

ESCAPING THE ECHO CHAMBER IN MOVIES RECOMMENDER SYSTEM

A Thesis

presented to

the Faculty of Natural and Applied Sciences

at Notre Dame University-Louaize

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

ANTHONY EL HADDAD

JULY 2022

© COPYRIGHT

By

Anthony El Haddad

2022

All Rights Reserved

Notre Dame University - Louaize
Faculty of Natural and Applied Sciences
Department of Computer Science

We hereby approve the thesis of

Anthony El Haddad

Candidate for the degree of Master of Science in Computer Science


Dr. Hoda Maalouf

Supervisor, Chair


Dr. Maya Samaha

Committee Member

Acknowledgments

I would like to thank my advisor Dr. Hoda Maalouf, whose guidance was essential in helping me decide on this thesis topic and determine its specifics through every step of all stages of the project while remaining objective and avoiding offering explicit directions instead of helpful guidance. I appreciate her patience, dedication, level of organization, and timeliness in our meetings and in responding to my concerns.

I would also like to extend our gratitude to Notre Dame University's Faculty of Natural and Applied Sciences Department of Computer Science, which has presented the opportunity to work with its faculty and contribute to its research projects. I genuinely appreciate the prospect of presenting my thesis to the jury Dr. Maya Samaha.

Table of Contents

Acknowledgments	v
Table of Contents	vi
List of Figures	viii
List of Tables	x
List of Code Listings	Error! Bookmark not defined.
List of Abbreviations	Error! Bookmark not defined.
Abstract	xi

Chapter 1: Introduction and Problem Definition 1

1.1 Introduction to the General Problem	1
1.2 Problem Definition	3
1.3 Research Objectives	3
1.4 Approach and Main Results.....	4
1.5 Thesis Organization.....	4

Chapter 2: Background and Motivation 5

2.1 Definition of the Basic Concepts	5
2.2 Descriptions of the Used Algorithms	7
2.3 Previous Work in the Subject	7
2.4 Research Motivation.....	29

Chapter 3: Original Work..... 30

3.1 Introduction.....	30
-----------------------	----

Chapter 4: Conclusion 54

4.1 Main Contributions of the Thesis.....	54
4.2 Possible Extensions and Future Work.....	54

Bibliography..... 55

List of Figures

Figure 1: Tokenization.....	10
Figure 2: Stemming Results Example	10
Figure 3: TF-IDF selected top words	12
Figure 4: Classifier Performance concerning the number of Features	16
Figure 5: Example Curse of Dimensionality: Example Dataset	17
Figure 6: Example Curse of Dimensionality: Encoding.....	17
Figure 7: Example Curse of Dimensionality: Features Estimation.....	18
Figure 8: Example Curse of Dimensionality: Highly Dimensional Encoding	18
Figure 9: DBSCAN Datapoint types	20
Figure 10: Number of clusters effect on distortion	23
Figure 11: Jaccard Similarity Example	26
Figure 12:Jaccard Similarity Calculation	27
Figure 13: Single Linkage Nearest/Farthest Neighbor.....	27
Figure 14: Average Linkage All Neighbors and Mean Linkage Mean Neighbor.....	28
Figure 15: Collecting IMDb's IDs.....	32
Figure 16: Data Collection using OMDb	32
Figure 17: Collected Data.....	33
Figure 18: Year Feature to Categorical Variable	33
Figure 19: Runtime to Categorical Variable	34
Figure 20: All Genres Plot	36
Figure 21: Genre One-hot Encoding	36

Figure 22: Actors in the Dataset.....	37
Figure 23: Adding Actors to Dataset.....	37
Figure 24: Directors considered as Predictors	38
Figure 25: Writers in the Dataset	39
Figure 26: Languages as Predictors.....	40
Figure 27: Dataset Shape	40
Figure 28: Movies Distribution in Countries	41
Figure 29: Synonym List	41
Figure 30: Replacing Special Characters.....	42
Figure 31: Tokenization.....	42
Figure 32: Lemmatization and Stemming	43
Figure 33: Analyzer Function	44
Figure 34: List of Terms with Highest Frequency	44
Figure 35: Terms with Highest TF-IDF Scores	45
Figure 36: Data frame.....	46
Figure 37: Features Standardization.....	47
Figure 38: PCA Application	47
Figure 39: Features of the First Principal Component	48
Figure 40: Clustering with DBSCAN.....	49
Figure 41: K-Means Clustering.....	52
Figure 42: Cluster Linking Using Jaccard Similarity.....	52

List of Tables

Table 1: Basic Concepts Definition.....	5
Table 2: Steps 1 and 2 results.....	12
Table 3: Step 3: TF- Results	13
Table 4: Step 4: IDF Results	13
Table 5: Step 5-Vectorization	13
Table 6: Results Comparison	14
Table 7: Genre Mapper	53

Abstract

With the considerable rise of Internet users and the massive and diverse web searches, an excess of data has become available online. A recommendation model, also known as an engine, handles this massive amount of data available to find patterns that reflect user behaviors. Recommendation models have been implemented in several industries, and the most popular implementation is in the entertainment industry, specifically video streaming and on-demand platforms. There are several types of recommendation systems. In this paper, we have proposed a way to escape the loop created by the recommendation systems, where, the more “same genre” movie we watch, the more of that same genre the recommendation system will propose.

Keywords: filter bubble, escape the echo chambers, recommendation system, recommender system.

Chapter 1: Introduction and Problem Definition

The Internet has revolutionized the industry, and several sectors have noticed huge booms. One of the most notable sectors is video-on-demand streaming. The availability of numerous choices on these platforms makes it hard for the user to find what suits his/her interests and makes it also challenging for the recommendation system to suggest options that can satisfy the user, which will ensure proper customer retention that will reflect in additional profit and longer streaming time. The following section aims to introduce the latest trend in the entertainment industry, set a frame for the proposed work, and define the problem that will be addressed in this thesis.

1.1 Introduction to the General Problem

The Internet revolutionized several industries, impacting the entertainment industry immensely. The high demand for video-on-demand services pushed for services enabling users to access the content based on their preferences from anywhere. There has been a notable increase in video-on-demand services such as Netflix, Amazon Prime, and YouTube. An extensive catalog of movies and series are available to users through different subscription models. Technology and precisely data science created a revolution in the entertainment industry.

In the entertainment industry, specifically in video-on-demand streaming such as Netflix, the options available to the users are endless. Several categories, genres, duration,

actors, language, and plots choose the user harder. It is essential to highlight that a "good" movie entails a degree of subjectivity. While a movie might receive a five-star rating from one user, it might receive a one-star rating from another based on personal preferences. Regardless of the subjectivity highlighted, there are some general guidelines that one can look at to determine if the movie has the potential to achieve success or not.

Streaming companies perceive technological advancements and the data science field of study as tools to enrich and personalize their content to attract additional traffic—aiming to track users and their preferences to provide better streaming experiences. Movie streaming websites and applications have extensively used data science. For example, Netflix tracks data points such as the time and date as user watched a movie or series, age, gender, location, scenes users have viewed repeatedly, browsing and scrolling behavior, and other points related to the activities during streaming [1].

The popularity of mobile devices makes people's daily lives dependent on mobile services, making data access necessary for recommendation systems easier to secure. Recommendations systems have been applied in the movie and series industry and several areas, especially with the thriving of e-commerce [2]. Recommendation systems are applied to music, movies, news, webpages, and online shopping. Movie recommendation systems are a powerful tool for creating personalized user experiences [3]. The recommendations are also called suggestions to support users in coping with the information overload available on streaming platforms and accordingly find appropriate movies.

The availability and abundance of data available might result in intellectual isolation for users. When website algorithms selectively guess the information that a user is interested to see based on the user previous experience such as location, past-click history, and search

history. Consequently, users will be isolated from the information that does not align with their viewpoints and will be trapped in their own cultural and ideological bubbles. Filter bubbling has become increasingly popular with the vast availability of information on the web. It is important to suggest methods that allow users to have access to information while escaping the filter bubble.

1.2 Problem Definition

Movie recommendation tools are powerful to both the user and the entertainment platform. Using data science to analyze metrics and features related to users' usage trends might result in relevant observations that can be used to build marketing strategies, movie plots, and revenue generation streams. However, movie recommendation systems are considered to be comprehensive and complicated. Handling cumbersome and extensive data can result in inefficient models and a long convergence time. The proposed approach also aims to escape the filter bubble and echo chamber that occurs when cumbersome data is handled.

1.3 Research Objectives

The aim is to build on existing literature and to suggest an optimal method on which recommendation systems can be built to allow the users to have the most enjoyable experience. The proposed research includes assessing and exploring different methods for similarity measures and clustering and aims to highlight their benefits and disadvantages. Eventually, they will be used to present a recommendation system algorithm based on all factors considered in the analysis. The aim to present a method to cluster a data then assess

the links between the different movies and series clusters. These links will be weighted and accordingly can be used in presenting an effective recommendation system to the user based on their preferences and their taste which will allow him/her to escape the filter bubble.

1.4 Approach and Main Results

1.5 Thesis Organization

A background and literature review on the topic will be first presented to understand the problem at hand and related concepts. The background section is presented as means for the reader to understand the main concepts related to the topic before presenting the original work. Once some background concepts are presented, the reasoning behind the original work will be presented along with the detailed steps of the analysis. Once the targeted approach is developed, a conclusion and future work section will be presented to determine the results of this thesis and what further needs development.

Chapter 2: Background and Motivation

This area of research has been extensively studied and implemented. Numerous methods and approaches are tested to converge to a well-performing recommendation system algorithm. Thus, the following section highlights previous literature and presents an overview of the main basic concepts on which this thesis will be built.

2.1 Definition of the Basic Concepts

Several concepts should be considered when addressing our topic. It is essential to lay down the basis on which the literature review will be conducted, and the proposed thesis will be developed. A brief description of the main concepts is presented in the table below. Further details will be presented in the previous work in the subject section.

Table 1: Basic Concepts Definition

Basic Concept	Definition
IMDb dataset	IMDB is one of the popular databases and includes data related to video games, movies, directors, actors, casts, and TV programs. The traffic and establishment of IMDB as a global reference for all users interested in movies and TV shows made the IMDB dataset vast and comprehensive to include several movie genres, actors, languages, and other factors.
Clustering	Clustering is the task of dividing the population of a dataset into a number of groups such that data points in the same group have similar characteristics.

Tokenization	Tokenization is the process of exchanging sensitive data for non-sensitive data called "tokens" that can be used in a database or internal system without bringing it into scope.
K-means	K-means clustering is one of the simplest and popular unsupervised machine learning algorithms.
DBSCAN	Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a base algorithm for density-based clustering. It can discover clusters of different shapes and sizes from a large amount of data, which is containing noise and outliers.
Filter Bubble	When a website algorithm anticipates what information, a user would like to see based on information about the user, such as location, past click-behavior, and search history, it might create a filter bubble or ideological frame, a condition of intellectual isolation. As a result, users are effectively isolated in their own cultural or ideological bubbles and cut off from information that contradicts their beliefs.
PCA	Principal Component Analysis, or PCA, is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.

2.2 Descriptions of the Used Algorithms

2.3 Previous Work in the Subject

The design of recommendation engines depends on the industry and the characteristics of the available dataset [4]. The complexity and comprehensiveness of the task created a stream of several potential techniques to present accurate recommendations to users. These techniques include but are not limited to the content-based recommender system, collaborative filtering recommender systems, and hybrid recommender systems [3]. The collaborative filtering approach aims to create a model that learns from the user's past behavior and similar decisions made by other users and accordingly recommends items or ratings [4]. Collaborative filtering methods can be divided into neighborhood-based, memory-based, and model-based approaches [4]. In neighborhood-based collaborative filtering, a subset of users is chosen based on their similarity to the active user. Then, a weighted combination of their ratings is used to generate predictions for the designated use [4]. When the data scale is immense, conventional neighborhood-based collaborative algorithms do not scale well since it is complex to compute the search of similar users [5]. In item-based collaborative filtering, the aim is to match a user's rated items to similar items. This method leads to faster online systems and improved recommendations [4].

On the other hand, content-based filtering adopts a series of discrete characteristics of an item to recommend additional items with similar properties [2]. When content-based and collaborative filtering are combined, the result would be a hybrid recommender system. Hybrid approaches come as a means to leverage the strengths of content-based and collaborative recommendation systems.

There are several challenges when deploying recommendation systems as per [6] and [4]. Sparsity is considered a problem in Collaborative Filtering systems. Most users do not rate the items, which leads to a scarce user rating matrix which reduces the probability of finding users with similar ratings. In addition to sparsity, the cold-start problem with new items and users is considered a significant challenge recommender system. For example, in Collaborative Filtering, an item cannot be recommended unless some user has rated it before [6]. Finally, fraud is also jeopardizing the accuracy of recommendations. Due to the impact of recommender systems on sellers' and platforms' benefits, they are attempts to inflate the desirability of certain products (push attacks) or lower competitors' ratings (nuke attacks).

