 -0.004847
basketball      -0.007338 -0.017483         0.053310 -0.005380
biography       -0.000788 -0.001877         0.016253 -0.000413
comedy           0.071072 -0.087373        -0.050685  0.009744
cricket         -0.018721 -0.044601         0.175467 -0.019583
crime           -0.001114  0.005621         0.018248 -0.000584
documentary      0.023778  0.100932        -0.050296 -0.011000
drama           -0.027997  0.046102        -0.038612 -0.006394
family          -0.001286 -0.003065         0.027347 -0.000675
fantasy         -0.001114 -0.002654         0.026120 -0.000584
football        -0.011028 -0.026274         0.111001 -0.012349
hockey          -0.004294 -0.010231         0.043073 -0.003817
horror          -0.021710 -0.037939        -0.013887  0.006512
musical         -0.000643 -0.001532         0.013533 -0.000337
mystery         -0.000643 -0.001532         0.021630 -0.000337
sci-fi          -0.018667 -0.024033         0.012909  0.008426
sport           -0.000643 -0.001532         0.009436 -0.000337
tennis          -0.004246 -0.010115         0.031127 -0.006446
bengali         -0.009703 -0.023117         0.005494  0.013977
english         -0.047708 -0.113662         0.086802  0.004853
gujarati        -0.015782 -0.037600        -0.031217 -0.022794
hindi           -0.098053 -0.233607        -0.000920 -0.009523
kannada         -0.013045 -0.031078         0.059154  0.008145
malayalam       -0.022931 -0.054632         0.023192  0.020708
marathi         -0.023155 -0.055165        -0.015152  0.034446
oriya           -0.008935 -0.021288        -0.071638  0.004390
punjabi          1.000000 -0.023705        -0.059538  0.013611
tamil           -0.023705  1.000000        -0.004603 -0.023092
normal_duration -0.059538 -0.004603         1.000000  0.003295
cluster          0.013611 -0.023092         0.003295  1.000000

[39 rows x 39 columns]
```

[55]:
```
sns.heatmap(df13.corr())
```

[55]:
```
<AxesSubplot:>
```



[56]:
```
corrMatrix=df13.corr().abs()
upperMatrix = corrMatrix.where(np.triu(np.ones(corrMatrix.shape), k=1).astype(np.
 ↪bool))

# Find index of feature columns with correlation greater than 0.95
corrFutures = [column for column in upperMatrix.columns if␣
 ↪any(upperMatrix[column] > 0.95)]

df13.drop(columns=corrFutures)
```

```
C:\Users\dell\AppData\Local\Temp\ipykernel_9756\3918593334.py:2:
DeprecationWarning: `np.bool` is a deprecated alias for the builtin `bool`. To
silence this warning, use `bool` by itself. Doing this will not modify any
behavior and is safe. If you specifically wanted the numpy scalar type, use
`np.bool_` here.
Deprecated in NumPy 1.20; for more details and guidance:
https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  upperMatrix = corrMatrix.where(np.triu(np.ones(corrMatrix.shape),
k=1).astype(np.bool))
```

[56]:

	rating	episode_count	season_count	movies	series	action	adventure	\
0	10	32	19	0	1	0	0	
1	4	21	15	0	1	0	0	
2	8	28	13	0	1	0	0	
4	2	29	10	0	1	1	0	
5	10	37	1	0	1	0	0	
...	...	...	...	...	...	...	...	
48640	6	15	6	0	1	0	0	
48641	4	14	1	0	1	0	0	
48642	6	5	33	0	1	0	0	
48643	5	1	9	0	1	0	0	
48644	8	4	2	0	1	0	0	

	animation	badminton	basketball	...	gujarati	hindi	kannada	\
0	0	0	0	...	0	0	0	
1	0	0	0	...	0	0	0	
2	0	0	0	...	0	0	0	
4	0	0	0	...	0	1	0	
5	0	0	0	...	0	1	0	
...	...	...	...	...	...	...	...	
48640	0	0	0	...	0	1	0	
48641	0	0	0	...	0	0	0	
48642	0	0	0	...	0	1	0	
48643	0	0	0	...	0	0	0	
48644	0	0	0	...	0	0	0	

	malayalam	marathi	oriya	punjabi	tamil	normal_duration	cluster
0	0	0	0	0	0	15.420940	2
1	0	0	0	0	0	14.914123	0
2	0	0	0	0	1	14.953344	0
4	0	0	0	0	0	15.112974	0
5	0	0	0	0	0	14.933925	2
...	...	...	...	...	...	...	...
48640	0	0	0	0	0	15.027452	0
48641	0	1	0	0	0	14.953344	1
48642	0	0	0	0	0	14.972392	3
48643	0	0	0	0	0	15.206792	1
48644	0	0	0	0	0	14.830741	1

[48113 rows x 38 columns]

7 Dropping highly correlated columns

[57]:
```
input = df13.drop("cluster", axis=1)
output = df13["cluster"]
```

8 Splitting column

[58]:
```
from sklearn.model_selection import train_test_split
```

[59]:
```
X_train, X_test, y_train, y_test = train_test_split(
    input, output, test_size=0.20, random_state=42)
```

9 Model building

[60]:
```
from sklearn.naive_bayes import CategoricalNB
from sklearn.model_selection import cross_val_score
```

[61]:
```
model = CategoricalNB()
```

[62]:
```
model.fit(X_train, y_train)
```

[62]:
```
CategoricalNB()
```

[63]:
```
model.predict(X_test)
```

[63]:
```
array([1, 1, 1, ..., 3, 1, 1])
```

[64]:
```
print(f"train Score:{model.score(X_train,y_train)}")
print(f"test Score:{model.score(X_test,y_test)}")
```

```
train Score:0.9797090153286568
test Score:0.9799438844435208
```

Results and Conclusion

The Naïve Bayes Partitioning model is very popular model where we cluster the data for generating recommendations and understanding the notion of similarity. Also, we come up the model whose accuracy on test data is 97.994% and error 2.03% on training data.

Future Work

We have already used Naïve Bayes Algorithm with K-means Clustering to build a model which have accuracy 97.994 %.

In part 2, We implement this model using different methods like by using Popularity model, Hybrid model, combined model etc. with different pre-processing method. Then analyse the difference created by these models and then compare the output created by these models. At last, we come up with the detailed analysis of these models.

References

[1] A M Fahim, A M Salem and F A Torkey "An efficient enhanced k-means clustering algorithm" Journal of Zhejiang University Science A vol. 10 pp. 1626-1633 July 2006

[2] K.A. Abdul Nazeer and M.P. Sebastian "Improving the Accuracy and Efficiency of the k-means Clustering Algorithm" Proceeding of the World Congress on Engineering vol. 1 2009

[3] Liu Ying. (2007) Analysis of the Application of Bayesian Method in Text Classification. Computer Knowledge and Technology, 4(22): 1074-1076

[4] https://python.org

[5] https://github.com

[6] https://stackoverflow.com