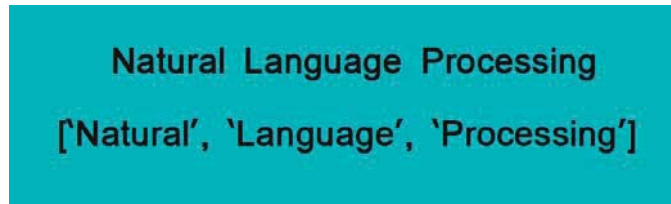
People's access to multiple online platforms for entertainment has created the availability of big data that can be used in several applications such as recommendation systems. Thus, several platforms created databases of movies and their related attributes, such as actors and directors [7]. IMDB is one of the popular databases and includes data related to video games, movies, directors, actors, casts, and TV programs. Through IMDB, users can gather data about movies and TV programs in addition to giving their votes for their favorite movie or actor [7]. Based on the users' input, IMDB rates the movies. The traffic and establishment of IMDB as a global reference for all users interested in movies and TV shows made the IMDB dataset vast and comprehensive to include several movie genres, actors, languages, and other factors.

The availability of a large dataset requires the availability of tools to analyze, extract, and process the data to deal with it quickly. Data analytics allows the optimal extraction of information from large, packed data sets. It is essential to clean the raw data [7] to make structured data easy for analysis.

It is crucial to cluster movies from large datasets to identify what movies form a natural cluster which can help and affect the recommendation system results [8]. These clusters can help the recommendation system to identify recurrent or repeating patterns. There are several clustering algorithms and techniques available that can support completing the job.

The dataset includes the top 250 movies based on the IMDb rating sourced directly from the IMDb's API, where the details of more than 5,000 movies are available. The movie's plot summaries were considered factors in the clustering; they were extracted from IMDb's website. Several factors are included in the raw dataset. Some of them are irrelevant to our analysis and only add additional unnecessary computing time. For example, the IMDb rating and ID are irrelevant in the analysis, while the runtime, plot, genre, actors, director, writer, year of release, country, and language are.

Since the movie's plot summaries were extracted from IMDB's website as features involved in the clustering, it is essential to extract meaningful words. It is essential to clean the raw data available to achieve the goal of including the plot summaries as features. Tokenization is a popular method to handle text data [9]. It aims to divide text documents into smaller units, such as single words, as shown in Figure 1: Tokenization. Each unit of the dissected text is called a token. For example, if tokenization is applied to the following sentence, "I am an outdoor enthusiast", the result would be ['I', 'am', 'an', 'outdoor', 'enthusiast']. Tokenization allows the analysis of the available text by interpreting the weight of different words based on the frequency of occurrence. There are several types of tokenization and including word tokenization and sentence tokenization. Through word tokenization, a sentence is split into words which are analyzed. Through sentence tokenization, a document or paragraph is split into analyzed sentences [9].



Natural Language Processing
['Natural', 'Language', 'Processing']

Figure 1: Tokenization

The aim is to present what dimensionality reduction techniques will be used while comparing them to other available methods and highlighting the reasons behind the clustering algorithm choice through the comparative analysis. The dimensionality reduction techniques considered are Principal Component Analysis (PCA), K-means, and DB-SCAN clustering to identify intrinsic clusters in the data.

Once tokenization is executed, it is essential to stem the available words. The aim is to convert a word form to its root. Stemming helps convert diverse forms of a word to their root form [10]. For example, words such as "fishing", "fished", and "fisher" are all derived from the word "fish" [11]. To understand stemming and tokenization, the following sentences will be assessed. Sentence 1 is "The audience witnesses the series of events that led to the tragic end," while sentence 2 is "Cyprus is considered an escape to many tourists that consider it a spot to witness the bliss of summer on the Mediterranean. Instead of building several dictionary entries for "witnesses" and "witness", stemming would result in only adding the word "wit". Several stemming algorithms include Porter, Lovin, Paice & Husk, Dawson, and Snowball. [11]. Snowball stemmer will be considered. To better portray the effect of stemming, the example below is considered in Figure 2: Stemming Results Example.

```
Without stemming: ['Today', 'May', 'is', 'his', 'only', 'daughter', "'s", 'wedding']  
After stemming:  ['today', 'may', 'is', 'his', 'onli', 'daughter', "'s", 'wed']
```

Figure 2: Stemming Results Example

As a first step, the summaries available are tokenized. The text is tokenized, and any punctuation and any other noise are removed. As a second step, all stop words are eliminated, especially those that did not provide critical information.

The term frequency-inverse document frequency (TF-IDF) was considered the basis for analyzing the plots [12]. TF-IDF measures word importance and is used to identify and highlight important and relevant features from an available text. It guarantees that words with an unusual tendency to appear in plots are selected as relevant words. TF measures the frequency in which a specific term recurs, while IDF measures the opposite, the scarcity of a term appearing in the plot text. The TD-IDF technique gives high weights to words that do not occur too frequently across all plots. In that way, a uniqueness indicator is given to movies; accordingly, movies with the same uniqueness indicator are clustered together. Evaluating the TF-IDF results shows how applicable a word to a sentence can be determined, how valuable a word is to a document, and how to identify misspelled words. For example, if in a document the word "coffee" was misspelled as "coffe", the words "coffee" and "coffe" might be processed as two separate words. In the case of TF-IDF, the mistake is corrected due to the IDF score since it will be determined that the word "coffee" is more important than "coffe" that will be treated as a non-useful word.

It is interesting to analyze the observations made on the selected top words. The TF-IDF resulted in selecting the most important words, as shown in Figure 1: TF-IDF selected top words. Some words are usual themes such as war, love, and friends. After determining and extracting the keywords, it is essential to encode them. All keywords for all movies are encoded. Each keyword is dedicated to a column. Then, for each movie representing a row in

the data frame matrix, a binary "0" is assigned if the movie does not contain the keyword in question, while a binary "1" is assigned if the movie contains the keyword.

```
top_words = ["young", "man", "help", "life", "war", "police", "family", "journey",
            "son", "boy", "world", "love", "save", "dark", "friends", "murder"]
```

Figure 3: TF-IDF selected top words

Thus, as a summary of what has been presented so far, the analysis to assess movies summaries includes the following steps:

1. Standardize, normalize (all lower case), and lemmatize data (all words to root words).
2. Tokenize words with frequency
3. Find TF for words
4. Find IDF for words
5. Vectorize the words

A simple example will be used to showcase the steps used to analyze movie summaries. The first document is "It is going to rain today". The second document is "Today, I am going outside. The third and last document is "I am going to watch the season premiere". Applying steps 1 and 2 to the data results in the following as shown in Table 2: Steps 1 and 2 results.

Table 2: Steps 1 and 2 results

Word	Count
going	3
to	2
today	2
i	2
am	2
it	1
is	1
rain	1

Applying step aims to compute the term frequency (TF) computed as the ratio between the number of repetitions of a word in a document and the total number of words in the document. Results are summarized in Table 3: Step 3: TF- Results.

Table 3: Step 3: TF- Results

Words/Documents	Document 1	Document 2	Document 3
going	0.16	0.16	0.12
to	0.16	0	0.12
today	0.16	0.16	0
i	0	0.16	0.12
am	0	0.16	0.12
it	0.16	0	0
is	0.16	0	0
rain	0.16	0	0

The fourth step of the analysis consists of finding the IDF ratio, which is computed as follows $IDF = \log \left[\frac{\text{Number of documents}}{\text{Number of documents containing the word}} \right]$. Results are summarized in Table 4: Step 4: IDF Results

Table 4: Step 4: IDF Results

Words	IDF Values
going	$\log(3/3)$
to	$\log(3/2)$
today	$\log(3/2)$
i	$\log(3/2)$
am	$\log(3/2)$
it	$\log(3/1)$
is	$\log(3/1)$
rain	$\log(3/1)$

The fifth step is to vectorize the different inputs and variables by building a model, as shown in Table 5: Step 5-Vectorization

Table 5: Step 5-Vectorization

Words	IDF Values	Words/Documents	Document 1	Document 2	Document 3
going	0	going	0.16	0.16	0.12
to	0.41	to	0.16	0	0.12
today	0.41	today	0.16	0.16	0
i	0.41	i	0	0.16	0.12
am	0.41	am	0	0.16	0.12

it	1.09	it	0.16	0	0
is	1.09	is	0.16	0	0
rain	1.09	rain	0.16	0	0

As for the final step, it is essential to compare the results as shown in Table 6: Results Comparison and conclude accordingly.

Table 6: Results Comparison

Words/Documents	going	to	today	i	am	it	is	rain
Document 1	0	0.07	0.07	0	0	0.17	0.17	0.17
Document 2	0	0	0.07	0.07	0.07	0	0	0
Document 3	0	0.05	0	0.05	0.05	0	0	0

Analyzing the results allows the discovery of similarities and differences between documents and words. It can be easily observed that the words "it," "is," and "rain" are essential to document 1 but not relevant to documents 2 and 3, which might result in the conclusion that documents 1, 2, and 3 address different perspectives and meanings when tackling the topic of rain. In addition to that, it can be observed that documents 1 and 2 discuss the event in the present based on the importance of "today".

It can be tiring to handle large datasets; thus, it is crucial to address the data dimensionality. PCA reduces the dimension while preserving the diversity between data points and reducing an extensive list of dimensions to a list of valuable and relevant features. Reducing the features helps present the data with a lighter computational load, reducing dimensional redundancy, and getting higher Effective Distance Metrics values, thus the importance of addressing the curse of dimensionality.

According to the Hughes phenomenon [12], when the number of samples is fixed while the number of dimensions increases, a model's predictive power first increases, but after a certain point, the predictive power decreases. To understand the effect of addressing data frame dimensionality, an example will be presented in which a set of images representing a cat or a dog will be considered. The aim is to create a classifier that can distinguish two categories: dogs and cats. Attributes should be considered to distinguish the objects apart. A potential attribute could be the animal's color. Based on that attribute, the classifier will map the data to a specific category. The classifier translates the calculation by the average red, green, and blue of the animal's picture. The mathematical model aims to translate the animal's color into numbers. The three colors to which correspond numbers consist of three distinct features, yet they are not enough to determine an accurate classification of the picture. Thus, additional features are needed, such as the image texture determined by getting the average edge or gradient intensity in the X and Y direction. After combining the colors and texture features, we obtained five distinct features that the classifier could use to distinguish cats from dogs.

One might think that adding additional features might make the differentiation process more straightforward and might decrease the margin of error. However, after a specific threshold, increasing the problem's dimensionality by adding additional features would result in degrading the classification algorithm performance, as shown in Figure 4: Classifier Performance concerning the number of Features.

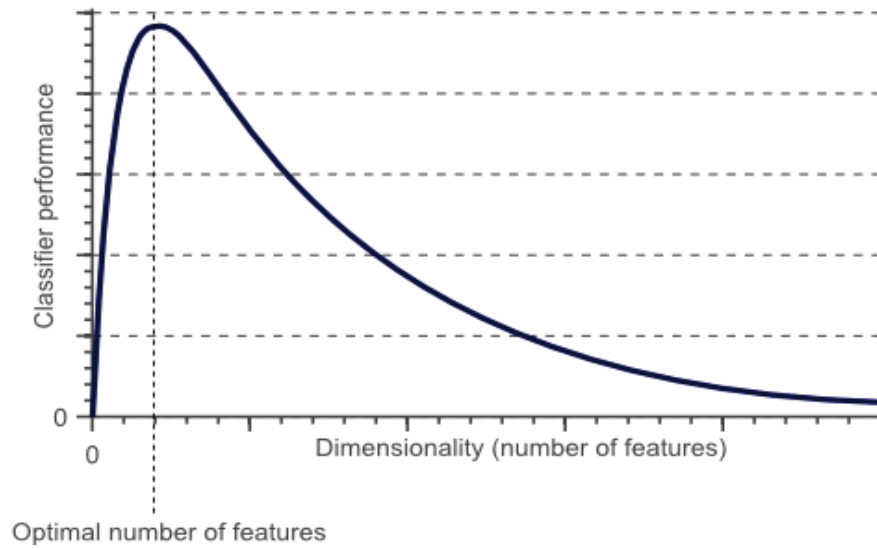


Figure 4: Classifier Performance concerning the number of Features

Having more data is helpful. However, too much information might sometimes result in an extended model training time, and the curse of dimensionality rises. Other potential issues that may arise from having a vast number of dimensions occur in the case of having more features than observations, leading to the risk of massively lifting the model considered. Moreover, another issue would be the difficulty of clustering observations; too many dimensions cause every observation to appear equidistant from all others [13]. To highlight the curse of high dimensional data will be elaborated in the following example. First, a dataset consisting of 8 candies is considered, as shown in Figure 5: Example Curse of Dimensionality: Example Dataset [13]. It can be observed that there are two clusters within the dataset: spicy and sweet. Based on the clustering, it can be observed that a bluish candy is sweet while the reddish candy is spicy. The two features considered for the clustering resulted in perfect clusters of taste, which means a perfect mod For the machine, the data should be presented properly as suggested in Figure 6: Example Curse of Dimensionality: Encoding.

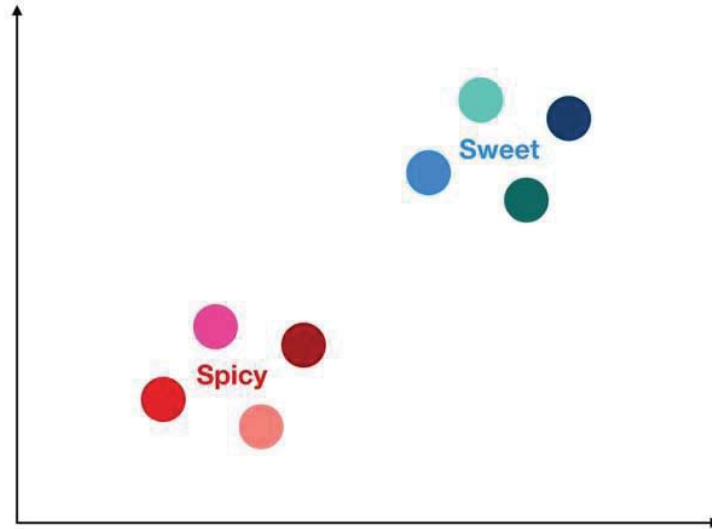


Figure 5: Example Curse of Dimensionality: Example Dataset

	Reddish	Bluish
	1	0
	1	0
	1	0
	1	0
	0	1
	0	1
	0	1
	0	1

Figure 6: Example Curse of Dimensionality: Encoding

However, if the data is more highly dimensional, as shown in Figure 8: Example Curse of Dimensionality: Highly Dimensional Encoding, there will be eight instead of two categories. Thus, dimensionality reduction could be adopted to reduce complexity. In the following example, the aim is not to explain a specific algorithm but rather, and it is a simple example that demonstrates some of the principles that dimensionality reduction algorithms follow. The first step in the algorithm is to locate the underlying trends, also called latent

features, in the available features, such as the primary colors red and blue. In the following steps, the algorithm estimates each feature in terms of its exposure to the latent features and plotted as shown in Figure 7: Example Curse of Dimensionality: Features Estimation. It was possible to produce two clusters using latent features such as red and blue. Then, the number of candies is used within each cluster as predictors.

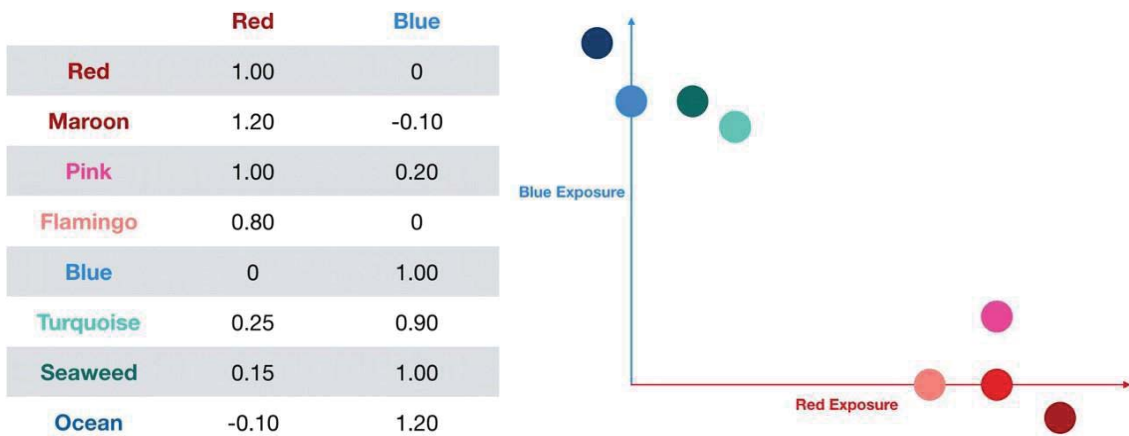


Figure 7: Example Curse of Dimensionality: Features Estimation

	Red	Maroon	Pink	Flamingo	Blue	Turquoise	Seaweed	Ocean
Red	1	0	0	0	0	0	0	0
Maroon	0	1	0	0	0	0	0	0
Pink	0	0	1	0	0	0	0	0
Flamingo	0	0	0	1	0	0	0	0
Blue	0	0	0	0	1	0	0	0
Turquoise	0	0	0	0	0	1	0	0
Seaweed	0	0	0	0	0	0	1	0
Ocean	0	0	0	0	0	0	0	1

Figure 8: Example Curse of Dimensionality: Highly Dimensional Encoding

Given a new candy, its color should be recorded. The color should be reflected concerning the latent features. Using the latent features exposures, it can be determined to what cluster the candy belongs to using simple techniques such as the Euclidean distance [13]. If the candy is

associated with the red cluster, it can be deduced that it is spicy. If the candy is associated with the blue cluster, it can be deduced that it is sweet. Thus, the model could be as follows in the example proposed to showcase the curse of dimensionality.

Various distance measurement techniques are used to calculate similarities between data points. These techniques include and are not limited to affinity propagation (graph distances), mean-shift (point distances), K-Means (distance between points), DBSCAN (distance between nearest points), Gaussian mixtures (Mahalanobis distance to centers), and spectral clustering (graph distance) [14]. Regardless of the numerous clustering techniques available, it should be highlighted that there is no one best algorithm for every purpose. For example, DBSCAN might be very accurate but can also be inefficient in other cases. DBSCAN might be more convenient to assess human intuitions that KMeans is [10]; thus, clustering is essential to address the curse of dimensionality.

Density-based clustering helps in identifying different clusters within a data frame. The rationale behind density-based clustering methods is that clusters in a dataset are adjoint regions of high density separated by similar regions of low density. Density-based Spatial Clustering of Applications with Noise (DBSCAN) is a density-based clustering algorithm [14]. The dataset available includes outliers and noise. Thus, the aim is to find different clusters of different shapes and sizes. In the DBSCAN algorithm, there are two parameters to consider. The first is MinPts, a threshold used to find if a region is dense when a minimum number of points are clustered together. The second parameter, Eps (ϵ), is used to locate points adjacent to a given point. Understanding the concepts of density reachability and density connectivity is crucial to recognizing the importance of the features MinPts and ϵ [15].

Reachability refers to reaching a point from another just by being at a distance ϵ from it. On the other hand, connectivity is finding if two points are located within the same cluster. Points a and e are connected if $a > b > c > d > e$. DBSCAN clustering results in three different types of data points, as shown in Figure 9: DBSCAN Datapoint types. A circle of radius ϵ surrounds each data point. The point is classified into the core, border, and noise points. If a circle around a point contains at least the 'MinPts' number of points, it is considered a core point. If several points are less than "MinPts", it is classified as Border Point. Finally, a point is classified as noise if no data points are around within an ϵ radius.

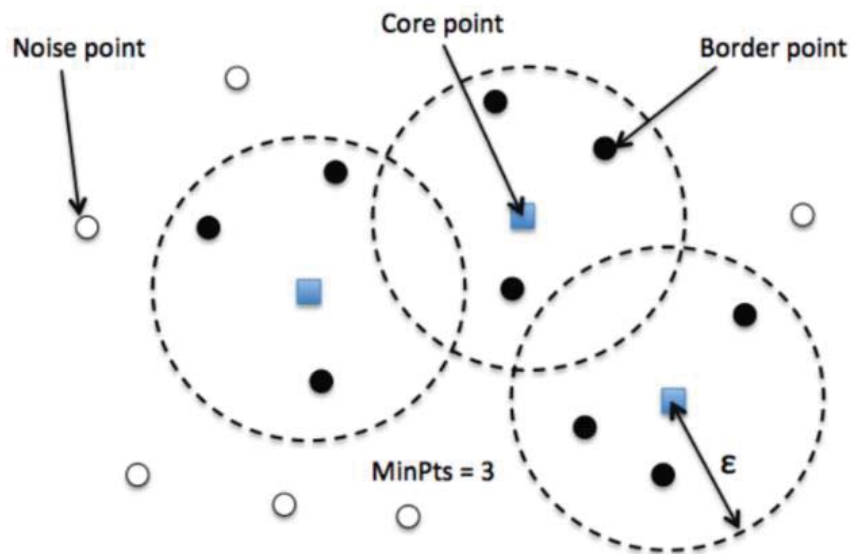


Figure 9: DBSCAN Datapoint types

Another clustering approach is K-Means clustering. A cluster will be formed between any two observations in a vector space, no matter how far the distance between them is. Every data point is vital since the average value of cluster elements determines the clustering. A slight variation in the data will affect the clustering outcome. However, this problem is significantly reduced by DBSCAN's logical cluster formation unless an odd shape data is

encountered, which is not usually a problem. Another challenge with K-Means is to specify before clustering the number of clusters we are interested in getting [16]. Usually, we will not determine a reasonable value of k beforehand. Thus, an iterative clustering algorithm is needed based on which K-Means aims to find a local maximum during each iteration. K-Means clustering is a standard algorithm in data science and machine learning. K-Means can be implemented based on the following logic:

1. Analyze the data quickly to determine how many classes to use. To use several classes/groups, the respective center points should be randomly selected.
2. Each data point is classified by finding the distance between it and the nearest group center. Then, the point is classified as belonging to the nearest group.
3. Based on this classification, the group center is calculated by taking the mean of all the vectors in the group.
4. This process should be repeated for several iterations until the center of the groups does not differ much between them. The group centers can also be initialized a few times randomly, and then the run that appears to have provided the best results is selected.

The K-Means method is fast since only distances are computed between points and group centers. Its complexity is linear and denoted by $O(n)$. K-Means clustering can handle an extensive data set since it has a linear complexity compared to other clustering types that could be quadratic $O(n^2)$. Moreover, K-Means works well when the clusters are hyper-spherical such as circles in 2D and spheres in 3D. Although the K-Means method is fast and typical, it has several disadvantages. Selecting how many classes are needed before the

analysis can be challenging. Thus, it is a disadvantage since it is difficult or the clustering algorithm to figure out the number of clusters. The K-Means also randomly chooses cluster centers, which might result in different clustering outcomes after each iteration [17].

Since determining the number of clusters prior to the analysis is a disadvantage to K-Means and since it is crucial to choose the correct number of clusters, there are several ways to determine K, which is the number of clusters. Three standard methods are field knowledge, business decision, and elbow method.

The elbow method can determine the number of clusters in a dataset. It works by plotting the increasing values of K against the total error obtained when using the specific value K [18]. The aim is to find the value of K that for each cluster will not significantly raise the variance. The percentage variance is the ratio between the variance between the group concerning the total variance. The variance represents the error. The error can be calculated based on the following steps:

1. Subtracting the Euclidean distance from each point of each cluster to the center of the respective group
2. The obtained result or value is squared to eliminate the negative terms
3. Finally, all squared values are added to obtain the total error.

Figure 10: Number of clusters effect on distortion represents an example that showcases how the distortion varies with the number of clusters

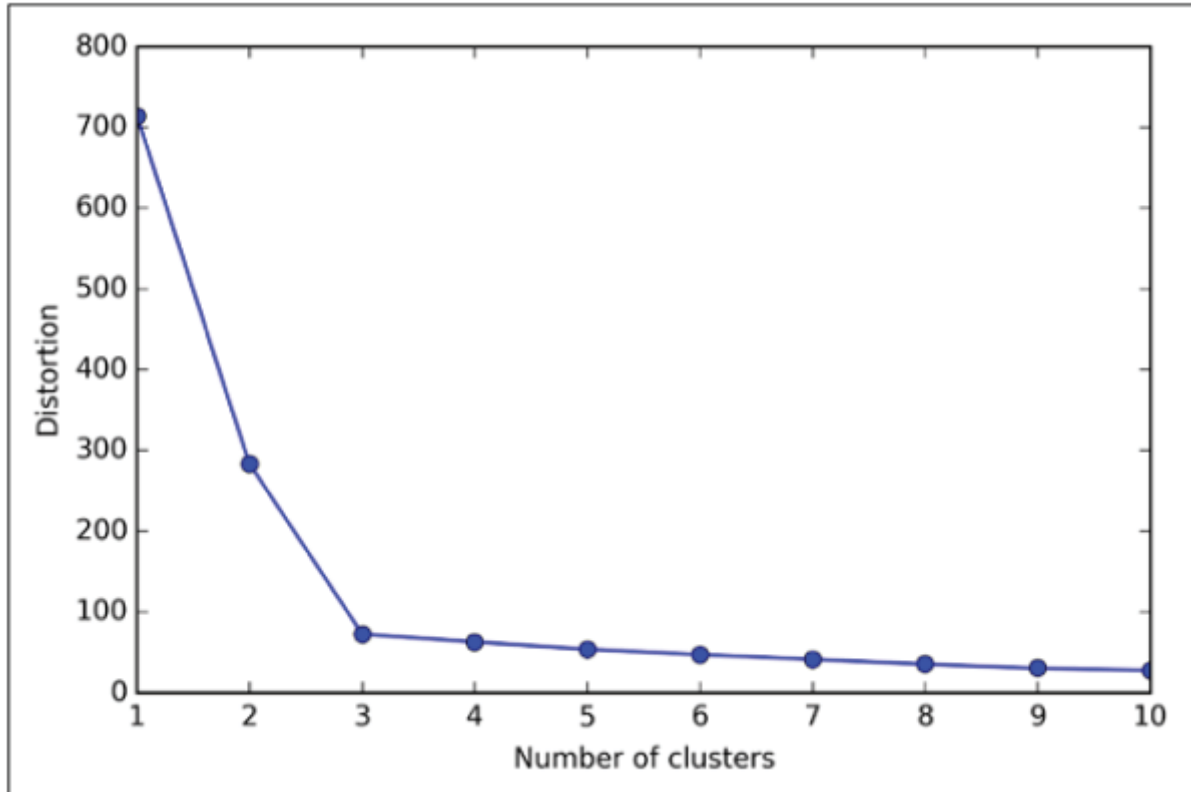


Figure 10: Number of clusters effect on distortion

Based on Figure 10: Number of clusters effect on distortion, $K=3$ will be selected; it is where the elbow is located. Thus, the elbow method could determine the convenient number of clusters in the dataset. The possible values of K range between 1 and the total number of elements in the data frame considered. There are two extreme cases for the possible values of K . If K is 1, then there is only one group to which all points belong. If $K =$ all data points, it can be concluded that each data point belongs to a separate group.

Now that both DBSCAN and K-Means have been introduced, it is essential to highlight why some scientists adopt DBSCAN instead of K-Means clustering, which is widely used, familiar, and straightforward. DBSCAN does not need to specify the number of clusters, as with K-Means. When the dataset includes several distributions, DBSCAN can

generate accurate results. With DBSCAN, a distance calculation function is needed in addition to guidance on how much distance is considered "close".

As highlighted so far, clustering, also called cluster analysis, is an unsupervised learning approach. Clustering separates data points into groups. The aim is to have data points of similar features included in the same group. Accordingly, data points that belong to different groups have different properties. To conduct clustering, and no matter the clustering method, the aim is to calculate similarities that will be used to group data points.

Principal component analysis (PCA) is considered one of the most popular multivariate statistical techniques. It can be compared to writing a book summary. It transforms high-dimensions data into lower-dimensions while conserving as much information as possible [13]. It has been applied in almost all scientific disciplines. It has been used in typical applications such as image processing genome research in which data included thousands of columns [13]. PCA is a technique used to analyze a data table representing observations derived from several inter-correlated variables. The PCA method's outcome is extracting the critical information from the raw data and building a set of new orthogonal variables called principal components obtained as linear combinations of the original variables [19]. The goals of PCA can be summarized in four points: extracting the critical information from the data table, compressing the size of the data set, simplifying the description of the data set, and analyzing the structure of the observations and the variables[20]. The components of the PCA are computed as follows: the first principal component is required to have the most considerable variance, and the remaining components are computed as being orthogonal to the previous component while having the most significant possible inertia [19].

Principal Component Analysis achieves dimensionality reduction based on the following steps [21]. In the first step, the data is standardized. Usually, variables that makeup sets have different units and means, leading to extremely heavy computation. A more efficient approach would be to center the data at mean zero and make it unitless. The second step is to find the covariance matrix, a symmetric matrix with the size of the data number of dimensions. The covariance shows how the features diverge from each other by computing the covariance pairwise. The third step is to find the eigenvalues and eigenvectors of the covariance matrix. Eigenvectors do not change direction when matrix transformation is applied; they are linearly independent. Eigenvalues indicate the magnitude of the Eigenvector. In the covariance matrix, the eigenvectors point toward the most considerable variance. The Eigenvector with the largest Eigenvalue refers to the first principal component. The first principal component represents the most variance. Since the principal components are combinations that ensure that the information does not overlap between features, they eliminate information redundancy. The variance reduces with every new principal component, and further reduction can be achieved in the fourth step by eliminating the minor critical principal components. In the fourth step, it should be carefully decided how much information loss is tolerated.

After clustering, it is crucial to assess the distances between clusters. To find the similarity between two examples, all the features' data for those two examples should be combined into a single numeric value—for example, a shoe data set with only one feature: shoe size. The difference between the two shoes' sizes should be calculated to quantify their similarities. The smaller the numerical difference between sizes, the greater the shoe similarity. The following approach is a simple, manual similarity measure [14]. If the sample

representing the shoes is more complex and includes an additional feature that is the color, the harder it will be to combine it with the numerical size data since the color is categorical data. Categorical data can be either single-valued such as a car's color that can be white or blue but never both, or multi-valued (multivalent), such as a movie's genre that can be action and comedy simultaneously or just one genre [22]. The manual similarity approach becomes more challenging since the dataset becomes harder. In that case, a supervised similarity measure should be adopted.

For univalent data matches, the similarity is represented by a binary "1"; otherwise, it is a binary "0". Multivariant data is harder to deal with. In that case, similarity can be calculated using the ratio of common values called the Jaccard similarity [22]. The Jaccard similarity can be further examined through the following example, as shown in Figure 11: Jaccard Similarity Example. The Jaccard index, also called the Jaccard similarity coefficient, is a statistical measure used to find the diversity and similarity between sample sets. The Jaccard coefficient is also known as the Tanimoto index or coefficient. The Jaccard coefficient measures the similarity between finite sample sets, and it is defined as the intersection between them divided by the size of the union of both sets, as shown in Figure 12: Jaccard Similarity Calculation.

- ["comedy", "action"] and ["comedy", "action"] = 1
- ["comedy", "action"] and ["action"] = $\frac{1}{2}$
- ["comedy", "action"] and ["action", "drama"] = $\frac{1}{2}$
- ["comedy", "action"] and ["non-fiction", "biographical"] = 0

Figure 11: Jaccard Similarity Example

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

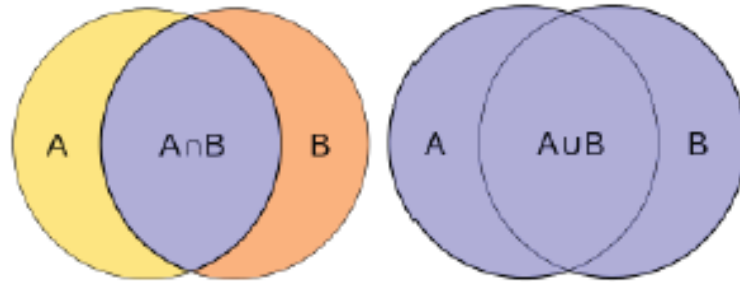


Figure 12: Jaccard Similarity Calculation

Another approach to replace the manual similarity measure is finding the distance between the clusters. The distance between the clusters will reflect the similarity between them. One possible approach to adopt is to find the closest two points in the two clusters, and it will be considered as a measure of similarity called the nearest neighbor method. Another approach would be to find the farthest two points in the cluster, called the furthest neighbor. The illustration in neighbor highlights both concepts.

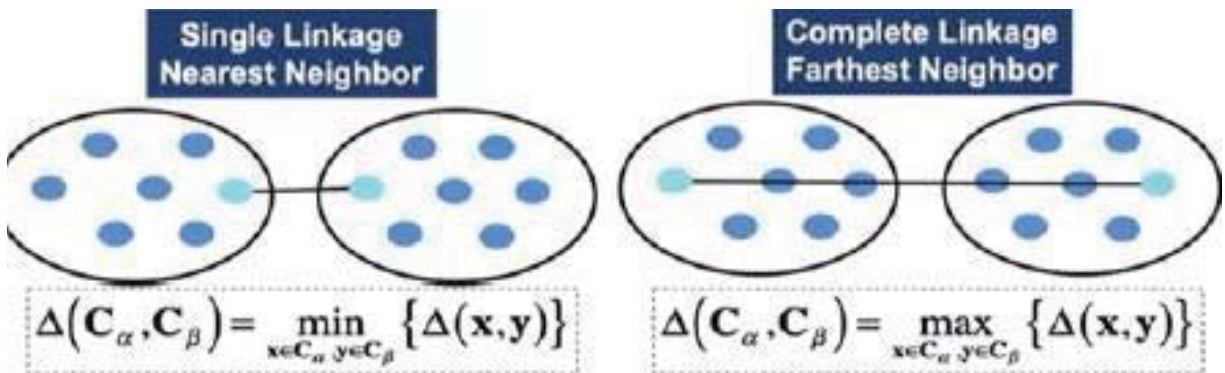


Figure 13: Single Linkage Nearest/Farthest Neighbor

An alternative approach is to find the average length from a point in one cluster to all other points in the other cluster, as shown in Figure 14: Average Linkage All Neighbors and Mean Linkage Mean Neighbor. The fourth and final approach would be to find the distance between the mean of one cluster and the other, considered in the similarity analysis as shown in Figure 14: Average Linkage All Neighbors and Mean Linkage Mean Neighbor.

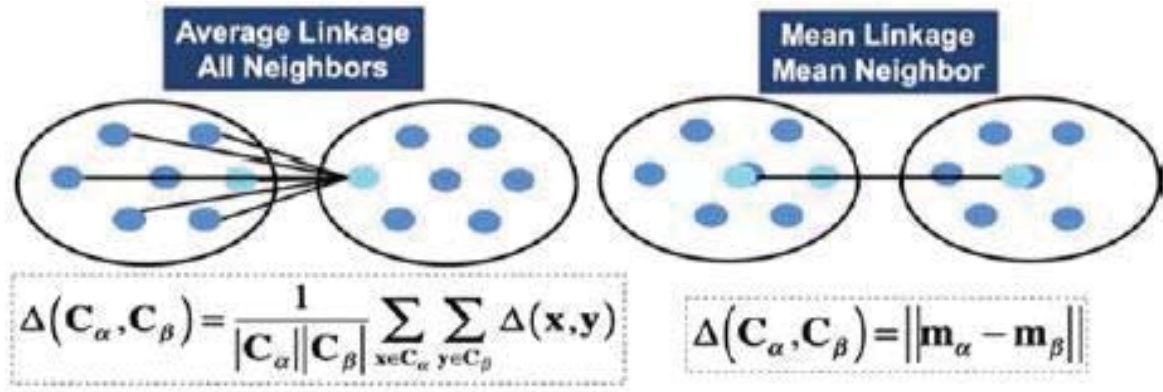


Figure 14: Average Linkage All Neighbors and Mean Linkage Mean Neighbor

Echo chambers are a social phenomenon where the filter bubbles of interacting individuals strongly overlap. Social media may limit the exposure to diverse perspectives and favor the formation of groups of like-minded users framing and reinforcing a shared narrative, that is, echo chambers [23]. The dangers of a society falling apart into distinct echo chambers can be described as a lack of society-wide consensus and a lack of at least some shared beliefs among otherwise disagreeing people that are needed for processes of democratic decision-making [24]. In a wide range of areas, including music, films, dating, commerce, and advertising, recommendation systems have an impact on how information is presented to people. On the one hand, the nearly universal practice of selecting content based on expected interests enables people to navigate the digital information glut. Models that have been trained to anticipate user preferences as accurately as possible are frequently used by recommender

systems. These models decide the content and information accessible to various users when the technologies are installed. The difference between these goals creates a chance for unexpected consequences, which can lead to phenomena like filter bubbles and polarization [23]. Preference predictions can help surface relevant and engaging content by taking use of user consumption or rating trends. On the other hand, this individualized curation has the ability to polarize and divide society. When used in feedback to form recommendations, the exploited patterns between users may actually encode harmful biases that reinforce one another.

Potential pitfalls occur when putting large scale machine learning-based systems in feedback with people, and highlights the importance of creating analytical tools to anticipate and prevent undesirable behavior. Such tools should seek to quantify the degree to which a recommender system will meet the information needs of its users or of society as a whole, where these “information needs” must be carefully defined to include notions like relevance, coverage, and diversity. In our proposed work, the proposed approach will address these potential pitfalls that result from bubble filtering.

2.4 Research Motivation

Chapter 3: Original Work

In the following section, and based on the extensive overview that has been provided in Chapter 2, the original work related to the thesis will be presented.

3.1 Introduction

- The student's original work could cover 1 or several chapters as required. It is preferable to put analytical and simulation results in separate chapters.
- Simulation results should be given in the form of curves and not tables filled with numerical values.
- Simulation and numerical results should be explained clearly and not left to open interpretation.

Listing 3.1 MyListingName

```
1.  class SimpleMath
2.  {
3.      public int AddTwoNumbers(int number1, int number2)
4.      {
5.          return number1 + number2;
6.      }
7.  }
```

Listing 3.2 MyListingName

```
1.  class SimpleMath
2.  {
3.      public int AddTwoNumbers(int number1, int number2)
4.      {
5.          return number1 + number2;
6.      }
7.  }
```

The dataset is based on IMDb's web API: RESTful web service. BeautifulSoup Library in Python was used to obtain plot summaries of movies. The final extracted data frame has rows and columns based on OMDb's web API and the movie plot summaries. The inputs of the data frame were all definite features. It is essential, as discussed previously, to apply clustering to the available data frame called the dataset. The dimensionality reduction techniques considered are Principal Component Analysis (PCA), K-means, and DB-SCAN clustering to identify intrinsic clusters in the data.

When it comes to analyzing and assessing the remaining features that include the language, actor, country, director, and other components, the top 20 occurrences were identified. The selection is based on frequency. Based on the top occurrences of each feature, additional columns are added to the matrix developed from the raw data set. Once added, each movie's features are encoded. Based on the approach explained, the data frame is constructed with 250 data points representing the 250 top IMDb movies and 112 columns representing all features. Once the dataset is built, addressing the data frame dimensionality is essential. Several dimensions make it complicated to acquire a fully functional model. There is a lack of statistical and standard computational techniques when handling large data sets. The Principal Component Analysis (PCA) will be suggested as a method to reduce all additional irrelevant features.

The first part of the analysis is to collect all the IMDb IDs of the top 250 movies. The latter task has been completed by scrapping IMDb's webpage, as shown in Figure 15: Collecting IMDb's IDs.

```
In [2]: url='http://www.imdb.com/chart/top'
page=get(url).content
soup=BeautifulSoup(page, 'html.parser')
class_=soup.find_all(name='div', attrs={'class': 'wlb_ribbon'})
movie_ids=[c['data-tconst'] for c in class_]
```

Figure 15: Collecting IMDb's IDs

The movie IDs are a collection of 250 movies per IMDb's website. OMDb will explicitly use OMDb's API to collect the data, as shown in Figure 16: Data Collection using OMDb.

```
In [3]: movie_info=[[[] for i in range(len(movie_ids))]

for i in range(250):
    url='http://www.omdbapi.com/?i='
    r=requests.get(url+movie_ids[i]+"&apikey=de12b217").json()
    for a in r.keys():
        movie_info[i].append(r[a])

df_omdb=pd.DataFrame(movie_info, columns=r.keys())

In [4]: df_omdb.head()
```

Figure 16: Data Collection using OMDb

Once all the information from the OMDb database is collected, it is essential to enrich the available data content by getting the IMDb database. A sample of the extracted data format is shown in Figure 17: Collected Data.

Out[4]:

	Title	Year	Rated	Released	Runtime	Genre	Director	Writer	Actors	Plot	Language	Country	Awards	
0	The Shawshank Redemption	1994	R	14 Oct 1994	142 min	Drama	Frank Darabont	Stephen King, Frank Darabont	Tim Robbins, Morgan Freeman, Bob Gunton	Two imprisoned men bond over a number of years...	English	United States	Nominated for 7 Oscars. 21 wins & 43 nominatio...	amazon.com/images/
1	The Godfather	1972	R	24 Mar 1972	175 min	Crime, Drama	Francis Ford Coppola	Mario Puzo, Francis Ford Coppola	Marlon Brando, Al Pacino, James Caan	The aging patriarch of an organized crime dyna...	English, Italian, Latin	United States	Won 3 Oscars. 31 wins & 30 nominations total	amazon.com/images/
2	The Godfather: Part II	1974	R	18 Dec 1974	202 min	Crime, Drama	Francis Ford Coppola	Francis Ford Coppola, Mario Puzo	Al Pacino, Robert De Niro, Robert Duvall	The early life and career of Vito Corleone in ...	English, Italian, Spanish, Latin, Sicilian	United States	Won 6 Oscars. 17 wins & 20 nominations total	amazon.com/images/M/
3	The Dark Knight	2008	PG-13	18 Jul 2008	152 min	Action, Crime, Drama	Christopher Nolan	Jonathan Nolan, Christopher Nolan, David S. Goyer	Christian Bale, Heath Ledger, Aaron Eckhart	When the menace known as the Joker wreaks havo...	English, Mandarin	United States, United Kingdom	Won 2 Oscars. 159 wins & 163 nominations total	amazon.com/images/

Figure 17: Collected Data

Once the data is extracted, it is crucial to manipulate, clean, and organize the available data. There are several steps in the data cleaning process. The first step is to convert the attribute "Year" to a categorical variable. The year 1990 is considered a convenient cutoff for the data available. Movies released before 1990 are represented by a binary "0" while movies released after 1990 are represented by a binary "1". One-hot encoding is performed, and a dummy variable is added to represent the feature Year (0, 1) in the data frame as shown in Figure 18: Year Feature to Categorical Variable

```
In [8]: df_omdb.Year=pd.to_numeric(df_omdb.Year)
for i in range(250):
    if df_omdb.Year[i]<1990:
        df_omdb.Year[i]=0
    else:
        df_omdb.Year[i]=1
dummy_year=pd.get_dummies(df_omdb.Year)

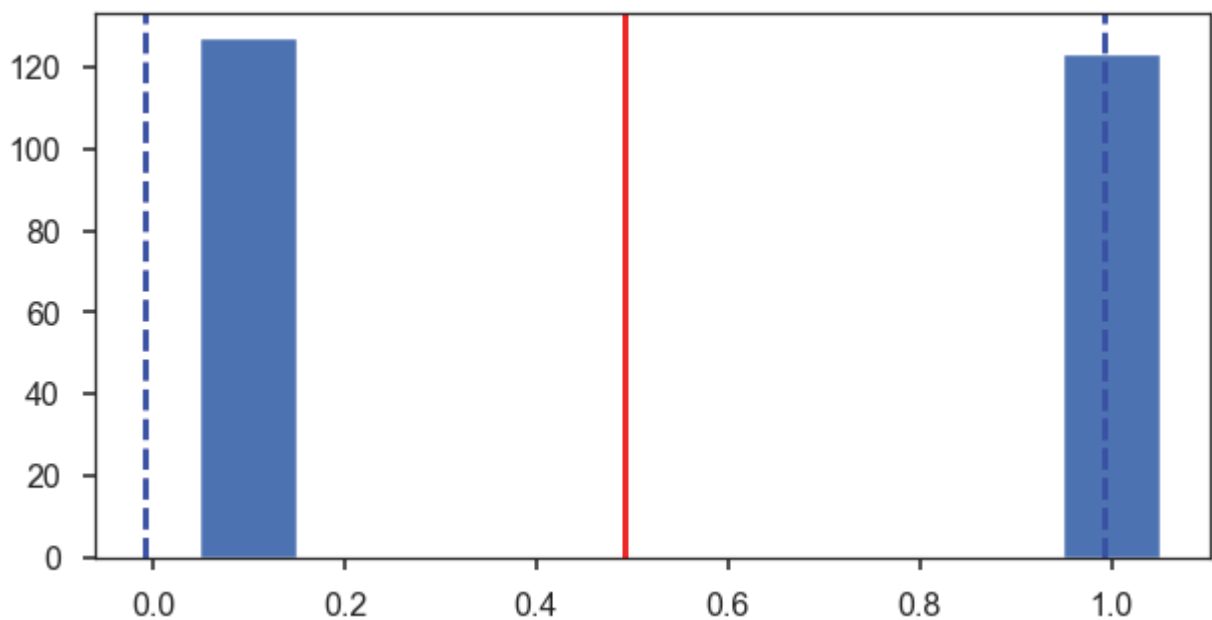
for i in range(250):
    df_omdb.Runtime[i]=df_omdb.Runtime[i].split()[0]
```

Figure 18: Year Feature to Categorical Variable

After converting the year feature to a categorical variable, it is essential to convert runtime too. A Runtime of 125 minutes, the median value of all data points, will be

considered a convenient cutoff as shown in **Error! Reference source not found.** Movies with a runtime of fewer than 125 minutes are represented by a binary "0," while movies with a runtime exceeding 125 minutes are represented by a binary "1".

```
In [9]: fig,ax=plt.subplots(1,1,figsize=(10,5))
ax.hist(df_omdb['Runtime'],edgecolor='white',align='right')
ax.axvline(x=np.mean(df_omdb['Runtime']),c='r')
ax.axvline(x=np.mean(df_omdb['Runtime'])-np.std(df_omdb['Runtime']),c='b',ls='--')
ax.axvline(x=np.mean(df_omdb['Runtime'])+np.std(df_omdb['Runtime']),c='b',ls='--')
plt.show()
```



```
In [10]: df_omdb['Runtime']=pd.to_numeric(df_omdb['Runtime'],errors='coerce')
for i in range(250):
    if df_omdb.Runtime[i]<=125:
        df_omdb.Runtime[i]=0
    else:
        df_omdb.Runtime[i]=1
```

Figure 19: Runtime to Categorical Variable

To display the most frequently occurring values, the function “plot_column(column_name, n_elem_display=0)” is developed. The main purpose is to plot a bar graph of a data frame column through the function. The function takes the column's name

and the number of elements to display as arguments. It returns a bar graph of the user-defined number of top elements frequently in that column. Within the function “plot_column(column_name, n_elem_display=0)”, another function “clean(column_clean)” is called. It takes a column from the data frame and splits two elements if a comma separates them. For example, in the "Actors" column, values such as Christian Bale and Morgan Freeman should be separated, and a variable should be created to represent each actor separately. Thus, the function "clean(column_clean)" will separate the two actors and store them individually in a list. Moreover, a function "top(column_name)" is created. It takes as its input the column's name and returns a sorted list of the elements that occur very frequently in that column in descending order.

All genres are then plotted as shown in Figure 20: All Genres Plot. All genres will be chosen as predictors in the considered dataset. One-hot encoding will be executed by adding one column for every genre in the data frame, as shown in Figure 21: Genre One-hot Encoding

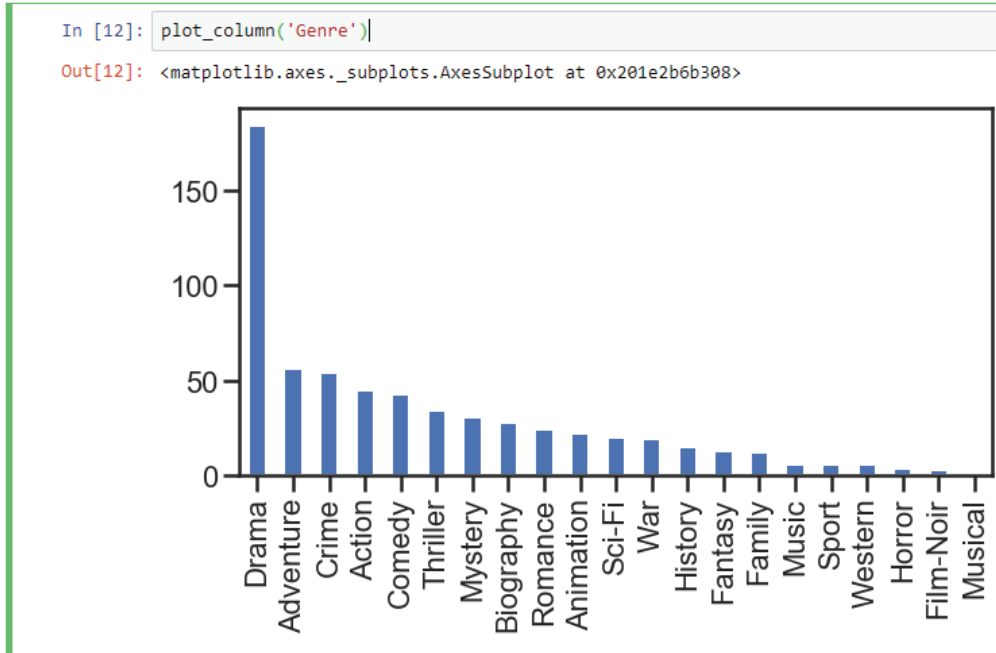


Figure 20: All Genres Plot



Figure 21: Genre One-hot Encoding

The following step analyzes the number of actors that should be considered predictors in the dataset. The top 30 actors with more than three movies in the IMDb's top 250 movies list are taken by analyzing the graph shown in Figure 22: Actors in the Dataset. One column is added for each actor, as shown in Figure 23: Adding Actors to Dataset.

```
In [15]: plot_column('Actors',30)|
```

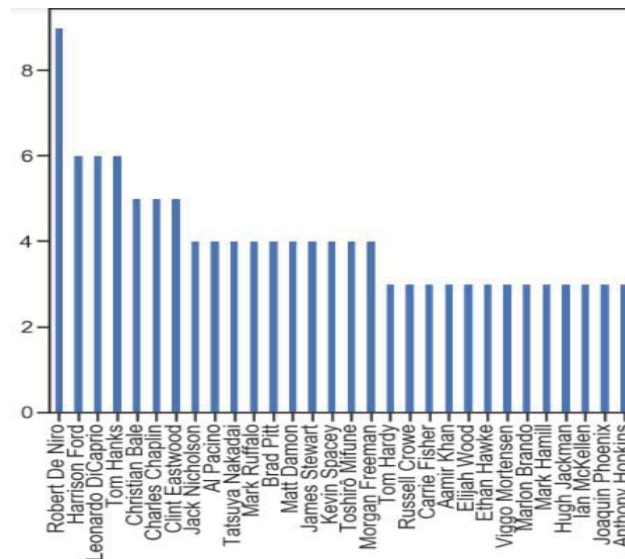


Figure 22: Actors in the Dataset

```
In [16]: #Adding actors to our dataset
actors=top('Actors')
actors
for actor in actors[:30]:
    df_omdb["Actor:"+actor] = [int(actor in a.split(', ')) for a in df_omdb.actors]
```

Figure 23: Adding Actors to Dataset

Another variable that should be assessed is the directors in the dataset. The top 20 directors, each having more than three movies in IMDb's top 250 movies list, are considered. Therefore, a column is created for each director, as shown in Figure 24: Directors considered as Predictors

```
In [17]: plot_column('Director',20)|
```

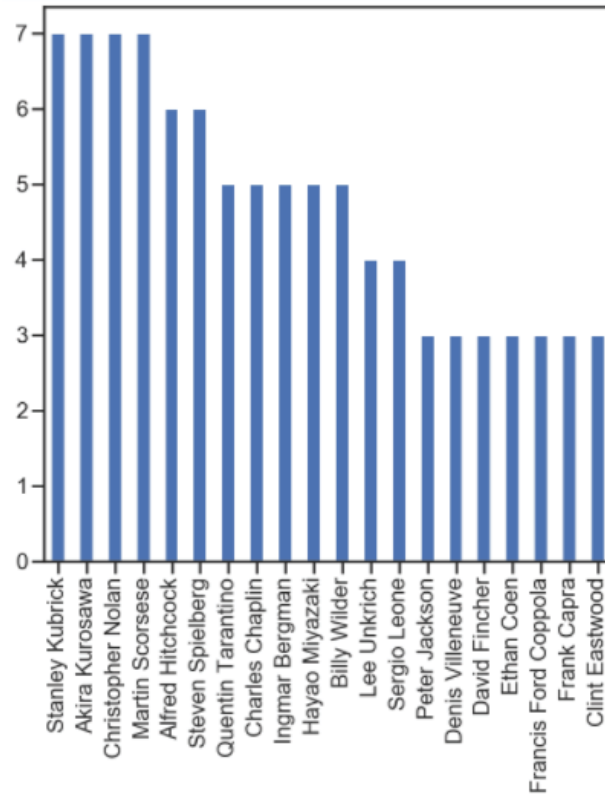



Figure 24: Directors considered as Predictors

The following step is to analyze and assess whether to take writers or not as predictors. Writers are plotted in a bar graph as shown in Figure 25: Writers in the Dataset. Based on the data available and after analyzing it, it can be deduced that not many actors have a significant number of movies. Therefore, writers are not considered one of the predictors in our model.

```
In [20]: writerlist=[w for w in writers2]
          wlt=dummy_writers[writerlist].sum()
          wlt=wlt.sort_values(axis=0,ascending=False)
          wlt.iloc[0:10].plot(kind = "bar",figsize=(10,10))
```

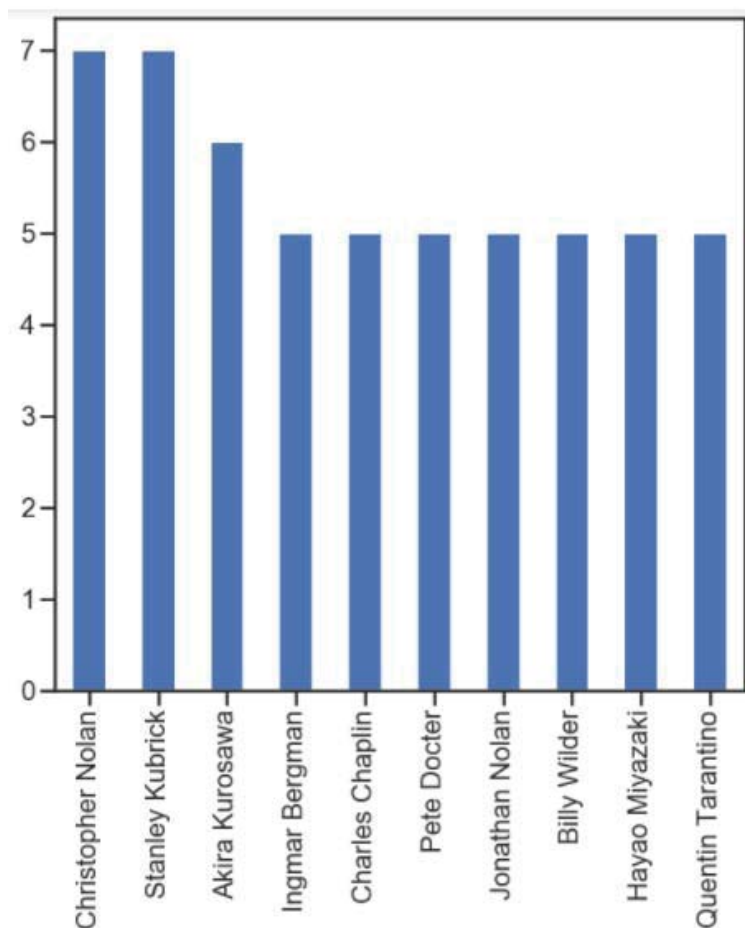


Figure 25: Writers in the Dataset

The language predictor is then explored. The top 11 languages are considered predictors based on the graph in Figure 26: Languages as Predictors. Since the language does not provide any distinguishing characteristics for clustering since there is always the option to add subtitles, language was not considered a predictor in our model.

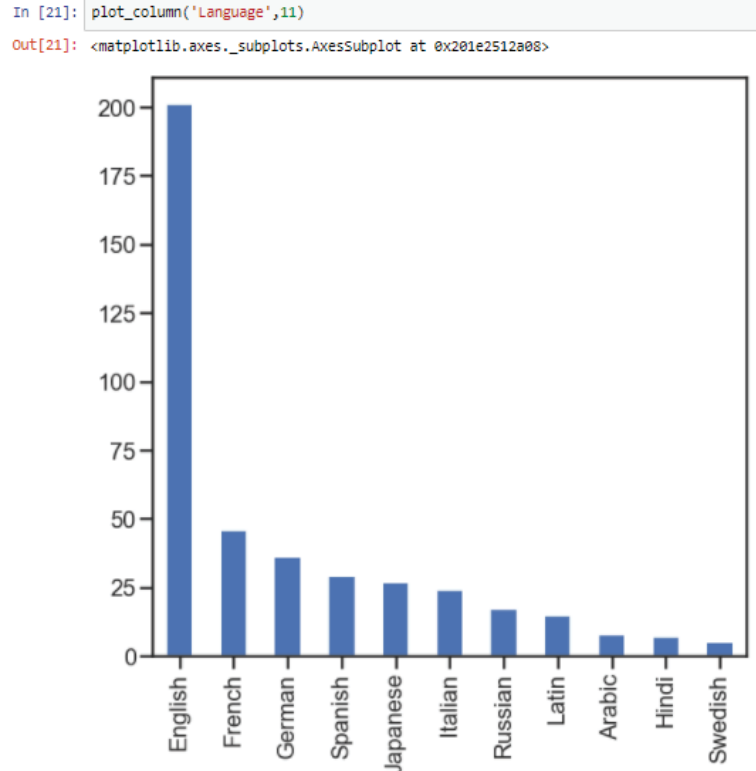


Figure 26: Languages as Predictors

So far, several columns have been added through the one-hot encoding. The shape of the dataset is a matrix of 250 rows representing the movies and 96 columns representing the different features, as shown in Figure 27: Dataset Shape

```
In [22]: df_omdb.shape
Out[22]: (250, 96)
```

Figure 27: Dataset Shape

The country of the movie is assessed next. Based on Figure 28: Movies Distribution in Countries, the movie's country is considered a predictor since the country has a significant role in a movie.

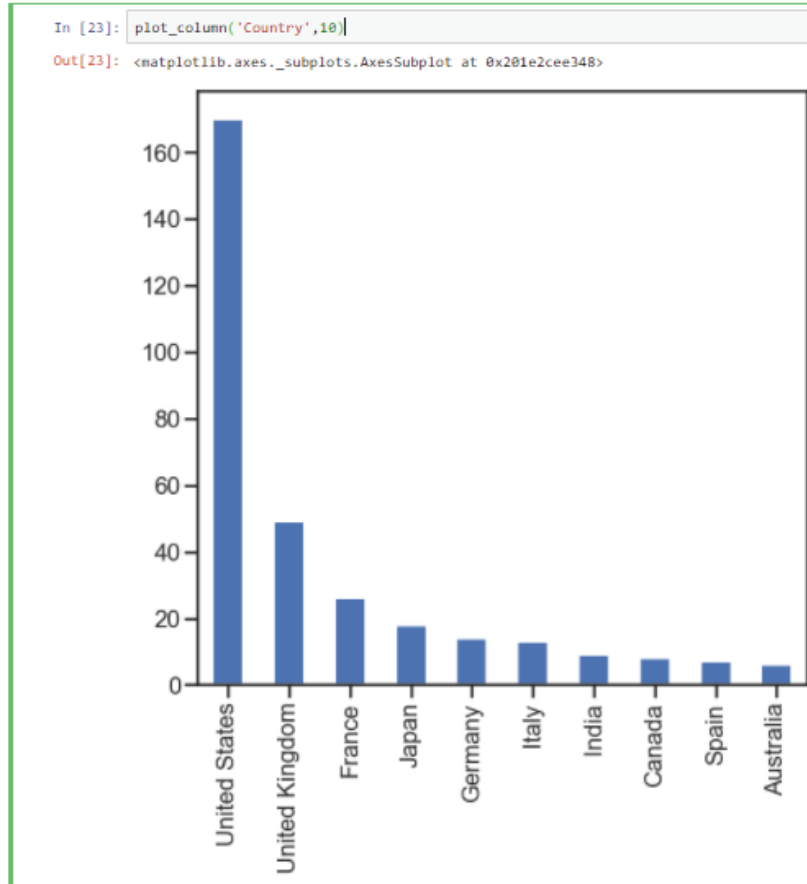


Figure 28: Movies Distribution in Countries

The plot processing is an important step that should be considered next. The plot of each movie is cleaned and tokenized. The aim is to extract words that can be used as features in the clustering algorithm. A function is created to clean, tokenize, lemmatize, and POS tagging to achieve the goal. First, the synonym list is created, as shown in Figure 29: Synonym List

```
# Synonym List
syms = {'wont':'would not', 'cant':'can not', 'cannot':'can not', \
'couldnt':'could not', 'shouldnt':'should not', \
'wouldnt':'would not', 'straightforward': 'straight forward' }
```

Figure 29: Synonym List

Then, the special characters are replaced with spaces in the text being cleaned. The "not contraction" will be replaced by "not". Details are shown in Figure 30: Replacing Special Characters

```
# Preprocess String s
s = s.lower()
#Replace special characters with spaces|
s = s.replace('-', ' ')
s = s.replace('_', ' ')
s = s.replace(',', '. ')
s = s.replace('\\', '')
s = s.replace('.', '')

# Replace not contraction with not
s = s.replace("'nt", " not")
s = s.replace("n't", " not")
...

```

Figure 30: Replacing Special Characters

Tokenization is then performed, and the synonym words will be mapped with the tokens extracted, as shown in Figure 31: Tokenization

```
# Tokenize
tokens = word_tokenize(s)

#tokens = [word.replace(',','') for word in tokens ]

tokens = [word for word in tokens if ('*' not in word) and \
('''' != word) and ("``" != word) and \
(word!='description') and (word !='dtype') \
and (word != 'object') and (word!="s")]

# Map synonyms
for i in range(len(tokens)):
    if tokens[i] in syns:
        tokens[i] = syns[tokens[i]]

```

Figure 31: Tokenization

The following step is to remove the stop words and then perform the lemmatization and stemming, as shown in Figure 32: Lemmatization and Stemming.

```

# Remove stop words
punctuation = list(string.punctuation)+['.', '...']
pronouns = ['i', 'he', 'she', 'it', 'him', 'they', 'we', 'us', 'them', 'he ']
others = ["'d", "co", "ed", "put", "say", "get", "can", "become", \
"los", "sta", "la", "use", "ask", "iii", "else", "doesn't", "dr.", "well", "let", "soon", "finally", "around", "little", \
"would", "set", "use", "place", "still", "three", "arrive", "next", "anoth", "keep", "must", "mr.", "bring", \
"much", "many", "eventually", "explain", "asks", "along", "may", "small", "hold", "realize", "think", "continue", \
"last", "behind", "discover", "something", "several", "end", "large", "high", "mr", "the", "dr", "mr"]
names = ["harry", "travis", "tommy", "joe", "jack", "dorothy", "mike", "george", "frank", "frankie", \
"frank ", "sarah", "andrew", "taylor", "arthur", "luke", "kane", "wallace", "parker", "danny", "tony", \
"michael", "luke", "kane", "danny", "john", "max", "tom", "paul", "ca", "neil", "maria", "barry", "anna", "jerry", \
"alex", "terry", "henry", "gordon", "leonard", "wayne", "vincent", "jimmy", "jordan", "sam", "nick", "nicholson", \
"jake", "rocky"]

stop = stopwords.words('english') + punctuation + pronouns + others + names
filtered_terms = [word for word in tokens if (word not in stop) and \
(len(word)>1) and (not word.replace('.', '', 1).isnumeric()) \
and (not word.replace("'", '', 2).isnumeric())]

# Lemmatization & Stemming - Stemming with WordNet POS
# Since Lemmatization requires POS need to set POS
tagged_words = pos_tag(filtered_terms, lang='eng')

# Stemming with for terms without WordNet POS
stemmer = SnowballStemmer("english")
wn_tags = {'N':wn.NOUN, 'J':wn.ADJ, 'V':wn.VERB, 'R':wn.ADV}
wnl = WordNetLemmatizer()
stemmed_tokens = []
for tagged_token in tagged_words:
    term = tagged_token[0]
    pos = tagged_token[1]
    pos = pos[0]
    try:
        pos = wn_tags[pos]
        stemmed_tokens.append(wnl.lemmatize(term, pos=pos))
    except:
        stemmed_tokens.append(stemmer.stem(term))
return stemmed_tokens

```

Figure 32: Lemmatization and Stemming

The text has been cleaned, tokenized, and lemmatized. Moreover, the POS tagging function has been developed. It is essential in the following steps to assess the words used and their frequency. The analyzer function called "my_analyzer" has been developed to create a word frequency list to identify the most frequent words in the plots, as shown in Figure 33:

Analyzer Function. The first few most occurring words are displayed as shown in Figure 34:

List of Terms with Highest Frequency.

```
# Create Word Frequency by Review Matrix using Custom Analyzer
cv = CountVectorizer(max_df=0.9, min_df=3, max_features=None, \
analyzer=my_analyzer, ngram_range=ngram)
tf1 = cv.fit_transform(plots)
terms1 = cv.get_feature_names()
term_sums = tf1.sum(axis=0)
term_counts = []
out1 = []
out2 = []
for i in range(len(terms1)):
    term_counts.append([terms1[i], term_sums[0,i]])
def sortSecond(e):
    return e[1]
term_counts.sort(key=sortSecond, reverse=True)
print("\nTerms with Highest Frequency:")
for i in range(400):
    print('{:<15s}{:>5d}'.format(term_counts[i][0], term_counts[i][1]))
    out1.append([term_counts[i][0], term_counts[i][1]])
np.savetxt("frequency_100.csv", out1, delimiter=",", fmt='%s')
print("")
```

Figure 33: Analyzer Function

```
Terms with Highest Frequency:
tell          2091
go            1840
find          1676
see           1330
back          1316
make          1208
man           1208
say           1072
time          1039
two           1024
kill           987
try           987
give          980
come          939
life          905
know          853
call          791
---
```

Figure 34: List of Terms with Highest Frequency

Once the word frequency has been determined, finding the TF-IDF score of every word is essential. The aim is to extract the words with the highest scores that can be used as features in clustering.

Terms with Highest TF-IDF Scores:	
father	7.15
kill	6.98
family	5.84
police	5.69
men	5.64
child	5.24
son	5.14
car	5.05
old	4.96
mother	4.87
wife	4.87
friend	4.82
young	4.77
room	4.76
money	4.65
boy	4.58
war	4.51
woman	4.43
love	4.39
live	4.24
run	4.20
house	4.14
german	4.11
night	4.01
train	3.96
murder	3.92

Figure 35: Terms with Highest TF-IDF Scores

A data frame for all frequent words from `count_vectorizer` has been created. Data has also been transposed and rearranged. As per the outcome in Figure 36: Data frame, there are 101 features. Each movie has 101 dimensions. The dataset has 250 data points and 101 dimensions, making it hard to model. To reduce the size of the data frame, principal component analysis is performed.


```
In [34]: df_final2.dropna(inplace=True)
print(df_final2.shape)
df_final2.head()
df_dbscan=df_final2|

(250, 101)
```

Figure 36: Data frame

After cleaning the data, the next step is to apply the Principal Component Analysis (PCA); however, features should be standardized first, as shown in Figure 37: Features Standardization. PCA is applied to standardized features as shown in Figure 38: PCA Application. We have selected the components that explain 1% of the dataset. The first component explains the highest variance in the dataset considered. Thus, the outcome is 39 components out of 101, representing 74.35% variation in our dataset.

The features that are the most important for the first principal component are checked as shown in Figure 39: Features of the First Principal Component. It can be observed that Action movies have the highest weight compared to family, crime, and drama. Moreover, Hugh Jackman and Aamir Khan have been assigned high weights among actors. Various values of epsilon ϵ and min_samples are scanned using DBSCAN to find a good silhouette score. The label output is taken, or the best performing model corresponding to ϵ equals six and min_samples equal 3, as shown in Figure 40: Clustering with DBSCAN.

```

first_comp = pca39.components_[0]
first_comps = pd.DataFrame(list(zip(first_comp, df_final2.columns)), columns=['weights', 'features'])
first_comps['abs_weights']=first_comps['weights'].apply(lambda x: np.abs(x))
first_comps.sort_values('abs_weights', ascending=False,inplace=True)

Top_39_features=first_comps
Top_39_features

```

	weights	features	abs_weights
50	0.314724	Actor:Hugh Jackman	0.314724
66	0.314724	Director:Sergio Leone	0.314724
44	0.314724	Actor:Aamir Khan	0.314724
3	0.289296	genre:Action	0.289296
2	0.267742	Runtime	0.267742
...
83	0.003649	Country:Australia	0.003649
51	0.002717	Actor:Ian McKellen	0.002717
43	-0.002404	Actor:Carrie Fisher	0.002404
35	-0.002215	Actor:Brad Pitt	0.002215
13	-0.000078	genre:History	0.000078

100 rows x 3 columns

Figure 39: Features of the First Principal Component

```
#Performing Clustering with DBSCAN

db = DBSCAN(eps=6, min_samples=3).fit(Xpca)
y = DBSCAN(eps=6, min_samples=3).fit_predict(Xpca)
dbscanlabels= db.labels_

np.unique(y)

array([-1,  0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11], dtype=int64)
```

Figure 40: Clustering with DBSCAN

K-Means clustering is then applied with k equal to 12. Then, the clusters are printed by genres and movie titles:

- Cluster 0
 - Cluster 0 words: genre: Crime, genre: Family, genre: Sport, genre: Animation, genre: War, genre: History.
 - Cluster 0 titles: Fight Club, Se7en

- Cluster 1
 - Cluster 1 words: genre: Comedy, genre: Sci-Fi, genre: Musical, genre: Crime, Actor: Tatsuya Nakadai, genre: History,
 - Cluster 1 titles: The Shawshank Redemption, 12 Angry Men, Pulp Fiction, Forrest Gump, Goodfellas, Cidade de Deus, La vita è Bella, It's a Wonderful Life, Saving Private Ryan, The Green Mile, Léon, The Usual Suspects, Back to the Future, The Lion King, American History X, The Departed, Das Leben der Anderen, Django Unchained, Paths of Glory, The Shining, Joker, Oldeuboi, Kimi no na wa., Once Upon a Time in America, American Beauty, Toy Story, Inglourious Basterds, Requiem for a Dream, A Clockwork Orange, Taxi Driver, Scarface, Snatch, To Kill a Mockingbird, Toy Story 3, Heat, LA Confidential, Monty Python and the Holy Grail, The Wolf of Wall Street, Casino, El laberinto del fauno, Raging Bull, Shutter Island, No Country for Old Men, Three Billboards Outside Ebbing, Missouri, Trainspotting, Fargo, Kill Bill: Vol. 1, The Deer Hunter, Prisoners, The Grand Budapest Hotel, Mary and Max., Catch Me If You Can, Room, Barry Lyndon, The Big Lebowski, Dead Poets Society, Harry Potter and the Deathly Hallows: Part 2, La haine, Spotlight, The Princess Bride

- Cluster 2
 - Cluster 2 words: genre: Biography, genre: Romance, genre: Adventure, genre: Music, Runtime, Actor: Robert De Niro,

- Cluster 2 titles: Schindler's List, The Pianist, Gladiator, Intouchables, Capharnaüm, Amadeus, Braveheart, Hamilton, Lawrence of Arabia, The Father, Green Book, Der Untergang, The Great Escape, A Beautiful Mind, Det sjunde inseglet, The Elephant Man, Gone with the Wind, Z, In the Name of the Father, Hacksaw Ridge, La passion de Jeanne d'Arc, Ford v Ferrari, 12 Years a Slave, Höstsonaten, Ben-Hur, Hachi: A Dog's Tale, Hotel Rwanda, Rush, Into the Wild, Strasti po Andreju, Fanny och Alexander,
- Cluster 3
 - Cluster 3 words: Year, genre: Mystery, genre: Animation, Actor: Leonardo DiCaprio, genre: Film-Noir, Runtime,
 - Cluster 3 titles: Il buono, il brutto, il cattivo, One Flew Over the Cuckoo's Nest, The Silence of the Lambs, Gisaengchung, Psycho, Hotaru no haka, Whiplash, Casablanca, C'era una volta il West, Rear Window, Nuovo Cinema Paradiso, Memento, Sunset Blvd., WALL·E, Witness for the Prosecution, Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb, Coco, Das Boot, 3 Idiots, Good Will Hunting, Reservoir Dogs, M - Eine Stadt sucht einen Mörder, Vertigo, Taare Zameen Par, Idi i smotri, Citizen Kane, Jagten, Singin' in the Rain, North by Northwest, Eternal Sunshine of the Spotless Mind, Ladri di biciclette, Full Metal Jacket, The Apartment, Incendies, Pather Panchali, Double Indemnity, Jodaeiye Nader az Simin, The Sting, Jai Bhim, 1917, Le fabuleux destin d'Amélie Poulain, Per qualche dollaro in più, Up, Die Hard, All About Eve, Some Like It Hot, Bacheha-Ye aseman, Judgment at Nuremberg, There Will Be Blood, The Treasure of the Sierra Madre, El secreto de sus ojos, Chinatown, Lock, Stock and Two Smoking Barrels, Dial M for Murder, The Sixth Sense, Klaus, Smultronstället, The Third Man, The Truman Show, Salinui chueok, Inside Out, The Bridge on the River Kwai, Finding Nemo, Tôkyô monogatari, On the Waterfront, Babam ve Oglum, Stalker, Relatos salvajes, Sherlock Jr., Persona, The General, Before Sunrise, Mr. Smith Goes to Washington, Gone Girl, Andhadhun, To Be or Not to Be, How to Train Your Dragon, Le salaire de la peur, Ah-ga-ssi, Stand by Me, Network, Cool Hand Luke, Gangs of Wasseypur, Les quatre cents coups, Platoon, Koe no katachi, Monsters, Inc., Rebecca, Life of Brian, Eskiya, Fa yeung nin wah, Amores perros, Rocky, It Happened One Night, Before Sunset, La battaglia di Algeri, Shin seiki Evangelion Gekijô-ban: Air/Magokoro wo, kimi ni, Le notti di Cabiria, Paris, Texas, Trois couleurs: Rouge, Soorarai Pottru,
- Cluster 4:
 - Cluster 4 words: genre: Drama, Year, genre: Fantasy, genre: History, genre: Family, genre: Biography,
 - Cluster 4 titles: Modern Times, City Lights, The Great Dictator, The Kid, The Gold Rush,
- Cluster 5:

- Cluster 5 words: genre: Adventure, genre: Animation, genre: Musical, genre: Horror, genre: Crime, genre: Family
- Cluster 5 titles: Unforgiven, Gran Torino, Million Dollar Baby,
- Cluster 6:
 - Cluster 6 words: genre: Action, Runtime, genre: Drama, Year, genre: Animation, genre: Sci-Fi,
 - Cluster 6 titles: Shichinin no samurai, Seppuku, Tengoku to jigoku, Ikiru, Yôjinbô, Ran, Rashômon, Dersu Uzala,
- Cluster 7:
 - Cluster 7 words: Title, genre: Biography, genre: Animation, Year, genre: Crime, genre: Adventure
 - Cluster 7 titles: Star Wars: Episode V - The Empire Strikes Back, Star Wars, Star Wars: Episode VI - Return of the Jedi
- Cluster 8:
 - Cluster 8 words: Year, genre: Music, genre: Comedy, Title, Actor: Al Pacino, genre: Mystery,
 - Cluster 8 titles: Sen to Chihiro no kamikakushi, Mononoke-hime, Hauru no ugoku shiro, Tonari no Totoro, Kaze no tani no Naushika,
- Cluster 9:
 - Cluster 9 words: genre: Comedy, genre: Family, genre: Horror, genre: Biography, genre: Action, Actor: Tom Hanks,
 - Cluster 9 titles: The Godfather, The Godfather: Part II, Apocalypse Now,
- Cluster 10:
 - Cluster 10 words: Title, genre: Drama, genre: Action, genre: Adventure, Actor: Christian Bale, genre: Horror,
 - Cluster 10 titles: The Dark Knight, Inception, The Matrix, Interstellar, Terminator 2: Judgment Day, The Prestige, Alien, Raiders of the Lost Ark, Avengers: Infinity War, Spider-Man: Into the Spider-Verse, The Dark Knight Rises, Aliens, Avengers: Endgame, 2001: A Space Odyssey, Dangal, Metropolis, Indiana Jones and the Last Crusade, Batman Begins, Dune: Part One, The Thing, Jurassic Park, V for Vendetta, Blade Runner, Warrior, Mad Max: Fury Road, Logan,
- Cluster 11:
 - Cluster 11 words: Title, Runtime, genre: Animation, genre: Biography, genre: Fantasy, genre: Comedy,
 - Cluster 11 titles: The Lord of the Rings: The Return of the King, The Lord of the Rings: The Fellowship of the Ring, The Lord of the Rings: The Two Towers,

```

3      101
1      60
2      31
10     26
6       8
8       5
4       5
11      3
9       3
7       3
5       3
0       2
Name: cluster, dtype: int64
    
```

Figure 41: K-Means Clustering

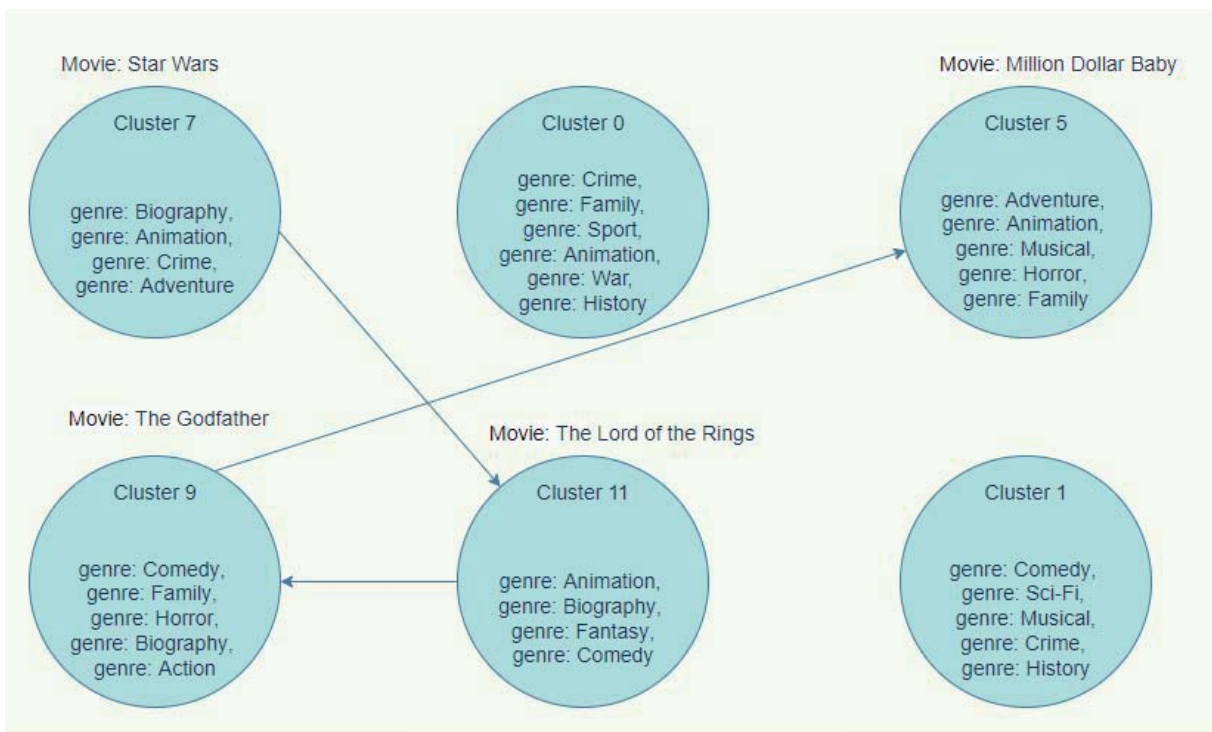


Figure 42: Cluster Linking Using Jaccard Similarity

Based on the results obtained as illustrated in Figure 42: Cluster Linking Using Jaccard Similarity from running our proposed approach to the problem, several clusters with different characteristics present different options to guide the users. For example, if two clusters have a high similarity when considering the movie genre, it would be more reasonable to suggest from the second cluster if the user's choice is already in the first cluster. However, it might be possible to get a user's choice from a cluster that does not share any similarity with another cluster. In that case and based on the Jaccard similarity, the lookup table shown in Table 7: Genre Mapper should be considered. The ratios in the lookup table are determined from the Jaccard similarity concept. Based on the lookup table, the highest ratio should be selected. It describes the highest similarity between the cluster that the user is currently in and the potential cluster from which the suggestion for the user should be drawn.

Table 7: Genre Mapper

Type:	Map to:
Crime	Horror
Family	Animation
Sport	Action
Animation	Family, Fantasy, Sci-Fi
War	Crime
History	War, Crime, Bio
Comedy	
Sci-Fi	Fantasy
Musical	Music, Romance
Biography	Crime, Romance, War, Drama
Romance	Drama, Musical
Adventure	Fantasy, Drama, Action
Mystery	Sci-Fi, Fantasy
Drama	Musical, Music
Fantasy	Animation, Sci-Fi
Horror	Crime, War
Action	Adventure, Crime, Animation

Chapter 4: Conclusion

4.1 Main Contributions of the Thesis

The main contribution of this thesis is a smooth flow of escaping the echo chamber where the following recommendation is related to the current movie but slightly different, in other words, cousins.

4.2 Possible Extensions and Future Work

Future work that can be built on the research presented in this paper is to try the algorithm on users and assess results effectiveness based on several methods. It is expected that the proposed algorithm will perform well when tried on different types of users and applications and its effectiveness can be proven through the available methods in literature. In addition, there is also room for improvement for the mapper lookup table. For instance, the enhancement would be done when the algorithm chooses a mapper to the "Genre," but taking into consideration of the context before choosing.

Bibliography

- [1] R. Meltzer, “How Netflix Utilizes Data Science,” *LightHouse Labs*, 2020.
<https://www.lighthouselabs.ca/en/blog/how-netflix-uses-data-to-optimize-their-product>.
- [2] D. Wang, Y. Liang, D. Xu, X. Feng, and R. Guan, “A content-based recommender system for computer science publications,” *Knowledge-Based Syst.*, vol. 157, no. February 2017, pp. 1–9, 2018, doi: 10.1016/j.knosys.2018.05.001.
- [3] Y. Wang, M. Wang, and W. Xu, “A Sentiment-Enhanced Hybrid Recommender System for Movie Recommendation: A Big Data Analytics Framework,” *Wirel. Commun. Mob. Comput.*, vol. 2018, 2018, doi: 10.1155/2018/8263704.
- [4] P. Melville and V. Sindhvani, “Encyclopaedia of Machine Learning: Recommender Systems,” *Encycl. Mach. Learn.*, pp. 829–838, 2010, [Online]. Available:
https://link.springer.com/content/pdf/10.1007/978-1-4899-7687-1_964.pdf
https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-30164-8_705
<https://link.springer.com/book/10.1007/978-0-387-30164-8>
<http://vikas.sindhvani.org/recommender.p>.
- [5] A. Roy and S. A. Ludwig, “Genre based hybrid filtering for movie recommendation engine,” *J. Intell. Inf. Syst.*, vol. 56, no. 3, pp. 485–507, 2021, doi: 10.1007/s10844-021-00637-w.
- [6] X. Su, T. M. Khoshgoftaar, X. Zhu, and R. Greiner, “Imputation-boosted collaborative filtering using machine learning classifiers,” *Proc. ACM Symp. Appl. Comput.*, no. 2,

- pp. 949–950, 2008, doi: 10.1145/1363686.1363903.
- [7] D. Joshi, “Predictive Modelling on IMDB ’ s Movie Data,” vol. 10, no. 05, pp. 283–290, 2021.
- [8] C. I. Pull, A. Projects, and W. Security, “Clustering of the Top 250 movies from IMDB Introduction,” 2019.
- [9] A. Hassani, A. Iranmanesh, and N. Mansouri, “Text mining using nonnegative matrix factorization and latent semantic analysis,” *Neural Comput. Appl.*, 2021, doi: 10.1007/s00521-021-06014-6.
- [10] N. Sandhya, Y. S. Lalitha, V. Sowmya, K. Anuradha, and A. Govardhan, “Analysis of Stemming Algorithm for Text Clustering,” *Int. J. Comput. Sci. Issues*, vol. 8, no. 5, pp. 352–359, 2011, [Online]. Available: http://wlv.summon.serialssolutions.com/2.0.0/link/0/eLvHCXMwXV2xDQIxDIwQEyCBKfkgKI4TJ64RLwb4Bd52UrJ_iZEoAA9gXePzXXMXAuZrin-c0F2IQzGSyqWxSrECWVjnuypqv2E738R_HIIu_E8hnW5r7dH_PQDRANoEFEIabQGWmlociGiTOxbformUm9svlHNBTOW-U6kTZNDruhIp7C3i32OicLiGV0gNzU7UqyLlo.
- [11] M. Haroon, “Comparative Analysis of Stemming Algorithms for Web Text Mining,” *Int. J. Mod. Educ. Comput. Sci.*, vol. 10, no. 9, pp. 20–25, 2018, doi: 10.5815/ijmeecs.2018.09.03.
- [12] A. Thakkar and K. Chaudhari, “Predicting stock trend using an integrated term frequency–inverse document frequency-based feature weight matrix with neural networks,” *Appl. Soft Comput. J.*, vol. 96, p. 106684, 2020, doi:

- 10.1016/j.asoc.2020.106684.
- [13] C. Cheng, “Principal Component Analysis (PCA) Explained Visually with Zero Math.” <https://towardsdatascience.com/principal-component-analysis-pca-explained-visually-with-zero-math-1cbf392b9e7d>.
- [14] M. Adibifard, A. Sheidaie, and M. Sharifi, “An intelligent heuristic-clustering algorithm to determine the most probable reservoir model from pressure–time series in underground reservoirs,” *Soft Comput.*, vol. 24, no. 20, pp. 15773–15794, 2020, doi: 10.1007/s00500-020-04908-6.
- [15] Nagesh Singh Chauhan, “DBSCAN Clustering Algorithm in Machine Learning,” 2022. <https://www.kdnuggets.com/2020/04/dbscan-clustering-algorithm-machine-learning.html>.
- [16] A. Huang, “Similarity measures for text document clustering,” *New Zeal. Comput. Sci. Res. Student Conf. NZCSRSC 2008 - Proc.*, no. April, pp. 49–56, 2008.
- [17] H. Man, “Data Clustering Using Unsupervised Learning— What type of movies are in the IMDB top 250?,” 2017. <https://medium.com/hanman/data-clustering-what-type-of-movies-are-in-the-imdb-top-250-7ef59372a93b>.
- [18] “Document Clustering with Python.” <http://brandonrose.org/clustering>.
- [19] H. Abdi and L. J. Williams, “Principal component analysis,” *Wiley Interdiscip. Rev. Comput. Stat.*, vol. 2, no. 4, pp. 433–459, 2010, doi: 10.1002/wics.101.
- [20] G. R. Naik, *Advances in principal component analysis: Research and development*. 2017.
- [21] Seb, “Principal Components Analysis Explained for Dummies,” *Class. Mach. Learn.*

- Mach. Learn.*, [Online]. Available: <https://programmatically.com/principal-components-analysis-explained-for-dummies/>.
- [22] “Machine Learning Glossary.” https://developers.google.com/machine-learning/glossary#categorical_data.
- [23] S. Dean, S. Rich, and B. Recht, “Recommendations and user agency: The reachability of collaboratively-filtered information,” *FAT* 2020 - Proc. 2020 Conf. Fairness, Accountability, Transpar.*, pp. 436–445, 2020, doi: 10.1145/3351095.3372866.
- [24] D. Geschke, J. Lorenz, and P. Holtz, “The triple-filter bubble: Using agent-based modelling to test a meta-theoretical framework for the emergence of filter bubbles and echo chambers,” *Br. J. Soc. Psychol.*, vol. 58, no. 1, pp. 129–149, 2019, doi: 10.1111/bjso.12286.

